THE PENNSYLVANIA STATE UNIVERSITY

THE GRADUATE SCHOOL

ANALYSIS OF FINITE-LENGTH LOW-DENSITY PARITY-CHECK CODES

A Dissertation in
Mathematics
by
Chenyiing Wang

© 2010 Chenying Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2010
The dissertation of Chenying Wang was reviewed and approved* by the following:

Wen-Ching Winnie Li
Professor of Mathematics
Dissertation Adviser
Chair of Committee

John J. Metzner
Professor of Computer Science and Engineering

Gary Mullen
Professor of Mathematics
Department Head

Stephen Simpson
Professor of Mathematics

*Signatures are on file in the Graduate School.
ABSTRACT

A low-density parity-check (LDPC) code is a binary linear code specified by a sparse parity check matrix. The min-sum decoding (MSD) algorithm can be applied on the Tanner graph modeling the structure of the LDPC code to do fast efficient decoding. We introduce the concept of maximal allowable error which describes the error-correcting capability of a code when MSD is implemented on LDPC codes over binary symmetric channel (BSC). Two powerful tools, called computation tree and deviation, were introduced by Wiberg to analyze the performance of MSD algorithm of finite-length LDPC codes. By using these tools, we obtain the maximal allowable error over BSC for cycle codes and a lower bound of that for general LDPC codes. By investigating the decoding process of codes $LU(2, q)$ and $LU(3, q)$ which were constructed by using certain $q$-regular bipartite graphs as Tanner graphs, we prove that the lower bound of the maximal allowable error for general LDPC codes is tight for odd $q$ and differs from the tight bound by 1 for even $q$. Furthermore, the Tanner graphs of codes $LU(2, q)$ and $LU(3, q)$ are
shown to be Ramanujan graphs.
## Contents

1 Introduction ............................................. 1

2 LDPC Codes and Min-Sum Decoding .............................. 7
   2.1 LDPC codes and Tanner graphs ......................... 7
   2.2 Min-Sum decoding ....................................... 8
   2.3 Computation tree and deviation .......................... 12
       2.3.1 Computation tree .................................. 13
       2.3.2 Deviation ........................................... 15

3 Analysis of Decoding Performance of Finite-Length LDPC Codes ............................................. 18
   3.1 The maximal allowable error of LDPC codes ............ 18
   3.2 Decoding performance for binary symmetric channel .... 20
       3.2.1 Cycle codes ........................................ 21
       3.2.2 The general LDPC codes .............................. 23
3.3 Decoding performance for AWGN channel

4 The Maximal Allowable Error for Codes $LU(2,q)$ and $LU(3,q)$

4.1 Graphs $D(m,q)$ and codes $LU(m,q)$

4.2 Notation and lemma

4.3 The maximal allowable error of $LU(2,q)$ over BSC

4.4 The maximal allowable error of $LU(3,q)$ over BSC

5 Ramanujan Graphs and Graphs $D(2,q)$ and $D(3,q)$

5.1 Introduction

5.2 Eigenvalues of $D(2,q)$

5.3 Eigenvalues of $D(3,q)$

Bibliography
Acknowledgments

First and foremost I would like to express my most sincere gratitude to my advisor Dr. Wen-Ching Winnie Li, who has supported me throughout my thesis with her immense knowledge and guidance whilst allowing me the room to work in my own way. I attribute the level of my Ph.D degree to her encouragement and effort and without her this thesis, too, would not have been completed or written. One simply could not wish for a better advisor.

I would like to thank Dr. Stephen Simpson for his tremendous help in my early years in graduate school. He taught me the art of studying like a mathematician and gave me the jumpstart for the later research works.

It is a pleasure to also thank all colleagues in my research group: MinLu, Ming-Hsuan Kang and Yang Fang for their enthusiastic discussions with me about my thesis work.

Words fail me to express my appreciation to my husband Yinkun for his support, love and persistent confidence in me. Without him there wouldn’t have been this thesis.
Finally, I wish to express my love and gratitude to my parents, my brother, and my parents in law, for their understanding and love throughout all my studies.
Chapter 1

Introduction

In 1962, Gallager [1] invented the low-density parity-check (LDPC) code in his thesis together with some iterative decoding algorithms. However, these codes and algorithms were forgotten for about 30 years. Then, in the mid-1990’s, they were rediscovered by Mackay and Neal [3, 2] and Wiberg [4, 5], and have received a lot of attention since then. An LDPC code is a linear code specified by a parity check matrix $H$ containing mostly 0’s and relatively few 1’s. To the parity check matrix $H$ we associate a bipartite Tanner graph $T(H)$ [6] whose two vertex sets consist of the bit nodes and check nodes, corresponding to the columns and rows of $H$, respectively. A bit node $v_i$ is adjacent to a check node $f_j$ if and only if the $j$th entry of $H$ is 1, in other words, the $j$th check equation checks the $i$th bit of a codeword. Thus a codeword is an assignment of 0 and 1 of the bit nodes such that the neighbors of each check node sum to zero. The feature that makes LDPC
codes attractive is the existence of computationally simple decoding algorithms, which are message-passing iterative decoding (MPID) algorithms. In the MPID, messages are sent iteratively back and forth between bit nodes and check nodes across the Tanner graph. These nodes try to make the best estimations based on the received information and then broadcast the estimations to their neighbors. The sparsity of the parity check matrix and Tanner graph is the key property that allows for efficient MPID algorithms efficiency of LDPC codes. While MPID is computationally far less demanding than the optimal maximum-likelihood decoding (MLD), which finds the most likely codeword that was originally sent, the performance of MPID is quite good. The analysis of MPID considers the asymptotical case where the block length of the code approaches infinity. Density evolution proposed in [16] is a valuable scheme for analyzing MPID in the asymptotic sense. However, the behavior of the algorithms for the case of finite-length code is at present not well understood. The pseudo-codewords and pseudo-weight introduced in [14] and [15] are good tools for analyzing finite-length LDPC codes, but the role played by pseudo-codewords is not fully understood for general finite-length LDPC codes.

One standard version of MPID algorithms is the min-sum decoding (MSD) algorithm. In this thesis, we restrict our attention to the MSD.
Unless otherwise stated, we also assume that the communication channel is binary symmetric channel (BSC), in which each bit of the codeword is either transmitted correctly with probability $1 - p$ or flipped independently with probability $p$. Since MLD is an optimal decoding algorithm, we are interested in comparing the performance of MSD with MLD. It is well known that in BSC if the number of errors in a received word, i.e., the number of the flipped bits, is less than half of the minimum weight of the LDPC codes, then MLD will correct the errors. Otherwise, it may fail. Similar to the role of half minimum weight in MLD, we introduce the concept of maximal allowable error in MSD, which describes the error-correcting capability of a code over BSC. The maximal allowable error is defined as the largest positive integer $r$ such that if the number of errors in a received word over BSC is at most $r$, then the MSD will stabilize and can correct the errors after some decoding iterations.

Wiberg [4, 5] characterized MSD algorithm convergent on Tanner graphs and provided some necessary and sufficient conditions for the algorithm to converge. He proved that the MSD is optimal if the Tanner graph is cycle-free and also introduced two powerful tools, called computation tree and deviation, which aid in the finite analysis of MSD by describing the algorithm using a cycle-free graph. In the first part of this thesis, by using these tools,
we obtain an upper bound of the errors and a lower bound of the decoding
iterations so that the error will be corrected and the solution will stabilize
when MSD is implemented over BSC. The values of these error bounds are
based on the girth and the degree of the Tanner graph. The error bound
for cycle codes is also the maximal allowable error of cycle codes because
it coincides with that of MLD in worst case. For general LDPC codes, the
error bound gives a lower bound of the maximal allowable error. The details
of these results and some similar results for AWGN channel are provided in
Chapter 3.

As shown in Chapter 3, the girth of a Tanner graph is essential to the
maximal allowable error of a LDPC code. It is desirable to construct codes
from graphs with large girth. An infinite bipartite q-regular graph \( D(q) \),
where \( q \) is a prime power and \( m \geq 2 \), was introduced by Lazebnik and
Ustimenko \[12\] in 1997. By keeping only the first \( m \) coordinates of the
vertices in \( D(q) \), they obtain the truncation graph \( D(m, q) \) of size \( 2q^m \). In
fact, for each \( m \geq 1 \), \( D(m+1, q) \) is a \( q \)-fold cover of \( D(m, q) \). The truncation
graphs \( D(m, q) \) form an infinite tower of covers. The girth of \( D(m, q) \) \[12, 10\]
is at least \( 2 \lceil \frac{m}{2} \rceil + 4 \) which is asymptotically optimal. Using \( D(m, q) \) as
Tanner graphs, Kim et al \[10\] in 2004 constructed a family of LDPC codes
called \( LU(m, q) \). In Chapter 4, the second part of this thesis, we work on the
analysis of decoding performance of codes $LU(2, q)$ and $LU(3, q)$. In order to find the maximal allowable error for these codes, we first bound them from below by the results in Chapter 3. Next, we investigate the decoding process of these two families of codes and bound them from above. Furthermore, these cases also show that the lower bound of the maximal allowable error for general LDPC codes obtained in Chapter 3 is tight for odd $q$ and differs from the tight bound by 1 for even $q$.

Ramanujan graphs, introduced by Lubotzky, Phillips, and Sarnak [8] in 1988, are connected $q$-regular graphs for which the largest nontrivial eigenvalue of the adjacency matrix is not greater than $2\sqrt{q-1}$. The extremal spectral property makes them optimal expanders, leading to wide applications. In addition, they are sparse and have large girth. Hence they appear to be good candidates to be used to associate LDPC codes. In 2000 Rosenthal and Vontobel [9] published the first construction of LDPC codes using Ramanujan graphs. In view of the nice properties $D(m, q)$ possess, we wonder if they happen to be Ramanujan graphs. In the third part of this thesis, working jointly with my advisor Wen-Ching W. Li, we determine the eigenvalues of graphs $D(2, q)$ and $D(3, q)$ and prove that they are Ramanujan graphs [7].

The rest of this thesis is structured as follows. Chapter 2 reviews the
backgrounds for LDPC codes, min-sum decoding, and two tools: computation tree and deviation. In Chapter 3 we obtain new results on the decoding performance of cycle codes and general LDPC codes over BSC and AWGN channel, respectively. Also the lower bounds of the maximal allowable error of LDPC codes over BSC is obtained. In Chapter 4, we first introduce the graphs $D(m, q)$ and codes $LU(m, q)$ and then investigate the decoding process of codes $LU(2, q)$ and $LU(3, q)$ and determine or bound the maximal allowable error of these codes. Then in Chapter 5, we prove that graphs $D(2, q)$ and $D(3, q)$ are Ramanujan graphs.
Chapter 2

LDPC Codes and Min-Sum Decoding

2.1 LDPC codes and Tanner graphs

Low-density parity-check (LDPC) codes and some iterative decoding algorithms were first introduced by Gallager [1] in 1962. However, these codes and algorithms were forgotten for quite a long time until it was rediscovered in 1990’s [3, 2, 4, 5].

A low-density parity-check (LDPC) code $C$ is a binary linear code defined by a sparse parity check matrix $H = (h_{ji})$, i.e. $C = \{x \in \mathbb{F}_2^n | Hx^t = 0\}$. To the parity check matrix $H$ we associate a bipartite graph $T(H)$, called the Tanner graph. The vertex set of the Tanner graph $T(H)$ consists of the bit nodes $V = \{v_1, \cdots, v_n\}$ and the check nodes $F = \{f_1, \cdots, f_r\}$. The set $\{v_i, f_j\}$ is an edge if and only if $h_{ji} = 1$. Thus a codeword $x$ is an assignment of values 0 or 1 to the bit nodes of the Tanner graph such that at each check
node the sum of its neighboring bit nodes is 0.

**Example 2.1.** Let $C$ be the code with the parity check matrix

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

Obviously, $C = \{(0,0,0,1,1,0,1), (0,0,0,1,0,1,1), (0,0,0,0,1,0,1), (0,0,0,0,0,0,0)\}$.

The Tanner graph $T(H)$ that is associated to $H$ is shown in Figure 2.1.

### 2.2 Min-Sum decoding

The feature that makes LDPC codes attractive is the existence of computationally simple decoding algorithms, which are message-passing iterative decoding (MPID) algorithms. The most two popular versions of the MPID are the min-sum decoding (MSD) and the sum-product decoding (SPD). In this thesis, we restrict our attention to the MSD. For the SPD, some results are the same as the MSD.

When $\tilde{y} = (\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n)$ is received, the maximum-likelihood decoding (MLD) algorithm, which is the optimal decoding, finds the most likely codeword that was sent, i.e., the codeword $y = (y_1, y_2, \ldots, y_n)$ that maximizes the
probability $Pr[y|\tilde{y}]$. Since the MLD is searching among the codewords exhausitively, it is time consuming, especially when the code is large. The MSD is a sub-optimal algorithm. It applies an iterative process on Tanner graph. In this thesis, we assume that all codewords are equally probable. Unless otherwise stated, we also assume the channel is a memoryless binary symmetric channel (BSC). By Bayes’ rule, this assumption implies that the codeword $y \in C$ which maximizes the probability $Pr[y|\tilde{y}]$ also maximizes $Pr[\tilde{y}|y]$. Moreover, the memoryless property of the channel implies that

$$Pr[\tilde{y}|y] = \prod_{i=1}^{n} Pr[\tilde{y}_i|y_i].$$

Thus the MLD algorithm is equivalent to finding the codeword $y$ that min-
imize the negative log-likelihood:

\[ G(y) = -\log Pr[\tilde{y}|y] = \sum_{i=1}^{n} -\log Pr[\tilde{y}_i|y_i], \]

which will be called our global cost function.

In the MSD, messages are sent back and forth between bit nodes and check nodes across the Tanner graph. More precisely, in each around of iteration, the message passed from a bit node \( v_i \) to a check node \( f_j \) is the cost (negative log-likelihood) that \( v_i \) has a certain value (0 or 1), given the local cost of that bit node, and all the values communicated to \( v_i \) in the previous round from check nodes incident to \( v_i \) which are not \( f_j \). Then, the message passed from \( f_j \) to \( v_i \) is the cost that \( v_i \) has a certain value given all the messages just passed to \( f_j \) from bit nodes which are not \( v_i \). Let \( \mu_{i\rightarrow j}^{(l)}(a), a \in \mathbb{F}_2 = \{0,1\} \) be the message passed from the bit node \( v_i \) to the check node \( f_j \) at the \( l \)-th round of the algorithm. Define \( \mu_{j\rightarrow i}^{(l)}(a) \) in a similar way. Here all \( \mu^{(l)}_{i\rightarrow j}(a) \) and \( \mu^{(l)}_{j\rightarrow i}(a) \) are called intermediate cost functions based on some bit nodes. And we use \( \gamma_i(y_i) = -\log Pr[\tilde{y}_i|y_i] \) as the local cost function which is based on the only node \( v_i \). At round 0, \( \mu^{(0)}_{i\rightarrow j} \) and \( \mu^{(0)}_{j\rightarrow i} \) are set to zero. The update rules for the messages can be described as follows:
\[
\mu_{[i \rightarrow j]}^{(l)}(a) = \gamma_i(a) + \sum_{j' \in N(i), j' \neq j} \mu_{[j' \rightarrow i]}^{(l-1)}(a),
\]

\[
\mu_{[j \rightarrow i]}^{(l)}(a) = \min_{x \in C_j, x_i = a} \left\{ \sum_{i' \in N(j), i' \neq i} \mu_{[i' \rightarrow j]}^{(l)}(x_{i'}) \right\}.
\]

Here \(N(j)\) denotes the set of the indices of the neighboring bit nodes of the
check node \(f_j\), \(N(i)\) is the set of the indices of the neighboring check nodes
of the bit node \(v_i\), and \(C_j\) is the set of all \(x \in \mathbb{F}_2^n\) such that \(x\) is locally valid
on the check node \(f_j\), i.e., \(C_j = \{ x \in \mathbb{F}_2^n | \sum_{i \in N(j)} x_i = 0 \}\).

After each iteration, we compute the following cost functions for all bit
nodes:

\[
\mu_i^{(l)}(a) = \gamma_i(a) + \sum_{j' \in N(i)} \mu_{[j' \rightarrow i]}^{(l)}(a).
\]

The estimate at a bit node \(v_i\) is

\[
v_i = \begin{cases} 
1 & \text{if } \mu_i(1) \leq \mu_i(0), \\
0 & \text{otherwise}.
\end{cases}
\]

If the estimate of the bit nodes happens to form a codeword, the iteration
stops. Otherwise, a new round of iteration starts.

Obviously, the above updating rules assume that the incoming messages
are independent. This assumption holds only if the associated Tanner graph
is a tree. For this case, we cite the following Theorem by Wiberg [5]:

11
Theorem 2.1. If the Tanner graph is finite and cycle-free, then the cost functions stabilize after finitely many iterations, and the final cost functions are
\[ \mu_i^{(l)}(a) = \min_{x \in \mathcal{C}, x_i = a} G(x). \]
In other words, the solution converges to the maximum-likelihood (ML) solution if the Tanner graph is cycle-free. For Tanner graphs that contain cycles, this is not true in general. This issue is discussed in the following section. For the max-product decoding, there is similar decoding procedure and Theorem 2.1 also holds, so do all the results in this chapter and Chapter 3.

2.3 Computation tree and deviation

In order to analyze the MSD for general Tanner graphs that contain cycles, Wiberg introduced two tools: computation tree and deviation.

Since we only consider the linear codes, we can assume (without loss of generality) that the all-zero codeword was transmitted and \( \tilde{y} = (\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n) \) was received in this thesis. Furthermore, we assume (also with full generality) that the local cost functions are normalized so that \( \gamma_i(0) = 0 \) for all bit nodes \( v_i \), and
\[
\gamma_i(1) = \begin{cases} 
-1 & \text{if } \tilde{y}_i = 1, \\
1 & \text{if } \tilde{y}_i = 0.
\end{cases}
\]
Thus the global cost may be written as
\[
G(x) = \sum_{i=1}^{n} \gamma_i(x_i) = \sum_{i \in \text{supp}(x)} \gamma_i(1).
\]

Here \( \text{supp}(x) \) is the support of \( x \), consisting of the indices where \( x \) has nonzero components.

2.3.1 Computation tree

Consider the cost function \( \mu_i^{(l)}(a) \) for the bit node \( v_i \) after \( l \) rounds of iterations. We can form a tree graph consisting of interconnected bit nodes and check nodes on which the computation of \( \mu_i^{(l)}(a) \) depends, in the same way as the original Tanner graph. The bit node \( v_i \) will be the root of the tree and the depth of the tree is equal to the iteration number \( l \). (For the depth of the tree, we only consider the layers of bit nodes. Here the root bit node \( v_i \) is layer 0.) In general, the same bit nodes and check nodes may occur at several places in the computation tree because the Tanner graph may have cycles.

We denote the bit nodes and the check nodes in the computation tree by \( \mathcal{V}^{(l)} \) and \( \mathcal{F}^{(l)} \), respectively, treating possible multiple occurrences distinctly. Associate with the computation tree is a tree code \( \mathcal{C}^{(l)} \) determined by \( \mathcal{V}^{(l)} \) and \( \mathcal{F}^{(l)} \). In fact, the computation tree is the Tanner graph of the code \( \mathcal{C}^{(l)} \). Also the global cost function \( \mathcal{S}^{(l)}(u) \) is defined for all tree codewords.
$u \in \mathcal{C}^{(l)}$. Call $(\mathfrak{V}^{(l)}, \mathfrak{F}^{(l)}, \mathcal{C}^{(l)})$ the tree system.

**Example 2.2.** Figure 2.2 illustrates the computation tree for the bit node $v_1$ after two iterations when decoding Code $\mathcal{C}$ in Example 2.1. The associated bit nodes set and check nodes set are

$$\mathfrak{V}^{(2)} = \{v_{1,1}, v_{2,1}, v_{3,1}, v_{2,2}, v_{1,2}, v_{3,2}, v_{4,1}, v_{6,1}, v_{5,1}, v_{6,2}, v_{7,1}, v_{1,3}, v_{4,2}, v_{6,3}\}$$

and

$$\mathfrak{F}^{(2)} = \{f_{1,1}, f_{2,1}, f_{2,2}, f_{3,1}, f_{4,1}, f_{1,2}, f_{3,2}\}$$

respectively. Here all nodes $v_{i,m}$ (or $f_{j,m}$) are the multiple occurrences of the same node $v_i$ (or $f_j$) in the original Tanner graph $T(H)$ in Figure 2.1.

With these definitions, the following corollary of Theorem 2.1 is obtained by Wiberg.

**Corollary 2.1.** When the min-sum algorithm is applied to a binary linear code with an associated Tanner graph (cycle-free or not), and the corresponding tree system is defined as above, the min-sum decoding computes

$$\mu_i^{(l)}(a) = \min_{u \in \mathcal{C}^{(l)}, u_i = a} \mathcal{G}^{(l)}(u)$$

for the bit node $v_i$ after $l$ rounds of iterations.

In summary, the MSD always operates as if the Tanner graph is cycle-free and provides optimal decision based on the tree system $(\mathfrak{V}^{(l)}, \mathfrak{F}^{(l)}, \mathcal{C}^{(l)})$ defined by the computation tree.
2.3.2 Deviation

From Corollary 2.1, it is clear that the bit node $v_i$ will be decoded incorrectly (i.e., to a one) after $l$ rounds of iterations if and only if some valid tree codeword $u \in \mathcal{C}^{(l)}$ with $u_i = 1$ has a lower global cost than all valid tree codewords with $u_i = 0$. Thus a necessary and sufficient condition for decoding error (at bit node $v_i$) is

$$\min_{u \in \mathcal{C}^{(l)}_0} \mathcal{G}^{(l)}(u) \geq \min_{u \in \mathcal{C}^{(l)}_1} \mathcal{G}^{(l)}(u),$$

where $\mathcal{C}^{(l)}_0 := \{u \in \mathcal{C}^{(l)} : u_i = 0\}$ and $\mathcal{C}^{(l)}_1 := \{u \in \mathcal{C}^{(l)} : u_i = 1\}$.

A simpler necessary condition is provided through the concept of deviation, introduced by Wiberg [5]:

Figure 2.2: Computation tree with root $v_1$ after two iterations when decoding code $\mathcal{C}$
Definition 2.1. Given a tree system rooted at $v_1$, a tree codeword $e \in C_1$ is called a deviation if it does not cover any other nonzero valid tree codeword, i.e., there is no $u \in C_1$, $u \neq 0$, such that $\text{supp}(u) \subset \text{supp}(e)$. The weight of a deviation $e$ is defined as the number of nonzero bit nodes in $e$, i.e., $\text{weight}(e) = |\text{supp}(e)|$.

Figure 2.3 illustrates two valid tree codewords in $C(2)$ of the tree Code determined by the computation tree in Figure 2.2, one of which is a deviation. Using the concept of deviation, the following result is proved by Wiberg in [5].

Theorem 2.2. A necessary condition for a decoding error to occur after $l$ rounds of iterations is that $G(l)(e) \leq 0$ for some deviation $e$. 
Figure 2.3: To the left, a valid codeword in $C_1^{(2)}$ is shown. However, it is not a deviation, because it covers a valid codeword (with "ones" only assigned to marked region). A deviation, shown to the right, is obtained by assigning zeros to those two marked bit nodes.
Chapter 3

Analysis of Decoding Performance of Finite-Length LDPC Codes

3.1 The maximal allowable error of LDPC codes

Since Theorem 2.1 does not apply when there are cycles in the Tanner graph, it is somewhat surprising that, in fact, the MSD often works quite well. This is primarily demonstrated by simulation results [1]. So we are interested in analyzing the MSD of LDPC codes with cycles in the Tanner graph.

Since MLD is an optimal decoding algorithm, we are interested in comparing the performance of MSD with MLD. In binary symmetric channel (BSC), it is well known that if the number of errors of a received word, i.e., the number of the flipped bit nodes, is less than half of the minimum weight of the LDPC codes, then MLD will correct the errors. Otherwise, it may fail. Although this is not true for MSD in general, there are some weak
solutions. By the definition of the normalized local cost functions $\gamma_i(y_i)$ in section 2.3, we obtain that the global cost of a deviation $\mathcal{C}^{(l)}(e) > 0$ if and only if less than half of the bit nodes of $\text{supp}(e)$ are flipped. Thus the following corollary of Theorem 2.2 is obtained.

**Corollary 3.1.** Given a tree system rooted at $v_i$, if the channel is BSC and the number of the flipped bit nodes in the tree bit node set $\mathcal{V}^{(l)}$ is less than half of the weight of all deviations, then MSD will correct the possible error at $v_i$ after $l$ iterations.

Similar to the role of half minimum weight in MLD, we introduce the concept of maximal allowable error in MSD as follows.

**Definition 3.1.** The maximal allowable error of an LDPC code is the largest positive integer $r$ such that if the number of errors in a received word over BSC is at most $r$, then the MSD will stabilize and can correct the errors after some decoding iterations.

By Theorem 2.1, the solution converges to the maximum-likelihood (ML) solution if the Tanner graph is cycle-free. So the maximal allowable error for MSD is just the largest integer which is less than half of the minimum weight if the Tanner graph of a LDPC code is cycle-free.

In 1981, Tanner gave a tree bound $b$ [6] of minimum weight of LDPC
codes which is indeed the minimum weight of the deviation in the biggest possible computation tree in which no bit nodes repeat. Tanner then designed a revised finite-step min-sum decoding algorithm in which the errors can be corrected if the number of errors in a received word is less than half of his tree bound $b$. Thus $\lfloor \frac{b-1}{2} \rfloor$ is indeed a lower bound of the maximal allowable error for his revised finite-step min-sum decoding.

In section 3.2, we discuss the performance of min-sum decoding for cycle codes and general LDPC codes. We also get lower bounds of the maximal allowable error for cycle codes and general LDPC codes. In section 3.3, we show some related results for LDPC codes over AWGN channel.

3.2 Decoding performance for binary symmetric channel

In Theorem 2.2, Wiberg found a necessary condition for a decoding error to occur using the two powerful tools: computation tree and deviation. In this section and section 3.3, we use Theorem 2.2 or the Corollary 3.1 to characterize the error-correcting capability of MSD for cycle codes and general LDPC codes over BSC and AWGN, respectively.

In order to use corollary 3.1, we shall investigate the minimal possible weight of any deviation in a computation tree of a given LDPC code. Notice that given a deviation $e$ of a tree, for any check node in the tree, the
assignment of its adjacent bit nodes contains either no "one" or two "ones". Otherwise, if there are more than two "ones" in its neighboring bit nodes, then this deviation must cover another nonzero tree codeword and contradicts the definition of deviations. Suppose the minimal degree of the bit nodes is $c$, that is, each bit node has at least $c$ adjacent check nodes. Let $v_i$ be the root bit node of the computation tree. Then the assignment of this root $v_i$ in any deviation $e$ must be "one". Since the number of its adjacent check nodes is at least $c$, there are at least $c$ "ones" in the neighboring bit nodes of the check nodes that $v_i$ is adjacent to except $v_i$, i.e. the bit nodes in the layer 1 of the tree. And each of these new "ones" gives rise to at least $c - 1$ new "ones" in the layer 2, and so on. So for a computation tree of depth $l$, which arises after $l$ rounds of iterations, the number of "ones" in any deviation $e$ is $\text{weight}(e) \geq 1 + c \sum_{s=1}^{l} (c - 1)^{s-1}$.

### 3.2.1 Cycle codes

A cycle code is an LDPC code whose associated Tanner graph is left 2-regular, i.e., all bit nodes have degree 2. The minimum weight of a cycle code is just half of the girth of the associated Tanner graph. Here the girth is the length of the shortest cycle contained in the graph. Let $g = \frac{\text{girth}}{2}$ and $d = \lfloor \frac{g - 1}{2} \rfloor$. So for cycle codes, the maximum number of errors that MLD can correct in worst case is $d$. We shall show that this result also holds for
Since all bit nodes have only two adjacent check nodes, it is easy to derive that the support of a deviation $e$ of a cycle code has the form of a path starting at a leaf bit node, going through the root, and ending at another leaf bit node in a computation tree. More precisely, such a path passes through alternating bit nodes and check nodes on the original Tanner graph with the property that the same edge never occurs twice successively. And obviously, we have $\text{weight}(e) = 2l + 1$, where $l$ is the depth of the tree or the numbers of iterations.

**Theorem 3.1.** (1). For a cycle code, if the number of errors in a received word is at most $d = \lfloor \frac{g-1}{2} \rfloor$, where $g = \frac{\text{girth}}{2}$, then the min-sum decoding will stabilize and can correct the errors after $\frac{2d^2+3d}{2}$ decoding iterations over BSC.

(2). The maximal allowable error for cycle codes is $d$.

**Proof.** (1). Because the number of decoding iterations is $\geq \frac{2d^2+3d}{2}$, the weight of a deviation $e$ is $\geq (2d+1)(d+1)$. Rewrite $\text{weight}(e) = m(2d+1)+k$, where $0 \leq k < 2d + 1$. So $m \geq d + 1$. Divide the corresponding path into $m+1$ connected short paths with $m$ paths of weight $2d+1$ and one path of weight $k$. Here the weight of a path is the number of the bit nodes. Since the girth of the Tanner graph is $\geq 2d + 1$ by assumption, any bit node of
the original Tanner graph can not occur more than once on a path of weight $\leq 2d + 1$. And since the number of errors is $\leq d$, the sum of the normalized local cost of all bit nodes on each short path is at least 1 by definition. So the globe cost of the deviation $e$ is $G(e) \geq (d+1) \times 1 - d = 1$. By Theorem 2.2, the MSD will correct the errors.

(2). Obviously this $d$ gives a lower bound of the maximal allowable error for cycle codes. Since MLD can correct up to $d$ errors in worst case, this $d$ is also an upper bound of the maximal allowable error for cycle codes. Therefore the maximal allowable error for cycle codes is just $d$.

Note that if the number of errors is at most $d$, the min-sum decoding can also correct the errors whenever $m > \min\{k, d\}$. For example, when the number of decoding iterations equals $d$, the min-sum decoding will correct up to $d$ errors. In fact, the stability of the iterated solution is the point of the above theorem.

3.2.2 The general LDPC codes

For an LDPC code with $c \geq 3$, where $c$ is the minimal degree of the bit nodes, the support of a deviation $e$ can be viewed as a tree. Using a similar idea, we obtain the following theorem. Instead of short paths for cycle codes,
the construction blocks are subtrees. Similarly, let $g = \frac{\text{girth}}{2}$ and $d = \left\lfloor \frac{q-1}{2} \right\rfloor$.

**Theorem 3.2.** (1). For an LDPC code whose bit nodes have minimal degree $c \geq 3$, if the number of errors in a received word is at most $\left\lfloor \frac{(c-1)d+1-1}{2c-4} \right\rfloor$, then the min-sum decoding will stabilize and can correct the errors after $d$ decoding iterations over BSC.

(2). $\left\lfloor \frac{(c-1)d+1-1}{2c-4} \right\rfloor$ is a lower bound of the maximal allowable error for general LDPC codes. This bound is tight for odd $q$ and differs from the tight bound by 1 for even $q$.

(3). There exist LDPC codes whose maximal allowable error are just $\left\lfloor \frac{(c-1)d+1-1}{2c-4} \right\rfloor$ or not greater than $\left\lfloor \frac{(c-1)d+1-1}{2c-4} \right\rfloor + 1$.

**Proof.** (1). Given a computation tree and a deviation $e$, a bit node on the computation tree is said to be efficient if it is assigned one in the deviation $e$.

If there are at least $r$ layers below an efficient bit node $v$ on a computation tree, let $T_r(v)$ be the subtree (of the computation tree) with depth $r$ and root $v$, denote by $\text{weight}(T_r(v))$ the number of efficient bit nodes on this subtree.

Let $\text{weight}(r)$ be the number of all efficient bit nodes on the $r$-th layer of a computation tree. Since the root is the layer 0, we have $\text{weight}(0) = 1$.

Suppose $l$ is the number of decoding iterations. Then if $l = d$, for
any deviation $e$ of the corresponding computation tree, $weight(e) \geq 1 + c \sum_{s=1}^{d} (c - 1)^{s-1} > 2 \frac{(c-1)^{d+1}-1}{2c-4}$. By the definition of the girth, any bit node of the original Tanner graph can not occur more than once on the tree. Thus by the bound of the number of the errors, the cost of the deviation $e$ is $G^{(l)}(e) > 0$.

On the other hand, if $l \geq d + 1$, then for any efficient bit node $v$ on the layer $l - d$, $weight(T_d(v)) \geq 1 + \sum_{s=1}^{d} (c - 1)^{s} = 2 \frac{(c-1)^{d+1}-1}{2c-4}$ because $v$ is not the root. So the sum of the local cost of all efficient bit nodes on the subtree $T_d(v)$ is at least 0. For any efficient bit node $v'$ on the layer $l - d - 1$, the same bit node of the original Tanner graph can not occur again on the subtree $T_{d+1}(v')$. So it can be derived that the sum of the local cost of all efficient bit nodes on $T_{d+1}(v')$ is at least 1. Thus the sum of the local cost of all efficient bit nodes on the last $d + 2$ layers, i.e., from layer $l - d - 1$ to layer $l$, is at least $weight(l - d - 1)$. And since $weight(l - d - 1) > \Sigma_{s=0}^{l-d-2} weight(s)$ by the assumption $c \geq 3$ and $\Sigma_{s=0}^{l-d-2} weight(s)$ is not more than the sum of the local cost of all efficient bit nodes on the first $l - d - 2$ layers, the sum of the local cost of all efficient bit nodes on the computation tree is more than 0, i.e., the globe cost of the deviation $e$ is $G^{(l)}(e) > 0$.

By Theorem 2.2, the MSD will correct the errors.

(2)(3). This $\lfloor \frac{(c-1)^{d+1}-1}{2c-4} \rfloor$ is obviously a lower bound of the maximal
allowable error for general LDPC codes. In Chapter 4, we shall investigate the decoding performance of special codes $LU(2,q)$ and $LU(3,q)$ and show that the maximal allowable error of codes $LU(2,q)$ and $LU(3,q)$ is just 
\[
\left\lfloor \frac{(c-1)d+1}{2c-4} \right\rfloor \text{ for odd } q \text{ or not greater than } \left\lfloor \frac{(c-1)d+1}{2c-4} \right\rfloor + 1 \text{ for even } q \text{ except a few cases. This also show that (2) is true.}
\]

From the tree bound on minimum weight given by Tanner [6], it can be shown that the error bound 
\[
\left\lfloor \frac{(c-1)d+1}{2c-4} \right\rfloor
\]
is always less than half of the minimum distance.

By Theorem 3.1 and 3.2, it’s obvious that the girth of a Tanner graph plays a crucial role in the decoding performance of LDPC codes. So it is desirable to construct codes from graphs with large girth.

For max-product decoding, since Theorem 2.2 also holds, so do Theorem 3.1 and 3.2(1).

### 3.3 Decoding performance for AWGN channel

In a AWGN channel, each input bit is $X \in \{0,1\}$ and $Y = (2X - 1) + N$ is the received bit, where $N$ has a Gaussian distribution with 0 mean and variance $\sigma^2$. As usual, we assume that the all-zero codeword was transmitted and \( \tilde{y} = (\tilde{y}_1, \cdots, \tilde{y}_n) \) was received. Furthermore, the local cost function can be normalized so that $\gamma_i(0) = 0$ for all bit nodes $v_i$, and $\gamma_i(1) = \tilde{y}_i$. The
codeword $x$ that minimize the global cost function $G(x) = \sum_{i=1}^{n} \gamma_i(x_i) = \Sigma_{i \in \text{supp}(x)} \tilde{y}_i$ is just the maximum-likelihood solution.

Let $\text{Max}(n)$ be the sum of the largest $n$ local functions $\gamma_i(1)$. Similar to the BSC channel, we got the following two results for cycle code and general LDPC codes.

**Theorem 3.3.** For a cycle code with minimum weight $w$, if $\text{Max}(w-1) \leq 0$, then the min-sum decoding can correct the errors after $w^2$ decoding iterations over AWGN channel.

Let $g = \frac{\text{girth}}{2}$ and $d = \lfloor \frac{g-1}{2} \rfloor$.

**Theorem 3.4.** For an LDPC code with the minimal degree $c$ of the bit nodes at least 3, if $\text{Max}(n) \leq 0$, where $n = 2\frac{(c-1)d+1-1}{2c-4}$, then the min-sum decoding can correct the errors after $d$ decoding iterations over AWGN channel.
Chapter 4

The Maximal Allowable Error for Codes $LU(2, q)$ and $LU(3, q)$

4.1 Graphs $D(m, q)$ and codes $LU(m, q)$

An infinite bipartite $q$-regular graph $D(q)$, where $q$ is a prime power and $m \geq 2$, was introduced by Lazăbnik and Ustimenko [12] in 1997: The vertices are divided into two sets $X = F_q^{\infty}$ and $Y = F_q^{\infty}$; a vertex $[x, x_1, x_2, \ldots]$ in $X$ is adjacent to a vertex $[y, y_1, y_2, \ldots]$, in $Y$ if and only if the following infinite equations are satisfied:

\[
y_1 = xy + x_1, \quad y_2 = xy_1 + x_2,
\]
\[
y_3 = yx_1 + x_3, \quad y_4 = yx_2 + x_4,
\]
\[
\ldots \ldots \ldots \ldots
\]
\[
y_{4i+1} = xy_{4i-1} + x_{4i+1}, \quad y_{4i+2} = xy_{4i} + x_{4i+2},
\]
By keeping only the first $m$ coordinates of the vertices in $D(q)$, they obtain the truncation graph $D(m, q)$ of size $2q^m$. In fact, for each $m \geq 1$, $D(m + 1, q)$ is a $q$-fold cover of $D(m, q)$. The truncation graphs $D(m, q)$ form an infinite tower of covers. The girth of $D(m, q)$ [12, 10] is at least $2\lceil \frac{m}{2} \rceil + 4$ which is asymptotically optimal. Hence graphs $D(m, q)$ appear to be good candidates for Tanner graphs.

Using $D(m, q)$ as Tanner graphs, Kim et al [10] in 2004 constructed a family of LDPC codes called $LU(m, q)$. In Chapter 5, we prove that graphs $D(2, q)$ and $D(3, q)$, are Ramanujan graphs, which means that these graphs are spectrally optimal and the codes $LU(2, q)$ and $LU(3, q)$ have good features. So it is interesting to analyze the decoding performance of $LU(2, q)$ and $LU(3, q)$. In [10], the minimum weight and the weight of the minimum stopping set, which are important measures of the decoding performance of a code using maximum-likelihood decoding (MLD) over Binary Symmetry Channel (BSC) and min-sum decoding (MSD) over Binary Erasure Channel (BEC), are either determined or bounded.

In the following sections, we investigate the decoding process of $LU(2, q)$
\[LU(3, q)\] and find or bound the maximal allowable error of these codes using MSD over BSC.

### 4.2 Notation and lemma

In sections 4.3 and 4.4, we shall prove the following theorem.

**Theorem 4.1.** The maximal allowable error \( r \) of \( LU(2, q) \) over BSC satisfies
\[
\frac{q-1}{2} \leq r \leq \frac{q+1}{2}.
\]
In particular, the maximal allowable error is equal to \( \frac{q}{2} \) for even \( q \).

The maximal allowable error \( r \) of \( LU(3, q) \) is the same as that of \( LU(2, q) \) if \( q \leq 3 \) or \( q \geq 11 \).

We need to introduce some notation and lemma first in this section.

As explained in section 2.2, after \( l \) rounds of iterations of MSD, for every bit node \( v_i \), the cost functions \( \mu_i^{(l)}(a) \) are computed based on the messages (intermediate cost functions) \( \mu_{[j' \rightarrow i]}^{(l)}(a) \) passed from all adjacent check nodes \( f_{j'} \). Since for every check node \( f_{j'} \), \( \mu_{[j' \rightarrow i]}^{(l)}(a) \) is determined by all adjacent bit nodes which are not \( v_i \). So given the messages passed from all bit nodes which have distance 2 to \( v_i \), the cost functions \( \mu_i^{(l)}(a) \) can be rewritten as follows:

\[
\mu_i^{(l)}(a) = \gamma_i(a) + \sum_{j' \in N(i)} \mu_{[j' \rightarrow i]}^{(l)}(a) = \gamma_i(a) + \sum_{j' \in N(i)} \min_{x \in C_{j'}, x_i = a} \left\{ \sum_{i' \in N(j'), i' \neq i} \mu_{[i' \rightarrow j']}^{(l)}(x_{i'}) \right\}.
\]
For the Tanner graphs whose girth are at least 6, any two bit nodes can share at most one common adjacent check node. In this case the functions \( \mu_{[i' \rightarrow j']}^{(l)} \) sent from the bit node \( v_{i'} \) to its neighboring check node \( f_{j'} \) can be denoted by \( \mu_{[i' \rightarrow i]}^{(l)} \) without confusion, where \( v_i \) is any adjacent bit node of \( f_{j'} \) which is not \( v_{i'} \). Thus we can rewrite the cost functions \( \mu_i^{(l)}(a) \) as follows.

\[
\mu_i^{(l)}(a) = \gamma_i(a) + \sum_{j' \in N(i)} \min_{x \in C_{j'}, x_i = a} \left\{ \sum_{i' \in N(j'), i' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\}.
\]

Similarly, the messages sent in the next iteration can also be rewritten as follows, where \( v_i \) and \( v_k \) are two bit nodes such that the distance between them is 2 and \( f_j \) is their common adjacent check node.

\[
\mu_{[i \rightarrow k]}^{(l+1)}(a) = \gamma_i(a) + \sum_{j' \in N(i), j' \neq j} \min_{x \in C_{j'}, x_i = a} \left\{ \sum_{i' \in N(j'), i' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\}.
\]

Notice that although \( \mu_i^{(l)}(a) \) are the cost functions computed after \( l \) rounds of iterations and \( \mu_{[i \rightarrow k]}^{(l+1)}(a) \) can be regarded as the messages passed from the bit node \( v_i \) to \( v_k \) along Tanner graph in \( l + 1 \) rounds of iterations, their updating rules are very similar. The only difference between them is that \( \mu_{[i \rightarrow k]}^{(l+1)}(a) \) does not count the messages passed from the neighboring bit nodes of \( f_j \).

As in section 2.1, \( \mu_{[i \rightarrow k]}^{(0)} \) are set to zero at round 0. The local cost functions \( \gamma_i(y_i) \) are normalized so that \( \gamma_i(0) = 0 \) for all bit nodes \( v_i \), and

\[
\gamma_i(1) = \begin{cases} 
-1 & \text{if } \tilde{y}_i = 1, \\
1 & \text{if } \tilde{y}_i = 0. 
\end{cases}
\]
Note that all $\mu^{(l)}_i(a)$ and $\mu^{(l+1)}_{[i \rightarrow k]}(a)$ are integer-valued functions.

To simplify the updating rules of $\mu^{(l)}_i$ and $\mu^{(l+1)}_{[i \rightarrow k]}$, we introduce new functions $\text{Diff}_i^{(l)}$, $\text{Diff}_{i \rightarrow k}^{(l+1)}$, $\text{Res}_i^{(l)}$ and $\text{Res}_{i \rightarrow k}^{(l+1)}$. Here $\text{Diff}_i^{(l)}$ is the absolute value of the difference of $\mu^{(l)}_i(0)$ and $\mu^{(l)}_i(1)$. And $\text{Res}_i^{(l)}$ is the result of decoding at the bit node $i$ after $l$ rounds of iterations. The precise definitions are given below.

**Definition 4.1.**

\[
\text{Diff}_i^{(l)} = |\mu^{(l)}_i(0) - \mu^{(l)}_i(1)|, \quad \text{Diff}_{i \rightarrow k}^{(l+1)} = |\mu^{(l+1)}_{[i \rightarrow k]}(0) - \mu^{(l+1)}_{[i \rightarrow k]}(1)|, \\
\]

\[
\text{Res}_i^{(l)} = \begin{cases} 
0 & \text{if } \mu^{(l)}_i(0) < \mu^{(l)}_i(1) \\
1 & \text{if } \mu^{(l)}_i(0) > \mu^{(l)}_i(1) \\
N & \text{if } \mu^{(l)}_i(0) = \mu^{(l)}_i(1) 
\end{cases}
\]

and

\[
\text{Res}_{i \rightarrow k}^{(l+1)} = \begin{cases} 
0 & \text{if } \mu^{(l+1)}_{[i \rightarrow k]}(0) < \mu^{(l+1)}_{[i \rightarrow k]}(1) \\
1 & \text{if } \mu^{(l+1)}_{[i \rightarrow k]}(0) > \mu^{(l+1)}_{[i \rightarrow k]}(1) \\
N & \text{if } \mu^{(l+1)}_{[i \rightarrow k]}(0) = \mu^{(l+1)}_{[i \rightarrow k]}(1) 
\end{cases}
\]

Note that when $\text{Res}_i^{(l)} = N$, which means that $\mu^{(l)}_i(0) = \mu^{(l)}_i(1)$, a decoding error may occur. The updating rules of $\mu^{(l)}_i$ and $\mu^{(l+1)}_{[i \rightarrow k]}$ and the above definitions yield the following lemma.

**Lemma 4.1.** Given a bit node $v_i$, let $S$ be the set of all check nodes $j' \in N(i)$ such that $\text{Diff}_{[i' \rightarrow i]}^{(l)} \neq 0$, i.e. $\text{Diff}_{[i' \rightarrow i]}^{(l)} \geq 1$, for all $i' \in N(j')$ and $i' \neq i$.

Let $S_0$ and $S_1$ be the two subset of $S$ which contain all $j' \in S$ such that
\[ \sum_{j' \in N(j), j' \neq i} {\text{Res}}_{[i' \rightarrow i]}^{(l)} \equiv 0 \mod 2 \text{ and } \equiv 1 \mod 2, \text{ respectively. Obviously} \]

\[ S = S_0 + S_1 \text{ and } S_0 \cap S_1 = \emptyset. \text{ Then the following statements hold.} \]

(1) \[ \mu_i^{(l)}(1) - \mu_i^{(l)}(0) = \sum_{j' \in S_0} \min_{j' \in N(j'), j' \neq i} \{ \text{Dif}_{[i' \rightarrow i]}^{(l)} \} + \gamma_i(1) \]

\[ - \sum_{j' \in S_1} \min_{j' \in N(j'), j' \neq i} \{ \text{Dif}_{[i' \rightarrow i]}^{(l)} \}. \]

(2) If \( S = S_0 \) and \( |S| \geq 2 \), then \( \text{Dif}_{[i' \rightarrow i]}^{(l)} = \gamma_i(1) + \sum_{j' \in S} \min_{j' \in N(j'), j' \neq i} \{ \text{Dif}_{[i' \rightarrow i]}^{(l)} \}, \)

and \( \text{Res}_{[i' \rightarrow i]}^{(l)} = 0 \).

(3) If \( S = S_1 \) and \( |S| \geq 2 \), then \( \text{Dif}_{[i' \rightarrow i]}^{(l)} = -\gamma_i(1) + \sum_{j' \in S} \min_{j' \in N(j'), j' \neq i} \{ \text{Dif}_{[i' \rightarrow i]}^{(l)} \}, \)

and \( \text{Res}_{[i' \rightarrow i]}^{(l)} = 1 \).

For all \( j \in N(i) \) and \( i' \in N(j'), j' \neq i \), if \( S_0, S_1, \) and \( S \) are replaced by \( S_0 \setminus \{j\}, S_1 \setminus \{j\}, \) and \( S \setminus \{j\} \) in (1) (2) (3), respectively, the same conclusion holds for corresponding \( \mu_{[i' \rightarrow i]}^{(l+1)}(1) - \mu_{[i' \rightarrow i]}^{(l+1)}(0), \text{Dif}_{[i' \rightarrow i]}^{(l+1)} \) and \( \text{Res}_{[i' \rightarrow i]}^{(l+1)} \).

**Proof.** First we note that if some check node \( j' \in N(i) \) is not in \( S \), that is, \( \mu_{[i' \rightarrow i]}^{(l)}(0) = \mu_{[i' \rightarrow i]}^{(l)}(1) \) for some \( i' \in N(j') \) and \( i' \neq i \) by the definition of function \( \text{Dif}_{[i' \rightarrow i]}^{(l)} \), then

\[ \min_{x \in C_j, x_i=0} \left\{ \sum_{i' \in N(j'), j' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\} = \min_{x \in C_j, x_i=1} \left\{ \sum_{i' \in N(j'), j' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\}. \]

Here \( C_j = \{ x \in \mathbb{F}_2^n \mid \sum_{i \in N(j)} x_i = 0 \} \) which is defined in section 2.2.

It’s also easy to get the following two results.

\[ \sum_{j' \in S_0} \min_{x \in C_{j', x_i=1}} \left\{ \sum_{i' \in N(j'), j' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\} - \sum_{j' \in S_0} \min_{x \in C_{j', x_i=0}} \left\{ \sum_{i' \in N(j'), j' \neq i} \mu_{[i' \rightarrow i]}^{(l)}(x_{i'}) \right\} \]
\[= \sum_{j' \in S_0} \min_{i' \in N(j'), i' \neq i} \{ \text{Dif}^{(l)}_{[i' \rightarrow i]} \}, \]

and

\[\sum_{j' \in S_1} \min_{x \in C(j')} \left\{ \sum_{i' \in N(j'), i' \neq i} \mu^{(l)}_{[i' \rightarrow i]}(x_{i'}) \right\} - \sum_{j' \in S_1} \min_{x \in C(j')} \left\{ \sum_{i' \in N(j'), i' \neq i} \mu^{(l)}_{[i' \rightarrow i]}(x_{i'}) \right\} \]

\[= \sum_{j' \in S_1} \min_{i' \in N(j'), i' \neq i} \{ \text{Dif}^{(l)}_{[i' \rightarrow i]} \}. \]

Combine all the above results, (1) of this lemma is proved. Since (2) and (3) are special cases of (1), they are proved straightforwardly.

And similarly, (4) is also true.

\[\square\]

### 4.3 The maximal allowable error of \( LU(2, q) \) over BSC

As introduced in section 4.1, the Tanner graph \( D(2, q) \) of the code \( LU(2, q) \) is defined as follows.

**Definition 4.2.** In \( D(2, q) \), the Tanner graph of \( LU(2, q) \), the set of all bit nodes and the set of all check nodes are both \( \mathbb{F}^2_q \). A bit node \((x, x_1)\) is adjacent to a check node \((y, y_1)\) if and only if \( y_1 = xy + x_1 \), where \( x, x_1, y, y_1 \) are in \( \mathbb{F}_q \).

The girth of \( D(2, q) \) is 6 for all \( q \). If \( q = 2 \), \( LU(2, q) \) is a cycle code.

So by Theorem 3.1, the maximal allowable error of \( LU(2, 2) \) over BSC is 1. Hence assume \( q \neq 2 \) for the rest of this section.
In order to find the maximal allowable error \( r \) of \( LU(2, q) \), we first get its lower bound \( \frac{q-1}{2} \) from Theorem 3.2. The major work is to investigate the decoding process of this code and bound \( r \) from above. Our strategy is to find a set of bit nodes for \( LU(2, q) \) such that if the bit nodes in this set are flipped, then it will cause decoding error after any number of rounds of iterations, in other words, for any \( l > 0 \), not all cost functions satisfy \( \mu_i^{(l)}(1) > \mu_i^{(l)}(0) \). To achieve this, we shall partition the bit nodes into sets, labeled by A, B, C, etc, so that for bit nodes in the same set, the values of their cost functions are close to each other after some iterations. We then observe that there are two sets of bit nodes which will be decoded to different values (0,1 or 1,0) after any finite number of rounds of iterations. So MSD fails in this case and an upper bound for \( r \) is obtained.

To prove Theorem 4.1 for \( LU(2, q) \), we distinguish two cases: \( q \) odd and \( q \) even.

Case 1: \( q \) is odd.

Divide all bit nodes into the following three sets:

- set \( A \) consists of any choice of \( \frac{q+3}{2} \) bit nodes of the form \((0, b)\), where \( b \) is in \( F_q \),

- set \( B \) consists of the remaining \( q - \frac{q+3}{2} \) bit nodes of the same form,
• set $C$ consists of the remaining $q^2 - q$ bit nodes.

Note that for any check node $(y, y_1)$, it is adjacent to a unique bit node $(x, x_1) = (0, y_1)$ in set $A \cup B$ by Definition 4.2. Thus all other adjacent bit nodes of $(y, y_1)$ are in set $C$. So we obtain the following lemma straightforwardly.

**Lemma 4.2.** (1). For any bit node in $A \cup B$, all the bit nodes which have distance 2 to it are in $C$.

(2). For any bit node in $C$, all the $q$ bit nodes in $A \cup B$ have distance 2 to it, corresponding to the $q$ different adjacent check nodes, and all the remaining bit nodes within distance 2 are in $C$.

Since for any bit node $v_i$, the cost function $\mu_i^{(l)}$ and $\mu_{[i\rightarrow k]}^{(l+1)}$ depend on all bit nodes which has distance 2 to it and all these bit nodes excepts ones who share the same adjacent check node with $k$ and $i$, respectively, it’s easy to understand the updating rule of $\mu_i^{(l)}$ and $\mu_{[iightarrow k]}^{(l+1)}$ by Lemma 4.2.

Suppose all bit nodes in $A$ are flipped, i.e., $\gamma_A(1) = -1$, and $\gamma_B(1) = \gamma_C(1) = 1$. Here we abused notation by denoting any bit node in $A$, $B$, $C$ again by $A$, $B$, $C$, resp., when there is no danger of confusion. Let $A \cup B$ denote a bit node in $A$ or $B$. The following lemma shows that bit nodes in $A \cup B$ and bit nodes in $C$ will be decoded to 0 and 1, respectively, after each iteration.
Lemma 4.3. For \( l \geq 1 \), \( \text{Res}^{(l)}_{A \cup B} = \text{Res}^{(l+1)}_{[A \cup B \to C]} = 0 \) and \( \text{Res}^{(l)}_C = \text{Res}^{(l+1)}_{[C \to A \cup B]} = 1 \).

Proof. The initial functions are \( \mu^{(1)}_{[A \to C]}(1) = \gamma_A(1) = -1 \), \( \mu^{(1)}_{[B \to C]}(1) = \gamma_B(1) = 1 \), \( \mu^{(1)}_{[C \to A \cup B]}(1) = \gamma_C(1) = 1 \), and \( \mu^{(1)}_{[A \to C]}(0) = \mu^{(1)}_{[B \to C]}(0) = \mu^{(1)}_{[C \to A \cup B]}(0) = 0 \) by the assumption of normalized initial cost functions and \( \gamma_i(0) = 0 \) for any bit node \( v_i \).

When \( l = 1 \), it follows from Lemma 4.2 and the updating rules of \( \mu^{(l)}_i \) and \( \mu^{(l+1)}_{i \to k} \) that \( \text{Res}^{(1)}_{A \cup B} = \text{Res}^{(2)}_{[A \cup B \to C]} = 0 \), \( \text{Res}^{(1)}_C = \text{Res}^{(2)}_{[C \to A \cup B]} = 1 \).

Suppose for \( l = k - 1 \geq 1 \), the statements in the lemma hold; consider the case \( l = k \).

By the assumption and Lemma 4.2, it is easy to check that all the functions \( \text{Diff} \) satisfy the conditions (2) or (3) in Lemma 4.1, we then get the following results:

\[
\begin{align*}
\text{Diff}^{(k)}_{A \cup B} & \geq -1 + q \min \{ \text{Diff}^{(k)}_{[C \to A \cup B]} \}, \\
\text{Diff}^{(k+1)}_{[A \cup B \to C]} & \geq -1 + (q - 1) \min \{ \text{Diff}^{(k)}_{[C \to A \cup B]} \}, \\
\text{Diff}^{(k)}_C & \geq -1 + q \min \{ \text{Diff}^{(k)}_{[A \cup B \to C]}, \text{Diff}^{(k)}_{[C \to A \cup B]} \}, \\
\text{Diff}^{(k+1)}_{[C \to A \cup B]} & \geq -1 + (q - 1) \min \{ \text{Diff}^{(k)}_{[A \cup B \to C]}, \text{Diff}^{(k)}_{[C \to A \cup B]} \},
\end{align*}
\]

\( \text{Res}^{(k)}_{A \cup B} = \text{Res}^{(k+1)}_{[A \cup B \to C]} = 0 \), and

\( \text{Res}^{(k)}_C = \text{Res}^{(k+1)}_{[C \to A \cup B]} = 1 \).

Here \( \min \{ \text{Diff}^{(k)}_{[C \to A \cup B]} \} \) denotes the minimal value of all functions in the
form $Dif_{[C\to A\cup B]}^{(k)}$. All other \textit{min} functions in this proof and the proofs of Lemma 4.4 and 4.8 are defined in the similar way.

The lemma is then proved by induction.

\[ \square \]

Case 2: $q$ is even.

The bit nodes are divided into three sets as in the first case with $\frac{q+3}{2}$ replaced by $\frac{q+2}{2}$. Lemma 4.2 also holds for this case obviously. Again we assume all bit nodes in $A$ are flipped. Then the following lemma shows that bit nodes in $A \cup B$ and bit nodes in $C$ will be decoded to different values after each iteration.

\textbf{Lemma 4.4.} For $l \geq 2$, $Res_{A\cup B}^{(l)} = Res_{A\cup B\to C}^{(l+1)} = l+1 \mod 2$ and $Res_{C}^{(l)} = Res_{C\to A\cup B}^{(l+1)} = l \mod 2$.

\textit{Proof.} All the initial functions are the same as those in the first case.

When $l = 1$, we have $Res_{A\cup B}^{(1)} = Res_{A\cup B\to C}^{(2)} = 0, Res_{C}^{(1)} = Res_{C\to B}^{(2)} = 1, Res_{C\to A}^{(1)} = -1$.

When $l = 2$, $Res_{A\cup B}^{(2)} = Res_{A\cup B\to C}^{(3)} = 1, Res_{C}^{(2)} = Res_{C\to A\cup B}^{(3)} = 0$.

Suppose for $l = k - 1 \geq 2$, the statements in the lemma hold; consider the case $l = k$.

All the functions $Dif$ are the same as those in the proof of the first case,
then
\[
Res_{A \cup B}^{(k)} = Res_{[A \cup B \rightarrow C]}^{(k+1)} = \begin{cases} 
0 & \text{if } Res_{[A \cup B \rightarrow C]}^{(k-1)} = 1, \text{ i.e., } k \text{ is odd} \\
1 & \text{if } Res_{[A \cup B \rightarrow C]}^{(k-1)} = 0, \text{ i.e., } k \text{ is even}
\end{cases}
\]
and
\[
Res_{C}^{(k)} = Res_{[C \rightarrow A \cup B]}^{(k+1)} = \begin{cases} 
1 & \text{if } Res_{[C \rightarrow A \cup B]}^{(k-1)} = 0, \text{ i.e., } k \text{ is odd} \\
0 & \text{if } Res_{[C \rightarrow A \cup B]}^{(k-1)} = 1, \text{ i.e., } k \text{ is even}
\end{cases}
\]
So by induction, the lemma holds.

\[\square\]

As shown in Lemmas 4.3 and 4.4, if the number of errors in the received word is equal to \( \frac{q+3}{2} \) for odd \( q \) or \( \frac{q+2}{2} \) for even \( q \) over BSC, then MSD may fail. So \( \frac{q+1}{2} \) is an upper bound of the maximal allowable error \( r \) of LU(2, q) over BSC. This completes the proof of Theorem 4.1 for code LU(2, q).

### 4.4 The maximal allowable error of LU(3, q) over BSC

The Tanner graph \( D(3, q) \) of the code LU(3, q) is defined as follows.

**Definition 4.3.** In \( D(3, q) \), the Tanner graphs of LU(3, q) codes, the set of all bit nodes and the set of all check nodes are both \( \mathbb{F}_q^3 \). A bit node \((x, x_1, x_2)\) is adjacent to a check node \((y, y_1, y_2)\) if and only if \( y_1 = xy + x_1 \) and \( y_2 = xy_1 + x_2 \), where \( x, x_1, x_2, y, y_1, y_2 \) are in \( \mathbb{F}_q \).
The girth of $D(3, q)$ is 8 for all $q$. And since $LU(3, 2)$ is a cycle code, the maximal allowable error of $LU(3, 2)$ over BSC is 1 by Theorem 3.1. Hence assume $q \neq 2$. We obtain a lower bound $\frac{q-1}{2}$ for the maximal allowable error $r$ of $LU(3, q)$ by Theorem 3.2.

To find an upper bound for $r$, the idea is similar. First we divide the bit nodes into following six sets, where all $\alpha, \beta, a, b, c$ are in $\mathbb{F}_q$.

- Set $A$ for odd $q$ consists of any choice of $\frac{q+3}{2}$ bit nodes of the form $(0, \alpha, \alpha)$, and set $B$ consists of the remaining $q - \frac{q+3}{2}$ bit nodes of the same form. For even $q$, replace $\frac{q+3}{2}$ by $\frac{q+2}{2}$.

- Set $C$ consists of bit nodes of the form $(1, \beta, 0)$.

- Set $D$ consists of bit nodes of the form $(0, b, c)$ with $b \neq c$, and bit nodes $(1, b, c)$ with $c \neq 0$.

- Set $E$ consists of all bit nodes $(a, b, c)$, $a \neq 0, 1$, which has distance 2 to set $A$.

- Set $F$ consists of all bit nodes $(a, b, c)$, $a \neq 0, 1$, which has distance 2 to set $B$.

In order to understand these six bit nodes set, we prove Lemma 4.5 first.
Lemma 4.5. (a) The distance between any two bit nodes in $A \cup B$ and $C$, is 2.

(b). If one neighboring bit node of a check node is in $A \cup B$ (resp. $C$), then this check node must be adjacent to a bit node in $C$ (resp. $A \cup B$) and is not adjacent to any other bit nodes in $D, C$, and $A \cup B$.

(c). If a bit node has distance 2 to two bit nodes in $A \cup B$, then it is a bit node in $C$.

Proof. (a). Given a bit node $(0, \alpha, \alpha)$ in $A \cup B$ and a bit node $(1, \beta, 0)$ in $C$, the check node $(\alpha - \beta, \alpha, \alpha)$ is adjacent to both of them by Definition 4.3, that is, both of the two pairs $((x, x_1, x_2) = (0, \alpha, \alpha), (y, y_1, y_2) = (\alpha - \beta, \alpha, \alpha))$ and $((x, x_1, x_2) = (1, \beta, 0), (y, y_1, y_2) = (\alpha - \beta, \alpha, \alpha))$ satisfy the equations $y_1 = xy + x_1$ and $y_2 = xy_1 + x_2$, respectively.

(b). Obviously, both of $A \cup B$ and $C$ contain $q$ bit nodes. Since the degree of graph $D(3, q)$ is $q$, each bit node has $q$ adjacent check nodes. By (a), if one neighboring bit node of a check node is in $A \cup B$ (resp. $C$), then this check node must be adjacent to a bit node in $C$ (resp. $A \cup B$). And if a check node is adjacent to a bit node $(0, \alpha, \alpha)$ in $A \cup B$ and a bit node $(1, \beta, 0)$ in $C$, then it is not adjacent to another bit node of the form $(a, b, c)$ with $a = 0, 1$ by Definition 4.3. This completes the proof of (b).

(c). By the results of (a) and (b), the set of all adjacent check nodes of a
bit node \((0, \alpha, \alpha)\) in \(A \cup B\) is \(\{(a, \alpha, \alpha) | a \in \mathbb{F}_q\}\). So if a bit node has distance 2 to two bit nodes \((0, \alpha, \alpha)\) and \((0, \alpha', \alpha')\) in \(A \cup B\), where \(\alpha \neq \alpha'\), then this bit node must be adjacent to two check nodes of the form \((a, \alpha, \alpha)\) and \((a', \alpha', \alpha')\). Thus by Definition 4.3, this bit node must has the form \((1, \beta, 0)\). So it is a bit node \(C\). 

Obviously, the union of the sets \(A, B, C\) and \(D\) is the set of all the bit nodes of the form \((a, b, c)\) with \(a = 0, 1\). And by lemma 4.5, it’s easy to prove that the sets \(E\) and \(F\) are disjoint, and their union is the set of all bit nodes of the form \((a, b, c)\) with \(a \neq 0, 1\). In order to understand the structure of the graph \(D(3, q)\) better, we observe that the check nodes are divided into three groups automatically.

**Lemma 4.6.** Any check node of \(LU(3, q)\) belongs to one of the following types.

1. One adjacent bit node is in \(A\), one in \(C\), and the remaining ones in \(E\);

2. One adjacent bit node is in \(B\), one in \(C\), and the remaining ones in \(F\);

3. Two adjacent bit nodes are in \(D\), and the remaining ones in \(E\) or \(F\). If \(q \geq 11\), then at least two adjacent bit nodes are in \(E\) and two in \(F\).
Proof. By Lemma 4.5, it is straightforward to get the three types of check nodes. Thus it remains to show that the numbers of the adjacent bit nodes in $E$ and in $F$ of any check node of type (3) are both at least 2 for the code $LU(3, q)$ when $q \geq 11$.

Here we only show the proof of the case of odd $q$. For even $q$, the proof is almost the same.

Since there are $\frac{q+3}{2}$ bit nodes in $A$, we divide all bit nodes in $E$ into $\frac{q+3}{2}$ types such that any bit node in $A$ has distance 2 to only one type of bit nodes in $E$. And similarly, we divide all bit nodes in $F$ into $\frac{q-3}{2}$ types corresponding to the $\frac{q-3}{2}$ bit nodes in $B$. Then it is easy to prove that in the adjacent bit nodes of any check node of type (3), there is at most one bit node for each type of bit nodes in $E$ and in $F$. Otherwise, the girth is at most 6 which results in a contradiction. Then the numbers of the adjacent bit nodes in $E$ and in $F$ of any check node of type (3) are at most $\frac{q+3}{2}$ and $\frac{q-3}{2}$. It is straightforward to verify from the definition of type (3) check node that the total number of its adjacent bit nodes in $E$ and $F$ is $q - 2$. So the numbers of the adjacent bit nodes in $E$ and $F$ of a type (3) check node are both at least 2 for the code $LU(3, q)$ when $q \geq 11$.

By Lemma 4.6 and Lemma 4.5 (3), we derive the following lemma. Then by this lemma and Lemma 4.6, it’s easy to understand the updating rule of
\[ \mu_i^{(l)} \text{ and } \mu_{[i \rightarrow k]}^{(l+1)} \]

**Lemma 4.7.** (a) All adjacent check nodes of a bit node in \( A \) are of type (1).

(b) All adjacent check nodes of a bit node in \( B \) are of type (2).

(c) For odd \( q \), \( \frac{q + 3}{2} \) adjacent check nodes of a bit node in \( C \) are of type (1), the remaining \( \frac{q - 3}{2} \) ones are of type (2). For even \( q \), replace \( \frac{q + 3}{2} \) by \( \frac{q + 2}{2} \).

(d) All adjacent check nodes of a bit nodes in \( D \) are of type (3).

(e) One of the adjacent check node of a bit node in \( E \) is of type (1), the remaining \( q - 1 \) ones are of type (3).

(f) One of the adjacent check node of a bit node in \( F \) is of type (2), the remaining \( q - 1 \) ones are of type (3).

Now assume that all bit nodes in \( A \) are flipped. We show that the bit nodes in \( A \cup B \) and bit nodes in \( C \) are the two sets of bit nodes which will be decoded to different values after each iteration.

**Lemma 4.8.** If \( q \geq 11 \), then the following statements hold for \( l \geq 1 \) when \( q \) is odd, and for \( l \geq 2 \) when \( q \) is even:

\[ Res_{A}^{(l)} = Res_{[A \rightarrow C]}^{(l+1)} = Res_{B}^{(l)} = Res_{[B \rightarrow C]}^{(l+1)} = l + 1 \mod 2, \]

\[ Res_{C}^{(l)} = Res_{[C \rightarrow A]}^{(l+1)} = Res_{[C \rightarrow B]}^{(l+1)} = l \mod 2, \]
\[
\begin{align*}
Res^{(l+1)}_{[D \rightarrow D]} &= Res^{(l+1)}_{[E \rightarrow A]} = Res^{(l+1)}_{[E \rightarrow D]} = Res^{(l+1)}_{[F \rightarrow B]} = Res^{(l+1)}_{[F \rightarrow D]} = 0, \\
Dif^{(l+1)}_{[E \rightarrow A]} &= Dif^{(l+1)}_{[F \rightarrow B]} < \frac{2q - 2}{q - 2} \min\{Dif^{(l+1)}_{[E \rightarrow D]}, Dif^{(l+1)}_{[F \rightarrow D]}\}, \\
Dif^{(l+1)}_{[D \rightarrow D]} &\geq \min\{Dif^{(l+1)}_{[E \rightarrow D]}, Dif^{(l+1)}_{[F \rightarrow D]}\},
\end{align*}
\]

and the values of all \( Dif \) functions which correspond to the same type bit nodes are the same.

Proof. For odd \( q \) and \( l = 1 \), it is easy to prove that the statements hold from Lemma 4.7 and the updating rules of \( \mu_i^{(l)} \) and \( \mu_i^{(l+1)} \). In particular,

\[ Dif^{(2)}_{[E \rightarrow A]} = Dif^{(2)}_{[F \rightarrow B]} = q, \ Dif^{(2)}_{[E \rightarrow D]} = q - 2, \ \text{and} \ Dif^{(2)}_{[F \rightarrow D]} = q. \]

For even \( q \) and \( l = 1 \), all the statements hold except \( Dif^{(2)}_{[C \rightarrow A]} = 0 \), i.e., \( Res^{(2)}_{[C \rightarrow A]} = -1 \). When \( l = 2 \), by the updating rules, all the statements hold and \( Dif^{(3)}_{[E \rightarrow A]} = Dif^{(3)}_{[F \rightarrow B]} = (q - 1)(q - 2) + 1, \ Dif^{(3)}_{[E \rightarrow D]} = (q - 2)^2 + 1, \) and \( Dif^{(3)}_{[F \rightarrow D]} = (q - 2)^2 - 1. \)

Suppose all the statements hold for \( l = k - 1 \geq 1 \) (for odd \( q \)) or \( l = k - 1 \geq 2 \) (for even \( q \)). Consider the case \( l = k \). By assumption and Lemma 4.7, either the condition (2) or (3) in Lemma 4.1 holds for any \( Dif \) functions except \( Dif^{(k+1)}_{[E \rightarrow D]} \) and \( Dif^{(k+1)}_{[F \rightarrow D]} \), it is straightforward to derive the following results in which \( s = \begin{cases} -1 & \text{if } Res^{(k-1)}_{A} = 0, \ \text{i.e., } k \text{ is even} \\ 1 & \text{if } Res^{(k-1)}_{A} = 1, \ \text{i.e., } k \text{ is odd} \end{cases} \) and \( p = \frac{q + 3}{2} \) for odd \( q \) and \( p = \frac{q + 2}{2} \) for even \( q \):

\[
\begin{align*}
\text{Diff}^{(k)}_{A} &= -s + q \min\{\text{Diff}^{(k)}_{[E \rightarrow A]}, \text{Diff}^{(k)}_{[C \rightarrow A]}\}, \\
\text{Diff}^{(k+1)}_{[A \rightarrow C]} &= -s + (q - 1) \min\{\text{Diff}^{(k)}_{[E \rightarrow A]}, \text{Diff}^{(k)}_{[C \rightarrow A]}\},
\end{align*}
\]

45
\[Dif_B^{(k)} = s + q \min\{Dif_{[F \to B]}^{(k)}, Dif_{[C \to B]}^{(k)}\},\]
\[Dif_{[B \to C]}^{(k+1)} = s + (q - 1) \min\{Dif_{[F \to B]}^{(k)}, Dif_{[C \to B]}^{(k)}\},\]
\[Dif_C^{(k)} = -s + p \min\{Dif_{[E \to A]}^{(k)}, Dif_{[A \to C]}^{(k)}\}\]
\[+ (q - p) \min\{Dif_{[F \to B]}^{(k)}, Dif_{[B \to C]}^{(k)}\},\]
\[Dif_{[C \to A]}^{(k+1)} = -s + (p - 1) \min\{Dif_{[E \to A]}^{(k)}, Dif_{[A \to C]}^{(k)}\}\]
\[+ (q - p) \min\{Dif_{[F \to B]}^{(k)}, Dif_{[B \to C]}^{(k)}\},\]
\[Dif_{[C \to B]}^{(k+1)} = -s + p \min\{Dif_{[E \to A]}^{(k)}, Dif_{[A \to C]}^{(k)}\}\]
\[+ (q - p) \min\{Dif_{[F \to B]}^{(k)}, Dif_{[B \to C]}^{(k)}\},\]
\[Dif_{[D \to D]}^{(k+1)} = 1 + (q - 1) \min\{Dif_{[D \to D]}^{(k)}, Dif_{[E \to D]}^{(k)}, Dif_{[F \to D]}^{(k)}\},\]
\[Dif_{[E \to A]}^{(k+1)} = 1 + (q - 1) \min\{Dif_{[D \to D]}^{(k)}, Dif_{[E \to D]}^{(k)}, Dif_{[F \to D]}^{(k)}\},\]
\[Dif_{[F \to B]}^{(k+1)} = 1 + (q - 1) \min\{Dif_{[D \to D]}^{(k)}, Dif_{[E \to D]}^{(k)}, Dif_{[F \to D]}^{(k)}\}.\]

And all the functions \(Res\) except \(Res_{[E \to D]}^{(k+1)}\) and \(Res_{[F \to D]}^{(k+1)}\) hold.

Now it suffices to prove that

\[Dif_{[E \to A]}^{(k+1)} = Dif_{[F \to B]}^{(k+1)} < \frac{2q - 2}{q - 2} \min\{Dif_{[E \to D]}^{(k+1)}, Dif_{[F \to D]}^{(k+1)}\},\]
\[Res_{[E \to D]}^{(k+1)} = Res_{[F \to D]}^{(k+1)} = 0,\]
\[Dif_{[D \to D]}^{(k+1)} \geq \min\{Dif_{[E \to D]}^{(k+1)}, Dif_{[F \to D]}^{(k+1)}\},\]

all the functions in the form \(Dif_{[E \to D]}^{(k+1)}\) have the same value and all the functions in the form \(Dif_{[F \to D]}^{(k+1)}\) have the same value.

Since \(Dif_{[D \to D]}^{(k+1)} \geq \min\{Dif_{[E \to D]}^{(k)}, Dif_{[F \to D]}^{(k)}\}\) by assumption,
\[
\min \{D_i f_{(D \to D)}^{(k)}, D_i f_{(E \to D)}^{(k)}, D_i f_{(F \to D)}^{(k)}\} = \min \{D_i f_{(E \to D)}^{(k)}, D_i s_{(F \to D)}^{(k)}\}.
\]

So \(D_i f_{(E \to A)}^{(k+1)} = D_i f_{(F \to B)}^{(k+1)} = 1 + (q - 1) \min \{D_i f_{(E \to D)}^{(k)}, D_i f_{(F \to D)}^{(k)}\}\).

By (1) of lemma 4.1,

\[
\mu_{(E \to D)}^{(k+1)}(1) - \mu_{(E \to D)}^{(k+1)}(0) = 1 + (q - 2) \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(D \to D)}^{(1)}, D_i f_{(E \to D)}^{(1)}\} - \min \{D_i f_{(A \to C)}^{(1)}, D_i f_{(C \to A)}^{(1)}, D_i f_{(E \to A)}^{(1)}\},
\]

\[
\mu_{(F \to D)}^{(k+1)}(1) - \mu_{(F \to D)}^{(k+1)}(0) = 1 + (q - 2) \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(D \to D)}^{(1)}, D_i f_{(E \to D)}^{(1)}\} - \min \{D_i f_{(B \to C)}^{(1)}, D_i f_{(C \to B)}^{(1)}, D_i f_{(F \to B)}^{(1)}\}.
\]

And since \(D_i f_{(E \to A)}^{(1)} = D_i f_{(F \to B)}^{(1)} < \frac{2q - 2}{q - 2} \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(F \to D)}^{(1)}\}\) by assumption, we have

\[
0 < 1 + (q - 2) \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(F \to D)}^{(1)}\} - \frac{2q - 2}{q - 2} \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(F \to D)}^{(1)}\}
\]

\[
< 1 + (q - 2) \min \{D_i f_{(E \to D)}^{(1)}, D_i f_{(F \to D)}^{(1)}\} - D_i f_{(E \to A)}^{(1)}
\]

\[
\leq \min \{D_i f_{(E \to D)}^{(k+1)}, D_i f_{(F \to D)}^{(k+1)}\},
\]

and all the functions in the form \(D_i f_{(E \to D)}^{(k+1)}\) have the same value and ones in the form \(D_i f_{(F \to D)}^{(k+1)}\) also have the same value.

Thus

\[
Res_{(E \to D)}^{(k+1)} = Res_{(F \to D)}^{(k+1)} = 0
\]

and

\[
D_i f_{(D \to D)}^{(k+1)} \geq \min \{D_i f_{(E \to D)}^{(k+1)}, D_i f_{(F \to D)}^{(k+1)}\}.
\]

47
To prove that
\[ \text{Diff}_{[E \rightarrow A]}^{(k+1)} = \text{Diff}_{[F \rightarrow B]}^{(k+1)} < \frac{2q - 2}{q - 2} \min \{ \text{Diff}_{[E \rightarrow D]}^{(k+1)}, \text{Diff}_{[F \rightarrow D]}^{(k+1)} \}, \]
it suffices to show that \( (q - 1) < \frac{2q - 2}{q - 2} ((q - 2) - \frac{2q - 2}{q - 2}) \) which holds because \( q \geq 11 \).

By Lemma 4.8, if the number of errors is equal to \( \frac{q + 3}{2} \) for odd \( q \) or \( \frac{q + 2}{2} \) for even \( q \), then MSD may fail. So \( \frac{q + 1}{2} \) is an upper bound of the maximal allowable error of \( LU(3, q) \) over BSC. So Theorem 4.1 for \( LU(3, q) \) with \( q \geq 11 \) holds.

If \( q = 3 \), the bit nodes set \( B \) is empty and thus \( E \) is empty. In other words, all bit nodes of the form \((0, a, a)\) are flipped. The graph \( D(3, q) \) is relatively simple. The same method shows that bit nodes set \( A \) and bit nodes set \( C \) will also be decoded to different numbers after any round of iteration. In conclusion, Theorem 4.1 is true for all \( LU(3, q) \) if \( q \leq 3 \) or \( q \geq 11 \). For the cases of \( 3 < q < 11 \), Lemma 4.2 doesn’t hold. So Lemma 4.3 may not hold. However, by computer simulation, we also found that for these cases, set \( A \cup B \) and set \( C \) will be decoded to different numbers in the first 20 rounds of iteration. Thus Theorem 4.1 is true for these cases if we only consider the first 20 rounds of iteration of MSD.
Chapter 5

Ramanujan Graphs and Graphs $D(2, q)$ and $D(3, q)$

5.1 Introduction

Ramanujan graphs were introduced by Lubotzky, Phillips, and Sarnak [8] in 1988. A connected $k$-regular graph is called Ramanujan graph if the largest nontrivial eigenvalue of the adjacency matrix is not greater than $2\sqrt{k-1}$. The extremal spectral property makes Ramanujan graphs optimal expanders, leading to wide applications. In addition, they are sparse and have large girth. Hence they appear to be good candidates for LDPC codes. In 2000 Rosenthal and Vontobel [9] published the first construction of LDPC codes using Ramanujan graphs. In view of the nice properties $D(m, q)$ possess which were introduced in section 4.1, we wonder if they happen to be Ramanujan graphs. Working jointly with my advisor Wen-Ching W. Li, we determine the eigenvalues of graphs $D(2, q)$ and $D(3, q)$ and obtain the
following result [7]. Our paper [7] proves Theorem 5.1 without computing the eigenvalues, and in this thesis we give another proof by exhibiting the eigenvalues explicitly.

**Theorem 5.1.** The largest non-trivial eigenvalue of graph \( D(2, q) \) is \( \sqrt{q} \), and that of graph \( D(3, q) \) is \( \sqrt{2q} \). Consequently, the graphs \( D(2, q) \) and \( D(3, q) \) are Ramanujan graphs.

The details of the proof is shown in sections 5.2 and 5.3. Here we introduce a few steps and two lemmas first.

Let \( H \) be the parity check matrix of a Tanner graph. Then \( A = \begin{pmatrix} 0 & H \\ H^t & 0 \end{pmatrix} \) is the adjacency matrix of the Tanner graph. Since the Tanner graph is bipartite, the eigenvalues of \( A \) are symmetric, namely, \( \pm \lambda \) occur as eigenvalues simultaneously and with the same multiplicity. Hence to prove that the Tanner graph is a Ramanujan graph, it suffices to check that the nontrivial eigenvalues of \( A^2 \) are not greater than \( (2\sqrt{q-1})^2 = 4(q-1) \), where \( A^2 = \begin{pmatrix} HH^t & 0 \\ 0 & H^tH \end{pmatrix} \). As \( H^tH \) and \( HH^t \) have the same eigenvalues, we only need to check the eigenvalues of \( H^tH \).

It's straightforward to check that the following two lemmas hold.

**Lemma 5.1.** Let \( M(m, r) = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix} \) be an \( mr \times mr \) matrix consisting of \( r^2 \) \( m \times m \) submatrices, with \( 0 \) on all diagonal positions.
and \( \mathbf{1} \) on all other positions, where \( \mathbf{0} \) and \( \mathbf{1} \) are the matrices of all 0 entries and all 1 entries, respectively. The following table lists the eigenvalues and corresponding eigenvectors of \( M(m, r) \).

<table>
<thead>
<tr>
<th>eigenvalues</th>
<th>multiplicity</th>
<th>eigenvectors</th>
</tr>
</thead>
</table>
| \( m(r-1) \) | 1 | \(
\begin{pmatrix}
  s \\
  s \\
  \vdots \\
  s
\end{pmatrix}, \text{ where } s \in \mathbb{C}
\) |
| 0 | \( r(m-1) \) | \(
\begin{pmatrix}
  S_1 \\
  S_2 \\
  \vdots \\
  S_r
\end{pmatrix}, \text{ where } S_i = \begin{pmatrix}
  s_{i1} \\
  s_{i2} \\
  \vdots \\
  s_{im}
\end{pmatrix} \in \mathbb{C}^m
\) and \( \sum_{i=1}^{m} s_{ij} = 0 \) for each \( i \) |
| \( -m \) | \( r-1 \) | \(
\begin{pmatrix}
  S_1 \\
  S_2 \\
  \vdots \\
  S_r
\end{pmatrix}, \text{ where } S_i = \begin{pmatrix}
  s_i \\
  s_i \\
  \vdots \\
  s_i
\end{pmatrix} \in \mathbb{C}^m
\) and \( \sum_{i=1}^{r} s_i = 0 \) |

**Lemma 5.2.** Let \( N(m, r) = \begin{pmatrix}
  I & I & \cdots & I \\
  I & I & \cdots & I \\
  \vdots & \vdots & \ddots & \vdots \\
  I & I & \cdots & I
\end{pmatrix} \) be a \( mr \times mr \) matrix consisting of \( r^2 \) \( m \times m \) identity matrices \( I \). The following table lists the eigenvalues and corresponding eigenvectors of \( N(m, r) \).
### Eigenvalues and Eigenvectors

<table>
<thead>
<tr>
<th>Eigenvalues</th>
<th>Multiplicity</th>
<th>Eigenvectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>$m$</td>
<td>$\begin{pmatrix} S \ S \ \vdots \ S \end{pmatrix}$, where $S \in \mathbb{C}^m$</td>
</tr>
<tr>
<td>$0$</td>
<td>$m(r-1)$</td>
<td>$\begin{pmatrix} S_1 \ S_2 \ \vdots \ S_r \end{pmatrix}$, where $S_i \in \mathbb{C}^m$ and $\sum_{i=1}^{m} S_i = 0$</td>
</tr>
</tbody>
</table>

Obviously the matrices $M(m, r)$ and $N(m, r)$ are diagonalizable.

### 5.2 Eigenvalues of $D(2, q)$

In order to determine the eigenvalues of $H^tH$ of graph $D(2, q)$, we first consider the matrix $H^t$ for $D(2, q)$. By definition 4.2, it has rows (resp. columns) indexed by the vertices $[x, x_1]$ (resp. $[y, y_1]$) in the truncated vertex set $X$ (resp. $Y$) of $D(2, q)$. Arrange them by blocks so that vertices of the same first coordinate are in the same block. Then we view $H^t$ as a $q \times q$ block matrix with rows and columns indexed by elements in $F_q$. One checks that the $xy$ entry is the block representing the translation $I_{xy}$, where $I_z$ denotes a $q \times q$ translation matrix on $F_q$ sending $\alpha$ to $\alpha + z$.

**Example 5.1.** For graph $D(2, 3)$,

$$H^t = \begin{pmatrix} I_{0,0} & I_{0,1} & I_{0,2} \\ I_{1,0} & I_{1,1} & I_{1,2} \\ I_{2,0} & I_{2,1} & I_{2,2} \end{pmatrix}$$
We notice that the product $H^tH = qI + M(q,q)$, where $I$ is the $q^2 \times q^2$ identity matrix. Since the only eigenvalue of $qI$ is $q$, and both of the matrices $qI$ and $M(q,q)$ are diagonalizable, the corresponding eigenvalues of $H^tH$ are just the summation of those of $qI$ and $M(q,q)$. The following table describes the eigenvalues of $H^tH$.

<table>
<thead>
<tr>
<th>eigenvalues of $M(q,q)$</th>
<th>eigenvalues of $H^tH = qI + M(q,q)$</th>
<th>multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q^2 - q$</td>
<td>$q^2$</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>$q$</td>
<td>$q^2 - q$</td>
</tr>
<tr>
<td>$-q$</td>
<td>0</td>
<td>$q - 1$</td>
</tr>
</tbody>
</table>

Therefore the eigenvalues of $A$, the adjacency matrix of $D(2,q)$, is obtained as shown in the following table.

<table>
<thead>
<tr>
<th>eigenvalues of $A = \begin{pmatrix} 0 &amp; H \ H^t &amp; 0 \end{pmatrix}$</th>
<th>multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>1</td>
</tr>
<tr>
<td>$-q$</td>
<td>1</td>
</tr>
<tr>
<td>$\sqrt{q}$</td>
<td>$q^2 - q$</td>
</tr>
<tr>
<td>$-\sqrt{q}$</td>
<td>$q^2 - q$</td>
</tr>
<tr>
<td>0</td>
<td>$2(q - 1)$</td>
</tr>
</tbody>
</table>

Hence the largest non-trivial eigenvalue of $D(2,q)$ is $\sqrt{q}$, which is not large than $2\sqrt{q} - 1$ for any prime power $q$. Therefore for any prime power $q$, $D(2,q)$ is a Ramanujan graph.
5.3 Eigenvalues of $D(3, q)$

By definition 4.3, the matrix $H^t$ of $D(3, q)$ is obtained from the matrix of $D(2, q)$ by replacing each 1 in the row $[x, x_1]$ and column $[y, y_1]$ by the matrix $I_{xy}$. Similar to the previous case, we regard $H^t$ as a block matrix with rows and columns indexed by the vertices in the $X$ set and $Y$ set of $D(2, q)$, respectively.

**Example 5.2.** For graph $D(3, 3)$,

$$
H^t = \begin{pmatrix}
I_{0,0} & 0 & 0 & I_{0,0} & 0 & 0 \\
0 & I_{0,1} & 0 & 0 & I_{0,1} & 0 \\
0 & 0 & I_{0,2} & 0 & 0 & I_{0,2} \\
I_{1,0} & 0 & 0 & I_{1,0} & 0 & 0 \\
0 & I_{1,1} & 0 & 0 & I_{1,1} & 0 \\
0 & 0 & I_{1,2} & I_{1,0} & 0 & 0 \\
I_{2,0} & 0 & 0 & I_{2,0} & 0 & 0 \\
0 & I_{2,1} & 0 & 0 & I_{2,1} & 0 \\
0 & 0 & I_{2,2} & I_{2,1} & 0 & 0
\end{pmatrix}
$$

The product matrix $H^tH = qI + B$, where $I$ is the $q^3 \times q^3$ identity matrix and $B$ is a matrix consisting of $q^4 \times q$ blocks $I_{xy} - x_1, y$ if $x \neq y$ and 0 otherwise. We use $(I_{x,y_1-x_1,y})$ to denote the $q^2 \times q^2$ big block of $B$ corresponding to fixed $x$ and $y$.

**Example 5.3.** For $D(3, 3)$,

$$
B = \begin{pmatrix}
0 & (I_{0,y_1-x_1,1}) & (I_{0,y_1-x_1,2}) \\
(I_{1,y_1-x_1,0}) & 0 & (I_{1,y_1-x_1,2}) \\
(I_{2,y_1-x_1,0}) & (I_{2,y_1-x_1,1}) & 0
\end{pmatrix},
$$
where

\[(I_{x\cdot y_1-x_1\cdot y}) = \begin{pmatrix} I_{x\cdot 0\cdot 0\cdot y} & I_{x\cdot 1\cdot 0\cdot y} & I_{x\cdot 2\cdot 0\cdot y} \\ I_{x\cdot 0\cdot 1\cdot y} & I_{x\cdot 1\cdot 1\cdot y} & I_{x\cdot 2\cdot 1\cdot y} \\ I_{x\cdot 0\cdot 2\cdot y} & I_{x\cdot 1\cdot 2\cdot y} & I_{x\cdot 2\cdot 2\cdot y} \end{pmatrix}.\]

In order to find the eigenvalues of the matrix \(B\), we rewrite \(B = D - C\), where all the \(q\) big blocks of \(D\) are \((I_{x\cdot y_1-x_1\cdot y})\) and the only nonzero big blocks of \(C\) are the blocks \((I_{x\cdot y_1-x_1\cdot y})\) with \(x = y\).

**Example 5.4.** For \(D(3,3)\),

\[
D = \begin{pmatrix} (I_{0\cdot y_1-x_1\cdot 0}) & (I_{0\cdot y_1-x_1\cdot 1}) & (I_{0\cdot y_1-x_1\cdot 2}) \\ (I_{1\cdot y_1-x_1\cdot 0}) & (I_{1\cdot y_1-x_1\cdot 1}) & (I_{1\cdot y_1-x_1\cdot 2}) \\ (I_{2\cdot y_1-x_1\cdot 0}) & (I_{2\cdot y_1-x_1\cdot 1}) & (I_{2\cdot y_1-x_1\cdot 2}) \end{pmatrix},
\]

and

\[
C = \begin{pmatrix} (I_{0\cdot y_1-x_1\cdot 0}) & 0 & 0 \\ 0 & (I_{1\cdot y_1-x_1\cdot 1}) & 0 \\ 0 & 0 & (I_{2\cdot y_1-x_1\cdot 2}) \end{pmatrix}.
\]

We then have \(B^2 = D^2 - BC - CB - C^2\), and \(D^2 = q^2I + qM(q, q^2), CB = BC = M(q^2, q), C^2 = qC\). Thus by Lemma 5.1, the matrices \(D^2, CB\) and \(BC\) are diagonalizable and the eigenvalues and corresponding eigenvectors of those are determined. Now it suffices to prove that the matrix \(C\) is also diagonalizable and determine the eigenvalues and eigenvectors of \(C\). Then we can derive those of \(B^2\).

Since \(C^2 = qC\), \(C\) has only two eigenvalues: 0 and \(q\). In order to find the corresponding eigenvectors of \(C\), we notice that \(C = PTP^{-1}\), where \(P\) is the matrix consisting \(q^2\) \(q \times q\) blocks \(I_{-xx}\) on all diagonal positions with \(x = y\) and \(x_1 = y_1\) and 0 otherwise, and \(T\) is the matrix consisting of \(q\) \(q^2 \times q^2\) big
blocks $N(q,q)$ on all diagonal positions with $x = y$ and $0$ otherwise. So if $\alpha$ is an eigenvector of $T$, then $P\alpha$ is an eigenvector of $C$ corresponding to the same eigenvalue. Since the eigenvalues and corresponding eigenvectors of all the diagonal big blocks $N(q,q)$ are determined by Lemma 5.2, we obtain those of the matrices $T$ and $C$ straightforwardly.

So all the eigenvalues and corresponding eigenvectors of $B^2$ are summarized in the following two tables. Here $B^2 = q^2 I + qM(q, q^2) - 2M(q^2, q) - qC$. 
<table>
<thead>
<tr>
<th>Eigenvalues of $B^2$</th>
<th>Eigenvalues $M(q, q^2)$</th>
<th>Eigenvalues $M(q^2, q)$</th>
<th>Multiplicity</th>
<th>Eigenvectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q^2(q - 1)^2$</td>
<td>$q^3 - q$</td>
<td>$q^3 - q^2$</td>
<td>$q$</td>
<td>$\begin{pmatrix} s \ s \ \vdots \ s \end{pmatrix}$, where $s \in \mathbb{C}$</td>
</tr>
<tr>
<td>$q^2$</td>
<td>$-q$</td>
<td>$-q^2$</td>
<td>$q$</td>
<td>$q - 1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\begin{pmatrix} S_1 \ S_2 \ \vdots \ S_q \end{pmatrix}$, where $S_i = \begin{pmatrix} s_i \ s_i \ \vdots \ s_i \end{pmatrix} \in \mathbb{C}q^2$ and $\sum_{i=1}^{q} s_i = 0$</td>
</tr>
<tr>
<td>0</td>
<td>$-q$</td>
<td>0</td>
<td>$0$</td>
<td>$q^2 - q$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\begin{pmatrix} S_1 \ S_2 \ \vdots \ S_q^2 \end{pmatrix}$, where $S_i = \begin{pmatrix} s_i \ s_i \ \vdots \ s_i \end{pmatrix} \in \mathbb{C}q$ and $\sum_{i=mq+1}^{mq+q} s_i = 0$ for all $m \in {0, 1, \ldots, q - 1}$</td>
</tr>
</tbody>
</table>
The eigenvalues of $B^2$, $M(q, q^2)$, and $M(q^2, q)$ are given by:

<table>
<thead>
<tr>
<th>Eigenvalues of $B^2$</th>
<th>Eigenvalues of $M(q, q^2)$</th>
<th>Eigenvalues of $M(q^2, q)$</th>
<th>Multiplicity</th>
<th>Eigenvectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$q$</td>
<td>$q^2 - q$</td>
</tr>
<tr>
<td>$q^2$</td>
<td>$0$</td>
<td>$0$</td>
<td>$q(q - 1)$</td>
<td>$q^2(q - 1)^2$</td>
</tr>
</tbody>
</table>

Then we obtain the eigenvalues of the matrix $H^t H$.

<table>
<thead>
<tr>
<th>Eigenvalues of $B$</th>
<th>Eigenvalues of $H^t H = qI + B$</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q(q - 1)$</td>
<td>$q^2$</td>
<td>$1$</td>
</tr>
<tr>
<td>$0$</td>
<td>$q$</td>
<td>$2(q^2 - q)$</td>
</tr>
<tr>
<td>$\pm q$</td>
<td>$2q$ or $0$</td>
<td>$q^3 - 2q^2 + 2q - 1$</td>
</tr>
</tbody>
</table>

Thus the eigenvalues of the adjacency matrix $A$ are $q, \pm \sqrt{q}, \pm \sqrt{2q}$, and $0$. Since the dimension of $LU(3, q)$, which is the multiplicity of the eigen-
value 0 of the matrix $H$ or $H^t$ of $D(3, q)$, is $\frac{q^4 - 2q^2 + 3q - 2}{2}$ if $q$ is odd and $2^{3t} + 2^{t+1} - 1 - (\frac{1+\sqrt{17}}{2})^{2t} - (\frac{1-\sqrt{17}}{2})^{2t}$ if $q$ is even and $q = 2^t$, which were determined by Sin and Xiang [11] in 2006 and Arslan [13] in 2008, respectively. The multiplicities of the eigenvalues 0 of $A$ for odd $q$ and even $q$ are just twice them, respectively. Hence the multiplicities of the eigenvalues $\sqrt{2q}$ and $-\sqrt{2q}$ of $A$ are determined as shown in the following table.

<table>
<thead>
<tr>
<th>eigenvalues of $A$</th>
<th>multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>1</td>
</tr>
<tr>
<td>$-q$</td>
<td>1</td>
</tr>
<tr>
<td>$\sqrt{2q}$</td>
<td>$\frac{q^4 - 2q^2 + 3q - 2}{2}$ if $q$ is odd, $-2q^2 + (\frac{1+\sqrt{17}}{2})^{2t} + (\frac{1-\sqrt{17}}{2})^{2t}$ if $q$ is even, where $q = 2^t$</td>
</tr>
<tr>
<td>$-\sqrt{2q}$</td>
<td>the same as above</td>
</tr>
<tr>
<td>$\sqrt{q}$</td>
<td>$2(q^2 - q)$</td>
</tr>
<tr>
<td>$-\sqrt{q}$</td>
<td>$2(q^2 - q)$</td>
</tr>
<tr>
<td>0</td>
<td>$2(q^3 + 2q - 1 - (\frac{1+\sqrt{17}}{2})^{2t} - (\frac{1-\sqrt{17}}{2})^{2t}$ if $q$ is odd, $2(q^3 + 2q - 1 - (\frac{1+\sqrt{17}}{2})^{2t} - (\frac{1-\sqrt{17}}{2})^{2t}$ if $q$ is even, where $q = 2^t$</td>
</tr>
</tbody>
</table>

Therefore the largest non-trivial eigenvalue of $D(3, q)$ is $\sqrt{2q}$, which is not large than $2\sqrt{q} - 1$ for any prime power $q$. So $D(3, q)$ is a Ramanujan graph.
Bibliography


Vita

Chenying Wang

Chenying Wang was born in Nanjing in China. She obtained her Bachelor’s degree in Applied Mathematics and Master’s degree in Computer Application Technology from Southeast University (China) in 1999 and 2003, respectively. In August 2003, Chenying Wang was admitted to graduate school of The Pennsylvania State University, majoring in Mathematics. Since then, she is doing mathematical research under the supervision of Prof. Wen-Ching Winnie Li.