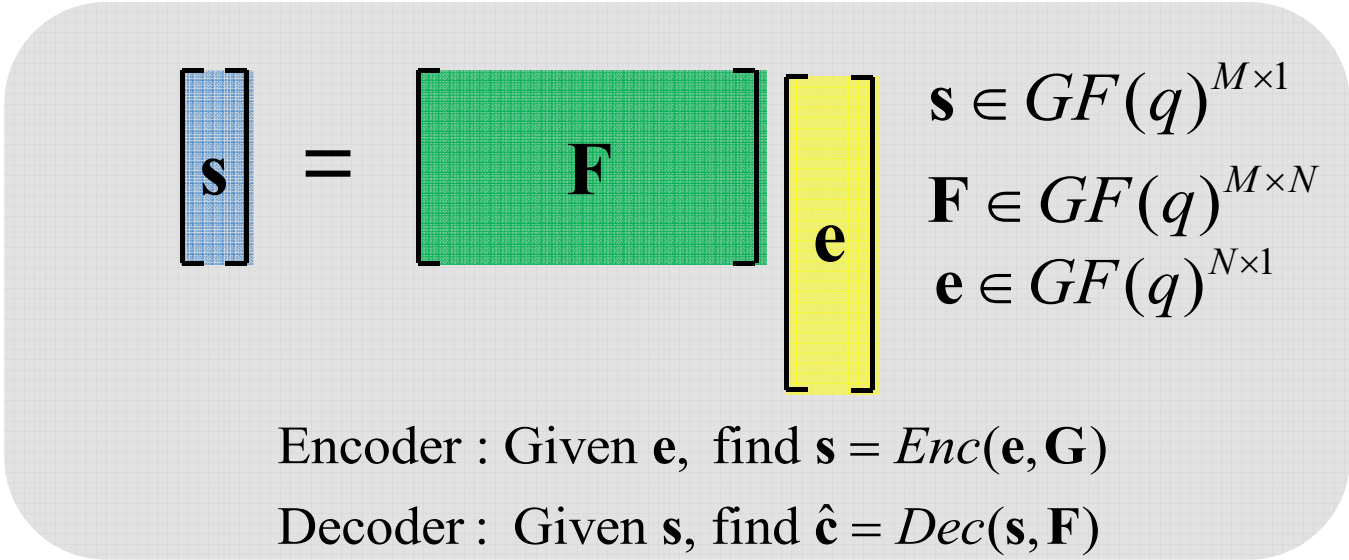
Scalable DeSecure ECCPoW Blockchains 난이도 조절 Algorithm

Heung-No Lee, GIST, South Korea
Home page: http://infonet.gist.ac.kr
Facebook/Publication ID: Heung-No Lee
E-mail: heungno@gist.ac.kr

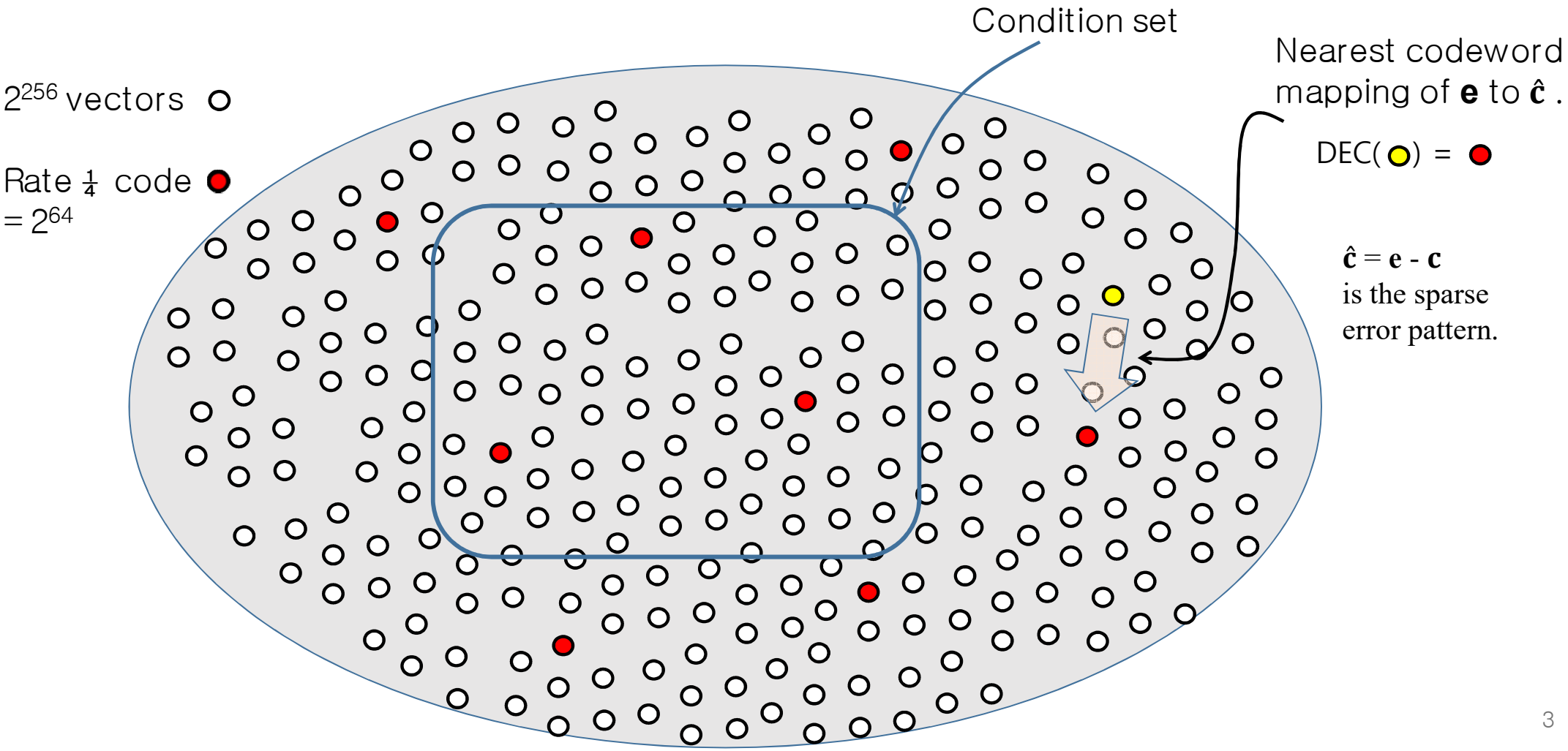
Block code, encoder and decoder

- ENC implies the encoder function, i.e., ENC takes the message vector **m** as the input and produces a codeword vector corresponding to it, e.g. **c** = ENC(**G**, **m**).

- DEC implies the decoding function; DEC takes an arbitrary vector e and returns a closest codeword **c** □i.e., **c** ⊟DEC(**F**, **e**).



Geometrical Explanation



Block code

- A block code C(N, Rate, **G**, **F**, ENC, DEC, GF(q)) is well defined as a collection of codewords. When, $q = 2$, it is a binary system.
- N is the dimension of the code (e.g. $N = 512$)
- Rate = $(N - M)/N$ is the rate of the code, where M < N.
- For example, with $N = 1024$ and $M = 256$, Rate = 3/4.

- **G** is the Generator matrix with dimension $N \times (N - M)$.
- **F** is the Check matrix with dimension, $M \times N$.
- **G** and **F** are orthogonal to each other, i.e., $\mathbf{FG} = \mathbf{0}$.

- A message vector m is an $(N - M) \times 1$ vector.
- A codeword **c**, an $N \times 1$ vector, is an element of the code and can be generated by multiplying a message vector m to the Generator matrix **G**, i.e., $\mathbf{c} = \mathbf{Gm}$.

- Galois Field of size q, GF(q), is used for addition and multiplication operations and storage of numbers in the system.

Decoder

- DEC is to find a codeword $\mathbf{c}$ most close to the input word $\mathbf{e}$.
- For the concept of distance, the Hamming distance can be used.
  For example, $DH(\mathbf{e}, \mathbf{c}) = |\mathbf{e} - \mathbf{c}|_0$ is the number of non-zero values in the $(\mathbf{e} - \mathbf{c})$ vector.

- There are many ways to find $\mathbf{c}$ satisfying $\mathbf{Fc} = \mathbf{0}$.

- We propose to use the message passing graph decoder for its excellency in accuracy and superiority in decoding speed.
  This is to prevent a cheating attack in which a smart miner comes up with a new decoder algorithm of his own developed and outpaces the regular miners using the designated decoder. If this is allowed, a hidden advantage goes to the smart miner.

Fine Difficulty Level Adjustment using Exact Distance Spectra for Parity Check Codes

Heung-No Lee

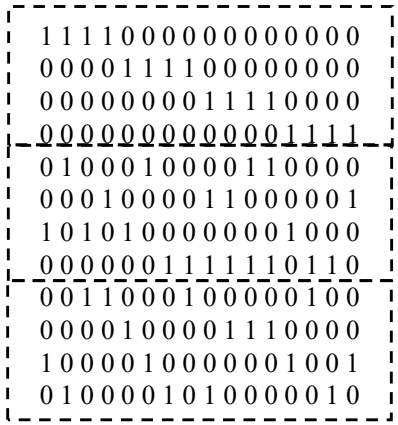
EECS
Gwangju Institute of Science and Technology (GIST)
Gwangju, South Korea
heungno@gist.ac.kr

***Abstract*—In this paper, we aim to provide fine difficulty level adjustment for crypto puzzles used in proof-of-work (PoW) systems. PoW provides secure solution to prevent the double spending problems in cryptocurrencies. We use the low-density parity check (LDPC) code based crypto puzzles. For this puzzles, the difficulty levels can be adjusted using the distance spectrum of the LDPC codes. A distance spectrum of a block code reveals how many number of codewords with a certain Hamming weight are there inside a given codebook. In this paper, we aim to calculate the distance spectrum of systematic and non-systematic parity check codes using a combinatorial method and compare the result with that using a traditionally known method.**

Keywords-component; Distance Sepctrum, LDPC code, LDGM code

In addition, we provide a very simple combinatorial routine to calculate the distance spectrum of short LDPC codes. In the past, researchers, such as Litsyn, Sheveley and Burshein, have focused on the asymptotic exponents of the distance spectrum[3][4]. The exponents from asymptotic bounds usually do not lead to accurate results. Instead, we propose a combinatorial way of calculating the distance spectrum as opposed to the traditional method, which relies on utility variable optimization. The traditional approach is an asymptotic bound on the distance spectrum. The new approach does not use any approximation.

We anticipate that this new distance spectrum analysis shall provide a clearer picture on the structure of this powerful class of codes and on the iterative receiver performance in comparison

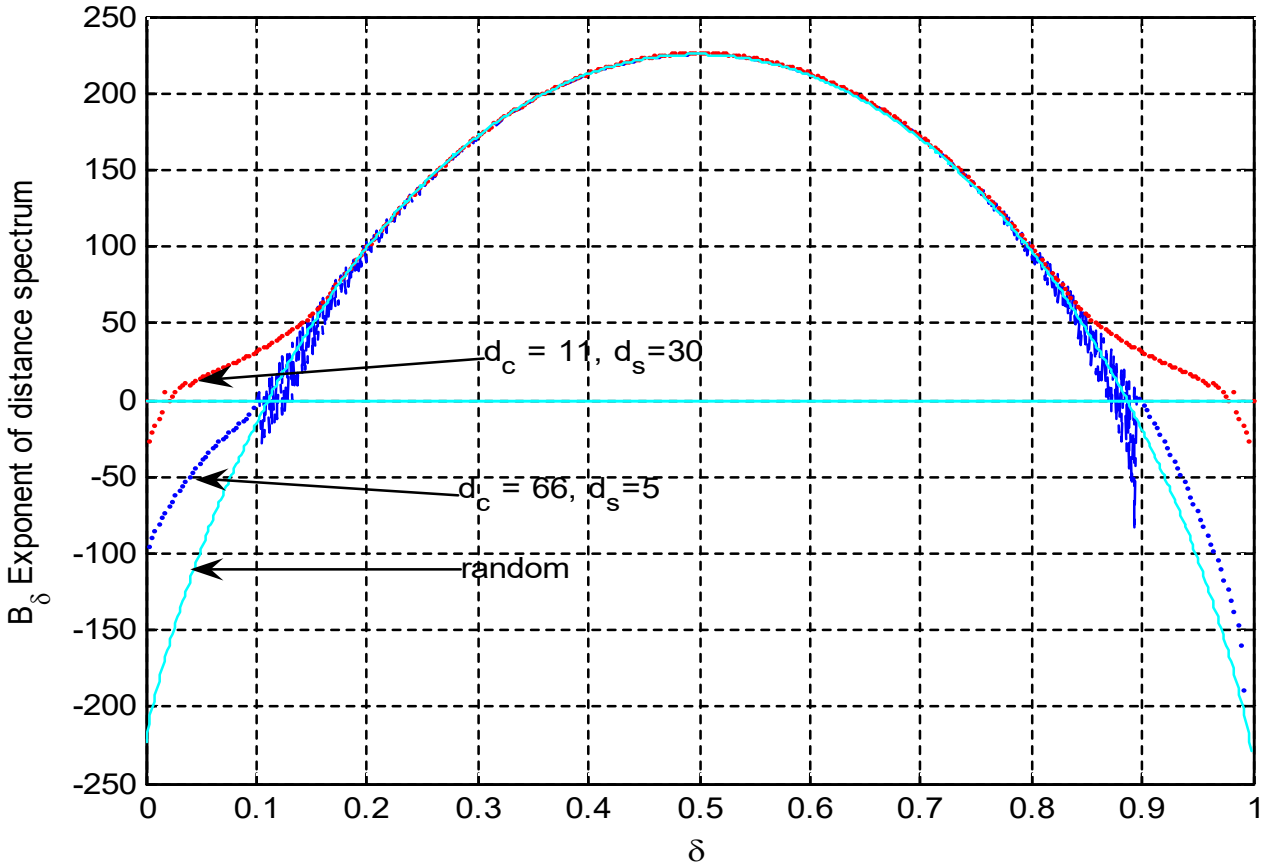


1st submatrix

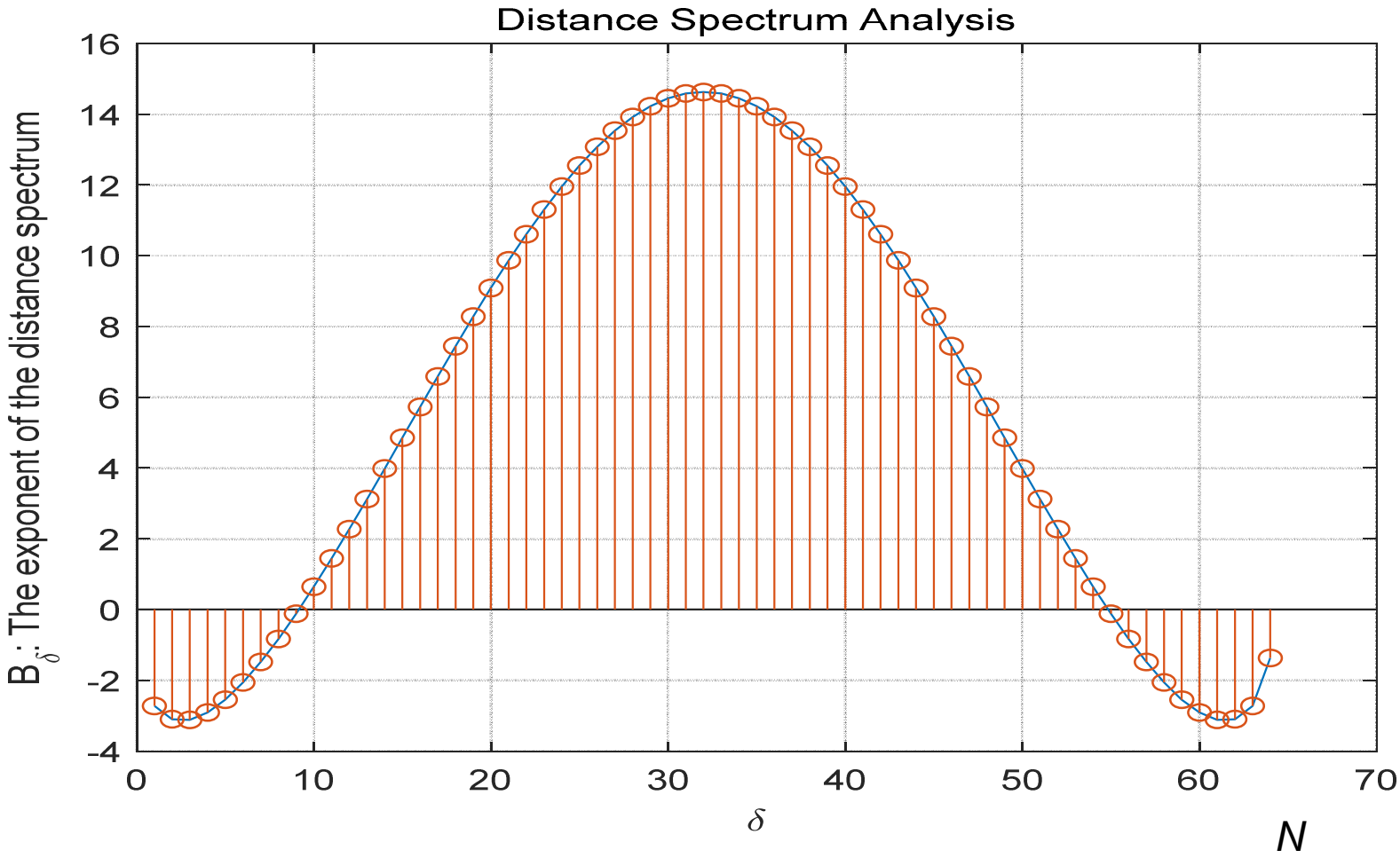
2nd submatrix

3rd submatrix

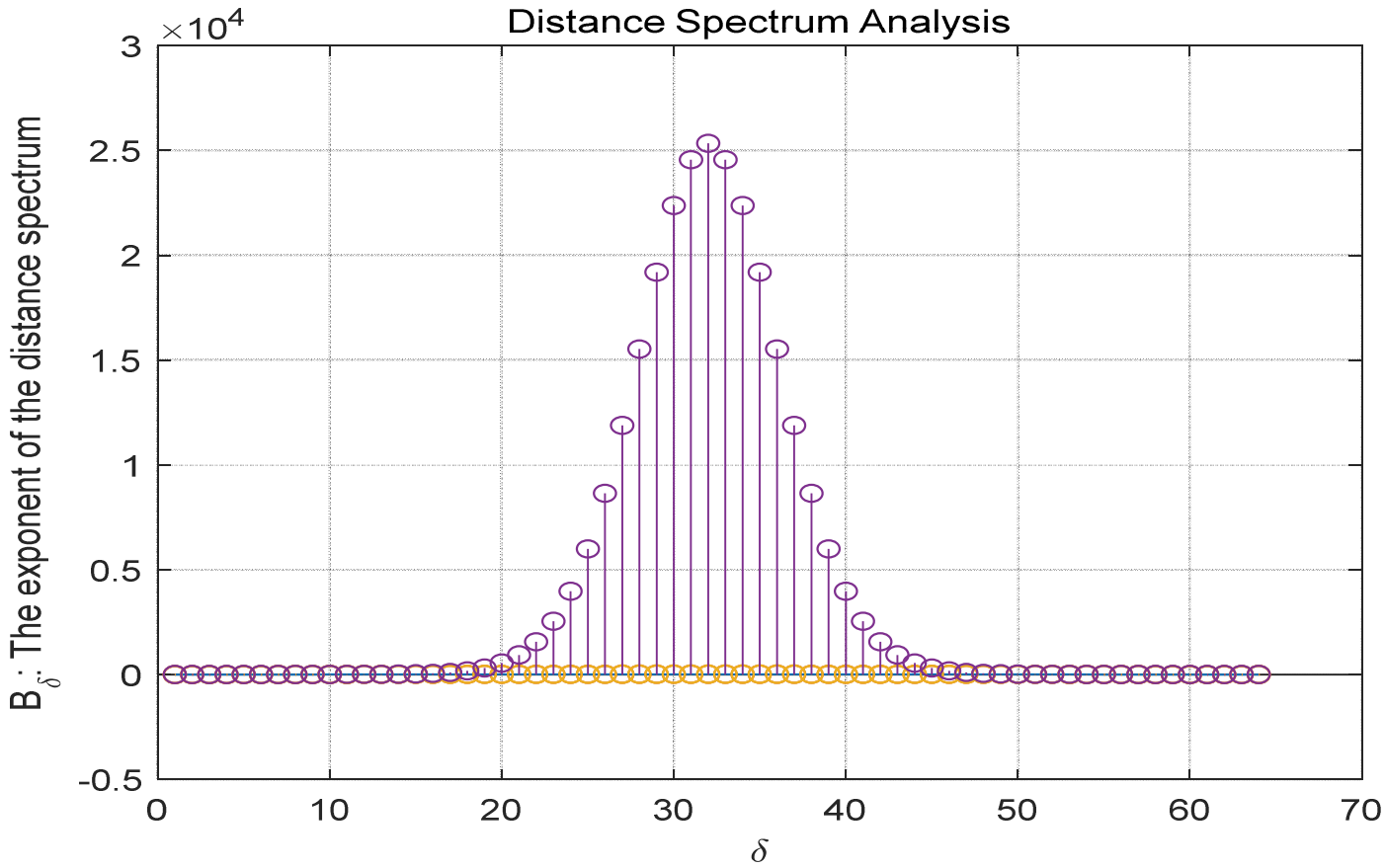
Distance Spectra of LDPC codes



Distance Spectrum of (N=64, 3, 4) Code



Distance Spectrum of (N=64, 3, 4) Code

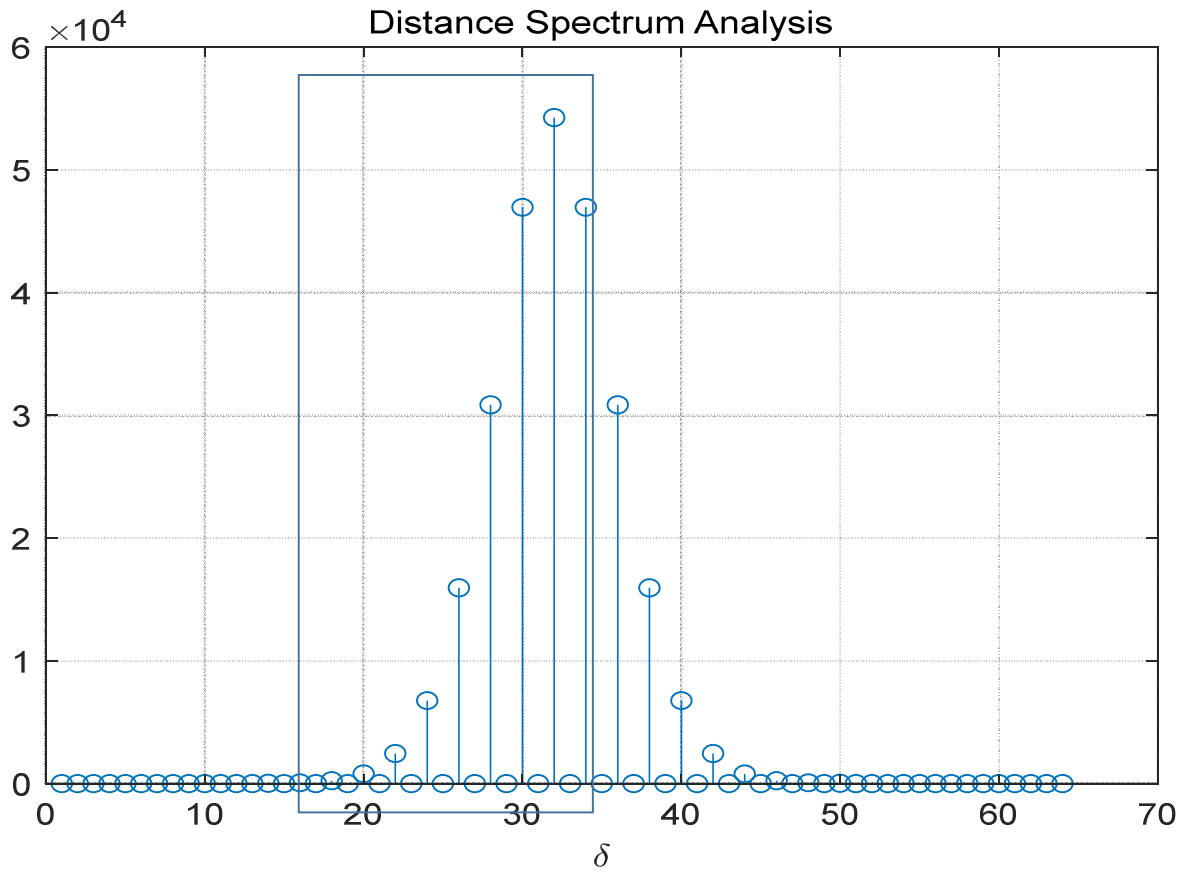


Decoder가 돌고,
c_hat을 뱉어냈다.

Condition Check:
1. codeword?
2. 난이도 만족?

이 코드에 기반한
작업증명의
최소 난이도와 최대
난이도는?

Distance Spectrum of (N=64, 3, 4) Code



Decoder가 돌고,
c_hat을 뱉어냈다.

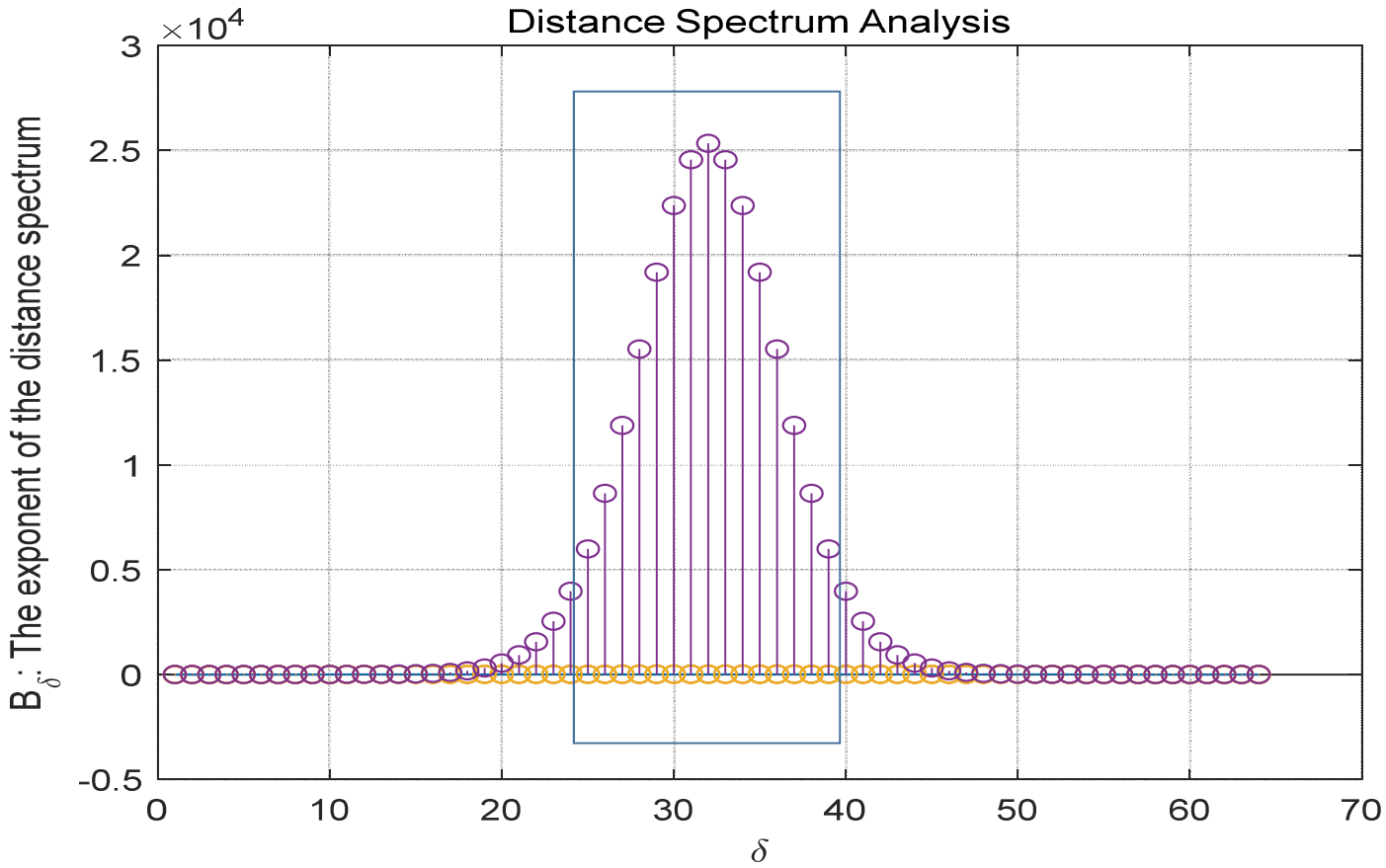
Condition Check:
1. codeword?
2. 난이도 만족?

이 코드에 기반한
작업증명의
최소 난이도와 최대
난이도는?

Spectrum within the Selected Box

- $h = 25, B_h = 5987$
- $h = 26, B_h = 8639$
- $h = 27, B_h = 11884$
- $h = 29, B_h = 15524$
- $h = 30, B_h = 19188$
- $h = 31, B_h = 22375$
- $h = 32, B_h = 24560$
- $h = 33, B_h = 25338$
- $h = 34, B_h = 24560$
- $h = 35, B_h = 22375$
- $h = 36, B_h = 19188$
- $h = 37, B_h = 15524$
- $h = 38, B_h = 11884$
- $h = 39, B_h = 8639$

Distance Spectrum of (N=64, 3, 4) Code



This is from using the Litsyn's result.

Spectrum within the Selected Box

- $h = 25, B_h = 5987$
- $h = 26, B_h = 8639$
- $h = 27, B_h = 11884$
- $h = 29, B_h = 15524$
- $h = 30, B_h = 19188$
- $h = 31, B_h = 22375$
- $h = 32, B_h = 24560$
- $h = 33, B_h = 25338$
- $h = 34, B_h = 24560$
- $h = 35, B_h = 22375$
- $h = 36, B_h = 19188$
- $h = 37, B_h = 15524$
- $h = 38, B_h = 11884$
- $h = 39, B_h = 8639$

There are 14 sets of codewords.

- 14C1 = 14 sets
- 14C2 = 91
- 14C3 = 364
- 14C4 = 1001
- …
- Suppose we cut at 4, then there are 1470 sets of difficulty.
- Order them in ascending order.

Distance Spectrum

- D(h): exponent on the number of codewords with Hamming weight h. Let h = 0, 1, 2, …, N
  - Too big a number this is. So it's better to use the exponent.
- d(h) = log(D(h));
- Ex) wc = 3, wr = 6, N 은 6*wc 의 배수, M = 3*wr
  - Rate R = (N-M)/N = d
- Ex2) D(h=0) = 1, D(1)=0, D(2)=0, D(3)= 0, D(4)= 0, D(5) = 4, … ➔ D = [1 0 0 0 0 4 7 7 4 0 0 0]

미세 난이도 조절 Algorithm

- 최소 난이도.
- 최고 난이도. Let dmin = min{h: D(h) neq 0} and choose the Condition Set C = {e: c_hat = DEC(e, H) whose $||c\_hat||_H$ = dmin}
- Algorithm의 목표: 최고 난이도와 최소 난이도 Range 안에서 난이도를 미세하게 조절하는 것.
  - 예를 들어, 난이도 두배 증가, 혹은 두 배 감소.
- EX) D = [1 0 0 0 0 4 7 7 4 0 0 0 0]
- 최소 난이도: 조건 만족 확률 = 22/22 = 1
- 최고 난이도: 조건 만족 확률 = 4/22 ~ 1/6

미세 난이도 조절 알고리즘의 예시

- 난이도 := 조건 만족 확률의 Inverse
- 가령 2가 가능?
- Yes. 가령 C = {c: $\|c\_hat\|_H$ = 5 or 6}. |C| = 4+7. 그러므로 확률은 ½. 난이도는 2.

미세 컨트롤 알고리즘 예시

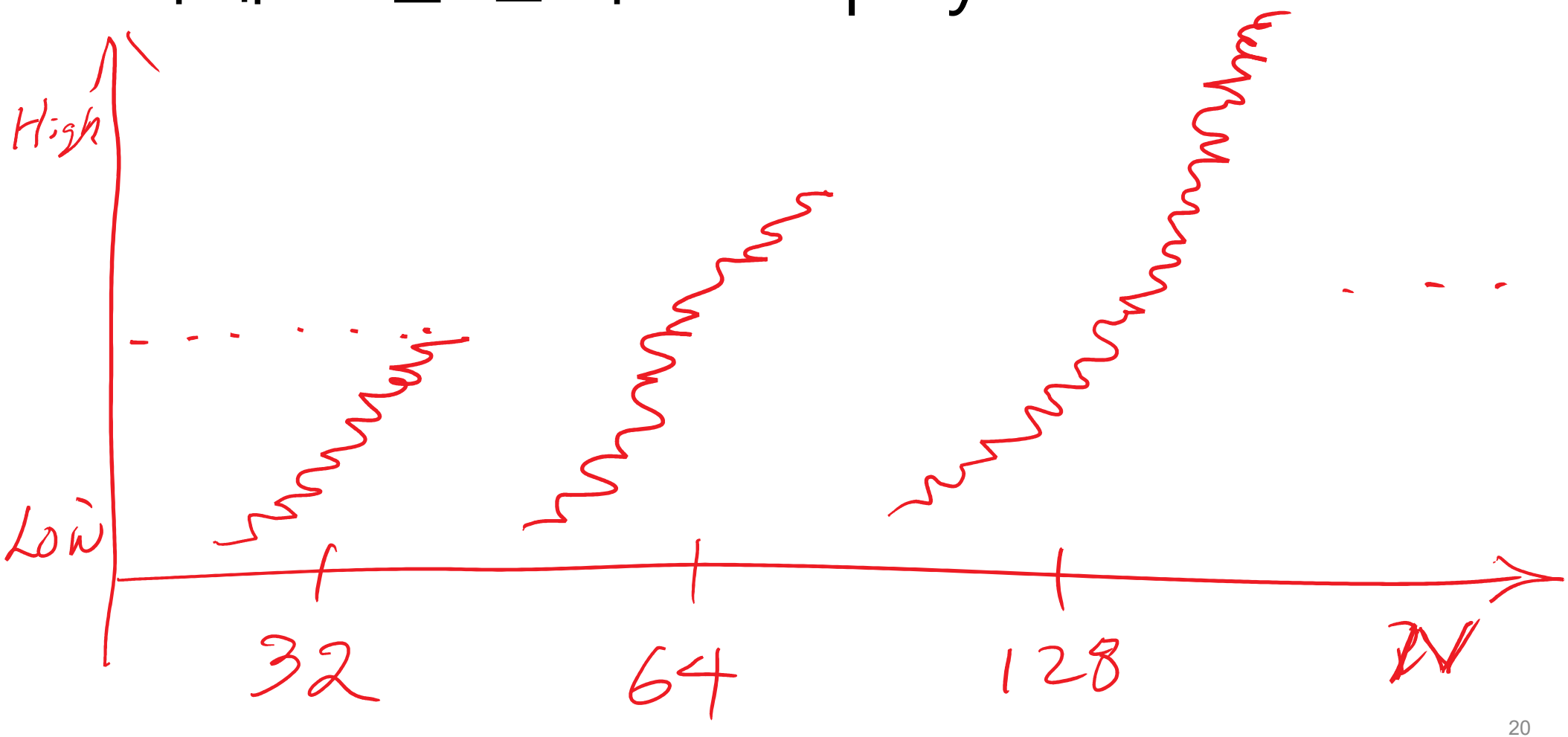
- 1. 몇 개의 미세 컨트롤 레벨이 존재하는가? 모든 가능한 조합을 고려한다.
- 2. 각 미세 컨트롤 레벨에 해당하는 난이도를 찾는다.
- 3. 난이도를 가장 작은 것부터 큰 것까지 Sorting한다. 각 난이도 별 조합도 같이 sorting된다. 각 난이도 별 조합이 씌여진 Table을 완성.
- 4. 최소 난이도 부터 최고 난이도 – epsilon 까지 조절 가능. 즉, 하나의 난이도에서 상향이나 하향 조절 가능.
- 5. 가능한 조합의 수는 nC2 + nC3 + nC4, … where n is the number of distance coefficients with non-zero Hamming weights. 우리 example에서 n = 4.

미세 컨트롤이 가능한 알고리즘
1. 몇 개의 미세 컨트롤 레벨이 존재하는가?

미세 컨트롤 알고리즘 Pseudo Code

- 1. 몇 개의 미세 컨트롤 레벨이 존재하는가? 모든 가능한 조합을 고려한다.
- 2. 각 미세 컨트롤 레벨에 해당하는 난이도를 찾는다.
- 3. 난이도를 가장 작은 것부터 큰 것까지 Sorting한다. 각 난이도 별 조합도 같이 sorting된다. 각 난이도 별 조합이 씌여진 Table을 완성.
- 4. 최소 난이도 부터 최고 난이도 – epsilon 까지 조절 가능. 즉, 하나의 난이도에서 상향이나 하향 조절 가능.
- 5. 가능한 조합의 수는 nC2 + nC3 + nC4, … where n is the number of distance coefficients with non-zero Hamming weights. 우리 example에서 n = 4.

미세 조절 난이도 Display



Summary

- 미세 난이도 조절 알고리즘 제시
- 컴퓨터 시뮬레이션 필요