Yonsei University

BEC 채널에서의 극 부호의 패리티 검사 행렬을 이용한 복호

한국통신학회 2012년도 하계종합학술발표회

라마다프라자 제주호텔, 제주

2012 / 6 / 21

박진수, 박기현, 송홍엽

연세대학교

Coding and Crypto Lab.

I. 서 론

- Arikan이 제안한 **극 부호**에 관한 연구가 활발히 진행되고 있음
- B-DMC(Binary-input Discrete Memoryless Channel)s 에서 채널 용량을 달성하는 것으로 알려짐
- 극 부호의 일반적 복호 방법인 SC(Successive Cancellation) 기법에 기반
- 극 부호의 SC 복호기에 더하여, 이에 해당하는 패리티 검사 행렬을 이용한 복호기를 추가적으로 덧붙임
- 이러한 패리티 검사 행렬을 LDPC(Low-Density Parity-Check) 부호의 복호 관점에서 최적화를 시도함
- 최종적으로, GF(2)상에서 극 부호의 복호 성능의 개선을 꾀함

II. 본 론(1)

- n회의 반복적인 채널 변형을 통해 얻어지는 극 부호의 생성 행렬
  - 길이 $2^n = N$
  - Kronecker 곱 연산기호 $\otimes$
  - n회 채널 변형으로 만들어지는 생성 행렬 G_N

$$F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad G_N = B_N F^{\otimes n}, \quad B_N = R_N (I_2 \otimes B_{N/2})$$

  - 이 때, $B_2 = I_2$ 이며 R_N은 reverse shuffle operator
- Notations
  - 어떤 row 벡터 a_i^j, $1 \le i \le j \le N$는 어떤 벡터 $(a_1, \cdots, a_N)$의 부 벡터 $(a_i, \cdots, a_j)$
  - 어떤 집합 $S \subset \{1, \cdots, N\}$ 에 대해 a_S 는 벡터 $(a_1, \cdots, a_N)$의 부 벡터로써, S 에 속한 인덱스의 element만을 모은 벡터
  - 생성행렬 G_N은 입력 벡터 u_1^N에 대하여 $x_1^N = u_1^N G_N$ 로 부호화
  - 정보 비트 인덱스 집합 $A \subset \{1,2,3, \cdots 2^N\}$
  - 고정 비트(frozen bits) 인덱스 집합 A^c
  - $G_N(A)$는 G_N 중 A에 속한 인덱스의 row만을 모아 만든 생성 행렬

II. 본 론(2)

- 생성 행렬의 변형
  - 정보 비트 수 $K = |A|$
  - 고정 비트의 값을 모두 0으로 두면 부호화 된 벡터는 $x_1^N = u_A G_N(A)$ 와 같음
  - $G_N(A)$ 의 크기는 $K \times N$
  - $G_N(A)$ 는 항상 full rank임을 쉽게 알 수 있음
  - $G_N(A)$ 에 상응하는 패리티 검사 행렬 $H_N(A)$
  - $G_N(A)$ 에 elementary row operation을 적용하여도 같은 벡터 공간을 갖음
  - 모든 $i \in \{1, \cdots, N\}$ 에 대해, i 번째 성분만 1이며 나머지 성분은 0인 길이 K 짜리 column 벡터 e_i
  - $G_N(A)$ 에 elementary row operation을 적용하여, 각 i 에 대하여 오직 하나의 e_i 를 갖도록 변형할 수 있으며, 그 때의 생성 행렬은 $G_N^e(A)$
  - $N \times N$ 크기의 permutation 행렬 Π 를 이용하여 다음과 같이 변형할 수 있음

$$G = \left[I_K \;\; P^T \right] = G_N^e(A)\Pi$$

  - 이 때, P 는 $(N-K) \times N$ 크기의 이진 행렬

II. 본 론(3)

- 극 부호의 패리티 검사 행렬
  - 구해진 G 를 이용하여 다음과 같은 패리티 검사 행렬을 구할 수 있음

$$H = [P \quad I_{N-K}]$$

  - $G_N^e(A)$ 에 맞게 역 permutation을 해주면,

$$H_N^e(A) = H\Pi^{-1}$$

  - 이며, $GH^T = 0$ (이 때, 0은 적당한 크기의 영행렬) 이므로

$$G_N^e(A)\,H_N^e(A)^T = 0$$

  - 따라서, $H_N^e(A)$는 $G_N^e(A)$의 패리티 검사 행렬이 될 수 있음
  - $G_N^e(A)$는 $G_N(A)$와 같은 벡터 공간을 가지므로 $H_N^e(A)$ 또한 $G_N(A)$의 패리티 검사 행렬

II. 본 론(4)

- 극 부호의 패리티 검사 행렬과 LDPC부호
  - $H_N^e(A)$를 보통의 LDPC 부호의 패리티 검사 행렬 관점에서 graph를 그릴 수 있음
  - 이 graph를 이용하여 BP(Belief Propagation) 복호를 수행할 수 있음
  - 하지만 $H_N^e(A)$는 LDPC 부호의 입장에서 전혀 최적화 되지 않았음

- Notations
  - $\lambda^o = (\lambda_1^o, \cdots, \lambda_{N-K}^o)$ 는 목표로 하는 최적화된 디그리별 변수 노드의 개수를 벡터로 적은 것
  - λ_i^o 는 최적화된 변수 노드들 중, i 의 디그리를 갖는 변수 노드의 개수
  - $\lambda = (\lambda_1, \cdots, \lambda_{N-K})$ 는 현재의 패리티 검사 행렬상의 각 디그리 별 변수 노드의 개수를 벡터로 적은 것
  - $d(\lambda^o, \lambda)$는 λ^o와 λ각각의 평균과 분산을 (m^o, ν^o)와 (m, ν)의 형태로 하여 2차원 공간상의 거리를 출력하는 함수
  - $d_{H_{now}}(i, j, \lambda^o)$는 H_{now}행렬의 i번째 row 벡터를 j번째 row 벡터에 더한 후의 λ를 $d(\lambda^o, \lambda)$에 넣은 결과값

II. 본 론(5)

입력 : λ^o, $H_N^e(A)$

$d_{now} = d(\lambda^o, \lambda)$; $H_{now} = H_N^e(A)$; $sw = true$;

while($d_{now} \neq 0$ & $sw \neq false$){

find a pair (i,j) such that $d_{now} > d_{H_{now}}(i,j,\lambda^o)$ and $i \neq j$

if(an (i,j) pair exists){

$d_{now} = d_{H_{now}}(i,j,\lambda^o)$

add ith row to jth row for H_{now}

}

else

$sw = false$

}

출력 : H_{now}

알고리즘 1. $H_N^e(A)$의 최적화 알고리즘

II. 본 론(6)

- 최적화 알고리즘의 출력
  - 위 알고리즘으로 구해진 H_{now}는 $H_N^e(A)$와 동일한 벡터 공간을 갖음
  - 따라서 H_{now} 또한 $G_N(A)$의 패리티 검사 행렬
- 극 부호의 복호기와 LDPC 복호기의 연접
  - H_{now} 에 해당하는 LDPC 부호의 BP 복호기를 극 부호의 SC 복호기 앞단에 연결
  - SC 복호기의 입장에서 더 나은 품질의 입력값을 얻기를 기대
  - 특히 BEC 에서는 LDPC 복호기가 erasure를 복호하지 못할 수는 있으나, 잘못 복호할 경우는 없기 때문에 SC 복호기 입장에서는 더 나은 채널 출력을 받는 것처럼 볼 수 있음

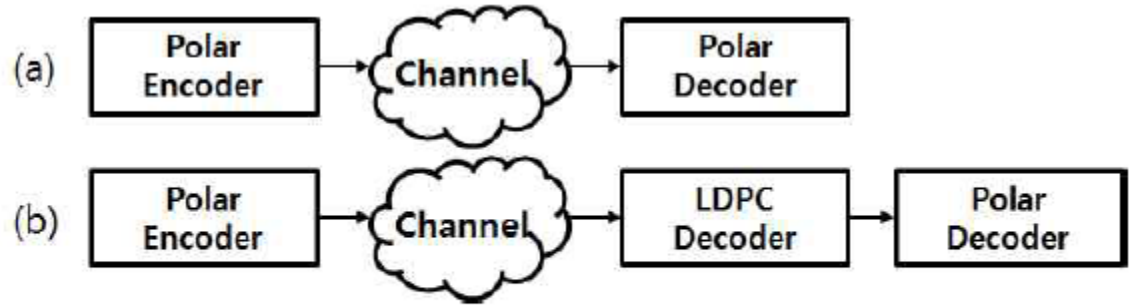

그림 1. (a) 기존 극 부호의 부, 복호화 과정, (b) 제안하는 극 부호의 부, 복호화 과정.

III. 결 론(1)

- 컴퓨터 시뮬레이션 파라미터
  - $n = 8, N = 256, K = 128$, 부호율 $1/2$
  - 고정 비트 인덱스 집합 A^c는 BEC상에서 Arikan이 제안한 방법으로 구함
  - 목표 최적 변수노드 수 벡터 $\lambda^o = (0, 0, N, 0, \cdots, 0)$
    - 즉, 모든 변수노드의 디그리가 3인 패리티 검사 행렬이 목표
  - 이에 대한 최적화 알고리즘의 결과로 $d_{now} = 21.8$인 H_{now}를 얻음
  - LDPC 복호기의 최대 반복 복호 횟수는 30회로 제한

III. 결 론(2)

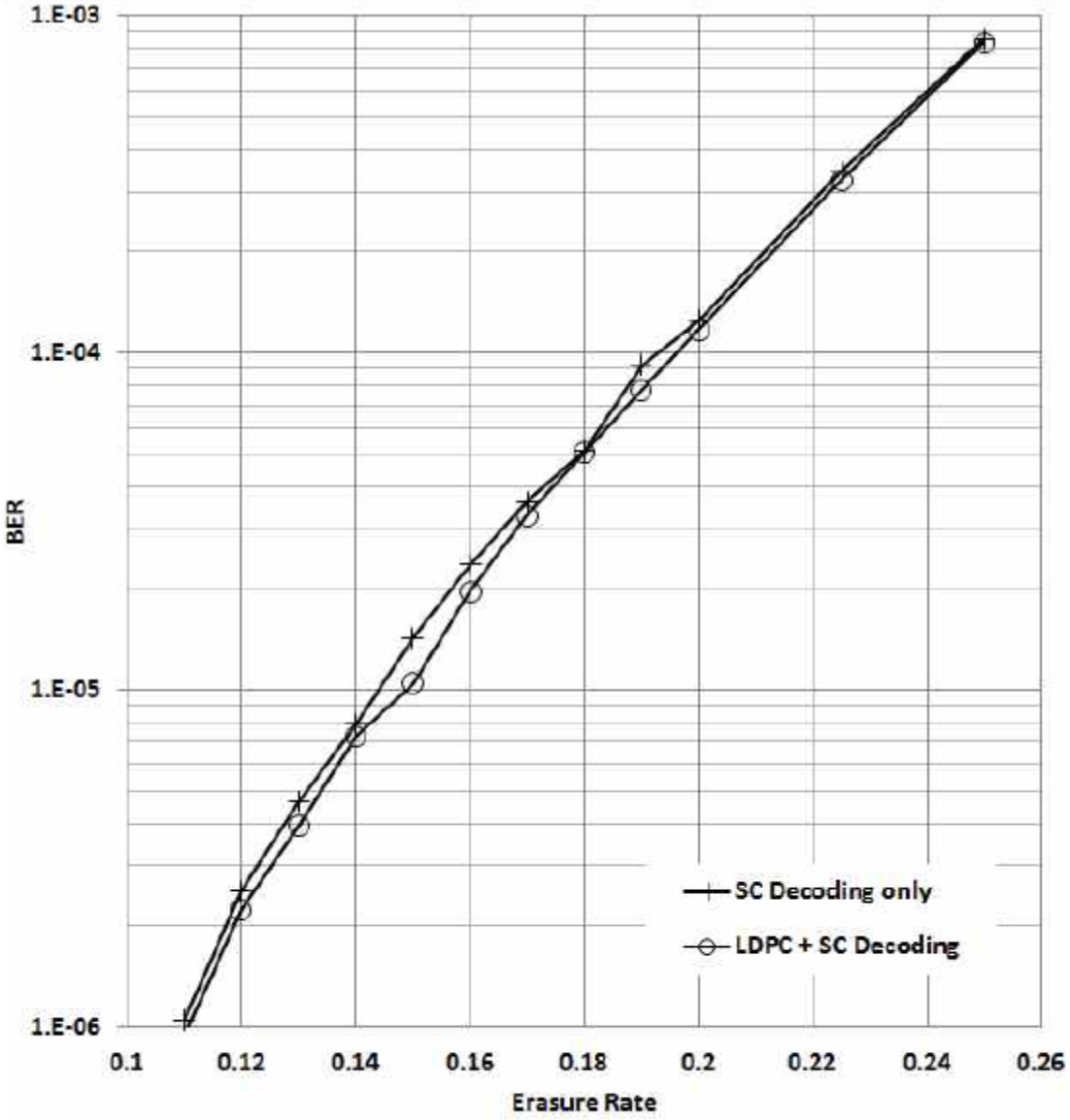

그림 2. 시뮬레이션 결과

III. 결　　론(3)

- 시뮬레이션 결과
  - 극 부호의 SC 복호만을 수행했을 때 보다 LDPC 복호기를 함께 사용했을 때 약간의 성능이 개선됨
  - LDPC 복호기가 사용한 계산 복잡도에 비해 그리 큰 이득이 아니라고 볼 수도 있음
  - 하지만, 다른 부호를 연접하여 부호율을 낮추지 않음
  - 또한 기존 SC 복호기보다 더 잘 복호할 수 있는 방법이 있다는 것을 보임에 의의가 있음
  - 만약 부호의 길이가 무한대로 길어지면,
    - Arikan의 증명에 따라 극 부호의 오류율은 SC 복호 만으로도 0으로 수렴함
    - 이 때에는 제안하는 복호 방법을 구지 사용할 필요는 없음
  - 하지만 실용적인 길이의 짧은 극 부호를 고려한다면, 이러한 접근도 고려할 가치가 있음