논문 99-24-7A-18 한국통신학회논문지 '99-7 Vol.24 No.7A

연쇄 컨볼루션 부호의 가중치 열거함수 계산 알고리듬

정회원 강 성 진*, 권 성 락**, 이 영 조**, 강 창 언***

An Algorithm for Computing the Weight Enumerating Function of Concatenated Convolutional Codes

Sung-Jin Kang*, Sung-Lark Kwon**, Young-Jo Lee**, Chang-Eon Kang*** *Regular Members*

요 약

병렬 연쇄 컨볼루션 부호 및 직렬 연쇄 컨볼루션 부호의 ML 연판정 복호에 대한 비트오류확률의 상한치는 가중치 열거함수(Weight Enumerating Function; WEF)를 통해서 구할 수 있으며, 이 상한치는 반복 복호를 통해 얻을 수 있는 비트오류확률의 하한치가 된다. 본 논문에서는 스택 알고리듬과 양방향 탐색 알고리듬을 혼합한 새로운 오류사건 탐색 알고리듬을 제안하고, 얻어진 오류사건을 이용하여 WEF를 계산하는 알고리듬을 제안한다. 컴퓨터 시뮬레이션을 통해, 반복복호를 통해 얻을 수 있는 비트오류확률의 하한치가 됨을 확인하였다.

ABSTRACT

The union upper bounds to the bit error probability of maximum likelihood(ML) soft-decoding of parallel concatenated convolutioanl codes(PCCC) and serially concatenated convolutional codes(SCCC) can be evaluated through the weight enumerating function(WEF). This union upper bounds become the lower bounds of the BER achievable when iterative decoding is used. In this paper, to compute the WEF, an efficient error event search algorithm which is a combination of stack algorithm and bidirectional search algorithm is proposed. By computor simulation, it is shown that the union boounds obtained by using the proposed algorithm become the lower bounds to BER of concatenated convolutional codes with iterative decoding.

I. 서 론

채널 부호는 블록 부호와 컨볼루션부호로 나눌 수 있다. 채널 부호 연구가들은 각각의 부호에 대하여 독립적으로 좋은 부호(또는, 발생다항식)를 찾는데 주력해왔다. 그러나, Forney는 부호를 병렬 혹은 직렬로 연결(concatenation)시켜서 사용하면 매우 우수한 오류 정정 능력을 보인다는 사실을 제시했다[1]. 기존의 연쇄부호는 외부부호로 블록 부호를, 내부부호로 컨볼루션 부호를 사용하는 직렬 형태가 주를 이루며, 현재도 강력한 오류 정정 능력이 필요한 곳에서 사용되고 있다.

근래에 와서, Berrou등에 의해 제안된 Turbo 코드는 컨볼루션 부호를 병렬로 연결시킨 연쇄 부호의 일종으로서 반복적인 복호를 통해 Shannon 한계에 1dB 이내로 접근하는 우수한 성능을 가진다[2]. 직렬 연쇄 컨벌루션 부호(Serially Concatenated Convolutiona Codes, SCCC)는 PCCC가 소개된 이후에 Benedetto등에 의해 소개된 부호로서, 두 컨벌루션 부호를 인터리버를 사이에 두고 직렬 연결하므로써 구성할 수 있다. SCCC는 PCCC에 비해 인터리빙 이득이 좋으며, 신호대 잡음비가 증가함에 따라 오류 확률의 플로어(floor)현상이 나타나지 않는 특징을 가진다[4].

* 한국전자통신연구원 무선방송기술연구소 방송기술연구부
** LG정보통신 중앙연구소 이동통신연구단
*** 연세대학교 전자공학과
논문번호 : 98217-0512, 접수일자 : 1998년 5월 12일

컨볼루션 부호의 비트오류확률에 대한 상한치는 부호의 전달함수(transfer function)를 이용하거나, 가중치 스펙트럼(weight spectrum)을 이용하여 계산할 수 있다. 입력되는 정보비트 열이 무한대라고 가정하면, 부호의 전달함수는 가중치 스펙트럼에 대한 closed-form으로 생각할 수 있다. 그러나 연쇄 컨볼루션 부호는 블록 단위로 전송이 이루어지기 때문에 전달함수에 대한 closed-form이 존재하지 않아서, 가중치 스펙트럼을 통해서 비트오류확률을 계산해야 한다. 최근에 와서, PCCC와 SCCC의 비트오류확률을 계산하기 위한 연구가 이루어졌으며, ML 연판정 복호에 대한 비트오류확률의 상한치가 반복 복호를 통해 얻을 수 있는 비트오류확률의 하한치가 됨을 보였다[3].

PCCC는 입력-잉여 가중치 열거함수(Input-Redundancy Weight Enumerating Function; IRWEF)를 통해 비트오류확률을 계산할 수 있으며, SCCC는 입-출력 가중치 열거함수(Input-Output Weight Enumerating Function; IOWEF)를 통해 계산할 수 있다[3,4]. 주어진 PCCC 및 SCCC 부호기에 대한 IOWEF와 IRWEF를 얻기 위해서는 오류사건의 개수와 출력 비트 가중치외에 입력 비트의 가중치 뿐만아니라 오류사건의 길이도 알아야 한다. 또한, 단일 오류사건과 다중 오류사건에 대해서도 고려를 해주어야한다. 컨볼루션 부호에 대한 가중치 스펙트럼을 탐색하는 몇몇 알고리듬들이 제안되어 있긴 하지만, 이들은 단일 오류사건만을 탐색하며, 특정 가중치에서 동작을 한다[5,6].

본 논문에서는 길이에 따른 오류사건을 탐색하고 정렬한 후에, 이들의 조합으로써 컨볼루션 부호의 IOWEF(혹은 IRWEF)를 찾는 알고리듬을 제안한다. 또한, 길이에 따른 오류사건을 탐색하는 알고리듬을 제안한다. 제안된 알고리듬을 사용하여 구한 비트오류확률의 상한치는 실제 반복 복호를 통해 얻을 수 있는 비트오류확률의 하한치가 됨을 컴퓨터 시뮬레이션을 통해 확인한다.

본 논문의 구성은 다음과 같다. II 장에서는 PCCC와 SCCC에 기본 성질과 각 부호에 대한 ML 상한치를 구하는 방법을 간단히 살펴보고, III장에서 ML 상한치를 구하기 위한 가중치 스펙트럼을 구하는 방법을 제안한다. IV장에서는 제안된 알고리듬을 통해 얻은 비트오류확률과 시뮬레이션을 통해 얻은 오류 성능을 비교하고, V 장에서 결론을 맺는다.

II. 연쇄 컨볼루션 부호의 비트오류확률

2.1 병렬 연쇄 컨볼루션 부호(PCCC)

병렬연결 컨볼루션 부호는 순환 체계적인 컨벌루션 부호(Recursive Systematic Convolutional Code)를 기본 부호로 사용하며, 그림 1과 같이 인터리버를 사이에 두고 기본 부호를 병렬 연결 시킴으로써 구성할 수 있다. PCCC는 비트오류확률 관점에서 최적의 복호 방법인 Maximum Aposteriori Probability(MAP) 알고리듬을 반복적으로 수행하여 복호할 수 있다. PCCC에서 사용되는 인터리버는 컨벌루션 부호의 자유 해밍거리가 최소가 되게 하는 입력 정보의 최악의 패턴을 제거하여 전체 부호의 자유 해밍 거리를 최대화하는데 기여한다. 따라서, 인터리버는 랜덤할수록 좋으며, 그 크기가 클수록 더 나은 성능을 보인다.

PCCC에 대한 IRWEF 및 조건부 가중치 열거함수(conditional weight enumerating function; CWEF)는 각각 다음과 같이 정의된다[3].

$$A^C(W,Z)=\sum_{w,j}A_{w,j}W^wZ^j \tag{1}$$

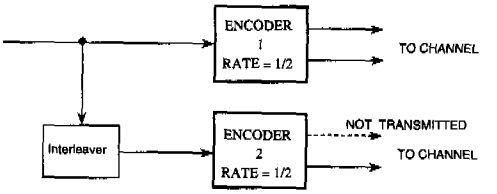


그림 1. PCCC 부호기

$$A_w^C(Z)=\sum_j A_{w,j}Z^j=\frac{1}{w!}\left.\frac{\partial^w A^C(W,Z)}{\partial W^w}\right|_{W=0} \tag{2}$$

여기에서 $A_{w,j}$는 입력 정보비트의 해밍 가중치가 w일 때 패리티비트의 가중치가 j인 코드워드의 수를 나타낸다. 따라서, 이러한 코드워드의 전체 해밍 가중치는 $w+j$이 된다. 식(1)의 IRWEF와 식(2)의 CWEF PCCC의 ML 연판정에 대한 비트 오류확률의 유니온 바운드(union bound)를 구하는 데 사용되어질 수 있다. 즉,

$$\begin{aligned}P_b &\le \left.\frac{W}{k}\frac{\partial A^C(W,Z)}{\partial W}\right|_{W=Z=e^{-R_cE_b/N_0}}\\ &=\left.\sum_{w=1}^{k}\frac{w}{k}W^wA_w^C(Z)\right|_{W=Z=e^{-R_cE_b/N_0}}\\ &=\left.\sum_m D_mH^m\right|_{H=e^{-R_cE_b/N_0}}\end{aligned} \tag{3}$$

여기에서, D_m은 식(4)와 같고, R_c는 부호율이다.

$$D_m = \sum_{j+w=m} \frac{w}{k} A_{w,j} \tag{4}$$

식 (3)으로부터 부호의 비트 오류 확률에 대한 좀 더 엄격한 바운드(bound)는 식(5)와 같이 얻을 수 있고, 유한개 항을 고려하면 식(6)과 같은 상한치의 근사값을 얻을 수 있다.

$$P_b \le \frac{W}{2k}\,\mathrm{erfc}\left(\sqrt{\frac{d_{\min} R_c E_b}{N_0}}\right) \cdot e^{\frac{d_{\min} R_c E_b}{N_0}} \left.\frac{\partial A^C(W,Z)}{\partial W}\right|_{W=Z=e^{-R_c E_b/N_0}} \tag{5}$$

$$P_b \approx \frac{1}{2}\sum_m D_m\,\mathrm{erfc}\left(\sqrt{m\frac{R_c E_b}{N_0}}\right) \tag{6}$$

PCCC에 대한 전체 IRWEF를 계산하기 위해서는 먼저 초월 트렐리스(hyper trellis)를 구성한 다음, 초월 함수의 각 가지에 해당하는 전달함수를 구해야하며 다음과 같이 된다.

$$A^C_{ij,ml}(w,Z) = \frac{A^{C_1}_{im}(w,Z)\cdot A^{C_2}_{jl}(w,Z)}{\binom{N}{w}} \tag{7}$$

여기에서, N은 인터리버의 길이이고, $A^{C_n}_{sn}(w,Z)$는 부호기의 상태 s에서 시작해서 N번째 후에 상태 n에서 끝나는 모든 경로를 포현하는 CWEF이다. $A^C_{ij,ml}(W,Z)$는 PCCC의 두 부호기가 초기 상태 s_{1i}, s_{2j}에서 시작해서 N번째 후에 s_{1m}, s_{2l}상태에 도달하는 모든 경로를 표현한다. 참고문헌 [3]에서는 0상태에서 0상태로 가는 $A^C_{00,00}(W,Z)$로 전체 초월 트렐리스의 전달함수를 대치하여 비트 오류 확률의 상한치에 대한 근사치를 구하는 방법을 제시하고, 인터리버의 크기가 기본 부호의 구속장보다 매우 크기만하면 근사치와 정확한 계산치가 유사하게 나타남을 보였다.

2.2 직렬 연쇄 컨볼루션 부호(SCCC)

SCCC는 그림 2와 같이 두 개의 컨벌루션 부호를 인터리버를 사이에 두고 직렬로 연결시켜서 구성한다. 내부 부호(inner code) C_i의 부호율이 $R^i_c = p/n$이고, 외부 부호(outer code) C_o의 부호율이 $R^0_c = k/p$일 때, SCCC의 전체 부호율 $R_c = R^i_c \times R^o_c = k/n$이 된다. SCCC에서 내부 부호는 항상 인터리버 이득을 얻을 수 있기위하여 RSC부호를 사용하고, 외부 부호는 NSC 부호를 사용할 때 가장 우수한 성능을 가진다고 알려져 있다. 이때 외부 부호는 d^o_f가 크고(가능하면 홀수인), $w_{M,f}$와 N^o_f는 작고 $d^i_{f,eff}$가 큰 부호를 선택한다. 여기에서 $w_{M,f}$는 가중치가 d^o_f인 출력 코드워드를 생성시키는 입력 가중치의 최대값을 나타내며, N^o_f는 그러한 출력 코드워드의 개수이다. $d^i_{f,eff}$는 입력 가중치 $w=2$일 때의 자유 해밍거리이며, 실효 자유 거리(effective free distance)라고 한다[4].

SCCC에 대한 IOWEF는 다음과 같이 정의한다.

$$A_C(W,H) = \sum_{w,h} A^C_{w,h} W^w H^h \tag{8}$$

여기에서 $A^C_{w,h}$는 가중치가 w인 입력 정보워드에 의해 발생되는 코드워드의 가중치가 h인 모든 코드워드의 개수이다. PCCC에서와 유사하게 여기에서도, 입력 정보 워드의 가중치 w에 대한 조건부 가중치 열거 함수(Conditional Weight Enumerating Function; CWEF) $A^C(w,H)$를 정의할 수 있으며 식(9)와 같은 관계식을 가진다. CWEF를 구하면 식(10)과 같이 비트 오류 확률에 대한 바운드를 구할 수 있다[4].

$$A^C(w,H) = \frac{1}{w!}\left.\frac{\partial^w A^C(W,H)}{\partial W^w}\right|_{W=0} \tag{9}$$

$$P_b(e) \le \sum_{w=1}^{k} \frac{w}{k} A^C(w,H)\Big|_{H=e^{-R_c E_b/N_0}} \tag{10}$$

내부 부호와 외부부호의 CWEF를 구하면 식(11)과 같이 SCCC에 대한 IOWEF를 얻을 수 있고, PCCC의 경우와 마찬가지로 $A^C_{00,00}(W,H)$로 SCCC에 대한 초월 트렐리스의 전달함수를 근사화시킬 수 있다.

$$A^C_{ij,ml}(W,H) = \sum_{l=0}^{N} \frac{A^{C_o}_{im}(W,l)\cdot A^{C_i}_{jl}(l,H)}{\binom{N}{l}} \tag{11}$$

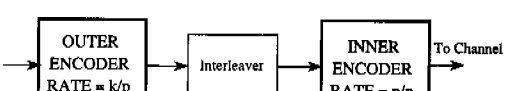


그림 2. 직렬 연쇄 컨벌루션 부호

Ⅲ. 제안된 가중치 열거함수 계산 알고리듬

트렐리스 길이가 N인 컨벌루션 부호기의 CWEF를 찾기 위한 전달함수 $T^C(W,Z,L,\Omega)$는 다음과 같이 정의된다.

$$T^C(W,Z,L,\Omega)=\sum_{i,j,m,n} T_{i,j,m,n} W^i Z^j L^m \Omega^n \qquad (12)$$

$T^C(W,Z,L,\Omega)$는 트렐리스의 처음 단계의 0상태에서 출발하여 N번째 단계 혹은 그 이전에 0상태로 돌아오는 모든 가능한 경로를 열거하는 전달함수이다. 이 전달함수에 표현된 경로는 N번째 단계 이전 단계에서 0상태로 재결합(remerge)되면 바로 다음 단계에서 0상태를 떠나거나 아니면, 끝까지 0상태를 유지하여야 한다. 이러한 경로는 트렐리스 상에 생길 수 있는 여러개의 오류 사건을 연달아 모아 놓은 형태이다. $T_{i,j,m,n}$는 입력 정보열의 가중치가 i, 잉여 비트의 가중치가 j, 전체 길이가 m, 0 상태로 n번 재결합(즉, n개의 오류 사건을 연달아 모아 놓은 형태)되는 경로의 개수이다. 따라서, $T_{i,j,m,n}$는 컨벌루션 부호의 출력열 중에서 입력 정보비트의 가중치와 잉여비트의 가중치가 같은 경로의 개수를 의미하며, 이것은 블록 부호의 코드워드 중 입력 비트와 잉여 비트의 가중치가 같은 코드워드의 개수와 유사한 개념이다. 컨벌루션 부호에서 (i,j,m,n)쌍은 하나의 블록 부호의 코드워드에 해당하는 등가 코드워드로 볼 수 있다. 하나의 (i,j,m,n) 등가 코드워드는 길이가 N인 트렐리스 상에서 식(13)에 나타나있는 개수 만큼의 실제 경로로 부터 나올 수 있다[3]. 즉,

$$K[m,n]={}_{N-m+n}C_n \equiv \binom{N-m+n}{n} \qquad (13)$$

따라서, 컨벌루션 부호의 CWEF는 식(14)와 같이 쓸 수 있다.

$$A_{(N)}^C(w,Z)=\sum_j A_{w,j} Z^j \qquad (14)$$

여기에서,

$$A_{w,j}=\sum_{m,n} K[m,n]\, T_{w,j,m,n} \qquad (15)$$

전달함수 $T^C(W,Z,L,\Omega)$를 살펴보면, 기존의 컨볼루션 부호 및 트렐리스 부호에 대한 가중치 스펙트럼 탐색 알고리듬을 적용할 수 없다는 사실을 알 수 있다. 즉, 기존의 가중치 스펙트럼 탐색 알고리듬은 오류 사건의 길이는 고려하지 않고, 입력 가중치와 출력 가중치가 다른 오류 사건만을 구별하고 이 들의 개수를 세기때문에 $T^C(W,Z,L,\Omega)$를 구하기 위해서는 변형이 되어야만 한다. 본 논문에서는 길이에 따른 오류 사건들을 스택에 쌓아가면서 트렐리스를 양방향으로 탐색하는 알고리듬을 제안한다. 이 과정을 거쳐 얻어진 오류사건들의 가능한 조합을 계산하면, 전달함수 $T^C(W,Z,L,\Omega)$를 구할 수 있다.

PCCC에 대한 가중치 열거 함수 계산 알고리듬

단계 P1) 트렐리스를 초기화하고 구하고자 하는 가중치 열거 함수의 최대 가중치 $D_{\max}$ 및 트렐리스 길이 N을 설정한다.

단계 P2) $D_{\max}$ 보다 작은 오류사건 탐색. 이 때, 오류 사건은 입력 정보비트의 가중치, 출력 잉여비트의 가중치 및 오류 사건의 길이가 다르면 다른 오류사건으로 구별하고 기억해야한다. 각각의 오류 사건은 $\{L_l, W_l, Z_l, N_l\}$로 구별된다.
여기에서, L_l은 오류사건의 길이, W_l은 입력 정보 비트의 가중치, Z_l는 출력 잉여비트의 가중치, N_l는 오류 사건의 개수를 나타낸다. 따라서, 이 오류사건에 의해 발생되는 출력 코드워드의 가중치는 $D_l = W_l + Z_l$이 된다.

단계 P3) 단계 P2에서 얻어진 오류사건에 대하여 다음 조건을 만족하는 모든 조합을 계산하여 $T_{i,j,m,n}$을 계산한다.

$$\sum^{n} D_l < D_{\max} \quad \text{and} \quad \sum^{n} L_l < N$$

이 때, $T_{i,j,m,n}$는 다음과 같이 계산된다.

$$T_{i,j,m,n} = \prod^{n} N_l, \quad \text{where} \begin{cases} i=\sum^{n} W_l\,,\ j=\sum^{n} Z_l\,, \\ m=\sum^{n} L_l\,,\ n=D_l \text{의 개수} \end{cases}$$

단계 P4) 식(14)과 (15)를 이용하여 기본 부호(constituent code)의 CWEF를 구하고, 식(7), (4)을 통해 전체 PCCC에 대한 가중치 열거 함수를 구한다.

SCCC에 대한 가중치 열거 함수 계산 알고리듬

단계 S1) 내부 부호의 트렐리스를 초기화하고 구하고자 하는 가중치 열거 함수의 최대 가중치 $H_{\max}$ 및 트렐리스 길이 N을 설정한다.

단계 S2) 내부 부호에 대하여 $H_{\max}$보다 작은 오류 사건 탐색. 이 때, 오류 사건은 입력 정보비트의 가중치, 출력 코드워드의 가중치, 오류 사건의 길이가 다르면 다른 오류사건으로 구별하고 기억해야한다. 오류 사건은 $\{L_l, W_l, H_l, N_l\}$로 구별된다. 여기에서, L_l은 오류사건의 길이, W_l은 입력 정보 비트의 가중치, H_l는 출력 코드워드의 가중치, N_l는 오류 사건의 개수를 나타낸다.

단계 S3) 단계 S2에서 얻어진 오류사건에 대하여 다음 조건을 만족하는 모든 조합을 계산하여 $T_{i,j,m,n}$을 계산한다. 이때, 입력 가중치의 최대값 $I_{\max}$ 를 저장한다.

$$\sum H_l < H_{\max} \quad \text{and} \quad \sum L_l < N$$

이 때, $T_{i,j,m,n}$는 다음과 같이 계산된다.

$$T_{i,j,m,n} = \prod N_l, \quad \text{where} \begin{cases} i = \sum W_l, \; j = \sum H_l \\ m = \sum L_l, \; n = H_l \text{의 개수} \end{cases}$$

단계 S4) 식(14)와 (15)를 이용하여 내부 부호의 CWEF를 구한다.

단계 S5) $H_{\max}$ 값을 $I_{\max}$ 값으로 치환한 후, 외부 부호에 대하여 단계 S2부터 단계 S4를 반복한다.

단계 S6) 식(11)을 통해 SCCC에 대한 가중치 열거 함수를 구한다.

오류 사건 탐색 알고리듬

위의 알고리듬에서 단계 P2 및 S2에 사용되는 오류사건 탐색 알고리듬은 다음과 같다. 단계 E1에서의 확장은 이전 경로의 가중치에 가지 가중치(branch weight)를 더하는 과정을 의미하며, 첨자 f와 b는 순방향과 역방향을 의미한다.

단계 E1) 순방향의 0 상태를 제외한 모든 상태에 대하여 트렐리스를 확장한다. 만약 이전 상태에 경로가 하나도 존재하지 않으면, 그 상태에 대해서는 확장을 하지않는다. 상태 s_i로 연결되는 모든 가지에서 정보비트 가중치와 출력비트 가중치가 같은 경로는 같은 경로로 판단하고 개수를 증가시킨후 하나는 제거한다. 이때, 확장된 경로의 가중치가 $D_{\max} - D_{b,min}$인 경로는 저장하지 않고 삭제하므로써 불필요한 계산과 메모리를 줄일 수 있다. $D_{b,min}$는 역방향 확장된 경로의 가중치중에서 가장 작은 값이다. 순방향 확장된 경로의 가중치중에서 가장 작은 값 $D_{f,min}$을 저장하고, L_f를 증가시킨다.

단계 E2) 0 상태를 제외한 모든 상태에서, 역방향 확장된 상태와 결합(merge)하는 가를 확인한다(구속장의 2배 만큼 순방향과 역방향 확장을 하고 나면, 0 상태를 제외한 모든 상태에서 결합이 일어난다. 그림 3(c) 참조).

단계 E3) 결합한 상태가 있으면, $L_l = L_f + L_b$의 길이를 갖는 오류 사건이 존재하는 것이며, 순방향 확장된 경로와 역방향 확장된 경로의 모든 조합으로부터 오류사건을 계산하고 저장한다. 이때, 오류사건의 입력 가중치는 $W_l = W_f + W_b$이고, 출력 가중치는 $Z_l = Z_f + Z_b$, 오류사건의 개수는 $N_l = N_f \times N_b$가 된다.

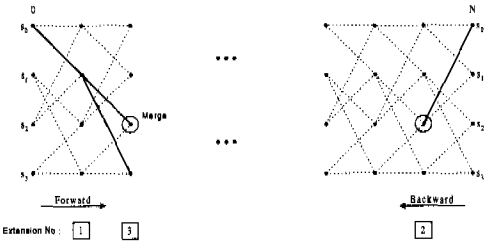
단계 E4) 결합이 일어난 모든 상태에 대하여 단계 E3을 수행한다. 만일, 중복되는 오류 사건(입력 가중치, 출력 가중치가 같은 오류사건)은 오류 사건의 개수를 증가시키고 나머지는 제거한다. 오류사건을 저장할 때, 출력 가중치에 대하여 오름차순으로 정렬을 한다.

단계 E5) 역방향에 대하여 단계 E1부터 단계 E4를 수행한다.

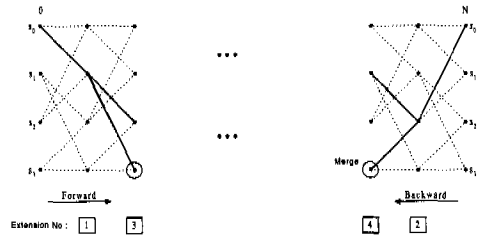
단계 E6) 순방향 확장과 역방향 확장을 계속해서 번갈아 가며 반복한다. 만약 $L_f + L_b$가 트렐리스 길이 N보다 크거나, 단계 E4에서 오류사건이 하나도 생기지 않으면 중단한다.

제안된 알고리듬은 동적 프로그래밍을 통해 쉽게 구현될 수 있으며, 순방향(forward) 확장과 역방향(backward) 확장을 통해 오류 사건을 쉽게 찾을 수 있다. 즉, 순방향 확장시에는 역방향 상태와 결합되

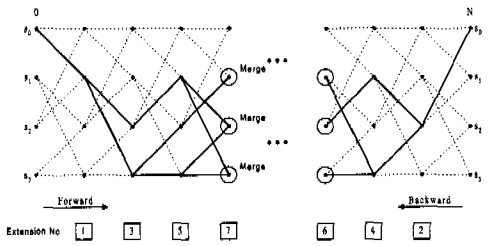
어야만 0상태로 갈 수 있고, 역방향 확장시에는 순방향 상태와 결합되어야만 0상태로 갈 수 있다.



(a) 3번째 순방향 확장



(b) 4번째 역방향 확장



(c) 7번째 순방향 확장

그림 3. 오류 사건 탐색 알고리듬 예

따라서 순방향과 역방향 확장시에 0상태로 가는 가지는 확장에서 제외된다. 단, 첫 번째 순방향 확장과 두 번째 역방향 확장은 0상태에서부터 시작한다. 그림 3은 제안된 알고리듬의 동작 예를 보여준다.

그림 3(a)는 3번째 순방향 확장을 나타내며, 순방향 확장된 상태중에서 s_2만 역방향 상태 s_2와 결합되고 있다. 그림 3(b)는 4번째 역방향 확장이며, 상태 s_3에서만 결합이 일어난다. 그림 3(c)는 7번째 순방향 확장을 나타내며, 0상태를 제외한 모든 상태에서 결합이 일어남을 볼 수 있다. 순방향과 역방향으로 번갈아 확장을 하면서 각 상태에서의 결합를 확인하면 오류 사건을 찾을 때 중복을 피할 수 있다.

단계 P3 와 단계 S3에서 계산량 줄이는 방법

단계 P3와 단계 S3는 앞서 설명한 오류 사건 탐색 알고리듬을 사용하여 얻어진 오류 사건들을 가지고 조건을 만족하는 모든 가능한 조합을 계산하는 단계이다.

이 때, 사건의 개수 n을 증가시키면서 가능한 모든 조합을 따지는 것이 효율적이다. 그러나, 오류 사건이 증가함에 따라, 모든 가능한 조합을 검사하는 것은 시간이 많이 소요된다. 그림 4는 n=2인 경우에 대한 오류사건 조합의 예를 보여준다. 오류 탐색 알고리듬에 의해 그림 4(a)의 E1~E5와 같이 오류 사건 5개를 찾았다고 가정하자. 여기에서 E1과 E2는 오류 사건의 길이가 5이고, E3과 E4는 6, E5는 7이다. 이들 오류사건 5개로서 가능한 조합은 그림 4(b)와 같이 $5^2=25$가지가 가능하다. 그림 4(b)의 조합중에서 E1+E2, E2+E1은 두 오류사건의 위치만 바뀐 형태이기 때문에, 식(12)에서 i,j,m,n이 같게 나타난다.

따라서, 이들 오류사건은 한번만 계산하고, 중복되는 경우의 수 만큼 곱해주면된다. 이렇게 중복되는 오류사건의 조합은 크게 두가지 형태, 즉, 같은 길이의 오류사건 조합에서 중복되는 경우와, 다른 길이의 오류 사건 조합에서 중복되는 경우로 나눌 수 있다. 예를 들면, E1+E2과 E2+E1는 같은 길이의 오류사건 조합에서 중복되는 경우이고, E1+E3과 E3+E1은 다른 길이의 오류사건 조합에서 중복되는 경우이다. 이러한 중복되는 경우를 제거하므로써 프로그램의 수행시간을 줄일 수 있다.

본 논문에서는 다음과 같은 과정을 통해 중복되는 경우를 피하면서 $T_{i,j,m,n}$를 계산한다. 먼저, 오류사건을 길이별로 묶어서 길이에 따른 오류사건 부집합을 만든 다음, 아래와 같은 조건에 의해 n개의 오류 사건 부집합을 선택한다.

$$\{S_l,\cdots,S_{l+n-1}\},\quad \text{Length}(S_l)\le\cdots\le\text{Length}(S_{l+n-1})$$

여기에서, S_l은 길이가 l인 오류사건의 집합이다. 그림 4의 예에서는 부집합은 S_1={E1,E2}, S_2={E3,E4} S_3={E5}이다.

이들 부집합으로부터 위의 조건을 만족하면서 부집합을 선택하는 경우는 그림 4(c)처럼 6가지가 생긴다. 이 때, 그림 4(c)의 ①,④,⑥은 같은 부집합이 선택된 경우이고, ②,③,⑤은 서로 다른 부집합이 선택된 경우이다.

이들 부집합에 따라, 중복되는 경우가 달라진다. 즉, ②,③,⑤의 경우는 부집합에서의 모든 경우에 대하여 2!개 중복이 나타나며, ①,④,⑥의 경우는 서로 다른 오류사건의 조합에 대하여 2!개 중복이 나타난다.

그림 4는 n=2인 경우에 대한 예이며, 일반적인 경우로 확장하면 Sfactor와 Efactor는 모두 같은 것이 들어 있는 순열의 수와 같게 된다.

즉, Sfactor는 부집합을 원소로 생각했을 때 같은 것이 들어 있는 순열의 수와 같으며, Efactor는 오류사건을 원소로 생각했을 때 같은 것이 들어 있는 순열의 수와 같다.

예를 들어, $\{S_1, S_2, S_3\}$인 경우에 대해서는 Sfactor= 3!/1!1!1! = 3이고, $\{S_1, S_1, S_2\}$인 경우는 Sfactor=3!/2!1!이 된다. 그림 4에서 각 조합에 Mfactor와 Sfactor를 곱하면 전체 조합을 계산한 결과를 얻게된다. 그림 4에서 조사할 오류사건의 조합이 총 25가지에서 15가지로 줄어 드는 것을 볼 수 있으며, n이 커질수록 중복되는 경우가 많이 생기며, 이러한 효과로 인해 프로그램 수행시간을 상당히 단축시킬 수 있다.

E1 : 5 1 2 4 , E2 : 5 1 4 3 , E3 : 6 2 4 3,
E4 : 6 1 4 4, E5 : 7 1 2 5

(a) 오류 사건(길이, 입력 정보 비트 가중치, 출력 잉여비트 가중치, 개수)

E1 + E1, E1 + E2, E1 + E3, E1 + E4, E1 + E5
E2 + E1, E2 + E2, E2 + E3, E2 + E4, E2 + E5
E3 + E1, E3 + E2, E3 + E3, E3 + E4, E3 + E5
E4 + E1, E4 + E2, E4 + E3, E4 + E4, E4 + E5
E5 + E1, E5 + E2, E5 + E3, E5 + E4, E5 + E5

(b) 가능한 오류 사건 조합(음영:중복되는 오류사건 조합)

① 길이 5 인 오류사건 + 길이 5 인 오류사건 => Sfactor = 2!/2!
E1 + E1, E2 + E2 => Efactor = 2!/2!
E2 + E1 => Efactor = 2!/1!1!

② 길이 5 인 오류사건 + 길이 6 인 오류사건 => Sfactor = 2!/1!1!
E1 + E3, E1 + E4, E2 + E3, E2 + E4 => Efactor = 2!/2!

③ 길이 5 인 오류사건 + 길이 7 인 오류사건 => Sfactor = 2!/1!1!
E1 + E5, E2 + E5 => Efactor = 2!/2!

④ 길이 6 인 오류사건 + 길이 6 인 오류사건 => Sfactor = 2!/2!
E3 + E3, E4 + E4 => Efactor = 2!/2!
E3 + E4 => Efactor = 2!/1!1!

⑤ 길이 6 인 오류사건 + 길이 7 인 오류사건 => Sfactor = 2!/1!1!
E3 + E5, E4 + E5 => Efactor = 2!/2!

⑥ 길이 7 인 오류사건 + 길이 7 인 오류사건 => Sfactor = 2!/2!
E5 + E5 => Efactor = 2!/2!

(c) 실제 검사되는 오류 사건 조합

그림 4. n=2인 경우에 대한 오류 사건 조합 예

Ⅳ. 실험 및 결과 고찰

본 논문에서 제안된 알고리듬을 사용하여 PCCC와 SCCC에 대한 ML 연판정 복호에 관한 비트오류확률의 상한치를 계산하였다. 1/3 부호율 PCCC의 기본 부호는 생성다항식 $G(D) = [1, 1+D^2/1+D+D^2]$를 사용하였으며, 1/3 부호율 SCCC는 1/2 부호율 외부부호(NSC)로서 $G(D) = [1+D+D^2, 1+D^2]$, 2/3 부호율 내부부호(RSC)로서 $G(D) = \begin{bmatrix} 1, & 0, & 1+D^2/1+D+D^2 \\ 0, & 1, & 1+D/1+D+D^2 \end{bmatrix}$를 사용하였다.

그림 5~7은 본 논문에서 제안한 알고리듬에 의한 1/3 부호율 PCCC 부호의 ML 연판정 복호에 대한 비트오류확률의 상한치이다. 그림 5은 식(4)의 D_m값을 35까지 고려한 결과이다.

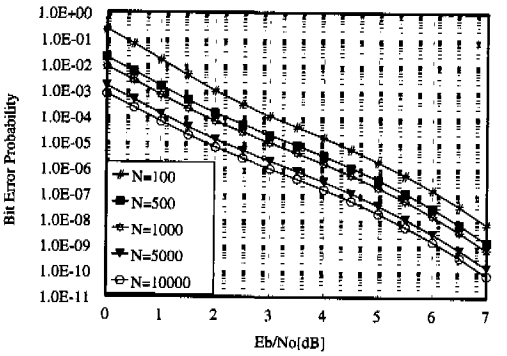


그림 5. PCCC 부호의 인터리버 길이에 따른 ML 연판정 복호에 대한 비트 오류 확률(D_m의 최대값 : 35)

그림 6은 인터리버의 길이를 100으로 고정시키고

D_m값을 25, 30, 35, 40까지 고려했을 때의 비트오류확률이며, D_m의 최대값이 바뀜에 따라 낮은 E_b/N_0에서의 비트오류확률이 증가하지만, 어느 정도 E_b/N_0가 증가하면 D_m의 최대값과 상관없이 비트오류확률이 일정하게 됨을 알 수 있다. 이 결과는 비트오류확률의 근사치의 유효성을 설명하는 것이다

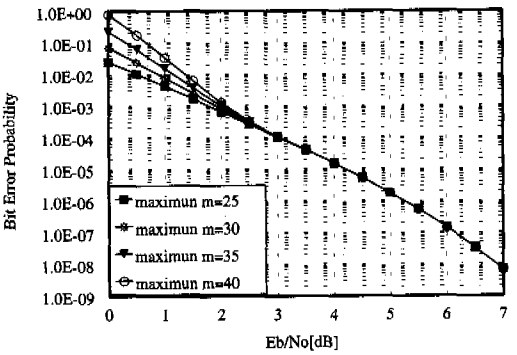


그림 6. PCCC부호에서 D_m의 최대값에 따른 ML 연판정 복호에 대한 비트 오류 확률(인터리버 길이=100)

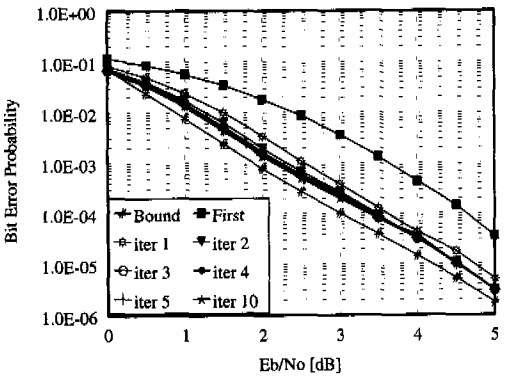


그림 7. PCCC부호의 ML 연판정 복호에 대한 비트 오류확률과 반복복호에 대한 시뮬레이션 결과와의 비교(D_m의 최대값 : 30, 인터리버 길이=100)

그림 7은 본 논문에서 제안된 알고리듬을 통해 구한 PCCC 비트오류확률의 ML 바운드와 실제 반복복호를 적용한 시뮬레이션 결과를 비교한 것이다. 그림에서 알 수 있듯이 ML 연판정 복호에 대한 비트오류확률의 상한치는 반복복호를 통해 얻을 수 있는 비트오류확률의 하한치가 됨을 알 수 있다.

그림 8~10는 1/3 부호율 SCCC(외부부호 NSC, 내부부호 RSC)에 대하여 본 논문에서 제안한 알고리듬을 통해 얻은 ML 연판정 복호에 대한 비트오류확률의 결과이다.

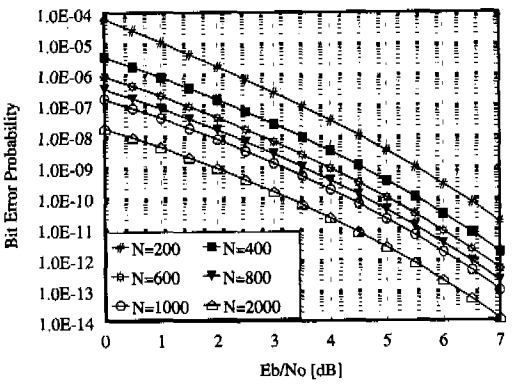


그림 8. SCCC 부호의 인터리버 길이에 따른 ML 연판정 복호에 대한 비트오류확률(D_m의 최대값 : 35)

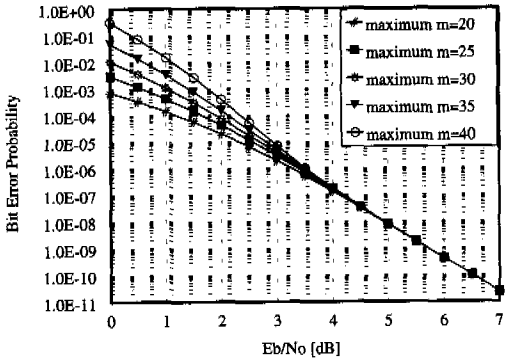


그림 9. SCCC 부호의 D_m의 최대값에 따른 ML 연판정 복호에 대한 비트오류확률(인터리버 길이=100)

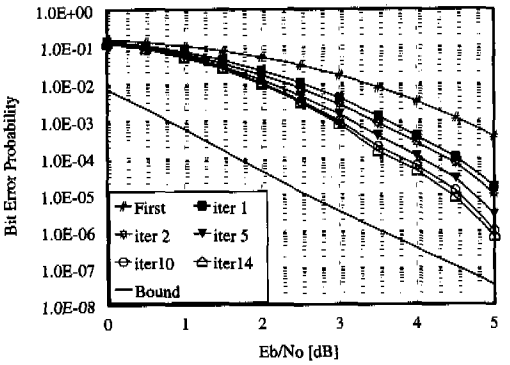


그림 10. SCCC 부호의 ML 연판정 복호에 대한 비트오류확률과 반복 복호에 대한 시뮬레이션 결과와의 비교 (D_m의 최대값 : 35, 인터리버 길이 =100)

이들 그림으로부터 내부부호로서 RSC부호를 사용했을 때 모든 신호대잡음비에서 인터리버 이득을 얻을 수 있음을 알 수 있다. 그림 8은 인터리버의

길이에 따른 SCCC부호의 비트오류확률을 나타내며, 그림 9는 인터리버의 길이가 100일 때 D_m의 최대값에 따른 비트오류확률이다. PCCC와 마찬가지로 신호대잡음비가 어느 정도 커지면 비트오류확률이 수렴함을 볼 수 있다. 그림 10은 ML 비트오류확률과 반복복호를 사용한 시뮬레이션의 결과와 비교한 것이다.

인터리버의 길이가 100이고 총 반복횟수는 15이다. 신호대 잡음비와 반복 횟수가 증가함에 따라 비트오류확률이 ML 상한에 접근해 간다.

V. 결 론

본 논문에서는 병렬 및 직렬 연쇄 컨볼루션 부호의 비트오류확률을 계산하기 위해 길이에 따른 오류사건을 이용하여, 가중치 열거 함수 계산 알고리듬을 제안하였다. 또한, 길이에 따른 오류사건을 탐색하는 알고리듬을 제안하였다. 제안된 알고리듬을 적용시켜본 결과, 효과적으로 가중치 열거 함수를 찾을 수 있었다. 실제 반복복호를 통한 실험 결과, 연쇄 컨볼루션 부호의 ML 연판정 복호에 대한 비트오류확률의 상한치는 실제 반복복호를 통해 얻을 수 있는 비트오류확률의 하한치가됨을 확인하였다.

참 고 문 헌

[1] G. D. Forney. Jr., *Concatenated codes*, The M.I.T. Press, 1966.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correctinf coding and decoding: Turbo codes," Proceeding of ICC '93, pp. 1064-1070, May 1993.

[3] S. Benedetto, G. Montorsi, "Unveiling Turbo Codes ; Some Results on Parallel Concatenated Coding Schemes," IEEE Trans. on Comm., Vol. 42, No. 2., pp.409-428., Mar., 1996.

[4] S. Benedetto, et al., "Serial Concatenation of Interleaved Codes : Performance Analysis, Design, and Iterative Decoding," TDA Progress Report 42-126, pp.1-26, Aug., 1996.

[5] M. Rouanne, D. J. Constello. Jr., "An algorithm for computing the distance spectrum of trellis codes," IEEE JSAC, Vol. 7, No. 6, pp. 929-940, Aug. 1989.

[6] M. Cedervall, R. Johannesson, "A fast alogrithm for computing distance spectrum of convolutional codes," IEEE Trans. on Info. Theory, Vol. 35, No. 6, pp.1146-1159, Nov. 1989.

[7] L. R. Bahl, et. al., "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. on Info. Theory, pp. 284-287, Mar. 1974.

[8] S. Benedetto, et al., "Performance evaluation of trellis-coded modulation schemes," Proceedings of the IEEE, Vol. 82, No. 6, pp. 833-855, June 1994.

[9] S. Benedetto, et al., "A Soft-Input Soft-Output Maximum A Posteriori(MAP) Module to Decode Parallel and Serial Concatenated Codes," TDA Progress Report 42-127, pp.1-20, Nov., 1996.

[10] D. Divsalar, S. Dolinar, F. Pollara, R. J. McEliece, "Transfer Function Bounds on the Performance of Turbo Codes", TDA Progress Report 42-122, pp.44-55, August 15, 1995.

강 성 진(Sung-Jin Kang) 정회원

1992년 2월 : 연세대학교 전자공학과 공학사

1994년 8월 : 연세대학교 대학원 전자공학과 공학석사

1998년 8월 : 연세대학교 대학원 전자공학과 공학박사

1998년 12월~현재 한국전자통신연구원 무선방송기술연구소 방송기술연구부 선임연구원

권 성 락(Sung-Lark Kwon) 정회원

1992년 2월 : 연세대학교 전자공학과 공학사

1994년 2월 : 연세대학교 대학원 전자공학과 공학석사

1998년 8월 : 연세대학교 대학원 전자공학과 공학박사

1998년 9월~현재 LG정보통신 중앙연구소 이동통신연구단 선임연구원

이 영 조(Young-Jo Lee) 정회원
1991년 2월 : 연세대학교 전자공학과 공학사
1993년 2월 : 연세대학교 대학원 전자공학과 공학석사
1997년 2월 : 연세대학교 대학원 전자공학과 공학박사
1997년 3월~현재 LG정보통신 중앙연구소 이동통신연구단 선임연구원

강 창 언(Chang-Eon Kang) 정회원
제23권 제10호 참조