효율적인 복잡도 감소 기법을 적용한 블럭 터보 부호

정회원 김 수 영*, 이 수 인*

Block Turbo Codes Using Efficient Reduced Search Trellis Decoding Method

Sooyoung Kim Shin*, Soo In Lee* *Regular Members*

요 약

본 논문에서는 트렐리스 복호 방식을 이용한 블록 터보 부호의 이론적인 성능 바운드와 함께 내부 구성 부호의 변형 및 내부 구성 부호의 연결 방법에 따른 성능의 변화를 살펴본다. 또한 효율적인 복잡도 감소 기법을 적용한 반복 복호 기법을 소개하고 시뮬레이션을 통하여 성능을 분석한 결과를 제시한다. 가우시안 채널에서의 시뮬레이션 결과에 따르면 본 논문에서 제시한 기법은 약 1/10정도의 복잡도를 가지고서도 전체 트렐리스를 탐색한 기법에 거의 근접하는 성능을 얻을 수 있음이 보여졌다.

ABSTRACT

In this paper, block turbo codes using trellis decoding method is introduced. Performance simulations of block turbo codes with various modifications of constituent codes and concatenation methods are demonstrated with theoretical performance bounds. An efficient complexity reduction method for trellis decoding is also introduced. Simulation results on an AWGN channel shows that the performance of the proposed complexity reduction method could approximate to that of the exhaustive search method with only 1/10 of the full load.

Ⅰ. 서 론

터보 부호[1]는 연판정 출력 값을 이용하여 반복 복호를 수행함으로써 엄청난 양의 부호화 이득을 얻을 수 있다. 이러한 강력한 오류정정 부호 능력으로 인하여 터보 부호는 여러 가지 응용 분야에 대하여 그 활용성이 거론되고 있다. 특히 차세대 이동통신 시스템인 IMT-2000 시스템에서도 데이터 전송을 위해서는 거의 모든 규격서에서 터보 부호의 채택을 고려 또는 확정하고 있는 상태이다. 원래 터보 부호의 개념은 길쌈 부호에 바탕을 둔 내부 구성 부호와 각각의 내부 구성부호를 연결한 인터리버로 구성이 되어있는 구조를 이용하여 반복적으로 복호를 수행하는 것이었는데, 최근에는 블록 부호를 내부 구성 부호로 사용하는 터보부호에 대한 논문들이 발표되고 있다[2].

이러한 블록터보부호가 가지고 있는 장점으로는 우선 블록 부호의 유한한 길이 특성으로 인한 간단한 구조와 길쌈 부호에서와는 달리 인터리버의 역할이 그다지 크지 않다는 것, 또한 비교적 높은 부호화 율에서 더 성능이 우수하다는 점 등이 있다. 특히 이러한 장점 때문에 블록터보 부호는 상용 칩으로도 이미 구현이 되어 있는 상태이다[3][4].

그러나 이제까지 제시된 거의 대부분의 블록 터보부호는 연판정 출력을 제공하기 위한 방법으로써 대수적인 복호 방식을 사용하고 있다. 본 논문에서는 연판정 입출력을 이용하여 복호할 수 있는 가장

* 한국전자통신연구원 무선.방송기술연구소 위성통신시스템연구부(sookim@etri.re.kr)
논문번호 : 99474-1129, 접수일자 : 1999년 11월 29일

효율적인 방법으로써 트렐리스 복호 방식을 사용하는 블록 터보 부호를 소개한다. 그러나 부호의 오류 정정 능력이 증가할수록 트렐리스의 복잡도가 증가하게 되므로 본 논문에서는 효율적으로 트렐리스의 복잡도를 줄일 수 있는 방법을 이용하여 거의 전체 트렐리스를 탐색했을 때와 동일한 성능을 얻을 수 있는 방식을 제시하고 그 시뮬레이션 성능을 비교해 보도록 한다.

따라서 본 논문에서는 블록 터보 부호의 기본 개념과 트렐리스 복호 방식, 특히 연판정 출력 비터비 알고리즘을 이용한 블록 터보 부호에 대하여 기술하고 이러한 알고리즘의 성능 향상 기법과 그 시뮬레이션 결과를 2장에서 살펴본다. 3장에서는 이러한 블록 터보 부호가 가질 수 있는 이론적인 성능 바운드를 도출하여 실제 시뮬레이션 성능과 비교해 볼 수 있도록 하며, 4장에서는 블록 부호의 변형과 내부 구성 부호의 연결에 따른 성능의 변화등 블록 터보 부호의 여러 가지 측면을 시뮬레이션 결과를 분석함으로써 살펴보도록 한다. 또한 본 논문에서 제시된 복잡도 감소 기법의 기본 개념을 5장에서 소개하고 이러한 복잡도 감소 기법을 적용하였을 경우의 성능에 대해서도 상세히 제시하기로 한다. 마지막으로 6장에서는 결론을 내리기로 한다.

Ⅱ. 블록 터보 부호

1. 연판정 출력 비터비 알고리즘을 이용한 블록 터보부호

블록 터보 부호는 곱 부호 (product code)를 이용하여 반복적으로 복호하는 알고리즘을 1994년 Pyndia가 처음으로 제안하였다[2]. 일반적인 블록 부호의 복호 방식에서는 연판정 정보를 사용하기가 매우 힘들기 때문에 그가 제안한 블록 터보 부호에서는 연판정 출력을 얻기 위하여 Chase 알고리즘[5]을 사용하였다. 그러나 원래의 Chase 알고리즘은 연판정 출력을 제공하기 위한 알고리즘이 아니라 블록 부호에 대하여 연판정 입력을 이용하여 복호를 수행하는 방식의 하나이므로 Pyndia는 연판정 출력을 제공하기 위한 별도의 루틴을 사용하였다. Pyndia 가 제안한 블록 터보 부호 방식은 기본적으로는 기존의 블록 복호에 대한 대수적인 복호 방식을 사용하는 것이다. 따라서 연판정 입력 및 출력을 사용하는 것이 그리 간단한 문제가 아니다.

상기의 방식이외에 선형 블록 부호는 트렐리스로 표현할 수 있고 따라서 길쌈부호에서와 같은 방식으로 연판정 정보를 이용한 비터비 복호가 가능하다[6]. 이러한 사실은 블록 부호에 대해서도 연판정 출력 비터비 복호 알고리즘이나(soft output Viterbi algorithm ; SOVA)[7] 최대 사후확률추정 (maximum a posteriori ;MAP) 알고리즘과[8] 같은 트렐리스를 이용한 반복 복호가 가능함을 시사해 준다고 할 수 있을 것이다. MAP 알고리즘을 이용한 블록 터보 부호의 성능에 대해서는 이미 간단한 시뮬레이션 결과가 제시된 바 있다[9]. 그러나 MAP 알고리즘에서 요구하는 엄청난 계산량의 단점을 극복하기 위하여 본 논문에서는 그 구현 방법이 비교적 간단하고 복잡도가 적은 SOVA를 이용한 블록 터보부호를 이용하였다.

SOVA를 이용한 길쌈 터보 부호에 대한 연구는 이미 여러 논문에서 발표된 바 있으며[10] 그 성능에 대해서도 다양한 결과가 제시되어 있다[11]. 이를 기본으로 하여 SOVA를 이용한 블록터보부호의 동작 원리를 정리해 보기로 한다. SOVA를 이용한 블록 터보부호에서도 마찬가지로 반복 복호를 수행하는 기본 동작 원리는 다른 터보부호에서와 동일하다. 그림 1은 (n,k) 블록 부호를 내부 구성 부호로 사용하고 사이즈가 k^2인 블록 인터리버를 이용한 블록 터보부호에서의 반복 복호 동작원리를 나타낸 것이다. 그림 1에 나타나 있는 블록 터보 부호는 정확하게 곱부호가 아니라 인터리버를 사이에 둔 병렬 연접 부호라고 볼 수 있다. 그림 2에는 그림 1의 블록 터보 부호의 동등 모델(equivalent model)이 나타나 있다.

그림 1에 나타나 있듯이 우선 열 부호에 대한 복호를 수행함에 있어서 채널에서 수신된 원래의 신뢰도 정보 $L(u)$및 $L(y)$와 행 부호에 대한 복호로부

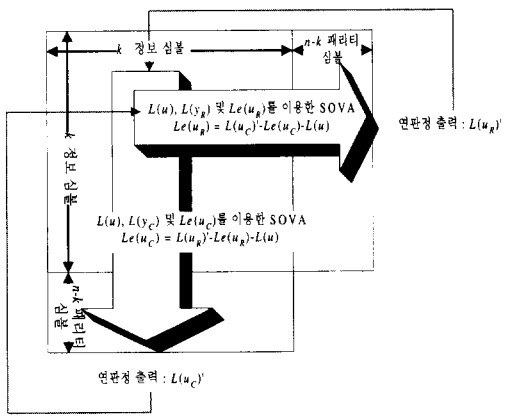


그림 1. (n,k) 블록 부호를 구성 부호로 하는 SOVA를 이용한 병렬연접 블록 터보 부호의 동작 원리

터 얻은 신뢰도 정보를 이용하여 구한 외부 신뢰도 정보 (extrinsic information) $Le(u_R)$를 함께 사용하여 비터비 복호를 수행한다.

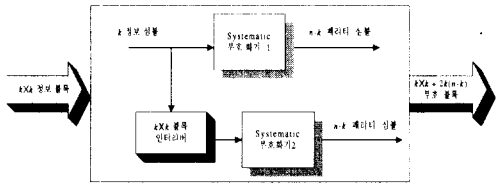


그림 2. (n,k) 블록 부호를 구성 부호로 하는 병렬연접 블럭 터보부호의 동등모델

여기서 $L(u)$는 systematic 심볼에 대한 채널에서의 신뢰도 정보이며, $L(y)$는 패리티 심볼에 대한 채널에서의 신뢰도 정보이다. 열 부호에 대한 외부 신뢰도 정보 $Le(u_R)$은 행부호에 대한 비터비 복호의 연판정 출력 값 $L(u_C)$에서 원래의 채널 신뢰도 정보 $L(u)$와 그 단계에서 사용된 외부 신뢰도 정보를 빼서 그 단계 복호로부터 얻어진 순수한 신뢰도 정보를 추출하여 사용한다. 다시 말하면 임의의 단계에서 출력된 신뢰도 정보는 그 단계의 입력으로 사용된 신뢰도 정보를 함께 포함하고 있으므로 이들을 빼줌으로써 순수한 그 단계에서 복호로부터 발생한 신뢰도를 나타내도록 하는 것이다. 즉, 행 부호와 열 부호에서의 외부 신뢰도 정보는 다음과 같이 구할 수 있다.

$$\begin{aligned} Le(u_R) &= L(u_C)' - Le(u_C) - L(u) \\ Le(u_C) &= L(u_R)' - Le(u_R) - L(u) \end{aligned} \quad (1)$$

상기의 SOVA를 이용한 블록 터보 복호 알고리즘을 정리하면 다음과 같다

(1) 열 부호에 대한 복호기에 $L(u)$, $L(y_R)$ 및 $Le(u_R)$을 입력하여 연판정 출력 비터비 복호를 수행한다. 초기의 $Le(u_R)$ 값은 모두 0으로 정해져 있다.

(2) 열부호에 대한 복호의 결과로 연판정 출력 $L(u_R)$을 얻을 수 있다.

(3) 열부호에 대한 연판정 출력 $L(u_R)$을 이용하여 행부호에 대한 외부 신뢰도 정보를 식 (1)로부터 구한다.

(4) 행 부호에 대하여 다시 $L(u)$, $L(y_C)$ 및 $Le(u_C)$을 입력하여 연판정 출력 비터비 복호를 수행한다.

(5) 행부호에 대한 복호의 결과로 연판정 출력 $L(u_C)$을 얻을 수 있다.

(6) 행부호에 대한 연판정 출력 $L(u_C)$을 이용하여 열부호에 대한 외부 신뢰도 정보를 식(1)로부터 구한다.

(7) (1)부터 반복한다.

2. SOVA에 대한 성능 향상 기법

일반적으로 SOVA는 MAP 알고리즘에 비하여 반복복호 알고리즘에서 열등한 성능을 나타내는데 그 이유는 SOVA에서 출력되는 신뢰도 정보가 지나치게 낙관적으로 계산되는데 있다. 따라서 본 절에서는 이러한 지나친 낙관적 정보를 교정하여 성능을 향상시킬 수 있는 방법을 살펴보기로 하는데, 연판정 출력 값의 갱신 방법과 연판정 출력 값의 정규화 방법이 이에 해당한다.

2.1 성능 향상을 위한 연판정 출력 값의 갱신 방법

SOVA의 신뢰도, 즉 연판정 출력 계산 방법에 의하면 현재 시점에서 생존 경로와 경쟁 경로의 미터릭 차이 값을 계산한 후 이전 시점에서의 연판정 출력 값을 갱신해 주어야 하는데, 이때 이전 시점에서 생존 경로와 경쟁 경로에서의 정보 비트가 다른 시점에 대해서만 그 값을 갱신해 주는 것이 원래의 SOVA에서 수행하는 방법이다. 그러나 이러한 방식이 지나치게 낙관적인 신뢰도 정보를 도출하는 것으로 알려졌고[10], 따라서 생존 경로와 경쟁 경로의 정보 비트 값이 다른 시점에 대해서도 그 값을 갱신해 주어야 할 필요가 있다. 따라서 여러 가지 선행 연구에서 증명된 바에 따르면 다음과 같은 신뢰도 갱신 방법을 사용하는 것이 최적이라고 할 수 있다. 또한 아래와 같은 신뢰도 갱신 방법을 사용할 경우 SOVA는 Max-log-MAP 복호 방식과 동일하는 것이 밝혀졌다[12].

k시점에서의 정보 비트 u^k에 대하여 계산된 신뢰도 값을 $L(\hat{u}_k)$라고 하고, k시점에서 트렐리스의 상태 m으로 들어오는 생존 경로와 경쟁 경로와의 미터릭 차이값이 Δ_k^m이라고 하면 k시점에서의 상태 m에서의 신뢰도 값 L_k^m의 초기 치는 Δ_k^m가 될 것이다. 그리고 최종 단계에서의 생존 경로는 하나 뿐일 것이므로 결국

$$L(u_k) = \hat{u}_k L_k^m \quad (2)$$

그러나 이러한 신뢰도 값은 시간이 진행됨에 따라 다음과 같은 규칙에 따라 계속해서 갱신해 주어야 한다. 임의의 j 시점($j<k$)에서의 생존 경로에서

계산된 정보 비트를 u_j^t라 하고 경쟁 경로에서 계산된 정보 비트를 u_j^c라고 하면 갱신해 주는 규칙은 다음과 같다.

$$\begin{aligned} &\text{if } u_j^t \neq u_j^c \text{ then } L_j' = \min(L_j', \Delta_k^m) \\ &\text{if } u_j^t = u_j^c \text{ then } L_j' = \min(L_j', \Delta_k^m + L_j^c) \end{aligned} \quad (3)$$

일반적으로 직렬 연접부호에서는 상기에서 언급한 새로운 갱신 규칙이 성능 개선에 있어서 별 효과가 없는 것으로 나타났다. 그러나 일반적인 병렬 연접 길쌈 터보 부호에서는 10^{-4} 정도의 비트오률 구간에서 400 비트 사이즈의 일반적인 블록 인터리버를 사용한 경우에 8번 반복 복호 후 약 0.5dB의 성능 개선이 있음이 밝혀졌다[13].

2.2 성능 향상을 위한 연판정 출력 값의 정규화 방법

일반적으로 가우시안 채널에서 SOVA의 입력이 되는 로그 확률로 표현된 신뢰도 값은 가우시안 확률 분포를 가진다고 가정할 수 있다. Papke와 Robertson[10]은 이를 이용하여 SOVA를 이용한 반복 복호의 성능을 향상시키는 기법을 제안하였는데, 여기에서는 이 기법에 대하여 좀 더 자세히 살펴보기로 하자. 그들은 SOVA를 사용하는 복호기에서의 연판정 출력 값이 가우시안 분포를 가진다는 것으로부터 SOVA의 입력이 되는 로그 확률로 표현된 신뢰도 값이 어떻게 표현되는지를 유도하였다. 임의의 시점에서 정보 비트 u가 1이었다는 가정하에 SOVA 복호기 출력 v의 확률 분포는 다음과 같이 정의할 수 있다.

$$p(v|u=1) = \frac{1}{\sqrt{2\pi}\sigma_v} \exp\left(-\frac{(v-m_v)^2}{2\sigma_v^2}\right) \quad (4)$$

여기서 m_v는 v의 평균이며 σ_v는 v의 표준편차이다. SOVA 복호기 출력이 주어졌다는 가정하에 조건부 로그 확률, 즉 다음 단에서의 SOVA의 외부 입력 정보을 계산하면 다음과 같이 표현할 수 있다.

$$\begin{aligned} L_u &= \ln \frac{p(v|u=1)}{p(v|u=-1)} \\ &= \ln\left[\exp\left(-\frac{(v-m_v)^2 - (v+m_v)^2}{2\sigma_v^2}\right)\right] \\ &= m_v \frac{2}{\sigma_v^2} v \end{aligned} \quad (5)$$

단, 여기서 $p(u=1) = p(u=-1)$ 이라고 가정하고 Bayes's rule을 적용하였다.

식 (5)의 결과의 의미는 연판정 출력이 다음 복호단에서 로그 확률 값(LLR)으로써 입력이 되기 위해서는 연판정 출력 값 v에 정규화 상수 c를 곱하여 주어야 한다는 것이다.

$$c = \frac{2m_v}{\sigma_v^2} \quad (6)$$

이러한 c 값, 즉 정규화 상수는 신호 대 잡음비가 높아질수록 증가하게 되는 값이다. 그 이유는 신호 대 잡음 비가 증가 할수록 수신되는 연판정 출력 값, 즉 복호되는 값에 대한 신뢰도의 평균 값 m_v는 증가할 것이며, 이에 대한 분산 값 σ_v는 점점 감소하게 될 것이기 때문이다.

그림 3은 통계적으로 조사한 신호 대 잡음 비의 변화에 대한 정규화 상수 c의 변화를 나타내 준다. 본 논문에서는 채널의 신호대 잡음비에 대한 사전 정보가 없이도 정규화 상수를 용이하게 사용할 수 있도록 각 블록에서의 연판정 출력 값의 평균과 분산을 구함으로써 실제 복호를 수행할 때 그리 크지 않은 연산량의 추가만으로 효율적으로 사용할 수 있게 하였다.

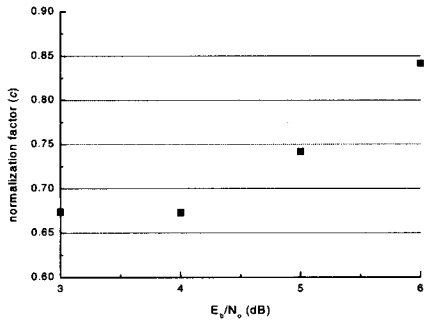


그림 3. (31,26) BCH 부호를 이용한 블록 터보 부호에서의 신호 대 잡음비에 따른 정규화 상수의 변화

2.3 시뮬레이션 결과

그림 4와 그림 5에는 이러한 BCH 블록 터보 부호에서의 여러 가지 복호 기법에 따른 BER 성능을 나타내었다. 그림 4의 (15,11) BCH 부호를 이용한 블록 터보 부호에서는 일반적인 SOVA를 적용할 경우 MAP 알고리즘에 비해 약 0.5dB 정도의 성능 열화를 나타내지만, 본 문서에서 제시한 정규화 기법을 적용할 경우 약 0.3dB 정도의 성능 향상을 보여 MAP에 비하여 0.2dB 정도 성능이 떨어진 BER 특성을 나타낸다고 할 수 있다.

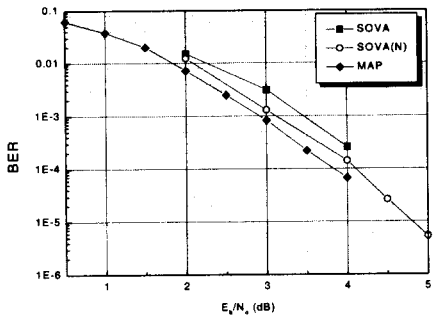


그림 4. (15,11) BCH 블록 부호를 병렬로 연접한 블록 터보 부호(R = 0.58)의 6회 반복 복호 후의 BER 성능, (SOVA(N) : 성능향상 기법을 적용한 SOVA)

그러나, 부호 자체의 오류정정 능력이 향상될수록 MAP 알고리즘과 SOVA와의 성능 차이는 점점 더 커지게 되는데, 그림 5의 (31,26) BCH 부호를 이용한 블록 터보 부호에서는 MAP 알고리즘과 일반적인 SOVA의 성능 차이가 10^{-4} 정도의 BER 값에서 약 0.8dB 정도가 됨을 알 수 있다. 그러나 정규화 기법을 적용한 SOVA를 이용할 경우 약 0.4dB 정도의 성능 향상을 보여 MAP 알고리즘과는 약 0.4dB 정도의 성능 차이만을 보임을 알 수 있다.

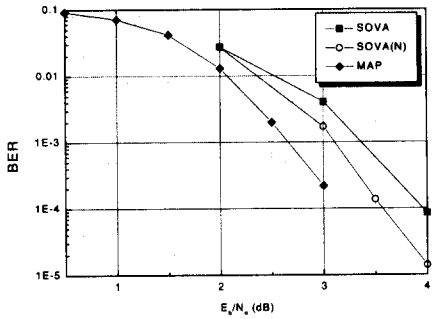


그림 5. (31,26) BCH 블록 부호를 병렬로 연접한 블록 터보 부호(R = 0.72)의 5회 반복 복호 후의 BER 성능, (SOVA(N) : 성능향상 기법을 적용한 SOVA)

Ⅲ. 병렬 연접 블록 터보 부호의 이론적인 성능

블록 부호에 대한 연판정 최우 복호의 이론적인 성능 바운드는 부호의 웨이트 분포를 이용하여 구할 수 있다[14]. 따라서 이러한 블록 부호를 내부 구성부호로 이용한 블록 터보 부호의 성능도 구할 수 있을 것이다. 그러나, 블록 터보 부호의 성능은 각 내부 구성 부호의 웨이트 분포를 알 수 있다 하더라도 인터리버에 따라 성능이 좌우되므로 좀 더 복잡하게 얽혀있다. 즉, 우선 임의의 인터리버에 대하여 두 번째 부호화기에 의해 생성되는 패리티 비트는 입력 정보 비트에만 의존하는 것이 아니라 이것들이 인터리버에 의해 어떤 식으로 배열되느냐에 따라서 좌우되기 때문이다.

이론적으로 이러한 경우에 웨이트 분포를 구할 수 있는 유일한 방법은 모든 경우에 대한 가능성을 다 조사해 보는 것이다. 즉 이진 부호인 경우에 $2^{(\text{interleaver length})}$ 가지에 대하여 조사하는 것이다. 그러나 이러한 방법은 인터리버의 길이가 큰 경우에는 거의 불가능 하다. 상기한 바와 같은 문제점을 해결하기 위하여 Benedetto[15]는 확률적인 개념을 가지는 유니폼 인터리버를 (uniform interleaver)[4] 이용하여 병렬로 연결된 블록 터보 부호의 이론적인 성능 바운드를 도출할 수 있는 방법을 제시하였다.

그림 6에는 (7,4) BCH 부호를 인터리버의 길이가 4×4=16인 블록 인터리버를 사용하여 병렬로 연접한 부호에 대한 시뮬레이션 결과와 상기한 바와 같은 유니온 바운드 계산 결과가 나타나 있다. 본 논문에서 제시된 모든 블록터보부호는 (n,k) 내부 구성 블록 부호에 대하여 구조가 가장 간단한 $k \times k$ 블록 인터리버를 사용하였다. 그림 6에 나타난 (7,4) BCH 부호에 대한 시뮬레이션에서는 SOVA를 이용한 반복 복호 기법을 사용하였고 6회 반복 후의 성능을 제시하였다.

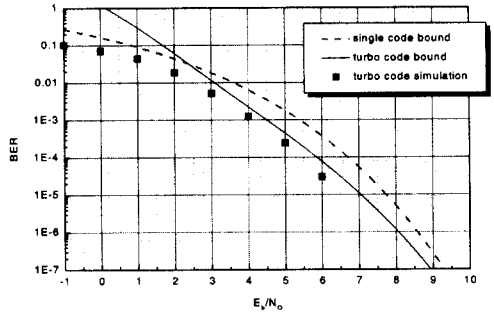


그림 6. (7,4) BCH 부호에 대한 병렬 연접 부호의 SOVA를 이용한 6회 반복 후의 시뮬레이션 성능과 유니온 바운드 비교

그림 6에 나타난 결과를 보면 비트 오율이 낮은 구간에서 시뮬레이션 결과는 유니온 바운드에 거의 근접함을 알 수 있다. (7,4) BCH 부호 자체의 오류정정 능력이 비교적 낮고 인터리버 사이즈도 16으로 비교적 적기 때문에 반복 횟수를 6회로 한정했다 하더라도 무한대로 반복한 성능과 거의 차이가

없다. 또한 3.3 절의 시뮬레이션 결과에서 살펴본 바에 의하면 오류정정 능력이 낮을수록 MAP 복호와의 성능 차이는 별로 없기 때문에 SOVA를 이용한 반복 복호의 성능이 (7,4) BCH 부호에서는 MAP 복호와 거의 성능 차이가 없을 것이다.

그림 7에 나타나 있는 (15,11) BCH 부호에 대한 시뮬레이션 결과와 유니온 바운드 계산 결과를 비교함으로써 좀 더 자세히 살펴보도록 하자. 여기에는 MAP 알고리즘을 이용한 시뮬레이션 결과도 동시에 나타나 있다. 그런데 여기에서는 부호어의 수가 16개 밖에 되지 않아 모든 부호어와의 거리 분포를 모두 고려할 수 있었던 (7,4) BCH부호에서와는 달리 전체 부호어의 수가 2^{11}개 이므로 최소 거리 d_{min} = 3인 부호어들만을 고려했을 때의 유니온 바운드와 거리가 7까지 되는 부호어들을 고려하였을 때의 유니온 바운드 두 가지를 나타내었다. 그러나 두 가지 바운드가 BER 값이 낮은 구간에서는 거의 차이가 없음을 알 수 있다.

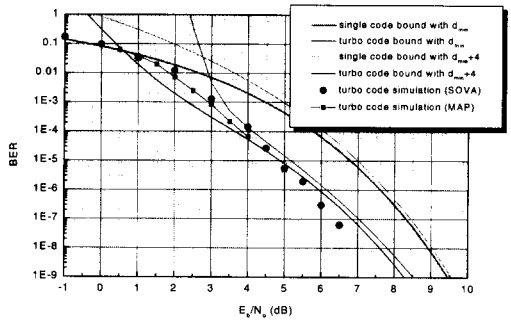


그림 7. (15,11) BCH 부호에 대한 병렬 연접 부호의 SOVA 및 MAP을 이용한 6회 반복 후의 시뮬레이션 성능과 유니온 바운드 비교

시뮬레이션 결과가 일부 구간에서 유니온 바운드보다 큰 값을 나타내는 것은 고려하는 부호어의 개수가 거리가 적은 일정 부호어들로 제한되었기 때문이며, BER 값이 낮아질수록 그 경향이 반전됨을 알 수 있다. 그러나 여기에서 BER 값이 점점 낮아질수록 바운드에서 멀어지는 정도가 점점 더 심화되는 듯한 경향을 볼 수 있다.

Ⅳ. 내부 구성 블록 부호의 변형과 연결 방법에 따른 성능 변화

1. 블록 터보 부호를 위한 블록 부호의 변형

앞 절에서 기술된 블록 터보 부호는 모두가 내부 부호를 병렬로 연결한 형태였다. 그러나 블록 터보 부호를 처음으로 제안한 Pyndia[2]는 내부 부호를 직렬로 연결한 형태를 사용하였는데 이는 터보 부호가 나오기 전에 이미 알려진 곱 부호(product code)의 형태이다. 이러한 곱 부호의 최소 거리는 각 구성 부호의 최소 거리의 곱이 된다. 이러한 점에서 변형된 블록 부호를 사용해야 하는 필요성이 제기되는 것이다. 즉 각 구성 부호어의 부호화율을 조금 감소시켜서 부호의 최소 거리를 1 증가 시키면 직렬로 연결된 블록 터보 부호 입장에서 볼때는 부호화율의 큰 감소 없이 최소 거리는 크게 증가 시킬 수 있기 때문이다.

예를 들어 (31,26) BCH 부호를 확장 시킨 (32,26) extended BCH 부호를 사용하여 직렬 블록 터보 부호를 구현할 경우 부호화율에서 약 0.3dB의 감소가 있지만 최소 거리는 9에서 16으로 크게 증가하게 된다. 따라서 본 절에서는 이러한 잇점을 활용할 수 있는 블록 부호의 변형과 이를 트렐리스로 어떻게 구현할 수 있는지에 대하여 살펴보기로 한다.

블록 부호의 여러 가지 변형 기법 중 최소 거리를 증가 시켜 그 성능을 향상 시킬 수 있는 방법으로는 부호의 확장(extension)과 부호의 삭제가 (expurgation) 있다[14]. 부호의 확장은 부호의 패리티 심볼을 하나 더 증가 시키는 작업이다, 즉, (n, k) 부호를 (n+1,k) 부호로 변형하는 것인데 일반적으로 부호의 최소 거리를 증가 시킬 수 있도록 패리티를 결정한다. 부호를 확장 시키는 가장 흔한 방법은 싱글 패리티 체크를 첨가하는 것이다. 이런 방식으로 변형이 되면 모든 부호어의 웨이트는 항상 짝수가 된다. 일반적으로 이러한 방식으로 변형된 부호는 시클릭 부호가 아니다. 예를 들어 최소 거리가 3인 (7,4) BCH 부호를 확장시키면 최소 거리가 4인 (8,4) 부호가 된다.

부호의 삭제는 부호어들 중 일부를 없애 버리는 것이다. 시클릭 부호에 대하여 이러한 작업은 생성다항식 $g(x)$에 x^n-1의 인자(factor)를 곱함으로써 수행할 수 있는데 가장 흔한 방법 중의 하나가 생성다항식 $g'(x)$를 $g(x)(x$-1)로 사용하는 것이다. 이러한 방법을 사용하면 부호어의 웨이트는 항상 짝수가 되는데 그 이유는 다음과 같다.

생성다항식에 (x-1)을 곱한다는 의미는 부호어를 왼쪽으로 한번 쉬프트 시킨 부호어를 원래 부호어에 더하는 것을 말한다. 따라서 이러한 작업으로 생성된 부호어의 웨이트는 항상 짝수가 되게 된다. 두 개의 1이 더해져서 사라지게 될 때는 두개가 동시

에 사라지게 되므로 홀수 더하기 홀 수는 짝수이며 짝수 더하기 짝수는 짝수가 되는 원리가 같은 것이다. 삭제(expurgation)라는 단어가 붙게 된 것도 임의의 부호어를 없애 버린다는 뜻을 의미하는 것이다. 이러한 방식으로 변형된 시클릭 부호는 시클릭 부호의 특성을 그대로 유지하게 된다. (7,4) BCH 부호를 삭제하게 되면 정보어의 길이가 1이 준 (7,3) 부호가 된다.

본 논문에서 다루고 있는 블록 터보 부호는 모두 트렐리스 기법을 이용하여 구현한 방법이므로 변형된 부호에 대한 트렐리스를 어떻게 구현해야 하는지가 가장 큰 문제가 된다. 삭제된 부호에 대한 트렐리스는 생성 다항식을 구할 수 있으므로 FIR 필터 모델[16]을 이용하여 쉽게 구할 수 있다.

확장 부호에 대해서는 두 가지 방법을 생각해 볼 수 있다. 가장 간단한 구현 방법은 원래 부호의 트렐리스를 그대로 사용하고 최종 단에서 싱글 패리티를 채크하는 방법이다. 즉 원래 부호와 같이 복호를 하고 최종 단에서 패리티를 검사하여 마지막 수신된 심볼과 비교하는 방법이다. 그 다음 방법은 확장 부호의 패리티 체크 메트릭스를 알 수 있으므로 [14] 패리티 체크 메트릭스를 사용하여 트렐리스를 구현하는 방식[6]이다. 본 논문에서는 위의 두 가지 방식에 대한 성능을 시뮬레이션 한 결과를 모두 비교 제시하기로 한다. 본 논문에서는 첫 번째 방식을 확장 1 방식이라고 하고 두 번째 방식을 확장 2방식이라고 한다.

2. 내부 부호의 연결 형태 및 변형에 따른 성능 변화

그림 8에는 길이 15인 BCH 부호와 이를 변형한 부호들을 병렬로 연결한 블록 터보부호에 대한 BER 성능이 나타나 있다. 그림 8에 나타나 있는 바와 같은 확장 부호와 삭제 부호에 대한 성능이 원래 부호의 성능보다는 우수하게 나타나며, 두 부호의 성능은 거의 동일함을 알 수 있다. 또한 그림 9에는 각 부호들의 연결 형태에 대한 성능이 나타나 있다.

그림 9에 나타나 있는 결과를 보면 직렬로 연결한 방식이 병렬로 연결한 방식에 비하여 10^{-5}-10^{-6} 정도의 BER 구간에서 약 0.5dB 정도의 이득이 있음을 알 수 있으며, 그림 8에서와 마찬가지로 확장 부호와 삭제 부호의 반복 복호 성능은 거의 동일함을 알 수 있다. 그러나 확장 부호의 트렐리스 구현 방법에 있어서 확장 2방식이 확장 1 방식에 비하여 성능이 더 우수함을 알 수 있는데 그 이유는 다음과 같다.

확장 1 방식의 경우에는 원래 부호의 트렐리스를 그대로 사용하고 맨 마지막에 패리티에 대해서만 미터릭을 추가로 계산하게 되므로 전체 존재하는 트렐리스, 즉 부호어들 중 일부만을 탐색하는 것과 같은 결과를 낳게 되어 열등한 결과를 보이게 되는 것이다.

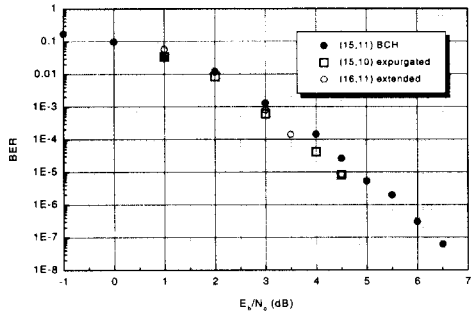


그림 8. (15,11) BCH 부호와 그 변형 부호에 대한 병렬 연접 블록터보부호의 6회 반복 후의 BER 시뮬레이션 성능

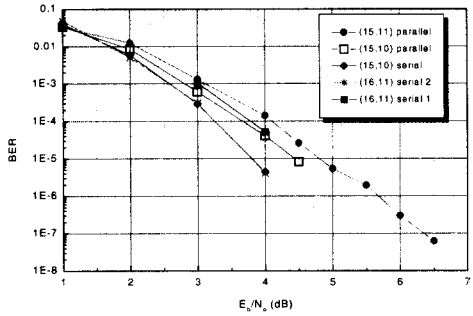


그림 9. (15,11) BCH 부호와 그 변형 부호에 대한 연결형태에 따른 블록터보부호의 6회 반복 후의 BER 시뮬레이션 성능 (serial 1 : 확장 1방식, serial 2 : 확장 2 방식)

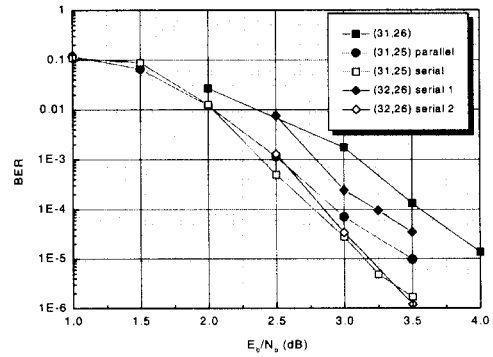


그림 10. (31,26) BCH 부호와 그 변형 부호에 대한 연결형태에 따른 블록터보부호의 6회 반복 후의 BER 시뮬레이션 성능 (serial 1: 확장 1방식, serial 2 : 확장 2 방식)

그림 10에는 각각 (31,26) BCH 부호와 그 변형 부호에 대한 연결형태에 따른 블록 터보 부호의 성능이 나타나 있다. 그림 10에서도 그림 9에서와 마찬가지로 직렬 연결 부호의 성능이 병렬 연결 부호의 성능보다는 우수하며 확장 또는 삭제 변형된 부호의 성능이 원래 부호보다는 더 우수함을 알 수 있다.

V. 블록 터보 부호에서의 효율적인 복잡도 감소 기법

1. 복잡도 감소 SOVA

SOVA는 기본적으로 비터비 알고리즘이므로 경로 미터릭의 통계치를 이용한 복잡도 감소 기법[17]을 쉽게 적용할 수 있다. 그러나, 여기서 문제는 이러한 SOVA가 블록 터보 부호에서와 같이 그 연판정 출력이 후단에서 효율적으로 사용되기 위해서는 신뢰도 정보를 어떻게 정의해 주느냐가 가장 큰 문제일 것이다. 본 논문에서는 복잡도 감소 기법을 사용한 SOVA에서 신뢰도 정보 부여 방법에 대하여 핵심적으로 다루고 본 논문에서 기술된 방법을 사용하였을 때의 간단한 BER 성능 시뮬레이션 결과를 제시하기로 한다.

경로 미터릭의 통계치를 이용한 복잡도 감소 기법의 원리는 경로 미터릭의 확률 분포가 정규 분포를 따른다는 가정 하에 트렐리스의 임의의 시점에서 존재할 수 있는 총 경로의 수 S보다 적은 B개의 경로만을 선택하기 위하여 경로 미터릭의 평균과 분산 값을 이용하여 기준 미터릭 값을 구한다는 것이다. 즉 이 기준 미터릭보다 적은[1] 이 값은 경로 미터릭을 어떻게 정의하느냐에 따라 큰 값이 될 수도 있고 적은 값이 될 수도 있다. 일반적으로 유클리디언 거리를 사용할 경우에는 적은 값이 된다. 경로 미터릭을 가진 경로의 수가 총 B개가 된다는 것이다. 따라서 이 방식에서는 각 시점에서 평균적으로 거의 B개의 경로를 유지하게 된다.

따라서 위의 방식을 사용하게 되면 임의의 시점에서 항상 총 경로의 수 보다 적은 경로가 존재하게 되어 임의의 노드에서 단 한 개의 경로만이 유입되어 경로 미터릭의 비교없이 생존 경로가 저절로 결정되는 경우가 발생하게 되는 것이다. 여기서 SOVA를 적용할 때의 문제가 발생하게 된다. 즉, SOVA에서 신뢰도 정보를 생산하는 기본 원리는 임의의 노드에서 생존 경로와 경쟁 경로와의 차이 값을 신뢰도로 정의하게 되는데 경쟁 경로가 존재하지 않게 되면 신뢰도 값을 정의하는데 어려움이 있게 되는 것이다. 이러한 과정을 간단한 BCH 부호에 대한 비터비 복호 과정을 이용하여 살펴보도록 하자.

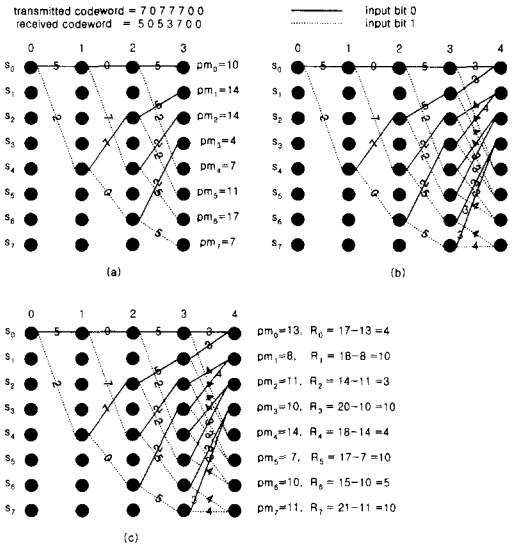


그림 11. (7,4) BCH 부호에 대한 SOVA 적용 예

그림 11은 (7,4) BCH 부호에 대하여 SOVA를 적용하는 예를 나타내준다. 3 비트로 양자화된 연판정 정보를 이용하여 복호를 수행한다고 가정해보자. 그림에서 s_i는 트렐리스의 각 상태를 나타내며, pm_i는 i번째 상태로 들어오는 경로의 미터릭 값을 나타낸다. 또한 각 노드에서 노드로 연결되는 가지 위의 숫자들은 해당 가지의 미터릭을 나타낸다. 트렐리스의 x축은 시간의 진행을 나타내는데 시점 4에서부터 각 노드에 여러 개의 경로가 동시에 유입되게 된다.

그림 11의 (c)에 나타나 있는 바와 같이 각 노드에 두 개의 경로가 동시에 유입되면 경로 미터릭 값이 적은 경로가 생존 경로가 되며, 생존 경로의 미터릭 값을 저장하게 된다. 또한 R_i는 i번째 상태로 들어오는 경로의 그 시점에서의 신뢰도(즉, 연판정 출력)를 나타내는데 그 값은 그림 11의 (c)에 나타나 있는 바와 같이 한 노드로 유입되는 두 경로의 미터릭의 차이 값이 된다.

1) 이 값은 경로 미터릭을 어떻게 정의하느냐에 따라 큰 값이 될 수도 있고 적은 값이 될 수도 있다. 일반적으로 유클리디언 거리를 사용할 경우에는 적은 값이 된다.

그러나, 만약 여기에 전체 경로 중 일부의 경로만을 선택하여 유지하는 복잡도 감소 기법을 적용한다고 가정해보자. (7,4) BCH부호에 대한 트렐리스에는 총 8개의 경로가 존재하게 되는데 예를 들어 이중 6개의 경로만을 선택하여 유지하는 복잡도 감소 기법을 적용한다고 가정해 보자. 그림 12에는 그림 11과 동일한 경우에서 전체 경로 중 가장 우수한 6개의 경로만을 선택하여 유지하는 경우를 보여준다.

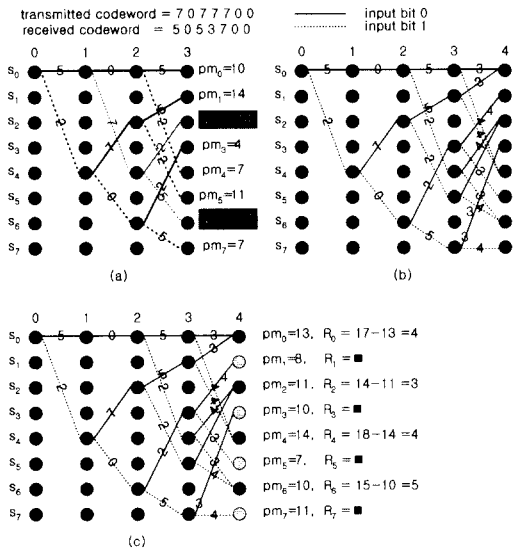


그림 12. (7,4) BCH 부호에 대한 복잡도 감소 SOVA 적용 예

그림 12의 (a)에 나타난 바와 같이 시점 3에서부터는 전체 경로의 수가 유지하고자 하는 경로의 수 6보다 크므로 이중 경로 미터릭 값이 큰 경로 두개를 제거한다. 그림 12의 (a)에 나타나 있듯이 상태 2로 들어오는 경로와 상태 7로 들어오는 두 경로의 미터릭이 가장 큰 값이므로 이 두 경로를 제거하였다고 가정한다. 따라서 그림 12의 (b)를 살펴보면 시점 3에서 상태 2와 상태 7에서는 경로가 존재하지 않고 총 6개의 경로만이 존재하고 있으며, 상태 2와 상태 7에 해당하는 노드에서는 시점 4로 갈 때 경로의 확장이(path extension) 일어나지 않음을 보여 준다.

복호를 진행하여 시점 4에 이르게 되면 시점 3에서의 6개 경로로부터 다시 경로를 확장하게 되는데, 시점 3에서 확장하지 않은 두 경로로 인하여 그림 12의 (c)에 나타난 바와 같이 시점 4에서는 두 개의 경로가 동시에 유입되지 않고 한 개의 경로만이 유입되는 노드가 발생하게 된다. 이러한 경우에는 일반적인 SOVA에서와는 달리 경쟁 경로가 존재하지 않기 때문에 신뢰도 값을 정의할 수가 없게 된다.

앞서 기술한 바와 같이 복잡도 감소 기법을 적용한 SOVA에서는 경쟁 경로가 존재하지 않는 노드가 생기기 때문에 이러한 경우에는 일반적인 방법으로는 신뢰도 값을 정의할 수가 없게 된다. 따라서 이처럼 경쟁 경로가 존재하지 않는 경우에 신뢰도 값을 어떻게 정의해주느냐가 복잡도 감소 SOVA의 효용성을 결정해 준다고 할 것이다.

그림12의 (c)에서 경쟁 경로가 존재하지않아 신뢰도를 정의할 수 없는 네 개의 노드의 원래 신뢰도 값, 즉 복잡도 감소 기법을 적용하지 않았을 때의 그림 12의 (c)에 나타나있는 신뢰도 값을 살펴보자. 신뢰도 값을 정의할 수 없는 노드에 대한 원래의 신뢰도 4개의 값은 총 8개의 신뢰도 값 중 가장 높은 값임을 알 수 있다. 그 이유는 간단하다. 왜냐하면 복잡도 감소 기법을 사용하여 제거된 경로들은 그 경로 미터릭이 가장 낮은 값들이었을 것이므로 신뢰도 값을 정의할 수 없는 노드로 들어올 예정이었으나 복잡도 감소 기법에 의해 제거된 경로와 현재 존재하는 경로와의 미터릭 차이 값은 비교적 클 것이기 때문이다. 따라서 이러한 부분에 대하여 신뢰도를 정의하는 몇가지 방법을 생각할 수 있다.

첫째는 이러한 신뢰도 정의 불능 노드에 대해서는 그 시점에서의 신뢰도 값 중 최대 값을 할당할 수 있을 것이다. 그림 12의 예제에서는 신뢰도 정의 불가능 노드로 할당될 신뢰도 값이 남아있는 신뢰도 중 최대 값, 즉 5가 될 것이다. 그러나 이러한 방법은 현 시점에서의 신뢰도 값들에 대하여 최대 값을 찾아내는 작업이 요구되기 때문에 알고리즘이 복잡도가 증가할 위험이 있다.

두번째 방법으로는 복호 진행 과정중 그 값을 임의의 최대치(∞)로 두고 복호가 진행되면서 원래의 SOVA에서처럼 갱신 작업을 계속한다. 이러한 과정은 일반적인 SOVA에서도 수행되는 과정이다. 신뢰도 갱신 작업 때 이전 시점에서의 신뢰도 값은 현재 시점과 비교하여 계속 최소 값으로 갱신되기 때문에 초기 신뢰도 값은 항상 최대치(∞)로 정해주는 것이 보통이다. 그러나 복잡도 감소 기법을 적용하면 신뢰도 정의 불가능 노드가 존재하기 때문에 이러한 노드에 할당된 최대치(∞)는 갱신되지 않고 그대로 남아있을 수 있다. 복호가 완료된 후에도 임의의 최대치 값으로 남아있으면 이 최대치를 제외한

그 경로에서의 최대 값으로 대치한다.

세번째 방식은 신뢰도 정의 불가능 노드로 유입될 예정이었던 경로의 미터릭 값이 적어도 복잡도 감소 기법에서 경로를 제거하기 위하여 사용되는 기준 경로 미터릭, pm_r, 보다는 클 것이라는 것을 이용하여 신뢰도 값을 유입 경로의 미터릭과 기준 경로 미터릭의 차이값으로 정의하는 것이다. 두 번째 방법에 비교해 볼 때 현 시점에서의 신뢰도를 더욱 잘 반영해 줄 수 있는 방법이 될 것이다.

즉, 그림 12의 예에서 세번째 방식으로 신뢰도 정의 불가능 노드에 대한 신뢰도를 정의해보면 다음과 같다. 시점 4에서 8개의 경로중 6개만을 선택하기 위해서는 기준 경로 미터릭, pm_r=13이 될 것이다. 따라서 신뢰도 정의 불능 노드 네 개에 대한 신뢰도 값을 복잡도 감소 기법을 적용하지 않았을 때의 신뢰도 값과 비교해보면 다음과 같다.

표 1. 복잡도 감소 기법을 적용하였을 때와 적용하지 않았을 경우 신뢰도 값의 비교

신뢰도 값 / state number	복잡도 감소 기법 적용 R*	R
1	13-8=5	10
3	13-10=3	10
5	13-7=6	10
7	13-11=2	10

표 1에 나타난 결과에 따르면 세 번째 방식을 사용하면 복잡도 감소 기법을 사용하였을 때보다 신뢰도 값이 더 적게 정의된다는 것을 알 수 있다. MAP 알고리즘과 비교하여 SOVA의 단점이 신뢰도 값을 지나치게 낙관적으로 평가한다는 것을 감안하면 세 번째 방식은 이러한 점을 어느 정도 보완한다는 점에서 볼 때 원래의 신뢰도 보다 적은 값으로 계산된다는 것은 바람직하다고 평가할 수 있을 것이다.

이러한 사실은 SOVA의 성능을 보완하기 위하여 사용하는 여러 가지 기법들이 지나치게 낙관적으로 평가된 신뢰도 정보를 좀 더 적은값으로 보정하는 기법들이라는데도 알 수 있다[10,13]. SOVA다른 관점에서 바라보면, 복잡도 감소 기법에서 신뢰도 정의 불가능 노드에 정의되는 신뢰도 값은 복잡도 감소 기법을 적용하지 않았을 때보다는 부정확한 값일 것이므로 더 적은 값으로 할당되는 것이 바람직할 것이다. 즉, 이 방식은 신뢰도 정의 불가능 노드가 가질 수 있는 신뢰도 값 중 최소값을 할당하는 방식이라고 할 수 있다.

2. 복잡도 감소 기법을 적용한 시뮬레이션 결과

그림 13에는 (31,25) expurgated BCH 부호를 사용하여 5.1에서 제시한 세가지의 복잡도 감소 기법에 대한 성능을 시뮬레이션한 결과가 나타나 있다.

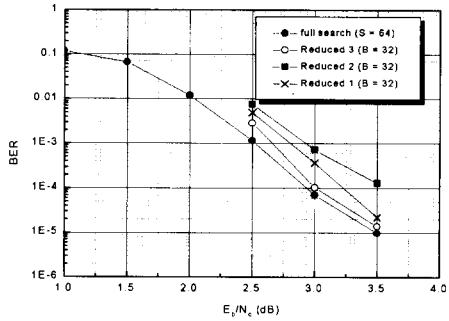


그림 13. (31,25) expurgated 부호를 내부 구성 부호로 사용한 병렬 연접 블록 터보 부호에 대하여 복잡도 감소 기법을 적용한 복호기의 6회 반복 후의 성능

그림 13에서 사용된 부호의 전체 최대 경로 수 S=64이며, 복잡도 감소 기법에서는 복호기가 항상 최대로 유지하는 경로수인 B를 32로 하였다. 시뮬레이션 결과를 살펴보면 그림 13에 Reduced 1으로 나타나 있는 5.1절의 첫번째 방식은 세번째 방식(Reduced 3)에 비하여 복잡도가 크면서도 세번째 방식에 비하여 성능저하가 더 큼을 알수 있다. 또한 Reduced 2로 표시되어 있는 두번째 방식은 전 경로 탐색 방법에 비하여 성능 저하가 매우 심각함을 알 수 있다. 그러나, 세번째 방식의 (Reduced 3) 경우 전경로 탐색 방법과 거의 근접하는 성능을 얻을 수 있음을 알 수 있다.

상기의 시뮬레이션 결과를 바탕으로 하여 그림 14에는 (31,21) BCH 부호를 병렬로 연접한 블록 터보 부호에 대하여 상기에서 세번째로 언급한 복

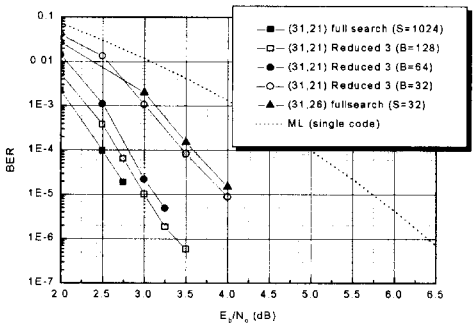


그림 14. 병렬 연접 블록 터보 부호에 대하여 복잡도 감소 기법을 적용한 복호기의 6회 반복 후의 성능

잡도 감소 SOVA를 적용하였을 때의 성능이 나타나 있다. (31,21) BCH 부호의 트렐리스에는 총 1024개의 경로가 존재하는데, 그림 14에 나타난 결과에 따르면 총 128개 즉, 약 1/10의 경로만을 유지하고서도 전체 1024개의 경로를 탐색했을 때의 성능과 거의 유사한 성능을 얻을 수 있음을 볼 수 있다.

Ⅵ. 결 론

본 논문에서는 트렐리스 복호 방식 특히 변형된 SOVA를 이용한 블록 터보 부호의 성능에 대하여 살펴보았다. 우선 SOVA의 성능을 개선시키기 위한 몇가지 알고리즘을 적용하여 효율적으로 반복 복호를 수행할 수 있도록 하였다. 또한 병렬로 연접된 블록 터보 부호의 이론적인 성능 바운드와 실제 시뮬레이션 결과를 비교해 보았으며, 내부 구성 부호의 연결 형태와 변형에 따른 성능의 변화를 살펴보았다. 특히 내부 구성 부호의 오류정정 능력이 증가됨에따라 트렐리스의 복잡도가 증가하게 되므로, SOVA에서 트렐리스의 복잡도를 감소시킬 때 발생하는 문제를 효율적으로 해결함으로써 복잡도를 줄일 수 있는 알고리즘을 제안하였다. 시뮬레이션 결과에서 살펴본 특성을 보면 제안된 알고리즘은 약 1/10의 복잡도만을 가지고서도 전체 트렐리스를 탐색했을 때와 비교하여 0.1dB 이하의 성능 열화만을 보인다는 것을 확인할 수 있었다.

참 고 문 헌

[1] C. Berrou, A. Glavieux, and P. Thitimajshma, "Near Shannon limit error-correction coding and decoding : Turbo-codes", *Proceedings of International Conference on Communications*, pp. 1064-1070, Geneva, Switzerland, May 1993.

[2] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes", *Proceedings of the IEEE Global Communications Conference GLOBECOM 1994*, pp. 339-343, San Francisco, U. S. A., Nov. 1994

[3] http://www. eccincorp.com/

[4] http://www.aha.com/

[5] D. Chase, "A class of Algorithm for Decoding Block Codes with Channel Measurement Information", *IEEE Transactions on Information Theory*, 18(1), pp.170-182, Jan. 1972

[6] Wolf, J. K., "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis", *IEEE Transactions on Information Theory*, 24(1), pp.76-80, 1978

[7] J. Hagenauger and P. Höher, "A Viterbi Algorithm with Soft Outputs and Its Application", *Proceedings of the IEEE Global Communications Conference GLOBECOM 1989*, pp. 47.1.1-47.1.7, Nov. 1989

[8] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory*, 42, pp. 409-428, Mar. 1996.

[9] 백동철, "MAP 복호 알고리즘을 이용한 블록부호 및 연접부호의 성능 분석", *한양대학교 석사 학위 논문*, 1998년 12월

[10] L. Papke and P. Robertson, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme", *Proceedings of ICC*, pp. 102-106, Dallas, U.S.A., June 1996.

[11] Fu-hua Huang, "Evaluation of Soft Output Decoding for Turbo Codes", *MS Thesis, Virginia Polytechnic Institute and State University*, May 1997

[12] M. C. Fossorier, F. Burkert, S. Lin and J. Hagenauer, "On the Equivalence between SOVA and Max-Log-MAP Decodings", *IEEE Communications Letters*, 2(5), pp. 137-139, May 1998.

[13] L. Lin and R. S. Cheng, "Improvements in SOVA-based decoding for turbo codes", *Proceedings of ICC*, pp. 1473-1478, Montreal, Canada, June 1997

[14] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenium Press, New York, 1981

[15] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes : Some Resultson Parallel Concatenated Coding Schemes", *IEEE Transactions on Information Theory*, 42(2), pp.

409-428, Mar. 1996

[16] S. K. Shin and P. Sweeney, "Soft Decision Decoding of Reed Solomon Codes Using Trellis Methods", *IEE Proceedings- Communications*, 141(5), pp. 303-308, Oct. 1994

[17] S. K. Shin and S. I. Lee, "Efficient complexity reduction technique in trellis decoding algorithm", *Electronics Letters*, 35(1), pp. 16-17, January 1999

김 수 영(Sooyoung Km Shin) 정회원

1990년 2월 : 한국과학기술원 전기 및 전자공학과 학사

1994년 2월~1991년 9월 : 한국전자통신 연구소 위성통신시스템연구부 연구원

1992년 10월 : Univ. of Suurey, U.K 공학 석사

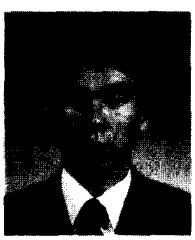
1995년 2월 : Univ. of Surrey, U.K 공학 박사

1994년 11월~1996년 6월 : Research Fellow, Univ. of Surrey, U.K

1996년 8월~현재 : 한국전자통신 연구원 위성통신 시스템연구 선임연구원

<주관심 분야> 오류정정 부호화 방식, 이동/위성통신 전송 방식 연구 등

이 수 인(Soo In Lee) 정회원



1985년 2월 : 경북대학교 전자공학과 공학사

1989년 2월 : 경북대학교 전자공학과 공학석사

1996년 2월 : 경북대학교 전자공학과 공학박사

1990년 2월~현재 : 한국전자통신연구원 무선방송기술연구소 팀장

<주관심 분야> 위성통신, 이동통신, 채널코딩