동적 BP 기법을 이용한 LT 부호의 복호

정호영*

요약

LT(Luby transform) 부호는 인터넷 망과 같이 erasure 채널을 통해 데이터를 전달하는데 효율적인 부호로 알려져 왔다. LT 부호의 복호 방식 중 BP(belief propagation) 알고리즘은 복호 속도 면에서 가장 빠른 방식으로 인정되고 있지만 k 값이 작을 경우 오버헤드 비중이 너무 큰 단점이 있다. 본 논문에서는 오버헤드 비중이 OFG(on the fly Gaussian elimination) 알고리즘과 같이 매우 낮으면서도 복호 복잡도는 OFG 알고리즘에 비해 크게 감소된 OFB(on the fly belief algorithm) 복호 방식을 제안하였다. 특히 OFB 알고리즘은 OFG 알고리즘과 같이 매 패킷이 도착할 때마다 복호 과정을 수행함으로서 복호 연산이 전 패킷으로 고루 분산되는 효과를 갖는다.

LT Decoding using On the Fly Belief Propagation

Ho-Young Cheong*

ABSTRACT

LT codes provide an efficient way to transmit data stream over erasure channels, like Internet. Belief Propagation(BP) algorithm is a fast decoding algorithm for LT codes but for small k it requires a large overhead to decode. In this paper, an efficient decoding algorithm for short k, called On the Fly Belief Propagation(OFB), is proposed. It provides a low decoding complexity while guaranteeing the same overhead with OFG. OFB performs useful processing at each coded packet arrival, so distributing the decoding operations during all packets receptions.

Key Words : Belief Propagation, Gaussian Elimination, Luby Transform(LT) code, Swap Heurisitic, Robust Soliton Distribution

---

* 남서울대학교 정보통신공학과
· 제1저자(First Author) : 정호영 · 교신저자(Correspondent Author) : 정호영
· 접수일(2009년 3월 8일), 수정일(1차 : 2010년 4월 10일), 게재확정일(2010년 4월 14일)

Ⅰ. 서 론

인터넷 망을 통해 패킷을 전송하는 경우와 같이 각각의 패킷이 오류 없이 수신되거나 혹은 전혀 수신되지 않는 erasure 채널이 점차 중요해져 가고 있다. 최근 erasure 채널을 통해 오류 없이 데이터를 전송하기 위한 FEC 부호로서 LT 부호[1]와 Raptor 부호[2]가 많은 관심을 받고 있다. LT 부호와 Raptor 부호 모두 디지털 fountain 부호에 속하는데, 예를 들어 전송해야 할 메시지 패킷이 k 개 있다면 LT 부호에서는 이들을 이용해 끊임없이 부호 패킷들을 발생시켜 전송하게 되며 수신 단에서는 패킷의 전송 순서에 상관없이 원래의 메시지 패킷이 완전히 복원될 때까지 부호화된 패킷을 랜덤하게 수신하여 복호하게 된다. 따라서 이들의 부호 율은 일정하지 않게 되므로 부호 율(code rate)이 정해지지 않는 무 부호 율 부호(rateless codes)라고 부르는 한편, erasure 채널에 적용될 경우 최적의 erasure 복원 성능을 나타내는 것으로 알려져 있다[1]. 이들은 마치 끊임없이 물방울을 뿜어내는 샘(fountain) 주위에 컵을 놓아 랜덤하게 떨어지는 물방울로 컵을 가득 채우는 모양과 유사하다고 하여 디지털 fountain 부호라고 부른다.

LT 부호의 복호는 BP(belief propagation) 알고리즘이 대표적인데 다른 복호 방식에 비해 복호 복잡도가 가장 낮은 것으로 알려져 있다. 그러나 전송하고자 하는 메시지 블록의 수 k가 크지 않을 경우 BP 보다 GE(Gaussian elimination) 알고리즘을 이용하는 것이 더 유리할 수 있는데 BP의 경우 k가 작을 경우 오버 헤드(overhead)가 지나치게 증가하기 때문이다. 반면 GE 복호 방식은 복호 속도가 BP에 비해 느리기는 하지만 오버헤드 비중이 훨씬 작다. BP와 GE 복호 방식은 두 방식 모두 패킷의 전송 과정 끝 부분에서 대부분의 복호 연산이 수행되므로 복호 처리 연산이 마지막 패킷들이 도착하는 시간 이후에 집중되는 특성을 갖는다.

[3]에서 제안한 OFG(on the fly Gaussian elimination) 복호 방식은 GE의 복호 연산과정을 패킷들이 도착하는 모든 시간대로 분산시켜 복호를 수행함으로서 마지막 패킷이 도착하는 시간대에 복호를 종료할 수 있다. GE 복호 방식에 기반을 두고 있기 때문에 BP에 비해 오버 헤드 비중이 훨씬 작으며 복호 복잡도 또한 GE에 비해 작은 특징을 갖는다. OFG 복호 방식에서는 복호 연산 양을 줄이기 위해 swap heuristic 과정을 두고 있는데 이는 삼각화가 진행되는 동안 행렬의 행 차수(row degree)를 줄여나가는 과정이라고 할 수 있으며 이를 통해 상측 삼각 행렬(upper triangle)에서 1의 밀도를 작게 하여 연산 양을 줄이는 기법을 사용하고 있다.

본 논문에서는 오버헤드 비중이 GE 복호 방식과 거의 비슷하면서도 복호 연산 양을 크게 줄인 LT 부호의 복호 기법을 제안한다. OFG 복호 방식의 swap heuristic 과정이 삼각 행렬의 행 차수(row degree)를 줄이는 역할을 한다면 본 논문에서 제안한 복호 방식은 삼각 행렬의 열 차수(column degree)를 줄임으로서 복호 연산 양을 줄이는 복호 과정을 갖게 되며 복호 과정이 BP 알고리즘의 복호 과정과 유사하여 OFB(on the fly belief propagation) 알고리즘이라고 부르기로 한다. 제안된 복호기법은 IPTV와 같은 멀티캐스트 전송에서 발생되는 손실 패킷을 효율적으로 복구하기 사용되며 특히 최근에 많은 관심을 받고 있는 네트워크 코딩 기법에 응용할 수 있다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 LT 부호의 복호 방식 중 BP 와 OFG 복호 알고리즘을 살펴보고, 제 3 장에서는 본 논문에서 제안한 OFB(on the fly belief propagation) 복호 알고리즘을 자세히 기술한다. 제 4 장에서 이들에 대해 시뮬레이션을 통해 오버헤드 비중과 복호 연산 양을 중심으로 한 실험 결과를 비교·분석하고 제 5 장에서 결론을 맺는다.

II. LT 부호의 부호화와 복호화

2.1 LT 부호의 부호화(encoding)

총 N 비트인 전송 데이터 스트림을 길이 l의 패킷들로 분할하여 $k\ (=N/l)$ 개의 메시지 소스 패킷 $m=[m_1, m_2, \ldots, m_k]^T$를 만든다. 패킷 길이 l은 임의로 선택해도 이론적으로 부호 성능에 영향을 미치지 않지만, 실질적으로 l 값이 클수록 부호화에 사용된 패킷들의 정보를 담은 정보 벡터에 의한 오버 헤드 비중이 감소하게 되는 효과를 얻는다[2]. k 개의 메시지 패킷을 이용해 다음과 같은 부호화 과정을 통해 부호화된 패킷 $y=[y_1, y_2, \ldots]$를 발생하게 된다.

① 부호 패킷(encoded packet)의 차수 d 를 랜덤하게 발생시킨다. 이때 d 값은 RSD(robust soliton distribution)를 가져야 한다.

② k 개의 메시지 패킷 중에서 임의로 d 개를 선택한 후 xor 연산을 수행하여 부호화된 패킷 y_i를 식 (1)과 같이 얻는다.

$$y_i = m_{r_1} \oplus m_{r_2} \oplus \ldots \oplus m_{r_d} \tag{1}$$

여기에서 m_{r_j}는 xor 연산에 사용된 r_j 번째 소스 패킷을 의미한다.

③ y_i의 xor 연산에 사용된 소스 패킷 정보를 담고 있는 정보 벡터 $b_i=[b_{i1}, b_{i2}, \ldots, b_{ik}]$와 함께 y_i를 전송하게 된다. 여기에서 m_j가 y_i를 생성하기 위해 xor 연산에 사용되었을 경우 $b_{ij}=1$ 값을 갖고 사용되지 않았을 경우 $b_{ij}=0$의 값을 갖게 된다.

수신단의 복호기에 n 개의 부호화된 패킷 $y=[y_1, y_2, \ldots, y_n]^T$과 해당 정보 벡터 $B=[b_1, b_2, \ldots, b_n]^T$ 가 수신되면 식 (2)의 방정식을 풀어 소스 패킷 m을 복호하게 된다.

$$Bm = y \tag{2}$$

여기에서 잉여 등식(linearly dependent equation)을 제거하면 B와 y로부터 각 각 $(k\times k)$ 행렬 G 와 $(k\times 1)$ 벡터 Y를 얻을 수 있고 식 (3)의 방정식을 얻을 수 있으며 이를 풀어 m을 복호하게 된다.

$$Gm = Y \tag{3}$$

기본적으로 식 (3)의 해는 GE(Gaussian elimination)을 통해 구할 수 있으나 GE를 이용할 경우 k 값에 따라 연산양이 급격히 증가하여 k 값이 클 경우 적용하기 어렵다.

LT 부호는 벡터 b_i의 차수가 RSD 특성을 갖도록 적절히 조절하여 수신 단에서 BP 복호 알고리즘이 적용될 수 있도록 한 부호이다. BP 복호 알고리즘이 적용될 경우 k 값에 대한 복호 연산양은 $O(k\log k)$인 것으로 밝혀져 있다[3].

2.2 BP(belief propagation) 복호

k 개의 부호화된 패킷들을 수신한 후 함께 전송되어 온 부호화 정보 벡터 B를 이용해 $Bm=y$를 구성할 수 있는데 이때 B는 $(k\times k)$ 행렬이며 행의 차수는 RSD(robust soliton distribution) 특성을 갖게 된다.

B 행렬의 i 번째 행을 $B[i]$라고 표시하기로 하자. 차수(degree)가 1인 행 $B[i]$에서 '1'의 열(column) 위치가 j라고 하면 $m_j=y_i$로 복호하고 j 번째 열에 있는 모든 '1'을 '0'으로 바꿈과 동시에 '0'으로 바뀌는 행에 해당하는 부호 패킷 y_l을 y_i와 xor 연산을 하게 된다.

다시 차수가 1인 행을 찾아 위와 같은 복호 과정을

B 행렬의 원소가 모두 0이 될 때까지 반복하는데 B 행렬의 원소가 모두 0이 되면 복호가 성공한 것이고 모두 0이 되지 않았는데도 불구하고 차수가 1인 행이 없으면 복호 과정이 중단되어 실패하게 된다.

차수가 1인 행이 없어 복호 과정이 중단되면 새로운 패킷을 수신한 후 복호된 위치의 '1'을 모두 '0'으로 변환하는 과정을 먼저 수행하고 B 행렬에 추가하여 행렬의 모든 원소가 '0'이 될 때까지 복호과정을 계속한다.

2.3 OFG 복호

GE 과정에 기반을 두고 있지만 OFG 복호 알고리즘은 k 개의 부호 패킷이 도착할 때까지 기다리지 않고, 처음 도착하는 패킷부터 시작하여 매 번 패킷들이 도착할 때 마다 이들을 이용하여 행렬 G 의 삼각 화(triangularization)를 진행한다.

또한 OFG에서는 삼각 행렬에서 '1'의 밀도가 작아지도록 하는 swap heuristic 과정을 수행하는데 '1'의 밀도가 작아지면 복호화 과정 중에 발생하는 복호 연산양을 크게 줄일 수 있다[3].

G 가 부분적으로 삼각 형태를 갖는 $(k \times k)$ 행렬이라고 가정하자. 즉 임의의 행에서 가장 왼쪽의 '1'의 위치가 대각(diagonal)에 있거나 행의 모든 원소가 0인 형태를 의미한다. y_i와 b_i를 수신하자마자 b_i에서 가장 왼쪽에 있는 '1'의 위치를 찾는다. 이때 그 위치를 s_i라고 할 경우 $G[s_i]$가 모두 0인 행이면 $G[s_i]$를 b_i로 대치하고 그렇지 않으면 $G[s_i]$와 b_i 및 y_i와 y_{s_i}에 대해 각 각 xor 연산을 취하여 새로운 식 $b_i' = b_i \oplus G[s_i]$와 $y_i' = y_i \oplus y_{s_i}$를 얻는다. 이때 b_i'에 대한 s_i'는 $s_i' > s_i$인 값을 갖게 되며 b_i'이 행렬 G 에 삽입되든지 b_i'의 모든 원소가 0이 될 때까지 위의 과정을 반복한다. 행렬 G 의 '1'의 밀도가 작을수록 위에서 기술한 복호 과정의 반복 횟수가 감소하는데[1] '1'의 밀도를 감소시키기 위해 다음과 같은 swap heurisitic 과정을 둔다.

s_i가 대각 위치일 경우 행 $G[s_i]$의 차수와 b_i의 차수를 비교하여 $\deg(b_i) < \deg(G[s_i])$이면 해당 부호 패킷과 함께 $G[s_i]$와 b_i를 서로 교환한 후 복호 과정을 계속한다.

OFG 복호 기법의 가장 큰 특징은 복호 연산을 k 개의 패킷들이 모두 도착할 때까지 기다리지 않고 첫 번째 수신 패킷부터 복호 과정을 시작하여 마지막 패킷이 도착하면 즉시 복호화 과정을 종료할 수 있다는 것이다.

즉 복호화 연산이 모든 수신 패킷 도착 시간에 골고루 분포되어 전체적인 복호 시간을 단축할 수 있으며 GE에 기반을 두고 있어 오버헤드가 적다는 장점을 갖는다.

III. OFG 복호

OFG 복호 과정에서는 행렬 G 에 대한 '1' 의 밀도를 작게 하기 위해 swap heuristic 과정을 사용하고 있다. 수신된 패킷의 정보 벡터 b_i가 좌측으로부터 s_i 열에 처음 '1'을 가질 때 b_i와 $G[s_i]$의 차수를 비교한 후 작은 차수를 갖는 벡터가 행렬 G 에 위치하도록 함으로서 '1'의 밀도를 줄여가고 있는 것이다. 그러나 이러한 방법은 삼각 화 과정이 어느 정도 진행되고 나면 더 이상 밀도가 감소하지 않는다. 예를 들어 삼각 화 후반부에는 G 행들의 차수가 2~3의 값을 가지게 되며 차수가 1인 패킷이 도착하지 않는 한 밀도가 감소하는 일은 거의 없다고 보아야 한다. 특히 삼각 화 과정이 완성되어 갈수록 대각 위치가 비어 있는 행을 채우기 위해 많은 복호 과정이 반복되며 상측 삼각 행렬에 '1'의 밀도가 클수록 반복 횟수는 더욱 더 증가하게 된다. 복호 연산 횟수가 삼각 화 후반부로 갈수록 커진다는 점을 감안할 때 '1'의 밀도 감소는 연산 횟수를 줄이는 데

큰 영향을 준다는 것을 실험적으로 알 수 있다. OFG 복호 방식의 swap heuristic 과정은 G에 대해 행(row) 방향으로 작용하여 '1'의 밀도를 줄이는 기법으로 볼 수 있다.

본 논문에서 제시한 OFB 알고리즘은 '1'의 밀도를 줄이기 위한 복호 과정이 열(column) 방향으로 작용한다. 이는 BP 알고리즘에서 열에 있는 '1'을 제거하는 과정과 유사하다. BP 알고리즘에서는 차수가 '1'인 패킷이 수신될 경우 이를 리플(ripple)에 저장했다가 k개의 패킷이 모두 수신되고 나면 복호과정이 시작되는데, 리플에 저장되어 있던 차수 1의 패킷에 대한 복호가 시작되면 복호되는 패킷의 '1' 위치에 해당하는 열에 대해 행 연산(row operation)을 통해 해당 열(column)의 모든 '1'을 제거하게 된다. BP와 달리 OFB에서는 k 개의 패킷이 모두 수신될 때까지 기다리지 않고 처음 패킷부터 복호화 과정을 시작하게 된다. 그림 1은 OFB 복호 알고리즘의 삼각 화 과정을 예로 든 것이다.

그림 1 (a)에서 네 번째 열에 대해 네 번째 행을 가지고 행 연산을 하여 (b)와 같이 네 번 째 열의 '1'을 모두 제거할 수 있다. 새롭게 수신된 패킷의 정보 벡터가 $b_1 = (01100)$이라고 하면 그림 1 (c)와 같이 두 번째 행을 채울 수 있고 다시 세 번 째 열에 대해 행 연산을 통해 그림 1 (d)와 같이 세 번 째 열의 '1'을 제거할 수 있다.

상측 삼각 행렬의 '1'의 밀도가 아직 감소되진 않았지만 빈 행의 대각 위치에 해당하는 열(다섯 번째 열)에 '1'이 집중되어 있음을 알 수 있다. 이제 다시 새로운 패킷 $b_2 = (11010)$이 수신되면 반복적인 복호 과정을 통해 새로운 패킷 $b_2' = (01011)$, $b_2'' = (00010)$, $b_2''' = (00001)$을 연속적으로 얻게 되고 그림 1 (f)의 행 연산을 통해 최종적으로 (g)를 완성하게 된다. 그림 1 (h)는 OFG 알고리즘을 적용하여 삼각 화를 완성했을 때의 결과를 나타낸 것으로 상측 삼각 행렬에 '1'이 완전히 제거되지 않아 삼각 화가 끝난 뒤에도 후방 대입(backward substitution) 기법을 적용해 복호해야 하며 k 값이 클 경우 후방 대입 연산양은 무시할 수 없다.

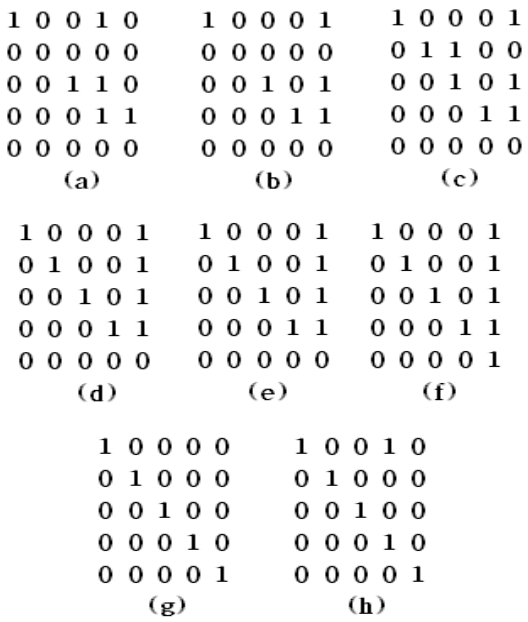

그림 1. OFB 삼각 화 과정의 예
Fig. 1. Example of OFB triangularization process

Ⅳ. 시뮬레이션 결과 및 분석

본 논문에서 제시한 OFB 알고리즘의 성능을 고찰하기 위해 다음과 같이 시뮬레이션 실험을 수행하였다. 우선 복호 알고리즘은 LT 부호의 전형적인 복호 방식인 BP 알고리즘과 GE(Gaussian elimination) 기법을 기반으로 하는 OFG 알고리즘 및 본 논문에서 제시한 OFB 알고리즘에 대해 각 성능 파라미터 별로 시뮬레이션 실험을 통해 결과를 도출하였다. 특별히 오버헤드 비중과 복호 복잡 도를 중심으로 실험하였으며 복호 복잡도의 경우 공정한 비교를 위해 OFG 알고리즘의 후방 대입(backward substitution) 연산 과정도 포함시켰다. 전송 패킷 수 k는 25~500까지의 값에 대

해 각 각 1,000 회 이상의 시뮬레이션을 통해 통계치를 얻었으며, δ 값은 0.01로 하고 리플의 크기에 영향을 주는 c 값은 0.1과 0.01에 대해 각 각의 경우를 실험하였다.

그림 2는 $\delta = 0.01$일 때 k 값에 따른 각 복호 알고리즘 별 오버헤드 비중을 나타낸 것이다. 총 수신된 패킷 수를 n이라고 할 때 오버헤드 비중은 $\epsilon = n/k - 1$으로 표현할 수 있다. BP 복호의 경우 리플의 크기를 결정하는 c 값에 따라 차이가 크며, 특히 $k < 100$의 경우 k 값이 작을수록 오버 헤드 비중은 급속히 증가함을 알 수 있다. 반면에 OFB와 OFG의 경우에는 $k < 100$은 경우 BP에 비해 오버헤드 비중이 크게 증가하지는 않는다.

또한 OFG와 OFB는 $k > 200$ 인 경우 오버 헤드 비중은 약 3% 내외로 거의 발생하지 않는다고 볼 수 있으며 이에 비해 BP 알고리즘의 오버헤드 비중은 $k = 500$일 경우에도 약 40.8%의 큰 비중을 나타낸다.

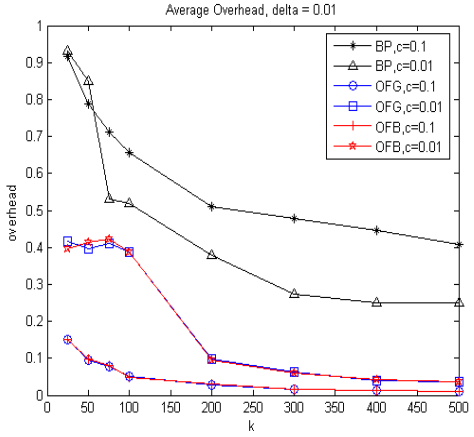

그림 2. k 에 따른 오버헤드 비중($\delta = 0.01$)
Fig. 2. Average overhead vs. k ($\delta = 0.01$)

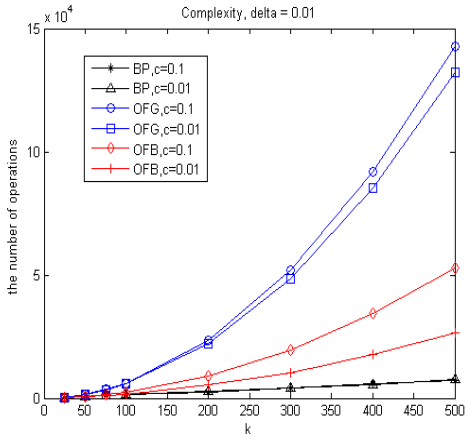

그림 3. k 에 따른 복호 복잡도 ($\delta = 0.01$)
Fig. 3. Decoding complexity vs. k ($\delta = 0.01$)

그림 3은 $\delta = 0.01$일 때 k 값에 따른 각 복호 알고리즘 별 복호 복잡도(complexity)를 나타낸 것이다. 여기에서 OFB와 BP 알고리즘은 삼각화가 완성되는 순간 복호 과정이 종료되는 반면 OFG 알고리즘은 삼각화 완료 후에 후방 대입 연산은 추가로 수행해야 복호가 종료되므로 OFG 알고리즘의 복잡 도를 계산할 때 후방 대입 과정도 포함시켰다. 그림 3에서 k 값이 작을 때에는 알고리즘 별 연산 량 차이가 크지 않지만 k 값이 증가할수록 알고리즘 별 연산 량 차이는 증가함을 알 수 있다. 특히 OFB 알고리즘의 경우 k 값에 관계없이 OFG 알고리즘의 경우와 오버헤드 비중은 같으나 복호 연산양이 크게 감소함을 알 수 있다. $k = 500$, $c = 0.1$일 경우 OFB는 OFG 알고리즘에 비해 약 63% 의 연산 량 감소를 보이고 있으며, $k = 500$, $c = 0.01$일 경우에는 BP의 복잡 도에 거의 근접하는 뛰어난 성능을 보인다.

V. 결 론

본 논문에서는 열 방향의 OFG 알고리즘을 이용하여 작은 오버헤드 비중을 가지면서도 OFG 알고리즘에 비해 복호 복잡도를 크게 감소시킨 OFB 알고리즘을 제안하였다. BP 알고리즘이 k 개의 패킷을 모두 수신한 후에 복호화 과정을 시작하는 반면 OFB 알고리즘은 매 패킷이 수신될 때 마다 동적으로(on the fly) 복호화 과정을 수행하여 마지막 패킷이 수신되자마자 복호화 과정을 종료하여 전체 복호 시간을 단축하는 효과를 갖는다.

k 값이 작을 경우 BP 알고리즘은 오버헤드 비중이 너무 큰 단점을 가지는 반면 OFG와 OFB 알고리즘의 경우 약 3% 이내의 작은 오버헤드 비중을 갖는다. 특히 본 논문에서 제안한 OFB 알고리즘의 경우 복호 복잡 도에 있어서 $k = 500,\ c = 0.1$일 경우 복호 연산양이 약 63% 감소함을 알 수 있었으며, $k = 500,\ c = 0.01$일 경우에는 오버헤드 비중은 OFG와 같으면서도 복잡 도는 BP에 근접하는 뛰어난 성능을 보였다. 이와 같은 특성은 k 값이 클 경우에도 유지될 것으로 보이며, 향 후 다양한 파라미터 값들에 따른 성능 특성 분석이 필요하다.

참고문헌

[1] M. Luby, "*LT codes*," Proceedings 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 271~280, Vancouver, Canada, Nov. 2002.

[2] A. Shokrollahi, "*Raptor codes*," IEEE Transaction on Information Theory, Vol. 52, No. 6, pp. 2551~2567, June 2006.

[3] Valerio Bilglio, Macro Grangetto, Rossano Gaeta, Matteo Sereno, "*On the fly Gaussian Elimination for LT codes*," IEEE Communication Letters, Vol. 13, No. 12, pp. 953~955, Dec. 2009.

[4] Saejoon Kim, Karam Ko, and Sae-Young Chung, "*Incremental Gaussian Elimination Decoding of Raptor Codes over BEC*," IEEE Communication Letters, Vol. 12, No. 14, pp. 307~309, April 2008.

[5] D. J. C. MacKay, "*Fountain Codes*," IEE Proceeding-Communications, Vol. 152, No. 6, pp. 1062~1068, Dec. 2005.

[6] 최재연, "디지털 무선 기지국용 저잡음 증폭기 설계 ", *한국지식정보기술학회 논문지*, 제4권 제3호, pp.13-18, 2009.

[7] 김석훈 외, "불완전도체 구조체 기반의 모서리 산란파 분석", *한국지식정보기술학회 논문지*, 제5권 제1호, pp.1-8, 2010.

감사의 글

본 논문은 2008학년도 남서울대학교 학술연구비 지원에 의해 연구되었음.

정호영(Ho-Young Cheong)

1989년 7월 : 연세대학교 본 대학원 전자공학과 (공학석사)
1994년 2월 : 연세대학교 본 대학원 전자공학과 (공학박사)

1995년 4월~현재: 남서울대학교 정보통신공학과 교수
2004년 3월 ~ 현재 : 남서울대학교 정보통신연구소 상임연구원

※ 관심분야 : 오류정정부호, Mobile IPTV 등