İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DÜŞÜK YOĞUNLUKLU EŞLİK DENETİM KODLARI İÇİN KODLAYICI VE KOD ÇÖZÜCÜ TASARIM TEKNİKLERİ

YÜKSEK LİSANS TEZİ

Müh. Tolga MATARACIOĞLU

Anabilim Dalı: ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

Programı: TELEKOMÜNİKASYON MÜHENDİSLİĞİ

HAZİRAN 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DÜŞÜK YOĞUNLUKLU EŞLİK DENETİM KODLARI İÇİN KODLAYICI VE KOD ÇÖZÜCÜ TASARIM TEKNİKLERİ

YÜKSEK LİSANS TEZİ
Müh. Tolga MATARACIOĞLU
504021303

Tezin Enstitüye Verildiği Tarih : 8 Mayıs 2006
Tezin Savunulduğu Tarih : 16 Haziran 2006

Tez Danışmanı : Prof. Dr. Ümit AYGÖLÜ (İ.T.Ü)

Diğer Jüri Üyeleri : Doç. Dr. İbrahim ALTUNBAŞ (İ.T.Ü)

Yrd. Doç. Dr. Oğuz KUCUR (G.Y.T.E)

Haziran 2006

ÖNSÖZ

Tez çalışmam boyunca bana yardımcı olan ve benden değerli yorumlarını esirgemeyen tez danışmanım Sayın Prof. Dr. Ümit Aygölü'ne ve TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü (UEKAE) Ağ Güvenliği Grubu Proje Yöneticisi Sayın Mert Üneri'ye, bu süre içerisinde manevi desteklerini hep yanımda hissettiğim annem Sayın Sevgi Mataracıoğlu, babam Sayın Oktay Mataracıoğlu ve nişanlım Sayın Özlem Yılmaz'a teşekkürü bir borç bilirim.

Ayrıca yüksek lisans öğrenimim boyunca tez çalışmam için maddi destek sağlamak amacıyla yurt içi yüksek lisans bursu veren TÜBİTAK Bilim Adamı Yetiştirme Grubu'na saygılarımı sunarım.

Kavaklıdere, Ankara — Tolga MATARACIOĞLU
Haziran 2006

İÇİNDEKİLER

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÖZET

DÜŞÜK YOĞUNLUKLU EŞLİK DENETİM KODLARI İÇİN KODLAYICI VE KOD ÇÖZÜCÜ TASARIM TEKNİKLERİ

Tez çalışmasında düşük yoğunluklu eşlik denetim (LDPC) kodları ele alınmıştır. Hem düzenli hem de düzensiz LDPC kodlar için tasarım teknikleri incelenmiş, yeni üstün kodlar sunulmuştur. Bu çalışmada literatürde yer almış kodlama oranlarının yanısıra, farklı kodlama oranları için de ikili (BPSK) AWGN kanalda iyi hata başarımı sağlayan LDPC kodlar tasarlanmıştır. Tasarlanan yeni düzensiz LDPC kodların, sadece kod sözcük uzunluğu sonsuza giderken hesaplanan eşik değeriyle değil, aynı zamanda sonlu kod sözcük uzunluklarıyla kanal sığasına çok yaklaştığı ve kod sözcük uzunluğu $N = 1000$ ve kodlama oranı $R = 0.5$ için daha iyi bit hata başarımı gösterdiği görülmüştür. Karşılaştırma yapmak gerekirse düzensiz kodlar düzenli kodlara göre daha iyi bit hata başarımı sunmaktadır. Ayrıca kodlama oranı arttığı zaman başarım da düşmektedir. Düzensiz kodlar için de bit düğümlerinin maksimum değeri ne kadar büyükse, ilgili kod o kadar kanal sığasına yaklaşmaktadır. Bu çalışmada optimizasyon, verici ve alıcı tekniği olarak farksal evrim tekniği, bit doldurma algoritması ve mesaj aktarma algoritması teknikleri kullanılmıştır. Çalışmada tasarlanan en üstün LDPC kod, $R = 0.5$ kodlama oranında elde edilmiş olup kanal sığasından sadece 0.02 dB daha uzakta eşik değerine sahip olan bir düzensiz LDPC kod olmaktadır.

SUMMARY

ENCODER AND DECODER DESIGN TECHNIQUES FOR LOW DENSITY PARITY CHECK CODES

In this dissertation, low density parity check (LDPC) codes have been considered. Design techniques have been examined and new LDPC codes with improved error performance have been suggested for both regular and irregular LDPC codes. Also for BPSK AWGN channel, superior LDPC codes have been successfully designed for $R = 0.5$ and $N = 1000$. For the designed superior irregular codes, it can be said that not only the performance of the codes is good when the code block length goes to infinity, but also they approach to the channel capacity and for code rate $R = 0.5$, they perform better when the code block length is finite ($N = 1000$). So as to compare, it can be said that irregular codes perform better than regular codes. Further, when the coding rate increases, the performance of the code starts decreasing. For irregular codes, when the degree of the bit node that possess the maximum number of branches is large, then that code approaches to the channel capacity. The combination of differential evolution technique, bit filling algorithm and message passing algorithm techniques as optimization, transmitter and receiver blocks for a communications system has been used. The best superior code designed in this dissertation is an irregular LDPC code with coding rate $R = 0.5$ and is only 0.02 dB away from the channel capacity.

1. GİRİŞ

Bu bölümde konuyla ilgili literatürde yapılmış çalışmalarla tezin konuya katkıları anlatılmaktadır.

1.1 Konuyla İlgili Literatürde Yapılmış Çalışmalar

Düşük yoğunluklu eşlik denetim (low density parity check - LDPC) kodları, diğer adıyla Gallager kodları, ilk olarak 1962'de Gallager tarafından bulunmuştur [1][2]. Bu kodlar, iteratif kod çözme yapılarının karmaşıklığının blok uzunluğuyla *doğrusal* olarak artmasından dolayı çok iyi kodlar olarak nitelendirilmektedir. Klasik kodlarda ise blok uzunluğuyla karmaşıklık arasındaki ilişki üstel olarak doğru orantılıdır. LDPC kodlar çok hızlı kodlama / kod çözme algoritmalarıyla donanmışlardır. Yoğunluk terimi, eşlik denetim matrisindeki sıfır harici elemanların sayısının tüm matris elemanlarının sayısına oranı olarak tanımlanır. LDPC kodlarının da düşük yoğunluklu olması özelliği, eşlik denetim (H) matrisindeki sıfır dışındaki elemanların sayısının tüm eleman sayısına oranının çok düşük olması anlamına gelmektedir. Bu çalışmada ikili diziler kullanılacağından dolayı yoğunluk tanımı, eşlik denetim matrisindeki birlerin sayısının tüm bitlerin sayısına oranı olarak yapılabilir.

Çözülmesi gereken problem ise, bu algoritmaların düşük işaret - gürültü oranlarında orijinal kod sözcüğünü onarabileceği şekilde kodların tasarlanmasıdır. LDPC kodlar hem daha iyi başarım hem de daha düşük kod çözme karmaşıklığı sunarlar. Kod sözcük uzunluğu 10 milyon olan düzensiz bir LDPC kodu, Shannon sınırından yalnızca 0.04 dB daha düşük başarım göstermektedir.

Ayrıca LDPC kodlarda ilgili eşlik denetim matrisindeki 1'lerin konumları rastgele olduğundan, serpiştirici (interleaver) fonksiyonu kendi içinde gerçekleşmektedir. Bu özellik LDPC kodları, rakipleri karşısında üstün kılmaktadır.

Kod sözcük uzunluğu sonsuza giderken en büyük olabilirlikli (ML) kod çözücünün hata olasılığını sıfıra götüren kodların varlığını Shannon kanıtlamıştır [3]. Ama bu türden kodların nasıl bulunacağı hakkında bir şey söylememiştir. Bu kod dizileri elde olsa bile, bu dizileri verimli olarak kodlamak / çözmek konusu berrak değildir. ML kod çözücü, alınan sözcüğe en yakın Hamming uzaklıklı kod sözcüğünü bulmaktadır. Shannon, sığa (kapasite) diye adlandırılan bir sayının var olduğunu ve bu sığaya yakın hızlarda güvenli iletimin mümkün olduğunu, sığanın yukarısındaki hızlarda güvenli iletimin mümkün olmadığını ispatlamıştır. Şekil 1.1'de bir iletişim sistemine ait blok diyagramı verilmiştir:

Şekil 1.1: İletişim Sistemi Blok Diyagramı

LDPC kodların en büyük rakibi olan turbo kodlara göre önemli birkaç avantajı bulunmaktadır [17]. Bunlardan birincisi, LDPC kodlar için mesaj aktarma algoritması kullanılarak kod çözme çok basit olmakla beraber kod çözme işi çok yüksek hızlarda gerçeklenebilmektedir. İkinci önemli avantajı ise, çok düşük karmaşıklığa sahip kod çözücülerin ancak LDPC kodlar için tasarlanabilmeleridir. Üçüncü avantajı da daha iyi hata tabanlarına (error floor) sahip olmalarıdır. Pratikte ise LDPC kodlara ilişkin tek bir itiraz vardır ki bu da, kodların inşa edilmesi esnasındaki kodlama işleminin karmaşıklığının yüksek olmasıdır.

LDPC kodların can alıcı özelliklerinin çok önceden biliniyor olmasına rağmen, o zamandaki bilgisayarların yeterince güçlü olmamalarından dolayı bu kodlar uzun bir süre rafa kaldırılmıştır. Bu kodların tekrar ele alınmaya başlaması 1995 yılında MacKay ve Neal [4] sayesinde olmuştur. Son zamanlarda getirilen tasarım iyileştirmeleri, LDPC kodların donanım gerçeklemesi ile bu kodların sunduğu basitlik, bir yandan düşük karmaşıklık getirirken bir yandan da turbo kodlardan daha iyi başarım sağlamaktadır [5].

Mesaj aktarma algoritmasıyla (message passing algorithm) çözülen LDPC kodlar şaşırtıcı bir eşik etkisi göstermiştir [6]. Richardson [6] tarafından öne

sürülen ve yoğunluk evrimi (density evolution) diye adlandırılan sayısal bir teknikle, çeşitli kanallarda mesaj aktarma algoritmasının başarımını incelemek mümkün olmaktadır.

Daha önce ele alınan düzenli (regular) LDPC kodların yanısıra Luby, Mitzenmacher, Shokrollahi ve Spielman [7], düzensiz LDPC kodları incelemişler ve bu tür kodlar kullanmanın yanında kod parametrelerini de optimize ederek sığaya yakın başarım gösteren kodlar bulmayı başarmışlardır. Aynı zamanda çok büyük kod sözcük uzunluğuna sahip kodlar için kod çözme analizi sonucu belirlenen bir gürültü eşiğinin altında düşük bit hata olasılığının elde edildiğini göstermişlerdir. Bu sonuçları kullanarak, büyük blok uzunluklarıyla, turbo kodlardan başarım olarak daha iyi olan LDPC kodlar tasarlamayı başarmışlardır. Bu çalışmayla ulaşılan başarım, Chung, Forney, Richardson ve Urbanke [8] tarafından daha da geliştirilmiş ve AWGN (Additive White Gaussian Noise) kanal için hesaplanmış sığanın 0.0045 dB yanına yaklaşmak mümkün olmuştur.

Prabhakar ve Narayanan [9] donanımın gerçekleştirilmesinde, graf bağlantılarının bellekte tutulması yerine kodun graf yapısının özyineleme (recursion) kullanarak üretilmesi sayesinde çok basit hale gelen kod çözücüler üretilebileceğini göstermişlerdir.

Kod optimizasyonu ile ilgili olarak da Hou, Siegel ve Milstein [10], kanaldan bağımsız olarak düzensiz LDPC kodların hangi yöntemle optimize edildiğinde kanal sığasına yakın kodların elde edilebileceğini göstermişlerdir.

Verici tasarımıyla ilgili olarak Campello ve Modha [11][12], gerçekleştirilmesi kolay, yüksek hızlı ve yüksek çevre ölçülü (girth) düzensiz LDPC kodlar tasarlamayı başarmışlardır.

Gerçekleştirme alanları olarak ele alındığında ise, turbo kodların pek çok uygulamada çoktan yerini aldığı ve üçüncü nesil (3G) kablosuz standartlarının bir parçası olduğu görülmektedir. LDPC kodlar turbo kodlara göre ayrıntılı bir biçimde daha sonradan ele alındığı için şu anki standartlarda yer alma şansını kaybetmiştir. Buna rağmen son DVB uydu iletişim standardı için LDPC tabanlı bir çözüm önerilmiştir. 3G sistemlerinin gelişmesi sürecindeki gecikme, LDPC tabanlı kodlama sistemlerinin sunumuna katkı sağlayacaktır.

Flarion Technologies programlanabilir bir LDPC kod çözücü gerçekleştirmiştir. Bu donanım pek çok LDPC tasarımını desteklemektedir.

LDPC kodlar üzerinde çalışan bir grup araştırmacı tarafından kurulan Digital Fountain, LDPC kodların analizine anahtar sonuçlar denebilecek önemli katkılarda bulunmaktadır [15]. Bu şirket internet üzerinde verimli ve güvenilir içerik dağıtımı sağlamak amacıyla LDPC benzeri kodlar kullanmaktadır.

Lucent Technologies optik şebekeler için bir LDPC kodlayıcı tasarlamıştır. Buradaki cihaz 10 Gb/s verimle, $R = 0.93$ kodlama oranıyla ve 10^{-15} hedef hata başarımıyla çalışmaktadır. LDPC kodlar Jet Propulsion tarafından Consultative Commitee for Space Data Systems (CCSDS)'e önerilmiştir.

Agere de optik şebekeler için bir LDPC kod gerçekleştirmiştir. Howland ve Blanksby düşük güçlü ve yüksek hızlı uygulamalarda LDPC kodların potansiyelini göstermek amacıyla, $R = 0.5$ oranlı ve 1024 kod sözcük uzunluğuna sahip bir LDPC kod gerçekleştirmiştir. Ayrıca depolama endüstrisi de gelecek nesil cihazlarda kullanılması amacıyla LDPC kodlarla ciddi bir biçimde ilgilenmektedir fakat henüz bu özellikte bir ürün duyurulmamıştır.

1.2 Tezin Konuya Katkıları

Bu çalışmada öncelikle LDPC kodların özellikleri tanıtılmaktadır. LDPC kodun tanımı, düzenli ve düzensiz LDPC kodlar ve Tanner grafının açıklamaları 2. bölümde verilmektedir.

3. bölümde LDPC kodlar için alıcı tasarımlarına yer verilmiştir. Öncelikle ilgili matematiksel altyapı anlatılmış, daha sonra bit çevirme algoritmasıyla (bit flip algorithm) mesaj aktarma algoritmasına yer verilmiştir.

Bir sonraki bölümde ise LDPC kodlar için verici tasarımları ele alınmıştır. Bu bölümde LDPC kodların yarı rastgele üretimi, düzenli LDPC kod tasarımıyla ilgili olan sözde rastgele tasarım (pseudo random construction) ve düzensiz LDPC kod tasarımıyla ilgili olan bit doldurma algoritması (bit filling algorithm) anlatılmıştır.

5. bölümde LDPC kod optimizasyonu konusu incelenmiştir. Bu bölümde Gauss yaklaşımı (Gaussian approximation), diğer adıyla yoğunluk evrim (density evolution) ve farksal evrim (differential evolution) tekniklerine yer verilmiştir. Yukarıdaki bölümlerden elde edilen bilgi birikimi sayesinde AWGN kanal için kanal sığasına çok yakın, farklı hızlara ve çeşitli yoğunluklara sahip düzensiz kodlar tasarlanmış, bir iletişim sisteminde yer alan verici, kanal, alıcı yapıları

bilgisayar ortamında oluşturulmuş ve farksal evrim tekniğiyle optimum kod dizisi belirlenip bu dizi kullanılarak bit doldurma algoritmasının LDPC kodları üretmesi sağlanmış, bu kodlar mesaj aktarma algoritması yardımıyla çözülüp başarımları, literatürdekilerle karşılaştırılmış ve kod sözcük uzunluğu sonsuza giderken üstün kodlar elde edilmiştir.

6. bölümde elde edilen sonuçlara yer verilmiştir. İletişim sisteminde optimizasyon, verici ve alıcı tekniği olarak önerilen farksal evrim tekniği, bit doldurma algoritması ve mesaj aktarma algoritması tekniklerinin oluşturdukları birleşim kullanılmıştır.

Tüm bölümlerde yer alan başarım sonuçları MATLAB programı kullanılarak, algoritmalar ise C programlama dili kullanılarak elde edilmiştir.

2. LDPC KODLARIN ÖZELLİKLERİ

Bu bölümde LDPC kodların özellikleri incelenmekte, eşlik denetim matrisi, düzenli ve düzensiz kodlarla Tanner grafı anlatılmaktadır.

2.1 Eşlik Denetim Matrisi

n uzunluklu bir eşlik denetim kodu, kod sözcükleri *n* - *k* adet doğrusal eşlik denetim kısıtlamasını sağlayan doğrusal ikili blok koddur. Geleneksel olarak, *(n – k) x n* boyutundaki eşlik denetim matrisi *H* ile tanımlanır. Bu matristeki *n* - *k* satır, *n* - *k* adet kısıtlamayı ifade eder. Örneğin, ilk kısıtlama 2. ve 5. bitlerin birbirine eşit olduğunu belirtsin. Buna göre *H* matrisinin ilk satırında 2. ve 5. sütunlar 1, diğer sütunlar 0 içerir. Eşlik denetim kodu $\mathbf{c}H^T = 0$ koşulunu sağlayan ***c*** kümesinden oluşmaktadır, $\mathbf{c} = (c_1, c_2, ..., c_n)$. Eğer *H* matrisinin rankı $r = rank(H) \leq n - k$, yani *H* matrisindeki doğrusal bağımsız satır sayısı *r* ise, kod sözcüklerinin sayısı 2^{n-r}, kodun boyutu ise $K = n - r$ olmaktadır. n uzunluklu her bir kod sözcüğü *k* bilgi biti taşıdığından dolayı kodlama oranı $R = \frac{K}{n} = \frac{n - r}{n}$ olarak tanımlanır.

Bir *(n, k)* doğrusal blok kodu ζ, *k* bilgi sözcük uzunluğuna ve *n* kod sözcük uzunluğuna sahiptir. Bir ζ kodunda $\mathbf{u} = (u_0, u_1, ..., u_{k-1})$ 2^k bilgi sözcüğü ve $\mathbf{c} = (c_0, c_1, ..., c_{n-1})$ 2^k kod sözcüğü bulunmaktadır. Bilgi sözcüğünden kod sözcüğüne eşleme ($\mathbf{u} - \mathbf{c}$) sonucunda kod sözcüğü vektörü ise aşağıdaki biçimde yazılabilir:

$$\mathbf{c} = u_0\mathbf{g_0} + u_1\mathbf{g_1} + ... + u_{k-1}\mathbf{g_{k-1}}. \qquad (2.1)$$

Burada **g** vektörleri, üreteç matrisini oluşturan bileşenler olmaktadır. Aynı denklem matris formunda yazılırsa aşağıdaki denklem elde edilir:

$$\mathbf{c} = \mathbf{u}G. \tag{2.2}$$

Burada $G = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ . \\ . \\ . \\ \mathbf{g_{k-1}} \end{bmatrix}_{kxn}$ olarak tanımlanır ve üreteç matrisi (generator matrix) adını alır.

H^T matrisinde, toplandığı zaman tüm sıfır vektörünü oluşturan minimum satır sayısına minimum uzaklık adı verilir.

Örnek olarak (7, 4) Hamming kodunu ele alalım:

$H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Bu matrise bakılarak herhangi 2 satırın toplamından ikili düzende tüm sıfır vektörü elde edilemez fakat 0, 1 ve 6. satırların toplamından tüm sıfır vektörü elde edilebilir. Bu yüzden kodun minimum uzaklığı $d_{\min} = 3$ olarak hesaplanır. Kodun minimum uzaklığı aşağıdaki gibi hesaplanır:

$$d_{\min} = \min\left\{w_H(\mathbf{c}), \mathbf{c} \neq \mathbf{0} : \mathbf{c}H^T = \mathbf{0}\right\}. \tag{2.3}$$

Burada $w_H(\mathbf{c})$, **c**'nin Hamming ağırlığı olarak tanımlanır ve değeri, ilgili vektördeki 1'lerin sayısına eşittir. Bu çalışmada aksi belirtilmediği sürece ikili düzen ele alınacaktır. Yani,

$\left\{\begin{matrix}0+0=0\\0+1=1\\1+0=1\\1+1=0\end{matrix}\right\}$ eşitlikleri geçerli olacaktır.

2.2 Düzenli ve Düzensiz LDPC Kodların Tanımı

Genel olarak diğer tüm doğrusal kodlara ait eşlik denetim matrislerinde sıfırların ve birlerin sayısı yaklaşık olarak aynı olmaktadır. Eşlik denetim matrisinde 1'lerin sayısı, tüm matris elemanlarının sayısının yanında çok seyrek kalıyorsa, bu matrisin oluşturduğu koda düşük yoğunluklu eşlik denetim kodu adı verilir.

2.2.1 Düzenli LDPC Kodlar

M x N boyutlu düzenli bir (*j, k*) LDPC matrisinde her sütunda tam olarak *j* adet, her satırda da tam olarak *k* adet 1 bulunur ve $j < k << N$ koşulu sağlanıyorsa bu matrisin oluşturduğu koda düzenli (regular) LDPC kod denir.

Düzensiz LDPC kodlara ait eşlik denetim matrisleri yine içerdiği 1 sayısı bakımından seyrek olmakla beraber, her satırdaki veya her sütundaki 1'lerin sayısının aynı olması gerekmemektedir.

Düzenli LDPC kodunun her eşlik denetim denkleminde tam olarak k bit vardır ve her bit, *j* tane denklemde kullanılmaktadır ($j \geq 3$). H matrisindeki toplam 1'lerin sayısı ise $M.k = N.j$ olarak hesaplanır. Bu hesap, her biri *k* tane 1 içeren *M* satır ve *j* tane 1 içeren *N* sütunun varolmasından doğar. Eğer *M* satır doğrusal bağımsızsa kodlama oranı *R* aşağıdaki şekilde ifade edilir:

$$R = \frac{K}{N} = \frac{N - M}{N} = 1 - \frac{M}{N} = 1 - \frac{j}{k}. \tag{2.4}$$

Burada *K* sabitine kodun boyutu adı verilir. Kodlama oranının pozitif bir değer olması gerektiği gerçeğinden yola çıkarak (2.4) denkleminden $j < k$ eşitsizliği elde edilir. Ayrıca $M.k = N.j$ denkleminden yola çıkarak *N, j* ve *k*'nın seçiminin bağımsız olmadığı, $N.\frac{j}{k}$ değerinin bir tamsayı olması gerektiği apaçık ortadadır.

Örneğin bir (3, 4) LDPC matrisi N = 1000 veya N = 1004 için tanımlı iken, N = 1002 için tanımlı olmamaktadır.

Düzenli bir (j, k) LDPC matrisinde 1'lerin tüm eleman sayısına oranı k / N'dir. Düşük yoğunluk terimi de N'nin sonsuza gittiği durumda bu oranın sıfıra yaklaşmasından gelmektedir. Yani,

$$\lim_{N\to\infty} \frac{k}{N} \to 0 . \tag{2.5}$$

Düzenli bir M x N boyutlu (j, k) LDPC matrisi genellikle aşağıdaki H_0 alt matrisiyle ifade edilebilir:

$$H_0 = \begin{bmatrix} 11\cdots1 & & & \\ & 11\cdots1 & & \\ & & \ddots & \\ & & & 11\cdots1 \end{bmatrix}. \tag{2.6}$$

(2.6) eşitliğinde H_0 matrisindeki satır sayısı $\frac{N}{k} = \frac{M}{j}$ olup, sütun sayısı N'dir. z. satır, ((z - 1).k + 1). sütundan $z.k$. sütuna kadar 1 içermektedir. Bu matris, (1, k) düzenli LDPC kodu tanımlar. Başarımı incelendiğinde, ilk 2 sütun doğrusal bağımlı olduğundan kodun minimum uzaklığı $d_{\min} = 2$'dir. Düzenli bir (j, k) LDPC matrisi, (2.6)'daki matrisin permütasyonları kullanılarak aşağıdaki şekilde oluşturulabilir:

$$H = \begin{bmatrix} \pi_1(H_0) \\ \pi_2(H_0) \\ \vdots \\ \pi_p(H_0) \end{bmatrix}. \tag{2.7}$$

Bu permütasyonların uygun seçilmesiyle $d_{\min} = 2$'nin üzerinde minimum uzaklıklı kodlar tanımlanabilir.

Örnek olarak LDPC matrisinin satır sayısı 15 olsun ve sadece 13 satır doğrusal bağımsız olsun. Bu durumda, bu 2 adet lineer bağımlı satırı H matrisinden eleyerek yeni bir tam ranklı eşlik denetim matrisi $\tilde{H}$, aşağıdaki şekilde tanımlanabilir:

$$\mathbf{c}H^T = \mathbf{c}\tilde{H}^T = 0. \tag{2.8}$$

Fakat her doğrusal bağımlı satırın atılması sonrasında yeni matristeki k sütundaki 1'lerin sayısı azalacak, dolayısıyla yeni matris düzenlilik özelliğine uymayacaktır.

Şu ana kadar anlatılanları bir örnek üzerinde tekrarlamak gerekirse, kod sözcük uzunluğu $N = 20$, $j = 3$ ve $k = 4$ olan ve eşlik denetim matrisi aşağıdaki şekilde verilen bir LDPC kod ele alınsın [1]:

$$H = \begin{bmatrix}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{bmatrix}. \tag{2.9}$$

Dikkatli incelenecek olursa, H matrisindeki 10. satır ilk dokuz satırın toplamı, 15. satır da 1'den 5'e ve 11'den 14'e kadar olan satırların toplamına eşittir.

Geriye kalan 13 satır ise doğrusal bağımsızdır. Bu yüzden H matrisinin rankı r = rank(H) = 13 olarak hesaplanır. LDPC kodun boyutu ise $K = N - r = 20 - 13 = 7$ olarak bulunur. Kodlama oranı ya da kodlama hızı $R = \frac{K}{N} = \frac{7}{20} = 0.35$ olarak hesaplanır. Eşlik denetim denklemi sayısı, yani H matrisindeki satırlar, $M = \frac{N.j}{k} = \frac{20.3}{4} = 15$ olarak bulunur.

2.2.2 Tanner Grafı

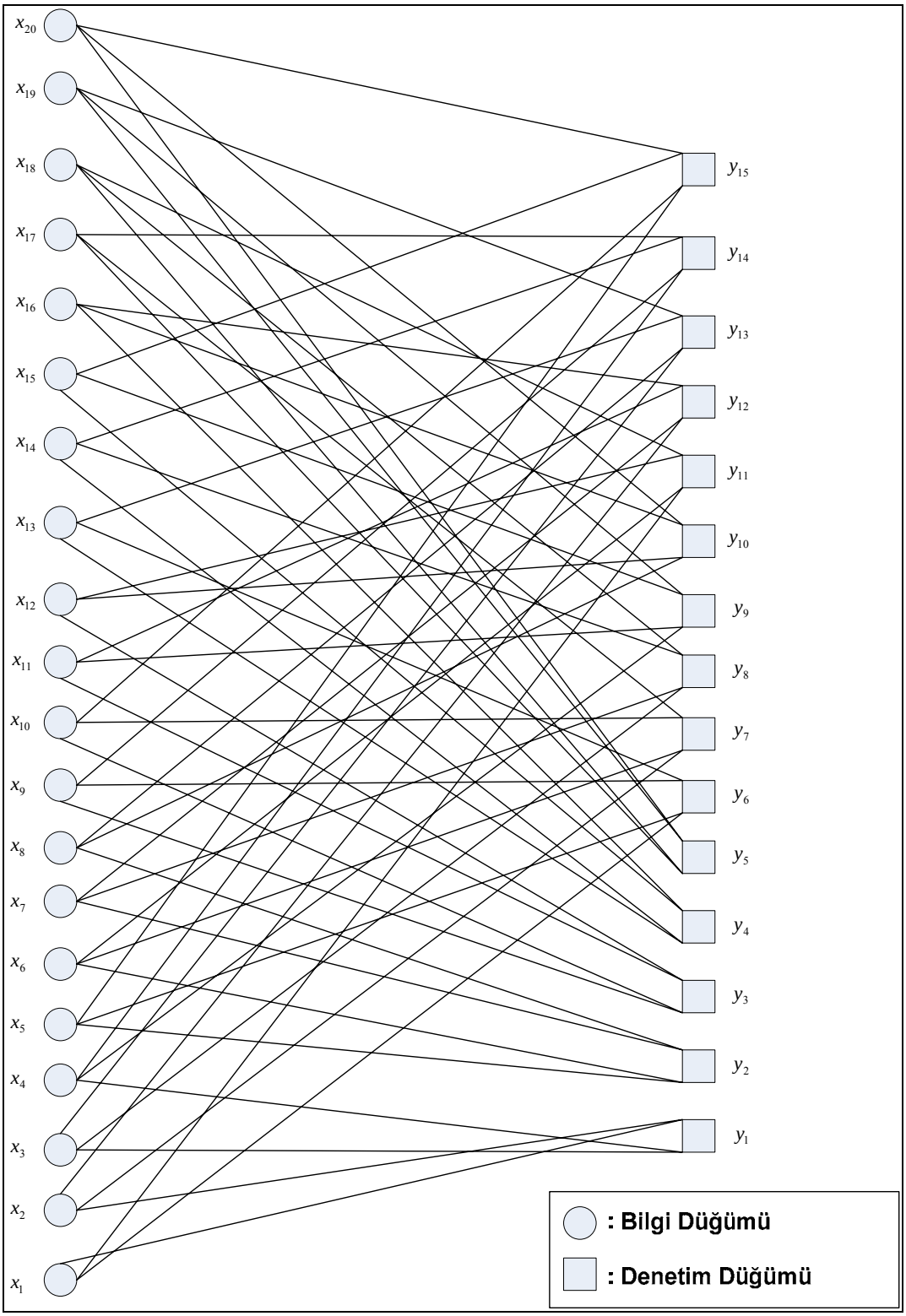
Herhangi bir eşlik denetim kodunu, LDPC kod dahil, görselleştirmek ve grafiksel olarak temsil etmek amacıyla Tanner grafı kullanılmaktadır [13][14][23]. Daha önceden açıklandığı gibi, bir M x N boyutlu eşlik denetim matrisi H, her bir kod sözcük uzunluğu N olan ve M adet eşlik denetim kısıtlamasını sağlayan kodu belirlemektedir. Tanner grafı her bir bit için N adet bit düğümü (bit node) ve her bir eşlik denetim denklemi için de M adet denetim düğümü (check node) içermektedir. Tez çalışması boyunca bit düğümleri daire ile, denetim düğümleri de kare ile gösterilecektir. Temel olarak m. denetim düğümüyle n. bit düğümü arasında ancak ve ancak eşlik denetim matrisindeki (m, n). elemanın değeri 1 ise, yani $H_{m,n} = 1$ ise, bu iki düğüm bir dalla (branch) birleşir. Aynı türden iki düğüm arasında doğrudan bağlantı olmadığından dolayı Tanner grafları çift yönlü (bipartite) olarak tanımlanmıştır. Şekil 2.1'de (2.9)'da verilen eşlik denetim koduna ilişkin Tanner grafı görülmektedir. Bu şekilde, bit düğümleri N = 20 adet daireyle, denetim düğümleri de M = 15 adet kareyle gösterilmiştir. $y_1, y_2, ..., y_5$ denetim düğümleri H_0'ı ($H_0 = \pi_1(H_0)$), $y_6, y_7, ..., y_{10}$ denetim düğümleri $\pi_2(H_0)$'ı, $y_{11}, y_{12}, ..., y_{15}$ denetim düğümleri de $\pi_3(H_0)$'ı temsil etmektedir. (2.22)'deki denklem takımı, bit düğümlerine ait kısıtlamaları (eşlik denetim denklemleri) ifade etmektedir:

$$
\begin{aligned}
y_1 &= x_1 + x_2 + x_3 + x_4 = 0, \\
y_2 &= x_5 + x_6 + x_7 + x_8 = 0, \\
y_3 &= x_9 + x_{10} + x_{11} + x_{12} = 0, \\
y_4 &= x_{13} + x_{14} + x_{15} + x_{16} = 0, \\
y_5 &= x_{17} + x_{18} + x_{19} + x_{20} = 0, \\
y_6 &= x_1 + x_5 + x_9 + x_{13} = 0, \\
y_7 &= x_2 + x_6 + x_{10} + x_{17} = 0, \\
y_8 &= x_3 + x_7 + x_{14} + x_{18} = 0, \\
y_9 &= x_4 + x_{11} + x_{15} + x_{19} = 0, \\
y_{10} &= x_8 + x_{12} + x_{16} + x_{20} = 0, \\
y_{11} &= x_1 + x_6 + x_{12} + x_{18} = 0, \\
y_{12} &= x_2 + x_7 + x_{11} + x_{16} = 0, \\
y_{13} &= x_3 + x_8 + x_{13} + x_{19} = 0, \\
y_{14} &= x_4 + x_9 + x_{14} + x_{17} = 0, \\
y_{15} &= x_5 + x_{10} + x_{15} + x_{20} = 0.
\end{aligned} \tag{2.10}
$$

Düzenli bir (*j*, *k*) LDPC kodunda her bir bit, *j* adet eşlik denetim denkleminde yer almaktadır. Dolayısıyla bir bit düğümünden çıkan dal sayısı her zaman *j* olmaktadır. Benzer olarak her bir eşlik denetim denklemi k bit içerdiğinden dolayı, her bir denetim düğümünden çıkan dal sayısı her zaman k olmaktadır.

Bir düğümün derecesi ise o düğüme gelen dal sayısı olarak tanımlanmaktadır. Düzenli LPDC kodu tekrar tanımlamak gerekirse, bir kod için eğer bit düğümlerinin dereceleri birbirine eşit ise ve denetim düğümlerinin dereceleri de birbirine eşitse bu koda düzenli LDPC kod denir. Düzensiz LDPC kodlar içinse her bir düğüm için farklı dereceler geçerli olabilmektedir.

Yukarıdaki açıklamaya dayanarak Şekil 2.1'de verilen kodun, tüm denetim düğümlerinin derecelerinin aynı ve 4 ve tüm bit düğümlerinin derecelerinin aynı ve 3 olması nedeniyle, bir (3, 4) düzenli LDPC kodu oluşturduğu söylenebilir.



Şekil 2.1: Düzenli LDPC Kodlar için Tanner Grafı Örneği

Tanner grafının en büyük avantajı, mesaj aktarma algoritmasının anlatılması sırasında anlaşılacaktır. Bu graf, kod çözme algoritmalarının kolaylıkla geliştirilebilmesine olanak sağlamaktadır.

Ayrıca LDPC kodlara ilişkin Tanner graflarının, eşlik denetim matrisindeki 1 sayısının az olması nedeniyle, seyrek yapıda olması, LDPC kodların algoritmik verimliliğinde anahtar özellik olarak gösterilmektedir. Aynı zamanda bu seyreklik sayesinde kodun minimum uzaklık değerinin maksimize edilmesi de mümkün olmaktadır.

Tanner graflarında bir bit düğümünden (bilgi düğümü de denir) başlayıp aynı bit düğümünde sona eren yola çevrim (cycle) adı verilmektedir. Graf iki yönlü olduğundan dolayı bir çevrimin minimum uzunluğu dört olmaktadır. Grafta eğer hiçbir şekilde çevrim mevcut değilse o grafa ağaç (tree) adı verilir. Çevrim içermeyen grafın, yani bir ağacın, iki önemli özelliği aşağıda verilmiştir:

- Herhangi bir dalın kaldırılması halinde iki ayrı alt graf meydana gelir.
- Herhangi bir çift bit düğümünü bağlayan tek bir yol vardır.

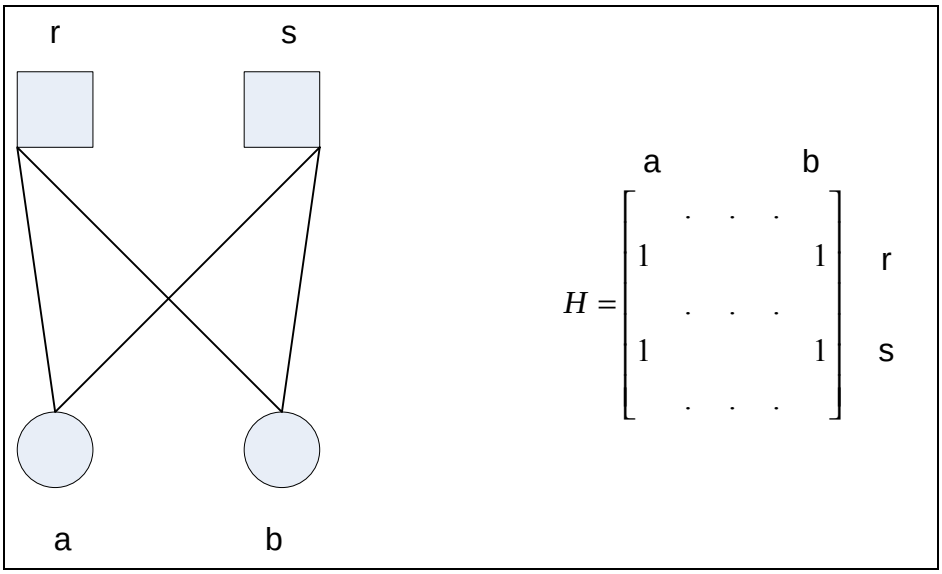
Rastgele bir biçimde üretilen bir eşlik denetim matrisine ait Tanner grafında 4 uzunluklu çevrimlerin olma olasılığı çok yüksektir [9].

Bir eşlik denetim matrisi *H*'daki 4 uzunluklu çevrimler, *H*'nın alt matrisinin köşelerinde yer alan 1'ler nedeniyle var olmaktadır. 4 uzunluklu çevrimlere ve buna ilişkin *H* matrisine örnek olarak Şekil 2.2 verilebilir.

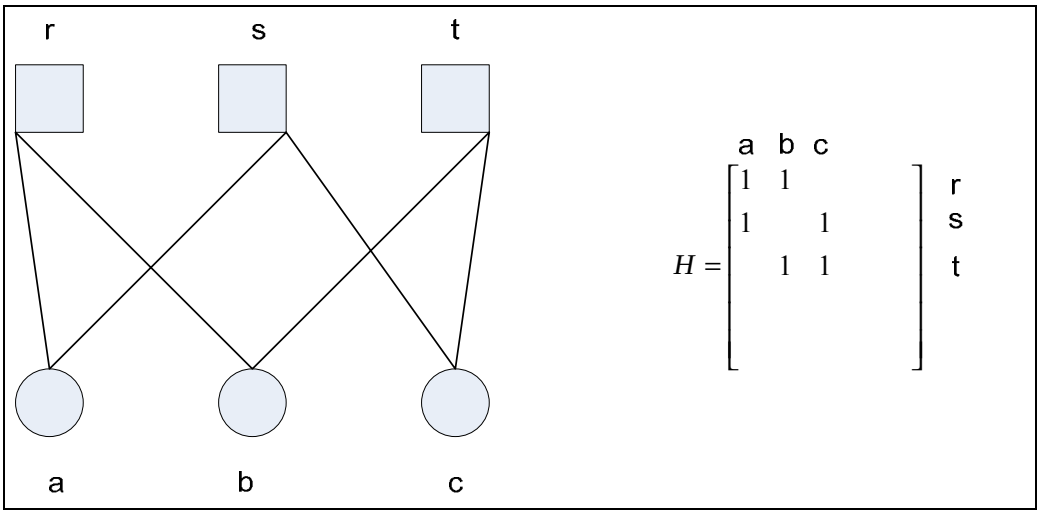
LDPC kodlara ilişkin Tanner grafındaki 6 uzunluklu çevrimlerin bulunması o kadar da kolay değildir. Şekil 2.3'te 6 uzunluklu çevrimlere ilişkin bir graf örneği ve buna ilişkin *H* matrisi yer almaktadır.

Teorem 2.1: İkili düzende bir eşlik denetim matrisi *H* ve $\mu_1 \geq \mu_2 \geq ... \geq \mu_r$ şeklinde sıralanmış özdeğerlere (eigenvalues) sahip $H^T H$ matrisi olsun (Tüm özdeğerler $H^T H$ matrisinin simetrik olması nedeniyle gerçel değerlere sahiptir). *G*, herbiri m derecesine sahip *n* bit düğümünün ve herbiri *j* derecesine sahip *r* denetim düğümünün oluşturduğu düzenli bağlantılı graf olsun. Bu durumda ilgili kodun minimum uzaklığı aşağıdaki eşitsizliği sağlamaktadır [16]:

$$d \geq \frac{n(2m-\mu_2)}{(mj-\mu_2)}. \qquad (2.11)$$



Şekil 2.2: 4 Uzunluklu Çevrimlere Örnek



Şekil 2.3: 6 Uzunluklu Çevrimlere Örnek

Düzenli bir (*j*, *k*) LDPC kodunun oluşturulmasında $2\ell(n)$'e eşit veya ondan kısa uzunluklu çevrimlerin bulunmaması istenir [6]. Burada $\ell(n)$ aşağıdaki şekilde ifade edilir:

$$\ell(n) = \frac{\ln n - \ln \frac{jk - k}{2k}}{\ln\left[(k-1)(j-1)\right]}. \tag{2.12}$$

Bir LDPC koda ilişkin Tanner grafındaki en kısa uzunluğa sahip çevrimin uzunluğuna minimum çevrim uzunluğu (girth) adı verilir.

2.2.3 Düzensiz LDPC Kodlar

Eğer eşlik denetim matrisinde her satır veya her sütundaki 1'lerin sayısı farklılık gösteriyorsa bu koda düzensiz (irregular) LDPC kodu adı verilir. Bu tür kodlarda her bir düğümün derecesi, belli bir dağılıma göre seçilmektedir. Bu nedenle örneğin düzensiz bir LDPC kodunun grafiksel gösteriliminde bit düğümlerinin yarısının derecesi 3, diğer yarısının derecesi 5 iken denetim düğümlerinin yarısının derecesi 6 ve diğer yarısının derecesi 8 olabilmektedir.

Düzensiz LDPC kodların kullanılmasının altında yatan ana neden, kod çözme sürecinde dalga etkisi (wave effect) adı verilen ve yüksek derecelere sahip bit düğümlerinin doğru değere çok çabuk bir şekilde yaklaştığı ve denetim düğümlerine ve daha düşük dereceli bit düğümlerine daha güvenilir bilgi sağladıkları anlamına gelen bir fenomenin varlığıdır [19].

Derece dağılımı çifti (λ, ρ)'ya sahip olan düzensiz bir LDPC kodu aşağıdaki şekilde ifade edilir:

$$C^n(\lambda, \rho). \tag{2.13}$$

Burada *n*, kod sözcük uzunluğunu göstermektedir. λ ise bit düğümlerinin derecelerinin dağılımını ifade etmektedir ve aşağıdaki biçimde tanımlanır:

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1} . \tag{2.14}$$

(2.13)'teki ρ ise denetim düğümlerinin derecelerinin dağılımını ifade etmektedir ve aşağıdaki biçimde tanımlanır:

$$\rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1} . \tag{2.15}$$

Özetle λ_i, i derecesine sahip bit düğümlerine gelen dalların tüm dallara oranını, ρ_i ise i derecesine sahip denetim düğümlerine gelen dalların tüm dallara oranını göstermektedir. (2.14) ve (2.15) denklemlerinde geçen d_v ve d_c değişkenleri sırasıyla maksimum bit ve maksimum denetim düğüm derecelerini göstermektedir. Tüm λ'ların toplamı, tüm ρ'ların toplamına eşit olup değeri 1'dir:

$$\sum_{i=2}^{d_v} \lambda_i = \sum_{i=2}^{d_c} \rho_i = 1 . \tag{2.16}$$

Aşağıda düzensiz LDPC kodlara ilişkin bir örnek bulunmaktadır [17]:

$$\begin{aligned} \lambda(x) &= 0.1575x + 0.3429x^2 + 0.0363x^5 + 0.0590x^6 + 0.2790x^8 + 0.1253x^9 \\ \rho(x) &= 0.8266x^{34} + 0.1345x^{35} + 0.0087x^{70} + 0.0302x^{71} \end{aligned} . \tag{2.17}$$

(2.18)'de verilen eşlik denetim matrisi ise düzensiz bir LDPC koda ilişkin örnek oluşturmaktadır.

Düzenli LDPC kodlar da (2.14) ve (2.15)'teki denklemler kullanılarak ifade edilebilir. Örnek olarak (3, 6) düzenli LDPC kodu ele alınsın. İlgili grafta bit düğümlerinin dereceleri sabit ve 3 olduğu için (2.14) denkleminde sadece λ_3 mevcut olup değeri 1'dir çünkü 3. dereceye sahip bit düğümlerindeki dalların toplamının tüm dallara oranı 1'dir. Aynı şekilde, ilgili grafta denetim düğümlerinin dereceleri sabit ve 6 olduğu için (2.15) denkleminde sadece ρ_6 mevcut olup değeri 1'dir çünkü 6. dereceye sahip denetim düğümlerindeki dalların toplamının tüm

dallara oranı 1'dir. bu yüzden düzenli bir (3, 6) LDPC kodunun dağılım parametreleri $\lambda(x) = x^2$ ve $\rho(x) = x^5$ olacaktır.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.18}$$

Genelleme yapmak gerekirse düzenli bir (*j*, *k*) LDPC kodunun dağılım parametreleri aşağıdaki gibi olmalıdır:

$$\lambda(x) = x^{j-1} \quad \text{ve} \quad \rho(x) = x^{k-1}. \tag{2.19}$$

Kodun *n* adet bit düğümü olduğu varsayılsın. *i* derecesine sahip bit düğümlerinin sayısı aşağıdaki biçimde hesaplanır:

$$n\frac{\lambda_i / i}{\sum_{j\geq 2} \lambda_j / j} = n\frac{\lambda_i / i}{\int_0^1 \lambda(x)dx}. \tag{2.20}$$

Tüm bit düğümlerinden çıkan dalların toplamı ise aşağıdaki biçimde hesaplanır:

$$n\sum_{i\geq 2} \frac{\lambda_i / i}{\int_0^1 \lambda(x)dx} = n\frac{1}{\int_0^1 \lambda(x)dx}. \tag{2.21}$$

Aynı biçimde *m* adet denetim düğümünden çıkan dalların toplamı da aşağıdaki biçimde hesaplanır:

$$m\frac{1}{\int_0^1 \rho(x)dx}. \tag{2.22}$$

(2.21) ve (2.22)'deki denklemler birleştirilirse aşağıdaki eşitlik elde edilir:

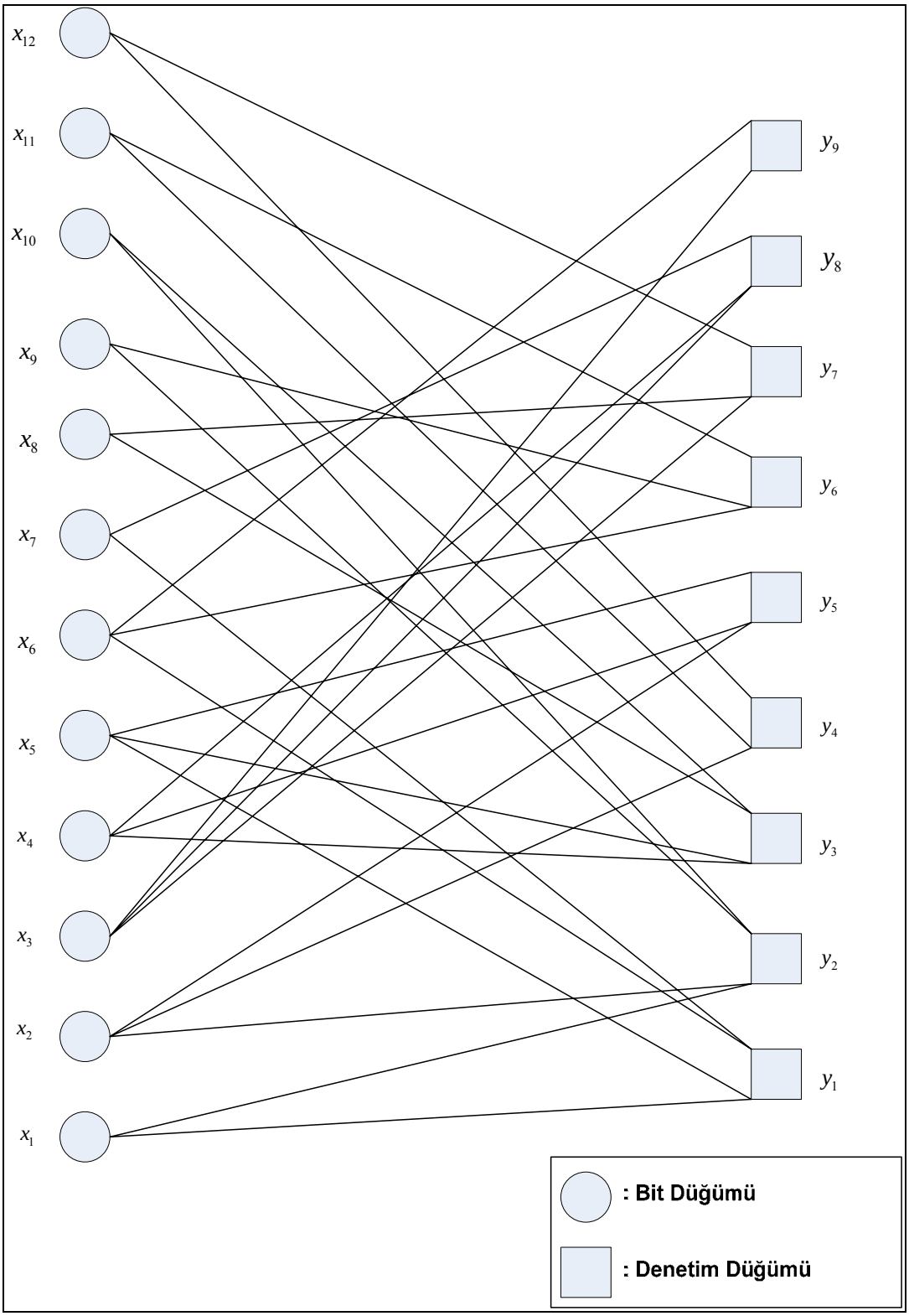
$$m = n\frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{2.23}$$

Tüm denetim denklemlerinin doğrusal bağımsız olduğundan yola çıkarak kodlama oranı aşağıdaki şekilde hesaplanır [18]:

$$R(\lambda,\rho) = \frac{n-m}{n} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{2.24}$$

(2.18)'de eşlik denetim matrisi verilen düzensiz LDPC koda ilişkin Tanner grafi ise Şekil 2.4'te verilmiştir. Ayrıca (2.18)'deki kısıtlama denklemleri de (2.25)'te verilmiştir:

$$\begin{aligned}
y_1 &= x_1 + x_5 + x_6 + x_7 = 0,\\
y_2 &= x_1 + x_2 + x_9 + x_{10} = 0,\\
y_3 &= x_4 + x_5 + x_8 + x_{10} = 0,\\
y_4 &= x_2 + x_{11} + x_{12} = 0,\\
y_5 &= x_2 + x_4 + x_5 = 0,\\
y_6 &= x_6 + x_9 + x_{11} = 0,\\
y_7 &= x_3 + x_8 + x_{12} = 0,\\
y_8 &= x_3 + x_4 + x_7 = 0,\\
y_9 &= x_1 + x_3 + x_6 = 0.
\end{aligned} \tag{2.25}$$



Şekil 2.4: Düzensiz LDPC Kodlar için Tanner Grafı Örneği

3. LDPC KODLAR İÇİN ALICI TASARIMI

Bu bölümde LDPC kodlar için gerekli matematiksel altyapı ve kod çözme problemiyle yumuşak ve sert kararlı kod çözme teknikleri anlatılmaktadır.

3.1 Matematiksel Altyapı

İkili rastlantı değişkeni $c \in \{0,1\}$'nin istatistiksel özellikleri aşağıdaki denklem takımıyla ifade edilsin:

$$p = \Pr[c = 1],$$

$$\Pr[c = 0] = 1 - p,$$

$$\lambda = \log \frac{\Pr[c = 1]}{\Pr[c = 0]}, \tag{3.1}$$

$$e^{\lambda} = \frac{p}{1 - p},$$

$$p = \frac{1}{1 + e^{-\lambda}} = \frac{e^{\lambda}}{1 + e^{\lambda}}.$$

λ'nın işareti c'nin değeriyle ilişkilidir. Yani c = 1'in olma olasılığı c = 0'ın olma olasılığından daha yüksekse λ'nın değeri pozitif, c = 0'ın olma olasılığı c = 1'in olma olasılığından daha düşükse λ'nın değeri negatif olacaktır.

Ayrıca λ'nın büyüklüğü olan $|\lambda|$, kesinliğin (certainty) bir ölçüsüdür. Yani eğer $\lambda = 0$ ise 0 ve 1'in olasılıkları eşittir ($p = \frac{1}{2}$). Eğer $\lambda = \propto$ ise c = 1'in olasılığı 1'dir. Eğer $\lambda = -\propto$ ise c = 0'ın olasılığı 1 olmaktadır.

$c \in \{0,1\}$ şeklinde rastgele bir bit verilmiş olsun. r de olasılık yoğunluk fonksiyonu (pdf) c'ye bağlı olan $f(r \mid c)$ fonksiyonunun bir gözlemini işaret etsin. Eğer c sabitse $f(r \mid c)$, r'nin bir fonksiyonu olarak görülür ve koşullu olasılık yoğunluk fonksiyonu (conditional pdf) adını alır. Öte yandan eğer r sabitse $f(r \mid c)$, c'nin bir fonksiyonu olarak görülür ve olabilirlik (likelihood) fonksiyonu adını alır.

Bir gözlem yapmadan önceki Pr[c = 1] ve Pr[c = 0] olasılıklarına önsel (a priori) olasılıklar, gözlem yaptıktan sonraki Pr[c = 1 | r] ve Pr[c = 0 | r] olasılıklarına sonsal (a posteriori) olasılıklar denmektedir. Bayes kuralına göre sonsal olasılıklar, olasılık fonksiyonuyla aşağıdaki denklemde ifade edildiği gibi orantılı olmaktadır:

$$\Pr[c = 1 \mid r] = \frac{f(r \mid c = 1)\Pr[c = 1]}{f(r)}. \tag{3.2}$$

Ayrıca sonsal olasılıkların oranının logaritması da aşağıdaki denklemde ifade edilmiştir:

$$\log \frac{\Pr[c = 1 \mid \mathrm{r}]}{\Pr[c = 0 \mid \mathrm{r}]} = \log \frac{f(r \mid c = 1)}{f(r \mid c = 0)} + \log \frac{\Pr[c = 1]}{\Pr[c = 0]}. \tag{3.3}$$

Eşitliğin sağ tarafındaki ilk terim logaritmik olabilirlik oranı (log – likelihood ratio, LLR) adını almaktadır. Ikinci terim ise önsel logaritmik olabilirlik oranı adını alır. Eşitliğin sol tarafındaki terime ise sonsal logaritmik olabilirlik oranı adi verilir. Eğer c'nin 1 ya da 0 olma değeri birbirine eşitse önsel LLR sıfır olur ve sonsal LLR ile LLR birbirine eşit olur.

3.2 Tanh Kuralı

$\phi(\mathbf{c}) \in \{0,1\}$, n bitten oluşan $\mathbf{c} = \lfloor c_1 c_2 .... c_n \rfloor$ kümesinin paritesini göstersin. Yani eğer **c**'de çift sayıda 1 varsa $\phi(\mathbf{c}) = 0$, eğer **c**'de tek sayıda 1 varsa $\phi(\mathbf{c}) = 1$ olmaktadır. Bu ifadenin matematiksel biçimi aşağıda verilmiştir:

$$\phi(\mathbf{c}) = \begin{cases} 0, & \sum_{i=1}^{n} c_i = 0 \,(\mathrm{mod}\, 2) \text{ ise} \\ 1, & \sum_{i=1}^{n} c_i = 1 \,(\mathrm{mod}\, 2) \text{ ise.} \end{cases} \tag{3.4}$$

Eğer bitler bağımsız ise parite için önsel LLR, tanh kuralına uyar [8][20].

$\lambda_i = \log \frac{\Pr[c_i = 1]}{\Pr[c_i = 0]}$, i. bit için önsel LLR olarak tanımlanmıştı. $\lambda_{\phi(c)} = \log \frac{\Pr[\phi(c) = 1]}{\Pr[\phi(c) = 0]}$ önsel LLR de aşağıdaki eşitlikteki gibi tanh kuralına uyar:

$$\tanh(\frac{\lambda_{\phi(c)}}{2}) = \prod_{i=1}^{n} \tanh(\frac{\lambda_i}{2}). \tag{3.5}$$

Burada $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ ve $\tanh(\frac{x}{2}) = \frac{e^x - 1}{e^x + 1}$ olarak tanımlıdır.

<u>*Tanıt:*</u> İkili vektör $\mathbf{c} = \lfloor c_1 c_2 .... c_n \rfloor$'nin paritesi $\phi(\mathbf{c})$, aşağıdaki şekilde ifade edilebilir:

$$\phi(\mathbf{c}) = \frac{1}{2}\left(1 - \prod_{i=1}^{n} (1 - 2c_i)\right). \tag{3.6}$$

Paritenin tek sayı olması olasılığı, yani tek sayıda 1 içermesi olasılığı, $\phi(\mathbf{c})'nin$ beklenen değerine eşittir:

$$\Pr[\phi(\mathbf{c}) = 1] = E[\phi(\mathbf{c})] \tag{3.7}$$

$$= \frac{1}{2}\left(1 - E\left[\prod_{i=1}^{n}(1-2c_i)\right]\right) \tag{3.8}$$

$$= \frac{1}{2}\left(1 - \prod_{i=1}^{n}(1-2E[c_i])\right) \text{ (bitlerin bağımsız olmasından dolayı)} \tag{3.9}$$

$$= \frac{1}{2}\left(1 - \prod_{i=1}^{n}\left(1 - \frac{2e^{\lambda_i}}{1+e^{\lambda_i}}\right)\right) \tag{3.10}$$

$$= \frac{1}{2}\left(1 - \prod_{i=1}^{n}\frac{1-e^{\lambda_i}}{1+e^{\lambda_i}}\right) \tag{3.11}$$

$$= \frac{1}{2}\left(1 - \prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right). \tag{3.12}$$

$\Pr[\phi(\mathbf{c})=0] = 1 - \Pr[\phi(\mathbf{c})=1]$ olduğundan dolayı $\phi(\mathbf{c})$'nin LLR'si aşağıdaki gibi ifade edilir:

$$\lambda_{\phi(\mathbf{c})} = \log\frac{\Pr[\phi(\mathbf{c})=1]}{\Pr[\phi(\mathbf{c})=0]} = \log\left(\frac{1 - \prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}{1 + \prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}\right). \tag{3.13}$$

(3.13)'ten yararlanılarak (3.5)'te yer alan tanh kuralının tanıtı, (3.14), (3.15) ve (3.16)'da yer alan denklemler sayesinde sonlanmaktadır:

$$\tanh\left(\frac{-\lambda_{\phi(\mathbf{c})}}{2}\right)=\frac{1-\left(\frac{1-\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}{1+\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}\right)}{1+\left(\frac{1-\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}{1+\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)}\right)} \tag{3.14}$$

$$=\frac{\left(1+\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right)-\left(1-\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right)}{\left(1+\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right)+\left(1-\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right)} \tag{3.15}$$

$$=\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right). \tag{3.16}$$

(3.5)'te $\lambda_{\phi(\mathbf{c})}$'nin çözümü aşağıda verilmiştir:

$$\lambda_{\phi(\mathbf{c})}=-2\tanh^{-1}\left(\prod_{i=1}^{n}\tanh\left(\frac{-\lambda_i}{2}\right)\right). \tag{3.17}$$

Alternatif olarak LLR'lerin işaretlerinin ve büyüklüklerinin ayrılması istensin. (3.5)'te yer alan denklemde - λ_i yerine $sign(-\lambda_i)|\lambda_i|$ konulursa şu denklem takımı çifti elde edilir:

$$sign\left(-\lambda_{\phi(\mathbf{c})}\right)=\prod_{i=1}^{n}sign\left(-\lambda_i\right), \tag{3.18}$$

$$\tanh\left(\frac{\left|\lambda_{\phi(\mathbf{c})}\right|}{2}\right)=\prod_{i=1}^{n}\tanh\left(\frac{|\lambda_i|}{2}\right). \tag{3.19}$$

(3.19) eşitliğinin her iki tarafı –log(.) fonksiyonuna sokulursa aşağıdaki denklem elde edilir:

$$f\left(\left|\lambda_{\phi(\mathbf{c})}\right|\right)=\sum_{i=1}^{n} f\left(\left|\lambda_i\right|\right). \tag{3.20}$$

Burada $f(x)=\log\frac{e^x+1}{e^x-1}=-\log\left(\tanh\left(\frac{x}{2}\right)\right)$ olarak tanımlanmıştır.

Her $x > 0$ değeri için $f(f(x)) = x$ olduğundan (3.20)'nin her iki tarafına $f(.)$ fonksiyonu uygulanırsa aşağıdaki denklem elde edilir:

$$\left|\lambda_{\phi(\mathbf{c})}\right|=f\left(\sum_{i=1}^{n} f\left(\left|\lambda_i\right|\right)\right). \tag{3.21}$$

(3.18) ile (3.21) denklemlerinin bir araya getirilmesiyle de

$$\lambda_{\phi(\mathbf{c})}=-\prod_{i=1}^{n} sign(-\lambda_i) f\left(\sum_{i=1}^{n} f\left(\left|\lambda_i\right|\right)\right) \tag{3.22}$$

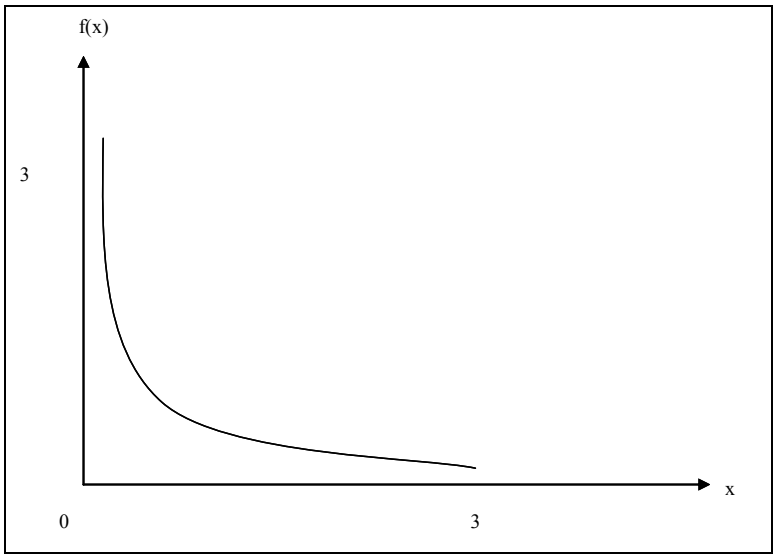
elde edilmiş olur.

Donanım gerçekleştirmelerinde (3.22)'de yer alan denklem, (3.5) ve (3.17)'de yer alan denklemlere tercih edilmektedir çünkü burada n adet terimin çarpımı yerine n adet terimin toplamıyla uğraşılmaktadır. Yine de analizlerde, kavramsal kolaylık getirdiğinden dolayı (3.5) denklemi tercih edilmektedir.

Şekil 3.1'de $f(x)$ fonksiyonu çizilmiştir. Görüldüğü üzere her $x > 0$ değeri için $f(x)$ pozitif ve monoton azalan bir fonksiyon olmaktadır. $f(0)=\propto$ ve $f(\propto)=0$ eşitlikleri geçerlidir. Aynı zamanda her $x > 0$ değeri için $f(f(x)) = x$ olmaktadır.

$\hat{c}=[\hat{c}_1\hat{c}_2....\hat{c}_n]$ c'nin en büyük olabilirlikli değerini göstersin. Burada aşağıdaki eşitlik geçerlidir:

$$\hat{c}_i=\begin{cases}0, \lambda_i \le 0\\ \\ 1, \lambda_i > 0.\end{cases} \tag{3.23}$$



Şekil 3.1: *f*(x) Fonksiyonu

$\phi(\mathbf{c})$'nin en yakın olasılıklı değerini gösteren $\lambda_{\phi(\mathbf{c})}$'nin işareti aşağıdaki gibi belirlenmektedir:

$$sign\left(\lambda_{\phi(\mathbf{c})}\right) = -\prod_{i=1}^{n} sign\left(-\lambda_i\right) = -(-1)^{\phi(\hat{c})}. \tag{3.24}$$

Daha önce $\left|\lambda_{\phi(\mathbf{c})}\right| = f\left(\sum_{i=1}^{n} f\left(\left|\lambda_i\right|\right)\right)$ olarak belirlenmişti. *c*'nin *k*. biti eşit olasılıkla 0 veya 1 olsun. Bu durumda (3.21)'deki toplam ifadesi sonsuz olmaktadır. $f(\infty) = 0$ olduğundan $\lambda_{\phi(c)} = 0$ olur, yani geriye kalan bitlerin önemi sıfırlanır. Genel olarak bir bit, diğer bitlerden önemli ölçüde az kesinliğe sahipse aşağıdaki yaklaşıklık yapılabilir:

$$\left|\lambda_{\phi(\mathbf{c})}\right| = f\left(\sum_{i=1}^{n} f\left(\left|\lambda_i\right|\right)\right) \cong f\left(f\left(\left|\lambda_{\min}\right|\right)\right) = \left|\lambda_{\min}\right|. \tag{3.25}$$

(3.25)'ten çıkan sonuca göre kesinlik, en az kesin olan bitin kesinliğine yaklaşık olarak eşit olmaktadır. (3.25)'te elde edilen eşitlik (3.22)'de kullanılırsa aşağıdaki yaklaşıklık elde edilir:

$$\lambda_{\phi(\mathbf{c})} \cong -(-1)^{\phi(\hat{c})}\left|\lambda_{\min}\right|. \tag{3.26}$$

Bir sonraki bölümde (3.17)'nin kod çözme probleminde nasıl kullanıldığı anlatılmaktadır. Daha az karmaşıklık getirmesi nedeniyle (3.17) yerine (3.26) kullanılacaktır.

3.3 Kod Çözme Problemi

Eşlik denetim matrisi *H*'ya ait bir kodun, toplamsal beyaz Gauss gürültülü (AWGN) bir kanaldan geçtikten sonra çözüldüğü düşünülsün. Alıcı tarafındaki gözlem olan $r = [r_1 r_2 .... r_N]$ dizisi, iletilen kod sözcüğü olan $c = [c_1 c_2 .... c_N]$ dizisiyle aşağıdaki biçimde ilişkilidir:

$$r = 2c - 1 + n. \tag{3.27}$$

Burada gürültü vektörü olan *n*'nin bileşenleri bağımsız, sıfır ortalamalı ve σ^2 varyanslı Gauss rastlantı değişkenleridir. Ayrıca (3.27) eşitliği antipodal bit – simge eşlemesini ifade etmektedir ($0 \rightarrow -1, 1 \rightarrow 1$).

n. bit için hata olasılığını minimize eden sezici, sonsal LLR'yi aşağıdaki denklem yardımıyla hesaplamaktadır:

$$\lambda_n = \log\frac{\Pr[c=1 \mid \mathrm{r}]}{\Pr[c=0 \mid \mathrm{r}]} = \log\frac{\Pr[c_n=1 \mid \mathrm{r}_n, \{r_i \neq n\}]}{\Pr[c_n=0 \mid \mathrm{r}_n, \{r_i \neq n\}]}. \tag{3.28}$$

Daha sonra da sezici aşağıdaki karşılaştırma sonucunda kestirilen değere karar vermektedir:

$$\hat{c}_n = \begin{cases} 1, \lambda_n > 0 \\ 0, \lambda_n \leq 0. \end{cases} \tag{3.29}$$

(3.28) eşitliğindeki oranın pay kısmına Bayes kuralı uygulanırsa

$$\Pr\left[c_n = 1 \mid \mathrm{r}_n, \{r_i \neq n\}\right] = \frac{f\left(r_n, c_n = 1, \{r_i \neq n\}\right)}{f\left(r_n, \{r_i \neq n\}\right)} \tag{3.30}$$

$$= \frac{f(r_n \mid c_n = 1, \{r_i \neq n\}) f\left(c_n = 1, \{r_i \neq n\}\right)}{f(r_n \mid \{r_i \neq n\}) f\left(\{r_i \neq n\}\right)} \tag{3.31}$$

$$= \frac{f(r_n \mid c_n = 1) \Pr\left[c_n = 1 \mid \{r_i \neq n\}\right]}{f(r_n \mid \{r_i \neq n\})} \tag{3.32}$$

elde edilir.

Aynı işlemler (3.28) eşitliğindeki kesrin payda kısmına uygulanabilir. Böylece (3.28) eşitliğinden

$$\lambda_n = \log \frac{f(r_n \mid c_n = 1) \Pr\left[c_n = 1 \mid \{r_i \neq n\}\right]}{f(r_n \mid c_n = 0) \Pr\left[c_n = 0 \mid \{r_i \neq n\}\right]} \tag{3.33}$$

$$= \log \frac{f(r_n \mid c_n = 1)}{f(r_n \mid c_n = 0)} + \log \frac{\Pr\left[c_n = 1 \mid \{r_i \neq n\}\right]}{\Pr\left[c_n = 0 \mid \{r_i \neq n\}\right]} \tag{3.34}$$

olup, sonuç olarak

$$\lambda_n = \underbrace{\frac{2}{\sigma^2} r_n}_{\substack{\text{Esas Bilgi} \\ \text{(Intrinsic)}}} + \underbrace{\log \frac{\Pr\left[c_n = 1 \mid \{r_i \neq n\}\right]}{\Pr\left[c_n = 0 \mid \{r_i \neq n\}\right]}}_{\substack{\text{Yan Bilgi} \\ \text{(Extrinsic)}}} \tag{3.35}$$

elde edilir.

Yukarıdaki denklemde AWGN kanal için geçerli olan

$$f(r_n \mid c_n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(r_n - 2c_n + 1)^2}{2\sigma^2}} \tag{3.36}$$

eşitliği kullanılmıştır.

(3.35) denklemindeki ilk terim, *n*. kanal gözleminin katkısını göstermekte olup esas bilgi (intrinsic information) adını alır. İkinci terim ise diğer gözlemlerin katkılarını göstermekte olup yan bilgi (extrinsic information) adını almaktadır. (3.35)'ten görüldüğü üzere bu katkılar basitçe toplanarak birleşmektedir. Ayrıca esas bilgi, *n*. kanal gözlemiyle doğru orantılıdır. Esas bilgide yer alan $\frac{2}{c^2}$ sabitine kanal güvenilirliği (channel reliability) adı verilir.

(3.35) denklemini daha da basitleştirmek amacıyla aşağıdaki eşitlik yazılabilir [21][22]:

$$\lambda_n = \frac{2}{\sigma^2} r_n - 2\sum_{i=1}^{j} \tanh^{-1}\left(\prod_{l=2}^{k} \tanh\left(\frac{-\lambda_{i,l}}{2}\right)\right). \tag{3.37}$$

Burada

$$\lambda_{i,l} = \log \frac{\Pr\left[c_{i,l} = 1 \,\middle|\, \{r_i \neq n\}\right]}{\Pr\left[c_{i,l} = 0 \,\middle|\, \{r_i \neq n\}\right]} \tag{3.38}$$

olarak tanımlanmıştır.

Tanner grafı yardımıyla (3.37) denklemi, düğümden düğüme aktarılan mesajlar olarak yorumlanabilir. $c_{i,l}$ bit düğümünün $\lambda_{i,l}$ mesajını *i.* denetim düğümüne aktardığı varsayılsın. *i.* denetim düğümü de c_n dışındaki $(k - 1)$ mesajı toplar, sonsal LLR'yi hesaplar ve bu mesajı *n.* bit düğümüne aktarır. *n.* bit düğümü de (3.37) denklemini kullanarak λ_n'yi hesaplar.

3.4 Sert Kararlı Kod Çözme: Bit Çevirme Algoritması

LDPC kodlar için sert kararlı kod çözme tekniği olarak bit çevirme algoritması (bit flip algorithm) kullanılmaktadır. Bu algoritmaya göre kod çözücü, tüm denetim denklemlerini ayrı ayrı hesaplar ve koşulu sağlanmayan denetim denklemi için ilgili bitlerin değerini değiştirir. Oluşan yeni bit değerlerine göre tekrar tüm denetim denklemleri ayrı ayrı hesaplanır ve bu işlemler tüm denetim denklemleri sağlanıncaya kadar veya daha önceden belirlenmiş bir iterasyon sayısına ulaşıncaya kadar sürdürülür.

Eğer alıcı maksimum iterasyon sayısına ulaştığında halen bir sonuç alınamıyorsa kod çözme hatası sezilir. Bu özellik de LDPC kodların hem hata sezme hem de hata düzeltme özelliklerine sahip olduğunun bir göstergesi olmaktadır.

Örnek olarak (7, 4) Hamming kodu ele alınsın. Bu koda ilişkin eşlik denetim matrisi aşağıda verilmiştir:

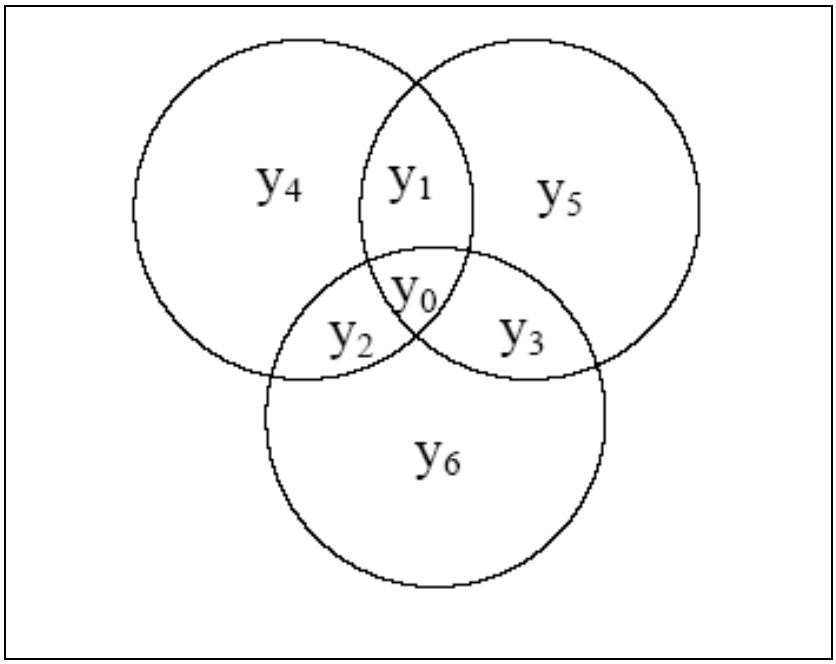
$$H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.39}$$

Matristeki ilk sütun ele alındığında algoritma aşağıdaki denklemin sağlanıp sağlanmadığını hesaplar:

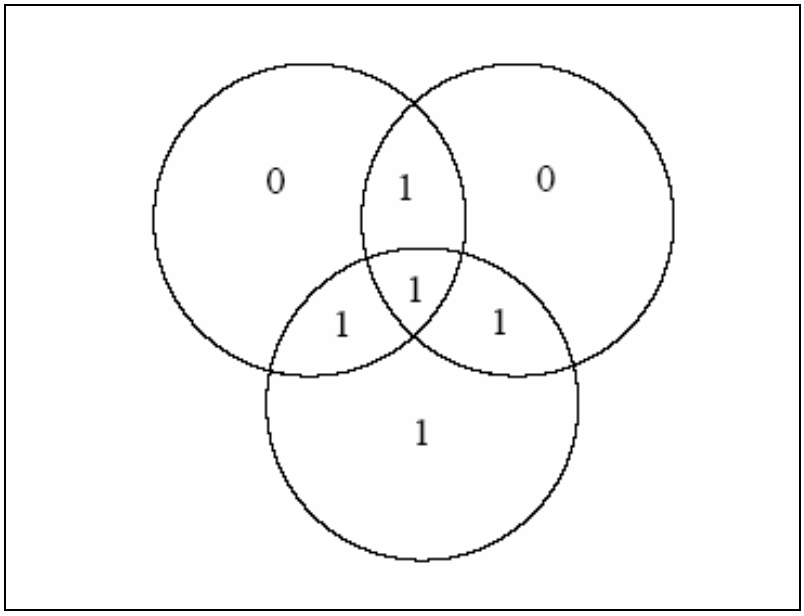
$$y_0 + y_1 + y_2 + y_4 = 0. \tag{3.40}$$

(3.39)'daki matrisin her üç sütunu da ele alındığında, hesaplamalara görsellik katılması amacıyla Şekil 3.2 verilmiştir. $\mathbf{y}H^T = 0$ denklem takımını sağlamak amacıyla Şekil 3.2'deki her çember için o çemberde yer alan bitlerin modülo 2 toplamının sıfır olması gerekmektedir. Buna örnek olarak da Şekil 3.3 verilmiştir.

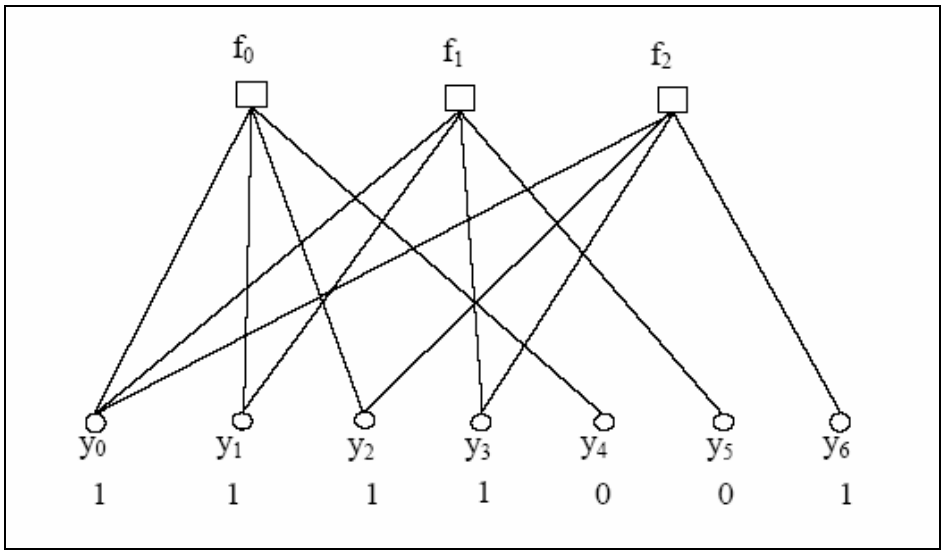
Şekil 3.2 ve Şekil 3.3 ele alındığında y_1'in hatalı bit olduğu ve eğer y_1'in değeri değiştirilirse (flipping) tüm eşlik denetim denklemlerinin sağlanacağı görülmektedir. Aynı görselleştirme Tanner grafı yardımıyla, ilgili şekillerdeki çemberler yerine denetim düğümleri getirilerek de yapılabilir. Buna örnek olarak Şekil 3.4 verilmiştir.



Şekil 3.2: (7, 4) Hamming Kodu için Görselleştirilmiş Denetim Denklemleri



Şekil 3.3: Denetim Denklemlerinde Bitlerin Bulunmasını Sağlayan Görselleştirilmiş Bit Çevirme Algoritması



Şekil 3.4: Tanner Grafı Yardımıyla Bit Çevirme Algoritmasının Uygulanması

3.5 Yumuşak Kararlı Kod Çözme: Mesaj Aktarma Algoritması

Mesaj aktarma algoritması (message passing algorithm), Tanner grafındaki dallar boyunca bir düğümden diğer düğüme mesaj aktaran bir kod çözme tekniğidir [24]. Bu algoritma kanı yayılımı (belief propagation) algoritması ve toplam – çarpım (sum - product) algoritması olarak da adlandırılmaktadır.

Kod sözcüklerinin blok uzunluğu arttıkça en büyük olabilirlikli ML kod çözme karmaşıklığı üstel olarak artmaktadır. Kanal sığasının (1 - δ) oranında iletim yapılmak istensin. Bu durumda mesaj aktarma algoritması iterasyon başına kod çözme karmaşıklığının blok uzunluğuyla $n\ln(1/\delta)$ ile orantılı olarak artmasından dolayı pratik olmaktadır. Genel olarak bu algoritma ML kod çözücüden daha güçsüz olarak nitelendirilmektedir. Eğer Tanner grafında çevrim bulunmuyorsa bu algoritma sonsal sınır olasılıklarını kesin olarak bulmaktadır. Çevrimlerin olduğu durumlarda ise doğru olasılıklara yaklaşıklık yapılmaktadır. Grafta çevrimler olsa bile algoritma iyi çalışmaktadır ve hala karmaşıklığı çok düşük olmaktadır.

Bu algoritma bir yumuşak kararlı kod çözme tekniğidir. Sert kararlı kod çözme tekniği daha az karmaşıklığına sahipken, yumuşak kararlı kod çözme tekniği daha iyi hata başarımı sağlamaktadır. Algoritmada aktarılan mesaj olasılıklardır. Çok büyük uzunluklu kod sözcüklerinden dolayı logaritmik olarak çalışmak avantaj

getirmektedir. Pratikte bu algoritma ya maksimum iterasyon sayısına ulaşıncaya kadar ya da olasılıkların kesinliğe yakın olmasına kadar sürdürülmektedir. Algoritma graftaki dalları gezmektedir. Dalların sayısı da eşlik denetim matrisinin seyrek olarak 1 içermesinden ötürü seyrektir. Dolayısıyla algoritmanın çalışma süresi de oldukça kısadır. Bu yapı dolayısıyla bellek gereksinimi de minimal olmaktadır. Graf temelli gerçekleştirme çok yüksek hızlarda bile LDPC kodunu kullanılır yapmaktadır. Eğer bilgi düğümü sayısı yeterince büyükse kod çözücünün planlanması zorlaşmaktadır.

Aktarılan mesajlar tamamen kullanılan kanala bağlı iken, algoritma tamamen kullanılan kanaldan bağımsızdır. Denetim düğümleri işlemci olarak, bilgi düğümleri de toplayıcı olarak düşünülebilir.

Bu algoritmada mesajlar graf üzerinden düğümden düğüme aktarılmaktadır. Düğümler ise gelen mesajları toplamakta ve giden mesajları üretmektedir. Algoritma adımları aşağıda verilmiştir:

Başlangıç:

(1) $u_{m,n}^{(0)} = 0, \forall m \in \{1,2,....M\}$ ve $n \in N_m$

(2) $\lambda_n^{(0)} = \frac{2}{c^2} r_n, \forall n \in \{1,2,....N\}$

İterasyon:

İterasyon sayacı $l = 1,2,....l_{maks}$

(3) Denetim düğümü güncelleme:

$m \in \{1,2,....M\}$ ve $n \in N_m$ için,

$$u_{m,n}^{(l)} = -2 \tanh^{-1}\left(\prod_{i \in N_m - n} \tanh\left(\frac{-\lambda_i^{(l-1)} + u_{m,i}^{(l-1)}}{2} \right) \right)$$

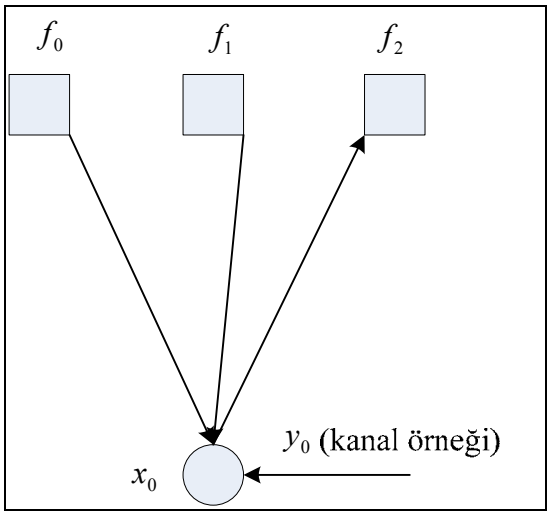
(4) Bilgi düğümü güncelleme:

$n \in \{1,2,....N\}$ için,

$$\lambda_n^{(l)} = \frac{2}{\sigma^2} r_n + \sum_{m \in M_n} u_{m,n}^{(l)}$$

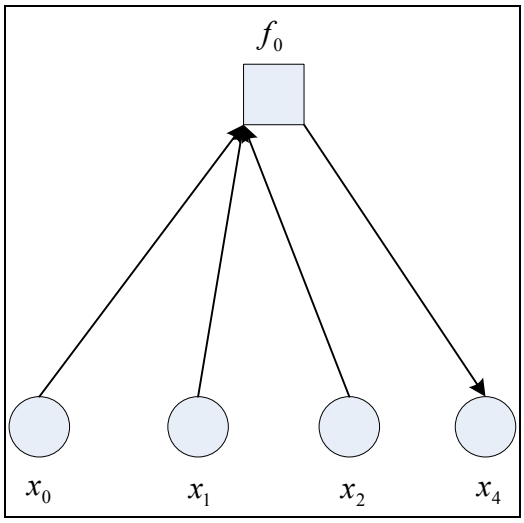
Burada $u_{m,n}^{(l)}$, m. denetim düğümünden n. bilgi düğümüne l. iterasyonda gönderilen mesajı göstermektedir. $\lambda_n^{(l)}$ ise l iterasyon sonundaki n. sonsal LLR'nin kestirimini ifade etmektedir. Ayrıca burada $M_n = \{m : H_{m,n} = 1\}$ ve $N_m = \{n : H_{m,n} = 1\}$ olmaktadır. $H_{m,n}$ ise eşlik denetim matrisinde m. satır ve n. sütunda yer alan değeri göstermektedir.

Şekil 3.5'te x_0'dan f_2'ye doğru olan durum incelenmiştir. Burada x_0'ın, f_2'nin halihazırda sahip olduğu bilgi dışında sahip olduğu tüm bilgi (yan bilgi) f_2'ye aktarılmaktadır.



Şekil 3.5: Bit Düğümünden Denetim Düğümüne Aktarılan Mesaj

Şekil 3.6'da ise denetim düğümünden bit düğümüne doğru olan durum incelenmiştir. Burada denetim düğümü, kendisine bağlı olan her bilgi düğümüne, bilgi düğümünün halihazırda sahip olduğu bilgi dışında, sahip olduğu tüm yan bilgiyi iletmektedir. Yani sadece $c_0 + c_1 + c_2 + c_4 = 0$ denklemiyle tutarlı olan bilgi gönderilmektedir.



Şekil 3.6: Denetim Düğümünden Bit Düğümüne Aktarılan Mesaj

3.5.1 Olasılık Bölgesinde Kod Çözücü

Mesaj aktarma algoritması hem olasılık bölgesinde hem de logaritmik bölgesinde kullanılmaktadır. Fakat logaritmik bölge, işlemlerde basitlik getirdiği için, daha fazla tercih edilmektedir.

Teorem 3.1: m adet bağımsız ikili sayıdan oluşan bir dizi ele alınsın. $\mathbf{a} = (a_1, a_2, ..., a_m)$ ve $\Pr[a_k = 1] = p_k$. Bu durumda bu dizinin çift ve tek sayıda 1 içerme olasılıkları sırasıyla,

$$\frac{1}{2} + \frac{1}{2}\prod_{k=1}^{m}(1-2p_k) \tag{3.41}$$

ve

$$\frac{1}{2} - \frac{1}{2}\prod_{k=1}^{m}(1-2p_k) \tag{3.42}$$

olmaktadır.

Tanıt: m üzerinden tümevarım yapılarak,

$$\Pr[\mathbf{a}=\text{çift}]=\Pr[a_1+a_2=0]=p_1p_2+(1-p_1)(1-p_2)=\frac{1}{2}+\frac{1}{2}(1-2p_1)(1-2p_2) \tag{3.43}$$

$$\Pr[\mathbf{a}=\text{tek}]=1-\Pr[\mathbf{a}=\text{çift}]=\frac{1}{2}-\frac{1}{2}(1-2p_1)(1-2p_2). \tag{3.44}$$

(3.41) denkleminin $m = L - 1$ için geçerli olduğu varsayılarak ve $z_L = a_1 + a_2 + \ldots + a_L$ olarak alınarak $\Pr[z_L = 0]$ için

$$\Pr[z_{L-1}+a_L=0]=\frac{1}{2}+\frac{1}{2}\left(1-\underbrace{2\Pr[z_{L-1}=1]}_{1-\prod_{k=1}^{L-1}(1-2p_k)}\right)(1-2p_L)=\frac{1}{2}+\frac{1}{2}\prod_{k=1}^{L}(1-2p_k) \tag{3.45}$$

elde edilir.

Teorem 3.2: $y_i = x_i + n_i$ olsun. Burada $n_i \approx N(0,\sigma^2)$ ve $\Pr[x_i = +1] = \Pr[x_i = -1] = \frac{1}{2}$ olmaktadır. Bu durumda

$$\Pr[x_i = x \mid y] = \frac{1}{1 + e^{\frac{-2yx}{\sigma^2}}},\ x \in \{\pm 1\} \tag{3.46}$$

eşitliği geçerli olacaktır.

Tanıt:

$$\Pr[x_i = x \mid y] = \frac{\Pr[y \mid x_i = x]\Pr[x_i = x]}{p(y)} \tag{3.47}$$

olarak tanımlı Bayes kuralı uygulandığında,

$$\Pr[x_i = x \mid y] = \frac{\frac{1}{2}e^{-(y-x)^2/2\delta^2}}{\frac{1}{2}e^{-(y-1)^2/2\delta^2} + \frac{1}{2}e^{-(y+1)^2/2\delta^2}} \tag{3.48}$$

elde edilir. (3.48) eşitliğinde biraz düzenleme yapılarak da

$$= \frac{e^{xy/\delta^2}}{e^{y/\delta^2} + e^{-y/\delta^2}} \tag{3.49}$$

$$= \frac{1}{e^{y(1-x)/\delta^2} + e^{-y(1+x)/\delta^2}} = \frac{1}{1 + e^{-2yx/\delta^2}} \tag{3.50}$$

eşitlikleri elde edilmiş olur.

Olasılık ve logaritmik bölgelerde algoritma adımlarını açıklarken aşağıdaki notasyon kullanılacaktır:

V_j = {denetim (c) düğümü j'ye bağlı olan bilgi (v) düğümleri}

$V_{j\backslash i}$ = {c düğümü j'ye bağlı olan v düğümleri} \ {i. v düğümü}

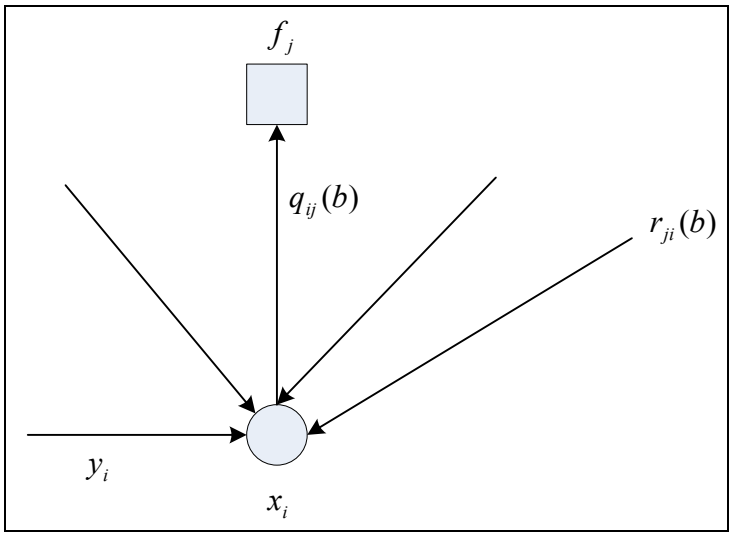
C_i = {i. v düğümüne bağlı c düğümleri}

$C_{i\backslash j}$ = {i. v düğümüne bağlı c düğümleri} \ {j. c düğümü}

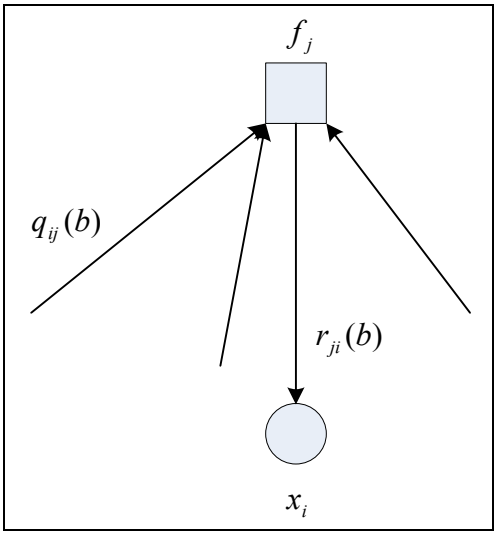
$P_i = \Pr[c_i = 1 \mid y_i]$

$q_{ij}(b)$ = i. bilgi düğümünden j. denetim düğümüne aktarılan yan bilgi. Bu bilgide kanal örneği y_i bulunmamakta olup $c_i = b, b \in \{0,1\}$ olasılığı göz önüne alınmaktadır. Grafiksel anlatım amacıyla Şekil 3.7 incelenebilir.

$r_{ji}(b)$ = j. denetim düğümünden i. bilgi düğümüne aktarılan bilgi. Grafiksel anlatım amacıyla Şekil 3.8 incelenebilir.



Şekil 3.7: $q_{ij}(b)$ 'nin Grafiksel Anlatımı



Şekil 3.8: $r_{ji}(b)$ 'nin Grafiksel Anlatımı

Olasılık bölgesinde mesaj aktarma algoritmasının adımları aşağıda verilmiştir. Buradaki döngüler $h_{ij} = 1$ eşitliğini sağlayan her i ve j için gerçekleşmelidir.

(0) <u>Başlangıç:</u>

$$q_{ij}(0) = 1 - P_i = \Pr\left[x_i = +1 \mid y_i\right] = \frac{1}{1 + e^{\frac{-2y_i}{\sigma^2}}}$$

$$q_{ij}(1) = P_i = \Pr\left[x_i = -1 \mid y_i\right] = \frac{1}{1 + e^{\frac{2y_i}{\sigma^2}}}$$

(1) $r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i' \in V_{j \setminus i}} \left(1 - 2q_{i'j}(1)\right)$

$r_{ji}(1) = 1 - r_{ji}(0)$

(2) $q_{ij}(0) = K_{ij}\left(1 - P_i\right)\prod_{j' \in C_{i \setminus j}} r_{j'i}(0)$

$$q_{ij}(1) = K_{ij} P_i \prod_{j' \in C_{i \setminus j}} r_{j'i}(1)$$

Buradaki K_{ij} sabitleri $q_{ij}(0) + q_{ij}(1) = 1$ denklemini sağlayacak şekilde seçilmektedir.

(3) Her i için,

$$Q_i(0) = K_i\left(1 - P_i\right)\prod_{j \in C_i} r_{j'i}(0)$$

$$Q_i(1) = K_i P_i \prod_{j \in C_i} r_{j'i}(1)$$

Bu eşitliklerde yer alan K_i sabitleri $Q_i(0) + Q_i(1) = 1$ denklemini sağlayacak şekilde seçilmektedir.

(4) Her i için,

$$\hat{c}_i = \begin{cases} 1, Q_i \geq 0.5 \\ 0, Q_i < 0.5 \end{cases}$$

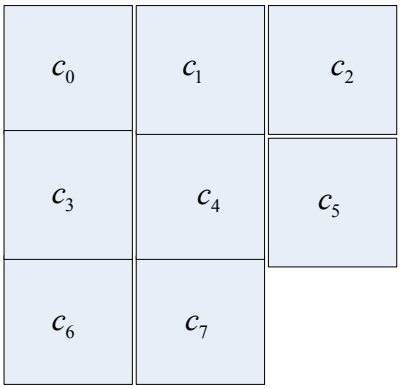
Eğer $\left(\hat{\mathbf{c}}H^T = \mathbf{0}\right)$ veya (iterasyon sayısı = maksimum iterasyon sayısı)

Dur

Değilse

(1). adıma geri dön.

Olasılık bölgesinde mesaj aktarma algoritmasının C programlama dilinde yazılmış şekli EK – A'da verilmiştir. Örnek olarak minimum uzaklığının değeri 4 olan (8, 4) çarpım kodu (product code) Şekil 3.9'daki gibi verilsin:



Şekil 3.9: (8, 4) Çarpım Kodu

Bu kod için aşağıdaki denklem takımı geçerli olmaktadır:

$$\begin{aligned} c_2 &= c_0 + c_1, \\ c_5 &= c_3 + c_4, \\ c_6 &= c_0 + c_3, \\ c_7 &= c_1 + c_4. \end{aligned} \tag{3.51}$$

Koda ilişkin ilgili eşlik denetim matrisi ise

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.52}$$

biçimindedir. Bu örnek için kod sözcüğü

$$\mathbf{c} = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 \end{matrix} \tag{3.53}$$

olarak seçilsin. Bu kod sözcüğüne ilişkin alınan sözcük dizisi de aşağıdaki gibi olsun:

$$\mathbf{y} = \begin{matrix} y_0 & y_1 & y_2 \\ y_3 & y_4 & y_5 \\ y_6 & y_7 & \end{matrix} = \begin{matrix} +0.2 & +0.2 & -0.9 \\ +0.6 & +0.5 & -1.1 \\ -0.4 & -1.2 & \end{matrix}. \tag{3.54}$$

y_0 ve y_4'te işaret hatalarının da bulunduğu görülmektedir. Bu koda ve ilişkin kod sözcüğüne algoritma uygulandığında aşağıdaki sayısal değerler elde edilir:

İterasyon: 1
Q = 0.7686 0.8694 0.9647 0.5076 0.7426 0.9479 0.7199 0.9853
c = 1 1 1 1 1 1 1 1

İterasyon: 2
Q = 0.1751 0.1836 0.9668 0.2330 0.4637 0.9722 0.8305 0.9919
c = 0 0 1 0 0 1 1 1

İterasyon: 3
Q = 0.7232 0.4609 0.9667 0.6308 0.8091 0.9498 0.6802 0.9909
c = 1 0 1 1 1 1 1 1

İterasyon: 4
Q = 0.5307 0.2683 0.9731 0.1477 0.4095 0.9811 0.8392 0.9922
c = 1 0 1 0 0 1 1 1

İterasyon: 5
Q = 0.7564 0.5799 0.9672 0.3425 0.7573 0.9558 0.8102 0.9896
c = 1 1 1 0 1 1 1 1

İterasyon: 6
Q = 0.4544 0.2089 0.9736 0.2136 0.6243 0.9686 0.8412 0.9924
c = 0 0 1 0 1 1 1 1

İterasyon: 7

Q = 0.7399 0.3381 0.9692 0.4086 0.7869 0.9567 0.7754 0.9923

c = 1 0 1 0 1 1 1 1

Görüldüğü gibi 7. iterasyon sonucunda istenilen kod sözcüğü elde edilebilmiştir.

3.5.2 Logaritmik Bölgede Kod Çözücü

Olasılık bölgesinde mesaj aktarma algoritmasının, fonksiyonlarının çarpım içermesinden dolayı gerçekleştirilmesi daha zordur. Halbuki logaritmik bölgede çarpım işlemleri yerini toplama işlemine bırakır. Dolayısıyla bu bölgede gerçekleştirme daha kolay olmaktadır.

Bu bölgeye özgü notasyon aşağıda listelenmiştir:

$$L(c_i) = \log \frac{\Pr[x_i = +1 \mid y_i]}{\Pr[x_i = -1 \mid y_i]},$$

$$L(r_{ji}) = \log \frac{r_{ji}(0)}{r_{ji}(1)},$$

$$L(q_{ij}) = \log \frac{q_{ij}(0)}{q_{ij}(1)},$$

$$L(Q_i) = \log \frac{Q_{ij}(0)}{Q_{ij}(1)}. \tag{3.55}$$

Olasılık bölgesinde verilen algoritmada (0). adım olan başlangıç adımı aşağıdaki eşitliğe dönüşür:

$$L(q_{ij}) = L(c_i) = \log \frac{\left(1 + e^{\frac{-2y_i}{\sigma^2}}\right)^{-1}}{\left(1 + e^{\frac{2y_i}{\sigma^2}}\right)^{-1}} = \frac{2y_i}{\sigma^2}. \tag{3.56}$$

(1). adım için aşağıdaki eşitlikler geçerli olacaktır:

$$2r_{ji}(0) = 1 + \prod_{i' \in V_{j \setminus i}} \left(1 - 2q_{i'j}(1)\right) \tag{3.57}$$

$$\Rightarrow 1 - 2r_{ji}(1) = \prod_{i' \in V_{j \setminus i}} \left(1 - 2q_{i'j}(1)\right) \tag{3.58}$$

$$L\left(r_{ji}\right) = 2\tanh^{-1}\left(\prod_{i' \in V_{j \setminus i}} \tanh\left(\frac{1}{2}L\left(q_{i'j}\right)\right)\right). \tag{3.59}$$

Yukarıdaki ifadelerde halen çarpım işlemleri görülmektedir. Bunlar

$$L\left(q_{i'j}\right) = \alpha_{ij}\beta_{ij} \tag{3.60}$$

$$\alpha_{ij} = sign\left(L\left(q_{i'j}\right)\right) \tag{3.61}$$

$$\beta_{ij} = \left|L\left(q_{i'j}\right)\right| \tag{3.62}$$

tanımları kullanılarak

$$L\left(r_{ji}\right) = \left(\prod_{i'} \alpha_{ij}\right) 2\tanh^{-1}\log^{-1}\sum_{i'}\log\tanh\left(\frac{1}{2}\beta_{i'j}\right) \tag{3.63}$$

$$= \left(\prod_{i'} \alpha_{ij}\right) \phi\left(\sum_{i' \in v_{j \setminus i}} \phi\left(\beta_{i'j}\right)\right) \tag{3.64}$$

denklemlerine dönüştürülür. (3.64) eşitliğinde yer alan ϕ fonksiyonu, (3.20)'deki f fonksiyonu ile aynıdır.

(2). adım içinse

$$L\left(q_{ij}\right) = L\left(c_i\right) + \sum_{j' \in C_{i \setminus j}} L\left(r_{j'i}\right) \tag{3.65}$$

eşitliği geçerli olmaktadır. (3). adım da aynı mantıkla logaritmik bölgeye taşınabilir.

Yukarıdaki eşitliklere dayanarak logaritmik bölgede mesaj aktarma algoritmasının adımları aşağıda verilmiştir. Buradaki döngüler $h_{ij} = 1$ eşitliğini sağlayan her *i* ve *j* için gerçekleşmelidir:

(0) $L(q_{ij}) = L(c_i) = \frac{2y_i}{c^2}$

(1) $L(r_{ji}) = \left(\prod_{i' \in V_{j \setminus i}} \alpha_{ij} \right) \phi \left(\sum_{i' \in v_{j \setminus i}} \phi(\beta_{i'j}) \right)$

(2) $L(q_{ij}) = L(c_i) + \sum_{j' \in C_{i \setminus j}} L(r_{j'i})$

(3) $L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji})$

(4) Her i için,

$$\hat{c}_i = \begin{cases} 1, L(Q_i) < 0 \\ 0, L(Q_i) \geq 0 \end{cases}$$

Eğer $(\hat{\mathbf{c}} H^T = \mathbf{0})$ veya (iterasyon sayısı = maksimum iterasyon sayısı)

Dur

Değilse

(1). adıma geri dön.

Logaritmik bölgede mesaj aktarma algoritmasının C programlama dilinde yazılmış şekli EK – B'de verilmiştir.

Logaritmik bölgede mesaj aktarma algoritması Şekil 3.9'da verilen çarpım koduna uygulandığında 7. iterasyon sonucunda istenilen kod sözcüğünün elde edildiği görülmektedir. Buna ilişkin sayısal değerler aşağıda verilmiştir ($c^2 = 0.5$ olarak alınmıştır):

İterasyon: 1

LQ = -1.2002 -1.8953 -3.3092 -0.0306 -1.0597 -2.9008 -0.9439 -4.2042

c = 1 1 1 1 1 1 1 1

İterasyon: 2

LQ = 1.5499 1.4922 -3.3721 1.1913 0.1455 -3.5547 -1.5889 -4.8064

c = 0 0 1 0 0 1 1 1

İterasyon: 3

LQ = -0.9605 0.1568 -3.3680 -0.5354 -1.4442 -2.9399 -0.7545 -4.6958

c = 1 0 1 1 1 1 1 1

İterasyon: 4

LQ = -0.1229 1.0031 -3.5876 1.7531 0.3659 -3.9473 -1.6520 -4.8420

c = 1 0 1 0 0 1 1 1

İterasyon: 5

LQ = -1.1331 -0.3222 -3.3854 0.6521 -1.1379 -3.0733 -1.4512 -4.5529

c = 1 1 1 0 1 1 1 1

İterasyon: 6

LQ = 0.1830 1.3318 -3.6083 1.3031 -0.5077 -3.4307 -1.6673 -4.8708

c = 0 0 1 0 1 1 1 1

İterasyon: 7

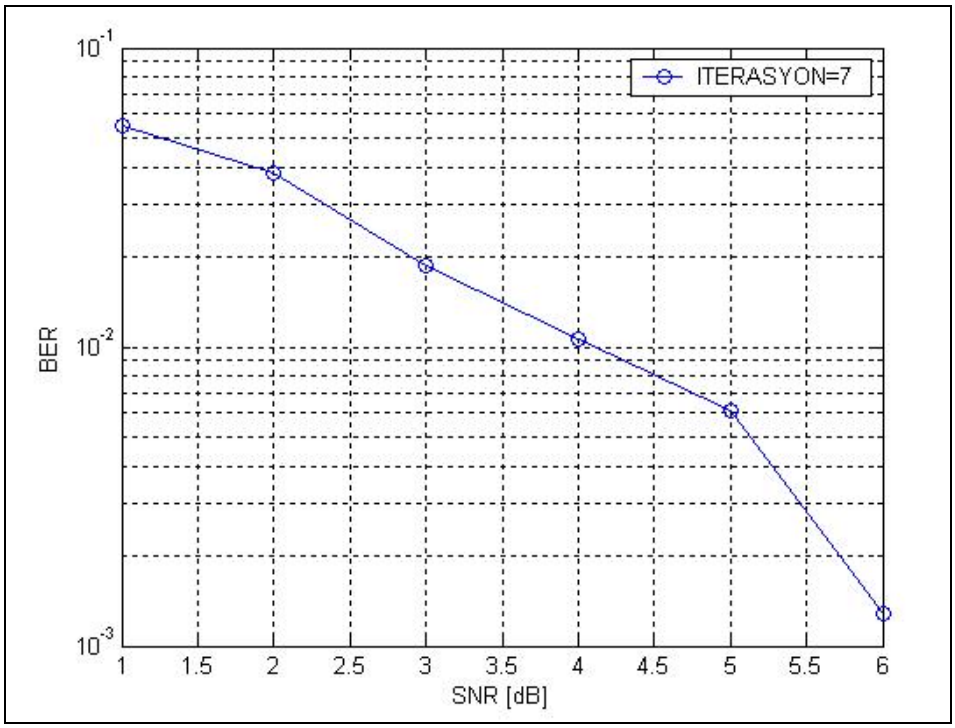
LQ = -1.0455 0.6718 -3.4495 0.3697 -1.3064 -3.0952 -1.2390 -4.8631

c = 1 0 1 0 1 1 1 1

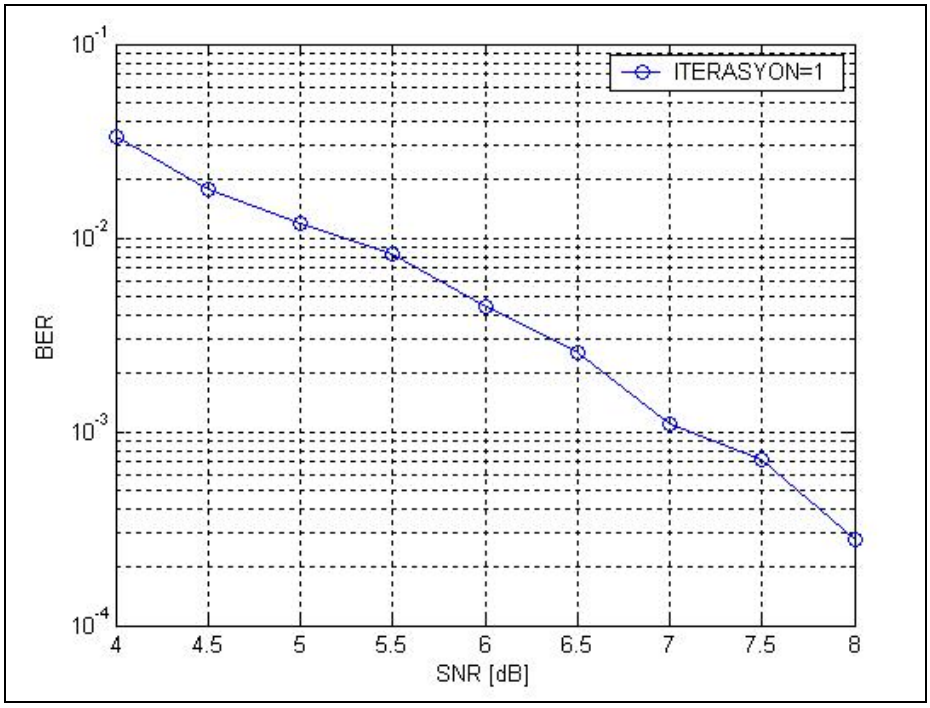
Şekil 3.10'da (7, 4) Hamming koduna uygulanmış logaritmik bölgede mesaj aktarma algoritmasının bit hata başarımı verilmiştir.

Şekil 3.11'de ise (20, 3, 4) Gallager koduna uygulanmış logaritmik bölgede mesaj aktarma algoritmasının bit hata başarımı verilmiştir.

Şekillerde yer alan SNR değeri, E_b/N_0 olarak tanımlıdır. Burada E_b bit başına enerjiyi, N_0 da varyansın 2 katını göstermektedir ($N_0/2 = \sigma^2$).



Şekil 3.10: (7, 4) Hamming Kodunun Logaritmik Bölgede Mesaj Aktarma Algoritması ile Bit Hata Başarımı



Şekil 3.11: (20, 3, 4) Gallager Kodunun Logaritmik Bölgede Mesaj Aktarma Algoritması ile Bit Hata Başarımı

4. LDPC KODLAR İÇİN VERİCİ TASARIMI

Bu bölümde düzenli ve düzensiz LDPC kod tasarlama teknikleri incelenmektedir.

4.1 LDPC Kodların Yarı Rastgele Üretimi

Aşağıda artan algoritma karmaşıklığına göre sıralanmış, LDPC kodlar için eşlik denetim matrisi *H*'nın üretilme yolları listelenmiştir [25][26]:

1) *H* matrisi tüm elemanları sıfır olan bir matristen üretilmeye başlanır ve her bir sütunda rastgele seçilmiş *j* adet bitin tersi alınır. Dolayısıyla oluşan LDPC kodu düzensiz olacaktır.
2) *H* matrisi rastgele olarak *j* ağırlıklı sütunların yaratılmasıyla üretilir.
3) *H* matrisi *j* ağırlıklı sütunlara ve mümkün olduğu kadar aynı ağırlığa sahip satırlara sahip olacak şekilde üretilir.
4) *H* matrisi *j* ağırlıklı sütunlara ve *k* ağırlıklı satırlara sahip olacak şekilde üretilir ve herhangi iki sütunun birbiri üzerine izdüşümünde birden fazla aynı bitin olması engellenir. Oluşan LDPC kodu düzenli olacaktır.
5) *H* matrisi 4. maddede yer alan algoritmaya göre üretilir, ayrıca üretim esnasında kısa uzunluğa sahip çevrimler yasaklanır.
6) *H* matrisi 5. maddede yer alan algoritmaya göre üretilir, ayrıca $H = [H_1 : H_2]$ ve H_2'nin tersi alınabilecek şekilde veya en azından *H* matrisi tam ranklı olacak biçimde oluşturulur.

Genel olarak *H* matrisinin üretilmesinde 5. maddede yer alan algoritma kullanılmaktadır. Bu algoritmanın adımları aşağıda verilmiştir:

(1) Kod parametreleri olan *n, m, j, k* ve $L_{\min}$ seçilir. Burada $L_{\min}$ minimum çevrim uzunluğu olmaktadır. Böylece *H* matrisi, her sütununda *j* adet ve her satırında *k* adet 1 içeren (*n x m*) boyutunda bir matris olacaktır.

(2) Sütun sayacı sıfırlanır, $i_c = 0$.

(3) Oluşturulmaya başlanan *H* matrisinin i_c sütunu için *j* ağırlıklı sütun vektörü üretilir ve o sütuna yerleştirilir.

(4) Bu noktada eğer her satırın ağırlığı *k*'dan küçük veya eşitse, iki sütunun birbiri üzerine izdüşümü sonrası birbirinin aynısı olan bit sayısı 1'den küçük veya eşitse ve tüm çevrim uzunlukları $L_{\min}$'den büyük veya eşitse i_c 1 birim arttırılır, $i_c = i_c + 1$.

(5) Eğer $i_c = n$ ise algoritma durur. $i_c = n$ koşulu sağlanmadıysa (3). adıma geri dönülür.

Bu algoritmada üretim, yarı rastgele olduğundan algoritmanın çalışması saatler boyu sürebilmektedir. Hatta hiçbir zaman sonlanmayabilir. Bu durumda ya yeni bir başlangıç değeriyle ya da yeni kod parametre değerleriyle algoritma yeniden başlatılmalıdır.

Ayrıca inşa edilen kodun düzenli olması garantisi de yoktur. Bu koşulu sağlamak amacıyla algoritma adımlarına birkaç yeni adımın eklenmesi gerekmektedir.

4.2 Düzenli LDPC Kod Tasarımı: Sözde Rastgele Üretim Algoritması

Sözde rastgele (pseudo random) üretim algoritmasının temel avantajı, Tanner grafındaki dalların bellekte kaydedilerek değil de özyineleme (recursion) kullanılarak, donanım gerçekleştirmesinin kolaylaştırılması olarak gösterilebilir [9].

Bu algoritma ile çevrim uzunluğu 4'ten daha yüksek düzenli LDPC kodlar oluşturulabilmektedir. Rastgele yapılan üretimde 4 uzunluklu çevrimlerin olma olasılığı oldukça yüksek olmaktadır. Bu dezavantajdan kurtulmak amacıyla algoritma adımlarına bir kısıtlama daha eklenmiştir. Düzenli kodlar için ilgili eşlik denetim matrisindeki her bir sütunun ağırlığı olan *j*'nin 3'ten büyük olması

gerektiği tanıtlanmıştır [1][2]. $j \geq 3$ koşulu sağlandığında minimum uzaklık değeri çok iyi olan kodlar tasarlanabilmektedir.

Düzensiz LDPC kodların düzenli LDPC kodlara göre daha üstün oldukları [6]'da gösterilmiştir. Buna rağmen düzenli LDPC kodların, düzgün yapılarının kod çözücü tasarımında sunduğu basitlik nedeniyle donanım gerçeklemelerinde avantajları bulunmaktadır. Ayrıca düşük kod sözcük uzunlukları için düzensiz LDPC kodların küçük minimum uzaklığa sahip olma olasılıkları daha yüksek olmaktadır. Bu gerekçelerle düzenli LDPC kodlar tercih edilmektedir. Mesaj aktarma algoritması altında düzenli LDPC kodların hata başarımlarının maksimum olduğu durum $j = 3$ olduğu durumdur [6]. Bu algoritma da bu eşitlik göz önünde bulundurularak ve cebirsel olarak tasarlanmıştır.

Kod sözcük uzunluğu N, eşlik denetim matrisindeki herbir sütunun ağırlığı p ve herbir satırın ağırlığı q olarak tanımlanmış düzenli bir (N, p, q) LDPC kodunun cebirsel olarak inşa edilmesi problemi aslında $A = A_0, A_1, ..., A_{M-1}$ dizisinin bulunması problemi olmaktadır. Burada $M = N.q$, graftaki dal sayısını göstermektedir. Ayrıca A_i, $\lfloor i/p \rfloor$. bit düğümü ile $\lfloor A_i/q \rfloor$. denetim düğümü arasındaki dalı göstermektedir. $\lfloor . \rfloor$ operatörü, işleme giren değeri kendisine en yakın fakat ondan büyük olmayan tamsayıya eşitlemektedir. Dizi elemanları arasındaki özyineleme bağıntısı

$$A_{n+1} = (aA_n + b) \bmod M, \quad 0 \leq n \leq M - 2 \tag{4.1}$$

biçimindedir. Burada A_0, 0 ile (M - 1) arasında herhangi bir tamsayı olabilmektedir. (4.1)'deki sabitler olan a ve b'nin seçimi ise aşağıdaki kısıtlamalara bağlıdır:

C1) $a < M$, $b < M$.

C2) b, M'ye göre asal sayıdır.

C3) (a - 1), M'yi bölebilen her asal p sayısı için p'nin çarpımından oluşur.

C4) (opsiyonel) a, M'ye göre asal sayıdır.

LDPC kodun iki yönlü grafını tasarlamak amacıyla (4.1) eşitliği, n = 0, 1, ..., M - 2 için tekrarlanmaktadır. N = 2048, K = 1024, p = 3 ve q = 6 parametrelerine sahip R = 0.5 oranlı düzenli LDPC kodu tasarlanmak istensin. İlgili graftaki dal sayısı $M = N.p$ = 2048.3 = 6144 olacaktır. Yukarıda verilen kısıtlamalara uygun olarak da a = 49, b = 11 ve A_0 = 0 seçilsin. Bu durumda (4.1)'de ifade edilen dizi elemanları A = 0, 11, 550, 2385, 140, 727,…. olacaktır. Bu bilginin ışığında ilgili Tanner grafı da aşağıdaki şekilde elde edilmektedir:

- Birinci bit düğümü, $\lfloor 0/6 \rfloor = 0$, $\lfloor 11/6 \rfloor = 1$ ve $\lfloor 550/6 \rfloor = 91$ numaralı denetim düğümlerine bağlıdır.
- Benzer şekilde ikinci bit düğümü de $\lfloor 2385/6 \rfloor = 397$, $\lfloor 140/6 \rfloor = 23$ ve $\lfloor 727/6 \rfloor = 121$ numaralı denetim düğümlerine bağlıdır.

Oluşturulan eşlik denetim matrisinin sistematik biçimde oluşturulmaması nedeniyle kodlama için üreteç matrisinin belirlenmesi amacıyla Gauss eliminasyon yöntemi uygulanmalıdır.

Bu tekniğe [9]'un kattığı en önemli yenilik ise 2 veya 4 uzunluklu çevrim uzunluklarını yasaklayan a ve b sabitlerinin seçimi konusudur. Bu kısıtlamaların da algoritmada yer alması amacıyla C1'den C4'e kadar olan kısıtlamalara ek olarak aşağıdaki kısıtlamaların sağlanması gerekmektedir:

C5) (a - 1) M'yi böler.

C6) $b > 2(q - 1)$.

C7) $(a - 1) > 4b + 2(q - 1)$.

C8) $X = \left(M/(a - 1)\right) \geq q$ veya eşdeğer olarak $(a - 1) \leq \left(M/q\right)$.

C9) $((a + 1)i) \bmod X = 0$ veya eşdeğer olarak $\left(\left(a^2 - 1\right)i\right) \bmod M = 0$ eşitliklerini sağlayan ve $1 \leq i \leq (q - 1)$ aralığında bulunan hiçbir i tamsayısı yoktur.

Burada eğer X çift sayı ise, C7 kısıtlaması $(a - 1) > 3b + 2(q - 1)$ kısıtlamasına karşılık gelmektedir.

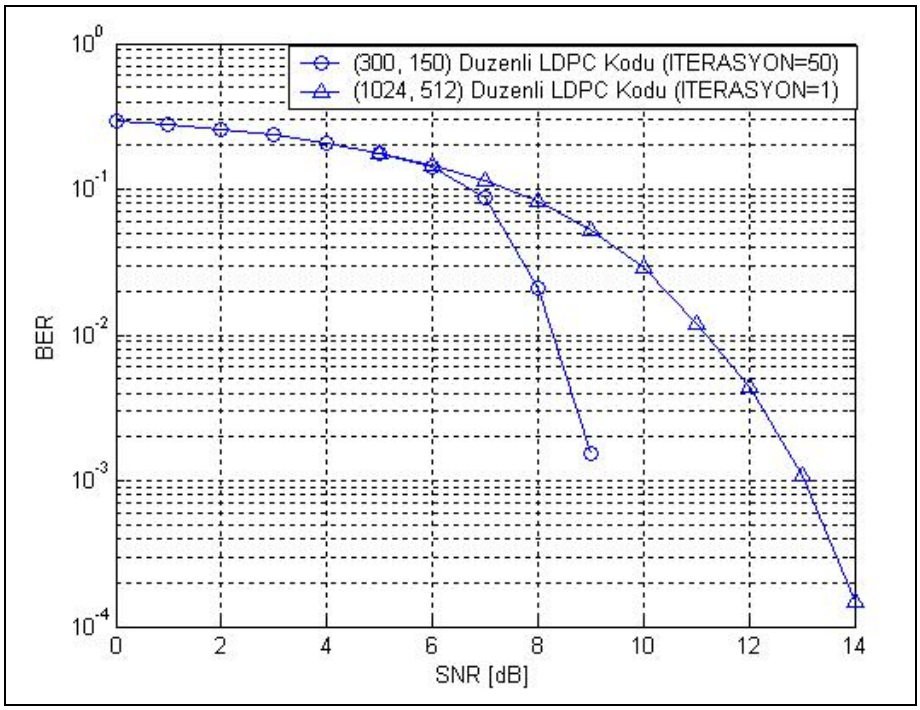
Test edilen kodlara dayanarak a ve b sabitlerinden düşük değere sahip olanlar daha iyi başarım göstermişlerdir. Ayrıca bit hata başarımının A_0 ilk koşuluna bağlı olduğu görülmektedir. Bu yöntem sayesinde kod çözücüdeki bellek gereksinimi önemli ölçüde düşmektedir.

Sözde rastgele üretim algoritmasının C programlama dilinde yazılmış biçimi EK – C'de verilmiştir.

$N = 300,\ K = 150,\ j = 3,\ k = 6,\ R = 0.5,\ A_0 = 0,\ a = 61,\ b = 11$ parametrelerine sahip düzenli LDPC kodu üretilmek istensin. Bu durumda A = 0, 1, 113, 35, 67, 59, 11, 72, 94, 76, 18,…. şeklinde olacaktır.

$N = 1024,\ K = 512,\ j = 3,\ k = 6,\ R = 0.5,\ A_0 = 0,\ a = 49,\ b = 11$ parametrelerine sahip düzenli LDPC kodu üretilmek istensin. Bu durumda A = 0, 1, 91, 397, 23, 121, 307, 196, 430, 112, 394,..... şeklinde olacaktır.

Yukarıda tanımlı 2 kodun 4-lü darbe genlik modülasyonu (4 PAM) ile AWGN kanaldan iletildikten sonra mesaj aktarma algoritması için bit hata başarımı Şekil 4.1'de verilmiştir. Ayrıca ilk verilen kodun BPSK modülasyonu ile AWGN kanaldan iletildiğinde bit hata başarımı Şekil 4.2'de yer almaktadır.



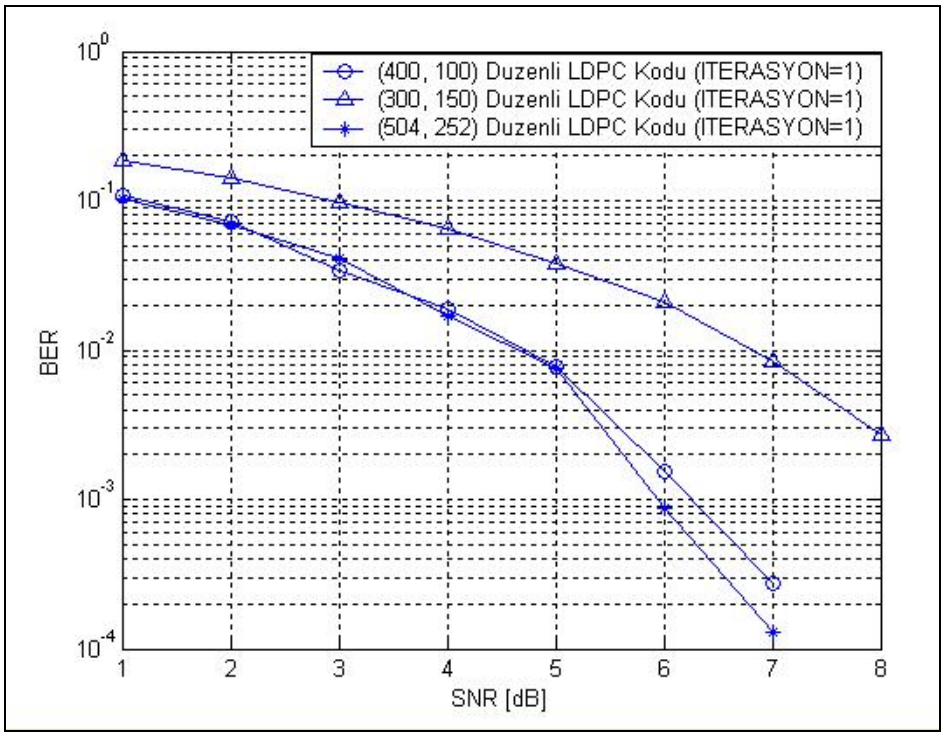
Şekil 4.1: 4 PAM Modülasyonlu (300, 150) ve (1024, 512) Düzenli LDPC Kodlarının Bit Hata Başarımı

Burada iterasyon sayısının önemi vurgulanmaktadır ve aynı kodlama oranına sahip 2 düzenli LDPC kodu incelenmiştir. Şekil 4.1 incelendiğinde, kod sözcük uzunluğu daha büyük olmasına rağmen (yani daha iyi başarım göstereceği beklenmesine rağmen) daha az iterasyon yapıldığı için (1024, 512) düzenli LDPC kodu, (300, 150) düzenli LDPC kodundan daha kötü başarım göstermiştir. 0.002'lik bit hata başarımına ulaşmak için (300, 150) kodunda yaklaşık 8 dB'e gereksinim varken bu değer (1024, 512) kodunda yaklaşık 12.5 dB'e çıkmaktadır.

$N = 400,\ K = 100,\ j = 3,\ k = 4,\ R = 0.25,\ A_0 = 0,\ a = 61,\ b = 7$ parametrelerine sahip düzenli LDPC kodu üretilmek istensin. Bu durumda A = 0, 1, 108, 20, 37, 158, 85, 117,…. şeklinde olacaktır.

$N = 504,\ K = 252,\ j = 3,\ k = 6,\ R = 0.5,\ A_0 = 0,\ a = 85,\ b = 11$ parametrelerine sahip düzenli LDPC kodu üretilmek istensin. Bu durumda A = 0, 1, 157, 47, 7, 121, 221, 138, 210, 16, 144,..... şeklinde olacaktır.

Yukarıda tanımlı kodların BPSK modülasyonu ile AWGN kanaldan iletildikten sonra mesaj aktarma algoritması ile çözülmesi durumunda bit hata başarımı Şekil 4.2'de verilmiştir.



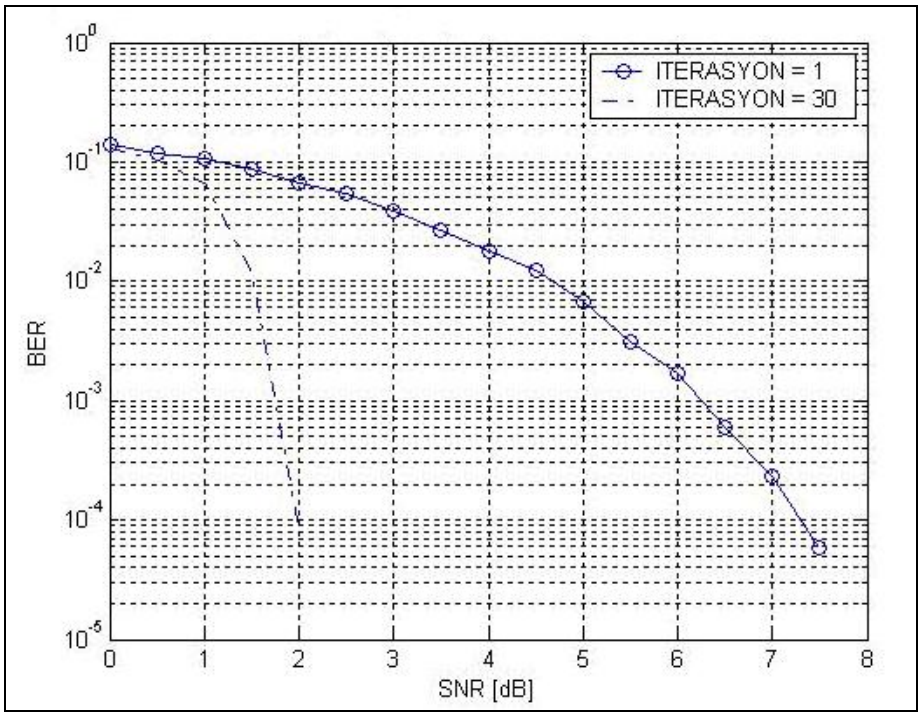
Şekil 4.2 : BPSK Modülasyonlu (300, 150), (400, 100) ve (504, 252) Düzenli LDPC Kodlarının Bit Hata Başarımı

Şekil 4.2 incelendiğinde R = 0.25 oranına sahip (400, 100) LDPC koduyla R = 0.5 oranına sahip (300, 150) LDPC kodu karşılaştırılırsa, aynı iterasyon sayısı için (400, 100) LDPC kodu hem kod sözcük uzunluğu daha büyük olduğu için hem de kodlama oranı daha düşük olduğu için diğer koda göre daha iyi başarım göstermiştir. 0.01 bit hata başarımına ulaşmak için (400, 100) LDPC kodunda yaklaşık 4.5 dB'e gereksinim varken bu değer (300, 150) LDPC kodunda yaklaşık 6.75 dB olmaktadır.

Farklı kodlama oranlarına sahip 2 kodun aynı bit hata başarımını vermesi istenirse, yüksek kodlama oranına sahip olan kod için ya iterasyon sayısı ya da kod sözcük uzunluğu arttırılır. Şekil 4.2'de kod sözcük uzunluğunun arttırılması durumu incelenmiştir. R = 0.5 kodlama oranlı (504, 252) LDPC kodu, kodlama oranı daha yüksek olmasına rağmen (400, 100) LDPC koduna yakın bir bit hata başarımı göstermiştir. Bunun nedeni kod sözcük uzunluğunun arttırılmasıdır. SNR değeri 4 dB'e kadar (400, 100) LDPC kodu biraz daha iyi başarım göstermiş fakat 4 dB'den sonra (504, 252) LDPC kodu daha iyi başarım göstermeye başlamıştır.

N = 2048, K = 1024, j = 3, k = 6, R = 0.5, $A_0 = 0$, a = 49, b = 11 parametrelerine sahip düzenli LDPC kodu üretilmek istensin. Bu durumda A = 0, 1, 91, 397, 23, 121, 819, 196, 430, 624, 906,..... şeklinde olacaktır. Bu kodun BPSK modülasyonu ile AWGN kanaldan iletildiğinde bit hata başarımı Şekil 4.3'te verilmiştir.

Şekil 4.3'te aynı kod sözcük uzunluğuna ve kodlama oranına sahip 2 farklı kod için iterasyon sayısı parametre olarak alınmış ve bit hata başarımları karşılaştırılmıştır. Buna göre iterasyon sayısı düşük olan LDPC kodu daha düşük başarım göstermiştir. 0.0001 bit hata başarımına ulaşmak için 30 iterasyon yapılmış LDPC kodda yaklaşık 2 dB'e gereksinim varken bu değer 1 iterasyon yapılmış kod için yaklaşık 7.5 dB'e çıkmaktadır. Yani yaklaşık 5.5 dB'lik kazanç sağlamak için 29 kez iterasyon yapmak gereklidir.



Şekil 4.3: BPSK Modülasyonlu (2048, 1024) Düzenli LDPC Kodunun Bit Hata Başarımı

Sığa hesabı içinse (4.2)'de verilen Shannon sınırından faydalanılabilir. (4.2)'de R kodlama oranını göstermektedir. Bu formülle hesaplanan değer, hata olasılığını 0'a götüren eşik SNR değeri olmaktadır:

$$\mathrm{SNR}_{\mathrm{e}}\,[dB] = 10\log_{10}\left(\frac{2^{2R}-1}{2R}\right). \tag{4.2}$$

4.3 Düzensiz LDPC Kod Tasarımı: Bit Doldurma Algoritması

Yüksek hızlarda çalışan ve yüksek minimum çevrim uzunluğuna sahip olan LDPC kodlar tasarlamak amacıyla bit doldurma algoritması (bit filling algorithm) kullanılmaktadır [11][12]. Bu algoritma hem düzenli hem de düzensiz LDPC kodlar üretebilmektedir.

Yüksek minimum çevrim uzunluğunun istenmesinin amacı, LDPC kodların kod çözme aşamasında mesaj aktarma algoritmasıyla çözülmesi esnasında birbirinden bağımsız iterasyon sayısının minimum çevrim uzunluğuyla orantılı

olmasından dolayıdır. Ayrıca minimum uzaklık, minimum çevrim uzunluğu arttıkça artmaktadır.

m, N, b ve $a_1, a_2, ..., a_N$ pozitif tamsayıları verilmiş olsun. Bu algoritma olabilecek en yüksek minimum çevrim uzunluğuna sahip ve *H* matrisinde *j*. sütunda tam olarak a_j adet 1 içeren, $1 \leq j \leq n$ ve her satırda en fazla *b* adet 1 olan (*m x N*) boyutlu eşlik denetim matrisi *H*'yı üretmektedir. Bu algoritma yardımıyla, MacKay [27] tarafından bulunan kodlara göre daha yüksek hızlara sahip kodlar üretilebilmiştir.

Buradaki temel tasarım ilkesi, grafa bitleri tek tek yerleştirirken aynı zamanda önceden tanımlanmış minimum çevrim uzunluğu kısıtlamasının da denetlenmesidir. Algoritmanın koşulması esnasında minimum çevrim uzunluğu kısıtlaması değişmektedir. Başlangıçta yüksek bir minimum çevrim uzunluğu $\bar{g}$ 'de ısrar edilerek algoritma koşturulur. Bu koşulu ihlal etmeden daha fazla bit eklenemiyorsa, minimum çevrim uzunluğu kısıtlaması 2 birim azaltılır ve algoritma çalışmaya devam eder.

Bu çalışma ilkesi ya tüm gerekli olan bitler (*N* adet) grafa yerleştirilinceye kadar ya da minimum çevrim uzunluğu kısıtlaması daha önceden belirlenmiş bir minimum değerin ($\underline{g}$) altına düşene kadar sürdürülmektedir.

Pratikte $a_1 \leq a_2 \leq .... \leq a_N$ olması, yani öncelikle düşük dereceye sahip bitlerin, daha sonra da yüksek dereceli bitlerin yerleştirilmesi, daha iyi başarım sağlamaktadır. Tüm kısıtlamaları sağlayan, yani *k*. sütunda tam olarak a_k, $k = 1, 2, \ldots, n$ adet 1 olan, her satırda en fazla *b* adet 1 olan ve grafın minimum çevrim uzunluğu en az $g' \geq \underline{g}$ koşullarını sağlayan *n* adet sütunun *H* matrisine yerleştirildiği varsayılsın. Bu durumda $(n + 1)$. sütunun *H* matrisine yerleştirilme kuralı açıklanmalıdır.

Başlangıçta boş olan ve en fazla a_{n+1} boyutuna sahip olan U_1 kümesi, yeni eklenecek sütunu göstersin. Bu küme denetim düğümlerinden oluşmaktadır, bu nedenle $\{1,2,....,m\}$ kümesinin alt kümesi olacaktır. Ayrıca *i* adet $(0 \leq i < a_{n+1})$ denetim düğümünün U_1 kümesine eklendiği varsayılsın. Aşağıda adımları verilen prosedür, U_1 kümesine $(i + 1)$. denetim düğümünü eklemeye çalışmaktadır. Bu işlem başarısızlıkla sonuçlanabilir, bu durumda tüm prosedür durdurulur.

Girdiler: m, N, b, $\{a_1, a_2, ..., a_N\}$, $\bar{g}$ ve $\underline{g}$.

(1) $n = 0$, A = {1, 2,…., m}, $U_1 = \phi$ ve $g' = \bar{g}$ olarak başlangıç değerleri verilir.

(2) $c \in$ {1, 2,…., m} için deg(c) = 0 ve $N_c = \phi$ yapılır.

(3) do {

(4) $\forall c \in U_1$ için $H_{c,n} = 1$ yapılır ve deg(c) 1 arttırılır.

(5) $i = 0$, $U_1 = \phi$ ve $U = \phi$ olarak kurulur.

(6) do {

(7) $F_0 = A \setminus U$ hesaplanır.

(8) if ($F_0 \neq \phi$) {

(9) F_0 içerisinden c^* seçilir. Birinci derece sezgisele göre (first order heuristic) en düşük dereceli olanlar bir yerde toplanır ve içlerinden biri seçilir.

(10) $\forall c \in U_1$ için $N_c = N_c \cup \{c^*\}$ ve $N_{c^*} = N_{c^*} \cup U_1$ güncellemeleri yapılır.

(11) $U_1 = U_1 \cup \{c^*\}$, $U = U \cup V_{(g'/2)-1}(c^*)$ ve A güncellemeleri yapılır.

(12) $i = i + 1$.

(13) } else {

(14) $g' = g' - 2$.

(15) $U = \bigcup_{1 \le j \le (g'/2)-1} U_j$ tekrar hesaplanır.

(16) }

(17) } while $((i < a_n) \& (g' \geq \underline{g}))$

(18) $n = n + 1$.

(19) } while $((n < N) \& (g' \geq \underline{g}))$

Çıktılar: n, g', $H_n \equiv H$.

Burada $\cup$, işleme giren kümelerin birleşimini göstermektedir. Algoritmada yer alan (3) - (19) arasındaki dış do while döngüsü, eşlik denetim matrisine sütun

eklemektedir. Eğer iterasyon başarıyla sonuçlanırsa (4). adım H matrisini günceller. (6) - (17) arasındaki iç do while döngüsü ise U_1 kümesine bir denetim düğümü eklemektedir.

Burada U_1, eklenecek olan (n + 1). bit düğümüne bağlı denetim düğümlerinin kümesi olarak düşünülebilir. Bir denetim düğümü $1 \leq c \leq m$ için N_c, kendisiyle bir bit düğümünü paylaşan denetim düğümlerinin kümesini göstersin. Başka bir deyişle N_c, c'den 2 dal uzak olan tüm denetim düğümlerinin kümesidir. $j \geq 2$ için aşağıdaki tanım yapılmaktadır:

$$U_j = \bigcup_{c \in U_{j-1}} N_c . \tag{4.3}$$

Eğer c^* U_2 kümesinin elemanıysa 4 uzunluklu çevrimlerin olduğu garanti edilmektedir. Bu yüzden bu tür çevrimlerden kurtulmak amacıyla U_2 kümesinde denetim düğümlerinin olmaması gerekmektedir. U_j kümesinde yer alan bir denetim düğümünü U_1 kümesine eklemek, 2j veya daha küçük uzunlukta çevrim uzunlukları yaratacaktır. Bu nedenle aşağıda verilen eşitlik sağlanmalıdır:

$$U = \bigcup_{1 \leq j \leq (g'/2)-1} U_j . \tag{4.4}$$

deg(c), c denetim düğümünün derecesini göstersin ve A kümesi de aşağıdaki gibi tanımlansın:

$$A = \{c \in \{1,2,....,m\} : \deg(c) < \mathrm{b}\}. \tag{4.5}$$

Bu durumda minimum çevrim uzunluğu ve denetim düğümleri dereceleri kısıtlamalarını bozmadan U_1 kümesine eklenebilecek denetim düğümlerinin kümesi aşağıdaki gibi olacaktır:

$$F_0 = A \setminus \mathrm{U} . \tag{4.6}$$

Eğer $F_0 = \phi$ ise mevcut durumdaki minimum çevrim uzunluğu azaltılır. Eğer g', $\underline{g}$'nin de altına düşerse prosedür durdurulur. F_0'dan hangi denetim düğümünün seçilmesi konusunda birinci dereceden sezgisel kural geçerli olmaktadır.

Eğer U_1 kümesine c denetim düğümü eklenirse, $U_1(c) = \{c\}$ olarak kurulur ve $2 \leq j \leq (g'/2) - 1$ için

$$U_j(c) = \bigcup_{c' \in U_{j-1}(c)} N_{c'} \tag{4.7}$$

$$V_{(g'/2)-1}(c) = \bigcup_{1 \leq j \leq (g'/2)-1} U_j(c) \tag{4.8}$$

eşitlikleri hesaplanır. Son olarak da U güncellemesi

$$U = U \bigcup V_{(g'/2)-1}(c) \tag{4.9}$$

biçiminde yapılmaktadır.

Bu algoritma, daha önceden belirlenmiş bit dereceleri dağılımı sayesinde düzensiz LDPC kodlar da üretmektedir. Bit derecelerinin dağılımı Tablo 4.1'de verilen düzensiz bir LDPC kodun tasarımı yapılmak istensin. Tabloya göre N kod sözcük uzunluklu LDPC kod için d_i derecesine sahip n_i adet bit düğümü olacaktır.

Tablo 4.1: Düzensiz LDPC Kod için Bit Dereceleri Dağılımı

Derece	Bit Oranı
d_1	n_1/N
d_2	n_2/N
. . .	. . .
d_r	n_r/N

Burada $d_1 < \mathrm{d}_2 < .... < \mathrm{d}_\mathrm{r}$ ve $N = \sum_{i=1}^{r} n_i$ olarak tanımlıdır. Bu dağılımı bit doldurma algoritmasında kullanmak amacıyla (4.10)'daki eşitlik her $1 \leq i \leq r$ için kullanılmaktadır:

$$a_j = \begin{cases} d_1, 0 < \mathrm{j} \leq \mathrm{N}_1, \\ \\ d_2, N_1 < \mathrm{j} \leq \mathrm{N}_2, \\ . \\ . \\ . \\ d_r, N_{r-1} < \mathrm{j} \leq \mathrm{N}_\mathrm{r} = N. \end{cases} \tag{4.10}$$

Ayrıca eğer $\overline{g} = g = 2m + 2$ koşulu sağlandığı takdirde çevrim bağımsız bir graf garanti edilmektedir.

Örnek olarak $N = 9$, $m = 4$, $b = 4$, $a_i = 2$, $\overline{g} = \underline{g} = 6$ parametrelerine sahip bir (9, 5) düzenli LDPC kodu tasarlanmak istensin. Bu kod tasarımına ait algoritma adımları aşağıda verilmiştir:

1) $n = 0, A = \{0, 1, 2, 3\}, U_1 = \phi$, $g' = 6$ başlangıç değerleri verilir.
2) $\forall c \in A$ için $\deg(c) = 0$, $N_c = \phi$, $U = \phi$ yapılır.
3) $F_0 = \{0,1,2,3\}$. Tüm elemanların dereceleri eşit olduğu için içlerinden herhangi biri seçilebilir. $c^* = 0$ seçilsin. $N_c = \phi$ ve $N_{c^*} = N_0 \bigcup U_1 = N_0 = \phi$ güncellemeleri yapılır.
4) $U_1 = U_1 \bigcup c^* = \{0\}$, $U_1(c) = c = \{0\}$, $V_2(c^* = 0) = U_1(c) \bigcup U_2(c)$ güncellemeleri yapılır: $U_2(c) = N_0 = \phi \Rightarrow V_2(c^* = 0) = \{0\}$.
5) $U = U \bigcup V_2(c^*) = \{0\}$ yapılır.
6) $i = 1$ yapılır.
7) $F_0 = A \setminus \mathrm{U} = \{1,2,3\}$ olarak güncellenir.
8) $c^* = 1$ seçilsin. $N_0 = N_0 \bigcup \{1\} = \{1\}$ ve $N_1 = N_1 \bigcup U_1 = \phi \bigcup U_1 = \{0\}$ güncellemeleri yapılır.

9) $U_1 = U_1 \bigcup \{1\} = \{0,1\}$, $U_1(1) = \{1\}$, $U_2(1) = N_1 = \{0\}$ güncellemeleri yapılır.

10) $V_2(c^* = 0) = \{0,1\} \Rightarrow U = \{0,1\}$, A = {0, 1, 2, 3} olarak güncellenir.

11) $i = 2$ yapılır.

12) $n = 1$ yapılır.

13) $H_{0,0} = H_{1,0} = 1$ yapılarak ve deg(0) ve deg(1)'in değerleri 1 arttırılarak güncellemeler yapılır.

14) $i = 0$, $U_1 = \not\in$, $U = \not\in$ ve $F_0 = \{0,1,2,3\}$ güncellemeleri yapılır.

15) $c^* = 2$ seçilsin. $U_1 = \{2\}$, $U = \{2\}$ ve $A = \{0, 1, 2, 3\}$ olarak güncellenir.

16) $i = 1$ yapılır.

17) $F_0 = \{0,1,3\}$ olarak güncellenir.

18) $c^* = 3$ seçilir. $N_2 = N_2 \bigcup \{3\}$ ve $N_3 = \{2\}$ olarak güncellemeler yapılır.

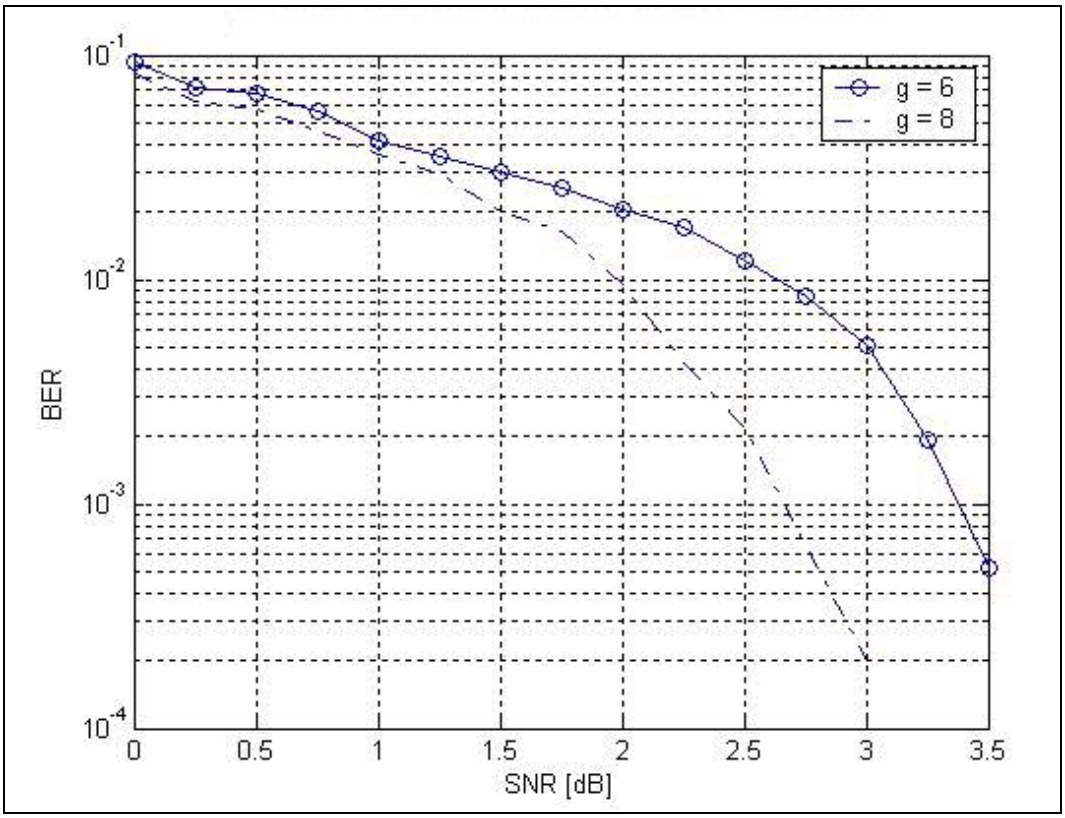
19) $U_1 = \{2,3\}$, $U_1(c) = c = \{3\}$, $U_2(c) = N_3 = \{2\}$, $V_2(c^*) = \{2,3\}$ ve $U = \{2,3\}$ şeklinde güncellemeler yapılır.

20) $i = 2$, $n = 2 =$ $H_{2,1} = H_{3,1} = 1$ yapılarak ve deg(2) ve deg(3)'ün değerleri 1 arttırılarak güncellemeler yapılır.

Algoritmanın geri kalan adımları da benzer şekilde hesaplanabilmektedir.

N = 3502, m = 2000, g = {6, 8} ve her sütunun ağırlığı j = 3 olan düzenli bir LDPC kodunun bit doldurma algoritması ile oluşturulup mesaj aktarma algoritması ile çözülmesi durumundaki bit hata başarımı Şekil 4.4'te verilmiştir.

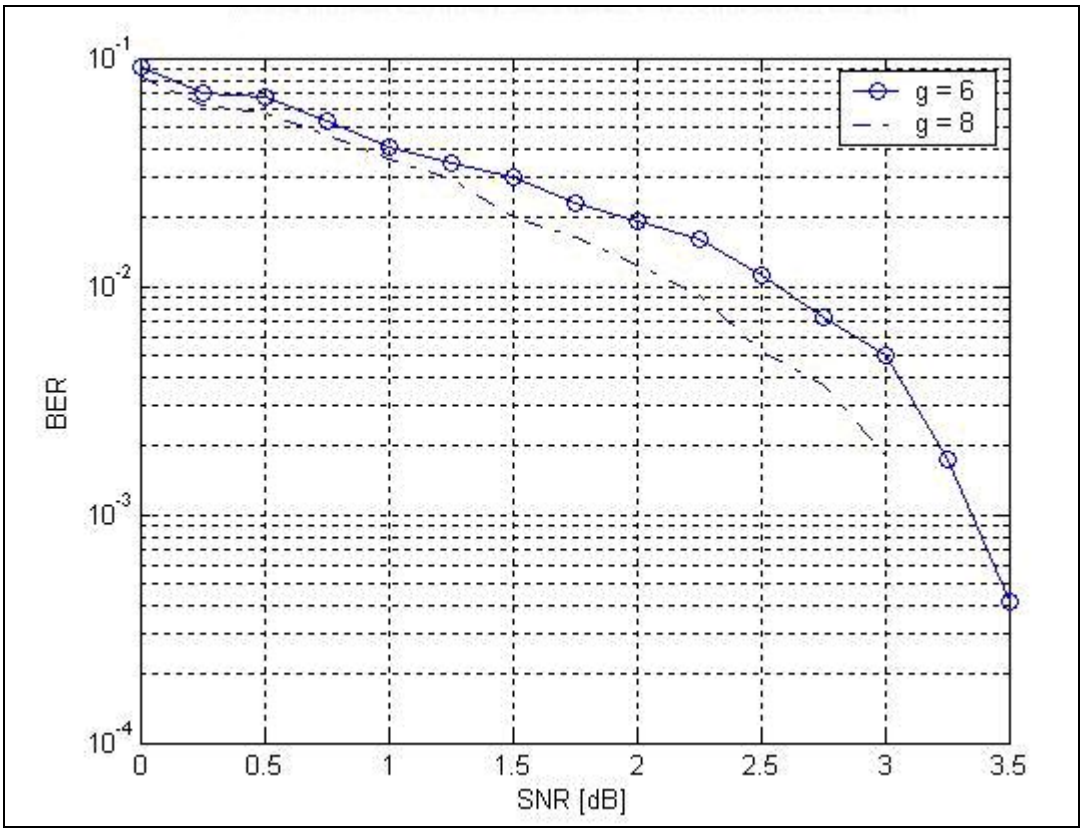
Burada aynı N ve m parametrelerine sahip 2 düzenli LDPC kodun bit hata başarımına minimum çevrim uzunluğunun etkisi görülmektedir. Minimum çevrim uzunluğu arttıkça hata başarımı da iyileşmektedir. g'yi 6'dan 8'e çıkararak 0.001 bit hata başarımı sağlamak için yaklaşık 0.5 dB kazanç elde edilmiştir. 1 dB'in altında ise minimum çevrim uzunluğunun etkisi azalmaktadır.



Şekil 4.4: Düzenli bir (3502, 1502) LDPC Kodu için Bit Doldurma Algoritmasının Bit Hata Başarımı

N = 1000, m = 700, g = {6, 8} olan düzensiz bir LDPC kodunun bit doldurma algoritması ile oluşturulup mesaj aktarma algoritması ile çözülmesi durumundaki bit hata başarımı Şekil 4.5'te verilmiştir.

Burada da aynı N ve m parametrelerine sahip 2 düzensiz LDPC kodun bit hata başarımına minimum çevrim uzunluğunun etkisi görülmektedir. 0.002 bit hata başarımına ulaşmak için minimum çevrim uzunluğunu g = 6'dan g = 8'e çıkarmak yaklaşık 0.25 dB'lik bir kazanç sağlamaktadır. 1 dB'in altında ise minimum çevrim uzunluğunun etkisi kaybolmaktadır.



Şekil 4.5: Düzensiz bir (1000, 300) LDPC Kodu için Bit Doldurma Algoritmasının Bit Hata Başarımı

5. LDPC KOD OPTİMİZASYONU

Bu bölümde düzenli ve düzensiz LDPC kodları için optimizasyon teknikleri anlatılmaktadır.

5.1 Yoğunluk Evrim Tekniği

Kod sözcük uzunluğu sonsuz olmayan kodlar için mesaj aktarma algoritmasını kullanan kod çözücülerin yakınsaklık özelliğinin analizi zor olmaktadır. Fakat kod sözcük uzunluğunun sonsuza götürülmesi analizi kolaylaştırmaktadır çünkü bu durumda var olan çevrimler yok sayılabilir, yani düğümlerdeki mesajlar bağımsız olarak kabul edilebilir. Yoğunluk evrim (density evolution), diğer adıyla Gauss yaklaşıklık (Gaussian approximation) tekniği bir LDPC kodda kod sözcük uzunluğu N sonsuza giderken ilgili Tanner grafındaki mesajların olasılık yoğunluk fonksiyonlarını (pdf) izlemek için kullanılan bir tekniktir.

Genelliği bozmadığı için tüm bitleri sıfır olan kod sözcüğünün iletildiği ve $0-1,\ 1-\ -1$ şeklinde bitten simgeye eşleme yapıldığı varsayılsın. Bu durumda AWGN kanal için n. bit düğümünden iletilen ilk mesaj (bilgi) aşağıdaki gibi olacaktır:

$$\lambda_n^{(0)} = \frac{2}{c^2}\left(1 + N\left(0, \sigma^2\right)\right). \tag{5.1}$$

(5.1) ifadesi analiz edilirse $\lambda_n^{(0)}$'ın varyansı ortalamasının iki katı olmaktadır:

$$\lambda_n^{(0)} \sim N(m,2m)\,. \tag{5.2}$$

Burada $m = 2/\sigma^2$ olmaktadır. Bu şekildeki bir Gauss rastlantı değişkeni tutarlılık (consistency) özelliğine sahip olmaktadır. İlk iletilen mesajlar Gauss dağılımlı olmasına rağmen diğer mesajlar Gauss dağılımlı değildir. Yine de analizi kolaylaştırmak amacıyla tüm mesajların tutarlı Gauss pdf'ine sahip oldukları varsayılır. Bu durumda izlenmesi gereken tek bir parametre bulunmaktadır, bu da ortalamadır [28][29][30][31]. Analiz edilecek LDPC kod, *j* dereceli bit düğümlerine ve *k* dereceli denetim düğümlerine sahip düzenli bir LDPC kod olacaktır.

$u^{(l)}$, l iterasyon sonunda denetim düğümünden bit düğümüne doğru giden rastgele seçilmiş mesajı ve μ_l de ortalamasını ifade etsin:

$$\mu_l = E\left[u^{(l)}\right]. \tag{5.3}$$

Tanh kuralına göre bu mesaj aşağıdaki eşitliği sağlamaktadır:

$$\tanh\left(\frac{u^{(l)}}{2}\right) = \prod_{i=1}^{k-1} \tanh\left(\frac{v_i^{(l)}}{2}\right). \tag{5.4}$$

Burada $v_i^{(l)}$'ler l iterasyon sonunda bit düğümünden denetim düğümüne doğru olan ve herbirinin ortalaması $m_l = E\left[v_i^{(l)}\right]$ olan ilgili mesajları ifade etmektedir. Yine $u^{(l)}$ ve $v_i^{(l)}$'lerin tutarlı Gauss rastlantı değişkenleri oldukları varsayılacaktır. Yani $u^{(l)} \sim N(\mu_l, 2\mu_l)$ ve $v^{(l)} \sim N(m_l, 2m_l)$ olacaktır. (5.4) eşitliğinin her iki tarafının beklenen değeri alınırsa ve $v_i^{(l)}$'lerin bağımsız oldukları varsayılırsa aşağıdaki eşitlik elde edilir:

$$\psi(\mu_l) = \psi(m_l)^{k-1}. \tag{5.5}$$

Burada $\psi(m)$ (5.6)'daki gibi tanımlı olmaktadır. Ayrıca bu fonksiyon $\tanh(m/2)$ fonksiyonuyla benzer karakteristiklere sahip olmakla beraber fonksiyonun tersi de tanımlıdır.

$$\psi(m) = E\left[\tanh\left(\frac{1}{2}N(m,2m)\right)\right]. \tag{5.6}$$

m_l'yi μ_{l-1} ile ilişkilendirmek amacıyla aşağıdaki denklem elde edilir:

$$v_i^{(l)} = v_i^{(0)} + \sum_{n=1}^{j-1} u_n^{(l-1)} . \tag{5.7}$$

Burada $v_i^{(0)}$, bit düğümünden denetim düğümüne iletilen ilk mesajı ifade etmektedir. Bu mesaj da $2/\sigma^2$ ortalamalı tutarlı bir Gauss rastlantı değişkenidir. $u_n^{(l-1)}$'ler ise her birinin ortalaması μ_{l-1} olan bağımsız ve denetim düğümünden bit düğümüne iletilen ilgili mesajları ifade etmektedir. (5.7)'nin her iki tarafının ortalaması alınırsa aşağıdaki eşitlik elde edilmektedir:

$$m_l = \frac{2}{c^2} + (j-1)\mu_{l-1} . \tag{5.8}$$

(5.8) eşitliği (5.5)'e yerleştirilirse

$$\psi(\mu_l) = \psi\left(\frac{2}{\sigma^2} + (j-1)\mu_{l-1}\right)^{k-1} \tag{5.9}$$

elde edilir.

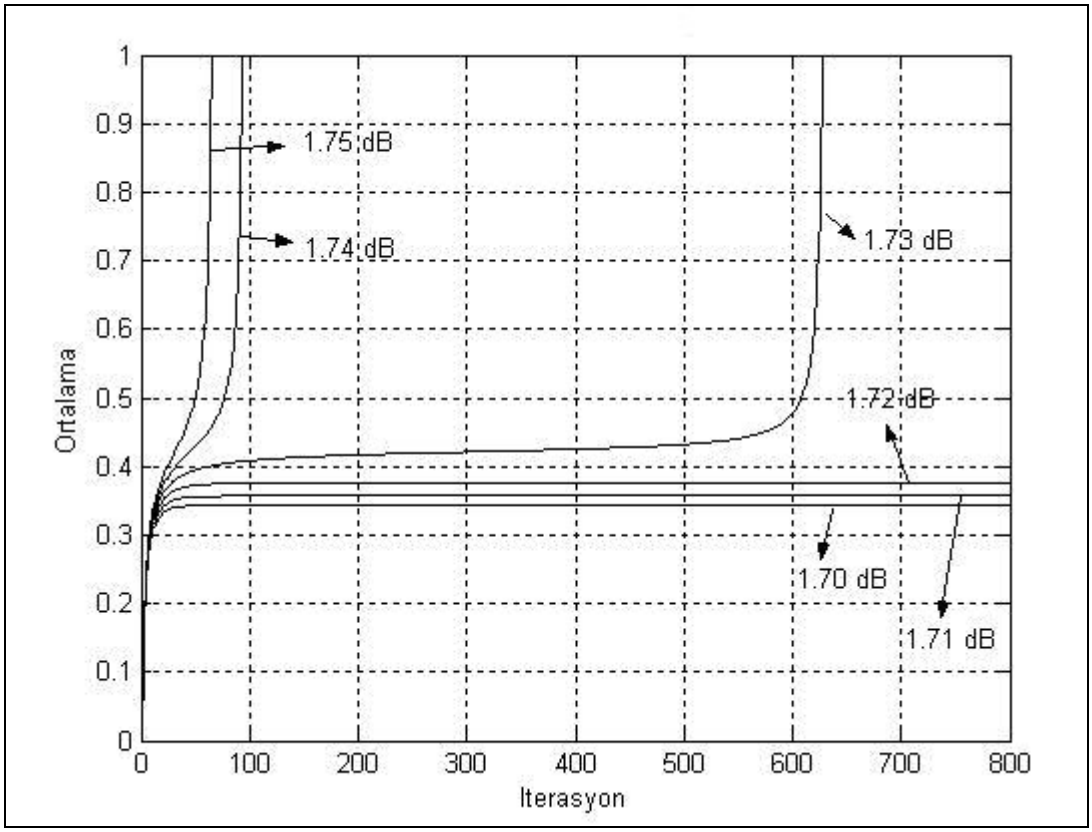
(5.9)'da μ_l yalnız bırakıldığında, l iterasyon sorunda rastgele seçilen ve denetim düğümünden bit düğümüne iletilen bir mesajın ortalaması aşağıdaki denklem sayesinde hesaplanmaktadır:

$$\mu_l = \psi^{-1}\left(\psi\left(\frac{2}{\sigma^2} + (j-1)\mu_{l-1}\right)^{k-1}\right). \tag{5.10}$$

Mesaj aktarma kod çözücüsünde ifade edildiği gibi denetim düğümünden bit düğümüne doğru iletilen ilk mesajın ortalaması sıfır olmaktadır. Bu başlangıçla

(5.10) eşitliği sürekli tekrarlanırsa μ_l'nin evrimi analiz edilmektedir. Bu yinelemenin dinamikleri *j, k* ve işaret gürültü oranı (SNR) $2/\sigma^2$ parametreleriyle belirlenmektedir. Düzenli LDPC kodlar için yoğunluk evrim algoritmasının MATLAB programlama diliyle yazılmış hali EK – D'de verilmiştir.

j, *k*, c^2'nin her türlü değer kümesiyle (5.10) eşitliği her zaman yakınsamaktadır. Ayrıca tüm $E_b/N_0 > \gamma$ için, *l* sonsuza giderken μ_l'yi sonsuza götüren bir SNR eşiği γ var olmaktadır. Bu durum da kod çözme hata olasılığının sıfıra yaklaşması anlamına gelmektedir. Şekil 5.1'de (4, 6) düzenli LDPC kodu için yoğunluk evriminin sonuçları verilmektedir. SNR değeri 1.73 dB ve daha yukarısı için iterasyon sonsuza giderken ortalama da sonsuza gitmektedir. Bu da hata olasılığının sıfıra yaklaşmasına neden olmaktadır. 0.33 oranlı kodlar için Shannon sınırı (4.2) eşitliğine göre hesaplandığında -0.55 dB bulunur. Şekil 5.1, yeterince uzun (4, 6) düzenli LDPC kodlar için SNR = 1.73 dB değerinde hata olasılığının çok düşük olduğunu ve sığayla arasındaki farkın 2.28 dB olduğunu göstermektedir. Bu fark düzensiz LDPC kodlar kullanılarak daha da azaltılabilmektedir [8][17].



Şekil 5.1: (4, 6) Düzenli LDPC Kodu için Yoğunluk Evrimi Analizi

Tablo 5.1'de ise çeşitli *j* ve *k* değerlerine sahip LDPC kodlar için hesaplanan eşik değerleri verilmiştir. Bu değerlere bakılarak düzenli LDPC kodlar için optimize kodun, $j = 3$ seçildiğinde elde edilebildiği söylenebilir. Buradaki eşik değeri, iterasyon sayısı sonsuza giderken ortalamayı da sonsuza götüren minimum işaret - gürültü oranı olmaktadır.

Tablo 5.1: Düzenli LDPC Kodlar için Yoğunluk Evrimi Analiziyle Hesaplanan Eşik Değerleri

j	*k*	Oran	Eşik [dB]
3	6	0.5	1.17
4	8	0.5	1.6
5	10	0.5	2.04
3	5	0.4	0.97
4	6	0.3333	1.73
3	4	0.25	1.06
4	10	0.6	1.78
3	9	0.6667	1.79
3	12	0.75	2.26
5	6	0.1667	3.96

Benzer işlemler düzensiz LDPC kodlara uygulandığında, *l* iterasyon sorunda rastgele seçilen ve denetim düğümünden bit düğümüne iletilen bir mesajın ortalaması

$$m_u^{(l)} = \sum_{j=2}^{d_c} \rho_j \phi^{-1}\left(1-\left[1-\sum_{i=2}^{d_v}\lambda_i \phi\left(\frac{2}{\sigma^2}+(i-1)m_u^{(l-1)}\right)\right]^{j-1}\right) \tag{5.11}$$

şeklinde elde edilmektedir.

Burada $\phi(.)$ fonksiyonu aşağıdaki şekilde tanımlanmaktadır:

$$\phi(x) = \begin{cases} e^{\alpha x^{\gamma}+\beta},\ 0 < x < 10,\ \alpha = \text{-}0.4527,\ \beta = 0.0218,\ \gamma = 0.86 \\ \sqrt{\dfrac{\pi}{x}} e^{\frac{-x}{4}} \left(1 + \dfrac{2}{x}\right),\ x \geq 10. \end{cases} \tag{5.12}$$

[29]'da optimize LDPC kodlar elde etmek için $\rho(x)$'in aşağıdaki gibi seçilmesi gerektiği gösterilmiştir:

$$\rho(x) = \rho x^{k-1} + (1-\rho)x^{k},\ k \geq 2,\ 0 \leq \rho \leq 1. \tag{5.13}$$

Bu tekniğe dayalı optimizasyonun ancak d_v'si düşük ve kodlama oranı yüksek LDPC kodlarda daha iyi sonuç verdiği gösterilmiştir [29].

5.2 Farksal Evrim Tekniği

Farksal evrim (differential evolution) tekniğinin yoğunluk evrim tekniğinden en büyük farkı, AWGN kanala uygulanabildiği gibi ayrıca Rayleigh sönümlemeli kanala da uygulanabilmesi olarak gösterilebilir. Ayrıca bu teknik hem düzenli hem de düzensiz LDPC kodlara uygulanabilmektedir.

AWGN kanal için ilk olarak iletilen mesajın yoğunluk fonksiyonu simetri özelliğine sahip olmaktadır. Bu özelliğe sahip yoğunluk fonksiyonları her pozitif x değeri için $f(x) = f(-x)e^{x}$ eşitliğini sağlarlar. Bu özelliği tanıtlamak amacıyla

$$P_0(q_0) = \frac{\sigma}{2\sqrt{2\pi}} e^{\left(\frac{-\left(q_0 - \frac{2}{\sigma^2}\right)^2}{2\left(\frac{4}{\sigma^2}\right)} \right)} \tag{5.14}$$

$$= \frac{\sigma}{2\sqrt{2\pi}} e^{\left(\frac{-\left(-q_0 - \frac{2}{\sigma^2}\right)^2}{2\left(\frac{4}{\sigma^2}\right)} \right)} e^{q_0} \tag{5.15}$$

$$= P_0(-q_0)e^{q_0} \tag{5.16}$$

eşitlikleri kullanılır. Burada $\sigma^2 = 1/2R(E_b/N_0)$ olarak seçilmektedir. Eğer iletilen mesajların yoğunluk fonksiyonları simetrik ise hatalı olan mesajların oranı belli bir değere, belki de sıfıra, yakınsamaktadır [17].

Yakınsaklık (convergence) ya da denge (stability) koşulu için de

$$\lambda_2 < \lambda_2^{\,*} = \frac{e^{1/2\sigma_n^2}}{\sum_{j=2}^{d_c} \rho_j (j-1)} \tag{5.17}$$

ilişkisi geçerli olmaktadır [10]. Burada λ_2, Tanner grafında 2 derecesine sahip bit düğümlerine gelen dalların tüm dallara oranını göstermektedir.

Ayrıca (λ, ρ) dağılım çiftine sahip bir LDPC kodun doğrusal zamanda kodlanabilir (linear time encodable) olabilmesi için aşağıdaki eşitsizliğin sağlanması gerekmektedir:

$$\lambda_2 \geq \frac{\sum_{i=2}^{\rho_{maks}} \frac{\rho_i}{i}}{2}. \tag{5.18}$$

Bu algoritmanın temeli, sürekli uzay parametreleriyle doğrusal olmayan maliyet fonksiyonu minimizasyonu problemini çözmeye dayanmaktadır [10][32][33]. Bu teknik hem silen kanalda hem AWGN kanalda hem de Rayleigh sönümlemeli kanallar için çok iyi düzensiz LDPC kodlar tasarlanmasında çok başarılı olmaktadır.

[17]'de 0.5 hızlı ve en iyi derece dağılımına sahip kodlar için bulunan eşik değerinin kanal sığasından farkı 0.06 dB olarak bulunmuştur. Bu çalışmada ise bu fark 0.02 dB'e indirilmektedir. (λ, ρ) dağılım çiftine sahip olan bir kod için aşağıdaki eşitlik sağlanmaktadır:

$$\sum_i \frac{\rho_i}{i} = K \sum_i \frac{\lambda_i}{i}, \quad K = 1 - R. \tag{5.19}$$

Ayrıca bu dağılım çifti için

$$\lambda_2 = 1 - \sum_{i=3}^{d_{lmaks}} \lambda_i, \quad \rho_2 = 1 - \sum_{i=3}^{d_{rmaks}} \rho_i \tag{5.20}$$

eşitlikleri de geçerli olmaktadır. Burada d_{lmaks} ve d_{rmaks} sırasıyla maksimum bit ve denetim düğüm derecelerini göstermektedir.

(5.19) ve (5.20) eşitlikleri kullanılarak $\lambda_{d_{lmaks}}$ çözüldüğünde aşağıdaki eşitlik elde edilir:

$$\lambda_{d_{lmaks}} = \frac{\frac{1-K}{2} + \sum_{i=3}^{d_{rmaks}} \rho_i \left(\frac{1}{i} - \frac{1}{2}\right) - K \sum_{i=3}^{d_{lmaks}-1} \lambda_i \left(\frac{1}{i} - \frac{1}{2}\right)}{K\left(\frac{1}{d_{lmaks}} - \frac{1}{2}\right)}. \tag{5.21}$$

L, (λ, ρ) dağılım çiftine ait bağımsız eleman sayısını göstersin. $\lambda_1 = \rho_1 = 0$ için (5.20) ve (5.21)'deki eşitlikler göz önüne alındığında bağımsız eleman sayısı $L = d_{lmaks} + d_{rmaks} - 5$ olacaktır. Algoritmada L boyutlu $p = (\lambda_3, ..., \lambda_{dlmaks-1}, \rho_3, ..., \rho_{drmaks})$ vektörü oluşturulmaktadır. Buradaki amaç, bu vektörün bileşenlerinin optimal olarak seçilerek ilgili derece dağılımı çiftinin en yüksek gürültü eşik değerine ulaşmasını sağlamaktır.

Farksal evrim tekniği, bir paralel arama tekniğidir. İlk vektörden başlanarak algoritma iteratif olarak her bir vektörü paralel olarak günceller. Eğer en iyi maliyet fonksiyonu değerine sahip olan üstün bir vektör bulursa algoritma sonlanır. Tüm vektörlerin paralel olarak güncellenmesi sayesinde yerel minimalardan ve ıraksaklıktan kurtulmak mümkün olmaktadır. Farksal evrim algoritmasının adımları aşağıda verilmiştir:

(1) <u>Başlangıç:</u>

Gürültü seviyesi c seçilir. İlk üretim $G = 0$ için rastgele olarak $10L$ adet L boyutlu $P_{i,G}$ (i = 0, 1, …., $10L$ - 1) vektörü oluşturulur. $10L$ değeri optimizasyon süreci boyunca sabit kalmaktadır. Her vektör için Yoğunluk Evrimi tekniği uygulanır ve eşik değeri hesaplanır. Burada iterasyon sayısı yaklaşık 1000 olarak seçilir. En düşük hata olasılığına,

yani en büyük $m_u^{(l)}$ değerine sahip olan vektör en iyi vektör $P_{best,G}$ olarak belirlenir. Her bir vektörün hatası $P_{ei,G}$ olarak hesaplanır. $P_{best,G}$ de en düşük $P_{ei,G}$ değerine sahip olan vektördür.

(2) <u>*Değişim:*</u>

Yeni üretim $G + 1$ için yeni vektörler üretirken, her $i = 0, 1, \ldots, 10L - 1$ için, 0'la $10L - 1$ arasında ve her biri i'den farklı dört adet r_1, r_2, r_3, r_4 tamsayıları seçilir ve aşağıdaki tanım yapılır:

$$V_{i,G+1} = P_{best,G} + F\left(P_{r1,G} - P_{r2,G} + P_{r3,G} - P_{r4,G}\right). \tag{5.22}$$

Burada F sabiti, farksal varyasyonun kuvvetlendiricisi olmaktadır ve genelde $F = 0.5$ seçilir. Yeni vektörler için de farksal evrim tekniği uygulanır ve her bir vektörün hatası $P_{vi,G+1}$ olarak kaydedilir.

(3) <u>*Seçme:*</u>

Her $i = 0, 1, \ldots, 10L - 1$ için $P_{ei,G}$ ile $P_{vi,G+1}$ karşılaştırılır.

Eğer $\left(P_{ei,G} > P_{vi,G+1}\right)$

$$P_{i,G+1} = V_{i,G+1}$$

Değilse

$$P_{i,G+1} = P_{i,G}$$

En küçük hatalı vektör de $P_{best,G+1}$ olarak atanır.

(4) <u>*Sonlandırma:*</u>

Eğer $P_{best,G+1}$'e ait olan hata 10^{-8}'den daha büyükse (2). adıma geri dönülür. Bu hata 10^{-8}'den daha küçükse c biraz arttırılır ve (1). adıma geri dönülür. Eğer uzun çalışma zamanı sonunda hata sıfıra yakınsamıyorsa algoritma durur. En yüksek c değeri için hatası sıfıra giden vektör en iyi vektör olarak atanır ve eşik gürültü düzeyi $c^* = c$ olur.

Tablo 5.2, 5.3, 5.4, 5.5 ve 5.6'da bu yöntemle tasarlanmış üstün düzensiz LDPC kodlar yer almaktadır. Bu tablolar yukarıdaki algoritmanın bilgisayar yardımıyla işletilmesi sonucu elde edilmiştir.

Şekil 5.2'de ise farksal evrim tekniğiyle tasarlanan kodların ikili AWGN kanal için bit doldurma algoritmasıyla gerçeklemesi yapıldıktan sonra kod çözücüde mesaj aktarma algoritması kullanıldığında bit hata başarımları verilmiştir.

Görülmektedir ki, kod sözcük uzunluğu sonsuza giderken bilgisayar benzetim sonuçları da hesaplanan eşik değerlerine yaklaşmaktadır. Yoğunluk evrimi tekniğiyle farksal evrim tekniğinin birleştirilmesi çok iyi LDPC kodlar tasarlanmasında anahtar rol oynamaktadır. Ayrıca tasarlanan tüm kodlar, denge özelliğine sahip olmakla beraber daha yüksek bit dereceli kodların daha iyi başarım sunduğu görülmüştür. Denetim düğümlerinin derecelerinin de düşük olması başarıma olumlu olarak yansımaktadır. Ayrıca üst üste gelen hatalı bitlerin sırasını değiştirmek ve böylece hata olasılığını düzeltmek amacıyla kullanılan serpiştirici (interleaver) LDPC kodların doğasında zaten varolduğundan üstün kodlar tasarlanabilmiştir.

Tablo 5.2: İkili AWGN Kanal için $R = 0.25$ Oranında Tasarlanmış Düzensiz LDPC Kodlar

$R = 0.25$	AWGN	BPSK	Sığa:	c = 1.549594		SNR = -0.79405 dB	
d_{lmaks}	10	15	20	25	30	40	50
λ_2^*	0.3251	0.3040	0.2715	0.2329	0.2387	0.2381	0.2387
λ_2	0.3135	0.3018	0.2690	0.2308	0.2384	0.2363	0.2386
λ_3	0.1584	0.2208	0.2382	0.2229	0.2238	0.2252	0.2234
λ_4	0.0242						
λ_5		0.0145					
λ_7		0.0048					
λ_{10}	0.5039						
λ_{12}		0.0097					
λ_{14}		0.4146					

Tablo 5.2 (devamı): İkili AWGN Kanal için R = 0.25 Oranında Tasarlanmış Düzensiz LDPC Kodlar

λ_{15}		0.0338					
λ_{16}							0.0004
λ_{17}					0.0127		
λ_{18}			0.4816				0.0002
λ_{19}						0.0757	
λ_{20}			0.0112	0.0098			0.0027
λ_{21}					0.0082		0.0032
λ_{22}					0.2308		0.0189
λ_{23}				0.1508		0.3338	0.4852
λ_{24}							
λ_{25}				0.3857	0.1467	0.0807	0.0005
λ_{26}					0.1044		
λ_{27}							0.0001
λ_{30}					0.0350		
λ_{34}							0.0009
λ_{35}							0.0005
λ_{37}							0.0001
λ_{40}						0.0483	
λ_{41}							0.0014
λ_{44}							0.0002
λ_{46}							0.0006
λ_{47}							0.0030
λ_{50}							0.0195
ρ_5	0.9846	0.8666	0.4274				
ρ_6	0.0154	0.1334	0.5726	0.6834	0.8199	0.8087	0.8259
ρ_7				0.3166	0.1801	0.1913	0.1741

Tablo 5.2 (devamı): İkili AWGN Kanal için R = 0.25 Oranında Tasarlanmış Düzensiz LDPC Kodlar

c^*	1.370	1.480	1.520	1.530	1.535	1.536	1.539
SNR^* [dB]	0.276	-0.395	-0.627	-0.684	-0.712	-0.718	-0.734

Tablo 5.2'de tasarlanan R = 0.25 oranındaki en üstün LDPC kod, sığadan sadece 0.06 dB daha uzaktadır.

Tablo 5.3: İkili AWGN Kanal için R = 0.33 Oranında Tasarlanmış Düzensiz LDPC Kodlar

R = 0.33	AWGN	BPSK	Sığa:	c = 1.296627		SNR = -0.4953 dB
d_{lmaks}	10	15	20	25	30	50
λ_2^*	0.3430	0.2814	0.2459	0.2192	0.2101	0.1878
λ_2	0.3378	0.2786	0.2446	0.2187	0.2088	0.1873
λ_3	0.2678	0.2474	0.2458	0.2314	0.2272	0.2042
λ_9	0.0900					
λ_{10}	0.3044					
λ_{12}		0.1746				
λ_{15}		0.2994				
λ_{18}			0.4440			
λ_{20}			0.0656			0.0725
λ_{21}				0.1657		
λ_{22}				0.0130		0.2852
λ_{24}				0.1727	0.4866	
λ_{25}				0.1985		
λ_{30}					0.0774	
λ_{36}						0.0475

Tablo 5.3 (devamı): İkili AWGN Kanal için R = 0.33 Oranında Tasarlanmış Düzensiz LDPC Kodlar

λ_{50}						0.2033
ρ_5	0.9722	0.1252				
ρ_6	0.0278	0.8748	0.5010			
ρ_7			0.4990	0.8437	0.5786	
ρ_8				0.1563	0.4214	0.8174
ρ_9						0.1826
c^*	1.244	1.258	1.287	1.292	1.292	1.292
SNR^* [dB]	-0.135	-0.233	-0.431	-0.464	-0.464	-0.464

Tablo 5.3'te tasarlanan R = 0.33 oranındaki en üstün LDPC kod, sığadan sadece 0.03 dB daha uzaktadır.

Tablo 5.4: İkili AWGN Kanal için R = 0.5 Oranında Tasarlanmış Düzensiz LDPC Kodlar

$R = 0.5$	AWGN	BPSK	Sığa:	c = 0.978694		SNR = 0.1871 dB	
d_{lmaks}	10	15	20	25	30	40	50
λ_2^*	0.2742	0.2626	0.2169	0.2168	0.1987	0.1850	0.1872
λ_2	0.2741	0.2607	0.2144	0.1943	0.1974	0.1819	0.1854
λ_3	0.2680	0.3116	0.2402	0.2942	0.2512	0.2428	0.2379
λ_9	0.0738	0.0142					
λ_{10}	0.3840						
λ_{11}			0.1994				
λ_{15}		0.4135					0.1539
λ_{17}						0.0548	
λ_{18}				0.4928			0.0499

Tablo 5.4 (devamı): İkili AWGN Kanal için R = 0.5 Oranında Tasarlanmış Düzensiz LDPC Kodlar

λ_{19}					0.5170		
λ_{20}			0.3460				
λ_{21}						0.2462	
λ_{23}						0.2070	0.2620
λ_{25}				0.0187			
λ_{26}							0.0043
λ_{30}					0.0344		
λ_{36}							0.0699
λ_{38}							0.0358
λ_{40}						0.0673	
λ_{50}							0.0009
ρ_{7}	0.6442	0.3746					
ρ_{8}	0.3558	0.6254	0.0168	0.0404			
ρ_{9}			0.9832	0.9596	0.4846		
ρ_{10}					0.5154	0.8626	0.9731
ρ_{11}						0.1374	0.0269
c^*	0.9486	0.9502	0.9545	0.9571	0.9750	0.9760	0.9760
SNR^* [dB]	0.458	0.444	0.404	0.381	0.220	0.211	0.211

Tablo 5.4'te tasarlanan R = 0.5 oranındaki en üstün LDPC kod, sığadan sadece 0.02 dB daha uzaktadır.

Tablo 5.5: İkili AWGN Kanal için $R = 0.66$ Oranında Tasarlanmış Düzensiz LDPC Kodlar

$R = 0.66$	AWGN	BPSK	Sığa:	c = 0.766578		SNR = 1.0594 dB	
d_{lmaks}	15	20	25	30	40	50	75
λ_2^*	0.3770	0.3114	0.2910	0.2440	0.2037	0.1845	0.1709
λ_2	0.2782	0.3054	0.2606	0.2427	0.1939	0.1831	0.1644
λ_3	0.3034	0.1394	0.4378	0.3745	0.2986	0.2688	0.2557
λ_4	0.3966						
λ_5		0.5360					
λ_{10}			0.2526				
λ_{13}					0.3494		
λ_{14}				0.2364		0.1821	
λ_{15}	0.0218						
λ_{16}							0.2435
λ_{17}				0.1085			
λ_{18}						0.2619	
λ_{19}							0.3129
λ_{20}		0.0192			0.1219	0.1005	
λ_{25}			0.0490				
λ_{30}				0.0379			
λ_{32}							0.0015
λ_{38}					0.0012		
λ_{40}					0.0350		
λ_{50}						0.0036	
λ_{75}							0.0220
ρ_8	0.1796						
ρ_9	0.8204	0.2192	0.1030				
ρ_{10}		0.7808	0.8970				

Tablo 5.5 (devamı): İkili AWGN Kanal için R = 0.66 Oranında Tasarlanmış Düzensiz LDPC Kodlar

ρ_{11}				0.9110			
ρ_{12}				0.0890			
ρ_{13}					0.9750		
ρ_{14}					0.0250	0.9671	
ρ_{15}						0.0329	0.9555
ρ_{16}							0.0445
c^*	0.6800	0.7050	0.7250	0.7450	0.7470	0.7550	0.7557
SNR^* [dB]	2.10	1.787	1.544	1.307	1.284	1.192	1.184

Tablo 5.5'te tasarlanan R = 0.66 oranındaki en üstün LDPC kod, sığadan sadece 0.12 dB daha uzaktadır.

Tablo 5.6: İkili AWGN Kanal için R = 0.75 Oranında Tasarlanmış Düzensiz LDPC Kodlar

R = 0.75 AWGN BPSK Sığa: c = 0.6770 SNR = 1.62637 dB					
d_{lmaks}	30	40	50	60	75
λ_2^*	0.2841	0.2385	0.2126	0.1997	0.1874
λ_2	0.2768	0.2329	0.2104	0.1905	0.1199
λ_3	0.4946	0.3834	0.3208	0.2860	0.3748
λ_5					0.0310
λ_8				0.1541	
λ_{12}		0.3530	0.4290		0.0835
λ_{13}	0.2256			0.3067	
λ_{16}					0.3750
λ_{20}			0.0181		

Tablo 5.6 (devamı): İkili AWGN Kanal için R = 0.75 Oranında Tasarlanmış Düzensiz LDPC Kodlar

λ_{23}			0.0153		
λ_{30}	0.0030			0.0471	
λ_{40}		0.0307			
λ_{50}			0.0064		
λ_{60}				0.0156	
λ_{75}					0.0158
ρ_{12}	0.5074				
ρ_{13}	0.4926				
ρ_{14}		0.4070			
ρ_{15}		0.5930			
ρ_{16}			0.9703		
ρ_{17}			0.0297	0.9983	
ρ_{18}				0.0017	0.9529
ρ_{19}					0.0471
c^*	0.6500	0.6520	0.6560	0.6560	0.6561
SNR^* [dB]	1.981	1.954	1.901	1.901	1.900

Tablo 5.6'da tasarlanan R = 0.75 oranındaki en üstün LDPC kod, sığadan sadece 0.27 dB daha uzaktadır.

Yukarıdaki tablolarda bulunan eşik SNR değerleri, kod sözcük uzunluğu N'nin sonsuza gitmesi durumunda elde edilen değerler olmaktadır. N'nin sonlu olması durumu için de Şekil 5.2 verilmektedir. Sonlu durum için N = 1000 değeri alınmıştır.

[17]'de R = 0.5 için bulunan eşik SNR değerlerine bakıldığında, maksimum denetim düğüm derecesi 50 için bulunan değer 0.2485 dB iken Tablo 5.4'te bu değer 0.211 dB'e indirilmiştir. Ayrıca maksimum dereceye sahip denetim düğümleri için (d_{lmaks}) derece sayısını arttırmak, işaret - gürültü oranı bakımından kazanç

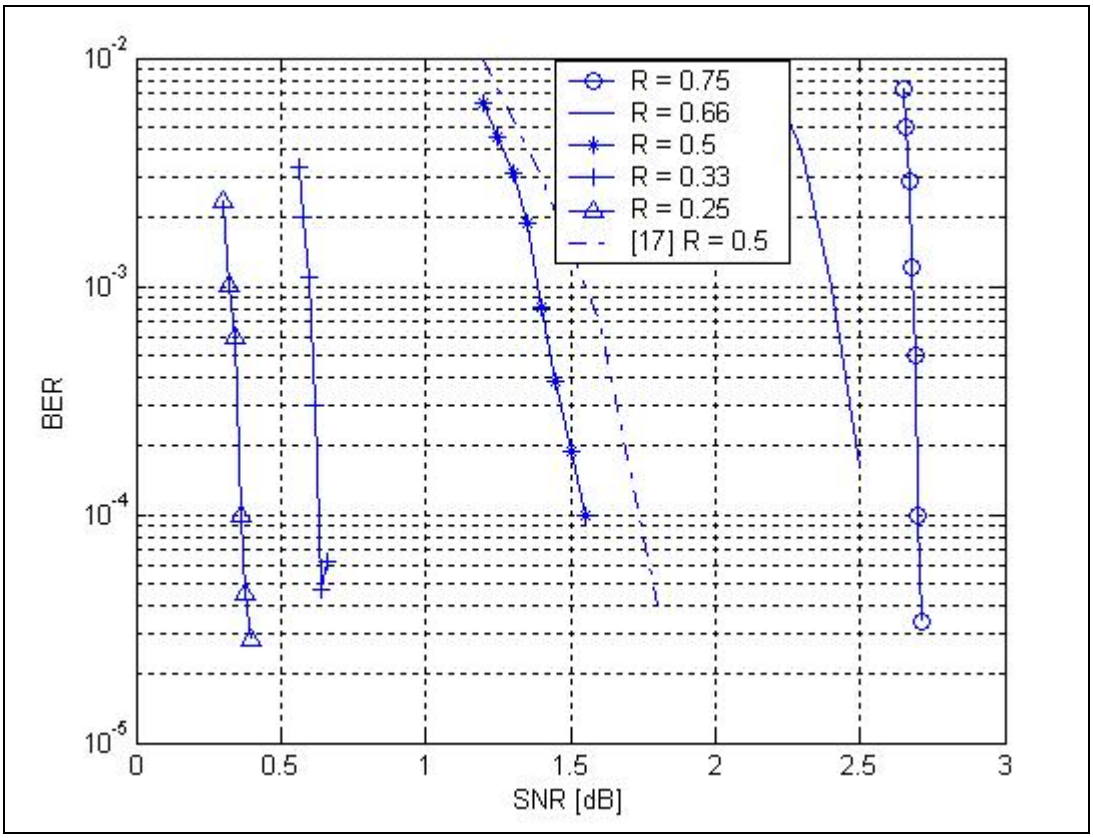
sağlamaktadır. Örneğin Tablo 5.4'te d_{lmaks} değerini 10'dan 50'ye çıkarmak, işaret - gürültü oranı bakımından 0.458 - 0.211 = 0.247 dB'lik bir kazanç sağlamaktadır.

[10]'da ise $R = 0.33$ için ve $d_{lmaks} = 50$ için bulunan eşik değeri -0.4225 dB iken aynı hıza sahip LDPC kod için bu değer Tablo 5.3'te -0.464 bulunmuştur. Bu da 0.04 dB'lik bir kazanç sağlamıştır. [10] ve [17]'de denetim düğümleri katsayıları ρ_i'lerin sayısı 3 ile sınırlandırılmışken bu çalışmada ρ_i'lerin sayısı 2 ile sınırlandırılmış ve bu sınırlandırma da [10] ve [17]'ye göre başarımı daha iyi LDPC kodlar üretilebilmesine neden olmuştur.

Ayrıca yukarıdaki tablolarda, literatürde sık olarak yer verilen $R = 0.33$ ve $R = 0.5$ kodlama oranlarına sahip LDPC kodların yanı sıra $R = 0.25$, $R = 0.66$ ve $R = 0.75$ kodlama oranlarına sahip LDPC kodlar da tasarlanmıştır. Farksal evrim algoritmasıyla düzensiz kodlar için yoğunluk evrim algoritmasının C programlama dilinde yazılmış biçimi EK – E'de, rastgele sayı üretim algoritması EK – F'de ve grafik çizim programı EK – G'de verilmiştir.

Şekil 5.2 analiz edildiğinde tasarlanan düzensiz LDPC kodların, sadece kod sözcük uzunluğu sonsuza giderken hesaplanan eşik değeriyle değil, aynı zamanda sonlu kod sözcük uzunluklarıyla da [17]'de yer alan koddan daha iyi başarım gösterdiği ve kanal sığasına yaklaştığı görülmüştür. Bu LDPC kodların kanal sığasından ve hesaplanan eşik değerinden daha uzakta başarım göstermesinin nedeni sonlu kod sözcük uzunluğu kullanılmasıdır. Halbuki hesaplanan eşik değerleri kod sözcük uzunluğunun sonsuza gitmesi durumunda geçerli olmaktadır.

Literatürde yer almayan kodlama oranlarına sahip ($R = 0.25$, $R = 0.66$ ve $R = 0.75$) optimum LDPC kodlar tasarlanmış ve başarımları Şekil 5.2'de verilmiştir. $R = 0.5$ için [17]'de yer alan LDPC koduyla bu çalışmada tasarlanan LDPC kodu karşılaştırılmış ve buradaki kodun daha iyi başarım gösterdiği görülmüştür. 10^{-4} bit hata oranına ulaşabilmek için [17]'de tasarlanan LDPC kodda yaklaşık 1.75 dB değerinde işaret - gürültü oranı gerekirken bu değer bu çalışmada tasarlanan LDPC kodda yaklaşık 1.55 dB'e indirilmiş ve yaklaşık 0.2 dB'lik bir kazanç elde edilmiştir.



Şekil 5.2: Farklı Kodlama Oranlarında Tasarlanan Düzensiz LDPC Kodlar için Bit Hata Başarımları

Ayrıca kodlama oranıyla bit hata başarımının ters orantılı olduğu da görülmüştür. Tasarlanan en iyi kod olan $R = 0.25$ kodlama oranına sahip düzensiz LDPC kod ile en kötü kod olan $R = 0.75$ kodlama oranına sahip düzensiz LDPC kodu karşılaştırıldığında 10^{-4} bit hata oranına ulaşmak için gerekli işaret - gürültü oranları arasındaki fark yaklaşık 2.4 dB olmaktadır.

6. SONUÇLAR

Bu bölümde çalışmada tasarlanmış düzenli ve düzensiz LDPC kodlara ilişkin elde edilen sonuçlara yer verilmektedir.

Düşük yoğunluklu eşlik denetim (LDPC) kodları, ya da diğer adıyla Gallager kodları ilk olarak 1962 yılında Gallager tarafından bulunmuş ve 1995 yılına kadar varolan bilgisayar sistemlerinin yetersiz kalmasından dolayı adeta unutulmuştur. 1995 yılında ise MacKay ve Neal sayesinde bu kodların üstün özellikleri yeniden keşfedilmiş ve daha sonraki yıllarda da konuyla ilgili pek çok gelişme ortaya çıkmıştır.

Tez çalışmasında düşük yoğunluklu eşlik denetim kodları incelenmiştir. Öncelikle LDPC kodların genel özellikleriyle düzenli ve düzensiz LDPC kodlar ve Tanner grafı konuları ayrıntılı bir biçimde ele alınmıştır. Daha sonra iletişim sisteminin bir parçası olan alıcı tasarımına yer verilmiştir. Burada yumuşak ve sert kararlı kod çözme teknikleri ayrıntılı bir biçimde anlatılmıştır. İletişim sisteminin diğer bir parçası olan verici tasarımına ise bir sonraki bölümde yer verilmiştir. Hem düzenli hem de düzensiz LDPC kodlar için tasarım teknikleri incelenmiş, yeni üstün kodlar sunulmuştur. Son bölümde de düzenli ve düzensiz kodlar için optimizasyon konusuna yer verilmiştir.

Bu çalışmada literatürde sıkça yer alan kodlama oranları dışında farklı kodlama oranlarına sahip LDPC kodlar tasarlanmıştır. Ayrıca ikili AWGN kanalda kod sözcük uzunluğu $N = 1000$ ve kodlama oranı $R = 0.5$ için iyi bit hata başarımı gösteren LDPC kodlar tasarlanmıştır. Tasarlanan yeni düzensiz LDPC kodların, sadece kod sözcük uzunluğu sonsuza giderken hesaplanan eşik değeriyle değil, aynı zamanda sonlu kod sözcük uzunluklarıyla da kanal sığasına yaklaştığı görülmüştür.

Karşılaştırma yapmak gerekirse düzensiz kodlar düzenli kodlara göre daha iyi başarım sunmaktadır. Ayrıca kodlama oranı arttığı zaman başarım da

düşmektedir. Düzensiz kodlar için de bit düğümlerinin derecelerinin maksimum değeri ne kadar büyükse, ilgili kod o kadar kanal sığasına yaklaşmaktadır.

İterasyon sayısı da bit hata başarımı üzerinde önemli bir rol oynamaktadır. İterasyon sayısı arttıkça tasarlanan LDPC kodlar daha iyi başarım göstermektedir.

Ayrıca kod sözcük uzunluğu N ile bit hata başarımı doğru orantılıdır. N sonsuza giderken başarım da kanal sığasına yaklaşmaktadır. Düzensiz LDPC kodlar için minimum çevrim uzunluğunun başarıma etkisi olduğu görülmüştür. Minimum çevrim uzunluğu arttığı takdirde bit hata başarımı da iyileşmektedir.

Tasarlanan üstün LDPC kodlar için işaret - gürültü oranı eşik değerleri, [10] ve [17] ile, bit hata başarımları da [17] ile karşılaştırılmış ve bu çalışmada elde edilen değerlerin daha iyi olduğu görülmüştür. R = 0.5 için [17]'de eşik SNR değerlerine bakıldığında, maksimum denetim düğüm derecesi 50 için bulunan değer 0.2485 dB iken bu çalışmada bu değer 0.211 dB'e indirilmiştir.

[10] ve [17]'de denetim düğümleri katsayıları ρ_i'lerin sayısı 3 ile sınırlandırılmışken bu çalışmada ρ_i'lerin sayısı 2 ile sınırlandırılmış ve bu sınırlandırma da [10] ve [17]'ye göre başarımı daha iyi LDPC kodlar üretilebilmesine neden olmuştur.

[10]'da R = 0.33 için bulunan eşik işaret - gürültü oranı ile bu çalışmada tasarlanan R = 0.33 kodlama oranına sahip üstün düzensiz LDPC kodu için bulunan işaret - gürültü oranı karşılaştırıldığında, bu çalışmada yer alan LDPC kodun yaklaşık 0.04 dB'lik bir kazanç sağladığı görülmüştür.

Sonlu kod sözcüklerine sahip LDPC kodlar ele alındığında ise [17]'de yer alan N = 1000 kod sözcük uzunluğuna ve R = 0.5 kodlama oranına sahip LDPC koda göre bu çalışmada tasarlanan LDPC kodun işaret - gürültü oranında yaklaşık 0.2 dB'lik bir kazanç sağladığı görülmüştür.

İletişim sisteminde optimizasyon, verici ve alıcı tekniği olarak önerilen farksal evrim tekniği, bit doldurma algoritması ve mesaj aktarma algoritması tekniklerinin oluşturdukları birleşim kullanılmıştır.

Ayrıca LDPC kodların doğal yapısında yer alan serpiştirici fonksiyonu da üstün LDPC kod tasarlama konusunda önemli bir rol oynamaktadır. Çalışmada tasarlanan en üstün LDPC kod, R = 0.5 oranında elde edilmiş olup kanal sığasından sadece 0.02 dB daha uzakta eşik değerine sahip olan bir düzensiz LDPC kod olmaktadır.

Literatürde yer almayan farklı kodlama oranlarına sahip LDPC kodlar da bu çalışmada tasarlanmıştır. Literatürde sık olarak yer verilen $R = 0.33$ ve $R = 0.5$ oranlarına sahip LDPC kodların yanı sıra $R = 0.25$, $R = 0.66$ ve $R = 0.75$ kodlama oranlarına sahip üstün LDPC kodlar da tasarlanmış ve sunulmuştur.

LDPC kodlar, özellikle düzensiz yapıya sahip olanları, en büyük rakibi olan turbo kodlara oranla hata tabanı (error floor) açısından çok daha iyi başarım sunmaktadır. Üçüncü nesil (3G) sistemlerinin gerçekleştirilmesi sürecinde yaşanan gecikme, bu sektörde LDPC kodların daha fazla kullanılabilir olmasında önemli bir rol oynayacaktır. Ayrıca çok daha az belleğe gereksinim duyan düzenli LDPC kodlar da, her ne kadar kanal sığasına düzensiz LDPC kodlar kadar yaklaşamasalar da, pratikte kullanılabilir niteliktedir.

KAYNAKLAR

[1] **R. G. Gallager**, 1962. Low - Density Parity - Check Codes, *IRE Trans. Inform. Theory*.

[2] **R. G. Gallager**, 1963. Low - Density Parity - Check Codes, *MIT Press, Cambridge, MA.*

[3] **C. E. Shannon**, 1948. A Mathematical Theory of Communication, *The Bell Syst. Techn. J.*

[4] **D. J. C. MacKay, R. M. Neal**, 1996. Near Shannon - Limit Performance of Low - Density Parity - Check Codes, *Electronics Letters*.

[5] **T. Richardson, R. Urbanke**, 2003. The Renaissance of Gallager's Low - Density Parity - Check Codes, *IEEE Communications Magazine*.

[6] **T. Richardson, R. Urbanke**, 2001. The Capacity of Low Density Parity Check Codes Under Message Passing Decoding, *IEEE Transactions on Inform. Theory*.

[7] **M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman**, 2001. Improved Low Density Parity Check Codes Using Irregular Graphs, *IEEE Trans. Inform. Theory*.

[8] **S. Y. Chung, G. D. Forney, T. J. Richardson, R. L .Urbanke**, 2001. On the Design of Low Density Parity Check Codes within 0.0045 dB of the Shannon Limit, *IEEE Commun. Letters*.

[9] **A. Prabhakar, K. Narayanan**, 2002. Pseudorandom Construction of Low Density Parity Check Codes Using Linear Congruential Sequences, *IEEE Trans. Comm. Theory*.

[10] **J. Hou, P. H. Siegel, L. B. Milstein**, 2001. Performance Analysis and Code Optimization of Low Density Parity Check Codes on Rayleigh Fading Channels, *IEEE Journal on Selected Areas in Communications*.

[11] **J. Campello, D. S. Modha, S. Rajsgopalan**, 2001. Designing LDPC Codes Using Bit - Filling, *IEEE*.

[12] **J. Campello, D. S. Modha**, 2001. Extended Bit - Filling and LDPC Code Design, *IEEE*.

[13] **R. M. Tanner**, 1981. A Recursive Approach to Low Complexity Codes, *IEEE Trans. Inform. Theory*.

[14] **F. R. Kschischang, B. J. Frey, H. A. Loeliger**, 2001. Factor Graphs and the Sum - Product Algorithm, *IEEE Transactions on Information Theory*.

[15] **M. Luby**, 2001. Efficient Erasure Correcting Codes, *IEEE Trans. Inform. Theory*.

[16] **R. M. Tanner**, 2001. Minimum Distance Bounds by Graph Analysis, *IEEE Trans. Inform. Theory*.

[17] **T. J. Richardson, M. A. Shokrollahi, R. L. Urbanke**, 2001. Design of Capacity Approaching Irregular Low Density Parity Check Codes, *IEEE Trans. Inform. Theory*.

[18] **M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann**, 1997. Practical Loss Resilient Codes, *Proc. 29th Annu. ACM Symp. Theory of Computing*.

[19] **M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman**, 1998. Analysis of Low Density Codes and Improved Designs Using Irregular Graphs, *Proc. ACM Symp*.

[20] **J. Hagenauer, E. Offer, L. Papke**, 1996. Iterative Decoding of Binary Block and Convolutional Codes, *IEEE Trans. Inform. Theory*.

[21] **J. R. Barry**, 2001. Low Density Parity Check Codes, *Georgia Institute of Technology*.

[22] **W. Ryan**, 2003. Low Density Parity Check Codes, *UCLA Extension Short Course*.

[23] **F. R. Kschischang**, 2003. Codes Defined on Graphs, *IEEE Communications Magazine*.

[24] **E. Yeo, B. Nikolic, V. Anantharam**, 2003. Iterative Decoder Architectures, *IEEE Communications Magazine*.

[25] **D. J. C. MacKay**, 1999. Good Error Correcting Codes Based on Very Sparse Matrices, *IEEE Trans. Inform. Theory*.

[26] **D. J. C. MacKay**, 1999. Comparison of Construction of Irregular Gallager Codes, *IEEE Trans. Comm. Theory*.

[27] **D. J. C. MacKay**, 1999. Encyclopedia of Sparse Graph Codes, *http://wol.ra.phy.cam.ac.uk/mackay/codes/data.html*.

[28] **S. Y. Chung, R. Urbanke, T. J. Richardson**, 2000. Gaussian Approximation for Sum Product Decoding of Low Density Parity Check Codes, *Proc. Int. Symp. Inform. Theory*.

[29] **S. Y. Chung, R. Urbanke, T. J. Richardson**, 2001. Analysis of Sum Product Decoding of Low Density Parity Check Codes Using a Gaussian Approximation, *IEEE Trans. Inform. Theory*.

[30] **H. El Gamal, A. R. Hammons, Jr.**, 2000. Analyzing the Turbo Decoder Using the Gaussian Approximation, *Proc. Int. Symp. Inform. Theory*.

[31] **S. Y. Chung, T. J. Richardson, R. L. Urbanke**, 2001. Gaussian Approximation, *http://lthcwww.epfl.ch/ldpc/gaopt.html*.

[32] **R. Storn, K. Price**, 1997. Differential Evolution – A Simple and Efficient Heuristic Adaptive Scheme for Global Optimization over Continuous Spaces, *J. Global Optimization*.

[33] **A. Shokrollahi, R. Storn**, 2000. Design of Efficient Erasure Codes with Differential Evolution, *Proc. IEEE Int. Symp. Inform. Theory*.

EKLER

Aşağıda listesi verilen algoritmaların yer aldığı ekler, tez çalışmasının arkasında bulunan CD'nin içindedir.

EK – A: Olasılık Bölgesinde Mesaj Aktarma Algoritması
EK – B: Logaritmik Bölgede Mesaj Aktarma Algoritması
EK – C: Sözde Rastgele LDPC Kodu Üretme Algoritması
EK – D: Düzenli LDPC Kodlar için Yoğunluk Evrim Algoritması
EK – E: Farksal Evrim ve Düzensiz LDPC Kodlar için Yoğunluk Evrim Algoritmaları
EK – F: Rastgele Sayı Üretimi
EK – G: Grafik Çizim Programı

ÖZGEÇMİŞ

Tolga Mataracıoğlu 07 Temmuz 1979 tarihinde İstanbul'da doğdu. İlköğretimini Özel Işık Lisesi İlkokul bölümünde tamamladıktan sonra ortaokul ve liseyi Vefa Anadolu Lisesi'nde okudu. 1997 yılında İstanbul Teknik Üniversitesi'ne girdi. 1998 yılında Prof. Dr. Faruk Umar bursu, 2000 yılında ise Vehbi Koç Vakfı bursu kazandı. 2002 yılında İstanbul Teknik Üniversitesi Elektrik - Elektronik Fakültesi Elektronik ve Haberleşme Mühendisliği bölümünden derece ile mezun oldu. Akabinde yine aynı üniversitenin Fen Bilimleri Enstitüsü'nde Telekomünikasyon Mühendisliği yüksek lisans programına girdi. 2003 yılında TÜBİTAK Bilim Adamı Yetiştirme Grubu yurt içi yüksek lisans bursu kazandı. İstanbul'da Superonline ve Ericsson'da staj yapmış ve Borusan Telekom'da çalışmış Tolga Mataracıoğlu halen Ankara TÜBİTAK Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü (UEKAE) Ağ Güvenliği Grubu'nda Sistem Güvenlik Çözümleri Sorumlusu olarak çalışmalarını sürdürmektedir.