



**İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

DOKTORA TEZİ

İTERATİF AYRIŞTIRILABİLİR BLOK KODLAR

Gökmen ALTAY

Elektrik Elektronik Anabilim Dalı

Elektrik Elektronik Programı

Danışman

Prof.Dr. Osman Nuri UÇAN

Aralık, 2005

İSTANBUL



**İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

DOKTORA TEZİ

İTERATİF AYRIŞTIRILABİLİR BLOK KODLAR

Gökmen ALTAY

Elektrik Elektronik Anabilim Dalı

Elektrik Elektronik Programı

Danışman

Prof.Dr. Osman Nuri UÇAN

Aralık, 2005

İSTANBUL

Bu çalışma 29/12/ 2005 tarihinde aşağıdaki jüri tarafından Elektrik Elektronik Anabilim Dalı Elektrik Elektronik programında Doktora Tezi olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Osman Nuri UÇAN (Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Serhat Şeker
İstanbul Teknik Üniversitesi
Elektrik Elektronik Fakültesi

Doç. Dr. Hakan Ali Çırpan
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Emin Anarım
Boğaziçi Üniversitesi
Mühendislik Fakültesi

Doç. Dr. Sedef Kent
İstanbul Teknik Üniversitesi
Elektrik Elektronik Fakültesi

ÖNSÖZ

Doktora öğrenimim sırasında bana mükemmel bir çalışma ortamı sunan, çalışmalarımı en doğru yönde yapmamı sağlayan ve her ihtiyaç duyduğumda güler yüzüyle yanımda olan çok değerli hocam Sayın Prof. Dr. Osman Nuri UÇAN'a saygı dolu teşekkürlerimi sunuyorum. Ayrıca tez izleme komitesinde yer alarak katkıda bulunan Sayın Doç. Dr. Hakan Ali ÇIRPAN ve Sayın Prof. Dr. Serhat ŞEKER'e teşekkür ederim.

Çalıştığım kurum olan Bahçeşehir Üniversitesi Mühendislik Fakültesi dekanı Sayın Prof. Dr. Şenay YALÇIN'a ve Bölüm Başkanım Sayın Prof. Dr. H. Ergun BAYRAKÇI'ya doktora çalışmalarımı yürütebilmem için sağladıkları tam destekleri için çok teşekkür ederim. Ayrıca eşime ve aileme en içten teşekkürlerimi sunarım.

Aralık, 2005

Gökmen ALTAY

İÇİNDEKİLER

ÖNSÖZ	i
İÇİNDEKİLER	ii
ŞEKİL LİSTESİ.....	iii
TABLO LİSTESİ	iv
SEMBOL LİSTESİ	v
ÖZET	vi
SUMMARY	vii
1. GİRİŞ	1
2. GENEL KISIMLAR	5
2.1. Blok Kodlar	5
2.2. Blok Kodlar İçin Kafes Yapıları	8
3. MALZEME VE YÖNTEM	11
3.1. Yumuşak Kararlı Kodçözme Ve Hata Başarımı	11
3.2. TBGG Kanal Modeli	12
3.3. Viterbi Algoritması.....	13
3.4. BCJR Algoritması.....	14
3.5. İteratif Kodçözme Algoritması	14
3.4. Toplam-Ürün Algoritması	16
4. BULGULAR	20
4.1. Geometrik Ürün Kodları İnşası	20
4.2. Hamming Mesafesi 4 olan Geometrik Ürün Kod Ailesi	24
4.3. Daha Büyük Hamming Mesafeli Geometrik Ürün Kodları İnşası.....	38
4.4. Geometrik Ürün Kodları İçin Kafes Yapıları	45
4.4.1. Hamming Mesafesi 4 olan Geometrik Ürün Kodlarının Kafesleri	49
4.4.2. Hamming Mesafesi 8 olan Geometrik Ürün Kodlarının Kafesleri	56

4.4.3. Geometrik Ürün Kodları Kafeslerinin Diğer Blok Kodlarının Kafesleri ile Karmaşıklık Yönünden Karşılaştırılması.....	60
4.5. Geometrik Ürün Kodları İçin Bilgisayar Benzetimi İle Hata Başarımı.....	69
4.5.1. Viterbi Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımları.....	70
4.5.2. BCJR Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımları	81
4.5.3. İteratif Kodçözümüyle Geometrik Ürün Kodunun Hata Başarımı	84
4.5.4. Toplam Ürün Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımları.....	85
4.6. Basamak Blok Kodları	90
4.6.1. Basamak Blok Kodları İnşası	90
4.6.2. Basamak Blok Kodlarının Bazı Üreteç Matrisleri	96
4.6.3. Toplam Ürün Algoritması Kodçözümüyle Basamak Blok Kodlarının Hata Başarımları.....	111
5. TARTIŞMA VE SONUÇ	117
KAYNAKLAR	120
ÖZGEÇMİŞ	127

ŞEKİL LİSTESİ

Şekil 1.1	: Basit bir sayısal haberleşme sistemi	2
Şekil 2.1	: (7,4) Hamming doğrusal blok kodu için kodlayıcı devresi	6
Şekil 2.2	: Basit bir kafes yapısı ve etiketlenmiş bitleri	8
Şekil 3.1	: İteratif kodçözücülü basit bir iletişim sistemi	15
Şekil 3.2	: Basit bir (8,4) ürün kodu için Tanner çizgesi	17
Şekil 4.1	: GÜ kodları için genel alt-kafes yapısı, T	45
Şekil 4.2	: GÜ kodları için genel kafes yapısı, T'	46
Şekil 4.3	: Üreteç kodkelimesinin, g_4 , alt-kafesine etiketlenmesi	49
Şekil 4.4	: $C = (12, 7, 4)$ GÜ kodu kafes yapısı	50
Şekil 4.5	: $C = (14, 9, 4)$ GÜ kodu kafes yapısı	51
Şekil 4.6	: $C = (16, 11, 4)$ GÜ kodu kafes yapısı	52
Şekil 4.7	: $C = (18, 12, 4)$ GÜ kodu kafes yapısı	53
Şekil 4.8	: $C = (20, 14, 4)$ GÜ kodu kafes yapısı	54
Şekil 4.9	: $C = (22, 16, 4)$ GÜ kodu alt-kafesi, T	55
Şekil 4.10	: $C = (24, 18, 4)$ GÜ kodu alt-kafesi, T	55
Şekil 4.11	: $C = (26, 20, 4)$ GÜ kodu alt-kafesi, T	56
Şekil 4.12	: $C = (28, 22, 4)$ GÜ kodu alt-kafesi, T	56
Şekil 4.13	: $C = (16, 5, 8)$ GÜ kodu kafes yapısı, T'	57
Şekil 4.14	: $C = (24, 9, 8)$ GÜ kodu kafes yapısı, T'	58
Şekil 4.15	: $C = (32, 16, 8)$ GÜ kodu kafes yapısı, T'	59
Şekil 4.16	: $C = (64, 19, 16)$ GÜ kodu alt-kafesi, T	60
Şekil 4.17	: $C = (12, 7, 4)$ GAC kodu kafes yapısı [82]	61
Şekil 4.18	: $C = (16, 11, 4)$ GAC kodu kafes yapısı [82]	62
Şekil 4.19	: $C = (12, 6, 4)$ Dizi kodunun kafes yapısı [82]	63
Şekil 4.20	: $C = (16, 9)$ Dizi kodunun kafes yapısı [82]	64
Şekil 4.21	: $C' = (7, 3, 5)$ RS kodu kafes yapısı [82]	65
Şekil 4.22	: $C = (12, 7, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	70
Şekil 4.23	: $C = (14, 9, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	71
Şekil 4.24	: $C = (16, 11, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	72
Şekil 4.25	: $C = (18, 12, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	73
Şekil 4.26	: $C = (20, 14, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	74
Şekil 4.27	: $C = (22, 16, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	75
Şekil 4.28	: $C = (24, 18, 4)$ GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	76

Şekil 4.29	: C = (28, 22, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	77
Şekil 4.30	: C = (16, 5, 8) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	78
Şekil 4.31	: C = (24, 9, 8) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BHO	79
Şekil 4.32	: C = (32, 15, 8) GÜ ve (32, 16) RM kodlarının ve kodlanmamış bitlerin AWGN kanalda Viterbi ile BHO	80
Şekil 4.33	: C = (12, 7, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BCJR algoritması yardımıyla elde edilen BHO.....	81
Şekil 4.34	: C = (16, 11, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BCJR algoritması yardımıyla elde edilen BHO.....	82
Şekil 4.35	: C = (20, 14, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda BCJR algoritması yardımıyla elde edilen BHO.....	83
Şekil 4.36	: C = (12, 7, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda iteratif kodçözme algoritması yardımıyla elde edilen BHO.	84
Şekil 4.37	: C = (12, 7, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile elde edilen BHO	85
Şekil 4.38	: C = (16, 11, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile elde edilen BHO	86
Şekil 4.39	: C = (28, 22, 4) GÜ kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO	87
Şekil 4.40	: (16, 11, 4) GÜ ve (15, 11, 3) Hamming kodlarının ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile 8 iterasyonda elde edilen BHO	88
Şekil 4.41	: (8, 4, 4) GÜ ve (7, 4, 3) Hamming kodlarının ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile 8 iterasyonda elde edilen BHO	89
Şekil 4.42	: C = (12, 7, 4) BB kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün kodçözme algoritması ile elde edilen BHO.....	111
Şekil 4.43	: C = (16, 11, 4) BB kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile elde edilen BHO.....	112
Şekil 4.44	: C = (28, 22, 4) BB kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile elde edilen BHO.....	113
Şekil 4.45	: C = (32, 26, 4) BB kodunun ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması yardımıyla elde edilen BHO.....	114
Şekil 4.46	: (8, 4, 4) BB ve (7, 4, 3) Hamming kodlarının ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile 16 iterasyonda elde edilen BHO.....	115
Şekil 4.47	: (12, 7, 4) BB ve (14, 8) LDPC kodlarının ve kodlanmamış bitlerin AWGN kanalda toplam-ürün algoritması ile 200 iterasyonda elde edilen BHO.....	116

TABLO LİSTESİ

Tablo 4.1	: Bilgisayar ile oluşturulan ve sağlaması yapılan bazı GÜ kodları	26
Tablo 4.2	: Bazı GÜ ve diğer blok kodlar için kafes karmaşıklıkları.....	66

SEMBOL LİSTESİ

GÜ	: Geometrik Ürün
BB	: Basamak Blok
TBGG	: Toplamsal beyaz Gauss gürültülü (Additive White Gaussian Noise)
G	: Üreteç matrisi
C	: Blok kod
E(C)	: Dal karmaşıklığı
 V 	: Durum karmaşıklığı
KK	: Kafes Karmaşıklığı
İFKA	: İkili Faz Kaydırmalı Anahtarlama
BHO	: Bit Hata Oranı
E_b	: Bir bilgi biti başına düşen enerji miktarı
N₀	: Güç tayf yoğunluğu
σ²	: Varyans (Değişinti)
BCJR	: Bahl, Cocke, Jelinek ve Raviv
T	: Alt-kafes yapısı
T'	: Genel kafes yapısı
GAC	: Genelleştirilmiş Dizi Kodları (Generalised Array Codes)
dB	: Desibel

ÖZET

İTERATİF AYRIŞTIRILABİLİR BLOK KODLAR

Güvenilir ve kaliteli sayısal haberleşme sistemlerinin geliştirilebilmesi için etkin kanal kodlama yöntemlerinin ve bunlara uygun kodçözme algoritmalarının bulunması halen gelişmekte olan önemli bir araştırma konusudur. Herhangi bir kanal kodlama tekniğinde gerçekleştirilen en temel işlem, iletilecek mesaj bitlerine fazlalık bitleri ekleyerek kodlamaktır. Kanal kodlama yöntemlerini birbirinden ayıran esas fark, eklenen bu fazlalık bitlerinin nasıl ve ne kadar eklendiğidir. İstenilen belirli bir özellikteki kodlamayı sağlayacak iki farklı kodlama yönteminden, pratikte uygulaması daha kolay olan ve daha az fazlalık bitleri ekleyen yöntem daha etkin bir tekniktir. Çünkü, bir çok uygulamada iletimin gerçekleştirileceği kanalın bant genişliği sınırlıdır. Bu tez çalışmasında, yeni bir kanal kodlama yöntemi olan Geometrik Ürün (GÜ) kodlaması sunulmuştur. Önerilen bu yöntem ile elde edilen blok kodlar genel yapıda olup uzunluğu 8'den büyük ve Hamming mesafesi 4 olan bütün en iyi çift blok kodları, bir kod ailesi olarak üretebilmektedir. Geliştirilen kodlama yöntemi genel bir yapıdadır ve daha büyük minimum Hamming mesafeli kodları da üretebilmektedir.

Blok kodların kodçözümünde, Viterbi ve BCJR gibi kafes yapısına ihtiyaç duyan kodçözme algoritmalarını kullanabilmek için, her bir blok kod tekniğine ayrı bir kafes yapısı tasarlanmalıdır. Bir blok kod için birden fazla yapıda kafes yapısı dizayn edilebilir; fakat önemli olan en az işlemle kodçözmeye olanak veren bir kafes yapısı tasarlamaktır. Bu tezde sunulan GÜ kodlarının kodçözümünde kullanılabilecek kafes yapıları, minimum karmaşıklık yapıda ve kolay uygulanabilir esneklikte tasarlanmıştır. Tasarlanan kafes yapılarının en az karmaşıklığa sahip olduğu literatürdeki diğer blok kodların kafes yapıları ile karşılaştırılarak gösterilmiştir.

Bu çalışmada ortaya konan GÜ kodlarının hata başarımları, çeşitli kodçözme algoritmaları ve tasarlanan kafes yapıları kullanılarak, toplamsal beyaz Gauss gürültülü kanal ortamında, bilgisayar benzetimi yardımıyla elde edilmiştir.

Önerilen GÜ kodlarından yola çıkılarak, kodlayıcı uygulamasında kolaylık sağlayabilen ve GÜ kodu ile aynı kod ölçülerine sahip bir başka blok kodlama tekniği de geliştirilmiş olup basamaklı üreteç matrisi yapısından dolayı Basamak Blok (BB) kodları olarak isimlendirilmiştir. BB kodlarının farkları ve avantajları detayları ile açıklanmıştır.

BB kodlarından bazıları için hata başarımları toplam-ürün algoritması kullanılarak, toplamsal beyaz Gauss gürültülü kanalda, bilgisayar benzetimi yardımıyla elde edilmiştir.

Bu tezde önerilen GÜ ve BB kodları ile üretilen minimum Hamming mesafesi 4 olan kod aileleri, genişletilmiş Hamming kodlarını ve Reed-Muller kodlarından minimum

Hamming mesafesi 4 olanlarını içine alır. Çünkü bu kodların uzunlukları 2'nin kuvvetleri olarak değişmektedir, diğer taraftan GÜ ve BB kod aileleri 2'nin katları olan kod uzunluklarına sahiptir. Dolayısıyla GÜ ve BB kodları, genişletilmiş Hamming kodlarını ve minimum Hamming mesafesi 4 olan Reed-Muller kodlarını kapsayan daha üst bir kod ailesi olarak karşımıza çıkmaktadır. İleride, bu tez ile sunulan GÜ ve BB kodları üzerine yapılabilecek çalışmaları: Daha büyük Hamming mesafeli GÜ ve BB kodlarını da, 4 Hamming mesafeli olan GÜ ve BB kodlarındaki gibi en iyi kodlama oranını verecek şekilde geliştirme, kodlayıcı ve kodçözücü çip tasarımı, değişik modülasyon teknikleriyle ve başka kanallarda hata başaım analizleri gibi çalışmalar olarak sıralayabiliriz..

SUMMARY

ITERATIVE DECOMPOSABLE BLOCK CODES

Reliable and quality communication systems require effective channel coding methods and suitable decoding algorithms. Channel coding process is basically adding redundant bit into message bits. The difference among channel coding techniques is how and how many redundant bits are added into message bits. In this study, a new channel coding technique, Geometric Product (GP) codes are introduced. The codes obtained from the technique are in general form and the technique can generate all the even length binary block codes with Hamming distance 4, which have the length greater than or equal 8. Since the proposed coding method is a general approach, it can also generate higher minimum Hamming distance codes.

Trellis structures are designed in order to use them in decoding GP codes. The designed trellises contain minimal complexity and are flexible regarding easier implementation. It is shown that the designed trellises are less complex than other block codes trellises in the literature.

Error performances of some of the proposed codes are obtained with the help of computer simulation in additive white Gaussian noise channel using variety of decoding techniques such as Viterbi, BJCR and iterative decoding algorithms.

Another new block coding technique, which is similar to GP coding but has less complex encoding structure, is also introduced and it is named as Step Block (SB) codes because of its generator matrix structure.

For some of SB codes, error performances are obtained with the help of computer simulation in additive white Gaussian noise channel using sum-product algorithm.

The minimum Hamming distance 4 code families of the proposed GP and SB codes include the extended Hamming codes and Reed-Muller codes of Hamming distance 4. It is because the lengths of these codes are the power of 2 whereas the lengths of the proposed codes are the multiple of 2. As the code sizes of GP and SB codes covers the extended Hamming codes and Reed-Muller codes of Hamming distance 4, they stand as superior code families.

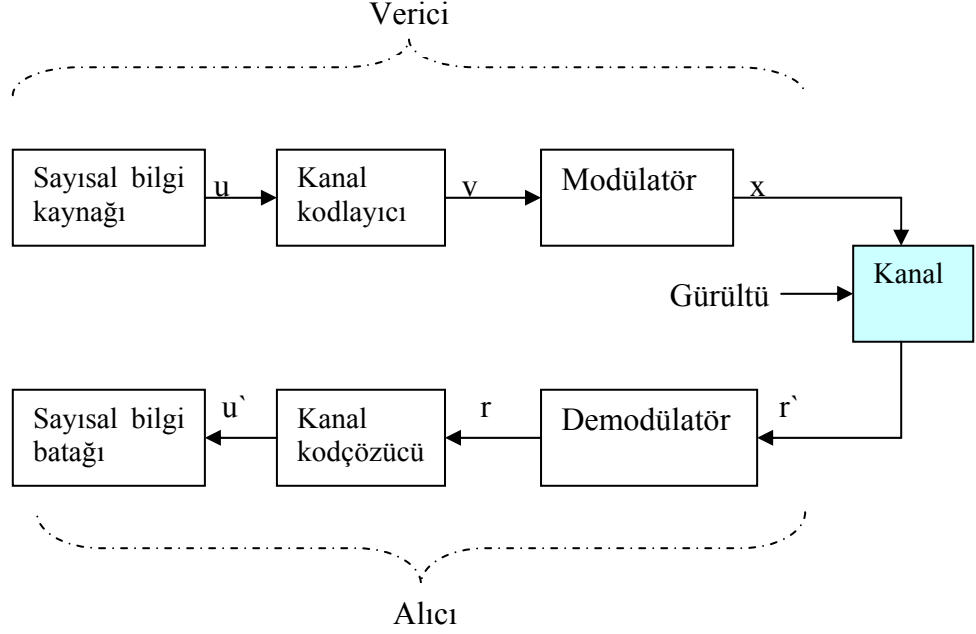
1. GİRİŞ

Bilgisayar ağıları ve cep telefonlarının hayatın her alanına yayılmasıyla, hızlı ve hatasız sayısal bilgi iletimi gereksinimi sürekli artmaktadır. Haberleşme sistemlerinde bilgiyi bir yerden başka bir yere iletmek için vericiden gönderilen mesaj işareti, işaretin gönderildiği kanaldan (örneğin kablo, hava, fiber optik) ve aynı zamanda verici ve alıcı devrelerinden kaynaklanan işareti bozucu etkilerden yani gürültüden dolayı, bozulmaya uğrar ve alıcıda gönderilenden farklı bir bilgi alınabilir. Bu durum haberleşme sisteminin hatalı mesaj göndermesi demektir. Kanal kodlama konusu, sayısal haberleşme sistemindeki alıcıda alınan bilgi bitlerinde meydana gelebilen bu hataları ortadan kaldırmayı amaçlayan çok geniş bir araştırma alanıdır. Kanal kodlama modern sayısal haberleşme ve bellek sistemlerinin ayrılmaz bir parçasıdır.

Kanal kodlama teoreminde kanal kapasitesinin teorik sınırı ile ilgili olarak, Shannon [1] 1948’de ortaya koyduğu teoremden, kodlama oranının kanal kapasitesinden küçük olması durumunda, bilginin uygun bir biçimde kodlanmasıyla, gürültülü bir kanalda bilgi iletimi esnasında oluşan hataların istenilen en düşük oranlara indirilebileceğini ispatlamış; fakat bunun nasıl yapılacağını açık bir problem olarak bırakmıştır [2]. Önerilen bu teorem ile yeni kanal kodlama araştırma alanı ortaya çıkmıştır. Bu alanda yapılan çalışmaların en önemli amaçlarından bir tanesi, kodlaması ve kodçözmesi kolay ve de mümkün olduğunca küçük hata oranları sağlayan kodlar bulmaktır [3].

Sayısal haberleşme sisteminde iletilecek bilgi bitleri çok değişik amaçlara yönelik bir çok işlemden geçirilir. Tez konusu kanal kodlama ile ilgili olduğundan, Şekil 1.1 ile gösterdiğimiz basit bir haberleşme sisteminde, en temel işlem blokları ve kanal kodlama işlemi ile ilgili bloklar gösterilmiştir. Verici tarafında, *sayısal bilgi kaynağından* üretilen bilgi dizisi, u , *kanal kodlayıcı* bloğunda bir kanal kodlama algoritmasıyla kodlanması, v , olarak kodlanır. *Modülatör*, kanal kodlayıcı çıkışındaki sembolleri, kanalda iletme uygun dalga biçimine çevirir. Kanalda gürültüye maruz kalan ve bozulmaya uğrayan dalga biçimi, r , alıcı kısmında ki demodülatör aracılığı ile alınan

mesaj dizisi, r , olarak *kanal kodçözücü* bloğuna iletilir. *kanal kodçözücü* bloğunda en büyük olasılıkla kestirilen bilgi dizisi, u' , elde edilir.



Şekil 1.1: Basit bir sayısal haberleşme sistemi.

Günümüzde iki ayrı temel yapıda kanal kodlama biçimi mevcuttur. Bunlar *blok kodlar* ve *katlamalı kodlardır*. Bir blok kodlama tekniği kullanan bir kanal kodlayıcı, sayısal bilgi kaynağından üretilen bilgi dizisini k bitlik mesaj bloklarına ayırır ve bu $\mathbf{u} = (u_0, u_1, \dots, u_{k-1})$ sembolü ile temsil edilir. Kodlayıcı bu $\mathbf{u}$ mesaj bloğunu n bitlik $\mathbf{v} = (v_0, v_1, \dots, v_{n-1})$ kodkelimesi sembollerine kodlar. Kodlayıcının *kodlama oranı* $R = k / n$ ile tanımlanır. Bu durumda n bitlik kodlayıcı çıkışı sadece k bitlik giriş bitlerine bağlıdır ve bunu gerçekleyen kanal kodlayıcı da belleksiz bir kombinasyonel lojik devre ile gerçekleştirilebilir. *Katlamalı kodlar* ise bellekli yapısından dolayı çıkış bitleri her zaman daha önce kodlayıcıya giren bitlere bağlı olacaktır [2, 3]. Tezin içinde bahsedilecek tüm kodlama işlemleri ikili (binary) kodlama işlemine göre yani mod-2 aritmetiğine göre yapılacaktır.

Literatürde bulunan en önemli blok kodlardan bazıları, Hamming kodları [4], Reed-Muller kodları [5, 6], ürün (product) kodları [7], $|u|u+v|$ inşası kodları [8, 9], Golay kod [10, 8, 11, 12], çevrimsel (cyclic) kodlar [13], BCH (Bose, Chaudhuri and

Hocquenghem) kodları [14,15], Reed-Solomon kodları [16], Genelleştirilmiş dizi kodları (Generalised array codes, GAC) [17], Turbo kodları [18, 19, 20], Düşük yoğunluklu parite kontrol kodları (Low Density Parity Check Codes, LDPC) [21, 22, 23] ve ürün yığılma kodları (Product Accumulate, PA, codes) [24] olarak sıralanabilir. Bu kodlar kullanılarak daha başka yapıda ve güçlü kodlar tasarlamak halen başvuru bir yöntemdir. Mesela Turbo kodları [18, 19] basit yapıda bazı katlamalı kodları yeni bir yöntemle birleştirilerek elde edilmiştir. Daha sonrada katlamalı kodlar yerine ürün kodları [7] gibi başka temel kodlar da kullanılarak Turbo kodlar elde edilmiştir. Bu tez çalışmasında literatürde bilinen ürün kodları temel alınarak, kodlama oranı, ürün kodlaması ile elde edilen kodlardan daha yüksek olan, yeni bir kodlama yöntemi geliştirilmiştir.

Bu tezde, temelde kodlayıcı yapısını kolaylaştıran, kolay uygulanabilir, pratik, istenilen kod uzunluğunu sağlayan esneklikte ve daha da önemlisi az karmaşık kafes yapılarını tasarlamayı sağlayan yeni bir blok kodlama tekniği önerilecektir. Buna ek olarak, önerilen kodun kodçözümünü az karmaşıklıkla çözmeyi sağlayan genel yapıdaki minimum kafes yapıları ve bu kafes yapıları kullanılarak, önerilen kodların *toplamsal beyaz Gauss gürültülü, TBGG, (Additive White Gaussian Noise, TBGG)* [25, 26] kanallardaki, bilgisayar benzetimleri ile elde edilen hata başarımları, bir çok değişik kod boyutları ve değişik kodçözme yöntemleri için sunulacaktır. Ayrıca önerilen yeni kodlama yöntemi ile aynı ölçülerde, ona benzer, fakat temel kod olan ürün kodlarını kullanmayan, basit kodlama yapısındaki bir başka yeni kodlama tekniği de avantajları belirtilerek sunulacaktır.

Tezin bölümleri şu şekilde düzenlemiştir: 2. bölümde Genel Kısımlar başlığı altında *blok kodların* ve tezde sıkça geçen parametrelerinin basit bir tanıtımı yapılacaktır. Ayrıca kodçözümünde kullanılan kafes yapıları ve önemli parametreleri tanıtılacaktır. 3. bölümde Malzeme ve Yöntem başlığı altında tez çalışması esnasında bilgisayar benzetimi yapılırken kullanılan her türlü malzeme, yöntem ve teorilerin tanıtımı yapılacaktır. 4. bölümde Bulgular başlığı altında kısımlar halinde, önerilen yeni kodlama yönteminin genel bir tanıtımı yapılacak ve alt kısımlar halinde bu kodlama yöntemi ile elde edilen kod aileleri sunulacak, önerilen kodların kodçözümünde kullanılması için tasarlanan yeni ve genel kafes yapıları detaylarıyla beraber

sunulacaktır. Önerilen kod yapılarına benzer fakat daha kolay kodlayıcı yapısı tasarlamaya olanak veren bir başka yeni blok kodlama yöntemi sunulacak ve son olarak önerilen kodların bilgisayar benzetimleri kullanılarak elde edilen hata başarımları sunulacaktır. 5. bölümde bulgular literatürle karşılaştırılarak değerlendirilecek, sonuçlar detaylı ve açık olarak ifade edilecek ve çalışmanın geliştirilmesi için ileride yapılacak araştırmalara yönelik önermeler yapılacaktır.

2. GENEL KISIMLAR

Bu bölümde tez çalışmasıyla doğrudan ilgili olan ve tez çalışması sonunda ortaya konan bulguların daha iyi anlaşılabilmesi için bilinmesi gerekli olan kanal kodlama ile ilgili temel konular incelenecektir.

2.1. BLOK KODLAR

Günümüzde veri iletiminde kodlama işlemi genellikle ikili biçimde 0 ve 1 ile yapıldığından, bu kısımda *doğrusal blok kodlar* tanıtılacaktır. Tez çalışması sonucu elde edilen kodlar da doğrusal blok kodların özelliğine sahiptirler. Doğrusal bir blok kod, C , en temelde bir *üreteç matrisi*, G , ile tanımlanır. Kodlanacak k bitlik mesaj bloğu (u) üreteç matrisi ile çarpılarak n bitlik kodkelimesine (v) kodlanır. Yapılan işlem matematiksel olarak ikili aritmetik uzayında şu denklemle tanımlanabilir.

$$v = u \cdot G \quad (2.1)$$

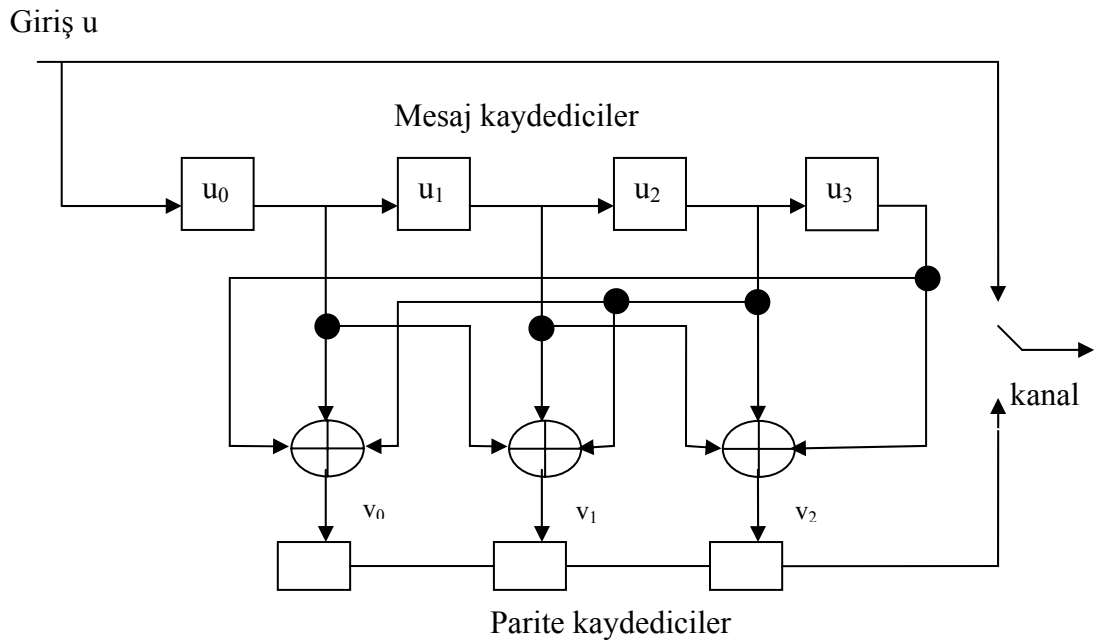
Bu denklemde G üreteç matrisinin boyutları u ve v dizilerinin boyutlarına bağlı olduğundan k satırlık ve n sütunluk bir matristir ve de k adet satırlarından her biri doğrusal bağımsız diziler, yani kodkelimeleridir. Bu matrisin doğrusal kombinasyonlarından oluşabilecek 2^k adet kodkelimleri vardır. Bunlardan oluşan kümeye *blok kod* denir ve böyle bir blok kod, C , (n, k) blok kodu olarak tanımlanabilir. Bir (n, k) doğrusal blok kodu, herhangi k adet doğrusal bağımsız kodkelimelerinin oluşturduğu üreteç matrisi ile elde edilebilir. Bu anlamı, bir blok kodun birden fazla üreteç matrisi olabilir demektir. Aşağıda verilen Örnek 2.1’de literatürde en çok karşılaşılan $[2, 3, 4]$ en basit yapıda olan bir $(7, 4)$ Hamming doğrusal blok kodu ve üreteç matrisi gösterilmektedir.

Örnek 2.1: Bir (7,4) Hamming doğrusal blok kodu, (2.1) denklemiyle ikili aritmetik uzayında şöyle elde edilebilir [2]:

<u>u</u>	<u>y</u>
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0100111
0101	0101100
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010011
1011	1011000
1100	1100010
1101	1101001
1110	1110100
1111	1111111

$$\mathbf{G} = \begin{bmatrix} 1000101 \\ 0100111 \\ 0010110 \\ 0001011 \end{bmatrix}$$

Böyle bir kodun kodlayıcı devresi aşağıdaki gibidir [2].



Şekil 2.1: (7,4) Hamming doğrusal blok kodu için kodlayıcı devresi.

Şekil 2.1 ile gösterilen kodlayıcı devresindeki bazı elemanlar şöyle tanımlıdır:

Dikdörtgen biçimindeki eleman, , bir kaydedicidir, bir sonra gelecek olan mesaj biti gelene kadar içindeki bilgiyi tutar.

Elips içinde artı biçimindeki eleman, , ikili toplayıcıdır.

İçi dolu daireler, , lehim noktalarını belirtir.

Genel itibariyle yukarıdaki örnekte gösterilen bir blok kodun bazı önemli özelliklerini temsil eden *minimum Hamming mesafesi*, *Hamming ağırlığı*, *parite kontrol matrisi* gibi parametreleri vardır. Bir v kodkelimesinin *Hamming ağırlığı* veya sadece ağırlığı $w(v)$ ile temsil edilir ve v 'nin sıfırdan farklı elemanlarının sayısı olarak tanımlanır. Aynı uzunlukta iki ayrı kodkelimesi (v_1 ve v_2) arasındaki *Hamming mesafesi* veya sadece mesafesi $d(v_1, v_2)$ ile temsil edilir ve her iki kodkelimesinin birbirinden farklı bitlerinin sayısı olarak tanımlanır. Ayrıca aynı uzunlukta iki ayrı kodkelimesinin Hamming mesafesi; v_1 ve v_2 toplamından oluşan kod kelimesinin Hamming ağırlığı olarak da tanımlıdır [2, 3, 26].

$$d(v_1, v_2) = w(v_1 + v_2) \quad (2.2)$$

Bir doğrusal blok kodun *minimum mesafesi* yani *Hamming mesafesi*, $d_{\min}$, o kodun kodkelimeleri içinde minimum Hamming ağırlıklı olan ağırlık değerine eşittir. Bir blok kodu C , (n, k, d) parametreleriyle gösterilebilir. Bu durumda n kodun uzunluğu, yani kodkelimelerindeki bitlerin sayısı, k kodun boyutu, yani mesaj bitlerinin sayısı, ve d ise kodun minimum Hamming mesafesidir.

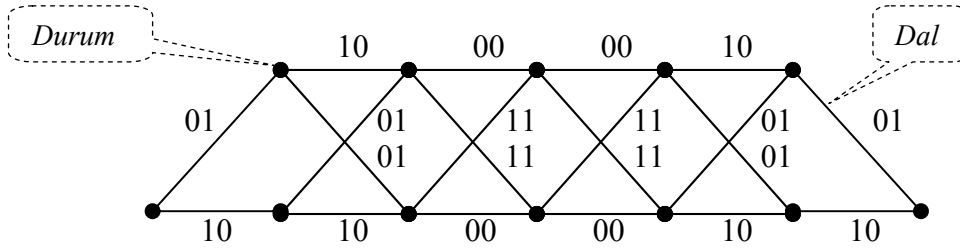
Blok kodların kodçözümünde kullanılan bir *parite kontrol matrisi* vardır ve genelde H ile temsil edilir. Parite kontrol matrisi, H , ait olduğu kodu üreten üreteç matrisi G 'ye diktir ve aşağıda (2.3) denklemleriyle tanımlanan özelliğe sahiptir.

$$G \cdot H^T = 0 \quad \text{veya} \quad H \cdot G^T = 0 \quad (2.3)$$

2.2. BLOK KODLAR İÇİN KAFES YAPILARI

Kafes yapıları 1973'te ilk defa Forney [27] tarafından katlamalı kodların Viterbi algoritması [28] yardımıyla kodçözümünü açıklamak için kullanılmıştır. Doğrusal blok kodları içinse kafes yapıları ilk defa 1974'te Bahl ve arkadaşları [29] tarafından ortaya konulmuştur. Bu konuda daha kapsamlı çalışmalar ile kafes yapılarının nasıl inşa edileceği ve önemli özellikleri gibi problemler üzerine araştırmalar yapılmıştır [30, 31, 32].

Blok kodların kodçözümünde de katlamalı kodlarda olduğu gibi *kafes* yapıları kullanılabilir. Viterbi algoritması ve Bahl, Cocke, Jelinek ve Raviv (BCJR) algoritması gibi kodçözme algoritmalarının uygulanması kafes yapıları üzerinden mümkündür [27, 28, 29]. Bir blok kodu grafiksel olarak temsil etmek için en bilinen teknik *kafes* yapılarıdır. Bir kodun *kafes diyagramı*, dalları etiketlenmiş ve yönlendirilmiş olmalı ve aynı zamanda baş durumdan son duruma kadar dalların birleşiminden oluşan her bir *yol* veya *patika*, kodun bir kodkelimesini temsil ediyor olmalıdır [2].



Şekil 2.2: Basit bir kafes yapısı ve etiketlenmiş bitleri.

Şekil 2.2.'de basit bir kafes yapısı ve üzerine etiketlenen kodkelimesi bitleri gösterilmiştir [85]. Ayrıca *durum* (düğüm) ve *dal* gibi kafeslerin önemli kavramları da şekil üzerinde belirtilmiştir. Kodkelimelerinden kafes dalları üzerine etiketlenen ardışık bit gruplarının arasındaki bağlantı noktası *durum* olarak tanımlanabilir. *Durum* aynı zamanda, kodkelimesi üzerindeki, her bir zamanını da temsil eder. Kafes yapısı içinde bir durumdan diğerine geçişte veya bir zaman anından diğer ana geçişte aradaki bağlantı dal ile temsil edilir. Kodkelimesindeki bitler bu dallara etiketlenir. Dal, durum ve bir dala etiketlenen bitlerin sayısı, bir kodun kafes yapısının karakteristiğini gösteren en temel öğelerdir.

Kafes yapıları, Viterbi algoritması veya Bahl, Cocke, Jelinek ve Raviv (BCJR) [29] algoritmasını kullanarak, bir blok kodun *en büyük olabilirlikli* (EBO) yumuşak kararlı kodçözümüne olanak verir. Diğer taraftan, blok kodlar için kafes yapısı tabanlı kodçözücü uygulamaları kafes yapılarının karmaşıklığıyla sınırlandırılmıştır. Genel olarak blok kodlar için kafes yapısı karmaşıklığı, kafesin dalları ve durumlarının sayısına bağlı olarak ifade edilir. Bazı çalışmalar [33, 34] *durum karmaşıklığı* bir kodun karakteristiğini ifade eden baskın parametre olarak yorumlasalar da, Forney [35] *dal karmaşıklığının* daha önemli bir parametre olduğunu vurgulayarak Viterbi algoritması ile kodçözümü için kafes dallarının sayısının ve ikili karşılaştırma durumlarının sayısının *karmaşıklık* için daha önemli parametreler olduğunu belirtir. Blok kodlarda kafes karmaşıklığı kod uzunluğu arttıkça üstel olarak arttığından, $n = 100$ gibi büyük uzunluklarda blok kodların kafes yapıları kullanılarak Viterbi algoritması ile kodçözümü imkansız hale gelir. Dolayısıyla az karmaşık ve pratikte uygulanabilir kafes yapılarının bulunması blok kodlar için önemli bir problemdir. Literatürde kafes yapıları üzerine çok önemli çalışmalar yapılmıştır [36 - 46].

McEliece, bir kodun kodçözümünün, Viterbi algoritması kullanılması durumunda, işlem sayısı niceliğini o kodun kafes karmaşıklığı olarak tanımlamıştır. Bunu da bir kodun *dal karmaşıklığı*, $E(C)$, ve *durum karmaşıklığı*, $|V|$, değerlerini kullanarak aşağıdaki gibi formüle etmiştir [43]:

$$E(C) = \sum_i P^{b_i} \quad (2.4)$$

Burada p kullanılan kodun alfabesidir. Daha önce açıklandığı gibi tez boyunca yapılan işlemlerde ikili kod alfabesi kullanılmaktadır ve dolayısıyla bu durumda $p = 2$ olarak alınır. Üstel ifade b_i ise dal uzayının i zamanındaki boyutu, yani bir dala i zamanında etiketlenen bitlerin sayısıdır.

Bu durumda Viterbi algoritması ile kodçözümü gerçekleştirilen bir kodun *kafes karmaşıklığı*, KK , şu formülle tanımlanır [43, 47].

$$KK = 2E(C) - |V| + 1 \quad (2.5)$$

Burada $|V|$ ise kafes içindeki durumların sayısı olarak tanımlıdır. Bu gösterilen eşitlik, ilerideki bölümlerde, önerilen kafes yapılarının kafes karmaşıklıklarının karşılaştırılması için kullanılacaktır.

3. MALZEME VE YÖNTEM

Bu çalışmada yeni bir kanal kodlama yöntemi ve yeni kafes yapıları gibi teorik sonuçlar elde edilmiş olmakla beraber, önerilen bu kodların en genel bir haberleşme sistemi olan TBGG kanalda iletilmesi durumundaki hata performansının incelenmesi için bilgisayar benzetimleri gerçekleştirilmiştir. Bilgisayar benzetimi için gerekli olan malzemeler ise yüksek hızlı bir bilgisayar, MATLAB ve C gibi programlama dilleridir. Benzetimler yapılırken *İkili Faz Kaydırmalı Anahtarlama, İFKA (Binary Phase Shift Keying, BPSK)*, modülasyonu ile modülatörden gönderilen işaretler TBGG kanalda bozulduktan sonra Viterbi algoritması, Bahl, Cocke, Jelinek ve Raviv (BCJR) algoritması, Toplam-ürün algoritması ve iteratif kodçözme yöntemleri ile kodçözümü yapıldığından, bu bölümde bu yöntemler hakkında genel bir bilgi verilecektir.

3.1. YUMUŞAK KARARLI KODÇÖZME VE HATA BAŞARIMI

Sert Kararlı Kodçözme tekniğinde alıcıda alınan sembol değeri, eşik değerine göre kıyaslanarak mantıksal 1 veya 0 bitine karar verilir. *Yumuşak Kararlı Kodçözme* tekniğinde ise sembol değerlerine göre elde edilen olasılıklardan yola çıkılarak karar verilir. Bu tez çalışmasında kullanılan İFKA modülasyonunda modülatör, mantıksal 0 bitini -1 ve mantıksal 1 bitini de +1 değerine atar.

Bit Hata Oranı (BHO), veya diğer ifadesiyle bit hatasının olasılığı, bir iletişim esnasında alıcıda bulunan kodçözücü çıkışında hatalı olarak kodu çözülen bilgi bitleri sayısının alınan toplam bilgi biti sayısına oranı olarak tanımlıdır. Bir kodun hata başarımı için en standart ölçüt BHO' dur. Bundan başka bir de *Kelime Hata Oranı, KHO*, veya *Blok Hata Oranı, BLHO*, vardır ki bu da kodçözücü çıkışındaki kodu çözülen kodkelimelerinin hatalı kodçözülme olasılığı olarak tanımlıdır.

Eğer vericiden gönderilen sembolün, yani elektriksel işaretin enerjisi E_s ise, gönderilen bir bilgi biti başına düşen enerji miktarı $E_b = E_s/R$ olur. Burada $R = k/n$ kod oranıdır.

Buna göre başarımlı analizinde kullanılan temel parametre *İşaret-Gürültü Oranı, İGO* (signal-to-noise ratio, SNR), E_b/N_0 olarak tanımlanır ki burada $N_0/2$ çift taraflı gürültü *Güç Tayf Yoğunluğudur, GTY*. İGO veya BHO'na bakılarak *kodlama kazancı* hesaplanabilir.

3.2. TBGG KANAL MODELİ

Bir iletim kanalını matematiksel modellemek, bir kodun başarımlı inceleme için çok faydalıdır. En genel ve en önemli kanal modeli *Toplamsal Beyaz Gauss Gürültülü (TBGG)* kanaldır. Kablo ve hava gibi ortamlar bu kanala örnek olarak verilebilir. Kanaldaki işaretin bozulmasının benzetimi, sıfır ortalamalı, toplamsal, beyaz, normal dağılımlı (gauss dağılımlı) ve varyansı $\sigma^2 = N_0/2$ olan gürültünün, iletilen işarete eklenmesi ile yapılır. Burada beyaz denilmesinin sebebi, gürültü gücü yoğunluğunun tüm frekans bölgesine eşit dağılmasındandır. TBGG sürekli, zamanla değişmeyen ve belleksiz bir kanaldır.

Vericiden gönderilen x vektörü, kanalda rasgele gürültü vektörü n ile bozulur. Bu durumda alıcıda alınan işaret i . anda, r_i olarak şu biçimde elde edilir:

$$r_i = x_i + n_i, \quad i \in [1, n]. \quad (3.1)$$

Rasgele Gauss gürültü değerinin, n_i , *Olasılık Yoğunluk Fonksiyonu, OYF*, şöyle gösterilebilir:

$$p(n_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left[-\frac{1}{2}\left(\frac{n_i}{\sigma}\right)^2\right]} \quad (3.2)$$

(3.1) eşitliğinden elde edilen n_i değeri bu eşitlikte yerine konulursa aşağıdaki ifade elde edilir.

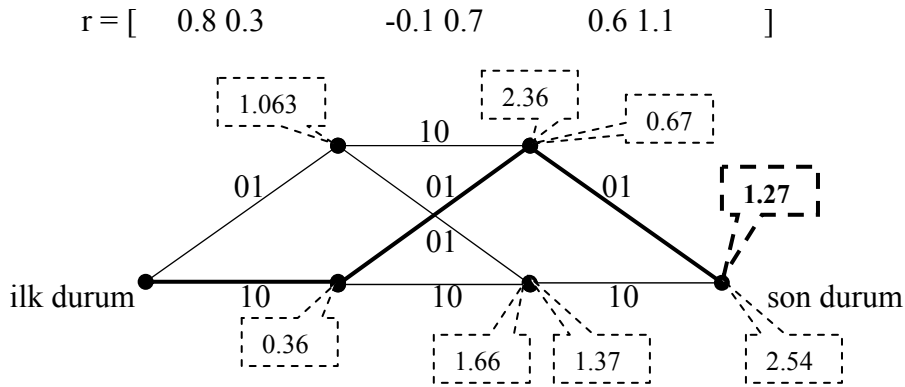
$$p(r_i/x_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left[-\frac{1}{2}\left(\frac{r_i - x_i}{\sigma}\right)^2\right]} \quad (3.3)$$

Bu eşitlikle vericiden bir x_i işareti TBGG kanalda gönderildiğinde, alıcıda r_i değerini alma olasılığı elde edilir.

3.3. VITERBI ALGORİTMASI

1967’de Viterbi [28] tarafından bulunan *Viterbi algoritması (VA)*, temel olarak bir kodun kafes yapısını kullanarak en büyük olabilirlikli kodçözümünü gerçekleştiren bir kodçözme algoritmasıdır. Viterbi algoritması bir t_i anındaki durumlara giren her bir kafes patikaları ile alınan işaret değerleri arasındaki benzerliği yada mesafeyi (metrik) hesaplar. Kullanılan metrik genelde *Öklid metriğidir* ve karşılaştırılacak sayıların ayrı ayrı farklarının karelerinin toplamının karekökü olarak hesaplanır. Bir anda aynı duruma giren kafes patikalarından metriği fazla olan elenir. Metriği az olan, yani en benzeri, en büyük olabilirli olan, tercih edilen patika olarak kalır ve işleme bu tercih edilen patika ve de metriği ile devam edilir. Bu metrik karşılaştırmaları ve elemeler son duruma kadar devam eder ve en sonunda en küçük metriğe sahip patika yani kodkelimesi seçilerek kodçözümü gerçekleştirir. Her durumda yüksek metrikli patikaları elemek işlem sayısında çok önemli miktarda azalma sağlar. Bu konuda çok önemli çalışmalar yapılmış ve yapılmaya devam etmektedir [48 - 51]. Anlatılanları açıklayıcı basit bir örnek aşağıda verilmektedir.

Örnek 3.1: Aşağıda görülen basit bir kafes yapısı üzerinde, alıcıda alınan bir işaret vektörü r ’nin kodu gösterildiği gibi çözülür.



Bu örnekte alınan işaret r vektörü olarak kafes üzerinden tüm olası kodkelimeleri ile karşılaştırılmış ve en büyük olabilirlikli patika, yani kodkelimesi kalın dallarla gösterilen patika olarak karar verilmiştir. Bu durumda karar verilen vektör $v = [\ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \]$ dizisidir.

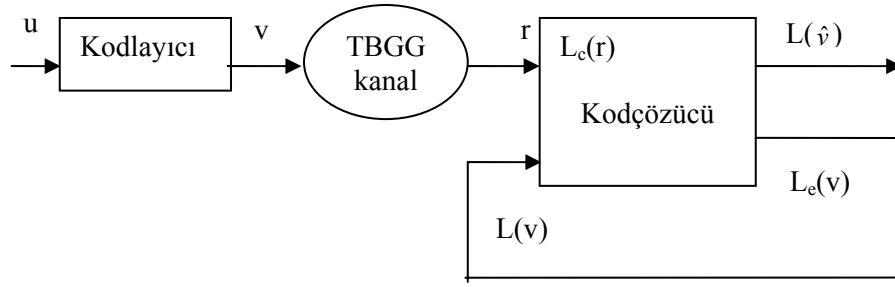
3.4. BCJR ALGORİTMASI

1974'te Bahl, Cocke, Jelinek ve Raviv [29] tarafından geliştirilen bir kodçözme algoritması olduğu için genelde BCJR algoritması olarak anılır, ayrıca MAP (Maximum a posteriori) algoritması olarak ta bilinmektedir. BCJR algoritması kodçözücüye giren yumuşak işaret değerlerini işler ve yumuşak sonuçlar üretir. Bu tip kodçözücülere Yumuşak Girişli – Yumuşak Çıkışlı, YGYÇ (Soft in- Sof out, SISO), kodçözücüler denir. Algoritma sembol hata olasılığını minimuma indirirken aynı zamanda sembol için karar verilen değerın güvenilirlik ölçüt değerini de verir. Algoritma alınan bir sembol dizisindeki her bir sembolü kodun kafes yapısını kullanarak ayrı ayrı kestirirken tüm sembol dizisini göz önüne alarak karar verir. BCJR algoritmasının Viterbi algoritmasına göre karmaşıklığı çok fazladır. Bu tezde yapılan hata başarımı analizlerinde Viterbi algoritması ve Toplam-ürün algoritması genel olarak kullanıldığından ve sonuçlar en çok bu iki algoritmaya göre değerlendirildiğinden BCJR algoritmasının detaylarına burada girilmeyecektir. Bu çalışmada kullanılan BCJR algoritmasının detayları referansı [29] ile gösterilen kaynaktan görülebilir.

3.5. İTERATİF KODÇÖZME ALGORİTMASI

İteratif (döngülü) kodçözme tekniğı modern kanal kodlama yöntemlerinin önemli bir parçasıdır. Bu çalışmada önerilen kodlar bir çeşit blok kodlama yöntemi olduğundan bu kısımda blok kodlar için iteratif kodçözme yöntemi açıklanacaktır. Bu tezde kullanılan iteratif kodçözme tekniğı, Hagenhauer [52] tarafından ortaya konulan yöntem olmakla beraber gösterim (simgelem) olarak [53] kaynağındakine uygun kullanılmıştır. Bu teknik, önerilen kodlara Şekil 3.1 ile resmedilen iletişim sistemi modeline göre uygulandı. Şekil 3.1 üzerinde görünen sembollerden bazılarının başında 'L' olmasının sebebi, ilgili değerlerin logaritmasının alındığını göstermek içindir. Ayrıca burada kullanılan tüm logaritmlar doğal logaritmalardır ve bu sayede işlem kolaylığı sağlanır.

Bu modelde r sembolü $r_j = v_j + n_j$ biçiminde (3.1) eşitliğı ile gösterildiğı gibidir. İteratif kodçözme işlemine, alınan r sembol dizisinin, v kodkelime dizisi olma olasılığını logaritmik gösterimde inceleyerek başlanır. Kodkelimleri kanala gönderilirken İFKA modülasyonu ile mantıksal 0, -1 ve mantıksal 1, +1 değerlerine çevrilir.



Şekil 3.1: İteratif kodçözücülü basit bir iletişim sistemi.

$$\begin{aligned}
 L(v|r) &= \log \left[\frac{P(v=+1|r)}{P(v=-1|r)} \right] = \log \left[\frac{P(r|v=+1)P(v=+1)}{P(r|v=-1)P(v=-1)} \right] \\
 &= \log \left[\frac{P(r|v=+1)}{P(r|v=-1)} \right] + \log \left[\frac{P(v=+1)}{P(v=-1)} \right]
 \end{aligned} \quad (3.4)$$

Bu durumda eşitlikteki ifadeler basit sembollerle ifade edilirse,

$$L(v|r) = L(r|v) + L(v) \quad (3.5)$$

elde edilir. Bu eşitlikte $L_c(r)$ ile gösterilen ifade r 'nin logaritmik kanal ölçütüdür. $L(v)$ ise v 'nin önsel değeridir. Her bir bitin gönderilme olasılığı eşit olduğundan başlangıç değeri sıfırdır. (3.5) eşitliği daha basit biçimde

$$L'(\hat{v}) = L_c(r) + L(v). \quad (3.6)$$

şeklinde gösterilebilir. Bu eşitlikteki $L_c(r)$ değeri gauss gürültülü kanalda aşağıdaki gibi hesaplanır:

$$L_c(r_j) = \log \left[\frac{P(r_j|v_j=+1)}{P(r_j|v_j=-1)} \right] = \log_e \left[\frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{r_j-1}{\sigma} \right)^2}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{r_j+1}{\sigma} \right)^2}} \right] \quad (3.7)$$

$$L_c(r_j) = -\frac{1}{2} \left(\frac{r_j - 1}{\sigma} \right)^2 + \frac{1}{2} \left(\frac{r_j + 1}{\sigma} \right)^2 = \frac{2}{\sigma^2} r_j \quad (3.8)$$

Diğer taraftan *katkı değeri* (extrinsic value) ile isimlendirilen $L_e(v)$ değeri, kodçözümü esnasında her bir iterasyon sırasında kodçözücü çıkışında elde edilir. Yumuşak kararlı kodçözücü çıkışındaki $L(\hat{v})$ değeri gerçek bir değerdir ve bu değer in işaretine bakılarak yani 0 eşik seviyesi ile kıyaslanarak ve bir *sert karar* verilerek kodçözümü gerçekleştirir.

$$L(\hat{v}) = L_c(r) + L(v) + L_e(v) \quad (3.9)$$

Burada $L(\hat{v})$ değerinin ‘+’ veya ‘-’ olması ile 1 veya 0 mantıksal değerlerine karar verilir. $L(\hat{v})$ değerinin büyüklüğü ise verilen kararın güvenilirliği hakkında bize bilgi verir. Denklemden geçen $L_e(v)$ değeri v kodkelimesindeki bitlerin her birinin diğeriyle olan eşitliğinden yola çıkılarak bulunur. İteratif kodçözme yöntemi temelinde çok fazla çalışma vardır ve en çok kullanıldığı konular, Turbo kodlarla ilgili olanlardır [54 - 64].

3.5. TOPLAM-ÜRÜN ALGORİTMASI

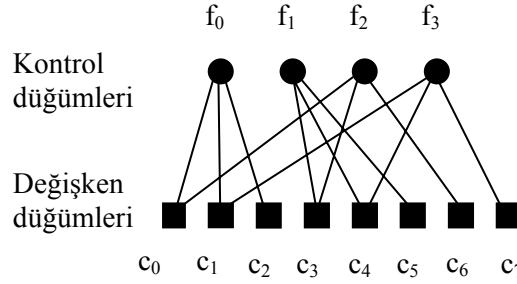
Düşük yoğunluklu parite kontrol kodlarını (Low Density Parity Check Codes, LDPC) 1960’ta geliştiren Gallager [21], aynı zamanda yaklaşık-en iyi bir kodçözme algoritması da önermiştir. Daha sonra bu algoritmaya benzer başka algoritmalar da çeşitli uygulamalar için başka araştırmacılar tarafından geliştirilmiştir [65, 66]. Bu kodçözme algoritması temelde, ilgili değişkenlerin iteratif olarak bir çizge (grafik) üzerinden hesaplanması şeklinde olur. Söz konusu algoritma literatürde, *Toplam-Ürün Algoritması*, (TÜA), *inanç-yayılma* algoritması ve *mesaj geçme* algoritması olarak geçmektedir. Bu algoritmalarda kodçözme işlemi *Tanner çizgeleri* [67] üzerinden yapılır.

Tanner [67] LDPC kodlarını etkin bir biçimde temsil edebilmek ve kodçözümünü gerçekleyebilmek için *iki parçalı* bir çizge (Tanner çizgesi) geliştirdi. İki parçalı bu çizge, iki düğüm grubundan oluşur ve bir gruptaki bir düğüm ancak karşı gruptaki düğümlerle bağlanabilir yani kendi grubuyla direk bağlantı yapılamaz. İki düğüm grubu değişken düğümleri (v-düğüm) ve kontrol düğümleri (k-düğüm) olarak

isimlendirilirler. Tanner çizgesi şu kurala göre çizilir: Parite kontrol matrisi H içindeki bir h_{ji} elemanı mantıksal 1 ise; j . kontrol düğümü i . değişken düğümüne bağlanır. Aşağıdaki Örnek 3.2 de bu durum daha açık görülebilir.

Örnek 3.2: Bir (8,4) ürün kodu parite kontrol matrisi ve Tanner çizgesi şöyledir.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (3.10)$$



Şekil 3.2: Basit bir (8,4) ürün kodu için Tanner çizgesi.

Toplam-ürün algoritması açıklanırken yukarıdaki örnekte bulunan şekildeki parametreler ve değişken gösterimleri kullanılacaktır. Ayrıca algoritma açıklanırken bu kısımda [68] nolu kaynaktaki sembol gösterimleri kullanılmıştır. Buna göre en büyük olabilirlikli kodçözme de, alınan bir y dizisi için bir c_i bitinin 0 veya 1 olarak gönderilme olasılığı oranı L ise,

$$L = \frac{\Pr(c_i = 1 | y)}{\Pr(c_i = 0 | y)} \quad (3.11)$$

olarak yazılabilir. Bu durumda sert karar aşağıdaki gibi verilir:

$$\hat{c}_i = \begin{cases} 1 & \text{if } L \geq 1 \\ 0 & \text{if } L < 1 \end{cases} \quad (3.12)$$

Buna göre Toplam-ürün algoritmasını, detaylarına girmeden önce bazı değişkenler şu şekilde tanımlanabilir:

- q_{ij} : değişken düğümünden, c_i , kontrol düğümüne, f_j , geçilen mesajlar.
- r_{ji} : kontrol düğümünden, f_j , değişken düğümüne, c_i , geçilen mesajlar.
- $R_j = \{i: h_{ji}=1\}$: j. satırdaki 1'lerin kolonlardaki yerlerinin kümesi.
- $R_{j\setminus i} = \{i': h_{ji'}=1\} \setminus \{i\}$: i.kolonun yeri hariç, j. satırdaki 1'lerin kolonlardaki yerlerinin kümesi.
- $C_j = \{i: h_{ji}=1\}$: i. kolondaki 1'lerin satırlardaki yerlerinin kümesi.
- $C_{i\setminus j} = \{i': h_{ji'}=1\} \setminus \{j\}$: j.satırın yeri hariç, i. kolondaki 1'lerin satırlardaki yerlerinin kümesi.
- $p_i = \Pr(c_i = 1 | y_i)$

Tanımlanan bu değişkenlerce algoritma, adım adım şu şekilde yapılır:

1. Başlatımlama,

$$\begin{aligned} q_{ij}(0) &= 1 - p_i = \Pr(c_i = 0 | y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}} \\ q_{ij}(1) &= p_i = \Pr(c_i = 1 | y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \end{aligned} \quad (3.13)$$

2. Birinci yarım döngü,

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{ji}} (1 - 2q_{ij'}(1)) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned} \quad (3.14)$$

3. İkinci yarım döngü,

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1 - p_i) \prod_{j' \in C_{i\setminus j}} r_{ji'}(0) \\ q_{ij}(1) &= K_{ij} p_i \prod_{j' \in C_{i\setminus j}} r_{ji'}(1) \end{aligned} \quad (3.15)$$

Burada k_{ij} sabitleri öyle seçilmelidirki $q_{ij}(0) + q_{ij}(1) = 1$ olsun.

4. Yumuşak karar,

$$\begin{aligned} Q_i(0) &= K_i(1 - p_i) \prod_{j \in C_i} r_{ij}(0) \\ Q_i(1) &= K_i p_i \prod_{j \in C_i} r_{ij}(1) \end{aligned} \quad (3.16)$$

Burada k_i sabitleri; $Q_i(0) + Q_i(1) = 1$ olacak şekilde seçilmelidir.

5. Sert karar,

$$\hat{c}_i = \begin{cases} 1 & \text{if } Q_i(1) > Q_i(0) \\ 0 & \text{aksi halde} \end{cases} \quad (3.17)$$

Bu algoritma maksimum döngü sayısına ulaşınca kadar, veya $\hat{c}^T H^T = 0$ şartı sağlanana kadar döndürülür. Günümüzde LDPC kodlarının koçözümünde ve diğer alanlarda da birçok yerde kullanılan bu algoritma ile ilgili literatürde bir hayli çalışma mevcuttur [69-79] .

4. BULGULAR

Bu tezde yapılan çalışmalarda, temelde kodlayıcı yapısını kolaylaştıran, kolay uygulanabilir, pratik, istenilen kod uzunluğunu sağlayan esneklikte ve daha da önemlisi az karmaşıklıkla kafes yapılarını tasarlamayı sağlayan yeni bir genel blok kod yapısı önerilmiş olup, geometrik bir yapı ile inşa edildiğinden *Geometrik Ürün (GÜ)* kodları olarak isimlendirilmiştir. Buna ek olarak, önerilen kodların kodçözümünü az karmaşıklıkla çözmeyi sağlayan genel yapıdaki minimum kafes yapıları tasarlanmıştır. Bu kafes yapıları kullanılarak, önerilen kodların *toplamsal beyaz Gauss gürültülü, TBGG*, kanallarda ki bilgisayar benzetimleri ile elde edilen hata başarımları, bir çok değişik kod boyutları ve değişik kodçözme yöntemleri için elde edilmiştir. Ayrıca önerilen yeni kodlama yöntemi ile aynı ölçülerde, ona benzer fakat daha basit kodlama yapısındaki bir başka yeni kodlama tekniği de geliştirilmiş olup, inşa yapısı tamamen basamaklı bir yapı gösterdiğinden bu kodlar da *Basamak Blok (BB)* kodları olarak isimlendirilmiştir.

4.1. GEOMETRİK ÜRÜN KODLARI İNŞASI

Geometrik Ürün (GÜ) kodları yöntemiyle inşa edilen bir blok kodu; $C = (n, k, d)$ olarak tanımlansın. Burada n kod uzunluğu, k kod boyutu yani saf bilgi bitleri sayısı, ve d kodun minimum Hamming mesafesidir. Bir C blok kodunu üreten üreteç matrisi G , üç veya daha fazla basit yapıdaki eleman kodları üreteç matrislerinin, GÜ kod inşası olarak tanımlanan bir yöntemle birleşimi sonucu oluşur. Bu birleşme işlemi sonucu elde edilen ürün, üreteç matrisi G olduğundan, bu üreteç matrisi kullanılarak üretilen kodlar GÜ kodları olarak adlandırılmışlardır. GÜ kodu üreteç matrisi inşası, genel bir *ürün kodu* üreteç matrisi üzerine, belirlenen başka basit yapıdaki eleman matrislerinin yine belirlenen kurallara göre eklenmesi ile gerçekleştirilir.

Genel bir ürün kodu $C_y = (n_y, k_y, d_y)$ ile ve üreteç matrisi G_y ile gösterilsin. C_y ürün kodu $C_1 = (n_1, k_1, d_1)$ ve $C_2 = (n_2, k_2, d_2)$ olan iki basit kodun birleşimi ile oluşur. Birleşim işlemi C_1 ve C_2 kodlarının G_1 ve G_2 üreteç matrislerinin *Kronoker çarpımı* (Kronocker product) sonucu oluşturulur ($G_y = G_2 \otimes G_1$). GÜ kod inşası bu şekilde elde edilen G_y matrisini temel alır ve ona başka bir basit eleman matrisi olan G_z matrisini belirli bir kurala göre ekler. Bazı durumlarda G_z haricinde başka eleman matrisleri de eklemek gerekebilir. Genel olarak tanıtilan GÜ kod inşasının detayları aşağıda açıklandığı gibidir.

Genel bir ürün kodu üreteç matrisi, G_y , basit matrislerin, G_1 ve G_2 , Kronoker çarpımları ile $G_y = G_2 \otimes G_1$ şeklinde aşağıdaki gibi elde edilir. Burada G_2 *tekil parite kontrol matrisidir*.

$$G_2 = \begin{bmatrix} 1 & & & 1 \\ & 1 & & 1 \\ & & \ddots & \vdots \\ & & & 1 & 1 \end{bmatrix}_{k_2 \times n_2} \quad (4.1)$$

ve

$$G_y = \begin{bmatrix} G_1 & & & G_1 \\ & G_1 & & G_1 \\ & & \ddots & \vdots \\ & & & G_1 & G_1 \end{bmatrix}_{k_y \times n_y} \quad (4.2)$$

Eleman matrislerinin ölçüleri belirlendiğinden ürün matrisinin, G_y , ölçülerini bulmak kolay olacaktır. Burada $k_2 = n_2 - 1$, $k_y = k_1 \times k_2$ ve $n_y = n_2 \times n_1$, olmak üzere, $n_2 \geq 4$ seçilmelidir. GÜ kod inşası bu denklemler sonucu elde edilen G_y matrisini temel olarak kullanır ve bu matrisin geometrik olarak altına başka bir basit yapıda olan G_z matrisini veya eğer gerekliyse başka basit eleman matrislerini de belirli bir kurala göre yerleştirir. Bu şekilde inşa edilen GÜ kodlarının sonuç üreteç matrisi, G , aşağıdaki denklemde gösterilen genel yapıda olur.

A_1 grubu için yerleştirme işlemi sona erer. A_1 grubu için yerleştirme işlemi tamamlandıktan sonra A_2 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda A_2 grubu için yerleştirilecek 4 adet G_z matrisi arasında birer aralık bırakılır. Buradaki ‘aralıktan’ kasıt bir G_z ölçeği kadarlık yerdir. A_2 grubunun ilk satırında da yerleştirme ilk kolondan başlar. A_2 grubunun ikinci satırında ise ardarda yerleştirilen 4 adet birer aralıklı G_z matrisi, 4 G_z matrisi ölçeğinde sağa kaydırılır. Böylece aynı grupta bulunan ardışık iki satır arasında ki 2’şer adet G_z matrisleri üst üste gelmiş olur. Bu işlemde bir öncekine benzer biçimde 4 adet G_z matrisini yerleştirecek kolon kalmayana kadar devam eder. A_2 grubu için yerleştirme işlemi tamamlandıktan sonra A_3 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda ise A_3 grubu için yerleştirilecek 4 adet G_z matrisi arasında üçer aralık bırakılır. Yerleştirme işlemi öncekiler gibi yapılmakla beraber sağa kaydırma 8 aralık yapılır. Bu işlemler aynı şekilde, eğer yerleştirmeye yer varsa, diğer gruplar içinde devam eder. Tüm yerleştirme işlemi 4 adet G_z matrisini yerleştirecek ekleme satırı kalmayınca sona erer. Genel bir kural olarak, sağa kaydırma işlemi A_1 grubu için 2, A_2 grubu için 4, A_3 grubu için 8 daha sonraki gruplar için 8, 16,... ve bu şekilde 2’nin kuvvetleri halinde devam eder. 4 adet G_z matrisi arasında bırakılan aralıklar ise A_1 grubu için 0, A_2 grubu için 1, daha sonraki gruplar için 3, 7, 15,... ve bu şekilde kaydırma miktarının yarısından 1 eksik olarak devam eder.

Şimdiye kadar açıklanan kurallar uygun eleman matrisleri seçildiği varsayımı altında, eleman matrislerinin nasıl yerleştirildiği ile ilgilidir. Eleman matrislerinin nasıl seçileceği ile ilgili genel kurallar konulmamakla beraber bunların nasıl seçileceğini en başta istenilen kodun uzunluğu belirler. İstenilen uzunluklu koda göre C_1 ve C_2 kodlarının üreteç matrisleri birleştirilerek genel ürün matrisi yapısı elde edilir. G_z matrisi seçilirken dikkat edilmesi gereken husus ise 4 adet G_z matrisinde bulunan mantıksal 1’lerin toplamı, kodun arzu edilen minimum Hamming mesafesine eşit olmasıdır. Bu konuda daha sonraki kısımlarda açıklayıcı örnekler verilecektir. Bunlara ilaveten, elde edilen GÜ kod yapısı sistematik olmamakla beraber, bilgi bitleri ve parite kontrol bitleri ayırt edilebildiği için yarı-sistematik bir yapıdadır.

4.2. HAMMING MESAFESİ 4 OLAN GEOMETRİK ÜRÜN KOD AİLESİ

GÜ kodlamasının (4.3) denklemiyle önerilen inşası kullanılarak, minimum Hamming mesafesi 4 olan en iyi özellikli çift kodlar elde edilmiştir. Burada ‘en iyi’ kavramından kasıt, minimum Hamming mesafesi 4 olan çift kodların alabileceği en büyük uzunluk, n , ve boyut, k , parametrelerinin bu kodlama yöntemi ile elde edilebilmesidir. Blok kodların belli bir minimum Hamming mesafesi için alabileceği en iyi yani maksimum uzunluk ve boyut değerleri ‘en iyi-bilinen kodlar tablosu’ olarak literatürde [80] yayınlanmış olup, bu tezde elde edilen kodların en iyi kodlardan olup olmadığına sözkonusu tablo ile kıyaslanarak karar verilmiştir.

Bu minimum Hamming mesafesi 4 olan en iyi çift kodların inşası için, eleman matrisleri $C_1 = (2, 1, 2)$ ve $C_2 = (2, 1, 1)$ boyutlarındaki basit kodlar seçilmiş olup bu kodların üreteç matrisleri; $G_1 = [1 \ 1]$ ve $G_2 = [1 \ 0]$ olan en basit yapıdaki matrislerdir. Temel eleman kodlarından olan C_2 kodunun üreteç matrisi önceki bölümde de açıklandığı gibi tekil-parite kontrol matrisi olup, bu kodun uzunluğunu temsil eden n_2 değeri 4’ten büyük veya 4’e eşit istenilen çift tamsayı değerini alabilir. Dolayısıyla bu n_2 değeri değiştirilerek istenilen kod uzunluğu elde edilebilir. Bu durum daha açık biçimde (4.1) denkleminde ve (4.2) denkleminde görülebilir. Çünkü $G_1 = [1 \ 1]$ gibi basit bir matris olarak seçilip değiştirilmediğinden, G_y matrisini oluşturan diğer matris G_2 değiştirilerek G_y temel matrisinin uzunluğu ayarlanabilir. Buradan elde edilen uzunlukta GÜ kodlarının uzunluk değerine (n) eşittir; çünkü $n = n_2 \times n_1$ olduğu matris özelliklerinden açıkça görülebilir. Sonuç matrisin, sadece, boyutu (k) ekleme matrisleri ile artırılır. Yani n_2 değeri istenilen kodun uzunluğunu elde edebilmek için ilk başta ayarlanması gereken temel parametredir. Bundan başka herhangi bir parametreyi keyfi değiştirebilme söz konusu değildir. Bir önceki bölümde açıklanan GÜ kod inşası kurallarına uyularak ve bu bölümde belirlenen eleman matrisleri kullanılarak minimum Hamming mesafesi 4 olan en iyi yapıda bir GÜ kodu üreteç matrisi, (4.3) denkleminde olduğu gibi otomatik olarak inşa edilebilir.

Bu şekilde minimum Hamming mesafesi 4, uzunluğu 8’den büyük ve çift olan, en iyi yapıda GÜ kodları bir kod ailesi olarak elde edilmiştir. Bulunan kod ailesinin ölçülerini formülleştirilebilmek için 500 adetten fazla üreteç matrisi bilgisayar programı vasıtasıyla

oluşturulmuş olup, bu matrislerin ölçüleri kaydedilmiş ve kod parametreleri n , k ve d arasındaki ilişki aşağıdaki gibi formülleştirilmiştir:

$$C = (n, k, d) = (n, n - \lceil \log_2 n + 1 \rceil, 4) . \quad (4.4)$$

Burada n değeri 8'den büyük veya 8'e eşit çift bir tamsayı olup $\lceil b \rceil$ gösterilimindeki işlem b değerini pozitif sonsuza doğru en yakın tamsayıya yuvarlar. Bu denklemde görülen minimum Hamming mesafesinin 4 değerine eşit olduğu Teorem 4.1 ile aşağıda ispatlanmıştır.

Teorem 4.1: Eğer eleman matrisleri $G_1 = [1 \ 1]$ ve $G_2 = [1 \ 0]$ seçilirse, o zaman, (4.3) denklemiyle inşa edilen bir GÜ üreteç matrisi, G , ile üretilen bir C kodunun minimum Hamming mesafesi 4 değerine eşittir.

İspat: G matrisi geometrik yerleştirilmelerine göre gruplandırılan satır gruplarından oluştuğundan, bu grupların kendisi ve grupların birbirleriyle olan ilişkilerinin incelenmesiyle Teorem 4.1 ispatlanabilir. (4.3) denkleminde matris inşasının yerleştirme kuralından dolayı hiç bir durumda G matrisinin herhangi iki matris satırı arasında 2 adetten fazla eleman matrisleri üst üste gelemez. A_0 grubunda en fazla 1 adet G_1 matrisi üst üste gelir ve dolayısıyla her bir satırdan 1'er adet G_1 matrisi, herhangi bir ikili toplama işleminde boşa kalır. Her bir G_1 matrisi 2 adet mantıksal 1 içerdiğinden A_0 grubunun doğrusal kombinasyonları sonucu oluşabilecek herhangi bir kodkelimesinin minimum Hamming mesafesi 4 değerine eşittir. Kalan satırlar A_i ($i = 1, 2, \dots$) gruplarına ait olup her bir satır 4 adet G_2 matrisi içerir ve bu gruplardaki herhangi iki satır arasında da en fazla 2 adet G_2 matrisi üst üste gelir. Bunun anlamı herhangi iki satır arasındaki ikili toplamada 4 adet G_2 matrisi bozulmadan sonuç satırda kalır. Her bir G_2 matrisi 1 adet mantıksal 1 içerdiğinden dolayı en kötü durumda, A_i gruplarındaki satırların doğrusal kombinasyonları sonucu oluşabilecek herhangi bir kodkelimesinin minimum Hamming mesafesi 4 değerine eşittir. İncelenmeyen tek durum A_0 ve A_i grupları arasındaki ilişkidir. Bu durumda da en kötü durumda her iki grup arasında üst üste gelebilecek matris sayısı en fazla 2'dir. Bu en kötü durumda oluşabilecek sonuçtaki kodkelimesinde 2 adet G_2 matrisi (yani toplam 2 adet mantıksal 1) ve 2 adet $G_1 \oplus G_2$ matrisi ($[1 \ 1] \oplus [1 \ 0] = [0 \ 1]$) yani yine toplam 2 adet mantıksal 1 ve toplam olarak bu

satırda 4 adet mantıksal 1 olur. Bu ise satırın minimum Hamming mesafesinin 4 değerine eşit olduğunu gösterir. Dolayısıyla (4.3) denklemiyle inşa edilen bir üreteç matrisi, G , ile üretilen bir C kodunun minimum Hamming mesafesi 4 değerine eşittir.

Aşağıda gösterilen Tablo 4.1 de (4.4) eşitliği ile ölçüleri belirlenen GÜ kod ailesinden, bilgisayar programı yardımıyla oluşturulan ve tüm parametrelerinin doğruluğu yine aynı program yardımıyla taranıp sağlanan bazı kodlar sunulacaktır.

Tablo 4.1 : Bilgisayar ile oluşturulan ve sağlaması yapılan bazı GÜ kodları

(8,4,4) (10,5,4) (12,7,4) (14,9,4) (16,11,4) (18,12,4) (20,14,4) (22,16,4)
(24,18,4) (26,20,4) (28,22,4) (30,24,4) (32,26,4) (34,27,4) (36,29,4) ...

Şimdiye kadar anlatılan kod inşasının daha açık olarak gösterilmesi için Örnek 4.1 aşağıda verilmektedir.

Örnek 4.1: Kod uzunluğu $n = 52$ ve minimum Hamming mesafesi 4 olan bir GÜ kodu üreteç matrisi, G , inşa edilmek istenirse, $n_2 = 52/n_1 = 52/2 = 26$ olur. Öte yandan $G_1 = [1 \ 1]$ ve $G_z = [1 \ 0]$ olduğu bilinmekte olup G_y matrisi (4.2) denkleminde olduğu gibi oluşturulduğunda (25×52) ölçülerinde bir temel matris elde edilir. Bundan sonra temel matrisin altına G_z eleman matrislerini ekleme işlemi (4.3) denkleminde gösterildiği gibi yapılır. Aşağıda elde edilen G üreteç matrisinde G_z , basitlik açısından sadece z harfi ile temsil edilmektedir. (4.3) denklemi ile elde edilen GÜ kod üreteç matrisi (45×52) ölçülerinde olup 4 ekleme grubundan oluşmaktadır. İlk grup olan A_1 grubunda 12 satır eklenebilmekte ve sağa kaydırma 2 aralıklı olup eleman matrisleri arasında aralık yoktur. İkinci grup olan A_2 grubunda 5 satır eklenebilmekte ve sağa kaydırma 4 aralıklı olup eleman matrisleri arasında 1 aralık vardır. A_3 grubunda 2 satır eklenebilmekte ve sağa kaydırma 8 aralıklı olup eleman matrisleri arasında 3 aralık vardır. Son olarak A_4 grubunda 1 satır eklenebilmekte ve eleman matrisleri arasında 7 aralık olup sağa kaydırma, yer kalmadığından yapılamamıştır. Bu aşamadan sonra G_z matrislerini kurala uygun biçimde yerleştirecek yer kalmadığından işlem tamamlanmıştır. GÜ kod inşası tekniği ile genel bir ürün kodu matrisi üzerine toplamda 20 satır eklenebilmiştir ki bu da

Örnek 4.2: $C = (12, 7, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 110000000011 \\ 001100000011 \\ 000011000011 \\ 000000110011 \\ 000000001111 \\ 101010100000 \\ 000010101010 \end{bmatrix} \quad (4.6)$$

Örnek 4.3: $C = (14, 9, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 11000000000011 \\ 00110000000011 \\ 00001100000011 \\ 00000011000011 \\ 00000000110011 \\ 00000000001111 \\ 10101010000000 \\ 00001010101000 \\ 10001000100010 \end{bmatrix} \quad (4.7)$$

Örnek 4.4: $C = (16, 11, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 1100000000000011 \\ 0011000000000011 \\ 0000110000000011 \\ 0000001100000011 \\ 0000000011000011 \\ 0000000000110011 \\ 0000000000001111 \\ 1010101000000000 \\ 0000101010100000 \\ 0000000010101010 \\ 1000100010001000 \end{bmatrix} \quad (4.8)$$

Örnek 4.5: $C = (18, 12, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 1100000000000000011 \\ 0011000000000000011 \\ 0000110000000000011 \\ 0000001100000000011 \\ 0000000011000000011 \\ 000000000011000011 \\ 000000000000110011 \\ 000000000000001111 \\ 1010101000000000000 \\ 0000101010100000000 \\ 000000001010101000 \\ 100010001000100000 \end{bmatrix} \quad (4.9)$$

Örnek 4.6: $C = (20, 14, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 11000000000000000011 \\ 00110000000000000011 \\ 00001100000000000011 \\ 00000011000000000011 \\ 00000000110000000011 \\ 00000000001100000011 \\ 00000000000011000011 \\ 00000000000000110011 \\ 00000000000000001111 \\ 10101010000000000000 \\ 00001010101000000000 \\ 00000000101010100000 \\ 00000000000010101010 \\ 10001000100010000000 \end{bmatrix} \quad (4.10)$$

Örnek 4.7: $C = (22, 16, 4)$ olan bir GÜ kodu üretic matrisi:

$$G = \begin{bmatrix} 1100000000000000000011 \\ 0011000000000000000011 \\ 0000110000000000000011 \\ 0000001100000000000011 \\ 0000000011000000000011 \\ 0000000000110000000011 \\ 0000000000001100000011 \\ 0000000000000011000011 \\ 0000000000000000110011 \\ 0000000000000000001111 \\ 1010101000000000000000 \\ 0000101010100000000000 \\ 0000000010101010000000 \\ 0000000000001010101000 \\ 1000100010001000000000 \\ 0000000010001000100010 \end{bmatrix} \quad (4.11)$$

Örnek 4.8: $C = (24, 18, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 11000000000000000000000011 \\ 00110000000000000000000011 \\ 00001100000000000000000011 \\ 00000011000000000000000011 \\ 00000000110000000000000011 \\ 00000000001100000000000011 \\ 00000000000011000000000011 \\ 000000000000001100000011 \\ 000000000000000011000011 \\ 0000000000000000001100011 \\ 00000000000000000000110011 \\ 00000000000000000000001111 \\ 10101010000000000000000000 \\ 00001010101000000000000000 \\ 00000000101010100000000000 \\ 000000000000101010100000 \\ 000000000000000010101010 \\ 10001000100010000000000000 \\ 000000001000100010001000 \end{bmatrix} \quad (4.12)$$

Örnek 4.9: $C = (26, 20, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 1100000000000000000000000011 \\ 0011000000000000000000000011 \\ 0000110000000000000000000011 \\ 0000001100000000000000000011 \\ 0000000011000000000000000011 \\ 0000000000110000000000000011 \\ 0000000000001100000000000011 \\ 0000000000000011000000000011 \\ 0000000000000000110000000011 \\ 00000000000000000011000011 \\ 00000000000000000000110011 \\ 00000000000000000000001111 \\ 1010101000000000000000000000 \\ 0000101010100000000000000000 \\ 0000000010101010000000000000 \\ 0000000000001010101000000000 \\ 00000000000000001010101000 \\ 1000100010001000000000000000 \\ 00000000100010001000100000 \\ 10000000100000001000000010 \end{bmatrix} \quad (4.13)$$

Örnek 4.11: $C = (30, 24, 4)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 110000000000000000000000000011 \\ 001100000000000000000000000011 \\ 000011000000000000000000000011 \\ 000000110000000000000000000011 \\ 000000001100000000000000000011 \\ 000000000011000000000000000011 \\ 000000000001100000000000000011 \\ 000000000000110000000000000011 \\ 000000000000011000000000000011 \\ 000000000000001100000000000011 \\ 000000000000000110000000011 \\ 000000000000000011000011 \\ 00000000000000000110011 \\ 0000000000000000001111 \\ 1010101000000000000000000000 \\ 0000101010100000000000000000 \\ 0000000010101010000000000000 \\ 0000000000001010101000000000 \\ 0000000000000001010101000000 \\ 00000000000000001010101000 \\ 1000100010001000000000000000 \\ 0000000010001000100010000000 \\ 000000000000000010001000100010 \\ 100000001000000010000000100000 \end{bmatrix} \quad (4.15)$$

4.3. DAHA BÜYÜK HAMMING MESAFELİ GEOMETRİK ÜRÜN KODLARI İNŞASI

GÜ kod inşası, uygun eleman üreteç matrislerinin seçildiği varsayımı altında, bu matrislerden oluşturulacak bir büyük üreteç matrisinin elde edilmesi için eleman matrislerinin nasıl yerleştirileceğine dair kesin, düzgün ve genel geometrik kurallar sağlar. Buna karşılık, kullanılacak eleman matrislerinin seçilmesi için kesin ve genel kurallar koymaz. Sonuç üreteç matrisinin istenilen ölçülerine göre gerekli eleman matrislerinin boyutları büyük ölçüde bellidir fakat kesin ölçütler olmadığından eleman matrisleri denenerek bulunmalıdır. Bu şekilde elde edilen eleman matrisleri kullanılarak elde edilen minimum Hamming mesafesi 4 olan en iyi kodlar bir önceki bölümde sunulmuştu. Bu bölümde ise minimum Hamming mesafesi 8 olan kodların inşası için gerekli eleman matrisleri ve ayrıca daha da büyük Hamming mesafeli kodların elde edilmesi için gerekli eleman üreteç matrisleri gösterilecek ve bu eleman matrisleri seçilirken nelere dikkat edilmesi gerektiğin hakkında bahsedilecektir. GÜ kod inşası ile elde edilen 4'ten büyük Hamming mesafeli kodlardan bazıları en iyi kodlardan olup bazıları değildir. Yani bu durumda 4 Hamming mesafeli kodlarda olduğu gibi tüm kodlar en iyi kodlardan olmayabilir. Bu bölümde öncelikle minimum Hamming mesafesi 8 olan kodların inşası için gerekli eleman matrisleri sunulacaktır. Eleman matrisleri

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \quad (4.19)$$

ve

$$G_z = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad (4.20)$$

olarak seçilmelidir. Bu eleman matrisleri kullanılarak ve (4.3) denklemiyle gösterilen GÜ kod inşası yönteminden yararlanılarak (16, 5, 8), (24, 9, 8), (32, 15, 8) kodlarının üreteç matrisleri inşa edilmiş ve kodların parametrelerinin doğruluğunun sağlanması bilgisayar programı yardımıyla test edilmiştir. Diğer taraftan inşa edilen kodların uzunlukları artırıldıkça, inşa edilen kodların ölçüleri en iyi olma özelliğinden gitgide sapmaktadır. Bu durumu bertaraf etmek için kod inşa yöntemi geliştirilmeye çalışılmış ve genel kod inşasına ufak bir ekleme ile (32,16,8) en iyi kodu elde edilebilmiştir. Ancak, her kod uzunluğu için bu durumu genelleştirebilmek mümkün olmamıştır. Buna

mukabil bu çalışmada ulaştığımız sonuç göstermektedir ki, GÜ kod inşası, kendi üzerinde yapılabilecek ufak eklemeler ile daha büyük minimum Hamming mesafeli kodlarda da en iyi kodları üretebilme potansiyeline sahiptir. Üretilen (32, 16, 8) geliştirilmiş GÜ kodu inşası Örnek 4.15 ile aşağıda gösterilmiştir.

Örnek 4.15: $C = (32, 16, 8)$ en iyi ölçülerindeki bir GÜ blok kodunu üretebilmek için önce n_2 değeri $n_2 = 32/n_1 = 32/4 = 8$ olarak hesaplanır. Burada n_1 (4.19) eşitliğinden görüldüğü gibi 4'tür. Daha sonra G_2 ve G_y matrisleri (4.1) ve (4.2) deki gibi oluşturulur. G_1 ve G_z (4.19) ve (4.20) eşitliklerinde belirtildiği gibi seçilir. En son olarak bu koda özel geliştirilmiş GÜ kod inşasında kullanılmak üzere bir başka eleman matrisi G_3 şu şekilde seçilir:

$$G_3 = [1 \ 0 \ 0 \ 0] \quad (4.21)$$

Bu işlemlerden sonra eleman matrisleri aşağıda (4.22) eşitliğinde görüldüğü gibi yerleştirilerek sonuç üreteç matrisi inşa edilir.

$$G = \begin{bmatrix} & & & G_y \{7 \times 32\} \\ G_z & G_z & G_z & G_z \\ & G_z & G_z & G_z \\ & & G_z & G_z \\ G_z & & G_z & G_z \\ G_z & G_z & G_z & G_z \\ G_z & G_z & G_z & G_z \\ G_z & G_z & G_z & G_z \end{bmatrix} \quad (4.22)$$

$\{16 \times 32\}$

Daha büyük minimum Hamming mesafeli kodlar için eleman matrislerini seçme işlemi daha da güçleşmektedir. Aşağıda minimum Hamming mesafesi 16 olan kodları üretmek için seçilmesi gereken eleman matrisleri gösterilmiştir.

$$G_3 = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1] \quad (4.23)$$

ve

$$G_z = \begin{bmatrix} 11110000 \\ 00111100 \\ 10101010 \end{bmatrix} \quad (4.24)$$

Bu eleman matrislerini kullanarak ve (4.3) ile gösterilen GÜ kod inşasından yararlanılarak (64, 19, 16) GÜ kodu üretilmiş ve parametrelerin doğruluğunun sağlaması bilgisayar programı ile test edilmiştir. Aşağıda verilecek örnekler ile büyük minimum Hamming mesafeli kodların üreteç matrislerinden bazıları sunulacaktır.

Örnek 4.16: $C = (16, 5, 8)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 1111000000001111 \\ 0000111100001111 \\ 0000000011111111 \\ 1100110011001100 \\ 0110011001100110 \end{bmatrix} \quad (4.25)$$

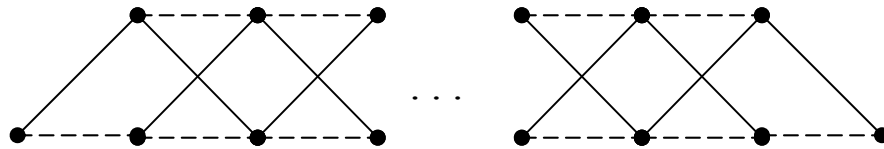
Örnek 4.17: $C = (24, 9, 8)$ olan bir GÜ kodu üreteç matrisi:

$$G = \begin{bmatrix} 111100000000000000001111 \\ 000011110000000000001111 \\ 000000001111000000001111 \\ 000000000000111100001111 \\ 000000000000000011111111 \\ 110011001100110000000000 \\ 011001100110011000000000 \\ 000000001100110011001100 \\ 000000000110011001100110 \end{bmatrix} \quad (4.26)$$

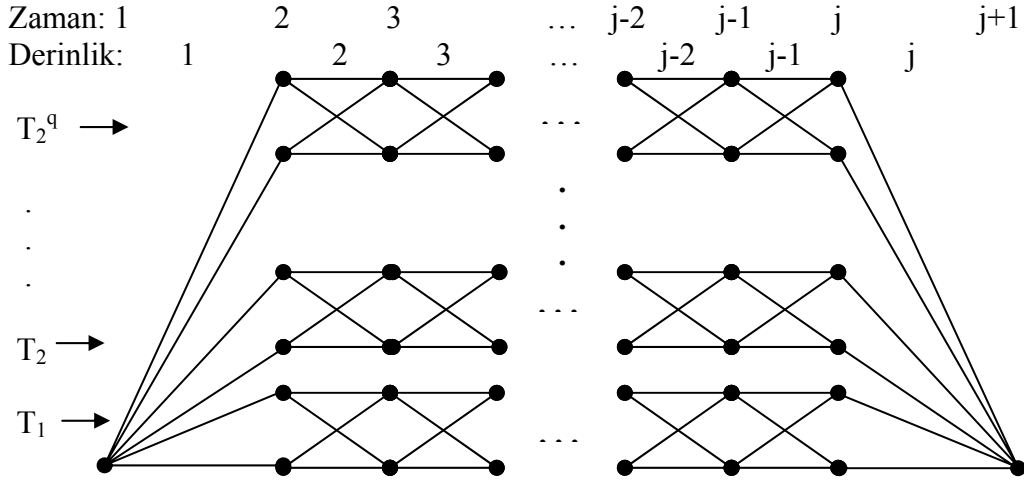
4.4. GEOMETRİK ÜRÜN KODLARI İÇİN KAFES YAPILARI

Blok kodların Viterbi algoritması (VA) ile kodçözümünün pratikte ki uygulamaları için basit yapıda, düzenli, esnek, özdeş, paralel ve az sayıda dal ile durum içeren kafes yapıları elzemdir. Bununla beraber genelde, blok kodun ölçüleri arttıkça kafes karmaşıklığı da üstel olarak artar. *Az karmaşıklıkli* kafes yapıları en az sayıda dal ve durum içerirler; ama bu pratikte kolay uygulanabilir yapıda oldukları anlamına gelmez. Dolayısıyla bir kafesin pratik ve teorik karmaşıklığı arasında bir denge noktası vardır denilebilir. İyi bir kafes yapısı, birçok paralel ve yapısal olarak özdeş alt-kafesler içermesinin yanında az boyutta durum uzayına sahip olmalı ve alt-kafesler arasında çapraz bağlantılar barındırmamalıdır. Bu birbirine özdeş ve az karmaşıklıkli Viterbi kodçözümler kullanarak paralel yani eşzamanlı olarak alt-kafesler üzerinde bağımsız biçimde işlem yapmaya ve sonuçta kodçözme işleminin hızını artırmaya olanak sağlar [81].

Önerdiğimiz GÜ kodları için tasarlanan genel yapıdaki kafes yapıları, büyük ölçüde, yukarıda bahsedilen gerekli özellikleri taşımaktadır. Şekil 4.1 ve Şekil 4.2 ile tasarlanan bu kafes yapılarının genel yapısı verilmiştir. Şekillerden de görüleceği gibi bu kafes yapısı birbirinden bağımsız, paralel, yapısal özdeş, 2-durumlu alt-kafes içeren ve alt-kafesler arasında çapraz bağlantı içermeyen yapıdadır. Dolayısıyla Şekil 4.1 ile gösterilen alt-kafesi inşa etmek aslında hemen hemen bütün kafesi inşa etmek anlamına gelmektedir.



Şekil 4.1: GÜ kodları için genel alt-kafes yapısı (T).



Şekil 4.2: GÜ kodları için genel kafes yapısı (T').

Burada ‘zaman’ ile her bir durumun indisi ve ‘derinlik’ ile de her bir dalın indisi gösterilmiştir. T_2^i ($i = 0, 1, \dots, q$) ise, her bir alt kafesin indisidir.

Alt-kafes olarak Şekil 4.1 ile gösterilen T az karmaşıklıkla yapıda olup, birden çok T alt-kafesinin paralel olarak başlangıç ve son durumların birleştirilmesiyle GÜ kodları için genel kafes yapısının bütünü Şekil 4.2 ile (T') gösterildiği gibi inşa edilir.

Bir GÜ kodunun bir T' az karmaşıklıkla kafesini inşa etmek için önce alt-kafes, T , oluşturulur. Derinliklerin sayısı, j , kod uzunluğunun, n , C_1 eleman kodu uzunluğu, n_1 , ile bölümünden elde edilir ($j = n / n_1$). Bu aşamadan sonra 2^q özdeş alt-kafes birbirine başlangıç ve son noktalarından Şekil 4.2 de gösterildiği gibi bağlanarak elde edilir. 2^q ifadesindeki q değişkeni,

$$q = k - k_y \quad (4.31)$$

denkleminde hesaplanır. Burada k kodun boyutu ve k_y ise temel ürün kodunun, C_y , boyutudur. Pratikteki uygulama yönünden bakıldığında alt-kafes, T , GÜ kodunun kafes gösterimi gibi kabul edilebilir. Çünkü, Viterbi algoritması kullanıldığında eğer hız önemli değilse, T alt-kafesinin sonunda elde edilen metrik, bir saklayıcıda tutularak, kalan alt-kafeslerin metrikleri de aynı Viterbi kodçözücü ile elde edilip bir saklayıcıda tutulabilir. Sonunda saklayıcıdaki metrikler kıyaslanarak doğru patika yani kodkelimesi seçilebilir. Kodlanmış bitlerin kafes dallarına atanması (etiketlenmesi) işlemi dinamik

şekilde gerçekleştirilebilir. Bu dinamik etiketleme olanağından dolayı T alt-kafesinin, bir yönden, GÜ kodlarının kafes yapılarını temsil ettiği de söylenebilir.

Alt-kafesin, T, en önemli özelliklerinden bir tanesi, etiketlenme işleminin kolaylığıdır. Çünkü her bir dal n_1 bit ile etiketlenir ve bu, kafesin hiçbir derinliğinde değişmez. Kafes yapısı, T', üzerinde görüldüğü gibi, bir alt-kafesin, T, sonuç metriğini hesaplamak ve o metrikle ilişik patikayı hesaplamak, tüm kafes yapısının, T', metriğini hesaplamak için en önemli aşamadır. Çünkü diğer alt-kafeslerin metriğini hesaplariken tek değişen dal etiketleridir. Alt-kafesi, T, daha açıkça analiz edebilmek için dalları 2 ayrı grupta tanımlanır. Bunlar *düz dallar* (Şekil 4.1 de kesikli çizgilerle gösterilen tüm düz dallar) ve *çapraz dallardır* (Şekil 4.1 de sabit çizgilerle gösterilen tüm çapraz dallar). T üzerindeki dallara etiketlenen bitlerin sayısı tüm kafes boyunca aynıdır ve her bir dal n_1 bit ile etiketlenir. Ayrıca bir diğer önemli özellik de, T alt-kafesinin herhangi bir derinliğinde birbirini kesen 2 adet çapraz dalın etiketlerinin birbirinin aynı olması ve birbirine paralel olan 2 adet düz dalın da etiketlerinin birbirinin aynı olmasıdır. Daha da önemlisi, herhangi bir derinlikteki düz dalların etiketleri, çapraz dalların etiketlerinin ikili sayı alfabesine göre tümleyenidir.

Kodlanmış bitleri T alt-kafesinin dallarına etiketlemek için *üreteç kodkelimleri* kullanılır. Bu üreteç kodkelimleri, GÜ kodunun üreteç matrisinin, G, ekleme satırlarının doğrusal kombinasyonlarından ve temel matris G_y' nin tüm satırlarının ikili toplamından oluşur. Bir alt-kafesin, T, üreteç kodkelimesi kullanılarak etiketlenmesi işleminde aşağıdaki yol izlenir:

- a) İşlemler soldan sağa doğru gitmekte olup, alt-kafesin, T, ilk derinliğinde bulunan ilk düz dala, etiketlenecek üreteç kodkelimesinin ilk n_1 adet biti etiketlenir. Daha sonra, aynı derinlikte bulunan çapraz dala, düz dala etiketlenen bitlerin tümleyeni etiketlenir.
- b) Daha sonra ikinci derinlikte bulunan düz dallara, üreteç kodkelimesinin ikinci n_1 adet biti etiketlenir. Daha önce olduğu gibi burada da aynı derinlikte bulunan çapraz dallara, düz dallara etiketlenen bitlerin tümleyeni etiketlenir.

- c) Bir önceki adımda yapılanlar, takip eden her bir derinlikteki dallar için de en son derinliğe kadar tekrarlanır. En son derinlikte, üreteç kodkelimesinin son n_1 adet biti ilk adımda olduğu gibi dallara etiketlenir.

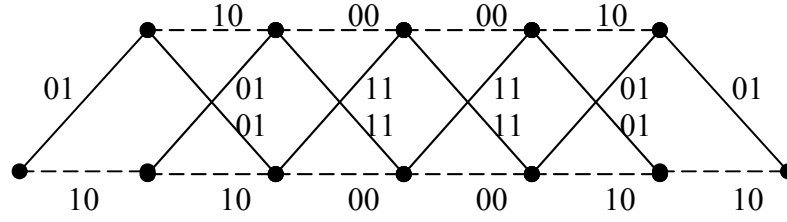
Yukarıda anlatılan esnek etiketleme işlemi, geriye kalan diğer tüm üreteç kodkelimeleri kullanılarak kalan alt-kafesler için de aynı işlem adımlarından geçirilerek tekrarlanır. Bu dinamik olarak alt-kafeslere etiketleme yapabilme imkanı, tek bir Viterbi kodçözücü ve bir bellek kullanarak her bir alt-kafes üzerinden sırayla metrik hesaplamaya olanak verir. Eğer kullanılacak uygulama için hız önemli bir parametre ise, bir çok (mesela alt-kafes sayısı kadar) Viterbi kodçözücü paralel olarak çalıştırılarak sonunda elde edilen metrikler karşılaştırılarak sonuca hızla ulaşılabilir. GÜ kodlarının paralellüğinden ve dinamik etiketlenme özelliğinden kaynaklanan bu esneklik, kodçözücü tasarımı uygulamalarında önemli kolaylık ve esneklik sağlar. GÜ kodlarının özellikle *tüm devre (TD)* uygulamalarında alt-kafesin bahsedilen avantajı çok yararlıdır. Örnek 4.21 bu esnek etiketleme işlemini göstermek için sunulmaktadır.

Örnek 4.21: $C = (12, 7, 4)$ GÜ kodunun kafes yapısının (T') , kodçözümü analizinin yapılabilmesi için önce üreteç matrisi aşağıdaki gibi oluşturulmalıdır.

$$G = \begin{bmatrix} G_y \\ G_z G_z G_z G_z \\ G_z G_z G_z G_z \end{bmatrix} = \begin{bmatrix} 110000000011 \\ 001100000011 \\ 000011000011 \\ 000000110011 \\ 000000001111 \\ 101010100000 \\ 000010101010 \end{bmatrix}$$

G üreteç matrisinin ilk 5 satırının ikili aritmetikte toplamı ile üreteç kodkelimelerinden bir tanesi olan $g_1 = [111111111111]$ elde edilir. Ekleme satırları olan son iki satırın doğrusal kombinasyonları ile de geri kalan üreteç kodkelimeleri $g_2 = [101010100000]$, $g_3 = [000010101010]$ ve $g_4 = [101000001010]$ elde edilir. Bütün alt kafesler i.in etiketleme işlemi aynı olduğundan, bir tanesini detaylı açıklamak işlemin tamamını anlatmak için yeterli olacaktır. Bu örnek için gerekli alt-kafeslerin sayısı $2^q = 2^2 = 4$ olur ki burada q değeri (4.31) denkleminde açıklandığı gibi $q = 7 - 5 = 2$ olarak elde edilir. Gösterim için g_4 üreteç kodkelimesi kullanılacaktır.

$$g_4 = [\quad 10 \quad \quad 10 \quad \quad 00 \quad \quad 00 \quad \quad 10 \quad \quad 10 \quad]$$



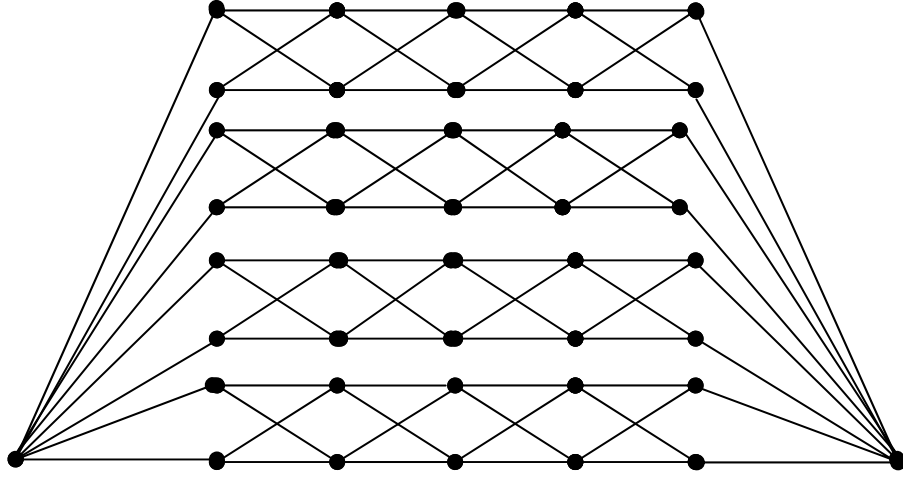
Şekil 4.3: Üreteç kodkelimesinin, g_4 , alt-kafese etiketlenmesi.

Yukarıdaki alt-kafeste ilk ve son düğümler hariç, her bir düğüme 2 dal sonlanmaktadır. Bir durumun, yani düğümün metriği hesaplanırken çıkarma, karesini alma, çarpma ve bir önceki metrikle toplama işlemleri yapılır. Yukarıdaki kafes yapısında bir derinlikte düz ya da çapraz aynı etiketli bitler bulunduğundan çıkarma ve çarpma işlemlerinin sonucu bir diğeri için tekrar kullanılabilir. Dolayısıyla kayda değer biçimde çıkarma ve çarpma işlemlerinden tasarruf edilmiş olur. Buna ilaveten; sonuçta elde edilen en iyi metrik değerinin ilişkili patikasını elde edebilmek için gerekli saklama işleminde çapraz dallar '1' ile, düz dallar '0' ile temsil edilirse daha az bellek ile saklama yapmak mümkün olabilecektir.

Aynı işlemler g_1 , g_2 ve g_3 içinde tekrarlanır ve en sonunda aralarında en iyi metrik ile ilişkili patika seçilir. Uygulamadaki en önemli dezavantaj, kod uzunluğu arttıkça doğrusal değil de üstel olarak artan alt-kafeslerin sayısı, yani bütün kafesin karmaşıklığıdır. Kafes yapılarının daha iyi kavranması için bazı GÜ kodlarının kafes yapıları, buradan sonraki kısımda şekillerle sunulacaktır.

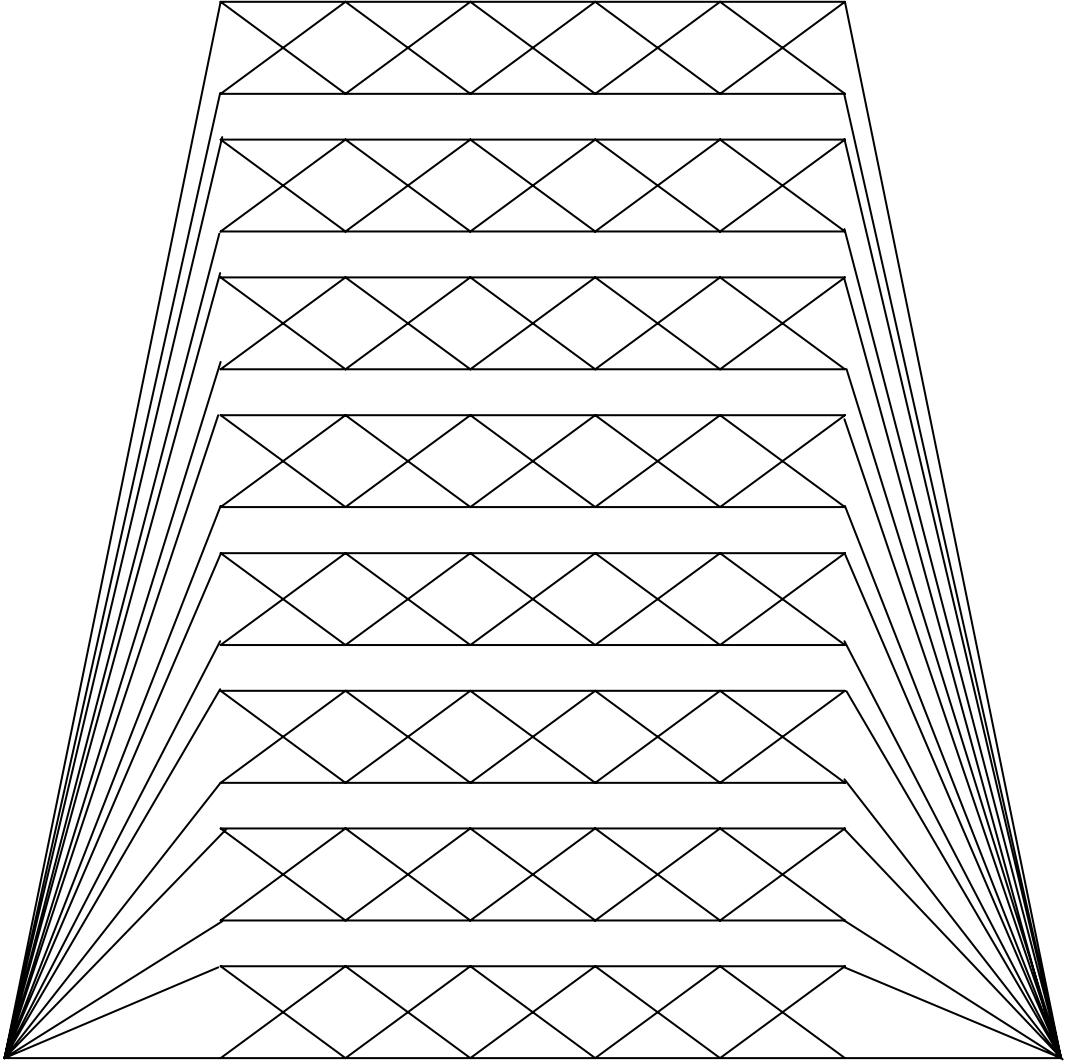
4.4.1. Hamming Mesafesi 4 olan Geometrik Ürün kodlarının kafesleri

Bu kısımda Hamming mesafesi 4 olan GÜ kodlarından bazıları için kafes yapıları, daha önce anlatıldığı gibi çizilerek, kafes parametreleri çizilen her şeklin altında açıklamalı olarak aşağıdaki gibi elde edilmiştir.



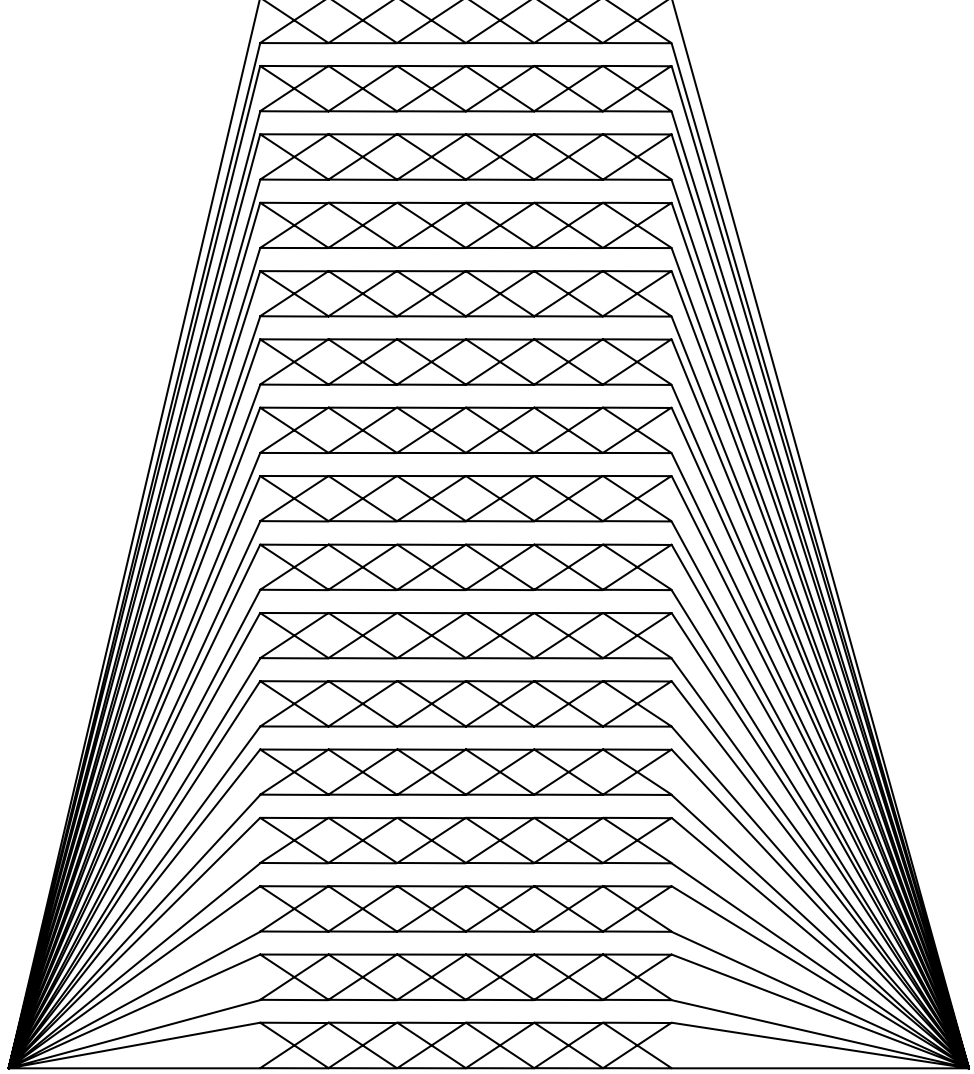
Şekil 4.4: $C = (12, 7, 4)$ GÜ kodu kafes yapısı.

Yukarıdaki $C = (12, 7, 4)$ GÜ kodu kafesinin her bir dalına etiketlenen bit sayısı yani dal boyutu 2'dir. Dolayısıyla tezin 2. bölümünde (2.4) ve (2.5) denklemleriyle bahsedilen kafes karmaşıklığı hesabı için gerekli parametreler: Dal karmaşıklığı, $E(C) = 320$, durum karmaşıklığı, $|V| = 42$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 320 - 42 + 1 = 599$ olarak hesaplanır.



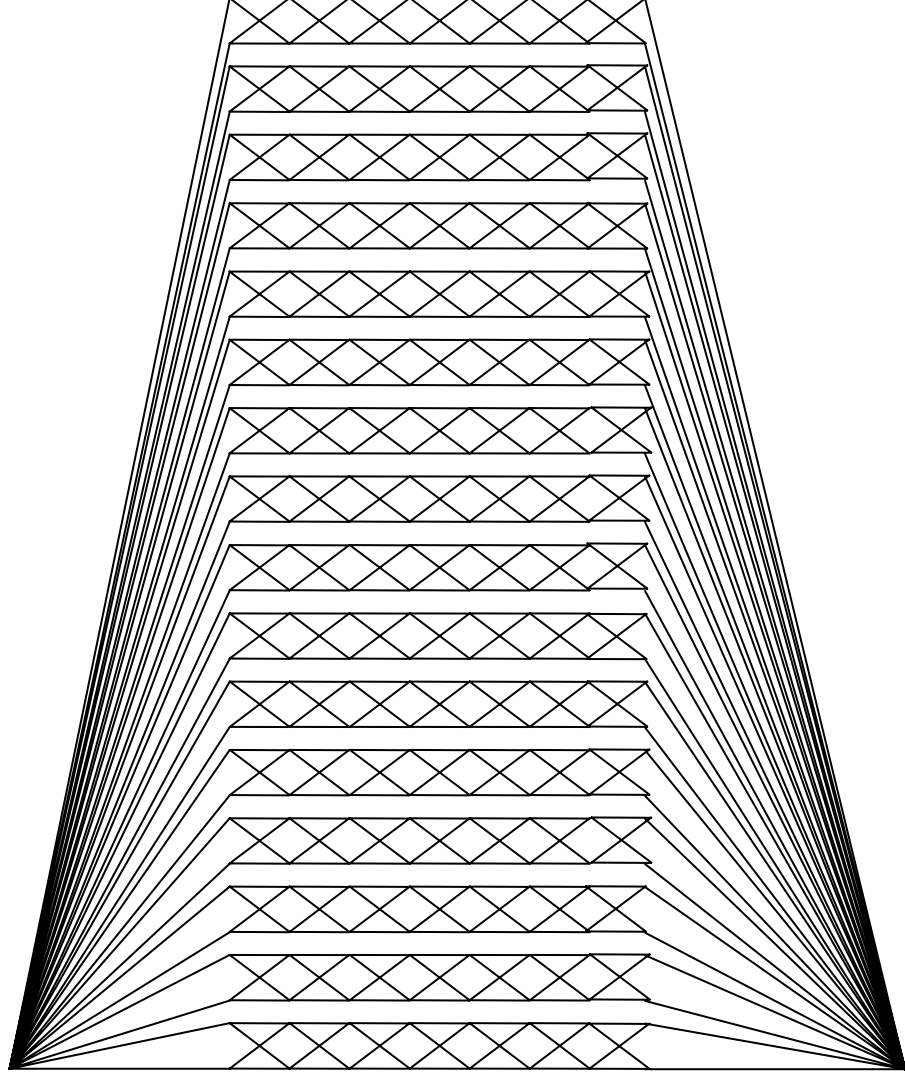
Şekil 4.5: $C = (14, 9, 4)$ GÜ kodu kafes yapısı.

Yukarıdaki $C = (14, 9, 4)$ GÜ kodu kafesi için; dal karmaşıklığı, $E(C) = 768$, durum karmaşıklığı, $|V| = 98$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklı $KK = 2 * 768 - 98 + 1 = 1439$ olan hesaplanır.



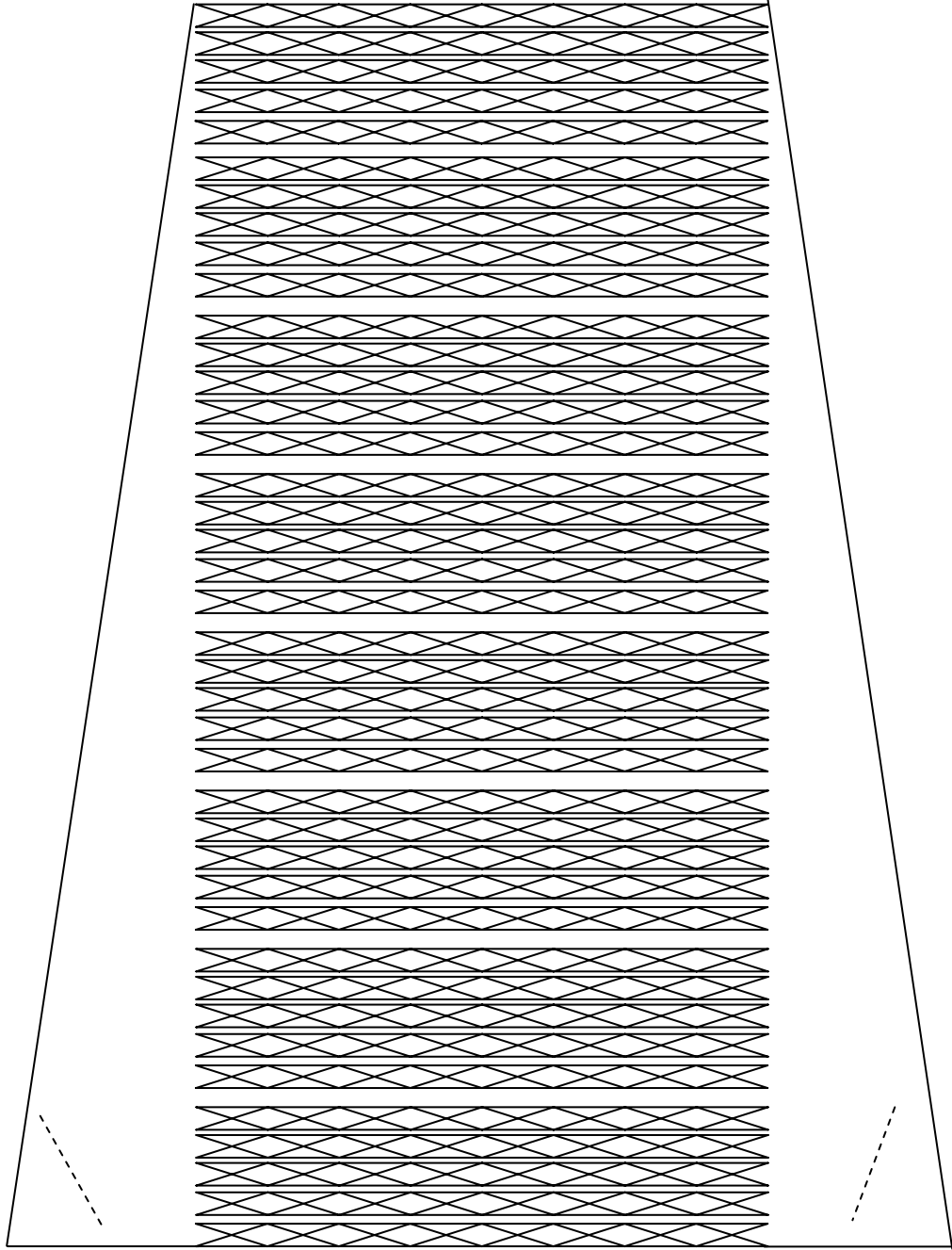
Şekil 4.6: $C = (16, 11, 4)$ GÜ kodu kafes yapısı.

Şekildeki $C = (16, 11, 4)$ GÜ kodu kafesi için; dal karmaşıklığı, $E(C) = 1792$, durum karmaşıklığı, $|V| = 226$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 1792 - 226 + 1 = 3359$ olarak hesaplanır.



Şekil 4.7: $C = (18, 12, 4)$ GÜ kodu kafes yapısı.

Şekildeki $C = (18, 12, 4)$ GÜ kodu kafesi için; dal karmaşıklığı, $E(C) = 2048$, durum karmaşıklığı, $|V| = 258$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 1792 - 226 + 1 = 3839$ olarak hesaplanır.

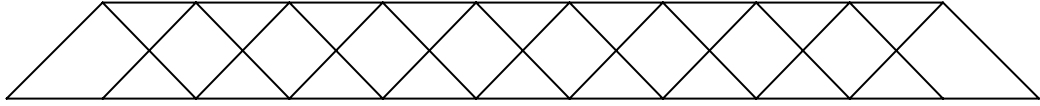


Şekil 4.8: $C = (20, 14, 4)$ GÜ kodu kafes yapısı.

Şekildeki $C = (20, 14, 4)$ GÜ kodu kafesi için; dal karmaşıklığı, $E(C) = 4608$, durum karmaşıklığı, $|V| = 1154$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 4608 - 1154 + 1 = 8063$ olarak hesaplanır. Çizilen şekilde kesikli

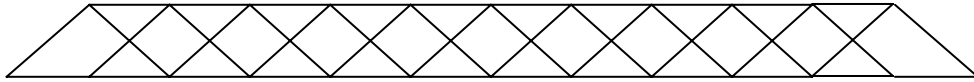
çizgi ile gösterilen çizgiler, ilk düğümden ve son düğümden kelebek alt-kafeslere olan girişleri basit bir şekilde temsil etmek amacıyla konulmuştur.

Şu ana kadar gösterilen kafes şekilleri bazı GÜ kodlarının bütün bir kafes yapısını göz önüne sermek için çizilmiştir. Burada dikkat edilmesi gereken husus, kod ölçüleri arttıkça kafes boyutlarının üstel bir şekilde arttığıdır. En son çizilen (20, 14, 4) GÜ kodundan bir büyük kodun kafes yapısını artık sayfaya sığdırmak mümkün olmamaktadır. Dolayısıyla bundan sonra çizilecek kafes şekillerinde sadece kodların alt-kafesleri, T, çizilecek ve alt-kafeslerden kaç tane olduğu belirtilecektir.



Şekil 4.9: $C = (22, 16, 4)$ GÜ kodu alt-kafesi, T.

Şekildeki $C = (22, 16, 4)$ GÜ kodu kafesinin alt-kafes sayısı, (4.31) denkleminde açıklanan q parametresi, $q = 16 - 10 = 6$, kullanılarak, $2^q = 2^6 = 64$ şeklinde kolayca hesaplanabilir. Söz konusu GÜ kodu kafesi için; dal karmaşıklığı, $E(C) = 10240$, durum karmaşıklığı, $|V| = 1282$ ve dal boyutu $b_i = 2$ olarak bulunur. Böylece kafes karmaşıklığı $KK = 2 * 10240 - 1282 + 1 = 19199$ olarak hesaplanır.



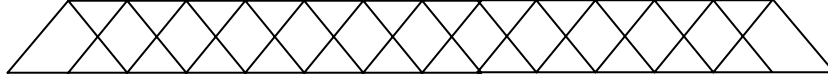
Şekil 4.10: $C = (24, 18, 4)$ GÜ kodu alt-kafesi, T.

Şekildeki $C = (24, 18, 4)$ GÜ kodu kafesinin alt-kafes sayısı $q = 18 - 11 = 7$, kullanılarak, $2^q = 2^7 = 128$ şeklinde hesaplanır. Aynı kodun kafesi için dal karmaşıklığı, $E(C) = 22528$, durum karmaşıklığı, $|V| = 2818$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 22528 - 2818 + 1 = 42239$ olarak hesaplanır.



Şekil 4.11: $C = (26, 20, 4)$ GÜ kodu alt-kafesi, T.

Şekildeki $C = (26, 20, 4)$ GÜ kodu kafesi için; alt-kafes sayısı $q = 20 - 12 = 8$, kullanılarak, $2^q = 2^8 = 256$ şeklinde kolayca hesaplanabilir. Bu durumda GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 49152$, durum karmaşıklığı, $|V| = 6146$ ve dal boyutu $b_i = 2$ olarak, kafes karmaşıklığı da; $KK = 2 * 49152 - 6146 + 1 = 92159$ olarak hesaplanır.

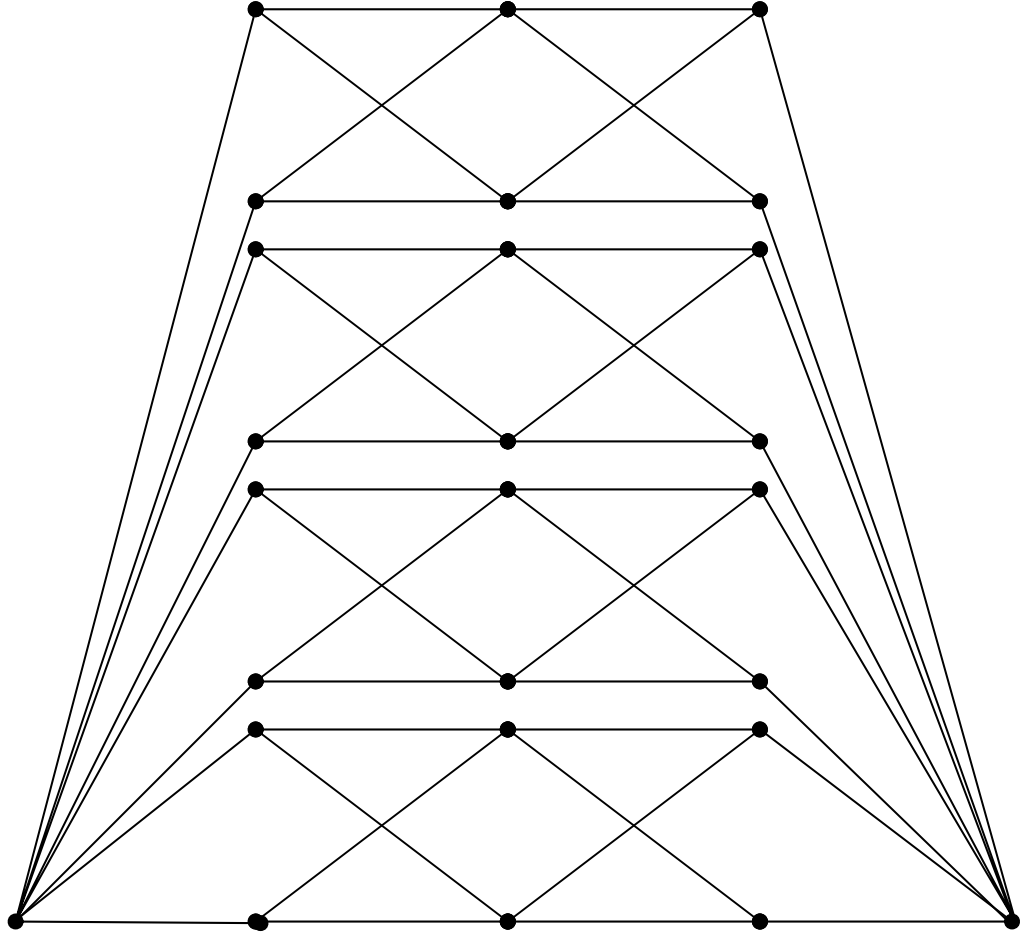


Şekil 4.12: $C = (28, 22, 4)$ GÜ kodu alt-kafesi, T.

Şekildeki $C = (28, 22, 4)$ GÜ kodu kafesi için alt-kafes sayısı $q = 22 - 13 = 9$, kullanılarak, $2^q = 2^9 = 512$ şeklinde elde edilir. Bu GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 106496$, durum karmaşıklığı, $|V| = 13314$ ve dal boyutu $b_i = 2$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 106496 - 13314 + 1 = 199679$ olarak hesaplanır.

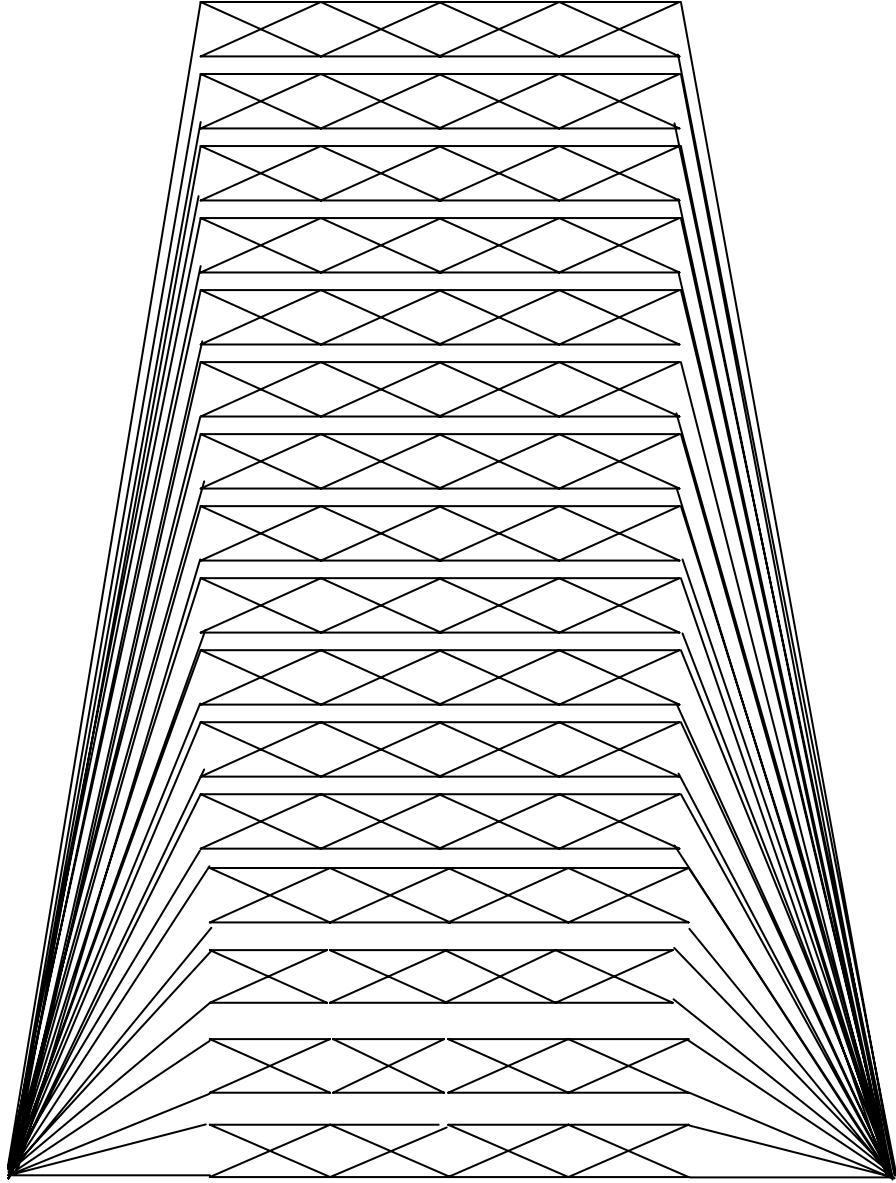
4.4.2. Hamming Mesafesi 8 Olan Geometrik Ürün Kodlarının Kafesleri

Bu kısımda ise Hamming mesafesi 8 olan GÜ kodlarından bazıları için kafes yapıları şekillerle gösterilip bunlara ait kafes parametreleri incelenecektir.



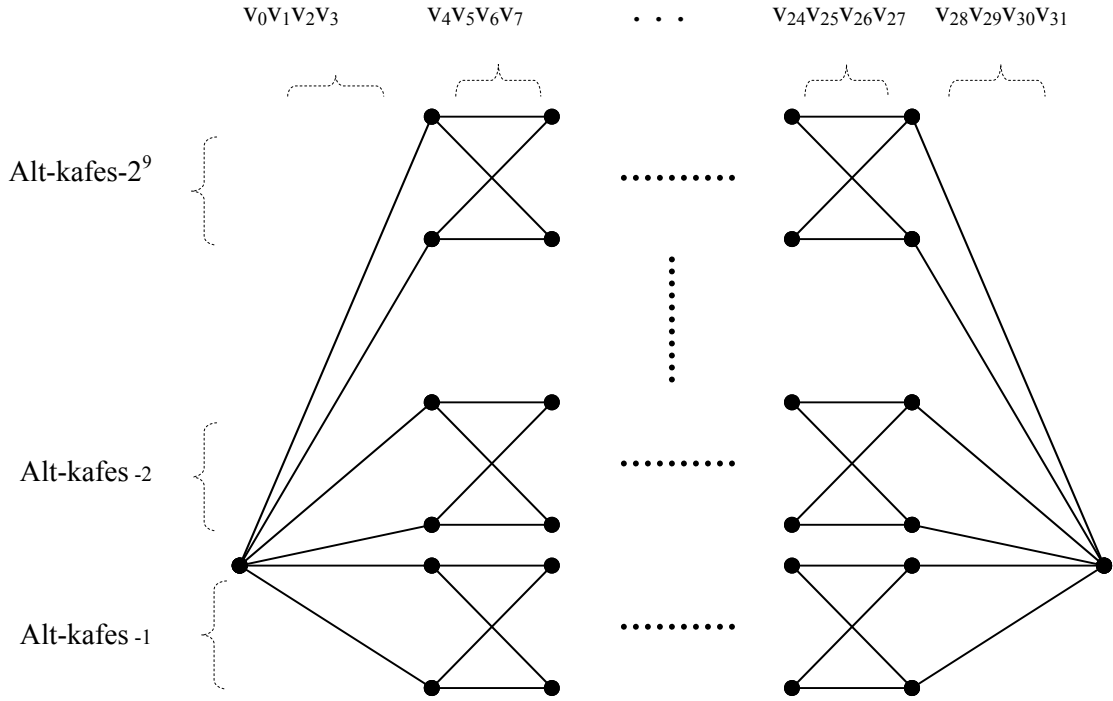
Şekil 4.13: $C = (16, 5, 8)$ GÜ kodu kafes yapısı, T' .

Şekildeki $C = (16, 5, 8)$ GÜ kodu kafesinin alt-kafes sayısı $q = 5 - 3 = 2$, kullanılarak, $2^2 = 2^2 = 4$ şeklinde kolayca hesaplanabilir. Bu GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 1280$, durum karmaşıklığı, $|V| = 26$ ve dal boyutu $b_i = 4$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 1280 - 26 + 1 = 2535$ olarak hesaplanır.



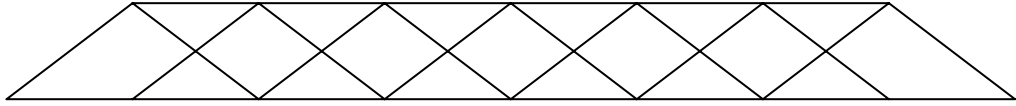
Şekil 4.14: $C = (24, 9, 8)$ GÜ kodu kafes yapısı, T' .

Şekildeki $C = (24, 9, 8)$ GÜ kodu kafesinin alt-kafes sayısı $q = 9 - 5 = 4$, kullanılarak, $2^q = 2^4 = 16$ şeklinde kolayca hesaplanabilir. Bu GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 5120$, durum karmaşıklığı, $|V| = 162$ ve dal boyutu $b_i = 4$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 5120 - 162 + 1 = 10079$ olarak hesaplanır. Sonuçtan da anlaşılacağı gibi dal boyutu parametresi b_i kafes karmaşıklığını üstel biçimde artırır.



Şekil 4.15: $C = (32, 16, 8)$ GÜ kodu kafes yapısı, T' .

Şekildeki kafes (4.22) denklemiyle inşa edilen $C = (32, 16, 8)$ geliştirilmiş GÜ kodunun kafesi olup bu kafesin alt-kafes sayısı $q = 16 - 7 = 9$, kullanılarak, $2^q = 2^9 = 512$ şeklinde hesaplanabilir. Bu geliştirilmiş GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 229376$, durum karmaşıklığı, $|V| = 7168$ ve dal boyutu $b_i = 4$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 229376 - 7168 + 1 = 451585$ olarak hesaplanır.



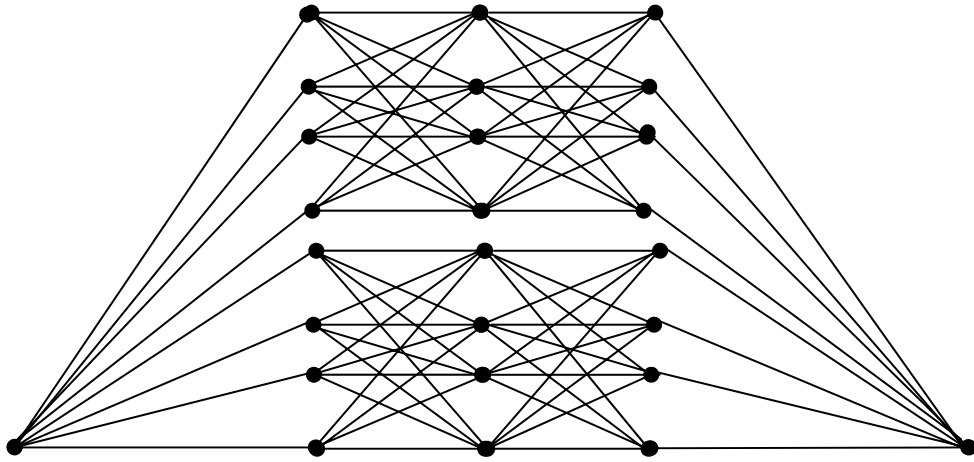
Şekil 4.16: $C = (64, 19, 16)$ GÜ kodu alt-kafesi, T .

Şekildeki $C = (64, 19, 16)$ GÜ kodu kafesinin alt-kafes sayısı $q = 19 - 7 = 12$, kullanılarak, $2^q = 2^{12} = 4096$ şeklinde kolayca hesaplanabilir. Bu GÜ kodu kafesinin dal karmaşıklığı, $E(C) = 29360128$, durum karmaşıklığı, $|V| = 57346$ ve dal boyutu $b_i = 8$ olarak bulunur. Bu durumda kafes karmaşıklığı $KK = 2 * 29360128 - 57346 + 1 = 58662910$ olarak hesaplanır.

4.4.3. Geometrik Ürün Kodları Kafeslerinin Diğer Blok Kodlarının Kafesleri İle Karmaşıklık Yönünden Karşılaştırılması

Bu bölümde; tez çalışmasında önerilen GÜ kodları ve bu kodların kodçözümü için tasarlanan kafes yapılarının karmaşıklığı diğer benzer kodların kafes yapılarıyla karşılaştırılacaktır. Kafes karmaşıklığı ölçütü olarak Viterbi Algoritması (VA) kullanıldığında gerekli işlem sayısını veren denklem takımı (2.4) ve (2.5) ikinci bölümde detayları ile anlatılmıştı. Karmaşıklık karşılaştırılması yapılırken bu denklemlerden yararlanılacaktır. Literatürde bulunan, GÜ kodları gibi ayrıştırılabilir yapıda olan blok kodlarının kafes yapıları ile karşılaştırma genel olarak Tablo 4.2’de yapılacaktır. Ayrıca, Genelleştirilmiş Dizi Kodları (Generalised Array Codes, GAC) [17] ile GÜ kodlarından (12, 7, 4) ve (16, 11, 4) ölçülerindeki kodların kafes yapıları ayrıntılı biçimde ele alınarak kıyas ve analiz yapılacaktır.

İlk karşılaştırma (12, 7, 4) GAC kodu kafesi [82] ile (12, 7, 4) GÜ kodu kafesi arasında yapılacaktır. Bunun için aşağıdaki şeklide GAC kodu kafesi ele alınacaktır.



Şekil 4.17: $C = (12, 7, 4)$ GAC kodu kafes yapısı [82].

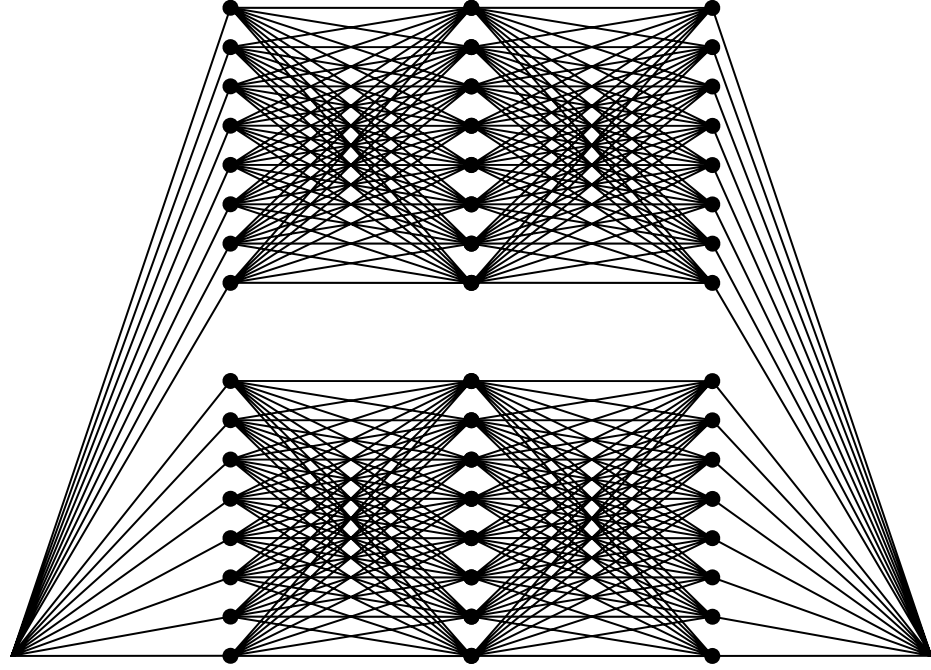
Yukarıdaki (12, 7, 4) GAC kodu kafesinde düğümlerin yani durumların sayısı 26 iken bu sayı (12, 7, 4) GÜ kodu kafesinde (Şekil 4.4) 42'dir. İncelenen GAC kafesindeki dalların sayısı 80 iken, GÜ kodu kafesinde de bu sayı 80'dir. Aralarındaki esas farkı ortaya koyan parametre; dallara etiketlenen bitlerin sayısıdır (b_i). (12, 7, 4) GAC kodu

kafesinde etiketlenen bit sayısı $b_1 = 3$ iken, bu parametre GÜ kodun kafesinde 2'dir. Dolayısıyla dal ve düğüm sayıları benzer olsa da, GÜ kodunun kafesi daha az karmaşıklığa sahiptir. Tablo 4.2'de gösterildiği gibi (12, 7, 4) GAC kodu kafesinin karmaşıklığı $KK = 1255$ iken GÜ kodu kafesinin karmaşıklığı $KK = 599$ dur. Görüldüğü gibi bu karşılaştırmada yarıya yakın bir fark söz konusudur.

Viterbi algoritması için işlem sayısı karmaşıklığı çok daha az olan GÜ kodu kafesi, esneklik yönünden de daha avantajlıdır. Şekil 4.4'ten görüleceği gibi (12, 7, 4) GÜ kodu kafesi 4 özdeş alt-kafesten oluşmakta olup 4 aynı Viterbi kodçözücüyü paralel çalıştırma olanağı sağlar ki bu da daha hızlı işlem yapmayı sağlar. Aynı zamanda bu yapı; bir tek Viterbi kodçözücüyü kullanarak tüm diğer 4 alt-kafeslerin kodçözümünü gerçeklemeye de olanak verir ki bu da hızın önemli olmadığı uygulamalar için maliyet açısından avantajlıdır. Diğer taraftan (12, 7, 4) GAC kodu kafesinin alt-kafes sayısı 2 olup GÜ kafesine göre yüzde elli daha az esnektir. Daha da önemlisi alt-kafesler kıyaslandığında bir düğüme giren dal sayısı GÜ kodunda 2 iken GAC kodunda 4'tür. Bu ise, bir düğümün metriğini hesaplarken, karmaşıklığı önemli ölçüde artırır. Çünkü bir düğüme 2 dal girdiğinde, 2 ayrı değer birbiriyle kıyaslanarak aradaki farkın sadece işaretine bakmak suretiyle büyük küçük ayrımı yapılabilir. GAC alt-kafesinde ise, bir düğümün metriğini hesaplarken 4 ayrı değer birbiri ile kıyaslanacak ve hassas değer kıyası yapılması gerekecektir.

Sonuç olarak (12, 7, 4) GÜ kodu kafesi, (12, 7, 4) GAC kodu kafesinden daha az karmaşık yapıda olup pratikteki uygulamalar içinde daha esnektir denebilir.

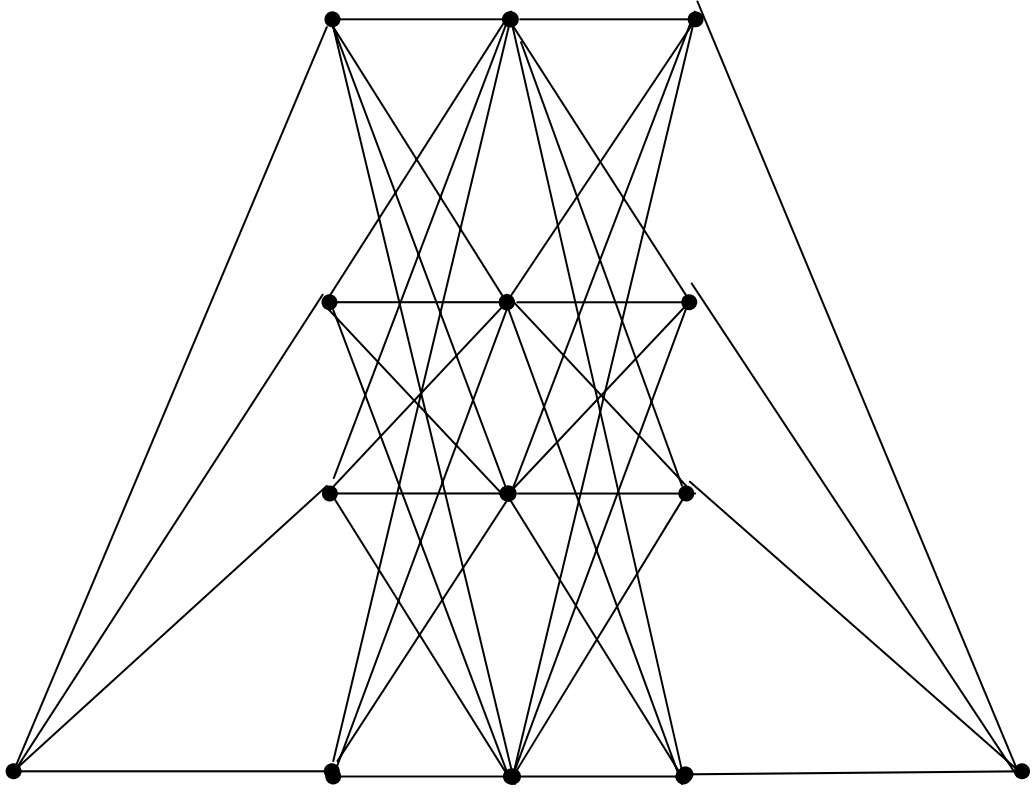
GÜ kod kafeslerinin az karmaşık yapısını daha net ortaya koyabilmek ve yukarıda varılan sonucu desteklemek amacıyla, (16, 11, 4) ölçülerindeki GAC ve GÜ kodlarının kafes yapıları da ayrıntılarıyla incelenecektir. Aşağıdaki şeklide (16, 11, 4) GAC kodu kafesi görülmektedir.



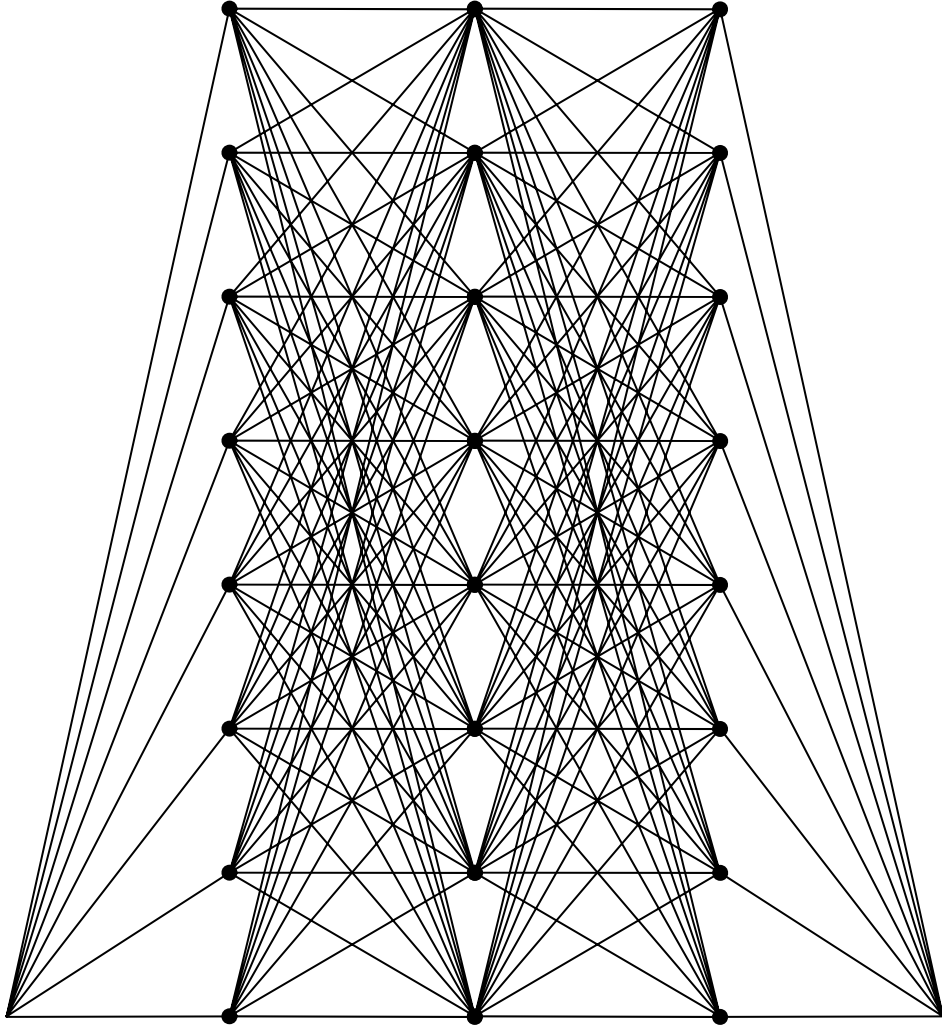
Şekil 4.18: $C = (16, 11, 4)$ GAC kodu kafes yapısı [82].

Şekilde görülen $(16, 11, 4)$ GAC kodu kafesinde, düğümlerin yani durumların sayısı 50 iken bu sayı $(16, 11, 4)$ GÜ kodu kafesinde (Şekil 4.6) 226 dir. İncelenen GAC kafesindeki dalların sayısı 288 iken GÜ kodu kafesinde bu sayı 448 dir. $(16, 11, 4)$ GAC kodu kafesinde etiketlenen bit sayısı $b_i = 4$ iken, bu parametre GÜ kodu kafesinde 2 dir. Tablo 4.2’de gösterildiği gibi; $(16, 11, 4)$ GAC kodu kafesinin karmaşıklığı $KK = 9167$ iken, GÜ kodu kafesinin karmaşıklığı $KK = 3359$ dur. Bu durumda aradaki fark daha da artmış olup, 3 katına yakın bir fark söz konusudur. Görüldüğü gibi kodlar büyüdükçe GÜ kod kafeslerinin karmaşıklık kazancı daha da artmaktadır.

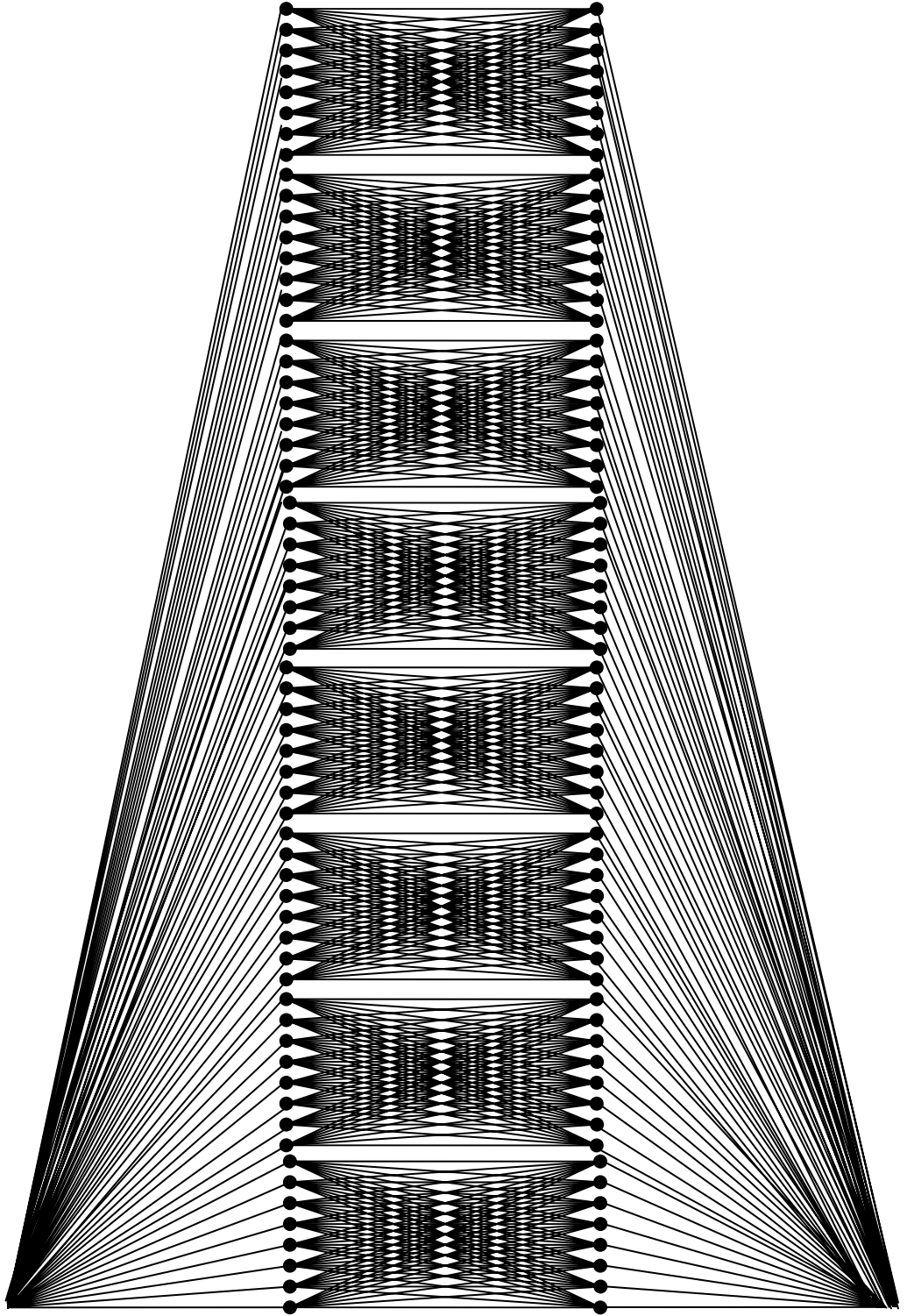
Aşağıdaki şekillerde, literatürde bulunan diğer bazı blok kodların kafes yapıları sunulduktan sonra kafes karmaşıklığı karşılaştırması da Tablo 4.2’de gösterilecektir.



Şekil 4.19: $C = (12, 6, 4)$ Dizi kodunun kafes yapısı [82].



Şekil 4.20: $C = (16, 9)$ Dizi kodunun kafes yapısı [82].



Şekil 4.21: $C' = (7, 3, 5)$ RS kodu kafes yapısı [82]; $C = (21, 9)$ blok koduna denktir.

Aşağıdaki tabloda genel olarak, GÜ kodlarının bir çoğu için kafes karmaşıklık değerleri ve benzer ölçülerdeki, literatürde bulunan, diğer bazı blok kodların kafes karmaşıklık değerleri sunulmuştur.

Tablo 4.2 : Bazı GÜ ve diğer blok kodlar için kafes karmaşıklıkları

Kodlar	Dal karmaşıklığı, $E(C)$, ve durumların sayısı, V .	Viterbi Algoritması için kafes karmaşıklığı, KK .
(12,7,4) GÜ	$E(C) = 320$, $V = 42$ ($b_i=2$ tüm dallar için)	$KK = 599$
(14, 9, 4) GÜ	$E(C) = 768$, $V = 98$ ($b_i=2$ tüm dallar için)	$KK = 1439$
(16,11,4) GÜ	$E(C) = 1792$, $V = 226$ ($b_i=2$ tüm dallar için)	$KK = 3359$
(18,12,4) GÜ	$E(C) = 2048$, $V = 258$ ($b_i=2$ tüm dallar için)	$KK = 3839$
(20, 14, 4) GÜ	$E(C) = 4608$, $V = 1154$ ($b_i=2$ tüm dallar için)	$KK = 8063$
(22, 16, 4) GÜ	$E(C) = 10240$, $V = 1282$ ($b_i=2$ tüm dallar için)	$KK = 19199$
(24, 18, 4) GÜ	$E(C) = 22528$, $V = 2818$ ($b_i=2$ tüm dallar için)	$KK = 42239$
(26, 20, 4) GÜ	$E(C) = 49152$, $V = 6146$ ($b_i=2$ tüm dallar için)	$KK = 92159$
(28, 22, 4) GÜ	$E(C) = 106496$, $V = 13314$ ($b_i=2$ tüm dallar için)	$KK = 199679$
(16, 5, 8) GÜ	$E(C) = 1280$, $V = 80$ ($b_i=4$ tüm dallar için)	$KK = 2535$
(24, 9, 8) GÜ	$E(C) = 5120$, $V = 162$ ($b_i=4$ tüm dallar için)	$KK = 10079$
(32, 16, 8) GÜ	$E(C) = 229376$, $V = 7168$ ($b_i=4$ tüm dallar için)	$KK = 451585$

(64, 19, 16) GÜ	$E(C) = 29360128$ $V = 57346$ ($b_i=8$ tüm dallar için)	KK = 58662910
(12,7,4) GAC	$E(C) = 640$, $V = 26$ ($b_i=3$ tüm dallar için)	KK = 1255
(12,6,4) Dizi kodu [82]	$E(C) = 320$, $V = 14$ ($b_i=3$ tüm dallar için)	KK = 627
(16,11,4) GAC	$E(C) = 4608$, $V = 50$ ($b_i=4$ tüm dallar için)	KK = 9167
(16,9) Dizi kodu [82]	$E(C) = 2304$, $V = 26$ ($b_i=4$ tüm dallar için)	KK= 4583
(7,3,5) RS ((21,9) blok koduna eşdeğer) [82]	$E(C) = 270336$, $V = 130$ ($b_i: 6, 9$ ve 6 gibi değişen değerler alır)	KK= 540543

Tablodan da görüldüğü üzere GÜ kod kafesleri benzerlerine kıyasla çok daha az karmaşık yapıdadır. Aynı zamanda basit paralel alt-kafesleri dolayısıyla daha esnek ve uygulanabilir özelliğe sahiptir.

Yukarıda detaylarıyla anlatılan ve Tablo 4.2 ile karşılaştırması yapılan GÜ kod kafes yapısından tüm diğer GÜ kodlarının kafes karmaşıklıklarının kolayca çıkarılabilmesi ve dolayısıyla tasarlanan GÜ kod kafeslerinin genelliğinin ispatlanması için Teorem 4.2 ve Teorem 4.3 önerilerek ispatlanacaktır.

Teorem 4.2: Bir GÜ kodunun, Şekil 4.1 ile gösterildiği biçimde elde edilen bir alt-kafesindeki (T) dallarının sayısı $4 \cdot (n/n_1 - 1)$ ve durumlarının sayısı $2 \cdot n/n_1$ ile tanımlıdır. Burada n kodun uzunluğu, n_1 ise eleman kodunun (C_1) uzunluğudur.

İspat: Önerilen teoremi ispatlayabilmek için Şekil 4.1 ile gösterilen genel alt-kafes yapısı özdeş parçalarına göre incelenecektir. Bu özdeş parçalar; kelebek parçası ($\bowtie$),

ile ilk ve son parçalardır, ($\swarrow$, $\searrow$). Şekil 4.1 den görüleceği gibi bir alt-kafesteki kelebek parçalarının sayısı $(n/n_1 - 2)$ 'dir. Her bir kelebek parçası 4 adet daldan oluştuğundan, bir alt-kafeste bulunan kelebek parçalarındaki toplam dal sayısı $4 \cdot (n/n_1 - 2)$ 'dir. Ayrıca her bir alt kafeste, 2 tane ilk ve 2 tanede son parçalardan gelen 4 adet dal bulunur. Dolayısıyla, bir GÜ kodunun alt-kafesindeki dalların sayısı $4 \cdot (n/n_1 - 2) + 4 = 4 \cdot (n/n_1 - 1)$ olur. Bir alt-kafesteki durumların sayısı, Şekil 4.1'den görüleceği üzere, kelebek kafeslerdeki durumların sayısı $2 \cdot (n/n_1 - 1)$ ile ilk ve son durumların eklenmesi ile, $2 \cdot (n/n_1 - 1) = 2 \cdot n/n_1$ olarak bulunur.

Teorem 4.3: Bir GÜ kodunun (C) , Viterbi kodçözümü kullanıldığında kafes karmaşıklığı, $KK = 2^{q+n_1+3} (n/n_1 - 1) - 2^{q+1} \cdot (n/n_1 - 1) + 3$ ile bulunur. Burada n kodun uzunluğu, n_1 ise eleman kodunun (C_1) uzunluğu ve q ise Denklem (4.31) ile tanımlı değişkendir.

İspat: Bu teorem Denklem (2.5) ile bulunan kafes kodçözme karmaşıklığı temel alınarak ispatlanacaktır. Bir GÜ kodunun kafes yapısının genel halinin gösterildiği Şekil 4.2 içinde bulunan alt-kafeslerin sayısının 2^q olduğu belirtilmişti. Bu sonuç ile birlikte Teorem 4.2 birleştirilerek, bir GÜ kod kafesinde bulunan dalların toplam sayısının $4 \cdot (n/n_1 - 1) \cdot 2^q = 2^{q+2} (n/n_1 - 1)$ olduğu açıkça görülebilir. Kafesteki kelebek parçalarında bulunan durumların toplam sayısı $2 \cdot (n/n_1 - 1) \cdot 2^q$ olarak Şekil 4.2'den elde edilir. Ayrıca ilk ve son durumlardan 2 durum gelir ve sonuçta bir GÜ kodunun kafesindeki toplam durum sayısı $(n/n_1 - 1) \cdot 2^{q+1} + 2$ olarak hesaplanır. Hesaplanan bu değerler, (2.5) kafes karmaşıklığı denkleminde yerine konulursa, bir GÜ kodu kafesinin, Viterbi kodçözme algoritması kullanıldığındaki kafes karmaşıklığı $KK = 2^{q+n_1+3} (n/n_1 - 1) - 2^{q+1} \cdot (n/n_1 - 1) + 3$ olarak elde edilir.

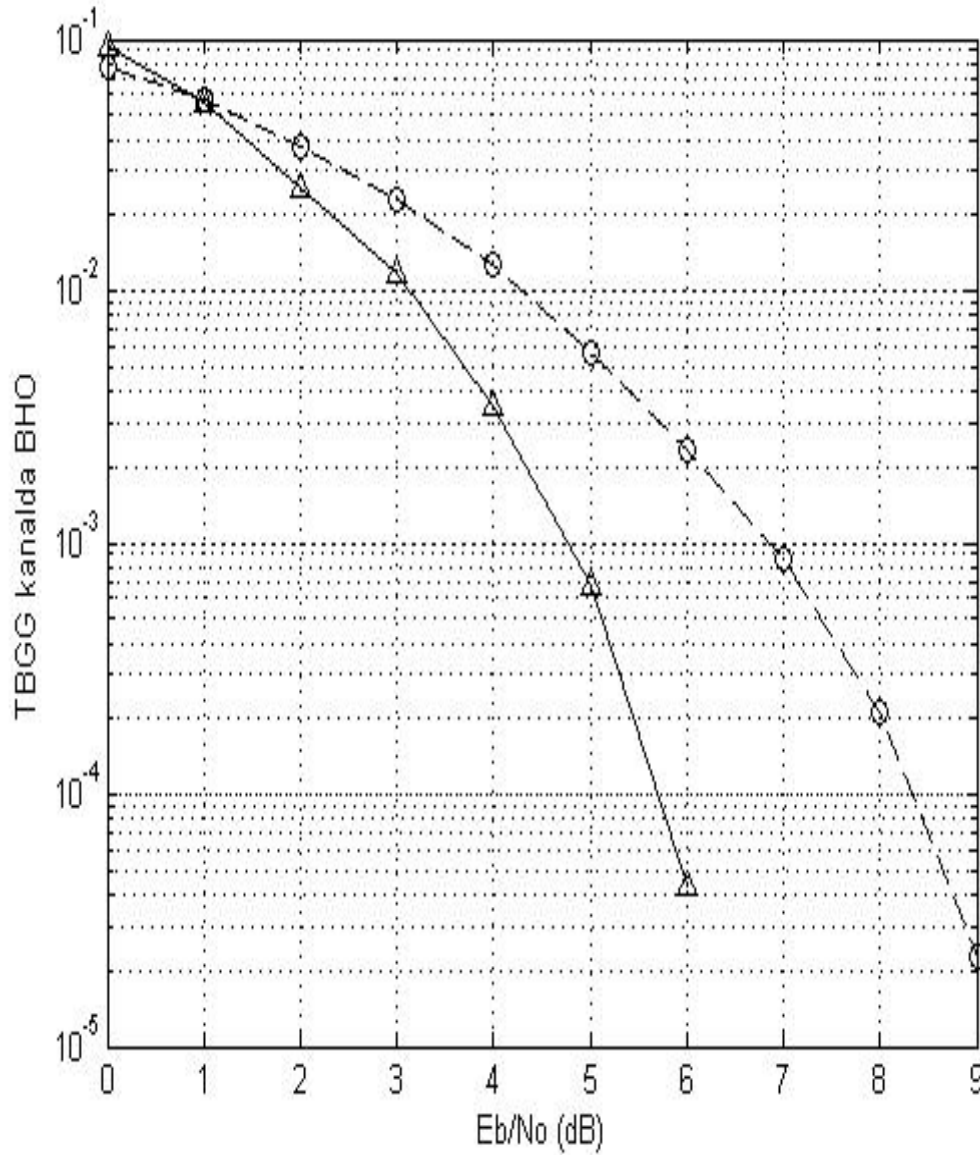
4.5. GEOMETRİK ÜRÜN KODLARI İÇİN BİLGİSAYAR BENZETİMİ İLE HATA BAŞARIMI

Önceki kısımlarda anlatılan kafes yapıları kullanılarak bilgisayar benzetimleri, sıfır ortalamalı ve varyansı $\sigma^2 = N_0/2$ olan, *Toplamsal Beyaz Gauss Gürültülü (TBGG)* kanal için gerçekleştirilmiştir. Burada $N_0/2$ çift taraflı *Güç Spektrum Yoğunluğudur (GSY)*. Bazı GÜ kodları için *hata başarımları*, E_b/N_0 (Desibel biriminde) ve *Bit Hata Oranı (BHO)* parametrelerine göre elde edilmiş olup burada E_b bir bilgi biti başına düşen enerji miktarıdır. Bilgisayar benzetimleriyle gerçekleştirilen iletişim modelinde, GÜ kodları *İkili Faz Kaydırmalı anahtarlama (İFKA)* modülasyonu kullanılarak mantıksal 1 biti +1 ve mantıksal 0 biti -1 değerlerine atanır. Alıcı kısımda, ideal kanal durum bilgisinin bulunduğu varsayılmış ve yumuşak kararlı Viterbi kodçözme uygulanmıştır. Ayrıca , BCJR ve İteratif kodçözme kullanılarak başarımların analizi bazı kodlar için yapılmıştır. İncelenen tüm GÜ kodların için, mükemmel bit ve blok senkronizasyonu (eşzamanlama) olduğu varsayılmıştır. Bu haberleşme modelinde bazı GÜ kodları için elde edilen bit hata oranları için eğriler aşağıda verilen şekillerdeki gibi elde edilmiştir. Diğer taraftan, karşılaştırma için kullanabilmek amacıyla 11 adet bitlerden oluşan bloklar İFKA modülasyonu gerçekleştirildikten sonra kodlayıcı ve kodçözücü kullanılmadan, TBGG kanalda, BHO değerleri elde edilmiş ve tüm eğrilerde kodlanmamış İFKA modülasyonunu temsil etmek üzere çizdirilmiştir. Bir örnek olarak bunun nasıl kullanılacağını göstermek gerekirse; (16, 11, 4) blok kodu kullanıldığında, kodlanmamış duruma göre, 2.6 dB (Desibel) kodlama kazancı sağlar.

Önce Viterbi algoritması kullanılarak bir çok GÜ kodu için elde edilen hata başarımları, sonra da BCJR ve İteratif kodçözme algoritmaları için bazı başarımların sonuçları sunulacaktır.

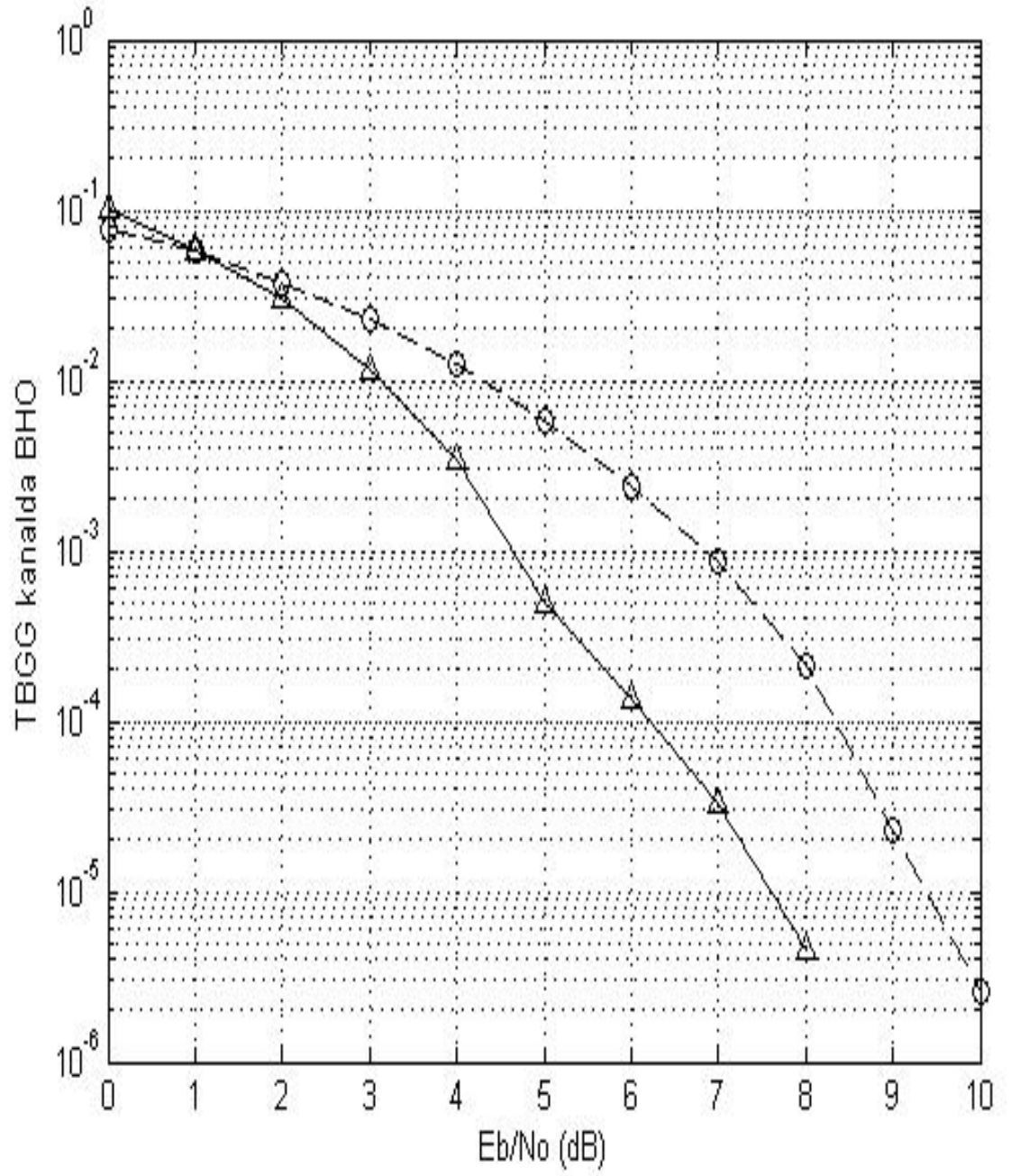
4.5.1. Viterbi Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımları

Bazı GÜ kodları için, Viterbi algoritması kullanılarak, en büyük olabilirlikli yumuşak kararlı kodçözümüyle elde edilen, TBGG kanal ortamında ki BHO başarımları bu bölümdeki şekillerle sunulmuştur.



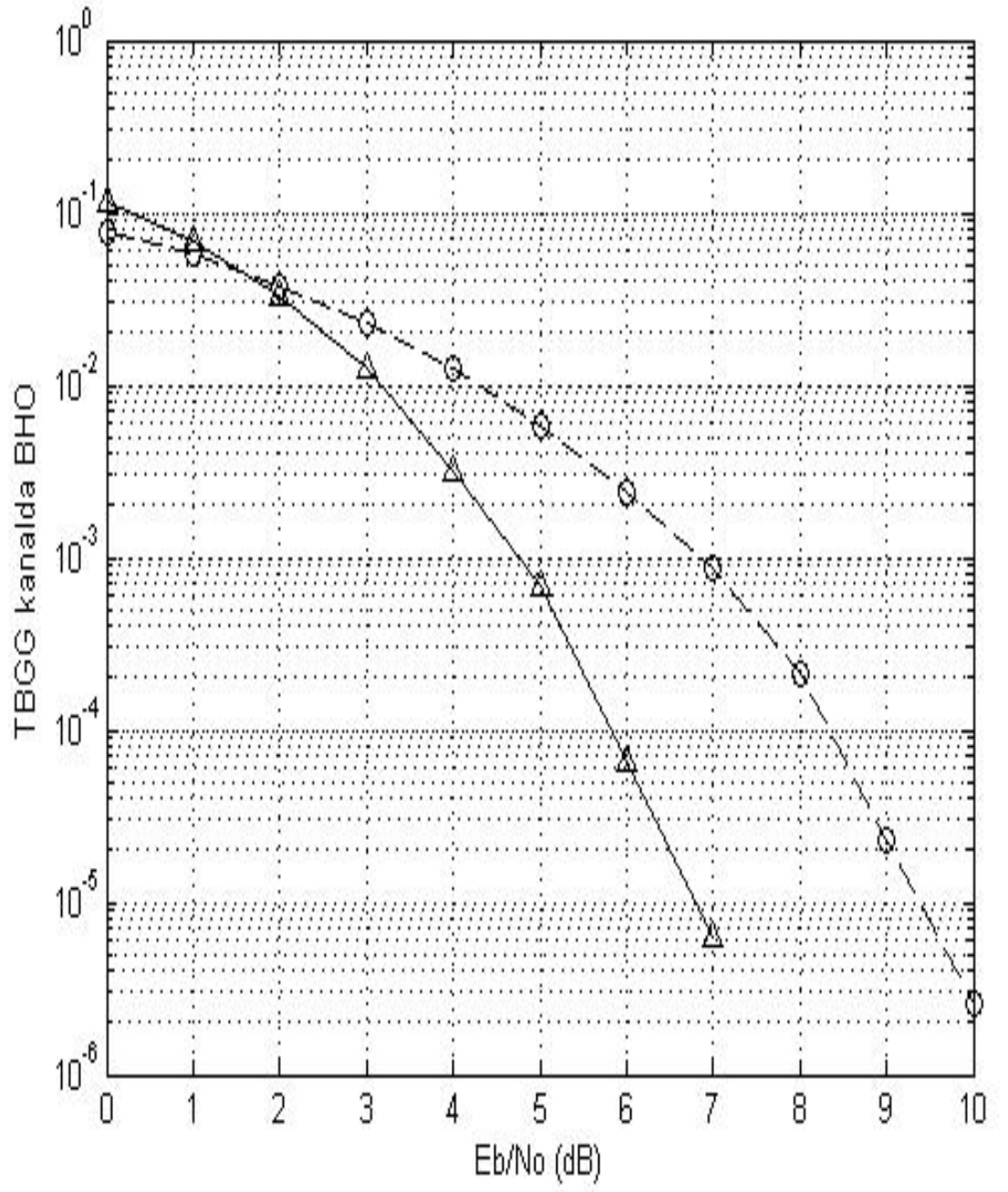
Şekil 4.22: $C = (12, 7, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(12, 7, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



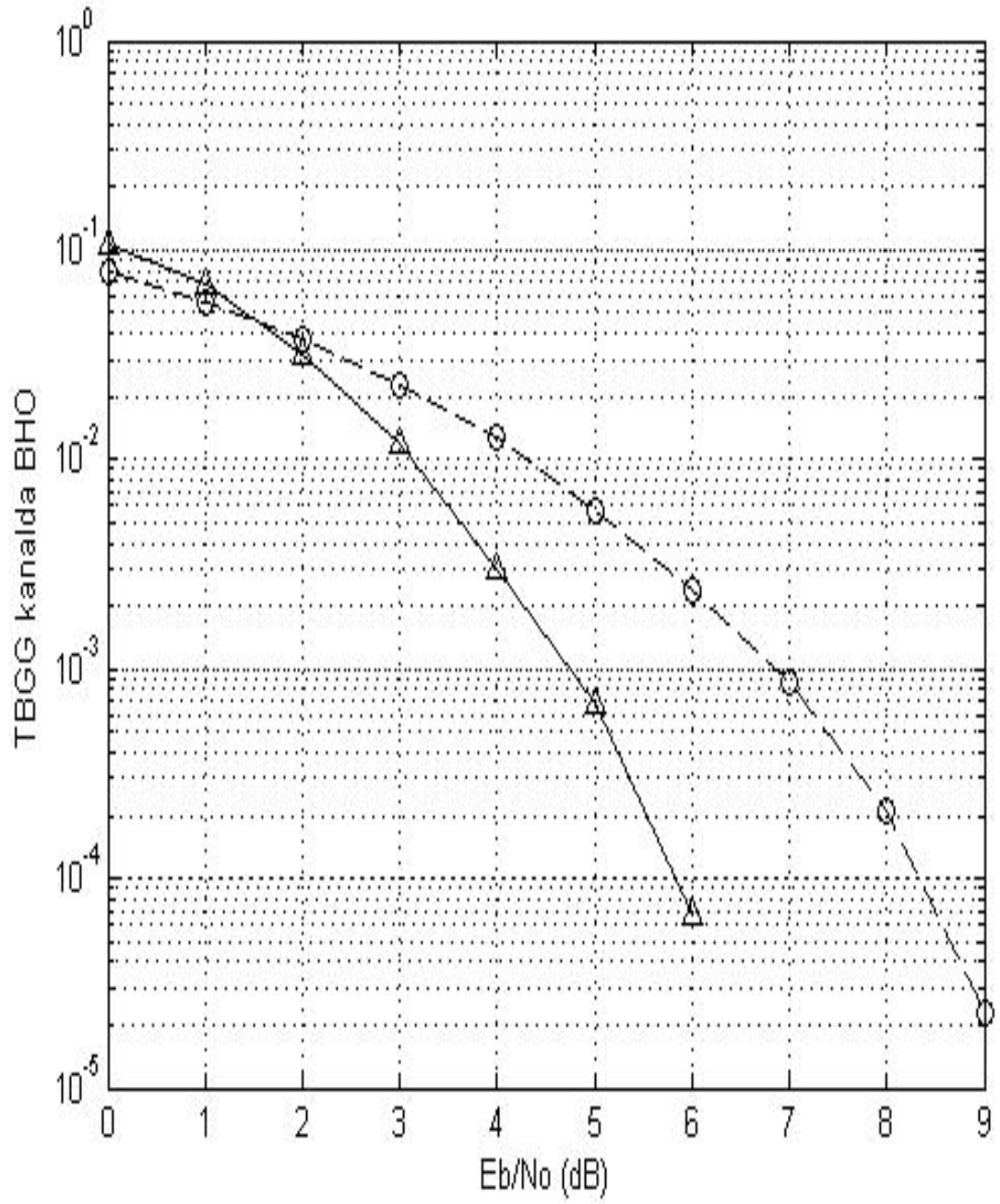
Şekil 4.23: $C = (14, 9, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(14, 9, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



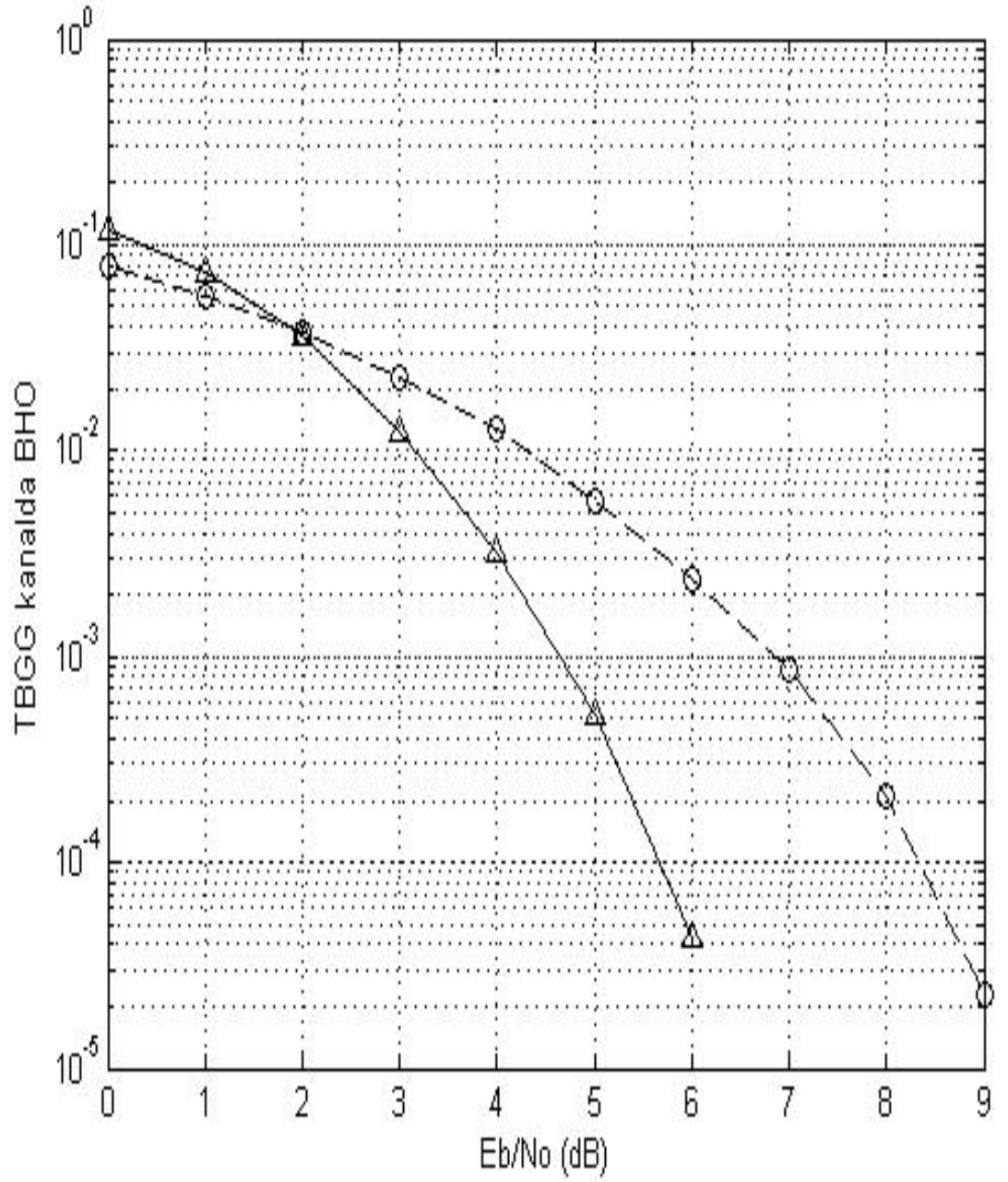
Şekil 4.24: $C = (16, 11, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(16, 11, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



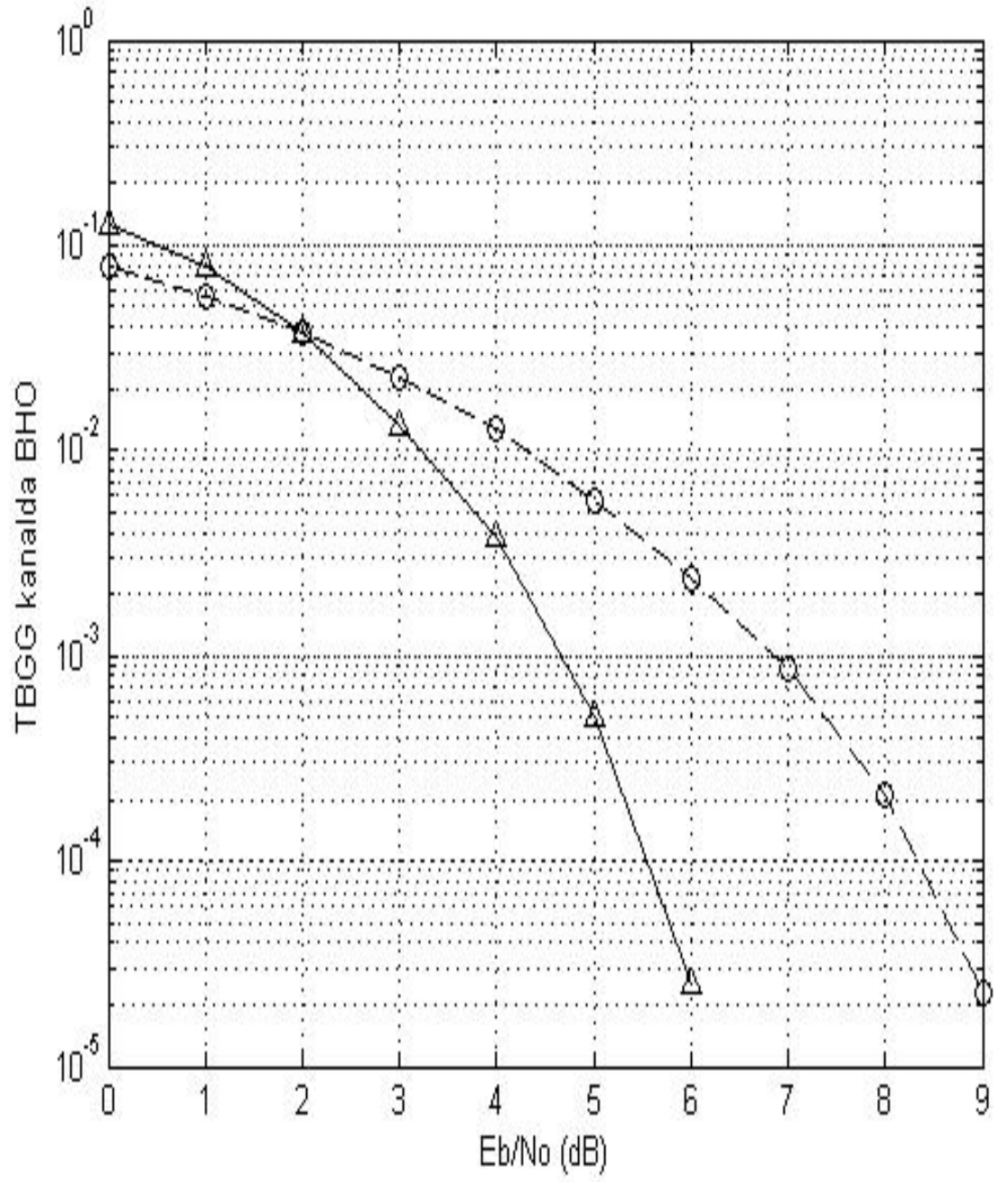
Şekil 4.25: $C = (18, 12, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(18, 12, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



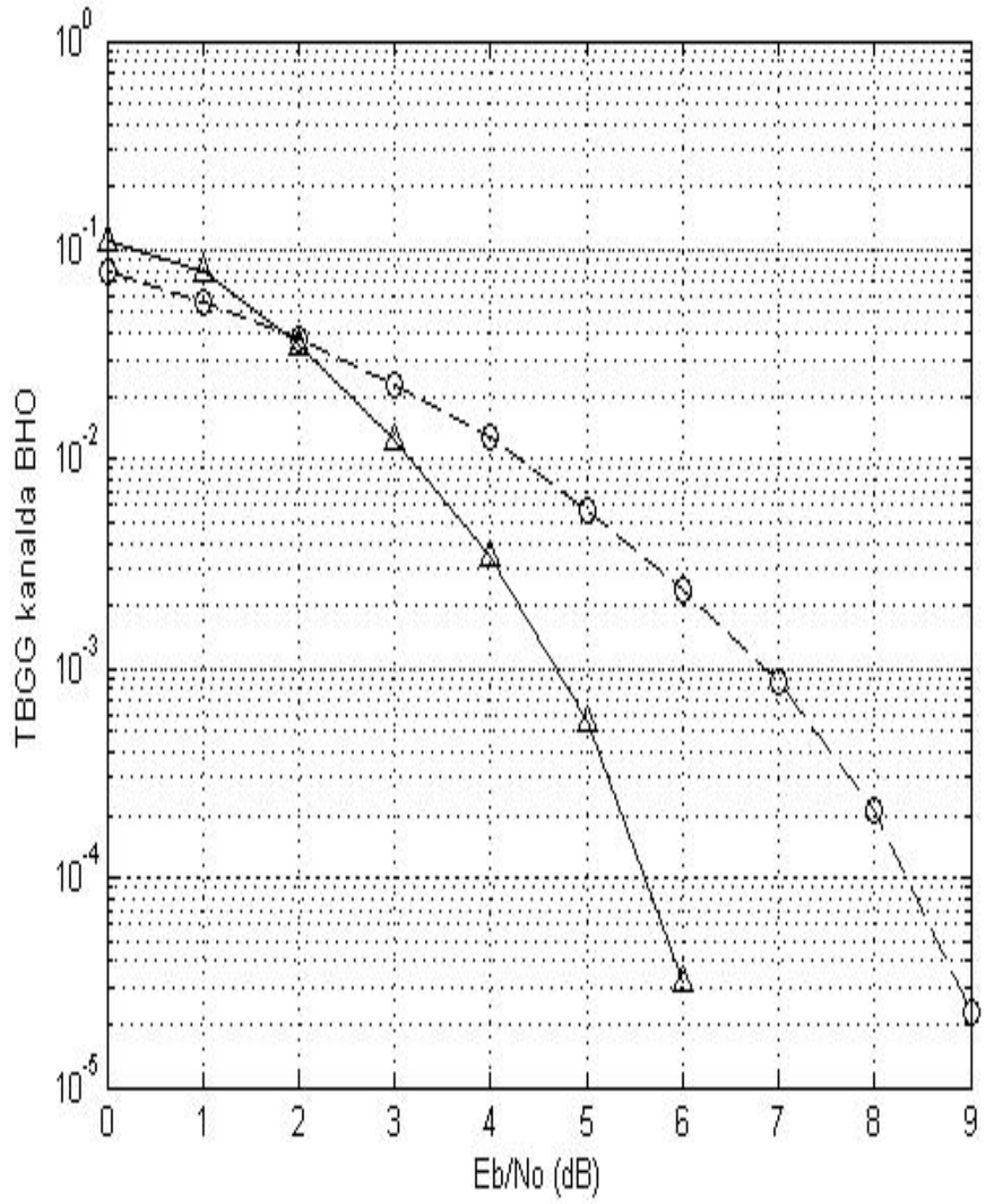
Şekil 4.26: $C = (20, 14, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(20, 14, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



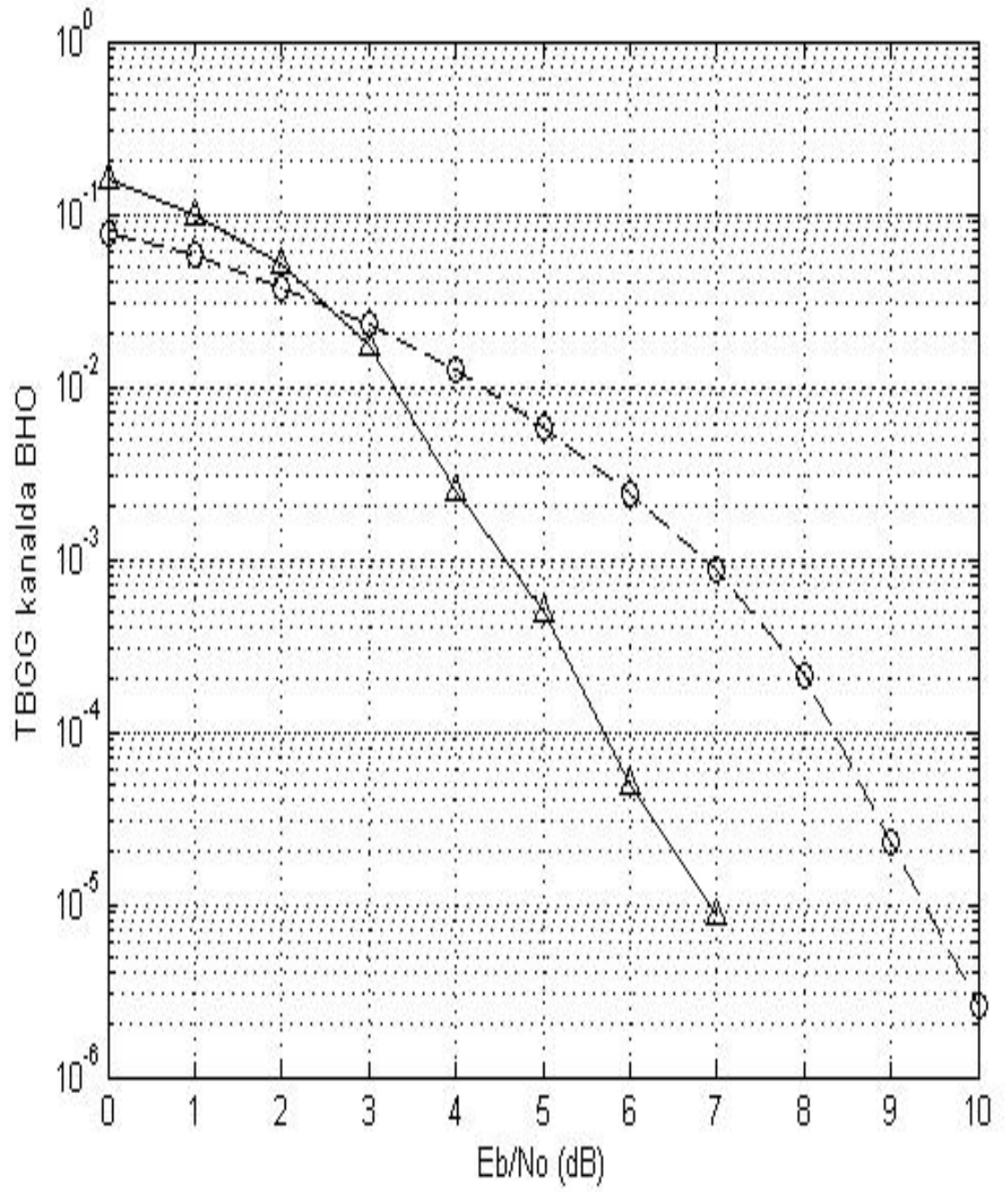
Şekil 4.27: $C = (22, 16, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(22, 16, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



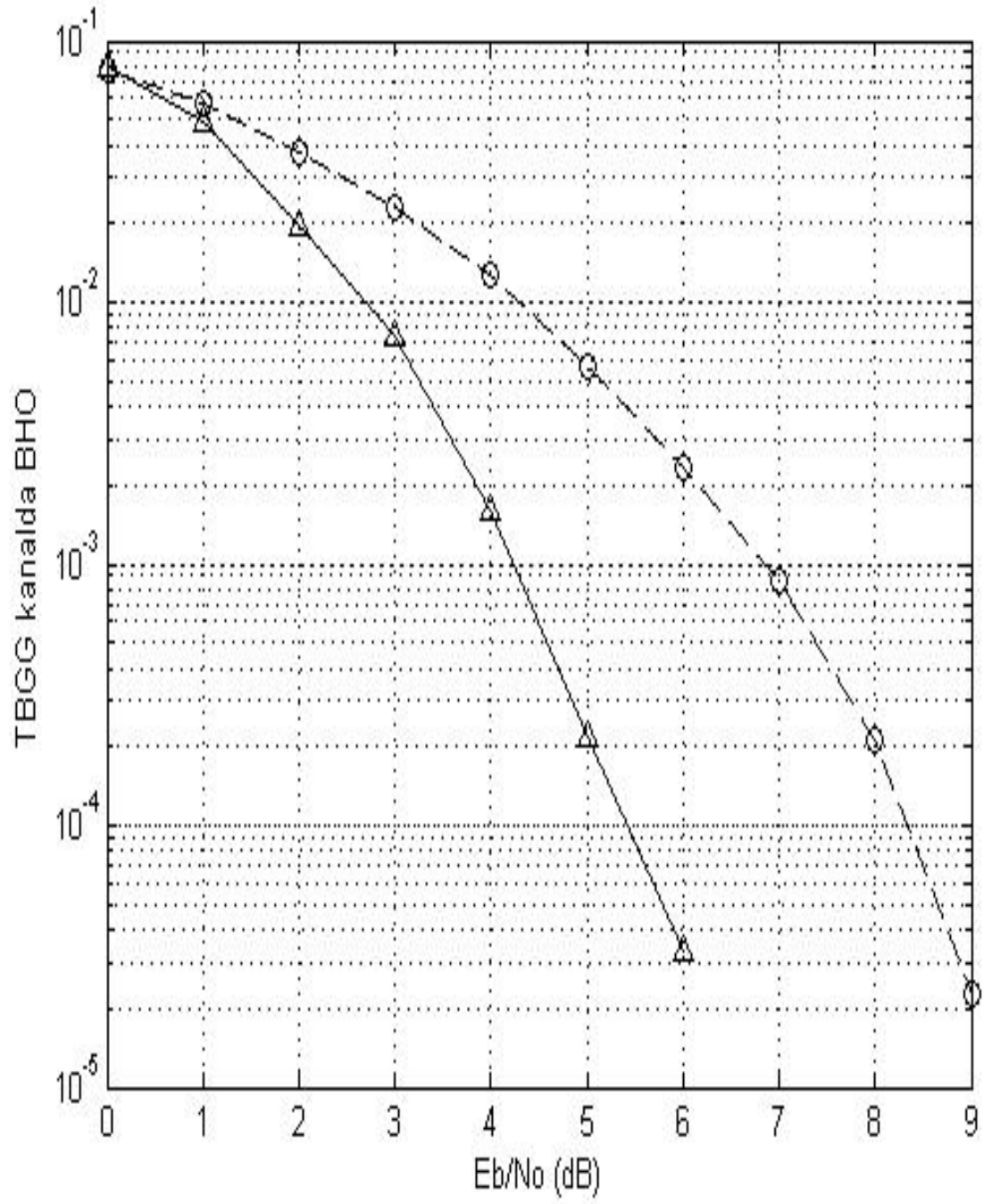
Şekil 4.28: $C = (24, 18, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(24, 18, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



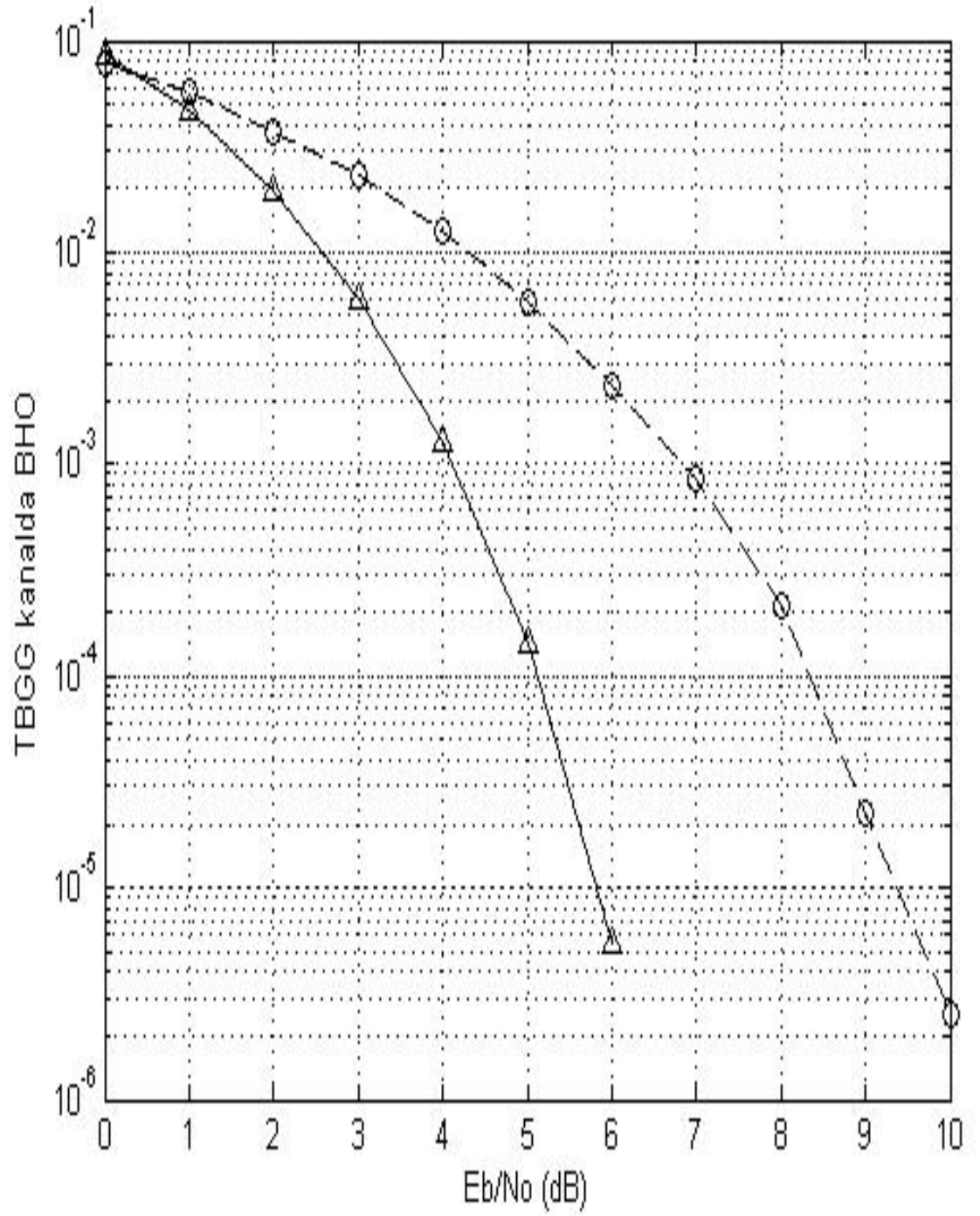
Şekil 4.29: $C = (28, 22, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(28, 22, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



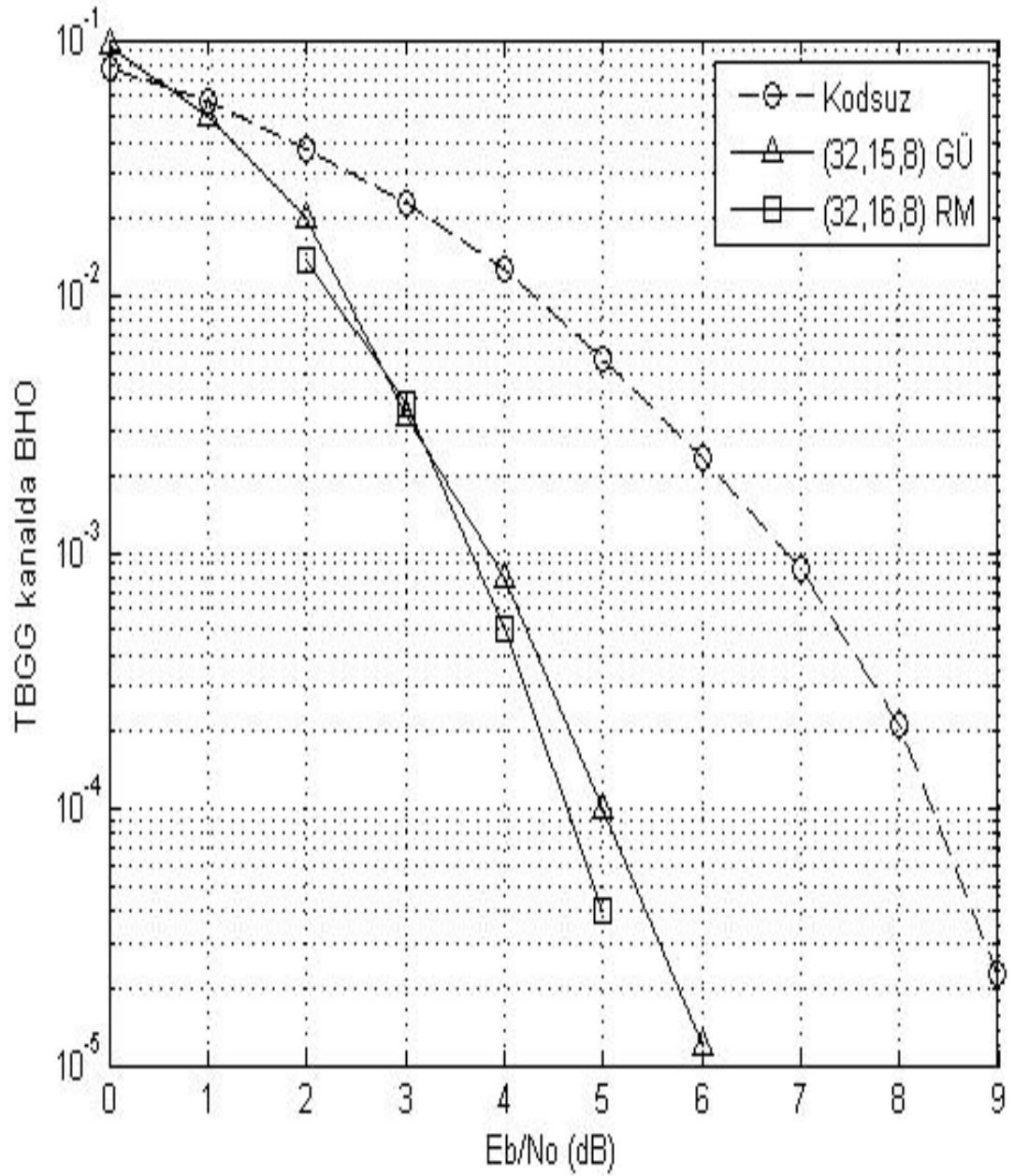
Şekil 4.30: $C = (16, 5, 8)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(16, 5, 8)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



Şekil 4.31: $C = (24, 9, 8)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BHO.

Şekilde üçgen işaretli düz çizgili eğri $(24, 9, 8)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.

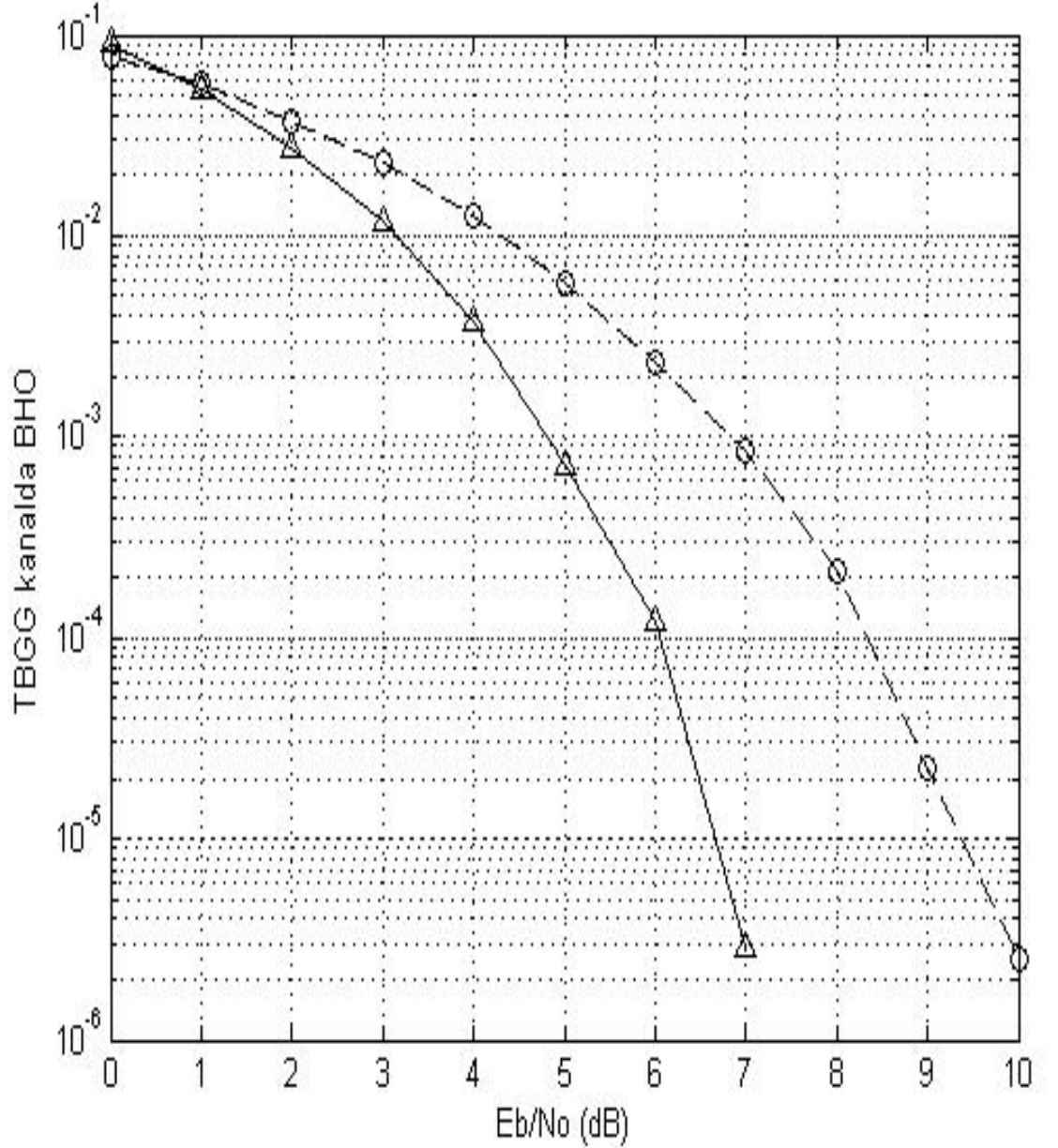


Şekil 4.32: $C = (32, 15, 8)$ GÜ kodunun, $(32, 16, 8)$ RM [2] kodunun ve kodlanmamış bitlerin TBGG kanalda BHO. Şekilde üçgen işaretli düz çizgili eğri $(32, 15, 8)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.

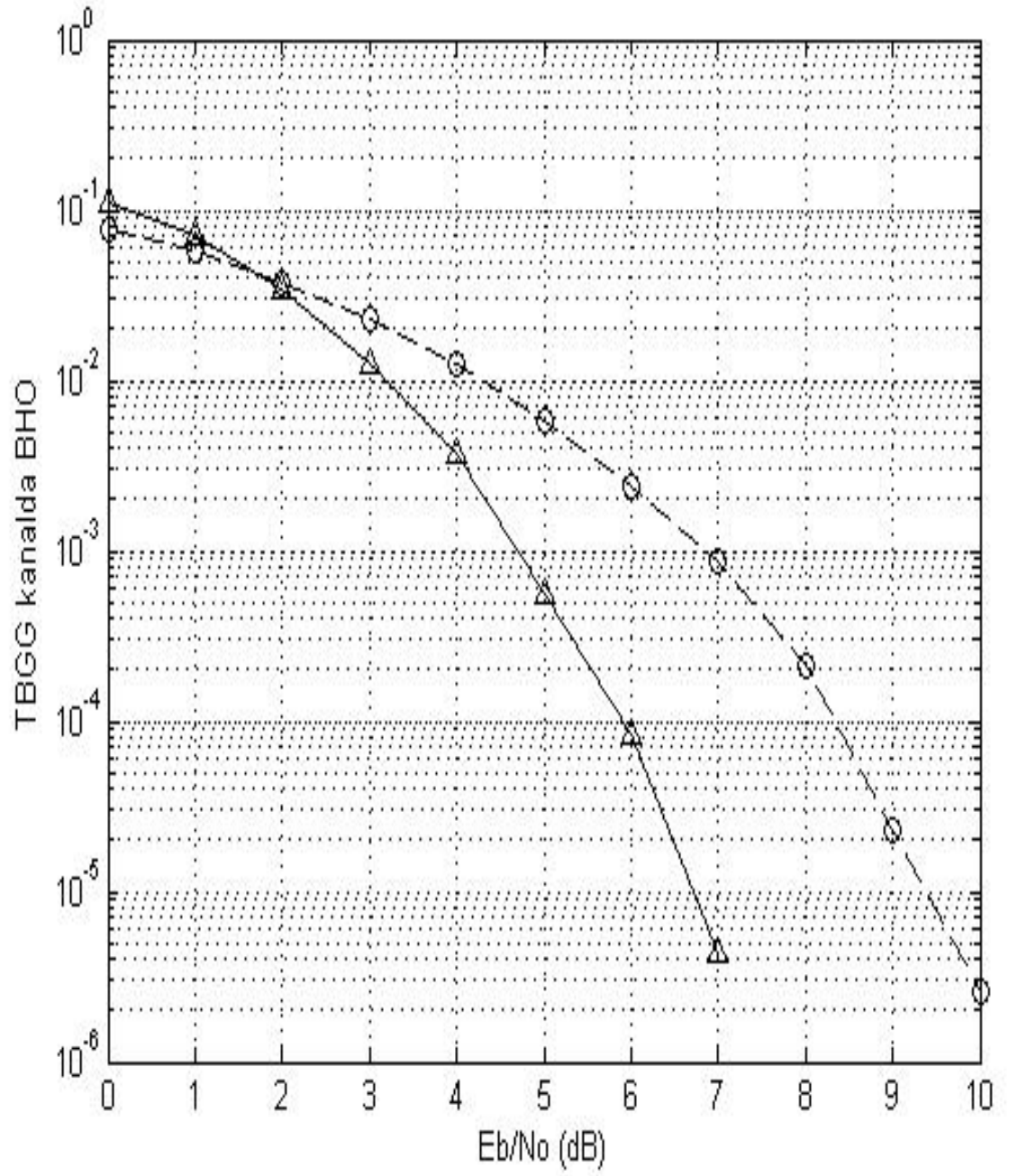
Şekildende görüldüğü $(32, 15, 8)$ GÜ kodunun hata başarımı ile $(32, 16, 8)$ RM kodunun hata başarımı birbirine çok yakındır.

4.5.2. BCJR Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımları

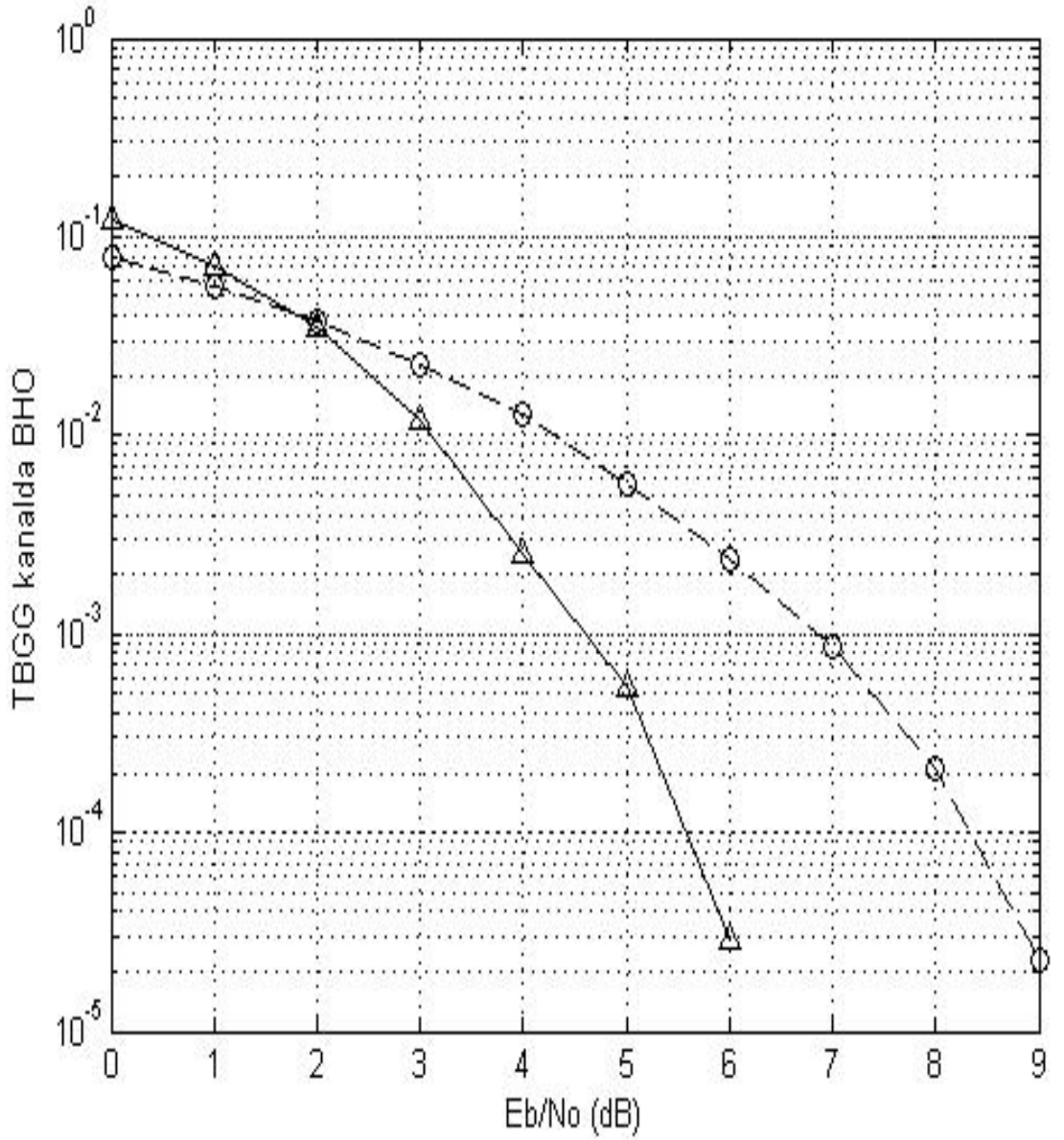
Bazı GÜ kodları için, BCJR algoritması kullanılarak, en büyük olabilirlikli yumuşak kararlı kodçözümüyle elde edilen, TBGG kanal ortamında ki BHO başarımları aşağıdaki şekillerde sunulacaktır.



Şekil 4.33: $C = (12, 7, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BCJR algoritması yardımıyla elde edilen BHO. Şekilde üçgen işaretli düz çizgili eğri $(12, 7, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.



Şekil 4.34: $C = (16, 11, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BCJR algoritması yardımıyla elde edilen BHO. Şekilde üçgen işaretli düz çizgili eğri $(16, 11, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.

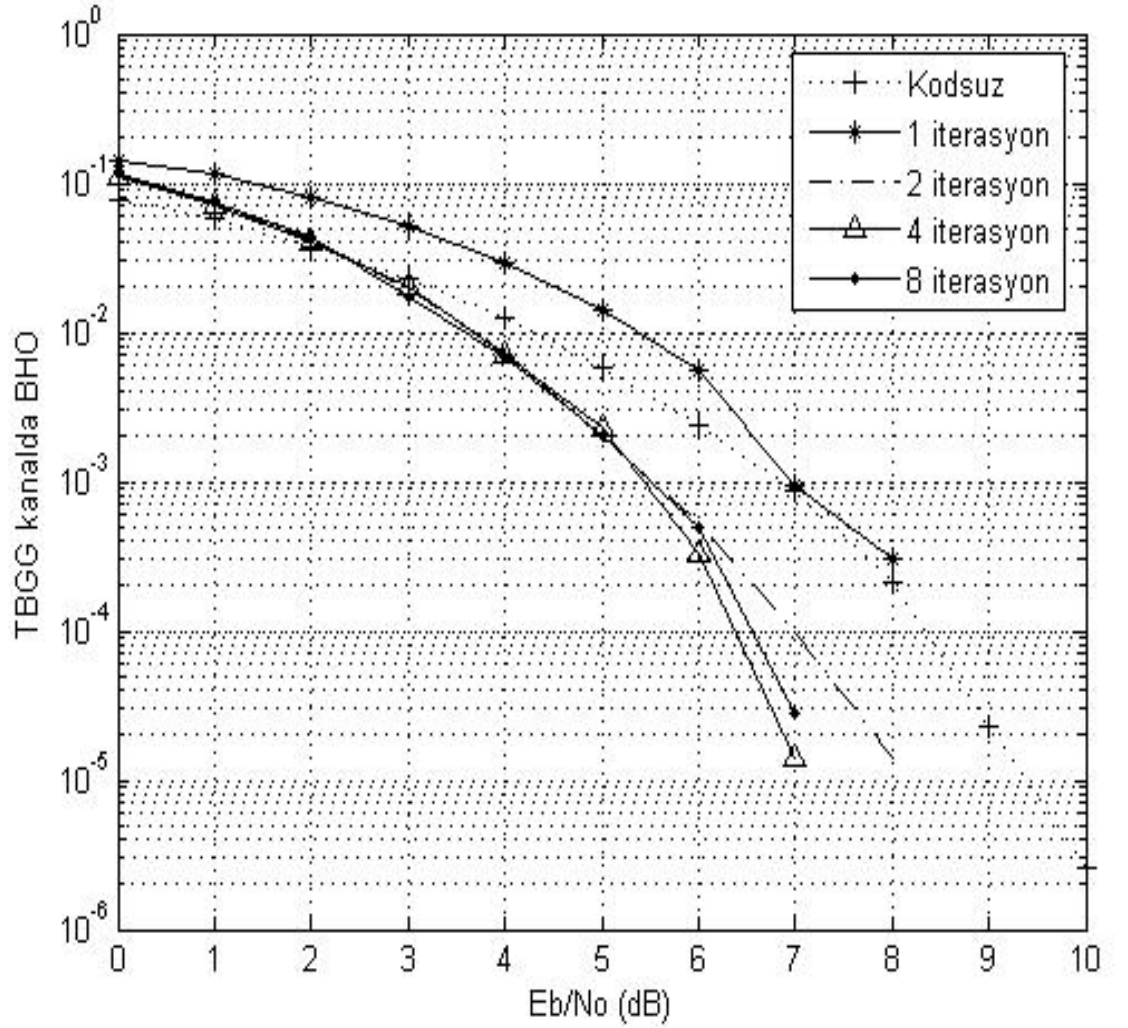


Şekil 4.35: $C = (20, 14, 4)$ GÜ kodunun ve kodlanmamış bitlerin TBGG kanalda BCJR algoritması yardımıyla elde edilen BHO. Şekilde üçgen işaretli düz çizgili eğri $(16, 11, 4)$ kodunun ve yuvarlak işaretli kesikli çizgili eğri kodlanmamış bitlerin BHO olarak temsil eder.

Sunulan eğrilerden de görüleceği üzere, aynı boyutlu iki kodun kodçözümünde, Viterbi algoritması yerine BCJR algoritması kullanıldığında hata başarımları pek değişmemekle beraber çok az bir iyileşme gözlenmektedir. Bununla beraber BCJR kodçözme algoritması Viterbi algoritmasına göre çok daha karmaşık bir algoritma olduğundan bilgisayarın kodçözme zamanı önemli ölçüde artmıştır.

4.5.3. İteratif Kodçözümüyle Geometrik Ürün Kodunun Hata Başarımı

Bir (12, 7, 4) GÜ kodu için, iteratif kodçözme algoritması kullanılarak, en büyük olabilirlikli yumuşak kararlı kodçözümüyle elde edilen, TBGG kanal ortamında ki BHO başarımı, değişik iterasyon sayıları için elde edilmiş ve aşağıdaki şekilde sunulmuştur.

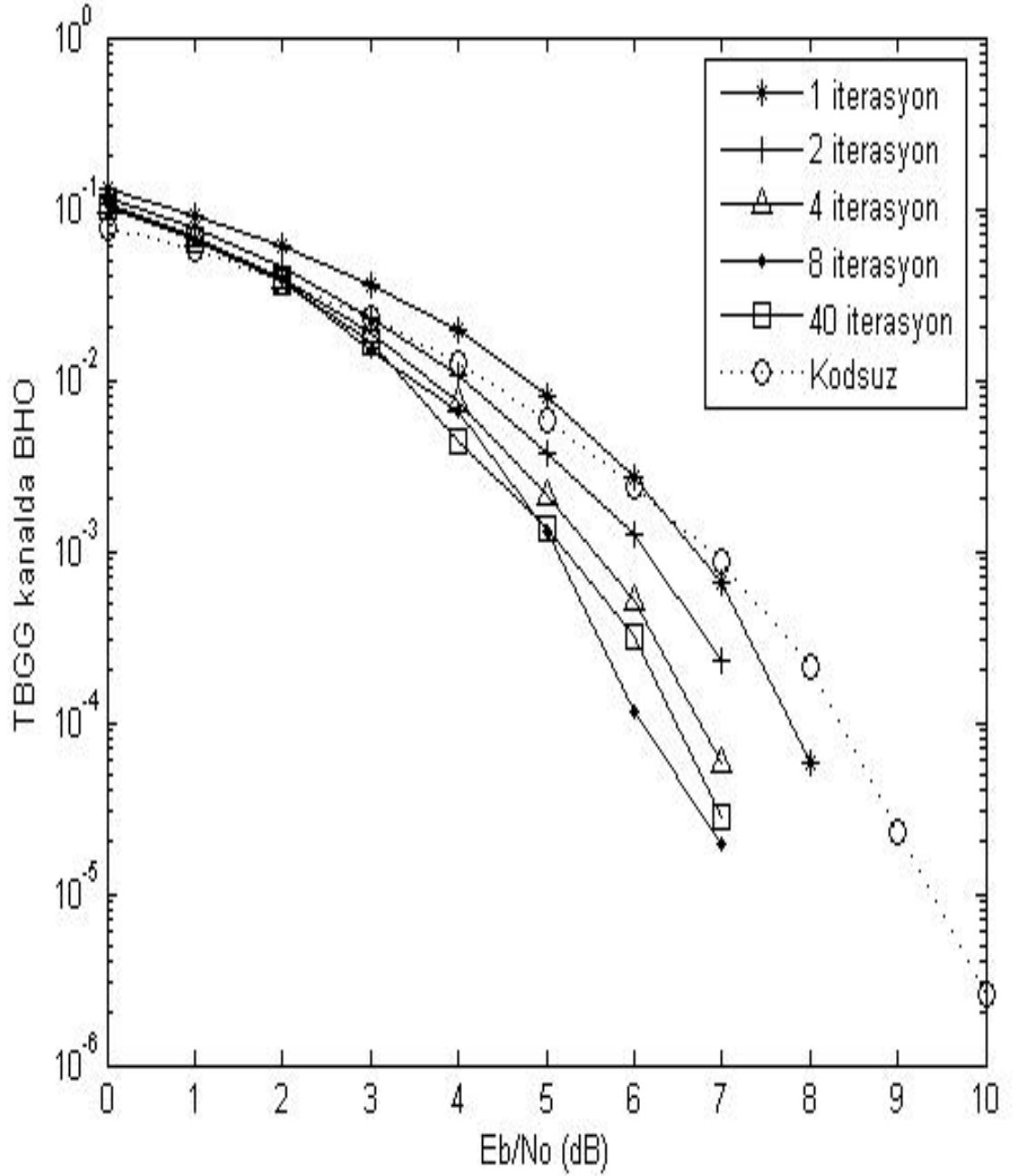


Şekil 4.36: $C = (12, 7, 4)$ GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda iteratif kodçözme algoritması yardımıyla elde edilen BHO eğrileri.

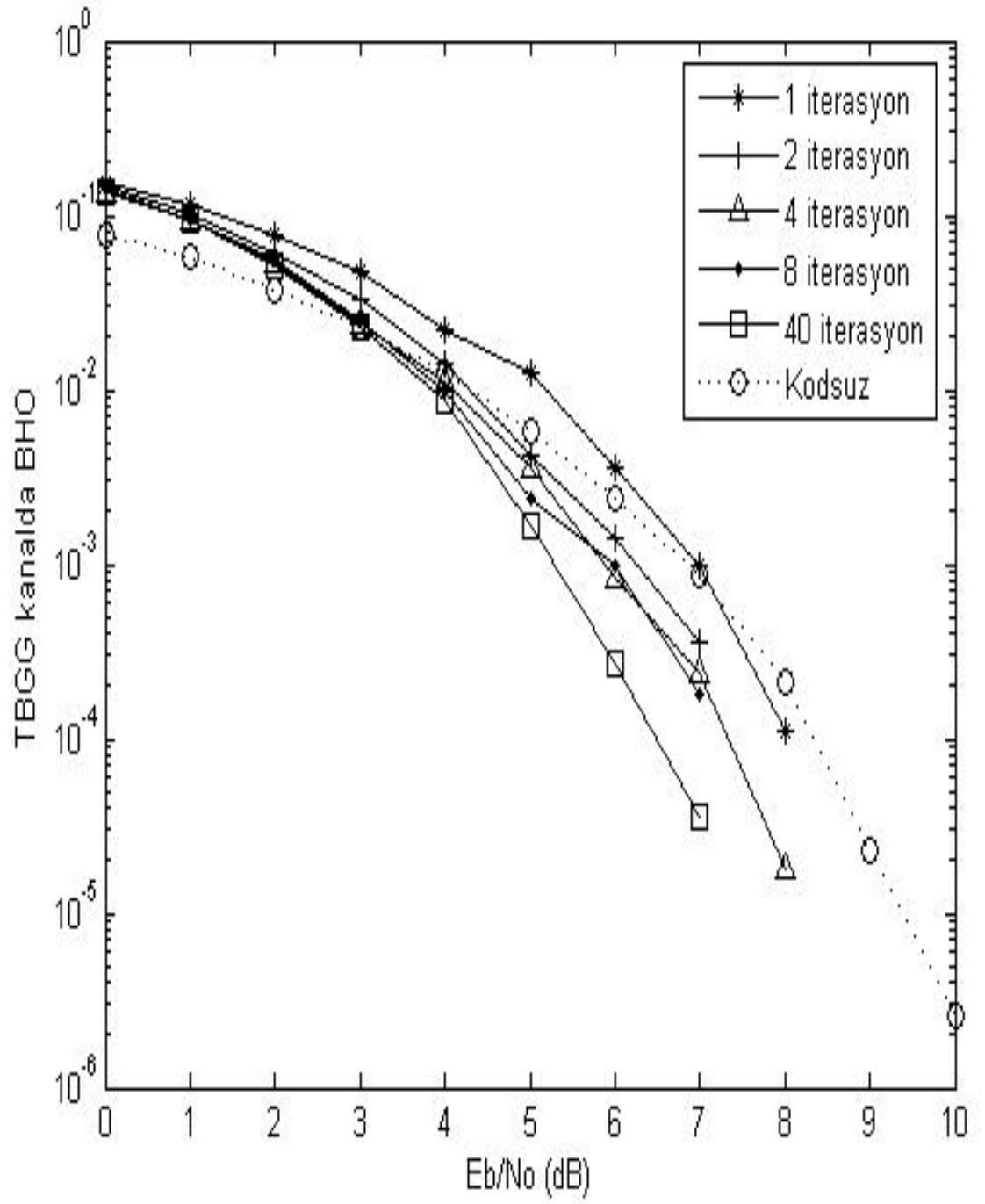
Yukarıda ki şekilden de görüldüğü gibi iteratif yöntemle kodçözümü gerçekleştirilen (12, 7, 4) GÜ kodu en iyi başarımlı eğrisine hızlı bir şekilde 4 iterasyon sonucu yakınsar. Daha büyük uzunluklu kodlar için bu kodçözme algoritmasının kullanılması aşırı karmaşıklığından dolayı kullanılmamıştır.

4.5.4. Toplam-Ürün Algoritması Kodçözümüyle Geometrik Ürün Kodlarının Hata Başarımı

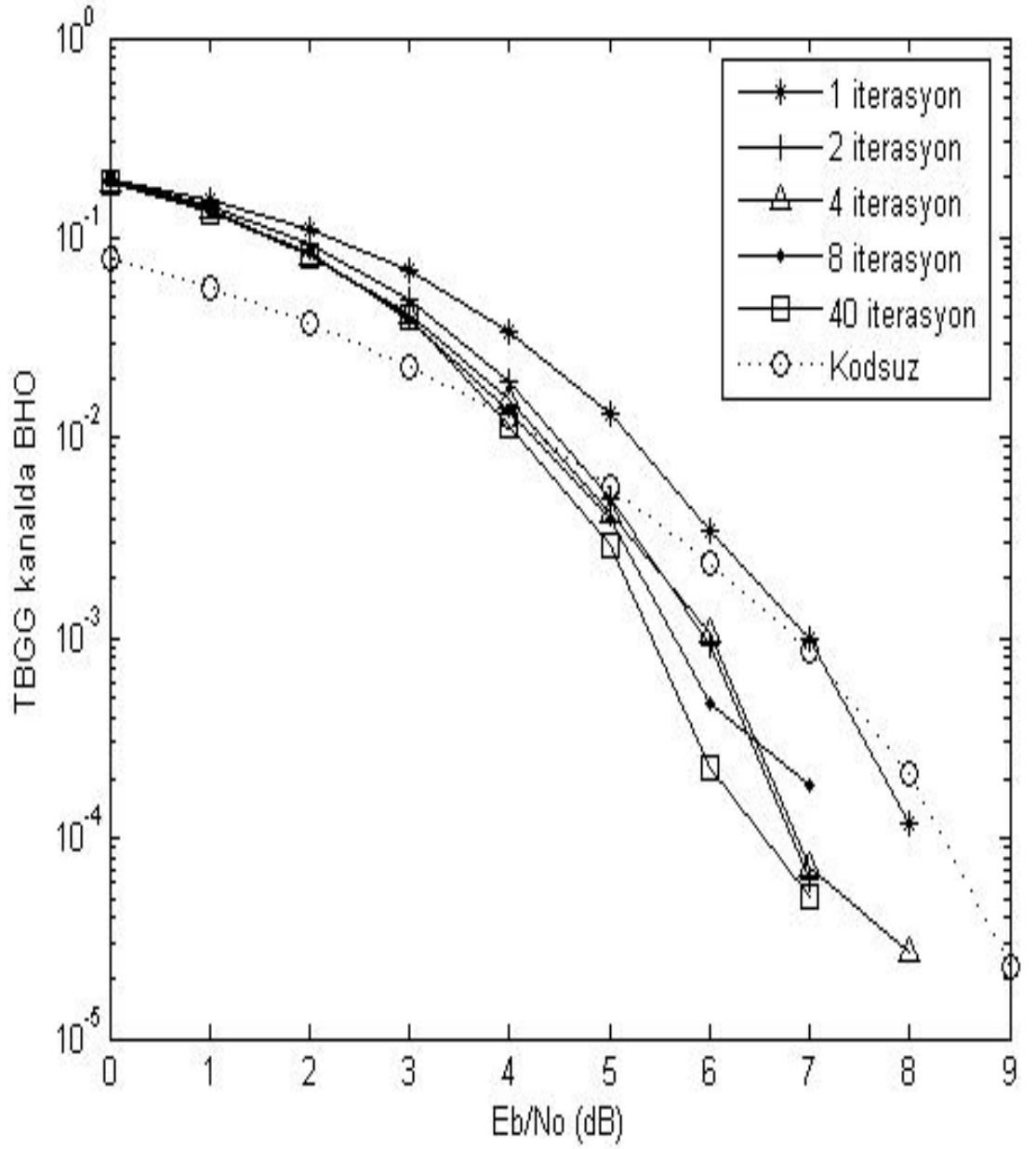
Önerilen bazı GÜ kodları için, toplam-ürün algoritması kullanılarak, en büyük olabilirlikli yumuşak kararlı kodçözümüyle elde edilen, TBGG kanal ortamında elde edilen BHO başarımları aşağıdaki şekillerde gösterilmiştir.



Şekil 4.37: $C = (12, 7, 4)$ GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO eğrileri.

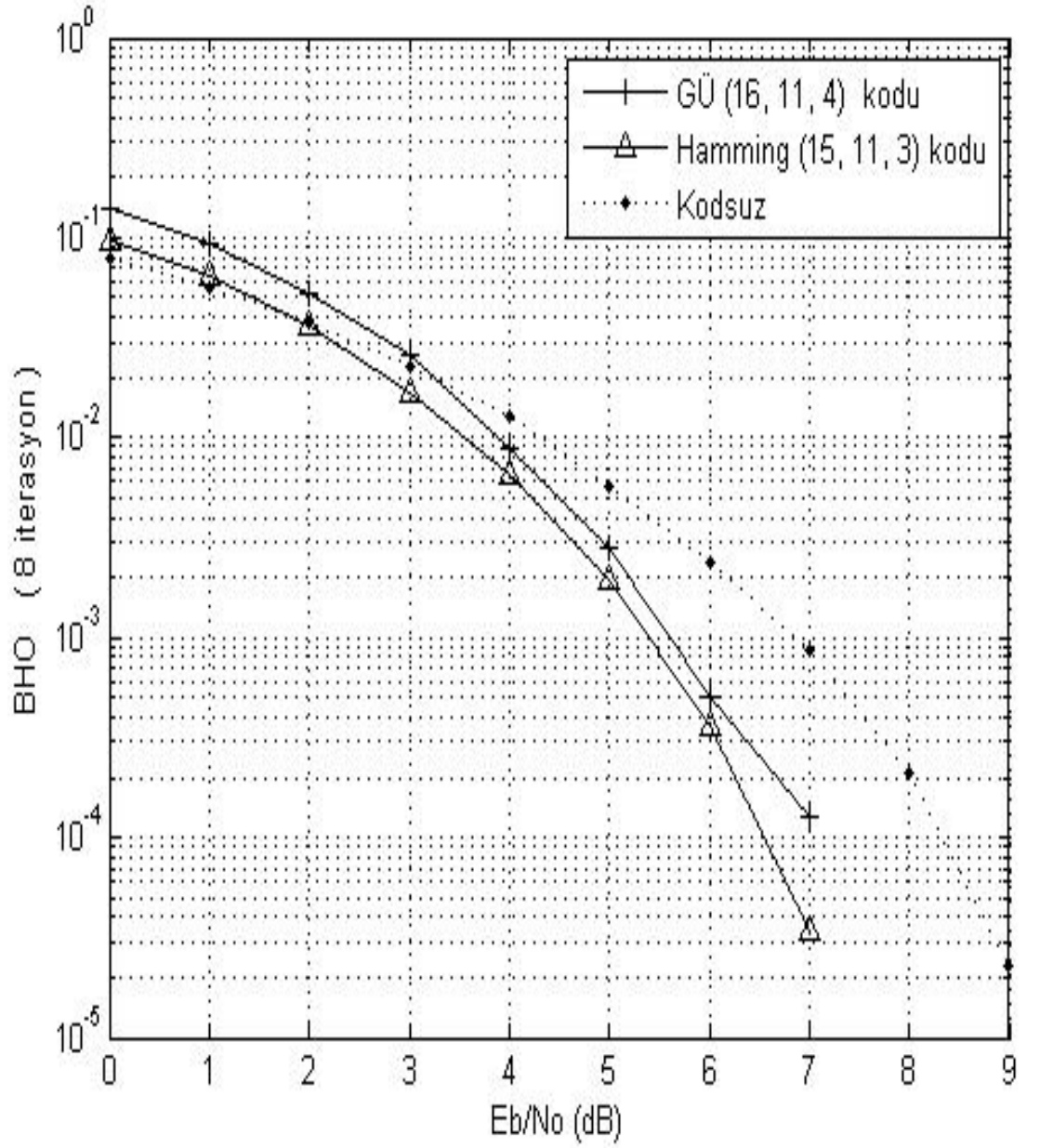


Şekil 4.38: $C = (16, 11, 4)$ GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO eğrileri.



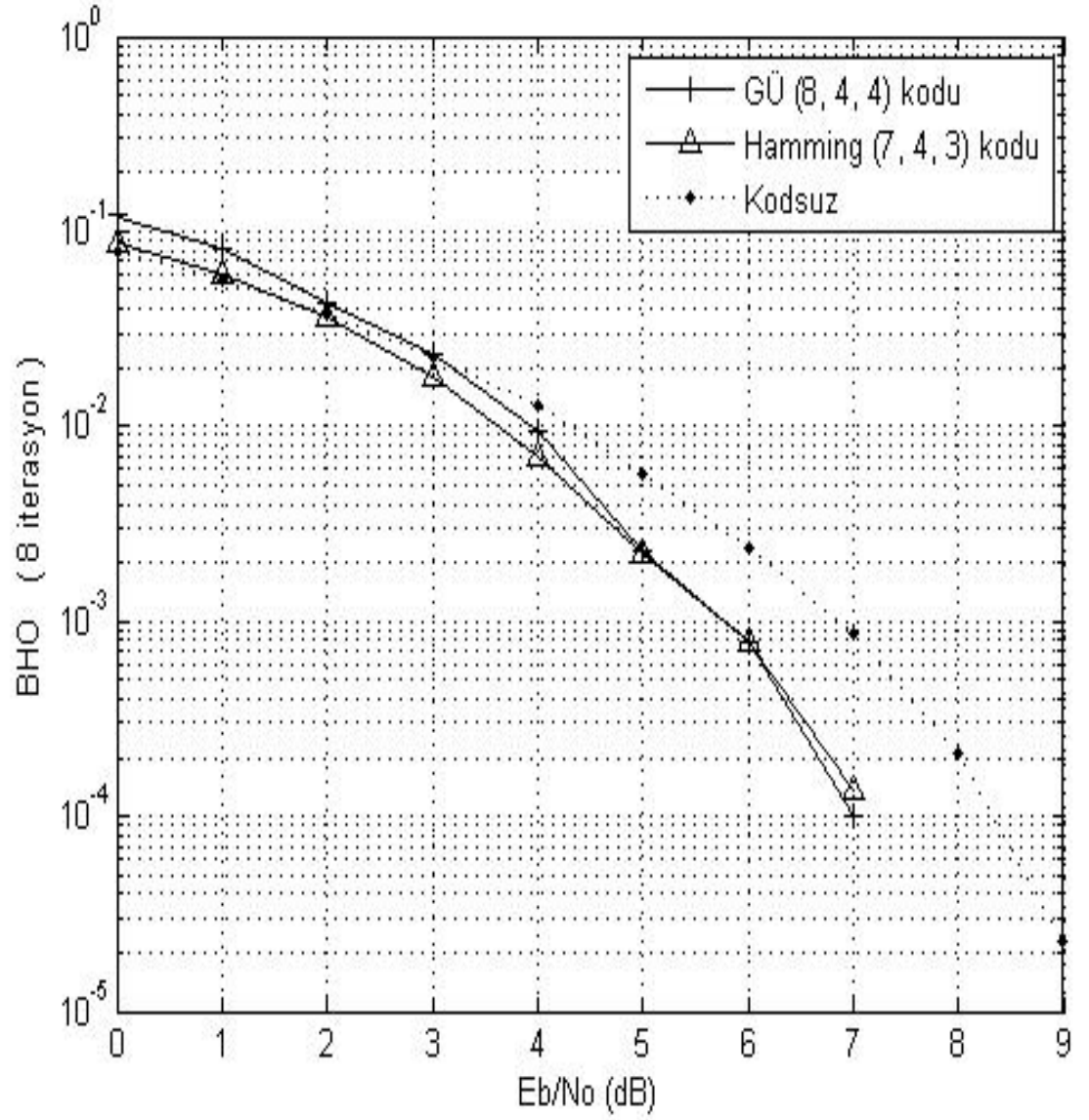
Şekil 4.39: $C = (28, 22, 4)$ GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO eğrileri.

Sunulan eğrilerden de görüleceği üzere, iterasyonların başarıma etkisi 4 iterasyona kadar yüksek, 4'ten sonra çok az bir etkisi vardır. Hata başarımları Viterbi veya BCJR algoritmalarıyla elde edilen sonuçlara oldukça yakınsamaktadır fakat onlar kadar iyi değildir.



Şekil 4.40: (16, 11, 4) GÜ ve (15, 11, 3) Hamming kodlarının TBGG kanalda toplam-ürün algoritması ile 8 iterasyonda elde edilen BHO eğrileri. (Ayrıca kodlanmamış durum da kodsuz olarak gösterilmiştir)

Sunulan eğrilerden de görüleceği üzere (16, 11, 4) GÜ ve (15, 11, 3) Hamming kodları birbirine yakın hata başarımlarına sahiptir.



Şekil 4.41: (8, 4, 4) GÜ ve (7, 4, 3) Hamming kodlarının TBGG kanalda toplam-ürün algoritması ile 8 iterasyonda elde edilen BHO eğrileri. (Ayrıca kodlanmamış durum da kodsuz olarak gösterilmiştir)

Sunulan eğrilerden de görüleceği üzere (16, 11, 4) GÜ ve (15, 11, 3) Hamming kodları birbirine yakın hata başarımlarına sahiptir.

4.6. BASAMAK BLOK KODLARI

Önceki kısımlarda anlatılan GÜ kodları, genel bir ürün kodu üreteç matrisi üzerine, GÜ kod inşasının belirlediği kurallara göre oluşturulan üreteç matrisi ile elde ediliyordu. Bu kısımda, genel bir ürün kodu üreteç matrisi kullanmadan tamamen sıfırdan üreteç matrisi inşa edilerek elde edilen ve *Basamak Kodları (BB)* olarak adlandırılan yeni kodlar sunulacaktır. Üreteç matrisi inşa edilirken yerleştirilen eleman matrisleri basamaklı bir yapı özelliği gösterdiği için bu üreteç matrisleri ile elde edilen kodlara Basamak Blok kodları adı verilmiştir. Genel olarak BB kodlarının GÜ kodlarından farkı, GÜ kodu üreteç matrisi inşasında kullanılan genel bir ürün kodu üreteç matrisinin, G_y , BB kodları üreteç matrisi inşasında kullanılmaması ve bunun yerine BB kodlarına has bir yapının koyulmasıdır. Geri kalan inşa özellikleri ve elde edilen kodların özellikleri GÜ kodları ile aynıdır. BB kodları üreteç matrislerinin GÜ kodları üreteç matrislerinden avantajlı olan kısmı, G_y genel ürün kodu üreteç matrisi yerine koyulan matrisin tıpkı kendisine eklenen eleman matrisleri gibi basamaklı yapıda olmasıdır. Kaydırmalı kaydedicilerle her matris satırı belli bir sayıda kaydırılarak bir sonraki satırı elde etme olanağına sahip olduğundan, kodlayıcı tasarımında önemli bir pratik avantaj sağlar.

4.6.1. Basamak Blok Kodları İnşası

Basamak Blok (BB) kodları yöntemiyle inşa edilen bir blok kodu $C = (n, k, d)$ olarak tanımlansın. Burada n kod uzunluğu, k kod boyutu yani saf bilgi bitleri sayısı, ve d kodun Hamming mesafesidir. C kodunu üreten üreteç matrisi G , üç veya daha fazla basit yapıdaki eleman kodları üreteç matrislerinin BB kod inşası olarak tanımlanan bir yöntemle birleşimi sonucu oluşur. BB kodu üreteç matrisi GÜ kodu üreteç matrisinin tüm özelliklerine sahiptir.

Hamming mesafesi 4 olan bir BB kodunun üreteç matrisi, G , $C_1 = (n_1, k_1, d_1) = (2,1,2)$ ve $C_2 = (n_2, k_2, d_2) = (2,1,1)$ ölçülerindeki basit blok kodların $G_1 = [1 \ 1]$ ve $G_2 = [1 \ 0]$ üreteç matrisleri kullanılarak (4.33) denkleminde göre yerleştirilmesi ile elde edilir. Elde edilen BB kodlarının ölçüleri de GÜ kodları ile aynı olup (4.32) eşitliği ile aşağıda gösterildiği gibidir.

$$C = (n, k, d) = (n, n - \lceil \log_2 n + 1 \rceil, 4) . \quad (4.32)$$

ve üreteç matrisi G ise,

$$G = \left[\begin{array}{ccccccccc} & G_1 G_1 & & & & & & & \\ & G_1 G_1 & & & & & & & \\ & & \cdot & & \cdot & & \cdot & & \\ & & & & & & & G_1 G_1 & \\ & & & & & & & G_1 G_1 & \\ & G_2 G_2 G_2 G_2 & & & & & & & \\ & G_2 G_2 G_2 G_2 & & & & & & & \\ & & \cdot & & \cdot & & \cdot & & \\ & & & & & & & G_2 G_2 G_2 G_2 & \\ G_2 & G_2 & G_2 & G_2 & G_2 & G_2 & & & \\ & & G_2 & G_2 & G_2 & G_2 & & & \\ & & & \cdot & \cdot & \cdot & G_2 & G_2 & G_2 & G_2 \\ G_2 & & G_2 & & G_2 & G_2 & G_2 & & G_2 & \\ & & & G_2 & G_2 & & & G_2 & & \\ & & & & & \cdot & \cdot & \cdot & & \end{array} \right] \quad \begin{matrix} \left. \vphantom{\begin{matrix} G_1 G_1 \\ G_1 G_1 \\ \cdot \\ G_1 G_1 \\ G_1 G_1 \end{matrix}} \right\} E_1 \\ \left. \vphantom{\begin{matrix} G_2 G_2 G_2 G_2 \\ G_2 G_2 G_2 G_2 \\ \cdot \\ G_2 G_2 G_2 G_2 \end{matrix}} \right\} E_2 \\ \left. \vphantom{\begin{matrix} G_2 \\ G_2 \\ G_2 \\ G_2 \\ G_2 \\ G_2 \end{matrix}} \right\} E_3 \\ \left. \vphantom{\begin{matrix} G_2 \\ G_2 \\ G_2 \\ G_2 \\ G_2 \\ G_2 \end{matrix}} \right\} E_4 \end{matrix} \quad (4.33)$$

eşitliği kullanılarak inşa edilir. Daha basit gösterim için matris aşağıdaki gibi de yazılabilir.

$$G = \left[\begin{array}{c} D_1 \\ D_2 \\ \vdots \\ D_t \end{array} \right] \begin{array}{l} \} E_1 \\ \} E_2 \\ \vdots \\ \} E_t \end{array} \quad (4.34)$$

Burada E_i ($i=1,2,\dots,t$) eleman matrislerinin geometrik yerleştirilmelerine göre ayrılan grup etiketleri olup, bu gruplar D_i matrisleri ile temsil edilmiştir. Burada t grupların

sayısını temsil etmektedir. (4.33) denklemiyle gösterilen BB kodlarının üreteç matrislerinin yerleştirilmesi aşağıda açıklandığı gibi yapılmıştır:

BB kodlarının üreteç matrisinin, G_1 satırları yerleştirilme biçimlerine göre $\{E_1, E_2, \dots\}$ olarak gruplandırılır. E_1 grubu için eleman matrislerinin satırlara yerleştirilmesinde 2 adet G_1 eleman matrisi ardışık olarak hiç aralık bırakmadan ilk sütundan başlayarak yerleştirilir. İlk satıra yerleştirilen kolonlar 1 tane G_1 matris ölçeği kadar sağa kaydırılarak ikinci satıra yerleştirilir. Bu çevrimsel işlem E_1 grubunun tüm satırları için aynen devam ettirilir ve G_1 matrisi en son sütuna ulaşınca işlem durdurulur. BB kodlarının GÜ kodlarından farklı olduğu inşa kısmı bu kısımdır. Çevrimsel özelliğinden dolayı kaydırmalı kaydedicilerle kodlamaya imkan sağladığından herhangi bir matris çarpma işlemi gerektirmediğinden GÜ kodunun inşasından daha avantajlıdır. Kalan kısımların inşası GÜ kodundakine benzer olmakla beraber burada BB kodları için detaylarıyla açıklanacaktır.

E_2 grubu için eleman matrislerinin satırlara yerleştirilmesinde E_2 grubundaki tüm ekleme satırlarında her bir satıra 4 adet G_2 matrisleri art arda hiç boşluk bırakılmadan yerleştirilir. E_2 grubunun ilk satırında yerleştirme ilk kolondan başlar. E_2 grubunun ikinci satırında ise art arda yerleştirilen 4 adet G_2 matrisi, 2 G_2 matrisi ölçeğinde sağa kaydırılır. Böylece bu iki satırda 2'şer adet G_2 matrisleri üst üste gelmiş olur. E_2 grubu için bu işlem 4 adet G_2 matrisini kaydirdıkça yerleştirilecek yer olduğu müddetçe devam eder. Eğer 4 adet G_2 matrisini yerleştirecek kolon kalmamışsa E_2 grubu için yerleştirme işlemi sona erer. E_2 grubu için yerleştirme işlemi tamamlandıktan sonra E_3 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda E_3 grubu için yerleştirilecek 4 adet G_2 matrisi arasında birer aralık bırakılır. Buradaki 'aralıktan' kasıt bir G_2 ölçeği kadarlık yerdir. E_3 grubunun ilk satırında da yerleştirme ilk kolondan başlar. E_3 grubunun ikinci satırında ise art arda yerleştirilen 4 adet birer aralıklı G_2 matrisi, 4 G_2 matrisi ölçeğinde sağa kaydırılır. Böylece aynı grupta bulunan ardışık iki satır arasındaki 2'şer adet G_2 matrisleri üst üste gelmiş olur. Bu işlemde bir öncekine benzer biçimde 4 adet G_2 matrisini yerleştirecek kolon kalmayana kadar devam eder. E_3 grubu için yerleştirme işlemi tamamlandıktan sonra, eğer yer varsa, E_4 grubu için yerleştirme işlemine başlanır. Bu durumda ise E_4 grubu için yerleştirilecek 4 adet G_2 matrisi arasında üçer aralık bırakılır. Yerleştirme işlemi öncekiler gibi yapılmakla

beraber sağa kaydırma 8 aralıklı yapılır. Yerleştirme için yer varsa, bu işlemler aynı şekilde diğer gruplar için de devam eder. Tüm yerleştirme işlemi 4 adet G_2 matrisini yerleştirecek ekleme satırı kalmayınca sona erer. Genel bir kural olarak, E_1 grubu ayrı tutulmak üzere, sağa kaydırma işlemi E_2 grubu için 2, E_3 grubu için 4, E_4 grubu için 8 daha sonraki gruplar için 8, 16, 32,..., 2^n 'nin kuvvetleri şeklinde devam eder. 4 adet G_2 matrisi arasında bırakılan aralıklar ise E_2 grubu için 0, E_3 grubu için 1, daha sonraki gruplar için 3, 7, 15,... şeklinde kaydırma miktarının yarısından 1 eksik olarak devam eder. Buraya kadar açıklanan yerleştirme işlemi C_1 ve C_2 kodlarının üreteç matrisleri G_1 ve G_2 kullanılarak yapılmıştır. Ancak, özellikle büyük Hamming mesafeli kodların inşasında karşılaşılabileceği gibi bazen G_2 matrisine ek olarak bir başka benzer ölçülerde eleman matrisi, mesela G_3 , eklenmesi gerekebilir. Böyle bir durumda yerleştirme işlemi aynen G_2 matrisine yapıldığı gibi yapılır ve yerleştirmeye G_2 matrisinden sonra devam edilir.

Şimdiye kadar açıklanan kurallar uygun eleman matrisleri seçildiği varsayımı altında eleman matrislerinin nasıl yerleştirildiği ile ilgiliydi. Eleman matrislerinin nasıl seçileceği ile alakalı genel kurallar koyulmamakla beraber, bunların nasıl seçileceğini istenilen kodun uzunluğu belirler. İstenilen uzunluklu koda göre C_1 ve C_2 kodlarının üreteç matrisleri birleştirilerek genel ürün matrisi yapısı elde edilir. G_2 matrisi seçilirken dikkat edilmesi gereken husus ise 4 adet G_2 matrisinde bulunan mantıksal 1'lerin toplamı, kodun arzu edilen minimum Hamming mesafesine eşit olmasıdır. Bununla ilgili olarak daha sonraki kısımlarda açıklayıcı örnekler verilecektir. BB kodlarının, bir kod ailesi olan minimum Hamming mesafeli kodların Hamming mesafelerinin 4 olduğunu ispatlamak için Teorem 4.4 önerilip ispatlanacaktır. Fakat, ondan önce ispatı yapabilmek için geliştirilen *yardımcı teoremler (Lemma)* ve bazı temel teoremler sunulacaktır.

Teoremlerin anlaşılmasında kolaylık sağlaması açısından aşağıdaki özdeşliği hatırlamak faydalı olacaktır: Eğer u ve v ikili vektörler ise, $u \bullet v$ işlemi u ve v 'nin her ikisinde de aynı yerde bulunan 1'lerin sayısını gösterir. Bundan yola çıkarak, çok bilinen ve çok faydalı bir eşitlik olan, $wt(u + v) = wt(u) + wt(v) - 2(u \bullet v)$ eşitliği elde edilir. Burada $wt(c)$ gösterimi, c 'nin Hamming ağırlığını gösterir.

Teorem 4.4: Eğer, bir ikili (n,k) kodu C için, G üreteç matrisinin satırları çift ağırlık değerine sahipse ve satırları birbirine dikse, C *kendi-diktir* [83].

Teorem 4.5: Eğer, bir ikili (n,k) kodu C için, G üreteç matrisinin satırları 4 ile bölünebilir bir ağırlığa sahipse ve birbirine dikse, o zaman C kendi-diktir ve C kodundaki tüm ağırlıklar 4 ile bölünebilir [83].

Yardımcı Teorem 4.1: Denklem (4.34) ile oluşturulan bir grup üreteç matrisi, D_i ile üretilen bir kod kelimesinin Hamming ağırlığı en az 4 tür.

İspat: Denklem (4.34) ile oluşturulan bir grup üreteç matrisinin, D_i satırlarının Hamming ağırlığı 4'tür. Ayrıca bu satırlar birbirlerine diktir. Her bir grup içinde diklik özelliği sağlanmaktadır. Bu durum Teorem 4.4 de aranan şartı sağlar, dolayısıyla bir D_i grup üreteç matrisi ile üretilen bir C_i kodu kendi-diktir. D_i grup üreteç matrislerindeki satırların ağırlıkları 4 olduğundan, yani 4 ile bölünebildiğinden, ve C_i kendi-dik olduğundan, Teorem 4.5'ten yola çıkılırsa, o zaman D_i ile üretilen bütün C_i kodlarının ağırlıkları 4 ile bölünebilir. D_i içindeki her bir satırın ağırlığı 4 olduğundan, yani 4 ile bölünebilir olduğundan, C_i kodunun minimum Hamming ağırlığı yani minimum Hamming mesafesi 4'tür. Yardımcı Teorem 4.1 şunu ispatlar: (4.33) veya (4.34) denklemlerindeki her bir grup minimum Hamming mesafesi 4 olan kodkelimeleri üretir.

Yardımcı Teorem 4.2: Eğer u ve v , (4.33) denklemiyle inşa edilen üreteç matrisinin, G , ayrık ikili satır vektörleri ise, $u \bullet v$ işleminin sonucu en fazla 2'dir.

İspat: (4.33) denklemiyle oluşturulan G üreteç matrisinden de gözlenebileceği gibi matrisin satırlarının yerleştirilme kuralı doğal olarak kendiliğinden bunu sağlar; çünkü G içindeki üst üste gelen eleman matrislerinin sayısı en fazla 2'dir.

Teorem 4.6: (4.33) denklemiyle oluşturulan bir G üreteç matrisi ile üretilen bir C kodunun minimum Hamming mesafesi 4'tür.

İspat: Eğer u ve v , (4.33) denklemiyle inşa edilen üreteç matrisinin, G , ayrık ikili satır vektörleri ise $wt(u + v) = wt(u) + wt(v) - 2(u \bullet v)$ özdeşliğinden yararlanılarak ispata

gidilebilir. Yardımcı Teorem 4.1 kullanılırsa, $wt(u) \geq 4$ ve $wt(v) \geq 4$ olur. Yardımcı Teorem 4.2 kullanılırsa, $u \cdot v \leq 2$ olur. Bu durumda en kötü senaryoya göre hesap yapıldığında minimum Hamming mesafesi $wt(u + v) = 4 + 4 - 2 \cdot 2 = 4$ bulunur. Dolayısıyla C kodunun minimum Hamming mesafesi 4'tür.

Bu teoremin sağlamasını yapmak için ayrıca bir bilgisayar araştırma programı yazılmış ve BB kodu olan C kodunun (4.33) denklemindeki gibi inşası sırasında minimum Hamming mesafesinin 4 olduğu teyit edilmiştir. Denklem (4.32) ile ölçülerin tek bir parametreye (n) göre formülleştirilmesi 4 Hamming mesafeli BB kodları için çok önemli bir özellik olup istenilen ölçüleri kontrol etme imkanı sağlar. Örneğin, minimum Hamming mesafesi 4 olan ve uzunluğu $n = 1200178$ olan bir ikili doğrusal blok kod inşa edilmek istenirse, bu durumda kodun boyutu $k = 1200156$ olur. Yani $C = (1200178, 1200156, 4)$ en iyi ölçüsündeki ikili doğrusal blok kodunu (4.33) denklemini kullanarak üretmek mümkündür. Bir çok uygulama için kod ölçüsünü kontrol edebilme özelliği önemlidir. Örneğin [84] kaynağında 256 uzunluklu bir kodu inşa edebilmek için bir adet (128, 120) ve iki adet (64, 57) *uzatılmış Hamming kodu* kullanılmış ve sonuçta kodlama oranı $R = (120 + 57 \cdot 2) / (128 + 64 \cdot 2) = 0.914$ olan bir kod elde edilmiştir. Aynı uzunluktaki kod, (4.33) denklemiyle üreteç matrisi oluşturulan BB kodu olarak (256, 247, 4) ölçülerinde inşa edilebilir ve bu kodun kodlama oranı $R = 247/256 = 0.964$ olur. Bu da gösteriyorki BB kodu 0.05 daha iyi kodlama oranı sağlar. Bu örnek, sadece BB kodlarının kodlama oranındaki avantajını göstermek için verilmiş olup, üreteç matrisinin yarı çevrimsel bir yapıda olmasından dolayı pratik kodlama uygulamaları için de çok avantajlı bir durumu ifade eder.

Daha büyük Hamming mesafeli BB kodları da (4.33) denklemi ile gösterilen üreteç matrisi G kullanılarak oluşturulabilir. Bu durumda sadece eleman matrislerini değiştirmek yeterlidir. Örneğin, minimum Hamming mesafesi 8 olan BB kodlarını inşa edebilmek için eleman matrisleri $G_1 = [1 \ 1 \ 1 \ 1]$ ve $G_2 = [1 \ 1 \ 0 \ 0]$ seçilebilir. Ancak, daha iyi kodlama oranı elde edebilmek için başka eleman matrislerini de eklemek mümkün ise eklenmelidir. Bilgisayar programlarıyla yapılan araştırmalar sonucu bir başka eleman matrisinin, $G_3 = [0 \ 1 \ 1 \ 0]$, tıpkı G_2 matrisi gibi G içine G_2 matrisleri yerleştirildikten sonra yerleştirilebileceği görülmüştür. Bu üç eleman matrisleri (4.33)

denkleminde açıklandığı biçimde yerleştirildiğinde minimum Hamming mesafesi 8 olan BB kodları inşa edilebilir. Bunlardan bazıları bir sonraki kısımda oluşturulacaktır.

Minimum Hamming mesafesi 16 olan BB kodları inşası için $G_1 = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$, $G_2 = [1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$, $G_3 = [0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0]$ ve $G_4 = [1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0]$ eleman matrisleri seçilir. Burada G_2 , G_3 ve G_4 birbirinin aynı biçim ve sırayla yerleştirilir. Daha büyük minimum Hamming mesafeli kodlar için de şimdiye kadar sözünü ettiğimiz eleman matrislerine benzer özellikte eleman matrisleri aranmalı ve bilgisayar programı ile oluşturulan kodun parametreleri test edilmelidir.

4.6.2. Basamak Blok Kodlarının Bazı Üreteç Matrisleri

Basamak Blok kodlarından bazıları, aşağıda verilen örneklerdeki gibi oluşturulmuştur.

Örnek 4.22: $C = (12, 7, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 111100000000 \\ 001111000000 \\ 000011110000 \\ 000000111100 \\ 000000001111 \\ 101010100000 \\ 000010101010 \end{bmatrix} \quad (4.35)$$

Örnek 4.23: $C = (14, 9, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000 \\ 00111100000000 \\ 00001111000000 \\ 00000011110000 \\ 00000000111100 \\ 00000000001111 \\ 10101010000000 \\ 00001010101000 \\ 10001000100010 \end{bmatrix} \quad (4.36)$$

Örnek 4.24: $C = (16, 11, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 1111000000000000 \\ 0011110000000000 \\ 0000111100000000 \\ 0000001111000000 \\ 0000000011110000 \\ 0000000000111100 \\ 0000000000001111 \\ 1010101000000000 \\ 0000101010100000 \\ 0000000010101010 \\ 1000100010001000 \end{bmatrix} \quad (4.37)$$

Örnek 4.25: $C = (18, 12, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 111100000000000000 \\ 001111000000000000 \\ 000011110000000000 \\ 000000111100000000 \\ 000000001111000000 \\ 000000000011110000 \\ 000000000000111100 \\ 000000000000001111 \\ 101010100000000000 \\ 000010101010000000 \\ 000000001010101000 \\ 100010001000100000 \end{bmatrix} \quad (4.38)$$

Örnek 4.26: $C = (20, 14, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000 \\ 00111100000000000000 \\ 00001111000000000000 \\ 00000011110000000000 \\ 00000000111100000000 \\ 00000000001111000000 \\ 00000000000011110000 \\ 00000000000000111100 \\ 00000000000000001111 \\ 10101010000000000000 \\ 00001010101000000000 \\ 00000000101010100000 \\ 00000000000010101010 \\ 10001000100010000000 \end{bmatrix} \quad (4.39)$$

Örnek 4.27: $C = (22, 16, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000 \\ 00111100000000000000 \\ 00001111000000000000 \\ 00000011110000000000 \\ 00000000111100000000 \\ 00000000001111000000 \\ 00000000000011110000 \\ 0000000000000011110000 \\ 0000000000000000111100 \\ 0000000000000000001111 \\ 10101010000000000000 \\ 00001010101000000000 \\ 0000000010101010000000 \\ 0000000000001010101000 \\ 1000100010001000000000 \\ 0000000010001000100010 \end{bmatrix} \quad (4.40)$$

Örnek 4.28: $C = (24, 18, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 111100000000000000000000 \\ 001111000000000000000000 \\ 000011110000000000000000 \\ 000000111100000000000000 \\ 000000001111000000000000 \\ 000000000011110000000000 \\ 000000000000111100000000 \\ 000000000000001111000000 \\ 000000000000000011110000 \\ 000000000000000000111100 \\ 000000000000000000001111 \\ 101010100000000000000000 \\ 000010101010000000000000 \\ 000000001010101000000000 \\ 000000000000101010100000 \\ 000000000000000010101010 \\ 100010001000100000000000 \\ 000000001000100010001000 \end{bmatrix} \quad (4.41)$$

Örnek 4.29: $C = (26, 20, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000000000000000 \\ 00111100000000000000000000000000 \\ 00001111000000000000000000000000 \\ 00000011110000000000000000000000 \\ 00000000111100000000000000000000 \\ 00000000001111000000000000000000 \\ 00000000000011110000000000000000 \\ 00000000000000111100000000000000 \\ 00000000000000001111000000000000 \\ 00000000000000000011110000000000 \\ 00000000000000000000111100000000 \\ 00000000000000000000001111000000 \\ 10101010000000000000000000000000 \\ 00001010101000000000000000000000 \\ 00000000101010100000000000000000 \\ 00000000001010101000000000000000 \\ 00000000000010101010000000000000 \\ 10001000100010000000000000000000 \\ 00000000100010001000100000000000 \\ 100000001000000010000000010 \end{bmatrix} \quad (4.42)$$

Örnek 4.30: $C = (28, 22, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000000000000000 \\ 00111100000000000000000000000000 \\ 00001111000000000000000000000000 \\ 00000011110000000000000000000000 \\ 00000000111100000000000000000000 \\ 00000000001111000000000000000000 \\ 00000000000111100000000000000000 \\ 00000000000001111000000000000000 \\ 00000000000000011110000000000000 \\ 00000000000000000111100000000000 \\ 00000000000000000001111000000000 \\ 00000000000000000000011110000000 \\ 00000000000000000000000111100000 \\ 000000000000000000000000011110000 \\ 0000000000000000000000000001111000 \\ 00000000000000000000000000000111100 \\ 00000000000000000000000000000001111 \\ 10101010000000000000000000000000 \\ 00001010101000000000000000000000 \\ 00000000010101010000000000000000 \\ 00000000000001010101000000000000 \\ 00000000000000000101010100000000 \\ 00000000000000000000010101010000 \\ 00000000000000000000000010101010 \\ 10001000100010000000000000000000 \\ 00000000010001000100010000000000 \\ 10000000010000000010000000010000 \end{bmatrix} \quad (4.43)$$

Örnek 4.31: $C = (30, 24, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000000000000000 \\ 00111100000000000000000000000000 \\ 00001111000000000000000000000000 \\ 00000011110000000000000000000000 \\ 00000000111100000000000000000000 \\ 00000000001111000000000000000000 \\ 00000000000011110000000000000000 \\ 00000000000000111100000000000000 \\ 00000000000000001111000000000000 \\ 00000000000000000011110000000000 \\ 00000000000000000000111100000000 \\ 000000000000000000000011110000 \\ 000000000000000000000000111100 \\ 000000000000000000000000001111 \\ 10101010000000000000000000000000 \\ 00001010101000000000000000000000 \\ 00000000101010100000000000000000 \\ 00000000000010101010000000000000 \\ 00000000000000001010101000000000 \\ 000000000000000000001010101000 \\ 10001000100010000000000000000000 \\ 00000000100010001000100000000000 \\ 000000000000000000001000100010 \\ 100000001000000010000000100000 \end{bmatrix} \quad (4.44)$$

Örnek 4.32: $C = (32, 26, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000000000000000 \\ 00111100000000000000000000000000 \\ 00001111000000000000000000000000 \\ 00000011110000000000000000000000 \\ 00000000111100000000000000000000 \\ 00000000001111000000000000000000 \\ 00000000000011110000000000000000 \\ 00000000000000111100000000000000 \\ 00000000000000001111000000000000 \\ 00000000000000000011110000000000 \\ 00000000000000000000111100000000 \\ 00000000000000000000001111000000 \\ 00000000000000000000000011110000 \\ 00000000000000000000000000111100 \\ 00000000000000000000000000001111 \\ 10101010000000000000000000000000 \\ 00001010101000000000000000000000 \\ 00000000101010100000000000000000 \\ 00000000001010101000000000000000 \\ 00000000000010101010000000000000 \\ 00000000000000101010100000000000 \\ 00000000000000001010101000000000 \\ 000000000000000000101010100000 \\ 10001000100010000000000000000000 \\ 00000000100010001000100000000000 \\ 00000000000000001000100010001000 \\ 10000000100000001000000010000000 \end{bmatrix} \quad (4.45)$$

Örnek 4.33: $C = (34, 27, 4)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 11110000000000000000000000000000 \\ 00111100000000000000000000000000 \\ 00001111000000000000000000000000 \\ 00000011110000000000000000000000 \\ 00000000111100000000000000000000 \\ 00000000001111000000000000000000 \\ 00000000000011110000000000000000 \\ 00000000000000111100000000000000 \\ 00000000000000001111000000000000 \\ 00000000000000000011110000000000 \\ 00000000000000000000111100000000 \\ 00000000000000000000001111000000 \\ 00000000000000000000000011110000 \\ 0000000000000000000000000011110000 \\ 0000000000000000000000000000111100 \\ 0000000000000000000000000000001111 \\ 10101010000000000000000000000000 \\ 00001010101000000000000000000000 \\ 00000000101010100000000000000000 \\ 00000000000010101010000000000000 \\ 00000000000000001010101000000000 \\ 00000000000000000000101010100000 \\ 0000000000000000000000001010101000 \\ 10001000100010000000000000000000 \\ 00000000100010001000100000000000 \\ 0000000000000000000010001000100000 \\ 10000000100000001000000010000000 \end{bmatrix} \quad (4.46)$$

Örnek 4.34: $C = (40, 33, 4)$ olan bir BB kodu üreteç matrisi:

[illegible]

Örnek 4.35: $C = (16, 5, 8)$ olan bir BB kodu üreteç matrisi:

$$G = \begin{bmatrix} 1111111100000000 \\ 0000111111110000 \\ 0000000011111111 \\ 1100110011001100 \\ 0110011001100110 \end{bmatrix} \quad (4.48)$$

Örnek 4.36: $C = (24, 9, 8)$ olan bir BB kodu üreteç matrisi:

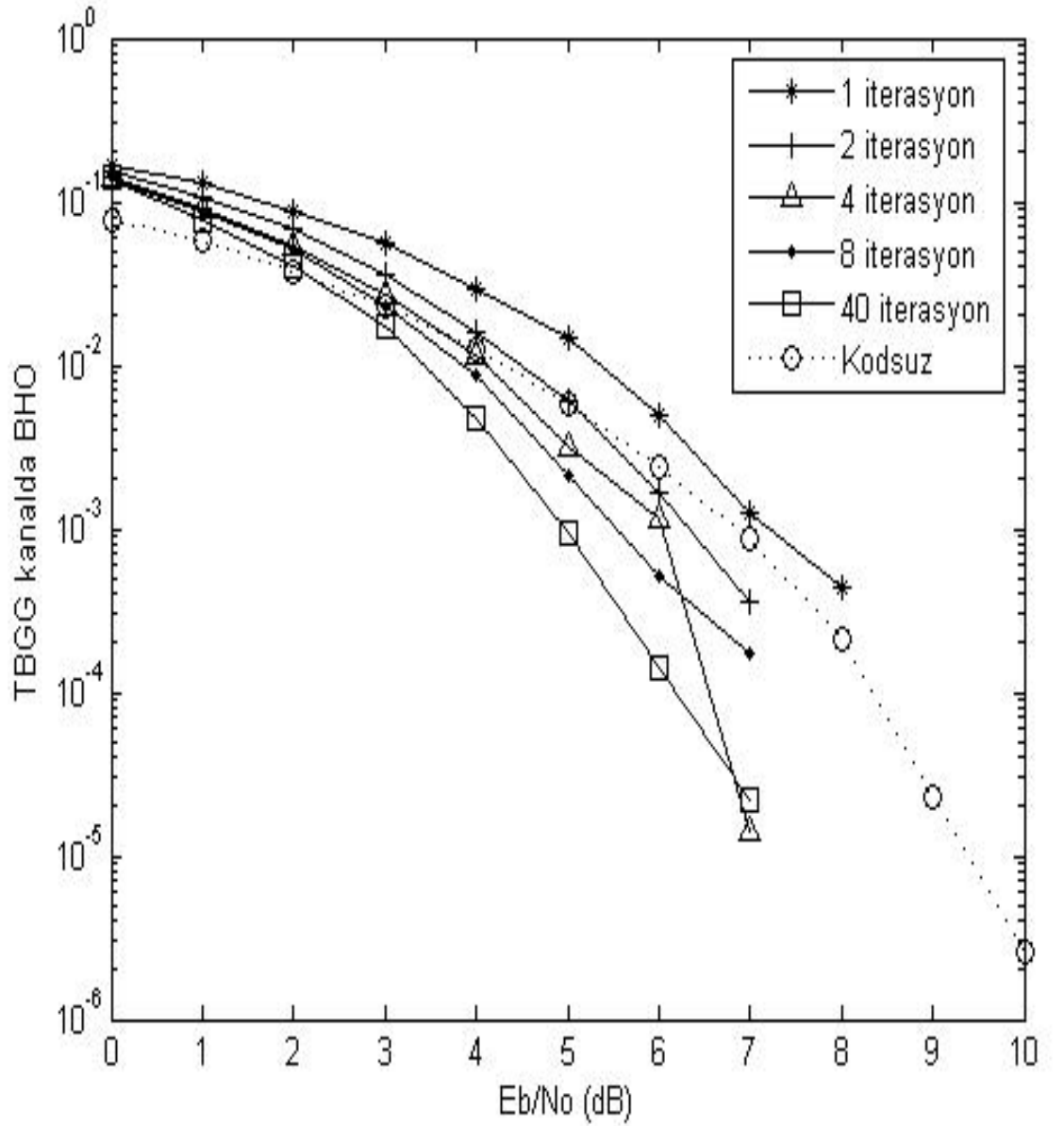
$$G = \begin{bmatrix} 111111110000000000000000 \\ 000011111111000000000000 \\ 000000001111111100000000 \\ 000000000000111111110000 \\ 000000000000000011111111 \\ 110011001100110000000000 \\ 000000001100110011001100 \\ 011001100110011000000000 \\ 000000000110011001100110 \end{bmatrix} \quad (4.49)$$

Örnek 4.37: $C = (32, 15, 8)$ olan bir BB kodu üreteç matrisi:

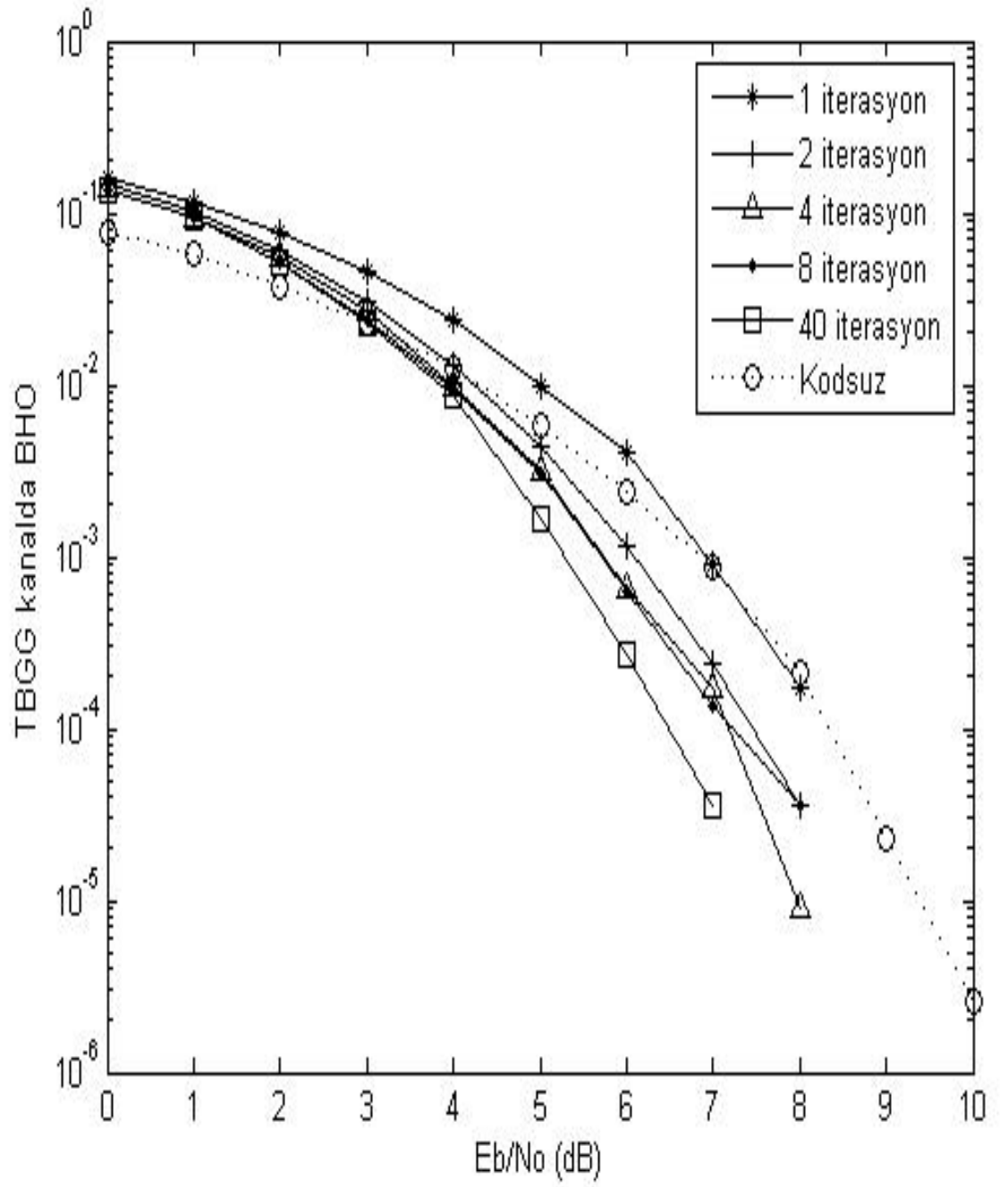
$$G = \begin{bmatrix} 11111111000000000000000000000000 \\ 00001111111100000000000000000000 \\ 00000000111111110000000000000000 \\ 00000000000011111111000000000000 \\ 00000000000000001111111100000000 \\ 00000000000000000000111111110000 \\ 00000000000000000000000011111111 \\ 11001100110011000000000000000000 \\ 00000000110011001100110000000000 \\ 000000000000000001100110011001100 \\ 11000000110000001100000011000000 \\ 01100110011001100000000000000000 \\ 00000000011001100110011000000000 \\ 000000000000000000110011001100110 \\ 01100000011000000110000001100000 \end{bmatrix} \quad (4.50)$$

4.6.3. Toplam-Ürün Algoritması Kodçözümüyle Basamak Blok Kodlarının Hata Başarımı

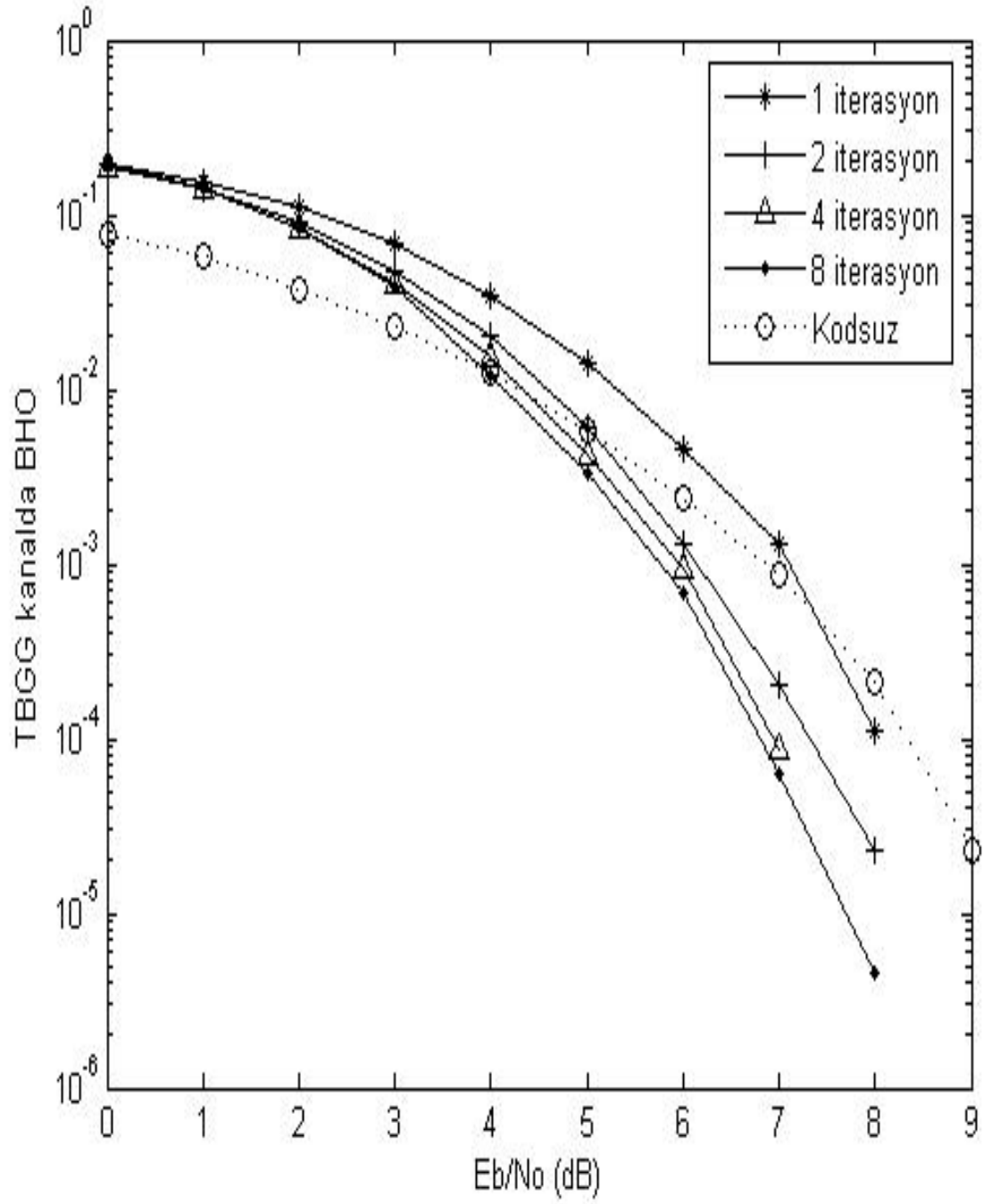
Bu kısımda BB kodlarının toplam-ürün algoritması ile kodçözülmesi halindeki hata başarımları bilgisayar benzetimleri yardımıyla elde edilmiştir. Daha önce olduğu gibi buradaki bilgisayar benzetimi ile oluşturulan modelde İFKA modülasyonu kullanılmış ve TBGG kanaldan işaretler iletilmiştir. Elde edilen sonuçlar aşağıdaki şekilde gösterilmiştir



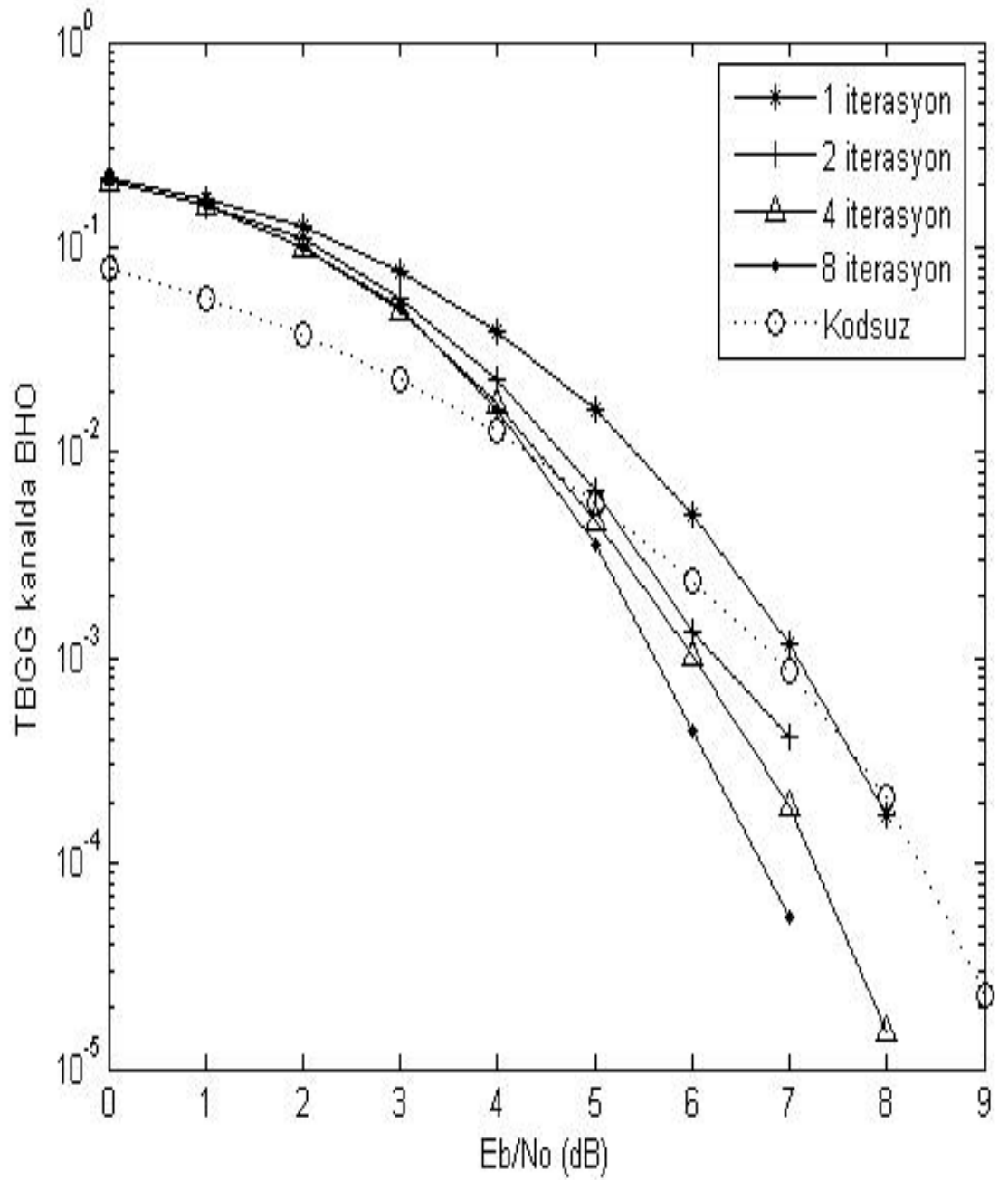
Şekil 4.42: $C = (12, 7, 4)$ BB kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO'ları.



Şekil 4.43: $C = (16, 11, 4)$ BB kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO'ları.

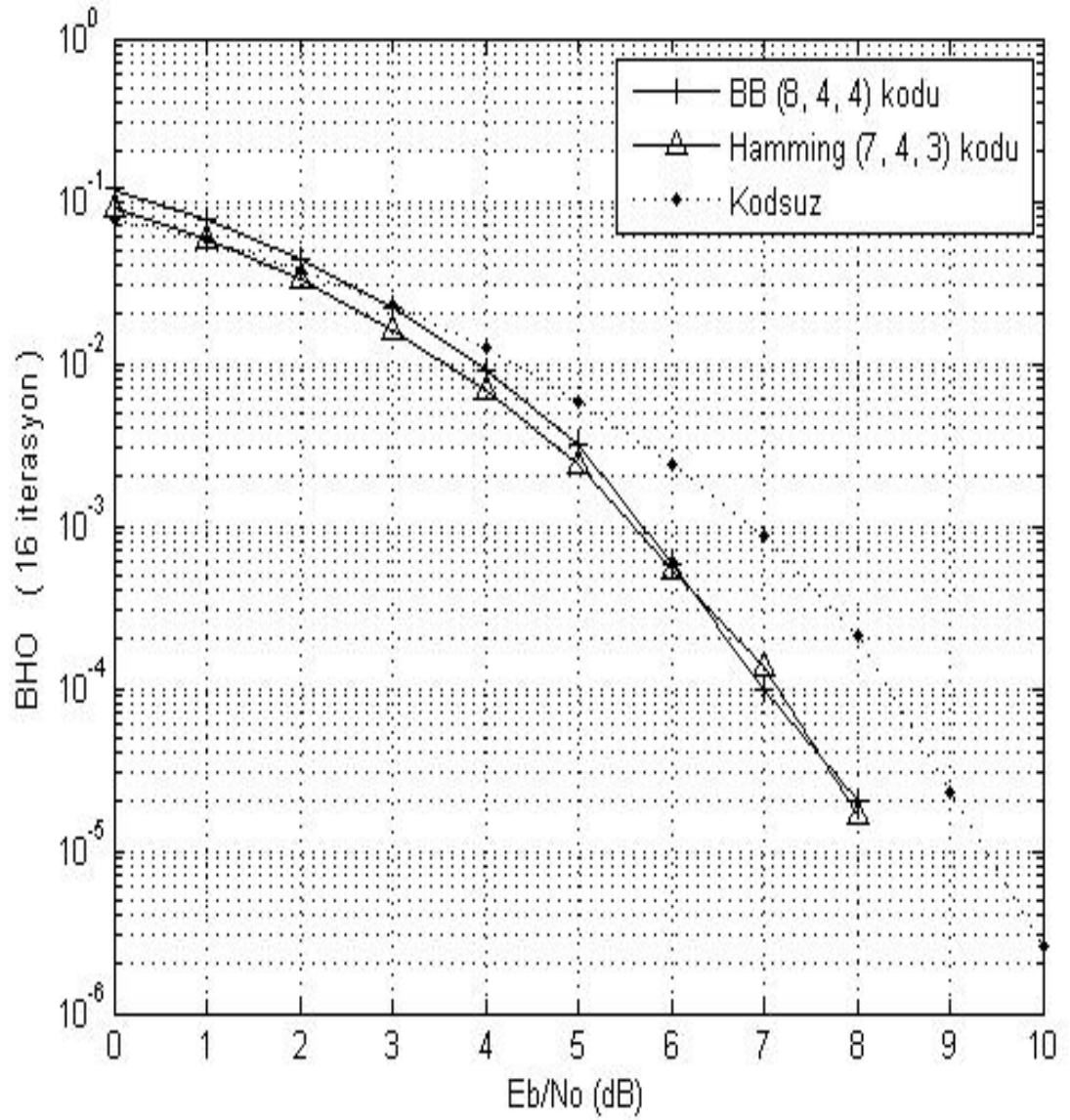


Şekil 4.44: $C = (28, 22, 4)$ BB kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO'ları.



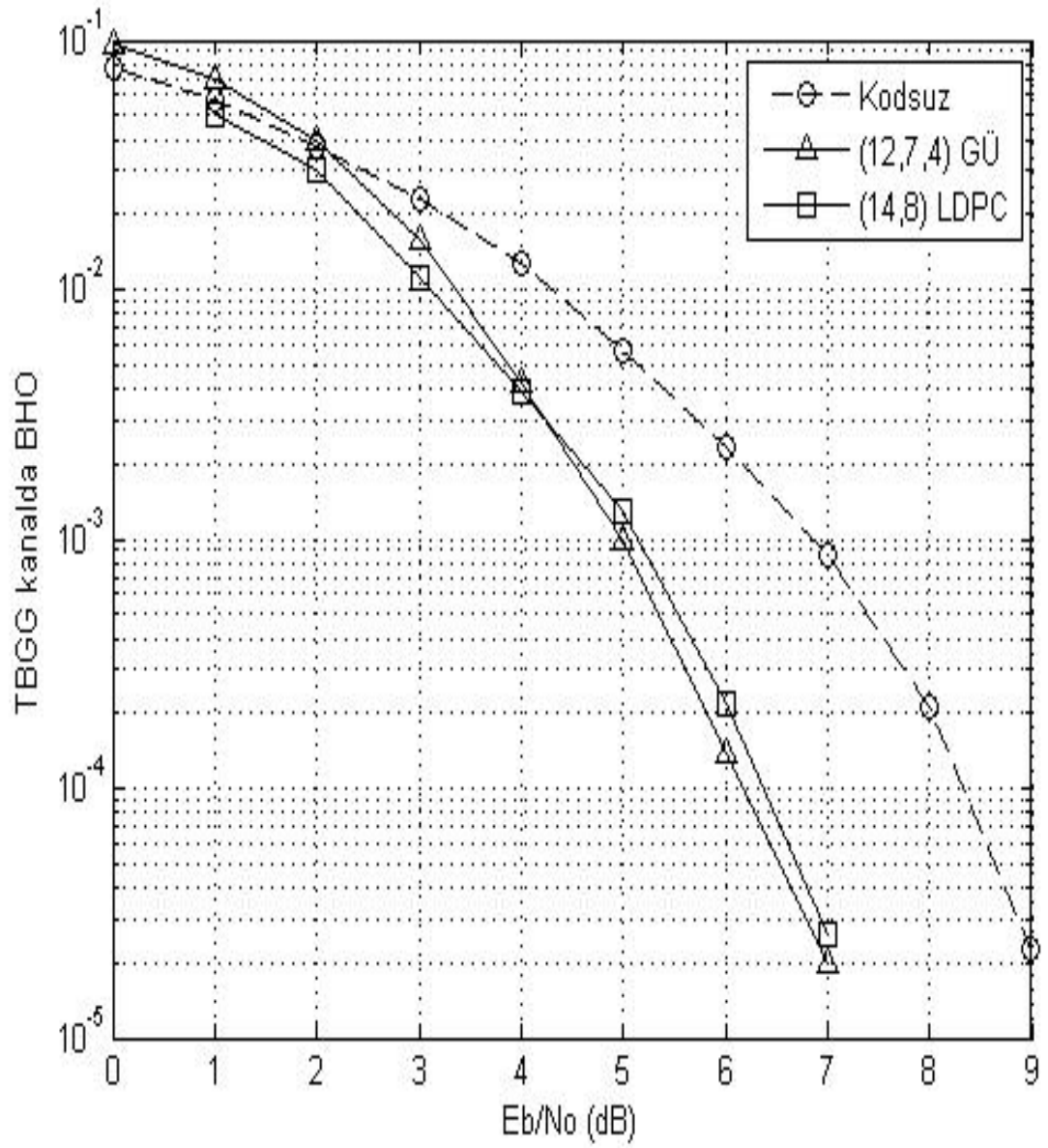
Şekil 4.45: $C = (32, 26, 4)$ BB kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kodçözme algoritması yardımıyla elde edilen BHO' ları.

BB kodlarında da gözlemleneceği gibi toplam-ürün kodçözme algoritmasıyla elde edilen hata başarımları 4 iterasyona kadar hızlı bir şekilde iyileşmekle beraber, 4 iterasyondan sonra iyileşme çok az olmaktadır.



Şekil 4.46: BB (8, 4, 4) ve (7, 4, 3) Hamming kodlarının TBGG kanalda toplam-ürün algoritması ile 16 iterasyonda elde edilen BHO' ları. Ayrıca, kodlanmamış bitlerin (kodsuz) hata eğriside verilmiştir.

Şekildende görüldüğü gibi hata BB (8, 4, 4) ve (7, 4, 3) Hamming kodlarının hata başarımları 16 iterasyonda hemen hemen aynıdır.



Şekil 4.47: BB (12, 7, 4) ve LDPC (14,8) [2] kodlarının TBGG kanalda toplam-ürün algoritması ile 200 iterasyonda elde edilen BHO' ları. Ayrıca, kodlanmamış bitlerin (kodsuz) hata eğriside verilmiştir.

Şekildende görüldüğü gibi hata BB (12, 7, 4) ve LDPC (14, 8) kodlarının hata başarımları 200 iterasyonda yaklaşık aynıdır.

5. TARTIŞMA VE SONUÇ

Her şeyden önce belirtmek gerekir ki bu çalışmada, var olan bir yöntemin uygulanması yerine, yeni yöntemler önerilmiştir. Bu çalışmada ile yeni bir kanal kodlama yöntemi olan Geometrik Ürün (GÜ) kodlaması sunulmuştur. Bu kodlar genel yapıda olup, uzunluğu 8'den büyük ve Hamming mesafesi 4 olan bütün en iyi çift ikili blok kodları, bir kod ailesi olarak, üretebilmektedir. Bu kodların minimum Hamming mesafesinin 4 olduğu Teorem 4.1 ile ispat edilmiştir.

Ayrıca önerilen yöntemlerle, Hamming mesafesi 8, 16 ve daha büyük olan bazı en iyi ve iyi kodları da üretmektedir. Ancak, kod inşasında kullanılan basit eleman matrislerinin seçilmesinde bir genelleştirme yapılmadığından en iyi kodları elde edebilmek güçleşmektedir.

Önerilen kodların kodçözümünü gerçekleştirebilmek için gerekli kafes yapıları; minimum, az karmaşık, basit, esnek ve uygulanabilir yapıda tasarlanmıştır. Bu önerilen kafes yapılarının Viterbi algoritması ile kodçözümü yapıldığında daha az karmaşıklığa sahip olduğunu göstermek için literatürde benzer kod ve kafes yapılarından örnekler verilerek karşılaştırma yapılmıştır.

Viterbi ve BCJR kodçözme algoritmaları kullanılarak önerilen kafes yapıları üzerinden GÜ kodlarının bazıları için hata başarımları, TBGG kanal ortamında, bilgisayar benzetimi aracılığıyla elde edilmiştir. Ayrıca iteratif kodçözme ve toplam-ürün kodçözme algoritmaları kullanılarak bazı GÜ kodları için hata başarımları analizi yapılmıştır.

Tez çalışması sırasında keşfedilen bir kodlama tekniği olan Basamak Blok (BB) kodları inşası, GÜ kod inşasından daha kolay yapıda fakat GÜ kodları ile aynı ölçülerde olacak şekilde elde edilmiştir. BB kodları, GÜ kodlarında temel olarak kullanılan genel bir

ürün kodu üreteç matrisini hiç kullanmadan, yeni bir üreteç matrisi yapısına sahiptir. GÜ kodlarından esas farkı da bu genel ürün matrisini kullanmamasıdır. BB kodları da GÜ kodlarında olduğu gibi Hamming mesafesi 4 olan bütün en iyi çift kodları, bir kod ailesi olarak, üretebilmektedir. Bu kodların minimum Hamming mesafesinin 4 olduğu Yardımcı Teorem 4.1, Yardımcı Teorem 4.2 ve Teorem 4.4 ile önerilerek ispat edilmiştir. BB kodları da genel bir yapıda olup, minimum Hamming mesafesi daha büyük olan kodları üretebilir.

BB kodlarından bazıları için hata başarımları, TBGG kanal ortamında, bilgisayar benzetimi aracılığıyla toplam-ürün kodçözme algoritması kullanılarak elde edilmiştir.

Bu tezde önerilen BB ve GÜ kodları ile üretilen minimum Hamming mesafesi 4 olan kod aileleri, uzatılmış Hamming kodlarını ve minimum Hamming mesafesi 4 olan Reed-Muller kodlarını da içine alır. Çünkü literatürde çok uzun zamandır var olan bu kodların uzunlukları 2'nin kuvvetleri olarak değişmektedir; yani 8, 16, 32 gibi kod uzunluklarına sahiptir. Diğer taraftan bu çalışmada önerilen GÜ ve BB kod aileleri 2'nin katları olan, 8, 10, 12, 14, 16, 18,..., gibi, kod uzunluklarına sahiptir. Bu bakımdan, GÜ ve BB kodları daha üst bir kod ailesi olarak karşımıza çıkmaktadır. Bu çalışmayla elde edilen sonuçlar çeşitli dergi ve konferanslarda sunulmuştur [87 - 97].

Düşük yoğunluklu üreteç matrisine sahip kodların kodlayıcı ve kodçözücü yapısını kolaylaştırdığı ve aynı zamanda çok iyi hata performansı gösterdiği literatürde bilinmektedir [85, 86]. Önerilen BB ve GÜ kodları, kod inşası çok kolay, düzenli ve yarı çevrimsel bir yapıdadır. Dolayısıyla kaydırmalı kaydediciler kullanarak çok büyük uzunluktaki kodları bile çok daha kolay üretebilme olanağı sağlamaktadır. Daha da önemlisi BB ve GÜ kodlarının üreteç matrisleri, düşük yoğunluklu üreteç matrisi yapılarına sahip olup, kodlayıcı uygulamalarında çok faydalı olan bir özelliği barındırmaktadır.

Önerilen GÜ ve BB kodları özellikle büyük oranlı kodlar gerektiren bant-genişliği etkin uygulamalarda, örneğin, bilgisayar içi haberleşmelerinde, saklayıcılarda, optik haberleşmede ve diğer bilinen uygulamalarda kullanılabilir.

Bundan sonra bu konuda yapılabilecek çok fazla çalışma olacağı değerlendirilmektir. Çünkü bu yeni bir yöntemdir ve bu yöntemin hangi uygulamalarda ne kadar etkin olduğu önemli bir araştırma alanıdır. Bunun dışında BB ve GÜ kodlarında kullanılan eleman matrislerinin her bir minimum Hamming mesafesi için ne olacağının genel bir yapıya dökülmesi çok önemli ikinci bir araştırma alanıdır. Ayrıca, çok büyük uzunluklardaki, mesela 4000, BB ve GÜ kodlarının hata başarımlarının elde edilmesi ve literatürde çok popüler olan düşük-yoğunluklu parite-kontrol kodları (LDPC) ile kıyaslanması ilginç olacaktır. Bundan başka önerilen kodların pratikte kullanılan kodlayıcı mimarı tasarımları da başka bir araştırma alanıdır. Elbette çok daha başka araştırmalar da BB ve GÜ kodları bağlamında gerçekleştirilebilir, burada sadece bazı önemli olanları vurgulanmıştır.

KAYNAKLAR

1. SHANNON, C. E., 1948, A mathematical theory of communication, *Bell Sys. Tech. J.*, pp.379-423 (Part 1); pp. 623-656 (Part 2).
2. LIN, S.ve COSTELLO, D. J., Jr., 2004, *Error control coding*, second edition, Prentice Hall, 0-13-017973-6.
3. BOSSERT, M., 1999, *Channel Coding*, Wiley, 0-471-98277.
4. HAMMING, R. W., 1950, Error detecting and error correcting codes, *Bell Sys. Tech. J.*, 29:147-60.
5. MULLER, D.E., September 1954, Applications of boolean algebra to swiKKhing circuits design and to error detection, *IRE Trans.*, EC-3: 6-12.
6. REED, I. S., September 1954, A class of multiple-error-correcting codes and the decoding scheme, *IRE Trans.*, IT-4: 38-49.
7. ELIAS, P., September 1954, Error-free coding, *IRE Trans. Inform. Theory*, PGIT-4: 29-37.
8. MACWILLIAMS, F. J. ve SLOANE, N. J. A., 1977, *The theory of error-correcting codes*, North Holland, Amsterdam, 0-444-85193-1.
9. PLOTKIN, M., November 1960, Binary codes with specific minimum distance, *IEEE Trans. Inform. Theory*, IT-34: 1152-1187.
10. GOLAY, M. J. E., June 1949, Notes on digital coding, *Proc. IEEE*, 37:657.
11. VANSTONE, S. A. ve VAN OORSCHOT, P. C., 1989, *An introduction to error correcting codes with applications*, Kluwer academic, Boston, 0-7923-9017-2.
12. WICKER, S. B., 1995, *Error control systems for digital communications and storage*, Prentice Hall, Englewood Cliffs,N.J., 0132008092.
13. PRANGE, E., September 1957, Cyclic error-correcting codes in two symbols, *AFCRC-TN-57,103*, Air force cambridge research center, Cambridge.
14. HOCQUENGHEM, A.,1959, Codes corecteurs d'erreurs, *Chiffres*, 2:147-56.

15. BOSE, R. C. ve RAY-CHAUDHURI, D. K., March 1960, On a class of error correcting binary group codes, *Inform. Control*, 3: 68-79.
16. REED, I. S. ve SOLOMON, G., June 1960, Polynomial codes over certain finite fields, *J. Soc. Ind. Appl. Math.*, 8: 300-304.
17. HONARY, B., MARKARIAN, G. S. ve FARREL, P. G., 1993, Generalised array codes and their trellis structure, *IEE Electron. Lett.*, Vol.29, no.6, pp. 541-542.
18. BERROU, C., GLAVIEUX, A. ve THITIMAJSHIMA, P., MAY 1993, Near Shannon limit error-correcting coding and decoding: Turbo codes, *Proc. IEEE Intl Conf. Commun. (ICC 93)*, pp.1064-70, Geneva, Switzerland.
19. BERROU, C. ve GLAVIEUX, A., October 1996, Near optimum error correcting and decoding: Turbo codes, *IEEE Trans. Commun.*, COM-44:1261-71.
20. BERROU, C. ve GLAVIEUX, A., June 1998, Reflections on the prize paper: Near optimum error-correcting coding and decoding: Turbo codes, *IEEE Inform. Theory Soc. Newsletter*, 48(2):24-31.
21. GALLAGER, R. G., January 1962, Low density parity check codes, *IRE Trans. Inform. Theory*, IT-8: 21-28.
22. GALLAGER, R. G., 1963, Low density parity check codes, *MIT Press*, Cambridge.
23. MACKAY, D. J. C. ve NEAL, R. M., 1996, Near Shannon limit performance of low density parity check codes, *Electron. Lett.*, 32(18): 1645-46.
24. LI, J., NARAYANAN, K. R. ve GEORGHIADES, C. N., January 2004, Product accumulate codes: A class of codes with near-capacity performance and low decoding complexity, *IEEE Trans. Inform. Theory*, vol.50, no.1, pp. 31-46.
25. WOZENCRAFT, J. M. ve JACOB, I. M., 1965, *Principles of communication engineering*, John Wiley, New York, 0881335541.
26. PROAKIS, J. G., 2000, *Digital communications*, 4th ed., Mc-Graw-Hill, New York, 0072321113.
27. FORNEY, G. D. Jr., 1973, The Viterbi algorithm, *Proc. IEEE*, 61: 268-78.
28. VITERBI, A. J., 1967, Error bounds for convolutional codes and asymptotically optimum decoding algorithm, *IEEE Trans. Inform. Theory*, IT-13: 260-69.
29. BAHL, L. R., COCKE, J., JELINEK, F. ve RAVIV, J., 1974, Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Trans. Inform. Theory*, IT-20: 284-87.

30. WOLF, J. K., 1978, Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory*, IT-24: 76-80.
31. MASSEY, J. L., 1978, Foundation and methods of channel encoding, in *Proc. Int. Conf. Inform. Theory and Systems*, NTG-Fachberichte, Berlin.
32. FORNEY, G. D. Jr., 1988, Coset codes II: Binary lattices and related codes, *IEEE Trans. Inform. Theory*, IT-34: 1152-87.
33. LAFOURCADE, A. ve VARDY, A., 1995, Asymptotically good codes have infinite trellis complexity, *IEEE Trans. Inform. Theory*, vol.41, pp. 555-559.
34. SCHUURMAN, P., Nov. 1996, A table of state complexity bounds for binary linear codes, *IEEE Trans. Inform. Theory*, vol.42, no.6, pp. 2034-2042.
35. FORNEY, G. D. Jr., Nov. 1994, Dimension / length profiles and trellis complexity of linear block codes, *IEEE Trans. Inform. Theory*, vol.40, no.6, pp. 1741-1752.
36. LIN, S., KASAMI, T., FUJIWARA, T. ve FOSSORIER, M. P. C., 1998, *Trellises and trellis-based decoding algorithms for linear block codes*, Kluwer Academic, Boston, 0-7923-8151-3.
37. VARDY, A., 1998, Trellis structure of codes, *Elsevier Science*, Amsterdam vol. 2, ch. 24, pp. 1989-2118.
38. MUDER, D. J., 1988, Minimal trellises for block codes, *IEEE Trans. Inform. Theory*, 34: 1049-53.
39. BERGER, Y. ve BE'ERRY, Y., 1993, Bounds on the trellis size of linear block codes, *IEEE Trans. Inform. Theory*, IT-39: 203-9.
40. FORNEY, G. D. ve TROTT, M. D., 1993, The dynamics of group codes: state spaces, trellis diagrams and canonical encoders, *IEEE Trans. Inform. Theory*, IT-39: 1491-513.
41. YTREHUS, O., 1995, On the trellis complexity of certain binary linear block codes, *IEEE Trans. Inform. Theory*, 40: 559-60.
42. LAFOURCADE, A. ve VARDY, A., 1996, Optimal sectionalization of a trellis, *IEEE Trans. Inform. Theory*, 42: 1072-92.
43. MCELIECE, R. J., 1996, On the BCJR trellis for linear block codes, *IEEE Trans. Inform. Theory*, 42: 1072-92.
44. SIDORENKO, V. R., MARKARIAN, G. ve HONARY, B., 1996, Minimal trellis design for linear block codes based on the shannon product, *IEEE Trans. Inform. Theory*, 42: 2048-53.

45. MOORTHY, H. T., LIN, S. ve KASAMI, T., May 1997, Soft-decision decoding of binary linear block codes based on iterative search algorithm, *IEEE Trans. Inform. Theory*, 43: 1030-40.
46. MORELOS-ZARAGOZA, R. H., FUJIWARA, T., KASAMI, T. ve LIN, S., March 1999, Constructions of generalized concatenated codes and their trellis-based decoding complexity, *IEEE Trans. Inform. Theory*, 45: 725-731.
47. VARDY, A. ve BE'ERY, Y., 1994, Maximum likelihood soft decision decoding of BCH codes, *IEEE Trans. Inform. Theory*, vol. IT-40, pp. 546-554.
48. OMURA, J. K., January 1969, On the Viterbi decoding algorithm, *IEEE Trans. Inform. Theory*, IT-15: 177-79.
49. HAGENAUER, J. ve HOEHER, P., November 1989, A Viterbi decoding algorithm with soft decision outputs and its applications, *Proc. IEEE Global conference on communications*, pp.1680-86, Dallas, Tex.
50. VITERBI, A. J., October 1971, Convolutional codes and their performance in communications systems, *IEEE Trans. Commun. Technol.*, COM-19: 751-72.
51. VITERBI, A. J. ve OMURA, J. K., 1979, *Principles of digital communication and coding*, McGraw-Hill, Newyork.
52. HAGENAUER, J., OFFER, E. ve PAPKE, L., March 1996, Iterative decoding of binary block and convolutional codes, *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445.
53. SKLAR, B., Dec. 1997, A primer on turbo code concepts, *IEEE Comm. Magazine*, pp. 94-102.
54. ANDERSEN, J. D., June 1994, The turbo coding scheme, *IEEE Intl. Symp. Inform. Theory (ISIT 1994)*, Trondheim, Norway.
55. ROBERTSON, P., November 1994, Illuminating the structure of code and decoder of parallel concatenated recursive systematic (Turbo) codes, *Proc. IEEE Global Commun.*, pp.343-46, Kingston, Ontario.
56. BENEDETTO, S. ve MONTORSI, G., March 1996, Unveiling turbo codes: some results on parallel concatenated coding, *IEEE Trans. Inform. Theory*, IT-42: 409-428.
57. BENEDETTO, S. ve MONTORSI, G., May 1996, Design of parallel concatenated convolutional codes, *IEEE Trans. Commun.*, COM-44: 591-600.
58. BATTAIL, G., February 1998, A conceptual framework for understanding turbo codes, *IEEE J. Select. Areas Commun.*, SAC-16:245-54.

59. DIVSALAR, D. ve POLLARA, F., November 1995, On the design of turbo codes, *TDA Progress report*, 42-123, Jet propulsion laboratory, Pasadena, Calif.
60. HEEGARD, C. ve WICKER, S. B., 1999, *Turbo Coding*, Kluwer Academic, Boston, Mass.
61. VUCETIC, B. ve YUAN, J., 2000, *Turbo codes*, Kluwer Academic, Boston, Mass.
62. RYAN, W. E., 2002, *Concatenated convolutional codes and iterative decoding*, in Wiley encyclopedia of telecommunications, edited by PROAKIS, J. G., John Wiley, New York.
63. BRINK, S. TEN, May 1999, Convergence of iterative decoding, *IEE Electron. Lett.*, 35: 806-8.
64. WIEBERG, N., LOELIGER, H. A., ve KOETTER, R., September 1995, Codes and iterative decoding on general graphs, *Eur. Trans. Telecommun.*, 513-26.
65. PEARL, J., 1988, *Probabilistic reasoning in intelligent systems*, San Mateo, CA: Morgan Kaufmann.
66. MACKAY, D., March 1999, Good error correcting codes based on very sparse matrices, *IEEE Trans. Inform. Theory*, pp. 399-431.
67. TANNER, R. M., Sept. 1981, A recursive approach to low complexity codes, *IEEE Trans. Inform. Theory*, pp. 533-547.
68. SUN, J., 2003, An introduction to low density parity check (LDPC) codes, Lane Dept. of Comp. Sci. & Elect. Eng., West Virginia University, USA, www.csee.wvu.edu/wcrl/public/slidelldpc.pdf, [Ziyaret tarihi: 12 July 2004].
69. MACKAY, D., July 1999, Sparse graph codes, *Proc. 5th Intl. Symp. Commun. Theory and Applications*, pp. 2-4, Ambleside, UK.
70. RICHARDSON, T., SHOKROLLAHI, A. ve URBANKE, R., February 2001, Design of capacity approaching irregular codes, *IEEE Trans. Inform. Theory*, 47(2): 619-37.
71. RICHARDSON, T. ve URBANKE, R., February 2001, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inform. Theory*, 47: 599-618.
72. CHUNG, S. Y., FORNEY, G. D. Jr., RICHARDSON, T. ve URBANKE, R., February 2001, On the design of low density parity check codes within 0.0045 dB of the Shannon limit, *IEEE Commun. Lett.*, 5:58-60.

73. KOU, Y., LIN, S. ve FOSSORIER, M., November 2001, Low density parity check codes based on finite geometries: A rediscovery and more, *IEEE Trans. Inform. Theory*, 47(6): 2711-36.
74. WIBERG, N., LOELIGER, H.-A. ve KÖTTER, R., 1995, Codes and iterative decoding on general graphs, *Eur. Trans. Telecommun.*, 6:513-25.
75. MCELIECE, R. J., MACKAY, D. J. C. ve CHENG, J. -F., February 1998, Turbo decoding as an instance of pearl's belief propagation algorithm, *IEEE J. Select. Areas Commun.*, 16: 140-52.
76. KSCHISCHANG, F. R. ve FREY, B. J., February 1998, Iterative decoding of compound codes by probability propagation in graphical models, *IEEE J. Select. Areas Commun.*, 16(12): 219-30
77. KSCHISCHANG, F. R., FREY, B. J. Ve LOELIGER, H.-A., February 2001, Factor graphs and the sum-product algorithm, *IEEE Trans. Inform. Theory*, 47: 498-519.
78. FOSSORIER, M., MIHALJEVIC, M. ve IMAI, H., May 1999, Reduced complexity iterative decoding of low density parity check codes, *IEEE Trans, Commun.*, 47: 673-80.
79. LUCAS, R., FOSSORIER, M., KOU, Y. ve LIN, S., June 2000, Iterative decoding of one step majority logic decodable codes based on belief propagation, *IEEE Trans, Commun.*, 48(6): 931-37.
80. BROUWER, A. E. ve VERHOEFF, T., March 1993, An updated table of minimum-distance bounds for binary linear codes, *IEEE Trans. Inform. Theory*, vol. 39, No. 2, pp. 664-677.
81. MOORTHY, H. T., LIN, S. ve UEHARA, G. T., Jan. 1997, Good trellises for IC implementation of Viterbi decoders for linear block codes, *IEEE Trans, Commun.*, vol. 45, No. 1, pp. 52-63.
82. HONARY, B. ve MARKARIAN, G., 1997, *Trellis decoding of block codes*, Kluwer Academic Publishers, USA., 0-7923-9860-2.
83. PLESS, V., 1989, *Introduction to the theory of error-correcting codes*, 2nd edition, John Wiley and Sons, Inc., 0-471-19047-0.
84. ISAKA, M. ve FOSSORIER, M., Feb. 2005, High-rate serially concatenated coding with extended Hamming codes, *IEEE Comm. Letter*, vol. 9, pp. 160-162, No.2.
85. OENING, T., R., MOON, J., February 2001, A low-density generator matrix interpretation of parallel concatenated single bit parity codes, *IEEE Trans. Magn.*, vol. 37, pp. 737-741.

86. FRIAS, J.,G., June 2003, Approaching shannon performance by iterative decoding of linear codes with low-density generator matrix, *IEEE Comm. Letter*, vol. 7, pp. 266-268, No.6.
87. ALTAY, G., UÇAN, O. N., Uğurdağ H. F., Geometric Augmented Product Codes, *IEE Proc. Communications*., accepted with revision..
88. ALTAY, G., UÇAN, O. N., 2005, Iterative Decoding of Geometric Augmented Product Codes, *Istanbul University - Journal of Electrical and Electronics Engineering*, Volume: 5, Number: 2.
89. ALTAY, G., YALCIN, S., July 2005, Belief Propagation Decoding of Some Decomposable Linear Block Codes, *International Conference on Recent Advances in Space Technologies (RAST)*, Istanbul, Turkey.
90. ALTAY, G., UÇAN, O. N., September 2005, Maximum A Posteriori Decoding of Geometric Product Codes, *Trends in the Development of Machinery and Associated Technology (TMT 2005)*, pp.1181-1184, Antalya, Turkey.
91. ALTAY, G., UÇAN, O. N., December 2005, Iterative Decoding Performance of Geometric Augmented Product Codes Over AWGN Channel, *International Conference on Electrical & Electronics Engineering, (ELECO)*, Bursa, Turkey.
92. ALTAY, G., UÇAN, O. N., Mayıs 2005, EAPCC Blok Kodları İçin Genel Kafes Yapısı Tasarımı, *IEEE SİU'05 Kurultayı*, Kayseri, Türkiye.
93. ALTAY, G., YALCIN, S., UÇAN, O. N., November 2003, Trellis Decoding of Some Decomposable Codes, *International Conference on Recent Advances in Space Technologies (RAST)*, Istanbul, Turkey.
94. ALTAY, G., ALTAY, S., UÇAN, O. N., July 2003, Trellis of (9,4,4) Decomposable Code of the $|a_1+x+z|a_2+x+z|a_1+a_2+x+z|$ Construction, *International Twelfth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN)*, Çanakkale, Turkey.
95. ALTAY, G., UÇAN, O. N., December 2003, Construction and Minimal Trellis for (32,16, 8) Decomposable Code, *International Conference on Electrical & Electronics Engineering, (ELECO)*, Bursa, Turkey.
96. ALTAY, G., UÇAN, O. N., September 2003, BER Performances of Some Distance-8 Decomposable Linear Block Codes, *International Conference on Signal Processing (ICSP)*, Çanakkale, Turkey.
97. ALTAY, G., UÇAN, O. N., Heuristic Construction of High -Rate Linear Block Codes, *International Journal of Electronics and Communications (AEUE: Archiv fuer Elektronik und Uebertragungstechnik)*, in press.

ÖZGEÇMİŞ

Gökmen ALTAY 20 Kasım 1978’de Ankara’da dünyaya gelmiştir. İlk, orta ve lise öğrenimini sırası ile Emirler İlkokulu, Sakarya İlköğretim Okulu, Mehmetçik Lisesi’nde Ankara’da tamamladıktan sonra 1999 yılında Trabzon’da Karadeniz Teknik Üniversitesi Elektrik Elektronik Mühendisliği Bölümünden Elektronik Mühendisi ünvanı ile mezun olmuştur. Daha sonra Londra’da, 2000-2001 yılları arasında Yüksek Lisans eğitimini London South Bank University, Telecommunications and Computer Network Engineering bölümünde tamamlamıştır.

2002 yılında Araştırma Görevlisi olarak çalışmaya başladığı Bahçeşehir Üniversitesi, Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği Bölümünde halen Öğretim Görevlisi olarak görev yapmaktadır.

Eylül 2002’de İstanbul Üniversitesi’nin Fen Bilimleri Enstitüsü’nün Elektrik-Elektronik programında Doktora eğitimine başlamıştır.

Gökmen ALTAY’ın tez çalışmaları sırasında elde edilen sonuçlar üzerine yazmış olduğu bildiri ve makaleler yurtiçi ve yurtdışında çeşitli dergi ve konferanslarda yayınlamıştır.