KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ İLETİŞİMDE KULLANILAN KATLAMALI KODLAMA TEKNİKLERİNİN KARŞILAŞTIRMALI BAŞARIM ANALİZİ

YÜKSEK LİSANS

Ali ÇALHAN

Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışman: Yrd. Doç. Dr. Celal ÇEKEN

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ İLETİŞİMDE KULLANILAN KATLAMALI KODLAMA TEKNİKLERİNİN KARŞILAŞTIRMALI BAŞARIM ANALİZİ

YÜKSEK LİSANS TEZİ

Ali ÇALHAN

Tezin Enstitüye Verildiği Tarih: 22 Aralık 2006

Tezin Savunulduğu Tarih : 11 Ocak 2007

Tez Danışmanı	Üye	Üye
Yrd. Doç. Dr. Celal ÇEKEN	Doç. Dr. İsmail ERTÜRK	Yrd. Doç. Dr. Ahmet Turan ÖZCERİT
(.....)	(.....)	(.....)

KOCAELİ, 2006

ÖNSÖZ VE TEŞEKKÜR

Kablosuz ortamın yüksek bit hata oranına sahip olması nedeniyle, veri iletiminde genellikle ileri hata düzeltme tekniklerinden olan katlamalı kodlama kullanır. Katlamalı kodlar özellikle gerçek zamanlı hata düzeltme uygulamalarında güçlü matematik yapıları sayesinde eşleniklerine göre üstünlük sağlarlar.

Bu tez çalışmasında; ileri hata düzeltme tekniklerinden Viterbi, SOVA ve Log-MAP kod çözme algoritmalarının karşılaştırmalı başarım analizi sunulmuştur. Ayrıca çalışma içerisinde bu algoritmalar, resim iletim uygulamasında test edilerek eğitim amaçlı olarak bir arayüz de hazırlanmıştır.

Çalışmalarım boyunca benden yardımlarını ve tavsiyelerini esirgemeyen değerli danışmanım Yrd. Doç. Dr. Celal ÇEKEN ve Doç. Dr. İsmail ERTÜRK'e teşekkür ederim.

İÇİNDEKİLER

SEMBOLLER

n	: Giriş bilgi dizisini oluşturan bit sayısı
k	: Kod kelimesini oluşturan bit sayısı
B	: Bant genişliği
SG	: Sinyal gücü
C	: Kanal kapasitesi
No	: tek taraflı gürültü güç yoğunluğu
Eb	: Ortalama bit enerjisi
Rb	: İletim bit oranı
Pr	: Hareketli birimde alınan sinyal gücü
Pt	: Vericiden iletilen sinyal gücü
Gt	: Anten kazancı
Gr	: Alıcıdaki anten kazancı
dtr	: Hareketli birim ile anten arasındaki mesafe
Ltr	: Kayıp faktörü
Φ	: İletilen sinyalin dalga boyu
Ae	: Fotometrik açıklık
cıh	: Işık hızı
F	: Taşıyıcı frekans
u	: Giriş bilgi dizisi
v	: Kod kelimesi
r	: Alınan kod dizisi
y	: Tahmini kod dizisi
h	: Kodlayıcıdaki hafıza sayısı
K	: Sınırlama uzunluğu
R	: Kod oranı
ç	: Giriş bilgi dizisi uzunluğu
P	: Olasılık
k1	: Küçük pozitif tamsayı
k2	: Küçük pozitif tamsayı
M	: Metrik
t	: zaman
S	: Kafes diyagram durumu
u’	: Tahmini bilgi dizisi
C1	: y fonksiyonunun terimi
C2	: y fonksiyonunun terimi
e	: exponansiyel
g	: Üreteç vektörü
MEM	: Saklama seviyesi
V	: Durum metriği
L	: Güvenilirlik
z	: Rasgele değişken
D	: Kaydıran yazmaç

σ	: Varyans
ort	: Ortalama
a	: Sönümleme genliği
m	: Kazanan ya da yarışan yol
Δ	: Son güvenilirlik değeri
CS	: Dairesel kaydıran yazmaç
I	: Serpiştirici
I^{-1}	: Düzenleyici
d	: veri biti
λ	: APP
Λ	: APP'lerin oranı
α	: İleri durum metriği
δ	: Dal metriği
β	: Geri durum metriği
f	: Geri durum metriği fonksiyonu
b	: Zamanda geri dönüş fonksiyonu
tu	: İletilen veri biti
tv	: İletilen eşlik biti
rx	: Alınan gürültülü veri biti
ry	: Alınan gürültülü eşlik biti
RX	: rx üzerinden değer alan rasgele değişken
p	: Olasılık yoğunluk fonksiyonu
A	: Sabit sayı
μ	: Düzeltme terimi
x	: Durum diyagramı giriş bilgi biti
s	: Durum diyagramı geçiş dizisi

Alt indisler

rout1	: Birinci çıkış
rout2	: İkinci çıkış
s	: Kazanan yol
c	: Yarışan yol
low	: en küçük seviye
K	: İletim kanalı
e	: Fazlalık bilgi
e1	: Birinci kod çözücüden gelen fazlalık bilgi
e2	: İkinci kod çözücüden gelen fazlalık bilgi
B	: Bit hatası
a	: a düğümü
b	: b düğümü
in	: Sıfır-Bir kararlı kod çözücü girişi
out	: Sıfır-Bir kararlı kod çözücü çıkışı

Kısaltmalar

BER	: Bit Error Rate (Bit Hata Oranı)
SNR	: Signal to Noise Ratio (Sinyal-gürültü Oranı)
FEC	: Forward Error Correcting (İleri Hata Düzeltme)

BCH : Bose Chaudury Hocquenquem
TCM : Trellis Coded Modulation (Kafes Kodlu Modülasyon)
MAP : Maximum A Posteriori (En Büyük Sonsallık)
SOVA : Soft Output Viterbi Algorithm (Yumuşak Çıkışlı Viterbi Algoritması)
Max Log-MAP: Maximum Logarithmic MAP (En büyük Logaritmik MAP
Log-MAP : Logarithmic MAP (Logaritmik MAP)
bps : bit per second (saniyedeki bit sayısı)
WAN : Wide Area Network (Geniş Alan Ağı)
AWGN : Additive White Gaussian Noise (Toplanır Beyaz Gaus Gürültüsü)
ARQ : Automatic Repeat Request (Otomatik Tekrar İsteme)
ACK : Acknowledge (Bilgi)
NAK : Negative Acknowledge (Olumsuz Bilgi)
MLDA : Maximum Likelihood Decoding Algorithm (En büyük Benzerlikli Kod çözme Algoritması
MLD : Maximum Likelihood Decoding (En büyük Benzerlikli Kod çözme)
ML : Maximum Likelihood (En büyük Benzerlik)
HDVA : Hard Decision Viterbi Algorithm (Sıfır-Bir Kararlı Viterbi Algoritması)
SDVA : Soft Decision Viterbi Algorithm (Yumuşak Kararlı Viterbi Algoritması)
BSC : Binary Symmetric Channel (İkili Simetrik Kanal)
RSC : Recursive Systematic Convolutional (Özyineli Sistematik Katlama)
APP : A posteriori Probability (Sonsal Olasılık)
LLR : Logaritmic Likelihood Ratio (Logaritmik Benzerlik Oranı)
GF : Galois Field (Galois Alanı)
GSM : Global System for Mobile Communications
Log : Logaritmik
IEEE: : Institute of Electrical and Electronics Engineers
IIR : Infinitive Impulsive Response
FIR : Finite Impulsive Response
ACS : Acknowledge
GHz : Giga Hertz (frekans birimi)

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

KABLOSUZ İLETİŞİMDE KULLANILAN KATLAMALI KODLAMA TEKNİKLERİNİN KARŞILAŞTIRMALI BAŞARIM ANALİZİ

Ali ÇALHAN

Anahtar Kelimeler: İleri Hata Düzeltme (FEC), Katlamalı kod, Viterbi, Turbo kod, SOVA, Log-MAP.

Özet: Son yıllarda özellikle hücresel, uydu ve bilgisayar haberleşmesinde çok büyük gelişmeler yaşanmaktadır. Haberleşme sistemlerindeki kablosuz ortamların bit hata oranı yüksek olduğundan dolayı aktarılan veriler üzerinde hata denetiminin yapılması zorunludur. Bu bağlamda sayısal haberleşme sistemlerinde bit hata oranını azaltmak ve sayısal bilgiyi girişim ve gürültülerden korumak için kanal kodlama kullanılmaktadır.

Kanal kodlamada çoğunlukla ileri hata düzeltme tekniklerinden biri olan katlamalı kodlar kullanılmaktadır. Katlamalı kodlar, güçlü matematiksel yapıya sahiptirler ve çoğunlukla gerçek zamanlı hata düzeltme uygulamalarında tercih edilmektedirler. Katlamalı kodlar için ana kod çözme stratejisi Viterbi algoritmasıdır. Katlamalı kodların geniş alanda kullanılması farklı kod çözme stratejilerinin geliştirilmesine neden olmuştur. Bu gelişmeler sayesinde Turbo kod adı verilen yeni bir hata düzeltme yöntemi tanımlanmıştır.

Bu tez çalışmasında, MATLAB yazılımı kullanılarak ileri hata düzeltme tekniklerinin temelini oluşturan Viterbi kod çözme algoritması ve Turbo kod çözme algoritmalarından Log-MAP ve SOVA algoritmaları sunulmaktadır. Bu tekniklerin karşılaştırmalı başarım analizi için örmek bir resim uygulaması gerçekleştirilmiştir. Elde edilen benzetim sonuçlarında, Log-MAP kod çözme algoritması özellikle artan SNR değerleri ile birlikte; Viterbi algoritmasına göre çok daha iyi BER başarımı sağlamaktadır. SOVA algoritması, yapısındaki iki SOVA kod çözücüsünden dolayı Viterbi algoritmasına göre iki kat daha karmaşıktır ve döngülü kod çözmenin bir sonucu olarak karmaşık hesaplama gerektirmektedir. Daha fazla döngü sayısı daha çok işlem karmaşıklığı demektir. Log-MAP algoritması, SOVA algoritmasından daha çok matematiksel işlem gerektirdiğinden aynı döngü sayısında, SOVA kod çözme algoritmasına göre daha iyi BER başarımı sağlamaktadır.

COMPARATIVE PERFORMANCE ANALYSIS OF CONVOLUTIONAL CODING TECHNIQUES FOR WIRELESS COMMUNICATIONS

Ali ÇALHAN

Keywords: Forward Error Correction (FEC), Convolutional code, Viterbi, Turbo code, SOVA, Log-MAP.

Abstract: Recently, there have been tremendous developments especially in the fields of cellular, satellite and wireless computer communications. In such systems, high bit error rate of the wireless medium requires employing error control codes on the data transferred. In this contex, channel coding is used in digital communication systems to reduce the rate of bit errors and to protect the digital information from noise and interference.

Convolutional codes are one of the most widely used forward error correction techniques for channel codes. They have strong mathematical structures and are mostly preferred for real time error correction applications. The main decoding strategy for convolutional codes is the Viterbi algorithm. Common use of convolutional codes has boosted development of different decoding schemes, resulting in a new error correcting method called Turbo code.

In this research study presented, Viterbi decoding algorithm that is the basis for Forward Error Correction (FEC) techniques, and Log-MAP and SOVA Turbo decoding algorithms are studied using MATLAB software. An example image transfer application has also been realized for comparative performance analysis of these techniques. The simulation results obtained show that the Log-MAP decoding algorithm achieves up to 100 times better BER performance especially for increasing SNR values than that of Viterbi algorithm. However, SOVA algorithm is more complicated than Viterbi algorithm since two SOVA decoders are employed and it requires complex calculations resulting from the iterative decoding algorithm utilized. The more the iteration number means the more processing complexity. Log-MAP algorithm has required more mathematical operations than SOVA. The studies have concluded Log-MAP decoding algorithm provides better BER performance compared to SOVA decoding algorithm for the same iteration number.

1. GİRİŞ

Kablosuz ortamlar, kablo kullanan eşleniklerinden farklı olarak hareketlilik, üretkenlik, düşük maliyet, kurulum kolaylığı ve ölçeklenebilirlik gibi birçok avantajı da beraberinde getirir. Bununla birlikte, kablosuz ortamın doğasından kaynaklanan bir takım sınırlamalar ve dezavantajlar bulunmaktadır. Kablosuz ortamda alıcı ile verici arasındaki iletişim kanalları çok çeşitlidir. Verici tarafından gönderilen sinyaller yansıma (reflection), kırılma (diffraction) ve dağılma (scattering) gibi etkiler nedeniyle alıcıya birçok kanalı kullanarak farklı güçlerde ve farklı zaman gecikmeleriyle ulaşabilirler (multipath) [1]. Ayrıca kablosuz ortamların bit hata oranı nispeten çok yüksektir (10^{-3} lere kadar yükselebilir). Bu sınırlamalar ve dezavantajlar kablosuz veri iletim başarımını olumsuz yönde etkilemektedir. Bu nedenle kablosuz veri iletiminde hata kontrolü zorunludur.

Sayısal veri iletimi ve depolama işlemleri sırasında başarım ölçütü olarak çoğunlukla bit hata oranı (BER, Bit Error Rate) kullanılır. BER genellikle, hatalı olarak iletilen bit sayısının toplam bit sayısına oranı olarak tanımlanmaktadır. Ortamda bulunan gürültü, sinyalin gücünü bastırarak sinyalde bozulmalara neden olur. Sinyal ile gürültü arasındaki ilişki SNR (Signal-to-Noise Ratio) ifadesiyle açıklanır. SNR, sinyal gücünün gürültü gücüne oranı olarak ifade edilir ve BER ile ters orantılıdır [2]. BER ne kadar küçükse SNR o kadar yüksektir ve kaliteli bir iletim sağlanmış olur.

Hata kontrolünde iki temel teknik kullanılmaktadır: iletim sisteminin tek yönlü olduğu durumlarda İleri Hata Düzeltme (FEC, Forward Error Correction) tekniği ve iletim sisteminin çift yönlü olduğu durumlarda Otomatik Tekrar Gönderme (ARQ, Automatic Repeat Request). Günümüzde kullanılan kodlanmış sistemlerin çoğu ileri hata düzeltme teknikleri yardımıyla işletilmektedir. Bit hata oranı çok yüksekse bozulan paketlerin yeniden gönderilmesi daha uygun olabilir ve bu durumlarda ARQ

tercih edilir. BER düşük ise bozulan paketi yeniden göndermek zaman kaybına neden olur, dolayısıyla FEC tercih edilir.

İleri hata düzeltme tekniğinde genel olarak iki tip kanal kodu vardır. Bunlar, blok kodlar ve katlamalı (convolutional) kodlardır. Blok kodlar çok fazla sonlu alan aritmetiği ve soyut matematik gerektirir. Bu kod çeşidi, hata bulma ve düzeltmede kullanılmaktadır. Blok kodlar, k bitlik bilgi bloğunu alır ve n bitlik kodlanmış bloğa çevirir. Önceden belirlenmiş kurallara göre, n-k adet fazlalık bit, k bitlik bilgi bitine eklenerek n bitlik kodlanmış bit haline getirilir. Genel olarak, (n,k) blok şeklinde ifade edilir. Genel blok kodları, Hamming kodlar, Golay kodlar, BCH (Bose Chaudury Hocquenquem) kodlar ve Reed Solomon kodlarıdır (ikili olmayan semboller kullanır). Blok kodları çözmenin ve k bitlik bilgiyi tahmin etmenin birçok yolu vardır.

Tez çalışmasına da konu olan katlamalı kodlar, pratik haberleşme sistemlerinde en çok kullanılan kanal kodlarıdır. Bunlar, farklı matematik yapıları ile geliştirilir ve gerçek zamanlı hata düzeltme tekniği kullanmaktadırlar. Katlamalı kodlar, tüm veri akışını bir tek kod kelimesine çevirir. Kodlanmış bitler, sadece o anki kodlayıcıya gelen k giriş bitine bağlı olmayıp daha önce kodlayıcıya giren bitlere de bağlıdır. Katlamalı kodlar için ana kod çözme stratejisi, Viterbi algoritması üzerine kurulmuştur. Katlamalı kodların yaygın olarak kullanılmasının nedeni, temel kodlama şeklinin geliştirilmesi ve genişlemesindeki ilerlemelerdir. Bu gelişmeler iki yeni kodlama şekli meydana getirmiştir. Bunlar TCM (Trellis Coded Modulation) ve Turbo kodlardır. TCM tekniği, kodlama, birleştirme ve modülasyonu tek işlemde yaparak fazlalık bitleri ekler. TCM tekniğinin tek avantajı, veri oranında bir azalma olmaması ya da bant genişliğinde bir genişleme olmamasıdır.

Diğer yeni kodlama şekli olan Turbo kodlar, çalışmamızda ayrıntılı olarak değerlendirilecektir ve yakın geçmişte en iyi hata düzeltme tekniği olarak gösterilmektedir. Bu hata düzeltme kodu, bilgiyi kanal üzerinden 0'a çok yakın hata ile iletebilmektedir [2]. Bu kod, iki katlamalı kodun birbirlerinden bir serpiştirici ile ayrılarak paralel bağlanması ile meydana gelmektedir. Turbo kodlar, döngülü kod

çözme algoritması sayesinde, Shannon[1] sınırına çok yakın bir başarım göstermektedir. Turbo kodlar ile geniş veri blokları kodlanacak olursa, 0.7 dB SNR değeri için 10^{-5} gibi çok küçük BER sonuçları elde edilebilir [3]. Turbo kodlarda, rasgele serpiştiricinin (random interleaver) kullanılmasından dolayı rasgele bir görünüm oluşmaktadır. Bununla beraber, kod çözme için yeterince kod yapısının bulunması bu kodları verimli yapmaktadır.

Turbo kodların öncelikli olarak iki kod çözme stratejisi vardır. Bunlar, en büyük sonsallık (MAP, Maximum A Posteriori) algoritması ve yumuşak çıkışlı Viterbi algoritması (SOVA, Soft Output Viterbi Algorithm)'dır. Hangi algoritmanın kullanılacağına bakılmadan, döngülü şekilde işlem yapılacaksa, aynı algoritmayı kullanan iki kod çözücü kullanılmalıdır. SOVA algoritması MAP algoritmasından daha az karmaşıktır ve her iki algoritma da karşılaştırılabilir başarım sonuçları sağlar. Ayrıca, SOVA algoritması Viterbi algoritmasının genişletilmiş şeklidir ve MAP algoritmasına göre daha kolay gerçekleştirilebilme avantajı vardır. MAP algoritması yapısından kaynaklanan hesaplama karmaşıklığına (doğrusal olmayan fonksiyonlar, çok sayıdaki toplama ve çarpım işlemleri) sahiptir. Bu nedenle MAP algoritmasından logaritmik alanda türetilen algoritmalar, MAP algoritmasının sayısal problemlerini ve hesaplama karmaşıklığını çözmektedir. Bununla birlikte, özellikle düşük SNR değerlerinde (Max-Log-MAP, Max fonksiyonu kullanmasından dolayı bu duruma bir örnektir), MAP algoritması kadar iyi sonuçlar verememektedir [4]. MAP algoritmasının bir diğer basitleştirilmiş şekli de aslında SOVA algoritmasıdır. Max-Log-MAP (Maximum Logarithmic MAP) algoritmasının düşük SNR değerlerindeki dezavantajını gidermek, MAP algoritmasına eşit bir şekilde sonuçlar üretmek ve MAP algoritmasının pratik uygulamalarda da kullanılmasını sağlamak amacıyla Log-MAP (Logarithmic MAP) algoritması geliştirilmiştir.

[1] 1948 yılında Shannon, gürültülü bir kanalda bilgi iletim oranında azaltma yapılmadan ve bu oranı istenilen düzeye getirerek bilgi kodlamayı göstermiştir [5]. Shannon limitine göre SNR≥-1.6 dB [6] olması gerekmektedir.

Bu tez çalışmasında kanal kodlama tekniklerinden katlamalı kodlar için kullanılan yumuşak çıkışlı Viterbi algoritması ve Turbo kod çözme algoritmalarından SOVA ile Log-MAP algoritmaları MATLAB yazılımı yardımıyla modellenerek örnek bir resim uygulaması için karşılaştırmalı başarım analizi sunulmaktadır.

1.1. Literatürde Yapılan Çalışmaların Özetleri

İleri hata düzeltme yönteminde yaygın olarak kullanılan katlamalı kodlar, Elias tarafından 1955 yılında blok kodlara alternatif olarak tanımlanmıştır. Bundan kısa bir süre sonra Wozencraft tarafından katlamalı kodlar için kullanılan ardışık kod çözme tekniği ifade edilmiş ve deneysel çalışmalar başlamıştır.

1963 yılında, Massey tarafından katlamalı kodlar için daha az verimli fakat gerçekleştirilmesi daha kolay olan eşik kod çözme tekniği geliştirmiştir. Bu ilerleme sayesinde katlamalı kodların pratik uygulamaları kablo ve radyo kanalları üzerinden sayısal iletim şeklinde gerçekleştirilmeye başlamıştır.

1967 yılında, Viterbi en büyük olasılıklı Viterbi kod çözme tekniği geliştirilmiştir. Viterbi kod çözme, ardışık kod çözme ile geliştirilerek uzak ve uydu haberleşmesinde 1970'li yılların başında kullanılmaya başlanmıştır. Bu süre içerisinde Omura, Viterbi algoritmasının, kod çözme grafiği üzerinde en kısa yolu bulma problemine karşı dinamik programlama çözümü olduğunu göstermiştir.

Katlamalı kodlar için ana kod çözme stratejisi, Viterbi algoritması üzerine kurulmuştur. Katlamalı kodların geniş olarak kullanılmasının nedeni, temel kodlama şeklinin geliştirilmesi ve genişlemesindeki ilerlemelerdir. Bu gelişmeler sayesinde Turbo kodlar geniş uygulama alanları bulmuştur.

Turbo kodlar ilk olarak 1993 yılında IEEE Uluslararası Haberleşme Konferansı'nda Berrou, Glavieux ve Thitimajshima tarafından tanıtılmıştır. 1993 yılında yapılan çalışma ile katlamalı kodların yeni bir sınıfı olan Turbo kodlar bulunmuştur. Bu çalışma ile iki adet özyineli sistematik katlamalı kodlayıcıdan oluşan Turbo

kodlayıcının bit hata olasılığı açısından başarımının Shannon sınırına çok yakın olduğu gösterilmiştir. Turbo kodların yapıtaşını oluşturan katlamalı kodlarda kullanılan Viterbi algoritmasının, verinin dizi hata olasılığını en aza indiren bir maksimum olasılıklı kod çözme tekniği olduğu bununla birlikte bit hata olasılığını en aza indirmesi gerekmediği ifade edilmiştir. Turbo kodlarda ise Viterbi algoritması yerine sonsal olasılık (APP, A Posteriori Probability) tekniğini kullanan Bahl Algoritması kullanılması gerektiği vurgulanmış ve sistematik katlamalı kodlar için değiştirilmiş Bahl kod çözme algoritması geliştirilmiştir [7].

2000 yılında Yan Wang, Chi-Ying Tsui, Roger S. K. Cheng tarafından yapılan çalışmada [8] Log-MAP algoritmasının karmaşıklığını azaltmayı sağlayan yöntemler geliştirilmiştir.

2001 yılında H. Peter, Y. Wu tarafından çalışmada [9] Turbo kod çözme algoritmalarının karmaşıklığı anlatılarak SOVA, Max Log-MAP ve Log-MAP algoritmalarının hesaplama karmaşıklıkları gösterilmiştir.

2003 yılında N. Ehtiati, M. R. Soleymani ve H. R. Sadjadpour yaptıkları çalışmalarında [10] çoklu Turbo kodlar için serpiştirici tasarlamışlardır.

2004 yılında M. Kouraichi, O. Ben Belghith tarafından yapılan çalışmada [11] SOVA algoritmasının kullanımı anlatılmıştır.

2004 yılında Xavier Jaspar ve Luc Vandendorpe tarafından yapılan çalışmada [12] Turbo kodlarda değişken uzunlukta kodlar için yeni bir döngülü kod çözme sunulmuştur.

S. Talakoub, L. Sabeti, B. Shahrrava ve M. Ahmadi tarafından yapılan çalışmada [13] basitleştirilmiş Log-MAP algoritması sunulmuştur.

2005 yılında Ali Ghrayeb tarafından yapılan çalışmada [14] depolama kanallarında SOVA tabanlı kod çözmedeki gelişmelerden bahsedilmiştir.

2006 yılında Bing Du tarafından yapılan çalışmada [15] kablosuz radyo kanalı üzerinden video yayımı uygulamasında FEC tekniklerini konu almıştır.
2006 yılında Chokri Chemak, Mohamed Salim [16] Bouhlel tarafından yapılan çalışmada kısa çerçeve boyutları için Shannon sınırına yakınsayan Turbo kodlardan bahsedilmiştir.

1.2. Tez Çalışmasının Amacı

Bu tez çalışmasının temel amacı kablosuz ortamda verinin kaynak ile hedef arasında en az hatayla iletilmesini sağlayacak ileri hata düzeltme tekniklerinin başarımını karşılaştırmalı olarak incelemektir. Bu doğrultuda aşağıdaki çalışmalar gerçekleştirilmiştir.

- İleri hata düzeltme tekniklerinin ve özel olarak katlamalı kodların incelenmesi.
- Katlamalı Kodlardan Viterbi ve Turbo Kodların Matlab yazılımı ile benzetim modellerinin gerçekleştirilmesi.
- Bu tekniklerin benzetimlerinin yapılarak karşılaştırmalı başarım değerlendirmelerinin yapılması.
- Matlab ile gerçekleştirilen modellerin örnek bir resim iletim uygulamasında kullanılarak başarımlarının incelenmesi.
- Eğitim amaçlı olarak kullanılabilecek bir yazılımın gerçekleştirilmesi.

1.3. Tez Çalışmasının Katkıları

Yukarıda verilen amaç ve hedefler doğrultusunda yapılan tez çalışmasının temel katkıları özetle şunlardır:

- İleri hata düzeltme teknikleri detaylı bir şekilde sunulmaktadır.
- Katlamalı kodların tarihsel gelişimi ve uygulama alanları ile kullanım amaçları açıklanmaktadır.
- Katlamalı kodlar için kullanılan ana kod çözme algoritması olan Viterbi kod çözme algoritması hakkında kapsamlı bilgi verilip uygulamaları sunulmaktadır.

- Turbo kodların ortaya çıkışı ve Turbo kodlara ait kod çözme teknikleri olan SOVA ve Log-MAP kod çözme algoritmaları hakkında detaylı bilgi verilmekte ve uygulamaları sunulmaktadır.
- Viterbi, SOVA ve Log-MAP algoritmalarına ait benzetim modelleri Matlab yazılımıyla gerçekleştirilmektedir.
- Katlamalı kodlardaki kod çözme algoritmalarının karşılaştırmalı başarım analizi sunulmaktadır.
- Örnek bir resim iletim uygulaması ile modelleri gerçekleştirilen tekniklerin başarımları incelenmektedir.
- Eğitim amaçlı olarak kullanılabilecek bir demonun gerçekleştirilmektedir.

1.4. Tez Organizasyonu

Tez çalışması sekiz ana başlıktan oluşmaktadır ve aşağıdaki şekilde organize edilmiştir.

Bölüm 2'de kablosuz haberleşmedeki genel kavramlar hakkında bilgi verilmekte ve kablosuz ortamın problemlerinden bahsedilmektedir.

Bölüm 3'de ileri hata düzeltme stratejileri kapsamlı ve detaylı bir şekilde incelenmektedir. Bu bağlamda ileri hata düzeltme stratejilerinden katlamalı kodlara değinilerek ve katlamalı kodlayıcının genel yapısı, kullanım ve gösterim şekillerinden bahsedilmektedir.

Bölüm 4'de, Viterbi kod çözme algoritmalarından olan sıfır-bir ve yumuşak kararlı Viterbi algoritmaları ile katlamalı kodların bu iki algoritmadaki başarım analizleri incelenmekte ve ayrıca benzetim modellerinden bahsedilmektedir. Kafes ve durum diyagramları hakkında kapsamlı bilgiler verilmektedir.

Bölüm 5'de, Turbo kodların gelişimi ve Turbo kodlayıcının yapısını oluşturan farklı katlamalı kodlayıcı yapılarından söz edilmektedir. Ayrıca Turbo kodlayıcı yapısında kullanılan serpiştirici yapısı ve çeşitlerinden bahsedilmekte ve kodların bağlanması hakkında bilgi verilmektedir.

Bölüm 6'da, Turbo kod çözme algoritma yapısı ve SOVA kod çözme algoritması hakkında kapsamlı bilgi verilip kullanılan matematiksel yapılardan söz edilmektedir. Aynı zamanda, SOVA algoritmasının benzetim modeli anlatılmaktadır.

Bölüm 7'de, MAP algoritmasından bahsedilmekte ve MAP algoritmasının yaygın olarak kullanılan türevi olan Log-MAP algoritması ve uygulaması sunulmaktadır.

Bölüm 8'de, tez kapsamında yapılan çalışmalardan elde edilen sonuçlar genel hatlarıyla değerlendirilmektedir.

2. KABLOSUZ HABERLEŞMEDE KULLANILAN GENEL KAVRAMLAR

2.1. Giriş

Elektronik haberleşme kanalında iletilen bilgi her zaman gürültüye maruz kalmaktadır ve bu nedenle sinyaller alıcıya hatalı bir şekilde iletilmektedir. Gürültü seviyesini düşürmek mümkün olmakla birlikte tamamen yok etmek olanaksızdır. Bu nedenle, gürültülü bir kanaldan bilgi iletilirken bilginin alıcıya yeterince doğru bir şekilde iletilmesi gerekmektedir.

Bu bölümde, kablosuz haberleşmede kullanılan genel kavramlar açıklanmaktadır. Ayrıca kablosuz ortamın problemlerinden söz edilmektedir.

2.2. Kanal Kapasitesi

Bilgi transferinin kalitesini belirleyen temel parametreler kanal bant genişliği (B) ve sinyal gücü (SG) dür. Bant genişliği, verilen bir zamanda kaynaktan hedefe ne kadar bilgi iletilebileceğinin ölçüsüdür ve ölçü birimi "bps"dir (bit per second, bit/saniye). Tüm haberleşme sistemleri için ortak olarak tanımlanan bu terim, sistemlerin kapasitesini ifade etmek için kullanılır. Tablo 2.1'de bant genişliğinin birimleri listelenmiştir.

Tablo 2.1:Bant genişliği birimleri

Bant Genişliği Birimi	**Kısa Ad**	**Eş Değeri**
Saniyedeki bit sayısı	bit/s	Temel bant genişliği birimi
Saniyedeki kilobit sayısı	Kbit/s	1 Kbit/s=10^3 bps
Saniyedeki megabit sayısı	Mbit/s	1 Mbit/s=10^6 bps
Saniyedeki megabit sayısı	Gbit/s	1 Gbit/s=10^9 bps

Bant genişliğinin fiziksel ve kullanılan teknolojilerden kaynaklanan sınırları vardır. Tablo 2.2'de bazı ortamların destekledikleri bant genişliği/uzaklık değerlerinin listesi verilmektedir [17]. Tablo 2.3 ise, farklı geniş alan ağ (WAN, Wide Area Network) servislerini ve bu servislerin bant genişliklerinin listesini göstermektedir [17].

Tablo 2.2: Bazı ortamların bant genişliği uzaklık değerleri

İletim Ortamı	**Bant Genişliği**	**Fiziksel Uzaklık**
50-Ohm Coaxial Kablo (Ethernet 10BASE2, ThinNet)	10–100 Mbit/s	185 m
50-Ohm Coaxial Kablo (Ethernet 10BASE, ThickNet)	10–100 Mbit/s	500 m
Category 5 UTP (Ethernet 10BASE-T)	10 Mbit/s	100 m
Category 5 UTP (Ethernet 10BASE-TX) (Fast ethernet)	100 Mbit/s	100 m
Multimode Fiber Optik 100BASE-FX	100 Mbit/s	2000 m
Singlemode Fiber Optik 1000BASE-LX	1000 Mbit/s	3000 m
Kablosuz Ortam	11 Mbit/s	Bir kaç yüz metre

Tablo 2.3: WAN servislerinin bant genişliği değerleri

WAN Servisi	**Kullanıcılar**	**Bant Genişliği**
Modem	Kişisel kullanıcılar	56 Kbit/s
ISDN	Küçük kuruluşlar	128 Kbit/s
Frame-Relay	Okullar, orta ölçekli kuruluşlar	56 Kbit/s-1,544 Mbit/s
T1	Daha büyük kuruluşlar	1,544 Mbit/s
T3	Daha büyük kuruluşlar	44,736 Mbit/s
E1	Büyük kuruluşlar	2,048 Mbit/s
E3	Büyük kuruluşlar	34,368 Mbit/s
STS-1 (OC-1)	Telefon şirketlerinin omurgaları	51,840 Mbit/s
STS-3 (OC-3)	Telefon şirketlerinin omurgaları	155,251 Mbit/s
STS-48 (OC-48)	Telefon şirketlerinin omurgaları	2,488320 Gbit/s

Kanal kodlama, iletilen verideki fazlalık bitler sayesinde sayısal veriyi hatalardan korur. Kanal kodları, hata bulma kodları olarak adlandırılan hata bulucu kodlar ve hata düzeltme kodları olarak adlandırılan hata bulucu ve düzeltici kodlardır.

1948 yılında Shannon, gürültülü bir kanalda bilgi iletim oranında azaltma yapılmadan ve bu oranı istenilen düzeye getirerek bilgi kodlamayı göstermiştir [5]. Shannon'un kanal kapasitesi AWGN (Additive White Gaussian Noise) kanalda şu şekilde ifade edilmektedir.

$$C=B\log_2\left(1+\frac{Pr}{NoB}\right)=B\log_2\left(1+\frac{SG}{No}\right) \tag{2.1}$$

Bu formülde C, kanal kapasitesini göstermektedir ve birimi bps (bit per second) dir. B, iletim bant genişliği (Hz), Pr alınan sinyal gücü (watt), No, tek taraflı gürültü güç yoğunluğu (watts/Hz) dur. Alınan güç,

$$Pr=EbRb \tag{2.2}$$

şeklinde ifade edilir. Burada Eb ortalama bit enerjisi ve Rb iletim bit oranını göstermektedir. (2.1) denklemi iletim bant genişliği ile normalize edildiğinde,

$$\frac{C}{B}=\log_2(1+\frac{Eb}{No}\frac{Rb}{B}) \tag{2.3}$$

elde edilir. Burada C/B oranı bant genişliği verimliliğini göstermektedir. Hata bulma ve düzeltme tekniklerinin temel amacı kablosuz iletim başarımını geliştirmek için veriye eklenen fazlalık bitleri tanımlamaktır. Fazlalık bitleri tanımlamak, veri oranının artmasına ve böylece bant genişliği ihtiyacının da artmasına sebep olur. Bant genişliği verimindeki azalma yüksek SNR değerlerinde giderilebilirken, düşük SNR değerlerinde, BER değerinde artış meydana getirir. Shannon limitine göre SNR≥-1.6 dB [6] olması gerekmektedir.

Haberleşmede başarım ölçütünü hata oranı belirler. Bu başarım ölçütüne BER adı verilir. Genel olarak BER, hataların toplam bit sayısına oranı olarak tanımlanır. Güçlü bir sinyal ve gürültüsüz bir ortam ile bit hata oranı önemsenmeyecek kadar düşüktür. Gürültünün ortamda artmasıyla bit hata oranı da artar. Gürültü, sinyalin gücünü bastırarak, sinyalde bozulmalara sebep olur. Bu şekilde sinyal ve gürültü arasındaki ilişki SNR ifadesiyle açıklanmaktadır.

$$(SNR)_{dB}=10\log_{10}SNR \tag{2.4}$$

2.3. Kablosuz Ortamın Problemleri

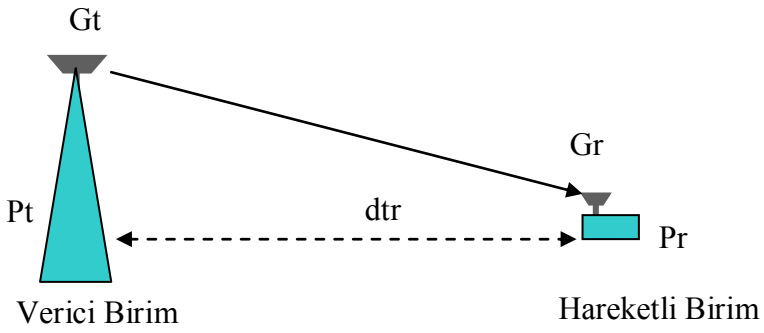
Kablosuz ortamlarda ağ bileşenleri birbirleriyle haberleşmek için radyo frekans teknolojilerini kullanmaktadır. Kablosuz ortamlar, kablo kullanan eşleniklerinden farklı olarak hareketlilik, üretkenlik, düşük maliyet, kurulum kolaylığı ve ölçeklenebilirlik gibi birçok avantajı da beraberinde getirir. Bununla birlikte, kablosuz ortamın doğasından kaynaklanan bir takım sınırlamalara ve dezavantajlara da sahiptir.

2.3.1. Yol kayıpları (Path loss)

Kablosuz ortamda alıcı ile verici arasındaki iletişim kanalları çok çeşitlidir. Verici tarafından gönderilen sinyaller yansıma (reflection), kırılma (diffraction) ve dağılma (scattering) gibi etkiler nedeniyle alıcıya birçok kanalı kullanarak farklı güçlerde ve farklı zaman gecikmeleriyle ulaşabilirler (multipath). Bu ise, ortalama gücün değişmesine neden olmaktadır.

Kablosuz haberleşme sistemlerinin, kullanıcılar değişik hızlarda hareket ederlerken ve değişik coğrafi bölgelerde iken servis vermeleri gerekecektir. Gezgin kullanıcıların bir erişim noktası bölgesinden diğerine geçerken verilen servisler kesilmemelidir. Bu nedenle, özel hareketlilik destek fonksiyonlarının yerine getirilmesi zorunludur. Bununla birlikte, gezgin kullanıcıların hızı nedeniyle gönderilen sinyaller ve bileşenleri (multipath) alıcıya gerçek frekansından farklı olarak ulaşacaktır (Doppler Shift).

Sinyal gücünün zayıflamasındaki en önemli temel çevresel etken, sinyalin alıcı ile verici arasında izlediği yolda sinyal gücündeki kayıplardır. Örneğin, yönsüz (non-directional) ya da eşyönlü (isotropic) antenler ile sinyal küresel dalga şeklinde yayılmaktadır. Sinyal gücü, kürenin yüzey alanı ile ters orantılıdır ve bu yüzden alıcı ve verici arasındaki mesafe ile ters orantılıdır. Bu boşluktaki kayba ek olarak atmosferik soğrulma (absorption) ve bitki örtüsünden dolayı çeşitli kayıplar meydana gelmektedir. 2 GHz dolaylarındaki sinyaller için şiddetli yağmurlar çok etkili olurken ağaçlandırılmış alanlar daha büyük kayıplara neden olmaktadır. Ağaçlandırılmış alanlarda kayba neden olan ağaç yapraklarıdır. Sinyal gücü, ağaçlandırılmış alanlarda 5–6 dB civarında olmak zorundadır fakat bu sinyal gücü kentsel alanlarda daha düşük olabilmektedir.



Şekil 2.1: Alıcı ve verici arasındaki ilişki

Boşlukta (free-space) yapılan iletim sırasında, alıcı, verici ve aralarındaki mesafe arasındaki ilişki şekil 2.1'de gösterilmiş olup aşağıdaki şekilde ifade edilir.

$$Pr=PtGt\frac{\Phi^2}{(4\pi)^2 dtr^2 Ltr}Gr \qquad (2.5)$$

Burada Pr hareketli birimde alınan sinyalin gücünü, Pt vericiden iletilen sinyalin gücünü, Gt anten kazancını, Φ iletilen sinyalin dalga boyunu, dtr hareketli birim ile anten arasındaki mesafeyi, Ltr kayıp faktörünü (sinyal yayılımı ile bağlantılı değildir, iletim hattı ve filtre kayıplarıdır), Gr alıcıdaki anten kazancını ifade etmektedir.
Anten kazancı şu şekilde ifade edilir:

$$Gt=\frac{4\pi Ae}{\Phi^2} \qquad \Phi=\frac{cıh}{F} \tag{2.6}$$

Ae fotometrik açıklığı, cıh ışık hızı (m/s), F taşıyıcı frekansını (Hz) göstermektedir. Kapalı alanlarda, duvar ve tabanlar sinyal için engel teşkil ederler ve bu da yol kayıplarına neden olmaktadır. Kapalı alanda bulunan engellerin yapısı da sinyal gücünde kayıplara sebep olmaktadır. Aşağıdaki tabloda engel yapıları ve sinyal güç kayıpları gösterilmiştir.

Tablo 2.4: Tipik engeller ve neden oldukları kayıplar (dB)

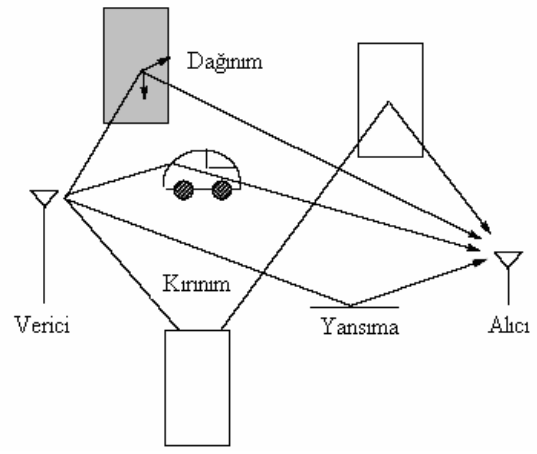
Engel Yapısı	**Kayıp (dB)**
Kumaş	1.4
Çift tahtalı	3.4
Folyo yalıtımı	3.9
Beton	13
Alüminyum kaplama	20.4
Bütün metaller	26

2.3.2. Çokyollu sönümleme (Multipath fading), gölgeleme (shadowing), yansıma (reflection), kırılma (diffraction) ve dağınım (scattering)

Gölgeleme (shadowing), alıcı ile verici arasında engeller nedeniyle sinyal gücünün soğurulması (absorption) olarak tanımlanmaktadır. Aradaki engeller sinyalin tüm gücünü soğurursa sinyal bloke edilmiş olur. Yol kayıplarından (path loss) dolayı oluşan sinyal gücündeki değişim geniş mesafelerde (100–1000 m) meydana gelmektedir. Gölgelemeden (shadowing) dolayı oluşan sinyal gücündeki değişim, alıcı ile verici arasındaki engellerin boyutu ile orantılıdır. Bu değişim açık alanlarda 10–100 m de, kapalı alanlarda ise daha az mesafelerde gerçekleşir. Yol kayıplarından ve gölgelemeden dolayı meydana gelen değişim geniş ölçekli yayılım etkileri (large scale propagation effects) ya da yerel ortalama zayıflama (local mean attenuation) olarak isimlendirilmektedir.

Sinyal yayılımı önündeki engeller yansımalara (reflection) neden olmaktadır. Bununla birlikte sıra halindeki engeller (örneğin ağaçlar) kırılmaya (diffraction) neden olurlar. Yansımaların bir avantajı, alıcı ve verici arasındaki büyük engellerden geçemeyen sinyaller küçük engellerden yansıyarak alıcıya ulaşmasıdır. Bununla birlikte iki büyük dezavantajı da vardır. Bunlar, yansımalardan dolayı oluşan sinyal gecikmeleri ile engellerin ve antenlerin hareketli olmasından dolayı meydana gelen doppler kaymalarından (Doppler shifts) oluşan rasgele frekans modülasyonudur. Ayrıca sinyaller engellere çarparak dağınık yansımalar da (scattering) yapabilmektedirler.

Sinyaller, yansımalar (reflection), kırınımlar (diffraction) ya da dağınımlar (scattering) sonucunda çoğullanır. Bu çoğullama, çokyollu (multipath) sinyal bileşenleri olarak adlandırılır. Bu sinyal bileşenleri alıcıda, sinyal gücünde zayıflama, zaman gecikmesi ve görüş hattı (line of sight) sinyal yolundan faz ya da frekans kaymalarına sebep olmaktadırlar. Çokyollu olma, kısa mesafelerde sinyalde değişimlere neden olur. Bu değişim küçük ölçekli yayılım etkileri (small scale propagation effects) ya da sönümleme (multipath fading) olarak adlandırılır. Aşağıdaki şekilde yansıma, kırınım ve dağınım gösterilmiştir.



Şekil 2.2: Yansıma, kırınım ve dağınım

2.4. Sonuç

Bilgi transferindeki temel parametreler bant genişliği ve sinyal gücüdür. Bant genişliğinin fiziksel ortamdan ve kullanılan teknolojilerden kaynaklanan sınırları vardır. Ayrıca kablosuz haberleşmede başlıca yol kayıpları, çok yollu sönümleme, gölgeleme, yansıma, kırılma, dağınım olarak sıralanabilecek temel problemler nedeniyle bilgi transferinde hatalar meydana gelebilmektedir. Kanal kodlama, iletilen verideki fazlalık bitler sayesinde sayısal veriyi hatalardan korur. Kanal kodları, hata bulma kodları olarak adlandırılan hata bulucu kodlar ve hata düzeltme kodları olarak adlandırılan hata bulucu ve düzeltici kodlardır.

3. HATA KONTROL STRATEJİLERİ

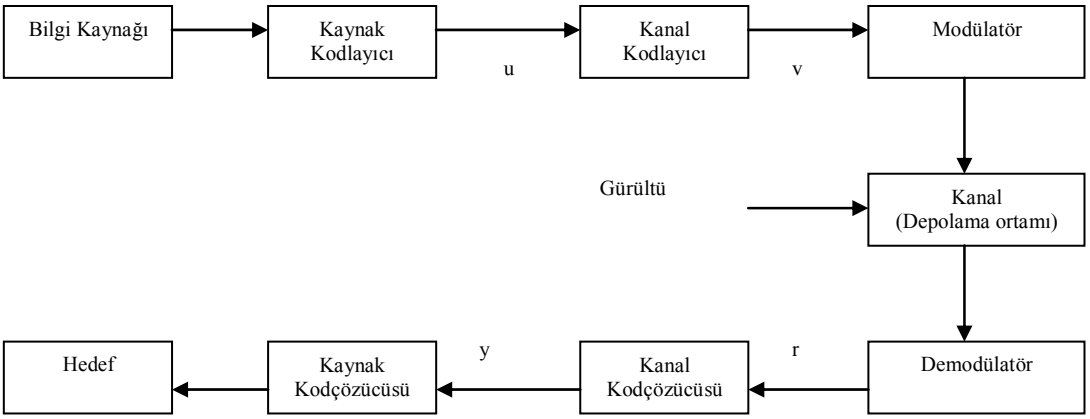
3.1. Giriş

Sayısal veri iletimi ve depolama sistemleri için verinin güvenilir ve doğru bir şekilde iletilmesi veya depolanması gerekmektedir. Buna askeri, resmi ve özel alanlarda sayısal bilginin değiştirilmesi, işlenmesi ve depolanması için büyük ölçekli ve yüksek hızlı veri ağlarında daha çok ihtiyaç duyulmaktadır. Bu gereksinimin karşılanması için haberleşme ve bilgisayar teknolojilerinin birleştirilmesi gerekmektedir. Burada önemli olan, iletim veya depolama sırasında oluşacak hataların kontrol edilmesi ve hatalı bilginin tekrar üretilmesidir. 1948 yılında, Shannon gürültülü bir kanalda bilgi iletim oranında azaltma yapılmadan ve bu oranı istenilen düzeye getirerek bilgi kodlamayı göstermiştir [5]. Hata kontrolü ileri hata düzeltme teknikleri sayesinde gerçekleştirilmektedir. En yaygın olarak kullanılan ileri hata düzeltme tekniği katlamalı kodlardır.

Bu bölümde iletim ve depolama sistemlerinin genel yapısından söz edilip ileri hata düzeltme stratejileri kapsamlı ve detaylı bir şekilde incelenmektedir. Bunun yanında ileri hata düzeltme stratejilerinden katlamalı kodlar ve katlamalı kodlayıcı ele alınıp genel yapısı, kullanım ve gösterim şekillerinden bahsedilmektedir.

3.2. İletim ve Depolama Sistemlerinin Genel Yapısı

Aşağıdaki şekilde tipik bir veri iletimi veya depolamasına ait blok diyagram gösterilmiştir.



Şekil 3.1: Tipik veri iletimi veya depolama sistemi blok diyagramı

Şekil 3.1'de bilgi kaynağı bir insan ya da bilgisayar, kaynak çıkışı, sürekli dalga şekli ya da ayrık semboller dizisidir. Kaynak kodlayıcısı, kanal çıkışını bilgi dizisi (u) ile ifade edilen ikili sayı sistemine çevirir. Kanal kodlayıcısı, bilgi dizisini (u) kod kelimesi (code word) denilen ayrık kodlanmış diziye (v) çevirmektedir. Kanal kodlayıcı kullanılmasının sebebi, gürültülü ortamdan geçecek bilgi dizisinin hatalardan korunması ve alıcıya en az hata ile iletmektir. Ayrık semboller, fiziksel ortamda veri iletimi ve sayısal veri depolama ortamında veri depolama için uygun değildir. Modülatör, kanal kodlayıcının her çıkış sembolünü iletim veya depolama için uygun olan dalga şekline çevirir. Bu dalga şekli iletim kanalına veya depolama ortamına girince gürültü tarafından bozulur.

Tipik iletim kanalları, telefon hatları, yüksek frekanslı radyo bağları, mikro dalga bağları ve uydu bağları gibi kanallardır. Tipik depolama ortamları, yarı iletken hafızalar, manyetik teypler ve optik hafıza birimleri gibi ortamlardır. Bu kanal ve ortamların her biri çeşitli bozukluklara maruz kalmaktadır. Telefon hatlarında, anahtarlama etki gürültüsü, ısı gürültüsü, diğer hatlardan gelen çapraz girişim gürültüsü ya da ışıklandırma gürültüsü meydana gelmektedir. Manyetik teyplerde, yüzey bozukluğu bir gürültü olarak nitelendirilebilir. Demodülatör ya da okuma

birimi, o andaki alınan her dalga şeklini alarak ayrık ya da sürekli forma dönüştürür. Demodülatörün çıkış dizileri, kodlanmış dizi (v) ye tekabül eder ve alınan dizi (r) olarak isimlendirilir.

Kanal kod çözücüsü, alınan diziyi ikili sayı sistemine göre düzenlenmiş tahmini diziye (y) çevirir. Kod çözme stratejisi, kanal kodlama ve kanalın gürültü karakteristiğine (ya da depolama ortamına) bağlıdır. İdeal olarak, tahmini dizi y, bilgi dizisi u'nun aynısı olmaktadır fakat iletim kanalındaki gürültü dolayısıyla kod çözme hataları meydana gelebilmektedir. Kanal kod çözücüsü tahmini diziyi (y), tahmini kaynak çıkışına çevirir ve bu çıkışı hedefe iletir.

Kaynak ve hedef arasında geçen bu iletim (depolama) işleminde asıl amaç, bilgiyi mümkün olduğu kadar hızlı bir biçimde iletmek, kanal kodlayıcı çıkışında bilgiyi tekrar üretebilmek, kodlayıcı ve kod çözücüyü uygun şekilde gerçekleştirebilmektir.

3.3. FEC ve ARQ

Şekil 3.1'de iletim (veya depolama) işlemi tek yönlü olarak gösterilmiştir. İletim (ya da depolama) işlemi, vericiden alıcıya doğru tek yönde olmaktadır. Tek yönlü iletim sistemi için hata kontrolü, ileri hata düzeltme (FEC, Forward Error Correction) tekniği kullanılarak yapılmaktadır. Bu teknik, hata düzeltme kodlarını kullanarak alıcıda meydana gelen hataları düzeltmektedir. Günümüzde kullanılan kodlanmış sistemlerin çoğu ileri hata düzeltme tekniklerini kullanmaktadır.

Bazı durumlarda iletim sistemi iki yönlü olmaktadır. Bu tip sistemlerde bilgi iki yönde de iletilmektedir. Alıcı aynı zamanda verici görevini yapmaktadır ve tersi de geçerlidir. İki yönlü iletim sistemlerine telefon hatları ve bazı uydu haberleşme sistemleri örnek verilebilir. İki yönlü iletim sistemlerinde hata kontrolü, otomatik yeniden gönderme (ARQ, Automatic Repeat Request) adı verilen hata bulma ve bilgiyi yeniden iletme şeklinde olmaktadır. Bu teknikte hata alıcıda bulunduktan sonra vericiye hatalı bilginin tekrar gönderilme isteği iletilir ve bu işlem bilgi doğru iletilinceye kadar devam eder.

İki tip otomatik yeniden gönderme (ARQ) tekniği vardır. Bunlar dur ve bekle (stop and wait) otomatik yeniden gönderme ve sürekli (continuous) otomatik yeniden gönderme teknikleridir. Dur ve bekle otomatik yeniden gönderme tekniğinde, verici alıcıya bir kod kelimesi gönderir ve alıcıdan pozitif (ACK, Acknowledge) ya da negatif (NAK, Negative Acknowledge) onaylama için bekler. Eğer alıcıdan pozitif bir onaylama dönerse verici sıradaki kod kelimesini gönderir. Alıcıdan negatif onaylama dönerse gönderilen bilgide hata vardır demektir ve verici aynı bilgiyi tekrar gönderir.

Sürekli otomatik yeniden gönderme tekniğinde, verici kod kelimelerini alıcıya sürekli şekilde gönderir ve sürekli olarak onaylamaları alır. Eğer negatif onaylama alınırsa verici tekrar iletime başlar. Hatalı kod kelimelerini yedeklenir ve iletim sonrasında hatalı kod kelimeler tekrar gönderilir. Bu işlem go-back-N otomatik yeniden gönderme olarak adlandırılır. Bunun yanında verici, sadece hatalı kod kelimelerini gönderebilir bu işleme de selective-repeat otomatik yeniden gönderme tekniği adı verilir. Selective-repeat otomatik yeniden gönderme tekniği go-back-N otomatik yeniden gönderme tekniğinden daha kullanışlı olmakla birlikte daha çok işleme (logic) ve tamponlamaya (buffering) gerek duymaktadır. Sürekli otomatik yeniden gönderme tekniği, dur ve bekle otomatik yeniden gönderme tekniğinden daha verimlidir fakat maliyeti fazladır. Uydu haberleşme sisteminde iletim oranı yüksek ve gidiş-geliş gecikmesi (round trip delay) yüksek olduğundan sürekli otomatik yeniden gönderme tekniği kullanılır. Dur ve bekle otomatik yeniden gönderme tekniği yarı çift yönlü (half duplex) kanallar, sürekli otomatik yeniden gönderme tekniği tam çift yönlü (full duplex) kanallar için tasarlanmıştır.

ARQ tekniğinin FEC tekniğine üstünlüğü daha basit kod çözme işlemi gerçekleştirmesidir. Bununla birlikte ARQ tekniği sadece hata meydana geldiğinde tekrar iletim yapmaktadır. FEC tekniği, düşük sabit gecikme sağlayabildiği için özellikle gerçek zamanlı trafikler için uygundur. Hata düzeltme kodları hata örneklerine bağlıdır ve en kötü durum için tasarlandığında iletim hızının azalmasına neden olmaktadır. Bit hata oranı düşük olan ortamlar için FEC yönteminin kullanılması daha uygundur. ARQ ise hatalı paketlerin yeniden gönderilmesi prensibine göre çalışır. Bu yöntemde sadece hatalı olarak alınan paketler yeniden

gönderilir. Değişken gecikmeli iletim sağladığından gerçek zamanlı uygulamalar için uygun değildir. Bit hata oranı yüksek olan ortamlar için ARQ yöntemi nispeten daha iyi sonuç vermektedir [18].

3.4. İleri Hata Düzeltme (FEC) Tekniği

Genel olarak blok kodlama ve katlamalı (convolutional) kodlama olarak iki tip ileri hata düzeltme tekniği kullanılmaktadır. Blok kodlamada bilgi dizisi, her birinde k bilgi biti bulunan bloklara ayrılır. Her mesaj bloğu ikili sayı sistemine göre, $u = (u_1, u_2, ..., u_k)$ şeklinde hazırlanır ve mesaj olarak adlandırılır. 2^k adet olası farklı mesaj vardır. Kodlayıcı, her bir u mesaj bloğunu kod kelimesi (code word) adı verilen ve $v = (v_1, v_2, ..., v_n)$ şeklindeki ayrık sembollere çevirir. Böylece farklı olası 2^k adet mesaja karşın 2^n adet olası kod kelimesi kodlayıcı çıkışlarında elde edilir. Bu şekilde ifade edilen blok kod, (n, k) şeklinde gösterilir. Burada $R = k/n$, kod oranını ifade eder ($k \leq n$ ya da $R \leq 1$). Kodlayıcıya gelen k bitlik giriş mesajına karşın çıkışta n bitlik çıkış kod kelimesi elde edilir. Blok kodlamada kodlayıcı hafızasızdır ve birleşik mantık devresi ile elde edilir. Her kod kelimesinde, n-k adet fazlalık (redundant) bit bulunmaktadır. Bu fazlalık bitler, kanal gürültüsü nedeniyle meydana gelen olası bit hatalarını yok etmek için kullanılır. Kod oranı sabit tutularak ve mesaj blok uzunluğu arttırılarak daha çok fazlalık bit eklenebilir.

Katlamalı (convolutional) kodlayıcı, kodlayıcıya gelen k bit uzunluğundaki bilgi bloklarını (u), kodlayarak n adet sembolden oluşan bloklar halinde kod kelimelerine (v) dönüştürür. Her kodlanmış blok sadece o anki k bit uzunluğundaki mesaj bloklarına bağlı olarak değil, kodlayıcıya gelen önceki h adet mesaj bloklarına da bağlı olarak oluşturulmaktadır. Bu yüzden h adet hafızaya sahiptir denilir. Bu şekilde ifade edilen katlamalı kodlayıcı, (n, k, h) olarak gösterilir. Burada $R = k/n$, kod oranını ifade eder. Katlamalı kodlayıcı bir hafızaya sahip olduğu için ardışık mantık devresi ile gerçekleştirilir.

İkili katlamalı kodda, kanal gürültüsünü yok etmeyi sağlayan fazlalık bitler bilgi dizisine eklenir ($k < n$ ya da $R < 1$). Katlamalı kodlayıcıda asıl problem, iletim

gürültülü bir kanaldan yapılırken doğru iletimin yapılabilmesi için kodlayıcı hafızasının nasıl kullanılacağıdır. Katlamalı kodlar hafızaya sahip olduğundan blok kodlara göre daha kullanışlıdır.

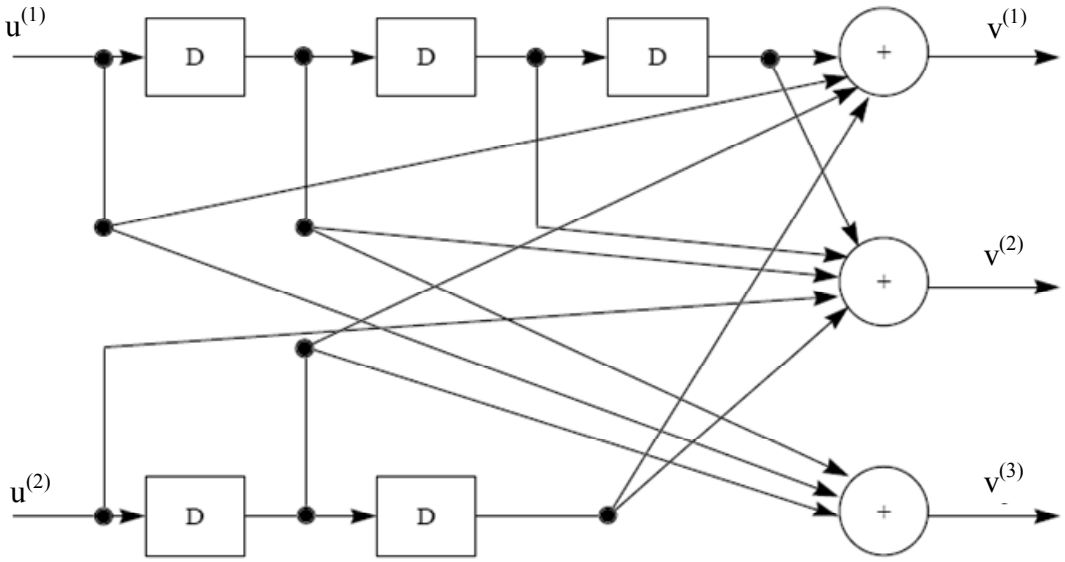
3.5. Katlamalı Kodlar

Katlamalı kodlar, ikili hata düzeltme kodları şeklinde 1955 yılında blok kodlara alternatif olarak Elias tarafından gerçekleştirilmiştir [19]. Elias, kablosuz haberleşmede (IMT–2000, GSM, IS–95), sayısal karasal ve uydu haberleşmesinde, yayın sistemlerinde ve uzay haberleşme sistemleri gibi daha birçok alanda kullanılan katlamalı kodları, katlamalı eşlik denetim sembol kodları (convolutional parity check symbols codes) olarak tanımlamıştır.

Wozencraft tarafından katlamalı kodlar için ardışık kod çözme işlemi (sequential decoding) esas alınarak çalışmalar yapılmıştır. 1963 yılında, Massey daha az verimli fakat basit bir kod çözme yöntemi kullanan eşik kod çözme (threshold decoding) yöntemini geliştirmiştir. Bu ilerleme, kablo ve radyo kanalları üzerinden yapılan sayısal iletim uygulamaları gerçekleştirilmesine neden olmuştur. 1967 yılında, Viterbi, en büyük olabilirlik kod çözme (MLD, Maximum Likelihood Decoding) tekniğini geliştirerek az sayıda kodlayıcı hafızası kullanarak işletimi kolay bir kod çözme algoritmasının doğmasına öncülük etmiştir. Ardışık kod çözme tekniğinin geliştirilmiş şekli olan Viterbi kod çözme tekniği 1970'li yılların başlarında uzak uzay (deep space) ve uydu haberleşmesinde kullanılmıştır.

3.6. Katlamalı Kodlayıcı Yapısı

Bir katlamalı kod, doğrusal kaydıran yazmaçlar (shift register) kullanılarak, veri akışına fazlalık bitlerin eklenmesi şeklinde ifade edilebilir. Şekil 3.2'de bir katlamalı kodlayıcı devresi gösterilmektedir.

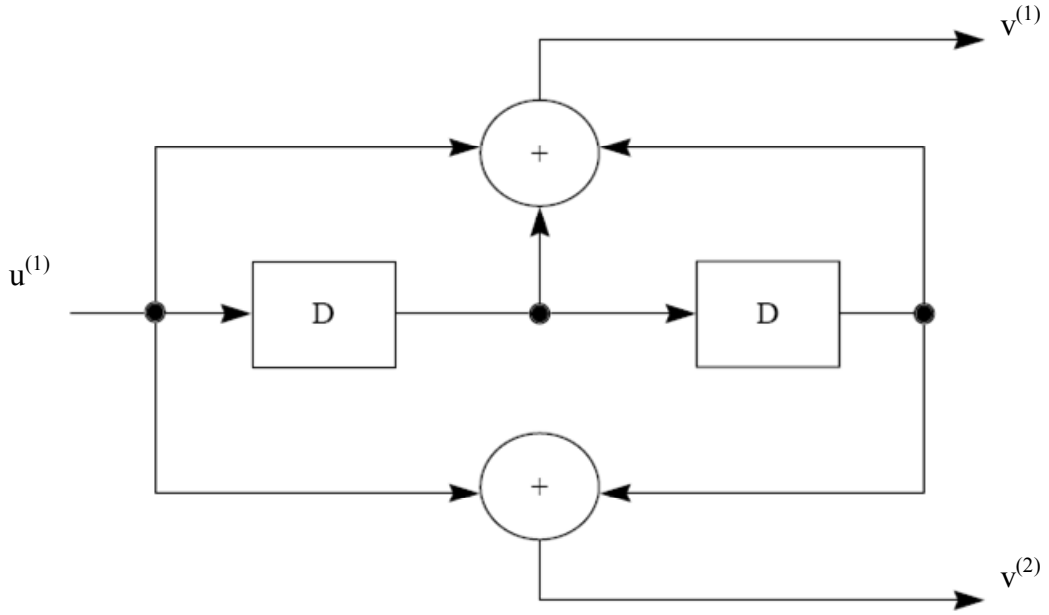


Şekil 3.2: $u^{(i)}$ 'nin giriş bilgi akışı ve $v^{(i)}$ 'nin kodlanmış çıkış bit akışı olarak gösterilen katlamalı kodlayıcı örneği [20]

Bilgi bitleri kaydıran yazmaçlara gönderilerek, giriş bitleri ve kaydıran yazmaçlarının içerikleri mod 2 işlemine sokularak kodlanmış çıkış bitleri elde edilir. Bir katlamalı kod için kod oranı, R şu şekilde tanımlanır.

$$R = \frac{k}{n} \tag{3.1}$$

Burada k, paralel giriş bilgi bitlerinin sayısı ve n, paralel kodlanmış çıkış bitlerinin sayısıdır. Sınırlama uzunluğu K, K=h+1 'e eşittir. h, kodlayıcı yapısında aynı sırada bulunan en fazla kaydıran yazmaç sayısıdır. Kaydıran yazmaçlar, katlamalı kodlayıcının durum bilgilerini saklarlar ve sınırlama boyu, her bir giriş bitinin etkileyeceği çıkışlarla ilgilidir. Şekil 3.2'de gösterilen kodlayıcıda, kod oranı 2/3, en büyük hafıza boyutu h=3 ve sınırlama boyu K=4 dür. Bir katlamalı kod, çeşitli kod oranları ve sınırlama boyları kullanılarak çok karmaşık hale getirilebilir. Şekil 3.3'de basit bir katlamalı kodlayıcı gösterilmiştir.



Şekil 3.3: k=1, n=2, R=1/2, h=2 ve K=3 değerlikli katlamalı kodlayıcı devresi

3.6.1. Katlamalı kodlayıcı gösterimi

Katlamalı kodlayıcı tasarımı birkaç farklı yolla gösterilebilir. Bunlar;

1. Üreteç gösterimi
2. Ağaç diyagramı gösterimi
3. Durum diyagramı gösterimi
4. Kafes diyagram gösterimi

şeklinde olmaktadır.

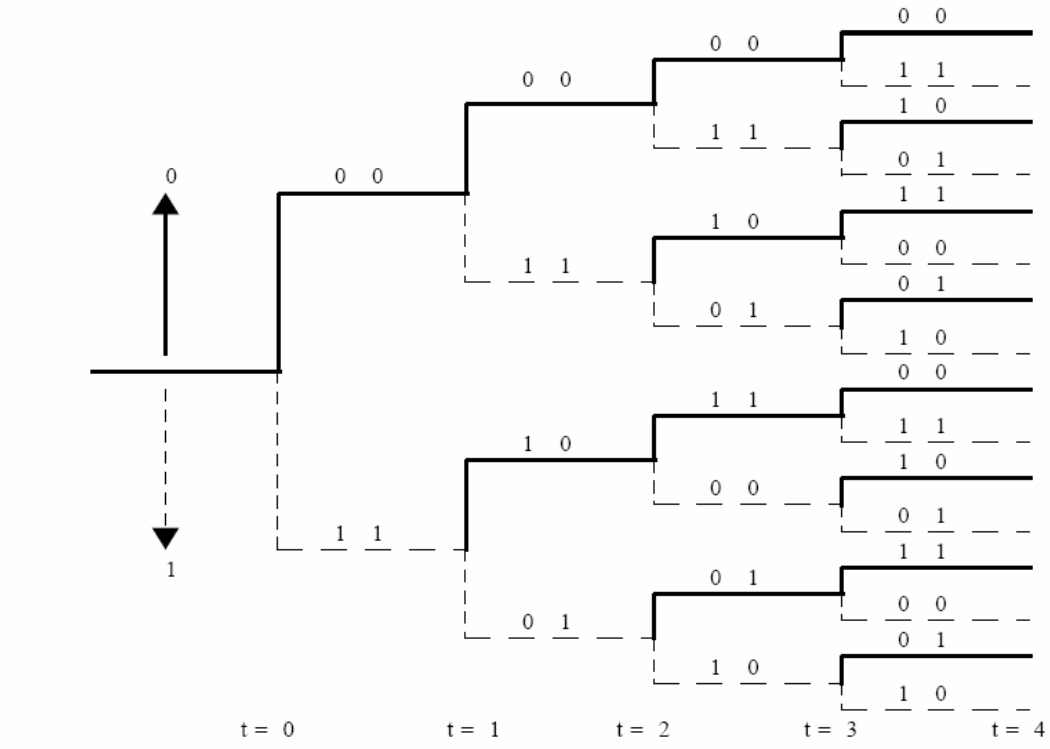
3.6.1.1. Üreteç gösterimi

Üreteç gösterimi, kaydıran yazmaçların modül 2 toplayıcılarına donanımsal bağlantısını göstermektedir. Bir üreteç vektörü, kaydıran yazmaçların bağlantılarının konumlarını gösterir. 1 bilgisi, bağlantının olduğunu, 0 bilgisi bağlantının olmadığını belirtir. Şekil 3.3 'deki katlamalı kodlayıcının üreteç vektörleri $g_1 = [111]$ ve $g_2 = [101]$ dir. Üreteç vektörleri sekizli sayı sisteminde ifade edilirler. Şekil 3.3 için bu ifade g_1=7 ve g_2=5 şeklindedir. Şekil 3.3'de görülebileceği üzere, kodlayıcının üst kısmındaki özel veya (EXOR) kapısına tüm kaydıran yazmaçlar bağlıdır. Yani tüm bağlantılar 1 konumundadır. Kodlayıcının alt kısmındaki özel veya kapısına birinci

kaydıran yazmacın çıkışı bağlı değildir yani 0 konumundadır. Bu yüzden ikinci üreteç vektörünün ikinci elemanı 0'dır. Üreteç vektörleri mantıksal işlemler yapılarak da bulunabilir. Katlamalı kodlayıcının girişine dürtü yanıtı (impulse response) uygulanarak üreteç vektörleri elde edilebilmektedir.

3.6.1.2. Ağaç diyagramı gösterimi

Ağaç diyagramı katlamalı kodlayıcı için olası tüm bilgileri ve kodlanmış dizileri gösterir. Şekil 3.4'de, şekil 3.3'deki dört giriş bit aralığı için ağaç diyagramı gösterilmektedir.

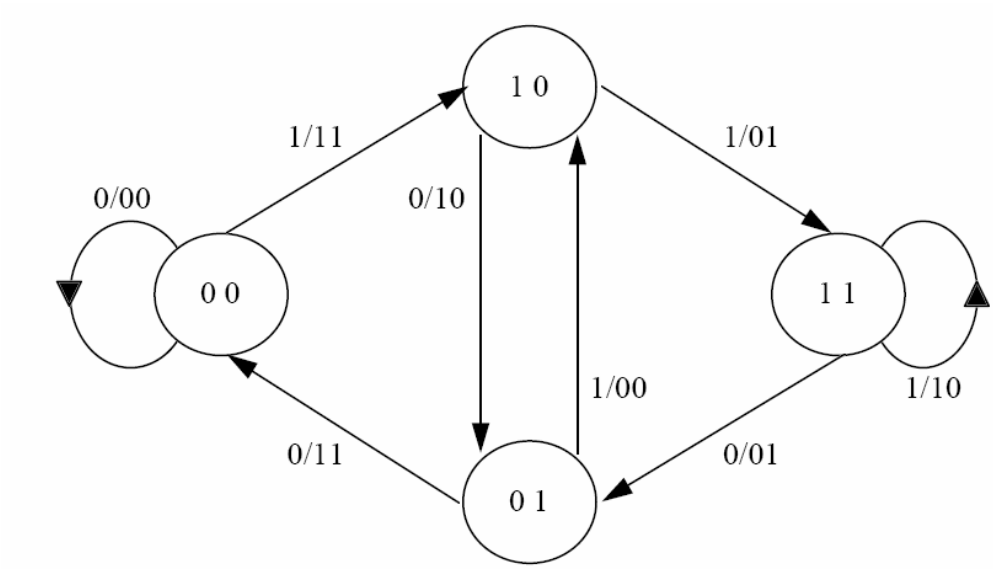


Şekil 3.4: Şekil 3.3 'deki dört giriş bit aralığı için ağaç diyagramı gösterimi

Ağaç diyagramında, kesiksiz çizgi 0 giriş bilgi bitini ve kesikli çizgi de 1 giriş bilgi bitini göstermektedir. Her giriş bitine karşılık gelen kodlanmış bitler ağacın dallarında (branch) gösterilmiştir. Bir giriş bilgi dizisi, ağaç diyagramında soldan sağa olacak şekilde bir yol (path) olarak tanımlanır. Örneğin, giriş bilgi dizisi, $u = \{1011\}$ olduğunda kodlanmış çıkış dizisi $v = \{11,10,00,01\}$ olur.

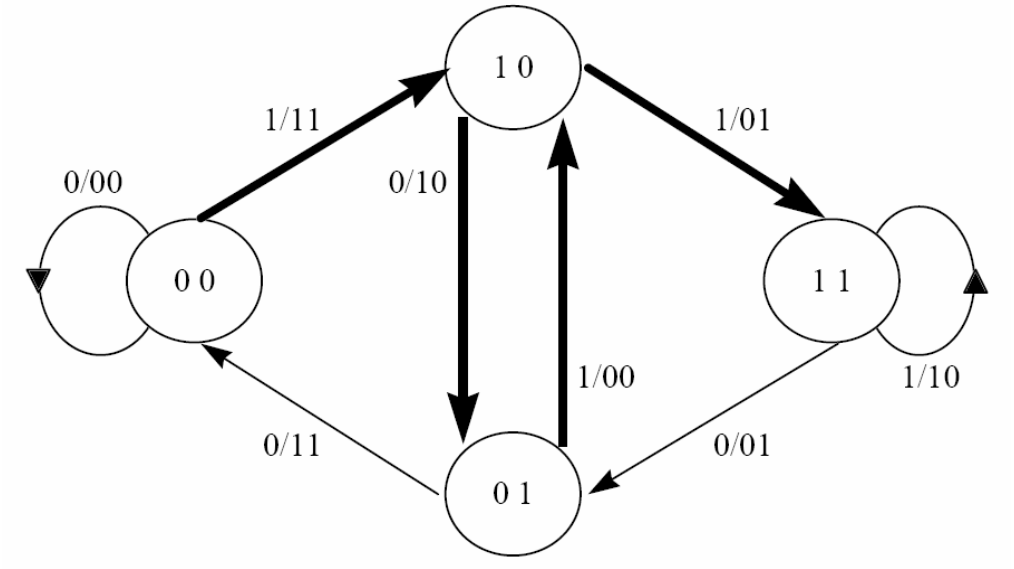
3.6.1.3. Durum diyagramı gösterimi

Durum diyagramı katlamalı kodlayıcının durum bilgisini göstermektedir. Katlamalı kodlayıcının durum bilgileri kaydıran yazmaçlara depolanmaktadır. Şekil 3.5'de şekil 3.3'deki kodlayıcının durum diyagramı gösterilmektedir.



Şekil 3.5: Şekil 2.2'deki kodlayıcının durum diyagramı

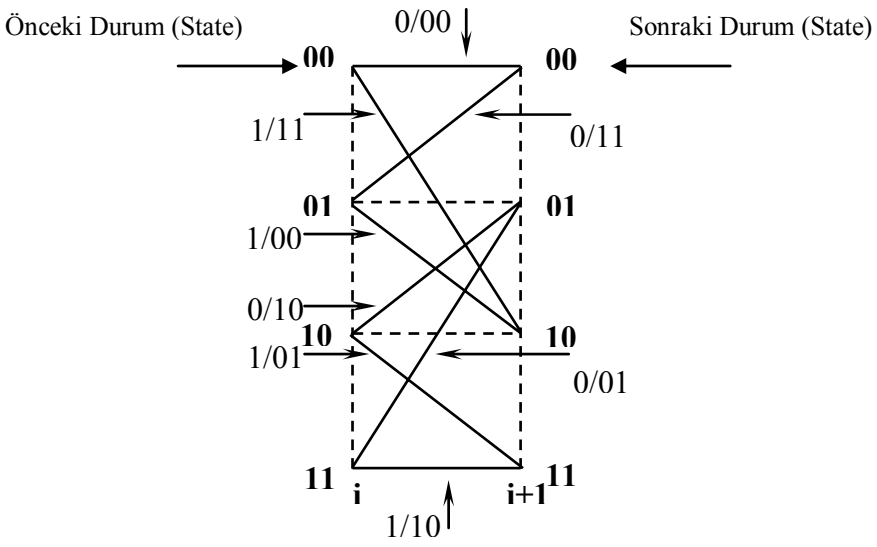
Durum diyagramında, kodlayıcının durum bilgileri yuvarlaklar içinde gösterilmektedir. Her yeni giriş bilgisi bir durumdan diğerine geçişe sebep olur. Durumlar arasındaki yol bilgileri u/v şeklinde gösterilmiştir. Burada u, giriş bilgi bitini ve v kodlanmış çıkış bitlerini gösterir. Kodlayıcının ilk durumları 0'dan başlar. Örneğin, giriş bilgi dizisi $u = \{1011\}$ olduğunda durum geçiş dizisi $s = \{10, 01, 10, 11\}$ ve kodlanmış çıkış dizisi, $v = \{11, 10, 00, 01\}$ olur. Şekil 3.6, verilen örnekte durum diyagramındaki yolları göstermektedir.



Şekil 3.6: {1011} giriş bilgi dizisi için durum geçişleri

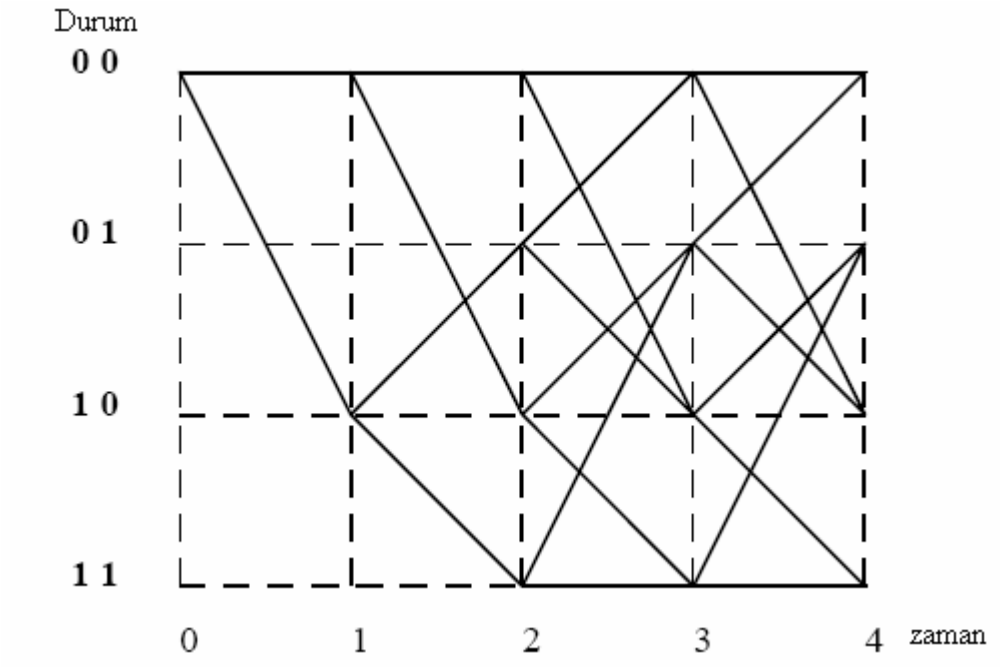
3.6.1.4. Kafes diyagram gösterimi

Kafes diyagramı (trellis diagram) temel olarak durum diyagramının tekrar çizilmesidir. Her zaman aralığında olası tüm durum geçişlerini göstermektedir. Kafes diyagramı katlamalı kodları çözmede çok yardımcı olmaktadır. Şekil 3.7'de genel kafes diyagramı ve şekil 3.8'de, şekil 3.3'deki gösterilen kodlayıcının kafes diyagramı gösterilmektedir.



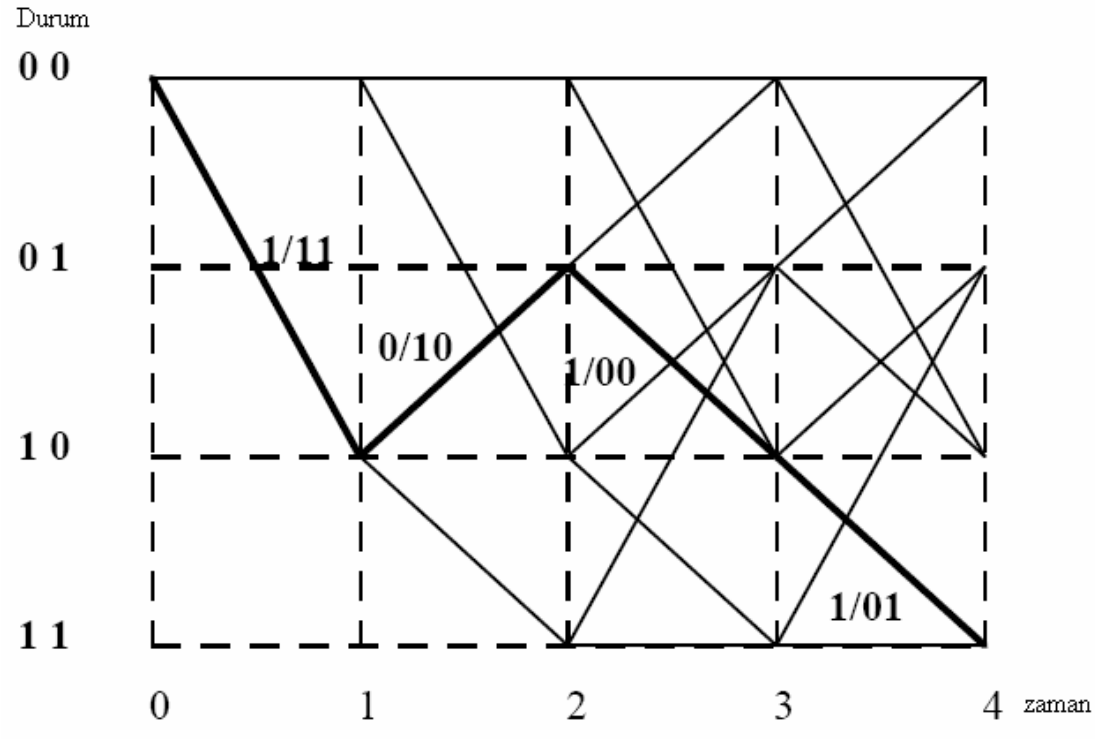
Şekil 3.7: Kafes diyagramının genel gösterimi

Şekil 3.7'de 00, 01, 10, 11 olarak gösterilen sayılar kaydıran yazmaçların durumlarıdır. 0/00, 1/10, 1/00, 0/11, 0/01, 1/11, 0/10, 1/01 şeklinde gösterilen ifadelerin / işaretinin solundaki rakamlar kodlayıcı girişlerine uygulanan bitleri, sağ tarafındaki rakamlar ise kodlanmış kodlayıcı çıkışlarını göstermektedir. Şekil 3.7'den de görüleceği üzere, örneğin, kaydıran yazmaçların önceki durumu 00 olduğu anda kodlayıcı girişine 1 biti gelirse, kaydıran yazmaçların sonraki durumu 10 olacak ve kodlanmış kodlayıcı çıkışı 11 olacaktır.



Şekil 3.8: Şekil 3.3'deki kodlayıcının girişine dört bitlik giriş bilgisi uygulandığındaki kafes diyagramı gösterimi

Şekil 3.9, Şekil 3.6 'daki durum geçişlerinin kafes yollarını göstermektedir.



Şekil 3.9: Şekil 3.6 ’daki durum geçişlerinin kafes yolları

3.7. Sonuç

Hata kontrol stratejileri olan FEC ve ARQ kullanım alanlarına göre seçilirler. Bit hata oranının düşük olduğu ve gerçek zamanlı uygulamalarda FEC kullanılırken bit hata oranının yüksek olduğu ve gecikmenin önemsiz olduğu durumlarda ARQ kullanımının uygundur. Ayrıca bu bölümde kullanılan FEC tekniklerinden katlamalı kod yapısının ayrıntılı ve her şekilde gösterimi açıklanmıştır.

4. VİTERBİ KOD ÇÖZME ALGORİTMASI VE BAŞARIM ANALİZİ

4.1. Giriş

Viterbi kod çözme algoritması, katlamalı kodlar için en çok uygulanan kod çözme algoritmasıdır ve en büyük olabilirlik (ML, Maximum Likelihood) kod çözme tekniğini kullanır [21]. Gürültülü bir kanal alıcıda bit hatalarına sebep olmakta ve Viterbi algoritması, alıcıda doğru bit dizisine en yakın bit dizisini kafes (trellis) diyagramını kullanarak bulmaktadır.

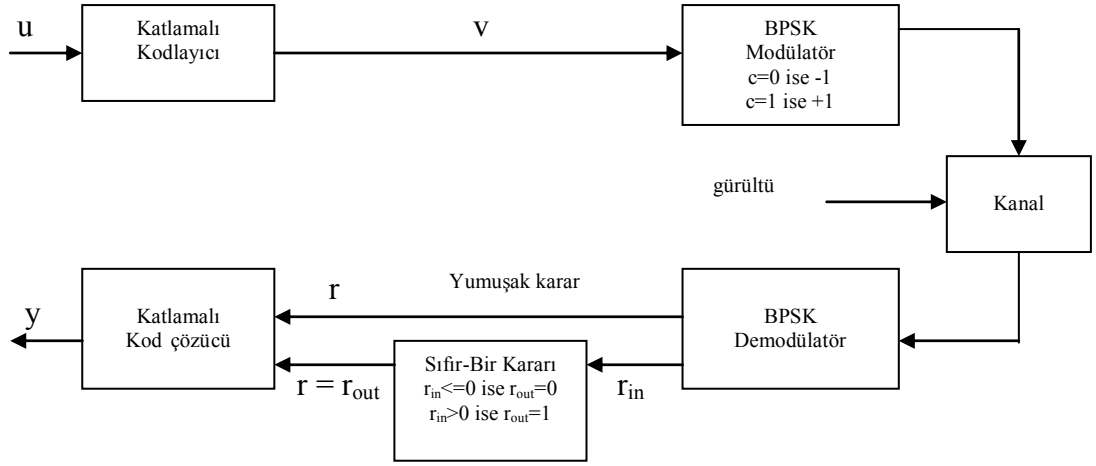
Viterbi kod çözme algoritması genel olarak iki farklı şekilde kullanılmaktadır. Bu farklılık alıcıya kodlanmış bilginin gelme şekli olarak ifade edilir. Kodlanmış bilgiler alıcıya, sıfır-bir kararlı ve yumuşak kararlı olmak üzere iki şekilde gelmektedir. Sıfır-bir kararlı işlemde kodlanmış bilgiler, kod çözücüde ±1 şeklinde ifade edilir. Yumuşak kararlı işlemde ise kodlanmış bilgiler çoklu seviyede ifade edilmektedir [21].

Bu bölümde, Viterbi kod çözme algoritmalarından olan sıfır-bir ve yumuşak kararlı Viterbi algoritmaları ile katlamalı kodların bu iki algoritmadaki başarım analizleri incelenmekte ve ayrıca benzetim modellerinden bahsedilmektedir. Kafes ve durum diyagramları hakkında kapsamlı bilgiler verilmektedir.

4.2. Sıfır-bir Kararlı (Hard Decision) ve Yumuşak Kararlı (Soft Decision) Viterbi Kod Çözme Algoritmaları

Sıfır-bir kararı ve yumuşak karar, alınan bitler üzerinde kullanılan nicemleme (quantization) tipleridir. Sıfır-bir kararlı kod çözme, alınan kanal değerleri üzerinde 1 bit nicemleme kullanarak işlem gerçekleştirir. Yumuşak kararlı kod çözme, alınan kanal değerleri üzerinde çok bitli nicemleme işlemini kullanmaktadır. İdeal yumuşak kararlı kod çözme için (sonsuz bit nicemleme), alınan kanal değerleri doğrudan kanal

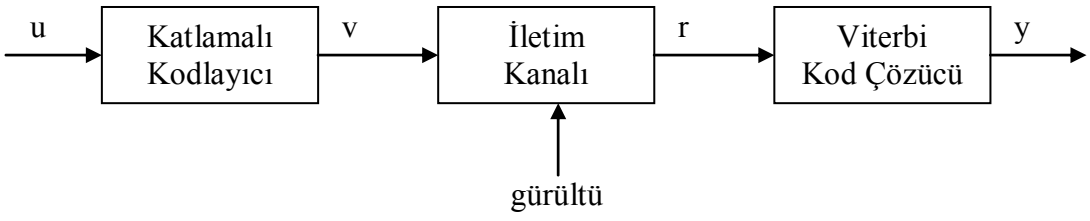
kod çözücüde kullanılır. Şekil 4.1, sıfır-bir ve yumuşak kararlı kod çözmeyi göstermektedir.



Şekil 4.1: Sıfır-bir ve yumuşak kararlı kod çözme [22]

4.2.1. Sıfır-bir kararlı Viterbi algoritması (HDVA)

Bir katlamalı kodlayıcı için, giriş dizisi u, çıkış dizisi v olacak şekilde, v dizisi gürültülü bir kanaldan iletilir ve alınan dizi, r meydana gelir. Viterbi algoritması, alınan r dizisi üzerinde en büyük olabilirlik algoritmasını kullanarak olası kod dizisi y 'yi üretir. $P(r|y)$ ile gösterilen olabilirlik fonksiyonu en yüksek yapılmak istenmektedir. Olabilirlik fonksiyonu, doğru veriyi (y), 2^{ς} olası kod kelimesi içinden seçer. y dizisi istenilen kod dizilerinden biri olmak zorundadır. Şekil 4.2 katlamalı kod sistem yapısını göstermektedir.



Şekil 4.2: Katlamalı kod sistemi

R kod oranına sahip katlamalı kod için, her zaman aralığında k (genelde k=1) bit kodlayıcı giriş bitine karşın n bit çıkış biti meydana gelmektedir. Giriş dizisi;

$$x = (x_0^{(1)}, x_0^{(2)}, ..., x_0^{(k)}, x_1^{(1)}, x_1^{(2)}, ...x_1^{(k)}, x_{ç+h-1}^{(1)}, ..., x_{ç+h-1}^{(k)}) \quad (4.1)$$

Ve kodlanmış çıkış dizisi,

$$v = (v_0^{(1)}, v_0^{(2)}, ..., v_0^{(n)}, v_1^{(1)}, v_1^{(2)}, ...v_1^{(n)}, v_{ç+h-1}^{(1)}, ..., v_{ç+h-1}^{(n)}) \quad (4.2)$$

olarak ifade edilir. Burada ç, giriş bilgi dizisinin uzunluğunu ve h, kaydıran yazmaçların sayısını göstermektedir. Dikkat edilmesi gereken bir nokta da, katlamalı kodlayıcının tüm durumlarının tekrar 0 konumuna gelmesi için h adet 0 bit, bilgi dizisinin kuyruğuna eklenmelidir. (4.1) ve (4.2) denklemlerinde, alt simge zaman indeksini, üst simge denklem (4.1) için k bit girişini, denklem (4.2) için n bit çıkış bit bloğunu göstermektedir. Alınan ve tahmin edilen (kod çözücü çıkışı) diziler, sırasıyla r ve y şu şekilde ifade edilir;

$$r = (r_0^{(1)}, r_0^{(2)}, ..., r_0^{(n)}, r_1^{(1)}, r_1^{(2)}, ..., r_0^{(n)}, r_{ç+h-1}^{(1)}, ..., r_{ç+h-1}^{(n)}) \quad (4.3)$$

$$y = (y_0^{(1)}, y_0^{(2)}, ..., y_0^{(n)}, y_1^{(1)}, y_1^{(2)}, ..., y_0^{(n)}, y_{ç+h-1}^{(1)}, ..., y_{ç+h-1}^{(n)}) \quad (4.4)$$

ML kod çözme algoritması için, Viterbi algoritması y dizisini $P(r|y)$'yi en büyükleyecek şekilde seçer. Kanal hafızasız varsayılır ve böylece gürültü sadece o anki alınan bitleri etkiler, o anki işlem gören bit, alınan diğer bitlerden bağımsız olarak işlem görür [20]. $P(r|y)$ işlemi, kod çözücüde üretilen y dizisinin r olma olasılığı olarak tanımlanabilir. Denklem (4.3) ve (4.4) göz önünde tutularak, r ve y bilgi dizileri açılacak olursa, denklem (4.5) ve (4.6) elde edilmektedir.

$$P(r|y) = \prod_{i=0}^{ç+h-1} [P(r_i^{(1)}|y_i^{(1)}) P(r_i^{(2)}|y_i^{(2)}) \cdots P(r_i^{(n)}|y_i^{(n)})] \quad [20] \quad (4.5)$$

$$P(r|y) = \prod_{i=0}^{ç+h-1} \left(\prod_{j=1}^{n} P(r_i^{(j)} | y_i^{(j)}) \right) \qquad [20] \qquad (4.6)$$

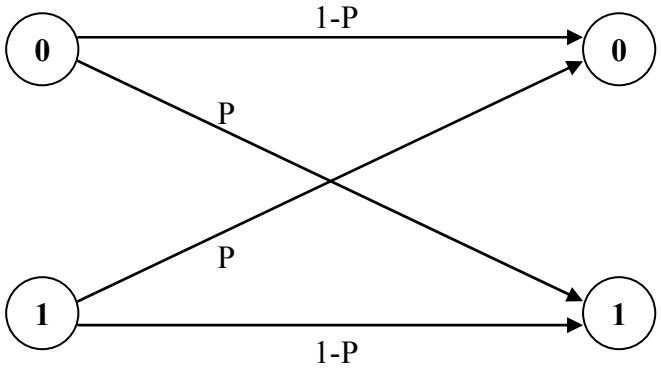
Denklem (4.6)'ya, alınan bilgi dizisine göre y'nin olasılık fonksiyonu denir [23]. $P(r|y)$ olasılığını en büyüklemek demek log $P(r|y)$ değerini en büyüklemek demektir. Çünkü logaritmalar tekdüze artış fonksiyonlarıdır. Böylece, logaritmik olasılık fonksiyonu,

$$\log P(r|y) = \sum_{i=0}^{ç+h-1} \left(\sum_{j=1}^{n} \log P(r_i^{(j)} | y_i^{(j)}) \right) \qquad [20] \qquad (4.7)$$

olarak elde edilir. Logaritmik fonksiyonların daha kolay bir şekilde toplanabilmesi için bit metrik tanımlanmıştır. Bit metrik şu şekilde tanımlanır,

$$M(r_i^{(j)} | y_i^{(j)}) = a[\log P(r_i^{(j)} | y_i^{(j)}) + b] \qquad [20] \qquad (4.8)$$

k1 ve k2 değerleri küçük pozitif tam sayılardır [20]. k1 ve k2 değerleri ikili simetrik kanal (BSC, Binary Symmetric Channel) ya da sıfır-bir kararlı kod çözme için tanımlanmıştır. Şekil 4.3'de bir BSC gösterilmiştir.



Şekil 4.3: İkili simetrik kanal modeli

BSC için, k1 ve k2 değerleri iki farklı yoldan seçilir. Basit yoldan, şu şekilde seçebiliriz;

$$k1 = \frac{1}{\log P - \log(1-P)} \qquad [20] \qquad (4.9)$$

$$k2 = -\log(1-P) \qquad [20] \qquad (4.10)$$

Sonuç olarak bit metrik,

$$M(r_i^{(j)} \left| y_i^{(j)}\right.) = \frac{1}{\log P - \log(1-P)} [\log P(r_i^{(j)} \left| y_i^{(j)}\right.) - \log(1-P)] \qquad (4.11)$$

BSC kanal modelinden, $P(r_i^{(j)} \left| y_i^{(j)}\right.)$ 'in alacağı değerlerin sadece P ve 1-P olabileceği görülmektedir (p, geçiş olasılığı). Tablo 4.1 bit metrik sonuçlarını göstermektedir.

Tablo 4.1: Basit bit metrik değerleri

$M(r_i^{(j)} \left\| y_i^{(j)}\right.)$	**Alınan bit** $r_i^{(j)} = 0$	**Alınan bit** $r_i^{(j)} = 1$
Kodu çözülmüş bit $y_i^{(j)} = 0$	0	1
Kodu çözülmüş bit $y_i^{(j)} = 1$	1	0

Tablo 4.1'deki bit metrikleri, kod çözücüde alınan ve kodu çözülmüş bitleri göstermektedir. Örneğin, kodlanmış bit $y_i^{(j)} = 0$ ve alınan bit $r_i^{(j)} = 0$ iken $M(r_i^{(j)} \left| y_i^{(j)}\right.) = 0$ olur. Bununla birlikte kodlanmış bit, $y_i^{(j)} = 0$ ve alınan bit $r_i^{(j)} = 1$ ise $M(r_i^{(j)} \left| y_i^{(j)}\right.) = 1$ olmaktadır. Bit metrikleri, hamming mesafesi ile ilgilidir ve hamming mesafe metriği olarak adlandırılır. Sonuç olarak, Viterbi algoritması, y

kod dizisini r dizisine göre kafes diyagramı üzerinden en küçük hamming mesafesine göre seçmeyi amaçlayan bir kod çözme algoritmasıdır.

Alternatif olarak, k1 ve k2 şu şekilde seçilebilir.

$$k1 = \frac{1}{\log(1-P) - \log P} \qquad [20] \qquad (4.12)$$

$$k2 = -\log P \qquad [20] \qquad (4.13)$$

Bit metrik sonuçları,

$$M(r_i^{(j)} \left| y_i^{(j)}\right.) = 1 = \frac{1}{[\log(1-P) - \log P]} [\log P(r_i^{(j)} \left| y_i^{(j)}\right.) - \log P] \qquad (4.14)$$

Tablo 4.2 bu değerler kullanılarak elde edilen bit metriklerini göstermektedir.

Tablo 4.2: Alternatif bit metrik değerleri

$M(r_i^{(j)} \left\| y_i^{(j)}\right.)$	**Alınan bit** $r_i^{(j)} = 0$	**Alınan bit** $r_i^{(j)} = 1$
Kodu çözülmüş bit $y_i^{(j)} = 0$	1	0
Kodu çözülmüş bit $y_i^{(j)} = 1$	0	1

Bu durum için, Viterbi algoritması, y kod dizisini r dizisine göre kafes diyagramı üzerinden en büyük hamming mesafesine göre seçer. Bundan başka, isteğe bağlı kanal için (BSC olmayabilir), k1 ve k2 değerleri deneyerek ve bit metrikleri kabul edilebilir hatalar içererek bulunur.

Bit metrikten yola çıkılarak yol (path) metrikleri tanımlanmıştır. Yol metrikleri şu şekilde tanımlanır,

$$M(r|y) = \sum_{i=0}^{ç+h-1} \left(\sum_{j=1}^{n} M(r_i^{(j)} | y_i^{(j)}) \right) \qquad [20] \qquad (4.15)$$

ve kafes diyagramındaki kodu çözülmüş bit dizisi y ile alınan bit dizisi r'nin olası toplam maliyetini gösterir. Ayrıca, k. Dal (branch) metriği şu şekilde tanımlanmaktadır,

$$M(r_k | y_k) = \sum_{j=1}^{n} M(r_k^{(j)} | y_k^{(j)}) \qquad [20] \qquad (4.16)$$

ve k. kısmi yol metriği,

$$M^k(r|y) = \sum_{i=0}^{k} M(r_i | y_i) \qquad [20] \qquad (4.17)$$

$$= \sum_{i=0}^{k} \left(\sum_{j=1}^{n} M(r_i^{(j)} | y_i^{(j)}) \right) \qquad [20] \qquad (4.18)$$

ile ifade edilmektedir. k. dal metriği kafes diyagramından seçilen bir dalın maliyetini gösterir. Burada k. kısmi yol metriği, k indeksli zamanda, kodu çözülmüş bit dizisi y'nin kısmi maliyetidir.

Viterbi algoritması, yol metriklerini hesaplamak için kafes diyagramını kullanmaktadır. Kafes diyagramındaki her durum, kısmi yol metriği değerine atanmıştır. Kısmi yol metriği, t=0 anında 0 durumundan t≥0 anında k durumuna tanımlanmaktadır. Her durumda, en iyi kısmi yol metriği, o durumda bitirilen yollardan seçilmektedir [20]. En iyi kısmi yol metriği, seçilen k1 ve k2 değerlerine göre büyük veya küçük metrik olabilir. Seçilen metrik, kazanan (survivor) yol olarak tanımlanır ve kalan metrikler kazanamayan metrikler olarak adlandırılmaktadır. Kazanan metrikler kaydedilir ve kazanamayan metrikler kafes diyagramından atılır.

Viterbi algoritması, işlemin sonucunda, tek kazanan yolu seçer ve bu yol ML yolu olarak adlandırılır. Kafes diyagramından ML yolu geriye doğru izlenirse, ML kodu çözülmüş dizi bulunabilir.

Sıfır-bir kararlı Viterbi algoritması (HDVA, Hard Decision Viterbi Algorithm)'nın işlem basamakları aşağıdaki şekilde tanımlanabilir [24,20]:

$S_{k,t}$, t anında kafes diyagramında S_k 'nın durumudur. Kafes diyagramındaki her durum $V(S_{k,t})$ ile gösterilen bir değere atanır.

1.(a) t=0 başlangıç değeri olarak atanır.

(b) $V(S_{0,0})=0$ ve tüm diğer $V(S_{k,t})=+\infty$ değerlerine atanır.

2.(a) t=t+1 olarak atanır.

(b) t anında S_k'ya giden tüm yollar için kısmi yol metrikleri hesaplanır.

İlk olarak t. dal metriği, $M(r_t|y_t)=\sum_{j=1}^{n} M(r_t^{(j)}|y_t^{(j)})$ bulunur. Bu değer hamming mesafesi $\sum_{j=1}^{n}|r_t^{(j)}-y_t^{(j)}|$'den bulunur. İkinci olarak, t. kısmi yol metriği, $M^t(r|y)=\sum_{i=0}^{t} M(r_i|y_i)$ hesaplanır. Bu değer $V(S_{k,t-1})+M(r_t|y_t)$'dan hesaplanmaktadır.

3. (a) $V(S_{k,t})$, t anında S_k durumuna giderken en iyi kısmi yol metriğine getirilir.

En iyi kısmi yol metriği, en küçük değere sahip kısmi yol metriğidir.

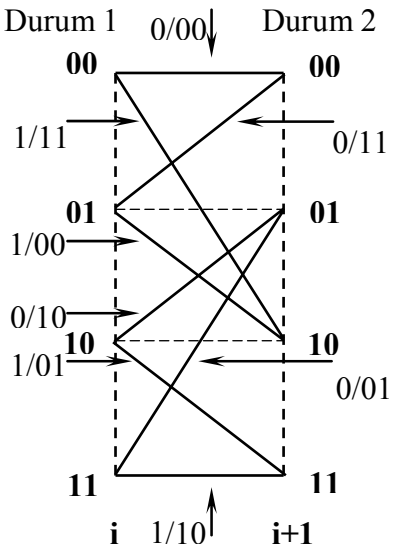
(b) Eğer birden fazla en iyi kısmi yol metriği varsa herhangi biri seçilebilir.

4.En iyi kısmi yol metrikleri kaydedilir ve kazanan bit ve durum yolları ile ilişkilendirilir.

5.Eğer t<ç+h-1, 2. adıma geri dönülür.

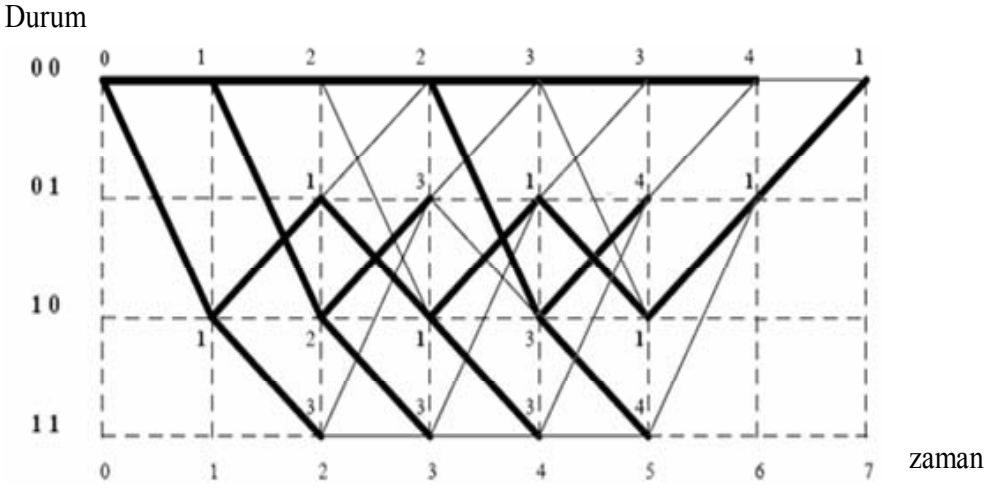
Viterbi algoritmasının sonucu ML kod kelimesine karşılık düşen tek kafes yoludur.

Aşağıda basit bir HDVA kod çözme örneği gösterilmektedir ve Şekil 3.3'deki katlamalı kodlayıcı kullanılmaktadır. Giriş dizisi, u={1010100} dir ve giriş dizisinin son iki biti kodlayıcının tüm durumlarını sıfır yapmak için eklenmiştir. Kodlanmış dizi, v={11, 10, 00, 10, 00, 10, 11} dir. Bununla birlikte, kod çözücüde alınan dizi, r={10, 10, 00, 10, 00, 11} ve bir bitlik hatalı olsun (hatalı bitin altı çizili). Şekil 4.4 örnek katlamalı kodlayıcının durum geçiş diyagramını göstermektedir.



Şekil 4.4: Örnek katlamalı kodlayıcının durum geçiş diyagramı

Durum geçiş diyagramı, tahmin edilen bilgiyi ve kod çözme işlemi için gerekli dallar üzerindeki kodlanmış bitleri göstermektedir. HDVA kod çözme, şekil 4.5'de gösterilen kafes diyagramı üzerinden ML yolunu seçer. Seçilen kısmi yol (toplanmış) metriği bu örnekte, her bir düğüm için en küçük hamming mesafesi şeklinde gösterilmiştir. Kalın çizilen kısmi yol metrikleri ML yolunu göstermektedir. Kazanan yollar kalın kesiksiz çizgilerle ve yarışan yollar sadece kesiksiz çizgilerle gösterilmiştir. Birden fazla en iyi yol çıkarsa her zaman ilk yol seçilir.



Şekil 4.5: Örnekteki HDVA kod çözme

Şekil 4.5'deki kafes diyagramından, tahmin edilen kod dizisi y={11, 10, 00, 10, 00, 10, 11} olarak bulunur ve bu v kod dizisidir. Şekil 4.4'deki durum geçiş diyagramı kullanılarak tahmin edilen bilgi dizisi, u'={1010100} olarak bulunur.

4.2.2. Yumuşak kararlı Viterbi algoritması (SDVA)

Genel olarak yumuşak kararlı Viterbi algoritmasını uygulamanın iki yolu vardır. Birinci metotta hamming mesafe metriği yerine öklit mesafe metriği kullanılır. Öklit mesafe metriğinde kullanılan kod çözücüde alınmış bitler, çoklu bit nicemlemesi ile işlem görür. İkinci metotta, çoklu bit nicemlemesi ile işlem yapan korelasyon metriği kullanılır.

4.2.3. Yumuşak kararlı Viterbi algoritması (1. metot)

Yumuşak kararlı kod çözmede, alıcı alınan bitleri bir ya da sıfır değerlerine atamaz (tek bit nicemlemesi) bunun yerine çoklu bit ya da sonsuz bit nicemlemesi kullanılmaktadır [20]. İdeal olarak, alınan bit dizisi r, sonsuz bit nicemlemesine tabi tutulur ve doğrudan yumuşak kararlı Viterbi kod çözücüsüne verilir. Yumuşak kararlı Viterbi algoritması, sıfır-bir kararlı Viterbi algoritmasına, hamming mesafesi yerine karesi alınmış öklit mesafesi kullanılması farkı dışında benzerdir.

Yumuşak kararlı Viterbi algoritması (SDVA1, Soft Decision Viterbi Algorithm 1)'nın işlem basamakları:

$S_{k,t}$, t anında kafes diyagramında S_k 'nın durumudur. Kafes diyagramındaki her durum $V(S_{k,t})$ ile gösterilen bir değere atanır.

1.(a) t=0 başlangıç değeri atanır.

(b) $V(S_{0,0})=0$ ve tüm diğer $V(S_{k,t})=+\infty$ olarak atanmaktadır.

2.(a) t=t+1 olarak atanır.

(b) t anında S_k'ya giden tüm yollar için kısmi yol metriklerini hesaplanır.

İlk olarak t. dal metriği, $M(r_t \mid y_t) = \sum_{j=1}^{n} M(r_t^{(j)} \mid y_t^{(j)})$ bulunur. Bu değer karesi alınmış öklid mesafesi $\sum_{j=1}^{n} (r_t^{(j)} - y_t^{(j)})^2$ 'den hesaplanır. İkinci olarak, t. kısmi yol metriği,

$M^t(r|y) = \sum_{i=0}^{t} M(r_i \,|\, y_i)$ hesaplanır. Bu değer $V(S_{k,t-1}) + M(r_t|y_t)$ 'dan bulunmaktadır.

3. (a) $V(S_{k,t})$, t anında S_k durumuna giderken en iyi kısmi yol metriğine getirilir.
En iyi kısmi yol metriği, en küçük değere sahip kısmi yol metriğidir.

b) Eğer birden fazla en iyi kısmi yol metriği varsa herhangi biri seçilebilir.

4.En iyi kısmi yol metrikleri kaydedilir ve kazanan bit ve durum yolları ile ilişkilendirilir.

5.Eğer t<ç+h-1, 2. adıma geri dönülür.

4.2.4. Yumuşak kararlı Viterbi algoritması (2. metot)

İkinci yumuşak kararlı Viterbi algoritması (SDVA2, Soft Decision Viterbi Algorithm 2) aşağıda gösterilmiştir. Benzerlik fonksiyonu, gauss olasılık yoğunluk fonksiyonu (pdf, probability density function),

$$P\left(r_i^{(j)} \,\middle|\, y_i^{(j)}\right) = \frac{1}{\sqrt{\pi No}} e^{-(r_i^{(j)} - y_i^{(j)}\sqrt{Eb})^2 / No} \tag{4.19}$$

Burada, Eb , her kod kelimesi başına düşen alınan enerjidir ve No tek taraflı yayılım yoğunluğudur [20]. Alınan bit, gauss rasgele değişkenidir ve $y_i^{(j)}\sqrt{Eb}$ anlamındadır aynı zamanda varyansı No/2'dır. Logaritmik olabilirlik fonksiyonu aşağıdaki şekilde tanımlanabilir [20].

$$\log P(r|y) = \sum_{i=0}^{ç+h-1} \left(\sum_{j=1}^{n} \log P(r_i^{(j)} \,|\, y_i^{(j)}) \right) \tag{4.20}$$

$$= \sum_{i=0}^{ç+h-1} \left\{ \sum_{j=1}^{n} \left[-\frac{(r_i^{(j)} - y_i^{(j)}\sqrt{Eb})^2}{No} - \log\sqrt{\pi No} \right] \right\} \tag{4.21}$$

$$= \frac{-1}{No} \sum_{i=0}^{ç+h-1} \left[\sum_{j=1}^{n} (r_i^{(j)} - y_i^{(j)}\sqrt{Eb})^2 \right] - \frac{(ç+h)n}{2} \log \pi No \tag{4.22}$$

$$= \frac{-1}{No} \sum_{i=0}^{ç+h-1} \left[\sum_{j=1}^{n} (r_i^{(j)^2} - 2r_i^{(j)} y_i^{(j)} \sqrt{Eb} + y_i^{(j)2} Eb \right] - \frac{(ç+h)n}{2} \log \pi No \quad (4.23)$$

Burada $y_i^{(j)2} = 1$ dir.

$$= C1 \sum_{i=0}^{ç+h-1} \left[\sum_{j=1}^{n} r_i^{(j)} y_i^{(j)} \right] + C2 \quad (4.24)$$

C1 ve C2 değerlerinin hepsi y fonksiyonunun terimleri değildir.

$$= C1(r \bullet y) + C2 \quad (4.25)$$

Buradan da görülebileceği üzere bit metrikler şu şekilde tanımlanabilir,

$$M(r_i^{(j)} \left| y_i^{(j)} \right.) = r_i^{(j)} y_i^{(j)} \quad (4.26)$$

Yumuşak kararlı Viterbi algoritması (SDVA2, Soft Decision Viterbi Algorithm 2) 'nın işlem basamakları:

$S_{k,t}$, t anında kafes diyagramında S_k 'nın durumudur. Kafes diyagramındaki her durum $V(S_{k,t})$ ile gösterilen bir değere atanır.

1. (a) t=0 başlangıç değeri atanır.

(b) $V(S_{0,0})=0$ ve tüm diğer $V(S_{k,t})=+\infty$ olarak atanmaktadır.

2.(a) t=t+1 olarak atanır.

(b) t anında S_k'ya giden tüm yollar için kısmi yol metriklerini hesaplanır.

İlk olarak t. dal metriği, $M(r_t \left| y_t \right.) = \sum_{j=1}^{n} M(r_t^{(j)} \left| y_t^{(j)} \right.)$ bulunur. Bu değer $r_i^{(j)}$ ve $y_i^{(j)}$ 'nin korelasyonlarından hesaplanır, $\sum_{j=1}^{n} r_i^{(j)} y_i^{(j)}$. İkinci olarak, t. kısmi yol metriği, $M^t(r \left| y \right.) = \sum_{i=0}^{t} M(r_i \left| y_i \right.)$ hesaplanır. Bu değer $V(S_{k,t-1}) + M(r_t \left| y_t \right.)$ 'dan bulunur.

3. (a) $V(S_{k,t})$, t anında S_k durumuna giderken en iyi kısmi yol metriğine getirilir.

En iyi kısmi yol metriği, en büyük değere sahip kısmi yol metriğidir.
b) Eğer birden fazla en iyi kısmi yol metriği varsa herhangi biri seçilebilir.
4. En iyi kısmi yol metrikleri kaydedilir ve kazanan bit ve durum yolları ile ilişkilendirilir.
5. Eğer t<ç+h–1, 2. adıma geri dönülür.

Genel olarak, AWGN kanalda, yumuşak kararlı kod çözme, sıfır-bir kararlı kod çözme üzerine yaklaşık olarak 2 dB kodlama kazancı sağlar.

4.3. Kod Çözme Derinliği

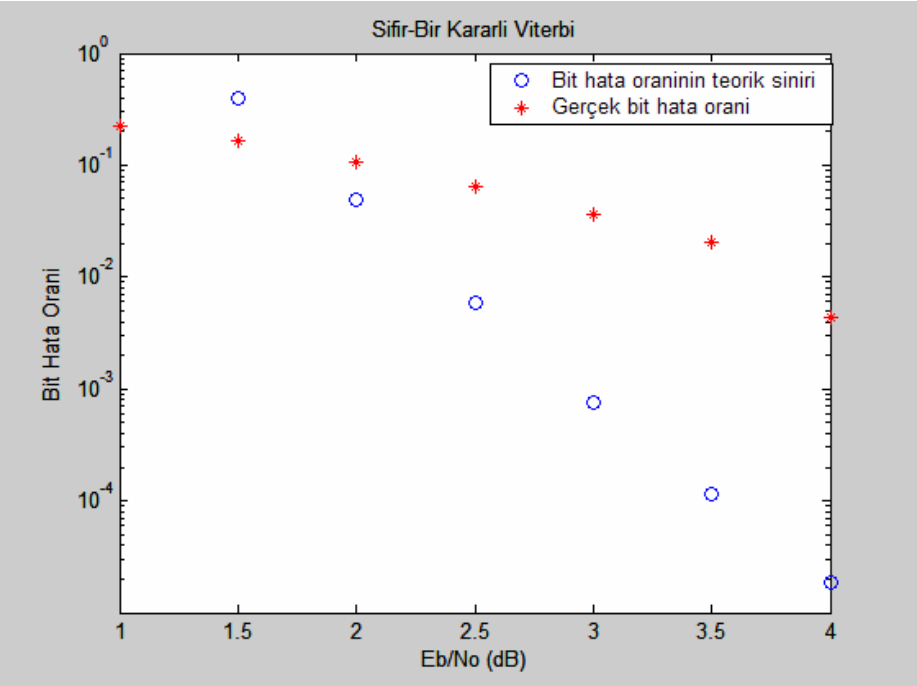
Kod çözme derinliğini zamanda bir pencere gibi düşünürsek, metrik hesaplaması için, pencerenin başında bitler üzerinde karar verir ve pencerenin sonunda bitleri onaylar. Bu işlem, daha az hafıza ve daha küçük kod çözme gecikmesi meydana getiren en uygun ML kod çözme işlemini sağlar. Denemeler sonucunda kod çözme derinliğinin sınırlama boyu (K) 'nın beş katı civarında alınmasının uygun olacağı görülmüştür [2].

4.4. Nicemleme Derecesi

Yumuşak kararlı Viterbi kod çözme işlemi için, alınan sinyal üzerindeki nicemleme derecesi kod çözücü başarımını etkilemektedir. Daha yüksek bit nicemlemesi, Viterbi kod çözücüsünün başarımını arttırmaktadır. Sonsuz bit nicemleme durumunun tersine sekiz seviyeli nicemleme işlemi başarımı biraz düşürdüğü bulunmuştur [20].

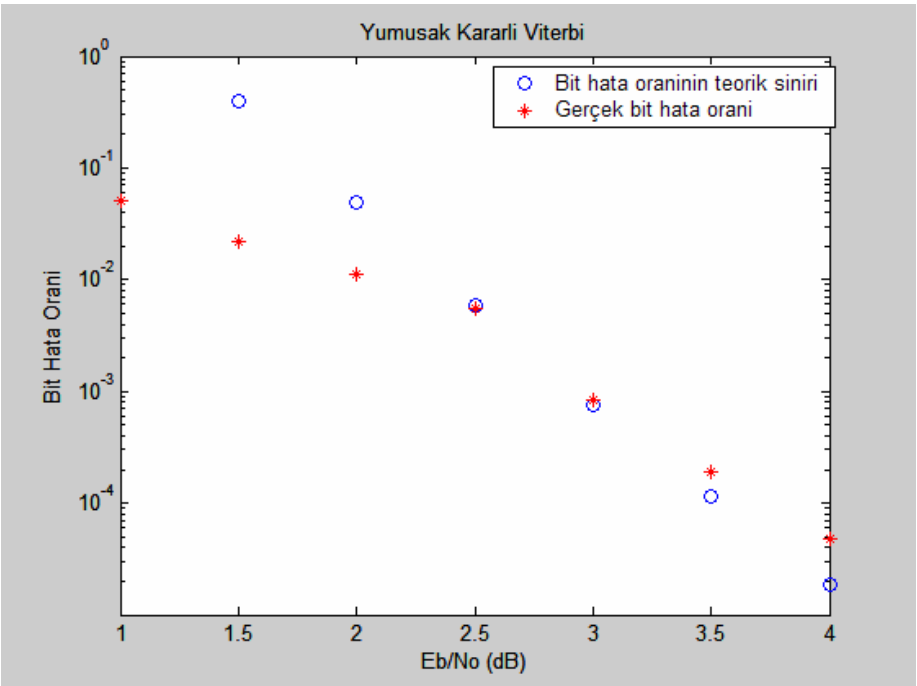
4.5. Viterbi Kod Çözme Algoritmasının Başarım Analizi

Aşağıdaki şekilde de görüleceği üzere sıfır-bir kararlı Viterbi kod çözme algoritması 4 dB' de 10^{-2}'den biraz fazla bit hata oranı başarımı göstermiştir.



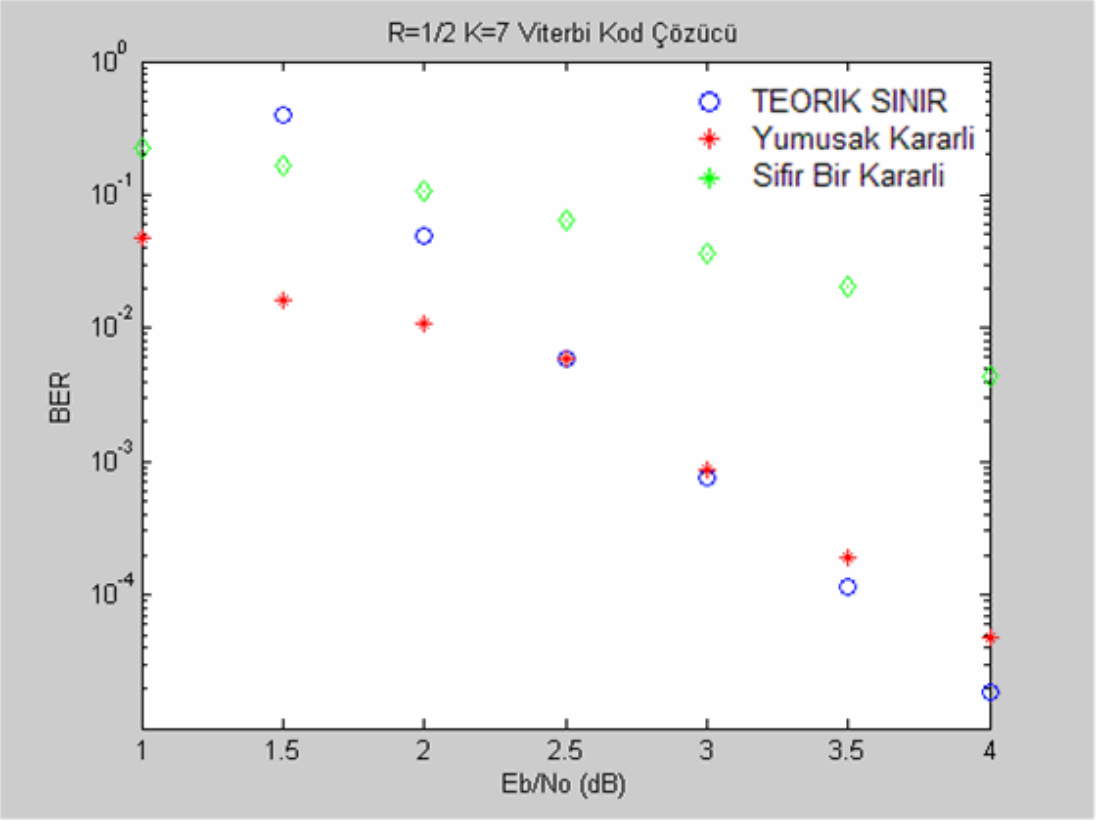
Şekil 4.6: Sıfır-Bir kararlı Viterbi kod çözme algoritmasının başarım grafiği

Şekil 4.8 'de yumuşak kararlı Viterbi kod çözme algoritmasının bit hata oranı başarımı gösterilmiştir. 4 dB'de 10^{-4} 'den fazla başarım göstermiştir.



Şekil 4.7: Yumuşak kararlı Viterbi kod çözme algoritmasının başarım grafiği

Sıfır-bir kararlı ve yumuşak kararlı Viterbi kod çözme algoritmaları karşılaştırılarak şekil 4.9'da başarımları gösterilmiştir.

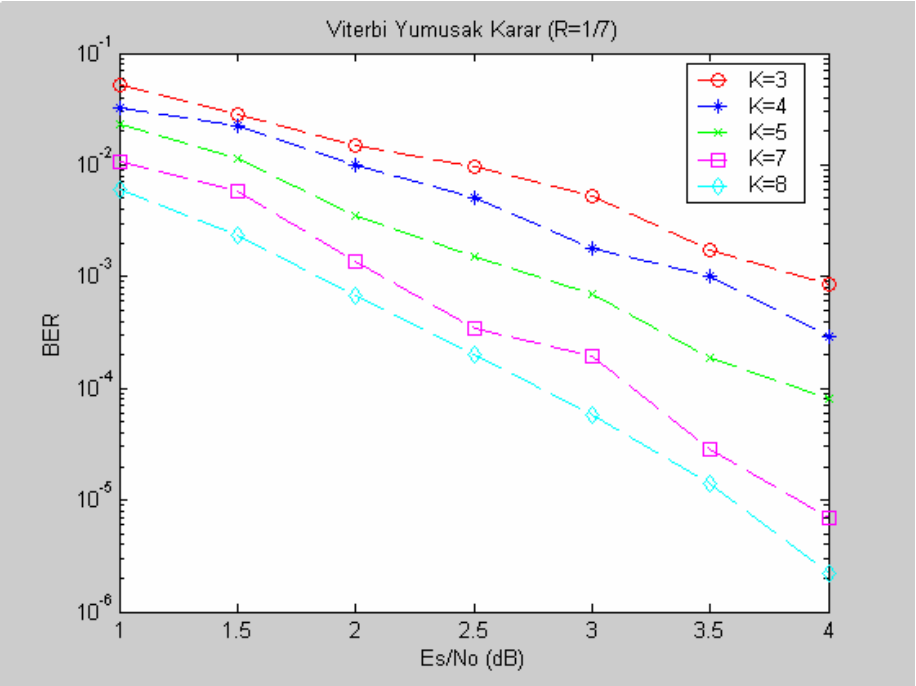


Şekil 4.8: Sıfır-bir ve yumuşak kararlı Viterbi kod çözme algoritmasının başarım gösterimi

Şekil 4.9'da görüleceği üzere yumuşak kararlı Viterbi kod çözme algoritması sıfır-bir kararlı Viterbi kod çözme algoritmasına göre çok daha iyi bir başarım göstermiştir. Şekil 4.9'da 1 dB ve 3 dB değerlerindeki sonuçlara dikkat edilecek olursa yumuşak kararlı Viterbi kod çözme algoritması sıfır-bir kararlı Viterbi kod çözme algoritmasına göre 2 dB'lik bir kazanç sağlamaktadır.

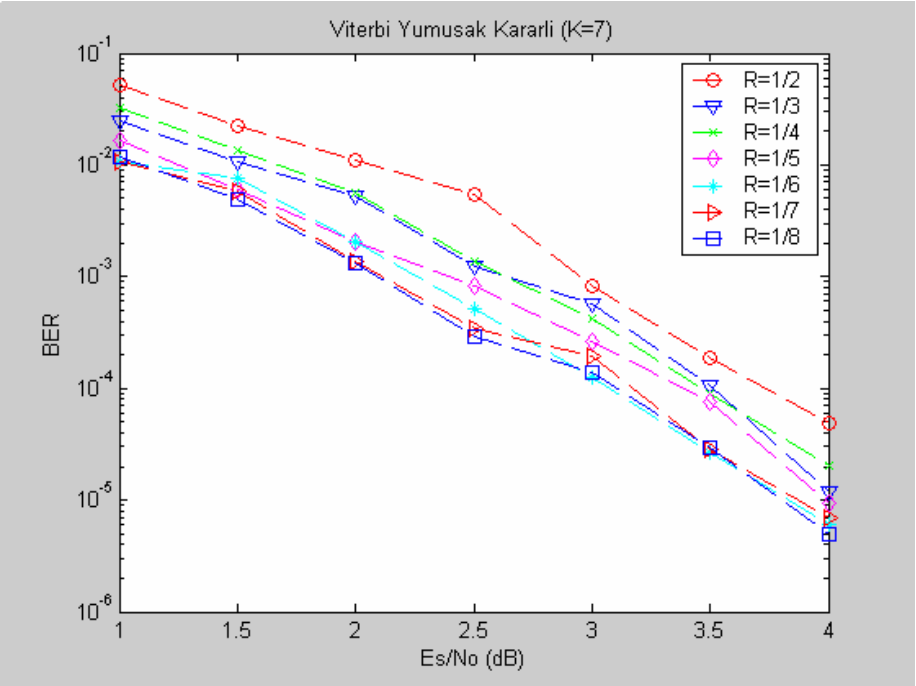
4.6. Kısıtlama Boyutu ve Kod Oranının Kod Çözme Başarımına Etkisi

Viterbi kod çözme algoritmasında kısıtlama boyutunun ve kodlayıcı hafızasının kod çözme başarımına etkisini göstermek için Yumuşak kararlı Viterbi algoritması kullanılmıştır.



Şekil 4.9: R=1/7 oranlı yumuşak kararlı Viterbi kod çözme algoritmasının farklı hafıza sayılarındaki başarım grafiği

R=1/7 oranlı yumuşak kararlı Viterbi kod çözme algoritmasının farklı hafıza sayılarındaki yani katlamalı kodlayıcıda kullanılan kaydıran yazmaç sayısına göre başarım sonucu gösterilmektedir. Hafıza ne kadar arttırılırsa başarım o kadar iyi olmaktadır.

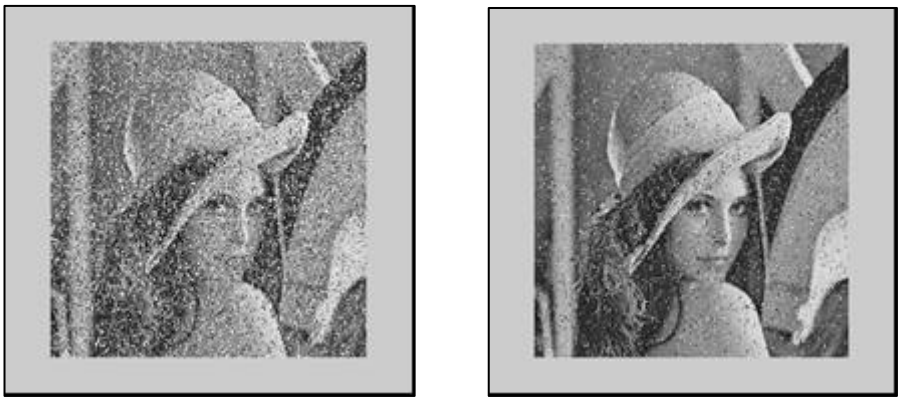


Şekil 4.10: K=7 değerine sahip yumuşak kararlı Viterbi kod çözme algoritmasının farklı kod oranlarında başarım grafiği

K=7 değerine sahip yumuşak kararlı Viterbi kod çözme algoritmasının farklı kod oranlarında başarım grafiği şekil 4.11'de gösterilmiştir. Kod oranı ne kadar arttırılırsa başarım o kadar iyi olmaktadır.

4.7. Viterbi Kod Çözme Algoritması Uygulaması

Viterbi kod çözme algoritması ile bmp uzantılı bir resim, Matlab Simulink programı kullanılarak vericiden alıcıya gönderilmiştir. Bu uygulama farklı SNR değerleri ile her iki Viterbi algoritması ile gerçekleştirilmiştir. Uygulamada kullanılan katlamalı kodlayıcının kısıtlama boyutu K=7 ve kod oranı R=1/2 olarak seçilmiştir.



Şekil 4.11: SNR=0 dB değerinde Viterbi sonuçları

Sol taraftaki resim sıfır-bir kararlı, sağ taraftaki resim yumuşak kararlı Viterbi algoritması kullanılarak ve SNR değeri 0 dB alınarak elde edilmiştir.

Şekil 4.12: SNR=1 dB değerinde Viterbi sonuçları

Sol taraftaki resim sıfır-bir kararlı, sağ taraftaki resim yumuşak kararlı Viterbi algoritması kullanılarak ve SNR değeri 1 dB alınarak elde edilmiştir.

Şekil 4.13: SNR=2 dB değerinde Viterbi sonuçları

Sol taraftaki resim sıfır-bir kararlı, sağ taraftaki resim yumuşak kararlı Viterbi algoritması kullanılarak ve SNR değeri 2 dB alınarak elde edilmiştir.

Şekil 4.14: SNR=3 dB değerinde Viterbi sonuçları

Sol taraftaki resim sıfır-bir kararlı, sağ taraftaki resim yumuşak kararlı Viterbi algoritması kullanılarak ve SNR değeri 3 dB alınarak elde edilmiştir.

Şekil 4.15: SNR=4 dB değerinde Viterbi sonuçları

Sol taraftaki resim sıfır-bir kararlı, sağ taraftaki resim yumuşak kararlı Viterbi algoritması kullanılarak ve SNR değeri 4 dB alınarak elde edilmiştir.

4.8. Sonuç

Bu bölümde sıfır-bir ve yumuşak kararlı Viterbi kod çözme algoritmalarının karşılaştırmalı başarım analizi gösterilmektedir. Viterbi algoritmalarının başarımına etki eden faktörlerden söz edilmiş ve yumuşak kararlı Viterbi algoritmasının sıfır-bir kararlı Viterbi algoritmasına göre 2dB'lik bir başarım göstermektedir.

5. TURBO KODLAR

5.1. Giriş

İleri hata düzeltme tekniklerindeki ilerlemeler sonucunda, yakın geçmişte en iyi hata düzeltme tekniği olarak Turbo kodlar gösterilmektedir. Turbo kodlar, bilim dünyasına sunulduğu günden bu yana uzay ve uydu iletişimleri gibi düşük güçlü uygulamalarda, 3G hücresel ve kişisel iletişim sistemleri gibi uygulama alanlarında kullanılmıştır.

Bu bölümde, Turbo kodların gelişimi ve Turbo kodlayıcının yapısını oluşturan farklı katlamalı kodlayıcı yapılarından söz edilmektedir. Ayrıca Turbo kodlayıcı yapısında kullanılan serpiştirici yapısı ve çeşitlerinden bahsedilmekte ve kodların bağlanması hakkında bilgi verilmektedir.

5.2. Turbo Kodların Gelişimi

Turbo kodlar ilk olarak 1993 yılında IEEE (Institute of Electrical and Electronics Engineers) Uluslararası Haberleşme Konferansı'nda Berrou, Glavieux ve Thitimajshima tarafından tanıtılmıştır. Turbo kodlar serpiştirici (interleaver) ile ayrılmış katlamalı (convolutional) kodlayıcılardan ve döngülü (iterative) kod çözücü yapısından oluşmaktadır. Kodlayıcının temel yapısını küçük kısıtlama uzunluğuna sahip özyineli sistematik katlamalı kod (RSC, Recursive Systematic Convolutional code) ve rasgele serpiştirici (Random Interleaver) oluşturmaktadır. 1993 yılında yapılan çalışma [3] ile katlamalı kodların yeni bir sınıfı olan Turbo kodlar bulunmuştur. Bu çalışma ile iki adet özyineli sistematik katlamalı kodlayıcıdan oluşan Turbo kodlayıcının bit hata olasılığı açısından başarımının Shannon sınırına çok yakın olduğu gösterilmiştir. Turbo kodların yapıtaşını oluşturan katlamalı kodlarda kullanılan Viterbi algoritması, verinin dizi hata olasılığını en aza indiren

en büyük olasılıklı kod çözme tekniği olduğu ancak bit hata olasılığını en aza indirmesi gerekmediği belirtilmiş, Turbo kodlarda ise Viterbi algoritması yerine sonsal olasılık (APP, A Posteriori Probability) tekniğini kullanan Bahl Algoritması kullanılması gerektiği vurgulanmış ve sistematik katlamalı kodlar için değiştirilmiş Bahl kod çözme algoritması geliştirilmiştir [7].

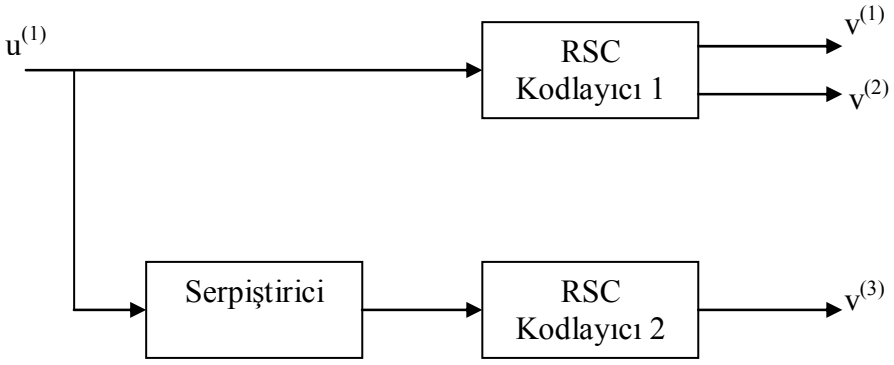
Turbo kod çözücü, sistematik veri bitlerinin yanında gönderilen eşlik bitlerini kullanarak iletim kanalında oluşan hataları düzeltmektedir. Turbo kod çözme işlemi, bileşen kod çözücüler arasında özyinelemeli olarak yapıldığından Turbo kod çözücüde kullanılabilecek olan katlamalı kod çözücüler, yumuşak-giriş (soft-in) ve yumuşak-çıkış (soft-out) özelliğine sahip olmalıdır. En büyük sonsal (MAP, Maximum A Posteriori) katlamalı kod çözme algoritması bu koşulları sağlamaktadır [3].

Turbo kodlarda yüksek başarımın temel nedeni serpiştiricidir. Seri ve paralel bağlı kodlamada serpiştirici, patlama hatalarını düzeltme kapasitesini geliştirmek ve kodun rasgele olmasını arttırmak için kullanılmaktadır. Serpiştirici, belli bir kurala göre kaynak bitlerinin sırasını değiştirir. Böylece iki kodlayıcı girişine, aralarında ilişki olmayan iki ayrı kaynak dizisi uygulanmış etkisi oluşturur. Serpiştiriciye ek iyileştirme uygulandığında, Turbo kodlar daha yüksek kodlama kazançlarına erişebilmektedirler. Serpiştirici uzunluğu, eşdeğer rasgele blok kod sözcük uzunluğuna denktir. Bu nedenle, Turbo kod başarımını iyileştirmenin çok etkili bir diğer yolu da serpiştirici uzunluğunu arttırmaktır. Serpiştirici uzunluğu ile orantılı olarak, kodlama ve kod çözme gecikmeleri de artar. Kodlama kuramından, kod sözcük uzunluğunun arttırılması ile daha yüksek kodlama kazancı sağlanabileceği bilinmektedir. Buna paralel olarak en büyük benzerlikli kod çözme algoritmasının (MLDA, Maximum Likelihood Decoding Algorithm) karmaşıklığı üstel olarak artmaktadır. Turbo kod çözücü, MLDA kod çözücüye göre çok daha basit yapıdadır ve başarımı, MAP kod çözücüler arasında yapılan özyinelemeli fazlalık bilgi (extrinsic information) aktarımı ile MLDA başarımına yakınsar. Turbo kod tasarımına ilişkin çalışmalar pek çok tasarımcı tarafından yapılmış ve bileşen kodların seçilmesine yönelik öneriler verilmiştir.

5.3. Turbo Kodlayıcı

Turbo kodlayıcının temel yapısını katlamalı kodlayıcılar oluşturmaktadır. Bir katlamalı kodlayıcı, girişine gelen veri dizilerini, uzunluğuna bakmaksızın bir kod kelimesine dönüştürür. Kodlayıcının temel yapısını, bellek görevi yapan kaydıran yazmaçlar, modül 2 toplayıcıları ve anahtarlar oluşturmaktadır.

Temel Turbo kodlayıcısı, iki aynı özyineli sistematik katlamalı kodun (RSC, Recursive Systematic Convolutional Code) paralel bağlanmasından oluşur [3]. Bir RSC katlamalı kod, tipik olarak R=1/2 değerine sahiptir. İki RSC kodlayıcı bir serpiştirici ile ayrılır. Şekil 5.1'de görüldüğü üzere Turbo kodlayıcıyı oluşturan iki RSC kodlayıcıdan ikinci RSC kodlayıcının girişi, ilk kodlayıcı girişinin değiştirilmiş (interleaving) şeklidir.

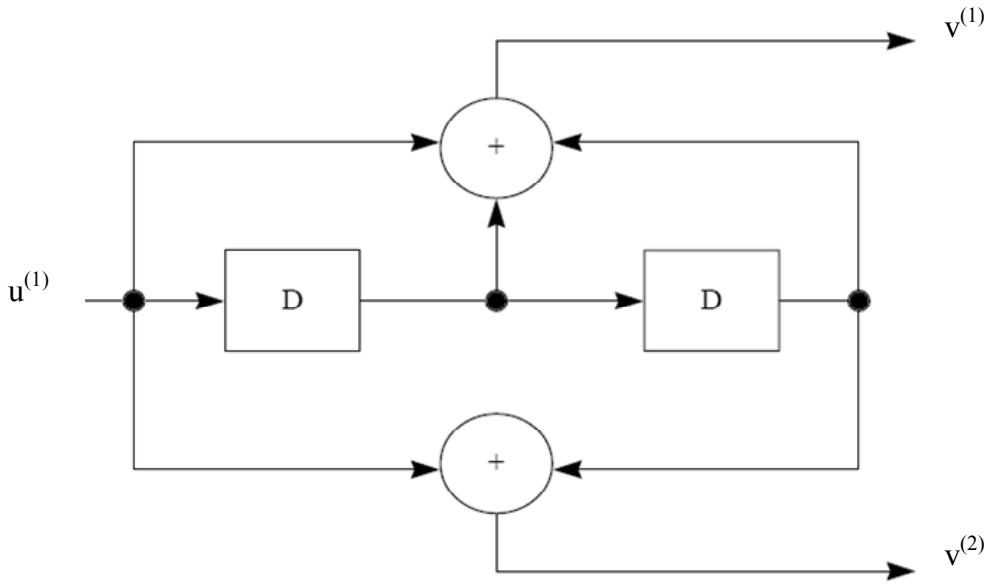


Şekil 5.1: Temel Turbo kod kodlayıcısı

Şekil 5.1, R=1/3 kod oranlı Turbo kod kodlayıcısını göstermektedir. İlk RSC kodlayıcının sistematik ($v^{(1)}$) ve eşlik kontrol çıkışı ($v^{(2)}$), diğer RSC kodlayıcının sadece eşlik kontrol çıkışı ($v^{(3)}$) kullanılarak kodlayıcı çıkış dizisi oluşturulmuştur. Bu kodlayıcıyı 1/2 kod oranlı şekle dönüştürmek için kodlayıcı çıkışlarına delikleme (puncturing) işlemi uygulanır.

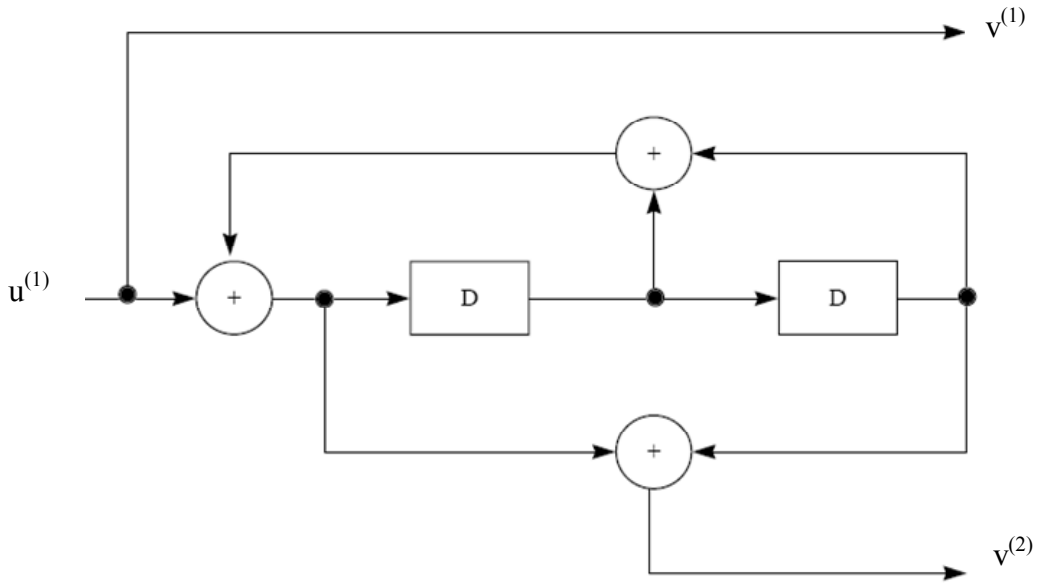
5.3.1. Özyineli sistematik katlamalı (RSC) kodlayıcı

Özyineli sistematik kodlayıcı (RSC), özyineli olmayan (klasik) katlamalı kodlayıcının çıkışlarından birinin geri besleme olarak girişine verilmesiyle elde edilmektedir. Şekil 5.2, klasik katlamalı kodlayıcıyı göstermektedir.



Şekil 5.2: Klasik katlamalı kodlayıcı (R=1/2 ve K=3)

Şekil 5.2'de gösterilen klasik katlamalı kodlayıcı g_1=[111] ve g_2=[101] üreteç dizileri ile hazırlanmaktadır. Üreteç dizileri genel şekilde gösterilecek olursa g=[g_1, g_2] şeklinde yazılır. RSC kodlayıcının üreteci g=[1, g_2/g_1] şeklinde gösterilir. Burada ilk çıkış (g_1) geri besleme olarak girişe verilmiştir. g=[1, g_2/g_1] gösteriminde 1, sistematik çıkışı, g_2 ileri besleme çıkışını, g_1 geri besleme girişini göstermektedir. Şekil 5.3, RSC kodlayıcıyı göstermektedir.

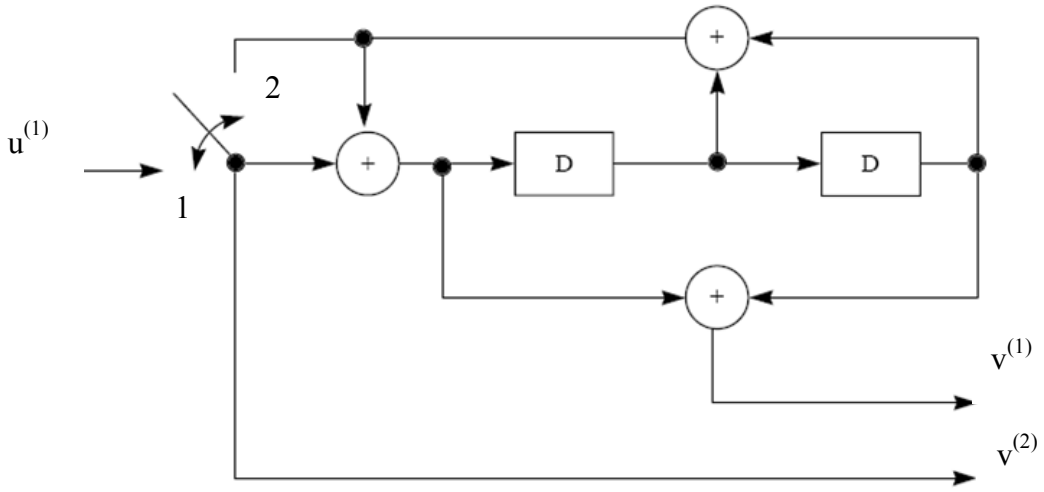


Şekil 5.3: Şekil 5.2'den elde edilen RSC kodlayıcı (R=1/2 ve K=3)

Bu tür özyineli kodlayıcılar [25]'de önerilmiştir, çünkü bu tür basit polinomlar en büyük uzunlukta diziler üretir ve Turbo koda rasgele olma özelliğini ekler. Ayrıca özyineli katlamalı kodlayıcılar düşük sinyal-gürültü oranında daha iyi başarım gösterirler [25].

5.3.2. Kafes (trellis) sonlandırma

Klasik katlamalı kodlayıcı giriş dizisinden sonra h=K-1 adet 0 girilmesiyle sonlandırılmaktadır. Fazlalık 0 bitleri, kafes diyagramının tüm durumlarını sıfıra getirmeyi sağlar (kafes sonlandırma). Bununla birlikte, RSC kodlayıcı için geri besleme söz konusu olduğundan dolayı bu işlem mümkün değildir. RSC kodlayıcı için, ilave sonlandırma bitleri kodlayıcının durumuna bağlıdır ve önceden tahmin edilmesi zordur [26]. Ayrıca, Turbo kodlayıcı yapısındaki bir kodlayıcı için sonlandırma bitleri bulunursa diğer kodlayıcı için h adet sonlandırma biti tüm durumları 0 yapmayabilir çünkü kodlayıcılar birbirinden serpiştirici ile ayrılmaktadır. Şekil 5.4'de, bu sorunun üstesinden gelmek için [26]'da basit bir strateji geliştirilmiştir.

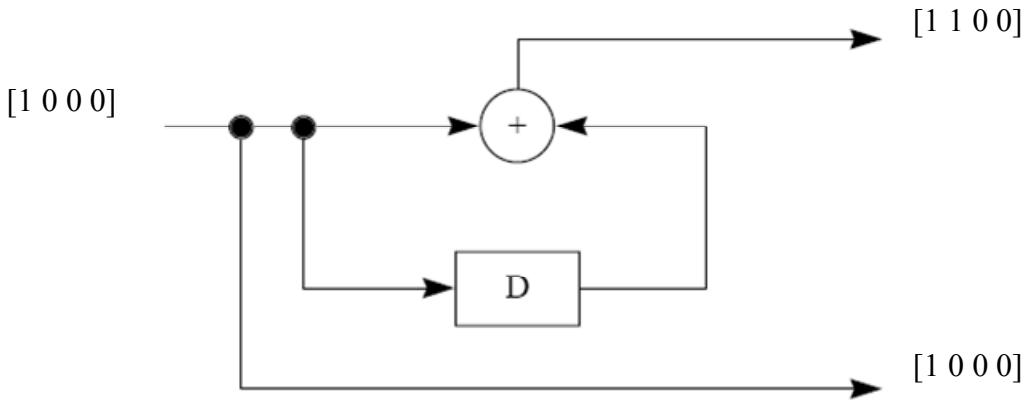


Şekil 5.4: RSC kodlayıcı için kafes sonlandırma stratejisi

Giriş dizisini kodlamak için, anahtar 1 konumuna getirilir ve kafesi sonlandırmak için anahtar 2 konumuna getirilir.

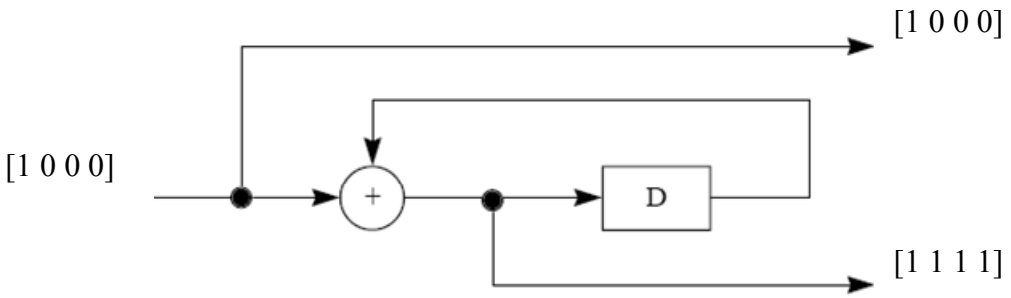
5.3.3. Özyineli (Recursive) ve özyineli olmayan (Non-Recursive) katlamalı kodlayıcılar

Şekil 5.5 basit bir özyineli olmayan katlamalı kodlayıcıyı göstermektedir. Bu kodlayıcı için üreteç dizileri, g_1=[11] ve g_2=[10] dır.



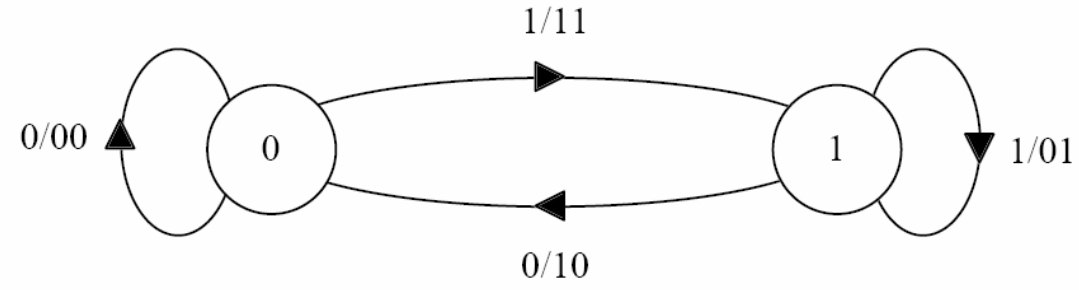
Şekil 5.5: Özyineli olmayan R=1/2 ve K=2 katlamalı kodlayıcı

Şekil 5.6'da, şekil 5.5'deki katlamalı kodlayıcıya denk özyineli kodlayıcı gösterilmektedir. Üreteç dizisi g=[1, g_2/g_1] dir.



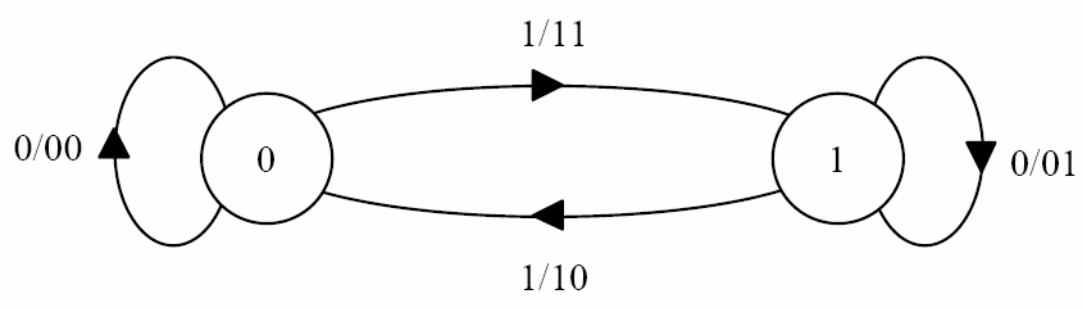
Şekil 5.6: Özyineli R=1/2 ve K=2 değerlikli katlamalı kodlayıcı

Bir sistematik kodlayıcı, zamanda ayrık ve zamanla doğrusal değişen sistemdir. Üreteç vektörleri rasyonel fonksiyonlardan oluştuğu için kodlayıcı, sonsuz dürtü yanıtlı (IIR, Infinite-Impulse-Response) doğrusal sistemdir. Sistematik olmayan kodlayıcı da sonlu dürtü yanıtlı (FIR, Finite Impulse Response) doğrusal sistem olarak ifade edilebilir. Şekil 5.5 ve şekil 5.6'da verilen aynı giriş dizilerine, özyineli olmayan kodlayıcı 3 bit ağırlığında çıkış kodu, özyineli kodlayıcı 5 bit ağırlığında çıkış kodu üretmektedir. Özyineli kodlayıcının özyineleme olmayan kodlayıcıya göre çıkış bit ağırlığı daha fazladır. Bunun bir sonucu olarak düşük ağırlıklı küçük boyutlu kod kelimeleri önlenerek daha iyi hata performansı göstermektedir. Turbo kodlar için RSC kodlayıcılar kullanmanın asıl amacı, kodlayıcıların sistematik olmasından çok özyineli olmasıdır [27]. Katlamalı kodlar için kullanılan en genel yapı, özyineli olmayan katlamalı kodlayıcıdır. Fakat bu tür kodlayıcılar sistematik olmadıkları için Turbo kodlayıcılarda kullanılmazlar. Bir kodlayıcının sistematik olması demek bir çıkışının kendi girişi olması demektir. Şekil 5.7, özyineli olmayan kodlayıcının durum diyagramını göstermektedir.



Şekil 5.7: Şekil 5.5'deki özyineli olmayan kodlayıcının durum diyagramı

Şekil 5.8, özyineli kodlayıcının durum diyagramını göstermektedir.

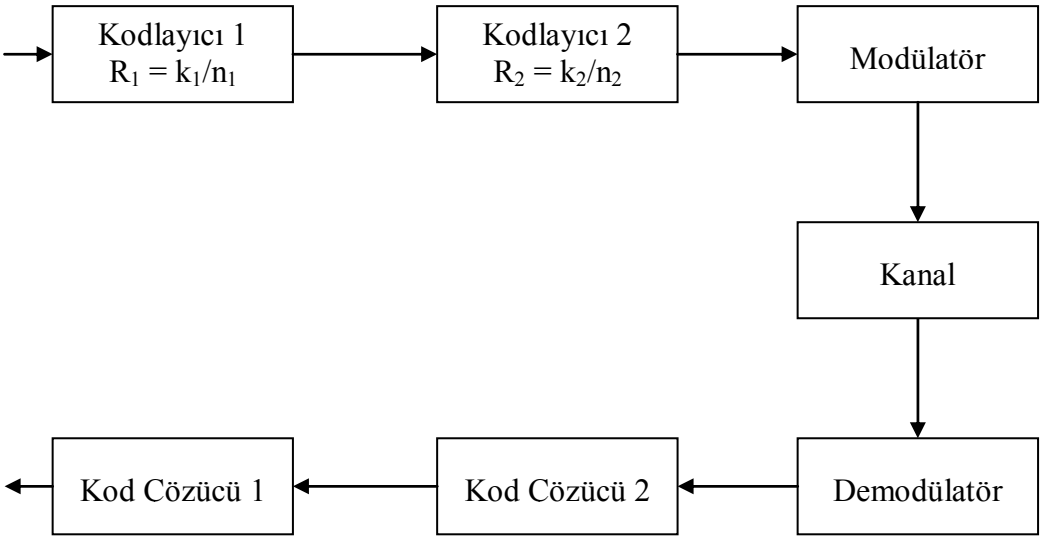


Şekil 5.8: Şekil 5.6'daki özyineli kodlayıcının durum diyagramı

Görüldüğü üzere, kodlayıcıların durum diyagramları benzerdir.

5.4. Kodların Bağlanması

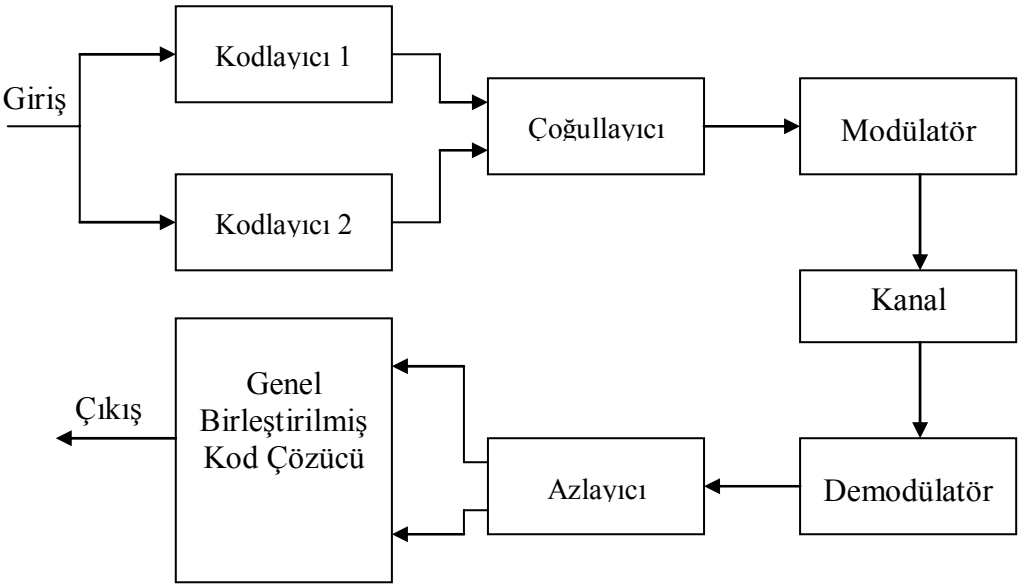
Kod bağlamadaki amaç iki farklı kodun daha büyük bir kod oluşturması için birleştirilmesidir [2]. Seri ve paralel olmak üzere iki tip kod bağlama şekli vardır. Şekil 5.9'da seri bağlama şeması gösterilmiştir.



Şekil 5.9: Seri bağlı kod yapısı

Seri bağlamada toplam kod oranı, iki kod oranının çarpımına eşittir. $R = \frac{k_1 k_2}{n_1 n_2}$ şeklindedir.

Şekil 5.10’ da paralel bağlı kod şeması görülmektedir.



Şekil 5.10: Paralel bağlı kod yapısı

Paralel bağlamada toplam kod oranı $R = \frac{k}{n_1 + n_2}$ şeklindedir.

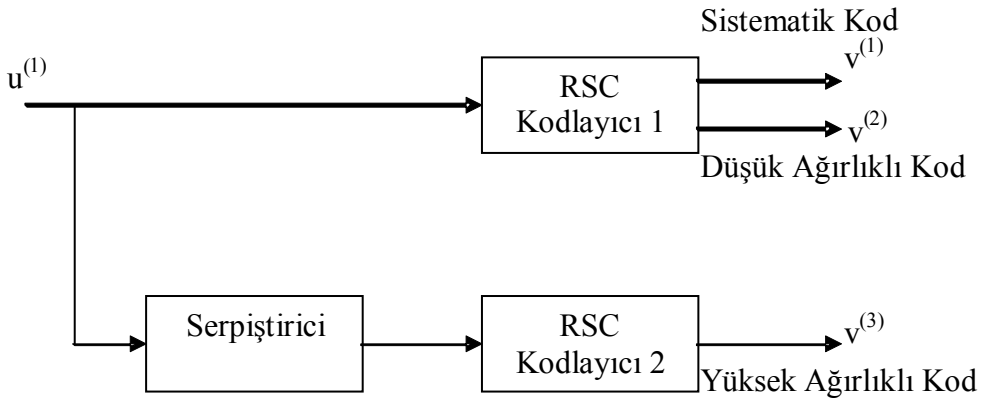
Turbo kodlar, paralel bağlı kodlamayı kullanır. Bununla birlikte Turbo kod çözücüsü seri bağlı kod çözmeyi kullanmaktadır. Seri bağlı kod çözücüler paralel bağlı kod çözücülerden daha iyi performans gösterirler. Çünkü seri bağlamada bilgi, bağlı kod çözücüler arasında paylaşılabilmekte fakat paralel bağlıda kod çözme işlemi bağımsız şekilde gerçekleştirilmektedir. Turbo kodlayıcının yapısını daha önceden de belirtildiği gibi özyineli sistematik katlamalı kodlayıcılar oluşturmaktadır.

5.5. Serpiştirici (Interleaver) Tasarımı

Serpiştirici, iki kodlayıcı arasında giriş dizisinin rasgele olma durumunu arttırmak için kullanılır. Bununla birlikte şekil 5.11’de de görüleceği üzere kod kelimelerinin ağırlıklarını arttırmak için de kullanılmaktadır. Serpiştiricinin görevi genellikle, birinci kodlayıcıda (RSC kodlayıcı 1) düşük ağırlıklı kodlayıcı çıkışı üretildiğinde diğer kodlayıcının (RSC kodlayıcı 2) çıkışını yüksek ağırlıklı şekle dönüştürmektedir. Serpiştirici kullanımında dikkat edilmesi gereken en önemli nokta, uygun olmayan serpiştirici kullanımında düşük ağırlıklı çıkışlar da oluşabilmesidir.

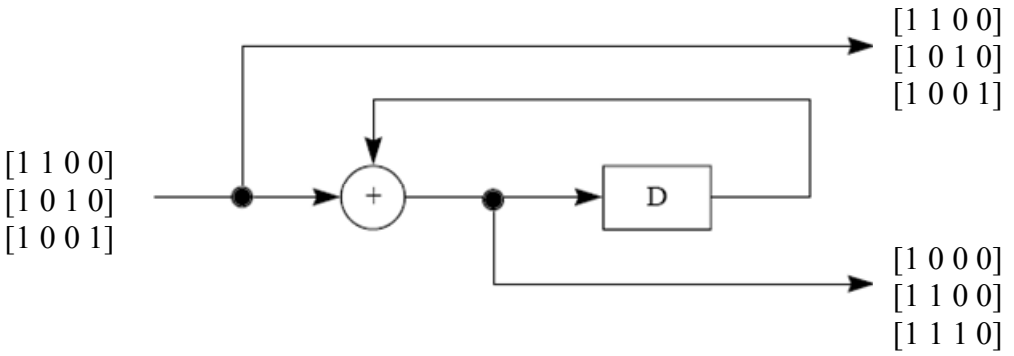
Serpiştirici aynı zamanda Turbo kod çözücüsü yapısında da kullanılmaktadır. Döngülü kod çözme işlemi sırasında serpiştirici kod çözücülerin yumuşak çıkışlarına karşılık gelen eşlik bitleri ile kodlayıcı bilgi giriş dizileri arasındaki ilişkiyi sağlayarak kod çözme işleminin gerçekleşmesini sağlamaktadır. Bu işlem döngülü kod çözme işleminin verimliğini etkilemektedir. Kodlayıcı girişleri birbirinden ne kadar az ilişkili ise döngülü kod çözme algoritmasının başarımı o kadar artmış olur.

Kısa bilgi blokları ile kullanılan Turbo kodların başarımında serpiştirici tasarımı oldukça önemlidir. Geniş bilgi blokları ile kullanılan Turbo kodlarda genellikle rasgele serpiştiriciler kullanılmakta ve yüksek başarım sağlanmaktadır.



Şekil 5.11: Serpiştirici RSC 1 kodlayıcısına göre RSC 2 kodlayıcısının kod ağırlığı

Şekil 5.11'de, giriş dizisi $u^{(1)}$, RSC 1 kodlayıcısı için düşük ağırlıklı özyineli katlamalı kod dizisi $v^{(2)}$ yi üretir. RSC 2 kodlayıcısının diğer bir düşük ağırlıklı özyineli çıkış dizisi üretmesini engellemek için serpiştirici giriş dizisi $u^{(1)}$'i değiştirerek yüksek ağırlıklı özyineli katlamalı kod dizisi $v^{(3)}$'ü üretmeyi sağlamaktadır. Böylece Turbo kodun kod ağırlığı dengelenmiş olmaktadır. Bu durum Şekil 5.12'de bir örnekle gösterilmiştir.



Şekil 5.12: Serpiştirici yeteneğini gösteren bir örnek

Şekil 5.12'de, giriş dizisine karşılık, çıkış dizileri üretilmiştir. Tablo 5.1, kod kelimeleri ve ağırlıklarını göstermektedir.

Tablo 5.1: Şekil 5.12'deki kodlayıcı için giriş ve çıkış dizileri

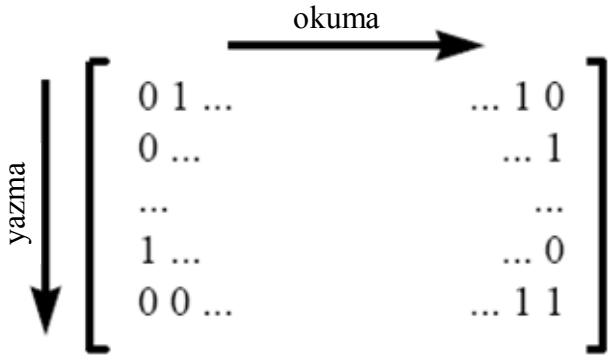
Giriş Dizisi	**Çıkış Dizisi** (üst çıkış)	**Çıkış Dizisi** (alt çıkış)	**Kod Kelime Ağırlığı**
1 1 0 0	1 1 0 0	1 0 0 0	3
1 0 1 0	1 0 1 0	1 1 0 0	4
1 0 0 1	1 0 0 1	1 1 1 0	5

Tablo 5.1'den de görüleceği üzere, kod kelime ağırlıkları bir serpiştirici kullanılarak arttırılmıştır. Burada, düşük ağırlıklı kod kelimeleri önlenerek Turbo kodun bit hata oranı önemli derecede düşürülmüştür. Bu yüzden serpiştirici tasarımı çok önemlidir.

5.6. Serpiştirici Çeşitleri

5.6.1. Blok serpiştirici

Blok serpiştirici, haberleşme sistemlerinde en çok kullanılan serpiştiricilerdir. Blok serpiştiriciler, sütun yönünde yukarıdan aşağıya ve soldan sağa yazar, satır yönünde soldan sağa ve yukarıdan aşağıya okur. Şekil 5.13'de bir blok serpiştirici görülmektedir.

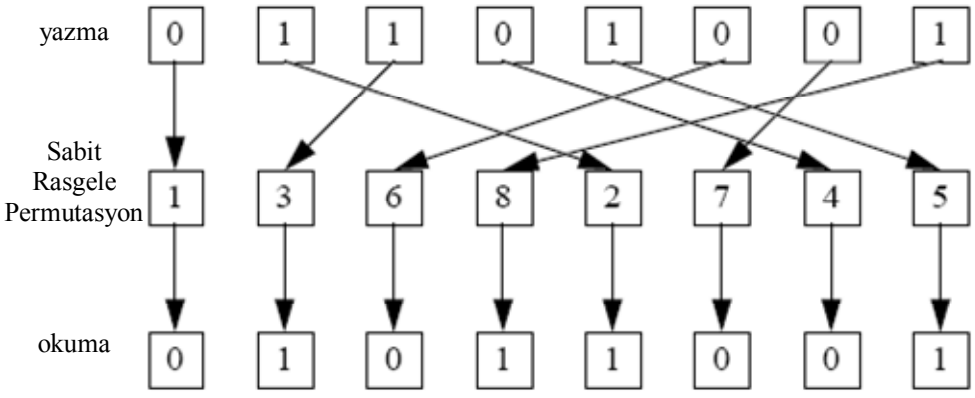


Şekil 5.13: Blok serpiştirici

Şekil 5.1.3'de serpiştirici [0 0 … 1 0 1…0…1…101…01] yazar ve [0 1…1 0 0…1…1…0 0 0…1 1] okur.

5.6.2. Rasgele (sözde rasgele) serpiştirici

Rasgele serpiştirici, giriş dizisini, sabit bir rasgele permutasyon kullanarak ve permutasyon sırasına göre eşleme yaparak serpiştirir. Şekil 5.14, uzunluğu 8 olan rasgele serpiştirici gösterilmektedir.



Şekil 5.14: Rasgele(sözde-rasgele) serpiştirici

Şekil 5.14 'de serpiştirici [0111001] yazar ve [01011001] okur. Bu tezde uzun bilgi blokları kullanıldığı için benzetimlerde rasgele serpiştirici kullanılmıştır.

5.6.3. Tek-çift serpiştirici

Tek-çift serpiştirici, kod oranı 1/2 olan Turbo kodlar için tasarlanmıştır. 1/2 kod oranlı Turbo kodlar 1/3 oranlı Turbo kodların delme (puncture) işleminden geçirilmesi ile elde edilir. Delme işlemi Turbo kodlayıcı çıkışlarında bazı bilgilerin delme matrisleri sayesinde kodlayıcı çıkış dizisinden atılması ile gerçekleştirilir. Bu atılan bilgiler sonucunda, kodlayıcıda oluşturulan kodlanmış diziler ve bu kodlanmış diziler ile ilişkili olan kodlayıcı bilgi giriş dizileri arasındaki ilişki bozulabilir. Sonuçta Turbo kod çözücüsünün başarımı düşmüş olabilmektedir. Tek-çift serpiştirici tasarımı bu problemin üstesinden gelmektedir [28].

Burada üç çeşit serpiştiriciden bahsedilmiştir. Daha birçok serpiştirici çeşidi mevcuttur.

5.7. Sonuç

Bu bölümde Turbo kod yapısından ve katlamalı kodların bağlanmasından bahsedilmektedir. Turbo kodlar için paralel bağlı kodlar kullanılmaktadır. Ayrıca Turbo kodların yapısında bulunan ve Turbo kodların başarımını büyük ölçüde etkileyen serpiştirici yapısıdır. Uzun boyutlu kod bloklarının kullanılmasında rasgele serpiştirici en iyi başarımı göstermektedir.

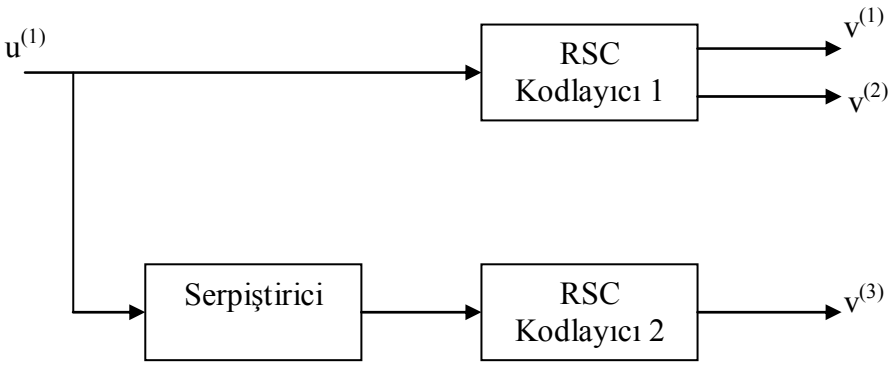
6. TURBO KOD ÇÖZME VE BAŞARIM ANALİZİ

6.1. Giriş

Turbo kodların yayımlandığı ilk makalede [3], Berrou 1974 yılında Bahl tarafından [29] hazırlanan döngülü kod çözme şemasına dayalı algoritmayı önermiştir. Bahl kod çözme algoritmasının Viterbi algoritmasından farkı yumuşak çıkışlar üretmesidir. Yumuşak çıkışlar elde edilmek üzere Viterbi algoritması geliştirilerek Turbo kod çözme algoritması olarak SOVA algoritması oluşturulmuştur. Viterbi algoritması her bir tahmini (kod çözücü çıkışı) bit için 0 ya da 1 çıkışı üretirken, Bahl algoritması, her bir tahmini bit için logaritmik olabilirlik çıkışlar üretir. Viterbi kod çözücüsünün amacı, iletilmiş kod kelimesinin kod kelime hatasını ML tahminini bularak en küçüklemektir. Bahl algoritması, kod kelimesindeki bitlerin APP'sini tahmin ederek bit hatasını en küçüklemeyi amaçlamaktadır.

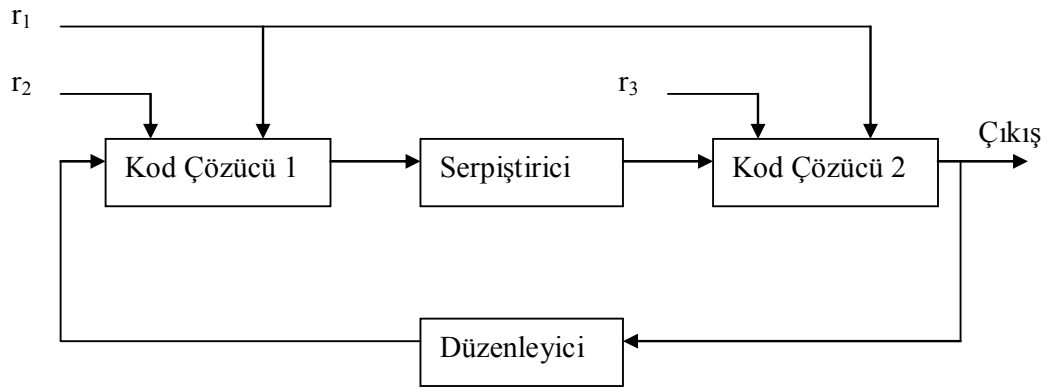
Bu bölümde, Turbo kod çözme algoritma yapısı ve SOVA kod çözme algoritması hakkında kapsamlı bilgi verilip kullanılan matematiksel yapılardan söz edilmektedir. Aynı zamanda, SOVA algoritmasının benzetim modeli anlatılmaktadır.

6.2. Turbo Kodlayıcı ve Kod Çözücü Yapısı



Şekil 6.1: Turbo kodlama şeması

Şekil 6.1'de Turbo kodlayıcı yapısı gösterilmektedir. Burada $v^{(1)}$, kodlayıcı giriş bilgi biti, $v^{(2)}$ kodlayıcı 1 çıkışında oluşan eşlik biti ve $v^{(3)}$ kodlayıcı 2 çıkışında oluşan eşlik bitidir.



Şekil 6.2: Turbo kod çözme şeması

Şekil 6.2'de genel Turbo kod çözme şeması gösterilmiştir. İki kod çözücü, Turbo kodlayıcı yapısına benzer olarak bir serpiştirici ile birbirine bağlanmıştır. Şekil 6.2'den de görüldüğü üzere her bir kod çözücünün üç girişi bulunmaktadır. Bu girişler, kodlayıcı giriş bilgi dizisine denk gelen sistematik kodlanmış kanal çıkış bitleri, kod çözücü ile ilişkilendirilmiş kodlayıcıdan gelen eşlik bitleri ve diğer kod çözücünün çıkış bitleridir. Kod çözücülerin üçüncü giriş bit dizisi, önceki bilgi (priori information) olarak isimlendirilir ve bu kod çözücünün üç girişi de yumuşak çıkışlar olmak zorundadır. Bu yumuşak çıkışlar, logaritmik olabilirlik (LLR, Log-Likelihood Ratio) oranları ile ifade edilmektedir. LLR hakkında bilgi SOVA ve log-MAP algoritmaları kısmında anlatılacaktır.

Şekil 6.2'deki kod çözücü döngülü şekilde çalışmaktadır ve ilk döngüde birinci kod çözücü, sadece kanal çıkış değerleri alır (ikinci kod çözücünün çıkışları 0 varsayılır) ve o veri bitlerine ait tahmini yumuşak çıkışları üretir. Birinci kod çözücünün yumuşak çıkışı, ikinci kod çözücü için ek bilgi olarak kullanılır ve bu bilgi kanal çıkışları ile kullanılarak tahmini veri bitlerini hesaplar. Birinci döngü bittikten sonra ikinci döngü başlar ve birinci kod çözücü kanal çıkışlarının kodlarını tekrar çözer, fakat birinci döngüdeki ikinci kod çözücünün çıkışları sayesinde ek bilgi ile giriş bitlerinin değeri değiştirilmiş olur. İkinci kod çözücü için önceki bilgi olarak

kullanılan bu ek bilgi, birinci kod çözücünün daha doğru yumuşak çıkışlar elde etmesini sağlar. Bu döngüler istenildiği kadar devam ettirilir ve her döngü bit hata oranının düşmesini sağlar.

Kodlayıcı devresinde serpiştirici kullanılmasından dolayı, LLR değerlerinin serpiştirilmesinde (interleaving) ve tekrar düzenlenmesinde (de-interleaving) aynı serpiştirici yapısı göz önünde bulundurulmalıdır.

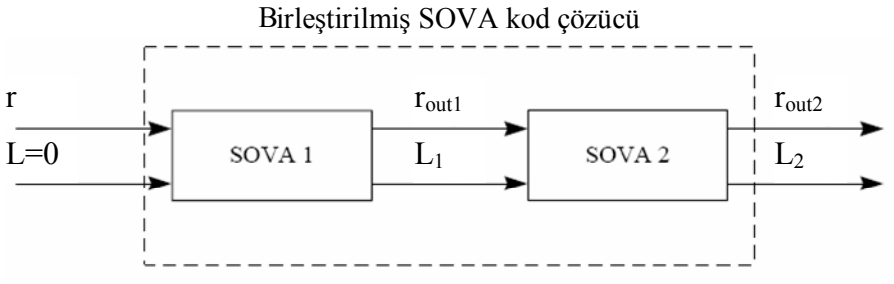
6.3. Turbo Kod Çözme Algoritmaları

Turbo kod çözme algoritmaları, Viterbi algoritması temel alınıp bu algoritmanın kod çözme başarımı arttırılarak gerçekleştirilmiştir. Turbo kod çözücüsü iki adet kod çözücünün seri bağlanmasından meydana gelmektedir. Seri bağlı bu kod çözücüler arasında tahmini çıkış bit dizisi döngülü bir şekilde iletilmektedir. Döngülü Turbo kod çözücü için bu çalışmada Log-MAP ve SOVA algoritmaları anlatılacaktır.

6.4. Yumuşak Çıkışlı Viterbi Kod Çözme Algoritması (SOVA)

Viterbi algoritması, katlamalı kodlar için en büyük olasılıklı ML çıkış dizisini üretmektedir. Bu algoritma tek aşamalı (one-stage) katlamalı kodlar için en uygun olası çıkış dizisini üretir. Viterbi kod çözme algoritmasının birleştirilmiş (çok aşamalı, concatenated, multistage) katlamalı kodlar için, iki dezavantajı vardır [21]. İlk olarak şekil 6.1'de içteki Viterbi kod çözücü (SOVA2, Soft Output Viterbi Algorithm 2), dıştaki Viterbi kod çözücünün (SOVA1, Soft Output Viterbi Algorithm 1) başarımını düşüren patlama hatalarını (burst error) meydana getirir [21]. İkinci olarak, içteki Viterbi kod çözücü, dıştaki Viterbi kod çözücünün yumuşak kararlılığından oluşan faydalarını engelleyerek sıfır-bir kararlı çıkışlar üretir. Eğer Viterbi kod çözücü doğru kararlar (yumuşak çıkış) üretebilirse, birleştirilmiş kod çözücünün başarımı arttırılabilir ve bu iki dezavantaj giderilebilir [30]. Doğru değerler (yumuşak çıkışlar), önceki bilgi (priori information) olarak sonraki Viterbi kod çözücüye geçerek kod çözme başarımını arttırmaktadır. Bu değiştirilmiş Viterbi kod çözme algoritması, yumuşak çıkışlı Viterbi kod çözme

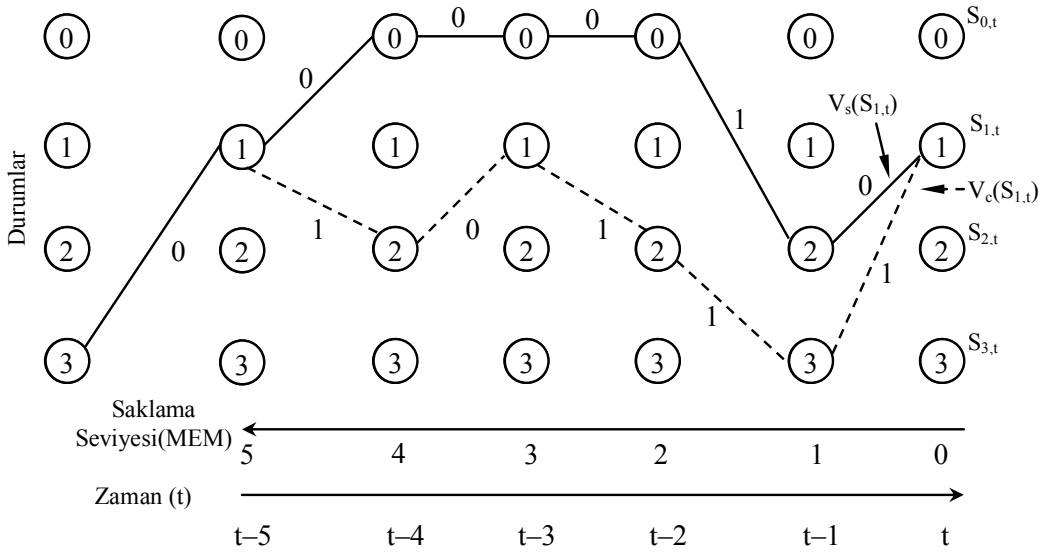
algoritması (SOVA, Soft Output Viterbi Algorithm) olarak isimlendirilir. Şekil 6.3'de birleştirilmiş kod çözücü gösterilmektedir.



Şekil 6.3: Alınan kanal değerleri (r), sıfır-bir kararlı çıkış değerleri (r_{out1} ve r_{out2}) ve ilişkilendirilmiş güvenilirlik değerleri (L) gösteren birleştirilmiş SOVA kod çözücü

6.4.1. SOVA kod çözücünün güvenirliği

SOVA kod çözücünün güvenirliği şekil 6.4'deki kafes diyagramından hesaplanabilir.

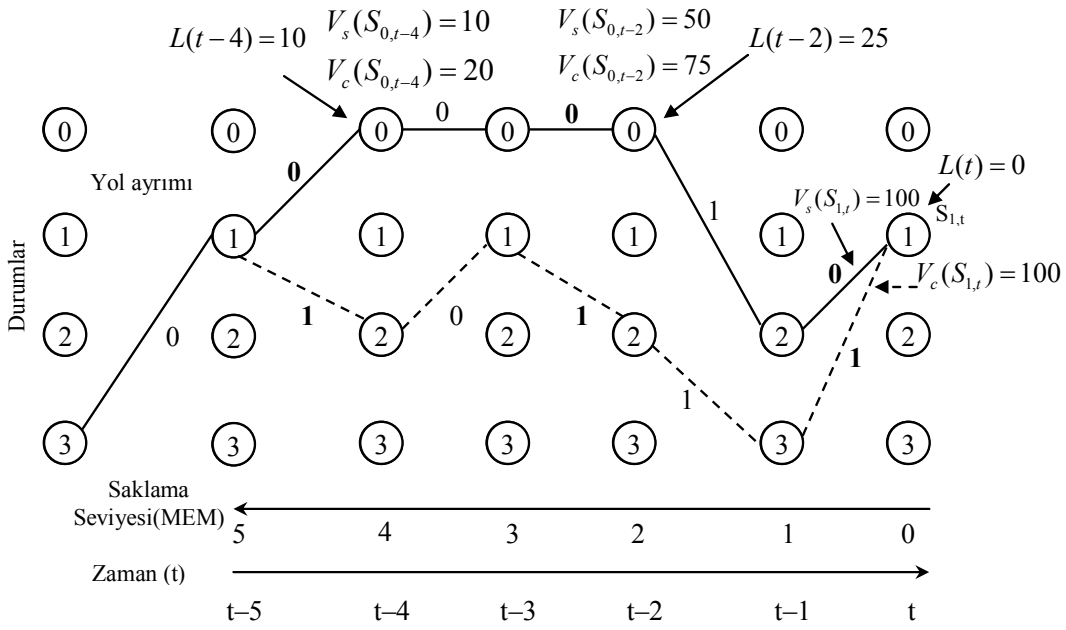


Şekil 6.4: t anında güvenilirlik tahmini için yarışan ve kazanan yolları gösteren kafes diyagramı örneği

Şekil 6.4'de, dört durumlu kafes diyagramı gösterilmektedir. Düz çizgiler kazanan yolu göstermektedir ve kesikli çizgiler ise t anında durum 1 için yarışan yolu göstermektedir. Şeklin daha fazla karmaşık görünmesini engellemek için diğer durumların kazanan ve yarışan yolları gösterilmemiştir. $S_{1,t}$ 1. durum ve t anını ifade

eden düğümü göstermektedir. Bununla birlikte {0,1} her bir yol için tahmini ikili kararları göstermektedir. $S_{1,t}$ düğümü için kazanan yol, $V_s(S_{1,t})$ ile ifade edilen biriktirilmiş metriğe ve aynı düğüm için yarışan yol, $V_c(S_{1,t})$ ile ifade edilen biriktirilmiş metriğe atanmaktadır. İki biriktirilmiş metrik farkının mutlak değeri, $L(t) = \left| V_S(S_{1,t}) - V_C(S_{1,t}) \right|$, $S_{1,t}$ düğümünün kazanan yolunun güvenilirlik değeri olarak adlandırılır. Bu fark ne kadar büyük olursa kazanan yol daha güvenilirdir denir. Bu güvenilirlik hesabı için, kazanan biriktirilmiş metrik, her zaman yarışan biriktirilmiş metrikten daha iyi varsayılır. Ayrıca, karmaşıklığı azaltmak için, burada sadece ML kazanan yolu için güvenilirlik değeri hesaplanmıştır.

Güvenilirlik kavramını açıklamak için iki örnek anlatılacaktır. Bu örneklerdeki Viterbi algoritması küçük olan biriktirilmiş metriği kazanan yol olarak seçer. Birinci örnekte, biriktirilmiş kazanan metrik olarak $V_s(S_{1,t})=50$ ve biriktirilmiş yarışan metrik olarak $V_c(S_{1,t})=100$ alınsın. Bu değerler için güvenilirlik değeri $L(t) = \left| 50 - 100 \right| = 50$ dir. İkinci örnekte, biriktirilmiş kazanan metrik aynı bırakılarak $V_c(S_{1,t})=75$ alınsın. Bu değerler için güvenilirlik değeri $L(t) = \left| 50 - 75 \right| = 25$ olarak hesaplanacaktır. İlk örnekteki güvenilirlik değeri daha büyük olduğundan ikinci örneğe göre kazanan yolun seçiminde ilk örnek daha doğru bir sonuç verecektir. Şekil 6.5'de, biriktirilmiş kazanan ve yarışan metriklerin mutlak değer farkı ile ilgili bir problem gösterilmektedir.



Şekil 6.5: Metrik değerlerini doğrudan kullanarak atamadan kaynaklanan güvenilirlik zayıflığını gösteren örnek

Şekil 6.5'de, $S_{1,t}$ düğümündeki kazanan ve yarışan yollar t–5 anında birbirlerinden ayrılmıştır. t, t–2 ve t–4 anlarında kazanan ve yarışan yollar, koyu olarak gösterilen farklı tahmini ikili kararlar üretmektedir. Örneği açıklamak için, $S_{1,t}$ düğümünde biriktirilmiş kazanan ve yarışan metriklerin eşit olduğunu ($V_S(S_{1,t})=V_C(S_{1,t})=100$) varsayalım. Bu değerlerin eşit olması, kod çözme işlemi sonucunda elde edilecek ML yolunun bulunmasında kazanan ve yarışan yolların olma olasılığının eşit olduğunu ifade etmektedir. Ayrıca, şekil 6.5'de gösterildiği üzere, t–2 ve t–4 anlarında biriktirilmiş kazanan metriğin, biriktirilmiş yarışan metrikten daha iyi olduğunu varsayalım. Şekil karmaşıklığını azaltmak için, t–2 ve t–4 anlarındaki yarışan metrikler gösterilmemiştir. Şekil 6.5'den t anında L(t)=0 olduğu görülmektedir. Bu değer, burada güvenilirlik değerinin kazanan yolun seçiminde herhangi bir etkisinin olmadığını göstermektedir. t–2 ve t–4 anlarında, kazanan yollara 0'dan farklı değerler atanmıştır. Bununla birlikte, t anında, yarışan yol ile kazanan yol aynı değere sahip olduğu için yarışan yol kazanan yol olabilir. Bu nedenle t, t–2 ve t–4 anlarında kazanan yol boyunca ilişkilendirilmiş güvenilirlik değerlerini azaltılmaksızın farklı tahmini ikili kararlar olabilmektedir.

Bu problemin üstesinden gelmek için yani kazanan yolun güvenilirlik değerini arttırmak ve güvenilirlik değerlerini güncellemek amacıyla geri izleme (trace back) işlemi önerilmiştir [21,30]. Bu güncelleme işlemi, Viterbi algoritması ile birleştirilmiştir [21]. Viterbi algoritması ile geri izleme işlemi birleştirildiğinde oluşan algoritma şu şekilde ifade edilebilir;

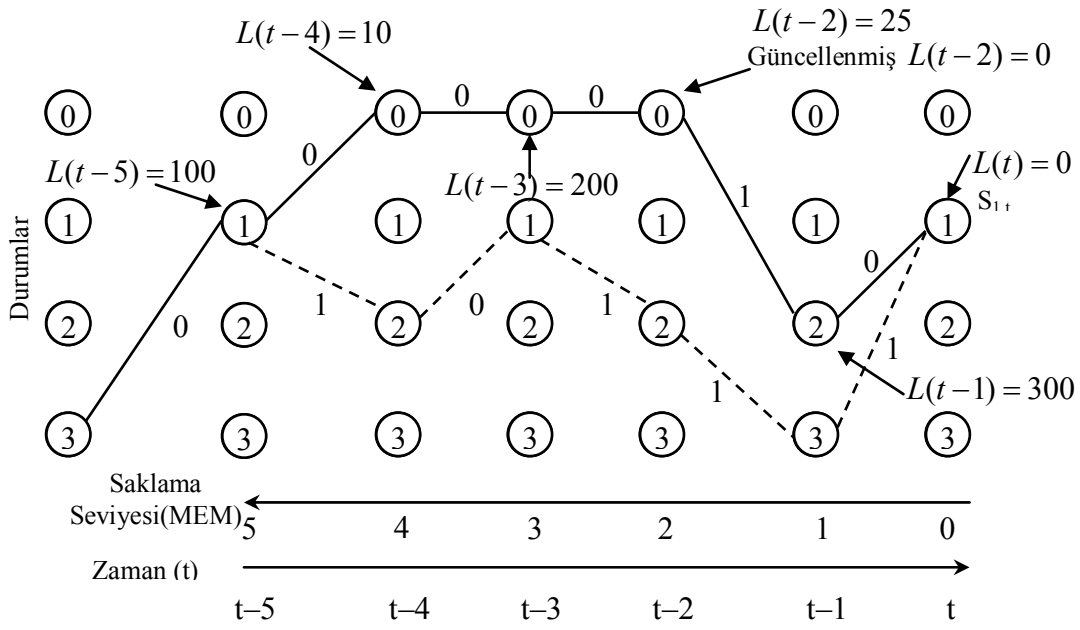
$S_{k,t}$ düğümü için, kafes diyagramında,

1. $L(t) = \left| V_S(S_{1,t}) - V_C(S_{1,t}) \right|$ değeri saklanır (Bazı yayınlarda L(t), Λ olarak da gösterilmiştir). Birden fazla yarışan yol varsa, tüm güvenilirlik değerleri hesaplanmak zorundadır ve en küçük güvenilirlik değeri L(t)'ye atanır.
2. $S_{k,t}$ 'nin güvenilirlik değeri, $+\infty$ olarak başlangıçta atanır (daha çok güvenilir anlamına gelir).
3. $S_{k,t}$ durumundaki kazanan ve yarışan yollar karşılaştırılır ve farklı iki yolun tahmini ikili kararları olan saklama seviyeleri (memorization levels) saklanır.
4. Aşağıdaki metoda göre bu saklama seviyelerindeki güvenilirlik değerleri güncellenir:

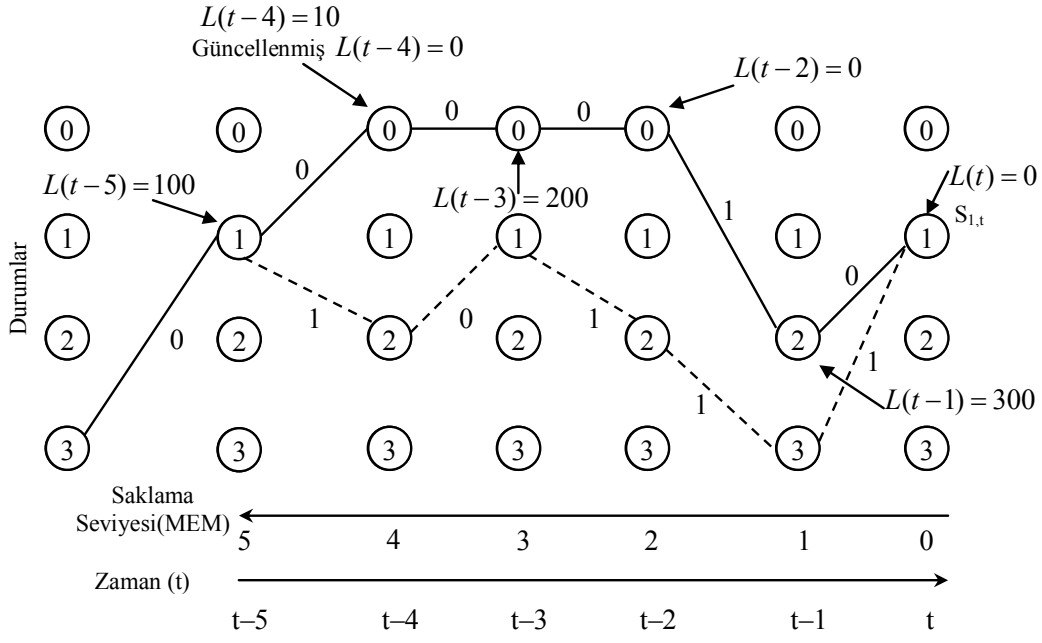
a) MEM_{low} olarak gösterilen en küçük saklama seviyesi (MEM>0) değeri bulunur (henüz güvenilirlik değeri güncellenmemiştir).

b) MEM=0 ve MEM=MEM_{low} arasındaki en küçük güvenilirlik değerinin atanması ile MEM_{low} 'un güvenilirlik değeri olan L(t-MEM_{low}) güncellenir.

Örneğe devam edilecek olursa, $S_{1,t}$ durumu için kazanan ve yarışan bit yolları arasındaki zıt bit tahminleri yerleştirilir ve MEM={0, 2, 4} olarak saklanır. Bu MEM bilgisi ile güvenilirlik güncelleme işlemi, şekil 6.6 ve şekil 6.7 'da gösterildiği üzere tamamlanmış olur. Şekil 6.6'de ilk güvenilirlik güncellemesi gösterilmektedir. Güvenilirlik değeri güncellenmemiş olan en küçük MEM>0, MEM_{low}=2 olarak saptanır. MEM=0 ve MEM=MEM_{low}=2 arasındaki güvenilirlik değeri L(t)=0 yapılır. Böylece ilişkilendirilmiş güvenilirlik değeri L(t–2)=25'den L(t–2)=L(t)=0'a güncellenir. Güncellenmemiş sonraki en düşük MEM>0 güvenilirlik değeri, MEM_{low}=4 olarak saptanır. MEM=0 ve MEM=MEM_{low}=4 arasındaki en düşük güvenilirlik değeri L(t–2)=L(t)=0 olur. Böylece ilişkilendirilmiş güvenilirlik değeri, L(t-4)'den L(t-4)=L(t)=L(t-2)=0 'a güncellenmiş olur. Şekil 6.7, ikinci güvenilirlik güncellemesini göstermektedir.



Şekil 6.6: t–2 anındaki (MEMlow=2) güncelleme işlemi



Şekil 6.7: t-4 anındaki (MEM_{low}=4) güncelleme işlemi

6.4.2. SOVA algoritması

Turbo kodlar için kullanılan SOVA kod çözme algoritması, değiştirilmiş Viterbi metriğini kullanarak oluşturulmaktadır. Değiştirilmiş Viterbi metriğini elde etmeden önce, logaritmik olabilirlik matematiğini (log-likelihood algebra) ve yumuşak kanal çıkışlarını (soft channel outputs) incelemek gerekmektedir. Şekil 6.8'de bu kavramları açıklayan sistem modeli gösterilmektedir.

Şekil 6.8: SOVA algoritmasını türetmek için kullanılan sistem modeli

Giriş bilgisi (u) kanal kodlayıcıya girerek kodlanır ve gürültülü iletim kanalından geçerek hatalı bir şekilde kanal kod çözücüsüne aktarılır. Kanal kod çözücüsünün çıkışında tahmini bit dizisi (u') elde edilmektedir.

6.4.3. Logaritmik olabilirlik matematiği

SOVA için kullanılan logaritmik olabilirlik matematiği, ikili rasgele değişken (u)'in, +1'in lojik 0 ve -1'in lojik 1 ile ifade edildiği GF(2) altında modüle 2 işlemiyle toplanmasıdır. Tablo 6.1'de bu faktörler altında iki adet ikili rasgele değişkenin toplama sonuçları gösterilmektedir.

Tablo 6.1: u_1 ve u_2 ikili rasgele değişkenlerinin toplamları

$u_1 \oplus u_2$	$u_2 = +1$	$u_2 = -1$
$u_1 = +1$	+1	-1
$u_1 = -1$	-1	+1

İkili rasgele değişken u için $L(u)$ ile ifade edilen logaritmik olabilirlik oranı şu şekilde tanımlanmaktadır;

$$L(u) = \ln \frac{P(u=+1)}{P(u=-1)} \tag{6.1}$$

$L(u)$, ikili rasgele değişken u'nun L değeri yani yumuşak değeri olarak ifade edilmektedir. $L(u)$'nun işareti u'nun sıfır-bir kararıdır ve $L(u)$'nun büyüklüğü bu kararının güvenilirliğidir. Tablo 6.2 logaritmik olabilirlik oranı $L(u)$'nun karakteristiklerini göstermektedir.

Tablo 6.2: $L(u)$ karakteristikleri

$P(u=+1)$	$P(u=-1)$	$L(u)$
1	0	$+\infty$
0.9999	0.0001	9.2102
0.9	0.1	2.1972
0.6	0.4	0.4055
0.5	0.5	0
0.4	0.6	-0.4055
0.1	0.9	-2.1972
0.0001	0.9999	-9.2102
0	1	$-\infty$

Tablo 6.2'den de görüleceği üzere $L(u)$, $+\infty$'a doğru gittikçe u=+1 olma olasılığı da artmaktadır. Aynı şekilde, $L(u)$, $-\infty$'a doğru gittikçe u=-1 olma olasılığı da artmaktadır.

Rasgele değişken olan u'nun olasılığı diğer bir rasgele değişken z ile koşullu olabilmektedir. Bu şekilde oluşturulan forma koşullu logaritmik olabilirlik oranı denir ve $L(u|z)$ şeklinde gösterilmektedir.

$$L(u|z) = \ln \frac{P(u=+1|z)}{P(u=-1|z)} \tag{6.2}$$

İki adet ikili rasgele değişkenin toplam olasılığı, $P(u_1 \oplus u_2 = +1)$, şu şekilde bulunur;

$$P(u_1 \oplus u_2 = +1) = P(u_1 = +1)P(u_2 = +1) + P(u_1 = -1)P(u_2 = -1) \tag{6.3}$$

Bu denkleme aşağıdaki ilişki eklenerek,

$$P(u = -1) = 1 - P(u = +1) \tag{6.4}$$

$P(u_1 \oplus u_2 = +1)$ olasılığı aşağıdaki şekli alır;

$$P(u_1 \oplus u_2 = +1) = P(u_1 = +1)P(u_2 = +1) + (1 - P(u_1 = +1))(1 - P(u_2 = +1)) \tag{6.5}$$

[31]'de gösterilen ilişki,

$$P(u = +1) = \frac{e^{L(u)}}{1 + e^{L(u)}} \tag{6.6}$$

Denklem (6.6) dikkate alınarak

$$P(u_1 \oplus u_2 = +1) = \frac{1 + e^{L(u_1)}e^{L(u_2)}}{(1 + e^{L(u_1)})(1 + e^{L(u_2)})} \tag{6.7}$$

$P(u_1 \oplus u_2 = -1)$ olasılığı şu şekilde hesaplanır,

$$P(u_1 \oplus u_2 = -1) = 1 - P(u_1 \oplus u_2 = +1) \tag{6.8}$$

$$P(u_1 \oplus u_2 = -1) = \frac{e^{L(u_1)} + e^{L(u_2)}}{(1 + e^{L(u_1)})(1 + e^{L(u_2)})} \tag{6.9}$$

Denklem (6.1)'den logaritmik olabilirlik oranı doğrudan yazılabilir;

$$L(u_1 \oplus u_2) = \ln \frac{P(u_1 \oplus u_2 = +1)}{P(u_1 \oplus u_2 = -1)} \tag{6.10}$$

Denklem (4.7) ve (4.9)'daki denklemler kullanılarak;

$$L(u_1 \oplus u_2) = \ln \frac{1 + e^{L(u_1)} e^{L(u_2)}}{e^{L(u_1)} + e^{L(u_2)}} \tag{6.11}$$

olarak bulunur. [32]'de bu sonuç yaklaşık olarak;

$$L(u_1 \oplus u_2) \approx sign(L(u_1)) sign(L(u_2)) \min(\left|L(u_1)\right|, \left|L(u_2)\right|) \tag{6.12}$$

elde edilir. Tablo 6.3, gerçek çözüme kıyasla bu yaklaşımın doğruluğunu göstermektedir.

Tablo 6.3'den görüleceği üzere, her iki değişken için kararın güvenirliği 0'a yaklaştıkça gerçek ve yaklaşık sonuçlar arasındaki sapma da artmaktadır.

Tablo 6.3: $L(u_1 \oplus u_2)$ 'nin kesin ve yaklaşık değerlerinin karşılaştırılması

$L(u_1)$	$L(u_2)$	$L(u_1 \oplus u_2)$ (6.11) Kesin Değer	$L(u_1 \oplus u_2)$ (6.12) Yaklaşık Değer
0	-1000	0	0
0	-100	0	0
0	-10	0	0
0	-1	0	0
0	0	0	0
0	1	0	0
0	10	0	0
0	100	0	0
0	1000	0	0
1	-1000	-1	-1
1	-100	-1	-1
1	-10	-0.9999	-1
1	-1	-0.4338	-1
1	0	0	0
1	1	0.4338	1
1	10	0.9999	1

Tablo 6.3 (devamı): $L(u_1 \oplus u_2)$ 'nin kesin ve yaklaşık değerlerinin karşılaştırılması

1	100	1	1
1	1000	1	1
10	-1000	-10	-10
10	-100	-10	-10
10	-10	-9.3069	-10
10	-1	-0.9999	-1
10	0	0	0
10	1	0.9999	1
10	10	9.3069	10
10	100	10	10
10	1000	10	10
100	-1000	-100	-100
100	-100	-99.3069	-100
100	-10	-10	-10
100	-1	-1	-1
100	0	0	0
100	1	1	1
100	10	10	10
100	100	99.3069	100
100	1000	100	100
1000	-1000	-999.3069	-1000
1000	-100	-100	-100
1000	-10	-10	-10
1000	-1	-1	-1
1000	0	0	0
1000	1	1	1
1000	10	10	10
1000	100	100	100
1000	1000	999.3069	1000

$$L(u_1)[+]L(u_2) = L(u_1 \oplus u_2) \tag{6.13}$$

İki yumuşak (L değerleri) değerin toplamı denklem (6.13) ile tanımlanır ve bununla birlikte aşağıdaki üç özelliğe sahiptir.

$$L(u)[+]\infty = L(u) \tag{6.14}$$

$$L(u)[+](-\infty) = -L(u) \tag{6.15}$$

$$L(u)[+]0 = 0 \tag{6.16}$$

Genel olarak yazılacak olursa;

$$\sum_{\substack{[+]\\ j=1}}^{J} L(u_j) = L\left(\sum_{\substack{\oplus\\ j=1}}^{J} u_j \right) \qquad \text{(6.13'ü takiben)} \tag{6.17}$$

$$= \ln \frac{P\left(\sum_{\substack{\oplus\\ j=1}}^{J} u_j = +1 \right)}{P\left(\sum_{\substack{\oplus\\ j=1}}^{J} u_j = -1 \right)} \qquad \text{(6.10'u takiben)} \tag{6.18}$$

$$= \ln \frac{\prod_{j=1}^{J} (e^{L(u_j)} + 1) + \prod_{j=1}^{J} (e^{L(u_j)} - 1)}{\prod_{j=1}^{J} (e^{L(u_j)} + 1) - \prod_{j=1}^{J} (e^{L(u_j)} - 1)} \tag{6.19}$$

$$\tanh(\frac{x}{2}) = \frac{e^x - 1}{e^x + 1} \tag{6.20}$$

Denklem (6.20)'deki ilişki kullanılarak, tümevarımı basitleştirecek olursak,

$$\sum L(u_j) = \ln \frac{1 + \prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2})}{1 - \prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2})} \tag{6.21}$$

$$= 2 \tanh^{-1}\left(\prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2}) \right) \tag{6.22}$$

Bu değerin hesaplanması çok uzun süreceği için yaklaşık olarak,

$$\sum_{\substack{[+]\\ j=1}}^{J} L(u_j) = L\left(\sum_{\substack{\oplus\\ j=1}}^{J} u_j\right) \tag{6.23}$$

$$\approx \left(\prod_{j=1}^{J} sign((L(u_j))\right) \min_{j=1,\ldots J}\{\left|L(u_j)\right|\} \qquad \text{(6.12'yi takiben)} \tag{6.24}$$

(6.24)'den görüleceği üzere yumuşak yani L değerleri toplamının güvenirliği yumuşak ya da L değer dizisinin en küçük elemanı ile tespit edilir.

6.4.4. Yumuşak kanal çıkışları

Şekil 6.8'deki sistem modelinde bilgi biti (u), kodlanmış bitler (v) ile eşleştirilmiştir. Kodlanmış bitler kanal üzerinden iletilerek kod çözücü girişinde alınan bitler (r) meydana gelmektedir. Bu sistem modelinden r ile koşullu olarak v'in logaritmik olabilirlik oranı şu şekilde hesaplanır,

$$L(v|r) = \ln\frac{P(v=+1|r)}{P(v=-1|r)} \tag{6.25}$$

Bayes teoremi kullanılarak bu logaritmik olabilirlik oranı aşağıdaki denkleme eşittir,

$$L(v|r) = \ln\left(\frac{p(r|v=+1)P(v=+1)}{p(r|v=-1)P(v=-1)}\right) \tag{6.26}$$

$$L(v|r) = \ln\frac{p(r|v=+1)}{p(r|v=-1)} + \ln\frac{P(v=+1)}{P(v=-1)} \tag{6.27}$$

Kanal modeli olarak AWGN ve gauss pdf f(z) 'si kullanılarak,

$$f(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-ort)^2}{2\sigma^2}} \tag{6.28}$$

Burada ort ortalamayı ve σ^2 varyansı göstermektedir.

$$\ln \frac{p(r|v=+1)}{p(r|v=-1)} = \ln \frac{e^{-\frac{Eb}{No}(r-a)^2}}{e^{-\frac{Eb}{No}(r+a)^2}} \tag{6.29}$$

$$= \ln \frac{e^{\frac{Eb}{No}2ar}}{e^{-\frac{Eb}{No}2ar}} \tag{6.30}$$

$$= 4\frac{Eb}{No}ar \tag{6.31}$$

Burada $\frac{Eb}{No}$, bit başına sinyalin gürültüye oranını ifade eder (doğrudan gürültü varyansı ile ilişkilidir) ve a sönümlenme genliğidir. Sönümlenmesiz gauss kanalı için a=1 dir. r koşullu v 'in logaritmik olabilirlik oranı $L(v|r)$,

$$L(v|r) = L_K r + L(v) \tag{6.32}$$

(6.27 ve 6.31 'e takiben) eşittir. Burada L_K, kanal güvenilirliği olarak tanımlanmıştır;

$$L_K = 4\frac{E_b}{N_o}a \tag{6.33}$$

Böylece, $L(v|r)$, ağırlıklı alınan değer $(L_K r)$ ile x'in logaritmik olabilirlik oranı $(L(v))$ 'nın toplamıdır.

6.4.5. Turbo kodlarda kullanılan SOVA kod çözücü bileşeni

SOVA kod çözücü bileşeni, Turbo kodlayıcısı tarafından üretilen iki kodlanmış veri akışından birini kullanarak bilgi dizisini tahmin eder. Şekil 6.9'de SOVA kod çözücüsünün giriş ve çıkışları gösterilmiştir.

Şekil 6.9: SOVA kod çözücü bileşeni

SOVA kod çözücü bileşeni $L(u)$ ile gösterilen bilgi dizisi (u)'nin önceki değerini ve $L_K r$ ile gösterilen ağırlıklı alınan diziyi (logaritmik olabilirlik oranı) işleme sokar. r, kanaldan alınan ve olası hata içeren bilgi dizisidir. Bununla birlikte $L(u)$ dizisi, önceki SOVA kod çözücü bileşeninden elde edilmektedir, fakat kod çözme işleminin başlangıcında, önceden herhangi bir işlem yapılmadığı için $L(u)$ dizisinin elemanları 0'dır. SOVA kod çözücü bileşeninin çıkışlarında u' ile gösterilen tahmini bilgi dizisi ve $L(u')$ ile gösterilen logaritmik olabilirlik oranı (yumuşak ya da L değeri) dizisi üretilir.

SOVA kod çözücü bileşeni, ML dizisi haricinde değiştirilmiş metrik kullanarak Viterbi kod çözücüsüne benzer şekilde çalışmaktadır. Bu değiştirilmiş metrik, diğer kod çözücüden gelen önceki değer (priori value) ile birleştirilir.

Temel Viterbi algoritması, $\mathrm{S}^{(m)}$ durum dizisi için arama yapar ya da $\mathrm{u}^{(m)}$ bilgi dizisi için sonraki olabilirliği (posteriori probability, $P(\mathrm{S}^{(m)}|\mathrm{r})$), en büyükleyen aramayı yapar. İkili kafesler (k=1) için m, kazanan ya da yarışan yollar için sırasıyla 1 ya da 2 olabilir. Bayes teoremi kullanılarak, sonraki olabilirlik şu şekilde ifade edilebilir;

$$P(\mathrm{S}^{(m)}|\mathrm{r}) = p(\mathrm{r}|\mathrm{S}^{(m)})\frac{P(\mathrm{S}^{(m)})}{p(\mathrm{r})} \tag{6.34}$$

Metrik hesaplaması için alınan dizi (r) sabittir ve m'ye bağlı değildir yani atılabilir. Böylece, en büyükleme,

$$\max_m p(\mathrm{r}\left|\mathrm{S}^{(m)}\right.)P(\mathrm{S}^{(m)}) \tag{6.35}$$

olmaktadır. t anında bitirilen durum dizisi olabilirliği $P(\mathrm{S}_t)$ dir. Bu olabilirlik şu şekilde hesaplanabilir;

$$P(\mathrm{S}_t) = P(\mathrm{S}_{t-1})P(S_t) \tag{6.36}$$

$$= P(\mathrm{S}_{t-1})P(u_t) \tag{6.37}$$

Burada $P(S_t)$, t anında durum olabilirliği, $P(u_t)$, bit olabilirliğidir. En büyükleme aşağıdaki şekilde genişletilebilir;

$$\max_m p(\mathrm{r}\left|\mathrm{S}^{(m)}\right.)P(\mathrm{S}^{(m)}) = \max_m \left\{ \prod_{i=0}^{t} p(\mathrm{r}_i \left| S_{i-1}^{(m)}, S_i^{(m)} \right.) P(\mathrm{S}_t^{(m)}) \right\} \tag{6.38}$$

Burada $(S_{i-1}^{(m)}, S_i^{(m)})$, i–1 ve i anları arasındaki durum geçişini, r_i durum geçişi için ilişkilendirilmiş alınan kanal değerlerini ifade eder. Düzenleme yapıldıktan sonra,

$$\max_m p(\mathrm{r}\left|\mathrm{S}^{(m)}\right.)P(\mathrm{S}^{(m)}) = \max_m \left\{ P(\mathrm{S}_{t-1}^{(m)} \prod_{i=0}^{t-1} p(\mathrm{r}_i \left| S_{i-1}^{(m)}, S_i^{(m)} \right.) P(u_t^{(m)}) p(\mathrm{r}_t \left| S_{t-1}^{(m)}, S_t^{(m)} \right.) \right\} \tag{6.39}$$

elde edilir.

$$p(\mathrm{r}_t \left| S_{t-1}^{(m)}, S_t^{(m)} \right.) = \prod_{j=1}^{N} p(r_{t,j} \left| v_{t,j}^{(m)} \right.) \qquad \text{göz önüne alınarak,} \tag{6.40}$$

$$\max_m \left\{ P(\mathrm{S}_{t-1}^{(m)} \prod_{i=0}^{t-1} p(\mathrm{r}_i \left| S_{i-1}^{(m)}, S_i^{(m)} \right.) P(u_t^{(m)} \prod_{j=1}^{N} p(r_{t,j} \left| v_{t,j}^{(m)} \right.) \right\} \qquad \text{olur.} \tag{6.41}$$

Eğer tüm denkleme m'den bağımsız olarak logaritma uygulanıp, 2 ile çarpılırsa ve iki sabit eklenirse en büyükleme işlemi değişmez.

$$\max_m \left\{ M_t^{(m)} \right\} = \max_m \left\{ M_{t-1}^{(m)} + [2 \ln P(u_t^{(m)}) - C_u] + \sum_{j=1}^{N} [2 \ln p(r_{t,j} \left| v_{t,j}^{(m)} - C_r \right.] \right\} \tag{6.42}$$

Burada,

$$\frac{M_{t-1}^{(m)}}{2} = \ln \left(P(\mathrm{S}_{t-1}^{(m)}) \prod_{i=0}^{t-1} p(\mathrm{r}_i \left| S_{i-1}^{(m)}, S_i^{(m)} \right.) \right) \tag{6.43}$$

$$C_u = \ln P(u_t = +1) + \ln P(u_t = -1) \tag{6.44}$$

$$C_r = \ln(p(r_{t,j} \left| v_{t,j} = +1 \right.)) + \ln(p(r_{t,j} \left| v_{t,j} = -1 \right.)) \tag{6.45}$$

Bu iki sabitin yerine konması ile SOVA metriği elde edilir;

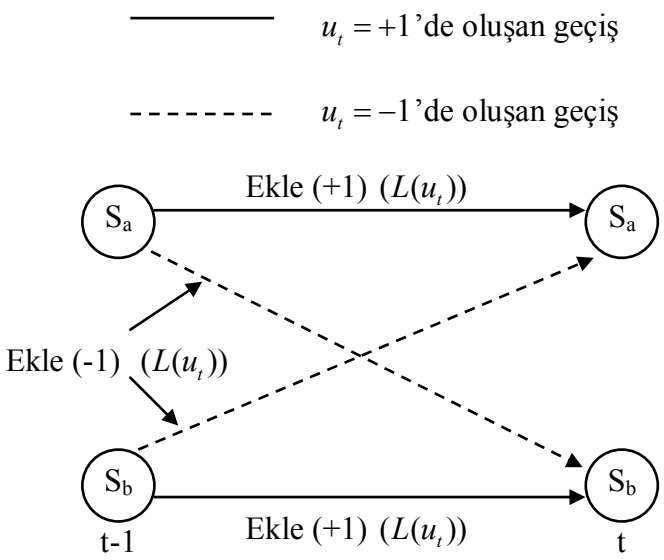
$$M_t^{(m)} = M_{t-1}^{(m)} + \sum_{j=1}^{N} v_{t,j} \ln \frac{p(r_{t,j} \left| v_{t,j} = +1 \right.)}{p(r_{t,j} \left| v_{t,j} = -1 \right.)} + u_t^{(m)} \ln \frac{P(u_t = +1)}{P(u_t = -1)} \tag{6.46}$$

$$M_t^{(m)} = M_{t-1}^{(m)} + \sum_{j=1}^{N} v_{t,j}^{(m)} L_K r_{t,j} + u_t^{(m)} L(u_t) \tag{6.47}$$

şeklinde düzenlenir. Sistematik kodlar için bu formül şu şekilde değiştirilebilir;

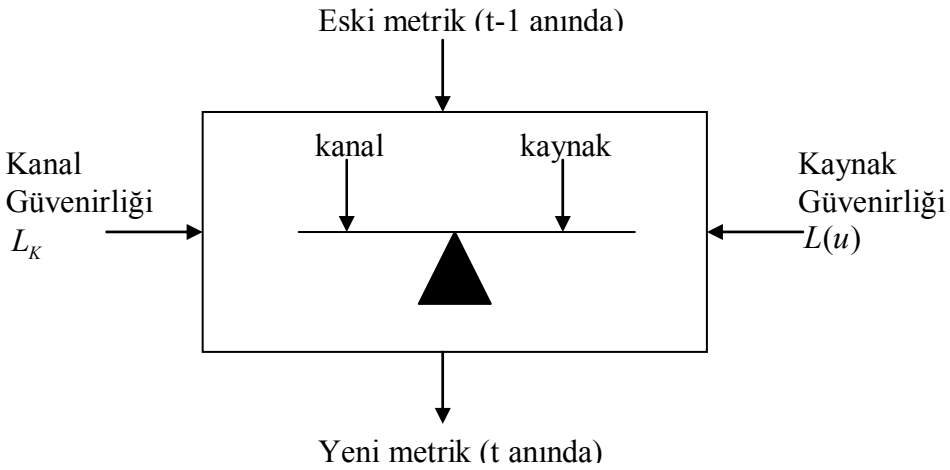
$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_K r_{t,1} + \sum_{j=2}^{N} v_{t,j}^{(m)} L_K r_{t,j} + u_t^{(m)} L(u_t) \tag{6.48}$$

(6.47) ve (6.48)'den görüldüğü üzere, SOVA metriği önceki metriği, kanal güvenirliğini ve kaynak güvenilirliğini (önceki değer) birleştirmektedir.



Şekil 6.10: SOVA metrik hesaplaması için kaynak güvenirliği

Şekil 6.10, S_a ve S_b durumlarına ait kafes diyagramlarını ve t ile t–1 anları arasındaki geçişi göstermektedir. Geçiş işlemi sırasında, düz çizgi u_t= +1 bilgi biti üretileceği zamanki geçişi, kesikli çizgi u_t= -1 bilgi biti üretileceği zamanki geçişi göstermektedir. Pozitif veya negatif olabilen kaynak güvenilirliği $L(u_t)$, önceki SOVA kod çözücü bileşeninden gelmektedir. 'Ekle' değeri, tahmini bilgi biti üzerinde, daha güvenilir karar elde etmek için SOVA metrik ile birleştirilir. Örneğin, $L(u_t)$ büyük bir pozitif sayı ise, kod çözme durumları arasında tahmini bit kararını +1'den -1'e değiştirmek oldukça zordur. Bununla birlikte, $L(u_t)$ küçük bir pozitif sayı ise, kod çözme durumları arasında tahmini bit kararını +1'den -1'e değiştirmek nispeten kolaydır. Sonuç olarak denilebilir ki, $L(u_t)$, önceki kod çözücüye zıt bit kararının gitmesini engellemek için tampon gibi davranır.



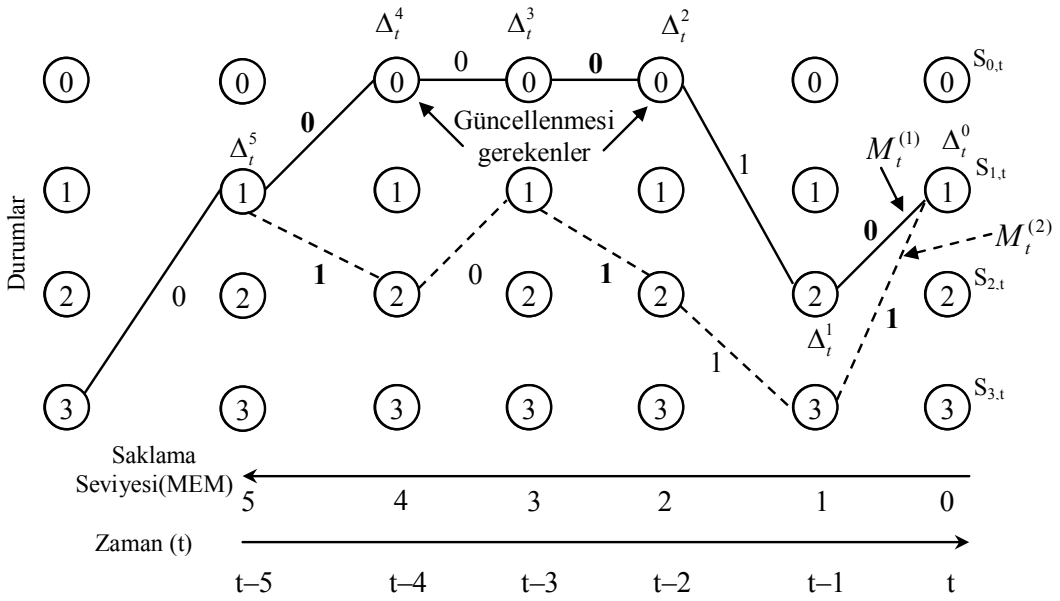
Şekil 6.11: SOVA metriğinin ağırlıklandırma özellikleri

Şekil 6.11'da gösterildiği üzere, SOVA metrik için kanal ve kaynak güvenilirliği arasındaki denge çok önemlidir. Bu durum, kanal ve kaynak güvenilirlik değerlerinin aynı büyüklükte olacağı demek değildir, fakat göreceli değerleri kanal ve kaynak koşullarını yansıtır. Örneğin, kanal iyi ise L_K, $|L(u)|$'dan daha büyüktür ve kod çözme işleminde alınan kanal değerleri etkilidir. Eğer kanal kötü ise, kod çözme işleminde önceki bilgi, $L(u)$ daha etkilidir. Denge sağlanmamış ise olumsuz etkiler meydana gelir ve kanal kod çözücünün başarımını düşürür.

t anında, güvenilirlik değeri (logaritmik olabilirlik oranının büyüklüğü) aşağıdaki denklemden sağlanan kafes diyagramındaki düğüme atanır.

$$\Delta_t^0 = \tfrac{1}{2}\left|M_t^{(1)} - M_t^{(2)}\right| \tag{6.49}$$

Δ_t^{MEM}, t anında saklama seviyesi MEM iken güvenilirlik değerini gösterir. Bu gösterim şekli L(t-MEM) ile benzerdir ve 6.12'de gösterilmiştir.



Şekil 6.12: Güvenilirlik tahmini için kazanan ve yarışan yolları gösteren SOVA örneği

t anında, m yoluna ait olabilirlik ve SOVA metriği, [32]'de gösterilmiştir;

$$P(yol(m)) = P(\mathrm{S}_t^{(m)}) \tag{6.50}$$

$$= e^{\frac{M_t^{(m)}}{2}} \tag{6.51}$$

$M_t^{(1)}$ t anında, bir düğüme ait kazanan metrik ve yarışan metrik $M_t^{(2)}$ ile gösterilmiştir. Böylece, doğru kazanan yolun seçim olabilirliği,

$$P(doğru) = \frac{P(yol(1))}{P(yol(1)) + P(yol(2))} \tag{6.52}$$

$$= \frac{e^{\frac{M_t^{(1)}}{2}}}{e^{\frac{M_t^{(1)}}{2}} + e^{\frac{M_t^{(2)}}{2}}} \tag{6.53}$$

$$= \frac{e^{\Delta_t^0}}{1 + e^{\Delta_t^0}} \tag{6.54}$$

Bu yol kararının olabilirliği şu şekilde hesaplanabilir;

$$\log \frac{P(doğru)}{1-P(doğru)} = \log \frac{\dfrac{e^{\Delta_t^0}}{1+e^{\Delta_t^0}}}{1-\dfrac{e^{\Delta_t^0}}{1+e^{\Delta_t^0}}} \tag{6.56}$$

$$= \Delta_t^0 \tag{6.56}$$

t anında, belli bir düğüme ait kazanan yol üzerindeki güvenilirlik değerleri, Δ_t^{MEM} ile gösterilir (MEM=0,…,t). t anında bu düğüm için MEM=k (t-MEM anı içinde aynı) iken kazanan yol üzerindeki bit, yarışan yol üzerindeki bit ile aynı ise ve eğer yarışan yol seçilmiş ise bit hatası yoktur denir. Böylece, bu bit durumunda güvenilirlik değeri değişmemiştir. MEM=k anında kazanan ve yarışan yoldaki bit farklı ise bir bit hatası vardır denir. Bit hata durumunda güvenilirlik değeri güncellenmelidir. Şekil 6.12'de bu durum söz konusu olduğu için MEM=2 ve MEM=4 için güvenilirlik güncellenmelidir. Güvenilirlik güncellemeleri, yumuşak ya da L değerlerinin düzeltilmesini sağlar. [33]'da gösterildiği üzere, bit kararının yumuşak yani L değeri;

$$L(u'_{t-MEM}) = u'_{t-MEM} \sum_{\substack{[+] \\ k=0}}^{MEM} \Delta_t^k \tag{6.57}$$

Denklem (6.24) ile yaklaşık olarak;

$$L(u'_{t-MEM}) \approx u'_{t-MEM} \min_{k=0,\ldots,MEM} \left\{ \Delta_t^k \right\} \tag{6.58}$$

Yumuşak çıkışlı Viterbi algoritması (güvenilirlik güncelleme işlemi ile birlikte), şu şekilde işletilir;

1.a) t=0 'dan başlatılır.

b) Kafes diyagramındaki sıfır durumu için $M_0^{(m)} = 0$ ve diğer tüm durumları -∞'dan başlatılır.

2. a) t=t+1 'e atanır.

b) $M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_K r_{t,1} + \sum_{j=2}^{N} v_{t,j}^{(m)} L_K r_{t,j} + u_t^{(m)} L(u_t)$ denklemi ile kafes diyagramındaki her durum için metrik hesaplanır.

Burada m, bir duruma ikili dal (branch) geçişini göstermektedir (m=1, 2).

$M_t^{(m)}$: m dalındaki t anındaki biriktirilmiş metrik.

$u_t^{(m)}$: m dalındaki t anındaki sistematik bit (N bitin birinci biti).

$v_{t,j}^{(m)}$: m dalındaki t anındaki N bitin j. biti ($2 \leq j \leq N$).

$r_{t,j}^{(m)}$: $v_{t,j}^{(m)}$'e karşılık gelen alınan değer.

$L_K = 4\frac{Eb}{No}$: kanal güvenilirlik değeri.

$L(u_t)$: t anındaki önceki güvenilirlik değeri. Bu değer önceki kod çözücüden gelen değerdir.

3.Her durum için $\max_m M_t^{(m)}$ bulunur. Basit olması için $M_t^{(1)}$ kazanan yol metriği olarak $M_t^{(2)}$ yarışan yol metriği olarak gösterilmektedir.

4. $M_t^{(1)}$ ve $M_t^{(1)}$ ile ilişkilendirilmiş kazanan bit ve durum yolları saklanır.

5. $\Delta_t^0 = \frac{1}{2}\left|M_t^{(1)} - M_t^{(2)}\right|$ hesaplanır.

6. t anında her durum için kazanan ve yarışan yollar kıyaslanır ve iki farklı yoldaki tahmini ikili kararların olduğu MEM ler saklanır.

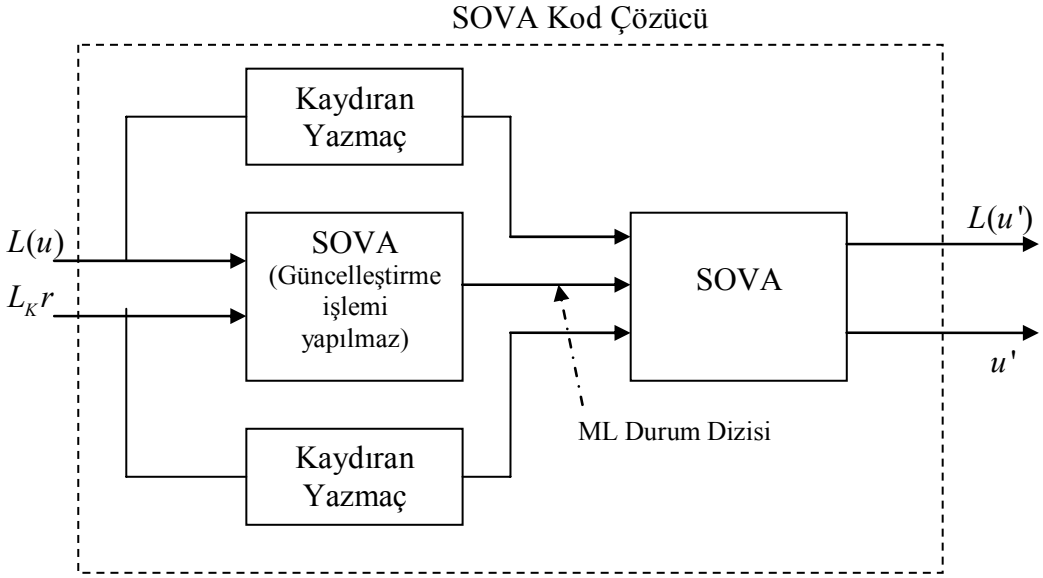
7. $\Delta_t^{MEM} \approx \min_{k=0,\ldots,MEM} \left\{\Delta_t^k\right\}$ değeri, en küçük MEM'den en büyük MEM'e kadar tüm MEM'ler güncellenir.

8. Alınan dizi bitinceye kadar 2. adıma geri gidilir.

9. Tahmini bit dizisi u' ve ilişkilendirilmiş yumuşak yani L değer dizisi $L(u') = u' \bullet \Delta$ çıkışta elde edilir. Burada, $\bullet$, bire bir çapımı (eleman elemana) ve Δ, son güncellenmiş güvenilirlik dizisini gösterir. $L(u')$ işleme sokularak $L(u)$ yani önceki dizi olarak kod çözücüye gider.

6.5. SOVA 'nın Gerçekleştirilmesi

SOVA kod çözücüsü çeşitli yollardan gerçekleştirilebilir. Uzun çerçeve boyutlarında ve büyük kısıtlama boyutunda SOVA kod çözücünün doğrudan gerçekleştirilmesi yoğun hesaplama gerektirmektedir. Çünkü tüm kazanan yolların güncellenmesi gerekir. Şekil 6.13 'de sadece ML yolu için güncelleme işlemi yapan kod çözücü şeması gösterilmektedir.



Şekil 6.13: SOVA kod çözücü şeması

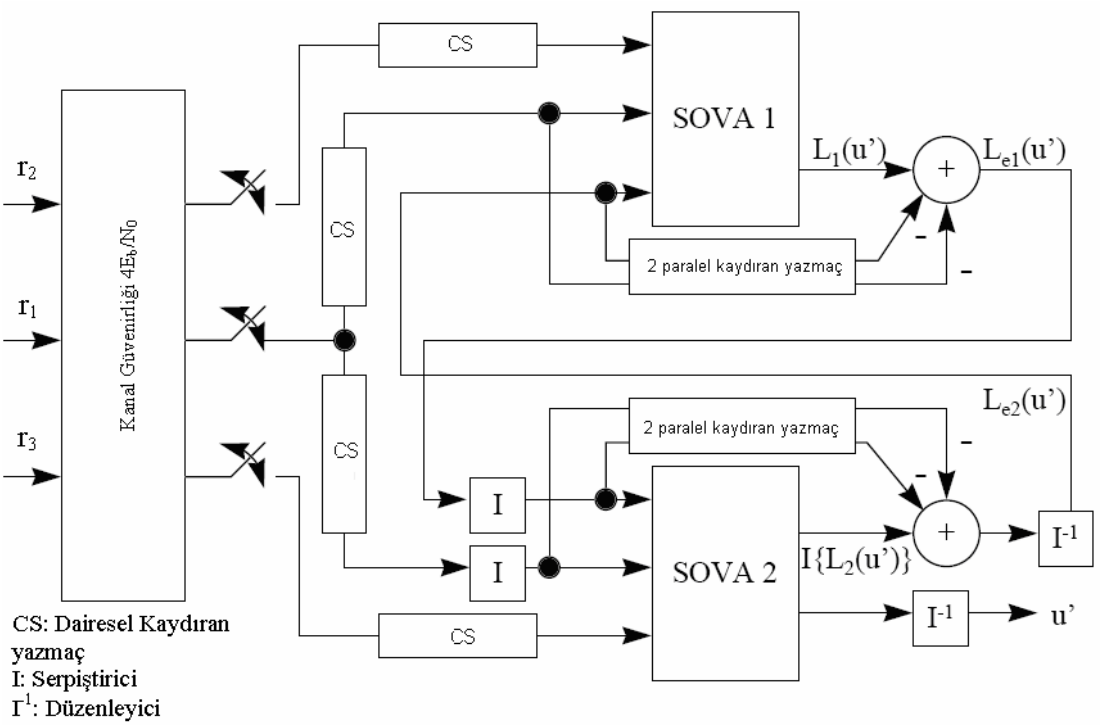
SOVA kod çözücünün girişleri, önceki değerler $L(u)$ ve ağırlıklı alınan değerler, $L_K r$ olarak gösterilmiştir. Çıkışları tahmini bit kararları u' ve ilişkilendirilmiş yumuşak yani L değerlerini içeren $L(u')$ dir.

SOVA kod çözücü yapısı iki ayrı SOVA kod çözücüsünden oluşmaktadır. İlk SOVA kod çözücüsü sadece ML yolu için metrikleri hesaplar, güvenilirlik değerlerini hesaplamaz. Kaydıran yazmaçlar, birinci SOVA kod çözücüsü ML yolunu işlerken girişleri tamponlamak için kullanılır. İkinci SOVA kod çözücüsü (ML yol bilgisi ile birlikte), ML yolunu tekrar hesaplar ve güvenilirlik değerlerini hesaplayarak günceller.

SOVA algoritmasının bu şekilde işletilmesi ile 2^m adet kazanan yolun yerine sadece ML yolunun saklanması güncelleme işleminin karmaşıklığını azaltmaktadır.

6.6. Döngülü SOVA Turbo Kod Çözücüsü

Döngülü Turbo kod çözücüsü iki SOVA kod çözücüsünden meydana gelmektedir. Şekil 6.14'de döngülü Turbo kod çözücü yapısı gösterilmektedir.



Şekil 6.14: SOVA döngülü Turbo kod çözücüsü

Turbo kod çözücüsü, çerçeve tabanlı alınan kanal bitlerini işletir. Şekil 6.13'de görüldüğü üzere, alınan kanal bitleri r_1 sistematik akışı r_2 ve r_3 isimli sırasıyla 1. ve 2. kodlayıcılardan gelen iki kontrol bit akışı olarak çoğullanır (demultiplexing). Bu bitler kanal güvenilirliği tarafından ağırlıklandırılır ve CS kaydıran yazmaçlarına yüklenir. CS'ler, bit dizilerini gerekinceye kadar tutarlar. Anahtarlar, o anki hazır bulunan çerçeve işletilene kadar bir sonraki çerçeveden gelen bitleri engellemek için açık durumdadır.

SOVA kod çözücüsü t anında, tahmini bit u' için yumuşak yani L değerlerini içeren $L(u'_t)$ üretir. $L(u'_t)$, üç farklı terime ayrıştırılabilir.

$$L(u_t^{'}) = L(u_t) + L_K r_{t,1} + L_e(u_t^{'}) \quad [32] \tag{6.59}$$

$L(u_t)$, önceki SOVA kod çözücü tarafından üretilen değerdir. $L_K r_{t,1}$, ağırlıklı alınan sistematik kanal değeridir. $L_e(u'_t)$, kod çözücü tarafından üretilen fazlalık değerdir. Kod çözücüler arasında iletilen fazlalık değer,

$$L_e(u_t^{'}) = L(u_t^{'}) - L(u_t) - L_K r_{t,1} \tag{6.60}$$

şeklindedir. Önceki değerin ($L(u_t)$), kod çözücüye tekrar girmesini engellemek için yumuşak yani L değer $L(u_t^{'})$'den çıkartılır. Bununla birlikte, ağırlıklı alınan sistematik kanal değeri $L_K r_{t,1}$, SOVA kod çözücülerin içinde ortak bilgiden çıkartılır. Şekil 6.13'de SOVA kod çözücülerin kapalı devre seri bağlanması ile oluşturulmuş Turbo kod çözücü görülmektedir. Bu kapalı devre kod çözme şemasında, her bir kod çözücü farklı ağırlıklı eşlik kontrol akışını kullanarak bilgi dizisini tahmin eder. Turbo kod çözücüsü, daha iyi kod çözme başarımı sağlamak için ve iki farklı ağırlıklı eşlik kontrol akışından daha doğru sonuçlar elde edebilmek için döngülü kod çözmeyi kullanmaktadır.

n. döngü için döngülü Turbo kod çözme algoritması aşağıdaki şekildedir;

1. SOVA1 kod çözücüsüne giren $4\frac{Eb}{No} r_1$ (sistematik) , $4\frac{Eb}{No} r_2$ (eşlik kontrol) dizileri ve $L_{e2}(u')$ dir. Çıkış dizisi ise $L_1(u')$ dır. Birinci döngü için, $L_{e2}(u') = 0$ dır, çünkü bir önceki değer başlangıçta yoktur (SOVA2 'den gelen fazlalık değer).
2. SOVA1'den gelen fazlalık bilgi,

$L_{e1}(u') = L_1(u') - L_{e2}(u') - L_K r_1$ den elde edilir ve $L_K = 4\frac{Eb}{No}$ dir.

3. $4\frac{Eb}{No}r_1$ ve $L_{e1}(u')$ dizileri serpiştirilir ve $I\left\{4\frac{Eb}{No}r_1\right\}$ ve $I\{L_{e1}(u')\}$ şeklinde gösterilirler.

4. SOVA2 kod çözücüsünün girişleri, $I\left\{4\frac{Eb}{No}r_1\right\}$(sistematik) , $I\left\{4\frac{Eb}{No}r_3\right\}$(Turbo kod kodlayıcısı tarafından serpiştirilmiş eşlik kontrol) ve $I\{L_{e1}(u')\}$ (önceki bilgi) dir. Çıkış dizileri, $I\{L_{e2}(u')\}$ ve $I\{u'\}$ dır.

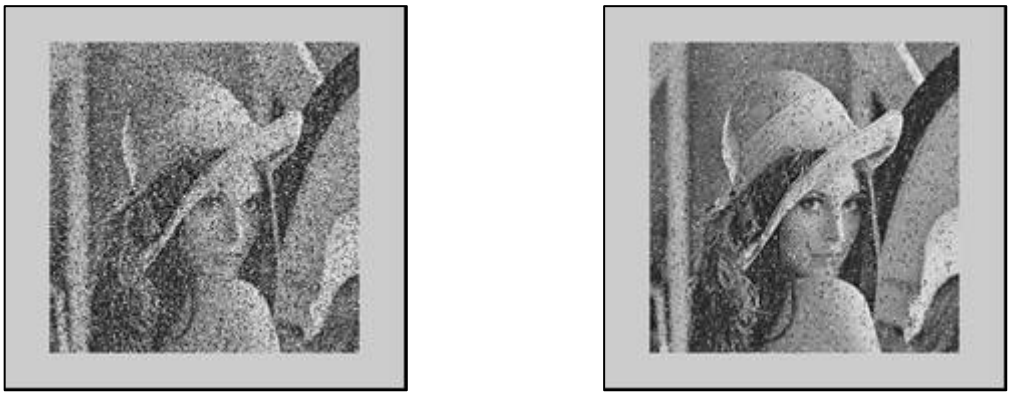
5. SOVA2'den gelen fazlalık bilgi,

$\{L_{e2}(u')\} = I\{L_2(u')\} - I\{L_{e1}(u')\} - I\{L_K r_1\}$ 'den hesaplanır.

6. $I\{L_{e2}(u')\}$ ve $I\{u'\}$ dizileri tekrar düzenlenerek (deinterleaved) ve $L_{e2}(u')$ ve u' elde edilir. Bir sonraki döngü için $L_{e2}(u')$ önceki bilgi olarak SOVA1 'e geri besleme yapılır ve u', n. döngü için tahmini çıkış bitleridir.

6.7. SOVA Kod Çözme Algoritması Uygulaması

SOVA kod çözme algoritması ile bmp uzantılı bir resim, Matlab Simulink programı kullanılarak vericiden alıcıya gönderilmiştir. Bu uygulama farklı SNR değerleri ile 5 döngü sayısı için gerçekleştirilmiştir.



Şekil 6.15: SNR=0 ve 1 dB değerlerindeki SOVA çıkışları

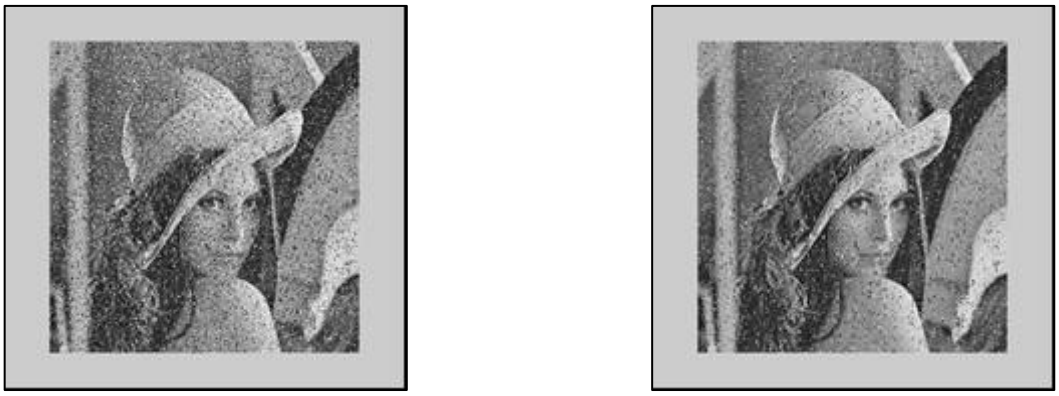
Beş döngülü kod çözme işlemi sonucunda sol tarafta SNR 0 değeri ve sağ tarafta SNR 1 değeri için elde edilmiştir.

Şekil 6.16: SNR=2 ve 3 dB değerlerindeki SOVA çıkışları

Beş döngülü kod çözme işlemi sonucunda sol tarafta SNR 2 değeri ve sağ tarafta SNR 3 değeri için elde edilmiştir.

Şekil 6.17: SNR=4 dB değerindeki SOVA çıkışı

Beş döngülü kod çözme işlemi sonucunda SNR 4 değeri için elde edilmiştir. Şekil 6.15, 6.16, 6.17'de görüldüğü üzere, SNR değeri arttıkça BER düşmekte ve kaliteli iletim yapılmış olmaktadır.



Şekil 6.18: SNR 1 değeri için solda 1 döngü sağda 5 döngü sayısı için sonuçlar

Şekil 6.17'de görüldüğü gibi, döngü sayısı arttıkça hata oranı azalmış ve daha iyi bir görüntü elde edilmiştir.

6.8. Sonuç

Bu bölümde Turbo kod çözme algoritmalarından SOVA kod çözme algoritması hakkında ayrıntılı bilgi verilmiştir. SOVA algoritması Viterbi algoritmasına göre çok daha iyi BER başarımı göstermektedir. Ayrıca çeşitli SNR değerlerinde uygulama sonuçları sunulmuştur.

7. EN BÜYÜK SONSAL KOD ÇÖZME ALGORİTMASI VE BAŞARIM ANALİZİ

7.1. Giriş

Turbo kod çözme işlemi, her veri biti için sonsal olasılıkların (APP) oluşturulması ile başlar ve bunu o veri bitine karşılık gelen en büyük sonsal olasılığın (MAP) seçimi takip eder. Bozulmuş kod-bit dizisinin alınması üzerine, APP değerleri ile karar verme işlemi başlar ve en büyük sonsal (MAP, Maximum A Posteriori) kod çözme algoritması, iletilmiş olan her bit için en doğru bilgiyi saptar. Bununla birlikte MAP kod çözme algoritmasında bazı metriklerin hesaplanması gerekmektedir. MAP algoritması, yapısından kaynaklanan hesaplama karmaşıklığına (doğrusal olmayan fonksiyonlar, çok sayıdaki toplama ve çarpım işlemleri) sahiptir. MAP algoritmasından logaritmik alanda türetilen algoritmalar (Max-Log-MAP ve Log-MAP algoritmaları), bu algoritmasının sayısal problemlerini ve hesaplama karmaşıklığını çözmektedir. Fakat özellikle düşük SNR değerlerinde (Max-Log-MAP, max fonksiyonu kullanmasından dolayı bu duruma bir örnektir), MAP algoritması kadar çok iyi sonuçlar vermemektedir [4]. MAP algoritmasının bir diğer basitleştirilmiş şekli de 6. bölümde anlatılan SOVA algoritmasıdır. Max-Log-MAP algoritmasının düşük SNR değerlerindeki dezavantajını gidermek, MAP algoritmasına eşit bir şekilde sonuçlar üretmek ve MAP algoritmasının pratik uygulamalarda da kullanılmasını sağlamak amacıyla Log-MAP algoritması geliştirilmiştir.

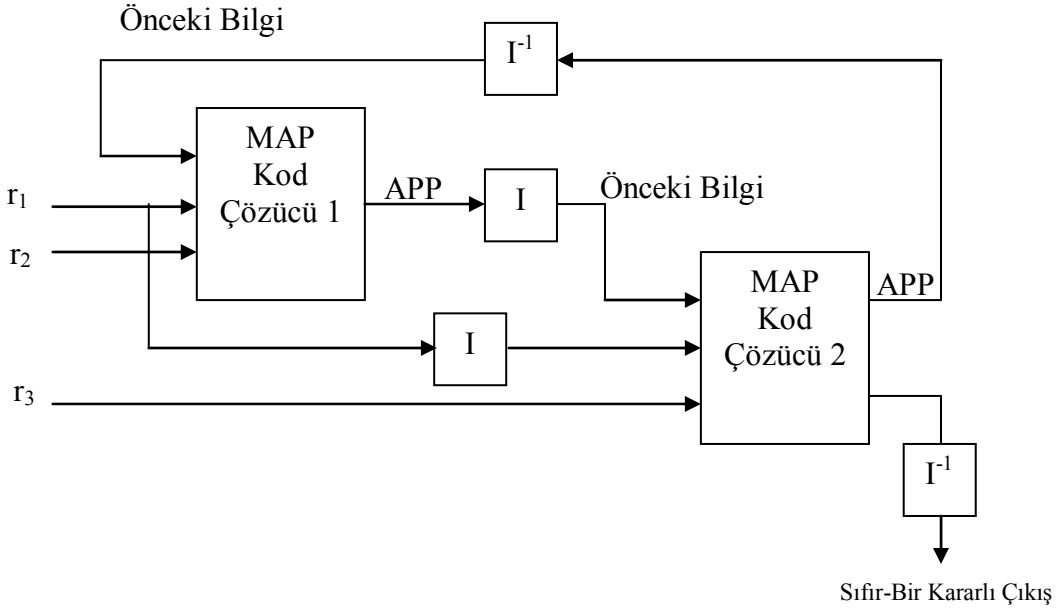
Bu bölümde öncelikle MAP algoritması ve ardından Log-MAP algoritması anlatılıp MAP algoritmasının pratik uygulamalarda kullanılan türevi Log-MAP algoritmasının benzetim sonuçları verilecektir.

7.2. SOVA ve MAP Algoritmaları

SOVA algoritmasında MAP algoritmasından farklı olarak, APP değerleri her veri biti için elde edilemez. Bu yüzden SOVA algoritması, iletilmiş olan veri dizisi için en doğru diziyi (ML) bulmaktadır. Bununla birlikte her iki algoritmanın işletilmesinde benzerlikler vardır. Kodu çözülmüş bit hata olasılığı P_B, küçük olduğunda, MAP ve SOVA algoritmalarının arasında çok az bir başarım farkı vardır. MAP algoritması, küçük Eb/No ve yüksek P_B değerlerinde, SOVA algoritmasına göre en az 0.5 dB başarım göstermektedir [21, 34]. Turbo kodlar için, bu durum ilk kod çözme döngüsünden itibaren zayıf hata başarımı sağladığı için çok önemlidir.

7.3. MAP Algoritması

MAP algoritmasının gerçekleştirilmesi, bir kod bloğu üzerinde iki yönlü şekilde yumuşak çıkışlı Viterbi algoritmasının yürütülmesi olarak ifade edilebilir. Bu kod bloğu için iki yönlü hesaplama, durum ve dal metriklerini meydana getirir ve bloktaki her veri biti için APP değerleri ve MAP elde edilir. Şekil 7.1'de MAP algoritmasının devre şeması gösterilmektedir.



Şekil 7.1: MAP algoritmasının devre şeması

Şekil 7.1'de r_1 ile gösterilen, kod çözücü 1 ve kod çözücü 2'nin girişi olan ve kod çözücü bloğunda elde edilen iletilmiş bilgi bloğudur (RSC kodlayıcının sistematik çıkışına karşılık gelir). r_2 ve r_3 sırasıyla, şekil 6.1'de gösterilen birleştirilmiş kodlayıcı yapısında, kodlayıcı 1 ve kodlayıcı 2'de üretilmiş eşlik kontrol bitlerine karşılık gelen kod çözücüde alınan eşlik bitleridir. Kodlayıcı 1 tarafından üretilen eşlik kontrol bitleri kod çözücü 1 yardımıyla algoritmaya sokulmaktadır. Aynı şekilde Kodlayıcı 2 tarafından üretilen eşlik kontrol bitleri kod çözücü 2 yardımıyla algoritmaya sokulmaktadır. Bu işlemler sırasında şekil 6.1'deki kodlayıcının sistematik çıkışı olan r_1 her iki kod çözücüde de kullanılmaktadır.

Tezin bu kısmında sistematik katlamalı kodlar için AWGN kanal modeli ile [34]'de sunulmuş olan MAP algoritması anlatılacaktır.

Şekil 7.1'de gösterilen olabilirlik oranı $\Lambda(\hat{d}_k)$ ya da LLR olarak da adlandırılan APP'lerin oranı;

$$\Lambda(\hat{d}_t)=\frac{\sum_m \lambda_t^{1,m}}{\sum_m \lambda_t^{0,m}} \tag{7.1}$$

$$L(\hat{d}_t)=\log\left[\frac{\sum_m \lambda_t^{1,m}}{\sum_m \lambda_t^{0,m}}\right] \tag{7.2}$$

şeklinde ifade edilir. Bu denklemlerde olasılık, veri biti $d_k=i$ ve kafes durumu $S_k=m$ iken, t=1 anından N anına kadar alınan ikili dizi r_1^N'e koşullu, $\lambda_t^{i,m}$ olarak gösterilen birleşik olasılık olarak ifade edilir ve şu şekilde tanımlanmaktadır;

$$\lambda_t^{i,m}=P(d_t=i,\ S_t=m\left|r_1^N\right.) \tag{7.3}$$

r_1^N, kanal üzerinden iletilip modüle edilmiş ve kod çözücüde yumuşak kararlı formda hazırlanmış bozuk kod dizisini ifade etmektedir. Aslında MAP algoritması, bir

andaki N bitten oluşan blok kod çözücü için hazırlanmış modülatör çıkışlarına ihtiyaç duymaktadır. r_1^N aşağıdaki şekilde yazılabilir;

$$r_1^N = \left\{ r_1^{t-1},\, r_t,\, r_{t+1}^N \right\} \tag{7.4}$$

Bayes teoreminin kullanımını kolaylaştırmak için, denklem (7.3) açık bir şekilde yazılmıştır. Böylece denklem (7.3) şu şekilde gösterilebilir;

$$\lambda_t^{i,m} = P(d_t = i,\, S_t = m \left| r_1^{t-1},\, r_t,\, r_{t+1}^N \right.) \tag{7.5}$$

$d_t = i,\, S_t = m \Rightarrow A$

$r_1^{t-1} \Rightarrow B$

$r_t \Rightarrow C$

$r_{t+1}^N \Rightarrow D$

alınarak, Bayes teoremi tekrar yazılacak olursa,

$$P\left(A \left| B,\, C,\, D\right.\right) = \frac{P(A,\, B,\, C,\, D)}{P(B,\, C,\, D)} = \frac{P\left(B \left| A,\, C,\, D\right.\right) P(A,\, C,\, D)}{P(B,\, C,\, D)} \tag{7.6}$$

$$= \frac{P\left(B \left| A,\, C,\, D\right.\right) P\left(D \left| A,\, C\right.\right) P(A,\, C)}{P(B,\, C,\, D)}$$

Bayes teoremi denklem (7.5)'e uygulanacak olursa,

$$\lambda_t^{i,m} = P\left(r_1^{t-1} \left| d_t = i,\, S_t = m,\, r_t^N \right.\right) P(r_{t+1}^N \left| d_t = i,\, S_t = m,\, R_t \right.) P(d_t = i,\, S_t = m,\, r_t) / P(r_1^N) \tag{7.7}$$

elde edilir. Bu denklemde $r_t^N = \left\{ r_t,\, r_{t+1}^N \right\}$ 'dir.

Denklem (7.7), $\lambda_t^{i,m}$'yı meydana getiren bileşenler de ifade edilirse daha açık şekilde anlaşılabilecektir. Denklem (7.7)'nin sağ tarafındaki paydada bulunan üç faktör sırasıyla, ileri durum metriği, geri durum metriği ve dal metriğidir.

7.3.1. Durum metrikleri ve dal metriği

Denklem (7.7)'nin sağ tarafındaki ilk pay, t anında ve m durumundaki ileri durum metriği olarak adlandırılır ve α_t^m olarak gösterilir. Böylece, i=1, 0 için;

$$P\left(r_1^{t-1} \middle| d_t=i,\, S_t=m,\, r_t^N\right)=P(r_1^{t-1} \middle| S_t=m) \triangleq \alpha_t^m \tag{7.8}$$

şeklinde ifade edilir. Burada i, 1 veya 0 bit bilgisidir. Denklem (7.8)'de $d_t=i$ ve r_t^N değerleri şu an için önemsizdir, çünkü $S_t=m$, t anından önceki işlemleri içerir, t anından sonraki işlemlerden etkilenmez. Diğer bir değişle, gelecek geçmişten etkilenmez, bu nedenle $P(r_1^{t-1})$, d=i ve r_t^N dizisinden bağımsızdır. Ancak, kodlayıcı bir hafızaya sahip ve $S_t=m$ geçmişe bağımlı olduğu için, bu nedenle $P(r_1^{t-1})$ denklem (7.8) ile alakalıdır. Denklem (7.8) t anında, önceki dizinin olasılığı olan ileri durum metriği α_t^m'yı ifade eder ve o anki duruma bağlıdır. Benzer olarak, denklem (7.7)'nin sağ tarafındaki ikinci pay, geri durum metriği, β_t^m, t anında ve m durumunda şu şekilde ifade edilir;

$$P(r_{t+1}^N \middle| d_t=i,\, S_t=m,\, r_t)=P(r_{t+1}^N \middle| S_{t+1}=f(i,m)) \triangleq \beta_{t+1}^{f(i,m)} \tag{7.9}$$

Burada $f(i, m)$, i giriş biti ve m durumu için bir sonraki durum ve $\beta_{t+1}^{f(i,m)}$, t+1 anında $f(i, m)$ durumundaki geri durum metriğidir. Denklem (7.9)'daki $\beta_{t+1}^{f(i,m)}$, t+1 anında ve o anki duruma bağlı olarak bir sonraki dizi için olasılık olur ve t anı için giriş bitini ve durumu içeren bir fonksiyon olarak döner. Bu sonlu durum makinesinin temel tanımıdır [35].

Denklem (7.7)'nin sağındaki üçüncü pay, t anında ve m durumundaki dal metriği olarak adlandırılır ve $\delta_t^{i,m}$ olarak gösterilmektedir. Dal metriği şu şekilde formülize edilir;

$$P(d_t=i, S_t=m, r_t) \triangleq \delta_t^{i,m} \tag{7.10}$$

Denklem (7.8) ve denklem (7.10) kullanılarak denklem (7.7) yeniden yazılacak olursa bileşik olasılık için daha küçük bir terim elde edilir;

$$\lambda_t^{i,m} = \frac{\alpha_t^m \delta_t^{i,m} \beta_{t+1}^{f(i,m)}}{P(r_1^N)} \tag{7.11}$$

Denklem (7.11) denklem (7.1) ve denklem (7.2) kullanılarak;

$$\Lambda(\hat{d}_t) = \frac{\sum_m \alpha_t^m \delta_t^{1,m} \beta_{t+1}^{f(1,m)}}{\sum_m \alpha_t^m \delta_t^{0,m} \beta_{t+1}^{f(0,m)}} \tag{7.12}$$

$$L(\hat{d}_t) = \log\left[\frac{\sum_m \alpha_t^m \delta_t^{1,m} \beta_{t+1}^{f(1,m)}}{\sum_m \alpha_t^m \delta_t^{0,m} \beta_{t+1}^{f(0,m)}}\right] \tag{7.13}$$

Burada, $\Lambda(\hat{d}_t)$ t. veri biti için olasılık oranı ve $L(\hat{d}_t)$, $\Lambda(\hat{d}_t)$'nin logaritması (genelde logaritma e tabanında hesaplanır) ve t anındaki veri bitinin LLR'sidir.

7.3.2. İleri durum metriğinin hesaplanması

Denklem (7.8)'den başlayarak, t–1 anından itibaren olası tüm durum geçiş olasılıklarının toplamı şu şekilde ifade edilir;

$$\alpha_t^m = \sum_{m'} \sum_{j=0}^{1} P(d_{t-1}=j,\ S_{t-1}=m',\ r_1^{t-1} \mid S_t=m) \tag{7.14}$$

Bu denklemde r_1^{t-1}, $\left\{r_1^{t-2}, r_{t-1}\right\}$ şeklinde yazılırsa ve Bayes teoreminden,

$$\alpha_t^m = \sum_{m'} \sum_{j=0}^{1} P\left(r_1^{t-2} \mid S_t=m,\ d_{t-1}=j,\ S_{t-1}=m',\ r_{t-1}\right) \times P\left(d_{t-1}=j,\ S_{t-1}=m',\ r_{t-1} \mid S_t=m\right) \tag{7.15a}$$

$$= \sum_{j=0}^{1} P\left(r_1^{t-2} \mid S_{t-1}=b(j,\ m)\right) P(d_{t-1}=j,\ S_{t-1}=b(j,m),\ r_{t-1}) \tag{7.15b}$$

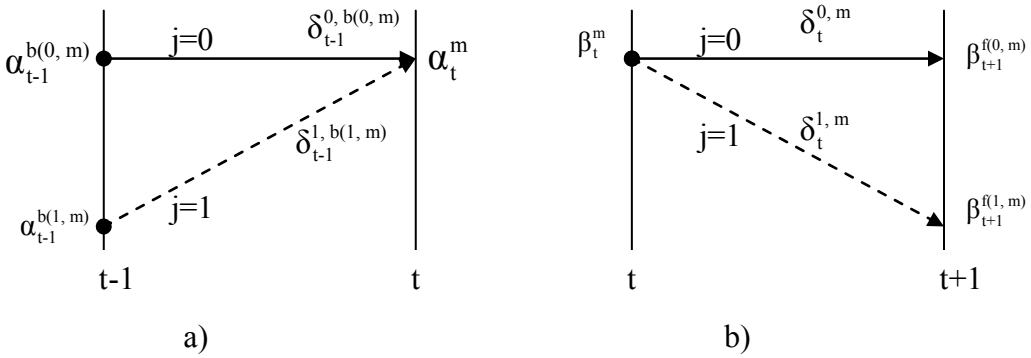
denklemleri elde edilmektedir. Denklem (7.15b)'de $b(j,\ m)$, j girişine karşılık gelen önceki dal üzerinden geçen, m durumundan zamanda geriye dönüş durumudur. Denklem (7.15b), t–1 anında j girişi ve m' durumu hakkında bilgi içerdiği için denklem (7.15a) ile yer değiştirebilir ve tamamen $S_t=m$ durumunda yol (path) oluşumunu tanımlamaktadır. Denklem (7.8) ve (7.10) kullanılarak denklem (7.15) basitleştirilecek olunursa;

$$\alpha_t^m = \sum_{j=0}^{1} \alpha_{t-1}^{b(j,\ m)} \delta_{t-1}^{j,\ b(j,\ m)} \tag{7.16}$$

elde edilmektedir. Denklem (7.16) t anında ve m durumunda yeni ileri durum metriğini göstermektedir ve t–1 anındaki iki ağırlıklı durum metriğini toplayarak elde edilir.

Ağırlıklandırma, 0 ve 1 veri bitlerine karşılık gelen durum geçişleri ile ilişkilendirilmiş dal metriklerinden oluşmaktadır. Şekil 7.2a'da, alpha parametresi

için iki farklı gösterim şekli görülmektedir. İki olası öncelikli durum (j=0 ya da 1 olmasına bağlı olarak) olduğu zaman, $\alpha_{t-1}^{b(j,\,m)}$, ileri durum metriği olarak kullanılır. t anında aynı durum üzerinde önceki zamanın bitiminde oluşan iki olası geçiş olduğu zaman, α_t^m, t anında ileri durum metriği kullanılmaktadır.



Şekil 7.2: α_t^m ve β_t^m 'nin grafiksel gösterimi [34]

Şekil 7.2'de dal metriği, ileri durum metriği ve geri durum metriği aşağıdaki şekilde hesaplanmaktadır.

$$\delta_t^{i,\,m}=\mu_t^i \exp(x_t u_t^i+y_t v_t^{i,\,m}) \tag{7.17}$$

$$\alpha_t^m=\alpha_{t-1}^{b(0,\,m)}\delta_{t-1}^{0,b(0,\,m)}+\alpha_{t-1}^{b(1,\,m)}\delta_{t-1}^{1,b(1,\,m)} \tag{7.18}$$

$$\beta_t^m=\beta_{t+1}^{f(0,\,m)}\delta_t^{0,\,m}+\beta_{t+1}^{f(1,\,m)}\delta_t^{1,\,m} \tag{7.19}$$

7.3.3. Geri durum metriğinin hesaplanması

Denklem (7.9) olan $\beta_{t+1}^{f(i,\,m)}=P\left[r_{t+1}^N \middle| S_{t+1}=f(i,\,m)\right]$'den başlanarak, β_t^m şu şekilde gösterilebilir;

$$\beta_t^m=P(r_t^N \mid S_t=m)=P(r_t,\, r_{t+1}^N \mid S_t=m) \tag{7.20}$$

β_t^m , t+1 anına olan olası tüm geçiş olasılıklarının toplamı olarak ifade edilebilir;

$$\beta_k^m = \sum_{m'} \sum_{j=0}^{1} P(d_t = j,\ S_{t+1} = m',\ r_t,\ r_{t+1}^N \mid S_t = m) \tag{7.21}$$

Bayes teoremi kullanılarak,

$$\beta_t^m = \sum_{m'} \sum_{j=0}^{1} P(r_{t+1}^N \mid S_t = m,\ d_t = j,\ S_{t+1} = m',\ r_t) \times P(d_t = j,\ S_{t+1} = m',\ r_t \mid S_t = m) \tag{7.22}$$

elde edilir. $S_t = m$ ve $d_t = j$ iken denklem (7.22)'un sağındaki ilk terim tamamen $S_{t+1} = f(j, m)$ 'de oluşan yolu yani j girişi ve m durumunu veren bir sonraki durumu tanımlar. Böylece bu şartlar, denklem (7.22)'un ikinci terimi $S_{t+1} = m'$ 'in $S_{t+1} = m$ ile yer değiştirmesine neden olur;

$$\beta_t^m = \sum_{j=0}^{1} P\left(r_{t+1}^N \mid S_{t+1} = f(j, m)\right) P(d_t = j,\ S_t = m,\ r_t)$$

$$= \sum_{j=0}^{1} \delta_t^{j, m} \beta_{t+1}^{f(j, m)} \tag{7.23}$$

Denklem (7.20) t+1 anında, iki ağırlıklı durum metriklerinin toplanması ile oluşan, m durumu ve t anındaki yeni geri durum metriklerini göstermektedir. Ağırlıklandırma, 0 ve 1 veri bitlerine karşılık gelen geçişler ile ilişkilendirilmiş dal metriklerinde oluşmaktadır. Şekil 7.1b'de beta parametresine ait iki farklı gösterim şekli gösterilmektedir. $\beta_{t+1}^{f(j, m)}$, iki farklı durum olduğu zaman (j=0 ya da 1 olması durumunda) t+1 anında geri durum metriği olarak kullanılmaktadır. t anında aynı durum üzerinde önceki zamanın bitiminde iki olası geçiş olduğu zaman, β_t^m , t anında geri durum metriği olarak kullanılmaktadır. Şekil 7.2, ileri ve geri durum metriklerinin hesaplanmasını gösteren grafiktir.

MAP kod çözme algoritması, Viterbi kod çözme algoritmasına bazı yönleriyle benzemektedir [35]. Viterbi algoritmasında, dal metrikleri durum metriklerine eklenmektedir. Daha sonra, bir sonraki durum metriği için, kıyaslama ile en küçük mesafe (en büyük olabilirlik) seçilir. Bu işleme kıyasla-seç (ACS) işlemi denir. MAP algoritmasında, durum metrikleri dal metrikleri ile çarpılır. Daha sonra bu metrikleri kıyaslamak yerine, şekil 7.2'de gösterildiği üzere, bir sonraki ileri (geri) durum metriği olarak toplanır. Viterbi algoritması ile en benzer dizi (yol) aranır, bu sırada sürekli kıyaslama işlemi gerçekleşmektedir ve en iyi yol bulunur. MAP algoritması ile yumuşak değer (olabilirlik ya da olabilirliğin logaritması) aranır; belirli bir zaman aralığındaki olası tüm geçişlerden alınan metrikler işleme sokulur ve zaman aralığı ile ilişkilendirilmiş veri biti hakkındaki en doğru bilgiler ele alınır.

7.3.4. Dal metriklerinin hesaplanması

Denklem (7.10) ile başlanacak olursa,

$$\delta_t^{i,m} = P(d_t = i,\ S_t = m,\ r_t)$$

$$= P(r_t \mid d_t = i,\ S_t = m) P(S_t = m \mid d_t = i) P(d_t = i) \tag{7.24}$$

Burada $r_t = rx_t, ry_t$ şeklinde ifade edilecek olursa, rx_t alınan gürültülü veri biti ve ry_t alınan gürültülü eşlik bitidir. Gürültü, veri bitini etkilemeden önce eşlik ve o anki durum girişten bağımsızdır ve 2^h durumdan herhangi biri olabilir. Burada h, katlamalı kod sistemindeki hafıza elemanlarının sayısıdır. Kısıtlama uzunluğu, K, h+1 değerindedir.

$$P(S_t = m \mid d_t = i) = \frac{1}{2^h}$$

ve

$$\delta_k^{i,m} = P(rx_t \mid d_t = i,\ S_t = m) P(ry_t \mid d_t = i,\ S_t = m) \frac{\mu_t^i}{2^h} \tag{7.25}$$

μ_t^i, $P(d_t=i)$ olarak tanımlanmıştır. $P(RX_t=rx_t)$, rasgele değişkenin olasılığı, RX_t, rx_t üzerinden değer alır ve olasılık yoğunluk fonksiyonu (pdf) $p_{rx_t}(rx_t)$ ile ilişkilidir [35]:

$$P(RX_t=rx_t)=p_{rx_t}(rx_t)drx_t \tag{7.26}$$

Gösterimin kolay olması için, rx_t üzerinden değer alan rasgele değişken RX_t, genelde rasgele değişken rx_t olarak gösterilir. Sıfır ortalamaya sahip ve varyansı σ^2 olan AWGN kanal için, denklem (7.26), denklem (7.25)'deki olasılık terimlerinin yerlerini değiştirerek pdf denklemleri ile kullanılacak olursa:

$$\delta_t^{i,m}=\frac{\mu_t^i}{2^h\sqrt{2\mu\sigma}}\exp\left[-\frac{1}{2}\left(\frac{rx_t-tu_t^i}{\sigma}\right)^2\right]drx_t\frac{1}{\sqrt{2\pi\sigma}}\exp\left[-\frac{1}{2}\left(\frac{ry_t-tv_t^{i,m}}{\sigma}\right)^2\right]dry_t \tag{7.27}$$

elde edilir. Bu denklemde, tu_t ve tv_t iletilen veri ve eşlik bitlerini göstermektedir, drx_t ve dry_t sırasıyla rx_t ve ry_t 'nin diferansiyelleridir ve sabit A_t ile absorbe edilir. tu_t^i parametresi, m durumundan bağımsız olan veriyi simgelemektedir. Bununla birlikte, $tv_t^{i,m}$ parametresi, kod hazıfaya sahip ise, m durumuna bağlı olan eşliği gösterir. Gösterimi basitleştirmek için, olabilirlik oranının pay ve paydasındaki terimleri atarsak;

$$\delta_t^{i,m}=A_t\mu_t^i\exp\left[\frac{1}{\sigma^2}\left(rx_ttu_t^i+ry_ttv_t^{i,m}\right)\right] \tag{7.28}$$

Denklem (7.28) denklem (7.1)'e göre düzenlenecek olursa;

$$\Lambda(\hat{d}_t)=\mu_t\exp\left(\frac{2rx_t}{\sigma^2}\right)\frac{\sum_m\alpha_t^m\exp\left(\frac{ry_ttv_t^{1,m}}{\sigma^2}\right)\beta_{t+1}^{f(1,m)}}{\sum_m\alpha_t^m\exp\left(\frac{ry_ttv_t^{0,m}}{\sigma^2}\right)\beta_{t+1}^{f(0,m)}} \tag{7.29a}$$

$$=\mu_t \exp\left(\frac{2rx_t}{\sigma^2}\right)\mu_t^e \tag{7.29b}$$

$$L(\hat{d}_t)=L(d_t)+L_K(rx_t)+L_e(\hat{d}_t) \tag{7.29c}$$

Her bir t anındaki $\mu_t=\mu^1/\mu^0$, önceki olasılık oranı girişi ve μ_t^e çıkış artık (extrinsic) olasılığıdır. Denklem (7.29b)'de μ_t^e, bir veri biti hakkında önceki giriş bilgisini değiştirmesinden dolayı düzeltme terimi olarak adlandırılır. Turbo kodlarda, bu düzeltme terimleri, kod çözme hata olasılığını en az hale getirmek ve her veri biti için olasılığı arttırmak için bir kod çözücüden diğerine geçmektedir. Böylece, kod çözme işlemi denklem (7.29b)'yi kullanarak, çeşitli iterasyonlar için $\Lambda(\hat{d}_t)$'yı hesaplar. Artık olasılık olan μ_t^e, belirli bir iterasyon sonucu oluşan ve bir sonraki iterasyon için öncelikli olasılık oranı μ_{t+1} ile yer değiştirir. Denklem (7.29b)'de $\Lambda(\hat{d}_t)$'nin logaritması alınarak denklem (7.29c) olan $L(\hat{d}_t)$ yani kod çözme işlemi sonucunda elde edilen üç LLR teriminden (öncelikli LLR, kanal ölçüm LLR ve artık LLR) oluşan yumuşak karar (sayı) elde edilir.

MAP algoritması, denklem (7.29a) ve (7.29b)'de görüldüğü üzere, olasılık oranı, $\Lambda(\hat{d}_t)$'nın terimlerinde işletilir. Bununla birlikte, işlem olasılık oranlarını kullanır. Olasılık oranları çarpma işlemlerinden dolayı çok karmaşıktır. MAP algoritmasının denklem (7.29c)'de gösterildiği şekilde logaritmik olarak işletilmesi [34,18] ile çarpma işlemlerinin yok edilmesinden dolayı karmaşıklık azaltılmaktadır.

7.4. Log-MAP (Logaritmic Maximum A Posteriori) Algoritması

MAP algoritmasının çok karmaşık matematik hesaplamaları gerektirdiğinden dolayı pratik uygulamalarda kullanımı uygun değildir. Hesaplama karmaşıklığını gidermek için MAP algoritması logaritmik alanda düzenlenerek Max-Log-MAP algoritması oluşturulmuştur [36]. Fakat MAP algoritmasının başarımı özellikle düşük SNR değerlerinde düşmüştür [36]. Log-MAP algoritmasında, her bir max işlemi sırasında

basit bir düzeltme fonksiyonu kullanılarak MAP algoritmasının verdiği başarım elde edilmiştir [37].

Log-MAP algoritmasını elde etmek için MAP algoritması aşağıdaki şekilde değiştirilmelidir;

- Çarpma işlemleri, toplama işlemlerine dönüştürülür.
- Toplama işlemleri, max∗(·) işlemine dönüştürülür.

$$\max^*(rx, ry)=\log(e^{rx}+e^{ry})=\max(rx, ry)+\log(1+e^{-|rx-ry|}) \quad (7.30)$$

şeklinde max işlemi hesaplanmaktadır.

İleri durum metriği : $\log(\alpha_t^m)$

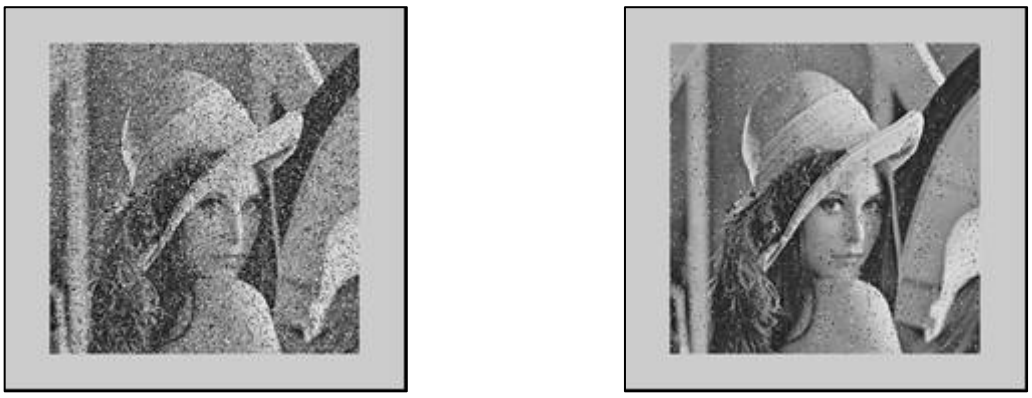
Geri durum metriği : $\log(\beta_t^m)$

Dal metriği : $\log(\delta_t^{i,m})$

şeklinde hesaplanmaktadır.

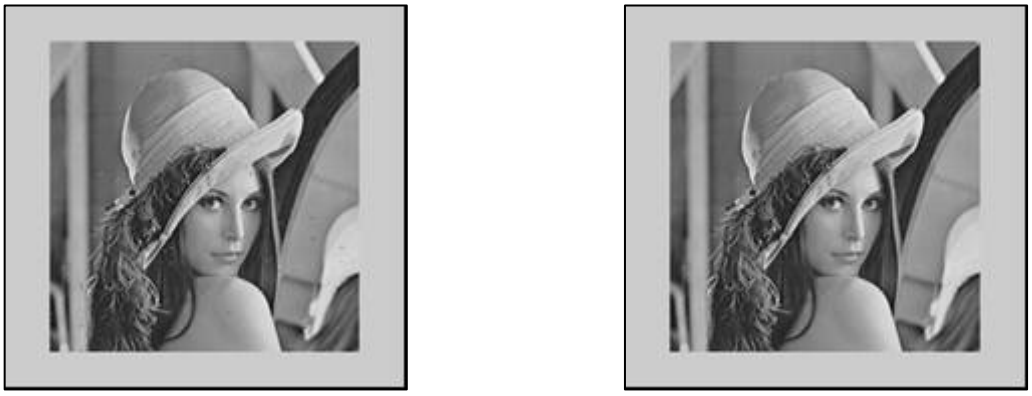
7.5. Log-MAP Kod Çözme Algoritması Uygulaması

SOVA kod çözme algoritması ile bmp uzantılı bir resim, Matlab Simulink programı kullanılarak vericiden alıcıya gönderilmiştir. Bu uygulama farklı SNR değerleri ile 5 döngü sayısı için gerçekleştirilmiştir.



Şekil 7.3: SNR=0 ve 1 değerlerindeki Log-MAP çıkışları

Beş döngülü kod çözme işlemi sonucunda sol tarafta SNR 0 değeri ve sağ tarafta SNR 1 değeri için elde edilmiştir.

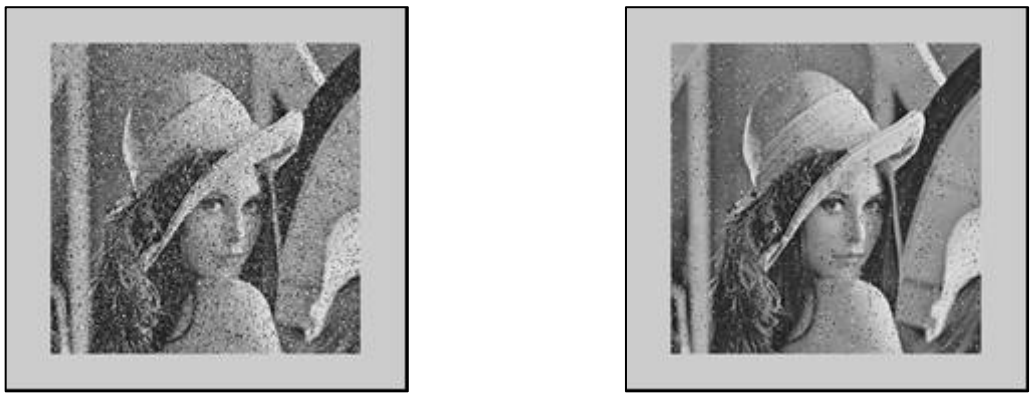


Şekil 7.4: SNR=2 ve 3 değerlerindeki Log-MAP çıkışları

Beş döngülü kod çözme işlemi sonucunda sol tarafta SNR 2 değeri ve sağ tarafta SNR 3 değeri için elde edilmiştir.

Şekil 7.5: SNR=4 değerindeki Log-MAP çıkışı

Beş döngülü kod çözme işlemi sonucunda SNR 4 değeri için elde edilmiştir. Şekil 7.3, 7.4, 7.5 ‘da görüldüğü üzere, SNR değeri arttıkça BER düşmekte ve kaliteli iletim yapılmış olmaktadır.



Şekil 7.6: SNR 1 değeri için solda 1 döngü sağda 5 döngü sayısı için Log-MAP çıkışları

Şekil 7.6’da görüldüğü gibi, döngü sayısı arttıkça hata oranı azalmış ve daha iyi bir görüntü elde edilmiştir.

7.6. Sonuç

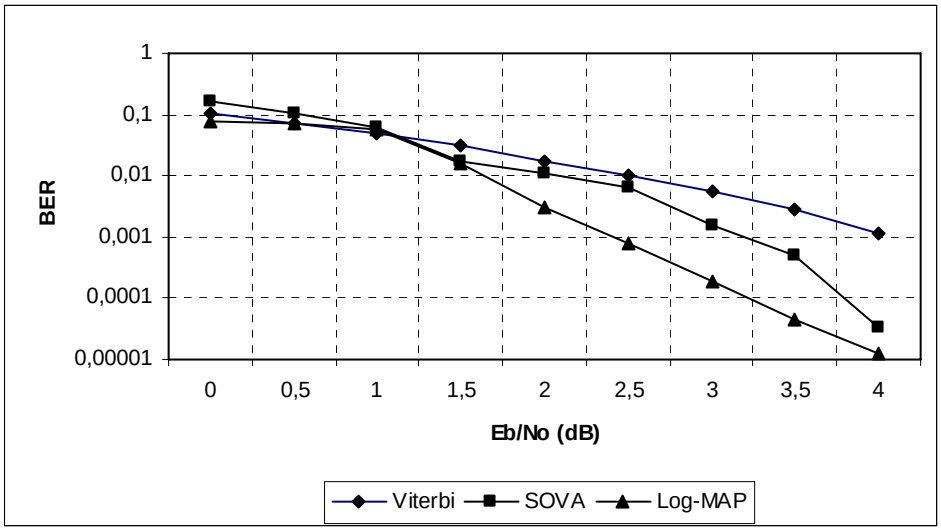
Bu bölümde Turbo kod çözme algoritmalarından uygulamalarda en çok kullanılan ve en iyi BER başarımını gösteren MAP algoritmasının logaritmik ifadesi olan Log-MAP algoritması hakkında detaylı bilgi verilmiştir.

8. SONUÇ VE ÖNERİLER

Katlamalı kodlarda yaygın olarak kullanılan başarım ölçütleri BER, hesaplama karmaşıklığı, kısıtlama boyutu ve hafıza sayısıdır. Tez çalışmasında bahsedilen başarım ölçütlerine göre hata düzeltme tekniklerinin karşılaştırılması yapılmaktadır.

Bu bölümde Turbo kod çözme ve Viterbi kod çözme algoritmalarının bit hata oranı (BER) ve hesaplama karmaşıklığına göre karşılaştırılmalı başarım analizleri yapılarak gerçekleştirilen örnek bir resim iletim uygulamasında elde edilen başarım sonuçları değerlendirilmektedir. Karşılaştırmalı başarım analizi için kablosuz ortamdan iletilecek trafikler AWGN (Additive White Gaussian Noise) kanal üzerinden ve BPSK (Binary Phase Shift Keying) modülasyon tekniği kullanılarak 100 bit uzunluğundaki veri paketleri şeklinde 50 bilgi bloğu olarak gönderilmektedir. Kodlama oranı olarak 1/2 seçilmiştir.

Modellerin benzetimleri süresince SNR değerleri 0 ile 4 arasında değiştirilerek, Viterbi, SOVA ve Log-MAP hata düzeltme algoritmalarının her birisi için elde edilen BER değerleri Şekil 8.1'de gösterilmiştir. Ayrıca, Şekil 8.2'de Viterbi algoritmasına göre normalize edilmiş SOVA ve log-MAP algoritmaları için BER başarım sonuçları sunulmaktadır. Normalizasyon işleminde, Viterbi algoritmasının BER değerleri 1 kabul edilerek diğer algoritmaların BER sonuçları göreceli olarak hesaplanmıştır.

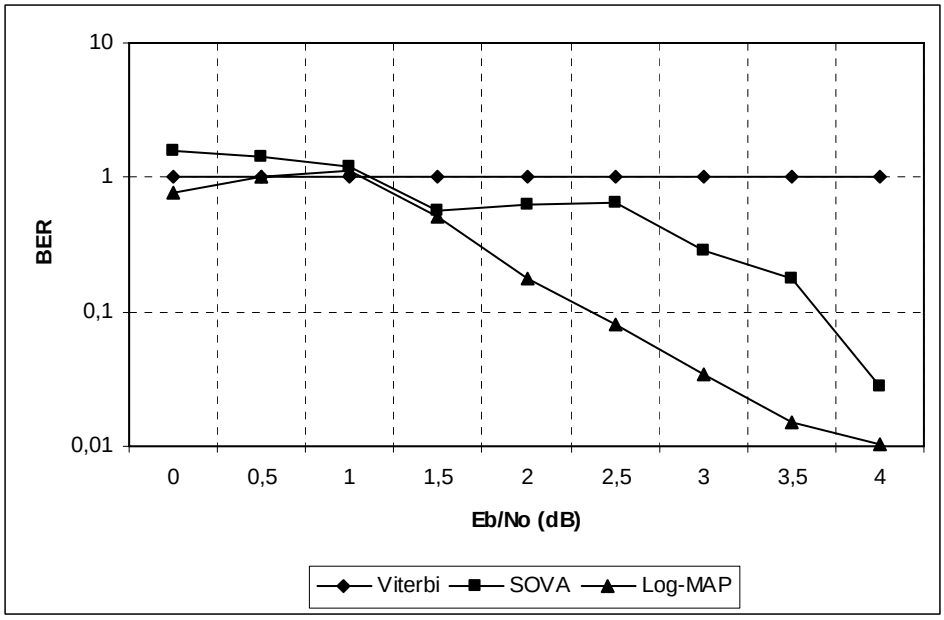


Şekil 8.1: Hata düzeltme tekniklerinin BER başarım sonuçları

Şekil 8.1'deki grafiğin sayısal sonuçları tablo 8.1'de gösterilmektedir.

Tablo 8.1 : Hata düzeltme tekniklerinin sayısal sonuçları

SNR (dB)	Viterbi	SOVA	Log-MAP
	BER Sonuçları		
0	0,1035	0,16327	0,079219
0,5	0,07344	0,10544	0,073638
1	0,04983	0,060544	0,05567
1,5	0,03079	0,017613	0,01565
2	0,0174	0,010908	0,003129
2,5	0,01001	0,0064838	0,0007988
3	0,005453	0,0015503	0,00018976
3,5	0,002887	0,00051793	0,000043416
4	0,001187	0,000033104	0,000012378



Şekil 8.2: Turbo kod çözme algoritmalarının normalize edilmiş BER başarım sonuçları

Turbo kod çözme algoritmaları, şekillerden de görülebileceği gibi, özellikle SNR değeri arttığında, daha iyi BER başarımı sağlamıştır. Viterbi kod çözme algoritmasında en iyi durumda yaklaşık 10^{-3} civarında BER değeri elde edilirken, Turbo kod çözme algoritmasında (log-MAP) bu oran 10^{-5} lere kadar düşmektedir. Buradan da anlaşılacağı üzere Turbo kodlar en iyi durumda yaklaşık olarak 100 kat daha iyi sonuç vermektedir.

Bununla birlikte, Turbo kod çözme algoritmaları için daha karmaşık işlemler gerekmektedir. Turbo kod çözme algoritmalarından biri olan SOVA algoritması, Viterbi algoritmasından iki kat daha karmaşıktır. Bunun nedeni, Turbo kod çözme algoritmasında iki adet SOVA kod çözücüsüne ihtiyaç vardır ve döngülü kod çözmeye dayalı bir algoritma olduğu için karmaşık hesaplama gerektirir. Döngü sayısı artışına bağlı olarak da işlem karmaşıklığı da artar. BER sonuçları elde edilirken Turbo kod çözme algoritmaları için döngü sayısı 5 olarak alınmıştır. Döngü sayısının artışı BER başarımını da arttırmaktadır. Diğer bir Turbo kod çözme algoritması olan Log-MAP algoritması ise, SOVA algoritmasından daha çok matematiksel işlem gerektirmektedir. Bu nedenle Log-MAP algoritması SOVA algoritmasından daha iyi BER başarımı göstermiştir.

Elde edilen sonuçlardan da anlaşılacağı üzere gecikme duyarlı uygulamalarda SOVA, veri kaybına duyarlı uygulamalarda ise Log-MAP algoritmasının kullanılması daha iyi sonuçlar vermektedir.

Turbo kodlar uzun veri bloklarına uygulandığında daha iyi sonuçlar verebilmektedir. Bununla birlikte, uzun veri bloklarında yapılan kodlama / kod çözme işlemleri oldukça uzun sürebilmektedir. Bu ise özellikle gecikme ve gecikme değişimine duyarlı gerçek zamanlı uygulamalar için uygun değildir. Bu çalışmada geçerli ve kullanılabilir bir başarım analizi yapabilmek amacıyla veri blokları ve blok uzunlukları kısa tutulmuştur.

Turbo kod algoritmaları da kendi aralarında karşılaştırıldığında; Log-MAP algoritması, kullandığı karmaşık hesaplama ve MAP kod çözme algoritmasına yakın başarım gösterdiği için SOVA algoritmasından daha iyi sonuçlar vermiştir. Benzetim süresi ve hesaplama karmaşıklığı nedeniyle 5 döngü sayısı dikkate alındığında Log-MAP algoritması Viterbi algoritmasından 20 kat ve SOVA algoritmasından 2 kat daha karmaşık yapıdadır.

Çalışmalarımızdaki benzetimlerden özetle aşağıdaki sonuçlar elde edilmiştir:

- Çerçeve (frame) sayısı arttıkça Turbo kodlayıcının başarımı da artmaktadır.
- Sabit çerçeve boyutunda, Turbo kodun başarımını kısıtlama boyutu ve kod oranı etkilemektedir. Kısıtlama boyutu arttıkça Turbo kodun başarımı artmaktadır. Turbo kodun kod oranı azaldıkça da başarımı artmaktadır.
- Turbo kod çözme işlemi bir döngüden fazla yapıldığında verimli bir kod çözme kazancı sağlanmaktadır.

Turbo kod çözme işleminde hesaplama yoğunluğundan dolayı çoğu benzetimler yüksek kod oranında, küçük kısıtlama uzunluğunda ve küçük çerçeve boyutunda yapılmaktadır. Turbo ve Viterbi kod çözme algoritmaları bmp uzantılı resim iletim uygulamasında test edilmiştir (Döngü sayısı 5 ve SNR 4dB). Uygulamada kullanılan resim 1024 çerceveden ve her çerceve 512 bitten oluşmaktadır. Tüm kod çözme algoritmaları şekil 8.3'de gösterilmiştir.

Viterbi, SOVA ve Log-MAP algoritmaları için sırasıyla $2x10^{-4}$, $5.7x10^{-6}$ ve 0 şeklinde BER sonuçları elde edilmiştir. Resim iletimimde oluşan hatalar daireler içerisinde gösterilmiştir.

a) Viterbi

b) SOVA

c) Log-MAP

Şekil 8.3: Resim iletimi sonuçları

Tez çalışması kapsamında ileri hata düzeltme tekniklerinde kullanılan Viterbi ve Turbo kod çözme algoritmalarının kullanımını içeren eğitim amaçlı bir uygulama da MATLAB grafik kullanıcı ara yüzü kullanılarak geliştirilmiştir.

KAYNAKLAR

[1] Çeken, C., "Kablosuz ATM Kullanarak Servis Kalitesi Desteği Sağlanmış Gerçek Zamanlı Veri Transferi", Doktora Tezi, ***Kocaeli Üniversitesi Fen Bilimleri Enstitüsü***, İzmit, 49-50, (2004).

[2] Proakis, J. G., "Digital Communications", 3rd edition, New York, ***McGraw-Hill***, (1995).

[3] C.Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," ***ICC 1993***, Geneva, Switzerland, pp. 1064–1070, May 1993.

[4] Robertson, P, Villebrun, E., and Hoeher, P., "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," ***Proc. of ICC '95***, Seattle, Washington, pp. 1009–1013, (1995).

[5] Shannon, C. E. "A mathematical theory of communications," ***Bell Systems Technical Journal***, Vol. 27, pp.379–423 and 623–656, (1948).

[6] Viterbi, A. J., and Omura, J. K., "Principles of Digital Communication and Coding," ***Mc-Graw Hill***, New York, (1979).

[7] Pietrobon S.S. ve Barbulescu A.S., "A simplification of the modified Bahl decoding algorithm for systematic convolutional codes", ***Australian Space Center for Signal Proc. Uni. of South Australia Res.*** Paper, Avustralya, (1996).

[8] Yan, W., Tsui C. Y., Cheng R. S. K., "A reduced complexity implemention of the Log-Map algorithm for Turbo-codes decoding", ***Acoustics, Speech, and Signal Processing, 2000, CASSP '00. Proceedings. 2000 IEEE International Conferenc***e, volume 5, 5-9 June 2000 Page(s):2621 - 2624 vol.5, 10.1109/ICASSP.2000.861007, (2000).

[9] Wu, P.H.-Y., "On the Complexity of Turbo Decoding Algorithms", ***Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd,*** volume 2, 6–9 May 2001 page(s):1439 – 1443 vol.2, 10.1109/VETECS.2001.944625, (2001).

[10] Ehtiati, N., Soleymani, M.R., Sadjadpour, H. "Joint interleaver design for multiple turbo codes", ***Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th***, volume 4, 26-29 Sept. 2004 page(s):2302 – 2306, 10.1109/VETECF.2004.1400463, (2004).

[11] Kouraichi, M., Belghith O, B., Kachouri, A., Kamoun, L., "Evaluation of SOVA algorithm in turbo code" ***Control, Communications and Signal Processing, 2004.***

First International Symposium, 2004 page(s):659 – 663, 10.1109/ISCCSP.2004.1296491, (2004).

[12] Jaspar, X., Vandendorpe, L., "New iterative decoding of variable length codes with turbo-codes", ***Communications, 2004 IEEE International Conference***, volume 5, 20-24 June 2004 Ppage(s):2606 - 2610 Vol.5, 10.1109/ICC.2004.1313003, (2004).

[13] Talakoub, S., Sabeti, L., Shahrrava, B., Ahmadi, M., **"**A linear log-MAP algorithm for turbo decoding and turbo equalization", ***Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference***, volume 1, 22-24 Aug. 2005 page(s):182 – 186 Vol. 1, 10.1109/WIMOB.2005.1512836, (2005).

[14] Ghrayeb, A., Chuan Xiu Huang, "Improvements in SOVA-based decoding for turbo-coded storage channels", ***Magnetics, IEEE Transactions***, volume 41, Issue 12, Dec. 2005 page(s):4435 – 4442,10.1109/TMAG.2005.857453, (2005).

[15] Bing Du, "Multimedia communication in wireless environment", ***Multi-Media Modelling Conference Proceedings, 2006 12th International***, 4–6 Jan. 2006 Page(s):4 pp., 10.1109/MMMC.2006.1651372, (2006).

[16] Chemak, C., Bouhlel, M.S., "Near Shannon Limit for Turbo Code with Short Frames" ***Information and Communication Technologies, 2006. ICTTA '06. 2nd***, volume 1, 24-28 April 2006 page(s):1994 – 1997, (2006).

[17] CISCO, 2001. ***Cisco Network Academy***. Cisco Systems, Inc.

[18] ILIC, Z., BAZENT, A., and KOS, M., "Data Link Control Schema for Wireless ATM Transmission", ***IEEE Proc.***, pp. 1400–1404, (2001).

[19] P. Elias, "Coding for Noisy Channels", ***IRE Conv. Rec.***, Part 4, pp. 37–47, (1955).

[20] Wicker, S. B., "Error Control Systems for Digital communication and Storage", New Jersey, ***Prentice-Hall***, (1995).

[21] Hagenauer, J. and Hoeher, P., "A Viterbi Algorithm with Soft-Decision Outputs and Its Applications," ***GLOBECOM 1989***, Dallas, Texas, pp.1680-1686, (Nov.1989).

[22] Woerner, B. D., "EE 5984 - Digital Communications - Lecture Notes", ***Virginia Tech***, (1994).

[23] Viterbi, A. J., "Convolutional Codes and Their Performance in Communication Systems," ***IEEE Transactions on Communications Technology***, Vol. COM–19, No. 5, pp. 751–772, (October 1971).

[24] Rappaport, T. S., "Wireless Communications Principles and Practice", New Jersey, ***Prentice-Hall***, (1996).

[25] Battail, G., Berrou, C., and Glavieux, A., "Psuedo-Random Recursive Convolutional Coding for Near-Capacity Performance," ***GLOBECOM 1993***, pp. 23–27, (Dec. 1993).

[26] Divsalar, D. and Pollara, F., "Turbo Codes for Deep-Space Communications," ***JPL TDA Progress Report***, page 42-120, (Feb. 15, 1995).

[27] Divsalar, D. and Pollara, F., "Turbo Codes for PCS Applications," ***Proceedings of ICC 1995***, Seattle, WA., pp. 54–59, (June 1995).

[28] Barbulescu, A. S. and Pietrobon, S. S., "Interleaver Design for Turbo Codes," ***Electronics Letters***, Vol. 30, No. 25, pp. 2107–2108, (Dec. 8, 1994).

[29] L. Bahl, J. Cocke, F. Jeinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", ***IEEE Trans. Inform. Theory***, vol. 20, pp. 248–287, (Mar. 1974).

[30] Berrou, C., Adde, P. Angui, E., and Faudeil, S., "A Low Complexity Soft-Output Viterbi Decoder Architecture," ***Proceedings of ICC 1993***, Geneva, Switzerland, pp. 737–740, (May 1993).

[31] Hagenauer, J., Offer, E., and Papke, L., "Iterative Decoding of Binary Block and Convolutional Codes," ***IEEE Transactions of Information Theory***, Vol. 42, No. 2, pp. 429–445, (March 1996).

[32] Hagenauer, J., Robertson, P., and Papke, L., "Iterative ("Turbo") Decoding of Systematic Convolutional Codes with the MAP and SOVA Algorithms," ***Proceeding of ITG***, pp. 21–29, (Oct. 1994).

[34] Pietrobon, S. S., "Implementation and Performance of a Turbo/MAP Decoder," ***Int'l. J. Satellite Communications***, vol. 15, pp. 23–46, (Jan-Feb 1998).

[35] Sklar, B., "Digital Communications: Fundamentals and Applications, Second Edition", Upper Saddle River, ***NJ: Prentice-Hall***, (2001).

[33] Hagenauer, J., "Source-Controlled Channel Decoding," ***IEEE Transactions on Communications***, Vol. 43, No. 9, pp. 2449–2457, (Sept. 1995).

[36] J.A.Erfanian, S.Pasupathy, and G.Gulak, "Reduced complexity symbol detectors with parallel structures for isi channels," ***IEEE Trans. Commun***., vo1.42 pp.1661-1671, (February/March/April 1994).

[37] Robertson P, Villebmn E, Hoeher P. "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", ***ICC '95*** pp.1009–13 v01.2, (1995).

KİŞİSEL YAYINLAR VE ESERLER

1. Calhan A., Ceken C., Erturk I., "Comparative Performance Analysis of Forward Error Correction Techniques Used in Wireless Communications", ***Third International Conference on Wireless and Mobile Communications, ICWMC 2007,*** 4-9 Mart 2007, Guadeloupe, French Caribbean (**Kabul edildi**).
2. Çalhan A., Çeken C., Ertürk İ., "İleri Hata Düzeltme Tekniklerinin Resim İletim Uygulamasında Karşılaştırılmalı Olarak İncelenmesi", ISSN:1306-3111, ***e-Journal of New World Sciences Academy 2007***, Volume: 2, Number: 1, Article Number: A0013.

ÖZGEÇMİŞ

1981 yılında Aydın'ın Söke ilçesinde doğdu. İlk, orta ve lise öğrenimini Söke'de tamamladı. 1999 yılında girdiği Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Öğretmenliği programından 2003 yılında Bilgisayar Teknik Öğretmeni olarak mezun oldu. Aynı yıl atandığı MEB'de Bilgisayar Öğretmeni olarak 3 yıl görev yaptı. Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'nda 2003 yılında başladığı yüksek lisans eğitimine devam etmektedir.

Halen Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik Bilgisayar Eğitimi Bölümünde Araştırma Görevlisi olarak çalışmaktadır.