T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS TEZİ

KODLAŞTIRMA TEORİSİ ÜZERİNE

Şefik YAMALI

Matematik Anabilim Dalı

Tezin Sunulduğu tarih : **29.06.2010**

Tez Danışmanı:

Prof. Dr. Yakup HACI

ÇANAKKALE

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

ŞEFİK YAMALI tarafından **PROF. DR. YAKUP HACI** yönetiminde hazırlanan **"KODLAŞTIRMA TEORİSİ ÜZERİNE"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. Yakup HACI
Danışman

Prof. Dr. Bilgehan GÜVEN
Jüri Üyesi

Prof. Dr. İsmail TARHAN
Jüri Üyesi

Sıra No : ……..
Tez Savunma Tarihi : 29/06/2010

Prof. Dr. İsmail TARHAN
Müdür
Fen Bilimleri Enstitüsü

İNTİHAL (AŞIRMA) BEYAN SAYFASI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belirttiğimi beyan ederim.

Şefik YAMALI

TEŞEKKÜR

Tezimin hazırlanması süresince matematiksel bakış açısını, tecrübesini ve öngörüsünü benimle paylaşan her türlü yardımını esirgemeyen ve bilimsel konularda daima yardımını gördüğüm tez yöneticisi hocam, Sayın Prof. Dr. Yakup HACI' ya ve tezimi hazırladığım sürede bana sevgi, güç ve destek veren aileme teşekkür ederim.

Şefik YAMALI

SİMGELER VE KISALTMALAR LİSTESİ

$\sum$	Toplam Sembolü
$\in$	Eleman
$\notin$	Eleman değil
$\leq$	Küçük Eşit
$\geq$	Büyük Eşit
$=$	Eşittir
$\neq$	Eşit değil
$G^{\perp}$	G kodunun duali
$\forall$	Her
$\cup$	Birleşim
$\cap$	Kesişim
$\supset$	Kapsar
$\subset$	Alt küme
$\subseteq$	Alt küme ve eşit
G	Üreteç matrisi
I_k	Birim matris
L(A)	Mesaj sözünün uzunluğu (blok uzunluğu)
$\Rightarrow$	İse

ÖZET

KODLAŞTIRMA TEORİSİ ÜZERİNE

Şefik YAMALI
Çanakkale Onsekiz Mart Üniversitesi
Fen Bilimleri Enstitüsü
Matematik Anabilim Dalı Yüksek Lisans Tezi
Danışman: Prof. Dr. Yakup HACI
29/ 06 /2010, 78

Kodlaştırmanın matematikte önemli rol oynaması bu teorinin daha derinden incelenmesini gerektirir. Birçok problemin çözümünde kodlaştırmada tek değerlilik özelliğinin belirlenmesi şarttır. Tez konusunun amacı kodlaştırmayı matematiksel biçimde tanımlayarak onun bazı önemli problemlerini incelemektir. Graf teorisi incelenip ardından alfabe kodlaştırması tanımlanarak böyle kodlaştırma için tek değerlilik problemi ele alınacaktır. Bu amaçla söz konusu problemin çözümü için yeterli, gerek ve yeterli koşullar incelenecektir. Daha sonra Hamming Kodlarının yapısı üzerine çalışmalar yapılacaktır.

Anahtar sözcükler: Graf teorisi, alfabe kodlaştırması, dekodlaştırma, kodlaştırmanın tek değerlilik problemi, Hamming kod yapısı, üreteç matrisler

ABSTRACT

ABOUT THE CODİNG THEORY

Şefik YAMALI
Çanakkale Onsekiz Mart University
Graduate School of Science and Engineering
Chair for Mathematic Thesis of Master of Science
Advisor: Prof.Dr. Yakup HACI
29/ 06 /2010, 78

That coding plays on important role in maths forces this theory to be examined in details. In the solution of most problems, it is obligatory to determine the feature of the unique valuableness in codding. The aim of the thesis's subject is to examine same important problems by defining coding mathematically. First, Graf theory will be examined; then the problem of the unique valuableness will be told for this kind of coding by defining the alphabeth coding. With this aim, the suitable conditions will be examined for the solution of this problem. Later the studies on the shape of Hamming codes will be done.

Keywords: Graph Theory, the alphabeth coding, decoding, the problem of the unique valuableness in coding, the shape of Hamming codes, generator matrix

İÇERİK

Sayfa

BÖLÜM 1
GİRİŞ

Kod, matematik alanında istatistik biliminde, bilgisayar alanında şifreleme konusunda, haberleşme alanında, haberleşme karmaşıklığını en düşük seviyeye indirmek ve kaynakların etkin biçimde temsil edilmesini sağlamak için kullanılmaktadır. Kodlaştırma, iletişim alanında kaynakların, kanalların, alıcıların bilgi karakteristiklerini incelemek, bilginin iletimini optimize etmek ve iletimin güvenilirliğinin düzeltilmesini sağlamak amacıyla kullanılmaktadır. Kodlaştırmanın temelini oluşturan İletişim Teorisi Kuramı 1948 yılında Claude Shannon tarafından geliştirilmiştir. İlk yıllarında teori akademik bir çalışma konusu olarak görülmüş, iletişim sistemlerinin bu yaklaşımla daha etkili ve güvenilir hale getirilmesi öngörülememiştir. Ancak 1970'lerin ortalarında bu yargı ortadan kalkmış ve teori haberleşme sistemlerinin büyük bir çoğunluğunun alt yapısında kullanılmaya başlamıştır. Bilginin iletimi için vericiden gönderilen mesaj sözcüğü, sözcüğün gönderildiği kanaldan ve aynı zamanda gönderen veya mesajı alandan kaynaklanan mesaj sözcüğü etkilerden dolayı, bozulmaya uğrar ve alıcıda gönderilen mesajdan farklı bir bilgi oluşabilir. Bu durum haberleşme sisteminin hatalı mesaj göndermesi olarak adlandırılır. Kodlaştırmanın amacı da bu hatalı mesajları düzelterek doğru bilgi aktarımını en iyi duruma getirmektir. Bu tezde bu durum, kodların matematiksel yapısı ile birlikte incelenmiştir.

Tezin 2. bölümünde, kodlaştırma teorisine yardımcı konu olarak Graf Teorisi Temel Terimleri incelenmiştir. İlk olarak terimlerin tanımları şu şekilde verilmiştir: Noktalar ve bunları birbirine bağlayan kenar adı verdiğimiz eğri parçalarından oluşan şekle graf (çizge) denmektedir. Matematiksel anlamda bir graf V kümesiyle V'nin iki elemanlı alt kümelerinden oluşan bir E kümesinden meydana gelmektedir. Eğer grafın bütün kenarları yöne sahipse grafa yönlü graf, aksi halde yönü olmayan graf denir. Eğer bir grafta hem yönlü hem de yönü olmayan kenarlar varsa grafa karışık graf denir. İki nokta arasında en az bir kenar varsa bunlara komşu noktalar denir. Bir noktayı kendisine bağlayan kenara şeklinden dolayı ilmek adını alır. İlmeği ve iki noktası arasında en çok bir kenarı bulunan graflara basit graflar denir. İki noktası arasında birden çok kenar bulunan graflara, multigraf (çoklu çizge) denir. Bir grafta her bir kenara bir reel sayı atanmışsa bu grafa ağırlıklı graf veya ağ denir. Hem ilmek, hem de çoklu kenarlara sahip yönü olmayan grafa pseudo graf denir. Bir noktadaki kenarların sayısına (yerel) derece denir.

Bütün noktaların dereceleri aynı olan graflara düzgün (regüler) graflar denir. Bir grafta tek başına kalmış noktalar bulanabilir. Böyle noktalara izole noktalar denir. Bütün noktalan izole olan bir grafa boş graf veya sıfır grafı denir. Her noktaya tam iki kenar değen tek parça n noktalı graflara döngü denir. Düğüm kümesi, tüm kenarların V_1'in bir düğümünü V_2'nin bir düğümüne bağlandığı $\{V_1,V_2\}$ şeklinde bir bölmelemeye sahip olan grafa iki parçalı(bipartite) graf denir. Noktaları n ve m elemanlı olmak üzere iki A ve B kümesine ayrılmış, A'daki her noktanın B' deki her noktaya bağlandığı başka da kenarı olmayan graflara iki parça ya da iki kümeli tam graf denir. $V^{|} \subseteq V$ (düğümler kümesi), $E^{|} \subseteq E$ (kenarlar kümesi) ve $\delta_G{}^{|}(e)= \delta_G(e)$ ise $G^{|}=(V^{|},E^{|})$ grafına G=(V,E) grafının alt grafı denir ve $G^{|} \leq G$ şeklinde gösterilir.

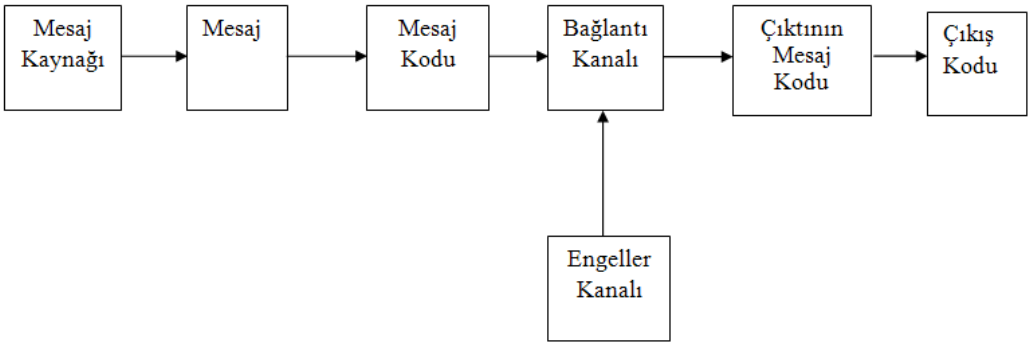
2. bölümün 2. başlığı altında tek hamlede çizilebilen graflar anlatılmıştır. 3. başlıkta ise Euler Döngüsü'ne değinilmiştir. Her kenardan tam bir kez geçen başladığı noktaya geri dönen yolculuklara Euler döngüsü denir. 4. başlıkta Hamilton Döngüsü verilmiştir. Bir G grafı verildiğinde her noktadan yalnız bir kez geçmek şartı ile kapalı bir yol oluşturabilen graflara Hamilton Grafı denir. İzomorfik graflar 5. başlıkta incelenmiştir. İki Graf verildiğinde bu iki grafta kenarlar ve konumlar arasındaki ilişki aynı olabilir. Bu durumda benzer özellikte iki farklı geometrik şekiller ortaya çıkabilir. Aynı durumu yansıtan, başka bir deyişle aynı yapıya sahip graflara eş yapılı graflar denir. Başka bir deyişle bunlara izomorfik graflar denir. İki grafın eş yapılı olabilmesi için;

- Kenar sayıları aynı olmalıdır.
- Nokta sayıları aynı olmalıdır.
- Nokta dereceleri aynı olmalıdır.
- Noktalar arasındaki ilişkiyi gösteren matrisler aynı olmalıdır.

2.bölümün son başlığında ise, komşuluk ve incidence matrisleri anlatılarak örnekler verilmektedir.

Tezin 3. bölümünde asıl araştırma konumuz olan kodlaştırma teorisi üzerinde çalışılmıştır. İlk olarak kodlaştırma teorisinin gelişim aşamalarından bahsedilmiştir. Kodlaştırma teorisi C.E.Shannon ve R.W.Hamming ile başladı. Daha önemlisi, J.C.Golay birden çok hata düzelten kodların iki yapısını verdi. R.W.Hamming ve J.C.Golay tarafından yapılan keşifler hem mühendisler hem de matematikçiler tarafından araştırma sahaları oluşturdu. Mühendisler ilk önce gelişmiş bilgi iletişimi için yeni olanakları kullanmak istedi. Diğer taraftan, matematikçiler ise daha çok kodların cebirsel yapılarına ilgi duydu. Dolayısıyla kodlaştırma teorisi bu şekilde gelişti.

3. bölümün 2. başlığı altında kodlaştırma teorisinin temel terimleri açıklanmıştır. Kodlaştırmanın asıl amacı olan ‘doğru bilgi iletimini’ gösteren tablomuz şu şekildedir:



Kodları 1 ve 0 dan oluşan diziler temsil etmektedir. Kodların en önemli özelliği, ikili dizilerle (binary strings) oluşturulmuş mesajların kodun öğelerini sıralayarak gösterilmesidir, bu sıralama eşsizdir. Eğer mesajın şifresi çözülmüşse, kodu temsil eden harflerin sıralamasında problem olmamalıdır. Bu koda “uniquely decipherable code”, “şifresi çözülebilir eşsiz kod” denilmektedir. Aynı uzunluktaki dizi çiftlerinden oluşan kodlara blok kodlar denir. Eğer kodun öğesinin başka bir kodun öğesinin başlangıç dizisi olmama özelliği varsa bu kodlara prefix kodlar denilmektedir. Prefix code’lara ayrıca “instantaneous code” (anlık kod) da denilmektedir. 1’lerin sonunda 0’ın takip ettiği dizilerden oluşan kodlara comma (virgül) kod denmektedir. Bazen depolama alanının küçültmek ve gönderme zamanının azaltmak için verilerin sıkıştırılması istenebilir. Alanları minimize etmekle en etkili kod “Huffman code”dur ayrıca bu kodun “anlık kod” yani Prefix kod olma avantajı vardır. Gönderme süresini azaltan kodlara verilebilecek bilindik bir örnek “Morse code”dur. Hata olduğunu algılama, tespit etme özelliği olan kodlara “error-detecting codes” , “hata bulma kodları” denir. Hataları düzeltme özelliği bulunan kodlara “error-correcting codes” , “hata düzeltme kodları” denir. Diğer bir kod çeşidimiz de Gray Code’udur. Yansımalı kodlar adıyla anılan Gray kodunda sayılar arasındaki geçişte sadece bir bit değişir.

Bu kod, hatayı azalttığından dolayı özellikle Analog-Sayısal dönüştürücülerde, bilgisayar kontrollü cihazlarda oldukça tercih edilen bir kodlamadır. Bilgi değişimlerini yani hata oluşumlarını denetleyebilmek ve gönderilen bilginin doğruluğunu kontrol etmek amacı ile Parity Kodu ortaya çıkmıştır. Parity bitinin genel kullanımı şu

şekildedir: Eğer kodumuzda tek sayıda 1 varsa parity bitimiz 1, çift sayıda 1 var ise 0 ayarlanır. Sonuçta kod parity biti ile çift sayıda 1 içerir. Bilgisayar dili olan "ASCII Code" u (American Standart Code for Information Interchange - Bilgi Alışverişi için Standart Amerikan Kodu) parity bit metodu kullanılarak oluşturulmuştur.

3. başlık altında dekodlaştırmanın tek değer problemini incelenmiştir. Bunun için öncelikle kodlaştırmayı tanımlanmıştır. $V = \{ b_1, b_2, \ldots, b_q \}$ alfabesi verilmiş olsun. Benzer olarak V alfabesindeki sözleri B ile ve V alfabesindeki bütün boş olmayan sözleri S(V) ile gösterelim. F, $S^l(U)$ dan S(V) ye bir fonksiyon olsun. O zaman, her $A \in S^l(U)$ için öyle $B \in S$ (V) bulunur ki bu durumda B=F(A) olur. Bu durumda B sözüne A mesajının kodu, A sözünden B koduna geçişe ise kodlaştırma denir. Ardından dekodlaştırma tanımlanmıştır. Verilmiş B koduna göre ilkel A mesajının (kodu B olan A mesajının) bulunmasına dekodlaştırma denir. Yani dekodlaştırma çıkıştaki bir mesajın düzeltilmesi ile elde edilen ilk mesaja geçiştir.

Alfabe kodlaştırması 4. başlığımızı oluşturmaktadır. U alfabesindeki harflerle B alfabesindeki sözler arasında aşağıdaki uygunluğun olduğunu varsayalım.

a_1 — B_1

a_2 — B_2

.....

a_r — B_r

Kodlaştırma teorisinde bu uygunluğa şema denir ve $\sum$ ile gösterilir. Bu yöntem alfabe kodlaştırmasını aşağıdaki gibi tanımlar.

$S^l(U) = S(U)$ kümesinden alınan her bir $A = a_{i1}, a_{i2} \ldots, a_{im}$ sözüne A ' nın kodu denilen $B = B_{i1}B_{i2} \ldots B_{in}$ sözü uygundur. Burada $B_1, B_2, \ldots, B_r$ sözlerine **basit kodlar** denir.

5. başlıkta dekodlaştırmanın tek değerliliği verilmiştir. U ve V alfabeleri için $\sum$ şemasıyla verilmiş alfabe kodlaştırmasını göz önüne alalım. $S'(U) = S(U)$ sağlandığını kabul edelim. Yani bu durumda, mesaj kaynağı U alfabesindeki bütün sözleri doğurur. Çok açıktır ki, alfabe kodlaştırması S(U) kümesinden S(V) kümesine geçişi sağlar. S(U) kümesinin böyle alfabe kodlaştırmasının görüntüsünü $S_{\sum}(V)$ ile gösterelim. Eğer S(U) kümesinin $S_{\sum}(V)$ görüntüsüne geçişi tek değerli ise o zaman

dekodlaştırma mümkündür. Alfabe kodlaştırmasının tek değerli olmadığı durumlarda dekodlaştırma mümkün değildir. O nedenle her bir gözönüne alınan alfabe kodlaştırmasının tek değerli olup olmadığının belirtilmesi çok önemlidir. Bu problemin çözümü için yöntem ya da kriter verilmelidir. Aksi halde sonsuz sayıdaki sözlerin incelenmesi gerekebilir. Bu durum ise, çözümü olmayan bir problemdir. Çözüme ulaşmak için prefix kodlardan yararlanırız. $B = B^{I} B^{II}$ ile yazılmış olsun. O zaman B^{I} sözüne B sözünün prefix'i (önek) , B" sözüne ise B sözünün sonu denir. B sözünün kendisi ile boş söz (Λ), B sözünün prefix'i ve sonu gibi alınır. $B = B\Lambda = B'B''$. Eğer istenilen $\forall$ i, j $(1 \le i, j \le r)$, i≠j için B_i sözü B_j sözünün prefix 'i değilse, o zaman $\sum$ şeması " prefix özelliğine sahiptir" denir. Eğer $\sum$ şeması prefix özelliğine sahip ise o zaman alfabe kodlaştırması tek değerlidir. $\sum$ ve $\tilde{\sum}$ şemaları ile belirtilen alfabe kodlaştırmasının, her ikisi aynı zamanda ya karşılıklı tek değerlidir, ya da karşılıklı tek değerli değildir. Her zaman yeterli koşul uygulanarak kodlaştırmanın tek değerli oluıp olmadığı ayarlanamaz. Bu amaçla dekodlaştırmanın tek değerlilik kriterinin verilmesi çok önemlidir.

Bu kriter 6. Başlık altında incelenmiştir. U alfabesindeki uzunluğu N' den büyük olmayan boş sözden farklı sözlerin oluşturduğu kümeyi $S^{N}(U)$ ile gösterildiğinde buradan, $S^{N}(U)$, $\sum_{i=1}^{N} r^{i}$ elemanlı sonlu sayılar kümesidir**.** Bu sayıyı sınırlandırmak için Markov Algoritması'ndan yararlanırız.

$$N_0 \le \left[\frac{(w+1)(L-r+2)}{2} \right]$$

Eşitsizliğini sağlayan öyle N_0 sayısı var ki , $\sum$ şemalı alfabe kodlaştırmasının karşılıklı tek değerlilik problemi, sonlu $S^{No}(U)$ kümesinin kodlaştırılması için benzer probleme indirgenir. A′ , A" $\in S^{No}(U)$ olduğundan tek değerli dekodlaştırma kriteri , $\sum$ şeması kullanılarak alfabe kodlaştırmasının tek değerli olup olmadığı U alfabesindeki uzunluğu N_0'dan fazla olmayan kelimeleri göz önüne alarak belirlememize olanak sağlayan basit bir algoritmayı doğurur. Böyle algoritmaların karmaşıklığı yaklaşık olarak r^{No} olarak hesaplanabilir. 7. başlıkta verdiğimiz örnekte r=5 *w*=2 ve L=16 olduğu durumdaki alfabe ile verilmiş ve uzunluğu 19' u aşmayan sözlerin sayısı en azından 5^{19} (r^{No}) olur ve tek değerliliği kontrol etmek için tek tek bu sözlerin incelenmesi gerekir. Bildiğimiz gibi, herhangi alfabe üzerinde verilebilen tüm sözlerin incelenmesi mümkün değildir. Görüldüğü gibi bu adımda

karşılıklı tek değerlilik kriteri biraz iyileşmesine karşılık, incelediğimiz örnekten de görüldüğü gibi bu kriterin de her zaman uygulanması iyi sonuç vermiyor, örneğin bu örnekte en azından 5^{19} sayıdaki sözlerin incelenmesi gerekmektedir. Kodlaştırma teorisinde bu tür alfabe kodlaştırmaları sık sık rastlanan problemlerdir. 8. başlıkta böyle problemleri daha basit şekilde çözebilmek için daha uygun farklı bir algoritma verilmiştir. Bu algoritma dekodlaştırmanın mümkün olup olmadığına karar vermek için gerekli ve yeterli derecede etkin olan bir algoritmadır. 2. bölümde incelediğimiz graf teorisinden bu başlıkta yararlanacağız. $\sum$ şemalı alfabe kodlaştırrmasının karşılıklı tek değerli olması için gerekli ve yeterli koşul (kriter) $\Gamma(\sum)$ grafının boş söze (Λ) uygun köşesinden geçen yönlendirilmiş kapalı döngünün olmamasıdır. Böylece göz önüne alınan alfabe kodlaştırmasının tek değerli olup olmaması, dolayısıyla dekodlaştırmanın mümkün olup olmaması graflar teorisinin elemanları uygulanarak oluşturulur. Bu durumu şu şekilde açıklayabiliriz: Her bir B_i elementer kodu için tüm basit olmayan $B_i=\beta' B_{i1}\ldots B_{iw}\beta''$ (1) ayrılışlarını göz önüne alalım. Burada β' ve β'' basit kodlardan farklı kodlardır.

Boş sözü Λ , uygun β' ve β'' kodlarının oluşturduğu kümeyi K_o olarak gösterelim.

K_o kümesini oluşturmak için aşağıdaki işlemler yapılır.

1- Her zaman için boş söz (Λ) içerir.

2- (1) yazılışında aynı zamanda hem başlangıçta hem de sonda bulunan β' ve β'' sözlerini içerir.

K_o kümesi ayarlandıktan sonra yapılacak adımlar şöyle sıralanabilir:

1- Göz önüne alınan alfabe kodlaştırması incelenerek her bir B_i elementer kodu için bütün (1) nolu ayrılışlar yazılır.

2- K_o elemanlarından yararlanarak $\Gamma(\sum)$ grafı çizilir.

3- Boş söze (Λ) uygun köşeden kapalı döngünün geçip geçmediği araştırılır.

Görüldüğü gibi bu algoritmanın uygulanması üç basit aşamadan oluşur ve her bir alfabe kodlaştırması için gerekli ve yeterlidir. β' , $\beta'' \in K_o$ olsun ve (1) ifadesindeki tüm ayrılışlar için β' , β'' kodlarına uygun gelen köşeleri bileştirdiğimizde oluşan grafı $\Gamma(\sum)$ ile

gösteririz. Eğer $\Gamma(\sum)$ grafında Λ(boş söz) üzerinden geçen bir döngü varsa o zaman $\sum$ şeması tek değerli değildir. $\Gamma(\sum)$ grafının çizilmesi algoritmanın yapılmasına bağlıdır. Eğer Λ (boş söz) üzerinden geçen bir döngü yoksa kodlaştırma karşılıklı tek değerlidir, yani dekodlaştırma mümkündür. Bu durum 3. bölümün 8. başlığı altında örneklerle detaylı olarak incelenmiştir.

4.bölümde kendi kendini düzeltebilen kodlar üzerine çalışılmıştır. { $A_1, A_2, \dots, A_s$ } kümesinden alınan $\alpha_1 \dots \alpha_m$ şeklinde olan A kelimeleri ℓ uzunluklu $\beta_1\beta_2 \dots \beta_\ell$ kelimeleri ile kodlanır ve kanalın çıkışında elde edilen $\beta_1\beta_2 \dots \beta_\ell$ 'ni ileten $\beta^l_1\, \beta^l_2 \dots \beta_\ell$ yardımıyla esas kelime olan $\alpha_1 \dots \alpha_m$ elde edilebilir mi? Böyle kodlara kendi kendini düzeltebilen kodlar denir. Eğer kelime kodu gönderilirken bir hata meydana gelirse, daha sonra $\alpha_i\ \alpha_i\ \alpha_i$ grubu içindeki bir sembol bozulur (değişir) ve tüm diğer semboller hatalı gönderilir. Kod uzunluğu $\ell = 3m$ olmadan önce, üçlü çözüm uygulanmak doğru değildir ve kendi kendini düzeltebilen kodlar p' den daha fazla hataya neden olur. Bu durumdan sonra orijinal mesajı tekil olarak yeniden yapmak her zaman mümkün olmayabilir. Kendi kendini düzeltebilen kodların doğru yapısı p = 1 durumunu ayrıntılarıyla inceleyen R.Hamming tarafından yapılan şu an kabul ettiğimiz yapıdır.
Daha sonra 2.başlık altında kendi kendini düzeltebilen kodlardan Hamming kod yapısı için algoritma tanımlanmıştır.
3.başlıkta Hamming kod yapısı ile dekokodlaştırma yapılmış ve örnek verilerek açıklanmıştır. Kod çözme, $\beta_1\beta_2 \dots \beta_\ell$ kodu içindeki $\alpha_1\ \alpha_2 \dots \alpha_m$ orijinal mesajının yapısından bilgi basamakları basamaklarından hangisinin yeterli olacağını seçmektir. m= 4 için kendi kendini düzelten kodu yapalım. Bu $2^4 \le \frac{2^l}{l+1}$ eşitsizliğinde ℓ'yi sağlayan en küçük sayı 7'dir. O zaman k = 3'tür. Hamming kod yapısına uygun olarak kontrol basamaklarının * ile işaretlendiği yerde Tablo 4.1'e uygun kendini düzelten kodlardan birini ele alınarak çözüm yapılmıştır. Kanal girişinde kodu 0110011 olarak girelim ve karışıklık sonucu S = 5. terim değişmiş olsun. Sonra çıkışta 0110111olurarak bulunsun. Bu durumda hatalı terimin numarasını hesaplayalım.

$$S'_1 = \beta'_1 + \beta'_3 + \beta'_5 + \beta'_7 = 0 + 1 + 1 + 1 = 1$$
$$S'_1 = \beta'_2 + \beta'_3 + \beta'_6 + \beta'_7 = 1 + 1 + 1 + 1 = 0$$
$$S'_1 = \beta'_4 + \beta'_5 + \beta'_6 + \beta'_7 = 0 + 1 + 1 + 1 = 1$$

Bu durumda $S' = 101 = 5$'tir. Hatalı terim bulunur ve $S' = S = 5$ olur. Sonuç olarak, geometrik özellikleri Hamming Kodu üzerine tanımlarız. ℓ-boyutlu birimi küp içinde bir

kod seti verilen engelleyici kaynağa rağmen kendi kendini düzelten kod ise ve eğer bu en büyük kuvveti ise bu maksimum demektir.

5.bölümde generator (üreteç) matrisler incelenmiştir; ardından 4.bölümde incelediğimiz Hamming kod yapısı ve Hamming hata bulma algoritması bu bölümde matris kullanılarak yapılmıştır. Bu bölümde kodların içindeki dizilerin sabit uzunluğuna 'n' diyeceğiz ve bu dizileri vektör veya 1xn matrisler olarak düşüneceğiz. Böylelikle vektörleri toplayacağız. Hesapladığımız bu kodlara "linear codes" denir. Bu kodlar ayrıca "group codes" olarak da bilinmektedir. Uzunluğu 'n' olan tüm ikili sistemlerin kümesi B_n ise, C, B_n'nin alt kümesidir. C kodu B_n'nin alt grubuysa "linear code"dur. İlk k satır ve sütunları kxk birim matris formunda ve her sütunu farklı matrise "generator matrix" (üreteç matris) denir. Üreteç matrisin satırlarını vektör veya kodun dizileri olarak sayarsak dizi kümesine 'S' denir. Mesajı göndermek istediğimizde, k uzunluğundaki mesaj dizisi kodlanmış dizinin ilk k biti olur. Çünkü G'nin ilk k sütunu olan I_k birim matris sadece orijinal dizide tekrar eder. Böylece kodlamış dizinin ilk k bitini alarak kodu kolayca çözebiliriz. Üretici matrisin satırlarının kümesi S = (s_1, s_2, s_3, …) ve mesaj kodu ω = (ω_1, ω_2, ω_3 …)'tir, böylece kodlanmış dizi $\omega_1 s_1 + \omega_2 s_2 + \omega_3 s_3 +$ … + $\omega_k s_k$ olur.S'deki dizilerin toplamı her ω_i ya 0 ya 1 olduğundan dolayı C'dedir. Çünkü C, S'ten üretilmiş bir gruptur. kodlanmış dizilerimiz ω_1, ω_2, ω_3, … ω_k … ω_n ve

$$G = \begin{bmatrix} 1 & 0 & 0 \ldots & 0 & A_{1,k+1} & A_{1,k+2} & \ldots & A_{1,n} \\ 0 & 1 & 0 \ldots & 0 & A_{2,k+1} & A_{2,k+2} & \ldots & A_{2,n} \\ 0 & 0 & 1 \ldots & . & A_{3,k+1} & A_{3,k+2} & \ldots & A_{3,n} \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & 0 & . & . & . & . \\ 0 & 0 & 0\ 0 & 1 & A_{k,k+1} & A_{k,k+2} & & A_{k,n} \end{bmatrix}$$

i > k için $\omega_i = \omega_1 A_{1,i} + \omega_2 A_{2,i} + \ldots \omega_k A_{k,i}$ olmalı ve n – k denkliğini sağlamalıdır. Sağlamadığı takdirde mesaj gönderilirken hata oluştuğu anlaşılır. Bu hata tekse düzeltmek için "coset leaders" kullanılır. İlk koset C'nin kendisidir. Diğer koseti, B_n 'nin uygun ağırlıktaki ve C'de bulunmayan elemanlarından seçeriz. Tekrar B_n'den uygun ağırlıktaki ve diğer kosetlerde de olmayan eleman seçeriz. Bu sistemde devam ettiğimizde B_n'nin kosetlere bölünmüş tablosunu elde ederiz. Ağırlığı az olan elemanlar önce listelenir. İlk sütundaki elamanlara "coset leaders" denir. Hata belirlemenin daha

kolay bir yolu da şu şekildedir: n uzunluğundaki v ve ω vektörlerimiz (veya dizi) olsun. Eğer nokta çarpımları v. ω = 0 ise v vektörü ω vektörünün "ortogonalidir". $C^{\perp}$ ile gösterilen C'nin "dual code"u C'nin bütün dizilerinin ortogonali olan B_n dizilerinin kümesidir. Bu $C^{\perp,}$ nin B'nin alt grubu olduğunu gösterir. Eğer C kodu [n,k]-kod ise k dizisinden üretilmiştir, o halde $C^{\perp}$ de n-k dizisinden elde edilmiş olur. Bu yöntem sendrom içeren satırı ve gönderilmiş mesajı bulmakta daha hızlıdır. Çünkü gönderilmiş mesajın transpozuyla $G^{\perp}$ 'yi sadece çarpıyoruz. Bu mesajlar için baş kosetler hatadır ve gönderilmiş mesajı içeren sütunun ilk satırı olan C'nin elemanı düzeltilmiş mesajdır.

5. bölümün ilk başlığında Hamming kod matrisleri üzerinde çalışılmıştır. Hamming kod matrisleri ağırlığı 1 olmayan kosetlerdeki problemlerin çözümü için kullanılır. Hamming matrisin satırlarından üretilen kodlara "Hamming code" denir. Hamming matrisleri üzerinde çalışmak için iki dizi arasındaki uzaklık ve ağırlıkları arasındaki ilişki bilinmelidir. Dizilerimiz c ve $c^{|}$ ise ağrılıkları arasındaki ilişki; $t (c + c^{|}) \leq \omega t (c) + \omega t (c^{|})$ dir. Aynı uzunluktaki $c^{|}$ ve c dizilerinin arasındaki uzaklık ve ya Hamming uzaklığı birinde 1 diğerinde 0 rakamının olduğu dizilerdeki uygun bitlerinin sayısıdır. Uzaklık fonksiyonunu $\delta (c, c^{|})$ ile gösteriyoruz. Rastgele seçilmemiş bir dizi gönderilirken iki dizi arasındaki uzaklık artıkça hata sayısı artar. δ ile gösterilen uzaklık fonksiyonlarının temel özellikleri şunlardır: c ve $c^{|}$, $c^{||}$ dizileri için;

a) Ancak ve ancak $c = c^{|}$ ise $\delta (c, c^{|}) = 0$

b) $\delta (c, c^{|}) = \delta (c^{|}, c)$

c) $\delta (c, c^{|}) \leq \delta (c, c^{|}) + \delta (c^{|}, c^{||})$

Koddaki herhangi iki dizi arasındaki minimum uzaklığı bilmek önemlidir. C kodsa, C kodundaki her hangi iki dizi arasındaki en kısa uzunluğa C kodunun minimum uzaklığı denir. D (C) ile gösterilir. C kodu için;

a) k kadar hata bulmak için D (C) = k + 1 olmalıdır.
b) k kadar hata düzeltebilmek için D (C) = 2k + 1 olmalıdır.

Bu bölümün geri kalanında diğer kodları kısaca inceleyeceğiz. Bunların ilki Golay koddur. Hamming kodları, 1950'de Hamming ve 1949'da Golay tarafından bağımsız olarak bulunmuştur. Bu kodlar Golay in 1949'daki makalesindedir. Bu kodlara aynı zamanda Hamming kodlarıda denilmektedir. Bu (23, 12, 7) modelinde Golay tarafından yayımlanmıştır. Bunun anlamı C'deki dizilerinin aralarındaki minimum uzaklığı 7 olan (23, 12) üretici matrisidir. Bu Golay kod $G = [I_{11} \mid A]$ dan üretilir. Golay üretici kodunun tekrar düzenlenip parity bit'i eklemesiyle oluşur. Bu matrisin

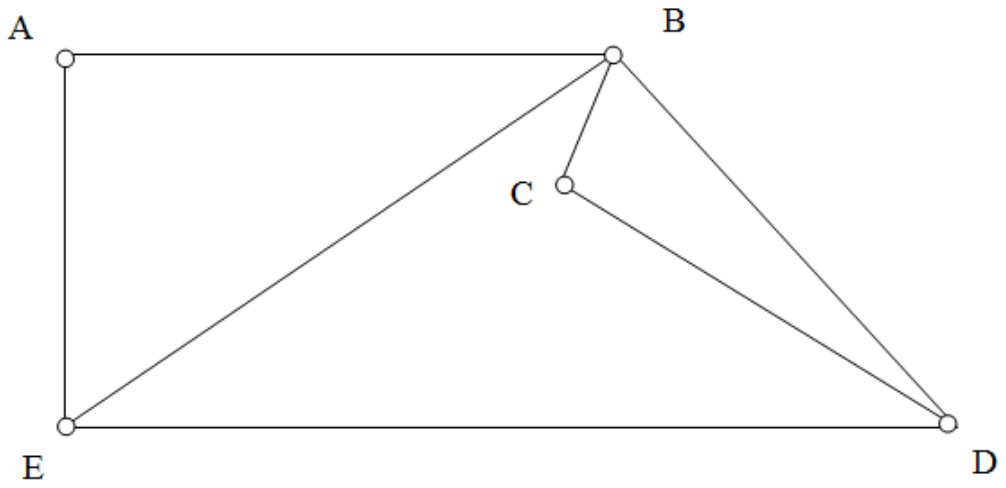
simetrisi çalışmamızı kolaylaştırıyor. G = [A │ I_{12}] 'yi görmek daha kolay, çünkü A = A^t'dır. Golay kodu (4096, 244, 8) code'u içeren çeşitli kodlar oluşturur. (4096, 244, 8) kodu Voyager Uzay gemisinde Jüpiter, Uranüs ve Neptün'ün görüntüleri göndermek için kullanılmıştır. n uzunluklu s dizisi için, $\bar{s}$ = s + 1 alalım, 1 bütün 1'leri içeren n uzunluğundaki dizidir. S dizilerinin kümesi verilsin, Plot (S) = { ss: $s \in S$ } $\cup$ { $s\bar{s}$: $s \in S$ }'dir. S = { 0000, 0011, 1100, 1111 } kümesi verilsin; S_1 = Plot (S), S_2 = Plot (S_1), S_3 = Plot (S_2), S_n = Plot (S_{n-1})'dir. Plot M Plotkin tarafından yaratılmış bir yapıdır. S_1 , S_2 , S_3 , … kümelerinden üretilen kodlar Reed – Muller kodları denir. S_3 kümesi (64, 32, 16) kodudur. Bu kod Marriner 9 uzay aracından gönderilen görüntülerin hatalarının düzeltilmesinde kullanılmıştır. Özel olarak Reed – Muller matrislerine Hadamard matrisi denir. Bu matrislerin $H_nH_n^t$ = nI özelliği vardır. I birim matrisidir. Bu matrislerden üretilen kodlara da 'Hadamard code' denir.

BÖLÜM 2
GRAF TEORİSİNİN TEMEL TERİMLERİ

2.1. Giriş

Noktalar ve bunları birbirine bağlayan kenar adı verdiğimiz eğri parçalarından oluşan şekle **graf (çizge)** denir.

Daha geniş olarak matematiksel tanımını şu şekilde yapabiliriz. Öncelikle G adını verdiğimiz grafımız şu şekilde olsun:



Şekil2.1. Graf

Bu şekildeki grafta A, B, C, D, E noktalarına **grafın noktaları ve ya düğümleri** denir. G grafının noktalar kümesi V(G) olarak gösterilir. Yani:

V(G) = {A,B,C,D,E} olarak gösterilir.

İki nokta arasındaki çizgilere **kenar** denir. Her kenarı bu çizgeye göre iki noktalı bir küme olarak gösterebiliriz. Örneğimiz olan G grafının kenarları şunlardır:

E(G) = { {A,B} , {A,E} , {B,C} , {B,D} , {B,E} , {C,D}, {D,E}}

G' nin kenarlar kümesi matematikte E(G) olarak gösterilir.

NOT Bir graf bir diyagram aracılığıyla temsil edilebilir. Fakat graf ile onu temsil eden diyagram aynı değildir. Şekil 2.1. kendi başına bir graf değildir, sadece bir grafın gösterimidir. Çünkü graf bir fonksiyon ile birlikte iki kümeden oluşur.

Tanım 2.1: Eğer grafın bütün kenarları yöne sahipse grafa **yönlü graf**, aksi halde **yönü olmayan graf** denir. Eğer bir grafta hem yönlü hem de yönü olmayan kenarlar varsa grafa **karışık graf** denir.

Bir kenar her iki ucunda da bir düğüm olacak şekilde tanımlandığından graftaki tüm kenarların uç noktalarını bir düğüm ile ilişkilendirmek gerekir. Bu nedenle her bir e kenarı için $\{V_1,V_2\}$ kümesi tanımlarız. Bunun anlamı e kenarının V_1 ve V_2 düğümlerini bağlandığıdır. $V_1=V_2$ olabilir.

$\{V_1,V_2\}$ kümesi J(e) ile gösterir ve düğümler kümesinin bir alt kümesidir.

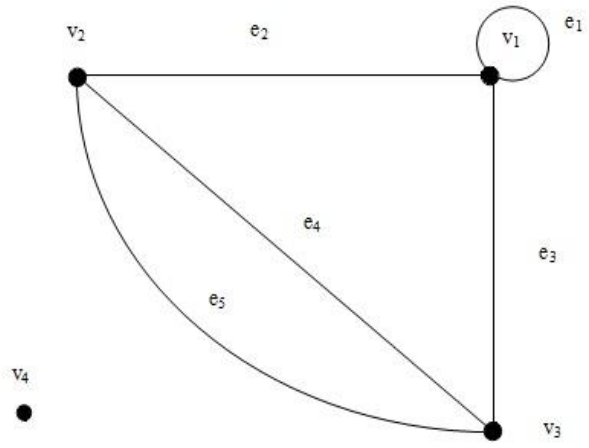
Bir yönü olmayan G grafı:

(i) Boş olmayan sonlu V düğümler kümesi,

(ii) Sonlu E kenarlar kümesi,

(iii) Her bir e kenarı için J(e), V nin bir veya iki elemanlı alt kümesi olan J: E → P(V) fonksiyonundan oluşur.

G:



Şekil 2.2. Graf

G grafının düğüm kümesi V= $\{V_1,V_2V_3,V_4\}$

G grafının kenar kümesi E= $\{e_1,e_2,e_3,e_4,e_5\}$

J : E → P(V)

$e_1 \rightarrow \{V_1\}$

$e_2 \rightarrow \{V_1,V_2\}$

$e_3 \rightarrow \{V_1,V_3\}$

$e_4 \rightarrow \{V_2,V_3\}$

$e_5 \rightarrow \{V_2,V_3\}$

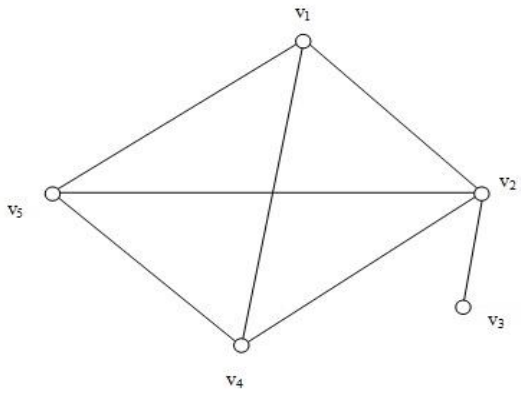
Tanım 2.2: Matematiksel anlamda bir graf V kümesiyle V'nin iki elemanlı alt kümelerinden oluşan bir E kümesinden meydana gelir.

V ile bütün kenarlardan oluşan E kümesinin ayrı ayrı sonlu veya sonsuz olmasına izin verilebilir. Buradaki çalışmada her iki kümenin sonlu olduğunu kabul edip bunun üzerinde işlem yapacağız.

Bir grafta V kümesinin boş olması çok anlamsızdır (Çünkü Tanım 2.1'den V nin iki elemanlı alt kümelerinden oluşan bir E kümesi vardır). Buna karşılık E kümesi boş olabilir. İki nokta arasında en az bir kenar varsa bunlara **komşu noktalar** denir.

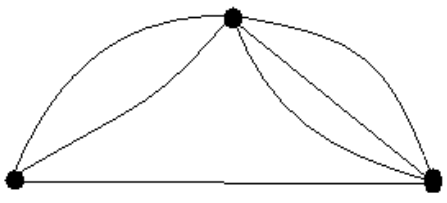
Eğer bir noktayı kendisine bağlayan kenar varsa bu nokta kendisiyle komşu olur. Ama genel olarak her nokta kendisine komşudur diyemeyiz. Bir noktayı kendisine bağlayan kenara şeklinden dolayı **ilmek** adını alır.

İlmeği ve iki noktası arasında en çok bir kenarı bulunan graflara **basit graflar** denir.



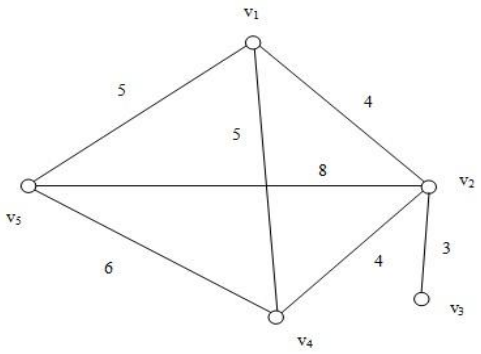
Şekil 2.3 Basit Graf

İki noktası arasında birden çok kenar bulunan graflara, **multigraf (çoklu çizge)** denir.



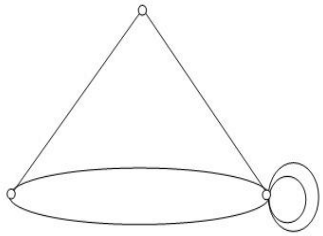
Şekil2.4 Multigraf

Bir grafta her bir kenara bir reel sayı atanmışsa bu grafa **ağırlıklı graf veya ağ** denir.



Şekil 2.5 Ağırlıklı Graf

Hem ilmek, hem de çoklu kenarlara sahip yönü olmayan grafa **pseudo graf** denir.



Şekil 2.6 Pseuodo Graf

Bir noktadaki kenarların sayısına (yerel) **derece** denir. Eğer V, bir G grafın bir noktası ise V deki derece L(v) ile gösterilir. L_{max} ve L_{min} ile bir G grafın derecelerinin içinde en büyüğü ve en küçüğü gösterilir.

İki noktası arasında birden çok kenarı bulunan graflarda noktaları saymak kolaydır. Ancak kenarları tek tek saymak zor olabilir. Bu durumda şu uygulamayı yapabiliriz. Her kenarın iki ucu vardır ve bu uçların her biri bir noktaya bağlanmıştır. Buna göre her noktasındaki kenar sayısını toplarsak bütün kenarları ikişer kez saymış oluruz. Yani bütün noktalardaki derecelerin toplamı, kenar sayısının iki katına eşittir. Bunu şöyle ifade edebiliriz. Bir G çizgesinde n nokta ve e kenar bulunsun.

n noktalar kümesi,

$V = \{v_1, v_2, \ldots, v_n\}$ ise

$$2e = \sum_{i=1}^{n} L(V)$$

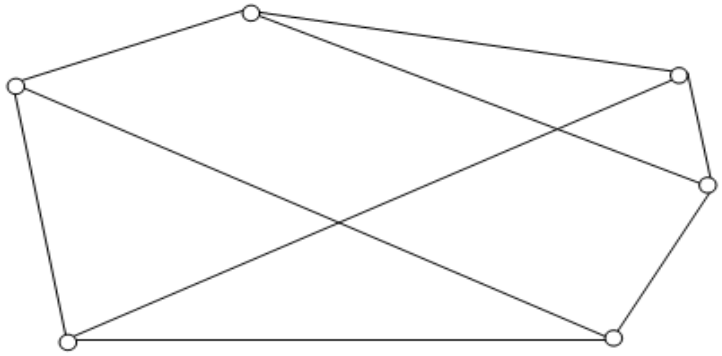
olur. (2.1)

Bütün noktaların dereceleri aynı olan graflara **düzgün (regüler) graflar** denir. Lokal dereceleri hep r ise r-li düzgün graf denir. |V| = n olan bir r-li düzgün grafta;

$|E| = e = \frac{1}{2}nr$ (2.2)

olacağı açıktır.

Bir örnek vermemiz gerekirse:



Şekil 2.7. 3-lü düzgün graf.

Bu grafta:

$V = 6, r = 3 \quad e = \frac{1}{2}nr \Rightarrow e = \frac{1}{2}.6.3 \Rightarrow e = 9$ (2.3) olur.

Bir grafta derecelerin toplamının kenarların sayısının iki katı olduğunu söylemiştik. O halde $L(v_1) + L(v_2) + \ldots + L(v_n)$ toplamı her zaman çift bir sayıdır. Tek sayıda tek sayının

toplamı yine tek sayı olduğundan yukarıdaki toplamda yer alan tek sayıların çift sayıda olması gerektiği anlaşılır. Yani bir çizgede derecesi tek olan noktaların sayısı çift olmak zorundadır. Bunu teorem olarak da yazabiliriz.

Teorem 2. 1: Her grafta derecesi tek olan noktaların sayısı çifttir.

İspat: Sonlu bir G = (V,E) grafında A grafın noktasını, L(A) o noktanın derecesini göstersin. Bu durumda:

$$\sum_{A \epsilon V} L(A) = 2\,|E| \qquad (2.4)$$

eşitliği geçerlidir.

V_0, derecesi çift olan V_1, derecesi tek olan noktalar kümesi olsun. Biraz önce bahsettiğimiz eşitlikten;

$$\sum_{A\in V_0} L(A) + \sum_{A\in V_1} L(A) = 2|E|$$

dir. O halde: (2. 4a)

$$\sum_{A\in V_0} L(A) + \sum_{A\in V_1} L(A)$$

çift bir sayıdır. (2. 4b)

Sol taraftaki toplamda L(A) sayılarının her biri çift ve toplamın sonucu da çift olduğundan,

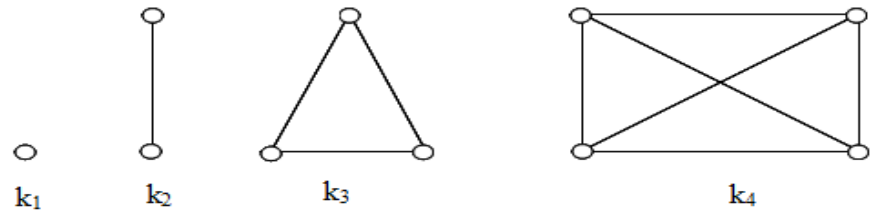
$\sum_{A\in V_0} L(A)$ sayısı da çift olmalıdır. Ama buradaki L(A)ların her biri tek sayıdadır .

Dolayısıyla L(A) ların toplamının çift olması için çift sayıda L(A) toplamalıyız. O halde V_1 çift sayı olmalıdır. Yani derecesi tek olan noktaların sayısı çifttir.

Bir grafta tek başına kalmış noktalar bulanabilir. Böyle noktalara **izole noktalar (küsmüş noktalar)** denir.

Bütün noktaları izole olan bir grafa **boş graf veya sıfır grafı** denir. Yani bir grafın boş olması kenarlar kümesi E' nin boş olması anlamına gelir.

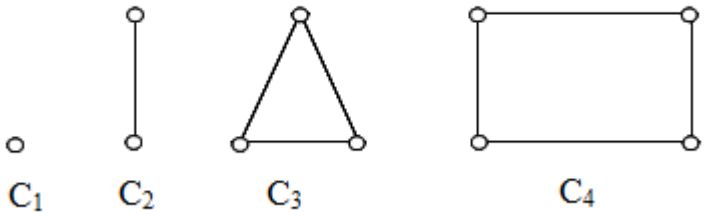
Herhangi iki noktası arasında en fazla bir kenar bulunan ve hiç ilmeği olmayan graflara tam graf denilmektedir. Nokta sayısı n olan bir tam graf K_n ile gösterilir.



Şekil 2.8. K_n tam grafı.

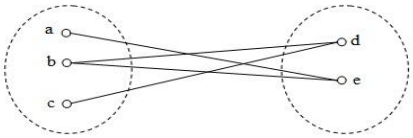
K_n grafı doğal olarak (n-l)-li düzgün graftır. $\frac{1}{2}$ n(n-l) kenara sahiptir.

C_n grafı, her noktaya tam iki kenar değen tek parça n noktalı graflardır. Bunlara **döngü** denir.



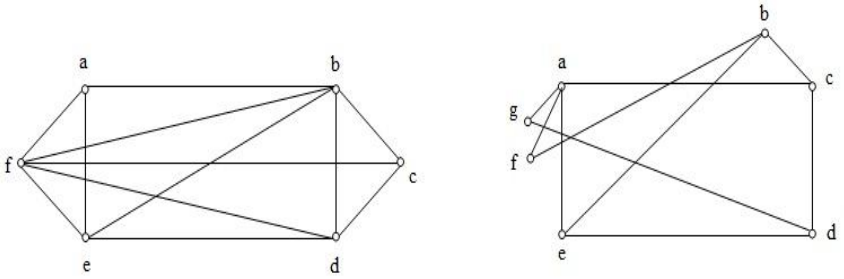
Şekil 2.9. C_n grafı.

Düğüm kümesi, tüm kenarların V_1'in bir düğümünü V_2'nin bir düğümüne bağlandığı $\{V_1,V_2\}$ şeklinde bir bölmelemeye sahip olan grafa **iki parçalı(bipartite) graf** denir.
G İki parçalı graf ise,



Şekil 2.10. İki parçalı graf

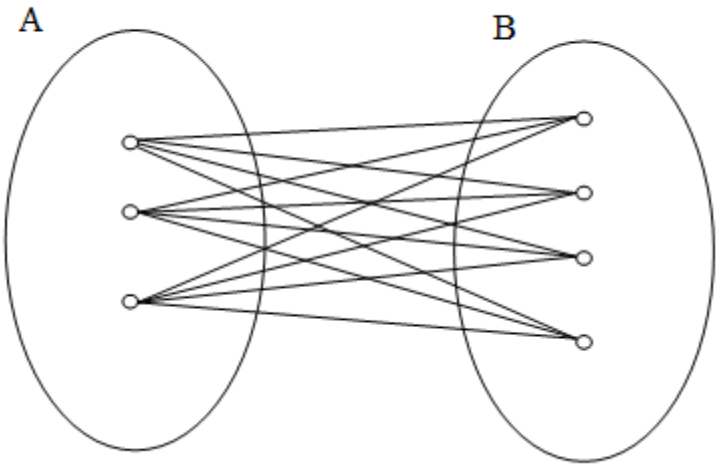
Mevcut küme içerisindeki düğümler birbirlerine herhangi bir kenar ile bağlanmamalıdır.



İki parçalı graf değil İki parçalı graf

Şekil2.11. Graflar

$K_{n,m}$ grafı, noktaların n ve m elemanlı olmak üzere iki A ve B kümesine ayrılmış, A'daki her noktanın B' deki her noktaya bağlandığı başka da kenarı olmayan graflardır. $K_{n,m}$' ye **iki parça ya da iki kümeli tam graf** denir. Bu graflarda nokta sayısı n+m. Kenar sayısı nm'dir.



Şekil 2.12. İki parçalı tam graf

Tanım 2.3: $V^{|} \subseteq V$ (düğümler kümesi), $E^{|} \subseteq E$ (kenarlar kümesi) ve $\delta_G{}^{|}(e) = \delta_G(e)$ ise $G^{|}=(V^{|},E^{|})$ grafına G=(V,E) grafının **alt graf**ı denir ve $G^{|} \leq G$ şeklinde gösterilir.

$G^{|}$ grafının tüm e kenarları için $\delta_G{}^{|}(e) = \delta_G(e)$ durumu $G^{|}$ alt grafının kenarlarının G de olduğu gibi aynı düğümleri bağlaması gerektiği anlamına gelir.

Bir G grafından bir V_i düğümü çıkartıldığında V_i hariç G'nin tüm düğümleri ve V_i ile bağlı olmayan tüm kenarlarından oluşan $G-V_i$ alt grafı elde edilir.

$G-V_i$, G'nin V_i düğümünü içermeyen maksimal alt grafıdır. Öte yandan G den bir x_j kenarı çıkartıldığında G'nin x_j haricindeki tüm kenarını içeren bir $G-x_j$ alt grafı elde edilir.

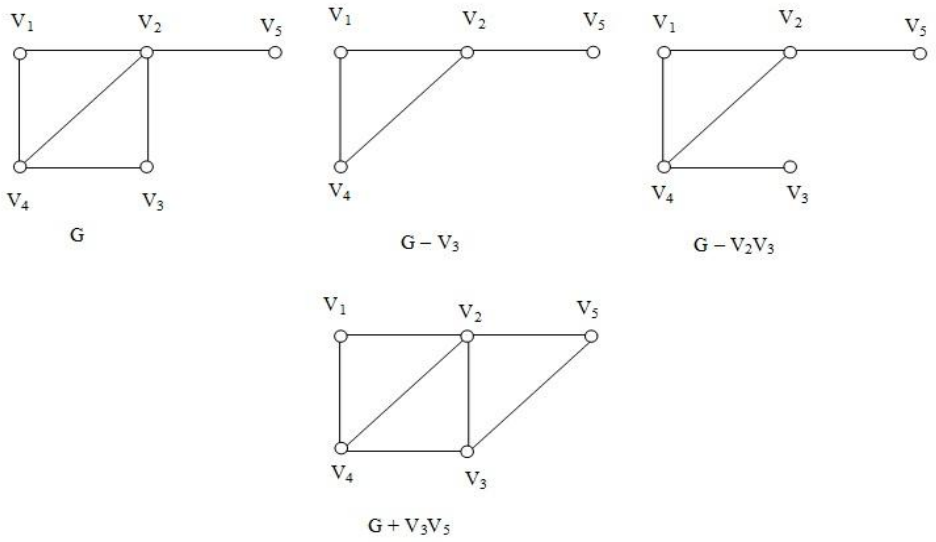
$G- x_j$, G nin x_j kenarını içermeyen maksimal alt grafıdır.

G_1 , G grafının bir alt grafı ise G ye G_1 grafının süper grafı denir.

Eğer V_i ve V_j G de komşu değilse V_iV_j kenarının eklenmesi V_iV_j kenarını içeren G'nin en küçük süper grafı olması ile sonuçlanır.

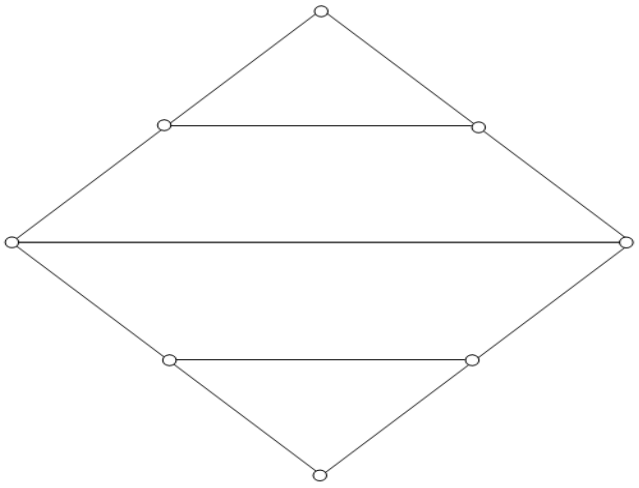
Yani, bu durumda G grafı, $G+V_iV_j$ grafının alt grafı olacaktır.

Bu durumlar aşağıda gösterilmiştir.



Şekil 2.13.

2.2. Tek Hamlede Çizilebilen Graflar



Şekil 2.14. Tek dereceli graf.

Bu şekildeki bir grafı elimizi en fazla iki kez kaldırarak yani en fazla üç çizimde çizebilir miyiz?

Öncelikle çizimde her noktaya 3 kenar değiyor. Yani her noktanın derecesi 3 tür. Şimdi çizimin ortasında olduğumuzu düşünelim. Bir noktaya doğru ilerliyoruz. O noktaya ulaştık. Ve o noktadan çıkıyoruz; demek ki çizimin ortasından geçtiğimiz her noktaya 2 derece kazandırırız: o noktaya ulaştığımızda ve o noktadan uzaklaştığımızda. Öte yandan çizime başladığımız ve çizimi bitirdiğimiz noktalara yalnızca 1 derece kazandırırız.

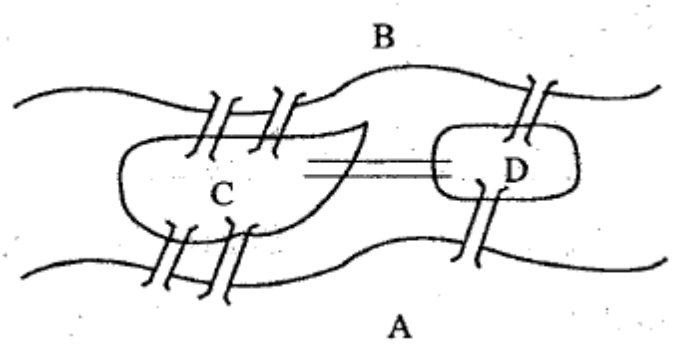
O halde hiç elimizi kaldırmazsak, elde ettiğimiz çizimin noktalarının en fazla ikisi dışında hepsinin derecesi çift olmak zorundadır.

Örneğimize dönersek en fazla 3 çizim olduğundan en fazla 6 noktasının derecesi tek olabilir. Oysa çizimimizde 8 tane noktanın derecesi tek olduğundan bu imkansızdır. Yani bu graf en fazla 3 hamlede çizilemez.

Aslında tek çizimde, elimizi kaldırmadan çizebileceğimiz grafların tek dereceli noktası ya hiç olmaz ya da sadece iki tane olabilir; eğer çizimi başladığımız yerde bitiriyorsak her nokta çift dereceli olmalı, eğer çizimi başladığımız yerde bitirmiyorsak sadece iki noktanın derecesi tek olabilir; çizime başladığımız ve çizimi bitirdiğimiz nokta.

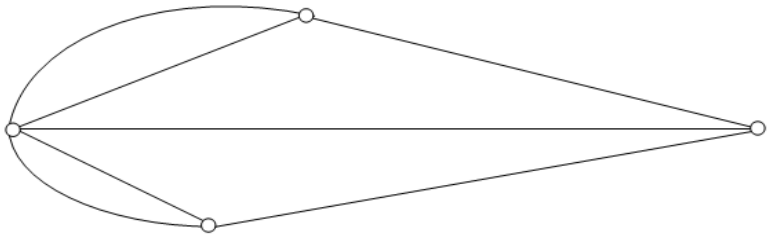
2.3. Euler Döngüsü

Graf kuramının bilinen en eski sorusu "Könisberg köprü problemidir. Burada Könisberg'deki Pregel nehrinin ve karalar arasında geçişi sağlayan 7 köprünün planını görüyorsunuz. Bu 7 köprünün her birinden sadece bir kez geçecek yolculuk mümkün müdür?



Şekil 2.15. Könisberg Köprüsü

Euler 1736' da bunun mümkün olmadığını göstermiştir. Bunu şu şekilde açıklayabiliriz. Nehrin iki yakasını ve adacıkları dört noktayla, 7 köprüyü de bu noktalar arasına koyacağımız kenarlarla gösterelim:

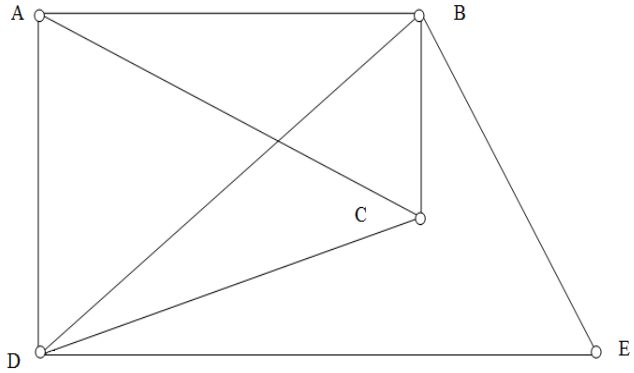


Şekil 2.16. Köprünün grafla gösterimi.

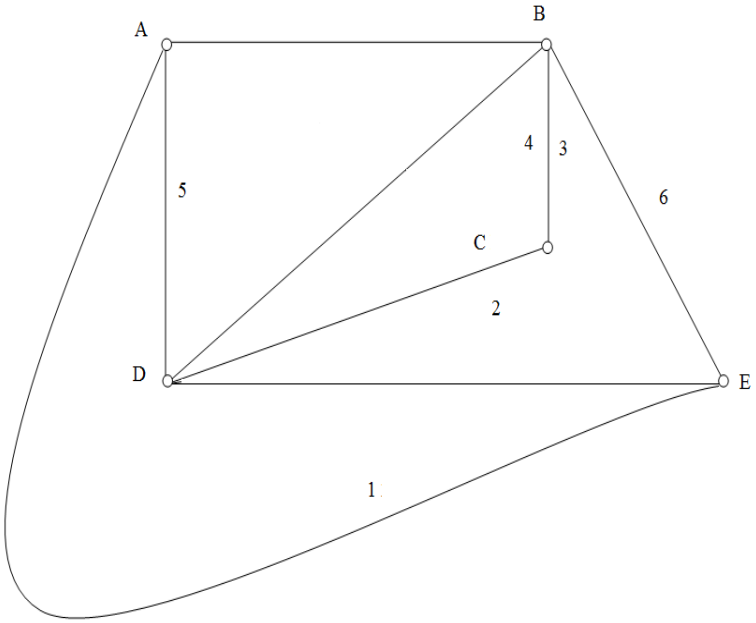
Tanım 2.4: Her kenardan tam bir kez geçen başladığı noktaya geri dönen yolculuklara Euler döngüsü denir. Euler döngüsü olan grafların tek parça ve her noktasının çift dereceli olması gerekir.

Teorem 2.2: Bir G grafının Euler grafı olması için gerek ve yeter koşul G grafındaki tüm noktaların derecesinin çift olmasıdır.

Bu teoremden yararlanılarak Şekil 2.16.' ya baktığımızda bu grafın Euler grafı olmadığı açıktır.



1. Graf



2. Graf

Şekil 2.17. Düzlemsel graf.

Kenarların sadece grafın noktalarında kesişecek biçimde düzleme çizilebilen graflara düzlemsel graf denir. Örneğin 1. graf düzlemsel değildir. Çünkü AC ve BE kenar çizgileri kesişmiştir. 2. graf düzlemseldir. Yani bir düzleme kenarları kesişmeyecek şekilde çizilmiştir. 2. grafta görüldüğü şekilde düzlemsel graf düzlemi bölgelere ayırır. Bu grafta düzlemi 6 parçaya ayırmıştır(grafin dışında kalan parçayı da sayıyoruz).

Teorem 2.3 (Euler Formülü): Tek parça düzlemsel bir grafın bölge sayısı b, kenar sayısı k, nokta sayısı n ise b-k+n=2 eşitliği geçerlidir.

İspat: İspatı b üzerine tümevarımla yapacağız.

Eğer b=l ise, o zaman grafta hiçbir döngü yok demektir, yani graf bir ağaçtır. Buradan k=n-1dir. Bunu yerine koyarsak:

b-k+n = 1-(n-l) + n = 2 (2.5)

olur ve eşitlik doğrudur.

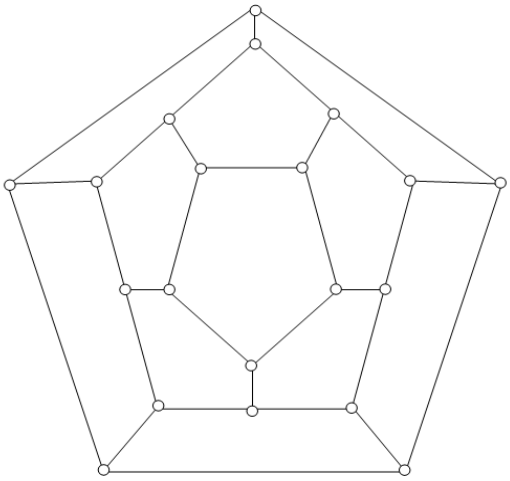
Şimdi b>l olsun. Grafa G adım verelim. AB de grafın bir döngüsünün bir kenarı olsun. AB kenarını kaldıralım. Geriye kalan grafa G_1 adını verelim. G_1 grafının bölge, kenar ve nokta sayılan sırasıyla b_1, k_1, n_1 olsun. AB kenarı G nin iki bölgesine de ortaktır. Dolayısıyla AB kenarını kaldırdığımızda b_1=b-l, k_1=k-l ve n_1= n eşitlikleri geçerlidir. M dolayısıyla tümevarım varsayımına göre Euler formülü G_1 grafi için doğrudur. Yani;

$b_1-k_1 + n_1=2$ dir.

b-k +n = (b_1+l) -(k_1+l) + n_1 = b_1-k_1 + n_1 = 2 bulunur. O halde b-k+n =2 doğrudur.

2.4. Hamilton Döngüsü

Bir G grafi verildiğinde her noktadan yalnız bir kez geçmek şartı ile kapalı bir yol oluşturabilen graflara Hamilton grafi denir. Aşağıdaki graf Hamilton grafidır.



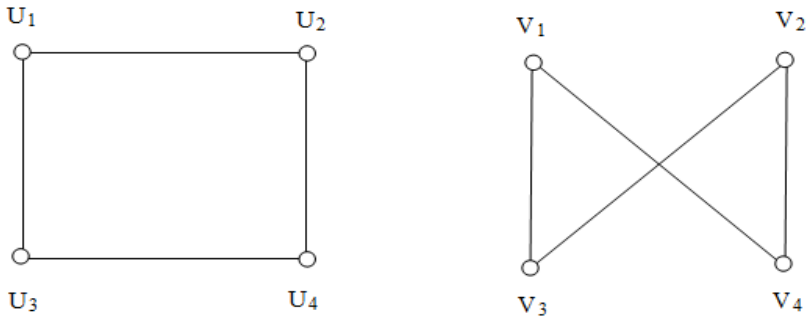
Şekil 2.18. Hamilton grafı

2.5. Eş Yapılı (İzomorfik) Graflar

İki Graf verildiğinde bu iki graf ta kenarlar ve konumlar arasındaki ilişki aynı olabilir. Bu durumda benzer özellikte iki farklı geometrik şekiller ortaya çıkabilir. Aynı durumu yansıtan, başka bir deyişle aynı yapıya sahip graflara eş yapılı graflar denir. Başka bir deyişle bunlara izomorfik graflar denir.

İki grafın eş yapılı olabilmesi için;

- Kenar sayıları aynı olmalıdır.
- Nokta sayıları aynı olmalıdır.
- Nokta dereceleri aynı olmalıdır
- Noktalar arasındaki ilişkiyi gösteren matrisler aynı olmalıdır. Bu matrislerdeki benzerlik satır ve sütunlardaki yer değişikliği ile de sağlanabilir.



Şekil 2.19. İzomorfik graflar.

Yukarıda görülen bu iki graf izomorfiktir. Kenar sayıları, nokta sayıları, nokta dereceleri aynıdır. Noktalar arasındaki ilişkiyi gösteren matrislerde aynıdır. Şöyle ki

	u_1	u_2	u_3	u_4
u_1	0	1	1	0
u_2	1	0	0	1
u_3	1	0	0	1
u_4	0	1	1	0

	V_1	V_2	V_3	V_4
V_1	0	0	1	1
V_2	0	0	1	1
V_3	1	1	0	0
V_4	1	1	0	0

Tablo 2.1. Şekil 2.19'daki grafların matrisi

Matrislerinde u_2 ve u_4 satır ve sütunları yer değiştirdiğinde elde edilen matrisle ikinci matriste eşit olur. Böylece iki grafın izomorfik olduğu kolayca görülür.

2.6. Komşuluk ve Incidence Matrisleri

Tanım 2.5:G, düğüm kümesi $\{V_1,V_2,\ldots,V_n\}$ olan bir graf olsun. G nin komşuluk matrisi; a_{ij} , V_i ve V_j yi bağlayan ayrı kenarların sayısı olmak üzere nxn lik $A=(a_{ij})$ matrisidir.

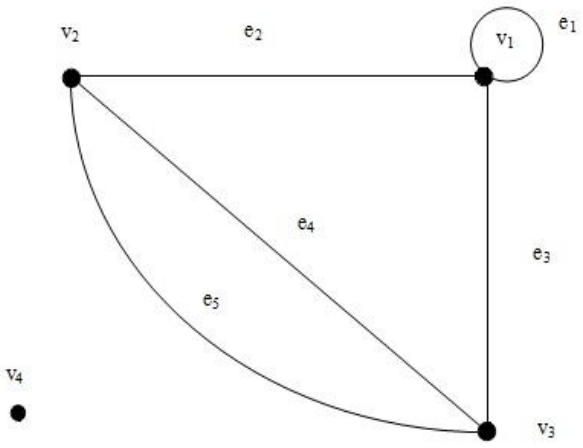
Komşu matrisi, V_i ve V_j yi bağlayan kenarların sayısı V_j ve V_i yi bağlayan kenarın sayısıyla aynı olduğundan simetrik olmalıdır.

V_i düğümünün derecesi komşuluk matrisinden belirlenebilir.

V_i de bir döngü yoksa bu düğümün derecesi matrisin j. sütununda ki değerlerin toplamıdır.

Her bir döngü dereceyi iki kez etkilediğinden i. sütunundaki değerleri toplarken a_{ii} diyagonel elemanının iki katı alınır.

Örnek:



Şekil2.20. G Grafı

G grafının komşuluk matrisi:

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{cccc} V_1 & V_2 & V_3 & V_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Grafın iki özelliği matrise bakılarak hemen görülebilir. Köşegene bakıldığında bir tek döngünün var olduğu görülür.

(v_1 den v_1 e)

İkincisi ise son satır veya sütundaki 0'lar v_4'un izole edilmişliğini gösterir.

Düğümlerin dereceleri matristen hesaplanabilir.

$p(v_1) = 2.1 + 1 + 1 = 4$

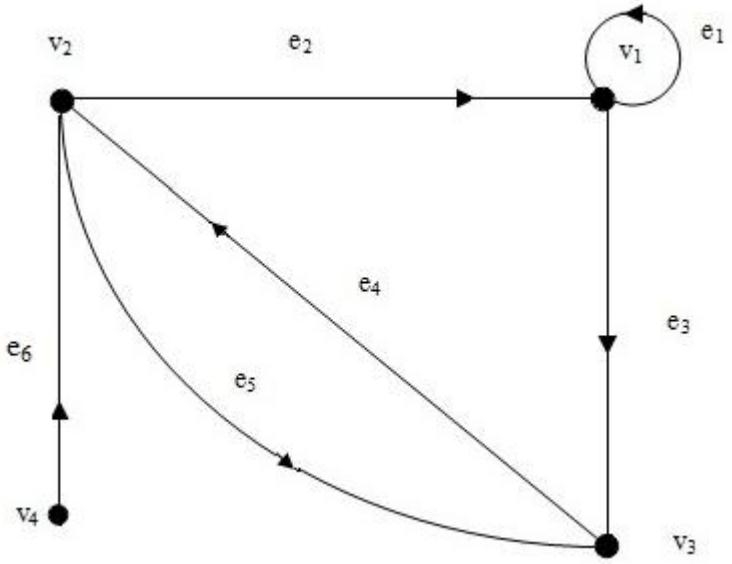
$p(v_2) = 1 + 2 = 3$

$p(v_3) = 1 + 2 = 3$

$p(v_4) = 0$

n kenar bir yönlü grafın komşuluk matrisi de nxn lik bir matristir.

Eğer i. düğümden j. düğüme bir kenar varsa $a_{ij} = 1$ aksi halde $a_{ij} = 0$ dır.



Şekil 2. 21. Graf

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{cccc} V_1 & V_2 & V_3 & V_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Hem yönlü hem de yönü olmayan graflar için diğer önemli matris incidence matrisidir.

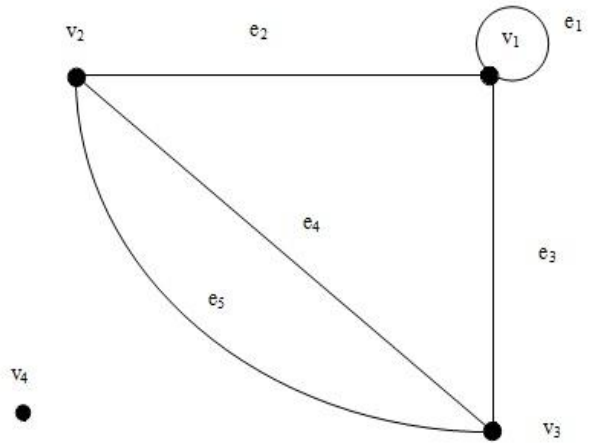
Komşuluk matrisinin tersine incidence matrisinde çoklu kenarlar gösterilebilir.

$V = \{ 1, 2, \dots, n \}$ ve $E = \{ e_1, e_2, \dots, e_m \}$ olmak üzere $G = (V, E)$ grafı verilsin.

G grafının incidence matrisi, nxm boyutlu olan ve her bir satırın bir düğüme ve her bir sütunun bir kenara karşılık geldiği bir $B = (b_{ik})$ matrisidir öyle ki eğer e_k, i ve j. düğümler arasındaki bir kenar ise k. sütunun elemanlarından $b_{ik} = b_{jk} = 1$, diğeri 0 dır.

Döngü olan kenarın sütununda sadece bir tek 1 vardır.

Örnek:



Şekil 2.22.Graf

Yukarıda verilen grafın incidence matrisi aşağıdaki gibidir.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

e_1, 1. düğüm kendisine bağlamıştır. Dolayısıyla 1. Sütunun elemanlarından $b_{11} = 1$, diğerleri 0'dır.

e_2, 1. düğüm ile 2. düğüm arasındaki kenardır. Dolayısıyla 2. Sütunun elemanlarından $b_{12} = b_{22} = 1$, diğeri 0'dır.

e_3, 1. düğüm ile 3. düğüm arasındaki kenardır. 0'dır. Dolayısıyla 3. Sütunun elemanlarından $b_{13} = b_{33} = 1$, diğeri 0'dır.

e_4, 2. düğüm ile 3. düğüm arasındaki kenardır. Dolayısıyla 4. Sütunun elemanlarından $b_{24} = b_{34} = 1$, diğeri 0'dır.

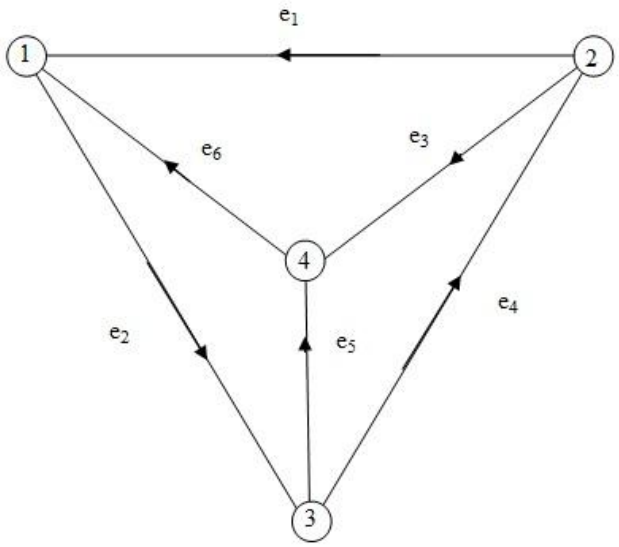
e_5, 2. ile 3. düğüm arasındaki kenardır. Dolayısıyla 5. Sütunun elemanlarından $b_{25} = b_{46} = 1$, diğerleri 0'dır.

Son olarak e_6, 2. ile 4. düğüm arasındaki kenardır. Dolayısıyla 6. sütunun elemanlarından

$b_{26} = b_{46} = 1$, diğerleri 0'dır.

Eğer G yönlü bir graf ve e_k, i. düğümden J. düğüme bir kenar ise, k. sütununda $b_{ik} = -1$ ve $b_{jk} = 1$ diğer birleşenler 0'dır

Örnek:



Şekil 2.23.Yönlü graf

Yönlü grafının incidence matrisi aşağıdaki gibidir:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

e_1, 1'den 2'ye bir kenardır. 1. sütundan b_{11} = -1, b_{21} = 1 diğerleri 0.

e_2, 1'den 3'e bir kenardır. 2. sütundan b_{12} = -1, b_{32} = 1 diğerleri 0.

e_3, 2'den 4'e bir kenardır. 3. sütundan b_{23} = -1, b_{43} = 1 diğerleri 0.

e_4, 3'ten 2'ye bir kenardır. 4. sütundan b_{34} = -1, b_{24} = 1 diğerleri 0.

e_5, 3'ten 4'e bir kenardır. 5. sütundan b_{35} = -1, b_{45} = 1 diğerleri 0.

e_6, 4'ten 1'e bir kenardır. 6. sütundan b_{46} = -1, b_{16} = 1 diğerleri 0.

BÖLÜM 3
KODLAŞTIRMA TEORİSİ

3.1. Giriş

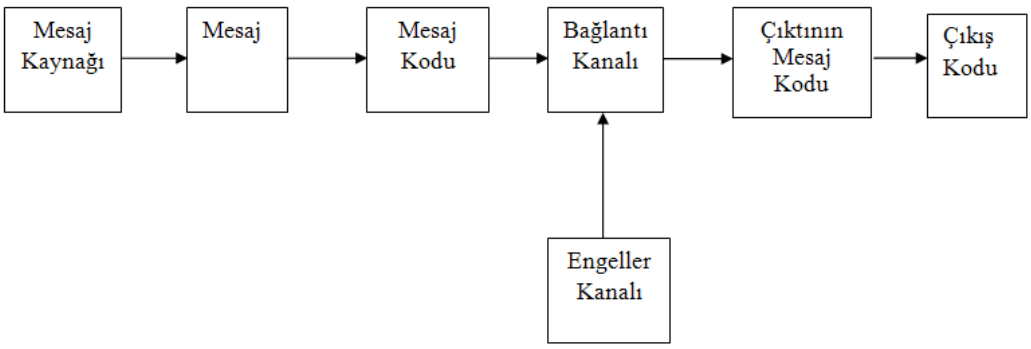
Kod denildiğinde, genellikle şifrelenmiş olan gizli kodları düşünürüz. Teorik olarak sadece ilgili kişinin anlayabileceği kodların iletilmesi (transmisyonu) ; ağların gizliliğinden emin olmak, askeri amaçlar ve yetkisi bulunmayan kişilerin ulaşabileceği bilgilerin bulunduğu diğer alanlar için kesinlikle çok önemlidir. Bu kodlar oluşturuldukları gibi kırılabilirler de. Mesajların engellenmesi ve şifrelerinin çözülmesi yetkisi bulunmayan kişilerce yapılmaktadır. Bu, ordu ve kodların güvenirliğinin korunması ve kırılmasıyla ilgilenen diğer kişiler için ciddi bir iştir. Ayrıca gittikçe popüler bir hobi olmaktadır. Böylece kodları kıranlarla kırılmaz kodlar yaratmaya çalışanlar arasında bir mücadele başlamıştır. Muhtemelen en önemli kod kırma olayı 2. Dünya Savaşındaki Almanların Enigma kodunun kırılmasıydı. Bunlarla ilgili kod teorisi bölümüne kriptoloji (gizlilik bilimi) denmektedir.

Kodlaştırma teorisi C.E.Shannon ve R.W.Hamming ile başladı. R.W.Hamming , 1950'de tek hata düzelten ikili kodların bir sınıfınının inşasını yapıp bunu makalesinde yayınladı. Bu kodlar, C.E.Shannon tarafından 1948'deki makalesinde söz edilen kodlardı. J.C.Golay , C.E.Shannon'ın makalesi aracılığıyla R.W.Hamming'in keşfini öğrendi. 1949'da J.C.Golay yapının genelleştirilmiş halini yayınladı. Hem R.W.Hamming'in orjinal ikili kodları hem de J.C.Golay'ın genelleştirilmeleri, şu anda Hamming kodları olarak bilinir. Daha önemlisi, J.C.Golay birden çok hata düzelten kodların iki yapısını verdi. Bu kodlar da kendi ismi ile Golay kodlar olarak bilinir. 1940'ların sonlarına doğru C.E.Shannon, R.W.Hamming ve J.C.Golay'ın yazdıkları makaleler yeni bilim sahalarının oluşmasına neden oldu. C.E.Shannon makalesinde iletişimin sınırları üzerine dayanan temel bir teori yayınladı. C.E.Shannon, bu makalesinde, herhangi bir kanal üzerinde kodları kullanarak düşük hata olasılığı elde etmenin mümkün olacağını gösterdi. Shannon'ın makalesi en azından iki araştırma alanın ortaya çıkmasına sebep oldu. Bunlardan biri başlıca performans üzerine dayalı sınırlarla ilgilenen bilgi teorisi (Information Theory) ve diğeri kodları kullanarak iyi iletişimler kurulması için metodlar geliştiren kodlaştırma teorisi (Coding Theory) oldu.
R.W.Hamming ve J.C.Golay tarafından yapılan keşifler hem mühendisler hem de matematikçiler tarafından araştırma sahaları oluşturdu. Mühendisler ilk önce gelişmiş bilgi iletişimi için yeni olanakları kullanmak istedi. Diğer taraftan, matematikçiler ise

daha çok kodların cebirsel yapılarına ilgi duydu. Dolayısıyla kodlaştırma teorisi bu şekilde gelişti.

3.2. Kodlaştırmanın Temel Terimleri

Kodlaştırma teorisi matematik için önemli bir unsurdur. Kodlaştırma, bazı nesneler üzerinde çalışmaların azaltılmasında önemli rol oynar. En basit örnek olarak, sayılarla ifade edilen ondalık sistem veya binary sistem verilebilir. Koordinat metodunun geometrideki rolünü göz önüne getirelim. Koordinat metodu matematiğin gelişmesinde önemli rol oynayan geometrik nesnelerin analitik açıklamalarla tanımlanmasını sağlamıştır. Ama bunun yanında kodlaştırma, bir araştırma konusu değil, bir lükstü. Daha sonra kontrol sistemlerinin oluşturulmasıyla kodlaştırma değişik bir önem kazandı ve kodlaştırma teorisini sistematik olarak araştırmak faydalı hale geldi . Hamming, Golay ve diğerleri her bir n bileşenli M vektörlerine belirli bir değer vererek M kapasitesinde ve n uzunluğunda bir kod oluşturmuşlardır. Şekil 3.1'de karekterize edilen diyagramda, kodlaştırmanın genel problemleri açıklanmıştır.



Şekil 3.1 Diyagram

Gürültü, bu süreçteki en önemli kısımdır, çünkü gürültünün olmayışı kodlama kuramına olan ihtiyacı ortadan kaldırır. Gürültü, teknik bir terim olup iletimin hatalı olmasına sebep olan etkenlerin geneline denir.

Sonlu sayıda harflerin oluşturduğu $U = \{ a_1, a_2, \ldots, a_r \}$ alfabesi verilmiş olsun. Bu alfabenin sembollerinden oluşan sonlu $A = a_{i1}a_{i2}\ldots a_{in}$ dizisine söz denir ve A sözünün uzunluğu olan n değeri L(A) ile tanımlanır. Eğer sözde hiçbir harf yoksa bu söz boş söz olarak adlandırılır. Farz edelim ki; S = S(U), U alfabesinin oluşturduğu bütün boş olmayan sözler kümesi olsun. S kümesinin herhangi S^l alt kümesini ele alalım. Böyle S^l alt kümesindeki sözleri doğuran nesneye mesaj kaynağı ve S^l sözlerine ise mesaj denir.

Mesaj kaynağı, bir otomasyon, kişi vb. tarafından verilebilir. Genelde kodlaştırma teorisi, problemleri incelediğinde mesaj kaynağı hakkındaki destekleyici bilgiyi kullanır. Mesaj kaynaklarını tanımlamanın birçok yöntemi vardır.

Kodlaştırma, iletişim alanında kaynakların, kanalların, alıcıların bilgi karakteristiklerini incelemek, bilginin iletimini en iyi duruma getirmek yani optimize etmek ve iletimin güvenilirliğinin düzeltilmesini sağlamak amacıyla kullanılmaktadır. Kod, birçok bilim dalı için önemli bir rol oynamaktadır. Bunlardan bazıları şöyle sıralanabilir; matematik alanında istatistik biliminde, bilgisayar alanında şifreleme konusunda, haberleşme alanında, haberleşme karmaşıklığını en düşük seviyeye indirmek ve kaynakların etkin biçimde temsil edilmesini sağlamak için kullanılmaktadır.

Kodları 1 ve 0 dan oluşan diziler temsil etmektedir. Bu belirli semboller genellikle alfabenin harflerini, klavyedeki rakamları bazen de kontrol tuşlarını içermektedir. Bu kodları bilgisayarda bilgilerin depolanması ve gönderilmesi (transmisyonu) için kullanılan dizi çiftlerini temsil eder. Birçok nedenden dolayı koda sahip olmak isteriz. Fakat bu nedenler birbirleriyle bağlantılı olmak zorunda değillerdir. Kodlarımızın sahip olmasını istediğimiz birkaç özelliği vardır. Kod koymak için bazılarını bulmalıyız. Bu bölümde kodlaştırmada kullanılan kodların değişik tiplerini inceleyeceğiz.

Kodların en önemli özelliği, ikili dizilerle (binary strings) oluşturulmuş mesajların kodun öğelerini sıralayarak gösterilmesidir, bu sıralama eşsizdir. Eğer mesajın şifresi çözülmüşse, kodu temsil eden harflerin sıralamasında problem olmamalıdır. Bu koda "**uniquely decipherable code**", "şifresi çözülebilir eşsiz kod" denilmektedir. Bu amacı gerçekleştirmek için bir çok yol vardır. Bir yolu bütün sembolleri aynı uzunluktaki dizi çiftleriyle kodlamaktır. Bunlara "**block code**" denir. Örneğin; eğer her bir sembolü 8 bit kullanarak kodlasaydık bunu her 8 bitin sonunda bilirdik, gönderilmiş mesaj simgesini temsil eden kod dizimiz olurdu. Eğer gönderilen

her simge ve harf için kodun uzunluğunu sınırlandırmamız gerekirse blok kodlar diğerlerine göre daha kullanışlıdırlar. Şifresi çözülebilir eşsiz kodlar yapmanın diğer bir yolu da "prefix code" kullanmaktır. Eğer kodun öğesinin başka bir kodun öğesinin başlangıç dizisi olmama özelliği varsa bu Prefix code'ları "C" ile tanımlıyoruz. Böylece, 1'ler ve 0'larla gösterilen A'nın sembolünü okuduğumuzda, o an A'nın dizisinin tamamlandığını biliyoruz. Prefix code'lara ayrıca "**instantaneous code**" (anlık kod) da denilmektedir.

Prefix code'ların bir tipi "**comma code**" (virgül kod)lardır. Her öğe 1'lerin sonunda 0'ın takip ettiği dizilerin oluşturduğu dizilerle kodlanır. Böylece dizilerden kurulan kodlar formlarını {0, 10, 110, 1110, 1111110,…} şeklinde alınırlar. Bu kodların en belirgin dezavantajı ise kod öğelerinin çok uzun olması ve depolama alanlarında çok geniş yer tutmasıdır.

Bazen depolama alanının küçültmek ve gönderme zamanının azaltmak için verilerin sıkıştırılması istenebilir. Alanları minimize etmekle en etkili kod "**Huffman code**"dur ayrıca bu kodun "anlık kod" yani Prefix kod olma avantajı vardır.

Gönderme süresini azaltan kodlara verilebilecek bilindik bir örnek "**Morse code**"dur. Hem Huffman hem de Morse kodlarında var olan harfler ve kullanılan semboller kısadır. Morse Alfabesi'nde harfler tek boşlukla (spaces), kelimeler ise üçer boşlukla ayrılır. Bu nedenle, boşluklar zamanın birimleridir.

Morse Kodu

A	· –	J	· – – –	S	· · ·
B	– · · ·	K	– · –	T	–
C	– · – ·	L	· – · ·	U	· · –
D	– · ·	M	– –	V	· · · –
E	·	N	– ·	W	· – –
F	· · – ·	O	– –	X	– · · –
G	– – ·	P	· – – ·	Y	– · – –
H	· · · ·	Q	– – · –	Z	– – · ·
I	· ·	R	· – ·		

Tablo 3.1

Veri aktarırken gönderme sürecinde hatalar oluşabilir. Bu, hatalar anlamsız sesler oluşturabilir. Örneğin; Galileo Voyager gibi uzak uzay araçlarından alınan verilerde çeşitli ses problemleri olabiliyor. Bazı durumlarda sadece hata bulmakla ilgilenebiliriz. Bu durum verinin gönderilemediğinden ya da tekrar tekrar gönderilmesinden kaynaklanabilir. Hata olduğunu algılama, tespit etme özelliği olan kodlara "**error-detecting codes**" , "hata bulma kodları" denir.

Diğer bir sorun, uzak uzay araçları gibi bilgilerini gönderilemediği yerlerde, veriyle ilgili yeterli bilgi istiyoruz; böylece sadece hatayı bulmakla kalmıyor ayrıca düzetiyoruz. Hataları düzeltme özelliği bulunan kodlara "**error-correcting codes**" , "hata düzeltme kodları" denir. Her zaman hata düzeltme kodları kullanmak bu nedenledir. Hata düzeltme kodlarıyla veya hata bulma kodlarıyla ilgili problemin çözülmesi için daha fazla bilgi gerektirir. Bu nedenle alanları minimize etmekte daha verimsizdirler. Ne yazık ki hata düzeltme kodlarının da, hata bulma kodlarının da hiç bir zaman hatayı bulup düzettiğinden emin olamayız. Problem hatanın çok olmasındandır. Eğer hata tekse bulmak ve düzeltmek mümkündür. Yapabileceğimiz en iyi şey bulunamamış ve düzeltilememiş hatalarla ilgili olasılıkları azaltmaktır. Fakat ihtimalleri azaltmaya engel olan çok problem, göndermemiz gereken daha çok bilgi ve olması gerekenden daha verimsiz kodlarımız var.

Diğer bir kod çeşidimiz de **Gray Code**'udur. Yansımalı kodlar adıyla anılan Gray kodunda sayılar arasındaki geçişte sadece bir bit değişir. Bu kodların kullanım amaçlarını gösteren klasik bir örnek var. Farz edelim bölümlere ayrılan döndürücü bir diskimiz ve diskin ne kadar uzakta döndüğü dijital bilgisini geri gönderen bir dizi fırça veya lazer ışınlarımız olsun. Eğer dizi çiftlerinin numaralandırılmış bitişik sektör (daire dilimi) boyutları yeterince farklıysa, yani bir sektörden diğerine geçerken bir çok rakam farkı varsa; bu durumda sektör değişiyormuş gibi okunması, diğer sektör numaralarından tamamen farklı numara üretmesindendir. Bu durumda sektörün numaralandırılması istenir çünkü bitişik sektörler arasında sadece bir rakam değişikliği vardır. Bu özellikteki kodlara "Gray Code" denir. Hatayı azalttığından dolayı özellikle Analog-Sayısal dönüştürücülerde, bilgisayar kontrollü cihazlarda oldukça tercih edilen bir kodlamadır.

Sayısal sistemler birbirleri ile haberleşirken bilginin değişmesi oldukça sıklıkla karşılaşılan bir konudur. Bilgi değişimlerini yani hata oluşumlarını denetleyebilmek ve gönderilen bilginin doğruluğunu kontrol etmek amacı ile Parity Kodu ortaya çıkmıştır. Parity bitinin genel kullanımı şu şekildedir: Eğer kodumuzda tek sayıda 1 varsa parity bitimiz 1, çift sayıda 1 var ise 0 ayarlanır. Sonuçta kod parity biti ile çift sayıda 1 içerir. Bu metodu "ASCII Code" u (American Standart Code for Information Interchange - Bilgi Alışverişi için Standart Amerikan Kodu) kullanarak gösteriyoruz. Bu 7 bit kullanılarak oluşturulmuş blok koddur, bu nedenle her sembol yedi bitlik 1 ve 0 dizisidir. Sekizinci bit hepsine bir parity biti eklenmesiyle oluşur. Parity hata kontrolü pek de etkili bir yöntem değildir çünkü kodumuzda çift sayıda bit değişirse parity bitimiz de geçerliliğini koruyacaktır ve bu durumda hatanın tespiti imkansızdır. Bunun dışında parity biti hata tespit edilse bile hatanın hangi bitte olduğunu gösteremez. Bu durumda kod tamamen çıkarılıp başlangıcından itibaren yeniden aktarılmak zorundadır. Bu yüzden gürültülü bir iletişim ortamında verinin iletilmesi çok uzun zaman alabilir hatta ve hatta hiçbir zaman gerçekleşmeyebilir.

Gönderilme hatası olasılığı 0,01 varsayılır, 1, 0'a; 0, 1'e çevrilir. Diğer varsayım, hata olasılığı mutlaka aynı hata yerindedir ve hatalı yerde 1, 0'a çevrilir ya da 0, 1'e. Biz ayrıca bir hata oluşumunun diğer oluşumunun olasılığını etkilemediği varsayarız.

Bir hatanın olasılığının $\binom{8}{1}$ $(0{,}01)(0{,}99)^7$, yaklaşık olarak 0,07 olduğu Binom teoreminden biliyoruz. Bununla birlikte iki hatanın olasılığı $\binom{8}{2}$ $(0{,}01)^2(0{,}99)^6$ yaklaşık olarak 0,002'dir ve yeterince küçüktür. Üç hata olduğunda hatalı bitlerin belirlenme olasılığı $\binom{8}{3}$ $(0{,}01)^3(0{,}99)^5$, yaklaşık olarak 0,00005'tir. Buradan da anlaşılacağı üzere, hata sayısı arttıkça hatalı bitlerin tespit olasılığı giderek azalmaktadır.

Kodların verilen sayıda kodlanmış tekrar eden diziler olduğunu varsayın. Örneğin; kodun her hangi dizisi kodlanırken bir kere tekrar eder; sonra 10110, 10110 10110 şeklinde kodlanır. Eğer iki kere tekrar ederse, sonra 10110 10110101101011 olur. Bir kere tekrar eden kodlanmış diziden sonra hata bulma kodumuz varsa hata oluştuğunda uygun pozisyonlar aynı olmayabilir.

Örneğin; kodlanmış dizi 1111110101 10111011 iken üçüncü ve son bitte hata oluşursa dizinin hangi kopyasında hata oluştuğunu bilmeden hatayı düzeltemeyiz. Hatayı en iyi şekilde dizimiz iki kere tekrar ettiğinde bulabiliriz. Eğer dizimizin üç kopyası varsa tek hata için kodu düzeltebiliriz. Dizinin uygun pozisyonlarında farklı bitler varsa iki kere yer alan değeri kabul ederiz. Örneğin; dizimizin uzunluğu 4 olsun ve 110110011101 alalım, sonra ikinci durumda iki 1 ve 0 aldık. Böylece doğru değerimiz 1 olmalı ve kodlanması gereken doğru dizimiz 1101 olmalıdır. Dizinin aynı pozisyonunda birden çok hata oluşursa, problemin bu noktada olduğu kesindir. Eğer dizimiz ‘ n ’ kopyalı şekilde tekrarlıysa hata düzeltici $\left[\frac{n}{2}\right]$ kerenin altındaki tekrarlarda aynı pozisyonda hata oluştuğu anda doğru sonucu verir.

Parity Bitiyle ASCII Kodu

Kod	*Sembol*	*Kod*	*Sembol*	*Kod*	*Sembol*	*Kod*	*Sembol*
00000000	NUL	10100000	SP	11000000	@	01100000	‘
10000001	SOH	00100001	!	01000001	A	11100001	A
10000010	STX	00100010	“	01000010	B	11100010	B
00000011	ETX	10100011	#	11100011	C	01100011	C
10000100	EOT	00100100	$	01000100	D	11100100	D
00000101	ENQ	10100101	%	11000101	E	01100101	E
00000110	ACK	10100110	&	11000110	F	01100110	F
10000111	BEL	00100111	/	01000111	G	11100111	G
10001000	BS	00101000	(	01001000	H	11101000	H
00001001	HT	10101001	)	11001001	I	01101001	İ
00001010	LF	10101010	*	11001010	J	01101010	J
10001011	VT	00101011	+	01001011	K	11101011	K
00001100	FF	10101100	‘	11001100	L	01101100	L
10001101	CR	00101101	-	01001101	M	11101101	M
10001110	SO	00101110	.	01001110	N	11101110	N
00001111	SI	10101111	/	11001111	O	01101111	O
10010000	DLE	00110000	0	01010000	P	01110000	P
00010001	DC1	10110001	1	11010001	Q	01110001	Q
00010010	DC2	10110010	2	11010010	R	01110010	R
10010011	DC3	00110011	3	01010011	S	11110011	S
00010100	DC4	10110100	4	11010100	T	01110100	T
10010101	NAK	00110101	5	01010101	U	11110101	U
10010110	SYN	00110110	6	01010110	V	11110110	V
00010111	ETB	10110111	7	11010111	W	01110111	W
00011000	CAN	10111000	8	11011000	X	01111000	X
10011001	EM	00111001	9	01011001	Y	11111001	Y
10011010	SUB	00111010	:	01011010	Z	11111010	Z
10011011	ESC	10111011	;	11011011	[	01111011	{
10011100	FS	00111100	<	01011100	\	11111100	_
00011101	GS	10111101	=	11011101	]	01111101	}
00011110	RS	10111110	>	11011110	^	01111110	˜
10011111	US	00111111	?	01011111	_	11111111	DEL

Tablo 3.2.

3.3. Dekodlaştırmanın Tek Değer Problemi

3.3.1 Kodlaştırma

Farz edelim, V = { $b_1, b_2, \ldots, b_q$ } alfabesi verilmiş olsun. Benzer olarak V alfabesindeki sözleri B ile ve V alfabesindeki bütün boş olmayan sözleri S(V) ile gösterelim. F, $S^l(U)$ dan S(V) ye bir fonksiyon olsun. O zaman, her $A \in S^l(U)$ için öyle $B \in S(V)$ bulunur ki bu durumda B=F(A) olur. Bu durumda B sözüne A mesajının kodu, A sözünden B koduna geçişe ise kodlaştırma denir.

3.3.2. Dekodlaştırma

Kodlaştırmadaki önemli konulardan biri de dekodlaştırma yani ters kodlaştırmadır. Dekodlaştırma aşağıdaki gibi tanımlanır.

Verilmiş B koduna göre ilkel A mesajının (kodu B olan A mesajının) bulunmasına dekodlaştırma denir. Yani dekodlaştırma çıkıştaki bir mesajın düzeltilmesi ile elde edilen ilk mesaja geçiştir. Dekodlaştırma gönderilecek mesajın eğer tersi F^{-1} mevcut ise mümkündür. Dekodlaştırma her mesaj için mümkün değildir sadece tek değerli kodlar için yapılabilir.

Kodlaştırma teorisinde F fonksiyonu genel olarak algoritmalarla verilir.

Bunlardan alfabe kodlaştırmasını inceleyelim.

3.4 Alfabe Kodlaştırması:

U alfabesindeki harflerle B alfabesindeki sözler arasında aşağıdaki uygunluğun olduğunu varsayalım.

Örnek1:

a_1 — B_1

a_2 — B_2

.....

a_r — B_r

Kodlaştırma teorisinde bu uygunluğa şema denir ve $\sum$ ile gösterilir. Bu yöntem alfabe kodlaştırmasını aşağıdaki gibi tanımlar.

$S^l(U) = S(U)$ kümesinden alınan her bir $A = a_{i1}, a_{i2} \ldots, a_{im}$ sözüne A 'nın

kodu denilen $B = B_{i1}B_{i2} \ldots B_{in}$ sözü uygundur. Burada $B_1, B_2, \ldots, B_r$ sözlerine basit kodlar denir.

3.5. Dekodlaştırmanın Tek Değerliliği

U ve V alfabeleri için $\sum$ şemasıyla verilmiş alfabe kodlaştırmasını göz önüne alalım. S'(U) = S(U) sağlandığını kabul edelim. Yani bu durumda, mesaj kaynağı U alfabesindeki bütün sözleri doğurur. Çok açıktır ki, alfabe kodlaştırması S(U) kümesinden S(V) kümesine geçişi sağlar. S(U) kümesinin böyle alfabe kodlaştırmasının görüntüsünü $S_{\sum}$(V) ile gösterelim. Eğer S(U) kümesinin $S_{\sum}$(V) görüntüsüne geçişi tek değerli ise o zaman dekodlaştırma mümkündür.

Örnek 2 : U ={ a_1,a_2 } ve V ={ b_1,b_2} alfabelerine uygun aşağıdaki şemayla verilen alfabe kodlaştırmasını göz önüne alalım.

A^{I} ve A^{II} nün, B^{I} ve B^{II} nin kodları olduğunu varsayalım.

$\sum$:

a_1 — b_1

a_2 — b_1b_2

Böyle alfabe kodlaştırması için dekodlaştırma aşağıdaki gibidir. Burada eğer $A^{I} \neq A^{II}$ ise $B^{I} \neq B^{II}$ olduğu kesindir.

Dekodlaştırma prosedürü şu şekilde gerçekleşir:

$B \in S_{\sum}$(V) sözü basit kodlara bölünür. Basit kodlara ayırmak için her bir b_2 harfinden önce gelen b_1 bulunur ve tüm (b_1b_2) çiftleri ayarlanır. Bu adımdan sonra B sözünün geri kalanında yalnızca b_1 harfleri bulunur.

Daha sonra şemaya uygun olarak, her bir (b_1b_2) çiftleri için a_2, geri kalan b_1 harfleri için a_1 değerleri yer değiştirilirse, kodu B olan A sözünü buluruz.

Varsayalım, $B = b_1 b_1 b_2 b_1 b_2 b_1 b_1 b_1 b_2$ verilmiş olsun, verilen şemaya uygun olarak tüm ($b_1 b_2$) çiftleri ayarlanırsa,

$B = b_1 (b_1 b_2)(b_1 b_2) b_1 b_1 (b_1 b_2)$ bulunur ve şemaya uygun olarak $a_1 \ldots b_1$, $a_2 \ldots b_1 b_2$ işlemleriyle,

$B = a_1 a_2 a_2 a_1 a_1 a_2$ yani kodu B olan A sözü (dekodlaştırma) bulunmuş olur.

Alfabe kodlaştırmasının tek değerli olmadığı durumlarda dekodlaştırma mümkün değildir. O nedenle her bir gözönüne alınan alfabe kodlaştırmasının tek değerli olup olmadığının belirtilmesi çok önemlidir.

Bu problemin çözümü için yöntem ya da kriter verilmelidir. Aksi halde sonsuz sayıdaki sözlerin incelenmesi gerekebilir. Bu durum ise, mümkünü olmayan bir problemdir. Alfabe kodlaştırması için genel bir kriter tanımlamadan önce, daha basit ve yeterli olan durumu göz önüne alalım.

Tanım 3.1: $B = B^{I} B^{II}$ ile yazılmış olsun. O zaman B^{I} sözüne B sözünün prefix'i (önek) , B" sözüne ise B sözünün sonu denir. B sözünün kendisi ile boş söz (**Λ**), B sözünün prefix'i ve sonu gibi alınır. $B = B\Lambda = B'B''$

Tanım 3.2: Eğer istenilen $\forall$ i, j $(1 \leq i, j \leq r)$, $i \neq j$ için B_i sözü B_j sözünün prefix 'i değilse, o zaman $\sum$ şeması " prefix özelliğine sahiptir" denir.

Yukarıda baktığımız örnek 2'den görüldüğü gibi, $\sum$ şeması prefix özelliğine sahip değildir.

3.5.1. Dekodlaştırma İçin Yeterli Koşul

Eğer $\sum$ şeması prefix özelliğine sahip ise o zaman alfabe kodlaştırması tek değerlidir. Ancak yeterli koşulu sağlamayan alfabe kodlaştırması da tek değerli olabilir. Bu durum

örnek 2' den de görülüyor.

$B_1 = b_1$

$B_2 = b_1 b_2$, burada $B_2 = B_1 b_1$ olduğundan B_1 sözü, B_2 sözünün prefix'i olmasına (prefix özelliğine sahip olmamasına) rağmen tek değerlidir.

$B = B_{i1}, B_{i2}, \ldots, B_{in}$ S(V) kümesinden alınmış herhangi bir söz olsun.

$\tilde{B} = B_{in}, B_{in-1}, \ldots, B_{i1}$ ile işaret edelim. Daha sonra,

$a_1 \quad — \quad \tilde{B}_1$

……..

$a_r \quad — \quad \tilde{B}_r$

gibi bir $\tilde{\sum}$ şemasını göz önüne alalım. Örneğin yukarıdaki $\sum$ şemasında;

$\sum$: a_1—b_1 $\tilde{\sum}$: a_1—b_1

a_2—b_1b_2 a_2—b_2b_1

Bu halde, görüldüğü gibi $\tilde{\sum}$ şeması prefix özelliğine sahiptir ve yeterli koşula göre bu şemaya karşılık gelen alfabe kodlaştırması tek değerlidir.

$\sum$ ve $\tilde{\sum}$ şemaları ile belirtilen alfabe kodlaştırmasının, her ikisi aynı zamanda ya karşılıklı tek değerlidir, ya da karşılıklı tek değerli değildir. Bunu göz önünde tutarak yeterli koşulu biraz daha iyileştirebiliriz.

Eğer $\sum$ şeması veya $\tilde{\sum}$ şeması prefix özelliğine sahip ise, o zaman bu şemaların belirttiği alfabe kodlaştırması tek değerlidir. Ancak şu durumu hatırlatmak gerekir ki, yeterli koşulu sağlamayan alfabe kodlaştırması da tek değerli olabilir.

Örnek 3: U={ a_1,a_2,a_3 } ve V ={ b_1,b_2,b_3 } alfabelerine uygun aşağıdaki şemayla verilen alfabe kodlaştırmasını göz önüne alalım.

$\sum$:

a_1—b_1

a_2—b_1b_2

a_3—b_3b_1

Görüldüğü gibi burada basit kodlar; $B_1=b_1$, $B_2=b_1b_2$ ve $B_3= b_3 b_1$ dir. Aynı zamanda

$B_2= B_1b_2$ olduğundan, B_1 basit kodu B_2 kodunun prefix'idir. Bu durumda verilen alfabe kodlaştırılması prefix özelliğine sahip değildir.

Öte yandan $\sum$ şemasına uygun $\tilde{\sum}$ şeması tanıma göre şöyledir.

$\tilde{\sum}$:

a_1—b_1
a_2—b_2 b_1
a_3—b_1 b_3

Burada basit kodlar; $B_1=b_1$, $B_2=b_2b_1$ ve $B_3=b_1b_3$ 'dir. Aynı zamanda $B_3= B_1b_3$ olduğundan, B_1 basit kodu B_3 kodunun prefix'dir. Bu durumda verilen alfabe kodlaştırlması prefix özelliğine sahip değildir.

Böylece örneğimizde, hem $\sum$ şeması hem de $\tilde{\sum}$ şeması prefix özelliğine sahip değildir. Fakat bu şemaların belirttiği alfabe kodlaştırması tek değerlidir.

Aslında eğer $B \in S_{\sum}(V)$ ise bu sözü basit kodlara bölmek mümkündür. Basit kodlara ayırmak için B sözündeki her bir (b_1b_2) çiftleri ve (b_3b_1) çiftleri ayarlanır. Bu adımlardan sonra B sözünün geri kalan kısmı yalnızca b_1 harflerinden oluşmuş olacaktır. Daha sonra $a_1= b_1$, $a_2=b_1b_2$ ve $a_3=b_3b_1$ dönüşümleri yapılacak olunursa, tek değerli olarak kodu B olan A sözü bulunmuş olur ve sonuç olarak $\sum$ şemasındaki basit kodların çifter çifter ayrık olduğunu söyleyebiliriz

Varsayalım ki, $B =b_1\ b_1\ b_2\ b_1\ b_1\ b_3\ b_1$ olsun.

$\sum$ şemasına göre ;	$\tilde{\sum}$ şemasına göre;
$B =b_1\ (b_1\ b_2)\ b_1\ b_1(b_3\ b_1)$	$B = b_1\ b_1\ (b_2b_1)\ (b_1\ b_3)\ b_1$
$A = a_1a_2a_1a_1a_3$	$A =a_1\ a_1\ a_2\ a_3\ a_1$

Bu örneklerden görüldüğü gibi, her zaman yeterli koşul uygulanarak kodlaştırmanın tek değerli oluıp olmadığı ayarlanamaz. Bu amaçla dekodlaştırmanın tek değerlilik kriterinin verilmesi çok önemlidir.

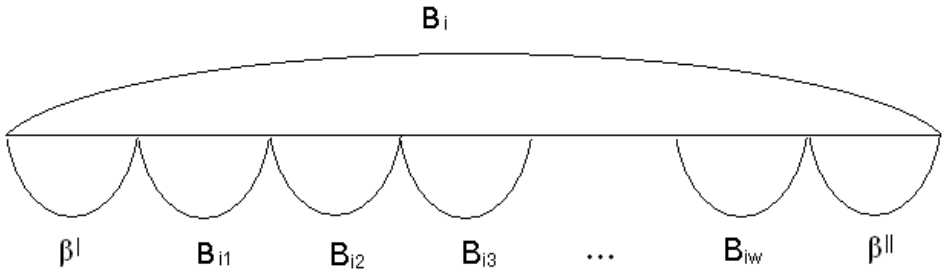
3.6. Dekodlaştırmanın Tek Değerlilik Koşulu

B sözünü oluşturan harflerin sayısına B sözünün uzunluğu demiştik. Bu durumda basit B_i kodları için $\ell(B_i) = \ell_i\ (i =1, 2,..., r)$ diyelim ve kodun uzunluğuna da

$L(B) = L$ (tüm kodu oluşturan harflerin uzunluğu) diyelim.

$L(B) = \ell(B_1) + \ell(B_2) + ... + \ell(B_r)$ dir.

$B_i = \beta' B_{i1} \ldots B_{iw} \beta''$ ile gösterelim. (1)



Şekil 3.2.

Burada β' ve β'' basit kodlardan farklı yazılışlardır. Eğer , $\beta' = \beta'' = \Lambda$ (boş söz) koşulu sağlandığında yalnızca bir doğru yazılış mümkün ise bu durumu dikkate almıyoruz. Böyle $B_i \neq B_i$ ayrılışlara B_i kodunun basit olmayan ayrılışından farklı ayrılış denir.

Çok açıktır ki, her bir B_i basit kodu için (1) nolu ayrılış sayısı sonludur.

W maximum w'yi belirtir ve B_i 'den alınan tüm dekompozisyonları belirtir**.**

$W = \max w$

U alfabesindeki uzunluğu N' den büyük olmayan boş sözden farklı sözlerin oluşturduğu kümeyi $S^N(U)$ ile gösterelim.

Çok açıktır ki, $S^N(U)$, $\sum_{i=1}^{N} r^i$ elemanlı sonlu sayılar kümesidir**.**

3.7. Markov Algoritması

$$N_0 \leq \left[\frac{(w+1)(L-r+2)}{2}\right] \qquad (3.1)$$

Eşitsizliğini sağlayan öyle N_0 sayısı var ki , $\sum$ şemalı alfabe kodlaştırmasının karşılıklı tek değerlilik problemi, sonlu $S^{No}(U)$ kümesinin kodlaştırılması için benzer probleme indirgenir.

$A' , A'' \in S^{No}(U)$ olmak üzere;

$$\ell(A'), \ell(A'') \leq \left[\frac{(w+1)(L-r+2)}{2}\right] \qquad (3.2)$$

Çok açıktır ki, eğer $N_0 = \max(\ell(A'), \ell(A''))$ yaparsak tek değerli dekodlaştırma $S^{No}(U)$ kümesi üzerinde daha fazla olmaz.

$A' , A'' \in S^{No}(U)$ olduğundan tek değerli dekodlaştırma kriteri , $\sum$ şeması kullanılarak alfabe kodlaştırmasının tek değerli olup olmadığı U alfabesindeki uzunluğu N_0'dan fazla olmayan kelimeleri göz önüne alarak belirlememize olanak sağlayan basit bir algoritmayı doğurur. Böyle algoritmaların karmaşıklığı yaklaşık olarak r^{No} olarak hesaplanabilir.

Örnek 4: $U=\{a_1, a_2, a_3, a_4, a_5\}$ ve $V=\{b_1, b_2, b_3\}$ alfabesi üzerinde aşağıdaki şema için N_0 ifadesini bulalım.

$\sum$:

a_1—b_1b_2

a_2—$b_1b_3b_2$

a_3—b_2b_3

a_4—$b_1b_2\ b_1b_3$

a_5—$b_2\ b_1b_2\ b_2b_3$

$2 \leq \ell_i < 6$

$W = \max W_i < 3$

Ayrıca, $B_5= b_2\ b_1b_2\ b_2b_3= b_2\ B_1B_3$ 'dir.

Bu nedenle W = 2 olarak hesaplanır.

r = 5 ve $L=\ell_1+\ell_2+\ell_3+\ell_4+\ell_5= 16$ dır. Bu durumda;

$$N_0 \le \left[\frac{(w+1)(L-r+2)}{2}\right]=\frac{3*13}{2}$$ bulunur.

Çok açıktır ki, böyle alfabe ile verilmiş ve uzunluğu 19'u aşmayan sözlerin sayısı en azından 5^{19} (r^{No}) olur ve tek değerliliği kontrol etmek için tek tek bu sözlerin incelenmesi gerekir.

Bildiğimiz gibi, herhangi alfabe üzerinde verilebilen tüm sözlerin incelenmesi mümkün değildir. Görüldüğü gibi bu adımda karşılıklı tek değerlilik kriteri biraz iyileşmesine karşılık, incelediğimiz örnekten de görüldüğü gibi bu kriterin de her zaman uygulanması iyi sonuç vermiyor, örneğin bu örnekte en azından 5^{19} sayıdaki sözlerin incelenmesi gerekmektedir.

Yukarıda incelediğimiz yeterli koşuldan yararlanarak göz önüne aldığımız alfabe kodlaştırmasının tek değerli olup olmadığını söyleyemeyiz. Çünkü böyle bir alfabe kodlaştırması prefix özelliğine sahip değildir. Görüldüğü gibi $B_4=B_1b_1b_3$ tür. Yani B_1 elementer kodu, B_4 elementer kodunun prefix'idir.

Böylece göz önüne alınan örnek için ne yeterli koşul uygulaması ile dekodlaştırma ayarlanabilir ne de Markov algoritması kullanmak amaca uygundur.

Kodlaştırma teorisinde bu tür alfabe kodlaştırmaları sık sık rastlanan problemlerdir. Böyle problemleri daha basit şekilde çözebilmek için daha uygun farklı bir algoritma verelim.

3.8. Dekodlaştırmanın Tek Değerliliğini Belirten Algoritma

Bu algoritma dekodlaştırmanın mümkün olup olmadığına karar vermek için gerekli ve yeterli derecede etkin olan bir algoritmadır. $\sum$ şemalı alfabe kodlaştırmasının karşılıklı tek değerli olması için gerekli ve yeterli koşul (kriter) $\Gamma(\sum)$ grafının boş söze (Λ) uygun köşesinden geçen yönlendirilmiş kapalı döngünün olmamasıdır.

Böylece göz önüne alınan alfabe kodlaştırmasının tek değerli olup olmaması, dolayısıyla dekodlaştırmanın mümkün olup olmaması graflar teorisinin elemanları uygulanarak ayarlanabilir.

Farz edelim ki, aşağıdaki gibi alfabe kodlaştırılması verilmiştir.

a_1—B_1

a_2— B_2

.....

a_r— B_r

Her bir B_i elementer kodu için tüm basit olmayan $B_i=\beta' B_{i1}\ldots B_{iw} \beta''$ (1) ayrılışlarını göz önüne alalım. Burada β' ve β'' basit kodlardan farklı kodlardır.

Boş sözü Λ , uygun β' ve β'' kodlarının oluşturduğu kümeyi K_o olarak gösterelim.

K_o kümesini oluşturmak için aşağıdaki işlemler yapılır.

1- Her zaman için boş söz (Λ) içerir.
2- (1) yazılışında aynı zamanda hem başlangıçta hem de sonda bulunan β' ve β'' sözlerini içerir.

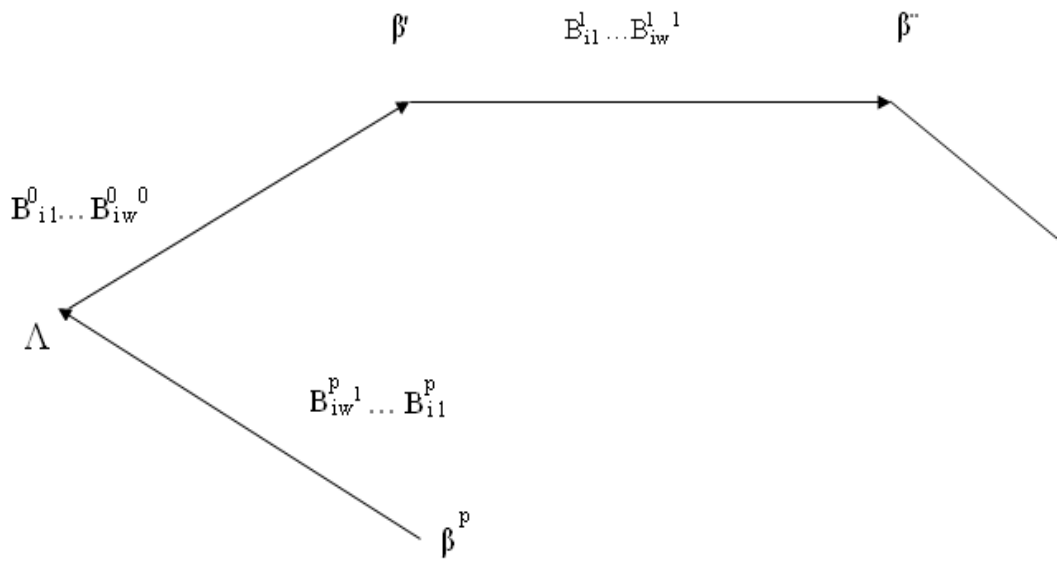
K_o kümesi ayarlandıktan sonra yapılacak adımlar şöyle sıralanabilir:

1- Göz önüne alınan alfabe kodlaştırması incelenerek her bir B_i elementer kodu için bütün (1) nolu ayrılışlar yazılır.

2- K_o elemanlarından yararlanarak $\Gamma(\sum)$ grafı çizilir.

3- Boş söze (Λ) uygun köşeden kapalı döngünün geçip geçmediği araştırılır**.**

Görüldüğü gibi bu algoritmanın uygulanması çok basittir ve her bir alfabe kodlaştırması için gerekli ve yeterlidir. β' , $\beta'' \in K_o$ olsun ve (1) ifadesindeki tüm ayrılışlar için β' , β'' kodlarına uygun gelen köşeleri bileştirdiğimizde oluşan grafı $\Gamma(\sum)$ ile gösteririz.



Şekil 3.3.Graf

Teorem 3.1: Eğer Γ(∑) grafında Λ(boş söz) üzerinden geçen bir döngü varsa o zaman ∑ şeması tek değerli değildir.

İspat**:** Alfabe kodlaştırmasının tek değerli olmadığını varsayalım. Bu durumda kısaltılamayan söz vardır.

$B = B^0_{i1} \dots B^0_{iw}{}^0 \beta' B^1_{i1} \beta'' \dots \beta^p B_i{}^p{}_1 \dots B_i{}^p{}_w{}^p$ sözü için,

$B_i{}^0 = B^0_{i1} \dots B^0_{iw}{}^0 \beta'$

$B_i{}^1 = \beta' B^1_{i\ 1} \beta'' \dots B^1_{i\ w} \beta''$

……

……

$B_i{}^p = \beta^p B_i{}^p{}_1 \dots B_i{}^p{}_w{}^p \beta''$ olur.

Bu nedenle de, Γ(∑) grafında Λ(boş söz) üzerinden geçen bir döngü vardır.

3.8.1. Yeterlilik

Γ(∑)) grafı Λ (boş söz) üzerinden geçen bir döngüye sahip olsun. Sonra,

$B = B^0_{i1} \ldots B^0_{iw}{}^0 \beta' B^1_{i1} \ldots B^1_{iw}1 \beta'' \ldots \beta^p B_i{}^p{}_1 \ldots B_i{}^p{}_w{}^p$ sözü için bölümlerle tanımlanmış aşağıdaki iki çevirim vardır.

$B = (B^0_{i1}) \ldots (B^0_{iw}{}^0)(\beta' B^1_{i1} \ldots B^1_{iw}1 \beta'') \ldots (B_i{}^p{}_1 \ldots B_i{}^p{}_w{}^p)$

$B = (B^0_{i1} \ldots B^0_{iw}{}^0 \beta')(B^1_{i1}) \ldots (B^1_{iw}1) \ldots (B_i{}^p{}_1) \ldots (B_i{}^p{}_w{}^p)$ olur.

ispat tamamlanmıştır.

Γ(∑) grafının çizilmesi algoritmanın yapılmasına bağlıdır. Eğer Λ (boş söz) üzerinden geçen bir döngü yoksa kodlaştırma karşılıklı tek değerlidir, yani dekodlaştırma mümkündür.

Örnek 5:

Aşağıdaki şemayla verilen alfabe kodlaştırmasının tek değerli olup olmadığını araştırınız.

a_1—b_1b_2

a_2—$b_1b_3b_2$

a_3—b_2b_3

a_4—$b_1b_2\ b_1b_3$

a_5—$b_2b_1b_2b_2b_3$

Önce verilen şemaya uygun tüm (1) nolu basit ayrılışları yazalım

$B_1 = (b_1)\,(b_2)$	$\beta' = b_1$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_2$
$B_2 = (b_1)\,(b_3\,b_2)$	$\beta' = b_1$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_3\,b_2$
$B_2 = (b_1\,b_3)\,(b_2)$	$\beta' = b_1\,b_3$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_2$
$B_3 = (b_2)\,(b_3)$	$\beta' = b_2$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_3$
$B_4 = (b_1)\,(b_2\,b_1\,b_3)$	$\beta' = b_1$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_2\,b_1\,b_3$
$B_4 = (b_1\,b_2)\,(b_1\,b_3)$	$\beta' = \Lambda$	$B_{i1} \ldots B_{iw} = B_1$	$\beta'' = b_1\,b_3$
$B_4 = (b_1\,b_2\,b_1)\,(b_3)$	$\beta' = b_1\,b_2\,b_1$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_3$
$B_5 = (b_2)\,(b_1\,b_2\,b_2\,b_3)$	$\beta' = b_2$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_1\,b_2\,b_2\,b_3$
$B_5 = (b_2)\,(b_1\,b_2)\,(b_2\,b_3)$	$\beta' = b_2$	$B_{i1} \ldots B_{iw} = B_1\,B_3$	$\beta'' = \Lambda$
$B_5 = (b_2\,b_1)\,(b_2\,b_2\,b_3)$	$\beta' = b_2\,b_1$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_2\,b_2\,b_3$
$B_5 = (b_2\,b_1\,b_2)\,(b_2\,b_3)$	$\beta' = b_2\,b_1\,b_2$	$B_{i1} \ldots B_{iw} = B_3$	$\beta'' = \Lambda$
$B_5 = (b_2\,b_1\,b_2\,b_2)\,(b_3)$	$\beta' = b_2\,b_1\,b_2\,b_2$	$B_{i1} \ldots B_{iw} = \Lambda$	$\beta'' = b_3$

Tablo 3.3

Yukarıdaki iki koşula uygun olarak;

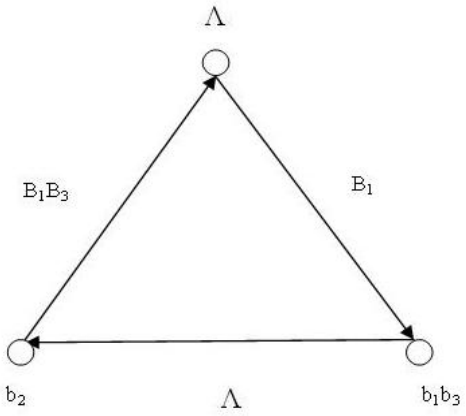
$K_0 = \{\ \Lambda\ ,\ b_2\ ,\ b_1\,b_3\ \}$ olarak bulunur. K_0 ifadesine uygun elementer kodlar aşağıdaki gibidir.

$B_2 = (b_1\,b_3)\,(b_2)$

$B_4 = (b_1\,b_2)\,(b_1\,b_3) = \Lambda\,B_1\,(\,b_1\,b_3\,)$

$B_5 = (b_2)\,(b_1\,b_2)\,(b_2\,b_3) = (b_2)B_1\,B_3\ \Lambda$

Algoritmaya uygun yazılan elementer kodların grafı çizilerek dekodlaştırmanın mümkün olup olmadığı araştırılır.



Şekil 3.4

Şekilden de görüldüğü gibi kapalı döngü olduğu için bu şema ile verilen kodlaştırma karşılıklı tek değerli değildir, yani dekodlaştırma mümkün değildir.

Dekodlaştırmanın imkansız olduğu, aşağıdaki B kodu için dekodlar bulunarak araştırılır.

$B = B_1 b_1 b_3 b_2 B_1 B_3$ ifadesi iki şekilde ifade edebiliriz.

$B = (B_1 b_1 b_3)(b_2 B_1 B_3)$ şeklindeki ayrılış için $A^{|} = a_4 a_5$ olan B'nin dekodu,

$B = B_1(b_1 b_3 b_2) B_1 B_3$ şeklindeki ayrılış için $A^{|} = a_1 a_2 a_1 a_3$ olan B' nin dekodu bulunur.

Aynı koda ait dekodlar görüldüğü gibi birbirinden farklı çıktığı için bu şema ile verilen alfabe kodlaştırması karşılıklı tek değerli değildir, dolayısıyla dekodlaştırma mümkün değildir.

Örnek 6) Aşağıdaki alfabe kodlaştırmasının tek değerli olup olmadığını araştıralım.

$\sum$:

a_1 —b_1

a_2 —$b_2 b_1$

a_3 —$b_1 b_2 b_2$

a_4 —$b_2 b_1 b_2 b_2$

a_5 —$b_2 b_2 b_2 b_2$

Önce verilen şemaya uygun tüm basit ayrılışları yazarız.

$B_2 = (b_2)(\ b_1) = (b_2)\ B_1$	$\beta^l = b_2$	$B_{i1} \ldots B_{iw} = B_1$	$\beta^{ll} = \Lambda$
$B_3 = (b_1)(\ b_2 b_2) = B_1\ (b_2 b_2)$	$\beta^l = \Lambda$	$B_{i1} \ldots B_{iw} = B_1$	$\beta^{ll} = b_2 b_2$
$B_3 = (b_1 b_2)(\ b_2)$	$\beta^l = b_1 b_2$	$B_{i1} \ldots b_{iw} = \Lambda$	$\beta^{ll} = b_2$
$B_4 = (b_2)(b_1)(\ b_2 b_2) = (b_2) B_1 (b_2 b_2)$	$\beta^l = b_2$	$B_{i1} \ldots B_{iw} = B_1$	$\beta^{ll} = b_2 b_2$
$B_4 = (b_2)(\ b_1 b_2 b_2) = (b_2)\ B_3$	$\beta^l = b_2$	$B_{i1} \ldots B_{iw} = B_3$	$\beta^{ll} = \Lambda$
$B_4 = (b_2 b_1)(\ b_2 b_2) = B_2\ (b_2 b_2)$	$\beta^l = \Lambda$	$B_{i1} \ldots B_{iw} = B_2$	$\beta^{ll} = b_2 b_2$
$B_4 = (b_2 b_1 b_2)(b_2)$	$\beta^l = b_2 b_1 b_2$	$B_{i1} \ldots b_{iw} = \Lambda$	$\beta^{ll} = b_2$
$B_5 = (b_2)(\ b_2 b_2 b_2)$	$\beta^l = b_2$	$B_{i1} \ldots b_{iw} = \Lambda$	$\beta^{ll} = b_2 b_2 b_2$
$B_5 = (b_2 b_2)(\ b_2 b_2)$	$\beta^l = b_2 b_2$	$B_{i1} \ldots b_{iw} = \Lambda$	$\beta^{ll} = b_2 b_2$
$B_5 = (b_2 b_2 b_2)(\ b_2)$	$\beta^l = b_2 b_2 b_2$	$B_{i1} \ldots b_{iw} = \Lambda$	$\beta^{ll} = b_2$

Tablo 3.4.

Yazılan tüm ayrılışlara uygun olarak;

$K_0 = \{\ \Lambda\ ,\ b_2\ ,\ b_2 b_2\ ,\ b_2 b_2 b_2\ \}$ olarak bulunur.

K_0 ifadesine uygun elementer kodlar aşağıdaki gibidir.

$B_2 = (b_2)\ B_1\ \Lambda$

$B_3 = \Lambda\ B_1\ (b_2 b_2)$

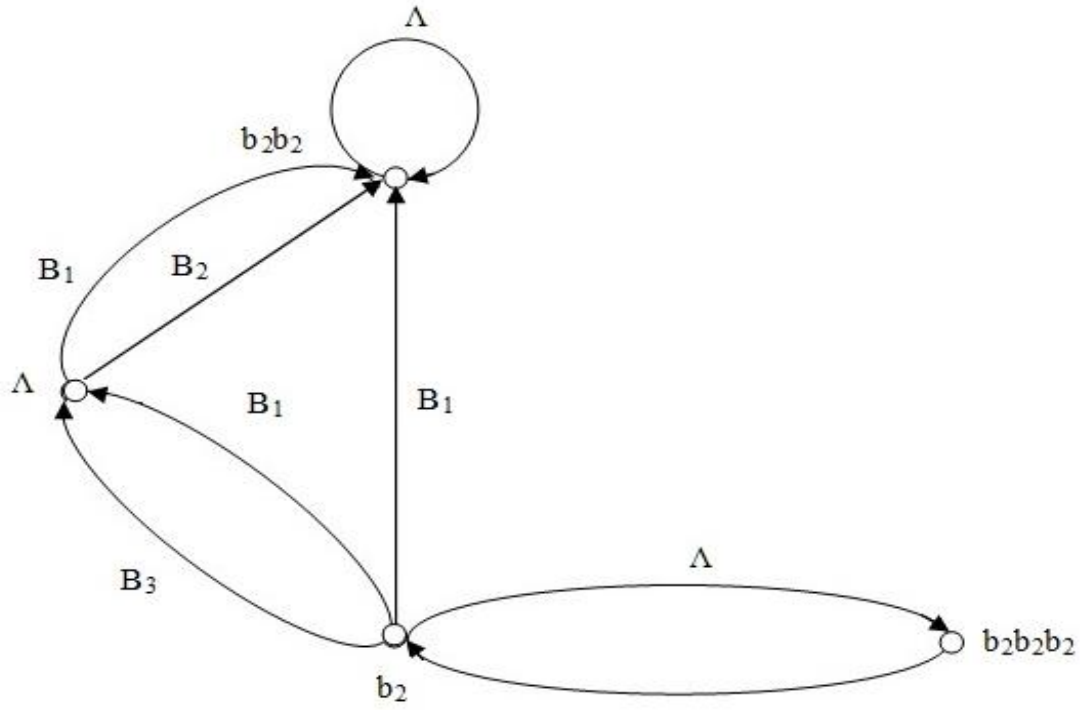
$B_4 = (b_2)\ B_1\ (b_2 b_2)$

$B_4 = (b_2)\ B_3\ \Lambda$

$B_4 = \Lambda\ B_2\ (b_2 b_2)$

$B_5 = (b_2)(\ b_2 b_2 b_2) = (b_2 b_2)(\ b_2 b_2) = (b_2 b_2 b_2)\ (b_2)$

Algoritmaya uygun yazılan elementer kodların grafı çizilerek dekodlaştırmanın mümkün olup olmadığı araştırılır.



Şekil 3.5

A'den geçen yönlü döngü içermeyen Γ(∑) grafi elde edilir. Bu yüzden ∑ şeması ile verilen alfabe kodlaştırması karşılıklı tek değerlidir. Yani dekodlaştırma mümkündür.

BÖLÜM 4
KENDİ KENDİNİ DÜZELTEBİLEN KODLAR

Bu bölümde, hata düzeltme kodlarının özel bir durumunu inceleyeceğiz.

$\mu = \{ 0, 1\}$ iki sembolden oluşan bir alfabe olsun. Ayrıca, $\{ A_1, A_2, \dots , A_s \}$ μ'de bulunan m sabit uzunluğunda $A = \alpha_1 \dots \alpha_m$ şeklindeki tüm kelimelerin oluşturduğu küme olsun. Burada

$s = 2^m$'dir.

Farz edelim ki, iletişim kanalında engelleyici bir kaynak çalışsın. Böyle bir durumda, bu kaynak uzunluğu yaklaşık olarak m olan $\{ A_1, A_2, \dots A_s \}$ kümesinden seçilmiş p tane kelimeden daha fazla hataya sebep olamaz. Buda kanalın çıkışında alınan ikili dizilimin, girişindekinden, p'den daha fazla yerde farklı olamayacağı anlamına gelir. Çok açıktır ki ön elemeler kodlaştırılması yapılmadan $\alpha_1 \dots \alpha_m$ şeklinde orijinal bir mesaj gönderildiği taktirde kanal çıkışında, hangi mesajın yollandığını belirlemek imkansızdır. O halde bu soru ortaya çıkar.

$\{ A_1, A_2, \dots , A_s \}$ kümesinden alınan $\alpha_1 \dots \alpha_m$ şeklinde olan A kelimeleri ℓ uzunluklu $\beta_1\beta_2 \dots \beta_\ell$ kelimeleri ile kodlanır ve kanalın çıkışında elde edilen $\beta_1\beta_2 \dots \beta_\ell$ 'ni ileten $\beta'_1\beta'_2 \dots \beta_\ell$ yardımıyla esas kelime olan $\alpha_1 \dots \alpha_m$ elde edilebilir mi?

Böyle kodlara kendi kendini düzeltebilen kodlar denir.

Burada önemsiz bir çözümün var olduğunu görmek oldukça basittir. Bunu engelliyici kaynağın 1 (p = 1) olduğu durumlar için kabul edebiliriz. Değişiklik sadece

$0 \rightarrow 1$ veya

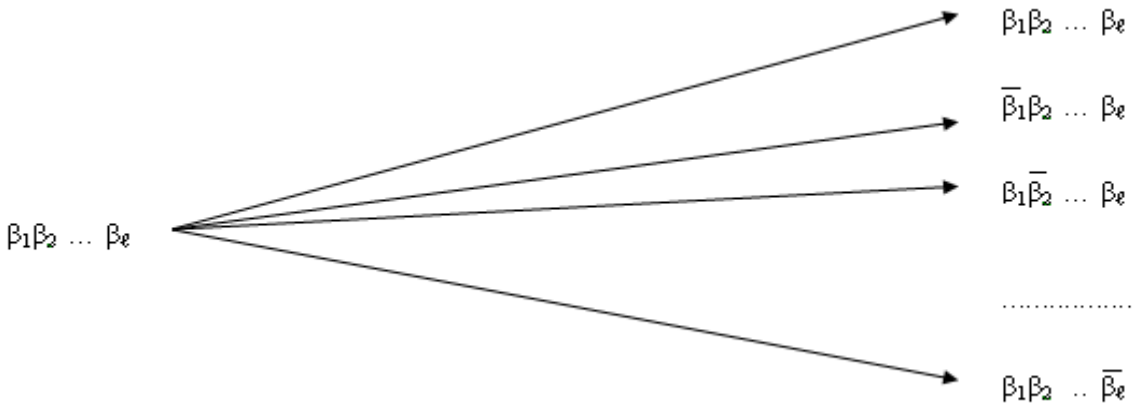
$1 \rightarrow 0$ ise mümkündür.

Kendi kendini düzeltebilen kod orijinal kodunun üçlemesi ile elde edilir.

$$\alpha_1 \alpha_2 \dots \alpha_m - \alpha_1 \alpha_1 \alpha_1 \alpha_2 \alpha_2 \alpha_2 \dots \alpha_m \alpha_m \alpha_m$$

Etkileme; Eğer kelime kodu gönderilirken bir hata meydana gelirse, daha sonra $\alpha_i \alpha_i \alpha_i$ grubu içindeki bir sembol bozulur (değişir) ve tüm diğer semboller hatalı gönderilir. Bu bize " ballot " metodu örneğini kullanılarak hatayı düzeltme, kodu yeniden yapma ($\alpha_1 \alpha_1 \alpha_1 \dots \alpha_m \alpha_m \alpha_m$) ve bundan dolayı orijinal mesajın ($\alpha_1 \dots \alpha_m$) elde etme olanağı sağlar.

(input) Girdi (output)Çıktı



Şekil 4.1.

Burada kod uzunluğu $\ell = 3m$ olmadan önce, üçlü çözüm uygulanmak doğru değildir ve kendi kendini düzeltebilen kodlar p' den daha fazla hataya neden olur. Bu durumdan sonra orijinal mesajı tekil olarak yeniden yapmak her zaman mümkün olmayabilir.

Kendi kendini düzeltebilen kodların doğru yapısı p = 1 durumunu ayrıntılarıyla inceleyen R. Hamming tarafından yapılan şu an kabul ettiğimiz yapıdır.

ℓ'nin kod uzunluğu ve $\ell = m + k$ olduğu durumda $\alpha_1\ \alpha_2 \ldots \alpha_m$ mesajları

$\beta_1\beta_2 \ldots \beta_\ell$

ifadeleri tarafından şifrelenir. Verilen engelleyici kaynak ile, çıkıştaki kodları elde etmek için olası yollar Şekil 4.1' deki gibidir.

Bu sebepten dolayı yol sayısı $\ell + 1$ 'dır. Eğer $\beta_1\beta_2 \ldots \beta_\ell$ kodu içinde kelime iletimi sırasında $\ell + 1$ tane durumu kodlamak için, yeterli sayıda koruyucu basamak olmasını istersek;

$2^k \geq \ell+1$ veya $2^m \leq 2^\ell / (\ell + 1)$ koşulu sağlanmalıdır.

ℓ, $2^m \leq 2^\ell / (\ell + 1)$ eşitsizliğini sağlayan en küçük tam sayı olarak seçilir

İleri üretim yapısı üç aşamada yerine getirilir.

4.1. Hamming Code Yapısı İçin Kodlaştırma Algoritmanın Tanımlanması

(1, 2, … , ℓ) doğal ölçülü cetveli k parçaya şu şekilde ayırırız:

V, $1 \leq V \leq \ell$ koşulunu sağlayan doğal bir sayı ve $V_k \ldots V_1$ binary ifade olsun.

1, 3, 5, 7, 9, … sırası $V_1 = 1$ olan bütün V'leri,

2, 3, 6, 7, 10, … sırası $V_2 = 1$ olan bütün V'leri,

4, 5, 6, 7, 12, … sırası $V_3 = 1$ olan bütün V'leri,

……

2^{k-1}, $2^{k-1} + 1$, …….. sırası $V_k = 1$ olan bütün V'leri kapsar.

Bunlardan ilk terimler, $2^0 = 1$, $2^1 = 1$, … , 2^{k-1} dir. Burada $2^{k-1} \leq \ell$ ve $2^k \geq \ell + 1$ koşulunu sağlayan 2'nin kuvvetleri şeklindeki sayılardır.

(1, 2, … , 2^{k-1}) kümesine dahil olan ℓ- indisli $\beta_1\beta_2 \ldots \beta_\ell$ kodundaki β_i terimleri kontrol basamakları olarak adlandırırlar ve bilgi basamağı olarak kalırlar. Buradaki k kontrolünü görmek çok kolaydır ve $\ell - k = m$ bilgi basamağıdır.

Şimdi ℓ- indisli $\beta_1\beta_2 \ldots \beta_\ell$ ve m- indisli $\alpha_1\ \alpha_2 \ldots \alpha_m$ için bir kural ifade edelim. İlk olarak bilgi basamaklarını belirleyelim.

$\beta_3 - \alpha_1$,

$\beta_5 - \alpha_2$,

$\beta_6 - \alpha_3$ vb.

Bilgi basamakları indisi normal düzen içinde $\alpha_1\ \alpha_2 \ldots \alpha_m$ 'e karşılık gelir.

Daha sonra kontrol sayılarını belirleyelim.

$\beta_1 = \beta_3 + \beta_5 + \beta_7 + \ldots$ (mod 2)

$\beta_2 = \beta_3 + \beta_6 + \beta_7 + \ldots$ (mod 2)

$\beta_4 = \beta_5 + \beta_6 + \beta_7 + \ldots$ (mod 2) vb. (4.1)

Çok açıktır ki, sağ taraftaki ifadeler daha önce belirlediğimiz bilgi basamaklarıdır.

$H^{|}_{\ell}$ tüm $\beta_1\beta_2 \dots \beta_\ell$ kodlarını kapsasın. Yani $\beta_1\beta_2 \dots \beta_\ell \subseteq H^{|}_{\ell}$ olsun.

4.2. Hamming Kodlarıyla Hata Bulma

$\beta_1\beta_2 \dots \beta_\ell \in H^{|}_{\ell}$ olsun ve $\beta_1\beta_2 \dots \beta_\ell$ kelime kodunu gönderirken S. terimde hata meydana gelsin. Sonra kelime ; $\beta^{|}_1\ \beta^{|}_2 \dots \beta^{|}_\ell = \beta_1 \dots \bar{\beta}_s \dots \beta_\ell$ kanal çıkışında bulunsun. $\beta^{|}_1\beta^{|}_2 \dots \beta^{|}_\ell$ kodu tarafından S'nin nasıl belirlendiğini gösterelim.

$S^{|}_1 = \beta^{|}_1 + \beta^{|}_3 + \beta^{|}_5 + \beta^{|}_7 + \dots$ (1. sıra)

$S^{|}_1 = \beta^{|}_2 + \beta^{|}_3 + \beta^{|}_6 + \beta^{|}_7 + \dots$ (2 sıra)

$S^{|}_1 = \beta^{|}_4 + \beta^{|}_5 + \beta^{|}_6 + \beta^{|}_7 + \dots$ (3. sıra) vb. olduğu durumda $S^{|} = S^{|}_k \dots S^{|}_1$ sayısını dikkate alalım.

$S = S^{|}$ olduğunu doğrulayalım. Gerçekten,

Eğer $S_1 = 0$ ise, o zaman S ilk sıraya dahil değildir ve
$\beta^{|}_1 + \beta^{|}_3 + \beta^{|}_5 + \beta^{|}_7 + \dots = \beta_1 + \beta_3 + \beta_5 + \beta_7 + \dots = 0$ dır. Bu yüzden $S^{|}_1 = 0$ olur…

$S_1 = S^{|}$

Eğer $S_1 = 1$ ise, o zaman S ilk sıraya dahildir. Ayrıca
$\beta^{|}_1 + \beta^{|}_3 + \beta^{|}_5 + \beta^{|}_7 + \dots = \beta_1 + \beta_3 + \beta_5 + \beta_7 + \dots = 1$ dir. Bu yüzden $S^{|}_1 = 1$ olur …
$S_1 = S^{|}_1$ 'dir.

Benzer şekilde $S_2 = S^{|}_2, \dots, S_k = S^{|}_k$ olduğu ispatlanır. Buradan $S = S^{|}$ olur.

Eğer kodu göndermede bir hata olmadıysa o zaman $S^{|} = 0$' dır. Bu yüzden $\bar{\beta}^{|}_s$ yerine $\beta^{|}_s$ koymak, $S^{|}$ değerinden bir hata olup olmadığını anlamamıza ve eğer hata varsa S içindeki terimin sayısını bulup, karmaşıklığı düzeltmeye, hangi durumda hatayı düzelttiğimizi öğrenmemize olanak sağlar.

4.3. Kodu Çözme (Dekodlaştırma)

Kod çözme, $\beta_1\beta_2 \ldots \beta_\ell$ kodu içindeki $\alpha_1\ \alpha_2 \ldots \alpha_m$ orijinal mesajının yapısından bilgi basamakları basamaklarından hangisinin yeterli olacağını seçmektir.

Örnek:

m = 4 için kendi kendini düzelten kodu yapalım.

Bu $2^4 \le \frac{2^l}{l+1}$ eşitsizliğinde ℓ'yi sağlayan en küçük sayı 7'dir.

O zaman k = 3'tür. Hamming kod yapısına uygun olarak kontrol basamaklarının * ile işaretlendiği yerde Tablo 4.1'e uygun kendini düzelten kodlardan birini ele alırız.

3., 5., 6. ve 7. sütunlardaki bilgi basamakları ilk olarak dörtlü 0000, … , 1111 şeklinde aşağı doğru doldurulur.

1*	2*	3	4*	5	6	7	1*	2*	3	4*	5	6	7
0	0	0	0	0	0	0	1	1	1	0	0	0	0
1	1	0	1	0	0	1	0	0	1	1	0	0	1
0	1	0	1	0	1	0	1	0	1	1	0	1	0
1	0	0	0	0	1	1	0	1	1	0	0	1	1
1	0	0	1	1	0	0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1	0	1	0	1	0	1
1	1	0	0	1	1	0	0	0	1	0	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1

Tablo 4.1.

1., 2. ve 4. sütunlar aşağıdaki formüllere göre doldurulur.

$$\beta_1 = \beta_3 + \beta_5 + \beta_7 + \ldots \text{ (mod 2)}$$

$$\beta_2 = \beta_3 + \beta_6 + \beta_7 + \ldots \text{ (mod 2)}$$

$$\beta_4 = \beta_5 + \beta_6 + \beta_7 + \ldots \text{ (mod 2)} \qquad (4.2)$$

Kanal girişinde kodu 0110011 olarak girelim ve karışıklık sonucu S = 5. terim değişmiş olsun. Sonra çıkışta 0110111olurarak bulunsun. Bu durumda hatalı terimin numarasını hesaplayalım.

$S'_1 = \beta'_1 + \beta'_3 + \beta'_5 + \beta'_7 = 0 + 1 + 1 + 1 = 1$

$S'_1 = \beta'_2 + \beta'_3 + \beta'_6 + \beta'_7 = 1 + 1 + 1 + 1 = 0$

$S'_1 = \beta'_4 + \beta'_5 + \beta'_6 + \beta'_7 = 0 + 1 + 1 + 1 = 1$

Bu durumda $S' = 101 = 5$'tir. Hatalı terim bulunur ve $S' = S = 5$ olur.

Sonuç olarak, geometrik özellikleri Hamming Kodu üzerine yerleştiririz.

ℓ- boyutlu küpü sınırlı bir uzay gibi düşünelim. Herhangi iki nokta $\beta' = (\beta'_1, \dots , \beta'_\ell)$ ve $\beta'' = (\beta'' , \dots , \beta''_\ell)$ için uzaklık ,

$p (\beta', \beta'') = \sum^{\ell}_{i=1} | \beta'_i - \beta''_i |$ ile hesaplanır.

Çok açıktır ki, β' ve β'' ifadelerinin farklı olduğu halde $p(\beta', \beta'')$ koordinatların sayısıdır.

Teorem 4.1: $\beta' \neq \beta''$ ve $\beta', \beta'' \in H^1_\ell$ olmak üzere β' ve β'' gibi herhangi iki kod için $p(\beta', \beta'') \geq 3$'tür.

İspat :

a) $p(\beta', \beta'') = 1$

b) $p(\beta', \beta'') = 2$

Eğer (a) ve (b) durumları hariç tutulursa ifade ispat edilir. Eğer $p (\beta', \beta'') = 1$ olursa, o zaman β''' sembolü $p (\beta', \beta''') = p (\beta''', \beta'') = 1$ … vb. gibi var olur. β', β'' 'nü β''' 'ne azaltmada tek bir hata mümkündür.

Bu yüzden her iki durumda da kanal çıkışında β' ve β'' den hangisinin gerçekten gönderdiğini saptamak olanaksızdır. İspat edilen teori, kendi kendini düzelten kod, H^1_ℓ kodunun tersidir.

β^o sembolü ℓ-boyutlu bir küpün bir noktası olsun.

Tanım 4.1: $p(\beta^{\circ}, \beta) \leq p$ gibi β noktaların $U^{p}_{\ell}(\beta^{\circ})$ toplamı β° merkezli küre olarak adlandırılır ve yarıçapı p'dir.

Tanım 4.2: $p(\beta^{\circ}, \beta) = p$ gibi β noktaların $V^{p}_{\ell}(\beta^{\circ})$ toplamı β° merkezli küre olarak adlandırılır ve yarıçapı p'dir.

Çok acıktır ki eğer β°'ın kod ise, o zaman iletişim kanalı tarafından yollanan karşı kaynak p hatalarından daha fazla hata içermez ve β°, $p(\beta^{\circ}, \beta) \leq p$ vb. gibi bir β noktası taşınırsa β°, β merkezli $U^{p}_{\ell}(\beta^{\circ})$ kümesi dahildir ve yarıçapı p'dir.

Buradan aşağıdakiler saptanır.

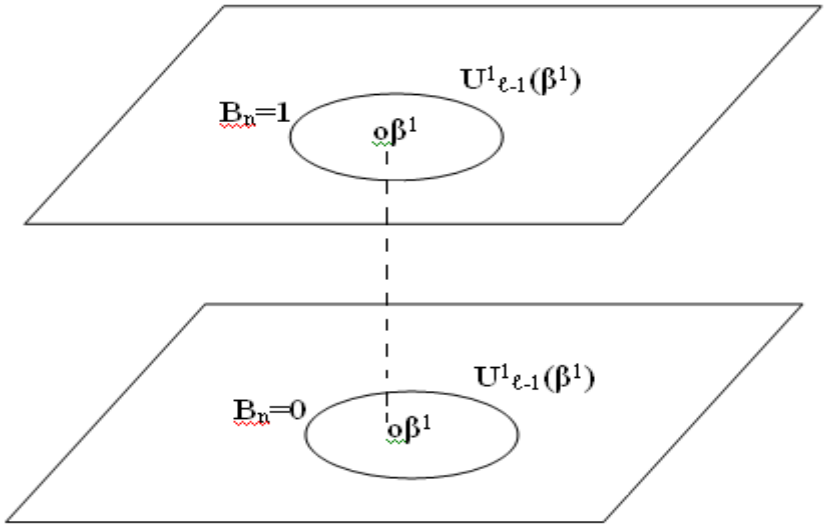
Teorem 4.2: H'nin, ℓ-boyutlu küpün kendi kendini düzelten kodu için alt kümesi p'den daha fazla hataya neden olmaz. Bu durum için,

$p(\beta^{\circ}, \beta) \geq 2p + 1$ ve herhangi $\beta^{|}$, $\beta^{||}$ için $\beta^{|} \neq \beta^{||}$ olması gerekli ve yerli koşuldur.

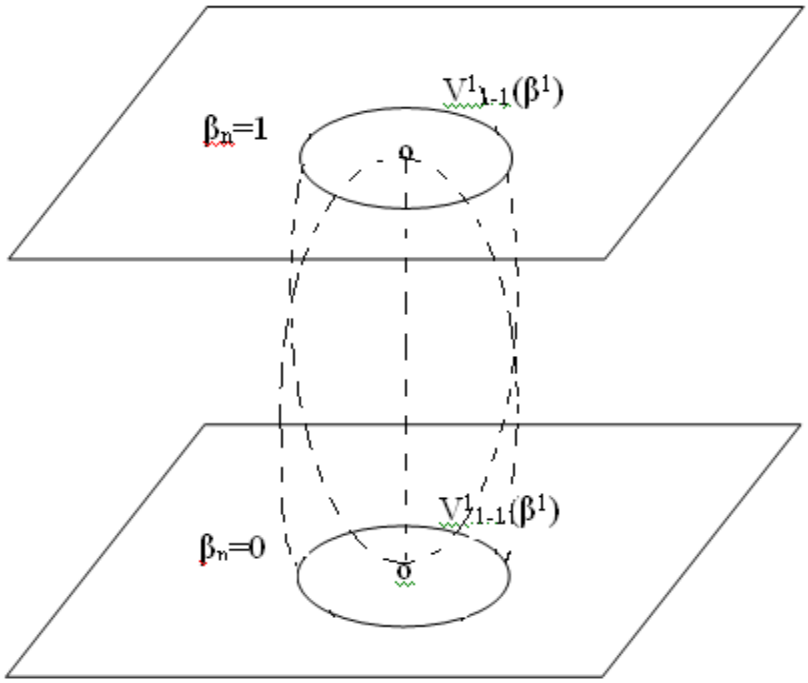
Teoremin ispatı için, yarıçapı p olan $\beta^{|}$ ve $\beta^{||}$ merkezli $U^{p}_{\ell}(\beta^{|})$ ve $U^{p}_{\ell}(\beta^{||})$ kümelerinin merkezlerinin eşit olması gerekir. İletişim kanalın çıkışında bulunan kod sadece bir kümeye ait olan noktaya uyar.

Teorem kendi kendini düzelten kod yapısına geometrik olarak yaklaşır ve genellikle aşağıdaki ifadelerle ilgilidir.

Teorem 4.3: (a) $\ell = 2^{t} - 1$ için, ℓ-boyutlu küp birimi, birim toplarının toplamına doğrudan ayrılır.



Şekil 4.2.



Şekil 4.3.

(b) $\ell=2^t$ için, ℓ-boyutlu küp,küre birimlerinin doğrudan toplamına ayrılır.

İspat :

(a) ℓ-boyutlu küp içinde Hamming kodunun H^l_ℓ alalım ve $\ell = 2^t - 1$ olsun.

Çok açıktır ki ; k = t, $m = 2^t - t - 1$

H^l_ℓ 'den her nokta için yarıçapı 1 (bir) olan kümeyi daire içine alalım ve gerekli ayırmayı belirten bunun gibi kürelerin setini gösterelim.

(Teorem 4.1'den) Bu setteki noktaların toplam sayısı;

$$(\ell + 1)\, 2^m = 2^t\, 2^{2t-t-1} = 2^t\, 2^{2t-1} = 2^\ell \text{ dir.}$$

Bu yüzden ℓ-boyutlu küpün bütün noktalarını kapsar.

(b) $\ell = 2t$ olsun. Son sabit koordinat, doğal bire-bir örten $\beta^\circ \leftrightarrow B^l$ var Olduğunda ve $\beta^\circ = (\beta_1 , \dots , \beta_{\ell-1}, 0)$, $\beta^1 = (\beta_1 , \dots , \beta_{\ell-1}, 1)$ olduğu durumda ; ℓ-boyutlu küpü $(\ell - 1)$ – boyutlu kübe ayırır.

$\ell - 1 = 2^t - 1$ ifadesinden dolayı, (ℓ- 1)-boyutlu küp (a)'nın yanında küplerin doğrudan toplamı ile ayrılır. (ℓ- 1)-boyutlu küpün içinden doğru toplanmış küplerden bir bölüm seçelim.

Küpler arasında doğal bire-bir örten özelliğini kullanabilmek için diğerleri için de ayırma yapabilir. Uygun kürelerden iki küreyi $U^1_{\ell-1}(\beta^\circ)$ ve $U^1_{\ell-1}(\beta^1)$ olarak düşünelim (şekil 4.2).

(ℓ- 1)-boyutlu iki küre ℓ-boyutlu iki küreye dönüşebilir (Şekil 4.3).

Doğrusu (ℓ- 1)-boyutlu toplar, (ℓ- 1)-boyutlu kürelerin birleşimidir ve bunların merkezleri;

$$U^1_{\ell-1}(\beta^\circ) = V^1_{\ell-1}(\beta^\circ) \cup \{\beta^\circ\}$$

$U^1_{\ell-1}(\beta^1) = V^1_{\ell-1}(\beta^1) \cup \{\beta^1\}$ olur.

Çok açıktır ki, eşitsizlikler

$$V^1_{\ell-1}(\beta^\circ) = V^1_{\ell-1}(\beta^\circ) \cup \{\beta^\circ\}$$

$V^1_{\ell-1}(\beta^1) = V^1_{\ell-1}(\beta^1) \cup \{\beta^1\}$ ℓ-boyutlu kürelerini kapsar.

Bunun için, eğer bu dönüşüm top çiftlerine uyan bölüm için yapılmışsa, o zaman kürelerin direkt toplamı için gereken küp bölümünü alırız. Böylece sağlama tamamlanmış olur.

Tanım 4.3: ℓ-boyutlu birim küp içinde bir kod seti verilen engelleyici kaynağa rağmen kendi kendini düzelten kod ise ve eğer bu en büyük kuvveti ise bu maksimum demektir.

Sonuç : $\ell = 2^\ell - 1$ için, Hamming kodu H^l_ℓ maksimumdur.

BÖLÜM 5
GENERATOR MATRİSLER (ÜRETEÇ MATRİSLER)

Bu bölümde kodların içindeki dizilerin sabit uzunluğuna ‘n’ diyeceğiz ve bu dizileri vektör veya 1xn matrisler olarak düşüneceğiz. Böylelikle vektörleri toplayacağız. Toplamı mod 2’de tanımlarsak 1 + 1 = 0 olacak. Böylece,

$$11110001 + 10100111 = 01010110$$

Hesapladığımız bu kodlara “linear codes” denir. Bu kodlar ayrıca “group codes” olarak da bilinmektedir. Daha önce belirttiğimiz gibi Binary sistemde uzunluğu n olan ‘C’ kodların vektör veya 1xn matris olarak düşünebiliriz. Uzunluğu ‘n’ olan tüm ikili sistemlerin kümesi B_n ise, C, B_n’nin alt kümesidir. B_n önceki toplamlarla kanıtlanmak istenirse grupların alt alta toplanmasıyla elde edilir. B_n’deki herhangi dizi kendisinin tersidir, bu nedenle aynı şeyler elenip çıkarılabilir. Buradan anlayacağımız gibi eklemek yerine çıkarmak da yapsak değişen bir şey olmaz. C kodu B_n’nin alt grubuysa “linear code”dur. Eğer lineer cebirde alırsak lineer cebirin bir çok özelliğinden dolayı C’nin gerçekten lineer vektör uzayı olduğunu inceleyebiliriz. Bu özellikler bize C’nin bir grup olduğunu, C’nin elemanlarının, vektör olduğunu ayrıca uygun boyutlardaki matrislerle çarpılabileceğini gösterir. Ayrıca matrislerin dağılma özelliğini kullanabiliriz. Bütün matrislerde,

$$A (B + C) = AB + AC$$

Şeklinde A’ nın çarpımının B ve C toplamı üzerine dağılmasını tanımlar. Eğer $u = (u_1, u_2, u_3, \ldots u_n)$ ve $v = (v_1, v_2, v_3, \ldots v_n)$ ve u ve v’nin iç çarpımını u.v ile gösterirsek, bunun eşiti,

$$u_1 v_1 + u_2 v_2 + u_3 v_3 + \ldots + u_n v_n \qquad (5.1)$$

olur.

Wt (C) ile gösterilen kod dizisinin ağırlı dizideki 1’lerin sayısı kadardır. Örneğin; C = 1011010 ise W t (C) = 4’tür.

kxn G matrisimiz olsun. İlk k satır ve sütunları kxk birim matris formunda ve her sütunu farklı olsun. O zaman $G = [\, I_k \mid A_{n-k} \,]$ şeklinde gösterilir. Örneğin;

$$\begin{bmatrix} 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1 \end{bmatrix}$$

Bu tarz bir matristir. Bu matrise "generator matrix" (üreteç matris) denir. Üreteç matrisin satırlarını vektör veya kodun dizileri olarak sayarsak dizi kümesine 'S' denir. Örneğin bu matriste

$$S = \{\, 100101, 010110, 001011 \,\}\text{'dir.}$$

C 'den alınan tüm vektörler S'in sonlu toplam dizileridir.

C'nin B_n'nin alt grubu olduğunu kanıtlayacak olursak, bizim örneğimizde S'teki ilk iki dizinin toplanmasıyla oluşan 110011'i alabiliriz, bu nedenle 110011 C'de de olabilir. C grubu, S kümesinden üretilmiştir. Bu ayrıca C'den üretilmiş en küçük kümedir. Çünkü S'in hiçbir elemanı S'in diğer elemanlarının toplamı değildir. Bunu $C = S^*$ şeklinde gösteriyoruz. $[\, I_k \mid A_{n-k} \,]$ formundaki "[n, k] – code" denir.

Teorem 5.1 : [n , k] – kod C kodu 2^k dizi içerir.

İspat: C dizisindeki ilk k bitleri C'nin elemanlarını belirler. C dizisindeki ilk k bitlerindeki 1'lerin pozisyonu, S'deki hangi dizilerin eklendiğini belirtir. Örneğin; C dizisinin birinci ve üçüncü bitlerinde 1 ortaya çıksın, sonra bu dizi G'nin 1.ve 3. satırları eklenerek üretilsin. Buradan ilk k bitlerini biçimlendiren 2^k farklı yol vardır, C'nin 2^k dizisi vardır.

Dizilerin uzunluğu k olan mesajı göndermek istersek, bunları sağdan G ile çarpıp kodlarız. Buradan $w = w_1, w_2, w_3 \ldots$ veya $(w_1, w_2, w_3 \ldots)$ ise diziyi wG şeklinde kodlarız.

Örneğimizde, 110 veya (1, 1, 0)'i

$$(1, 1, 0) \begin{bmatrix} 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1 \end{bmatrix} = (1, 1, 0, 0, 1, 1) \text{ veya}$$

110011 şeklinde kodlayabiliriz. Mesaj dizisi kodlanmış dizinin ilk üç bitidir. Genellersek k uzunluğundaki mesaj dizisi kodlanmış dizinin ilk k biti olur. Çünkü G'nin ilk k sütunu olan I_k birim matris sadece orijinal dizide tekrar eder. Böylece kodlamış dizinin ilk k bitini alarak kodu kolayca çözebiliriz.

(1, 1, 0)'i önceki gibi G'ye çarptığımızda

(1, 1, 0, 0, 1, 1,) = 1.(1, 0, 0, 1, 0, 1) + 1.(0, 1, 0, 1, 1, 0) + 0.(0, 0, 1, 0, 1, 1)

şeklini alır. Herhangi bir v vektöründe; 1.v = v ve $0.\vartheta$ = (0, 0, 0, 0, 0,0) olur. Bu nedenle kodlanmış diziler S'deki vektörün toplamıdır ve ayrıca C'dedir çünkü C gruptur. Genellersek, üretici matrisin satırlarının kümesi S = (s_1, s_2, s_3, …) ve mesaj kodu ω = (ω_1, ω_2, ω_3 …)'tir, böylece kodlanmış dizi

$$\omega_1 s_1 + \omega_2 s_2 + \omega_3 s_3 + \ldots + \omega_k s_k \text{ olur.}$$

S'deki dizilerin toplamı her ω_i ya 0 ya 1 olduğundan dolayı C'dedir. Çünkü C, S'ten üretilmiş bir gruptur.

G'yi [I_k | A_{n-k}] formunda, mesajı I_k biçiminde aldık.

Şöyle ki;

$$(\omega_1, \omega_2, \omega_3 \ldots) \begin{bmatrix} 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1 \end{bmatrix} = (\omega_1\ \omega_2,\ \omega_3\ \omega_1,\ \omega_2\ \omega_2,\ + \omega_2\ \omega_2,\ \omega_3\ \omega_1,\ \omega_1\ \omega_2)$$

Bu nedenle kodlanmış dizinin 4. biti $\omega_1 + \omega_2$, 5. biti $\omega_2 + \omega_3$, 6. biti $\omega_1 + \omega_3$, olmalıdır. (ω_1, ω_2, ω_3, ω_4, ω_5, ω_6,) dizisini oluşturur. Eğer gönderildikten sonra kodlanmış dizi eşitliği sağlamıyorsa, göndermede hata olduğunu biliriz. Örneğin; dizimizi 101100 alalım; ω_1 = 1 ω_2 = 0 ve ω_3 = 1 olduğu için ω_4 = 1 = 0 + 1 ,

ω_5=0=0+1 ve $\omega_6 = 0 = 1 + 1$ olmalı ω_5'in sonucu doğru olmadığı için hata oluştuğunu biliriz. Böylece A_{n-k} matrisi göndermenin doğruluğunu parity kontrol kadar çabuk kontrol etmemizi sağlar. Genelleme yaparsak, kodlanmış dizilerimiz $\omega_1, \omega_2, \omega_3, \ldots \omega_k \ldots \omega_n$ ve

$$G = \begin{bmatrix} 1 & 0 & 0 \ldots & 0 & A_{1,k+1} & A_{1,k+2} & \ldots & A_{1,n} \\ 0 & 1 & 0 \ldots & 0 & A_{2,k+1} & A_{2,k+2} & \ldots & A_{2,n} \\ 0 & 0 & 1 \ldots & . & A_{3,k+1} & A_{3,k+2} & \ldots & A_{3,n} \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & 0 & . & . & . & . \\ 0 & 0 & 0\ 0 & 1 & A_{k,k+1} & A_{k,k+2} & & A_{k,n} \end{bmatrix}$$

i > k için $\omega_i = \omega_1 A_{1,i} + \omega_2 A_{2,i} + \ldots \omega_k A_{k,i}$ olmalı ve n – k denkliğini sağlamalıdır.

Diğer problemimiz oluşan tek hatayı düzeltmektir. Aşağıdaki yöntem "coset leaders" kullanılarak bilinmektedir. Bu metodu bizim örneğimiz de gösterelim;

$$G = \begin{bmatrix} 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1 \end{bmatrix}$$

S = { 100101, 010110, 001011 } olduğu için

C = (000000, 100101, 010110, 001011, 110011, 011101, 101110, 111000)'dir.

İlk koset C'nin kendisidir. Diğer koseti, B_n 'nin uygun ağırlıktaki ve C'de bulunmayan elemanlarından seçeriz.

Örneğin;

b_1 = 100000'i seçebiliriz.

Koset,

b_1 + C = (100000, 000101, 110110, 101011, 010011, 111101, 001110, 011000) olur.

Tekrar B_n'den uygun ağırlıktaki ve diğer kosetlerde de olmayan eleman seçeriz. Örneğin;

b_2 = 010000'ı seçebiliriz.

Koset,

$b_2 + C$ = (010000, 110101, 000110, 011011, 100011, 001101, 111110, 101000) olur.

Bu sistemde devam ettiğimizde B_n'nin kosetlere bölünmüş tablosunu elde ederiz. Ağırlığı az olan elemanlar önce listelenir. İlk sütundaki elamanlara "coset leaders" denir.

000000	100101	010110	001011	110011	011101	101110	111000
100000	000101	110110	101011	010011	111101	001110	011000
010000	110101	000110	011011	100011	001101	111110	101000
001000	101101	011110	000011	111011	010101	100110	110000
000100	100001	010010	001111	110111	011001	101010	111100
000010	100111	010100	001001	110001	011111	101100	111010
000001	100100	010111	001010	110010	011100	101111	111001
100010	000111	110100	101001	010001	111111	001100	011010

Tablo 5.1

Son kosette ağırlığı 2 olan dizi almalıyız. 100010, 010001 veya 001100'ı seçebiliriz. Açıkçası, bir tane bile bulabilmek sevindirici. Süreç şöyle işliyor: 110110'u seçtiğimizi varsayalım bunu tabloda buluyoruz. 110110'u içeren satırın en başına bakıp 100000'i buluyoruz. Böylece hatanın ilk bitte oluştuğunu anlıyoruz. Çünkü 100000 C'nin herhangi bir elemanına eklenerek bu koset elde edilmiştir. 110110'u içeren sütunun en başına baktığımızda 010110'u buluyoruz. Bu C'nin dizisinin doğrusudur. Yine 001010 dizisini alalım. 001010'un bulunduğu satırın başı 000001'i buldurur. Bu 6. bitte hata olduğunu gösterir. 001010'un bulunduğu sütunun en başına bakarsak 001011'i buluruz. Bu C'nin doğru dizisidir.

Şimdi hata belirlemenin daha kolay bir yoluna bakalım, n uzunluğundaki v ve ω vektörlerimiz (veya dizi) olsun. Eğer nokta çarpımları v. ω = 0 ise v vektörü ω vektörünün "ortogonalidir". $C^{\perp}$ ile gösterilen C'nin "dual code"u C'nin bütün dizilerinin ortogonali olan B_n dizilerinin kümesidir. Bu $C^{\perp,}$ nin B'nin alt grubu olduğunu gösterir. Eğer C kodu [n,k]-kod ise k dizisinden üretilmiştir, o halde $C^{\perp}$ de n-k dizisinden elde edilmiş olur. Bunun ispatını lineer cebirin özelliklerini kullanarak yapabiliriz. Doğru olduğunu kabul ederek aşağıdaki teoremi verebiliriz.

Teorem 5.2: C bir grup kodu ve $C^{\perp}$, onun dual kodu olsun $C^{\perp,}$ de bulunan t dizisi, C'nin üreteçlerinin kümesi olan S'in her dizisinin ortogonalidir.

İspat: Açıkça, eğer t, $C^{\perp}$'de ise S'deki her diziye ortogonaldir. Bu nedenle C'deki ve $S \subseteq$ C'deki tüm dizilere ortogonaldir. Farzedelim $S=\{s_1, s_2, s_3, \ldots, s_k\}$ ve t, $s_i \in$ S'teki her s_i için ortogonal, buradan tüm $s_i \in S$ için $t \cdot s_1 = 0$ olur.

w_1'in 0 ve ya 1 olduğu yerde C'nin her elemanı, $w_1.s_1+w_2.s_2+w_3.s_3+\ldots+w_k.s_k$ formundan olur.

Matrisin lineer özelliğinden dolayı,

$$t\cdot(w_1.s_1+w_2.s_2+w_3.s_3+\ldots+w_k.s_k) = w_1(t\cdot s_1)+ w_2 (t\cdot s_2) +\ldots + w_k (t\cdot s_k)$$
$$= 0+ 0+ \ldots +0$$
$$= 0 \qquad (5.2)$$

Örneğimize geri dönersek;

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [\, I_3 \mid A_3 \,] \text{ şimdi bunu,}$$

$$G^{\perp} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = [\, A_3{}^t \mid I_3 \,] \text{şeklinde tanımlıyoruz}$$

$A_3{}^t$ A_3 'ün transpozudur ve $A_{3,}$ ün satırlarının sütuna çevrilmesiyle elde edilir. $G^{\perp}$ matrisine "parity check matrix" yani "parity kontrol matrisi"denir.

Örneğimizdeki deki bütün olasılıkları incelediğimizde, G'nin herhangi satırıyla $G^{\perp}$'nin herhangi satırının iç çarpımının 0'a eşit olduğunu görüyoruz. G'nin i. Satırının transpozu olan $r_i{}^t$'yle $G^{\perp} r_i{}^t = 0$ olduğunu biliyoruz. G'nin iç satırının $r_i{}^t$ transpozu sütuna çevrilerek tanımlanır. Bunu sütun konuma getirdiğimizde $G^{\perp}$ matrix'yle sağdan çarpabiliriz.

$$\begin{aligned} G^{\perp}(\omega_1 r_1{}^t + \omega_2 r_2{}^t + \omega_3 r_3{}^t) &= \omega_1 G^{\perp} r_1{}^t + \omega_2 G^{\perp} r_2{}^t + \omega_3 G^{\perp} r_3{}^t \\ &= 0 + 0 + 0 \\ &= 0 \end{aligned} \qquad (5.3)$$

Bu nedenle, $G^{\perp}$ ve C'nin transpozu alınmış herhangi elemanını çarptığımızda 0 elde ederiz.

Genellersek,

$$G = [\, I_k \mid A_{n-k} \,] = \begin{bmatrix} 1 & 0 & 0 \ldots & 0 & A_{1,k+1} & A_{1,k+2} & \ldots & A_{1,n} \\ 0 & 1 & 0 \ldots & 0 & A_{2,k+1} & A_{2,k+2} & \ldots & A_{2,n} \\ 0 & 0 & 1 \ldots & . & A_{3,k+1} & A_{3,k+2} & \ldots & A_{3,n} \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & . & . & . & . & . \\ . & . & . \ldots & 0 & . & . & . & . \\ 0 & 0 & 0\ 0 & 1 & A_{k,k+1} & A_{k,k+2} & & A_{k,n} \end{bmatrix} \text{ ise}$$

$$G^{\perp} = [\, A^t_{n-k} \mid I_{n-k} \,] = \begin{bmatrix} A_{1,k+1} & A_{2,k+1} & \ldots & A_{k,.k+1} & 1 & 0 & 0 & \ldots & 0 \\ A_{1,k+2} & A_{2,k+2} & \ldots & A_{k,.k+2} & 0 & 1 & 0 & \ldots & 0 \\ A_{1,k+3} & A_{2,k+3} & \ldots & A_{k,.k+3} & 0 & 0 & 1 & \ldots & . \\ . & . & \ldots & . & . & . & . \ldots & & . \\ . & . & \ldots & . & . & . & . & \ldots & . \\ . & . & \ldots & . & . & . & . & \ldots & 0 \\ A_{1,n} & A_{2,n} & \ldots & A_{k,.n} & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ olur.}$$

G'nin i. satırıyla $G^{\perp,}$ nın j. satırının iç çarpımın sonucu,

$0 + 0 + \ldots + 0 + A_i,j + 0 + \ldots + 0 + A_i,j + \ldots + 0 + 0 = 0$'dır. Çünkü G'nin i. Satırının transpozu olan $r_i{}^t$'yle $G^{\perp} r_i{}^t = 0$'dır. Bunu kullanarak, $G^{\perp}$ ile C'nin transpozu alınmış herhangi elemanını çarptığımızda 0 elde ederiz.

Ayrıca bir başka dikkat edilmesi gereken sonuca daha ulaşırız. Daha önceden yaptığımız gibi C grubunu kullanarak B_n'den oluşturduğumuz kosetlerden b_1 ve b_2 aynı kosetin elemanı olsun

$G^{\perp} b_1{}^t = G^{\perp} b_2{}^{t'}$ dir.

b_1 ve b_2 aynı kosetteyse, $c \in C$ için $b_1 = b_2 + c$'dir. Bu nedenle $b_1{}^t = b_2{}^t + c^t$ ve

$$G^{\perp} b_1{}^t = G^{\perp}(b_2{}^t + c^t)$$
$$= G^{\perp} b_2{}^t + G^{\perp} c^t$$
$$= G^{\perp} b_2{}^t + 0$$
$$= G^{\perp} b_2{}^t \text{ 'dir.} \qquad (5.4)$$

Çünkü bütün $c \in C$ için $G^{\perp} c^t = 0$'dır.

$G^{\perp}$ b^t kosetteki bütün b'lerde aynı olduğu için kosetten herhangi bir b seçip değerini hesaplayabiliriz. Bu nedenle $G^{\perp}$ imgesi olan bu ortak değeri önceki tabloda istediğimiz satıra yazabiliriz. Çünkü herhangi satırın elemanları koseti belirleyebilir. En sadeleri "baş kosetler" olduğu için onları seçtik ve imgelerinin değerini ikinci sütuna yerleştirdik. Bu değerlere "syndromes" (sendrom) denir ilk sendromun $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ olduğunu biliyoruz.

$$\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&1&0&0&1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1\\0\\1\end{bmatrix}$$ olduğunu bulduk, yani $\begin{bmatrix}1\\0\\1\end{bmatrix}$ ikinci sendromdur. Ve

$$\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&1&0&0&1\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1\\1\\0\end{bmatrix}$$ olduğundan $\begin{bmatrix}1\\1\\0\end{bmatrix}$ üçüncü sendromdur. Tablodaki

gibi devam ediyor …

000000	000	100101	010110	001011	110011	011101	101110	111000
100000	101	001010	110110	101011	010011	111101	001110	011000
010000	110	110101	000110	011011	100011	001101	111110	101000
001000	011	101101	011110	000011	111011	010101	100110	110000
000100	100	100001	010010	001111	110111	110111	011001	111100
000010	010	100111	010100	001001	110001	011111	101100	111010
000001	001	100100	010111	001010	110010	011100	101111	111001
100010	111	000111	110100	101001	010001	111111	001100	011010

Tablo 5.2.

Tabloyu tamamladıktan sonra gönderilmiş dizimizi 101100 aldığımızı varsayalım. Sonra transpozunu alıp $G^{\perp,}$ yla çarparak

$$\begin{bmatrix} 1&1&0&1&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&0&0&1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$ elde edelim. Böylece 101100'ün 6. satırıda

olduğunu biliyoruz. 101100'ün bulunduğu satırın en soldaki sütunun elemanı olan 000010 "baş kosettir". 101100'ün içeren sütunun en üstteki satırındaki C'nin elemanı 101110'dur. Bu yolla 101100 = 101110 + 000010'un tablonun neresinde yapıldığını biliyoruz. Böylece gönderilmiş mesaj olan 101100'ün 101110 olabileceğini varsayıp 5. bitte hata olduğunu söylüyoruz.

Bu yöntem sendrom içeren satırı ve gönderilmiş mesajı bulmakta daha hızlıdır. Çünkü gönderilmiş mesajın transpozuyla $G^{\perp}$ 'yi sadece çarpıyoruz. Bu mesajlar için baş kosetler hatadır ve gönderilmiş mesajı içeren sütunun ilk satırı olan C'nin elemanı düzeltilmiş mesajdır.

Bu işlemi daha çabuk yapabiliriz ve önceki tabloda sadece ilk iki sütuna ihtiyacımız var. Gönderilmiş mesajımızın 110000 olduğunu varsayalım. Transpozunu alıp $G^{\perp}$ ile çarptığımızda

$$\begin{bmatrix} 1&1&0&1&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&0&0&1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$ elde edelim. Yani sendrom $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$'dir ve baş koset 001000'dır.

Baş koset bize 3.bitte hata olduğunu söyler. Eğer 110000'a 001000 eklersek 111000 elde ederiz. Bu doğru koddur. Bu nedenle bizim yöntemimiz basittir. Sendromu bulmak için gönderilmiş mesajın transpozuyla $G^{\perp}$ 'yı çarp. Sendromdan baş koseti bul ve onu gönderilmiş mesaja ekleyerek doğru kodu elde et. Uyarı şu: tablonun sadece ilk iki sütunu kullanıyoruz.

Ama şöyle bir problemimiz var;

Gönderilmiş mesajımız 101001 olsun. 0 halde sendrom $\begin{bmatrix}1\\1\\1\end{bmatrix}$' dir. Uygun satırda ağırlığı 2 olan üç dizi var. 100010'un rastgele seçildiğini biliyoruz. Herhangi biri hata olabilir, bu nedenle sendrom kullanarak düzeltme yapmak işe yaramaz. Yine de bir değil de iki hatası olan dizileri düzeltmeye çalışıyoruz.

5.1. Hamming Kod Matrisleri

Geçen bölümün sonunda bazı dizilerin kodlarını düzeltmekte problem olduğunu gördük. Çünkü her baş kosetin ağılırlığı 1 değil. Bunların çözümü Hamming matris adı verilen üretici matrisler kullanmaktır. Hamming matrislere (G_H) bakmadan önce parity kontrol matrislere ($G_H^{\perp}$) bakalım $G_H^{\perp}$, 0'ların dışındaki r uzunluklu dizilerin tümü olasılıklarını içeren sütunlar gibi r satırılı matris olsun. $r \geq 3$ alalım $2^r - 1$ dizisi var. Bu nedenle $G_H^{\perp}$ $n = 2^r - 1$ olduğu için rxn matristir. Son r sütunlarında ağırlığı 1 olan birim matris formundaki sütunları kullanırız. Burada A^t'nın rx(n – r) matris olduğu yerde $G_H^{\perp}$ $[A^t \mid I_r]$ biçimindedir. A'nın (n – r) xr matrisi olduğu yerde G_H $[I_{n-r} \mid A]$ biçiminde (n – r)xn matristir. Hamming matrisin satırlarından üretilen kodlara "Hamming code" denir. Örneğin;

r = 3 alalım

Hamming matrisleri çalışmamız için iki dizi arasındaki uzaklık ve ağırlıkları arasındaki ilişki lazım. Ağırlıkla ilgili teoremle başlayalım:

Teorem 5.3: Dizilerimiz c ve $c^{|}$ ise

$\omega\, t\,(c + c^{|}) \leq \omega\, t\,(c) + \omega\, t\,(c^{|})$ dir.

İspat: $c = c_1\, c_2\, c_3 \ldots c_n$ ve $c^{|} = c^{|}_1\, c^{|}_2\, c^{|}_3 \ldots c^{|}_n$ olsun.

Eğer $c_i + c^{|}_i = 1$ ise ya $c_i = 1$ ya da $c^{|}_i = 1$'dir. Bu nedenle $c+c^{|}$'de 1'in oluştuğu her durum için c'de ya da $c^{|}$'de 1 bulunmalıdır.

Aynı uzunluktaki $c^{|}$ ve c dizilerinin arasındaki uzaklık ve ya Hamming uzaklığı birinde 1 diğerinde 0 rakamının olduğu dizilerdeki uygun bitlerinin sayısıdır. Uzaklık fonksiyonunu $\delta\,(c, c^{|})$ ile gösteriyoruz. Örneğin; c = 101011 ve $c^{|}$ = 110010 olsun

δ (c, $c^{|}$) = 3'tür. İki dizi arasında 2. , 3. ve 6. pozisyonda fark vardır. Rastgele seçilmemiş bir dizi gönderilirken iki dizi arasındaki uzaklık artıkça hata sayısı artar. δ ile gösterilen uzaklık fonksiyonlarının temel özellikleri şunlardır:

Teorem 5.4: c ve $c^{|}$, $c^{||}$ dizileri için;

a) Ancak ve ancak c = $c^{|}$ ise δ (c, $c^{|}$) = 0

b) δ (c, $c^{|}$) = δ ($c^{|}$, c)

c) δ (c, $c^{|}$) $\leq$ δ (c, $c^{|}$) + δ ($c^{|}$, $c^{||}$)

İspat: a ve b bölümleri doğrudan görülmektedir. C bölümünde; c ve $c^{|}$ dizileri için wt (c + $c^{||}$)= δ (c, $c^{||}$)'dir. Bunu görmek için;

c= c_1 c_2 c_3 . . . c_n ve $c^{||}$= $c^{||}_1$ $c^{||}_2$ $c^{||}_3$ . . . $c^{||}_n$ ise ancak ve ancak c_i =0 ve $c^{||}_i$=1 ve ya c_i =1 ve $c^{||}_i$ =0 olduğunda, c_i+ $c^{||}_i$, 1'den wt(c+ $c^{||}$)'ye katılır.

Fakat bu doğru ancak ve ancak 1'den δ (c, $c^{||}$)'ye katılan c_i ve $c^{||}_i$ farklı ise gerçekleşir. Ayrıca herhangi $c^{|}$ dizisi için; $c^{|}$ + $c^{|}$ dizisi sadece 0'ları içeriyorsa buna "0 dizisi" denir. Toplamın tanımıyla, her c dizisi için 0+c=c'dir. Bu nedenle,

$$\begin{aligned}\delta\ (c, c^{||}) &= wt\ (c+ c^{||}) \\ &= wt\ (c+0+ c^{||}) \\ &= wt\ (c+ c^{|}+ c^{|}+ c^{||}) \\ &\leq wt\ (c+ c^{|})+ wt\ (c^{|}+ c^{||}) \\ &= \delta\ (c, c^{|})+ \delta\ (c^{|}, c^{||}) \qquad (5.5)\end{aligned}$$

Koddaki herhangi iki dizi arasındaki minimum uzaklığı bilmek önemlidir. C kodsa, C kodundaki her hangi iki dizi arasındaki en kısa uzunluğa C kodunun minimum uzaklığı denir. D (C) ile gösterilir.

Aşağıdaki teorem kod kullanarak bulunmuş veya düzeltilmiş hata sayısında önemli bir ölçüm yoludur.

Teorem 5.5: C kodu için;

a) k kadar hata bulmak için D (C) = k + 1 olmalıdır.

b) k kadar hata düzeltebilmek için D (C) = 2k + 1 olmalıdır.

İspat: a) D (C)=k+1 ve c ∈ C ise c, koddaki en az k+1 yerlerindeki diğer dizilerden farklı olur. Bu nedenle, c iletilmişse ve k ve ya daha az hatası varsa; koddaki başkabir dizi olma ihtimali yoktur ve hata bulunur.

b) Eğer c dizisi, k veya daha az hatayla $c^{|}$ olarak iletilmişse; δ (c, $c^{|}$) ≤k'dir. C'deki bazı $c^{||}$ dizileri için δ ($c^{|}$, $c^{||}$) ≤k ise δ (c, $c^{|}$)+ δ ($c^{|}$, $c^{||}$)≤2k'dır. Fakat, δ ($c^{|}$, $c^{||}$) ≤ δ (c, $c^{|}$)+ δ ($c^{|}$, $c^{||}$) ve δ (c, $c^{||}$)≥2k+1 çelişki verir. Bu nedenle, $c^{|}$; $c^{|}$'ye uzaklığı k+1'den az tek dizi olan c'ye düzeltilebilir.

C kodundaki her hangi iki dizi arasındaki en kısa uzaklığı, D(C), yi bulmak için aşağıdaki teoreme ihtiyacımız vardır.

Teorem 5.6: D (C), W (C) = min { ω t (c): c ∈ C ve c ≠ 0 }'dır.

İspat: D (C)'nin tanımıyla; c, $c^{|}$∈ C ise δ (c, $c^{|}$)=D (C)'dir.

Ama δ (c, $c^{|}$)= wt(c+$c^{||}$) ve böylece c+$c^{||}$∈ C, W (C) ≤ wt(c+$c^{||}$)'dir.

Bu nedenle, W(C) ≤ D (C)'dir. Tersine, c∈ C için

wt (c)=wt (c+0) = δ (c, 0)≥ D (C)'dir. Bu nedenle, W (C) ≥ D (C)'dir.

Buradan; W (C) = D (C) olur.

w (c) ≥ 3 olan C Hamming kod'unu gösterlim. 1 ağırlında c ∈ C yoktur. Eğer varsa ve c j. Yerdeki 1'lerin haricinde bütün 0 ların dizisi ise ; $G_H^{\perp}$ 'nin bütün satırları için c ortagonaldır ve $G_H^{\perp,}$ nin j. sütunu $G_H^{\perp,}$ nin yapısıyla çelişen bütün 0'ları içerir. Yine, 2 ağırlığında da c ∈ C yoktur. Eğer olsaydı; c, i. ve j. pozisyondaki iki 1'in dışındaki bütün 0'lar içeren bir dizi olurdu. Yine bu nedenle c $G_H^{\perp}$ 'nin her satırı için ortogonoldir ve $G_H^{\perp}$ ' nin her satırındaki i. Ve j. sütunlar ya iki 1'den yada iki 0'dan olmalıdır. Ama $G_H^{\perp,}$ nin i. ve j. sütunları $G_H^{\perp,}$ nin yapısıyla çelişmelidir. Bu nedenle, w (c) ≥ 3 ve C, tek hata için hata düzeltici olarak kullanılabilir.

Şimdi şunu göstermek istiyoruz: herhangi bir kosette C'nin kendisi dışında 1 ağırlığında bir elemanımız var.

Eğer bunu yaparsak, problemin çözümü çok kolaylaşır. Teorem 5.1'e göre [n, k] kod C, 2^k dizi içerir. Bu nedenle G_H [I_{n-r} | A] şeklinde biçimlendirilir, C Hamming kod'u [n, n – r] kod' dur. Yani C 2^{n-r} eleman içerir. B_n 2^n elemandan oluşur.

Yani,

$\frac{2^n}{2^{n-r}} = 2^r$ koset C içerir.

C'deki diziler $2^r - 1$ uzunluğundadır. Bu nedenle 1 ağırlığında $2^r - 1$ tane dizi vardır. Şimdi şunu göstermeliyiz. Hiçbir koset ağırlığı 1 olan iki dizi içermesin. s ve s' aynı kosetteki ağırlığı 1 olan iki dizi olsun. Buradan, koset tanımına göre bazı $c \in C$ için $s = s' + c$'dir. Buradan $c = s + s'$'dir. Teorem 5.3'e göre,

$$w t (c) \leq w + (s) + w t (s') \leq 1 + 1 = 2$$

Ama, wt (c) $\geq$ 3, yani bu kesinlikle imkansızdır. Bu nedenle her koset, ağırlığını 0 olan dizi içeren C'nin dışında, tam olarak ağırlığı 1 olan bir dizi içerir.

Örneğimize geri dönersek, $G_H^{\perp}$ matrisi,

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

G_H matrisi

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$ 'dir

Buradan, her koset 0010000 dizisinden oluşan 1 ağırlıklı baş koset içerir.

Eğer bunu $G_H^{\perp}$ ile çarparsak,

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$ elde ederiz.

Yani $G_H^{\perp}$ 'nin 3. sütununun sendromu $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$'dir. Bu nedenle, 1 ağırlıklı dizinin j. sayısında 1 oluşursa, dizinin transpozuyla $G_H^{\perp}$ çarpıldığında, syndrome $G_H^{\perp}$' deki l. sütundur.

Yani, gönderilmiş mesaj dizisi aldığımızda ve bunu $G_H^{\perp}$ ile çarptığımızda, eğer gönderi doğruysa, sendrom hep 0'dır. Eğer bir hata varsa, $G_H^{\perp}$'nin sütunlarında birini elde ederiz, çünkü gönderilmiş mesaj dizisi kosetlerden birinde olmak zorundadır, bu nedenle ağırlığı 1 olan baş koset vardır. Yani, eğer $G_H^{\perp}$'nin i. sütunu sendromsa, baş kosetin i. sütununda 1 olduğunu biliriz, yani hata i. sütundadır yada dizinin i. bitindedir.

Örneğin; gönderilmiş mesajımız 1110110 olsun; bunun transpozunu $G_H^{\perp}$ ile çarpalım,

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$ elde ederiz.

Bu $G_H^{\perp}$'nin 2. sırasıdır. Yani 2. bittedir ve doğru mesaj 1010110 olmalıdır.

5.2. Golay - Reed Muller - Hadamard Kodları

Bu bölümün geri kalanında diğer kodları kısaca inceleyeceğiz. Bunların ilki Golay koddur. Hamming kodları, 1950'de Hamming ve 1949'da Golay tarafından bağımsız olarak bulunmuştur. Bu kodlar Golay in 1949'daki makalesindedir. Bu kodlara aynı zamanda Hamming kodlarıda denilmektedir. Bu kodları C(n,k,d) biçiminde gösterebiliriz. Burada n kodun uzunluğu yani kod kelimelerindeki bitlerin sayısı; k kodun boyutu yani mesaj bitlerinin sayısı ve d ise kodun minimum Hamming uzaklığıdır.

Bu (23, 12, 7) modelinde Golay tarafından yayımlanmıştır. Bunun anlamı C'deki dizilerinin aralarındaki minimum uzaklığı 7 olan (23, 12) üretici matrisidir. Bu Golay kod

$G = [\, I_{11} \mid A \,]$ dan üretilir.

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(24, 12, 8) şeklinde genişletilmiş üretici matris kullanarak çalışmak daha kolaydır.

$G = [\, I_{12} \mid A \,]$'dır.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Bu Golay üretici kodunun tekrar düzenlenip parity bit'i eklemesiyle oluşur. Bu matrisin simetrisi çalışmamızı kolaylaştırıyor.

$G^{\perp} = [\, A \mid I_{12} \,]$ 'yi görmek daha kolay, çünkü $A = A^{t}$'dır.

Golay kodu (4096, 244, 8) code'u içeren çeşitli kodlar oluşturur.

(4096, 244, 8) kodu Voyager Uzay gemisinde Jüpiter, Uranüs ve Neptün'ün görüntüleri göndermek için kullanılmıştır.

n uzunluklu s dizisi için, $\bar{s} = s + \mathbf{1}$ alalım, **1** bütün 1'leri içeren n uzunluğundaki dizidir. S dizilerinin kümesi verilsin,

Plot (S) = $\{ ss: s \in S \} \cup \{ s\bar{s} : s \in S \}$'dir.

S = { 0000, 0011, 1100, 1111 } kümesi verilsin;

$S_1 = Plot (S)$, $S_2 = Plot (S_1)$, $S_3 = Plot (S_2)$, $S_n = Plot (S_{n-1})$'dir.

Plot M Plotkin tarafından yaratılmış bir yapıdır. S_1 , S_2 , S_3 , … kümelerinden üretilen kodlar Reed – Muller kodları denir. S_3 kümesi (64, 32, 16) kodudur. Bu kod Marriner 9 uzay aracından gönderilen görüntülerin hatalarının düzeltilmesinde kullanılmıştır. Özel olarak Reed – Muller matrislerine Hadamard matrisi denir. Şimdi bunu tanımlayacağız.

Aşağıdaki gibi tekrarlanarak tanımlanmış matrisleri hesaplayalım.

$$A_1 = [\ 0\ ]$$

$$A_{2n} = \begin{bmatrix} A_n & A_n \\ A_n & \overline{A}_n \end{bmatrix} \text{'dir.}$$

$\overline{A}_n$ Matrisi $1 \le i, j \le n$ için $\overline{A}_{ij} = A_{ij}$ şeklinde tanımlanır.

Buradan,

$$A_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

ve

$$A_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \text{ 'dir.}$$

H_n, $1 \le i$ ve $j \le n$ olan her A_{ij} için 0'ların 1, 1'lerin -1'e dönüşmesiyle elde edilen A_n sonuçlarından oluşur. Bu nedenle,

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{'dir}$$

H_n matrisinin $H_n H_n^t = nI$ özelliği vardır. I birim matrisidir. Bu özellikteki matrislere Hadamard matrisler denir. Üretilen kodlara da 'Hadamard code' denir.

KAYNAKLAR

Carretero, R. (March 20.2002). Nonlinear Dynamical Systems at sdsu. Retrived February 15, 2006, from http://nlds.edu

Kara, İ. 1986. Yöneylem Araştırması, Doğrusal Olmayan Modeller. T.C. Anadolu Üniversitesi Mühendislik-Mimarlık Fakültesi Yayınları, No: 29, Eskişehir.

Türker, E. S. Ve Can, E. 1997. Bilgisayar Uygulamalı Sayısal Analiz Yöntemleri. Değişim Yayınları, Adapazarı. 95-139.

Aksoy, Y. 1996. Boole Cebiri Ve Lojik Devre Sentezi, Yıldız Teknik Üniversitesi. İstanbul. 219-272 s.

Anderson, A. J. 2004. Discrete Mathematics with Combinatorics. New Jersey. (2 nd ed.). 9-38 p.

Minez, Z. G. 1999. Diskret Sistemlerin Minimalleştirilmesi, Dumlupınar Üniversitesi, Kütahya. Yüksek Lisans Tezi. 7-60 s.

Rosen, H. K. 1995. Discrete Mathematics and Its Aplications, (3 rd. Ed) , New York: Mc Graw-Hill 611-622 p.

Yablonsky, S. V. 1989. Introduction to Discarte Mathematics, Translated from Russion Mr. Publishers Moscow. 44-49 p.

Owen G., 1995. Game Theory (3th ed.) Department of Mathematics Novel Postgroduate School Montery, California. (8th chap.) 164-172 p.

Weistein E. W., (February 13,2006) Dynamical Systems. Retrived February 13,2006 from http://mathworld.wolfram.com/DynamicalSystem.html

Harary F., 1969 Graf Theory, Universty of Michigan. Cambrige, Massachusetts. (4th chap.) 32-40 p.

Çapkın A. 2009 Sonlu Sistemlerde Deneme Problemleri (Yüksek Lisans Tezi) Çanakkale Onsekiz Mart Üniversitesi. Çanakkale

TABLOLAR LİSTESİ

Sayfa No

ŞEKİLLER LİSTESİ

Sayfa No

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Şefik YAMALI

Doğum Yeri: Çanakkale

Doğum Tarihi: 01/01/1976

EĞİTİM DURUMU

Lisans Öğrenimi: Çanakkale Onsekiz Mart Üniversitesi

Yüksek Lisans Öğrenimi: Çanakkale Onsekiz Mart Üniversitesi

Bildiği Yabancı Diller: İngilizce

İŞ DENEYİMİ

Çalıştığı Kurumlar ve Yıl: Çanakkale Merkez Dershanesi 1998 – 2001

Eceabat Kumköy Okulu 2001 – 2004

Çanakkale Lisesi 2004 - ……

İLETİŞİM

E-Posta Adresi: sefikyamali@hotmail.com