



**YILDIZ TEKNİK ÜNİVERSİTESİ**  
**ELEKTRİK-ELEKTRONİK FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**YAPAY ZEKA DÖNEM PROJESİ**

**KNAPSACK PROBLEMİNİN GENETİK ALGORİTMA**  
**İLE ÇÖZÜMÜ**

Proje Yöneticisi: Yrd. Doç. Dr. M. Fatih AMASYALI

Proje Grubu:

10011904 Napam BIAGUE

11011019 Enis AMUK

İstanbul, 2015

## İÇİNDEKİLER

İÇİNDEKİLER.....	2
ŞEKİL LİSTESİ .....	3
1. GİRİŞ .....	4
1.1 Knapsack 0-1 Problemi .....	4
1.2 Genetik Algoritma Yaklaşımı.....	4
1.3 Knapsack Probleminin Genetik Algoritma ile Çözüm Adımları.....	4
1.3.1 Eşyaların oluşturulması .....	4
1.3.2 Kromozomların oluşturulması.....	4
1.3.3 Kromozomların Uygunluk Yüzdelerinin Bulunması .....	5
1.3.4 İki Kromozomun Çaprazlanması .....	5
1.3.5 Mutasyon.....	5
1.3.6 Yeni Nesil Üretilmesi.....	5
2. PERFORMANS ANALİZİ .....	5
KAYNAKLAR.....	9

## ŞEKİL LİSTESİ

Şekil 2.1 Rastgele eşyaları kullanılarak çözüm.....	6
Şekil 2.2 Rastgele çözüm örneği .....	6
Şekil 2.3 Sonucu belli olan çözüm .....	7
Şekil 2.4 Örnek bir soru .....	7
Şekil 2.5 Sonucu belli olan çözüm 2 .....	8

## 1. GİRİŞ

Günümüzde yapay zeka çalışmaları olarak bir sürü çalışma yapılmaktadır. Bu çalışmalar yeni konu başlıkları olmakla beraber daha önce çözülmüş problemler de yapay zeka bakış açısıyla çözülmeye çalışılmıştır. Biz de bu fikirden yola çıkarak bu çalışmada var olan bir problemi yapay zeka algoritmasıyla çözmeye çalıştık. Bu çalışmada Knapsack problemi genetik algoritma yaklaşımıyla çözülmeye çalışılmıştır.

Bu çalışma JAVA programlama dili kullanılarak tasarlanmıştır. IDE olarak Eclipse kullanılmıştır. Çalışmada arayüz de bulunduğundan eclipse'nin Swing kütüphanesi de kullanılmıştır.

### 1.1 Knapsack 0-1 Problemi

Bu problemde isimleri 1 ile n arasında sayı ile ifade edilen n değişik madde bulunur. Her bir madde i için değerin  $v_i$  ve ağırlığın  $w_i$  olduğu bilinmektedir. Genel olarak her bir değer ve her bir ağırlık negatif olamazlar. Çanta içinde taşınabilecek tüm maddelerin toplam ağırlığının en çok W olup, bunun bir üst sınır olup aşılamayacağı bilinir [1].

Bu problemde amaç sırt çantasına, verilen çanta kapasitesini aşmayacak şekilde eşyalar yerleştirilerek maksimum değerin bulunmasıdır.

### 1.2 Genetik Algoritma Yaklaşımı

Genetik algoritmalar, doğada gözlemlenen evrimsel sürece benzer bir şekilde çalışan arama ve eniyileme yöntemidir. Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar [2]. Çözümü bulma adımları olarak kromozomların oluşturulması, bu kromozomlardan popülasyondaki uygunluk yüzdelere göre çaprazlama yapılması ve mutasyondur.

### 1.3 Knapsack Probleminin Genetik Algoritma ile Çözüm Adımları

Problemin çözüm adımları alt başlıklar halinde anlatılmıştır.

#### 1.3.1 Eşyaların oluşturulması

Problemde eşya sayısı ve buna bağlı olarak eşyaların ağırlıklarının ve değerlerinin kullanıcı tarafından el ile girilmesi uzun ve zahmetli olmasından dolayı kullanıcıdan eşya sayısının girilmesi istenmiştir. Kullanıcıdan alınan bu eşya sayısı Eşya\_Olustur(int eşya) metodu çağrılarak rastgele eşya sayısı kadar eşya oluşturulmuştur. Oluşturulan eşyaların ağırlıkları 1-10 arasında ve değerleri de 1-50 arasında değişmektedir. Oluşturulan eşyalar “eşyalar” adlı arraylist’te tutulmaktadır.

#### 1.3.2 Kromozomların oluşturulması

Kullanıcıdan eşya sayısının yanında çanta kapasitesi ve kromozom sayısının girilmesi istenmiştir. Girilen bu değerler ile Kromozom\_Olustur(int eşya, int çantaKapasitesi) fonksiyonu çağrılarak çanta kapasitesini aşmayacak şekilde rastgele kromozomlar oluşturulmuştur. Oluşturulan kromozomların uzunluğu eşya sayısı uzunluğundadır. Her oluşturulan kromozom “kromozomlar” adlı parent arraylist’inde tutulmaktadır.

### 1.3.3 Kromozomların Uygunluk Yüzdelerinin Bulunması

Kromozomlar üretildikten sonra çözüme olan uzaklıklarına göre seçilme yüzdeleri hesaplanmıştır. Bu hesaplama yapılırken her bir kromozomun değeri ile çanta kapasitesi/kromozomağırlığı oranı ile çarpılarak bir değer elde edilmiştir. Her bir kromozom için elde edilen bu değerlerin toplamı hesaplanmıştır. Her bir değerın toplam değere bölünüp 100 ile çarpılmasıyla kromozomların yüzdeleri hesaplanmıştır. Hesaplanan yüzdelere kadar 100 elemanlı bir yüzde dizisinde baştan itibaren her bir kromozomun yüzdesi kadar eleman kromozomun ID'si ile etiketlenmiştir. Bu sayede uygunluk fonksiyonu pasta dilimi haline çevrilmiştir.

### 1.3.4 İki Kromozomun Çaprazlanması

Çaprazlama işlemi için crossover(int indis1, int indis2, int esya, int çantaKapasitesi) fonksiyonu çağrılır. Bu fonksiyona rasgele seçilmiş iki kromozomun indisi gönderilir. Burada orijinal kromozomlar listesi üzerinde işlem yapmamak için yeni bir "yeniNesilKromozomlar1" adlı arraylist oluşturulmuştur. Orijinal liste yeni oluşturulan listeye kopyalanmıştır. Daha sonra kromozomları ortadan ikiye ayırarak çaprazlama işlemi yapılmıştır. Yani ilk kromozomun ilk yarısı ikinci kromozomun ikinci yarısı ile birleştirilmiştir. Çaprazlama yapıldıktan sonra mutasyon işlemi yapılmıştır. Bu mutasyon sonucunda oluşan iki kromozomun her birinin ağırlıkları çanta kapasitesinin aşmıyor ve sıfır ağırlıklı değilse "child" nesli için oluşturulmuş olan yeniNesilKromozomlar2 adlı listeye eklenerek çaprazlama işlemi sonlanmaktadır.

### 1.3.5 Mutasyon

Mutasyon işlemi için mutasyon(int indis, int esya, ArrayList<Kromozom> arr) adlı fonksiyon çağrılmaktadır. Bu fonksiyona gelen indis ile kromozomun rastgele bir noktasındaki değer sıfır ise bir, bir ise sıfır yapılarak mutasyon işlemi gerçekleştirilmiştir.

### 1.3.6 Yeni Nesil Üretilmesi

Yeni nesil üretebilmek için yeniNesilUret(int esya, int çantaKapasitesi) adlı fonksiyon çağrılmaktadır. Bu fonksiyon ilk olarak her bir kromozomun uygunluk yüzdesini hesaplamaktadır. Uygunluk yüzdesi hesaplandıktan sonra rastgele iki sayı üretilip 100'e göre mod'u alındıktan sonra uygunluk fonksiyonu ile elde edilen yüzde dizisinden iki kromozom indisi elde edilir. Bu iki indis crossover() fonksiyonuna gönderilerek çaprazlama yapılır. Bu işlemler "child" sayısı parent sayısına eşit olana kadar yapılır ve son olarak üretilen "child"lar "parent" listesi olan "kromozomlar" listesine kopyalanır. Yeni nesil üretilmiş olur.

## 2. PERFORMANS ANALİZİ

Program çalıştırıldıktan sonra yapılan denemelerde genetik algoritmanın çözümü bulma sıklığı çok düşük olduğu gözlemlenmiştir. Çözümü bulma yüzdesi yaptığımız denemeler sonucunda yaklaşık olarak %6 ila %15 arasında değişmektedir.

Rastgele eşyalar üretildiğinde çözümü bilmediğimizden çözümü bulma yüzdesini hesaplayamıyoruz. Fakat 100 defa çalıştırıldığında en az bir defa çözüm bulunduğundan program bize tam bir sonuç vermektedir. Şekil 2.1 ve Şekil 2.2'de örnek rastgele oluşturulmuş eşyalar kullanılarak elde edilmiş bir çözüm gösterilmektedir.

Eşya sayısı 16 Kromozom Sayısı 64

Canta kapasitesi 10

Hesapla

Maksimum Değer 104.0 Çözüm Bulma Yüzdesi :

Maksimum degerin bulunduğu kromozom

0000010010001000

Şekil 2.1 Rastgele eşyaları kullanılarak çözüm

Eşya sayısı 20 Kromozom Sayısı 64

Canta kapasitesi 18

Hesapla

Maksimum Değer 177.0 Çözüm Bulma Yüzdesi :

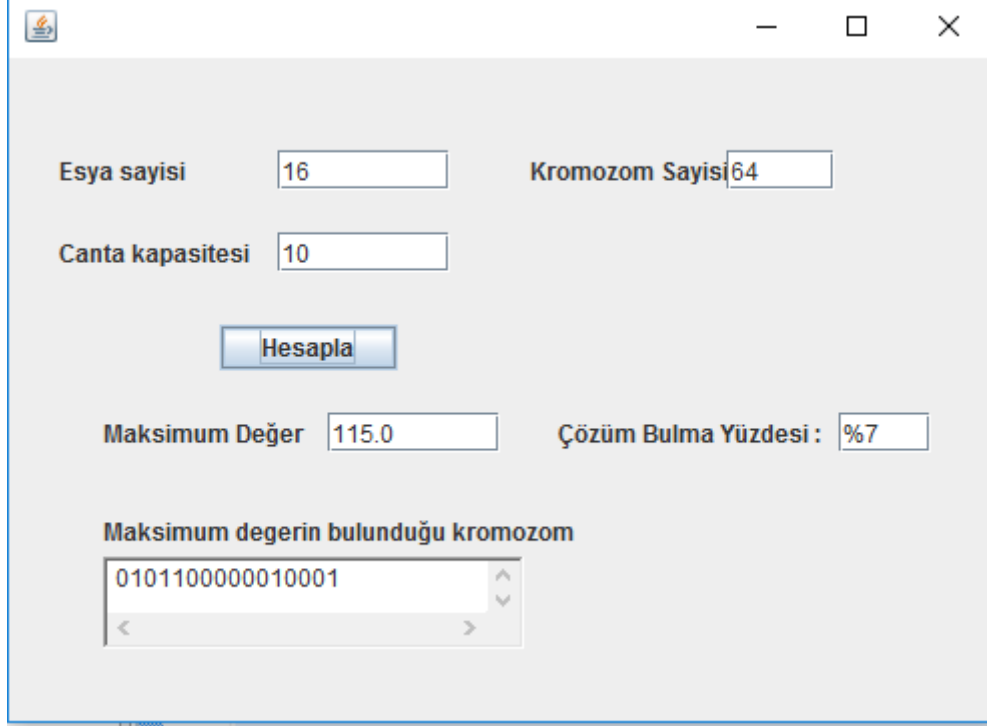
Maksimum degerin bulunduğu kromozom

01001100000100000001

Şekil 2.2 Rastgele çözüm örneği

Belli değerler ile program çalıştırıldığında da çözümü bilindiğinden çözümü bulma yüzdesi hesaplanabilmektedir. İnternette bulduğumuz bir örnek sayesinde sonucu bildiğimiz için çözümü bulma yüzdesi hesaplanmış program çıktısı Şekil 2.3’de gösterilmektedir. Programın başarısını kodu 100 defa çalıştırıp her adımda bulunan sonucu gerçek sonuç ile karşılaştırıp

eğer gerçek sonuca eşitse başarıyı tutan bir değişkenin değerini 1 arttırıyoruz. Bu şekilde program 100 defa çalıştığında kaçında çözümü bulduğumuzu hesaplamış oluyoruz.



The screenshot shows a software window with the following fields and controls:

- Eşya sayısı**: 16
- Kromozom Sayısı**: 64
- Canta kapasitesi**: 10
- Hesapla** button
- Maksimum Değer**: 115.0
- Çözüm Bulma Yüzdesi**: %7
- Maksimum degerin bulunduğu kromozom**: 0101100000010001

Şekil 2.3 Sonucu belli olan çözüm

İnternette bulduğumuz bir başka örnek ile program çalıştırıldığında eşya sayısı az olduğundan dolayı genetik algoritmanın çözümü bulma yüzdesi Şekil 2.5’de gösterildiği gibi %85 çıkmıştır.

Aşağıdaki şekilde kullanılan örnek gösterilmiştir [3].

Let  $W = 10$  and

$i$	1	2	3	4
$v_i$	10	40	30	50
$w_i$	5	4	6	3

Şekil 2.4 Örnek bir soru

Eşya sayısı  Kromozom Sayısı

Canta kapasitesi

Maksimum Değer  Çözüm Bulma Yüzdesi :

Maksimum degerin bulunduğu kromozom

Şekil 2.5 Sonucu belli olan çözüm 2



## **KAYNAKLAR**

[1] “Knapsack Problemi” ,

[https://tr.wikipedia.org/wiki/S%C4%B1rt\\_%C3%A7antas%C4%B1\\_problemi](https://tr.wikipedia.org/wiki/S%C4%B1rt_%C3%A7antas%C4%B1_problemi) , 10.12.2015

[2] “Genetik Algoritma” ,

[https://tr.wikipedia.org/wiki/Genetik\\_algoritma](https://tr.wikipedia.org/wiki/Genetik_algoritma) , 10.12.2015

[3] “Örnek soru” ,

<http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf> ,24.10.2015