GENETİK ALGORİTMALAR

Dr. Mahir Kaya
Tokat Gaziosmanpaşa Üniv.
Bilgisayar Mühendisliği
TOGU -CENG

GENETİK ALGORİTMALAR

Günümüzdeki zor ve karmaşık koşullar, problemlere hızlı ve kolay sonuç veren yeni çözüm yöntemleri geliştirilmesine neden olmuştur. Yapılan çalışmalarda klasik metotların yerine evrimsel algoritma kullanımı ön plana çıkmıştır. Evrimsel yaklaşımlardan olan genetik algoritmalar, yapılan çalışmalarda önemli bir yer tutmaktadır

Genetik algoritmalar, doğal seçim ilkelerine dayanan bir arama ve optimizasyon yöntemidir.

Genetik algoritmalar problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan bir çözüm kümesi üretir. Böylelikle, arama uzayında aynı anda birçok nokta değerlendirilmekte ve sonuçta bütünsel çözüme ulaşma olasılığı yükselmektedir. Çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır. Her biri çok boyutlu uzay üzerinde bir vektördür

GENETİK ALGORİTMALAR

Genetik Algoritma yaklaşımının ortaya çıkışı 1970'lerin başında olmuştur. ***1975***'te ***John Holland***'ın makine öğrenmesi üzerine yaptığı çalışmalarda canlılardaki evrimden ve değişimden etkilenerek, bu genetik evrim sürecini bilgisayar ortamına aktarması ve böylece bir tek mekanik yapının öğrenme yeteneğini geliştirmek yerine, çok sayıdaki böyle yapıların tamamını "çiftleşme, çoğalma, değişim…" gibi genetik süreçler sonunda üstün yeni bireylerin elde edilebileceğini gösteren çalışmasından çıkan sonuçların yayınlanmasından sonra geliştirdiği yöntemin adı "Genetik Algoritmalar" olarak tanınmıştır.

- Bir probleme olası pek çok çözümün içerisinde en uygununu (en iyisini) bulmaya çalışan algoritmalardır.
- Popülasyon nesilden nesile geliştikçe kötü çözümler yok olma, iyi çözümler ise daha iyi çözümler oluşturmak için kullanılma eğilimindedirler.

Genetik algoritmalar problemlerin çözümü için evrimsel süreci bilgisayar ortamında taklit ederler. Diğer optimizasyon yöntemlerinde olduğu gibi çözüm için tek bir yapının geliştirilmesi yerine, böyle yapılardan meydana gelen bir küme oluştururlar.

Problem için olası pek çok çözümü temsil eden bu küme genetik algoritma terminolojisinde **popülasyon** (nüfus) adını alır.

Popülasyonlar **vektör**, **kromozom** veya **birey** adı verilen sayı dizilerinden oluşur. Birey içindeki her bir elemana **gen** adı verilir. Popülasyondaki bireyler evrimsel süreç içinde genetik algoritma işlemcileri tarafından belirlenirler.

Problemin bireyler içindeki gösterimi problemden probleme değişiklik gösterir. Genetik algoritmaların problemin çözümündeki başarısına karar vermedeki en önemli faktör, problemin çözümünü temsil eden bireylerin gösterimidir.

Popülasyon içindeki her bireyin problem için çözüm olup olmayacağına karar veren bir **uygunluk (amaç)** fonksiyonu vardır.

Uygunluk fonksiyonundan dönen değere göre yüksek değere sahip olan bireylere, Popülasyondaki diğer bireyler ile çoğalmaları için fırsat verilir.

Bu bireyler çaprazlama işlemi sonunda **çocuk** adı verilen yeni bireyler üretirler. Çocuk kendisini meydana getiren ebeveynlerin (anne, baba) özelliklerini taşır.

Yeni bireyler üretilirken düşük uygunluk değerine sahip bireyler daha az seçileceğinden bu bireyler bir süre sonra nüfus dışında bırakılırlar.

Yeni nüfus, bir önceki nüfusta yer alan uygunluğu yüksek bireylerin bir araya gelip çoğalmalarıyla oluşur. Aynı zamanda bu nüfus önceki nüfusun uygunluğu yüksek bireylerinin sahip olduğu özelliklerin büyük bir kısmını içerir. Böylelikle, pek çok nesil aracılığıyla iyi özellikler nüfus içerisinde yayılırlar ve genetik işlemler aracılığıyla da diğer iyi özelliklerle birleşirler.

Uygunluk değeri yüksek olan ne kadar çok birey bir araya gelip, yeni bireyler oluşturursa arama uzayı içerisinde o kadar iyi bir çalışma alanı elde edilir.

Probleme ait en iyi çözümün bulunabilmesi için;

- Bireylerin gösterimi doğru bir şekilde yapılmalıdır.
- Uygunluk fonksiyonu etkin bir şekilde oluşturulmalıdır.
- Doğru genetik işlemciler seçilmelidir.

Bu sayede, çözüm kümesi bir noktada birleşecektir.

Genetik algoritmalar, oldukça büyük arama uzayına sahip problemlerin çözümünde başarı göstermektedir.

Genetik algoritmalar, bir problemin bütünsel en iyi çözümünü bulmak için garanti vermezler. Ancak, problemlere makul bir süre içinde, kabul edilebilir, iyi çözümler bulurlar.

Genetik algoritmaların asıl amacı, hiçbir çözüm tekniği bulunmayan problemlere çözüm aramaktır. Kendilerine has çözüm teknikleri olan özel problemlerin çözümü için, mutlak sonucun hızı ve kesinliği açısından genetik algoritmalar kullanılmazlar. Genetik algoritmalar ancak;

- Arama uzayının büyük ve karmaşık olduğu,
- Mevcut bilgiyle sınırlı arama uzayında çözümün zor olduğu,
- Problemin belirli bir matematiksel modelle ifade edilemediği,
- Geleneksel optimizasyon yöntemlerinden istenen sonucun alınmadığı, alanlarda etkili ve kullanışlıdır.

Genetik algoritmaların avantajları;

- Sürekli ve ayrık parametreleri optimize ederler
- Türevsel bilgiler gerektirmemesi
- Amaç fonksiyonu geniş bir spektrumda araştırması
- Çok sayıda parametrelerle çalışma imkanı olması
- Karmaşık amaç fonksiyonu parametrelerini, lokal minimum veya maksimumlara takılmadan optimize edebilmesi

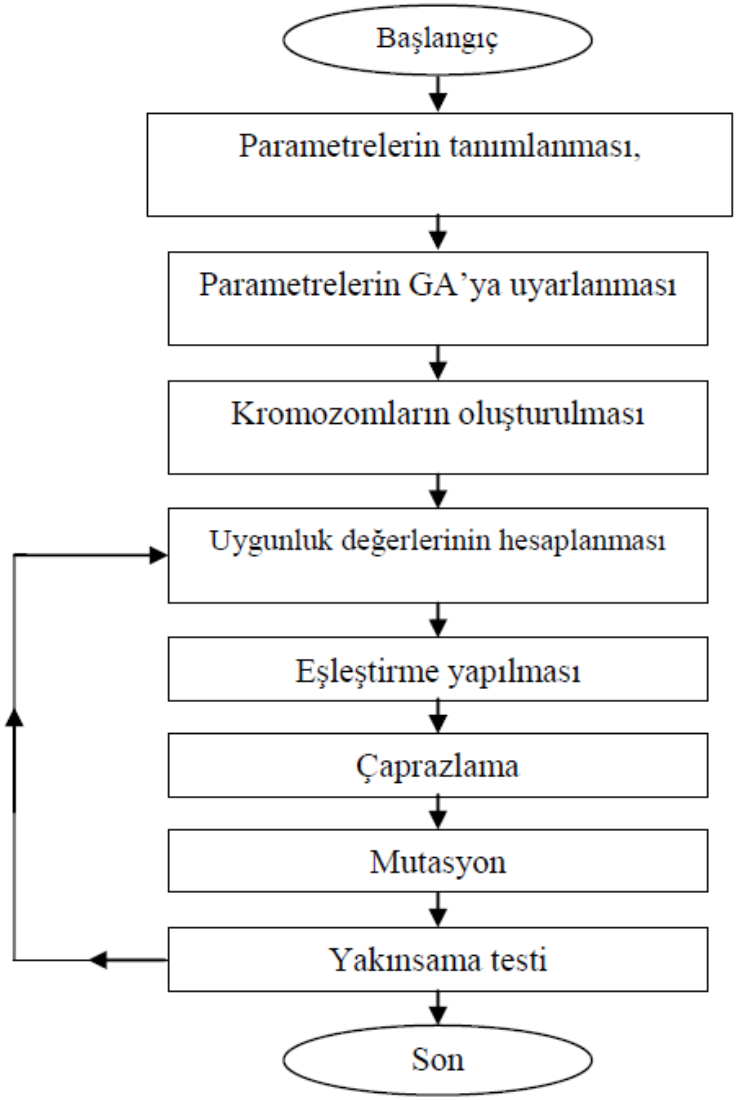
Genetik algoritmaların diğer yöntemlerden farkları;

- Genetik algoritmalar problemlerin çözümünü parametrelerin değerleriyle değil, kodlarıyla arar. Parametreler kodlanabildiği sürece çözüm üretilebilir. Bu sebeple genetik algoritmalar ne yaptığı konusunda bilgi içermez, nasıl yaptığını bilir.
- Genetik algoritmalar aramaya tek bir noktadan değil, noktalar kümesinden başlar. Bu nedenle çoğunlukla yerel en iyi çözümde sıkışıp kalmazlar.
- Genetik algoritmalar, türev yerine uygunluk fonksiyonunun değerini kullanır. Bu değerin kullanılması ayrıca yardımcı bir bilginin kullanılmasını gerektirmez.
- Genetik algoritmalar gerekli kuralları değil, olasılıklı kuralları kullanır.

GENETİK ALGORİTMALARIN UYGULAMA ALANLARI

- Genetik algoritma kullanımının en uygun olduğu problemler geleneksel yöntemler ile çözümü mümkün olmayan veya çözüm süresi problemin büyüklüğü ile üstel orantılı olarak artanlardır. Bugüne kadar GA ile çözümüne çalışılan konulardan bazıları şunlardır.
- **a. Optimizasyon:** Sayısal optimizasyon ve kombinasyona dayalı optimizasyon problemleri olan devre tasarımında, doğrusal olmayan denklem sistemlerinin çözümünde ve fabrika-üretim planlamasında
- **b. Otomatik programlama :** Bilgisayar programları yardımıyla network sıralamasında (sorting), ders programı hazırlanmasında
- **c. Makine öğrenmesi :** Robot sensörlerinde, yapay sinir ağlarında, VLSI yonga tasarımı ve protein yapısal analizinde
- **d. Ekonomi :** Ekonomik modellerin geliştirilmesinde ve işlemesinde
- **e. İmmün sistemler :** Doğal immün sistem modellerinde
- **f. Sosyal sistemler :** Sosyal sistemlerin analizinde
- **g. Finans :**Finansal modelleme uygulamalarında
- **h. Pazarlama :** Tüketicilere ait verileri analiz etmede

Genetik Algoritma Akış Şeması



Genetik algoritmaların işlem sırası

- Olası çözümlerin kodlandığı bir çözüm grubu oluşturulur.
- Popülasyonda bulunacak birey sayısı için bir standart yoktur, problemin türüne göre bu sayı değişebilir. Genellikle popülasyon rastgele oluşturulur.
- Popülasyondaki bireylerin verilen probleme göre uygunluk değerleri hesaplanır. Genetik algoritmanın başarısı çoğu zaman bu fonksiyonun verimli olmasına bağlıdır.
- Eşleşme havuzu oluşturulurken uygunluk değerleri baz alınır ve çeşitli seçim yöntemleri kullanılır.
- Seçilen kromozomlar eşlenerek yeniden kopyalama ve değiştirme uygulanır. Eşleşme havuzunda bulunan bireylerin çaprazlanması ve yeni bireylerin mutasyona uğraması bu aşamada yapılır.
- Yeni nesil bireylerin, ebeveyn bireylerle yer değiştirmesi sağlanarak popülasyonun sabit büyüklükte kalması sağlanır. Daha sonra, tüm kromozomların uygunluk değerleri tekrar hesaplanır.
- Belli bir nesil sayısına kadar algoritma döngüsü devam eder. Popülasyonun uygunluk değerleri tekrar hesaplanır, 2. adımdan itibaren işlemler tekrar edilir. Son olarak üretilen nesiller içinden en uygun değere sahip olan birey çözüm olarak kabul edilir.

Genetik Algoritmanın Performansını Etkileyen Faktörler

- **Kromozom sayısı:** Kromozom sayısının artırılması çalışma zamanını arttırmakta iken; kromozom sayısının azaltılması ise kromozom çeşitliliğini azaltmaktadır.
- **Mutasyon oranı:** Popülasyondaki bireyler birbirine benzemeye başladığında algoritmanın yakınsaması halen çözüm noktalarının uzağında bulunuyorsa, mutasyon işlemi GA'nın sıkıştığı yerden kurtulmak için tek yoldur. Ancak, mutasyon oranının yüksek bir değer seçilmesi GA'yı kararlı bir noktaya ulaşmaktan alıkoyacaktır.
- **Noktalı çaprazlama sayısı:** Normal olarak çaprazlama tek noktada gerçekleştirilmekle beraber, yapılan araştırmalar bazı problemlerde çok noktalı çaprazlamanın çok yararlı olduğunu göstermiştir.
- **Çaprazlamanın sonucu elde edilen bireylerin nasıl değerlendirileceği:** Elde edilen iki bireyin, hemen kullanılıp kullanılmayacağı yakınsama hızı açısından önemlidir.
- **Nesillerin birbirinden ayrık olup olmadığı:** Normal olarak her nesil tümüyle bir önceki nesle bağlı olarak oluşturulur. Bazı durumlarda, yeni nesli eski nesille birlikte, yeni neslin o ana kadar elde edilen bireyleri ile oluşturmak yararlı olabilir.
- **Parametre kodlanmasının nasıl yapıldığı:** Bir parametrenin doğrusal ya da logaritmik kodlanması genetik algoritmanın performansında önemli bir farka yol açmaktadır.
- **Kodlama gösteriminin nasıl yapıldığı:** Bu da nasıl olduğu yeterince açık olmamakla beraber genetik algoritmanın performansını etkileyen bir noktadır. İkilik düzen, kayan nokta aritmetiği ve gray kodu ile gösterim en yaygın yöntemlerdir.

Başlangıç Popülasyonu

• GA, çözüm adımlarına belirlenen gösterim şekline uygun kodlanmış bireylerden oluşan bir başlangıç popülasyonu oluşturarak başlarlar.
• Başlangıç popülasyonundaki her bir kromozom, problemin olası bir çözümünü temsil eder.
• Popülasyon sürekli daha iyi çözümler oluşturmaya çalıştığı için, zaman içinde değişir.
• Popülasyon büyüklüğü, problemin yapısına göre belirlenmelidir.

Genetik İşlemler

• Kodlama
  – Kodlama genetik algoritmanın çok önemli bir kısmını oluşturmaktadır.
  – Bir problem çözümüne başlarken sorulması gereken ilk sorudur.
  – Kurulan genetik modelin hızlı ve güvenilir çalışması için bu kodlama doğru yapılmalıdır.

Kodlama Türleri

- İkili Kodlama

Kromozom1	1101100100110110
Kromozom2	1101111000011110

- Değer Kodlama

Kromozom1	1.2324 5.3243 0.4556 2.3293 2.4545
Kromozom2	ABDJEIFJDHDIERJFDLDFLFEGT
Kromozom3	(back), (back), (right), (forward), (left)

- Permütasyon Kodlama

Kromozom1	1 5 3 2 6 4 7 9 8
Kromozom2	8 5 6 7 2 3 1 4 9

Seçim

- Yeni topluluğu oluşturmak için mevcut topluluktan çaprazlama ve mutasyon işlemine tabi tutulacak bireylerin seçilmesi gerekir.
- Teoriye göre iyi olan bireyler yaşamını sürdürmeli ve bu bireylerden yeni bireyler oluşturulmalıdır.
- Bu nedenle tüm seçilim yöntemlerinde uygunluk değeri fazla olan bireylerin seçilme olasılığı daha yüksektir.
- En bilinen seçilim yöntemleri Rulet Seçilimi, Turnuva Seçilimi ve Sıralı Seçilimdir.

Seçim Yöntemleri

- Rulet Seçilimi: Topluluktaki tüm bireylerin uygunluk değerleri toplanır ve her bireyin seçilme olasılığı, uygunluk değerinin bu toplam değere oranı kadardır.

- Sıralı Seçilim: En kötü uygunlukta olan kromozoma 1 değeri verilir, ondan daha iyi olana 2, daha iyisine 3 değeri verilerek devam edilir.

- Turnuva Seçilimi: Topluluk içerisinden rastgele k adet (3,5,7..) birey alınır. Bu bireylerin içerisinden uygunluk değeri en iyi olan birey seçilir.

Çaprazlama

Amaç, ata kromozomun yerlerini değiştirerek çocuk kromozomlar üretmek ve böylelikle zaten uygunluk değeri yüksek olan ata kromozomlardan daha yüksek uygunluklu çocuk kromozomlar üretmektir

A) Tek noktalı çaprazlama :

- Kromozom-1 : 11000|00100110110
- Kromozom-2 : 11011|11000011110
- Çocuk-1 : 11000011000011110
- Çocuk-2 : 11011100100110110

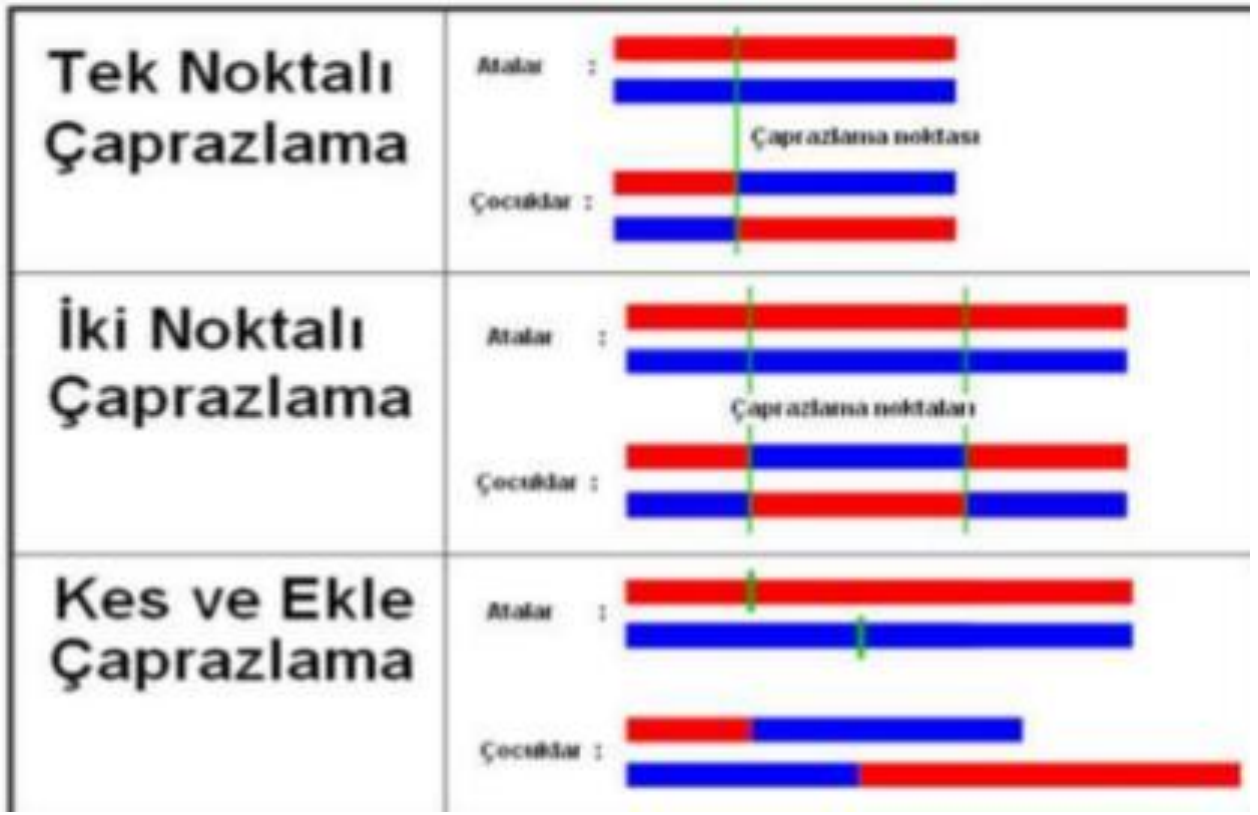
B) Çift noktalı çaprazlama :

- Kromozom-1 : 11000|00100|110110
- Kromozom-2 : 11011|11000|011110
- Çocuk-1 : 11000011000110110
- Çocuk-2 : 11011001000011110

C) Sıralı Çaprazlama

- A = 9 8 4| 5 6 7| 1 3 2 10
- B = 8 7 1| 2 3 10| 9 5 4 6
- A = 9 8 4| 2 3 10| 1 H H H
- B = 8 H 1 |5 6 7 |9 H 4 H
- AI = 9 8 4 |2 3 10 |1 5 6 7
- BI = 8 2 1| 5 6 7 |9 3 4 10 Sıra ile

soldan sağa doğru



Mutasyon

• Kromozomların kendi genleri veya genleri oluşturan küçük birimleri üzerinde değişiklik yapılmasını sağlayan işlemcidir.

• GA'da değişimin sağladığı avantaj, problemin çözüm alanını araştırmada yön değişikliklerini sağlayarak 'Mutasyon(Değişim) yardımıyla araştırmanın kısır döngüye girmesini önlemektir. ' (Lokal Maksimum)

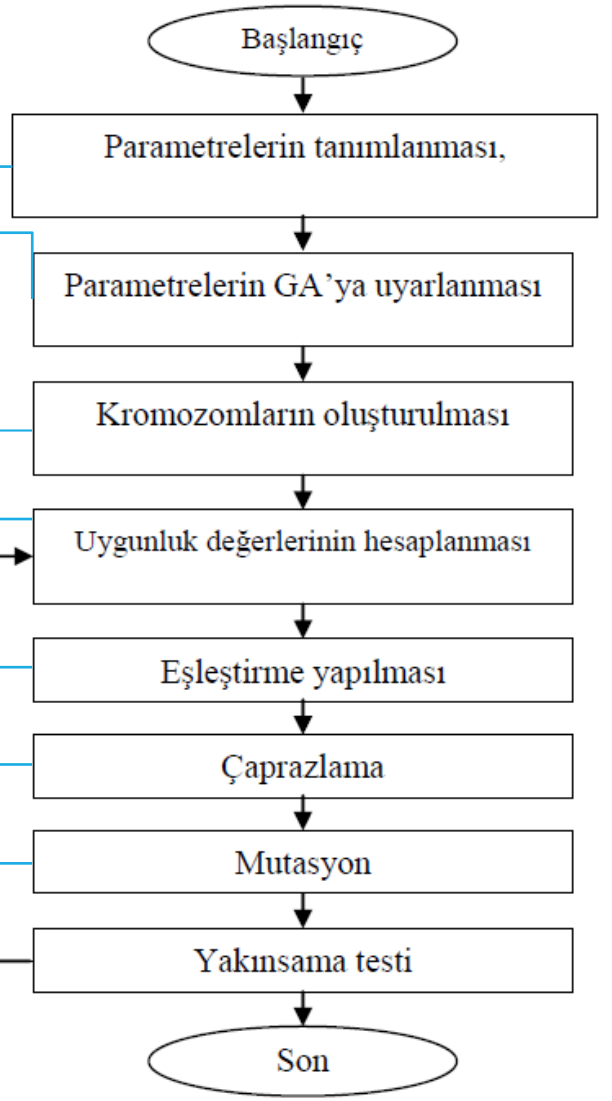
• Pozisyona göre değişim : Rasgele seçilen karakterlerin(genlerin) yerleri değiştirilerek gerçekleştirilir.

• Sıraya göre değişim : Kromozomun rasgele seçilen iki karakterinden ikincisinin, birincinin önüne getirilmesiyle olur

	Önce	Sonra
Pozisyona göre değişim	A B C D E F	F B C D E A
Sıraya göre değişim	A B C D E F	F A B C D E
Kromozom (Rast)	1 1 0 1 0 1 1 0	1 1 0 1 0 1 0 0

Dr. Mahir Kaya
TOGU -CENG

```
62  alt_sinir = -10
63  ust_sinir = 10
64  problem_boyutu = 4
65  populasyon_boyutu = 4
66
67  populasyon = np.random.ranf([populasyon_boyutu,problem_boyutu]) * (ust_sinir-alt_sinir) + alt_sinir
68
69  objit = list()
70
71  eniyideger = 100000000
72
73  for i in range(500):
74      obj = np.zeros(populasyon_boyutu)
75
76      for i in range(populasyon_boyutu):
77          obj[i]=fun(populasyon[i,:])
78
79      if min(obj) < eniyideger:
80          eniyideger = min(obj)
81          idx = np.where(obj==eniyideger)[0][0]
82          eniyicozum = populasyon[idx:]
83
84      objit.append(eniyideger)
85
86      araPopulasyon = dogalSecilim(populasyon,obj,populasyon_boyutu)
87      araPopulasyon = caprazla(araPopulasyon,populasyon_boyutu,0.95,problem_boyutu)
88      populasyon = mutasyon(araPopulasyon,0.05,populasyon_boyutu,
89                            problem_boyutu,0.05,ust_sinir,alt_sinir)
```



```python
def dogalSecilim(populasyon, obj, populasyon_boyutu):
    obj = 1/obj
    sumObj = sum(obj)
    probs = obj / sumObj
    cprobs = probs

    for i in range(1,populasyon_boyutu):
        cprobs[i] = cprobs[i-1] + probs[i]


    rs = np.random.ranf(populasyon_boyutu)
    araPopulasyon = populasyon.copy()

    for i in range(populasyon_boyutu):
        idx = np.argwhere(rs[i]<cprobs)[0][0]
        araPopulasyon[i,:] = populasyon[idx,:].copy()

    return araPopulasyon
```

Rulet tekeri seçim yönteminde amaç fonksiyonunun daha az olduğu bireyler daha değerli olduğu için çarpma işlemine göre tersini aldık.

Çıkan oranları kümülatif olarak topladık.

Birey sayısınca rasgele sayı ürettik.

Üretilen rasgele sayılara göre, daha önce kümülatif toplama göre oluşturduğumuz oranlara göre seçim işlemini gerçekleştirdik.

```python
def caprazla(araPopulasyon, populasyonBoyutu, caprazlamaOlasiligi, problem_boyutu):
    ciftler = np.random.permutation(populasyonBoyutu)

    for i in range(populasyonBoyutu // 2):
        parent1idx = ciftler[2*i]
        parent2idx = ciftler[2*i+1]
        parent1 = araPopulasyon[parent1idx,:]
        parent2 = araPopulasyon[parent2idx,:]

        rs = np.random.ranf()

        if rs < caprazlamaOlasiligi:
            caprazlamaNoktasi = np.random.randint(0, problem_boyutu)
            temp = parent1[caprazlamaNoktasi:].copy()
            parent1[caprazlamaNoktasi:] = parent2[caprazlamaNoktasi:].copy()
            parent2[caprazlamaNoktasi:] = temp
            araPopulasyon[parent1idx,:] = parent1
            araPopulasyon[parent2idx,:] = parent2

    return araPopulasyon
```

Popülasyondaki bireyler, çiftler halinde eşleniyor.

Rasgele bir sayı üretiliyor. Bu sayı çaprazlama olasılığından küçükse çaprazlama işlemimiz gerçekleştiriliyor.

Her çift birey için, rasgele bir çaprazlama noktası seçilip, o noktadan itibaren gen değişimi gerçekleştiriliyor.

Dr. Mahir Kaya
TOGU -CENG

```
50  def mutasyon(araPopulasyon,mutasyonOlasiligi,populasyon_boyutu,problem_boyutu,delta,ust_sinir,alt_sinir):
51
52      rs = np.random.ranf([populasyon_boyutu,problem_boyutu])
53
54      for i in range(populasyon_boyutu):
55          for j in range(problem_boyutu):
56              if rs[i,j] < mutasyonOlasiligi:
57                  rs2 = 2*np.random.ranf()-1
58                  araPopulasyon[i,j] = araPopulasyon[i,j] + rs2*delta*(ust_sinir-alt_sinir)
59
60      return araPopulasyon
```

Popülasyondaki her bireydeki her gen için rasgele sayı üretiliyor.

Oluşturulan rasgele sayı mutasyon olasılığından küçükse mutasyon işlemi yapıyor.

Mutasyon işlemi seçilen gende rasgele bir artırma veya azaltma şeklinde gerçekleştiriliyor.

```python
import numpy as np
import matplotlib.pyplot as plt


def fun(x):
    return sum(np.power(x,2))

def dogalSecilim(populasyon, obj, populasyon_boyutu):
    obj = 1/obj
    sumObj = sum(obj)
    probs = obj / sumObj
    cprobs = probs

    for i in range(1,populasyon_boyutu):
        cprobs[i] = cprobs[i-1] + probs[i]


    rs = np.random.ranf(populasyon_boyutu)
    araPopulasyon = populasyon.copy()

    for i in range(populasyon_boyutu):
        idx = np.argwhere(rs[i]<cprobs)[0][0]
        araPopulasyon[i,:] = populasyon[idx,:].copy()

    return araPopulasyon


def caprazla(araPopulasyon, populasyonBoyutu, caprazlamaOlasiligi, problem_boyutu):
    ciftler = np.random.permutation(populasyonBoyutu)

    for i in range(populasyonBoyutu // 2):
        parent1idx = ciftler[2*i]
        parent2idx = ciftler[2*i+1]
        parent1 = araPopulasyon[parent1idx,:]
        parent2 = araPopulasyon[parent2idx,:]

        rs = np.random.ranf()

        if rs < caprazlamaOlasiligi:
            caprazlamaNoktasi = np.random.randint(0, problem_boyutu)
            temp = parent1[caprazlamaNoktasi:].copy()
            parent1[caprazlamaNoktasi:] = parent2[caprazlamaNoktasi:].copy()
            parent2[caprazlamaNoktasi:] = temp
            araPopulasyon[parent1idx,:] = parent1
            araPopulasyon[parent2idx,:] = parent2

    return araPopulasyon
```

```python
def mutasyon(araPopulasyon,mutasyonOlasiligi,populasyon_boyutu,problem_boyutu,delta,ust_sinir,alt_sinir):

    rs = np.random.ranf([populasyon_boyutu,problem_boyutu])

    for i in range(populasyon_boyutu):
        for j in range(problem_boyutu):
            if rs[i,j] < mutasyonOlasiligi:
                rs2 = 2*np.random.ranf()-1
                araPopulasyon[i,j] = araPopulasyon[i,j] + rs2*delta*(ust_sinir-alt_sinir)

    return araPopulasyon


alt_sinir = -10
ust_sinir = 10
problem_boyutu = 4
populasyon_boyutu = 4

populasyon = np.random.ranf([populasyon_boyutu,problem_boyutu]) * (ust_sinir-alt_sinir) + alt_sinir

objit = list()

eniyideger = 100000000

for i in range(500):
    obj = np.zeros(populasyon_boyutu)

    for i in range(populasyon_boyutu):
        obj[i]=fun(populasyon[i,:])

    if min(obj) < eniyideger:
        eniyideger = min(obj)
        idx = np.where(obj==eniyideger)[0][0]
        eniyicozum = populasyon[idx:]

    objit.append(eniyideger)

    araPopulasyon = dogalSecilim(populasyon,obj,populasyon_boyutu)
    araPopulasyon = caprazla(araPopulasyon,populasyon_boyutu,0.95,problem_boyutu)
    populasyon = mutasyon(araPopulasyon,0.05,populasyon_boyutu,
                          problem_boyutu,0.05,ust_sinir,alt_sinir)


plt.plot(objit)
plt.xlabel("iterasyon")
plt.show()
```

Dr. Mahir Kaya
TOGU -CENG

Kaynaklar

http://kergun.baun.edu.tr/20172018Guz/YZ_Sunumlar/Genetik_Algoritmalar_Busra_Guracar.pdf

ARSLAN, M., DİFERANSİYEL EVRİM ALGORİTMASI YARDIMIYLA ASENKRON MOTOR PARAMETRELERİNİN BELİRLENMESİ, Yüksek Lisans Tezi, 2010.

https://ab.org.tr/ab16/sunum/202.pdf