GENETİK ALGORİTMALAR

1

GENETİK ALGORİTMALAR

- Genetik algoritmalar, Darwin'in doğal seçim ve evrim teorisi ilkelerine dayanan bir arama ve optimizasyon yöntemidir.
- Bu yöntem ilk olarak, John Holland ve arkadaşlarının yaptığı çalışmalarda (1970'li yıllarda) ortaya çıkmıştır.
- Geleneksel optimizasyon yöntemlerine göre farklılıkları olan genetik algoritmalar, parametre kümesini değil kodlanmış biçimlerini kullanırlar.

2

GENETİK ALGORİTMALAR

- 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.

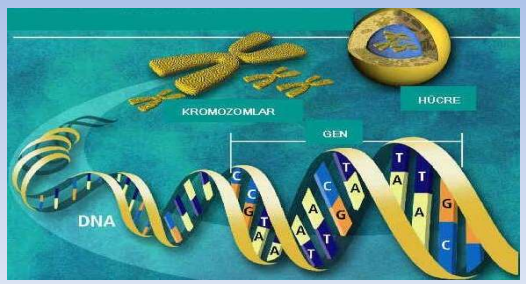
3

GENETİK ALGORİTMALAR

- Olasılık kurallarına göre çalışan genetik algoritmalar, yalnızca amaç fonksiyonuna gereksinim duyar. Çözüm uzayının tamamını değil belirli bir kısmını tararlar.
- Böylece, etkin arama yaparak çok daha kısa bir sürede çözüme ulaşırlar.
- Diğer bir önemli üstünlükleri ise; çözümlerden oluşan popülasyonu eş zamanlı incelemeleri ve böylelikle yerel en iyi çözümlere takılmamalarıdır.

4

Genetik Algoritmaların Biyolojik Altyapısı



5

GENETİK ALGORİTMALAR

- Gen: Kalıtsal molekülde bulunan ve organizmanın karakterlerinin tayininde rol oynayan kalıtsal birimlere denir.
- Bir gen A,B gibi bir karakter olabileceği gibi 0 veya 1 ile ifade edilen bir bit veya bit dizisi olabilir. Örneğin bir cismin x koordinatındaki yerini gösteren bir gen **101** şeklinde ifade edilebilir.

6

GENETİK ALGORİTMALAR

- Kromozom (Birey): Birden fazla genin bir araya gelerek oluşturduğu diziye denir.
- Kromozomlar toplumdaki bireyler yada üyelere karşılık gelirler.
- Ele alınan problemde alternatif çözüm adayıdır.
- Örneğin kromozom bir problemde açı, boyut ve koordinat değişkenlerinden veya bir dikdörtgen prizmasının ölçülerinden (yükseklik, genişlik, derinlik) oluşabilir.
- 001 101 111 -> 1,5,7 değerleri kromozomu oluşturan genlerdir.

- Popülasyon: Kromozomlardan oluşan topluluğa denir. Popülasyon üzerinde durulan problem için alternatif çözümler kümesidir.
- Popülasyondaki kromozom sayısı arttıkça çözüme ulaşma süresi azalır.

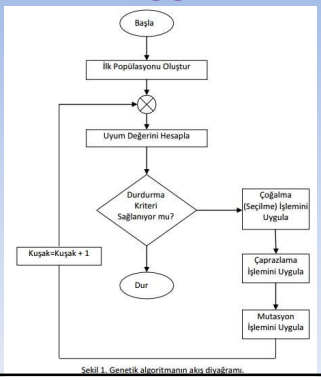
7

GENETİK ALGORİTMALAR
Temel Kavramlar

- Genetik algoritmalarda kromozomlar, problem için olası çözümleri temsil ederler.
- Uygunluk değeri, çözümün kalitesini belirler ve uygunluk fonksiyonu kullanılarak hesaplanır.
- Yeni nesiller oluşturmak için, olası çözümleri temsil eden kromozomlara genetik bilimindekine benzer bir biçimde çaprazlama ve mutasyon işlemleri uygulanır.

8

GENETİK ALGORİTMALAR



Şekil 1. Genetik algoritmanın akış diyağramı.

9

GENETİK ALGORİTMALAR

```
begin
  t = 0;
  Pt başlangıç yığınını oluştur;
  Pt' yi değerlendir;
  while not {bitiş koşulu} do
    begin
      t = t +1;
      Pt-1' den Pt' yi seç ;   { Yeniden üretim operatörü }
      Pt' yi değişime uğrat ;  { Çaprazlama ve mutasyon operatörü }
      Pt' yi değerlendir ;
    end
end
```

Genetik Algoritmaların Genel Yapısı

10

GENETİK ALGORİTMALAR
Genel Yapısı

- Basit bir GA'nın ilk aşamasında, tüm mümkün çözümlerin alt kümesinden oluşan bir başlangıç yığını elde edilir.
- Yığının her elemanı (bireyi) bir dizi olarak kodlanır.
- Her dizi biyolojik olarak bir kromozoma eşdeğerdir.
- GA'nın herhangi bir adımındaki yığın, nesil (generation) olarak adlandırılır.
- Yığındaki her dizi bir uygunluk değerine (fitness value) sahiptir.

11

GENETİK ALGORİTMALAR

- Uygunluk değeri, hangi bireyin bir sonraki yığına taşınacağını belirler.
- Bir dizinin uygunluk değeri, problemin amaç fonksiyonu değerine eşittir.
- Bir dizinin gücü uygunluk değerine bağlı olup iyi bir dizi, problemin yapısına göre maksimizasyon problemi ise yüksek, minimizasyon problemi ise düşük uygunluk değerine sahiptir.

12

GENETİK ALGORİTMALAR

- 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.

13

GENETİK ALGORİTMALAR

- Genetik Algoritmaları uygulama aşamasında aşağıdaki adımlara karar verilmesi son derece önemlidir:
- Kodlama
- Çözümleri Değerlendirme (Uygunluk Fonksiyonu)
- Birey Üretimi (Çaprazlama ve Mutasyon, Seçilim ve üretilen bireylerin popülasyona eklenmesi)

14

GENETİK ALGORİTMALAR

KODLAMA:

- Çözümdeki parametrelerin (gen'lerin) nasıl temsil edileceğidir. Bu parametreler, yani genler ikili temsil, tamsayılar, kayan noktalı sayılar, ağaç veri yapısı, dizi vs. olarak temsil edilebilmektedir.
- Bu temsil işlemi probleme bağlı olan bir işlemdir.
- Parametrelerin kodlanması, probleme özgü bilgilerin genetik algoritmanın kullanacağı şekle çevrilmesine olanak tanır.

15

GENETİK ALGORİTMALAR

- **İkili kodlamada** her kromozom 1 ve 0'lardan oluşan bir karakter dizisi şeklinde ifade edilir.
- **Değer Kodlama :** Bu kodlama gerçel gibi kompleks sayıların yer aldığı problemlerde kullanılır.
- **Permütasyon kodlamada** ise her kromozom, ilgili karakterin sıralamadaki pozisyonunu belirten sayılardan oluşan bir dizi ile ifade edilir.
- Permütasyon kodlama, genelde sıralama problemlerinde kullanılır.

16

GENETİK ALGORİTMALAR

•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

17

GENETİK ALGORİTMALAR

ÇÖZÜMLERİ DEĞERLENDİRME (UYGUNLUK FONKSİYONU)

- GA'daki her çözümün yani kromozomun, problemi ne derecede çözebildiğini hesaplamamızı sağlayan fonksiyona "uygunluk fonksiyonu" denir.
- Uygunluk fonksiyonu problem bağımlı bir fonksiyondur ve GA'nın en önemli kavramlarından biridir.

18

GENETİK ALGORİTMALAR

SEÇİLİM

- 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.

19

GENETİK ALGORİTMALAR

- 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.

20

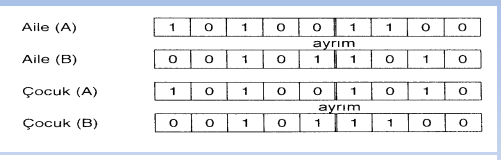
GENETİK ALGORİTMALAR

ÇAPRAZLAMA

- Amaç, ata kromozomun yerlerini değiştirerek çocuk kromozomlar üretmek ve böylelikle zaten uygunluk değeri yüksek olan ata kromozomlardan daha yüksek uygunluklu çocuk kromozomlar üretmektir.
- Çaprazlamanın en kolay yolu rastgele bir çaprazlama noktası belirleyip, bu noktadan önceki bölümü ilk ebeveynden, sonraki bölümü ise diğer ebeveynden alarak yeni bir birey oluşturmaktır.

21

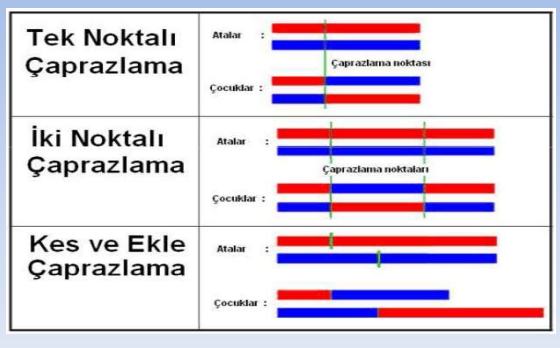
GENETİK ALGORİTMALAR



	1	2	3	4	5	6	7	8	9	10	11	12
A_1 :	0	1	0	0	1	1	1	0	1	0	1	0
A_2 :	1	0	0	1	1	1	0	0	0	1	0	0
A_1' :	0	1	0	0	1	1	1	0	0	1	0	0
A_2' :	1	0	0	1	1	1	0	0	1	0	1	0

22

GENETİK ALGORİTMALAR



23

GENETİK ALGORİTMALAR

MUTASYON

- Yeniden ve sürekli yeni nesil üretimi sonucunda, belli bir süre sonra nesildeki kromozomlar birbirlerini tekrar edebilir. Böylece farklı kromozom üretimi durur veya azalır.
- İşte bu nedenle nesildeki kromozom çeşitliliğini artırmak için kromozomlardan bazıları mutasyona tabi tutulur.
- Mutasyon olasılığı çok düşük (Ör:%0.01) tutulmalıdır.
- Yüksek mutasyon olasılığı uygun çözümlerin de bozulmasına yol açabilir.

24

GENETİK ALGORİTMALAR

MUTASYON

	1	2	3	4	5	6	7	8	9	10	11	12
A_1 :	0	**1**	0	0	1	1	1	0	1	**0**	1	0
A_1' :	0	**0**	0	0	1	1	1	0	1	**1**	1	0

Mutasyona Bir Örnek

Şekilde verilen örnek, bir diziye mutasyon operatörünün uygulanışını göstermektedir. A_1 dizisinin 2. ve 10. elemanları mutasyona uğratılarak A_1' dizisi elde edilir.

25

GENETİK ALGORİTMALAR
Performansı Etkileyen Faktörler

- Popülasyon büyüklüğü / Kromozom sayısı: Kromozom sayısını arttırmak çalışma zamanını arttırırken, azaltmak da kromozom çeşitliliğini yok eder.
- Mutasyon Oranı: Kromozomlar birbirine benzemeye başladığında hala çözüm noktalarının uzağında bulunuyorsa mutasyon işlemi GA'nın sıkıştığı yerden (tüm kromozomlar aynı platoda) kurtulmak için tek yoludur. Ancak yüksek bir değer vermek GA'nın kararlı bir noktaya ulaşmasını engelleyecektir.
- Kaç Noktalı Çaprazlama Yapılacağı: 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 birden kullanılıp kullanılamayacağı bazen önemli olmaktadır.

26

GENETİK ALGORİTMALAR
Performansı Etkileyen Faktörler

- Durum kodlanmasının nasıl yapıldığı: Bir parametrenin doğrusal yada logaritmik kodlanması GA'nın performansında önemli bir farka yol açabilir.
- Başarı değerlendirmesinin nasıl yapıldığı: Akıllıca yazılmamış bir değerlendirme işlevi, çalışma zamanını uzatabileceği gibi çözüme hiçbir zaman ulaşılamamasına da neden olabilir.

27

GENETİK ALGORİTMALAR

Avantajları

- Çok amaçlı optimizasyon yöntemleri ile kullanılabilmesi
- Çok karmaşık ortamlara uyarlanması
- Kısa sürelerde iyi sonuçlar verebilmesi
- Çaprazlama ve mutasyon sayesinde yerel maksimum ve platolardan kurtulabilir.

28

GENETİK ALGORİTMALAR

Dezavantajları

- Son kullanıcının modeli anlaması güç
- Problemi GA ile çözmeye uygun hale getirmek zor
- Uygunluk fonksiyonunu belirlemek zor
- Çaprazlama ve mutasyon tekniklerini belirlemek zor
- Ayarlanabilir çok fazla parametreye sahip
- Bir problemde gösterdiği başarıyı başka bir problemde tekrarlaması zor

29

GENETİK ALGORİTMALAR
Uygulama Alanları

- **Optimizasyon**
- **Otomatik Programlama**
- **Makine Öğrenmesi** (Yapay Sinir Ağlarında öğrenmeyi sağlayan ağırlık hesaplamaları, robotlar)
- **Ekonomi**
- **Tıp**
- **Ekoloji**
- **Sosyal Sistemler** (Böcek Kolonileri, Çok etmenli sistemlerde işbirliği)

30

Örnek Problem Çözümü

- Goldberg Problemi…
- Amaç: $f(x)=x^2$, $x=[0,31]$ şeklinde verilen bir fonksiyonun, verilen aralıkta maksimizasyonunun yapılması istenmektedir.

31

Örnek Problem Çözümü

- **Adım 1: Başlangıç popülasyonunun oluşturulması…**
- İlk olarak x sayısının kodlanması işlemi yapılmalıdır. x'in 0 ve 1'lerden oluşan 2 tabanındaki gösterimi kullanılacaktır.
- Dolayısıyla x, 5 bit uzunluğunda bir kodla (string) temsil edilecektir. Öyle ki 0: "00000" ve 31: "11111" olacaktır.

32

Örnek Problem Çözümü

Adım 2: ***Popülasyon içindeki her kromozomun uygunluk değerinin hesaplanması…***

Toplumun birey sayısı n:4 olarak seçilmiştir. Toplumu oluşturan dört birey, her biri 5 bit uzunluğunda birer kromozomla temsil edildiği için toplam 20 kere yazı tura atmak suretiyle belirlenmiştir. Elde edilen birey kromozomları ve uygunluk değerleri aşağıdadır.

Birey 1: 01101, x = 13 , x^2 = 169
Birey 2: 11000, x = 24 , x^2 = 576
Birey 3: 01000, x = 8 , x^2 = 64
Birey 4: 10011, x = 19 , x^2 = 361

33

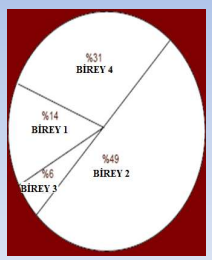
Örnek Problem Çözümü

- **Adım 3: Tekrar üretme, çaprazlama ve mutasyon işleminin uygulanması için seçme işlemi…**
- Yukarıda belirlenen bireyler için $f(x)=x^2$, bireylerin uygunluk değerlerini verir.
- Dört bireyin toplam uygunluk değerleri "169+576+64+361=1170" dir. Dolayısıyla her bir bireyin rulet tekerleğinde kaplayacağı alan şu şekilde hesaplanır:
- Birey 1: 169/1170=0.14 : %14
- Birey 2: 576/1170=0.49 : %49
- Birey 3: 64/1170=0.06 : %6
- Birey 4: 361/1170=0.31 : %31

34

Örnek Problem Çözümü

- Bu değerler, rulet tekerleğinin her çevrilişinde hangi olasılıkla hangi bireyin seçileceğini belirtir, örneğin 0.14 olasılıkla 1 numaralı birey seçilecektir.
- Rulet tekerleği ve bireylerin tekerlek üzerindeki dağılımları şekilde gösterilmiştir.
- Toplumda ki birey sayısının sabit kaldığı varsayıldığından dolayı, rulet tekerleği 4 kere çevrilerek çaprazlama havuzu oluşturulacaktır.
- Rulet tekerleği döndürülmüş ve şu sonuçlar elde edilmiştir:



35

Örnek Problem Çözümü

Bunun sonucunda elde edilen çaprazlama havuzu şu şekildedir;

- Aday 1 : 01101 (Birey 1)
- Aday 2 : 11000 (Birey 2)
- Aday 3 : 11000 (Birey 2)
- Aday 4 : 10011 (Birey 4)

36

Örnek Problem Çözümü

Adım 4: ***Çaprazlama işleminin uygulanması…***

Çaprazlama havuzu belirlendikten sonra iki aşamalı çaprazlama uygulanır. İlk aşamada adaylar çaprazlanmak üzere rastgele olarak eşlenirler. Her ikili grup için bir kere zar atılarak çaprazlaşmanın oluşacağı nokta belirlenir. Rastgele eşleştirme yapılmış ve bunun sonucunda, (Aday 1, Aday 2) ve (Aday 3, Aday 4) ikili grupları oluşmuştur. Çaprazlaşma noktaları da zar atılarak 1. Grup için k=4 ve 2. Grup içinde k=2 olarak belirlenmiştir. Bu aşamadan sonra çaprazlaşma gerçekleştirilmiş ve şu sonuçlar oluşmuştur;

37

Örnek Problem Çözümü

Çaprazlama grubu 1: (k=4)

- Aday 1 : 0110/1 oluşan Birey 1 : 01100
- Aday 2 : 1100/0 oluşan Birey 2 : 11001

Çaprazlama grubu 2 : (k=2)

- Aday 3 : 11/000 oluşan Birey 3 : 11011
- Aday 4 : 10/011 oluşan Birey 4 : 10000

38

Örnek Problem Çözümü

Adım 5: ***Mutasyon işleminin uygulanması…***

Bu aşamada, mutasyon bitler düzeyinde uygulanır. Bu örnekte her bir bit için (toplam 20 bit var) mutasyon olma olasılığı 0.01 olarak seçilmiştir. Dolayısıyla her bir bit için ağırlıklı yazı/tura (mutasyon olasılığına göre) atılarak hangi bitlerin mutasyona uğrayacağı belirlenir. Bu işlem yapılmış ve sonuçta oluşan birey 3'ün 2 numaralı bitinde mutasyon olacağı ortaya çıkmıştır.

39

Örnek Problem Çözümü

Çaprazlama Sonucu Oluşan Birey 3 : 11011

Mutasyon sonucu oluşan Birey 3 : 10011

Bu adımın tamamlanmasıyla bir sonraki kuşağı oluşturacak toplumun bireyleri belirlenmiş olur. Yeni toplum şu şekildedir;

Birey 1 : 01100 x=12, x^2=144
Birey 2 : 11001 x=25, x^2=625
Birey 3 : 10011 x=19, x^2=361
Birey 4 : 10000 x=16, x^2=256

40

Örnek Problem Çözümü

Adım 6: ***Kötü kromozomların popülasyondan elenmesi…***

Yeni oluşturulan kromozomların değerleri 2. adımda yapıldığı gibi hesaplanır. Bu işlemden sonra en kötü değerli kromozom popülasyondan elenecektir.

Eleme işlemi popülasyon sayımız sabit olduğundan popülasyondan atma şeklinde değil, en kötü kromozomu değiştirme şeklinde olacaktır.

Yeni popülasyon içerisindeki uygunluğu en düşük olan Birey 1 (01100) 'in kendini yeniden üretme olasılığı en düşük olduğundan değişime uğratılarak elenmesi gerekmektedir.

41

Örnek Problem Çözümü

Adım 7: ***2 – 6. arasındaki adımların tekrarı…***

Genetik algoritmaların değişkenlerinden biride, yazılan programın kaç adım çalıştırılacağıdır. Bu sayı problemin büyüklüğüne göre değişmekle birlikte, genellikle 50 – 500 aralığında bir değerdir.

Bu örnekte tek bir iterasyon yapılmış ve başlangıç toplumundan bir sonraki kuşak oluşturulmuştur, ancak genetik algoritmanın çalışmasının tam olarak gözlenebilmesi için tek bir iterasyon yeterli değildir.

42

Örnek Problem Çözümü

İşlemler tekrarlanarak aşağıdaki nesile ulaşılacaktır:

Birey 1 : 11111, x=31, x^2=961
Birey 2 : 11111, x=31, x^2=961
Birey 3 : 11111, x=31, x^2=961
Birey 4 : 11111, x=31, x^2=961

Amaç fonksiyonumuzun maksimizasyonunu sağlayan değer 31 'dir.

43

Farklı bir problem

source = "jiKnp4bqpmAbp"
target = "Hello, World!"

44

Farklı bir problem

```
def fitness(source, target):
   fitval = 0
   for i in range(0, len(source)):
      fitval += (ord(target[i]) - ord(source[i])) ** 2
   return(fitval)
```

Kaynak ve hedef karakterler arasındaki mesafenin karesi
Fitness = 0 → Mükemmel sonuç
Fitness = 1 → Bir karakter farklı "Hfllo" and "Hdllo"

45

Farklı bir problem

```
def mutate(source):
   charpos = random.randint(0, len(source) - 1)
   parts = list(source)
   parts[charpos] = chr(ord(parts[charpos]) +
 random.randint(-1,1))
   return(''.join(parts))
```

Kaynaktan rasgele bir karakter seçilir ve pozisyonu 1 arttırılır veya 1 azaltılır.

46

Farklı bir problem

```
fitval = fitness(source, target)
i = 0
while True:
   i += 1
   m = mutate(source)
   fitval_m = fitness(m, target)
   if fitval_m < fitval:
      fitval = fitval_m
      source = m
      print "%5i %5i %14s" % (i, fitval_m, m)
   if fitval == 0:
      break
```

Her bir iterasyonda, string mutasyona uğratılır ve yeni bir string oluşturularak fitness değeri hesaplanır. Bu değer orjinal (parent) değerden daha iyi ise yeni string seçilir, değilse atılır. Fitness değeri 0 ise hedefe ulaşılmıştır.

47

Farklı bir problem

```
    1 15491  jjKnp4bqpmAbp
   20 15400  jiKnp3bppoAbp
   40 15377  jiKlo2bpooAdp
   60 15130  iiKlo2aoooAdp
```

48

Farklı bir problem

500 9986 \eTlo,YaorNdf

1200 4186 Heglo,LWorhdP

1500 3370 Hello,GWorldL

49

Farklı bir problem

3078 2 Hello, Vorld"
3079 2 Hfllo, World"
3080 2 Hfllo, World"
3081 0 Hello, World!

50

Farklı bir problem

Bu basit örnekte gen havuzunda sadece 1 organizma var ve sadece 1 karakter mutasyona uğruyor. Bunu biraz daha geliştirelim.

51

Farklı bir problem

Fitness fonksiyonuna dokunmuyoruz.

Gen havuzunu büyütelim

52

Farklı bir problem

```
GENSIZE = 20
genepool = []
for i in range(0, GENSIZE):
    dna = [random.choice(string.printable[:-5]) for j in range(0, len(target))]
    fitness = calc_fitness(dna, target)
    candidate = {'dna': dna, 'fitness': fitness }
    genepool.append(candidate)
```

20 rastgele string seçilir ve fitness değerleri hesaplanır. Buna aynı zamanda popülasyon da diyebiliriz.

53

Farklı bir problem

```
1   7617  'iSx{$,K`u~(B
1   9284  SQf`1N#UdrPIT
1  12837  sYIu<E"Fq'^_.
1  15531  DC8Dg1I$*mUs-
1  16064  L~*}JBVdF7bu2
1  16533  1,XU%)5$q[YuO
1  16588  ff],ceW<0fud&
1  17316  [V3@2'VgY\{KV
1  17356  kWw#v/P<#apG9
1  17581  <Lrh(1hN_Bd)3
1  18777  TM]_]TbtxFY:q
1  19656  $zS+EI?BS>%z(
1  19841  =S;B~((W8 D,6
1  20398  P_A$D|NPJPio/
1  21957  J&f=O:g\8'{S2
1  22543  5*T2c"pMZ80L'
1  24954  A&lZ#A_}MxI"P
1  25186  &9MrI|0&x)q,N
1  28110  OIXT/Q{y3{"LR
1  29656  8WB99hx%0]}h[
```

54

Farklı bir problem

```
def mutate(parent1, parent2):
    child_dna = parent1['dna'][:]

    # Mix both DNAs
    start = random.randint(0, len(parent2['dna']) - 1)
    stop = random.randint(0, len(parent2['dna']) - 1)
    if start > stop:
        stop, start = start, stop
    child_dna[start:stop] = parent2['dna'][start:stop]

    # Mutate one position
    charpos = random.randint(0, len(child_dna) - 1)
    child_dna[charpos] = chr(ord(child_dna[charpos]) + random.randint(-1,1))
    child_fitness = calc_fitness(child_dna, target)
    return({'dna': child_dna, 'fitness': child_fitness})
```

Mutasyon fonksiyonunu geliştirelim. Sadece tek bir karakteri mutasyona uğratmak yerine popülasyondan rasgele 2 bireyi alıp ikisini **çaprazlayalım.** Daha sonra tek karakteri **mutasyona** uğratıp fitness değerini hesaplayalım.

55

Farklı bir problem

```
2   7617  'iSx{$,K`u~(B
2   8742  SQf`1N#UdfumT
2   9284  SQf`1N#UdrPIT
2  12837  sYIu<E"Fq'^_.
2  15531  DC8Dg1I$*mUs-
2  16064  L~*}JBVdF7bu2
2  16533  1,XU%)5$q[YuO
2  16588  ff],ceW<0fud&
2  17316  [V3@2'VgY\{KV
2  17356  kWw#v/P>#apG9
2  17581  <Lrh(1hN_Bd)3
2  18777  TM]_]TbtxFY:q
2  19656  $zS+EI?BS>%z(
2  19841  =S;B~((W8 D,6
2  20398  P_A$D|NPJPio/
2  21957  J&f=O:g\8'{S2
2  22543  5*T2c"pMZ80L'
2  24954  A&lZ#A_}MxI"P
2  25186  &9MrI|0&x)q,N
2  28110  OIXT/Q{y3{"LR
```

56

Farklı bir problem

```
6   7617  'iSx{$,K`u~{B
6   8742  SQf`1N#UdfumT
6   9284  SQf`1N#UdrPlT
6  10198  SQfD1N#UdfumT
6  12837  sYIu<E"Fq'^_.
6  15531  DC8Dg1I$*mUs-
6  16064  L~*}JBVdF7bu2
6  16387  SQf`1N"MZ80LT
6  16533  1,XU%)5$q[YuO
6  16588  ff],ceW<0fud&
6  17316  [V3@2'VgY\{KV
6  17356  kWw#v/P>#apG9
6  17356  kWw#v/P>#apG9
6  17581  <Lrh(1hN_Bd}3
6  18777  TM]_]TbtxFY:q
6  19656  $zS+El?BS>%z(
6  19841  =S;B~((W8 D,6
6  20287  fe],1eW<0fud&
6  20398  P_A$D|NPJPio/
6  21957  J&f=O:g\8'{S2
```

57

Farklı bir problem

```
40  3306  RQSw{$-KcfumB
40  4204  RQf`{$,KdfumT
40  4229  RQf`|$,KdfumT
40  4242  RQe`|$,KdfumT
40  4795  RQSw{$-KdfumT
40  4971  RQSwz$*K`uSnT
40  4973  RQSwz$+K`uSmT
40  4992  RQSwz$+K`uSnT
40  5017  SQSxz$+K`uSmT
40  5017  SQSxz$+K`uSmT
40  5951  (QSxz$+KdfSmT
40  5985  'QSxz$+K`uSmT
40  6421  SQfx{$+K`u~(B
40  6444  TQf`{$+K`u~(B
40  6489  SQfx{$+KdfS(B
40  6492  TQf`{$-K`u~(B
40  7034  SQSxy$+KdfS(B
40  7617  'iSx{$,K`u~(B
40  7617  'iSx{$,K`u~(B
40  7625  'iS`{$,Kdg~(B
```

58

Farklı bir problem

```
67  3138  RQSw{$+KdfukA
67  3161  RQSw{$+KcfukA
67  3176  RQSw{$,KdfulA
67  3176  RQSw{$+KcfulA
67  3218  RQSw{$-LcfumA
67  3222  RQSw{%,KefumB
67  3237  RQSw{$-LcfvmA
67  3241  RQSw{$-KcfumA
67  3241  RQSw{$-KcfumA
67  3266  RQSw{$-KceumA
67  3266  RQSw{$-KceumA
67  3267  RRSw{$-KcfumB
67  3289  RQSw{%,KefumC
67  3306  RQSw{$-KcfumB
67  3306  RQSw{$-KcfumB
67  3323  RQSw{#-KcfumB
67  3324  RPSw{$-KdfumB
67  3331  RQSw{$-KbfumB
67  3348  RQSw{#-KbfumB
67  3489  RQSw{$+KdfumA
```

59

Farklı bir problem

```
1600   19  Hdllo+ Worle%
1600   20  Hdklo+ Worle%
1600   20  Hdklo+ Worle%
1600   20  Hdklo+ Worle%
1600   20  Hdklo+ Worle%
1600   20  Hdklo+ Workd%



1904    0  Hello, World!
1904    1  Hello, World"
1904    1  Hello, World"
1904    2  Hello, Wprld"
1904    2  Helmo, World"
1904    2  Helmo, World"
1904    2  Hdllo, World"
1904    2  Hello, Worle"
```

60

Örnek

https://youtu.be/uwz8JzrEwWY

https://www.youtube.com/watch?v=FKbarpAlBkw

https://www.youtube.com/watch?v=u2t77mQmJiY

61