Abdulkadir TÜRE

abdulkadir.ture@std.yildiz.edu.tr

GENETİK ALGORİTMA

https://youtu.be/djHNyB3Qyb0

YAPISAL PROGRAMLAMAYA GİRİŞ DERSİ

Ders Yürütücüsü

DOÇ. DR. MEHMET FATİH AMASYALI

İÇİNDEKİLER

GENETİK ALGORİTMA NEDİR ?

Genetik algoritmalar, doğada gözlemlenen evrimsel mekanizmalara benzer mekanizmalar kullanarak çalışan eniyileştirme yöntemidir. Çok boyutlu uzayda belirli bir maliyet fonksiyonuna göre en iyileştirme amacıyla iterasyonlar yapan ve her iterasyonda en iyi sonucu üreten kromozomun hayatta kalması prensibine dayanan en iyi çözümü arama yöntemidir.

Genetik algoritmaların temel ilkeleri ilk kez Michigan Üniversitesi'nde John Holland tarafından ortaya atılmıştır. Holland 1975 yılında yaptığı, evrim yasalarını genetik algoritma içinde eniyileştirme problemleri için kullandığı çalışmaları "Adaptation in Natural and Artificial Systems" adlı kitabında bir araya getirmiştir.

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 global çözüme ulaşma olasılığı yükselmektedir. Çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır.

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 eniyileme yöntemlerinden istenen sonucun alınmadığı alanlarda etkili ve kullanışlıdır.

Gezgin Satıcı Problemi Üzerinden Genetik Algoritmanın Modellenmesi

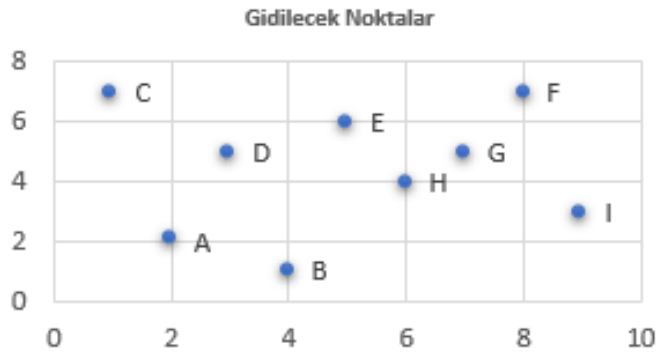
Bir kargo uçağı için haritada belirlen noktaları(N adet nokta) yalnızca bir kez ziyaret edecek şekilde en kısa rota oluşturulmak isteniyor.

Problemin zorluğu;

N = 30 için;

30!= 265.252.859.812.191.058.636.308.480.000.000 adet farklı rota bulunmaktadır. Bu rotaların tamamı için uzunluk hesaplanmalı ve en kısa uzunluğun hangi rotada olduğunun araması gerçeklenmelidir. Wikipedia kaynağına göre N=200 için bile şuanki bilgisayarlarla minimum uzunluk değerine sahip rotayı bulmak yıllar almaktadır.

Rotaları Kromozomlarla İfade Etmek

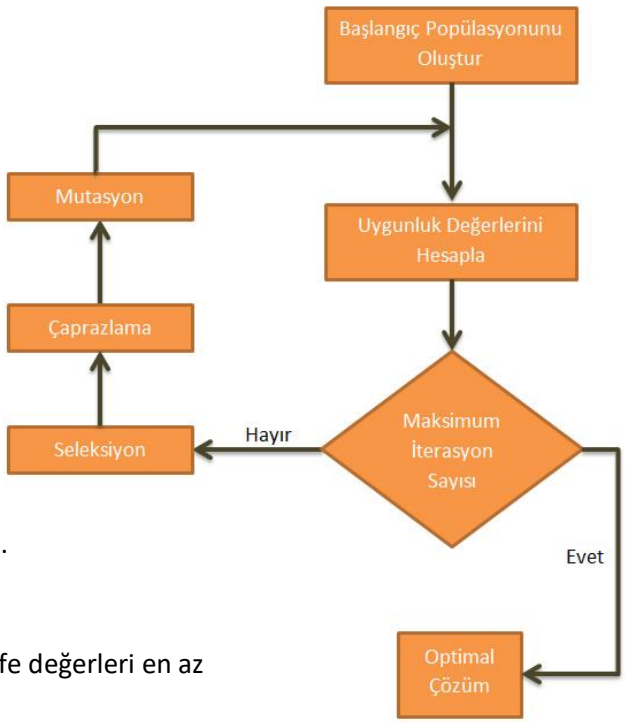


Genetik algoritma problem üzerinde modellenirken olası her rota kromozom olarak ifade edilir. Kromozom üzerindeki her gen ise olası rotadaki bir konumdur.

Yanda görülen noktalar kümesi için rastgele oluşturulan 2 kromozomun yapısı verilmiştir.

KROMOZOM	D	E	F	G	H	I	B	A	C
X	3	5	8	7	6	9	4	2	1
Y	5	6	7	5	4	3	1	2	7
KROMOZOM	A	B	C	H	I	F	G	E	D
X	2	4	5	6	9	8	7	1	3
Y	2	1	6	4	3	7	5	7	5

Genetik Algoritma Akış Şeması



Başlangıç Popülasyonu:

K Birey (K Kromozom(rota)) (Rastgele oluşturulacak).

Seçim:

Her kromozomun mesafe değerini hesaplanır. Mesafe değerleri en az olan ilk A birey seçilir. Diğerleri öldürülür.

Çaprazlama:

A Birey çaprazlanır ve yeni bireyler mutasyon aşamasına yönlendirilir.

Mutasyon:

Yeni A/2 birey mutasyona uğrar ve popülasyona (kromozom havuzuna) katılır.

Nesil Sayısı Kontrol(Nesil (Düngü sayısı) L' ye ulaştı mı? Kontrolü.)

Ulaşmadıysa Seçim aşamasına geri dön.

Ulaştıysa popülasyondaki en az mesafe değerine sahip olana kromozomu (rotayı) optimal çözüm olarak kabul et.

Kullanım Yerleri:

Genetik algoritmalar; Parametre ve sistem tanılama, kontrol sistemleri, robot uygulamaları, görüntü ve ses tanıma, mühendislik tasarımları, planlama, yapay zeka uygulamaları, uzman sistemler, fonksiyon ve kombinasyonel eniyileme problemleri ağ tasarım problemleri, yol bulma problemleri, çizelgeleme problemleri, sosyal ve ekonomik planlama problemleri için diğer eniyileme yöntemlerinin yanında başarılı sonuçlar vermektedir.

Örnek olarak;

Çok Kollu Robotların Çarpışmasız Hareketi:

Bu çalışmada birden fazla koldan oluşan robot sistemin sabit engellere çarpmadan hareketinin yanı sıra hareketli çevre ve engellerle de çarpışmadan hareket etmesi sağlanmaya çalışılmıştır. Yörünge denklemleri dizilere çevrilmiş ve böylece GA çarpışmasız minimum hareket yolunu hesaplamak için kullanılmıştır.

Robot Eli:

Bu çalışmada insan eli benzeri beş parmaklı bir robot elinin bir nesneyi kavraması için gereken hareketler incelenmiş ve bu karmaşık problemin çözümünde GA kullanılmıştır. Bu problem çarpışmasız hareket yörüngesi saptamaya benzemektedir. Her bir parmak diğeri için çarpmaması gereken bir engeldir. Uyumluluk her bir parmağın kontak noktasına (nesneye dokunduğu nokta) olan uzaklığı, stabilite, manipulasyon ve çarpışmasız hareket gözönünde bulundurularak hesaplanmaktadır. Klasik GA kullanılmasına rağmen çalışma uzayı oldukça geniş olduğundan (2230) GA' nın biraz daha seçici davranması sağlanmıştır. Ayrıca GA operatörleri de bu sebepten dolayı biraz modifiye edilerek kullanılmıştır.

Genetik Algoritmanın Avantajları

Genetik algoritmalar, diğer eniyileme yöntemleri kullanılırken büyük zorluklarla karşılaşılan, oldukça büyük arama uzayına sahip problemlerin çözümünde başarı göstermektedir. 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. Bu amaç da genetik algoritmanın diğer algoritmalara göre sahip olduğu en iyi avantajlardan biridir. Diğer avantajları da şu şekilde sıralanabilir:

– Çok amaçlı optimizasyon yöntemleri ile kullanılabilmesi

– Çok karmaşık ortamlara uyarlanması

– Kısa sürelerde iyi sonuçlar verebilmesi

Genetik Algoritmanın Dezavantajları

Genetik algoritma aşamalarının çözülecek probleme özel olarak tasarlanması problemin modellenmesi açısından zor bir durum oluşturmaktadır.

Son kullanıcının modeli anlaması güç bir durumdur.

Çözülmesi gereken problem için hangi çaprazlama yönteminin kullanılacağı veya mutasyon oranının ne olacağının belirlenmesi zor bir durumdur.

Algoritmanın Sınırları

Genetik algoritmada kullanılan genetik operatörlerin optimizasyonunun tam yapılmamasından dolayı genetik algoritma lokal optimum noktaya yakınsar ve global optimum noktayı bulamaz veya rastgele arama gerçekleştirerek verimli çalışamaz. Bu durum algoritmanın sınırlandırılmasına yol açar. Örneğin mutasyon oranının çok yüksek olması algoritmanın rastgele arama yapmasına neden olur. Bu durum optimum sonuç bulmayı imkansıza yakın hale getirir. Mutasyon oranının çok düşük olması ise genetik çeşitlilik yaratmayacağından lokal optimum noktaya yakın bir alanda sonuç değerlerini sınırlandırır.

Genetik Algoritmaya Rakip Algoritmalar

Arama algoritmaları veya arı kolonisi algoritması, karınca kolonisi algoritması genetik algoritmalara rakip olarak verilebilir algoritmalardır. Karşılaştırma yapılacak algoritmalar farklı yaklaşımlara sahip olduğundan 2 algoritma için karşılaştırma yapılacaktır. Öncelikle genetik algoritma ve lineer arama* algoritmasını kıyaslayalım.

N=100 için;(N ziyaret edilecek nokta sayısı)

100!=93.326.215.443.944.152.681.699.238.856.266.700.490.715.968.264.381.621.468.592.963.895.217.599.993.229.915.608.941.463.976.156.518.286.253.697.920.827.223.758.251.185.210.916.864.000.000.000.000.000.000.000.000

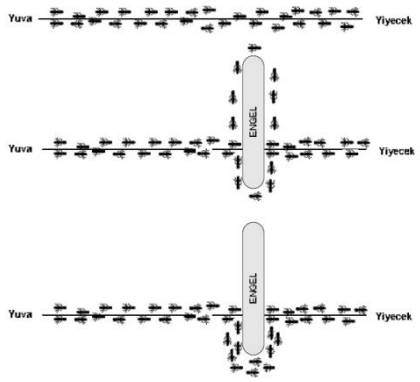
Uzunluğunda bir diziye ihtiyaç vardır. Bu dizi içerisinde 100! Olasılığın her birinin mesafe değeri hesaplanmalı ve diziye yazılmalıdır. Sonrasında lineer arama kullanılarak dizideki minimum değer bulunmalı ve bu değere karşılık gelen rota çözüm olarak kabul edilmelidir. Lineer arama algoritmasının karmaşıklığı tüm durumlar için O(n)'dir çünkü dizinin minimum elemanının bulunması için dizi elemanlarının tamamı gezilmek zorundandır. Wikipedia kaynağında durumla ilgili "*sadece yüz şehirlik liste olmasına rağmen çözüm yapılırken çözümün yıllar alabileceği iddia edilmektedir."* İfadesi yer almaktadır.

Aynı N değeri için tam optimize yazılan bir genetik algoritma kodu lineer aramaya göre bellek ve zaman açısından oldukça avantajlı durumdadır. *Journal Of Emerging Economies And Policy Vol.2 (1) | July 2017 GEZGİN SATICI PROBLEMİNİN GENETİK ALGORİTMALARLA ÇÖZÜMÜNDE BAŞLANGIÇ POPÜLASYONUN BELİRLENMESİ* araştırmasında populasyon büyüklüğü(kullanılacak dizi boyutunun en ideal durumu) 2000 olarak belirlenmiştir. Bu da 100! Elemana sahip bir dizi için bellekten yer ayırmak yerine 2000 elemana sahip bir dizi için bellekten yer ayrılacağı anlamına gelir. Zaman açısından aynı araştırmada verilen veriye göre en optimum sonuç 92,96 saniyede bulunmuştur. Lineer arama için ise üstte belirtildiği gibi N sayısına bağlı olarak yıllar alabilecek bir zamandan bahsedilmektedir.

*Çözüm kümesi elemanlarının uzunluk değeri sıralı olmayacağından dolayı lineer arama algoritmasıyla kıyaslaması yapılmıştır.

Karınca Kolonisi Algoritması

Genel olarak karıncaların gerçek hayattaki yiyecek arama yöntemini modellemiş halidir. Algoritma işleyişi şu şekilde incelenebilir: Karıncalar yuvalarından çıktıktan sonra rastgele yuva etrafında dağılım göstererek yiyecek aramaktadırlar. Yiyecek arayan karıncalar gittikleri yol üzerinde feromon sıvısını bırakarak birnevi arkasından gelecek olan karıncalar için iz oluşturmaktadır. Rastgele arama gerçekleyen karıncalar yiyecek bulup yuvaya döndüklerinde, yuvadan çıkan karıncalar yuvaya dönen karıncaların bıraktıkları sıvıları takip ederek yiyeceğe ulaşırlar. Burada önemli olan faktör feromon sıvısının buharlaşmasıdır. Uzun bir yiyecek yolunda feromon sıvısı buharlaşması yüksek oranda olacağından yuvadan yeni çıkan karınca feromon sıvısının yoğun olduğu yolu tercih edecek ve yiyeceğe en kısa yoldan gidecektir. En kısa yolun birçok karınca tarafından kullanılması üzerine diğer yollar işlevini kaybedecek ve en kısa yol feromon sıvısının en yoğun olduğu yol olarak karıncaların kullanımında olacaktır. Aşağıdaki görsel karıncaların en kısa yolu nasıl seçtiklerini temsil etmektedir. İlk başta yol üzerinde engel konulduğunda karıncalar engeli farklı yönlerde aşmaya çalışarak feromon sıvısının yoğun olduğu ana yola gelmeye çalışmış daha sonrasında yeni oluşturdukları ara yollardan uzun olanının kısa olana göre feromon yoğunluğu düşük olmasından dolayı uzun yol karıncalar tarafından tercih edilmeyip kısa yol karıncalar tarafından kullanılmıştır.



Karınca kolonisi algoritması ve genetik algoritma rapordaki problem için karşılaştırıldığında:

Başlangıç Değerleri:

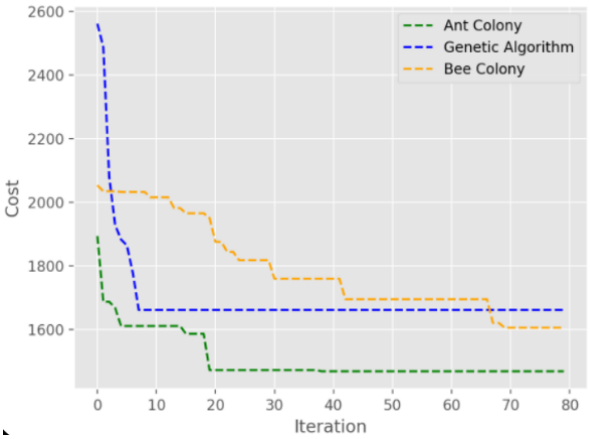
Genetik Algoritma

- Birey Sayısı: 15
- Mutasyon Olasılığı: 0,1
- Çaprazlama Olasılığı: 0,9
- Nokta Sayısı: 13
- Nesil Sayısı: 80

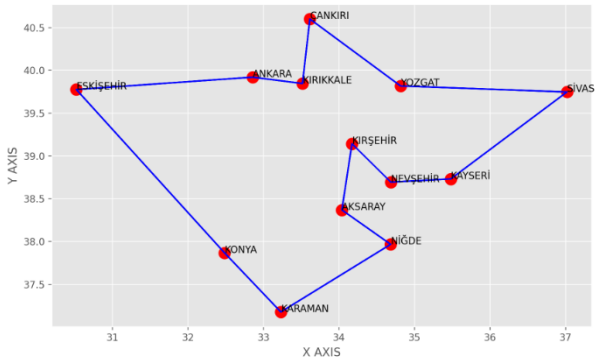
Karınca Kolonisi Algoritması

- Karınca Sayısı: 15
- Feromon Buharlaşma Oranı: 0,3
- Alfa: 1
- Beta: 2,5
- Nokta Sayısı:13
- İterasyon sayısı:80

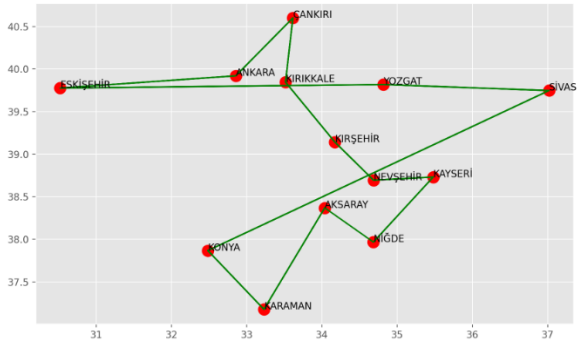
Algoritmaların iterasyon sayısı – bulduğu yol mesafesi grafiği aşağıdaki gibidir.



Genetik Algoritma En İyi Rota Sonucu



Karınca Kolonisi Algoritması En İyi Rota Sonucu



Grafiklerde görüldüğü gibi Karınca Kolonisi Algoritması Genetik Algoritmaya göre daha iyi sonuç bulmaktadır.

*Karşılaştırma yapılırken TOGU CS - OPTIMIZATION & ML aracı kullanılmıştır. Kullanılan araç hakkında detaylı bilgiye kaynakçada verilen bağlantıdan erişebilirsiniz.

Algoritma Karmaşıklığı

n adet nokta, k popülasyon büyüklüğü, p adet ata kromozom, L nesil (iterasyon) sayısı olmak üzere;

- Random Başlangıç Populasyonu Oluşturma Fonksiyonu:

  Girilen Nokta Dizisini popülasyon büyüklüğü kadar kopyalama işlemi: k*n

  Her kopyalanan dizi için n defa random yer değiştirme işlemi: k* n^2

- En İyi Ata Kromozom Seçim Fonksiyonu:

  p adet en kısa mesafeye sahip kromozomu işaretleme: p*k

  İşaretlenen kromozomları matriste yerleştirme işlemi: k*n

  Kullanılmayacak kromozomlardan matrisi temizleme: (6000)*n

- Çaprazlama Fonksiyonu: $(p/2)*(n^2/4)$
- Mutasyon Fonksiyonu: (p/2)
- Arama Fonksiyonu: (3*p)/2

Genetik Algoritma Karmaşıklık Analizi: $k*n+k*n^2+L*[(p*k)+(k*n)+(6000*n)+(p/2)*(n^2/4)+(p/2)]+(3*p)/2$

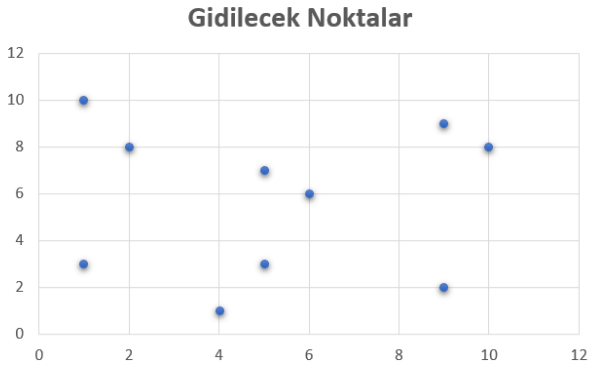
O notasyonu: $O(k*n+k*n^2+L*[(p*k)+(k*n)+(n)+(p)*(n^2)+(p)]+(p))$

*Tek değişkene bağlı olarak ifade edilmesi n , p , k, L değerlerine bağlı olarak değişmektedir. Bu nedenle genel ifade verilmiştir.

*Başlangıç populasyonunda minimum değeri arama asıl algoritmada yer almadığından karmaşıklık analizine dahil edilmemiştir.

Programın Çalışması: (Bu çalışma, sürecin anlaşılması için tüm aşamlar ekrana bastırılacak şekilde düzenlenerek hazırlanmıştır.Yazılımın normal çalışma durumu bir sonraki aşamada bulunmaktadır.)

```
Kac nokta icin rota hesaplanacak: 10
Baslangic populasyon buyuklugu: 8
Baslang²c populasyonundan kac ata birey kullanilacak: 6
Problem icin kac nesil ilerlenecek: 10
1. nokta icin (x,y):1,3
2. nokta icin (x,y):5,7
3. nokta icin (x,y):9,2
4. nokta icin (x,y):4,1
5. nokta icin (x,y):2,8
6. nokta icin (x,y):6,6
7. nokta icin (x,y):5,3
8. nokta icin (x,y):1,10
9. nokta icin (x,y):9,9
10. nokta icin (x,y):10,8
```



Başlangıç Populasyon Büyüklüğü: Random oluşturulacak popülasyondaki kromozom(rota) sayısı

Başlangıç Populasyonundan Kaç Ata Birey Kullanılacak: Başlangıç popülasyonunda en iyi mesafeye sahip kaç kromozomun ebeveyn olarak kullanılacağını temsil eder.

Problem İçin Kaç Nesil İlerlenecek: Tüm genetik süreçlerin kaç kez tekrarlanacağını ifade eder.(Algoritmada belirtilen maksimum iterasyon sayısı)

```
Random Ata Kromozomlar Olusturuldu:

1 2 3 4 5 6 7 8 9 10 57 0
6 10 5 1 2 5 1 9 4 9 0 0
6 8 7 3 8 3 10 9 1 2 0 0
1 2 3 4 5 6 7 8 9 10 61 0
2 5 9 4 1 6 1 9 5 10 0 0
8 3 9 1 3 6 10 2 7 8 0 0
1 2 3 4 5 6 7 8 9 10 52 0
9 4 2 5 5 6 9 1 1 10 0 0
2 1 8 7 3 6 9 10 3 8 0 0
1 2 3 4 5 6 7 8 9 10 46 0
1 6 5 2 5 9 9 10 4 1 0 0
10 6 3 8 7 2 9 8 1 3 0 0
1 2 3 4 5 6 7 8 9 10 49 0
4 5 6 10 2 1 9 9 1 5 0 0
1 7 6 8 8 10 9 2 3 3 0 0
1 2 3 4 5 6 7 8 9 10 52 0
2 9 10 5 6 1 1 9 5 4 0 0
8 2 8 7 6 3 10 9 3 1 0 0
1 2 3 4 5 6 7 8 9 10 50 0
1 5 5 1 10 6 9 9 4 2 0 0
3 3 7 10 8 6 9 2 1 8 0 0
1 2 3 4 5 6 7 8 9 10 49 0
5 4 9 10 6 1 5 9 1 2 0 0
3 1 9 8 6 10 7 2 3 8 0 0
```

Kromozomun Yapısı:

1 2 3 4 5 6 7 8 9 10 (57) → Rotanın Toplam Uzunluğu

6 10 5 1 2 5 1 9 4 9 x koordinatı

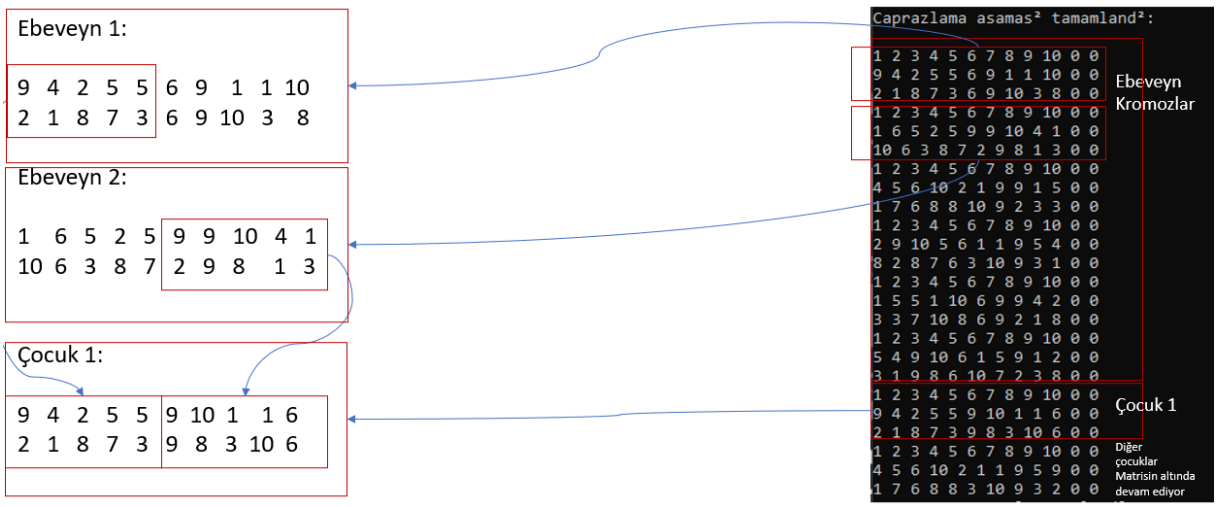
6 8 7 3 8 3 10 9 1 2 y koordinatı

Rota: (6,6) -> (10,8) -> (5,7) -> (1,3) -> (2,8) -> (5,3) -> (1,10) -> (9,9) -> (4,1) -> (9,2)

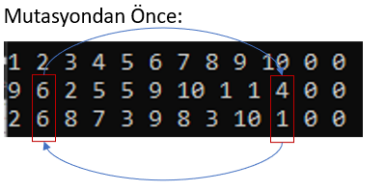
```
Secilen Kromozomlar 1 ile isaretlendi:

1 2 3 4 5 6 7 8 9 10 57 0
6 10 5 1 2 5 1 9 4 9 0 0
6 8 7 3 8 3 10 9 1 2 0 0
1 2 3 4 5 6 7 8 9 10 61 0
2 5 9 4 1 6 1 9 5 10 0 0
8 3 9 1 3 6 10 2 7 8 0 0
1 2 3 4 5 6 7 8 9 10 0 1
9 4 2 5 5 6 9 1 1 10 0 0
2 1 8 7 3 6 9 10 3 8 0 0
1 2 3 4 5 6 7 8 9 10 0 1
1 6 5 2 5 9 9 10 4 1 0 0
10 6 3 8 7 2 9 8 1 3 0 0
1 2 3 4 5 6 7 8 9 10 0 1
4 5 6 10 2 1 9 9 1 5 0 0
1 7 6 8 8 10 9 2 3 3 0 0
1 2 3 4 5 6 7 8 9 10 0 1
2 9 10 5 6 1 1 9 5 4 0 0
8 2 8 7 6 3 10 9 3 1 0 0
1 2 3 4 5 6 7 8 9 10 0 1
1 5 5 1 10 6 9 9 4 2 0 0
3 3 7 10 8 6 9 2 1 8 0 0
1 2 3 4 5 6 7 8 9 10 0 1
5 4 9 10 6 1 5 9 1 2 0 0
3 1 9 8 6 10 7 2 3 8 0 0
```

Başlangıç Populasyonundan en kısa mesafeye sahip seçim adeti kadar kromozom 1 ile işaretleniyor

```
1 ile isaretlenen kromozomlar secilerek diger kromozomlar yok edildi:

1 2 3 4 5 6 7 8 9 10 0 0
9 4 2 5 5 6 9 1 1 10 0 0
2 1 8 7 3 6 9 10 3 8 0 0
1 2 3 4 5 6 7 8 9 10 0 0
1 6 5 2 5 9 9 10 4 1 0 0
10 6 3 8 7 2 9 8 1 3 0 0
1 2 3 4 5 6 7 8 9 10 0 0
4 5 6 10 2 1 9 9 1 5 0 0
1 7 6 8 8 10 9 2 3 3 0 0
1 2 3 4 5 6 7 8 9 10 0 0
2 9 10 5 6 1 1 9 5 4 0 0
8 2 8 7 6 3 10 9 3 1 0 0
1 2 3 4 5 6 7 8 9 10 0 0
1 5 5 1 10 6 9 9 4 2 0 0
3 3 7 10 8 6 9 2 1 8 0 0
1 2 3 4 5 6 7 8 9 10 0 0
5 4 9 10 6 1 5 9 1 2 0 0
3 1 9 8 6 10 7 2 3 8 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

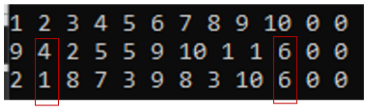


Mutasyondan Önce:



Çocuk kromozda rastgele 2 nokta seçilip bu noktaların yer değiştirmesi.

Mutasyondan Sonra:

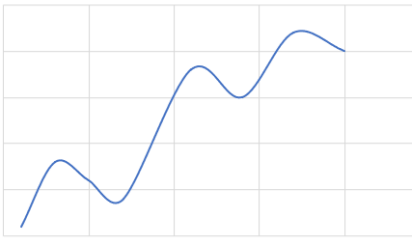


Bu Aşamadan Sonra Çocuk Bireylerin Üretimi Tamamlanmıştır. Çocuk Bireyler Populasyon Havuzuna Katılır Ve En Kısa Mesafeye Sahip Kromozomlar Seçilerek Süreç Tekrarlanır.

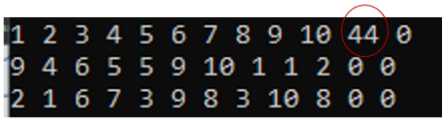
Neden Gerekli:

Mutasyon yapılmasının nedeni; birbirini izleyen daha uygun bireylerin atılmasından gelmektedir. Kromozomlarda rasgele değişiklikler yapılarak arama uzayında yeni kısımlara ulaşılmasını sağlar.

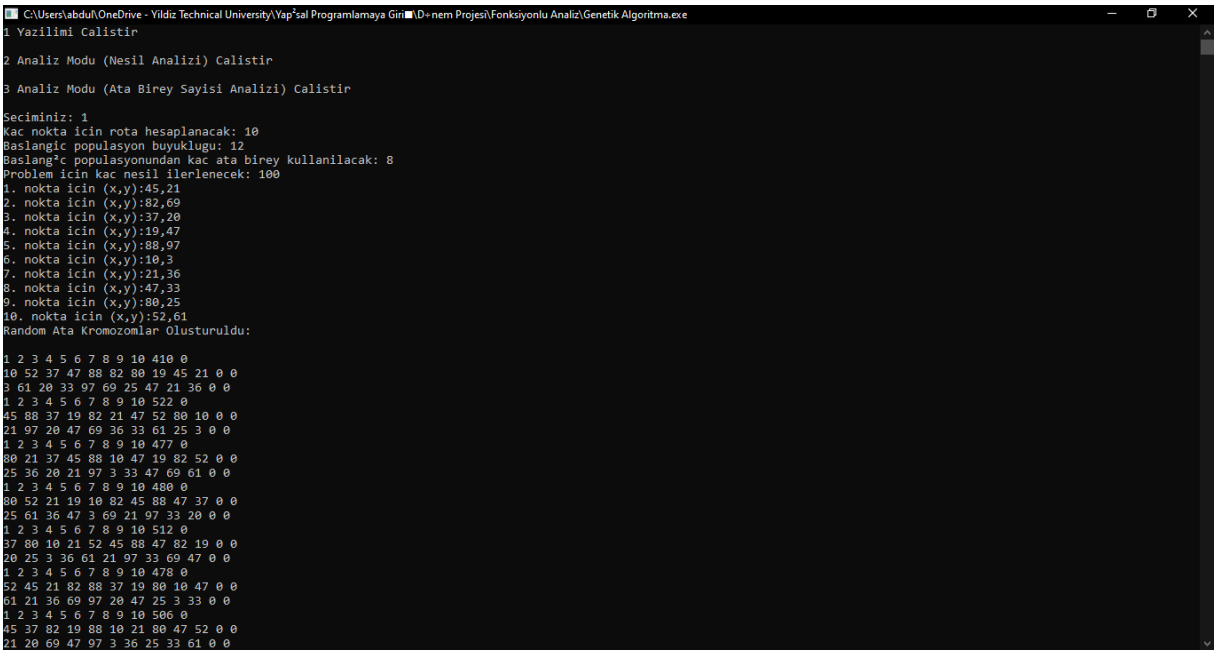
Yerel optimum noktalarının geçilmesini sağlar.

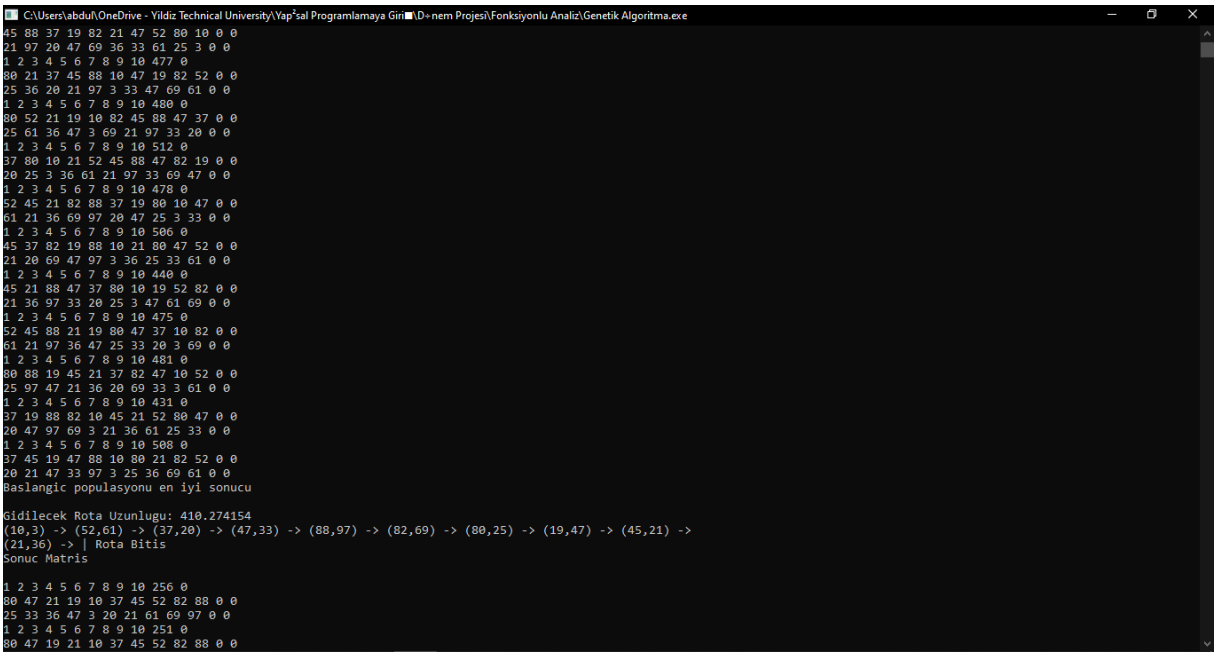


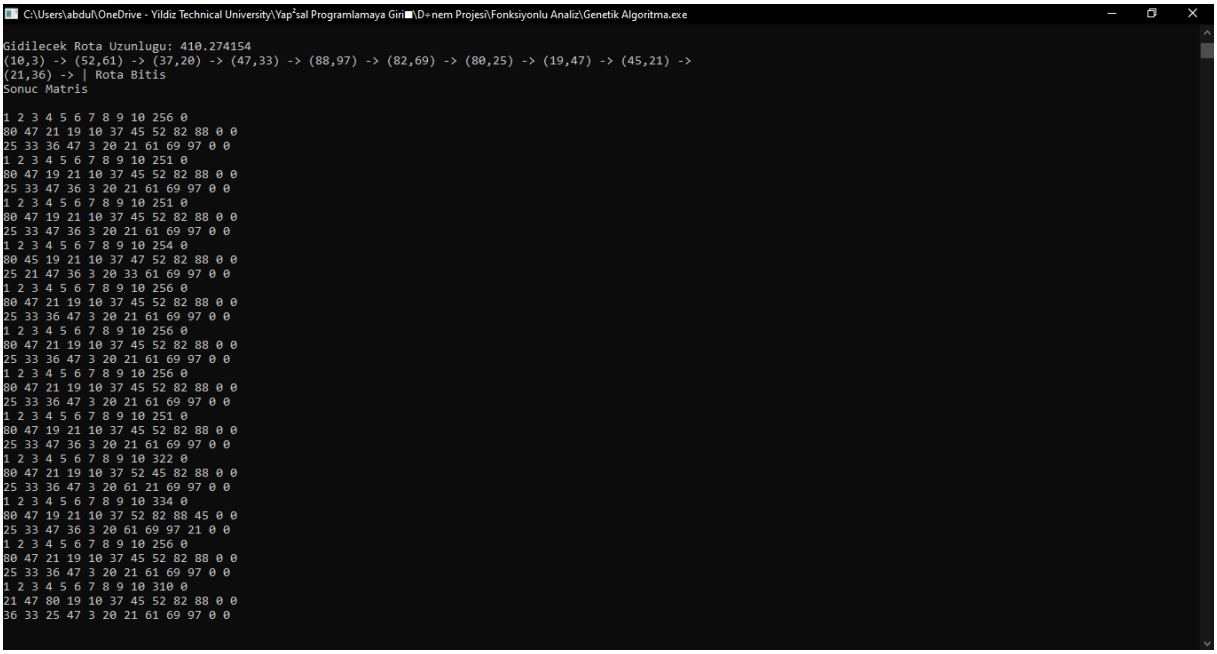
Bu örnek veriler için algoritma 10 nesil (iterasyon) devam etmiş ve sonuç popülasyon havuzundaki en kısa mesafeye sahip kromozom aşağıdaki gibi bulunmuştur.

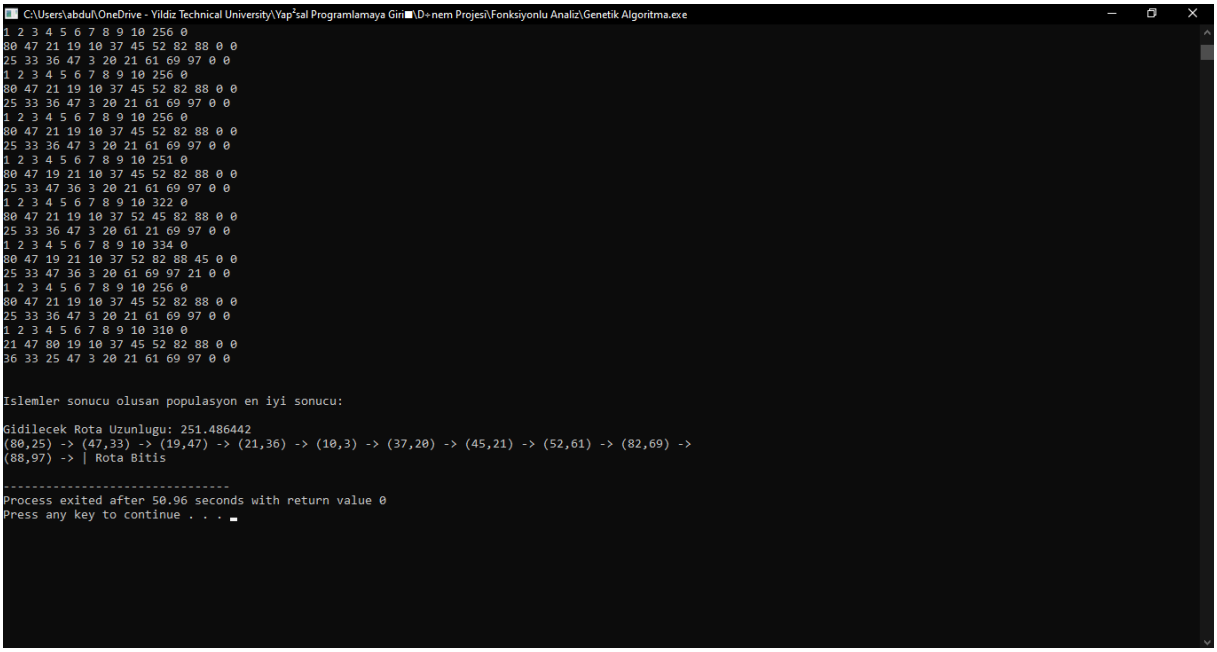


Programın Çalışması:



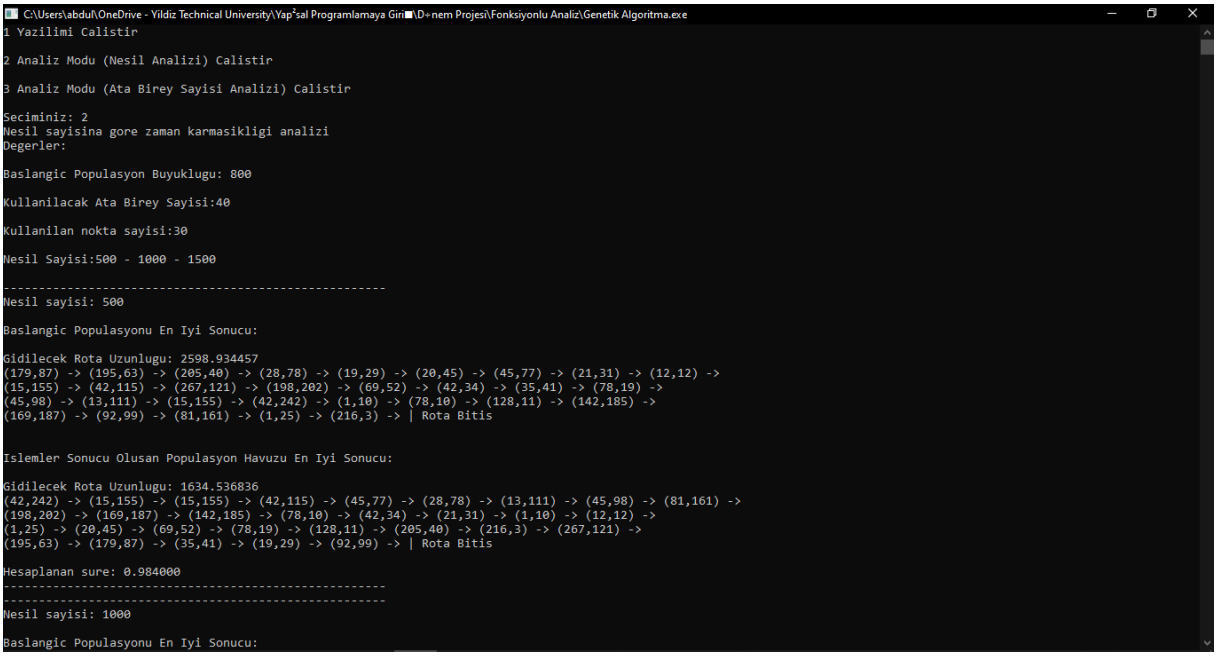


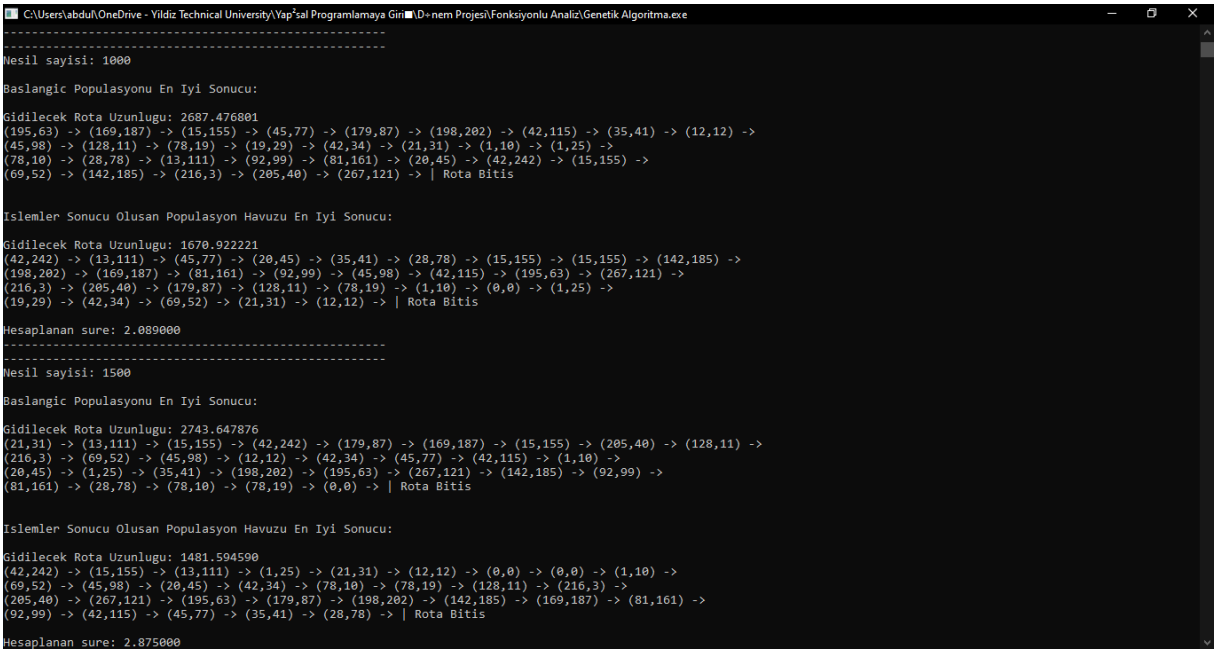




*Random ata kromozomlar oluşturuldu yazısının altındaki matris başlangıç populasyonunu ifade etmektedir. Sonuç matrisi yazısının altındaki matris iterasyonlar sonucu oluşan matrisi ifade etmektedir. Gidilecek rota, hemen üstündeki matrisin minimum değerini yazdırmıştır. Populasyon ve sonuç matrisleri programın çalışmasını göstermek amacıyla ekrana yazdırılmıştır. Ara matrisler iterasyon sayısı boyunca yazdırılacağından çok fazla ekran görüntüsü oluşturmaktadır. Bu nedenle ara işlemlerin açık hali yazdırılmamış, kod içerisine yorum satırı olarak ara değerleri ekrana yazdıran kod satırları eklenmiştir. Test amacıyla koddaki ilgili satırları aktif ederek tüm süreci ekrana yazdırabilirsiniz.

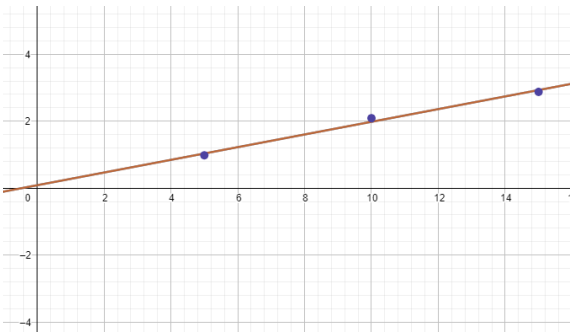
Analiz – 1 (Nesil Sayısına Göre Zaman Karmaşıklığı Analizi)





```
C:\Users\abdul\OneDrive - Yildiz Technical University\Yap²sal Programlamaya Giri■\D÷nem Projesi\Fonksiyonlu Analiz\Genetik Algoritma.exe

Gidilecek Rota Uzunlugu: 2743.647876
(21,31) -> (13,111) -> (15,155) -> (42,242) -> (179,87) -> (169,187) -> (15,155) -> (205,40) -> (128,11) ->
(216,3) -> (69,52) -> (45,98) -> (12,12) -> (42,34) -> (45,77) -> (42,115) -> (1,10) ->
(20,45) -> (1,25) -> (35,41) -> (198,202) -> (195,63) -> (267,121) -> (142,185) -> (92,99) ->
(81,161) -> (28,78) -> (78,10) -> (78,19) -> (0,0) -> | Rota Bitis


Islemler Sonucu Olusan Populasyon Havuzu En Iyi Sonucu:

Gidilecek Rota Uzunlugu: 1481.594590
(42,242) -> (15,155) -> (13,111) -> (1,25) -> (21,31) -> (12,12) -> (0,0) -> (0,0) -> (1,10) ->
(69,52) -> (45,98) -> (20,45) -> (42,34) -> (78,10) -> (78,19) -> (128,11) -> (216,3) ->
(205,40) -> (267,121) -> (195,63) -> (179,87) -> (198,202) -> (142,185) -> (169,187) -> (81,161) ->
(92,99) -> (42,115) -> (45,77) -> (35,41) -> (28,78) -> | Rota Bitis

Hesaplanan sure: 2.875000
--------------------------------------------------------
                    ANALIZ

--------------------------------------------------------


500 nesil
*********0.984(sn)


1000 nesil
*********************2.089(sn)


1500 nesil
*****************************2.875(sn)



--------------------------------
Process exited after 7.754 seconds with return value 0
Press any key to continue . . .
```

Hesaplanan sonuçlar için çizilen eğri doğrusal fonksiyon davranışı gösterir.



Eğri denklemi:

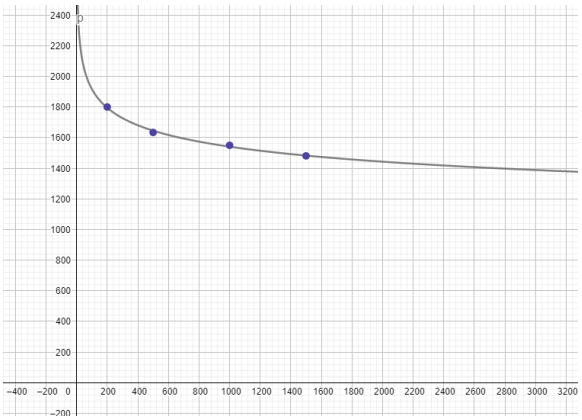
f(x) =a.x+b

Test verileri için a ve b değerleri:

F(x)=0.1895x+0.0863

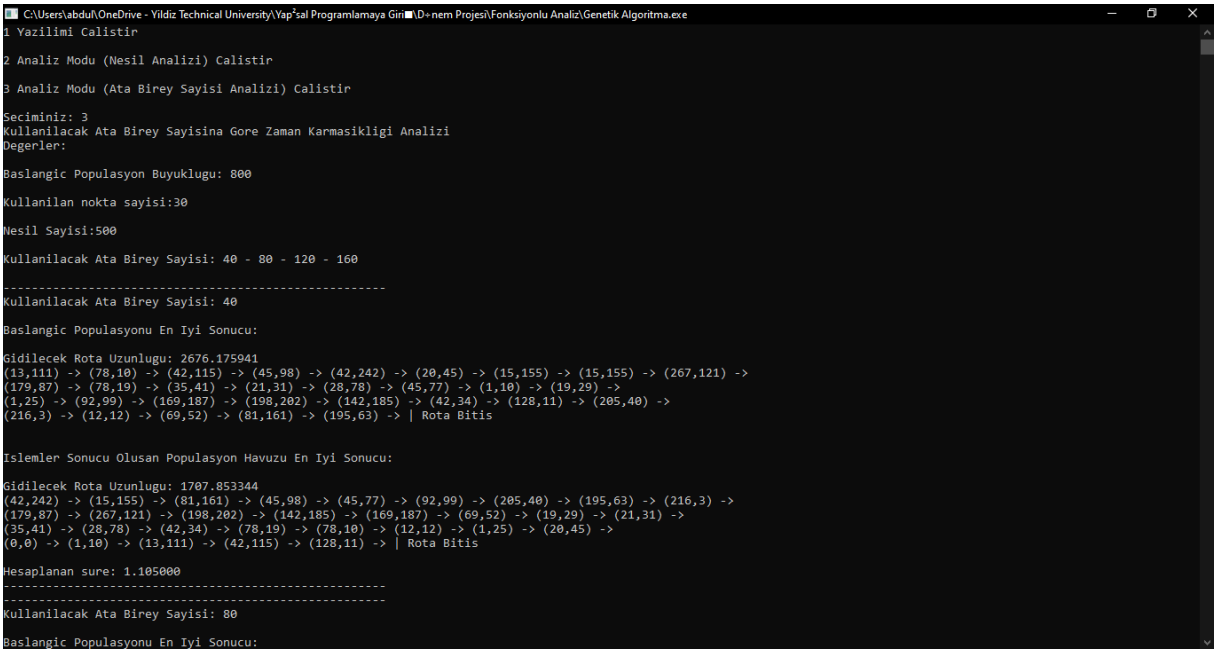
1500 - 1000 - 1500 değerleri grafik davranışının küçük boyutta

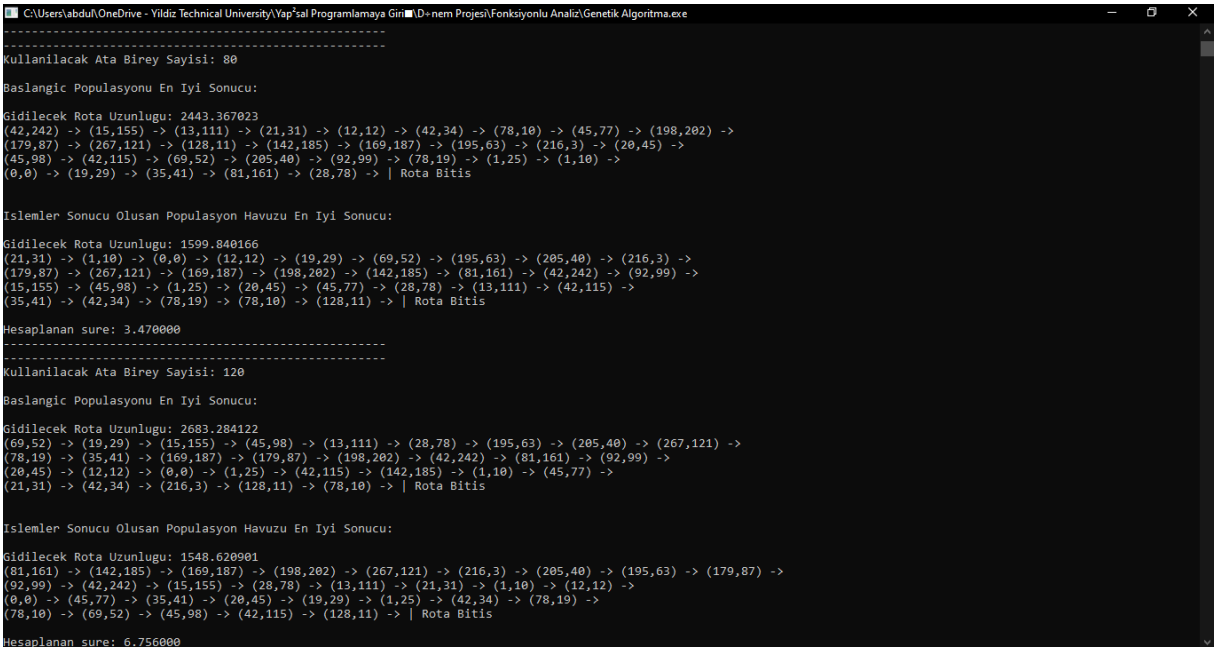
görünebilmesi için 5 - 10 - 15 olarak değiştirilmiştir.



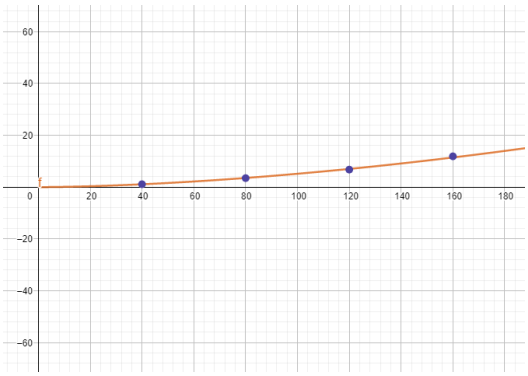
Nesil sayısı – Bulunan en kısa yol arasında ortalama test değerlerine bağlı olarak oluşturulan grafiğin davranışı yandaki gibidir.

Analiz – 2 (Ata Birey Sayısına Göre Zaman Karmaşıklığı Analizi)





```
C:\Users\abdul\OneDrive - Yildiz Technical University\Yap²sal Programlamaya Giri■\D÷nem Projesi\Fonksiyonlu Analiz\Genetik Algoritma.exe
-----------------------------------------------------
Kullanilacak Ata Birey Sayisi: 160

Baslangic Populasyonu En Iyi Sonucu:

Gidilecek Rota Uzunlugu: 2659.340540
(81,161) -> (142,185) -> (169,187) -> (42,242) -> (128,11) -> (216,3) -> (205,40) -> (42,34) -> (78,10) ->
(35,41) -> (69,52) -> (45,77) -> (45,98) -> (13,111) -> (21,31) -> (19,29) -> (1,10) ->
(12,12) -> (28,78) -> (15,155) -> (78,19) -> (1,25) -> (0,0) -> (267,121) -> (92,99) ->
(198,202) -> (42,115) -> (195,63) -> (179,87) -> (20,45) -> | Rota Bitis


Islemler Sonucu Olusan Populasyon Havuzu En Iyi Sonucu:

Gidilecek Rota Uzunlugu: 1541.945308
(42,242) -> (142,185) -> (198,202) -> (169,187) -> (267,121) -> (179,87) -> (195,63) -> (205,40) -> (216,3) ->
(92,99) -> (69,52) -> (19,29) -> (20,45) -> (42,34) -> (21,31) -> (45,98) -> (42,115) ->
(81,161) -> (15,155) -> (13,111) -> (28,78) -> (45,77) -> (78,10) -> (78,19) -> (128,11) ->
(12,12) -> (1,25) -> (0,0) -> (35,41) -> (1,10) -> | Rota Bitis

Hesaplanan sure: 11.881000
-----------------------------------------------------
                    ANALIZ

-----------------------------------------------------


40 Ata Birey
***********1.105(sn)


80 Ata Birey
*********************************3.470(sn)


120 Ata Birey
*************************************************************6.756(sn)


160 Ata Birey
*****************************************************************************************************************************************11.881(sn)
```



Hesaplanan sonuçlar için çizilen eğri üstel fonksiyon davranışı gösterir.

Eğri denklemi:

f(x) =a.x+x^(b)

Test verileri için a ve b değerleri

F(x)=0.00207x^(1.69)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define MAX 6000

void mesafeHesapla(double **populasyon,int aramaUzayi,int n); //Rotanın toplam uzunlugunu hesaplar ve matriste rotanın en sonuna yerlestirir.
void swap(double *x,double *y);
void randomAtaKromozomOlustur(double **populasyon,int size,int n);
void enIyiSecimi(double **populasyon,int secim,int aramaUzayi,int size,int n);
void caprazlaFunc(double **populasyon,int secim,int n);
void mutasyonFunc(double **populasyon,int secim,int n);
void matrisYazdir(double **populasyon,int size, int n);
void aramaFunc(double **populasyon,int aramaUzayi,int n);
void nesilAnalizFunc(double **populasyon);
void ataBireyAnalizFunc(double **populasyon);

int main(){
        srand(time(NULL));
        double **populasyon;
        int i,n,size,secim,iterasyon=0,nesil,mod;
        //509x509 matristen buyuk matris olusturmak icin asagıdaki kod blokunu kullanıyorum.
  populasyon=(double**) calloc(6000,sizeof(double*));
  for(i=0;i<6000;i++){
     populasyon[i]=(double*) calloc(6000,sizeof(double));
  }
  printf("1 Yazilimi Calistir\n\n2 Analiz Modu (Nesil Analizi) Calistir\n\n3 Analiz Modu (Ata Birey Sayisi Analizi) Calistir\n\nSeciminiz: ");
  scanf("%d",&mod);
  if(mod==1){
                printf("Kac nokta icin rota hesaplanacak: ");
                scanf("%d",&n);
                printf("Baslangic populasyon buyuklugu: ");
                scanf("%d",&size);
                printf("Baslangıc populasyonundan kac ata birey kullanilacak: ");
```

```
		scanf("%d",&secim);
		printf("Problem icin kac nesil ilerlenecek: ");
		scanf("%d",&nesil);
		for(i=0;i<n;i++){		//Butun noktalarin koordinat bilgilerini aliyorum.
			populasyon[0][i]=i+1;
			printf("%d. nokta icin (x,y):",i+1);
			scanf("%lf,%lf",&populasyon[1][i],&populasyon[2][i]);
		}
		randomAtaKromozomOlustur(populasyon,size,n);
		//printf("Random Ata Kromozomlar Olusturuldu: \n\n");
		//matrisYazdir(populasyon,size,n);
		//Buradan itibaren populasyon matrisinde populasyon buyuklugu(size) kadar rota ve bunların uzunluk degeri bulunuyor.
		enIyiSecimi(populasyon,secim,size*3,size,n);
		//printf("En iyileri sectim matris:\n");
		//matrisYazdir(populasyon,size,n);
		//Nesil sayisi kadar dongüye girip islemleri baslatıyorum.
		printf("Baslangic populasyonu en iyi sonucu\n");
		mesafeHesapla(populasyon,secim*3,n);
		aramaFunc(populasyon,secim*3,n);
		while(iterasyon<nesil){
			if(iterasyon>0){		//ilk iterasyondan sonra cocuk ve ata bireylerin bulundugu matris icin mesafe hesapliyor.
				mesafeHesapla(populasyon,(secim*9)/2,n);
				//printf("Mutasyon asaması tamamlandi: \n\n");
				//matrisYazdir(populasyon,size,n);
				//printf("En iyileri secimi gerceklesti\n");
				// SEÇİM AŞAMASI BASLANGICI (En kısa mesafeye sahip secim adetinde rota(kromozom) secilecek.
				enIyiSecimi(populasyon,secim,(9*secim)/2,size,n);
				//SECİM ASAMASI BİTİSİ
				//matrisYazdir(populasyon,size,n);
			}
			//CAPRAZLAMA ASAMASI BASLANGICI
			caprazlaFunc(populasyon,secim,n);
			//printf("Caprazlama asamasi tamamlandi: \n\n");
```

```
                //matrisYazdir(populasyon,size,n);
                //CAPRAZLAMA ASAMASI BİTİSİ
                //MUTASYON ASAMASI BASLANGIC
                mutasyonFunc(populasyon,secim,n); //cocuk bireyleri mutasyona ugratiyor

                //printf("Mutasyon asaması tamamlandı: \n\n");
                //matrisYazdir(populasyon,size,n);
                //MUTASYON ASAMASI BİTİSİ
                iterasyon++;
        }
        mesafeHesapla(populasyon,(secim*9)/2,n);
        //printf("Sonuc Matris \n\n");
        //matrisYazdir(populasyon,secim+(secim/2),n);
        //printf("\n\n");
        printf("Islemler sonucu olusan populasyon en iyi sonucu:\n");
        aramaFunc(populasyon,(9*secim)/2,n);
    }
    else if(mod==2){
        nesilAnalizFunc(populasyon);
    }
    else if(mod==3){
        ataBireyAnalizFunc(populasyon);
    }
    else{
        printf("Hatali deger girdiniz.");
    }
    return 0;
}


void matrisYazdir(double **populasyon,int size, int n){
    int i,j;
    for(i=0;i<size*3;i++){
        for(j=0;j<=n+1;j++){
            printf("%.0lf ",populasyon[i][j]);
```

```
		}
		printf("\n");
	}
}

void mesafeHesapla(double **populasyon,int aramaUzayi,int n){
	int i,k;
	double tempx,tempy,temp,mesafe;
	for(k=0;k<aramaUzayi;k+=3){
		tempx=0,tempy=0,temp=0,mesafe=0;
		for(i=1;i<n;i++){
			tempx=populasyon[k+1][i]-populasyon[k+1][i-1];  //2 nokta arasi uzaklık hesaplama bloku
			tempx=tempx*tempx;
			tempy=populasyon[k+2][i]-populasyon[k+2][i-1];
			tempy=tempy*tempy;
			temp=tempx+tempy;
			temp=sqrt(temp);
			mesafe+=temp;
		}
		populasyon[k][n]=mesafe;
	}
}

void enIyiSecimi(double **populasyon,int secim,int aramaUzayi,int size,int n){
	int j,i,k,min,indis;
	for(j=0;j<secim;j++){  //Secim degiskeni adetinde en kısa mesafeye sahip rotayı secip isaretliyorum
		min=99999999;
		for(i=0;i<aramaUzayi;i+=3){
			if(min>populasyon[i][n]&&populasyon[i][n]!=0){
				min=populasyon[i][n];
				indis=i;
			}
		}
		populasyon[indis][n]=0;   //aynı min degerini bulmasın diye ilgili mesafeyi sıfırlıyorum.
		populasyon[indis][n+1]=1;    // oldurulmeyecek olan kromozomları 1 ile isaretliyorum.
```

```
	}
	//printf("Secilen Kromozomlar 1 ile isaretlendi: \n\n");
	//matrisYazdir(populasyon,size,n);
	k=0;
	for(i=0;i<aramaUzayi;i+=3){	//Bu blokta 1 ile isaretledigim rotaları bulup bunları sirasiyla en üstten baslayarak matrisin altina dogru diziyorum.
		if(populasyon[i][n+1]==1){ //islem sonucunda secmek istedigim sayida kromozom matrisin en ustune dizilmis oluyor.
			for(j=0;j<n+2;j++){
				//printf("tasinacak deger :%lf\n",populasyon[i+1][j]);
				//printf("tasinan yer   %lf\n",populasyon[k+1][j]);
				populasyon[k][j]=populasyon[i][j];
				populasyon[k+1][j]=populasyon[i+1][j];
				populasyon[k+2][j]=populasyon[i+2][j];
			}
			k+=3;
		}
	}
	for(i=secim*3;i<MAX;i++){	//Burada populasyondaki secilmis kromozlar haricindeki kromozomları yok ediyorum.
		populasyon[i-(secim*3)][n+1]=0; //isaretleme yaptıgım birleri de burada 0a ceviyorum.
		for(j=0;j<n+2;j++){
			populasyon[i][j]=0;
		}
	}
	//printf("1 ile isaretlenen kromozomlar secilerek diger kromozomlar yok edildi:\n\n");
	//matrisYazdir(populasyon,size,n);
}

void caprazlaFunc(double **populasyon,int secim,int n){
	int i,j,k,m,l,flag=0,temp;
	k=0;	//CrossOver asaması (tek noktalı çaprazlama) Kromozomun yarısı bir ata kromozomdan, diger yarısı digerinden.
	for(i=0;i<(secim*3)-3;i+=6){	//Bu blok cocuk kromozomun ilk yarısını alıyor.Yeni bireyler matriste aşağıya diziliyor.
		for(j=0;j<n/2;j++){
			populasyon[secim*3+k][j]=populasyon[i][j];
			populasyon[secim*3+k+1][j]=populasyon[i+1][j];
```

```
                populasyon[secim*3+k+2][j]=populasyon[i+2][j];
        }
        k+=3;
    }
    k=0;
    for(i=3;i<secim*3;i+=6){        //Bu blok cocuk kromozomun ikinci yarısını alıyor.
        temp=n/2; // temp degiskeni eger aday elemanım kromozomun ilk kısmında bulunuyorsa onu alamayacağımdan sıranın kaymasını engeller.Boylece alamadıgım elemanlar icin kromozomun sonunda yerim olur.
        for(j=n/2;j<n;j++){
            populasyon[secim*3+k][j]=populasyon[i][j];
            for(l=0;l<n/2;l++){

    if(populasyon[i+1][j]==populasyon[secim*3+k+1][l]&&populasyon[i+2][j]==populasyon[secim*3+k+2][l]){
//2.kromozomdan aldıgın noktayı ilk kromzomdan almadıysam cocuk kromozoma aktarıyorum

                    flag=1;
                }
            }
            if(flag!=1){
                populasyon[secim*3+k+1][temp]=populasyon[i+1][j];
                populasyon[secim*3+k+2][temp]=populasyon[i+2][j];
                temp++;
            }
            flag=0;
        }
        k+=3;
    }
    flag=0;
    //2. Kromozomun sonran yarısını aldım ve bu parcada olup cocuk kromozomun ilk yarısında olmayan noktaları cocuk kromozoma ekledim.
    //alt blokta 2. kromozomun basından baslayıp cocuk kromozomda olmayan noktaları cocuk kromozoma ekliyorum. Boylece caprazlama islemi bitiyor.
    for(i=secim*3;i<(secim*9)/2;i+=3){
        temp=n/2;
        for(j=n/2;j<n;j++){             // kromozomda 0,0 olan yer ilk nerde baslıyor onu buluyorum.
            if(populasyon[i+1][j]!=0||populasyon[i+2][j]!=0){
                temp++;
```

```
                }
            }
            for(k=3;k<secim*3;k+=6){
                for(m=0;m<n;m++){
                    for(l=0;l<temp;l++){

    if(populasyon[k+1][m]==populasyon[i+1][l]&&populasyon[k+2][m]==populasyon[i+2][l]){
                                flag=1;
                        }
                    }
                    if(flag!=1){
                        populasyon[i+1][temp]=populasyon[k+1][m];
                        populasyon[i+2][temp]=populasyon[k+2][m];
                        temp++;
                    }
                    flag=0;
                }
            }
        }
        //printf("\n\n");
        //matrisYazdir(populasyon,secim+(secim/2),n);
}

void mutasyonFunc(double **populasyon,int secim,int n){
        int i,j,random1,random2;
        for(i=secim*3;i<(9*secim)/2;i+=3){
            for(j=0;j<(n*1)/10;j++){  //Her kromozom için mutasyon oranı kadar çalışacak olan döngü
                random1= rand() % n;
                random2= rand() % n;
                swap(&populasyon[i+1][random1],&populasyon[i+1][random2]);
                swap(&populasyon[i+2][random1],&populasyon[i+2][random2]);
            }
        }
}
```

```
void randomAtaKromozomOlustur(double **populasyon,int size,int n){
	int i,j,random1,random2;
	for(i=3;i<size*3;i+=3){   //Burada populasyon büyüklügü kadar ata kromozomu(girilen noktaları) kopyalıyorum. Elimde populasyon kadar es kromozom olusacak
		for(j=0;j<n;j++){
			populasyon[i][j]=populasyon[0][j];
			populasyon[i+1][j]=populasyon[1][j];
			populasyon[i+2][j]=populasyon[2][j];
		}
	}
	for(i=0;i<size*3;i+=3){    //Random ata kromozom populasyonu olusturmak icin kopyaladıgım kromozomlardan rastgele 2 nokta seciyorum ve o noktaları
		for(j=0;j<n;j++){     //swap ediyorum(Her kromozomu n defa). Boylece elimde rastgele dizilime sahip kromozomlar olusuyor.
			random1=rand() % n+1;
			random2=rand() % n+1;
			swap(&populasyon[i+1][random1-1],&populasyon[i+1][random2-1]);
			swap(&populasyon[i+2][random1-1],&populasyon[i+2][random2-1]);
		}
	}
	mesafeHesapla(populasyon,size*3,n);
}

void swap(double *x,double *y){
	double temp;
	temp=*x;
	*x=*y;
	*y=temp;
}

void aramaFunc(double **populasyon,int aramaUzayi,int n){
	int i,indis,min=999999999;
	for(i=0;i<aramaUzayi;i+=3){
		if(populasyon[i][n]<min){
			min=populasyon[i][n];
			indis=i;
```

```
		}
	}
	printf("\nGidilecek Rota Uzunlugu: %lf\n",populasyon[indis][n]);
	for(i=0;i<n;i++){
		printf("(%0.lf,%0.lf) -> ",populasyon[indis+1][i],populasyon[indis+2][i]);
		if(i==8||i==16||i==24){
			printf("\n");
		}
	}
	printf("| Rota Bitis \n");
}

void nesilAnalizFunc(double **populasyon){
	int size=800,n=30,secim=40,j,i,temp=0,nesil=0,mod=0,iterasyon;
	printf("Nesil sayisina gore zaman karmasikligi analizi\n");
	printf("Degerler:\n\nBaslangic Populasyon Buyuklugu: 800\n\nKullanilacak Ata Birey Sayisi:40\n\nKullanilan nokta sayisi:30\n\nNesil Sayisi:500 - 1000 - 1500\n\n");
	double sure[3]={0};
{	populasyon[1][0]=1;
	populasyon[2][0]=25;
	populasyon[1][1]=42;
	populasyon[2][1]=115;
	populasyon[1][2]=19;
	populasyon[2][2]=29;
	populasyon[1][3]=35;
	populasyon[2][3]=41;
	populasyon[1][4]=78;
	populasyon[2][4]=10;
	populasyon[1][5]=20;
	populasyon[2][5]=45;
	populasyon[1][6]=92;
	populasyon[2][6]=99;
	populasyon[1][7]=128;
	populasyon[2][7]=11;
	populasyon[1][8]=21;
```

```
populasyon[2][8]=31;
populasyon[1][9]=69;
populasyon[2][9]=52;
populasyon[1][10]=142;
populasyon[2][10]=185;
populasyon[1][11]=12;
populasyon[2][11]=12;
populasyon[1][12]=198;
populasyon[2][12]=202;
populasyon[1][13]=15;
populasyon[2][13]=155;
populasyon[1][14]=45;
populasyon[2][14]=98;
populasyon[1][15]=1;
populasyon[2][15]=10;
populasyon[1][16]=28;
populasyon[2][16]=78;
populasyon[1][17]=42;
populasyon[2][17]=34;
populasyon[1][18]=13;
populasyon[2][18]=111;
populasyon[1][19]=169;
populasyon[2][19]=187;
populasyon[1][20]=45;
populasyon[2][20]=77;
populasyon[1][21]=195;
populasyon[2][21]=63;
populasyon[1][22]=205;
populasyon[2][22]=40;
populasyon[1][23]=267;
populasyon[2][23]=121;
populasyon[1][24]=15;
populasyon[2][24]=155;
populasyon[1][25]=216;
populasyon[2][25]=3;
```

```
	populasyon[1][26]=42;
	populasyon[2][26]=242;
	populasyon[1][27]=78;
	populasyon[2][27]=19;
	populasyon[1][28]=81;
	populasyon[2][28]=161;
	populasyon[1][29]=179;
	populasyon[2][29]=87;
	}
	while(mod<3){
		clock_t start_t, end_t;
		start_t = clock();
		nesil+=500;
		printf("-----------------------------------------------------\n");
		printf("Nesil sayisi: %d \n\n",nesil);
		randomAtaKromozomOlustur(populasyon,size,n);
		printf("Baslangic Populasyonu En Iyi Sonucu:\n");
		enIyiSecimi(populasyon,secim,3*size,size,n);
		mesafeHesapla(populasyon,secim*3,n);
		aramaFunc(populasyon,3*secim,n);
		iterasyon=0;
		while(iterasyon<nesil){
			if(iterasyon>0){		//ilk iterasyondan sonra cocuk ve ata bireylerin bulundugu matris icin mesafe hesapliyor.
				mesafeHesapla(populasyon,(secim*9)/2,n);
				enIyiSecimi(populasyon,secim,(9*secim)/2,size,n);
			}
			caprazlaFunc(populasyon,secim,n);
			mutasyonFunc(populasyon,secim,n);
			iterasyon++;
		}
		mesafeHesapla(populasyon,(secim*9)/2,n);
		printf("\n\nIslemler Sonucu Olusan Populasyon Havuzu En Iyi Sonucu:\n");
		aramaFunc(populasyon,(9*secim)/2,n);
		end_t = clock();
```

```
		sure[mod]=(double)(end_t - start_t) / CLOCKS_PER_SEC;
		printf("\nHesaplanan sure: %f\n", (double)(end_t - start_t) / CLOCKS_PER_SEC);
		printf("-------------------------------------------------------");
		printf("\n");
		mod++;
	}
	printf("		ANALIZ		\n\n");
	printf("-------------------------------------------------------\n\n\n");
	for(i=0;i<3;i++){
		temp+=500;
		printf("%d nesil\n",temp);
		for(j=0;j<sure[i]*10;j++){
			printf("*");
		}
		printf("%.3lf(sn)",sure[i]);
		printf("\n\n\n");
	}
}

void ataBireyAnalizFunc(double **populasyon){
	int size=800,n=30,secim=0,j,i,temp=0,nesil=500,mod=0,iterasyon;
	double sure[4]={0};
	printf("Kullanilacak Ata Birey Sayisina Gore Zaman Karmasikligi Analizi\n");
	printf("Degerler:\n\nBaslangic Populasyon Buyuklugu: 800\n\nKullanilan nokta sayisi:30\n\nNesil
Sayisi:500\n\nKullanilacak Ata Birey Sayisi: 40 - 80 - 120 - 160\n\n");

{	populasyon[1][0]=1;
	populasyon[2][0]=25;
	populasyon[1][1]=42;
	populasyon[2][1]=115;
	populasyon[1][2]=19;
	populasyon[2][2]=29;
	populasyon[1][3]=35;
	populasyon[2][3]=41;
	populasyon[1][4]=78;
```

```
populasyon[2][4]=10;
populasyon[1][5]=20;
populasyon[2][5]=45;
populasyon[1][6]=92;
populasyon[2][6]=99;
populasyon[1][7]=128;
populasyon[2][7]=11;
populasyon[1][8]=21;
populasyon[2][8]=31;
populasyon[1][9]=69;
populasyon[2][9]=52;
populasyon[1][10]=142;
populasyon[2][10]=185;
populasyon[1][11]=12;
populasyon[2][11]=12;
populasyon[1][12]=198;
populasyon[2][12]=202;
populasyon[1][13]=15;
populasyon[2][13]=155;
populasyon[1][14]=45;
populasyon[2][14]=98;
populasyon[1][15]=1;
populasyon[2][15]=10;
populasyon[1][16]=28;
populasyon[2][16]=78;
populasyon[1][17]=42;
populasyon[2][17]=34;
populasyon[1][18]=13;
populasyon[2][18]=111;
populasyon[1][19]=169;
populasyon[2][19]=187;
populasyon[1][20]=45;
populasyon[2][20]=77;
populasyon[1][21]=195;
populasyon[2][21]=63;
```

```
	populasyon[1][22]=205;
	populasyon[2][22]=40;
	populasyon[1][23]=267;
	populasyon[2][23]=121;
	populasyon[1][24]=15;
	populasyon[2][24]=155;
	populasyon[1][25]=216;
	populasyon[2][25]=3;
	populasyon[1][26]=42;
	populasyon[2][26]=242;
	populasyon[1][27]=78;
	populasyon[2][27]=19;
	populasyon[1][28]=81;
	populasyon[2][28]=161;
	populasyon[1][29]=179;
	populasyon[2][29]=87;
	}
	while(mod<4){
		clock_t start_t, end_t;
		start_t = clock();
		secim+=40;
		printf("-----------------------------------------------------\n");
		printf("Kullanilacak Ata Birey Sayisi: %d \n\n",secim);
		randomAtaKromozomOlustur(populasyon,size,n);
		printf("Baslangic Populasyonu En Iyi Sonucu:\n");
		enIyiSecimi(populasyon,secim,size*3,size,n);
		mesafeHesapla(populasyon,(secim*3),n);
		aramaFunc(populasyon,3*secim,n);
		iterasyon=0;
		while(iterasyon<nesil){
			if(iterasyon>0){		//ilk iterasyondan sonra cocuk ve ata bireylerin bulundugu matris icin
mesafe hesapliyor.
				mesafeHesapla(populasyon,(secim*9)/2,n);
				enIyiSecimi(populasyon,secim,(9*secim)/2,size,n);
			}
```

```
                caprazlaFunc(populasyon,secim,n);
                mutasyonFunc(populasyon,secim,n);
                iterasyon++;
            }
            mesafeHesapla(populasyon,(secim*9)/2,n);
            printf("\n\nIslemler Sonucu Olusan Populasyon Havuzu En Iyi Sonucu:\n");
            aramaFunc(populasyon,(9*secim)/2,n);
            end_t = clock();
            sure[mod]=(double)(end_t - start_t) / CLOCKS_PER_SEC;
            printf("\nHesaplanan sure: %f\n", (double)(end_t - start_t) / CLOCKS_PER_SEC);
            printf("-----------------------------------------------------");
            printf("\n");
            mod++;
        }
        printf("                ANALIZ                \n\n");
        printf("-----------------------------------------------------\n\n\n");
        for(i=0;i<4;i++){
            temp+=40;
            printf("%d Ata Birey\n",temp);
            for(j=0;j<sure[i]*10;j++){
                printf("*");
            }
            printf("%.3lf(sn)",sure[i]);
            printf("\n\n\n");
        }
}
```

Kaynakça:

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

https://stringfixer.com/tr/Genetic_algorithms

https://www.veribilimiokulu.com/genetik-algoritma/

https://tr.wikipedia.org/wiki/Genetik_algoritma

https://acikerisim.cumhuriyet.edu.tr/xmlui/bitstream/handle/20.500.12418/12058/10245372.pdf?sequence=1&isAllowed=y

https://www.youtube.com/watch?v=gL5iw5cvy0M&t=1854s&ab_channel=BilgisayarKavramlari

https://medium.com/algorithms-data-structures/algoritma-karma%C5%9F%C4%B1kl%C4%B1%C4%9F%C4%B1-big-o-5f14316890a4

https://tr.wikipedia.org/wiki/Sezgisel_algoritma

https://tr.wikipedia.org/wiki/Gezgin_sat%C4%B1c%C4%B1_problemi

http://tr-metaheuristic-tsp.herokuapp.com/travellingsalesmanproblem

https://tr.wikipedia.org/wiki/Kar%C4%B1nca_kolonisi_optimizasyon_algoritmas%C4%B1

Journal Of Emerging Economies And Policy Vol.2 (1) | July 2017 GEZGİN SATICI PROBLEMİNİN GENETİK ALGORİTMALARLA ÇÖZÜMÜNDE BAŞLANGIÇ POPÜLASYONUN BELİRLENMESİ

GENETĐK ALGORĐTMA VE UYGULAMA ALANLARI Mustafa KURT, Cumali SEMETAY**

Kemal ÇAKAR GENETİK ALGORİTMALAR YARDIMIYLA ACİL SERVİS İSTASYONU YERLEŞİMİNİN OPTİMİZASYONU