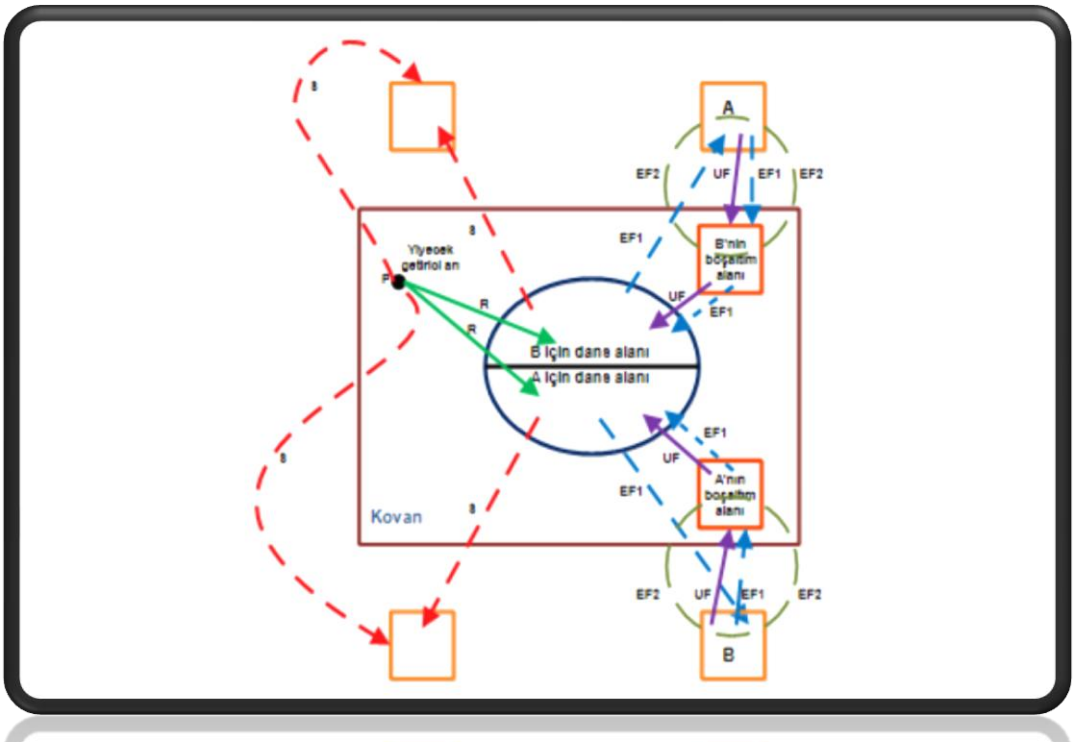
İçindekiler

Yapay Arı Kolonisi Algoritması

Doğada arıların besin arama davranışları insanlara ilham kaynağı olmuş ve bunun neticesinde ise Yapay Arı Kolonisi Algoritması(YAKA) geliştirilmiştir. YAKA'da arıların bütün davranışları bire bir modellenmemiş ve bunun yanında da bazı varsayımlarda bulunulmuştur. Bu varsayımlar her bir nektarın çıkarılmasında sadece bir görevli arının olmasıdır. Dolayısıyla algoritmada yer alan ve kullanılacak olan besin sayısı ile görevli arı sayısının birbirine eşit olması gerekmektedir. Bir diğer varsayım ise işçi arı ile gözcü arı sayısının birbirine eşit olmasıdır. Böyle bir varsayımda bulunulmasına rağmen aslında bir nektara gidip gelen arının görevli olduğu besin kaynağı tükendiğinde bu arının kâşif arı olması da söz konusudur. Bir besinin kalitesi ne kadar yüksekse o kaynağın uygunluk değeri de o denli iyidir. Dolayısıyla YAKA ile optimum çözümün elde edilmesine çalışılır. Bu noktada algoritmayı kullanan kişinin amacı maksimizasyon ya da minimizasyon olsun nektar kalitesi çözümün uygunluk değerine denk gelmektedir



Yukarıdaki şekil' de gösterildiği üzere kâşif arılar kovan çevresinde rastgele olarak besin kaynağı aramaya başlarlar. Besin kaynağı keşfinde bulunan kâşif arı bulduğu besin kaynağından kovana nektar taşımaya başlar. Kovana gelen arı nektarı boşalttıktan sonra üç olasılık söz konusudur. Bunlar; dans alanına giderek besin kaynağı ile ilgili bilgiyi diğer arılarla paylaşmak, hiç bilgi vermeden doğrudan besin kaynağına yönelmek ya da bulduğu besin kaynağını terk ederek yeniden kâşif arı olmaya devam etmektir. Kovanda bekleyen gözcü arılar da izledikleri dansa göre ilgili besin kaynağına yöneleceklerdir.

YAKA'nın adımları:

Adım 1: Rastgele besin kaynakları oluşturulur. Bu besin kaynaklarına sadık kalınarak işçi arı sayısı ve gözcü arı sayısı belirlenir. Ayrıca limit değeri de tespit edilir ve kontrol amaçlı sayaç değişkeni oluşturulur.

Adım 2: Oluşturulan bu besin kaynaklarına ait her bir besinin çözüm değerleri amaç fonksiyonunun türüne göre hesaplanır.

Adım 3: Maksimum döngü sayısı belirlenerek işçi arılar besin kaynaklarına gönderilir. İşçi arılar rastgele bir besine yönelerek bu besin kaynağını işlemeye başlarlar. Besin kaynağı işlendikten sonra bu besine ait yeni besin kalitesi(çözüm değeri) hesaplanır. Elde edilen çözüm değeri önceki çözüm değerinden daha iyi ise bu besin ve besinle ilgili bilgiler hafızaya alınırlar. Eğer çözüm değerinde iyileşme sağlanırsa limit değeri sıfırlanır aksi takdirde limit değeri bir arttırılır. Limit değeri için belirli bir üst değer belirlemek algoritmanın çalıştırılması esnasında sonsuz döngüye girmeye engel olacaktır.

Adım 4: İşçi arılardan sonra gözcü arılar devreye girerler. Besinlerin uygunluk değerine göre bir besin kaynağı seçilir. Gözcü arılar bu besin kaynağı üzerinde çalışmaya başlarlar. Aynı şekilde elde edilen çözüm değeri önceki çözüm değerinden daha iyi ise bu besin ve besinle ilgili bilgiler hafızaya alınırlar. Eğer besin kaynağında iyileşme sağlanırsa limit değeri sıfırlanır aksi takdirde limit değeri bir arttırılır. Bu safhada gözcü arılar işçi arılardan farklı olarak uygunluk değerine göre seçim yaparlar.

Adım 5: İşçi arı ve gözcü arı safhasından sonra kâşif arı devreye girer. Kâşif arı safhasının esas nedeni algoritmanın yerel minimum ya da maksimumda takılmasına engel olmaktır. Dolayısıyla elde edilmiş olan çözümü tamamıyla bozarak yani limit değerleri tamamen sıfırlanarak yeni bir çözüm değeri üretilmesini sağlar. Elde edilen çözüm değeri ile önceden hafızaya alınmış olan çözüm değeri karşılaştırılır. Bu iki çözüm değerinden iyi olanının hafızada tutulmasına devam edilir.

Adım 6: Maksimum döngü sayısı sağlanıncaya kadar işçi arı, gözcü arı ve kâşif arı safhası devam ettirilir. Durdurma kriteri sağlanınca algoritma sonlandırılır.

Yukarıda yer alan YAKA 'nın adımlarından da anlaşıldığı üzere YAKA 'yı dörde ayırmak mümkündür. Bunlar rastgele besin kaynaklarının üretilmesi, işçi arıların besin kaynaklarına gönderilmesi, gözcü arıların uygunluk değerine göre besin kaynağı seçmesi ve en son olarak da nektarı tükenen besin kaynağının terk edilmesidir. Bu noktada sırası ile yukarda yer alan bu dört adımdan bahsetmek yerinde olacaktır.

Rastgele Besin Kaynaklarının Üretilmesi

Besin kaynakları arama yapılan çözüm uzayında yer alacaktır. Dolayısıyla algoritmada ilk önce yapılması gereken şey bu besin kaynaklarının yerinin tespit edilmesidir. Besin kaynaklarının yerlerinin tespit edilmesi ile ilgili eşitlik aşağıda yer almaktadır.

$$x_{ij} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min})$$

Burada besin kaynağı sayısı *i* ile parametre sayısı ise *j* ile ifade edilmektedir. Yani önceden belirlenmiş olan bir alt değer ile üst değer arasındaki değerlerden oluşan besin kaynaklarının üretilmesi sağlanmış olur.

İşçi Arıların Besin Kaynaklarına Gönderilmesi

Arama uzayında çözüm değerleri araştırılırken işçi arılar besin kaynaklarından bir tanesini rastgele olarak belirlerler ve bu besin kaynağının kalitesini yani çözüm değerini hesaplarlar. Elde edilen çözüm değeri hafızaya alınır. Daha sonra işçi arılar besin kaynaklarına yöneldikçe hafızadaki bilgiler problemin amacına göre güncellenerek hafızada korunmaya devam edilir. Burada çözüm değerini iyileştiren değerlerin hafızada tutulacağını hatırlatmakta fayda vardır. Bu durum aşağıda yer alan eşitlikte yer almaktadır. $\upsilon_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ Eşitlikte yer alan υ_{ij} ile parametrelerin önceden belirlenmiş olan parametre sınırları arasında yer alması sağlanmaya çalışılmaktadır. Bu durum aşağıda yer alan eşitlikte yer almaktadır.

$$\upsilon_{ij} = \begin{cases} x_{ij} & , \upsilon_{ij} < x_j^{min} \\ \upsilon_{ij} & , x_j^{min} \le \upsilon_{ij} \le x_j^{max} \\ x_j^{max} & , \upsilon_{ij} > x_j^{max} \end{cases}$$

Bu bilgiler ışığı altında besin kaynağının uygunluk değeri aşağıda yer alan eşitliğe göre hesaplanır.

$$u_i = \begin{cases} 1/(1+u_i) & , \quad u_i \ge 0 \\ 1 + mutlak\ değer(u_i) & , \quad u_i < 0 \end{cases}$$

Burada u_i ile besin kaynağının uygunluk değeri ifade edilmektedir. Uygunluk değerinin hesaplanmasında problemin yapısı ön plana çıkmaktadır yani problemin maksimizasyon ya da minimizasyon olması durumuna göre uygunluk hesaplaması

yapılmaktadır. Uygunluk değerine göre seçilen besin kaynağının çözüm değeri hesaplanır. Eğer elde edilen çözüm önceki çözümden daha kötü ise sayaç bir arttırılarak önceden belirlenmiş olan limit değeri ile karşılaştırılır. Aksi halde önceki çözüm değerinden daha iyi bir çözüm değeri elde edilmesi durumunda ise sayaç sıfırlanır. Daha önce de belirtildiği üzere limit değeri ile karşılaştırma yapılmasının nedeni artık daha fazla iyileştirilemeyen besin kaynaklarını değerlendirme dışı bırakarak sonsuz döngüye girmeye engel olmaktır. Bu noktada kovanda bekleyerek dans alanındaki işçi arıları izleyen gözcü arılar önceden hesaplanmış olan uygunluk değerine göre ilgili besin kaynağına yöneleceklerdir. Uygunluk değerlerinin hesaplanmasında çeşitli yöntemler mevcuttur. Bunlar rulet tekerleği seçim yöntemi, sıralamaya dayalı seçim yöntemi, stokastik örnekleme, turnuva yöntemi gibi yöntemlerdir. YAKA'da rulet tekerleği seçim yöntemi kullanılmıştır. Bu yöntemde rulet tekerleği bir pasta gibi düşünülebilir. Pastanın her bir dilimi bir uygunluk değerine denk gelmektedir dolayısıyla uygunluk değeri yüksek olan çözüm değerinin seçilme olasılığı diğerlerinin seçilmesi olasılığından daha yüksektir. Aşağıda yer alan eşitlikte rulet tekerleği seçim yönteminde seçim olasılığının hesaplanış şekli yer almaktadır.

$$\rho_i = \frac{uygunluk_i}{\sum_{j=1}^{SN} uygunluk_i}$$

Yukarıdaki eşitlikte $uygunluk_i$ ile *i.* kaynağın uygunluk değeri, *SN* ile işçi arı sayısı ifade edilmektedir. Yani hesaplanan uygunluk değerinin toplam uygunluk değerine oranlanması ile pastanın dilimlerinin bir diğer ifade ile rulet tekerleğinde yer alan parçaların genişlikleri elde edilmiş olmaktadır.

Gözcü Arıların Besin Kaynaklarına Gönderilmesi

Yukarıda yer alan eşitlikte hesaplanan uygunluk değerine göre gözcü arılar kovandan ayrılarak ilgili besin kaynaklarına yönelerek yeni bir çözüm değeri hesaplar. Elde edilen çözüm değeri önceden hesaplanmış olan ve hafızada tutulan çözüm değeri ile karşılaştırılır. İlgili çözüm değeri önceki çözüm değerinden daha iyi ise sayaç sıfırlanır aksi halde sayaç bir arttırılır. Bütün gözcü arılar besin kaynağına gidene kadar bu süreç böyle devam eder.

Besin Kaynağının Terk Edilmesi ve Kâşif Arı Üretilmesi

Yukarıda yer alan ikinci ve üçüncü aşama yani işçi arıların besin kaynaklarına gönderilmesi ile gözcü arıların besin kaynaklarına gönderilmesi aşamaları tamamlandıktan sonra eğer sayaç limit değerini aşmışsa yani artık çözüm değeri daha fazla iyileştirilemiyorsa kâşif arılar görevi devralırlar. Gerçek hayatta bu durumu şöyle açıklamak mümkündür. Bir besin kaynağının nektarı tükenmişse nektarın çıkarılmasından sorumlu olan işçi arı yeni besin kaynaklarının araştırılmasından sorumlu olmak üzere kâşif arı olmaktaydı. İşte aynı gerçek arıların besin arama davranışlarında olduğu gibi YAKA'da da belirli bir limit adedince iyileştirilemeyen çözüm değeri için kâşif arılar üretilmekte ve bu kâşif arılar aracılığıyla yeni bir besin kaynağı oluşturulup bu besin kaynağının çözüm değeri hesaplanmaktadır. Oluşturulan yeni besin kaynağının çözüm değeri önceki çözüm değeri ile karşılaştırılmakta ve elde edilen çözüm değeri iyiyse hafızaya alınmakta aksi takdirde ihmâl edilmektedir. Bütün bu adımlar önceden belirlenmiş olan döngü adedince gerçekleştirilerek durdurma kriteri sağlandığında algoritma sonlandırılarak döngüden çıkılır.

Yapay arı kolonisi algoritması kullanarak gezgin satıcı probleminin Türkiye 'deki il merkezlerine uygulanması

Bilgisayar teknolojisindeki gelişmelerin artmasıyla optimizasyon yöntemleri fen, sosyal ve sağlık bilimleri alanlarındaki birçok uygulamanın çözümünde kullanılmaktadır. Sezgisel algoritmalar, özellikle birden farklı çözüme sahip olan problemlere uygulanan güçlü bir alternatif yöntemdir. Bu problemlerden biri de üzerinde sıkça çalışılan ve literatürde gezgin satıcı problemi olarak bilinen en kısa yol bulma problemidir. Bu problem, haritada yer alan bir noktadan hareket eden satıcının her noktaya bir kere uğradıktan sonra en kısa şekilde başlangıç noktasına tekrar dönmesi üzerinedir. Gelişime dayalı ve sürü zekâsı temelli olmak üzere iki ana başlığa ayrılan sezgisel algoritmalar çeşitlilik göstermektedir. Bu çalışmada sürü zekâsı temelli bir sezgisel algoritma olan yapay arı kolonisi algoritması kullanılarak Türkiye'deki il merkezleri için gezgin satıcı problemine çözüm bulunması hedeflenmiştir. Son yıllarda oldukça popüler olan yapay arı kolonisi algoritması bal arılarının doğadaki besin arama davranışından

esinlenilerek geliştirilen etkili yöntemdir. Bu çalışmada arı kolonisi optimizasyonu (BCO) olarak bilinen algoritma ile yapay arı kolonisi algoritmasının (ABC) bir arada kullanıldığı hibrit yapıda bir algoritma kullanılmıştır. Bununla birlikte bir yerel arama algoritması olan Opt-2 tekniği bu hibrit yapı içerisine dahil edilerek lokal alanda iyileştirme gerçekleştirilmiştir. Literatürde sonuçları bilinen simetrik ve asimetrik yapıda küçük, orta ve büyük boyutlu gezgin satıcı test problemlerine hibrit yapıdaki algoritma uygulanmış ve sonuçlar karşılaştırılmıştır. Bununla birlikte ülkemizdeki 81 il merkezi için deneysel uygulamalar gerçekleştirilmiş ve elde edilen sonuçlar literatür ile karşılaştırmalı olarak değerlendirilmiştir. Çalışmada alınan sonuçlara göre, gezgin satıcı probleminin çözümünde yapay arı kolonisi algoritması parçacık sürü optimizasyonu ve genetik algoritmaya oranla daha yüksek performans göstermektedir.

EKLER

EK1-Algoritma.m

```
clear all;
clc;
%sehirsayisi=81;
sehirsayisi=77;
isciarisayisi=15;
gozcuarisayisi=15;
besinsayisi=50;
limit=100;
%ilmesafe=xlsread('ilmesafe.xls');
ilmesafe=xlsread('eceson.xls');
eniyicozumdegeri=1000000;
besin=besinmatrisi(besinsayisi,sehirsayisi);
denemesayisi=zeros(size(besin(:,1)));
for i=1:besinsayisi
    cozumdegeri(i)=amacfonksiyonu(besin(i,:),ilmesafe,sehirsayisi);
    if eniyicozumdegeri>cozumdegeri(i)
        eniyicozumdegeri=cozumdegeri(i);
        eniyicozum=besin(i,:);
    end
end

for iterasyon=1:1000
%işçi arı safası başlar
for i=1:isciarisayisi
    degisecekbesinno=randi([1 besinsayisi]);
    r=randi(1);
    indis1=randi([1 sehirsayisi]);
    indis2=randi([1 sehirsayisi]);
    while indis1==indis2
        indis2=randi([1 sehirsayisi]);
    end
    if r<1/3
        yenibesin=yerdegistir(besin(degisecekbesinno,:),indis1,indis2);
    elseif r<2/3
        yenibesin=solakaydir(besin(degisecekbesinno,:),indis1,indis2);
    else

        yenibesin=diziyiterscevir(besin(degisecekbesinno,:),indis1,indis2);
    end
    yenicozumdegeri=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
    if yenicozumdegeri<cozumdegeri(degisecekbesinno)
        cozumdegeri(degisecekbesinno)=yenicozumdegeri;
        besin(degisecekbesinno,:)=yenibesin;
```

```
        denemesayisi(degisecekbesinno)=0;
        if eniyicozumdegeri>yenicozumdegeri
            eniyicozumdegeri=yenicozumdegeri;
            eniyicozum=yenibesin;
        end
    else
        denemesayisi(degisecekbesinno)=denemesayisi(degisecekbesinno)+1;
    end
end

%işçi arı safası bitti

%gözcü arı safası baslar
sabit=1;
for i=1:besinsayisi
    minicinuygunluk(i)=sabit/cozumdegeri(i);
end
cozumdegerleritoplami=0;
for i=1:besinsayisi
    cozumdegerleritoplami=cozumdegerleritoplami+minicinuygunluk(i);
end
for i=1:besinsayisi
    uygunluk(i)=minicinuygunluk(i)/cozumdegerleritoplami;
end
for i=1:gozcuarisayisi
    degisecekbesinno=besinsec(uygunluk);
    r=rand(1);
    indis1=randi([1 sehirsayisi]);
    indis2=randi([1 sehirsayisi]);
    while indis1==indis2
        indis2=randi([1 sehirsayisi]);
    end
    if r<1/3
        yenibesin=yerdegistir(besin(degisecekbesinno,:),indis1,indis2);
    elseif r<2/3
        yenibesin=diziyiterscevir(besin(degisecekbesinno,:),indis1,indis2);
    else
        yenibesin=solakaydir(besin(degisecekbesinno,:),indis1,indis2);
    end
    yenicozumdegeri=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
    if yenicozumdegeri<cozumdegeri(degisecekbesinno)
        cozumdegeri(degisecekbesinno)=yenicozumdegeri;
        besin(degisecekbesinno,:)=yenibesin;
        denemesayisi(degisecekbesinno)=0;
        if eniyicozumdegeri>yenicozumdegeri
            eniyicozumdegeri=yenicozumdegeri;
            eniyicozum=yenibesin;
        end
```

```
    else
    denemesayisi(degisecekbesinno)=denemesayisi(degisecekbesinno)+1;
    end
end
%gozcuarı safası bitti
%kasif ari safası başlar
for i=1:besinsayisi
    if denemesayisi(i)>limit;
        denemesayisi(i)=0;

[besin(i,:),cozumdegeri(i)]=yerelarama1(besin(i,:),cozumdegeri(i),ilmesafe,sehirsayisi);

if eniyicozumdegeri>cozumdegeri(i)
    eniyicozumdegeri=cozumdegeri(i);
    eniyicozum=besin(i,:);
end

[besin(i,:),cozumdegeri(i)]=yerelarama2(besin(i,:),cozumdegeri(i),ilmesafe,sehirsayisi);

if eniyicozumdegeri>cozumdegeri(i)
    eniyicozumdegeri=cozumdegeri(i);
    eniyicozum=besin(i,:);
end
    end
end
% kasif safası bitti
fprintf('iterasyon:%d Eniyicozum: %d \n', iterasyon,eniyicozumdegeri);


end
```

EK2-Amacfonksiyonu.m

```
function uzaklik=amacfonksiyonu(besin,ilmesafe,sehirsayisi)
ustlimit=(sehirsayisi-1);
uzaklik=0;
for i=1:ustlimit
    yeniuzaklik=ilmesafe(besin(i),besin(i+1));
    uzaklik=uzaklik+yeniuzaklik;
end
yeniuzaklik=ilmesafe(besin(i+1),besin(1));
uzaklik=uzaklik+yeniuzaklik;
```

EK3-Besinmatrisi.m

```
function matris=besinmatrisi(besinsayisi,sehirsayisi)
```

```
altlimit=1;
ustlimit=sehirsayisi;
matris=[];
for i=1:besinsayisi
   elemanlarifarklidizi=rakamlarifarklidizi(sehirsayisi,altlimit,ustlimit);
   matris=[matris;elemanlarifarklidizi];
end
```

EK4-Besinmatrisi2.m

```
function matris=besinmatrisi2(besinsayisi,sehirsayisi)
matris=[];
for i=1:besinsayisi
   elemanlarifarklidizi=randperm(sehirsayisi);
   matris=[matris;elemanlarifarklidizi];
end
```

EK5-Besinsayisi2.m

```
function matris=besinmatrisi2(besinsayisi,sehirsayisi)
matris=[];
for i=1:besinsayisi
   elemanlarifarklidizi=randperm(sehirsayisi);
   matris=[matris;elemanlarifarklidizi];
end
```

EK6-Besinsec.m

```
function bs=besinsec(uygunluk)
toplam=0;
r=rand(1);
for i=1:length(uygunluk)
   toplam=toplam+uygunluk(i);
   if r<toplam
      bs=i;
      break;
   end
end
```

EK7-DiziyiTersCevir.m

```
function yenibesin=diziyiterscevir(besin,indis1,indis2)
yenibesin=besin;
```

```
kucuk=min(indis1,indis2);
buyuk=max(indis1,indis2);
while kucuk<buyuk
   bosdegisken=yenibesin(kucuk);
   yenibesin(kucuk)=yenibesin(buyuk);
   yenibesin(buyuk)=bosdegisken;
   kucuk=kucuk+1;
   buyuk=buyuk-1;
end
```

EK8-Icindemevcutmu.m

```
function var=icindemevcutmu(dizi,aranandeger)
var=0;
for i=1:length(dizi)
   if dizi(i)==aranandeger
      var=i;
      break;
   end
end
```

EK9-Rakamlarifarklidizi.m

```
function dizi=rakamlarifarklidizi(aralik,altlimit,ustlimit)
if aralik>=1 && aralik<=(ustlimit-altlimit+1)
   dizi=[];
   for i=1:aralik
      rastgelesayi=round(altlimit+(ustlimit-altlimit)*rand(1));
      while icindemevcutmu(dizi,rastgelesayi)
         rastgelesayi=round(altlimit+(ustlimit-altlimit)*rand(1));
      end
      dizi(i)=rastgelesayi;
   end
else
   disp('hata');
end
```

EK10-Solakaydir.m

```
function yenibesin=solakaydir(besin,indis1,indis2)
yenibesin=besin;
kucuk=min(indis1,indis2);
buyuk=max(indis1,indis2);
ilkeleman=yenibesin(kucuk);
for i=kucuk:(buyuk-1)
   yenibesin(i)=yenibesin(i+1);
```

```
end
yenibesin(buyuk)=ilkeleman;
```

EK11-Yerdegistir.m

```
function yenibesin=yerdegistir(besin,indis1,indis2)
yenibesin=besin;
bosdegisken=yenibesin(indis1);
yenibesin(indis1)=yenibesin(indis2);
yenibesin(indis2)=bosdegisken;
end
```

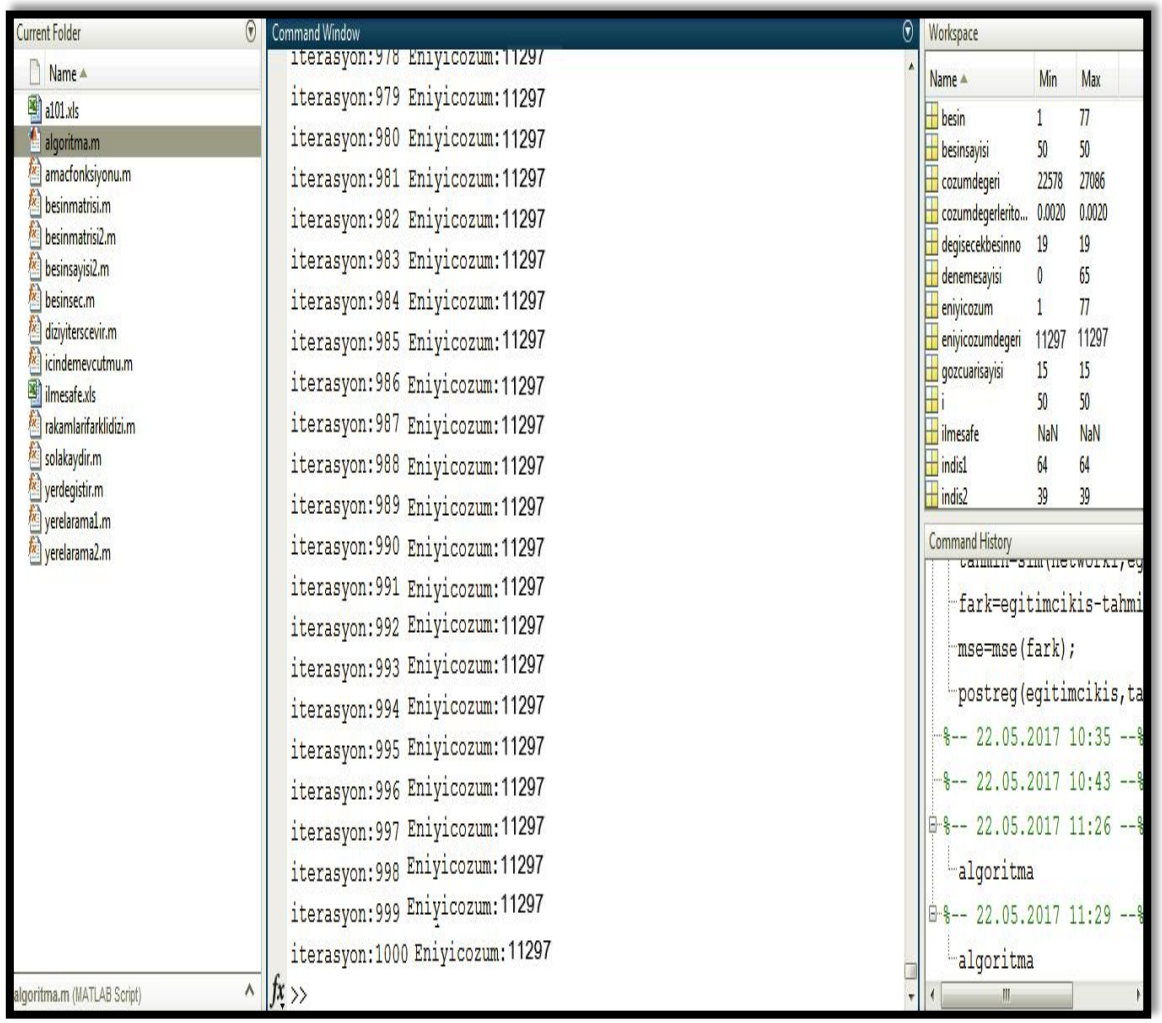
EK12-Yerelarama1.m

```
function
[sonucbesin,sonuccozumdegeri]=yerelarama1(besin,cozumdegeri,ilmesafe,sehirsayisi)
sonucbesin=besin;
sonuccozumdegeri=cozumdegeri;
for i=1:length(sonucbesin)-1
   for j=1:length(sonucbesin)
      yenibesin=solakaydir(sonucbesin,i,j);
      yeniamacfonksiyonu=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
      if yeniamacfonksiyonu<sonuccozumdegeri
         sonuccozumdegeri=yeniamacfonksiyonu;
         sonucbesin=yenibesin;
      end
   end
end
```

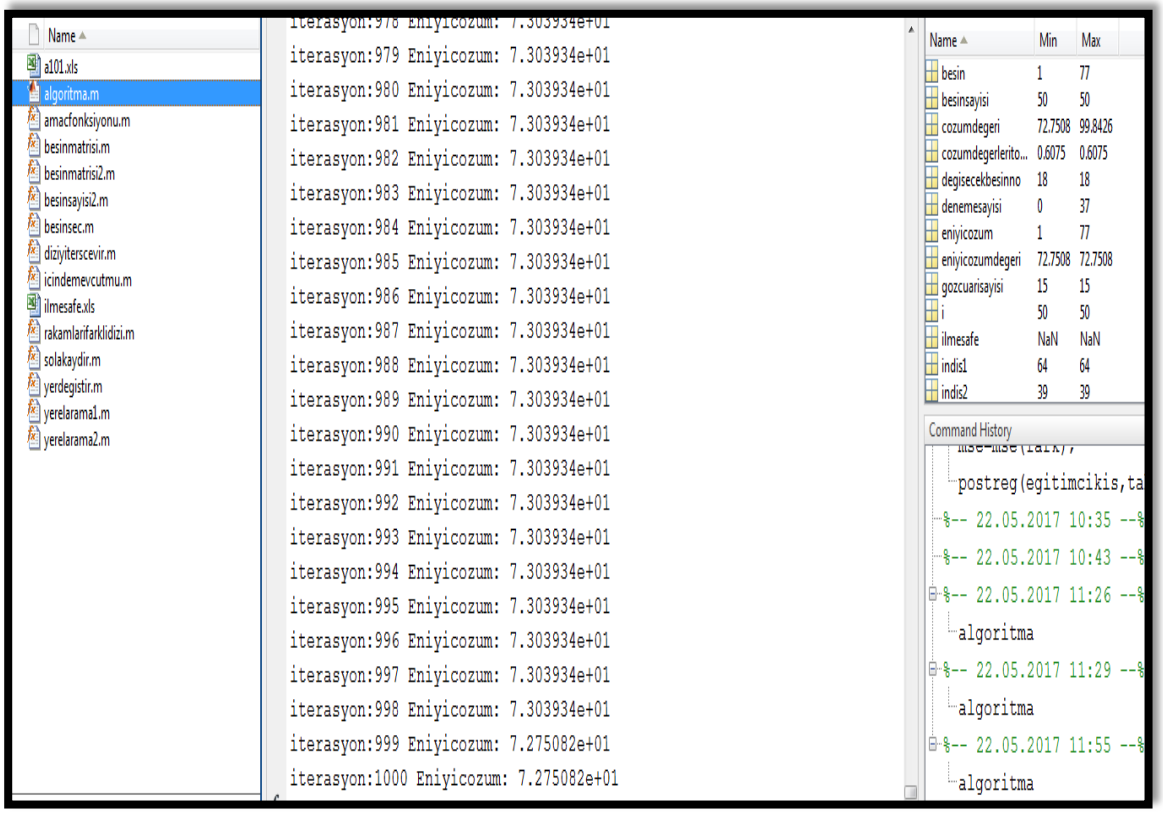
EK13-Yerelarama2.m

```
function
[sonucbesin,sonuccozumdegeri]=yerelarama2(besin,cozumdegeri,ilmesafe,sehirsayisi)
sonucbesin=besin;
sonuccozumdegeri=cozumdegeri;
for i=1:length(sonucbesin)-1
   for j=1:length(sonucbesin)
      yenibesin=diziyiterscevirme(sonucbesin,i,j);
      yeniamacfonksiyonu=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
      if yeniamacfonksiyonu<sonuccozumdegeri
         sonuccozumdegeri=yeniamacfonksiyonu;
         sonucbesin=yenibesin;
      end
   end
end
```

TÜRKİYE İLLER ARASI MESAFE ÇIKTISI



İZMİR İLİ A101 ŞUBELER ARASI ÇIKTI



KAYNAKÇA

http://www.kgm.gov.tr/SiteCollectionDocuments/KGMdocuments/Root/Uzakliklar/ilmesafe.xls

Abbass, H.A. 2001 MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. In: Proceedings of the Congress on Evolutionary Computation. Seoul, South Korea 207- 214

Afshar, A., Bozorg Haddada, O, Marin, M.A., Adams, B.J. 2007. Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. J. Frank. Instit. 344 452–462

Akay B. 2009, Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificıal Bee Colony) Algoritmasının Performans Analizi, Doktora Tezi, Kayseri

AslantaĢ ve ark., 2010, Multi-focus Image Fusion in Spatial Domain using ABC Optimization Algorithm , Erciyes University, Engineering Faculty, Intelligent Image Processing Lab, Kayseri

Baykasoglu, A., Özbakýr, L., Tapkan, P. 2007: Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem. In: Eds. Felix T. S. Chan and Manoj Kumar Tiwari: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Itech Education and Publishing, Vienna, Austria113-143

Benatchba, K., Admane, L., Koudil, M. 2005. Using Bees to Solve a Data-Mining Problem Expressed as a Max-Sat One. In: J. Mira and J.R. ´Alvarez (Eds.): IWINAC 2005, LNCS, 3562, Springer-Verlag Berlin Heidelberg 212–220

Bicthahn. J. ve Nissen, V., 1995. "An Introduction to Evolutionary Algorithms. Evolutionary Algorithms in Management Applications", s. 3-43, Biethahn, J. & Nissen. V.. Springer Verlag, Berlin Heidelberg New York.

Bingül, Z., Sekmen, A.S. and Zein, S., 1999. An Application of Multi-Dimensional Optimization Problems Using Genetic Algorithms. Proceedings of the IASTED International Conference Intelligent Systems and Control, Santa Barbara, CA, USA.

Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L. 2006. A Bee Colony Optimization Algorithm to Job Shop Scheduling Simulation. In: L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, (Eds.): Proceedings of the Winter Conference, Washington, DC 1954 - 1961

Drias, H., Sadeg, S. and Yahi, S. 2005. Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. In : Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science 3512, Springer Berin/Heilderberg 318-325

Erçin, Ö. 2011, Multi-objective Bee Colony Optimization To Tuning PID Controller, Yüksek Lisans Tezi, Adana

http://www.muhlisozdemir.com/blog/yapay-ari-kolonisi-algoritmasi/?lang=TR

http://acikerisim.selcuk.edu.tr:8080/xmlui/handle/123456789/1810