SEZGİSEL YÖNTEMLER VE UYGULAMALARI FİNAL SINAV PROJE RAPORU – BAHAR 2018

SEZGİSEL ALGORİTMALAR- YAPAY ARI KOLONİSİ (ARTİFİCİAL BEE COLONY-ABC) ALGORİTMASI KULLANILARAK GEZGİN SATICI PROBLEMİNİN UYGULANMASI

16 Mayıs 2018

SADIK İZGİ - 14253607

İçindekiler

1.Giriş

GSP ilk olarak 1930'lu yıllarda matematiksel olarak tanımlanmıştır. Problem tanımı basit olmasına rağmen çözümü zordur. Problemde kullanılan şehir sayısının artışına paralel olarak çözüm uzayı genişlemekte, problemin çözüm zamanı ve zorluğu artmaktadır. Bu nedenle problemin çözümünde analitik çözüm yöntemleri yetersiz kalmaktadır. Tüm çözüm uzayını taramak yerine mantıksal çıkarımlar ile çözüm uzayında kısmi taramalar yapan sezgisel yöntemler problemin çözümünü garanti etmemekle beraber, çözüm maliyetini azaltmaktadır.

2.Sezgisel Algoritmalar

Sezgisel ya da buluşsal (heuristic) bir problem çözme tekniğidir. Sonucun doğruluğundan kanıtlanabilir olup olmadığını önemsemez. Çeşitli alternatif hareketlerden etkili olanlara karar vererek iyiye yakın çözüm yolları elde etmeyi amaçlar. Makul bir süre içerisinde bir çözüm elde edeceklerini garanti ederler. Genellikle en iyi yakın olan çözüm yoluna hızlı ve kolay ulaşırlar.

Neden tercih edilirler?

- Optimizasyon problemi, kesin çözümü bulma işleminin tanımlandığı bir yapıya sahip olabilir.
- Anlaşıla bilirlik açısından sezgisel algoritmalar kara verici olduğu için çok daha basit olabilir.
- Öğrenme amaçlı ve kesin çözüm bulma işleminin bir parçası olabilir.
- Matematiksel olarak yapılan tanımlamalarda genellikle birçok parametre/Kısıt ihmal edilir.
- Bu durumda, model parametrelerini belirleme aşamasında kullanılan verinin hatalı olması, sezgisel yaklaşımın üretebileceği alt optimal çözümden daha büyük hatalara neden olabilir.

Sezgisel Yöntemlerden bazıları:

- Genetik Algoritma
- Karınca Kolonisi Optimizasyonu
- Parçacık sürü Optimizasyonu
- Yapay arı kolonisi
- Diferansiyel Gelişim Algoritması
- Isı Transferi Arama
- Yerçekimi Arama Algoritması
- Yasak Arama Algoritması
- Orman Optimizasyonu Algoritması
- Ağaç-Tohum Algoritması
- Kasırga Temelli Optimizasyon Algoritması
- Ağırlıklı Süperpozisyon Çekimi

3.Gezgin Satıcı Problemi

GSP, n adet şehir arasındaki mesafelerin bilindiği durumda, şehirlerin her birine yalnız bir kez uğramak şartıyla, başlangıç noktasına geri dönülmesi esasına dayalı, tur boyunca kat edilen toplam yolun en kısa olduğu şehir sıralamasının (optimal rota) bulunmasının amaçlandığı bir problemdir. Dağıtım, rotalama, kuruluş yeri belirleme, planlama, lojistik gibi problemlerde geniş bir uygulama alanına sahip olan gezgin satıcı problemi, aynı zamanda optimizasyon alanında, araştırmacılar tarafından üzerinde uzun yıllardır çalışmalar yapılan NP-hard (çözümü zor) sınıfında yer alan bir problemdir.

Eğer şehirler düğümlerle, yollar ise hatlar ile gösterilirse problem çizge üzerinde minimum maliyetli kapalı yolun bulunmasına karşılık gelmektedir.
Problemde düğüm sayısı arttıkça problemin çözümünde harcanan zaman üstel olarak artmaktadır.

GSP'de artan düğüm sayısına paralel olarak çözüm zamanının üstel olarak artışı Tablo 1'de gösterilmektedir.

Düğüm Sayısı	Döngü Sayısı(n-1)!	Gerekli Zaman
12	39.916.800	0,004 Saniye
13	479.001.600	0,05 Saniye
14	6.227.020.800	1 Saniye
15	87.178.291.200	9 Saniye
16	1.307.647.368.000	2 Dakika
17	2.1 * 1013	35 Dakika
18	3.6 * 1014	10 Dakika
19	6.4 * 1015	7.5 Gün
20	1.2 * 1017	140 Gün
21	2.4 * 1018	7.5 Yıl
22	5.1 * 1019	160 Yıl
23	1.1 * 1021	3,500 Yıl
24	2.6 * 1022	82.000 yıl
25	6.2 * 1023	2 milyon yıl

Tablo 1 Hamilton Döngülerinin Değerlendirilmesi [2]

Problemde başlangıç şehri verilmişse, mümkün olan Hamilton yolları sayısı geriye kalan (n-1) adet şehrin yer değişmesine, yani (n-1)!'e eşit olmaktadır. Bu durumda problem basit olmasına rağmen, problemin çözümünü çözüm uzayının tamamını taramakla bulmak çok iyi bir yaklaşım değildir. Problemin en azından bir basit çözümünün olacağı kesindir. Bu sebeple GSP problemlerinin çözümünde sezgisel ve metasezgisel tekniklerin kullanılması etkin bir yoldur.

Gezgin Satıcı probleminin çözümü;

- Başlangıç için seçilebilecek **n** tane şehir vardır.
- Bu şehirlerden birisi başlangıç noktası olacağı için, satıcı **n-1** farklı şehirde satış yapabilir.
- İkinci şehire geçince satıcının satış yapabileceği şehir sayısı **n-2** olur.

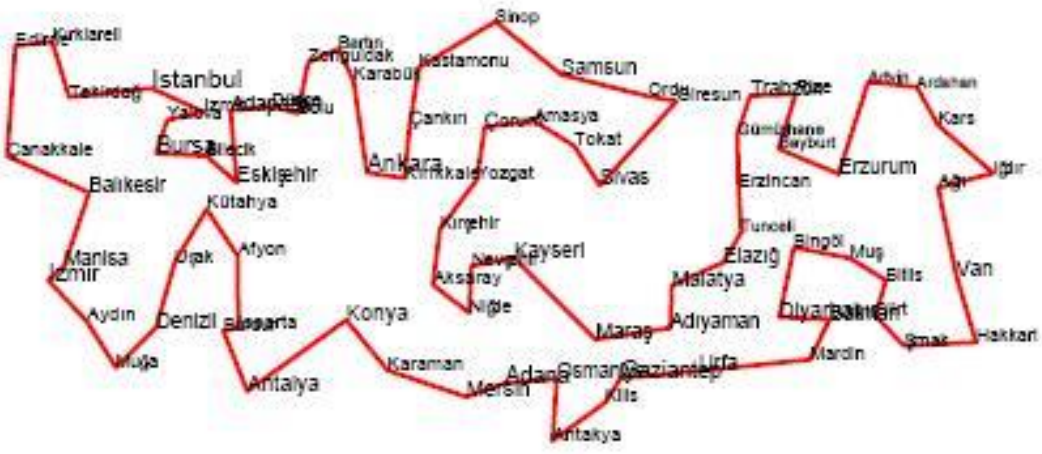
Buradan yola çıkılarak satıcının **n!** değişik tur arasından seçimi olacaktır. Bu ise 100 şehirlik bir tur için **9,33*10^157** değişik tur demektir.

Gezgin Satıcı Probleminin uygulama alanları;

- GSM operatörlerinin baz istasyonları için yer belirlemesinde,
- Malzeme akış sistemi tasarımında,
- Posta kutusu dağıtım probleminde,
- Araç rotalama probleminde,
- Uçaklar için havaalanı rota lamasında,
- Elektronik devre tasarımında

Gibi birçok alanda kullanımları mevcuttur.

Gezgin satıcı Örneği:

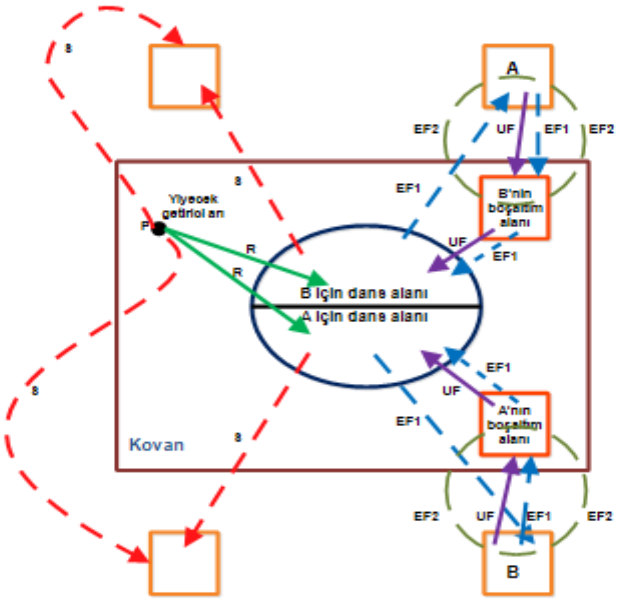


ŞEKİL -1 Bir gezgin satıcı yol modelidir. [internet -1]

Gezgin satıcının Türkiye İller haritasında kullanılmasının bir örneğidir. Bu örnekte en kısa mesafe kuş uçuşu olarak tasarlanmış ve bulunmuştur.

4.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 biri birine 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.



ŞEKİL -2 Arıların Yem Arama Davranışları [1]

Şekil 1' de gösterildiği üzere kaşif 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 ile ilgili kaba kod aşağıda yer almaktadır.

4.1 Yapay Arı Kolonisi Algoritması (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.

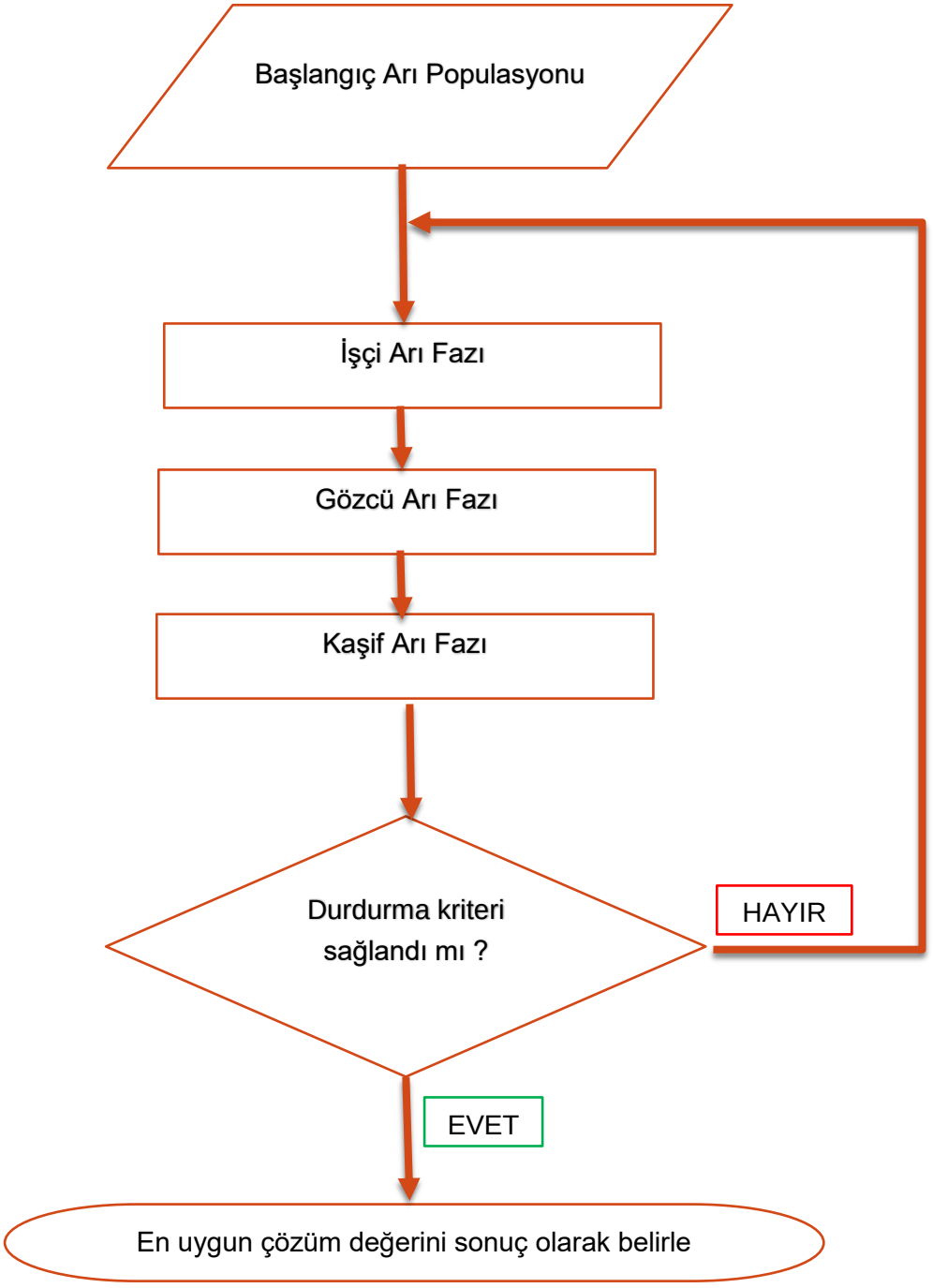
Adım 3: Maksimum döngü sayısı belirlenerek işçi arılar besin kaynaklarına gönderilir. İşçi arılar rastgele bir besine yönelerek bu besin kaynağını işlemeye başlarlar. Besin kaynağı işlendikten sonra bu besine ait yeni besin kalitesi (çözüm değeri) hesaplanır. Elde edilen çözüm değeri önceki çözüm değerinden 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.



ŞEKİL -2 Yapay Arı Kolonisine Akış Çizelgesi [2]

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.

$$X_{ij} = X_j^{min} + rand(0,1)\left(X_j^{max} - X_j^{min}\right)$$

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.

$$V_{ij} = X_{ij} + \emptyset_{ij}(X_{ij} - X_{kj})$$

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.

$$\boldsymbol{V_{ij}} = \begin{pmatrix} X_{\text{İ}J} & , & V_{ij} < X_j^{max} \\ V_{\text{İ}j} & , & X_j^{min} \leq V_{ij} \leq X_j^{max} \\ X_j^{max} & , & V_{ij} > X_j^{max} \end{pmatrix}$$

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

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

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.

$$P_i = \frac{uygunluk_i}{\sum_{j=1}^{SIN} uygunluk_i}$$

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 Kaşif 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.

5.Sözde Kod (Kaba kod)

Ari → rasgele yiyecek kaynağı çözümleriyle başlangıç arı popülasyonu

Eniyi →rasgele başlangıç yiyecek kaynağı çözümü

Maxziyaretsayısı → işçi arıların geliştirme olmaksızın aynı kaynağa yapacağı ziyaret sayısı

HataOlasılığı≈0,01 →işçi arının daha iyi bir komşu yiyecek kaynağını reddetme ya da daha kötü bir komşu yiyecek kaynağını kabul etme olasılığı.

İknaOlasılığı≈0,90 → gözcü arının daha iyi bir çözümden etkilenme olasılığı.

```
for Ari ∈ B do
        if (NektarMiktarı(Ari) > NektarMiktarı(Eniyi)) then
            Eniyi ← Ari
Repeat
    for Ari ∈ Kaşif Arı do
        RF ← rasgele yiyecek kaynağı çözümü
        if (NektarMiktarı(RF) > NektarMiktarı(Ari)) then
            Ari ← RF
            if (NektarMiktarı(Ari) > NektarMiktarı(Best)) then
            Eniyi ← Ari
            SalınımDansıYap(i)
    for Ari ∈ İşçi Arı do
        NF ← komşu yiyecek kaynağı çözümü
        if (NektarMiktarı(NF) > NektarMiktarı(Ari)) then
            if (RasgeleSayı[0,1] > HataOlasılığı) then
                Ari ← NF
                Ari.ziyaretsayısı = 0
                if (NektarMiktarı(Ari) > NektarMiktarı(Best)) then
                Eniyi ← Ari
                SalınımDansıYap(i)
```

Else

Ari.ziyaretsayısı++

if (Ari.ziyaretsayısı > maxziyaretsayısı) **then**

Bi'yi gözcü arı yap.

Rasgele bir Gözcü Arı seçip İşçi Arı yap.

Else

if (RasgeleSayı [0,1] > HataOlasılığı) **then**

Ari.ziyaretsayısı++

if (Ari.ziyaretsayısı > maxziyaretsayısı**) then**

Bi'yi gözcü arı yap.

Rasgele bir Gözcü Arı seçip İşçi Arı yap.

Else

Ari ← NF

Ari.ziyaretsayısı = 0

if (NektarMiktarı(Ari) > NektarMiktarı(Best)) **then**

Best ← Ari

SalınımDansıYap(i)

Until durma kriteri sağlanana kadar -iterasyon süresince ya da en iyi sonucu bulana dek

SalınımDansıYap(i)

for Bj ∈ Gözcü Arı **do**

if (NektarMiktarı(Ari) > NektarMiktarı (Bj) **then**

if (İknaOlasılığı > RasgeleSayı [0,1])

Bj ← Ari

6.Kullanılacak Teknolojiler

6.1 MATLAB

Matlab, teknik hesaplamalar ve matematiksel problemlerin çözümü ve analizi için tasarlanmış bir yazılım geliştirme aracıdır. “MATrix LABoratoty” kelimesinin kısaltması olan MATLAB, adında da anlaşılacağı üzere matrisler yani diğer bir deyişle diziler ile çalışır. Özellikle mühendislik alanındaki sistemlerin analizinde kullanılan MATLAB, görüntü işleme, yapay sinir ağları, sayısal işaret işleme, optimizasyon, veri elde etme, veri tabanı, süzgeç tasarımı, bulanık mantık, sistem kimliklendirme, dalgacıklar gibi araçları sunar.

MATLAB ’in nasıl bir yazılım olduğunu anlamak için onu çok gelişmiş özellikleri olan, programlanabilen bir bilimsel hesap makinesine benzetebiliriz. MATLAB ’de yazılan programlar, MATLAB ‘in kendine özgü dili kullanılarak yazılır ve MATLAB içinden çalıştırılır. Ayrıca yazdığınız programları DLL ve EXE olarak oluşturabildiğiniz gibi C/C++ kodlarına da çevirebilirsiniz.

Problemlerinizi MATLAB ‘de komut satırında çalışan programlar yazarak çözebildiğiniz gibi MATLAB GUI geliştirme aracını kullanarak, formlar ve butonlar gibi nesnelerden oluşan görsel yazılımlar geliştirebilirsiniz.

MATLAB ile;

- Veri elde etme
- Veri analizi ve inceleme
- Görsellik ve görüntü işleme
- Algoritma prototipi oluşturma ve geliştirme
- Modelleme ve simülasyon
- Programlama ve uygulama geliştirme yapabilirsiniz.

6.2 JULİA

Julia yüksek seviyeli, yüksek performanslı ve kullanıcısına verebileceği en iyi performansı bilgisayarı yormadan çalışan bir dil olarak tasarlanmıştır. Hızı ve kullanışlı bir dil olması onu ileride dilin kullanımının yaygınlaşacağını işaret ediyor. MIT tarafından geliştirilen bu dil büyük hesaplamalar ile uğraşan kişiler için biçilmiş kaftan olarak görülüyor. Sürekli güncellenen ve gün geçtikçe yararlı kütüphaneler eklenen Julia içinde çoğu şeyi barındırıyor. Geneli C ve Fortran tabanlı olan bu dil lineer cebir, hızlı fourier dönüşümü gibi birçok işlem için özel kütüphaneler barındırıyor. Yaptığı ve yapacağı işlemleri yüksek doğruluk oranı ve hızı sayesinde kullanıcıları bu yöne çekmeyi iyi başarıyor.

Julia’yı hız konusunda bir hayli ileri geçiren etken JIT(just-in-time) derleme şeklini kullanmasıdır. Yine Julia içerisinde C, Fortran ve Python kodlarını kolayca çağırabiliyoruz. Bu

özellik sayesinde Julia'da yapılamayan işleri bu dillerde yazılmış kütüphaneler aracılığıyla çözebiliyoruz.

6.3 SCİLAB

Oldukça yetenekli bir program olan Scilab mühendislik ve bilimsel uygulamalar için sayısal hesaplamalarda güçlü açık kaynaklı bir ücretsiz yazılım olması ile de dikkat çekiyor. Scilab 1994 yılında ortaya çıkarılmıştır. Zamanlar endüstriyel alanlarda kullanılmaktaydı daha sonra Mayıs 2003 yılında piyasaya sürüldü. Matlab 'a olan benzerliği ile Matlab gibi ücretli programlar kadar yüksek seviye işler yapmayı ve uygulamayı sağlayabiliyor. Lineer cebir, matrisler ve polinomlar üzerinde rahatlıkla işlemler yapılabiliyor. Diferansiyel denklemler üzerinde 2B ve 3B görsel simülasyonlar hazırlanabiliyor. Paralel sanal makinalar kullanılarak Scilab hesaplamalarını ağ üzerindeki bilgisayarlara paylaşabiliyor.

7.Yapılacak Projenin Açıklanması

Matlab, Julia, Scilab dillerinde Gezgin satıcı problemini sezgisel yöntem olan Yapay Arı kolonisi algoritması ile çözüp, her ikisinde de en iyi (best), en uygun (optimal), en kötü (worst) çalışma değerlerini on tekrar sonucunda tabloya dökmek ve raporlamak, raporlar sonucunda elde edilen verilerin kıyaslanması ile yeni bir rapor haline dönüştürerek sunmaktır.

8.Yapılan Algoritmanın Basit Açıklaması

Yapılan Gezgin Satıcı probleminin sezgisel yöntem olan Yapay arı ile en iyi çözümü üretilmeye çalışıldı. Algoritmada ülkemizin seksen bir ilini gezen ve en iyi değeri iterasyonlarla üretilmeye çalışıldı. Algoritmada veriler matrisler halinde işlem gördü. İl mesafeleri Karayolları Genel Müdürlüğünün son yayınladığı mesafeler göz önüne alındı. Bu mesafeler Excel şeklinde Matlab koduna yerleştirildi ve okutuldu. Aynı şekilde julia da aynı işlem yapıldı. Aramayı geliştirmek için çeşitli arama yöntemleri fonksiyonel olarak geliştirildi ve algoritmada kullanıldı. Fonksiyonların ne işe yaradıkları ve nerede kullanıldıkları komut satırlarında tek tek işlenmiştir. Algoritmanın Julia ve Matlab Performansları iterasyonlar sonucun en iyi , en kötü , en uygun olarak çalışma zamanları ile birlikte incelemesi yapıldı. Sonuçlar ve tüm program açıklamaları aşağıda fotoğraflar ve tablolar halinde verilmiştir.

9.Problemin MATLAB' da Gerçekleştirilmesi

- Besinmatrisi.m

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

%Şehir sayısı kadar besin oluşturudu.Alt ve üst limitleri belirtelen sayılar kadar
```

- Amacfonksiyonu.m

```
function uzaklik = amacfonksiyonu(besin,ilmesafe,sehirsayisi)
ustlimit =(sehirsayisi-1);
uzaklik = 0;
for i=1:ustlimit
    yeniuzaklik=ilmesafe(besin(i),besin(i+1));
    %üretmiş olduğumuz rastgele değerden 1.sinde başlayı i değeri artıcak
    uzaklik=uzaklik+yeniuzaklik;
end
%ilk şehirden son şehire kadar gidicek .burda da amaç fonctionum 2 ye
%ayrılcak
%son şehirden ilk şehire dönme olucak ayrılan kısım
yeniuzaklik=ilmesafe(besin(i+1),besin(1));
uzaklik=uzaklik+yeniuzaklik;
```

- Yerdegistir.m

```
%minimum mesafeyi kat eden yolu bulmaya çalışıcaz yer değiştirme op. belirleme
function yenibesin=yerdegistir(besin,indis1,indis2)
yenibesin = besin;
degisgen=yenibesin(indis1);
yenibesin(indis1)=yenibesin(indis2);
yenibesin(indis2) = degisgen;
end
```

- Solakaydir.m

```
%sola kaydırma
function yenibesin=solakaydir(besin,indis1,indis2)
% çözümde iğleştirme olmassa eski olan bozulmasın die bu kısım
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;

%burda da min mesafeyi kateden yolu bulmaya çalışacağız bu da bir arama
%yönetmi
```

- Diziyiterscevir.m

```
function yenibesin=diziyiterscevir(besin,indis1,indis2)
yenibesin=besin;
kucuk=min(indis1,indis2);
buyuk=max(indis1,indis2);
while kucuk<buyuk
    degisgen=yenibesin(kucuk);
    yenibesin(kucuk)=yenibesin(buyuk);
    yenibesin(buyuk)=degisgen;
    kucuk = kucuk+1;
    buyuk = buyuk-1;
end

%burda da min mesafeyi kateden yolu bulmaya çalışacağız bu da bir arama
%yönetmi.Eğer algoritma gelişmez ise eski çözümüde aklında tutuyor
```

- 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
%rulet tekerleği seçim yöntemi ile besin seçilmesi
```

- YerelArama1.m

```
%2 tane yerel arama yapıcam bunlardan 1.si burda yerelaramal olucak
%bunlar kaşif arilar açin aramadır
function [sonucbesin,sonuccozumdegeri]=YerelAramal(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
% elde ettiğimiz çözüm değeri diğerinden daha iyi ise değişicek. eğer
% değilse aynı kalıcak
%SOLAKAYDİRMA KULLANDIM bu aramada
```

- YerelArama2.m

```
%yerelarama2 yi burda yazıcam
%kaşif arının araması için
function [sonucbesin,sonuccozumdegeri]=YerelArama2(besin,cozumdegeri,ilmesafe,sehirsayisi)
    %ilgili çözüm bozulmasın die burda hafızaya alıyorum
    sonucbesin=besin; %ilgili çözüm bozulmasın die burda hafızaya alıyorum
    sonuccozumdegeri=cozumdegeri;
    for i=1:length(sonucbesin)-1
    for j=1:length(sonucbesin)
        yenibesin=diziyiterscevir(sonucbesin,i,j);
        yeniamacfonksiyonu=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
        if yeniamacfonksiyonu<sonuccozumdegeri
            sonuccozumdegeri=yeniamacfonksiyonu;
            sonucbesin=yenibesin;
        end
    end
    end

    %DİZİYİ TERSÇEVİRME KULLANDIM bu aramada
```

- Algoritma.m

```
clear all;
clc;
tic
sehirsayisi = 81; % Gezilecek olan şehir sayisi
isciarisayisi=15;
gozcuarisayisi=15;
%kaşif arisayisini 1 olarak kabul edicem onu girmicem
besinsayisi=50;
limit = 100;  %limitimiz
ilmesafe=xlsread('ilmesafe.xls');
% eniyicozumdegeri = 1000000;
eniyicozumdegeri = Inf;
besin=besinmatrisi(besinsayisi,sehirsayisi);
denemesayisi=zeros(size(besin(:,1)));
%sonsuz döngüye girmemesi için denemesayisi yaptık
for i=1:besinsayisi
    cozumdegeri(i)=amacfonksiyonu(besin(i,:),ilmesafe,sehirsayisi);
    %herbir çözüm satırının çözüm degerini elde edicez ve karşılaştırmyı
    %sağlıyacağız
    if eniyicozumdegeri>cozumdegeri(i)
        eniyicozumdegeri=cozumdegeri(i);
        eniyicozum=besin(i,:);
    end
end
for iterasyon=1:1000

%işçi arı safhası başlar
for i=1:isciarisayisi
    degisecekbesinno= randi([1 besinsayisi]); % rastgele bir satırı seçtik
    r=rand(1);
    indis1=randi([1 sehirsayisi]);
    indis2=randi([1 sehirsayisi]);
    while indis1 == indis2
        indis2=randi([1 sehirsayisi]);
    end
```

```
    %Besin oluşurken kullanılan fonksiyonlarla çözümü iyileştirmek
    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


% eğer daha iyi bir çözüm kalitesi elde edersek bunu hafızaya alalım
yenicozumdegeri=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
if yenicozumdegeri<cozumdegeri(degisecekbesinno)
    cozumdegeri(degisecekbesinno)=yenicozumdegeri;
    besin(degisecekbesinno,:)=yenibesin; % matrise atma da yapalım
    denemesayisi(degisecekbesinno)=0;
    if eniyicozumdegeri>yenicozumdegeri
        eniyicozumdegeri=yenicozumdegeri;
        eniyicozum=yenibesin;
    end
else
    denemesayisi(degisecekbesinno) = denemesayisi(degisecekbesinno)+1;
end
end
```

```
%işçi ari safhası bitti.Şimdide gözcü arı safhasına geçtim

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
```

--Gözcü arılar yeni besinlerin daha iyi olanlarını seçiyor.

```
% indis1 ile indis2 değerinin birbirine eşit olmasını istemiyorum. bu
% yüzden rastgele bir degeer daha oluşucak
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
```

```
%gözcü arı safhası bitti
%kaşif arı safhası başlar
for i=1:besinsayisi % ilgili besin satırında tek tek bakıcak ve rastgele besin satırını seçicek
    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
%kaşif arı safhası bitti
fprintf('iterasyon : %d En iyi çözüm: %d \n',iterasyon,eniyicozumdegeri);
```

```
%figüre başlangıcı
for i=1:sehirsayisi
   x(i) = [i];
   y(i)= [eniyicozum(i)];
%fprintf('deger: %d Cozum : %d\n',i,eniyicozum(i));


end
figure

plot(x,y,'r-')
hold on
plot(x,y,'s','MarkerSize',5)
grid
toc
 %figure bitişi
```

Burada algoritmam sona geliyor. Figürü çizdirip bitiriyor ve sonucu gösteriyor.

ÖNEMLİ NOT

Programın çalışmasından sonra ortaya çıkan figür çok büyük boyutta olduğu için dosyaya ek olarak JPG formatında verilmiştir.

10.Problemin JULİA' da Gerçekleştirilmesi

- Besinmatrisi.jl

```
function besinmatrisi(besinsayisi,sehirsayisi)
ustlimit = sehirsayisi;
matris =zeros(Int,besinsayisi,sehirsayisi);
for i = 1:besinsayisi
  matris[i,:]=randperm(sehirsayisi);
end
return matris
end
```

Random şeklinde besin matrisi oluşturuldu. Bu matris şehir sayısı kadar oluşacak

- Amacfonksiyonu.jl

```
function amacfonksiyonu(besin,ilmesafe,sehirsayisi)
ustlimit =(sehirsayisi-1);
uzaklik = 0;
y = besin[:];

for i=1:ustlimit
  all = y[i];

  all2 = y[(i+1)];
    yeniuzaklik=ilmesafe[all,all2];
    #üretmiş olduğumuz rastgele değerden 1.sinde başlayı i değeri artıcak
    uzaklik=uzaklik+yeniuzaklik;
end
#ilk şehirden son şehire kadar gidicek .burda da amaç fonctionum 2 ye ayrılacak
#son şehirden ilk şehire dönme olucak ayrılan kısım

return uzaklik
end
```

- Yerdegistir.m

```
#min mesafeyi kat eden yolu bulmaya çalışıcaz yer değiştirme op. belirle
function yerdegistir(besin,indis1,indis2)
yenibesin = besin;
degisgen=yenibesin[indis1];
yenibesin[indis1]=yenibesin[indis2];
yenibesin[indis2] = degisgen;
return yenibesin
end
```

- Solakaydir.jl

```
#sola kaydırma
function solakaydir(besin,indis1,indis2)
# çözümde iğleştirme olmassa eski olan bozulmasın die
yenibesin=besin;
kucuk=min.(indis1,indis2);
buyuk=max.(indis1,indis2);
ilkeleman=yenibesin[kucuk];
for i=kucuk[1]:buyuk[1]-1
    yenibesin[i] = yenibesin[i+1];
end
yenibesin[buyuk]=ilkeleman;
return yenibesin
end
```

- Diziyiterscevir.jl

```
function diziyiterscevir(besin,indis1,indis2)
yenibesin=besin;
kucuk=min.(indis1,indis2);
buyuk=max.(indis1,indis2);
while kucuk[1] < buyuk[1]
    degisgen=yenibesin[kucuk];
    yenibesin[kucuk]=yenibesin[buyuk];
    yenibesin[buyuk]=degisgen;
    kucuk = kucuk+1;
    buyuk = buyuk-1;
end
return yenibesin
end
```

- Besinsec.jl

```
function besinsec(uygunluk)
toplam=0.0;
r=rand();
bs = 0;
for i=1:length(uygunluk)
    toplam=toplam+uygunluk[i];
    if r<toplam
        bs=i;
        break;
    end
end
#rulet tekerleği seçim yöntemi ile besin seçilmesi
return bs
end
```

- YerelArama1.jl

```
#2 tane yerel arama yapıcam bunlardan 1.si burda yerelarama1 olucak
#bunlar kaşif arılar açın aramadır
function 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
#elde ettiğimiz çözüm değeri diğerinden daha iyi ise değişicek. eğer
#değilse aynı kalıcak
#SOLAKAYDİRMA KULLANDIM bu aramada
return [sonucbesin,sonuccozumdegeri]
end
```

- YerelArama2.jl

```
#yerelarama2 yi burda yazıcam
#kaşif arının araması için
function YerelArama2(besin,cozumdegeri,ilmesafe,sehirsayisi)
    #ilgili çözüm bozulmasın die burda hafızaya alıyorum
    sonucbesin=besin; #ilgili çözüm bozulmasın die burda hafızaya alıyorum
    sonuccozumdegeri=cozumdegeri;
    for i=1:length(sonucbesin)-1
    for j=1:length(sonucbesin)
        yenibesin=diziyiterscevir(sonucbesin,i,j);
        yeniamacfonksiyonu=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
        if yeniamacfonksiyonu<sonuccozumdegeri
            sonuccozumdegeri=yeniamacfonksiyonu;
            sonucbesin=yenibesin;
        end
    end
    end
    return [sonucbesin,sonuccozumdegeri]
  end
    #DİZİYİ TERSÇEVİRME KULLANDIM bu aramada
```

- Kaşif arılarımın her zaman daha iyi yolu bulması için yapılan arama çeşidi.

- Algoritma.jl

```
tic()
#Fonksiyonların pathleri gösterildi
path = "C:/Users/sadik/Desktop/yapayarijulia"
path2 = "C:/Users/sadik/Desktop/yapayarijulia/data"
#Fonksiyonlar burada okutuldu .içindekileri kullanabilmek için
include(joinpath(path2, "ilmesafe.jl"));
include(joinpath(path, "icindemevcutmu.jl"));
include(joinpath(path, "rakamlarifarklidizi.jl"));
include(joinpath(path, "besinmatrisi.jl"));
include(joinpath(path, "amacfonksiyonu.jl"));
include(joinpath(path, "yerdegistir.jl"));
include(joinpath(path, "solakaydir.jl"));
include(joinpath(path, "diziyiterscevir.jl"));
include(joinpath(path, "besinsec.jl"));
include(joinpath(path, "yerelarama1.jl"));
include(joinpath(path, "yerelarama2.jl"));

ilmesafe = deepcopy(ilmes);
sehirsayisi = 81;
isciarisayisi = 15;
gozcuarisayisi = 15;
#kaşif arisayisini 1 olarak kabul edicem onu girmicem
besinsayisi = 50;
limit = 100;
```

Buradaki ‘inculude’ ler fonksiyonlarımın yolunu belirtiyor. Onları kullanabilmemi sağlıyor.

```
eniyicozumdegeri = Inf;
besin = besinmatrisi(besinsayisi, sehirsayisi);
denemesayisi = zeros(Int, size(besin[:, 1]));
cozumdegeri = zeros(Float64, besinsayisi);
#cozumdegeri = zeros(besinsayisi + 1);
#sonsuz döngüye girmemesi için denemesayisi yaptık
for i = 1:besinsayisi
  cozumdegeri[i] = amacfonksiyonu(besin[i, :], ilmesafe, sehirsayisi);
  #herbir çözüm satırının çözüm degerini elde edicez ve karşılaştırmyı
  #sağlıyacağız
  if eniyicozumdegeri > cozumdegeri[i]
    eniyicozumdegeri = cozumdegeri[i];
    eniyicozum = besin[i, :];
  end
end
```

```
yenibesin = zeros(sehirsayisi);
degisecekbesinno = 0;
for iterasyon = 1:1000
  #işçi arı safhası başlar
  for i = 1:isciarisayisi
    degisecekbesinno = rand(1:besinsayisi, 1); # rastgele bir satırı seçtik
    r = rand();
    indis1 = rand(1:sehirsayisi, 1);
    indis2 = rand(1:sehirsayisi, 1);
    while indis1 == indis2
      indis2 = rand(1:sehirsayisi, 1);
    end
    #En iyi çözüm için dizi yi  yenileme operatörleri uygulandı
    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
  end
```

```
if any(eniyicozumdegeri.<cozumdegeri[degisecekbesinno])
  cozumdegeri[degisecekbesinno] = yenicozumdegeri;
  besin[degisecekbesinno, :] = yenibesin; # matrise atma da yapalım
  denemesayisi[degisecekbesinno] = 0;
  if eniyicozumdegeri > yenicozumdegeri
    eniyicozumdegeri = yenicozumdegeri;
    eniyicozum = yenibesin;
  end
else
  denemesayisi[degisecekbesinno] = denemesayisi[degisecekbesinno] + 1;
end
```

Yukarıdaki fotoğrafta en iyi çözüm değerini iyileştirebilmek için çözümleri yeni sonuç değerinde tutuyor. Yeni gelen sonuç en iyi mi diye kontrol ediliyor. Bir öncekinden daha iyi ise bu çözüm en iyi çözümüm oluyor.

```
#işçi ari safhası bitti şimdide gözcü arı safhasına geçtim

sabit = 1;
minicinuygunluk = zeros(Float64, besinsayisi);
uygunluk = zeros(Float64, besinsayisi);
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();
  indis1 = rand(1:sehirsayisi, 1);
  indis2 = rand(1:sehirsayisi, 1);
  while indis1 == indis2
    indis2 = rand(1:sehirsayisi, 1);
  end
```

```
# indis1 ile indis2 değerinin birbirine eşit olmasını istemiyorum. bu
# yüzden rastgele bir degeer daha oluşucak
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
```

```
  #gözcü arı safhası bitti
  #kaşif arı safhası başlar
  for i = 1:besinsayisi # ilgili besin satırında tek tek bakıcak ve rastgele besin satırını seçicek
    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
  #kaşif arı safhası bitti
  @printf("iterasyon : %d En iyi çözüm: %d \n",iterasyon,eniyicozumdegeri);

end


toc()
```

Bu son kısımda kaşif arılar tek tek tüm besinlere bakıyorlar ve rastgele bir satırı seçiyorlar. Satırlar seçildikten sonra algoritmada işlemleri görüyor ve bu sonuç en iyisi ise hafızaya alınıyor. Eğer değil ise hafızada ki kısımla karşılaştırıp eleme yapıyor . Bunların tamamı iterasyonlar sonucunda belli ediliyor ve en iyi çözüme ulaşılmaya çalışılıyor.

NOT: iterasyon sayısını artırmak daha iyi sonuçlar üretmeye yardımcı olacaktır.

10.Problemin SCİLAB' da Gerçekleştirilmesi

- Besinmatrisi.sce

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

- YerelArama1.sce

```
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
endfunction;
```

- YerelArama2.sce

```
function [sonucbesin,sonuccozumdegeri]=YerelArama2(besin,cozumdegeri,ilmesafe,sehirsayisi)

    sonucbesin=besin;
    sonuccozumdegeri=cozumdegeri;
    for i=1:length(sonucbesin)-1
    for j=1:length(sonucbesin)
        yenibesin=diziyiterscevir(sonucbesin,i,j);
        yeniamacfonksiyonu=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
        if yeniamacfonksiyonu<sonuccozumdegeri
            sonuccozumdegeri=yeniamacfonksiyonu;
            sonucbesin=yenibesin;
        end
    end
    end
    endfunction;
```

- Solakaydir.sce

```
function yenibesin=solakaydir(besin,indis1,indis2)
//çözümde iğleştirme olmassa eski olan bozulmasın die bu kısım
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;
endfunction;
```

- İcindemevcutmu.sce

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

- Besinlerin aynı olmaması için yapılan fonksiyon

- Rakamlarifarklidizi.sce

```
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
endfunction;
```

-Random sayılar üreterek iş yapan fonksiyon. Buradaki çıkan random sayılar besin oluşmasını sağlamaktadır.

- Diziyiterscevir.sce

```
function yenibesin=diziyiterscevir(besin,indis1,indis2)
yenibesin=besin;
kucuk=min(indis1,indis2);
buyuk=max(indis1,indis2);
while kucuk<buyuk
    degisgen=yenibesin(kucuk);
    yenibesin(kucuk)=yenibesin(buyuk);
    yenibesin(buyuk)=degisgen;
    kucuk = kucuk+1;
    buyuk = buyuk-1;
end
endfunction;
```

- İyi bir arama yapılabilmesi için yapılmıştır. Farklı arama türleri ile daha iyi sonuçlar elde edilmesi hedeflenmiştir.

- Besinsec.sce

```
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
endfunction;
```

- Yenibesin.sce

```
function yenibesin=yerdegistir(besin,indis1,indis2)
yenibesin = besin;
degisgen=yenibesin(indis1);
yenibesin(indis1)=yenibesin(indis2);
yenibesin(indis2) = degisgen;
endfunction;
```

- Tüm besinlerin hedeflenmesinde yeni oluşan besinlerin işleme girmesini sağlar.

- Yerdegistir.sce

```
function yenibesin=yerdegistir(besin,indis1,indis2)
yenibesin = besin;
degisgen=yenibesin(indis1);
yenibesin(indis1)=yenibesin(indis2);
yenibesin(indis2) = degisgen;
endfunction;
```

- Amacfonksiyonu.sce

```
function [uzaklik] = amacfonksiyonu(besin,ilmesafe,sehirsayisi)
ustlimit =(sehirsayisi-1);
uzaklik = 0;
for i = 1:ustlimit

    yeniuzaklik=ilmesafe(besin(i),besin(i+1));

    //üretmiş olduğumuz rastgele değerden 1.sinde başlayı i değeri artıcak
    uzaklik = uzaklik + yeniuzaklik;

end;
//ilk şehirden son şehire kadar gidicek .burda da amaç fonctionum 2 ye
//ayrılcak
//son şehirden ilk şehire dönme olucak ayrılan kısım
yeniuzaklik=ilmesafe(besin(i+1),besin(1));
uzaklik=uzaklik+yeniuzaklik;
endfunction;
```

- İlmesafe.sce

```
function ilmesafe1 = deger()
    ds = readxls('ilmesafe.xls')
    ds2 = ds(2)
    ilmesafe1 = ds2(2 : 82 , 4 : 84)
endfunction
veri = deger()
disp(veri)
```

- ilmesafe.xls dosyasından verileri çekip matrix haline dönüştürülmüştür.

- Algoritma.sce

```
sehirsayisi = 81; // Gezilecek olan şehir sayisi
isciarisayisi=15;
gozcuarisayisi=15;
//kaşif arisayisini 1 olarak kabul edicem onu girmicem
besinsayisi=50;
limit = 100; //limitimiz
ilmesafe= deger();
//ilmesafe=readxls('C:/Users/sadik/Desktop/scilab/ilmesafe.xls')
eniyicozumdegeri = 1000;
eniyicozumdegeri = %inf;
x=besinmatrisi(besinsayisi,sehirsayisi);
besin = matrix(x, [besinsayisi sehirsayisi]);
a = size(besin);
at=a(1,1);
denemesayisi=zeros(at,1);
//sonsuz döngüye girmemesi için denemesayisi yaptık
for i=1:besinsayisi
    cozumdegeri(i)=amacfonksiyonu(besin(i,:),ilmesafe,sehirsayisi);
    //herbir çözüm satırının çözüm degerini elde edicez ve karşılaştırmyı
    //sağlıyacağız
    if eniyicozumdegeri>cozumdegeri(i)
        eniyicozumdegeri=cozumdegeri(i);
        eniyicozum=besin(i,:);
    end
end
```

```
for iterasyon=1:1000

//işçi arı safhası başlar
for i=1:isciarisayisi
    degisecekbesinno= grand(1,1,"uin",1,besinsayisi); // rastgele bir satırı seçtik
    r=rand(1);
    indis1=grand(1,1,"uin",1,sehirsayisi);
    indis2=grand(1,1,"uin",1,sehirsayisi);
    while indis1 == indis2
        indis2=grand(1,1,"uin",1,sehirsayisi);
    end
    //Besin oluşurken kullanılan fonksiyonlarla çözümü iyileştirmek
    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
```

```
//eğer daha iyi bir çözüm kalitesi elde edersek bunu hafızaya alalım
yenicozumdegeri=amacfonksiyonu(yenibesin,ilmesafe,sehirsayisi);
if yenicozumdegeri<cozumdegeri(degisecekbesinno)
    cozumdegeri(degisecekbesinno)=yenicozumdegeri;
    besin(degisecekbesinno,:)=yenibesin; //matrise atma da yapalım
    denemesayisi(degisecekbesinno)=0;
    if eniyicozumdegeri>yenicozumdegeri
        eniyicozumdegeri=yenicozumdegeri;
        eniyicozum=yenibesin;
    end
else
    denemesayisi(degisecekbesinno) = denemesayisi(degisecekbesinno)+1;
end
end

//işçi arı safhası bitti.Şimdide gözcü arı safhasına geçtim

sabit = 1;
for i=1:besinsayisi
    minicinuygunluk(i)=sabit/cozumdegeri(i);
end
cozumdegerleritoplami = 0 ;
for i=1:besinsayisi
    cozumdegerleritoplami=cozumdegerleritoplami+minicinuygunluk(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=grand(1,1,"uin",1,sehirsayisi);
    indis2=grand(1,1,"uin",1,sehirsayisi);
    while indis1 == indis2
        indis2=grand(1,1,"uin",1,sehirsayisi);
    end
    //indis1 ile indis2 değerinin birbirine eşit olmasını istemiyorum. bu
    //yüzden rastgele bir degeer daha oluşucak
    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
    //gözcü ari safhası bitti
    //kaşif arı safhası başlar
    for i=1:besinsayisi // ilgili besin satırında tek tek bakıcak ve rastgele besin satırın
ı seçicek
        if denemesayisi(i)>limit
            denemesayisi(i)=0;
            [besin(i,:),cozumdegeri(i)]=YerelAramal(besin(i,:),cozumdegeri(i),ilmesafe,sehi
rsayisi);
            if eniyicozumdegeri>cozumdegeri(i)
                eniyicozumdegeri=cozumdegeri(i);
                eniyicozum=besin(i,:);
```

```
    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
    //gözcü ari safhası bitti
    //kaşif arı safhası başlar
    for i=1:besinsayisi // ilgili besin satırında tek tek bakıcak ve rastgele besin satırını seçicek
        if denemesayisi(i)>limit
            denemesayisi(i)=0;
            [besin(i,:),cozumdegeri(i)]=YerelAramal(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
    //kaşif arı safhası bitti
    printf('iterasyon : %d En iyi çözüm: %d \n',iterasyon,eniyicozumdegeri);
```

11.SONUÇ TABLOLARI

100 iterasyon için sonuçlar					
Deneme Sayısı	**Dil**	**En iyi(km)**	**En köyü(km)**	**En uygun(km)**	**Zaman(sn)**
1. Deneme Sayısı	**Scilab**	38899	50307	9920(km)	2.7549
	Julia	47319	53958	9920(km)	3.1311
	Matlab	44721	54340	9920(km)	5.4879
2. Deneme Sayısı	**Scilab**	44004	52954	9920(km)	2.9003
	Julia	49828	54736	9920(km)	0.5736
	Matlab	42463	50642	9920(km)	1.2560
3. Deneme Sayısı	**Scilab**	43367	52902	9920(km)	3.0415
	Julia	46920	54613	9920(km)	0.5640
	Matlab	42644	53371	9920(km)	1.2799
4. Deneme Sayısı	**Scilab**	41505	52418	9920(km)	2.8252
	Julia	45580	50658	9920(km)	0.5935
	Matlab	42506	54279	9920(km)	1.4106
5. Deneme Sayısı	**Scilab**	41918	51947	9920(km)	2.9246
	Julia	43158	53498	9920(km)	0.5757
	Matlab	42585	53450	9920(km)	1.4024
6. Deneme Sayısı	**Scilab**	36838	50876	9920(km)	2.9623
	Julia	47473	55092	9920(km)	0.6199
	Matlab	44389	53530	9920(km)	1.3542
7. Deneme Sayısı	**Scilab**	43355	54085	9920(km)	3.1491
	Julia	44891	54224	9920(km)	0.5991
	Matlab	43001	53646	9920(km)	1.3103
8. Deneme Sayısı	**Scilab**	40959	51840	9920(km)	2.6848
	Julia	41717	55567	9920(km)	0.5925
	Matlab	43494	53259	9920(km)	1.4091
9. Deneme Sayısı	**Scilab**	40883	53681	9920(km)	2.8043
	Julia	35199	43294	9920(km)	0.5987
	Matlab	44945	56833	9920(km)	1.2624

10. Deneme Sayısı	**Scilab**	39980	51071	9920(km)	3.0579
	Julia	41698	50498	9920(km)	0.5821
	Matlab	40672	50110	9920(km)	1.3585

- 100 iterasyon sonucu

Scilab , Julia ve Matlab ' da 100 iterasyonluk 10 ' ar çalıştırma yapılmıştır. Yapılan incelemede Scilab'ın diğerlerine göre azda miktarda da olsa daha iyi çözüm sonuçları elde ettiği gözlenmiştir. Anca Scilab diğerlerine göre çalışma hızı bakımından neredeyse yarı yarıya yavaş bir zamanda problemi çözmüştür. En hızlı ise Julia olarak görülmüştür. Buradaki En Uygun(optimal) sonuç bu güne kadar literatürde bulunmuş en iyi sonuçtur.

500 iterasyon için sonuçlar

Deneme Sayısı	Dil	En iyi(km)	En köyü(km)	En uygun(km)	Zaman(sn)
1. Deneme Sayısı	**Scilab**	27475	53728	9920(km)	9.3218
	Julia	36671	53397	9920(km)	0.7236
	Matlab	28775	50642	9920(km)	1.6197
2. Deneme Sayısı	**Scilab**	27415	49930	9920(km)	8.0498
	Julia	32795	53658	9920(km)	0.7578
	Matlab	30063	53159	9920(km)	1.6843
3. Deneme Sayısı	**Scilab**	26811	53565	9920(km)	7.8844
	Julia	35919	53267	9920(km)	0.7677
	Matlab	30688	53646	9920(km)	1.7021
4. Deneme Sayısı	**Scilab**	26032	53252	9920(km)	8.0387
	Julia	32518	53281	9920(km)	0.7504
	Matlab	28171	55521	9920(km)	1.6903
5. Deneme Sayısı	**Scilab**	26642	49887	9920(km)	8.2750
	Julia	36126	54185	9920(km)	0.7251
	Matlab	28246	52835	9920(km)	1.6688

6. Deneme Sayısı	**Scilab**	27813	53258	9920(km)	7.7400
	Julia	36206	55097	9920(km)	0.7424
	Matlab	30082	54521	9920(km)	1.7040
7. Deneme Sayısı	**Scilab**	26940	54042	9920(km)	7.8510
	Julia	33340	53881	9920(km)	0.7532
	Matlab	30200	56154	9920(km)	1.6952
8. Deneme Sayısı	**Scilab**	29022	54183	9920(km)	8.0547
	Julia	34052	55031	9920(km)	0.7493
	Matlab	29852	55777	9920(km)	1.6486
9. Deneme Sayısı	**Scilab**	26663	51699	9920(km)	8.2823
	Julia	33942	53632	9920(km)	0.7601
	Matlab	30241	52674	9920(km)	1.6606
10. Deneme Sayısı	**Scilab**	29318	52702	9920(km)	8.1495
	Julia	36166	53125	9920(km)	0.7535
	Matlab	28091	53884	9920(km)	1.6930

- 500 iterasyon sonucu

Scilab, Julia ve Matlab ' da 500 iterasyonluk 10 ' ar çalıştırma yapılmıştır. Çalıştırmalar sonucunda en iyi problemin çözümleri Matlab tarafından gerçekleştirilmiştir. Julia çözüm kalitesi olarak diğerlerinden kötü olsa bile çalıştırma hızı olarak en iyi Julia olduğu görülmektedir. Her biri optimal sonucun çok uzağında kalmıştır.

1.000 iterasyon için sonuçlar					
Deneme Sayısı	**Dil**	**En iyi(km)**	**En köyü(km)**	**En uygun(km)**	**Zaman(sn)**
1. Deneme Sayısı	**Scilab**	21766	53352	9920(km)	14.2511
	Julia	30880	53291	9920(km)	2.9513
	Matlab	24465	56526	9920(km)	2.1122
2. Deneme Sayısı	**Scilab**	22811	49622	9920(km)	14.6232
	Julia	30810	53929	9920(km)	0.8201
	Matlab	10983	55681	9920(km)	1.7490

3. Deneme Sayısı	**Scilab**	20246	53449	9920(km)	14.5430
	Julia	28880	54245	9920(km)	0.7831
	Matlab	23302	53779	9920(km)	1.6181
4. Deneme Sayısı	**Scilab**	22490	51243	9920(km)	14.6159
	Julia	28804	54377	9920(km)	0.7914
	Matlab	24793	54279	9920(km)	1.6772
5. Deneme Sayısı	**Scilab**	23161	53010	9920(km)	15.5025
	Julia	30327	54296	9920(km)	2.8375
	Matlab	24942	50631	9920(km)	1.6519
6. Deneme Sayısı	**Scilab**	20341	52427	9920(km)	14.2877
	Julia	28748	53746	9920(km)	0.7890
	Matlab	23669	55328	9920(km)	1.6431
7. Deneme Sayısı	**Scilab**	19425	54140	9920(km)	14.524048
	Julia	31271	54232	9920(km)	0.8125
	Matlab	24118	53480	9920(km)	1.6938
8. Deneme Sayısı	**Scilab**	17123	52701	9920(km)	18.46929
	Julia	30275	55399	9920(km)	0.8551
	Matlab	22225	54759	9920(km)	1.5948
9. Deneme Sayısı	**Scilab**	22578	51384	9920(km)	12.401036
	Julia	30523	52858	9920(km)	0.8262
	Matlab	23790	52845	9920(km)	1.6839
10. Deneme Sayısı	**Scilab**	23997	50571	9920(km)	12.4684
	Julia	30318	52844	9920(km)	0.8428
	Matlab	23391	55611	9920(km)	1.6630

- 1000 iterasyon sonucu

Scilab, Julia ve Matlab ' da 1000 iterasyonluk 10 ' ar çalıştırma yapılmıştır. Yapılan çalıştırmalar sonucunda hız olarak en kötü hıza sahip Scilab olduğu belirlenmiştir. Çözüm kalitesi çoğunluklu olarak Scilab değerleri diğerlerine göre çok az bir farkla daha iyi sonuçlar elde etmeyi başarmıştır. En hızlı olan ise Julia olmuştur. Julia her zaman 1 sn. daha kısa sürede problemi

çözüme kavuşturabilmiştir. Ama julia çözüm kalitesi diğerlerinin çok az gerisinde kalmıştır. Çözüm kalitesinde optimal sonuca en yaklaşan Matlab değerleri olmuştur.

12.Kaynakça

1. [1] Karaboğa D., sy. 206,Yapay Zeka Optimizasyon Algoritmaları, Nobel Yayın Dağıtım, Genişletilmiş 2. Basım, 2011.

2. [2]Akay B., Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi Algoritmasının Performans Analizi, sy. 61, 2009

3. [3]D. Karaboğa, An İdea Based On Honey Bee Swarm For Numerical Optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

4. [4]J. McCaffrey, "Natural Algorithms: Use Bee Colony Algorithms to Solve Impossible Problems", MSDN Magazines, erişim linki: http://msdn.microsoft.com/en-us/magazine/gg983491.aspx, 24 Ocak 2014.