



T.C

SAKARYA ÜNİVERSİTESİ

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

BSM 310 – YAPAY ZEKÂ

Grup üyeleri:

B171210002 EMRULLAH KARAKOÇ

B171210098 YUNUS ŞEN

B171210032 AHMETCAN ÖRTEN

G161210302 OKAY TONKA

**Sakarya
2020**



Yapay Arı Kolonisi (ABC) algoritması, enson tanıtılan sürütabanlı algoritmalarından biridir. 2005 yılında Derviş Karaboğata tarafından ortaya atılmıştır. ABC, birbalarısı sürüsünün akıllı yiyecek arama davranışını simüle eder. Bu çalışmada ABC çok sayıda sayısal test fonksiyonunu optimize etmek için kullanılmıştır.



GERÇEK ARILARIN DAVRANIŞLARI

yapılacak işler o iş için özelleşmiş arılar tarafından yapılır.Yani yapılacak işlere göre arılar arasında bir iş bölümü vardır ve kendi kendilerine organize olabilmektedirler. İş bölümü yapabilme ve kendi kendine organize sürü zekâsının iki önemli bileşenidir.

DİĞER POPÜLASYON TABANLI ALGORİTMALARDAN AVANTAJI

ABC'nin performansının, daha az kontrol parametresi kullanma avantajı ile diğer popülasyon tabanlı algoritmalarından daha iyi veya buna benzer olduğunu göstermektedir.

DANSLAR

Kaynağın kovana olan mesafesine göre çeşitli danslar mevcuttur: dairesel dans, kuyruk dansı ve titreme dansı gibi.

Daire Dansı Belirlenen yiyecek kaynağının kovana olan uzaklığı maksimum 50-100 metre civarında olduğundan bu dans yön ve uzaklık bilgisi içermez.

Titreme Dansı Arıların petek üzerinde düzensiz tarzda ve yavaş tempoda bacaklarını titreterek ileri, geri, sağa ve sola hareketleri söz konusudur. Arı zengin bir nektar kaynağı bulunduğunu ancak kovana işlenebileceğinden fazla nektar geldiğini ve bundan dolayı nektar işleme görevine geçmek istediğini belirtmektedir. Bu dansın amacı kovan kapasitesi ve yiyecek getirme aktivitesi arasındaki dengeyi sağlamaktır.

Kuyruk Dansı 100 metreden 10 kilometreye kadar olan geniş bir alan içerisinde bulunan kaynaklarla ilgili bilgi aktarımında kullanılır. Bu dans 8 rakamına benzeyen figürlerin yapıldığı dans çeşididir. Dansı izleyen arıların bir titreşim oluşturması ile dansa son verilir. Dansın her 15 saniyede tekrarlanma sayısı, nektar kaynağının uzaklığı hakkında bilgi vermektedir. Daha az tekrarlanma sayısı daha uzak bölgeleri ifade etmektedir.

Tüm zengin kaynaklarla ilgili bilgiler dans alanında gözcü arılara iletildiğinden, gözcü arılar birkaç dansı izledikten sonra hangisini tercih edeceğine karar verir. Yiyecek arayıcı arıların daha iyi anlaşılabilmesi için Şekil-2'deki verilen modelin incelenmesi faydalı olacaktır.

YAPAYARI KOLONİSİ'NİN ADIMLARI

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 besin kaynaklarının her bir çözüm değerleri amaç fonksiyonuna göre hesaplanır.

Adım 3: Bu adımda döngü başlamaktadır. Maksimum döngü sayısı belirlenerek işçi arılar besin kaynaklarına gönderilir.

İşçi arılar besin kaynağından rastgele bir besine yönelerek işlemeye başlarlar. Arılar besini işler ve kalitesini ölçerler. Elde edilen çözüm değeri(kalite) bir önceki çözüm değerinden daha iyiye bu besin ve besinle ilgili değerler hafızaya alınır ve limit değeri sıfırlanır.

Tersi durumda ise limit değeri 1 artırılır. Limit değeri için belirli bir üst değer belirlemek algoritmanın çalıştırılması esnasında sonsuz döngüye girmeye engel olacaktır.

Adım 4: İşçi arılardan sonra besin kaynaklarına gözcü arılar yönlendirilir.

Besinlerin uygunluk değerine göre bir besin kaynağı seçilir. Gözcü arı gittiği kaynağın kalitesini ölçer ve çözüm değerlerini hesaplar.

Bu değerler içinde en iyi olan önceki çözüm değeriyle kıyaslanır. Önceki çözüm değerinden daha iyi ise bu besin ve besinle ilgili bilgiler hafızaya alınırlar ve limit değeri sıfırlanır. Tersi durumda limit değeri bir arttırılır. Limit değeri üst değeri aşmış olması kontrol edilir.

Adım 5: İşçi arı ve gözcü arı adımlarından sonra kaşif arı aşamasına

geçilir. Kaşif arılar diğer besin kaynaklarından sonra tamamen yeni besin kaynakları belirler. Kaşif arı safhasının esas nedeni algoritmanın yerel minimum ya da maksimumda takılmasına engel olmaktır. Yeni besin kaynaklarının çözüm değerleri hesaplanır ve bir önceki çözümle kıyaslanır. Eğer yeni çözüm değeri, bir öncekinden iyi ise en iyi çözüm olarak hafızada tutulur ve limit değeri sıfırlanır. Tersi durumda limit değeri bir arttırılır. Limit değeri üst değeri aşmış olması kontrol edilir.

Adım 6: Maksimum değerde ki döngü sayısına ulaşana kadar görevli arı, kaşif arı ve gözcü arı adımları tekrar edilir. Durdurma kriteri sağlanınca algoritma sonlandırılır.

Adımları Özetleyecek Olursak:

- Bunlar sırasıyla başlangıç besin kaynaklarının belirlenmesi,
- İşçi arıların besin kaynaklarına gönderilmesi,
- Gözcü arıların uygunluk değerine göre besin kaynağı seçmesi ve En son olarak da tükenen besin kaynaklarının terk edilmesi ve kaşif arıların yenibesin kaynakları üretmeleridir.

1-BAŞLANGIÇ YİYECEK KAYNAĞI BÖLGELERİNİN ÜRETİLMESİ

Arama uzayını yiyecek kaynaklarını içeren kovan çevresi olarak düşünürsek algoritma, arama uzayındaki çözümlere karşılık gelen rastgele yiyecek kaynağı yerleri üretmek çalışmaya başlamaktadır . Rastgele yer üretme süreci her bir parametrelerinin alt ve üst sınırları arasında rastgele değer üretmek gerçekleşir (Eşitlik-1).

$$x_{ij} = x_{\min j} + rand(0,1) * (x_{\max j} - x_{\min j}) \quad (1)$$

Burada $i=1 \dots SN$, $J=1 \dots D$ ve SN yiyecek kaynağı sayısı ve D is optimize edilecek parametre sayısıdır. $x_{\min j}$ parametrelerin alt sınırıdır.

2- İŞÇİ ARILARIN YİYECEK KAYNAĞI BÖLGELERİNE GÖNDERİLMESİ

Daha öncede belirtildiği gibi her bir kaynağın bir görevli (işçi) arısı vardır. Dolayısıyla yiyecek kaynakların sayısı görevli arıların sayısına eşittir. İşçi arı çalıştığı yiyecek kaynağı komşuluğunda yeni bir yiyecek kaynağı belirler ve bunun kalitesini değerlendirir. Yeni kaynak daha iyi ise bu yeni kaynağı hafızasına alır. Yeni kaynağın mevcut kaynak komşuluğunda belirlenmesinin benzetimi Eşitlik-2 tanımlanmaktadır.

$$v_{ij} = x_{ij} + \phi_{ij} (x_{ij} - x_{kj}) \quad (2)$$

x_i ile gösterilen her bir kaynak için bu kaynağın yani çözümünün tek bir parametresi (rastgele seçilen parametresi, j) değiştirilerek x_i komşuluğunda v_i kaynağı bulunur.

Eşitlik 2 de $j, [1, D]$ aralığında rastgele üretilen bir tamsayıdır. Rastgele seçilen j parametresi değiştirilirken, yine rastgele seçilen x_k komşu çözümünün ($k \in \{1, 2, SN\}$) j . parametresi ile mevcut kaynağın j parametresinin farkları alınıp $[-1, 1]$ arasında rastgele değer alan sayısı ile ağırlandırıldıktan sonra mevcut kaynağın j parametresine eklenmektedir.

$$v_{ij} = x_{ij} + \phi_{ij} (x_{ij} - x_{kj}) \quad (2)$$

Eşitlik-2'den de görüldüğü gibi $x_{i,j}$ ve $x_{k,j}$ arasındaki fark azaldıkça yani çözümler birbirine benzedikçe $x_{i,j}$ parametresindeki değişim miktarı da azalacaktır. Böylece bölgesel optimal çözüme yaklaştıkça değişim miktarı da azalacaktır. • Bu işlem sonucunda üretilen $v_{i,j}$ 'nin daha önceden belli olan parametre sınırları aşması durumunda j . parametreye ait olan alt veya üst sınır değerlerine ötelenmektedir (Eşitlik-3).

$$v_{ij} = \begin{cases} x_j^{\min}, & v_{ij} < x_j^{\min} \\ v_{ij}, & x_j^{\min} \leq v_{ij} \leq x_j^{\max} \\ x_j^{\max}, & v_{ij} > x_j^{\max} \end{cases} \quad (3)$$

Sınırlar dâhilinde üretilen v_i parametre vektörü yeni bir kaynağa temsil etmekte ve bunun kalitesi hesaplanarak bir uygunluk değeri atanmaktadır (Eşitlik-4).

$$fitness_i = \begin{cases} 1/(1+f_i) & f_i \geq 0 \\ 1/abs(f_i) & f_i < 0 \end{cases} \quad (4)$$

Burada f_i ve v_i kaynağının yani çözümünün maliyet değeridir. Yeni Bulunan v_i çözümü daha iyi ise görevli arı hafızasından eski kaynağın yerini silerek v_i kaynağının yerini hafızaya alır. Aksi takdirde görevli arı x_i kaynağına gitmeye devam eder ve x_i çözümü geliştirilemediği için x_i kaynağı ile ilgili geliştirememeye sayacı (failure) bir artar, geliştirdiği durumda ise sayaç sıfırlanır.

3. GÖZCÜ ARILARIN SELEKSİYONDA KULLANACAKLARI OLASILIK DEĞERLERİNİN HESAPLANMASI (DANS BENZETİMİ)

Tüm görevli arılar bir çevrimde araştırmalarını tamamladıktan sonra kovana donüp buldukları kaynakların nektar miktarları ile ilgili gözcü arılara bilgi aktarırlar. Bir gözcü arı dans aracılığıyla paylaşılan bilgiden faydalanılarak yiyecek kaynaklarının nektar miktarları ile orantılı bir olasılıkla bir bölge(kaynak) seçer. BuABC'nin altında çoklu etkileşim sergilendiğinin bir örneğidir.Olasılıksal seçme işlemi, algoritmada nektar miktarlarına karşılık gelen uygunluk değerleri uygulanarak yapılmaktadır.

Uygunluk değerine bağlı seçme işlemi rulet tekerliği, sıralamaya dayalı,stokastik ,örnekleme, turnuva yöntemi yada diğer seleksiyon şemalarından herhangi biri ile gerçekleşir. TemelABC algoritmasında bu seleksiyon işlemi rulet tekerliği kullanılarak yapılmıştır. Tekerlekteki her bir dilimin açısı uygunluk değeri toplamına oranı o kaynağın diğer kaynaklara göre nispi seçilme olasılığı olduğunu vermektedir (Eşitlik-5).

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad (5)$$

Burada kaynağın kalitesini SN görevli arı sayısını göstermektedir. Bu olasılık hesaplama işlemine göre bir kaynağın nektar miktarı arttıkça (uygunluk değeri arttıkça) bu kaynak bölgesini seçecek gözcü arı sayısı da artacaktır. Bu özellik ABC' nin pozitif geri besleme özelliğine karşılık gelmektedir.

4. GÖZCÜ ARILARIN YİYECEK KAYNAĞI BÖLGESİ SEÇMELERİ

Algoritma da olasılık değerleri hesaplandıktan sonra be değerler kullanılarak rulet tekerleğine göre secim işleminde her bir kaynak için [0.1] aralığında rastgele sayı üretilen ve pi değeri bu üretilen sayıdan büyükse görevli arılar gibi gözcü arı da Eşitlik-2'yi kullanarak bu kaynak bölgesinde yeni bir çözüm üretir. Yeni çözüm değerlendirilir ve kalitesi hesaplanır.

Sonra yeni çözümle eski çözümün uygunluklarının karşılaştırıldığı en iyi olanın seçildiği açgözlü seleksiyon işlemine tabi tutulur. Yeni çözüm daha iyi ise eski çözüm yerine bu çözüm alınır ve çözüm geliştirememeye sayacı (failure) sıfırlanır. Eski çözümün uygunluğu daha iyi ise bu çözüm muhafaza edilir ve geliştirememeye sayacı (failure) bir artırılır. Bu süreç, tüm gözcü arılar yiyecek kaynağı bölgelerine dağılana kadar devam eder.

5. KAYNAĞI BIRAKMA KRİTERİ: LİMİT VE KÂŞİF ARI ÜRETİM

Bir çevrim sonunda tüm görevli ve gözcü arılar arama süreçlerini tamamladıktan sonra çözüm geliştirememeye sayaçları (failure) kontrol edilir. Bir arının bir kayaktan faydalanıp faydalanmadığı, yani gidip geldiği kaynağın nektarının tükenip tükenmediği çözüm geliştirememeye sayaçları aracılığıyla bilinir. Bir kaynak için çözüm geliştirememeye sayacı belli bir eşik değerinin üzerindeyse, artık bu kaynağın görevli arısının tükenmiş olan o çözümü bırakıp kendisi için başka bir çözüm araması gerekir.

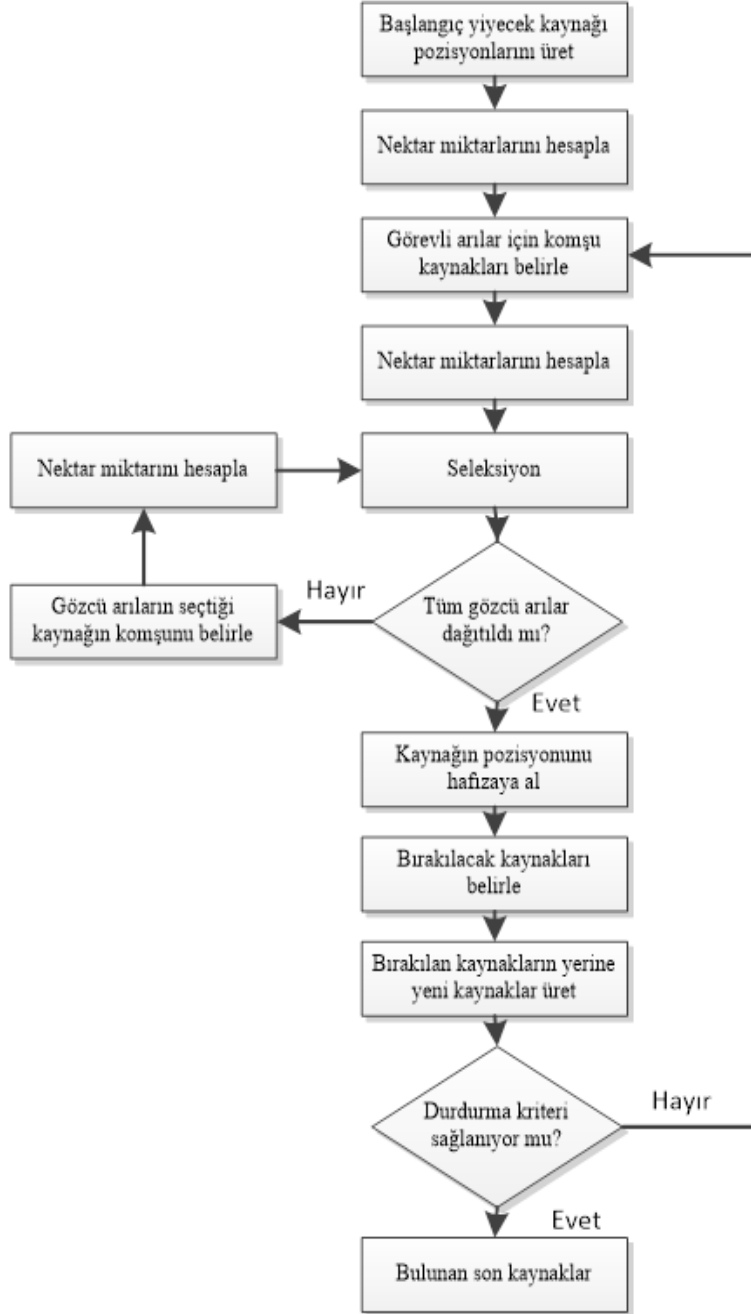
Bu da biten kaynakla ilişkili olan görevli arının kâşif arı olması anlamına gelmektedir. Kâşif arı haline geldikten sonra, bu arı için rastgele çözüm arama süreci başlar (Eşitlik-1). Kaynağın terk ettiğinin belirlenmesi için kullanılan eşik değeri ABC algoritmasının önemli bir kontrol parametresidir ve “limit” olarak adlandırılmaktadır. Temel ABC algoritmasında her çevrimde sadece kâşif arının çıkmasına izin verilir.

6. SELEKSİYON MEKANİZMALARI

ABC algoritması 4 farklı seleksiyon işlemi kullanmaktadır. Bunlar ; • Potansiyel iyi kaynaklarının belirlenmesine yönelik eşitlik 5 olasılık değerlerinin hesaplandığı global olasılık temelli seleksiyon süreci. • Görevli ve gözcü arıların renk, şekil, koku gibi nektar kaynağının türünü belirlenmesi sağlayan görsel bilgiyi kullanarak bir bölgede kaynağın bulunmasına vesile olan bölgesel olasılık tabanlı seleksiyon işlemi (Eşitlik-2).

$$v_{ij} = x_{ij} + \phi_{ij} (x_{ij} - x_{kj}) \quad (2)$$

İşçi ve gözcü arıların daha iyi olan kaynağı belirlemek amacıyla kullandıkları açgözlü seleksiyon. KâşifArılar tarafından eşitlik 1 aracılığıyla gerçekleştirilen rastgele seleksiyon. Bütün bu seleksiyon metotların bir arada kullanılmasıyla ABC algoritması hem iyi bir global araştırma hem de bölgesel araştırma yapabilmektedir.



ÖRNEKLER

Yapay arı kolonisi algoritması ile günümüzde bir çok projeye örnek olmuştur.

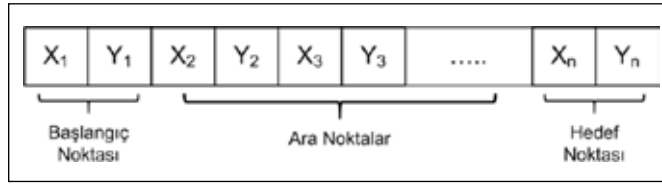
Bunlar:

- İnsansız Hava Araçları İçin Yapay Arı Kolonisi Algoritması Kullanarak Rota Planlama
- Modifiye Yapay Arı Koloni Algoritması ile Nümerik Fonksiyon Optimizasyonu
- Yapay Arı Kolonisi Algoritması İle Elektrik Güç Sistemi Optimal Yakıt Maliyetinin Belirlenmesi

İNSANSIZ HAVA ARAÇLARI İÇİN YAPAY ARI KOLONİSİ ALGORİTMASI ROTA HESAPLAMA

İnsansız Hava Araçları (İHA) için rota planlama, başlangıç noktasından verilen hedef noktaya kadar engellerden ve tehlikeli bölgelerden sakınarak güvenli bir rotanın bulunmasıdır. İHA'ların rota planlama problemine çözüm bulmak için pek çok yöntem kullanılmaktadır. Bu çalışmada rota planlama için Yapay Arı Kolonisi (YAK) algoritması kullanılmıştır. YAK algoritması son yıllarda optimizasyon problemlerinde yaygın bir şekilde kullanılan sezgisel algoritmalarından bir tanesidir. Çalışmada, YAK algoritmasının benzetim uygulamalarını gerçekleştirmek amacı ile C# programlama dili kullanılarak kullanıcı etkileşimli bir arayüz tasarlanmıştır. Rota planlaması için yapılan deneysel çalışmalar, YAK algoritmasının uygun rotalar bulmada başarılı sonuçlar verdiğini göstermektedir.

Yapay Arı Kolonisi (YAK) Algoritması ile Rota Planlama Rota planlama probleminde, İHA'nın başlangıç ve hedef noktaları arasında geçilecek bağlantı koordinat noktalarının tespit edilip, o noktalar üzerinden geçerek hedefe varması hedeflenmektedir. Geçilecek bağlantı noktaları belirlenirken iki temel kriter dikkate alınmıştır. Bu kriterler; rota üzerindeki engellerin dikkate alınması ve rotanın en kısa uzunlukta olacak şekilde belirlenmesi. Rota planlama problemine çözüm üretmek için kullanılan YAK algoritması ile başlangıç aşamasında aday rotalar denklem (1) kullanılarak rastgele üretilmektedir. Başlangıç ve hedef arasındaki mesafeye ve engel sayısına göre değişen haritalarda sabit bağlantı noktaların kullanılması dezavantaj oluşturmaktadır. Bu bakımdan çalışmada, rota üretimi sırasında başlangıç ve hedef koordinatları arasında üzerinden geçilecek olan bağlantı koordinat noktalarının sayısı dinamik olarak belirlenmektedir. Örneğin; algoritma başlangıç olarak iki bağlantı koordinat noktası ile algoritmaya başlamakta, iki adet bağlantı koordinat noktası ile hedefe çözüm mümkün olmadığı durumda aradaki koordinat nokta sayısını birer arttırarak çözüm aramaya devam etmektedir. Her bir bağlantı koordinat noktası için rastgele enlem ve boylam bilgisi üretilmektedir. Algoritmada kullanılan rota yapısı Şekil 1'de gösterilmektedir.



Rastgele rotaların üretiminden sonra her bir rotanın maliyeti (amaç fonksiyon) hesaplanmaktadır. Maliyet hesaplanırken her bir rotanın uzunluğu ve geçiş güzergâhında engellerin olup olmadığı kontrol edilmektedir.

Amaç Fonksiyon Her bir rotanın amaç fonksiyon değeri hesaplanırken, önce rotanın mesafesi hesaplanmakta daha sonra rota üzerinde herhangi bir engel olup olmadığı tespit edilip, var ise amaç fonksiyon değerine ek olarak engel maliyeti de eklenmektedir. GPS verileri üzerinden iki nokta arasındaki uzaklığı hesaplarırken dünyanın geometrik şekli göz önüne alınmıştır ve bunun için Haversine (Montavont ve Noel 2006) formülü kullanılmıştır. Haversine formülü enlemi ve boylamı bilinen iki koordinat arasındaki mesafenin hesaplanması için kullanılmaktadır. Enlemler arasındaki fark Δ enlem, boylamlar arasındaki fark Δ boylam ve R dünyanın yarıçapı ($R=6,371\text{km}$) olarak ifade edilmiştir. İki koordinat noktası arasındaki mesafe d, aşağıdaki denklemler ile hesaplanmaktadır (Montavot ve Noel 2006).

$$h = \text{haversin}(\Delta\text{enlem}) + \cos(\text{enlem1}) \times \cos(\text{enlem2}) \times \text{haversin}(\Delta\text{boylam})$$

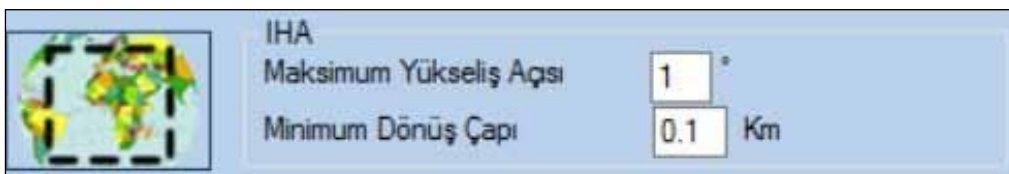
Algoritmada başlangıç aşamasında rastgele rotalar üretildiğinden, rotalar üzerinde engelli bölgelerin de olma ihtimali bulunmaktadır. Öncelikle herhangi bir engele çarpmadan veya engelli bir alana girmeden rota planlaması yapılması gerektiğinden, engelli alanların belirlenmesi gerekmektedir. Rota üzerindeki engellerin tespiti için, denklemi bilinen bir doğrunun bir noktaya olan uzaklığı formülü kullanılmıştır. Örneğin; $A(x_1, y_1)$ ve $B(x_2, y_2)$ noktaları arasındaki doğrunun herhangi bir engelden geçip geçmediğini bulmak için öncelikle denklem (8) kullanılarak doğrunun denklemi edilmektedir. $xx \ yy \ xx \ yy \ 21 \ 21 \ 2 \ 2 = -$ (8) Denklemin uygulanması sonucunda, $a=(y_2-y_1)$, $b=(x_2-x_1)$ ve $c=(y_1 \times x_2 - y_2 \times x_1)$ olarak bulunmaktadır. Bu durumda, $O(x_3, y_3)$ merkezli bir engelin AB doğrusuna uzaklığı denklem (9) ile hesaplanmaktadır.

$ab \ ax \ by \ c \ engel \ 22 \ 33 = + \ + \ +$ (9) denklem, engelin yarıçap değerinden küçük veya eşit olduğunda rotanın engelli bir alandan geçtiğini ifade etmektedir. Rota üzerindeki noktalar arasında bulunan engel sayısı kadar, mevcut rotanın amaç fonksiyon değerine engel maliyeti eklenmektedir. Algoritma en az maliyetli amaç fonksiyon değerlerini bulmaya çalıştığından, engel maliyetli amaç fonksiyon değerlerine sahip rotalar algoritmanın iteratif çalışması ile birlikte silinecektir. Bu bilgilere dayanarak, bir rotanın amaç fonksiyon değeri denklem (10) ile hesaplanmaktadır. $f_i = d_i + E_i$ (10) f_i değeri i. rotanın amaç fonksiyon değerini, d_i değeri i. rotanın uzunluğunu ifade etmektedir.


Ei değeri ise i. rota üzerindeki toplam engel sayısı ile engel maliyetinin çarpımını göstermektedir. Çalışmada, İHA için en uygun rota bulunduğundan sonra, takip edilecek rota üzerinde bulunan coğrafi engeller (dağ, tepe vb.) de dikkate alınmaktadır. Coğrafi engellerin koordinat, yükseklik bilgileri Google gMapControl eklentisi aracılığıyla tespit edilmektedir. Belli adımlarda, yükseklik bilgileri alınarak rota üzerinde bulunan coğrafi engellerin tespit edilmesi için tarama mesafesi kullanılmıştır. Yüksekliklerin belirlenmesi için başlangıç noktasından hedef noktaya kadar olan rota üzerinde herhangi bir coğrafi engel olup olmadığının kontrolü için yükseklik matrisi oluşturulmuştur. Tarama mesafesi, rota uzunluğuna göre orantısız olarak değişmekle birlikte, sabit olarak ta kullanılmıştır. Benzetim çalışmaları için tarama mesafesi 50m olarak kullanılmıştır. Bu durumda rota boyunca her 50m’de bir coğrafi engel olup olmadığı ve varsa yüksekliği kontrol edilmektedir. Rota üzerinde coğrafi engelli bölgeye yaklaşıldığında, bu engellerden sakınım için, İHA’nın kendi etrafında dönerek engelleri aşması hedeflenmiştir. İHA’nın kendi etrafında dönerek ulaşması gereken coğrafi engeli aşması için (11) numaralı denklemden yararlanılmıştır.
$$h = \frac{2\pi r \times \tan(\alpha)}{360} \quad (11)$$
 h, İHA’nın kendi etrafında bir tur döndüğünde aldığı yükseklik mesafesidir. α yükseliş açısı ve r ise İHA’nın kendi etrafında döneceği minimum yarıçap değeridir. Coğrafi engelin yüksekliği bilindiğinde, h değeri ile İHA’nın kendi etrafında kaç tur dönmesi gerektiği tespit edilmektedir.



Şekil 2. YAK ile rota planlama için kullanıcı arayüzü ekranı.



Şekil 3. İHA ayarları

	YAK			
	Besin Sayısı	40	Adım Sayısı	2
	Limit Değeri	50	İterasyon Sayısı	100

Şekil 4. YAK algoritması ayarları

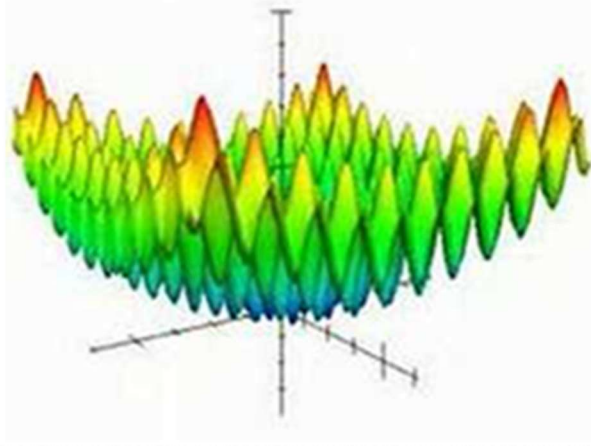
Sonuç

Bu çalışmada, otonom İHA'lar için rota planlaması problemi YAK algoritması kullanılarak çözülmüştür. Rota üzerinde bulunan bağlantı koordinat noktaları dinamik olarak belirlenmiştir. Bu sayede algoritma engellerin sayısına göre bağlantı koordinat sayısını değiştirerek en uygun yolu bulmaya çalışmaktadır. C# programlama dili kullanılarak kullanıcı etkileşimli bir arayüz geliştirilmiştir. Geliştirilen arayüz uygulamasında Google çevrimiçi harita kullanılmış ve rota için gerekli olan koordinat ve yükseklik bilgileri bu harita ile sağlanmıştır. Arayüz yardımıyla kullanıcıların başlangıç, hedef ve engellerin konumlarını girerek, İHA ile YAK algoritması için gerekli ayarları yaparak benzetim uygulamaları yapabilmelerine olanak sağlanmıştır. Rota üzerindeki dağ, tepe gibi coğrafi bölgeler harita üzerinde kullanıcıya gösterilmiş ve bu bölgelerde İHA'nın kendi etrafında dönüp yükselerek coğrafi engeli aşması gerektiği tespit edilmektedir. Yapılan farklı benzetim uygulamaları, İHA için rota planlama probleminde YAK algoritmasının başarılı sonuçlar verdiğini göstermektedir.



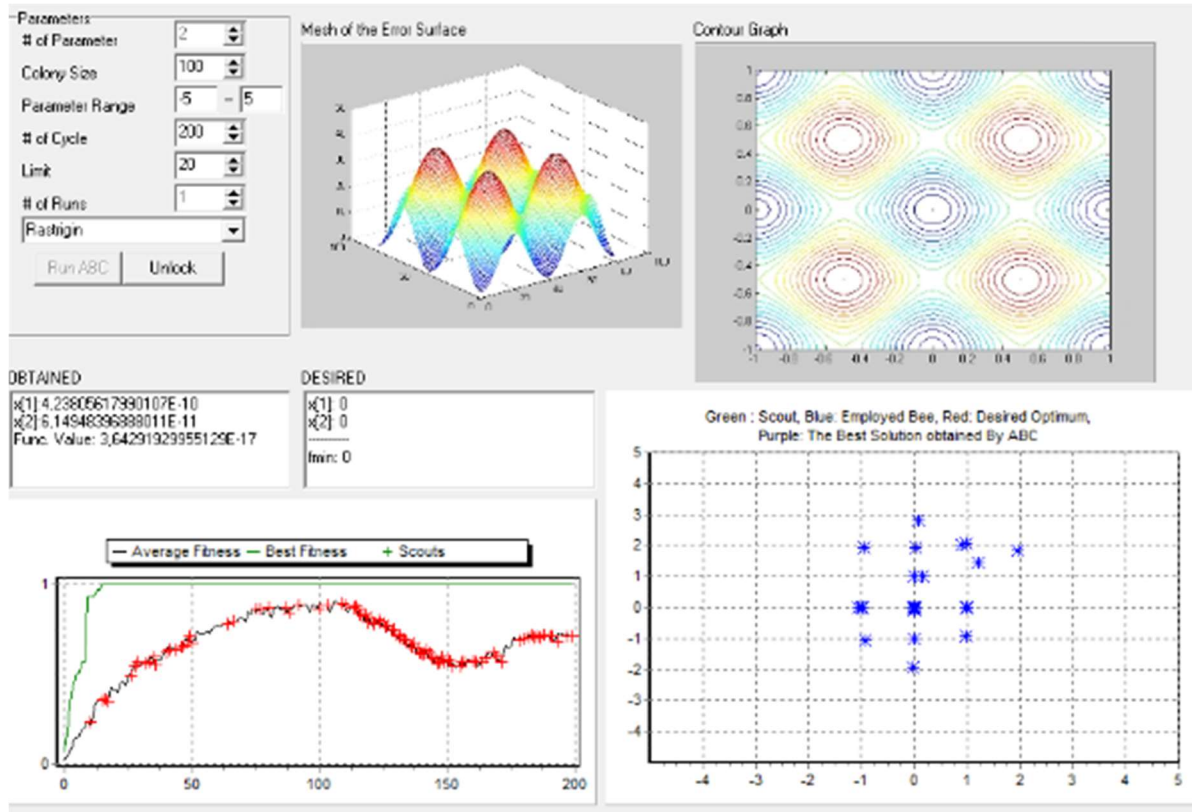
UYGULAMA VE DENEY SONUÇLARI

Algoritmanın performansını Rastrigin Problemi üzerinde görmeye çalıştık. Bu anlamda bu fonksiyonu çözmeye çalıştık. Deney sonuç ve performanslarına geçmeden önce Rastrigin Problemi ve Fonksiyonu nedir ve ne işe yarar bundan bahsetmek istiyorum. Rastrigin fonksiyonu içerisinde birçok lokal minimumu içeren ve bu yüzden de optimizasyon tekniklerinin performansını ölçmek için ideal olan bir test fonksiyonudur. Fonksiyonun global minimumu iki boyutlu uzay için [0,0] noktası, üç boyutlu bir uzay için ise [0,0,0] noktasıdır. Fonksiyonun grafik üzerindeki görünümü aşağıdaki şekilde gibidir.



$$f(\mathbf{x}) = An + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)]$$

Kullandığım demo uygulama <http://mf.erciyes.edu.tr/abc/> web sitesinden alınmıştır. Çeşitli optimizasyon problemleri üzerinde ABC Algoritması' nın performansını gözlemlemektedir. Biz yukarıda da bahsettiğimiz üzere bu fonksiyonlardan Rastrigin Fonksiyonunu kullandık. Demo uygulamanın genel görünümü aşağıdaki gibidir.



Öncelikle uygulamanın parametrelerine bakalım. Demo üzerinde parametreler ve açıklamaları aşağıda sırasıyla verilmiştir.

- # of parameter: Üzerinde çalışılan uzayın boyutu. Biz uygulamamızda anlaşılması ve gözlemi kolay olması maksadıyla 2 boyutlu uzayda çalışmayı tercih ettik.
- Colony Size: Kolonide bulunan arı miktarı.
- Parameter Range: Algoritmanın çalışmasını istediğimiz aralık. Yukarıdaki görülen şekilde aralık [-5,5] olarak seçilmiştir. Böylelikle alan daraltılarak arıların hareketi daha rahat görülmektedir.
- # of Cycle: Algoritmanın durdurma kriteri olarak çalışma sayısı belirtilmiştir.
- Limit: Her döngüde görevli ve gözcü arılar arama süreçlerini tamamladıktan sonra çözüm geliştirememeye sayacılarına bakarlar eğer ki limit değeri aşılmışsa yani algoritma belli bir süre boyunca artık iyi değer vermemeye başlamışsa bu bölgeden vazgeçilir. Limit değeri bu sayacı belirtmektedir.
- # of Runs: Algoritmanın baştan itibaren kaç defa çalışacağı. Biz her defasında varsayılan değer kabul edilen bir kez çalıştırdık.

Ayrıca demo şekil üzerinde görülen Obtained kısmı algoritmanın bulduğu en iyi değerleri, Desired kısmı fonksiyonun beklenen değerlerini, üst soldaki grafik Rastrigin Fonksiyonu'nun grafiksel olarak üç boyutlu görünümünü, üst sağdaki grafik fonksiyon grafiğinin üstten görünümünü, sol alttaki grafik en iyi değerlerin kaçınıcı döngüde bulunduğunu ve bulunan değerlerin ortalama iyiliğini göstermektedir. Sağ alttaki grafik ise arıların hareketini her döngüde izleme imkânı sunmaktadır. Buradaki mavi noktalar, görevli işçi arılar; yeşil noktalar, kaşif; kırmızı noktalar, beklenen değer; ve son olarak da mor noktalar, o ana kadar ki bulunan en iyi çözüm noktasını göstermektedir.

Deney Sonuçları: Yaptığımız deneylerde Colony Size, Cycle Sayısı ve Limit Sayısı parametrelerinin algoritmanın performansı üzerindeki etkilerine baktık. Her deney çeşidi için en az 5'er adet deney yapılmıştır.

Colony Size Deneyi

Sabit Değerler: Parameter Range (-5,5), Cycle (100), Limit (20)

1. Deney Colony Size Değeri: 30
2. Deney Colony Size Değeri: 50
3. Deney Colony Size Değeri: 60
4. Deney Colony Size Değeri: 75
5. Deney Colony Size Değeri: 100

Deney Değerlendirmesi: Bulunan en iyi değerlere bakıldığında yaptığımız deney için Colony Size değişimi çok etki etmemiştir fakat Colony Size'in artışı algoritmanın daha az döngüde en iyi değerlere ulaşmasını sağlamıştır. Yani koloni büyüklüğü problemin daha kısa sürede çözülmesini sağlamaktadır. Bu durumda eğer ki Colony Size'i artırma imkânımız yoksa döngü (cycle) sayısının yüksek tutulması gerekir. Aksi takdirde en iyi çözüme çok yakın olan çözümlere ulaşamayabiliriz.

of Cycle Deneyi (Döngü Sayısı)

Sabit Değerler: Colony Size(50), Parameter Range (-5,5), Limit (20)

6. Deney Cycle Değeri: 10
7. Deney Cycle Değeri: 20
8. Deney Cycle Değeri: 50
9. Deney Cycle Değeri: 70
10. Deney Cycle Değeri: 100

Deney Değerlendirmesi: Colony Size deneyinde 10 ile 100 döngü arasında deneyler yaptık. 10 beklenen değerler için çok bir sayı oldu ve hata oranı çok fazla çıktı. Belirlenen parametrelerde için bu deneyde cycle değerini artırmaya başladık ilk anlamlı sonuç 20'de çıkmıştır ve beklenen değere yüzde bir oranında hataya sahip değerler bulunmuştur. Bundan sonraki artışlar hata oranını giderek düşürmeye başlamış fakat yaklaşık 70 cycle'dan sonra hata oranında çok fazla bir değişim olmamaya başlamıştır. Belirlenen sabit parametrelere göre bu deney için yaklaşık 70 cycle değeri uygun bir değer olarak belirlenmiştir. Cycle değeri belli bir yere kadar olumlu etki etmekte, belli bir yerden sonra ise etmemektedir.

Limit Deneyi

Sabit Değerler: Colony Size(50), Parameter Range (-5,5), Cycle(80)

11. Deney Limit Değeri: 5
12. Deney Limit Değeri: 10
13. Deney Limit Değeri: 20
14. Deney Limit Değeri: 40
15. Deney Limit Değeri: 50
16. Deney Limit Değeri: 60
17. Deney Limit Değeri: 70

Deney Değerlendirmesi: Limit deneyinde diğer parametreler sabit kalmak koşuluyla limit değeri olarak 5'ten 70'e kadar deneyler yaptık. 5 değerinde yüksek cycle değerlerinde beklenen değere yakın değerler elde ettik. Limit değerini artırmaya başladık ve yavaş yavaş daha az cycle'da beklenen değerlere çok yakın değerler elde etmeye başladık. Fakat sabit değerlerle birlikte yaklaşık 20 limit değerinden sonraki değerlerde çok tutarlı sonuçlar elde edemedik. Yani 20 için daha az iken, 40 ta cycle değeri arttı, ama 50'de yine azaldı. Bu deneyler sonucunda limit değerinin cycle sayısına göre ayarlanması gerektiğini düşünüyoruz. Çünkü bu deneyde cycle 80 iken limit değerini 70'e kadar artırdık. Belli bir noktadan sonra anlamsız sonuçlar elde ettik. Bu sebeple limit için bizim düşüncemiz cycle sayısının yaklaşık yüzde 10'u ile %25'i arasında bir değer alması ideal olacaktır.