YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ARI KOLONİSİ YÖNTEMİ İLE İNSANSIZ HAVA ARAÇLARI İÇİN YOL PLANLAMA

YÜKSEK LİSANS TEZİ

Volkan ÇAVUŞ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

ŞUBAT 2017

YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ARI KOLONİSİ YÖNTEMİ İLE İNSANSIZ HAVA ARAÇLARI İÇİN YOL PLANLAMA

YÜKSEK LİSANS TEZİ

Volkan ÇAVUŞ
125105013

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Adem TUNCER

ŞUBAT 2017

YALOVA Üniversitesi Fen Bilimleri Enstitüsü'nün 125105013 numaralı Yüksek Lisans Öğrencisi **Volkan ÇAVUŞ**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"YAPAY ARI KOLONİSİ YÖNTEMİ İLE İNSANSIZ HAVA ARAÇLARI İÇİN YOL PLANLAMA"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Yrd. Doç. Dr. Adem TUNCER**
Yalova Üniversitesi .....

Jüri Üyeleri : **Prof. Dr. Mehmet YILDIRIM**
Kocaeli Üniversitesi .....

Yrd. Doç. Dr. Adem TUNCER
Yalova Üniversitesi .....

Yrd. Doç. Dr. Osman Hilmi KOÇAL
Yalova Üniversitesi .....

Teslim Tarihi : **05 Ocak 2017**
Savunma Tarihi : **08 Şubat 2017**

Kızım Alya Zehra'ya,

ÖNSÖZ

Teknolojik gelişmeler her alanda olduğu gibi insansız hava araçları (İHA'lar) için de hızlı bir şekilde ilerleme kaydetmektedir. Özellikle otonom olarak hareket eden İHA'ların en önemli görevlerinden biri, başlangıç ve hedef noktası üzerinde en kısa mesafe ve sürede ulaşabileceği güvenli bir rota planlaması yapabilmesidir. Rota planlama konusunda yıllardır birçok yöntem geliştirilmiş ve uygulanmıştır. Bu yöntemler dışında son yıllarda sezgisel tabanlı algoritmalar da rota planlama için kullanılmaktadır. Bu tez çalışmasında, C# ile tasarlanan kullanıcı etkileşimli arayüz ile İHA'nın başlangıç ve hedef noktaları arasında coğrafi ve sanal engellerden sakınarak güvenli bir şekilde ulaşımını sağlamak için rota planlaması yapılmıştır. Rota planlamasında sezgisel tabanlı algoritmalardan bir tanesi olan yapay arı kolonisi (YAK) algoritması kullanılmıştır.

Tez çalışması süresince desteğini hiçbir zaman esirgemeyen, bilgi ve tecrübelerinden yararlanırken göstermiş olduğu hoşgörü ve sabrından dolayı değerli danışmanım Yrd.Doç.Dr.Adem TUNCER'e teşekkürü bir borç bilirim.

Bugünlere gelmemde büyük emeği geçen anne ve babama, tez çalışmamın her aşamasında beni destekleyen, teşvik eden ve manevi desteği ile yanımda olan sevgili eşime ve aileme sonsuz teşekkür ederim.

Şubat 2017 Volkan ÇAVUŞ

İÇİNDEKİLER

Sayfa

KISALTMALAR

ABD : Amerika Birleşik Devletleri
CPU : Central Processing Unit
CUDA : Compute Unified Device Architecture
ÇEGOTA : Çok Etmenli Grid Ortam Tarama Algoritması
GA : Genetik Algoritma
İHA : İnsansız Hava Aracı
KKA : Karınca Koloni Algoritması
MALE : Medium Altitude Long Endurance
MFC : Çoklu Robot Orman Tarama Algoritması
MSTC : ÇokluRobot Yayılan Ağaç Tarama Algoritması
NASA : Ulusal Havacılık ve Uzay Dairesi
PSO : Parçacık Sürü Optimizasyonu
SSM : Savunma Sanayii Müşteşarlığı
YAK : Yapay Arı Kolonisi

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

YAPAY ARI KOLONİSİ YÖNTEMİ İLE İNSANSIZ HAVA ARAÇLARI İÇİN YOL PLANLAMA

ÖZET

İnsansız Hava Aracı (İHA), otonom veya uzaktan kontrol olmak üzere iki farklı kullanım şekli olan pilotsuz hava araçlarıdır. Uzun yıllar istihbarat, keşif, gözetleme gibi askeri alanlarda kullanılan İHA'lar son yıllarda uzaktan algılama, veri toplama, arama-kurtarma, yangın söndürme, tarım uygulamaları, görüntüleme ve taşımacılık gibi sivil alanlarda da kendine uygulama alanı bulmuştur. İHA'ların bahsedilen bu görevleri otonom olarak yerine getirebilmeleri için pek çok çalışma yapılmaktadır. Bu çalışmalardan önemli bir tanesi de rota planlamadır.

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 çözüm yöntemlerinde geleneksel algoritmaların yanı sıra sezgisel algoritmalar da kullanılmaktadır. Sezgisel algoritmaların kullanılmasındaki en önemli nedenlerden bir tanesi, kısa zamanda iyi sonuçların elde edilmesidir. Geleneksel algoritmaların genellikle hesaplama maliyetleri yüksektir.

Bu çalışmada rota planlama için Yapay Arı Kolonisi (YAK) algoritması kullanılmıştır. Rota planlama işlemi en kısa yolun bulunması olarak düşünüldüğünde aynı zamanda bir optimizasyon problemi olarak ta ele alınabilmektedir. YAK algoritması son yıllarda optimizasyon problemlerinde yaygın bir şekilde kullanılan sezgisel algoritmalardan bir tanesidir. Çalışmada, YAK algoritmasının benzetim uygulamalarını gerçekleştirmek amacı ile C# programlama dili kullanılarak kullanıcı etkileşimli bir arayüz tasarlanmıştır. Geliştirilen kullanıcı arayüzü üç aşamadan oluşmaktadır. Birinci aşamada; rotanın başlangıç ve bitiş konumları girilerek harita üzerinde tespit edilmekte, ikinci aşamada; enlem-boylam bilgileri harita üzerinde fare yardımıyla veya el ile girilerek sanal engeller oluşturulmakta, üçüncü aşamada; İHA ve YAK için gerekli parametre ayarları yapılmaktadır. Uygulamada Google çevrimiçi harita kullanılmış ve rota için gerekli olan koordinat ve yükseklik bilgileri bu harita ile sağlanmıştır. Rota planlaması için yapılan deneysel çalışmalar, YAK algoritmasının uygun rotalar bulmada başarılı sonuçlar verdiğini göstermektedir.

ROUTE PLANNING FOR UNMANNED AERIAL VEHICLES WITH ARTIFICIAL BEE COLONY

SUMMARY

Unmanned Air Vehicle (UAV) is a non-pilot air vehicle which has two different modes of use, autonomous or remote control. UAVs that have been used for many years in military fields such as intelligence, reconnaissance and surveillance, have been also begun to applied to civilian fields such as remote sensing, data collection, search and rescue, fire fighting, agricultural applications, imaging and transportation recently. Much work has been doing to ensure that UAVs can autonomously to perform these tasks. An important one of these studies is planning of the route.

A route planning for UAV is to find a feasible route from the starting point to the target point avoiding obstacles and dangerous areas. Many methods are used in order to solve route planning of UAVs. In addition to traditional algorithms, heuristic algorithms are also used to solve these problems. One of the most important reasons for using heuristic algorithms is to get good results in a short time. Traditional algorithms usually have high computational costs.

In this study, Artificial Bee Colony (ABC) algorithm is used for route planning. Route planning could be defined as an optimization problem when it has considered to find shortest route. The ABC algorithm is one of the heuristic algorithms widely used in optimization problems in recent years. In the study, an interactive user interface is designed using C# programming language to implement the simulation of the ABC algorithm. At the first phase, the starting and ending points of route have been determined on map;at the second phase imaginary barriers have made by inputting latitude and longitude coordinates with using mouse on map or keyboard. At third phase some parameters have adjusted. In practice, the Google online map has been used and the coordinates and altitude information required for the route have been provided with this map. Experimental studies for route planning show that the ABC algorithm provides successful results in finding suitable routes.

1. GİRİŞ

Kendi güç sistemi olan, otomatik veya uzaktan kontrol sistemi ile uçurulan pilotsuz hava araçlarına İnsansız Hava Aracı (İHA) denilmektedir. ABD'nin öncülüğünde bulunan İHA'lar "pilotsuz uçak", "robot uçak", "drones" gibi isimlerle adlandırılmaktadır (Akyürek ve ark., 2012). ABD Savunma Bakanlığı Ofisi Mart 2003 İHA Yol Haritasında, İHA'nın tanımını "İnsan operatörü taşımayan, hava aracını kaldırmak için aerodinamik kuvvetleri kullanan, bağımsız ya da uzaktan kontrollü uçabilen, tekrar kullanılabilen ya da kullanılamayan, öldürücü veya öldürücü olmayan yük taşıyabilir. Balistik veya yarı balistik araçlar, kruz füzeleri ve topçu mermileri insansız hava araçlarında kabul edilmez" olarak kullanır (USA Department of Defense, 2002). Sivil Havacılık Otoritesi tarafından Mayıs 2002'de yayınlanan klavuzda İHA'nın tanımı "İnsansız uçmak için tasarlanmış ya da değiştirilmiş bir uçak, uzaktan kontrol veya otonom modda işletilir" olarak yapılmıştır (Civil Aviation Authority, 2002).

İHA'lar XX. yüzyılın başından beri aşırı dikkat gerektiren ve uzun süreli insan odaklanmasının azalacağı düşünülen tehlikeli ve riskli ortamlarda kulanılmaktadır. Savunma Sanayii Müsteşarlığımızın 2004 yılından sonra yürüttüğü insansız hava aracı projeleri Şekil 1.1'de gösterilmiştir (SSM, 2011).

Şekil 1.1 : SSM 2004 sonrası yürüttüğü İHA projeleri.

1.1 İnsansız Hava Araçlarının Kullanım Alanları

İHA'lar genellikle Birinci Dünya Şavaşı'ndan bu yana bir dizi askeri uygulamalar için kullanılmıştır. İHA'ların ilk yaygın olarak kullanımı ABD tarafından Vietnam'daki savaş sırasında olmuştur (Ingham, 2008). İHA'lar askeri amaçlarının dışında sivil amaçlar için de kullanılmaktadır (SSM, 2011).

Askeri kullanım alanları;

- Keşif /Gözetleme Desteği
    - Taktik saha keşif/gözetleme
    - Stratejik keşif/gözetleme
- Taarruz
    - İç güvenlik

    - Yakın hava desteği
    - Hava savunma sistemlerinin imhası
    - Hava sahası savunma
- Hedef Benzetimi
    - Hedef uçak
    - Sahte uçak
- Elektronik Harp
    - Sinyal istihbaratı
    - Radar elektronik harp
    - Muharebe elektronik harp
    - Önleyici elektronik harp
- Özel Görevler
    - Haberleşme desteği
    - Mayın patlayıcı desteği
    - Arama-kurtarma/lojistik
    - Kentsel harp
    - Kimyasal, biyolojik, radyoaktif, nükleer tespiti
    - Çoklu İHA görevi
    - Deniz karakol/denizaltı savunma harbi
    - Kargo taşıma

Sivil kullanım alanları;

- Çevresel Kullanım
    - Atmosfer araştırması
    - Okyanus gözlemleri
    - Kasırga oluşumu ve araştırması
    - Jeolojik araştırmalar
    - Volkan çalışmaları ve patlama uyarıları
    - Hava durumu tahmini
- Acil Durum Uygulamaları
    - Yağ kaçağı gözleme
    - Sel izleme
    - Kasırga izleme
    - Afet operasyon yönetimi
    - Felaket durum değerlendirmesi
    - Arama kurtarma

    - Yangınla mücadele
    - Deprem gözleme
    - Volkan gözleme
- İletişim Uygulamaları
    - Geniş bant iletişim
    - Telekominikasyon role hizmetleri
    - Cep telefonu iletişimi
    - Küresel konumlandırma ve uydu sistemleri
- İzleme Operasyonları
    - Deniz devriyesi
    - Balıkçılık izleme
    - Kıyı şeridi izleme
    - Uluslararası sınır devriyesi
    - Uyuşturucu trafiği izleme
    - Yol trafiği izleme ve kontrolü
    - Kanun uygulamaları
    - Orman yangını tespiti
    - Ekin ve hasat durumu izleme
    - Çevre durumu izleme
    - Arazi haritalama
    - Yüksek voltajlı güç hattı izleme

1.2 Literatür Araştırması

Leong ve diğ. (2016) tarafından yapılan çalışmada, demiryolu ve metro ulaşım sistemlerinde gezgin satıcı problemini çözümü için bir optimizasyon algoritmasının geliştirilmesi amaçlanmıştır. Birden fazla hedefe en uygun rota seçimi için geliştirilmiş YAK algoritması kullanılmıştır. Geliştirilen algoritmada, 5 örnekte 4 istasyon, 2 örnekte 5 istasyon ve 3 örnekte 6 istasyon kullanılarak elde edilen toplam 10 örneğin sonuçları, gerçek sonuçlarla karşılaştırılmış ve algoritmanın performansı değerlendirilmiştir. Ulaşım sisteminin karmaşık olmasından dolayı, en iyi rotayı bulmak için algoritmanın yavaş olmasına rağmen doğru sonuçlar verdiği belirtilmiştir.

Baghel ve Mishra (2016) tarafından gerçekleştirilen çalışmada, mevcut araç navigasyon cihazlarında başlangıç noktasından hedef noktasına kadar olan rotanın planlamasında YAK algoritması kullanılmıştır. Çalışmada rota planlamayı gerçekleştirirken rota uzunluğu, seyahat süresi ve sürüş kolaylığı dikkate alınmıştır. Rota planlamada yol sınıfı, şerit sayısı, düğüm noktaları ile uyarı işaretlerinin sayısı, yolun yüzeyi ve trafik yoğunluğu gibi kriterlerin amaç fonksiyon değerini etkilediği belirtilmiştir. YAK algoritmasıyla elde ettikleri deneysel sonuçlar, GA ve Dijkstra algoritmasıyla karşılaştırılmıştır.

Çalık (2016) çalışmasında, Karınca Koloni Algoritması (KKA) kullanarak İHA'larda rota planlaması yapmıştır. Etmen tabanlı programlama ortamı olan Netlogo ile simüle edilen çalışmada, iki boyutlu düzlemde gerçekleştirilen rota planlaması yapılmış ve engellere değmeden İHA'nın hedef noktalarına uğrayarak rotasını tamamlaması sağlanmıştır.

Li ve diğ. (2014) tarafından gerçekleştirilen çalışmada, insansız savaş hava araçları için muharebe alanındaki tehditler ve kısıtlamalar dikkate alınarak en uygun yolun elde edilmesi amaçlanmıştır. Bu çalışmada denge-evrim stratejisi ile geliştirilen YAK algoritması kullanılmıştır. Bu yeni algoritma, yineleme sırasında yakınsama bilgileri, keşif ve kullanım hassasiyetini değiştirmek, yerel kullanım ve küresel keşif yetenekleri arasında denge kurmak için kullanılmıştır. Simülasyon sonuçlarına göre geliştirilen algoritmanın, YAK, modifiye edilmiş YAK ve iç geribildirim YAK algoritmalarına göre daha fazla kararlılık gösterdiği belirtilmiştir.

Çekmez (2014) çalışmasında, Genetik Algoritma (GA) ve KKA'yı grafik kartları üzerinde paralelleştirilerek İHA'nın rota planlamasında kullanmıştır. Çalışmada İHA'nın belirtilen noktalara uğrayarak ve sanal engellere çarpmadan iki boyutlu düzlemde rota planlaması yapılmıştır. Problem çözümünde CUDA mimarisi kullanan Çekmez aynı zamanda çalışmayı geleneksel CPU mimarisiyle de karşılaştırmıştır. Karşılaştırma işlemi 52, 76, 100, 225, 439 ve 1002 noktadan oluşan rotalarda, 1024, 2048, 4096 ve 8192 popülasyon sayıları kullanılarak hata ve işlem sürelerine göre yapılmıştır.

Aydemir (2014) çalışmasında, Grid tarama algoritmasını kullanarak İHA'da güzergah problemini çözen simülasyon tabanlı karar destek yazılımı geliştirmiştir. İki boyutlu düzlem üzerine yerleştirilen engellere göre rota planlaması yapılmıştır. Çalışmada arazi tipi boş, dış mekan ve iç mekan, etmen sayısı 2, 8, 14 ve 20, gruplandırma oranı 30, 60 ve 100 kriterlerine göre Çok Etmenli Grid Ortam Tarama Algoritması (ÇEGOTA), Çoklu Robot Yayılan Ağaç Tarama Algoritması (MSTC) ve Çoklu Robot Orman Tarama Algoritması (MFC) kullanılmış ve bu algoritmalar tarama sayı ve oranlarına göre karşılaştırılmıştır. Aydemir, çalışmada boş arazi ve iç mekanda MFC ve ÇEGOTA arasında fark olmadığı, dış mekanda ise MFC'nin daha iyi olduğu sonucuna ulaşmıştır.

Geng ve Zhao (2013) yaptıkları çalışmada, İHA rota planlamasının temel modelini incelemişlerdir. Daralma faktörü ile geliştirilmiş Hibrit Parçacık Sürü Optimizasyonu (PSO) kullanarak İHA'da rota planlaması yapan Geng ve Zhao rota planlamasını iki boyutlu düzlemde gerçekleştirerek Matlab ile simüle etmişlerdir. Çalışmada Hibrit PSO yönteminin basit ve etkili olduğunu ve İHA'larda yol planlaması için tüm gereksinimleri karşılayacağını göstermişlerdir.

Özalp (2013) çalışmasında, NASA'ya ait çevrimdışı üç boyutlu uydu görüntülerini kullanarak GA yardımıyla tekli ve çoklu İHA için rota planlaması gerçekleştirmiştir. Çalışmada rota üzerindeki dinamik engele değmemek için engellerin etrafında belirli uzaklık ve açılarda sanal noktalar oluşturulmuştur. İHA bu sanal noktalara uğrayarak engelin etrafından dolaştırılmıştır. Statik engellerden kurtulmak için İHA'nın kendi etrafında dönerek yükselmesi sağlanmış ve engeller aşılmıştır. Rota planlaması sırasında İHA'lara ait kinematik kısıtlar ve arazi şartları gözönünde bulundurulmuştur. Özalp kontrol noktası 202, 411, 662 ve 963, popülasyon birey sayısı 100, mutasyon oranı %20, elitizm oranı %20 olan GA değerleri ile 1,2,3 ve 4 adet İHA için test sonuçları elde etmiştir.

Ergezer ve Leblebicioğlu (2012) tarafından gerçekleştirilen çalışmada, GA kullanarak bir veya daha fazla İHA için güzergah planlaması yapılmıştır. İki boyutlu düzlemde gidilmesi ve gidilmemesi gereken noktalar belirlenerek, başlangıç noktasından başlayıp istenilen noktalar gezilerek tekrar başlangıç noktasına gelebilecek en kısa güzergahın bulunması amaçlanmıştır. Arama uzayının nasıl indirgenebileceğini göstermesi açısından tahmin edici sonuçlara ulaşlmıştır.

Arıca ve diğ. (2012) tarafından gerçekleştirilen çalışmada, MALE tipi İHA'lar için A* algoritması kullanılarak Java platformu üzerinde çok kriterli güzergah planlaması yapılmıştır. Coğrafi yapı eş yükselti eğrilerinin bulunduğu sayısal haritalardan faydalanılarak üç boyutlu arama uzayına aktarılmış ve grid yapısı kullanılmıştır.

Akça (2011) çalışmasında, Hibrit YAK algoritmasını kullanarak gezgin satıcı problemini Türkiye'deki il ve ilçe merkezlerine uygulamıştır. Arı Kolonisi Optimizasyonu temel alınarak hibrit bir algoritma geliştirilmiştir. Opt-2 arama algoritması hibrit yapı içerisine dahil edilerek yerel alanda iyileşme gerçekleştirmiştir. Simetrik ve asimetrik yapıda küçük, orta ve büyük boyutlu gezgin satıcı test problemlerine hibrit yapıdaki algoritmayı uygulamıştır. Türkiye'deki 81 il ve 888 ilçe merkezi için deneysel uygulamalar deneyen Akça, daha önce PSO ve GA ile yapılan çalışmalarla sonuçları karşılaştırmıştır.

Pakkan ve Ermiş (2010) tarafından gerçekleştirilen çalışmada, çevrimdışı olarak rota planlaması yapılmıştır. İstenilen hedeflerin coğrafi koordinatları Google Maps üzerinden veri paketleri şeklinde Matlab Simulink'e aktarılmış ve GA ile bu veri paketleri girdi olarak alınıp İHA için en uygun rota hesaplanmıştır. Yineleme sayısı 6000, popülasyon boyutu 80 olan GA'da 3, 5 ve 10 adet İHA için 15, 30, 45 ve 60 adet sabit hedefe göre izlenecek olan rota sonuçları elde edilmiş ve farklı parametre değerlerine göre sonuçlar karşılaştırılmıştır.

Kanok ve Hara (2008) tarafından gerçekleştirilen çalışmada, rota planlama problemini tek amaçlı bir problem yerine çok amaçlı bir problem olarak ele alınmış ve sürüş süresi, sürüş kolaylığı ile rota uzunluğu olmak üzere eş zamanlı üç amaç fonksiyonu kullanılmıştır. Çok amaçlı hibrid GA (GA+Dijkstra) algoritması kullanılarak en uygun rotanın planlaması yapılmıştır. Dijkstra algoritması ve tek yönlü GA ile karşılaştırılan deneysel sonuçlarda gerçek trafik bilgileri ile gerçek yol haritası kullanılmıştır. Çalışmanın değerlendirme aşamasında, yanlızca bir harita ve bir günlük verilerden faydalanılmıştır.

1.3 Çalışmanın Amacı

Rota planlama için kullanılan Dijkstra, A* gibi geleneksel algoritmaların yanı sıra sezgisel algoritmalar da kullanılmaktadır. Geleneksel algoritmaların özellikle hesaplama maliyetlerinin yüksek olmasından dolayı, kısa zamanda iyi sonuçlar üreten sezgisel algoritmalar da pek çok çalışmada kullanılmıştır. Örneğin Tuncer (2015) çalışmasında sezgisel bir yöntem olan Genetik Algoritma ile A* algoritmasını karşılaştırmıştır. Benzetim sonuçları Genetik Algoritma ile karşılaştırıldığında, küçük arama uzayında A* algoritmasının daha hızlı sonuç bulmasına karşın, arama uzayının genişlemesiyle A* algoritmasının logaritmik olarak arttığını göstermiştir. Bu bakımdan bu çalışmada da gelenekel yöntemler kullanmak yerine hızlı sonuç bulmasından dolayı sezgisel yöntem tercih edilmiştir.

Bu tez çalışmasında, İHA'ların coğrafi ve sanal engellerden sakınarak başlangıç noktasından hedef noktasına en kısa yolu kullanarak güvenli bir şekilde hareket edebileceği rota planlaması yapılmıştır. İHA'lar için rota planlama problemi YAK algoritması kullanılarak çözülmüştür. Rota planlaması işlemi yapılırken gerçek harita bilgileri kullanılmış ve haritadaki coğrafi koşullar dikkate alınmıştır. Çalışmada YAK algoritmasının benzetim uygulamalarını görsel olarak gerçekleştirmek amacı ile kullanıcı arayüzü tasarlanmıştır.

2. ROTA PLANLAMA İÇİN KULLANILAN ALGORİTMALAR

2.1 Dijkstra Algoritması

En kısa yol bulma problemlerinde kullanılan algoritmalarından bir tanesi olan Dijkstra algoritması, 1959 yılında bilgisayar bilimcisi Edsger W. Dijkstra tarafından açıklanmıştır (Dijkstra, 1959). Başlangıç düğümü ile diğer düğümler arasındaki en kısa yolun bulunması için kullanılır ve bu bakımdan tüm düğümleri sırasıyla kontrol eder.

Dijkstra algoritmasında tüm düğümler geçici ve kalıcı olmak üzere iki gruba ayrılır. Her tekrarlamada, taranacak bir sonraki düğüm geçici olarak belirlenmiş en kısa mesafeli düğüm olarak seçilir. Taranan düğümler kalıcı olarak işaretlenir ve tüm düğümler kalıcı olarak işaretlendiğinde tekrarlama biter. (Zhan, 1997).

Dijkstra algoritmasında en kısa rotayı bulmak için (Al-Tameemi, 2014);

1. İlk kalıcı düğüm olarak kaynak düğümü seçilir ve maliyeti sıfır olarak belirlenir.
2. Önceki kalıcı düğümlerin komşu düğümleri kontrol edilir.
3. Kontrol edilen komşu düğümlerin kümülatif maliyetleri hesaplanır ve geçici düğüm olarak belirlenir.
4. Geçici düğümler aşağıdaki adımlarla kontrol edilir;
    a. En düşük kümülatif maliyetli düğüm seçilir ve kalıcı düğüm olarak belirlenir.
    b. Düğümlere ulaşmak için daha fazla yol varsa, en kısa kümülatif yollar seçilir.
5. Tüm düğümleri kalıcı düğüm yapmak için 2 ile 4 arasındaki bütün adımlar tekrarlanır.

2.2 A* (A-Yıldız) Algoritması

Çok yüksek veya zorlu karmaşık hesaplamalı arama problemleri için sezgisel arama teknikleri uygulanmıştır. En yaygın olarak kullanılan sezgisel arama algoritması olan A* algoritması (Passino ve Antsaklis, 1994), 1968 yılında Peter E. Hart, Nils J. Nilsson ve Bertram Rapheal tarafından açıklanmıştır (Hart ve diğ., 1968).

Başlangıç düğümünden son düğüme kadar olan maliyetin en az olanını bulmayı sağlayan A* algoritması (Aygün ve Akçay, 2015), ağaçta ziyaret edilen düğümleri belirlemek ve yönlendirmek için eşitlik (2.1)'de verilen *f(n)* değerlendirme fonksiyonunu kullanır. Eşitlikteki *g(n)*, ilk düğümden *n* düğümüne olan gerçek maliyeti ve *h(n)*, *n* düğümünden hedef düğüme olan tahmini maliyeti temsil etmektedir (Alshawi ve diğ., 2012).

$$f(n) = g(n) + h(n) \tag{2.1}$$

A* algoritmasının adımları aşağıdaki gibidir (Yao ve diğ., 2010);

1. Başlangıç düğümü *s*, AÇIK tablosuna eklenir ve *f(s)=0+h(s)*'dir.
2. Hedef düğüm bulunana kadar aşağıdaki adımlar tekrarlanır
    a. AÇIK tablosu boş ise döngüden çıkılır
    b. AÇIK tablosundaki en küçük *f* değerine sahip düğüm AÇIK tablosundan çıkarılıp KAPALI tablosuna eklenir.
    c. *n* düğümü hedef düğüm ise döngüden çıkılır ve en iyi yol bulunur.
    d. En iyi *n* düğümü hedef düğüm değil ise *n* düğümü genişletilir. *n* düğümü yerine *m* düğümü eklenir.
    e. *m* düğümü AÇIK ve KAPALI tablosunda değil ise AÇIK tablosuna eklenir.
    f. *g(m)<g(k)* ve *m* düğümünün yedeği *k* düğümü AÇIK tablosunda ise AÇIK tablosundan *k* düğümü çıkartılıp *m* düğümü eklenir.
    g. KAPALI tablosunda yerine geçen düğümler varsa aşağıdaki işlemler uygulanır;
        i. KAPALI tablosundaki *k* düğümü ile *m* düğümü değiştirilir.

ii. Yerine geçen öğelerin sıradaki k düğümü içeren AÇIK ve KAPALI tablosundaki f değeri, yedeği g ile değiştirilir.

h. f değerine göre AÇIK tablosundaki düğümlere küçükten büyüğe doğru gidilir.

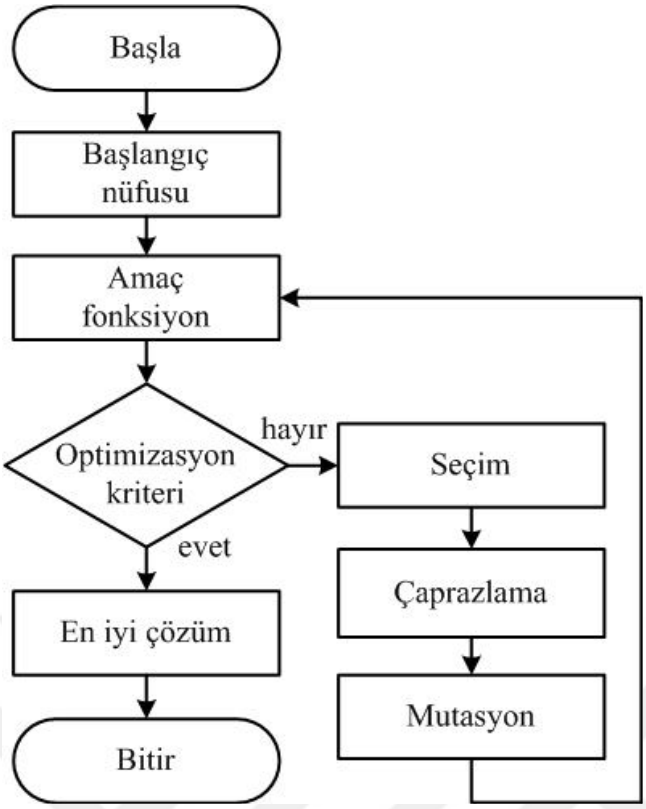
3. 2. adıma geri dönülür ve döngü tekrarlanır.

2.3 Genetik Algoritma (GA)

GA evrime dayalı algoritmaların bir türüdür. 1975 yılında John Holland, arkadaşları ve öğrencileri tarafından geliştirilmiştir (Karaboğa, 2004). Temelleri genetik bilimine ve bu bilimde yer alan seleksiyon, çaprazlama ve mutasyon kavramlarına dayandırılan optimizasyon tekniği (Tuncer, 2007) olan GA, gezgin satıcı problemi, rota planlama, mekanik öğrenme, otomatik programlama ve bilgi sistemleri, çizelgeleme problemleri ve sistem güvenliği problemleri gibi birçok alanda uygulanmaktadır (Emel ve Taşkın, 2002).

Şekil 2.1'de akış diyagramı gösterilen GA'nın çalışma adımları kısaca aşağıdaki gibi açıklanabilir (Tuncer, 2013);

- Çözüm uzayındaki olası tüm çözümler bir dizi olarak kodlanır ve bu kodlar birey olarak ifade edilir.
- Bireyler içinden rastgele bir çözüm alınarak başlangıç nüfusu oluşturulur ve birey sayısı bu nüfusun büyüklüğünü gösterir.
- Nüfus içindeki her birey için amaç fonksiyon değeri hesaplanır.
- Amaç fonksiyonunun değeri başlangıçta belirtilen kriteri sağlıyorsa algoritma sonlandırılır aksi halde algoritma çalıştırılmaya devam ettirilir.
- Amaç fonksiyon değerlerine göre bireylerin seçilme olasılığı belirlenir ve bu olasılığa göre rastgele olarak seçilme işlemi uygulanır.
- Çaprazlama ve mutasyon işlemleri uygulanarak daha iyi bireyler bulunur.
- Yeni bireylerin amaç fonksiyon değeri hesaplanır.



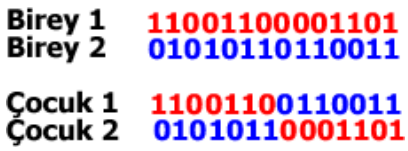
Şekil 2.1 : GA'nın akış diyagramı (Tuncer, 2013)

Başlangıç nüfusu, çözüm bilgilerini içeren bireylerin bir araya gelmesiyle oluşan olası çözüm yığınına denir (Küçük, 2016). Çözümü amaçlanan problemin özellikleri başlangıç nüfusunun büyüklüğünün belirlenmesinde etkilidir. Başlangıç nüfusunun büyüklüğü ise sonuca ulaşma süresi ve global optimuma yakın çözümün elde edilmesi olasılığı üzerinde etkilidir (Esmeray, 2016).

GA'ların ihtiyaç duyduğu şey problemin karar değişkenlerinin uygun bir yöntemle kodlanması ve neyin iyi olduğunu GA'ya belirtmek üzere tasarlanan bir amaç fonksiyonudur (Kahraman ve Özdağlar, 2004)

Goldberg'e göre, yeni nesiller için bireylerin belirlenmesi sürecinde önceki nüfustan gelen bazı bireyler yeni nüfusa aktarılması gerekmektedir (Küçük, 2016). Başlangıç nüfusu oluştuktan sonra, yeni bireylerin uyum değerleri hesaplanır. Yeni nesil oluşturacak bireyler, eşleştirmek için seçilir (Yakut ve Çankal, 2016). Nüfusta yer alan bireylerden hangilerinin ebeveyn olarak seçileceği ile ilgili çeşitli yöntemler mevcuttur. Bu yöntemlerden en çok kullanılanları rulet tekerleği, sıralama ve turnuva şeklindedir (Esmeray, 2016)

Çaprazlama işlemi, iki adet birey genlerini birleştirerek iki yeni birey oluşturmasıdır. Bu iki yeni bireyin optimum çözüme daha yakın olması beklenir (Yıldırım ve ark, 2002). Problem göre farklı çaprazlama yöntemleri kullanılmaktadır. Yaygın olarak kullanılan çaprazlama yöntemleri tek noktalı (Şekil 2.2), iki noktalı, çok noktalı ve tek düze çaprazlamalardır (Okkalı, 2013).



Şekil 2.2 : Tek noktalı çaprazlama

Mutasyon işlemi, bireylerin özelliklerinin birbirlerine benzemeye başlaması durumunda bireyin genlerini rastgele değiştirme işlemidir (Tuncer, 2007). Çözümler uzayında yeni alanların keşfedilmesi yani çeşitliliğin artmasını sağlayan mutasyon işlemi (Yıldırım ve diğ., 2002), mutasyon oranına göre yapılır. Mutasyon oranı genellikle birin bireyin gen uzunluğuna bölümü olarak kullanılır. Bir bireyin gen sayısının 100 olduğu varsayılırsa, mutasyon oranı 0.01 yani mutasyona uğrama olasılığı %1'dir (Er ve ark, 2005).

2.4 Karınca Koloni Algoritması (KKA)

Sezgisel algoritmalardan biri olan KKA, gerçek karıncaların yiyecek arama davranışından esinlenen Dorigo ve arkadaşları tarafından 1991 yılında önerilmiştir (Dorigo ve diğ., 1991). Karıncaların diğer canlılar gibi etrafı yeterince görüp izleme gibi özellikleri yoktur. Ancak yine de karıncalar yuvalarından, keşfettikleri bir yiyecek kaynağına en kısa yolu bulurlar. Karıncaların en kısa yolu bulmalarını sağlayan en önemli maddenin feromon maddesi olduğu bilinmektedir (Gangal, 2015). Feromon maddesi kokulu bir kimyasaldır. Yiyecek kaynağını arayan karınca, feromonu takip ederek yiyecek kaynağına ulaşır ve aynı kokuyu takip ederek yiyeceği yuvasına taşır. Bu işlem sırasında ardından gelen karıncalar da bu yolu takip edebilsin diye feromon bırakır. Takipçi karınca sayısı yeterli değil ise, buharlaşma yoluyla bu feromon izleri silinir (Demiray, 2008). Tam olarak karınca kolonilerinin davranışlarını modellenmesi yerine, yapay karınca modellerinin bir optimizasyon aracı olarak değerlendirilmesinden dolayı, gerçek karınca davranışlarından biraz farklıdır. Örneğin yapay karıncalar tamamen kör değil ve belirli bir hafızaya sahiptirler (Öztürk, 2012). Popülasyon temelli bir yaklaşım olan KKA, ilk olarak Gezgin Satıcı Problemi üzerinde uygulanmıştır (Keskintürk ve Söyler, 2006).

$\tau_{ij}(t)$, t zamanında kenar(i,j) üzerindeki iz yoğunluğu (gerçek arılarda feromon), ρ buharlaşma katsayısıdır. t+1 anındaki iz yoğunluğu eşitlik (2.2) ile hesaplanır (Dorigo ve diğ., 1991).

$$\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \Delta\tau_{ij}(t,t+1) \qquad \textbf{(2.2)}$$

$\Delta\tau^{k}_{ij}(t,t+1)$, t ve t+1 zaman aralığında k. karıncanın kenar(i,j) uzunluk birimi başına koyduğu iz yoğunluğudur. Kenar(i,j) üzerine birim zamandaki iz yoğunluğu eşitlik (2.3) verilmiştir (Dorigo ve diğ., 1991).

$$\Delta\tau_{ij}(t,t+1) = \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t,t+1) \qquad \textbf{(2.3)}$$

k. karıncanın i düğümünden j düğümüne geçiş olasılığı eşitlik (2.4) ile hesaplanır. η_{ij} seçilebilirlik parametresi $1/d_{ij}$ olarak tanımlanır. α ve β kullanıcıya geçiş olasılığını hesaplarken iz yoğunluğu ile seçilebilirlik arasındaki önemi kontrol etmemizi sağlar (Dorigo ve diğ., 1991).

$$p_{it} = \frac{\left[\tau_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta}}{\sum\left[\tau_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta}} \quad \textbf{(2.4)}$$

Karınca yoğunluk modeli, bir karıncanın *i* düğümünden *j* düğümüne gidişinde bu düğümler arasındaki (*i,j*) hattına, birim uzunluk başına Q_1 miktarınca iz bırakmasıdır. Karınca miktar modeli ise *i* düğümünden *j* düğümüne giden karınca Q_2/d_{ij} miktarınca birim uzunluk başına iz bırakmasıdır. Karınca yoğunluk modeli eşitlik (2.5), karınca miktar modeli eşitklik (2.6) ile hesaplanmıştır (Karaboğa, 2004).

$$\Delta\tau_{ij}(t,t+1) = \Delta\tau_{ij}(t,t+1) + Q_1 \quad \textbf{(2.5)}$$

$$\Delta\tau_{ij}(t,t+1) = \Delta\tau_{ij}(t,t+1) + Q_2 / d_{ij} \quad \textbf{(2.6)}$$

KKA temel adımları (Dorigo ve diğ., 1991);

1- Başla

   a. Her düğüme $b_i(t)$ karıncaları yerleştir. {$b_i(t)$: *t* zamanında *i* düğümündeki karınca sayısı}

   b. $\Delta\tau_{ij}(t,t+1)$ değerini sıfırla

2- Tabu listesi dolana kadar bu adımları tekrarla

   a. For i=1 to düğüm_sayısı do

      i. For *k*=1 to $b_i(t)$ do

         1. Eşitlik (2.3) ile $P_{ij}(t)$ değerini hesapla
         2. $P_{ij}(t)$ ihtimaline göre *j* düğümünü seç
         3. *k.* karıncayı *j* düğümüne taşı ve yeni değerler ile $b_i(t+1)$ hesapla
         4. *j* düğümünü $tabu_k(s)$ listesine ekle
         5. $\Delta\tau_{ij}(t,t+1)$ değerini eşitlik (2.5) veya (2.6) göre hesapla

   b. Eşitlik (2.2) ile her kenara(*i,j*) ait $\tau_{ij}(t+1)$ değerini hesapla

3- Şuana kadar bulunan en kısa değeri sakla

4- Döngü sayısına (NC) ulaşıldıysa işlemi bitir değilse devam et

   a. Tabu listesini boşalt

   b. Tüm düğümlere belirli miktarda karınca yerleştir

   c. $\Delta\tau_{ij}(t,t+1)$ değerini sıfırla

   d. 2. adıma git

3. YAPAY ARI KOLONİSİ ALGORİTMASI

3.1 Sürü Zekası

Kendi kendine organize olan ve birbirleri ile etkileşime giren akıllı birey topluluklarını inceleyen bilim dalı olan sürü zeka (Kadioğlu ve diğ., 2010), özellikle canlı bireylerin çevreleri ve birbirleri ile etkileşimi sonucu ortaya çıkan davranışların modellenmesi ve araştırılması ile ilgilenen yapay zeka disiplinidir (Öztürk ve diğ., 2014). Karıncalar, kuşlar, arılar ve balık sürüleri, sürü zekasına dayanan optimizasyon tekniklerinden sadece birkaçıdır (Tokaş ve Akdağlı, 2012).

3.2 Doğadaki Arılar

Doğadaki en ilginç sürülerden biri de bal arılarıdır. Sosyal bir çevrede yaşayan arılar, kendi kendilerine içgüdüsel olarak bir düzen kurar ve bu düzen içinde hareket ederler (Kartal, 2015).

Arılar koloniler olarak yaşayan sosyal böceklerdir. Bir kolonide Erkek arılar (drones), Kraliçe arı (queen) ve İşçi arılar (workers) olmak üzere üç çeşit arı bulunmaktadır (Karaboğa ve Akay, 2009).

Kraliçe arı: Kovan içinde yumurtlama yeteneğine sahip tek arı olan kraliçe arı, genellikle hayatında bir kez çiftleşir ve çiftleşmede depolanan spermler sayesinde iki veya daha fazla yıl döllenir (Karaboğa ve Akay, 2009). Arı sayısı azaldığında üremeye başlar, belirli bir eşik değerine ulaştığında üremeye ara verir. Üreme yeteneğini kaybedince kraliçenin kızlarından bir tanesi kraliçe arı olarak seçilir (Kıran, 2010). Sağlıklı bir kraliçe arı günde ortalama 2000'e yakın, yılda 175.000-200.000 arasında yumurta üretmektedir (Karaboğa ve Akay, 2009). Kovan içindeki arıları, salgıladığı maddeler sayesinde işaretleyen kraliçe arı, bu işaretleme ile kovana girebilecek olan yabancı arıları tespit etmektedir (Kartal, 2015).

Erkek arılar: Kolonide sadece üreme işinden sorumlu olan erkek arılar 6 aydan fazla yaşayamazlar ve kraliçe ile çiftleşmesi sonucunda ölürler (Kartal, 2015; Kıran, 2010).

İşçi arılar: Koloninin sağlıklı bir yaşam sürdürmesinde önemli göreve sahip olan işçi arılar, besinlerin toplanması ve saklanması, kovanın ölü arı ve molozlardan temizlenmesi, havalandırılması ve güvenliğinin sağlanması gibi görevlere sahiptir (Kartal, 2015). İşçi arılar, dişi arı olmalarına rağmen üreme dışındaki tüm işlerden sorumludur (Kıran, 2010). İşçi arılar yaz aylarında 6 hafta, kış aylarında ise 4 ile 9 ay arasında yaşarlar (Karaboğa ve Akay, 2009).

3.3 Arıların Besin Arama Davranışları

Arı kolonisinde yaşamın devam edebilmesi için en temel ihtiyaç besindir. Kovan içinde biriktirilen besinler, ortamda bulanabilecek besin kaynakları ve arıların etkileşimi besin arama sürecinde önemli etkenlerdir (Küçüksille ve Tokmak, 2011). Arılar kendi arasında merkezi bir otorite olmadan iş dağılımı gerçekleştirebildikleri için kendi kendine organize olabilmektedirler. Arılar arasında bir iş bölümü olduğunda kolonide yapılacak işler o iş için özelleşmiş arılar tarafından yapılır (Akay, 2009).

Kovandaki çoğu işten işçi arılar sorumludur ve kovandaki düzen, işçi arıların sorumluluklarını yerine getirmesiyle sağlanır (Öztürk, 2011). Bal arıları yiyecek arama davranışlarını ne tür dış etkenlerin (kuyruk dansından gelen konum bilgisi, koku, başka arıların kaynağa yönelmiş olmaları gibi) ve iç etkenlerin (hafızada tutulan kaynak ya da koku gibi) etkilediğini inceleyen birçok çalışma yapılmıştır (Akay, 2009). Arılar yiyecek kaynaklarına nektar, bal ya da polen bulmak için gitmektedirler. Gidecekleri yiyecek kaynağının değeri, nektar yoğunluğu, çeşidi, nektrarların çıkarılma kolaylığı, yuvaya yakınlığı gibi birçok etkene bağlıdır (Akyol ve Alataş., 2012). Arıların yiyecek arama süreci kovandan ayrılması ile başlar. Rastgele yiyecek araştırması yapan arılar buldukları besin kaynaklarında yiyecek miktarının azalması ile diğer arılardan aldıkları bilgi ile başka besin kaynaklarına ya da rastgele bir besin kaynağına yönelirler (Küçüksille ve Tokmak, 2011).

Yiyecek arama modelinde Tereshko ve Loengarow'a (2005) göre yiyecek kaynakları, görevi belli işçi arılar ve görevi belirsiz işçi arılar olmak üzere üç temel bileşen vardır.

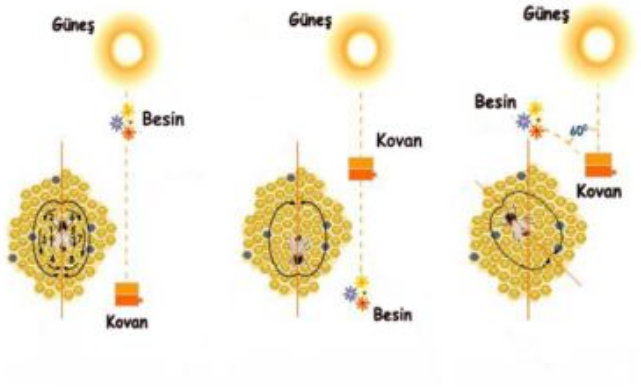
Yiyecek Kaynakları: Arılar besin ihtiyaçlarını karşılamak için gerekli olan nektar ve polenleri bulmak için yiyecek kaynaklarına giderler. Yiyecek kaynağının değeri, zenginliği, yuvaya yakınlığı, konsantrasyonu, çıkarılma kolaylığı gibi birçok faktöre bağlıdır.

Görevi Belirli İşçi Arılar: İşçi arılar ziyaret ettikleri kaynakların yuvadan uzaklığı, yönü ve değeri hakkında bilgileri diğer arılara iletirler. Besin kaynağındaki nektarları kovana taşıyan arılar nektar bittiğinde görevi belirsiz işçi arı olurlar.

Görevi Belirsiz İşçi Arılar: Yeni yiyecek kaynağı arayışında olan görevi belirsiz işçi arılar iki çeşittir; yuvada bekleyen ve görevli arılardan aldıkları bilgilerle kaynağa giden gözcü arılar ve yuvanın çevresinde 14km yarıçapında yeni besin kaynağı arayan kaşif arılardır.

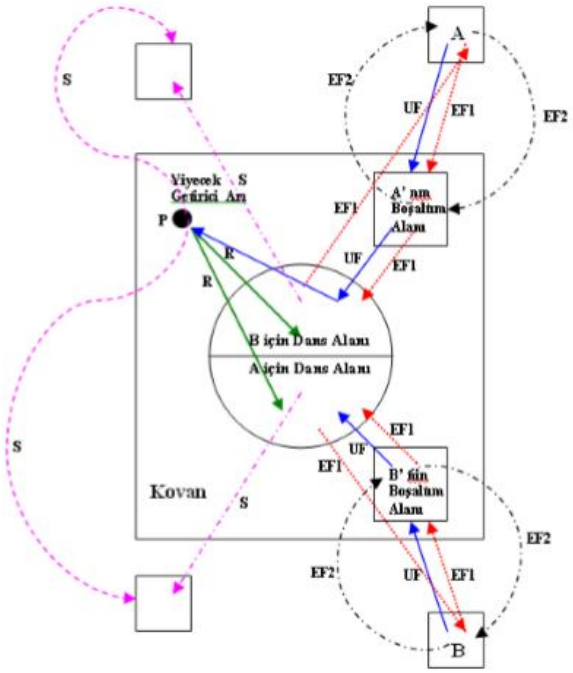
Besin arayıcı arılar, kaynak hakkında bilgiyi diğer arılara bildirmek üzere dans adı verilen özel bir hareket yaparlar (Kartal, 2015). Kaynağın kovana olan mesafesine göre dairesel dans (round dance), kuyruk dansı (waggle dance) ve titreme dansı (tremble dance) gibi çeşitli danslar mevcuttur (Akay, 2009). Besin kaynağının kovana uzaklığı 100 metreye kadar dairesel dansı (Şekil 3.1), 100 metreden sonra kuyruk dansını (Şekil 3.2) kovandaki diğer bireyleri bilgilendirmek için kullanır (Korkmaz, 2015). Zengin bir nektar kaynağını bulduğunu, ancak kovana işlenebileceğinden fazla nektar geldiğini bundan dolayı nektarı işleme görevine geçmek istediğini belirtmek için titreme dansını kullanır (Akay, 2009).

Şekil 3.1 : Arıların dairesel dansı (Korkmaz, 2015)



Şekil 3.2 : Arıların kuyruk dansı (Korkmaz, 2015)

Yiyecek arayıcı arıların davranışları Şekil 3.3'de gösterilmiştir. Kovan etrafındaki arılar kaynağın yeri hakkında bilgi sahibi değildirler. Kaynağa ulaşmak için *S* ile gösterilmiş kaşif arı olarak kaynağı arayabilir veya *R* ile gösterilmiş kuyruk danslarını izleyen gözcü arı olarak kaynağı bulabilirler. Arılar kaynaktaki nektarları kovana getirdikten sonra kaynağı bırakarak *UF* ile gösterilmiş izleyici olabilir, gittiği kaynağı dans ederek diğer arılara gösteren *EF1* ile gösterilmiş arı olabilir veya diğer arıları yönlendirmeden kaynağa gidebilen *EF2* ile gösterilmiş arı olabilir (Akay, 2009).



Şekil 3.3 : Yiyecek arama davranışları (Akay, 2009)

Bir birey komşuluk bilgilerini kullanarak sürüdeki diğer bireyler ile etkileşim kurar. Banabeau ve diğ. (1999) göre kendi kendine organize olabilmenin dört temel özelliği vardır;

Pozitif Geri Besleme (positive feedback): Pozitif geri besleme sürüdeki bireyleri uygun yapıların oluşturulmasına teşvik eden bir başparmak kuralıdır. Arıların dansı buna örnek olarak verilebilir.

Negatif Geri Besleme (negative feedback): Pozitif geri beslemeyi dengelemek için kullanılan negatif geri besleme, doygunluk, tükenme veya rekabet şeklini alabilir.

Rastgelelik-Salınım (fluctuation): Sürü içindeki rastgele olarak yapılan gezinme veya görev değişimleri gibi rastgele davranışlar bazı durumlarda yeni çözümlerin üretilmesine katkı sağlayabilir.

Etkileşim (multiple interaction): Etkileşim halinde bulunan sürüdeki bireyler toplu hareket edip birbirlerini etkilerler. Birbirlerinden etkilenen bireyler yaptıkları davranışların sonuçlarını paylaşarak daha iyi davranışlar ortaya çıkmasını sağlarlar. Etkileşim, kendi kendine organize olabilen bireylerin en önemli özelliğidir.

3.4 Yapay Arı Kolonisi Algoritmasının Temel Özellikleri

YAK algoritması sürü zekasına dayalı bir algoritmadır. Oldukça az kontrol parametresine sahip, basit ve esnektir. Gerçek yiyecek arayıcı arıların davranışlarına yakın şekilde simüle eder. Nümerik problemler için geliştirilen algoritma, ayrık problemler içinde kullanılabilir. Kaşif arıları küresel, işçi ve gözcü arılar bölgesel araştırma özelliğine sahiptirler ve iki araştırma özelliği paralel yürütülmektedir (Akay, 2009).

3.5 Yapay Arı Kolonisi Algoritmasının Temel Adımları

YAK algoritması rastgele besin kaynağı üretildikten sonra üç temel adımın tekrarlanmasıyla gerçekleştirilir. Bu adımlar şu şekildedir;

- rastgele besin kaynağının üretilmesi
- while (sonlandırma koşulu)
    - işçi arıların besin kaynağına gönderilmesi

- gözcü arıların besin besin kaynağına gönderilmesi
- besin kaynağını terk etme ve kaşif arıların üretilmesi

- do

3.5.1 Rastgele besin kaynağı üretme

Arama uzayını yiyecek kaynaklarını içeren kovan çevresi olarak düşünürsek algoritma arama uzayındaki çözümlere karşılık gelen rastgele yiyecek kaynağı yerleri üretmektedir (Özcan, 2016). Kovan çevresinde popülasyon adedine göre parametrelerin alt ve üst sınırları içerisinde yiyecek kaynağı üretilmesi için eşitlik (3.1) kullanılmaktadır (Çelik, 2015).

$$x_{ij} = x_j^{\min} + rand(0,1)(x_j^{mak} - x_j^{\min}) \quad \textbf{(3.1)}$$

Problemdeki besin kaynağının sayısı *SN*, optimize edilecek parametre sayısı *D* ile gösterilmektedir. Denklemdeki *i* değeri işlem yapılacak besin kaynağını (*1...SN*), *j* değeri ise işlem yapılacak parametreyi (*1...D*) ifade etmektedir. x_j^{mak} *j.* parametrenin üst sınırı, x_j^{min} ise *j.* parametrenin alt sınır değeridir (Akay, 2009).

3.5.2 İşçi arının besin kaynağına gönderilmesi

Her besin kaynağına ait bir işçi arı görevlendirilmektedir. İşçi arılar görevli oldukları besin kaynaklarının komşuluğunda yeni bir besin kaynağı eşitlik (3.2) ile belirlenir (Akay, 2009). Elde edilen besin kalitesi önceki besin kalitesinden daha iyi ise bu komşuluğu hafızaya alır ve önceki besin kaynağıyla değiştirir (Kartal, 2015) ve geliştirme sayacını sıfırlar, aksi durumda geliştirme sayacını bir arttırır (Çelik, 2015).

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad \textbf{(3.2)}$$

x_{ij} işçi arının görevli olduğu kaynağı, x_{kj} rastgele seçilen kaynağı, φ_{ij} ise [-1,1] aralığında rastgele seçilen bir değeri ifade etmektedir. her x_i besin kaynağı için [*1,D*] aralığında rastgele seçilen *j* parametresinin değiştirilmesi sonucu v_i kaynağı hesaplanır. Hesaplanan v_{ij} değeri önceden belirlenen alt ve üst sınırlarının dışında bir değer elde edilmiş ise eşitlik (3.3) kullanılarak ötelenmektedir (Çelik, 2015).

$$v_{ij} = \begin{cases} x_j^{\min}, & v_{ij} < x_j^{\min} \\ v_{ij}, & x_j^{\min} \le v_{ij} \le x_j^{mak} \\ x_j^{mak}, & v_{ij} < x_j^{mak} \end{cases} \quad \textbf{(3.3)}$$

Sınırlar dahilinde üretilen v_i kaynağı yeni bir kaynağı temsil etmekte ve bunun kalitesi hesaplanarak bir uygunluk değeri eşitlik (3.4) ile atanmaktadır. Denklemdeki f_i, v_i kaynağının yeni çözüm maliyetidir (Özcan, 2016).

$$fitness_i = \begin{cases} 1/(1+f_i), & f_i \ge 0 \\ 1/abs(f_i), & f_i < 0 \end{cases} \quad \textbf{(3.4)}$$

3.5.3 Gözcü arının besin kaynağına gönderilmesi

Kovanda bulunan gözcü arı sayısı besin sayısına eşittir. İşçi arıların aksine gözcü arılar istedikleri besin bölgesine gitmektedirler (Ünsal, 2014). Tüm işçi arılar bir çevrimde araştırmalarını tamamladıktan sonra kovana dönüp buldukları kaynakların nektar miktarları ile ilgili gözcü arılara bilgi aktarırlar (Akay, 2009). Gözcü arılar gelen bilgileri değerlendirir ve kaynakların kalitesiyle orantılı bir olasılıkla kaynağı seçer. Uygunluk değerine göre yapılan bu seçim rulet tekerleği, sıralama tabanlı, stokastik örnekleme, turnuva yöntemi veya diğer seleksiyon şemalarından herhangi biri ile gerçekleştirilebilir. Temel YAK algoritması bu seçme işlemini rulet tekerleği yöntemine göre yapmaktadır (Çelik, 2015; Akay, 2009).

Rulet tekerleği: Topluluktaki bireylerin seçilme işlemi uygunluk değerine göre yapılmaktadır. Bireylerin seçilme olasılığı eşitlik (3.5)'te gösterildiği gibi bireyin uygunluk değerinin, tüm bireylerin uygunluk değerlerinin toplamına bölümü ile elde edilir. Rulet tekerleği seçiminde, uygunluk değerleri arasındaki farklılığın büyük olması aynı çözümlerin etrafında dolanma sorununa yol açabilir (Cevre ve diğ., 2007).

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad \textbf{(3.5)}$$

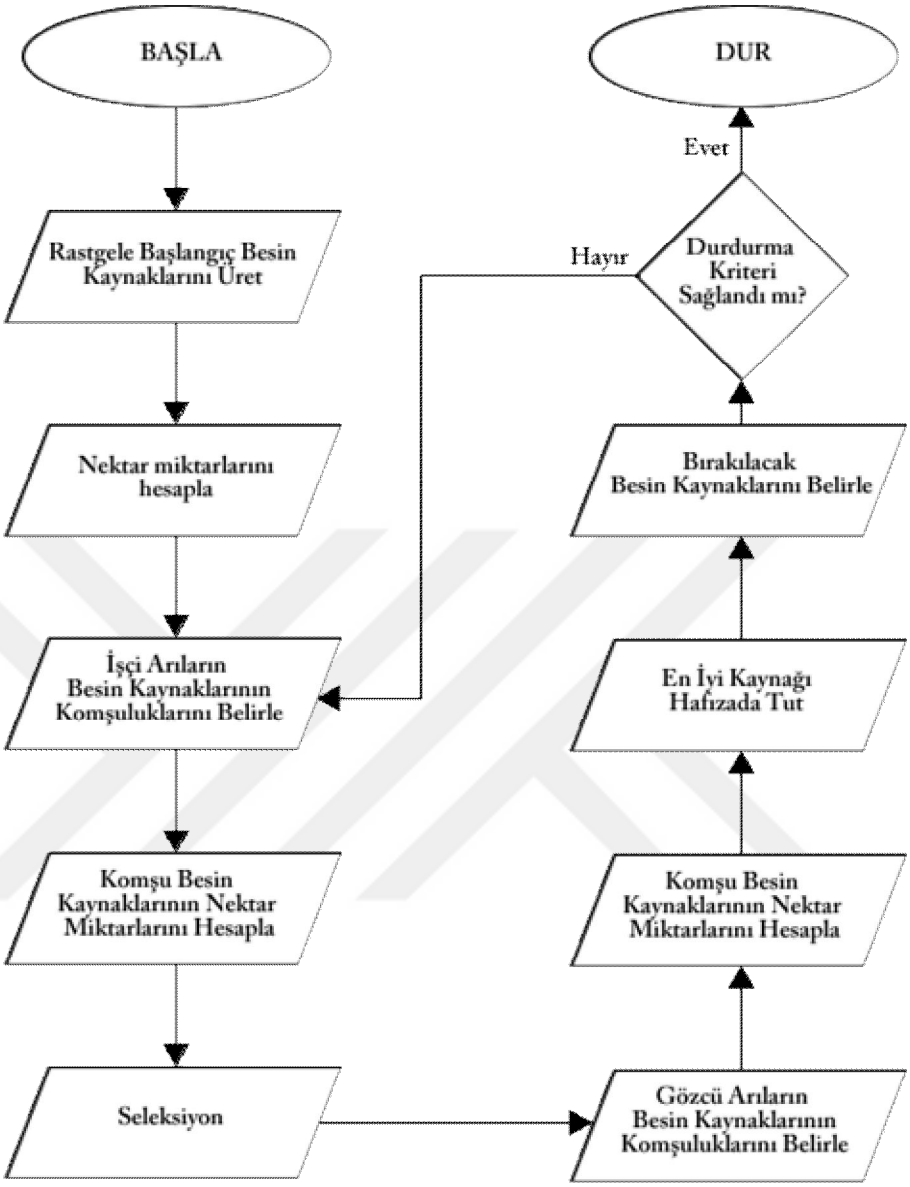
$fitness_i$ *i.* kaynağın kalitesini, *SN* gözcü arı sayısını göstermektedir. Bu olasılık hesaplama işleminde kaynaktaki nektar miktarı arttıkça uygunluk değeri artacaktır ve kaynak bölgesini seçecek gözcü arı sayısıda artacaktır. Bu özellik YAK'ın pozitif geribesleme özelliğine karşılık gelmektedir (Akay, 2009).

Bu aşamada her kaynak için [0,1] arasında rastgele bir değer üretilir. Bu değer olasılık değerinden küçük ise gözcü arılar işçi arılar gibi eşitlik (3.2)'yi kullanarak bu kaynağa komşu bir çözüm üretir. Bu komşu çözümün kalitesini değerlendirir ve eski çözümden daha iyi ise değerleri değiştirir ve geliştirme sayacını sıfırlar. Eski çözüm daha iyi ise geliştirme sayacını bir arttırarak devam eder (Çelik, 2015).

3.5.4 Besin kaynağını terk etme ve kaşif arının üretilmesi

Bir arının bir kaynaktan faydalanıp faydalanmadığı, o kaynağın nektar miktarının tükenip tükenmediğini geliştirme sayacı sayesinde öğrenilebilir. Bir çevrim sonunda tüm işçi ve gözcü arıların geliştirme sayaçları kontrol edilir. Sayaç değeri belli limit değerin üzerinde ise o kaynaktaki işçi veya gözcü arının kaşif arı olması anlamına gelmektedir. Görevli arı kaşif arı olduktan sonra eşitlik (3.1) kullanılarak rastgele bir çözüm üretir (Akay, 2009).

3.6 Yapay Arı Kolonisi Algoritmasının Akış Şeması



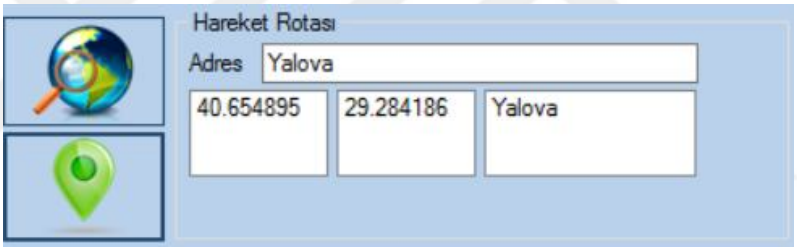
Şekil 3.4 : YAK algoritması akış şeması

4. GELİŞTİRİLEN UYGULAMA

4.1 Uygulama Arayüzü

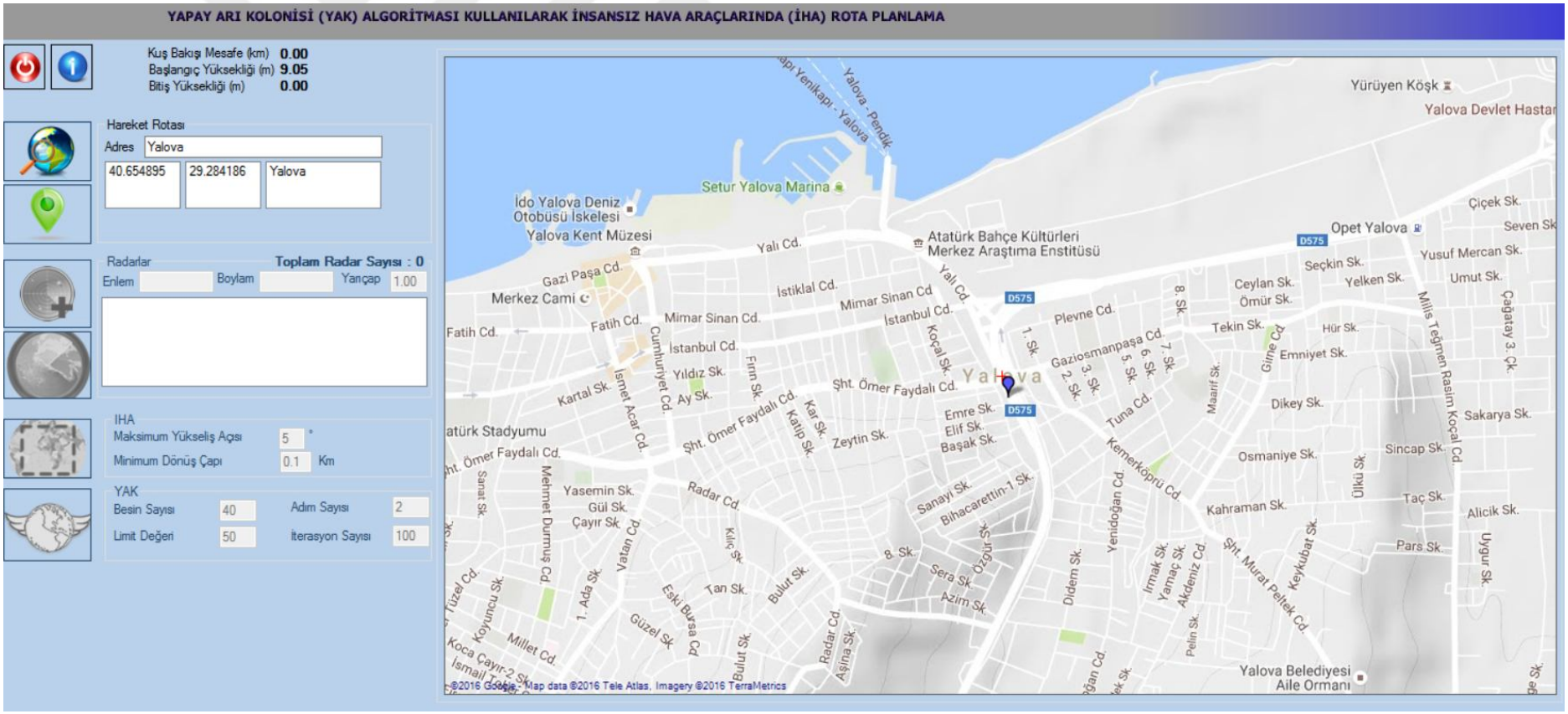
İHA'larda rota planlama için geliştirilen kullanıcı erişimli arayüz Şekil 4.2'de gösterilmiştir. Uygulama ile harita üzerinde çevrimiçi işlemler yapabilmek için gMapControl eklentisi kullanılmıştır (Url-1). Bu eklenti ile Google haritası kullanılarak gerçek zamanlı bilgilerden faydalanılmıştır. Hazırlanan arayüz üç aşamadan oluşmaktadır.

Birinci aşamada; İHA'nın başlangıç ve hedef adresleri arayüz ekranından girilmektedir (Şekil 4.1). Adres bilgisine göre koordinatlar bulunarak, harita üzerine işaretlenmektedir.



Şekil 4.1 : Başlangıç ve bitiş noktası giriş ekranı

Arayüz ekranından girilen noktaların koordinat bilgileri, *http://maps.googleapis.com* web adresindeki JSon verisi alınarak bulunmaktadır. Web adresindeki JSon verisini almak için gerekli C# kod bloğu Şekil 4.3'de gösterilmiştir.



Şekil 4.2 : YAK ile rota planlama için kullanıcı arayüzü ekranı

```
public double[] Enlem_Boylam_Bulma(string adres)
{
   double[] Koordinatlar=new double[2];
   string Sunucu="http://maps.googleapis.com/maps/api/"
   Sunucu+="geocode/json?sensor=true&address="+adres;
   string Veri;
   using (WebClient Cevap=new WebClient())
      Veri=cevap.DownloadString(Sunucu);
   JObject J_Veri=JObject.Parse(Veri);
   JArray J_Koordinat=(JArray)J_Veri["results"];
   Koordinatlar[0]=Convert.ToDouble(J_Koordinat[0]["geometry"]["location"]["lng"]);
   Koordinatlar[1]=Convert.ToDouble(J_Koordinat[0]["geometry"]["location"]["lat"]);
   return Koordinatlar;
}
```

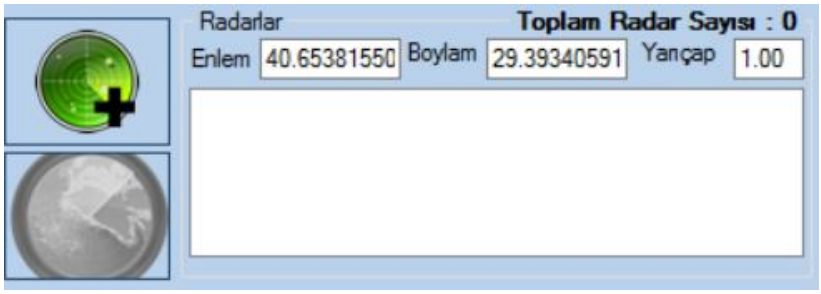
Şekil 4.3 : Adres bilgisine göre koordinat bilgisi bulma

Elde edilen koordinat bilgilerine göre arayüzde bulunan haritaya işaretçi eklemek için Şekil 4.4'teki C# kod bloğu kullanılmıştır.

```
public void Isaretci_Ekle(double Enlem, double Boylam)
{
   GMarkerGoogleType Tip=GMarkerGoogleType.blue_small;
   GMapOverlay Isaretci=new GMapOverlay("isaretci");
   GMarkerGoogle Isaret=new GMarkerGoogle(new PointLatLng(Enlem,Boylam),Tip);
   Isaretci.Markers.Add(Isaret);
   gMapControl1.Overlays.Add(Isaretci);
}
```

Şekil 4.4 : Enlem-boylam bilgisine göre işaretçi ekleme

İkinci aşamada; sanal engellerin (radar, tehlikeli bölge, vb.) Şekil 4.5'te gösterildiği gibi harita üzerine enlem-boylam bilgileri el ile girelerek ya da fare yardımıyla işaretlenerek oluşturulması sağlanmaktadır. Eklenecek olan sanal engelin yarıçap bilgisi girilerek ne kadar genişlikte bir alana uygulandığı belirtilmekte olup harita üzerinde gösterilmektedir.



Şekil 4.5 : Radarların harita üzerine eklenmesi

C# kod bloğunda enlem, boylam bilgisi girilen radarın etrafında yarıçap mesafesinde tüm açılarda bir nokta olmak üzere 360 adet nokta oluşturulmuş ve bu noktalar birleştirilerek sanal engel elde edilmiştir. C# gMapControl eklentisi üzerinde daire çizim metodu olmadığından bu şekilde bir yöntem geliştirilmiştir. Dünyanın geoit şeklinden dolayı elde edilen açılardaki koordinatlar yerine o açıda yarıçap uzaklığındaki noktaların koordinatları kullanılmıştır.

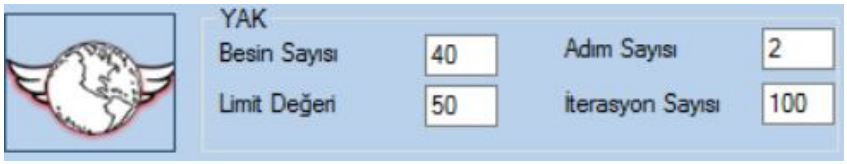
Üçüncü aşamada; İHA ve YAK için gerekli parametre ayarları yapılmaktadır. Kullanıcı arayüzünde İHA için yapılacak ayarlar Şekil 4.6'da, YAK için yapılacak ayarlar Şekil 4.7'de gösterilmektedir.

İHA ayarlar bölümündeki Maksimum Yükseliş Açısı, İHA'nın dikey eksende yapacağı açıyı (yunuslama) belirtmektedir. Her İHA'nın yükseliş açısı farklı olduğundan, kullanılacak olan İHA'ya ait bu açının kullanıcı tarafından girilmesi gerekmektedir. Minimum Dönüş Çapı ile İHA'nın kendi etrafında yapacağı minimum dairesel hareketin çap bilgisini belirtilmektedir.

Şekil 4.6 : İHA ayarları

YAK ayarlar bölümünde Besin Sayısı aday rota sayısını, Adım Sayısı bağlantı koordinat nokta sayısını, Limit Değeri işçi ve gözcü arıların kaşif arı olma kriterini,

İterasyon Sayısı YAK algoritmasının tekrarlama sayısını ifade etmektedir. Besin sayısı aynı zamanda YAK algoritmasındaki işçi ve gözcü arı sayılarınada eşittir. Adım sayısı kullanıcı tarafından girilen bir değerden başlayabildiği gibi, herhangi bir değer girilmediği takdirde algoritma tarafından dinamik olarak ta belirlenebilmektedir. Limit değeri geliştirme sayacına eşit olduğunda, ait olduğu işçi veya gözcü arı kaşif arıya dönüşecektir.



Şekil 4.7 : YAK algoritması ayarları

Bu üç aşama gerçekleştikten sonra uygulama, girilen kriterlere göre en uygun rotayı bularak harita üzerinde göstermekte ve kullanıcı için bir rapor oluşturulmaktadır. Bu raporda, İHA ve YAK ayarları, başlama ve bitiş koordinat noktaları, algoritma ile bulunan bağlantı koordinat noktaları ve sanal engellerin koordinatları bulunmaktadır. Rota üzerindeki coğrafi engellerin koordinatları, yükseklikleri ve İHA'nın kendi etrafında dönerek ne kadar tur atması gerektiği ile ilgili bilgiler de raporda kullanıcıya sunulmaktadır.

4.2 Engeller

4.2.1 Coğrafi engeller

İHA'ların rota üzerinde karşılaşabileceği birçok coğrafi engel vardır. Bunlar dağ ve tepe gibi yerleri değişmeyen engellerdir. Bu çalışmada Google Maps haritası yardımıyla gerçek koordinat ve yükseklik verileri kullanılmıştır.

Coğrafi engellerin tespiti için yükseklik matrisi oluşturulmuştur. Yükseklik verilerinden elde edilen eğim bilgilerine göre İHA'nın maksimum yükseliş açısının aşamayacağı noktalar belirlenmekte ve bu noktalarda İHA minimum dönüş çapı bilgisi kullanılarak kendi etrafında dönerek yükselmesi sağlanmaktadır. İHA'nın bu engelleri aşması için eşitlik (4.1) kullanılmaktadır. Eşitlikteki r değeri İHA'nın minimum dönüş yarıçapını, α ise maksimum yükseliş açısını ifade etmektedir. Coğrafi engelin yüksekliği bilindiğinden, h değeri ile İHA'nın kendi etrafında kaç tur dönmesi gerektiği tespit edilmektedir.

$$h = 2\pi r \times \tan(\alpha) \quad \textbf{(4.1)}$$

4.2.2 Sanal engeller

İHA'lar rota üzerinde coğrafi engellerin dışında radar ve tehlikeli bölge gibi farklı engellerle de karşılaşabilmektedir. Bu engeller için geliştirilen uygulama üzerinde sanal engeller tanımlanabilmektedir. Uygulamada sanal engeller, radar olarak kabul edilmiştir.

Örneğin $A(x_1,y_1)$ ve $B(x_2,y_2)$ noktaları arasındaki doğrunun $O(x_3,y_3)$ merkezli r yarıçaplı bir sanal engele değip değmediğini kontrol etmek için aşağıdaki adımlar uygulanmıştır;

1. Eşitlik (4.2) kullanılarak iki noktası bilinen doğrunun denklemi bulunmaktadır. Denklemin uygulanması sonucunda $a=y_2-y_1$, $b=x_2-x_1$ ve $c=(y_1.x_2)-(y_2.x_1)$ olarak bulunmaktadır.
2. Sanal engelin merkez noktasının, doğru parçasına uzaklığı eşitlik (4.3) kullanılarak hesaplanmaktadır.
3. Eşitlik (4.3) ile elde edilen değer, sanal engelin yarıçap bilgisinden;
    a. Büyük ($d > r$) ise, rotanın sanal engele değmediğini,
    b. Küçük veya eşit ($d \leq r$) ise, rotanın engele en az bir noktada değdiğini göstermektedir.

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_2}{x - x_2} \quad \textbf{(4.2)}$$

$$d = \frac{\left|ax_3 + by_3 + c\right|}{\sqrt{a^2 + b^2}} \quad \textbf{(4.3)}$$

4.3 Yapay Arı Koloni Algoritmasının Uygulanması

4.3.1 Rastgele rota üretme

Rastgele rota üretilirken eşitlik (4.4) kullanılmıştır. Denklemdeki eşitlikte r=1…BS, i=0...AS+1, j=0,1'dir. BS besin sayısını, AS adım sayısını ve j ise enlem/boylam bilgisini ifade etmektedir. j'nin 0 olması enlem, 1 olması boylam olduğunu belirtmektedir. x_j^{min} ve x_j^{mak} enlem ve boylam bilgilerinin alt ve üst sınır değerleridir.

$$x_{rij} = x_j^{\min} + rand(0,1)(x_j^{mak} - x_j^{\min}) \tag{4.4}$$

4.3.2 İşçi arı safhası

Üretilen her rotaya bir işçi arı yerleştirilir. Her işçi arı eşitlik (4.5) ile bulunduğu rotanın komşuluğunda yeni bir rota oluşturur. Eşitlikteki φ_r [-1,1] aralığında rastgele üretilen bir değeri, x_r işçi arının görevli olduğu rotayı, x_k rastgele seçilen rotayı ifade etmektedir. r=k=1…BS, i=0…AS+1, j=0,1'dir.

$$v_{rij} = x_{rij} + \varphi_r (x_{rij} - x_{kij}) \tag{4.5}$$

Her r değeri için rastgele seçilen i ve j değerlerindeki rota bilgisinin değiştirilmesi sonucu v_r kaynağı hesaplanır. Elde edilen kaynağın değeri önceden belirlenen üst sınırın üstünde ise üst sınıra, alt sınırın altında ise alt sınıra eşitlenir. Rotanın amaç fonksiyon değeri eşitlik (4.6) ile hesaplanmaktadır.

$$f_r = d_r + E_r \tag{4.6}$$

Eşitlikteki f_r değeri r. rotanın amaç fonksiyon değerini, d_r değeri r. rotanın uzunluğunu, E_r değeri r. rota üzerindeki toplam engel sayısı ile engel maliyetinin çarpımını göstermektedir. GPS verileri üzerinden rotanın uzunluğu hesaplanırken dünyanın geometrik şekli göz önünde bulundurulmuştur ve bunun için Haversine (Montavont ve Noel, 2006) formülü kullanılmıştır. Haversine formülü enlem 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 km) olarak ifade edilmiştir (Montavont ve Noel, 2006).

$$h = haver\ \sin(\Delta_{enlem}) + \cos(enlem_1) \times \cos(enlem_2) \times haver\ \sin(\Delta_{boylam}) \quad \textbf{(4.7)}$$

$$haver\sin(\delta) = \sin^2(\frac{\delta}{2}) \quad \textbf{(4.8)}$$

$$d_r = 2 \times R \times \arcsin(\sqrt{h}) \quad \textbf{(4.9)}$$

Amaç fonksiyon değerine göre uygunluk değeri eşitlik (3.4) ile hesaplanır. Bulunan uygunluk değeri eski uygunluk değerinden daha iyi ise rota bilgileri yeni hali ile değiştirilir ve geliştirme sayacı sıfırlanır, uygunluk değeri eski uygunluk değerinden iyi değilse geliştirme sayacı bir artırılır.

4.3.3 Gözcü arı safhası

İşçi arı safhası tamamlandıktan sonra gözcü arılar eşitlik (3.5) kullanılarak rotaların olasılık değerlerini hesaplar ve bu olasılık değerlerine göre gidecekleri rotaları belirlerler. Her rotanın olasılık değeri, uygunluk değerinin toplam uygunluk değerine oranı ile bulunur. Her gözcü arı, işçi arı gibi eşitlik (4.5) ile bulunduğu rotanın komşuluğundan yeni bir rota oluşturur ve eşitlik (4.6) ile rotanın amaç fonksiyon değerini hesaplar. Amaç fonksiyon değerine göre eşitlik (3.4) ile hesaplanan uygunluk değeri, eski uygunluk değerinden daha iyi ise rota bilgileri yeni hali ile değiştirilir.

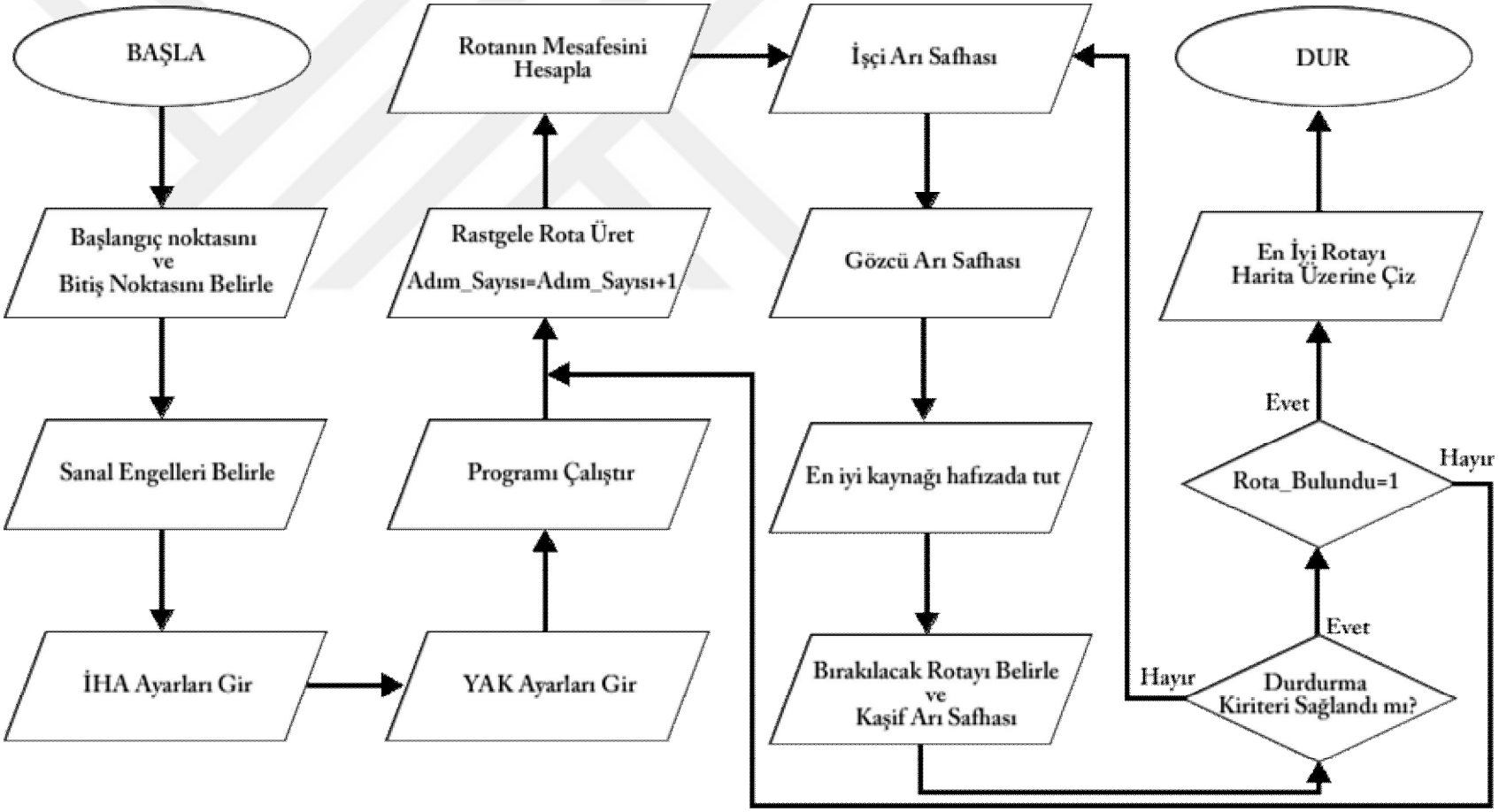
4.3.4 Kaşif arı safhası

Geliştirme sayacı değeri limit değerine ulaştığında görevli işçi veya gözcü arı kaşif arıya dönüşür. Bu aşamada eşitlik (4.4) kullanılarak kaşif arı rastgele bir rota üretir ve geliştirme sayacını sıfırlar.

Yapılan çalışmaya ait akış diyagramı Şekil 4.8'de gösterilmiştir. Uygulamadaki temel adımlar aşağıdaki gibidir.

- başlangıç ve bitiş noktalarını belirle
- sanal engelleri belirle
- İHA'ya ait kriterleri belirle
- YAK algoritmasının kriterlerini belirle
- Rastgele rota üret

- repeat
    - işçi arı safhası
    - gözcü arı safhası
    - kaşif arı safhası
- until iterasyon=maksimum çevrim sayısı
- harita üzerinde rotayı göster



Şekil 4.8 : Geliştirilen uygulamaya ait akış diyagramı

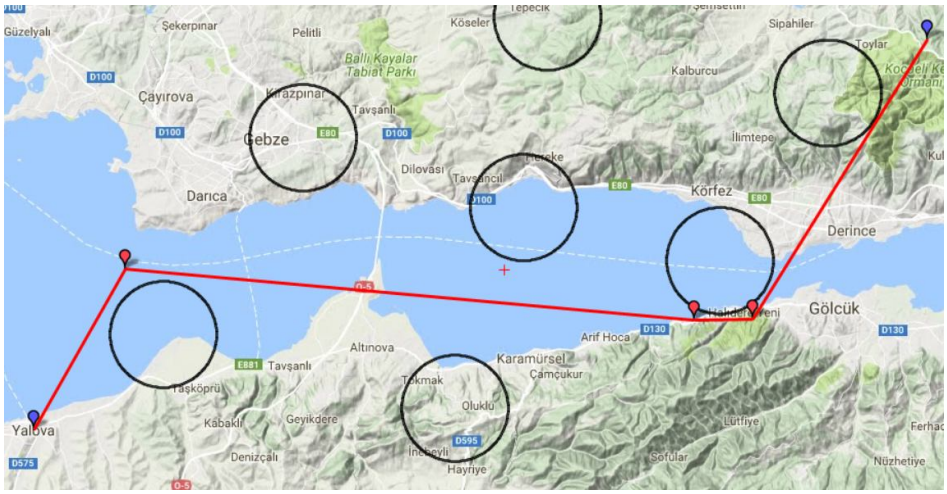
5. UYGULAMA ÖRNEKLERİ

Çalışmada, YAK algoritması ile rota planlama probleminin çözümü ve kullanıcı arayüzünün benzetim uygulamaları iki farklı senaryo ele alınmıştır. Benzetim çalışmalarında rota üzerinde görülen sarı işaretli noktalar İHA'nın aşamadığı coğrafi engelleri ifade etmektedir. İHA'nın rota üzerinde ilerlerken bu noktalara geldiği zaman kendi etrafında dönerek engeli aşması gerektiği öngörülmektedir.

Birinci senaryo; Yalova'dan Kocaeli'ye gidecek olan İHA için farklı yerlere yedi adet sanal engel eklenmiştir. İHA ve YAK algoritmasına ait kriterler Çizelge 5.1'de verilmiştir. Belirtilen kriterler ele alındığında bulunan en uygun rota Şekil 5.1'de gösterilmektedir. Üç bağlantı noktası kullanılarak ulaşılan hedefte şekilde görüldüğü gibi, başlangıç ve bitiş noktaları arasındaki engelli bölgelere girmeden elde edilen rotanın uzunluğu 64.602km olarak bulunmuştur.

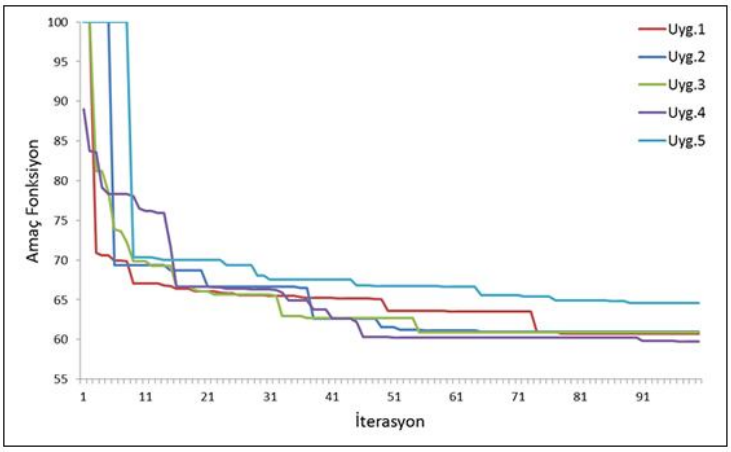
Çizelge 5.1 : Birinci senaryoya ait İHA ve YAK algoritması kriterleri

Kriterler	Değerler
Mak. yükseliş açısı	20°C
Min. dönüş çapı	0.1km
Besin sayısı	40
Limit değeri	50
İterasyon sayısı	100
Adım sayısı	3



Şekil 5.1 : Birinci senaryo için bulanan İHA rotası

Şekil 5.2'de birinci senaryo için YAK algoritmasının 5 kez çalıştırılması durumunda amaç fonksiyonun iterasyona bağlı olarak oluşan yakınsama grafiği gösterilmektedir. Grafikte de görüldüğü gibi uygulamalardan elde edilen değerlerin bir birine çok yakın sonuçlar verdiği görülmektedir. İterasyon sayısı ilerledikçe amaç fonksiyon değeri optimum çözüme yakınsamaktadır.

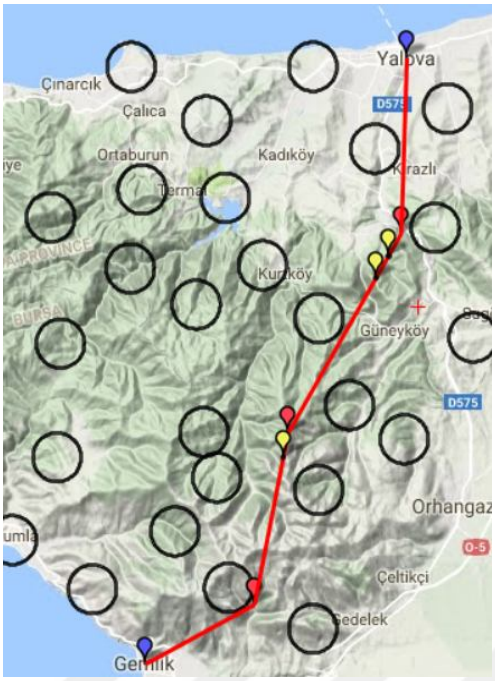


Şekil 5.2 : Birinci senaryoya ait amaç fonksiyonun iterasyona bağlı olarak yakınsama grafiği

İkinci senaryo; Yalova'dan Gemlik'e hareket edecek olan İHA için 26 adet sanal engel yerleştirilmiştir. İHA ve YAK algoritmasına ait kriterler Çizelge 5.2'de verilmiştir. Belirtilen kriterler ele alındığında bulunan en uygun rota Şekil 5.3'de gösterilmektedir. Üç bağlantı noktası kullanılarak ulaşılan hedefte İHA'nın coğrafi koşullardan dolayı kendi etrafında yükselerek aşması gereken üç farklı bölgenin bulunduğu görülmektedir. Şekilde görüldüğü gibi, başlangıç ve bitiş noktaları arasındaki engelli bölgelere girmeden elde edilen rotanın uzunluğu 27.289km olarak bulunmuştur.

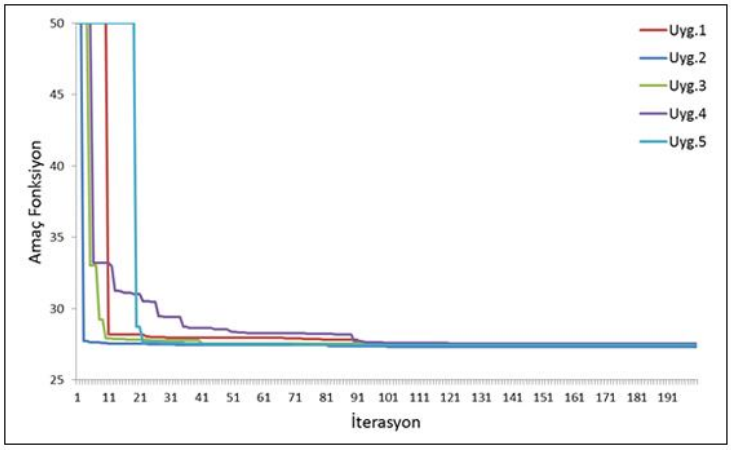
Çizelge 5.2 : İkinci senaryoya ait İHA ve YAK algoritması kriterleri

Kriterler	Değerler
Mak. yükseliş açısı	4°C
Min. dönüş çapı	0.1km
Besin sayısı	20
Limit değeri	50
İterasyon sayısı	200
Adım sayısı	3



Şekil 5.3 : İkinci senaryo için bulanan İHA rotası

Şekil 5.4'de ikinci senaryo için YAK algoritmasının 5 kez çalıştırılması durumunda amaç fonksiyonun iterasyona bağlı olarak oluşan yakınsama grafiği gösterilmektedir. Grafikte de görüldüğü gibi uygulamalardan elde edilen değerlerin bir birine çok yakın sonuçlar verdiği görülmektedir. İterasyon sayısı ilerledikçe amaç fonksiyon değeri optimum çözüme yakınsamaktadır.



Şekil 5.4 : İkinci senaryoya ait amaç fonksiyonun iterasyona bağlı olarak yakınsama grafiği

6. SONUÇ VE ÖNERİLER

Bu çalışmada, otonom İHA'lar için engelli ve tehlikeli bölgelerden sakınarak en uygun 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 tespit edilmekte ve bu bölgelerde İHA'nın kendi etrafında dönüp yükselerek coğrafi engeli aşması gerektiği belirlenmektedir. Yapılan farklı benzetim uygulamaları, İHA için rota planlama probleminde YAK algoritmasının başarılı sonuçlar verdiğini göstermektedir.

KAYNAKLAR

Akay, B., 2009: Nümerik Optimizasyon Problemlerinde Yapay Arı kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Doktora Tezi.

Akça, M.R., 2011:Yapay Arı Kolonisi Algoritması Kullanılarak Gezgin Satıcı Probleminin Türkiyedeki İl ve İlçe Merkezlerine Uygulanması, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi.

Akyol, S. ve Alataş, B., 2012: Güncel Sürü Zekası Optimizasyon Algoritmaları, Nevşehir Üniversitesi Fen Bilimleri Enstitüsü Dergisi,Cilt.1,sy.36-50

Akyürek, S., Yılmaz, M.A. ve Taşkıran, M., 2012: İnsansız Hava Araçları Muharebe Alanında ve Terörle Mücadelede Devrimsel Dönüşüm Raporu, Bilge Adamlar Stratejik Araştırmalar Merkezi, sy.1.

AlShawi, I.S., Yan, L., Pan, W. ve Luo, B., 2012: Lifetime Enhancement in Wireless Sensor Networks Using Fuzzy Approach and A-Star Algorithm, IEEE Sensors Journal, Vol.12,pp.3010-3018.

Al-Tameemi, H.L.H., 2014: Using Dijkstra aLgorithm in Calculating Alternative Shortest Paths for Public Transportation with Transfer and Walking CaseStudy: Ankara, Çankaya Üniversitesi.

Arıca, N., Cicibaş, H. ve Demir K.A., 2012: İnsansız Hava Araçları için Çok Kriterli Güzergah Planlama Modeli, Savunma Bilimleri Dergisi, sy.251-270.

Aydemir, H., 2014: İnsansız Hava Araçlarının Rotalama Problemi için Simülasyon Tabanlı Karar Destek Sistemi, Kara Harp Okulu Savunma Bilimleri Enstitüsü Hareket Araştırması ABD.

Aygün, S. ve Akçay, M., 2015: Matlab Paralel Hesaplama Aracı ile A* Algoritmasının RotaPlanlama İçin Analizi, 2.Genç Mühendisler Sempozyumu.

Baghel, P.K. ve Mishra A.K., 2016: A New Approach for a Car Navigation System, International Research Journal of Engineering and Technology, Vol.3,pp.377-381.

Bonabeau, E., Dorigo, M. ve Theraulaz, G., 1999: Swarm Intelligence: From Natural to Artificial Systems, New York, NY: Oxford University Press.

Cevre, U., Özkan, B. ve Aybars, U., 2007: Gezgin Satici Probleminin Genetik Algoritmalarla En İyilemesi ee Etkileşimli Olarak Internet Üzerinde Görselleştirilmesi, XII. Türkiye'de İnternet Konferansı, sy.104-112

Civil Aviation Authority, 2002: Unmanned Aerial Vehicle Operations in UK Airspace-Guidance, pp.2.

Çalık, S.K., 2016: Çok Etmenli Karınca Kolonisi Yaklaşımıile İHA Rota Planlaması,

Çekmez, U., 2014: İnsansız Hava Araçlarında Büyük Ölçekli Yol Planlama Problemlerinin GPU Üzerinde CUDA Yardımı ile Çözümü, Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsü, Bilgisayar Mühendisliği ABD.

Çelik, M., 2015: Yapay Arı Koloni Algoritması ile Bir Saldırı Tespit Sistemi, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans Tezi.

Demiray, D., 2008: Duyarga Ağları için Karınca Kolonisi Tabanlı Bir Yönlendirme Algoritması Çözümlemesi ve Tasarımı, İstanbul Teknik Üniveritesi Bilişim Enstitüsü, Yüksek Lisans Tezi.

Dijkstra, E.W., 1959: A Note on Two Problems in Connexion with Graphs, Numerische Mathematik 1, pp.269-271.

Dorigo, M., Maniezzo, V. ve Colorni, A., 1991: Ant System: An Autocatalytic Optimizing Process, Technial Report 91-016, Milano, İtaly.

Emel, G.G. ve Taşkın, Ç., 2002: Genetik Algoritmalar ve Uygulama Alanları, Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, Cilt.21, Sayı.1, sy.129-152.

Er, H., Çetin, M.K. ve Çetin E.İ., 2005: Finansta Evrimsel Algoritmik Yaklaşımlar: Genetik Algoritma Uygulamaları, Akdeniz İ.İ.B.F. Dergisi, sy.73-94.

Ergezer, H. ve Leblebicioğlu, K., 2012: Çoklu İnsansız Hava Araçları için Güzergah Planlaması,

Esmeray, M., 2016: Sevkiyat Süreçlerinde Araç Yükleme Probleminin Genetik Algoritma Yaklaşımı ile Çözümü ve Bir Uygulama, Ege Üniversitesi Sosyal Bilimler Enstitüsü, Yüksek Lisans Tezi.

Gangal, V., 2015: Kablosuz Algılayıcı Ağlarda Karınca Koloni Algoritmalı Rotalama ile Enerji Etkin Rotalamanın İncelenmesi, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi.

Geng, Q. ve Zhao, Z., 2013: A Kind of Route Planning Method for UAV Based on Improved PSO Algorithm, 25th ChineseControl and Decision Conference,pp.2328-2331.

Hart, P.E., Nilsson, N.J. ve Raphael, B., 1968: AFormal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems and Cybernetics,pp.100-107.

Ingham, L.A., 2008: Considerations for a Aroadmap for the Operation of Unmanned Aerial Vehicles (UAV) in South African Airspace, Stellenbosh University, pp.2

Kadioğlu, T., Vural, R.A. ve Yıldırım, T., 2010: Yapay Arı Kolonisi Tabanlı Butterworth Filtre Optimizasyonu, In Electrical, Electronics and Computer Engineering,sy.425-428.

Kahraman, A.M. ve Özdağlar, D., 2004: Su Dağıtım Sistemlerinin Genetik Algoritma ile Optimizasyonu, DEÜ Mühendislik Fakültesi Fen ve Mühendislik Dergisi, Cilt.6,Sayı.3,sy.1-18.

Kanoh, H. ve Hara, K., 2008: Hybrid genetic algorithm fordynamic multi-objective route planning with predicted traffic in a real-world road network, 10th annual conference on Genetic and evolutionary computation,pp.657-664.

Karaboğa, D., 2004: Yapay Zeka Optimizasyon Algoritmaları, Cilt.2, Atlas Yayın Dağıtım, İstanbul.

Karaboğa, D. ve Akay, B., 2009 A survey: Algorithms Simulating Bee Swarm Intelligence, Artificial Intelligence Review,pp.61-85.

Kartal, B., 2015: Yapay Arı Kolonisi Algoritması ile Finansal Portföy Optimizasyonu, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü Doktora Tezi.

Keskintürk, T. ve Söyler, H., 2006: Global KarıncaKolonisi Optimizasyonu, Gazi Üniversitesi Müh. Mim. Fak. Dergisi, Cilt.21,sy.689-698.

Kıran, M.S., 2010: Arı Kolonisi ile Şöför Hat Zaman Optimizasyonu, Selçuk Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

Korkmaz, A. 2015**:** Bal Arısı Polinasyonu, Samsun Gıda Tar. ve Hayv. İl Müdürlüğü yayını.

Küçük, A., 2016: Hemşire Çizelgeleme Problemlerinin Genetik Algoritmalarla Optimizasyonu ve Bir Uygulama, Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü,Yüksek Lisans Tezi.

Küçüksille, E.U veTokmak, M., 2011: Yapay Arı Kolonisi Algoritması Kullanılarak Otomatik Ders Çizelgeleme, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi,Cilt.15,sy.203-210.

Leong, K.H., Abdull-Rahman, H., Wang, C., Onn, C.C. ve Loo S.C., 2016: Bee Inspired Novel Optimization Algorithm and Mathematical Model for Effective and Efficient Route Planning in Railway System, PLoS One Vol.11,pp.1-24.

Li, B., Gong, L.G. ve Yang, W.L., 2014: An Improved Artificial Bee Colony Algorithm Based on Balance-Evolution Strategy for Unmanned Combat Aerial Vehicle Path Planning, The Scientific World Journal.

Montavont, J. ve Noel, T., 2006: IEEE 802.11 Handovers Assisted by GPS Information, IEEE International Conference on Wireless and Mobile Computing, Networking and Communications,pp.166-172.

Okkalı, A., 2013: Genetik Algoritmalar ile Aydınlatma Hesabı Optimizasyonu, Kocaeli Üniversitesi FBE Elektronik ve Bilgisayar Eğitimi.

Özalp, N., 2013: 3 Boyutlu Arazi ÜZerinde Çoklu Otonom İnsansız Hava Aracı Rota Planlaması, Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsü Bilgisayar Mühendisliği ABD.

Özcan, M., 2016: Atölye Tipi Çizelgeleme Problemlerinde Evrimsel Algoritmalar ile Yapay Arı Kolonisi Algoritmasının Bütünleşik Bir Yaklaşımı, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi.

Öztürk, C., 2011: Yapay Sinir Ağlarının Yapay Arı Kolonisi Algoritması ile Eğitilmesi, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi.

Öztürk, C., Hancer, E. ve Karaboğa, D., 2014: Küresel En İyi Yapay Arı Koloni Algoritması ile Otamatik Kümeleme, Gazi Üniv. Müh. Mim. Fak. Dergisi, Cilt.29,sy.677-687

Öztürk, D., 2012: İzalatör Yüzey Kaçak Akımlarının KarıncaKoloni Algoritması Yardımıyla İncelenmesi, Fırat Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Doktora Tezi,sy.48-49.

Pakkan, B. ve Ermiş, M., 2010: İnsansız Hava Araçlarının Genetik Algoritma Yöntemiyle Çoklu Hedeflere Planlanması, Havacılık veUzak Teknolojileri Dergisi, Cilt.4,Sayı.3,sy.77-84

Passino, K.M. and Antsaklis, P.J., 1994: A Metric Space Approach to the Specification of the Heuristic Function for the A* Algorithm, IEEE Transactions on System Man. And Cybernetics, Vol.24,pp.159-166.

SSM, 2011: Türkiye İnsansız Hava Aracı Sistemleri Yol Haritası 2011-2030, sy.2-37.

Tereshko, V. ve Loengarov, A. 2005: Collective Decision-Making in Honey Bee Foraging Dynamics, Computing and Informaton Systems,pp.1-7.

Tokaş, A. ve Akdağlı, A., 2012: E ŞEklili Kompakt Mikroşerit Antenlerin Rezonans Frekansının Hesaplanması, Gazi Üniv. Müh. Mim. Fak. Dergisi, Cilt.27,sy.847-854.

Tuncer, A., 2015: Performance Comparison of Genetic Algorithm and A* in Path Planning for Mobile Robots, İnternational Journal of Advanced Computational Engineering and Networking, vol.3, pp.15-18

Tuncer, A., 2013: Otonom Araçlar için Yol Bulma Probleminin Genetik Algoritmalar ve FPGA ile Çzöümü ve Gerçekleştirilmesi, Kocaeli Üniversitesi FBE Elektronik ve Bilgisayar Eğitimi.

Tuncer, A., 2007: Genetik Algoritmalar için Uzak Sanal Laboratuar, Kocaeli Üniveristesi FBE Elektronik ve Bilgisayar Eğitimi

Url-1 <*http://greatmaps.codeplex.com*>, alındığı tarih: 09.10.2016.

USA Department of Defense, 2002: Unmanned Aerial Vehiclec Roadmap 2002-2027, pp.2.

Ünsal, Ü., 2014: Yapay Arı Koloni Algoritması Kullanılarak Çokgensel Güven Bölgesinin Belirlenmesi ve Maden Ocaklarına Uygulanması, Karadeniz Teknik Üniversitesi Fen bİlimleri Enstitüsü, Yüksek Lisans Tezi.

Yakut, E. ve Çankal, A., 2016: Çok Amaçlı Genetik Algoritma ve Hedef Programlama Metodlarını Kullanarak Hisse Senedi Portföy Optimizasyonu: BİST-30'da Bir Uygulama, Business and Economics Research Journal,vol.7,Numb.2,pp.43-62

Yao, J., Lin, C., Xie, X., Wang A.J.A. ve Hung, C.C., 2010: Path Planning for Virtual Human Motion Using Improved A* Algorithm, 7th International Conference on Informnation Technology,pp.1154-1158.

Yıldırım, M., Erkan, K. ve Öztürk, S., 2002: Benzetilmiş Tavlamalı Genetik Algoritmalar ile Uzun Dönem Elektrik Enerjisi Üretim Genişletme Planlaması, Elektrik-Bilgisayar Mühendisliği Sempozyumu ELECO-2002.

Zhan, F.B., 1997: Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures, Journal of Geographic Information and Decision Analysis, Vol.1,pp.70-82.

EKLER

EK 1 : C# kodları

```
//---------Harita başlangıç ayarları
public void Harita_Ayarlari(GMapControl Harita, string Adres)
{
   Harita.MapProvider=GMap.NET.MapProviders
      .GoogleTerrainMapProvider.Istance;
   GMap.NET.Gmaps.Istance.Mode=GMap.NET.AccessMode.ServerAndCache;
   Harita.SetPositionByKeywords(Adres);
}

//---------Başlangıç ve bitiş noktasını belirleme
private void button_Click(object sender, EventArg e)
{
   double[] Koor=new double[3];
   Koor= Enlem_Boylam_Bulma(textbox1.Text); // Şekil 4.3
   Isaretci_Ekle(Koor[0],Koor[1]); // Şekil 4.4
   Koor[2]=Yukseklik_Bilgisi(Koor[0],Koor[1]); // Şekil 4.9
}

//---------Sanal engelleri ekleme C# kodu
private void button_Click(object sender, EventArg e)
{
   string EngelBilgileri=new string[3];
   for(int i=0;i<EngelSayisi;i++)
   {
      string bilgi=listbox1.Items[I].ToString();
      EngelBilgileri=bilgi.Split(';');
```

```
        double enlem=Convert.ToDouble(EnlemBilgileri[0]);
        double boylam=Convert.ToDouble(EnlemBilgileri[1]);
        double yaricap=Convert.ToDouble(EnlemBilgileri[2]);
        Color EngelRengi=Color.Black;
        Haritaya_Engel_Ekle(enlem,boylam,yaricap,EngelRengi);
    }
}

//---------ABC işlemleri
private void button_Click(object sender, EventArg e)
{
    double[,,] Rotalar=new double[200,10,2];
    double[,] RotaBilgileri=new double[200,3];
    Rotalar=Rastgele_Rota_Uret(RotaSayisi,BaslangicBitisKoor,SinirDegerleri,
        AdimSayisi); //Şekil 4.10
    for(int iter=1;iter<=IterasyonSayisi;iter++)
    {
        for(int isci=1;isci<=Isci_Ari_Sayisi;isci++)
            ISCI_ARI(isci);
        for(int gozcu=1;gozcu<=Gozcu_Ari_Sayisi;gozcu++)
            GOZCU_ARI(gozcu);
    }
    Rotayi_Haritaya_Ciz(En_Kisa_Rota,Color.Red);
    Rota_Yukseklik_Bilgilerini_Oku(En_Kisa_Rota);
    Cografi_Engelleri_Belirle(MakYukselisAcisi,MinDonusCapi/2);
}

ISCI_ARI(int isci)
{
    double Mesafe=Rota_Mesafesi(Rotalar,isci);
    double UygunlukDegeri=Uygunluk_Degerini_Hesapla(Rotalar,isci);
    int nokta=Rastgele_Tam_Sayi_Uret(1,Adim_Sayisi);
    int rota=Rastgele_Tam_Sayi_Uret(1,Rota_Sayisi);
    int koor=Rastgele_Tam_Sayi_Uret(0,1);
```

```
    double Vij,Qij;
    Qij=Rastgele_Ondalikli_Deger_Uret(-1,1);
    Vij=Rotalar[isci,nokta,koor]+
       Qij*(Rotalar[isci,nokta,koor]-Rotalar[rota,nokta,koor]);
    double EskiBilgi=Rotalar[isci,nokta,koor];
    Rotalar[isci,nokta,koor]=Vij;
    double YeniMesafe=Rota_Mesafesi(Rotalar,isci);
    double YeniUygunlukDegeri=Uygunluk_Degeri_Hesapla(Rota_Bilgileri,isci);
    if(UygunlukDegeri>YeniUygunlukDegeri)
       Rotalar[isci,nokta,koor]=EskiBilgi;
    else
    {
       RotaBilgileri[isci,1]=YeniUygunlukDegeri;
       RotaBilgileri[isci,0]=YeniMesafe;
    }
}

public void GOZCU_ARI(int gozcu)
{
    double Mesafe=Rota_Mesafesi(Rotalar,gozcu);
    double UygunlukDegeri=Uygunluk_Degerini_Hesapla(Rotalar,gozcu);
    int nokta=Rastgele_Tam_Sayi_Uret(1,Adim_Sayisi);
    int rota=Uygunluga_Gore_Rota_Sec(Rota_Bilgileri,Rota_Sayisi);
    int koor=Rastgele_Tam_Sayi_Uret(0,1);
    double Vij,Qij;
    Qij=Rastgele_Ondalikli_Deger_Uret(-1,1);
    Vij=Rotalar[gozcu,nokta,koor]+
       Qij*(Rotalar[gozcu,nokta,koor]-Rotalar[rota,nokta,koor]);
    double EskiBilgi=Rotalar[gozcu,nokta,koor];
    Rotalar[gozcu,nokta,koor]=Vij;
    double YeniMesafe=Rota_Mesafesi(Rotalar, gozcu);
    double YeniUygunlukDegeri=Uygunluk_Degeri_Hesapla(Rota_Bilgileri, gozcu);
    if(UygunlukDegeri>YeniUygunlukDegeri)
       Rotalar[gozcu,nokta,koor]=EskiBilgi;
```

```
    else
    {
        RotaBilgileri[gozcu,0]=YeniUygunlukDegeri;
        RotaBilgileri[gozcu,1]=YeniMesafe;
    }
}
public void KASIF_ARI(double[,] BBK, double[,] SD, int AS, int kasif)
{
    int f=AS+1;
    for(int j=0;j<2;j++)
    {
        Rotalar[kasif,0,j]=BBK[0,j];
        Rotalar[kasif,f,j]=BBK[1,j];
        for(int i=1;i<=AS;i++)
        {
            Random rastgele=new Random();
            double rastgele_sayi=rastgele.NextDouble();
            Rotalar[kasif,i,j]=SD[0,j]+(SD[f,j]-SD[0,j])*rastgele_sayi;
            if(Rotalar[kasif,i,j]<SD[0,j]) Rotalar[r,i,j]=SD[0,j];
            if(Rotalar[kasif,i,j]>SD[1,j]) Rotalar[r,i,j]=SD[1,j];
        }
    }
}

public double Uygunluk_Degerini_Hesapla(double[,] RotaBilgileri,int RotaNo)
{
    double sonuc;
    sonuc=1/(1+RotaBilgileri[RotaNo,1]);
    return sonuc;
```

```
}
public int Rastgele_Tam_Sayi_Uret(int Baslangic, int Bitis)
{
   Random rastgele=new Random();
   int sonuc=rastgele.Next(Baslangic,Bitis+1);
   return sonuc;
}
public double Rastgele_Ondalikli_Deger_Uret(int Baslangic, int Bitis)
{
   Random rastgele=new Random();
   double sonuc=rastgele.NextDouble();
   int Kat=Bitis-Baslangic;
   sonuc*=Kat;
   sonuc+=Baslangic;
   return sonuc;
}

public int Uygunluga_Gore_Rota_Sec(double[,] RotaBilgileri, int RotaSayisi)
{
   double toplam=0;
   Random rastgele=new Random();
   double veri=rastgele.NextDouble();
   for(int i=1;i<=RotaSayisi;I++)
   {
      toplam+=(1/(RotaBilgileri[I,1]);
      if(veri<toplam)
      {
         sonuc=i;
         break;
      }
   }
   return sonuc;
}
public double Rota_Mesafesi(double[,,] Rotalar, int gozcu)
```

```
{
   double mesafe=0;
   int AS=Adim_Sayisi;
   for(int i=0;i<AS+1;i++)
   {
      mesafe+=Iki_Nokta_Arasindaki_Mesafe(Rotalar[gozcu,i,0], Rotalar[gozcu,i,1],
         Rotalar[gozcu,i+1,0], Rotalar[gozcu,i+1,1]);
   }
   mesafe+=Iki_Nokta_Arasindaki_Mesafe(Rotalar[gozcu,AS+1,0],
      Rotalar[gozcu,AS+1,1],Rotalar[gozcu,0,0], Rotalar[gozcu,0,1]);
   return mesafe;
}
public double Iki_Nokta_Arasindaki_Mesafe(double Enlem1, double Boylam1,
   double Enlem2, double Boylam2)
{
   double R=6371
   double D_enlem=Enlem1-Enlem2;
   double D_boylam=Boylam1-Boylam2;
   double enlem=D_enlem*Math.PI/180;
   double boylam=D_boylam*Math.PI/180;
   double a=Math.Pow(Math.Sin(enlem/2),2)+Math.Cos(Enlem1*Math.PI/180)*
      Math.Cos(Enlem2*Math.PI/180)*Math.Pow(Math.Sin(boylam/2),2);
   double b=2*Math.Asin(Math.Min(1,Math.Sqrt(a)));
   double sonuc=Math.Round(R*b,3);
   return sonuc;
}

public void Rotayi_Haritaya_Ciz(int EnKisaRota,Color RotaRengi)
{
   for(int i=0;i<Adim_Sayisi;i++)
      Iki_Koordinat_Noktasini_Birlestir(Rotalar[EnKisaRota,i,0],
         Rotalar[EnKisaRota,i,1], Rotalar[EnKisaRota,i+1,0],
         Rotalar[EnKisaRota,i+1,1],RotaRengi,gMapControl1);
}
```

```
public Iki_Koordinat_Noktasini_ Birlestir(double Enlem1, double Boylam1,
   double Enlem2, double Boylam2, color CizgiRengi, GMapControl Harita)
{
   GMapOverlay isaretci=new GMapOverlay("cizgi");
   List<PointLatLng> Cizgi=new List<PointLatLng>();
   Cizgi.Clear();
   Cizgi.Add(new PointLatLng(Enlem1,Boylam1);
   Cizgi.Add(new PointLatLng(Enlem2,Boylam3);
   GMapPolygon Cizgim=new GMapPolygon(Cizgi,"IkiCizgi");
   Cizgim.Stroke=new Pen(CizgiRengi, 3);
   Isaretci.Polygons.Add(Cizgim);
   Harita.Overlays.Add(isaretci);
}
public double[] Rota_Yukseklik_Bilgilerini_Oku(int EnKisaRota)
{
   int Adim=20;
   int f=0;
   double mesafe=Rota_Bilgileri[EnKisaRota,0];
   int OkumaAdeti=Math.Round(mesafe*100/Adim,0);
   double[] Yukseklik=new double[OkumaAdeti];
   for(int i=0;i<Adim_Sayisi;i++)
   {
      double DogruMesafesi= Iki_Nokta_Arasindaki_Mesafe(
         Rotalar[EnKisaRota,i,0], Rotalar[EnKisaRota,i,1],
         Rotalar[EnKisaRota,i+1,0], Rotalar[EnKisaRota,i+1,1]);
      double AdimSayisi=Convert.ToIn32(DogruMesafesi/Adim);
      double D_enlem=Rotalar[EnKisaRota,i,0]-Rotalar[EnKisaRota,i+1,0];
      double EnlemAdim=D_enlem/AdimSayisi;
      double D_boylam=Rotalar[EnKisaRota,i,1]-Rotalar[EnKisaRota,i+1,1];
      double BoylamAdim=D_boylam/AdimSayisi;
      for(int j=0;j<AdimSayisi;j++)
      {
         f++;
         double OlcumEnlemi=Rotalar[EnKisaRota,i,0]+(j*EnlemAdim);
```

```
            double OlcumBoylami=Rotalar[EnKisaRota,i,1]+(j*BoylamAdim);
            Yukseklik[f]=Yukseklik_Ogrenme(OlcumEnlemi,OlcumBoylami);
        }
    }
}
public double Yukseklik_Bilgisi(double Enlem, double Boylam)
{
    string Sunucu="https://maps.googleapis.com/maps/api/"
    Sunucu+="elevation/json?location="+Enlem.ToString()+";"+Boylam.ToString();
    string Veri;
    using (WebClient Cevap=new WebClient())
        Veri=cevap.DownloadString(Sunucu);
    JObject J_Veri=JObject.Parse(Veri);
    JArray J_Yukseklik=(JArray)J_Veri["results"];
    double h=Convert.ToDouble(J_Yukseklik[0]["elevation"]);
    return h;
}
public double[,,] Rastgele_Rota_Uret(int BS, double[,] BBK, double[,] SD, double
AS)
{  int f=Adim_Sayisi+1;
    double[,,] Rota_Bilgileri=new double[BS,f+1,2];
    for(int r=1;r<=BS;r++)
    {  for(int j=0;j<2;j++)//j=0 enlem bilgisi, j=1 boylam bilgisini
        {
            Rota_Bilgileri[r,0,j]=BBK[0,j]; //başlangıç enlem/boylam bilgisi
            Rota_Bilgileri[r,f,j]=BBK[1,j]; //bitiş enlem/boylam Bilgisi
            for(int i=1;i<=Adim_Sayisi;i++)
            {  Random rastgele=new Random();
                double rastgele_sayi=rastgele.NextDouble();
                Rota_Bilgileri[r,i,j]=SD[0,j]+(SD[f,j]-SD[0,j])*rastgele_sayi;
                if(Rota_Bilgileri[r,i,j]<SD[0,j]) Rota_Bilgileri[r,i,j]=SD[0,j];
                if(Rota_Bilgileri[r,i,j]>SD[1,j]) Rota_Bilgileri[r,i,j]=SD[1,j];
            }
        }
```

```
    }
return Rota_Bilgileri;
}
```

ÖZGEÇMİŞ

Ad Soyad: Volkan ÇAVUŞ

Doğum Yeri ve Tarihi: İstanbul / 04.05.1982

E-Posta: cvsvlkn@gmail.com

Lisans: Yalova Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü

Lisans: Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Öğretmenliği

Mesleki Deneyim:

2009/ Sinop Üniversitesi, Öğretim Görevlisi

2003/2007 Kocaeli Halk Eğitim Merkezi ve ASO, Bilgisayar Öğretmeni

2002/2004 Kocaeli Valiliği, Bilgi İşlem

Yayın Listesi:

Duran İ.U. ve **Çavuş V.**, 2016: Akıllı Telefon Tabanlı Sanal Gerçekliğin Mesleki Eğitimde Uygulanması. *Mesleki Bilimler Dergisi*, Cilt.5(2), sy.6-11.

Çavuş V., Tuna R. ve Duran İ.U., 2016: Raspberry Pi Kullanılarak Sıramatik Tasarımı (Aile Hekimliği Sinop Örneği). *İnternational Mediterranean Science and Engineering Congress*. Sy.4694-4698, Ekim 26-28, 2016 Adana-Türkiye.

Çavuş V., Tuna R. ve Duran İ.U., 2016: İki Boyutlu Devre Resminin Tasarlanan Üç Eksenli Bir Platform ile Plaket Baskı Üzerine Çizimi. *V. Uluslararası Meslek Yüksekokuları Sempozyumu (UMYOS-2016)*, Cilt.1, sy.181-184, Mayıs 18-20, 2016 Prizren-Kosova.

Çavuş V. ve Tuncer A., 2015: An Application Interface Design for Backpropagation Artificial Neural Networks. *International Conference on Computer Sciance, Data Mining and Mechanical Engineering (ICCDMME-2015)*, pp.100-103, Nisan 20-21, 2015 Bangkok-Thailand.

Çavuş V., Tuncer A. ve Gök M., 2015: Sinop İline Ait Deniz Suyu Sıcaklığının Yapay Sinir Ağı ile Tahmini. *IV. Uluslararası Meslek Yüksekokulları Sempozyumu (UMYOS-2015)*, sy.2326-2330, Mayıs 21-23, 2015 Yalova-Türkiye.

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

▪ **Çavuş V.** ve Tuncer A., 2016: İnsansız Hava Araçları İçin Yapay Arı Kolonisi Algoritması Kullanarak Rota Planlama. *Karaelmas Fen ve Mühendislik Dergisi*.