



**MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ  
2017-2018 AKADEMİK YILI  
BITİRME PROJESİ RAPORU**

***“SÜRÜ OPTİMİZASYON ALGORİTMALARI  
UYGULAMALARI”***

**Öğrenciler:** Eray ALTAY - 21493632  
Nisanur SUNGUR - 21494986  
Figen Gizem TOSUN - 21495793  
Mehmet Emre YORULMAZ - 21495368

**Proje Danışmanı:** Prof. Dr. A. Ziya AKTAŞ

# **SÜRÜ OPTİMİZASYON ALGORİTMA UYGULAMALARI**

## **COLONY / SWARM OPTIMIZATION ALGORITHM APPLICATIONS**

**Öğrenciler:** Eray ALTAY - 21493632  
Nisanur SUNGUR - 21494986  
Figen Gizem TOSUN - 21495793  
Mehmet Emre YORULMAZ - 21495368

**Proje Danışmanı:** Prof. Dr. A. Ziya AKTAŞ

Başkent Üniversitesi  
Mühendislik Fakültesi  
BİLGİSAYAR Mühendisliği 2017-2018 Akademik Yılı  
BITİRME PROJESİ RAPORU  
olarak hazırlanmıştır.

Mayıs 2018

## SÜRÜ OPTİMİZASYON ALGORİTMA UYGULAMALARI

### ÖZ

Optimizasyon, bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçme işlemidir. Optimizasyon problemleri için birçok algoritma önerilmiştir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalarıdır. Bazen tek başlarına hiçbir iş yapamayan varlıklar, toplu hareket ettiklerinde çok zekice davranışlar sergileyebilmektedir. Bir topluluğa ait bireyler, en iyi bireyin davranışından ya da diğer bireylerin davranışlarından ve kendi deneyimlerinden yararlanarak değerlendirme yapmakta ve bu bilgileri ilerde karşılaşacakları problemlerin çözümleri için bir araç olarak kullanmaktadır. Canlıların sürü içerisindeki bu hareketleri gözlemlenerek sürü zekâsı tabanlı optimizasyon algoritmaları geliştirilmiştir. Bu projede uygulamada kullanılan bazı sürü zekâsı tabanlı sezgisel optimizasyon algoritmaları incelenmiştir. Üzerinde çalışılan bu algoritmalar arı, karınca, kuş ve kurt sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir. Sürü zekâsı hayvanların kolektif davranışları sayesinde kendi aralarında ve çevreleri ile etkileşim halinde gösterdikleri davranışlardan esinlenerek geliştirilen bir zeka tipidir. Bu projenin temel amacı; arı, karınca, kuş ve kurtlarla ilgili sezgisel optimizasyon algoritmalarının (sırasıyla ABC (Artificial Bee Colony), ACO (Ant Colony Optimization), PSO (Particle Swarm Optimization) ve GWO (Gray Wolf Optimizer)) çalışma prensiplerinin ve temel adımlarının incelenmesi ve anlaşılmasıdır. Çalışma sırasında dört örnek problem seçilerek bu algoritmaların uygulaması gerçekleştirilmiştir. Örnek problemler ve algoritmaları kullanılan hayvanlar sıra ile şöyle seçilmiştir: Tarım Alanlarının Sınıflandırılması – Arı, Aritmi Sınıflama – Karınca, Sensör Yerleştirme Problemi – Kuş ve İnsansız Hava Aracı Yol Bulma Problemi – Kurt. Çalışmalar sırasında MATLAB yazılımı, uygulama kolaylığı ve matematik-istatistik işlemler desteği nedeniyle kullanılmıştır. Projenin en son adımı olarak algoritmaların karşılaştırması on farklı benchmark fonksiyonu ile yapılmıştır.

**ANAHTAR KELİMELER:** Hayvan, Optimizasyon, Problem, Sezgisel Algoritma, Sürü Zekası.

# **COLONY/SWARM OPTIMIZATION ALGORITHM APPLICATIONS**

## **ABSTRACT**

Optimization is the process of choosing the best from among the alternatives possible under certain conditions in a problem. Many algorithms have been proposed for optimization problems. Heuristic algorithms are algorithms that can give optimal solutions for large size optimization problems in acceptable Time (sec.). SomeTime (sec.)s individuals who can not do anything alone can exhibit very clever behaviors when they act collectively. Individuals belonging to a community make use of the best individual's behavior or behavior of other individuals and their own experiences and make use of this information as a means of solving the problems they will face in the future. Observing these movements in the flock, the herd-based optimization algorithms have been developed. In this project, some herd-based heuristic optimization algorithms used in the application are examined. These algorithms have been developed by examining the movements of bee, ant, bird and wolf. It is a type of intelligence that is developed by the herd of intelligent animals inspired by their collective behaviors and the behaviors they show in interaction with each other and with their surroundings. The main purpose of this project is; ABC (Artificial Bee Colony), ACO (Ant Colony Optimization), PSO (Particle Swarm Optimization) and GWO (Gray Wolf Optimizer), and basic steps of the heuristic optimization algorithms (bee, ant, bird and wolf) are studied and understood. During the study, four sample problems were selected and the application of these algorithms was performed. The animals that use sample problems and algorithms are selected as follows: Classification of Agricultural Areas- Bee, Arrhythmia Classification- Ant, Sensor Placement Problem - Bird and Unmanned Air Vehicle Routing Problem - Kurt. During the work, MATLAB software has been used for ease of application and support for mathematical-statistical operations. The last step of the project was to compare algorithms with ten different benchmark functions.

**KEY WORDS:** Animal, Herd Intelligence, Heuristic Algorithm, Optimization, Problem.

## **İÇİNDEKİLER**

**ÖZ.....i**

**ABSTRACT.....ii**

**I GİRİŞ.....1**

1.1	Problem Tanımı.....	1
1.2	Kaynak Taraması.....	1
1.3	Çalışmanın Amacı.....	2
1.4	Kısıtlar ve Gereksinimler.....	2
1.4.1	Gerçekçi Kısıtlar.....	2
1.4.2	Fonksiyonel Gereksinimler.....	2
1.4.3	Fonksiyonel Olmayan Gereksinimler.....	3
1.5	Başarı Ölçütleri.....	3
1.6	Güz Döneminde Yapılan İşler.....	3
1.7	Bahar Dönemi Hedefleri.....	3
1.8	İş – Zaman Diyagramı.....	4
1.9	Görev Dağılımı.....	5

**II. OPTİMİZASYON NEDİR?.....6**

2.1	Genel Optimizasyon Tanımı.....	6
2.2	Optimizasyon Problemlerinin Sınıflandırılması.....	9
2.3	Optimizasyon Methodlarının Sınıflandırılması .....	9
2.3.1	İteratif Araştırma İşlemi Algoritma Formu.....	10
2.3.2	En-Dik İniş Metodu .....	11
2.3.3	Eşlenik Gradiyent Metodu .....	11
2.3.4	Modifiye Edilmiş Newton Metodu.....	11
2.4	Optimizasyon İçin Sezgisel Algoritmalar.....	13
2.4.1	Sezgisel Algoritmala Gerek Duyulmasının Sebepleri.....	13
2.4.2	Sezgisel Algoritmaların Değerlendirilmesi İçin Kriterler.....	14
2.4.3	Sürü Zekası Tabanlı Sezgisel Algoritmalar.....	14
2.4.3.1	Çözümün Gösterimi.....	15
2.4.3.2	Çözümün Uygunluğu.....	15

2.4.3.3 Popülasyon.....	15
2.4.3.4 Seçim Mekanizmaları.....	15
2.4.3.5 Yeni Çözümün Üretilmesi.....	16
2.4.3.6 İlklenidleme.....	16
2.4.3.7 Sonlandırma.....	16
<b>III . ARILAR İLE SÜRÜ OPTİMİZASYONU.....</b>	<b>17</b>
3.1    Yapay Arı Koloni Algoritması(ABC).....	17
3.1.1    ABC Algoritmasının Temel Adımları.....	17
3.1.2    ABC Algoritmasının Temel Özellikleri.....	20
3.2    Arıların Yiyecek Arama Davranışları.....	20
3.2.1    Yiyecek Kaynakları.....	20
3.2.2    Görevli İşçi Arılar.....	20
3.2.3    Görevsiz İşçi Arılar.....	20
3.3    Uygulama Alanları.....	25
3.4    Seçilen Örnek Problem.....	25
3.5    Tarım Alanlarının Sınıflandırılması.....	25
3.5.1    ABC Algoritması ile Tarım Alanlarının Sınıflandırılması Probleminin Çözümü .....	26
3.5.2    Örnek Çalışma .....	27
3.6    İş – Zaman Diagramı .....	30
3.7    Sonuç.....	31
3.8    Kaynakça.....	32
<b>IV KARINCALAR İLE SÜRÜ OPTİMİZASYONU.....</b>	<b>33</b>
4.1    Ant Colony Optimization (ACO) Algoritması.....	33
4.1.1    Geçiş Kuralı.....	37
4.1.2    Feromon güncellemesi.....	38
4.1.3    Lokal Feromon Güncellemesi.....	39
4.1.4    Global Feromon Güncellemesi.....	40
4.1.5    Optimum Karınca Sayısı.....	40
4.1.6    Parametre Değerleri.....	40
4.2    Karıncalar İle İlgili Diğer Algoritmalar.....	41

4.2.1 Maksimum-Minimum Karınca Sistemi.....	41
4.2.2 Rank Temelli Karınca Sistemi.....	42
4.2.3 Çoklu Karınca Koloni Algoritmaları.....	42
4.2.4 Melez ACO.....	42
4.3 Uygulama Alanları.....	43
4.4 Seçilen Örnek Problem.....	43
4.5 Problemin ACO ile Çözülmesi.....	44
4.6 Örnek Çalışma.....	46
4.7 Sonuç.....	47
4.8 İş – Zaman Diyagramı.....	48
4.9 Kaynakça.....	49
<b>V KUŞLAR İLE SÜRÜ OPTİMİZASYONU.....</b>	<b>50</b>
5.1 Kuşlar İle Süre Optimalizasyonu - Parçacık Süre Optimalizasyonu (PSO) Algoritması.....	50
5.2 Parçacık Süre Optimalizasyonu'nda Kullanılan Terimler.....	51
5.3 Parçacık Süre Optimalizasyonu Algoritması.....	52
5.4 Parçacık Süre Optimalizasyonu Parametreleri.....	55
5.4.1 Sürenin Büyüklüğü.....	55
5.4.2 İterasyon Sayısı.....	55
5.4.3 Bilişsel ve Sosyal Bileşenler.....	55
5.4.4 Atalet Bileşeni.....	55
5.4.5 En Yüksek Hız.....	55
5.4.6 Kısıtlama Faktörü.....	56
5.5 Uygulama Alanları.....	56
5.6 Seçilen Örnek Problem: PSO ile Sensör Yerleştirme Problemi .....	56
5.6.1 Arazi.....	57
5.6.2 Sensör.....	57
5.6.3 Problemin Formüle Edilmesi.....	57
5.6.4 Problemin İşleyişi.....	58
5.6.5 Problemin Parçacık Süre Optimalizasyonuna Uyarlanması.....	59
5.7 Sonuç.....	60

5.8	İş – Zaman Diyagramı.....	61
5.9	Kaynakça.....	62
<b>VI</b>	<b>KURTULAR VE SÜRÜ OPTİMİZASYONU.....</b>	<b>63</b>
6.1	Kurt Sürülerı.....	63
6.2	Kurt Sürülerini Konu Alan Süre Optimizasyonu Algoritmaları.....	64
6.3	Grey Wolf Optimizer Algorithm (GWO).....	65
6.3.1	Çalışma Prensipleri.....	65
6.4	Uygulama Alanları.....	69
6.5	Seçilen Örnek Problem .....	70
6.6	İHA Yol Bulma Problemi.....	70
6.7	GWO ile İHA Yol Bulma Probleminin çözümü .....	70
6.8	Örnek Çalışma.....	72
6.9	Sonuç.....	73
6.10	İş – Zaman Diyagramı.....	74
6.11	Kaynakça.....	75
<b>VII</b>	<b>KARŞILAŞTIRMA VE SONUC.....</b>	<b>76</b>
7.1	Performans Ölçümü ve Karşılaştırma.....	76
7.2	Artificial Bee Colony Algoritmasının Benchmark Fonksiyonları ile İncelenmesi.....	78
7.3	Ant Colony Optimization Algoritmasının Benchmark Fonksiyonları ile İncelenmesi.....	79
7.4	Particle Swarm Optimization Algoritmasının Benchmark Fonksiyonları ile İncelenmesi.....	80
7.5	Gray Wolf Optimizer Algoritmasının Benchmark Fonksiyonları ile İncelenmesi.....	81
7.6	Sonuçların Karşılaştırılması.....	82
7.7	Benchmark Fonksiyonlarının MATLAB Kodu.....	83
<b>VIII</b>	<b>EKLER.....</b>	<b>86</b>

## ŞEKİLLER LİSTESİ

<b>Şekil 2.1</b> Bir fonksiyonun yerel (local) ve genel (global) minimum noktaları.....	7
<b>Şekil 2.2</b> Gradiyent vektörü.....	7
<b>Şekil 2.3</b> Optimizasyon algoritmaları şeması.....	10
<b>Şekil 3.1</b> Arıların dansı.....	21
<b>Şekil 3.2</b> Yiyecek arama çevrimi.....	23
<b>Şekil 3.3</b> ABC algoritmasının akış diyagramı.....	24
<b>Şekil 3.4</b> Oluşturulan Harita.....	28
<b>Şekil 3.5</b> Örnek Çıktı.....	29
<b>Şekil 4.1</b> Gerçek karıncaların en kısa yolu bulma aşamaları.....	34
<b>Şekil 4.2</b> Karıncaların yol seçimleri.....	35
<b>Şekil 4.3</b> ACO algoritmasının adımları.....	36
<b>Şekil 4.4</b> ACO algoritması için sözde kod.....	37
<b>Şekil 4.5</b> ACO algoritması için akış diyagramı.....	37
<b>Şekil 4.6</b> Parametre seçimi.....	41
<b>Şekil 4.7</b> Parametre listesi.....	41
<b>Şekil 4.8</b> EKG görüntüsü nokta formu.....	44
<b>Şekil 4.9</b> Problem çözümünün görsel taslağı.....	45
<b>Şekil 4.10</b> Aritmi sınıflama probleminin ACO ile çözüm adımları.....	46
<b>Şekil 4.11 (a)</b> Sinüs taşikardisi ve örnek aritmi benzeşimi.....	46
<b>Şekil 4.11 (b)</b> Multifokal atrial taşikardi ve örnek aritmi benzeşimi.....	46
<b>Şekil 4.11 (c)</b> PAC taşikardi ve örnek aritmi benzeşimi.....	46
<b>Şekil 5.1</b> PSO için sözde kod.....	53
<b>Şekil 5.2</b> Parçacık sürü optimizasyonu akış diyagramı.....	54
<b>Şekil 5.3</b> Seçilen arazinin görünümü.....	58
<b>Şekil 5.4</b> Seçilen arazinin grafiğe dökümü.....	58
<b>Şekil 5.5</b> Sensörlerin tarama grafiği.....	59
<b>Şekil 5.6</b> PSO ve Sensör Yerleştirme Problemi sözde kodları.....	59
<b>Şekil 6.1</b> Hiyerarşik yapı.....	63
<b>Şekil 6.2</b> Avlanma örneği.....	64

<b>Şekil 6.3</b> Kurtların sonraki pozisyonlarının 2 ve 3 boyutlu ortamda belirlenmesi.....	66
<b>Şekil 6.4</b> Kurtların pozisyon güncelleme mekanizması.....	67
<b>Şekil 6.5</b> Kurtların avlarına saldırma durumları.....	67
<b>Şekil 6.6</b> GWO için sözde kod.....	68
<b>Şekil 6.7</b> Akış diyagramı.....	69
<b>Şekil 6.8</b> İHA GWO Sözde kod.....	72
<b>Şekil 6.9</b> Örnek Çıktı.....	73
<b>Şekil 7.1</b> Benchmark fonksiyonları.....	76
<b>Şekil 7.2</b> İki boyutlu Benchmark fonksiyonları.....	77

## TABLOLAR LİSTESİ

<b>Tablo 7.1</b> ABC Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri.....	78
<b>Tablo 7.2</b> ACO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri.....	79
<b>Tablo 7.3</b> PSO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri.....	80
<b>Tablo 7.4</b> GWO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri.....	81
<b>Tablo 7.5</b> Algoritmaların karşılaştırılması.....	82

# I GİRİŞ

## 1.1 Problem Tanımı

Optimizasyon bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçme işlemidir.

Sürü (Swarm) birbirleriyle etkileşim halinde bulunan ve dağınık yapıya sahip bireylerin oluşturduğu yapılara denir. Canlılar topluluğunu oluşturan arılar, karıncalar, kurtlar, kuşlar gibi canlılar bu sürülere örnek olarak gösterilebilir.

Zekâ, insanların öğrenme, anlama, soyut düşünme, sebeplendirme, planlama, problem çözme ve kanya varma gibi zihinsel yeteneklerine verilen genel bir sözcüktür.

Sürü zekâsı ise hayvanların kolektif davranışları sayesinde çevreleri ile etkileşim halinde gösterdikleri yem bulma, taşıma içinde yardımlaşma, kümelenme, tehlikeden uzaklaşma gibi davranışlarından esinlenerek geliştirilen bir zeka tipidir.

Bazen tek başına hiçbir iş yapamayan varlıklar, toplu hareket ettiklerinde çok zekice davranışlar sergileyebilmektedir. Bir topluluğa ait bireyler, en iyi bireyin davranışından ya da diğer bireylerin davranışlarından ve kendi deneyimlerinden yararlanarak yorum yapmakta ve bu bilgileri ileride karşılaşacakları problemlerin çözümleri için bir araç olarak kullanmaktadır. Örneğin, bir canlı sürüsünü oluşturan bireylerden birisi bir tehlike sezdiğinde bu tehlikeye karşı tepki verir ve bu tepki sürü içinde ilerleyip tüm bireylerin tehlikeye karşı ortak bir davranış sergilemesini sağlar. Canlıların sürü içerisindeki bu hareketleri gözlemlenerek sürü zekâsı tabanlı optimizasyon algoritmaları geliştirilmiştir.

“Sürü Optimizasyon Algoritma Uygulamaları” konulu bitirme projemizin amacı seçilmiş olan dört adet hayvan sürüsü ile seçilen örnek problemlere MATLAB yardımıyla çözüm üretmektir.

Raporun ilerleyen bölümlerinde “Sürü Optimizasyon Algoritma Uygulamaları” problem tanımı daha detaylı bir şekilde ele alınacaktır.

Bitirme projemiz, 2017-2018 öğretim yılı güz ve bahar dönemi olmak üzere iki akademik dönem içerisinde gerçekleştirilmiştir. Belirlmiş olduğumuz iş-zaman diyagramında birinci ve ikinci dönem çalışmaları haftalara göre belirlenmiş ve bu çizelgeye çoğulukla uyulmuştur.

## 1.2 Kaynak Taraması

Kaynakları incelemek proje konusunun anlaşılmasına ve örnek problem seçimine büyük katkı sağladı. Alt Bölüm 1.8’deki iş-zaman diyagramında da görüleceği gibi kaynak taraması projenin en uzun süren adımlarından birisi olmuştur. Yaptığımız kaynak taramasının adımları şunlardır:

- Konuyu anlamamıza yardımcı olabilmesi amacıyla, internet sayfalarından ve benzeri kaynaklardan bir ön taslak oluşturulmuştur.
- Proje konumuzun temelini oluşturan sürü optimizasyon algoritmalarını konu alan “Yapay Zeka Optimizasyon Algoritmaları” kitabını ortak kaynak olarak belirledik ve kitabı detaylı bir şekilde taradık.

- Taradığımız kaynaklardan elde ettiğimiz bilgileri düzenli aralıklarla proje danışmanımıza ve birbirimize ilettik. Aynı zamanda bu bilgileri hem internet ortamında hem de basılı kopya şeklinde muhafaza ettik.
- Örnek problem seçimi için birçok makale okumamız gerekiği için uzun bir süre elektronik kaynak taraması yaptık.
- Taradığımız kaynakların en güncel olanları her bölümün sonunda ayrı ayrı verilmiştir.

### **1.3 Çalışmanın Amacı**

Çalışmamızın amaçları aşağıda maddeler halinde verilmiştir:

- Optimizasyon kavramını anlamak;
- Sürü optimizasyon algoritmalarının çalışma prensiplerini matematiksel ve algoritmik yönden inceleyerek kullanım alanlarını saptamak;
- Algoritmaları uygun bir örnek problem üzerinde MATLAB yardımıyla uygulamak ve uygulama sonuçlarını karşılaştırmak;
- MATLAB üzerinde daha fazla deneyim kazanmak;
- Benchmark Fonksiyonlarının uygulamasını yapmak.

### **1.4 Kısıtlar ve Gereksinimler**

#### **1.4.1 Gerçekçi Kısıtlar**

Gerçekçi kısıtlar aşağıda maddeler halinde belirtilmiştir:

- Yazılım ve donanım gereksinimlerini yeterince sağlamaşı,
- Karşılaştırılabilirlik.

Sürü optimizasyon algoritmaları seçilirken her algoritmanın diğer algoritmalarдан en az biri ile aynı problemleri çözebilmesi gerekmektedir. Bu kısıt projemizin karşılaştırma aşaması için önemlidir.

- Uygulanabilirlik

Seçilen algoritmaların kaynak kodları gizli olmamalıdır; algoritmaların uygulanması için kaynak kodlarına ihtiyaç duyulmaktadır.

#### **1.4.2 Fonksiyonel Gereksinimler**

Fonksiyonel gereksinimler aşağıda maddeler halinde sıralanmıştır:

- Algoritmalar verilen girdileri alabilmeli ve bir çıktı verebilmelidir.
- Algoritmalar çeşitli problemler üzerinde uygulanabilmeli ve çözüme ulaşabilmelidir.
- Algoritmalar MATLAB platformunda çalışabilmelidir.
- Algoritmalar sürü optimizasyonu kullanarak problemin özelliğine bağlı optimum sonuçlar vermelidir.

### **1.4.3 Fonksiyonel Olmayan Gereksinimler**

Fonksiyonel olmayan gereksinimler aşağıda maddeler halinde sıralanmıştır:

- Algoritmalar çözükleri problemlerde sonuca belirli bir iterasyon sonucu ulaşabilmelidir.
- Algoritmalar problemlerin çözüm aşamasında üzerinde çalışıkları bilgisayar sisteminin işlemcisini ve belleğini aşırı düzeyde zorlamamalıdır.
- Sonuçlar en fazla yarım saat içinde alınabilmelidir.

### **1.5 Başarı Ölçütleri**

Başarı ölçütleri aşağıda maddeler halinde sıralanmıştır:

- Algoritmaların örnek problemlere uygulanabilmesi;
- Uygulamaların karşılaştırılabilir veriler elde etmesi;
- Algoritmaların farklı girdiler ile farklı performans verilerinin elde edilebilmesi;
- Benchmark Fonksiyonları ile karşılaştırılabilir olması.

Bölüm 2’de bu bölümdeki alt başlıklar 1.4 ve 1.5 ile ilgili detaylı bilgi verilmiştir.

### **1.6 Güz Döneminde Yapılan İşler**

Güz döneminde yapılan işler aşağıda sırasıyla verilmiştir:

- Sürü optimizasyon uygulamaları araştırılması, konunun anlaşılması ve kaynakların analizi;
- Üzerinde çalışılacak hayvan sürülerinin belirlenmesi ve araştırılması;
- Belirlenen hayvan sürüleriyle ilgili algoritmaların incelenmesi ve örnek problem seçimi;
- Sunuların ve raporun hazırlanması.

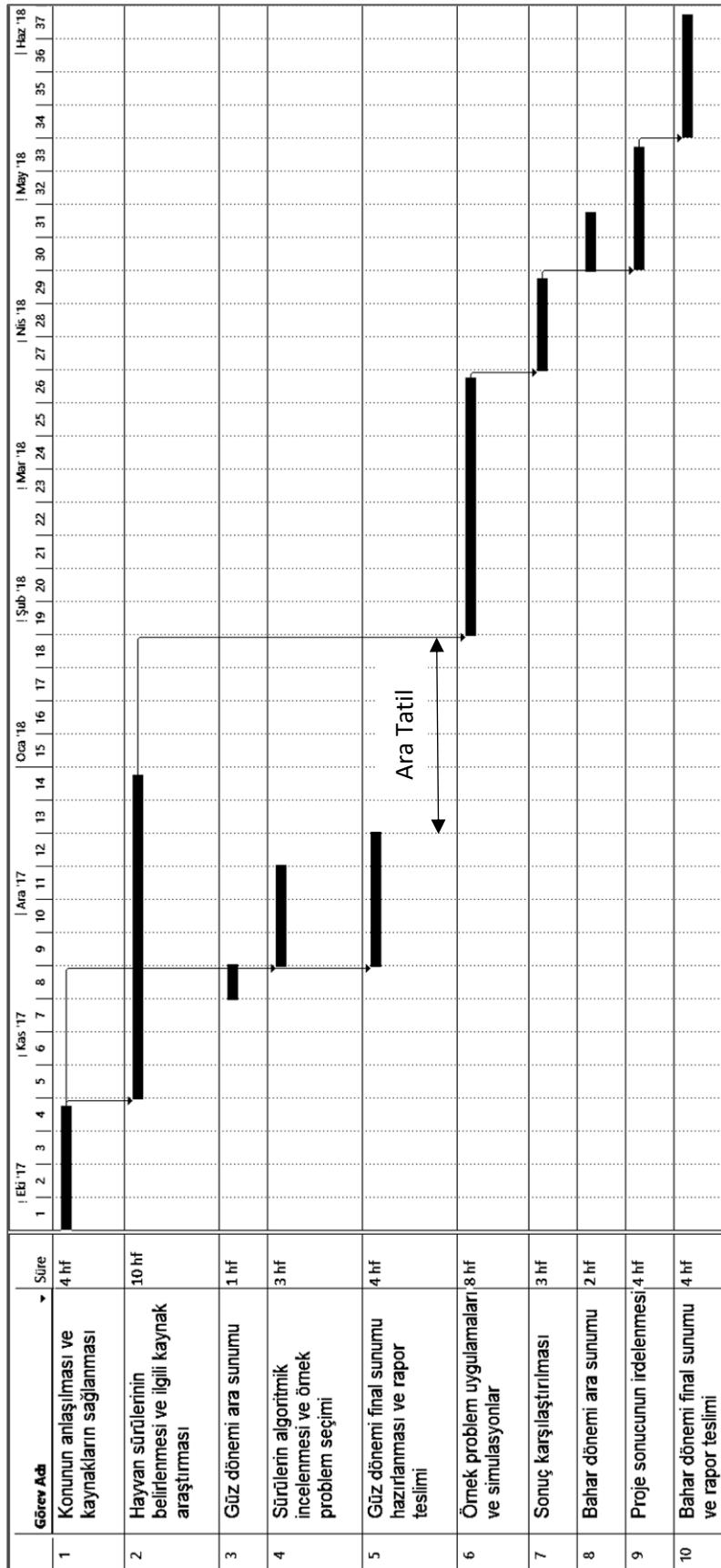
### **1.7 Bahar Dönemi Hedefleri**

Bahar dönemi hedefleri aşağıda sırasıyla verilmiştir:

- MATLAB kullanarak algoritmanın seçilen örnek probleme uygulanması ve ulaşılan sonuçların karşılaştırılması;
- Projenin sonucunun belirlenmesi ve irdelenmesi;
- Sonuçların Benchmark Fonksiyonları ile karşılaştırılmaları;
- Proje sürecinde yapılan tüm çalışmaların detaylı raporunun ve posterinin hazırlanması.

## BİL493-94 BITİRME PROJESİ 2017-2018 AKADEMİK YILI İŞ-ZAMAN PLANI

### 1.8 İş-Zaman Diyagramı



**Danışman:** Prof. Dr. A. Ziya Aktas  
**Öğrenciler :** Eray Altay  
 Figen Gizem Tosun  
 Mehmet Emre Yorulmaz  
 Nisanur Sungur

## **1.9 Görev Dağılımı**

Projemizin sunuları, raporu ve proje ile ilgili genel görevler birlikte yapılmıştır. Ekip lideri olarak Eray Altay seçilmiştir. Buna ek olarak her birimiz bir hayvan sürüsüyle ilgili algoritmayı inceledik.

İncelenen hayvan sürüleri ve görevli ekip üyeleri aşağıda belirtilmiştir:

- Eray Altay: Arılar ile sürü optimizasyonu;
- Nisanur Sungur: Karıncalar ile sürü optimizasyonu;
- Figen Gizem Tosun: Kuşlar ile sürü optimizasyonu;
- Mehmet Emre Yorulmaz: Kurtlar ile sürü optimizasyonu.

Yukarıda adı geçen başlıklar Bölüm 3,4,5 ve 6'da detaylı bir şekilde incelenmiştir.

## II OPTİMİZASYON NEDİR?

### 2.1 Genel Optimizasyon Tanımı

Optimizasyon, en iyileme anlamına gelmektedir. Bir problem için, verilen şartlar altında tüm çözümler arasından en iyi çözümü elde etme işidir. Belirli sınırlamaları sağlayacak şekilde, bilinmeyen parametre değerlerinin bulunmasını içeren herhangi bir problem, optimizasyon problemi olarak adlandırılabilir (Murty, 2003).

Bir optimizasyon işleminde önce karar parametrelerinin tanımlanması gereklidir. Sonra, bu parametrelere bağlı olarak bir maliyet fonksiyonu veya bir kar fonksiyonu ve sınırlama fonksiyonları tanımlanmalıdır. Maliyet fonksiyonu daha iyi çözümlerin parametrelerine daha düşük, Kar fonksiyonu ise daha yüksek değerler üretmektedir. Sınırlamalar ise, parametrelerin alamayacağı değerleri tanımlamaktadır. Sınırlamaların bazıları eşitlikler, bazıları ise eşitsizlikler biçiminde olabilir. Maliyet fonksiyonu ve sınırlamalar matematiksel formda n değişkenli bir vektor olarak aşağıdaki Denklem 2.1'deki gibi tanımlanabilir (Karaboğa, 2004).

$$f(x) = f(x^1, x^2, x^3, \dots, x^n) \quad (2.1)$$

Bu fonksiyonda  $x^i$ ,  $i$  parametresinin değerini göstermektedir.

Aynı zamanda aşağıdaki Denklem 2.2'de tanımlanan  $p$  tane eşitlik sınırlamaları gibi (Karaboğa, 2004).

$$H_j(x) = H_j(x^1, x^2, x^3, \dots, x^n) = 0, 1 \leq j \leq p \quad (2.2)$$

Veya aşağıdaki Denklem 2.3'de tanımlanan  $m$  tane eşitsizlik sınırlamaları gibi tanımlanabilir (Karaboğa, 2004).

$$G_i(x) = G_i(x^1, x^2, x^3, \dots, x^n) \leq 0, 1 \leq i \leq m \quad (2.3)$$

Bazı optimizasyon problemlerinde birden fazla maliyet fonksiyonu vardır, bu problemlere çok amaçlı optimizasyon problemi denir.

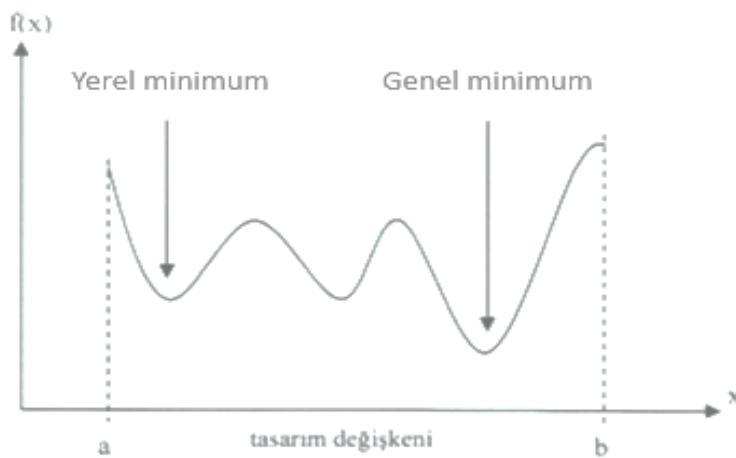
Optimizasyon problemlerinin sınırlarını sağlayan tüm çözümlerin oluşturduğu bölgeye uygun (feasible) çözüm bölgesi denir. Problemin optimum çözümü en düşük maliyet değerine ve en yüksek amaç fonksiyon değerine sahip çözümüdür. Bir maliyet fonksiyonu  $n$   $x^*$  çözümü için aşağıdaki verilen fonksiyonu verilen bölge içindeki tüm çözümler sağlıyor ise, bu fonksiyon  $x^*$  küresel minimuma sahiptir denir. Şekil 2.1'de ve Denklem 2.4'de gösterilmiştir (Karaboğa, 2004).

$$f(x^*) \leq f(x) \quad (2.4)$$

Aynı özelliklere sahip olan bir  $(x^*)$  fonksiyonu eğer dar bir komşuluğu ( $N$ ) içindeki tüm muhtemel çözümleri için sağlanıyorsa o fonksiyon bölgesel minimuma sahiptir denir.

Bu komşuluk  $\delta$  parametresine bağlı olarak aşağıdaki Denklem 2.5'deki gibi tanımlanabilir (Karaboğa, 2004).

$$N = \{x \mid x \in S \text{ ve } \|x - x^*\| < \delta\} \quad (2.5)$$



**Şekil 2.1** Bir fonksiyonun yerel (local) ve genel (global) minimum noktaları

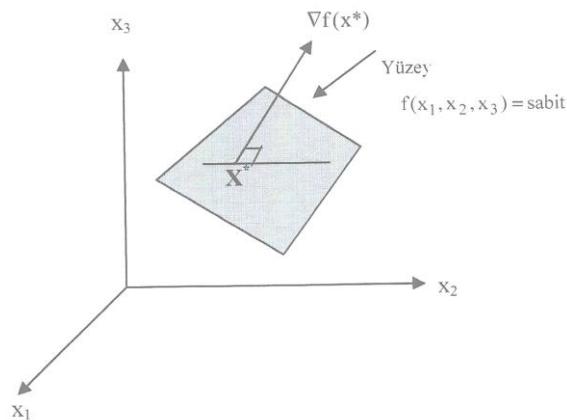
$f(x)$  fonksiyonunun  $x^*$  noktasındaki  $x^1$ 'e göre kısmi türevi Denklem 2.6'daki gibi tanımlanır (Karaboğa, 2004).

$$\frac{df(x)}{dx_1} |_{x=x^*} \quad (2.6)$$

Tüm kısmi türevlerin sütün vektörü formunda düzenlenmesi durumunda vector gradient vektörü olarak adlandırılır ve Denklem 2.7'de olduğu gibi gösterilir (Karaboğa, 2004).

$$\nabla f \text{ veya } \text{grad } f \quad (2.7)$$

Gradyent vektörü  $x^*$ noktasına teğet olan  $f(x) = \text{sabit}$  düzleme diktir ve  $f(x)$  fonksiyonunun en hızlı artış gösterdiği yönü belirlemektedir. Gradyent vektörünün bu özelliği, klasik gradiyente dayalı optimizasyon algoritmalarının temelini ortaya çıkarmaktadır. Üç değişkenli bir fonksiyon için bu durum Şekil 2.2'de gösterilmiştir (Karaboğa, 2004).



**Şekil 2.2** Gradiyent vektörü, x noktasına teğet olan düzleme dik konumdadır ve yönü de fonksiyonun en hızlı artış sağladığı doğrultuyu göstermektedir.

Gradyent vektörünün ikinci türevi ile hessian matris elde edilir, bu matris  $f(x)$  fonksiyonunu ikinci dereceden kısmi türevlerinden oluşmaktadır. Hessian matris, simetrik özelliğe sahiptir ve optimallik için yeterli şartların belirlenmesinde önemli rol oynar.

Optimum noktada sağlanması gereken şartlar, gerek şartlar olarak adlandırılır, ancak bu şartları sağlayamayan bir nokta optimum olamayacağı gibi bu şartların sağlanması bir noktanın optimum olmasını garantilemez. Bu yüzden gerek şartları sağlayan optimum ve optimum olmayan noktaları ayırt etmek için yeter şartlar kullanılır. Gerek şartlar sağlandıktan sonra yeter şartlar da sağlanıyorsa o noktanın optimum olduğu söylenir.

$x^*$   $f(x)$  fonksiyonunun bölgesel minimum olan bir nokta ve  $x$ ,  $x^*$ 'in yakın civarında digger bir nokta olsun. Bu durumda çözümlerin  $x^*$  civarındaki değişimi  $d$  ve bundan dolayı amaç fonksiyonunda ortaya çıkan  $f(x^*)$  civarındaki değişimi sırasıyla Denklem 2.8'deki gibi tanımlanabilir (Karaboğa, 2004).

$$d = x - x^* \text{ ve } \Delta f = f(x) - f(x^*) \quad (2.8)$$

$x^*$  bölgesel minimum olduğu için yakın civarında, yani küçük  $d$  değerleri için aşağıdaki Denklem 2.9'u sağlanması gereklidir (Karaboğa, 2004).

$$\Delta f = f(x) - f(x^*) \geq 0 \quad (2.9)$$

Bu eşitsizlik bölgesel minimum için gerek ve yeter şartların türetilmesi için kullanılabilir.  $d$  oldukça küçük olduğundan deltaf,  $x^*$  noktasında taylor serisine açılarak optimallik şartları türetebilir. Tek değişkenli fonksiyon için birinci dereceden gerek şart aşağıda Denklem 2.10'da belirtilmiştir (Karaboğa, 2004).

$$f'(x^*) = 0 \quad (2.10)$$

Minimumluk için yeter şart Denklem 2.11'de belirtilmiştir (Karaboğa, 2004).

$$f''(x^*) > 0 \quad (2.11)$$

Eğer  $f''(x^*) = 0$  ise  $x^*$ 'in bir minimum noktası olmadığı sonucuna varılmaz, bu durumda daha yüksek dereceden türevinin hesaplanması gereklidir. İkinci dereceden şart Denklem 2.12'de tanımlanmıştır (Karaboğa, 2004).

$$f''(x^*) \geq 0 \quad (2.12)$$

Bu şartı sağlayamayan bir nokta bölgesel minimum olamaz.  $f''(x^*) = 0$  ise, bu noktanın bölgesel minimum olabilmesi için gereken şartlar Denklem 2.13'de belirtilmiştir (Karaboğa, 2004).

$$f'''(x^*) = 0 \quad f''''(x^*) > 0 \quad (2.13)$$

Bu türev sıfırda eşitse bir üst dereceden türeve bakılmalıdır, yani çift dereceli türev sıfırdan küçükse aday nokta kesinlikle bölgesel minimum olamaz demektir ama sıfırda eşitse üst tek dereceli türevin sıfırda eşit ve bir üst çift derece türevin ise sıfırdan büyük olup olmadığı kontrol edilmelidir.

## **2.2 Optimizasyon Problemlerinin Sınıflandırılması**

Optimizasyon problemleri sınır durumları, lineerlik durumları ve süreklilik durumları gibi gruplar içerisinde incelenmektedir. Sahip olduğumuz  $f(x)$  fonksiyonunun,  $x$  ile ilgili herhangi bir sınırlama olmaksızın oluşan minimizasyon veya maksimizasyonuna ‘sınırlamasız optimizasyon’, sınırlamaya sahip minimizasyon veya maksimizasyonuna ise ‘sınırlı optimizasyon’ problemi denilmektedir.

Bir optimizasyon problemi, lineer amaç ve sınırlama fonksiyonlarına sahip ise bu problem ‘lineer optimizasyon’ problemi, bu fonksiyonlardan herhangi biri ‘non-lineer optimizasyon’ problemi olarak adlandırılır.

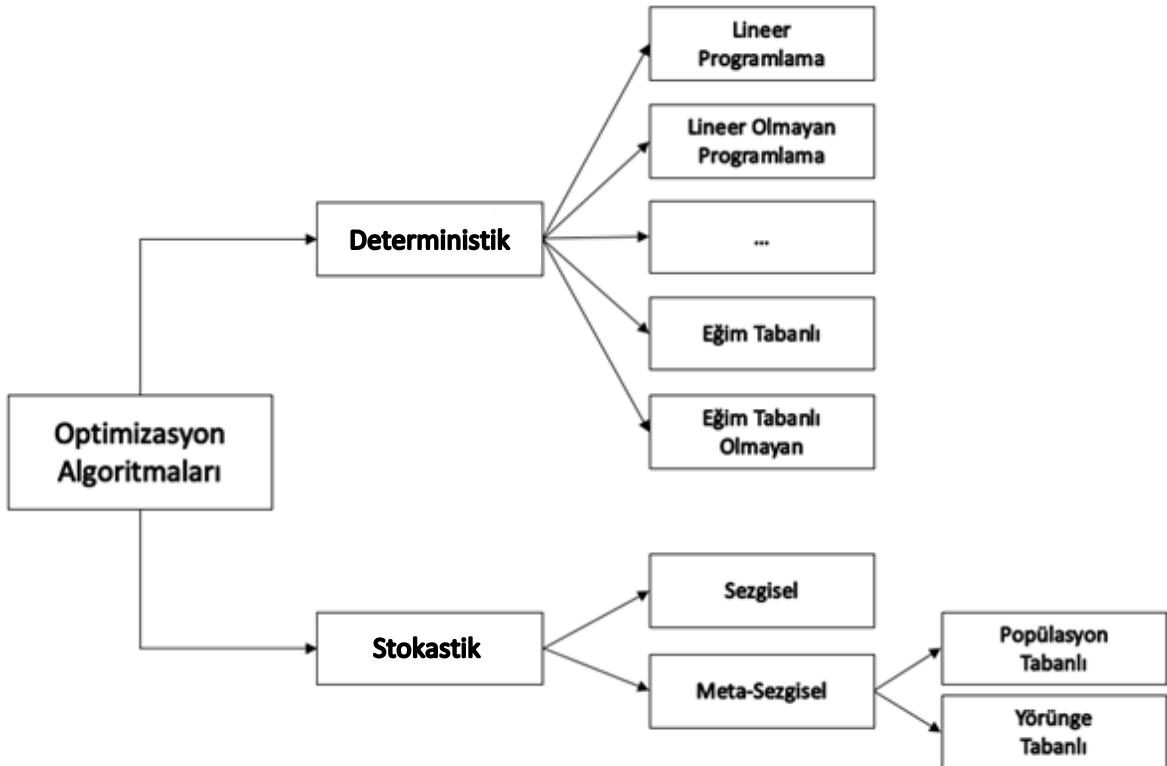
Diğer bir grulama şekli ise, değişkenlerin veya parametrelerin alacağı değerler sürekli ise bu tür problemlere ‘sürekli optimizasyon’ problemi denilmektedir. Ek olarak süreklilik grubunda incelenen ‘ayrık optimizasyon’ problemi ise ayrık niceliklerin düzenlenmesi, gruplanması, sıraya konulması veya seçilmesi üzerine olan problem grubudur.

Optimizasyon problemlerinde nonlinear kavramı vardır. Bu nonlinear kavramında, nonlinear problemleri çözmek için geliştirilen numerik methodlar vardır. Optimizasyon problemlerinde nonlinear programlamaya ihtiyaç duyulmasının sebepleri olarak şunlar söylenebilir:

- Değişkenlerin ve sınırlamaların çok fazla olması durumunda, oldukça fazla sayıda denklem oluşabilir ve bu nedenle gerekli şartların çözülmesi çok zor olabilir. Bu durum özellikle sınırlı optimizasyon problemlerinde ortaya çıkmaktadır.
- Problemelerin fonksiyonları yüksek seviyede nonlinearik içerebilir. Bu nedenle problemin boyutu küçük olsa bile gerekli şartlar da nonlinear yapıda olabilir.
- Çoğu mühendislik uygulamalarında fonksiyonlar tasarım değişkenleri açısından net olamayabilir. Böylelikle bağıntılar arasındaki bağ da net kurulamayabilir.

## **2.3 Optimizasyon Metotlarının Sınıflandırılması**

Optimizasyon problemlerinin çözümünde kullanılan algoritmalar iki kategoride değerlendirilebilir. Bunlardan ilki belirli bir prosedürü takip eden, takip edilen yolu, tasarım değişkenleri ve fonksiyon değerlerinin tekrarlanabildiği deterministik (rastgele olmayan, belirli) algoritmalardır. Yani algoritma ne zaman çalıştırılırsa çalıştırılsın aynı girdiler için aynı sonuçları her zaman üretebilen algoritmalar deterministik yaklaşımla açıklanmaktadır. Bu algoritmalar optimum sonucu bulabilse de özellikle çözüm uzayı büyük olan problemlerin çözümlerinde çok fazla zaman harcamaktadırlar. Diğer ise rassallığı içinde barındıran stokastik (olasılıksal) algoritmalardır. Bu algoritmalar Şekil 2.3’de gösterilmiştir (Karaboğa, 2004).



**Şekil 2.3** Optimizasyon algoritmaları şeması.

### 2.3.1 İteratif Araştırma İşlemi Algoritma Formu

- Bir başlangıç çözümü ( $x(0)$ ) alınır ve iterasyon sayacı sıfırlanır ( $t=0$ ).
- Araştırma uzayı için araştırma yönü belirlenir ( $d(t)$ ).
- Araştırmamanın yakınsamasını kontrol et. Eğer yakınsanmış ise araştırma işlemini bitir. Tersi mevcut ise araştırmaya devam et.
- Pozitif adım uzunluğunu hesapla ( $a(t)$ ).
- Yeni çözümü  $x(t+1) = x(t) + a(t)d(t)$  formülünü kullanarak belirle.
- Iterasyon sayacını bir arttır ve araştırma yönü belirlemeye geri dön.

Iteratif araştırma işleminde Mevcut çözümden bir sonraki çözüme geçiş adımları yukarıda belirtildiği gibidir ve görüldüğü üzere iteratif araştırma işlemi iki alt adımdan (araştırma yönünün ( $d^{(t)}$ ) ve adım uzunluğunun ( $a^{(t)}$ ) belirlenmesi) oluşur.

Araştırma yönü belirlendikten sonra adım büyüklüğünün hesaplanması tek boyutlu bir optimizasyon problemi haline gelir. Adım büyüklüğünün belirlenmesi için kullanılan metodlar iki gruba ayrılır: Analitik ve Nümerik metodlar.

En yaygın olarak kullanılan nümerik araştırma metodları şunlardır: En-Dik iniş metodu, eşlenik gradyent metodu, değiştirilmiş Newton metodu.

### **2.3.2 En-Dik İniş Metodu**

Bu metod en eski ve en basit araştırma yönü hesabı için kullanılan metottur. Ancak bu algoritmanın performansı çok iyi olmadığından genel uygulamalar için pek tavsiye edilmez. Bu metodun temel adımları ise aşağıdaki gibidir:

- Bir başlangıç çözümü ( $x(0)$ ) al ve iterasyon sayacını sıfırla ( $t=0$ ) ve durdurma kriteri için bir tolerans noktasını seç.
- $X(t)$  noktasında  $g(t)$  gradiyentini ve  $|g(t)|$ ’yi hesapla. Eğer  $|g(t)|$ , seçilen tolerans noktasından küçükse araştırmayı durdur. Değilse devam et.
- $D(t) = -g(t)$  şeklinde araştırmanın yönünü belirle.
- Yeni noktayı  $x(t+1) = x(t) + a(t)d(t)$  ifadesinden hesapla
- $T$ ’yi bir arttır ve ikinci noktadan devam et.

### **2.3.3 Eşlenik Gradiyent Metodu**

Bu da oldukça basit bir metoddür ve ilk metodun çok etkili şekilde değiştirilmesiyle elde edilmiştir. Bu metodun temel adımları ise aşağıdaki gibidir:

- Bir başlangıç çözümü ( $x(0)$ ) al ve iterasyon sayacını sıfırla ( $t=0$ ) ve bir yakınsama parametresi seç.  $D(t) = -g(t)$  olarak al ve eğer  $|g(t)| <$  yakınsama parametresi ise araştırmayı durdur. Değilse direk dördüncü adıma git.
- $X(t)$  noktasında gradiyent  $g(t)$  ve  $|g(t)|$ ’yi hesapla. Eğer  $|g(t)| <$  yakınsama parametresi ise araştırmayı durdur. Yoksa devam et.
- Eşlenik gradiyent yönü  $d(t) = -g(t) + b(t)d(t-1)$  bağıntısından hesapla. ( $b(t) = (|g(t)| / |g(t-1)|) / 2$ )
- $F(x(t) + \alpha(t)d(t))$  ifadesini minimize edecek şekilde adım büyüklüğünün değerini hesapla.
- Yeni noktayı  $x(t+1) = x(t) + \alpha(t)d(t)$  ifadesinden faydalılarak belirle ve  $t=t+1$  alarak ikinci adıma git.

### **2.3.4 Modifiye Edilmiş Newton Metodu**

Standart Newton metodu araştırmanın yönünü belirlemek için maliyet fonksiyonunun Hessian matrisini yani ikinci dereceden türevlerini kullanır. Bu metodun temel adımları ise aşağıdaki gibidir:

- Bir başlangıç çözümü ( $x(0)$ ) al ve iterasyon sayacını sıfırla ( $t=0$ ) ve durdurma kriteri için bir tolerans ( $\epsilon$ ) seç.
- $X(t)$ ’de gradiyent  $g(t)$  ve  $\|g(t)\|$ ’yi hesapla. Şayet,  $(\|g(t)\|) < \epsilon$  ise araştırmayı durdur. Yoksa, devam et.
- Hessian matrisi  $H(x(t))$ ’yi hesapla.
- $D(t) = -H^{-1}g(t)$  ifadesinden faydalananarak araştırmanın yönünü belirle.
- $X(t+1) = x(t) + \alpha(t)d(t)$  ifadesinden faydalananarak yeni çözümü hesapla.
- $T = t + 1$  olarak al ve ikinci adıma git.

En-dik iniş metodunda, araştırma yönünü belirlemek amacıyla maliyet fonksiyonun sadece birinci dereceden türev bilgisi kullanılmaktadır. İkinci dereceden türev bilgisi var ise maliyet yüzeyini bu tür bilgiyi kullanarak daha hassas yapmak amacıyla araştırma yönü daha doğru olarak belirlenebilir. İkinci dereceden bilginin kullanılması ile daha iyi bir yakınsama oranı sağlanabilir. Newton metodunun temel fikri, mevcut nokta etrafında fonksiyonun ikinci dereceden Taylor seri açılımını kullanmaktadır. Bu fikir, yeni çözüme ulaşmak için eski çözümde yapılması gereken değişimi tanımlamak için karesel bir ifade üretir. Bu metodun birinci ve ikinci dereceden türevlerinin olduğu kabul edilir. Öncelikle gradyent ( $g$ ) ve Hessian ( $H$ ) hesaplanır ve aşağıdaki Denklem 2.14 kullanılarak yeni çözüm elde edilir (Karaboga, 2004).

$$x(t+1) = x(t) + \Delta x(t) \text{ ve } \Delta x(t-1) = -\alpha[H(x(t))]^{-1} g(x(t)) \quad (2.14)$$

Burada  $H$ , Hessian matrisini temsil etmektedir ve bu matris tekil değildir.

Klasik Newton metodları, gradyent tabanlı bilgi aracılığıyla belirlenen doğrultuda adım büyüklüğünü 1 alır. Bu yüzden bu metod her problem için yakınsamayı garanti etmez. Bu dezavantajı ortadan kaldırmak için Değiştirilmiş Newton metodu önerilmiştir.

Değiştirilmiş Newton metodu da çeşitli zorluklara sebep olacak dezavantajlara sahiptir. Bu yüzden Marquardt, yön bulma işleminde bir değişiklik önermiştir (1963). Bu değişiklik sayesinde algoritma hem en-dik iniş algoritmasını hem de Newton metodunun avantajlarına sahip hale gelmiştir. Yeni metot, çözüm noktasının uzaklarında en-dik iniş algoritması gibi davranışırken çözüm civarında Newton metodу gibi davranışmaktadır. Bu yeni metot da Hessian matrisi aşağıdaki Denklem 2.15'deki gibi hesaplanmaktadır (Karaboga, 2004).

$$d(t) = -(H + \lambda I)^{-1} g(t) \quad (2.15)$$

Burada  $\lambda$ , pozitif değerli bir sabittir. Bu sabit yeterince büyük bir değere sahip olduğunda  $H$ 'ın etkisi ihmali edilebilir. Bu yön, en-dik iniş yönüne eşit olmaktadır. Burada  $(1/\lambda)$ , adım büyüklüğüne karşı gelmektedir.  $\lambda$  yeterince küçükse o zaman  $\lambda I$  terimi ihmali edilebilir ve yön Newton metodunda kullanılan yöne ve algoritma da Newton algoritmasına eşdeğer hale gelir.

Yarı-Newton metotlar Newton metoduna dayalı metotlardır. Aralarındaki fark Hessian matrisinin elde ediliş şevidir. Hessian matrisinin hesaplanması ve tersinin alınması uzun sürediği için, Yarı-Newton metotları da bu işlemlerin tam olarak yapılması yerine, sadece birince dereceden türevler kullanılarak ikinci dereceden türevlerin yaklaşıkları elde edilir. Temel fikir, çözüm ve gradyent vektörü ile ilgili değişimleri kullanarak mevut yaklaşılığı yenilemektir. Bu sınıfın yaygın olarak kullanılan algoritmalarından biri Davidon-Fletcher Powell (DFP) algoritmasıdır (Fletcher ve Powell, 1963).

Yukarıda anlatılan sınırlamasız optimizasyon problemleri için geliştirilen metotlar olduğu gibi, sınırlamalı optimizasyon problemleri için de metotlar geliştirilmiştir. Örneğin, Simplex metodu. Bu metot, bir lineer denklemler setinin çözümü için kullanılan standart Gauss-Jordan eliminasyon işleminin geliştirilmiş formudur. Bazı problemler çok amaçlı olabilir ve bu amaçları sağlayacak çözüm bulunamayabilir. Böyle durumlarda “hedef programlama” teknikleri kullanılmaktadır. Bunlara örnek olarak “Branch-and-bound” teknikleri verilebilir (Arora, 1989).

Sınırlamalı optimizasyon problemlerde genellikle, ilk önce Taylor seri açılımı vasıtasyyla nonlinear problem lineerleştirilir. Sonra bu problem Quadratik Programlama (QP) problemlerine dönüştürülür. QP problemleri simplex metodunun geliştirilmiş formları ile çözülebilir.

Geçmiş yıllarda araştırmacılar, özellikle ayrik optimizasyon sahasındaki problemlerin çözümü için büyük çaba sarf etmişlerdir. Ayrik optimizasyon, ayrik objelerin optimal olarak düzenlenmesinin, gruplandırmasının, sınırlanmasının veya seçilmesinin matematiksel çalışılması olarak tarif edilebilir. Ayrik problemlerin çoğu oldukça zor problemlerdir. Bu yüzden, büyük boyutlu problemlerin çözümünde uygun hesaplama süresi içerisinde optimal çözümleri bulabilen yaklaşık algoritmaları (approximate algorithms) geliştirmek ve kullanmak hale gelmiştir. Bu yaklaşım genellikle biyoloji, zooloji, fizik, bilgisayar ve karar üretme bilimlerinden türetilmiştir (Reeves, 1995; Pham ve Karaboga, 2000).

## 2.4 Optimizasyon İçin Sezgisel Algoritmalar

Sezgisel (heuristic) kavramı deneme yanılma yoluyla çözümün bulunması olarak ifade edilebilir. Optimizasyon problemlerinde deterministik algoritmaların代替に çok daha makul zamanlarda kaliteli çözümleri bulabilen sezgisel algoritmalar, her zaman optimal çözümü bulma garantisini veremeyebilirler. Aynı zamanda sezgisel yöntemler, çözüm uzayının tümünü ele almadan sezgisel bir şekilde çok kısa sürelerde optimum sonuçlara ya da optimuma çok yakın sonuçlara ulaşabilmektedirler.

Sezgisel algoritmalar, herhangi bir amacı gerçekleştirmek veya hedefe varmak için doğal fenomenlerden esinlenen algoritmalarıdır.

Anlaşırlık yönünden sezgisel algoritmaların karar verici açısından çok daha basit olabilirlerinden, optimizasyon problemlerinin kesin çözümü bulma işleminin tanımlanamadığı bir yapıya sahip olmasından, öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanabildiğinden sezgisel algoritmaları ihtiyaç duyulmaktadır.

Sağ-el kuralına dayalı olan bu algoritmalar, bir amacı gerçekleştirmek için alternatiflerden en etkili olanlara karar vermek amacıyla tanımlanan kriterler veya bilgisayar metodlarıdır. Bu tür algoritmalar yakınsama özelliğine sahiptir, ancak kesin çözümü garanti edemezler.

### 2.4.1 Sezgisel Algoritmalarla Gerek Duyulmasının Sebepleri

- Optimizasyon problemi, kesin çözümü bulma işleminin tanımlanamadığı bir yapıya sahip olabilir.
- Sezgisel algoritmalar, anlaşırlılık açısından karar verici için daha basit olabilir.
- Sezgisel algoritmalar, öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanılabilir.
- Matematik formülleriyle yapılan tanımlamalarda genellikle gerçek dünya problemlerinin en zor tarafları ihmal edilir. Model parametrelerini belirlemeye kullanılan verinin hatalı olması, sezgisel yaklaşımın üretebileceği alt optimal çözümden daha büyük hatalara sebep olabilir.

## 2.4.2 Sezgisel Algoritmaların Değerlendirilmesi İçin Kriterler

- **Çözüm kalitesi ve hesaplama zamanı:** Çözüm kalitesi ve hesaplama zamanı bir algoritmanın etkinliğinin değerlendirilmesi için önemli kriterlerdir. Bundan dolayı bir algoritma ayarlanabilir parametreler setine sahip olmalı ve bu parametreler kullanıcıya etkinlik açısından hesaplama maliyeti ile çözüm kalitesi arasında bir vurgulamanın yapılabilmesine imkân vermelidir.
- **Kod basitliği ve gerçeklenebilirlik:** Algoritma prensipleri basit olmalı ve genel olarak uygulanabilir olmalıdır. Bu durum problem yapısı ile ilgili başlangıçta çok az bilgiye sahip olunması halinde bile algoritmanın yeni alanlara kolaylıkla uygulanabilmesini sağlar.
- **Esneklik:** Algoritmalar modelde, sınırlamalarda ve amaç fonksiyonlarında yapılacak değişiklikleri kolayca karşılayabilmelidir.
- **Dinçlik:** Yöntem başlangıç çözümünün seçimine sahip olmaksızın her zaman yüksek kaliteli, kabul edilebilir çözümleri üretme kabiliyetine sahip olmalıdır.
- **Basitlik ve analiz edilebilirlik:** Karmaşık algoritmalar, esneklik ve çözüm kalitesi açısından basit algoritmalardan daha zor analiz edilebilmektedir. Algoritma kolayca analiz edilebilir olmalıdır.
- **Etkileşimli hesaplama ve teknoloji değişimleri:** Algoritma içinde insan-makine etkileşimi kullanma fikri çoğu sistemde yaygın olarak gerçekleştirilmektedir. Bunun en önemli avantajı çözümleri grafiksel olarak sergilenebilmesidir.

Sezgisel algoritmalarının performanslarını karşılaştırma aşamasında farklı algoritmalar aynı problem üzerinde çalıştığında çözüme ulaşma zamanı, hafıza maliyeti gibi detayların göz önünde bulundurulması gerekmektedir. Sezgisel algoritmaları incelerken “no free lunch” isimli teorem bu karşılaştırma konusunda önemli bir kural ortaya koymaktadır.

Bu teorem farklı algoritmaların tüm problemler üzerindeki performanslarının ortalamalarının eşdeğer olduğunu belirtmektedir. Bir optimizasyon algoritması bir problemi yüksek bir performansta çözebildiği gibi başka bir problemde çok düşük bir performans gösterebilir. Bu durum bir optimizasyon algoritmasının bütün problemlerde başarılı olamayacağını göstermektedir.

## 2.4.3 Sürü Zekası Tabanlı Sezgisel Algoritmalar

Sürü zekâsı birlikte yaşayan bireylerin veya organizmaların bir problemi çözmek için ortaya koyduğu tecrübe ve bilgi birikimidir. Sürü zekâsını oluşturan bireyler merkezi bir hiyerarşi ile kontrol olunmazlar ve kollektif tecrübeye katkıda bulunurlar. Nitekim herhangi bir karıncanın geçtiği yol üzerine feromon adı verilen bir kimyasal madde bırakması diğer karıncaları etkileyerek ya yiyecek kaynağının ya da yuvanın bulunmasına yardımcı olması bakımından önemlidir. Birey olarak zeki olmayan canlıların sürü olarak hayatlarını idame ettirebilmeleri için ortaya koydukları iş bölümü, bilgi paylaşımı vb. zeki davranışları araştırmacıların ilgisini çekmiş ve çeşitli sürülerin bu davranışlarını temel alan yöntemler ile gerçek dünya problemlerini çözmeye çalışmışlardır. Karıncaların yiyecek kaynağı ve yuva arasındaki zeki davranışları (feromon bırakma, feromon takip etme gibi) karınca kolonisi optimizasyon algoritmasının, kuş veya balık sürülerinin sosyal davranışları parçacık sürü optimizasyonunun ve bal arısı kolonilerinin yiyecek toplama ve bilgi paylaşımı davranışları da yapay arı kolonisi algoritmasının temelini oluşturur.

Sürü zekâsına dayanan yöntemler pozisyon güncellemeyi temel alırlar ve evrimsel hesaplama tekniklerindeki yeni bir nesil oluşturmak için gerekli operatörleri kullanmazlar. Bundan dolayı sürü zekâsı yöntemlerinde genetik operatörler (çaprazlama ve mutasyon) bulunmamaktadır. Bu operatörler yerine sürü zekâsına dayanan yöntemlerde uzayın verimli şekilde araştırılmasını sağlayacak pozisyon güncelleme teknikleri/kuralları/denklemleri bulunmaktadır.

#### **2.4.3.1 Çözümün Gösterimi**

Probleme ait olası çözümlerin evrimsel veya sürü zekâsına dayanan yöntemlerin hesap yapabileceği şekilde kodlanması gerekmektedir. İkili kodlama, tam sayı kodlama, ağaç gösterimi, sürekli kodlama formları çözümün gösterimi için kullanılabilir.

Optimizasyon probleminin yapısına bağlı olmak şartıyla bu kodlamalar arasında da dönüşüm yapılabilir. Örneğin ikili bir optimizasyon problemi için problemin uygun çözüm kodlaması ikili kodlama olarak görülmektedir fakat sürekli kodlama kullanılarak algoritma işlemlerini sürdürübilir ve amaç fonksiyonu değerlendirilmeden hemen önce ikili kodlamaya geçilebilir. Aynı şekilde sürekli bir gösterim yerine ikili gösterim kullanılarak (özellik genetik algoritma için) genetik operatörler koşturulabilir fakat amaç fonksiyonu değerlendirilmeden önce kodlama sürekli hale getirilebilir. Kısacası çözümün gösterimi optimizasyon problemine bağlı olduğu kadar, yönteme ve geliştiricinin seçimine de bağlıdır. Bu noktada seçici yöntemin performansı iyileştiren kodlamayı tercih etmeye dikkat etmelidir.

#### **2.4.3.2 Çözümün Uygunluğu**

Belirli bir gösterimle kodlanmış bireylerin çözüm kalitesini ölçmek amacıyla uygunluk fonksiyonu kullanılır. Optimizasyon probleminin amaç fonksiyonu çözümün uygunlığını belirlemek amacıyla kullanılabileceği gibi yönteme özel bazı durumlardan (seçim mekanizmalarının çalışabilmesi gibi) dolayı amaç fonksiyonunu da kullanan farklı uygunluk fonksiyonları geliştirilebilir.

#### **2.4.3.3 Popülasyon**

Birden fazla uygun veya uygun olmayan çözümlerin oluşturduğu kümeye popülasyon adı verilir. Popülasyon kromozomlardan, parçacıklardan, yapay karıncalardan, yapay arılarından veya yiyecek kaynaklarından oluşabilir. Popülasyon yöntem içerisinde çözümlerin (karar setinin) tutulması için kullanılır. Popülasyon için bazı durumlar söz konusudur ve popülasyon yöntemin verimli çalışabilmesi için bu durumlardan kurtarılmalıdır. Örneğin popülasyon yeni çözüm üretmeyecek duruma (durağanlaşma – stagnation) gelebilir veya yerel minimumlara takılabilir. Etkili bir yöntemede bu durumlarla başa çıkabilecek ve popülasyonun durumunu kontrol edecek mekanizmalar bulunmalıdır.

#### **2.4.3.4 Seçim Mekanizmaları**

Araştırmancıların sürdürülebilmesi yani aday çözümlerin veya yeni nesillerin oluşturulabilmesi için ebeveyn çözümlere ihtiyaç vardır. Yeni çözümlerin oluşturulabilmesi için hangi ebeveyn veya aktüel çözümlerin kullanılacağıının belirlenmesi gerekmektedir. Bunun için seçim mekanizmaları uygulanır. Bu seçim mekanizmaları çözümün kalitesine bağlı olabileceği gibi rastgele de olabilir. Ayrıca bu seçim mekanizmaları komşu çözümlerin belirlenmesinde de kullanılabilir.

#### **2.4.3.5 Yeni Çözümün Üretilmesi**

Yeni çözümün elde edilmesi evrimsel tekniklerde çaprazlamayla, sürü zekâsına dayanan yöntemlerde ise pozisyon güncellemesiyle sağlanır. Evrimsel tekniklerde ise iyi çözümlerin çaprazlanmasıyla daha iyi çözümlerin elde edilmesi amaçlanır. Sürü zekâsına dayanan yöntemlerde ise popülasyondaki çözümlerin kullanılmasıyla (popülasyonun en iyi çözümü, o ana kadar elde edilmiş en iyi çözüm, birey tarafından elde edilmiş en iyi çözüm vb.) yeni çözümler elde edilir. Bunun için genellikle farka dayalı yöntemler kullanılmaktadır ve rassallık da bu fark tabanlı işlemin içeresine entegre edilmiştir.

#### **2.4.3.6 İkkilendirme**

Sezgisel tekniklerde ilklendirme genellikle rastgele yapılır ve basitçe hesaplanabilir. Yöntemlerin yetenekleri test edilirken başlangıç şartlarına bağlılığı da test edilmektedir. Bu bağlamda düşünüldüğünde verimli ve etkili bir yöntemin başlangıç durumlarına karşı duyarsız olması beklenmektedir, yani farklı başlangıç durumlarında da iyi bir çözümün elde edilebilmesini yöntem sağlamalıdır.

#### **2.4.3.7 Sonlandırma**

Sezgisel yöntemler algoritma olarak adlandırılmaktadır ve algoritmalar sonlu bir işlem kümесini ifade ederler. Bu bağlamda kara kutu optimizasyon (black-box optimization) problemleri için belirli bir çevrim sayısı veya amaç fonksiyonun değerlendirilme sayısı durdurma kriteri olarak kullanılabilir. Sezgisel yöntemler kullanılarak eğitilen bir sınıflandırıcıda ise sezgisel yöntemin durdurma kriteri olarak sınıflandırıcının eğitimdeki başarısı veya hatası kullanılabilir.

### III ARILAR İLE SÜRÜ OPTİMİZASYONU

Doğal bir arı kolonisinde yapılacak o iş için özelleşmiş arılar tarafından yapılır. Yapılacak işlere göre arılar bir iş bölümü vardır ve herhangi bir merkezi otorite olmadan bu iş dağılımını gerçekleştirdikleri için kendi kendilerine organize olabilmektedirler. İş bölümü yapabilme ve kendi kendine organize olabilme sürü zekasının iki önemli özelliğidir.

#### 3.1 Yapay Arı Koloni Algoritması (YAKA) (Artificial Bee Colony Algorithm-ABC)

Doğada var olan zeki davranışlar içeren süreçlerin incelenmesi araştırmacıları yeni optimizasyon metotları geliştirmeye sevketmiştir. Arıların yiyecek arama davranışları modellenerek Yapay Arı Kolonisi (Artificial Bee Colony, ABC) algoritması geliştirmiştir.

Bu modele ait süreç adımları aşağıdaki gibi verilebilir:

- Yiyecek arama sürecinin başlangıcında, kaşif arılar çevrede rastgele arama yaparak yiyecek aramaya başlarlar.
- Yiyecek kaynakları bulunduktan sonar, kaşif arılar artık görevli arı olurlar ve buldukları kaynaklardan kovana nektar taşımaya başlarlar. Her bir görevli arı kovana dönüp getirdiği nektarı boşaltır ve bu noktadan sonar ya bulduğu kaynağa geri döner ya da kaynakla ilgili bilgiyi dans alanında sergilediği dans aracılığı ile diğer arılara aktarır.
- Kovanda bekleyen gözcü arılar zengin kaynakları işaret eden dansları izlerler ve yiyeceğin kalitesi ile orantılı olan dans frekansına bağlı olarak bir kaynağı tercih ederler. ABC algoritmasının bu süreçleri ve temel adımları aşağıda verilmektedir.

Yiyecek arayan arılarda görülen zeki davranış ile bu davranışları simüle eden ABC algoritmasının birimleri takip eden alt bölümlerde açıklanmaktadır.

##### 3.1.1 ABC Algoritmasının Temel Adımları

**Adım 1.** Başlangıç yiyecek kaynağı bölgelerinin üretilmesi.

##### REPEAT

**Adım 2.** Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi.

**Adım 3.** Olasılıksal seleksiyonda kullanılacak olasılık değerlerinin görevli arılardan elen bilgiye göre hesaplanması.

**Adım 4.** Gözcü arıların olasılık değerlerine göre yiyecek kaynağı bölgesi seçimleri.

**Adım 5.** Bırakılacak kaynakların bırakılması ve kaşif arı üretimi.

**UNTIL** (çevrim sayısı = maksimum çevrim sayısı)

Arama uzayının yiyecek kaynaklarını içeren kovan çevresi olarak düşünürsek, algoritma arama uzayındaki çözümlere karşılık gelen rastgele yiyecek kaynağı yerleri üreterek çalışmaya başlamaktadır. Rastgele yer üretme süreci her bir parametrenin alt ve üst sınırları arasında rastgele değer üreterek gerçekleştirilir. Aşağıda Denklem 3.1'de gösterilmiştir (Karaboğa, 2004).

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

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

Arama uzayında çözüm değerleri araştırılırken işçi arılar besin kaynaklarından bir tanesini rastgele olarak belirlerler ve bu besin kaynağının kalitesini yani çözüm değerini hesaplarlar.

Elde edilen çözüm değeri hafızaya alınır. Daha sonra işçi arılar besin kaynaklarına yöneldikçe hafızadaki bilgiler problemin amacına göre güncellenerek hafızada korunmaya devam edilir. Burada çözüm değerini iyileştiren değerlerin hafızada tutulacağını hatırlatmakta fayda vardır. Bu durum aşağıda yer alan Denklem 3.2'de yer almaktadır (Karaboğa, 2004).

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

Denklemde yer alan  $v_{ij}$  ile parametrelerin önceden belirlenmiş olan parametre sınırları arasında yer olması sağlanmaya çalışılmaktadır. Bu durum Denklem 3.3'de yer almaktadır (Karaboğa, 2004).

$$v_{ij} = \begin{cases} x_{ij} & , v_{ij} < x_j^{\min} \\ v_{ij} & , x_j^{\min} \leq v_{ij} \leq x_j^{\max} \\ x_j^{\max} & , v_{ij} > x_j^{\max} \end{cases} \quad (3.3)$$

Bu bilgiler ışığı altında besin kaynağının uygunluk değeri Denklem 3.4'e göre hesaplanır (Karaboğa, 2004).

$$u_i = \begin{cases} 1 / (1 + u_i) & , u_i \geq 0 \\ 1 + \text{mutlak değer}(u_i) & , u_i < 0 \end{cases} \quad (3.4)$$

Burada  $U_i$  ile besin kaynağının uygunluk değeri ifade edilmektedir. Uygunluk değerinin hesaplanması problemin yapısı ön plana çıkmaktadır yani problemin maksimizasyon ya da

minimizasyon olması durumuna göre uygunluk hesaplaması yapılmaktadır. Uygunluk değerine göre seçilen besin kaynağının çözüm değeri hesaplanır. Eğer elde edilen çözüm önceki çözümden daha kötü ise sayaç bir arttırılarak önceden belirlenmiş olan limit değeri ile karşılaştırılır. Aksi halde önceki çözüm değerinden daha iyi bir çözüm değeri elde edilmesi durumunda ise sayaç sıfırlanır. Daha önce de belirtildiği üzere limit değeri ile karşılaştırma yapılmasının nedeni artık daha fazla iyileştirilemeyen besin kaynaklarını değerlendirme dışı bırakarak sonsuz döngüye girmeye engel olmaktadır. Bu noktada kovanda bekleyerek dans alanındaki işçi arıları izleyen gözcü arılar önceden hesaplanmış olan uygunluk değerine göre ilgili besin kaynağına yöneleceklərdir. Uygunluk değerlerinin hesaplanmasıda çeşitli yöntemler mevcuttur. Bunlar rulet tekerliği seçim yöntemi, sıralamaya dayalı seçim yöntemi, stokastik örneklemme, turnuva yöntemi gibi yöntemlerdir. ABC algoritmasında rulet tekerliği seçim yöntemi kullanılmıştır. Bu yöntemde rulet tekerliği bir pasta gibi düşünülebilir. Pastanın her bir dilimi bir uygunluk değerine denk gelmektedir dolayısıyla uygunluk değeri yüksek olan çözüm değerinin seçilme olasılığı diğerlerinin seçilmesi olasılığından daha yüksektir. Aşağıda yer alan denklemde rulet tekerliği seçim yönteminde seçim olasılığının hesaplanması şekli Denklem 3.5'de yer almaktadır (Karaboğa, 2004).

$$\rho_i = \frac{uygunluk_i}{\sum_{j=1}^{SN} uygunluk_j} \quad (3.5)$$

Yukarıdaki Denklem 3.5'de uygunluk ile  $i$ . kaynağın uygunluk değeri,  $SN$  ile işçi arı sayısı ifade edilmektedir. Yani hesaplanan uygunluk değerinin toplam uygunluk değerine oranlanması ile pastanın dilimlerinin bir diğer ifade ile rulet tekerliğinde yer alan parçaların genişlikleri elde edilmiş olmaktadır.

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

Yukarıda yer alan ikinci ve üçüncü aşama yani işçi arıların besin kaynaklarına gönderilmesi ile gözcü arıların besin kaynaklarına gönderilmesi aşamaları tamamlandıktan sonra eğer sayaç limit değerini aşmışsa yani artık çözüm değeri daha fazla iyileştirilemiyorsa kâşif arılar görevi devralırlar. Gerçek hayatı bu durumu şöyle açıklamak mümkündür. Bir besin kaynağının nektarı tükenmişse nektarin çıkarılmasından sorumlu olan işçi arı yeni besin kaynaklarının araştırılmasından sorumlu olmak üzere kâşif arı olmaktadır. İşte aynı gerçek arıların besin arama davranışlarında olduğu gibi ABC algoritmasında da belirli bir limit adedince iyileştirilemeyen çözüm değeri için kâşif arılar üretilmekte ve bu kâşif arılar aracılığıyla yeni bir besin kaynağı oluşturulup bu besin kaynağının çözüm değeri hesaplanmaktadır. Oluşturulan yeni besin kaynağının çözüm değeri önceki çözüm değeri ile karşılaştırılmakta ve elde edilen çözüm değeri iyiise hafızaya alınmakta aksi takdirde ihmâl edilmektedir. Bütün bu adımlar önceden belirlenmiş olan döngü adedince gerçekleştirilerek durdurma kriteri sağlandığında algoritma sonlandırılarak döngüden çıkarılır. Şekil 3.1'de akış diyagramı olarak verilmiştir (Yuan, 2012).

Bütün bu seleksiyon metodlarının bir arada kullanılmasıyla ABC algoritması hem iyi bir küresel araştırma hem de bölgesel araştırma yapabilmektedir.

### **3.1.2 ABC Algoritmasının Temel Özellikleri**

ABC algoritması ile ilgili temel özellikler aşağıdaki gibi özetlenebilir.

- Oldukça basit ve esnektir.
- Gerçek yiyecek arayıcı arıların davranışlarını oldukça yakın şekilde simüle eder.
- Sürü zekasına dayalı bir algoritmadır.
- Nümerik problemler için geliştirilmiş olmasına rağmen ayrik problemler için de kullanılabilir.
- Oldukça az control parametresine sahiptir.
- Kaşif arılar tarafından gerçekleştirilen küresel ve görevli ile gözcü arılar tarafından gerçekleştirilen bölgesel araştırma kabiliyetine sahiptir ve iki araştırmayı da dengeli bir şekilde yürütmektedir.

## **3.2 Arıların Yiyecek Arama Davranışları**

Kollektif zekanın ortaya çıkışmasını sağlayan minimal yiyecek arama modelinde temel üç bileşen vardır: Yiyecek kaynakları, görevli işçi arılar ve görevsiz işçi arılar. Bu minimal model iki şekilde çalışmaktadır: Bir yiyecek kaynağına yönelme, kaynağı bırakma.

### **3.2.1 Yiyecek Kaynakları**

Arıların, nektar, polen veya bal elde etmek için gittikleri kaynaklara verilen isimdir. Yiyecek kaynağının değeri, çeşidi, kovana yakınlığı, nektar konsantrasyonu veya nektar çıkarılmasının kolaylığı gibi bir çok faktöre bakılarak kaynak seçilir.

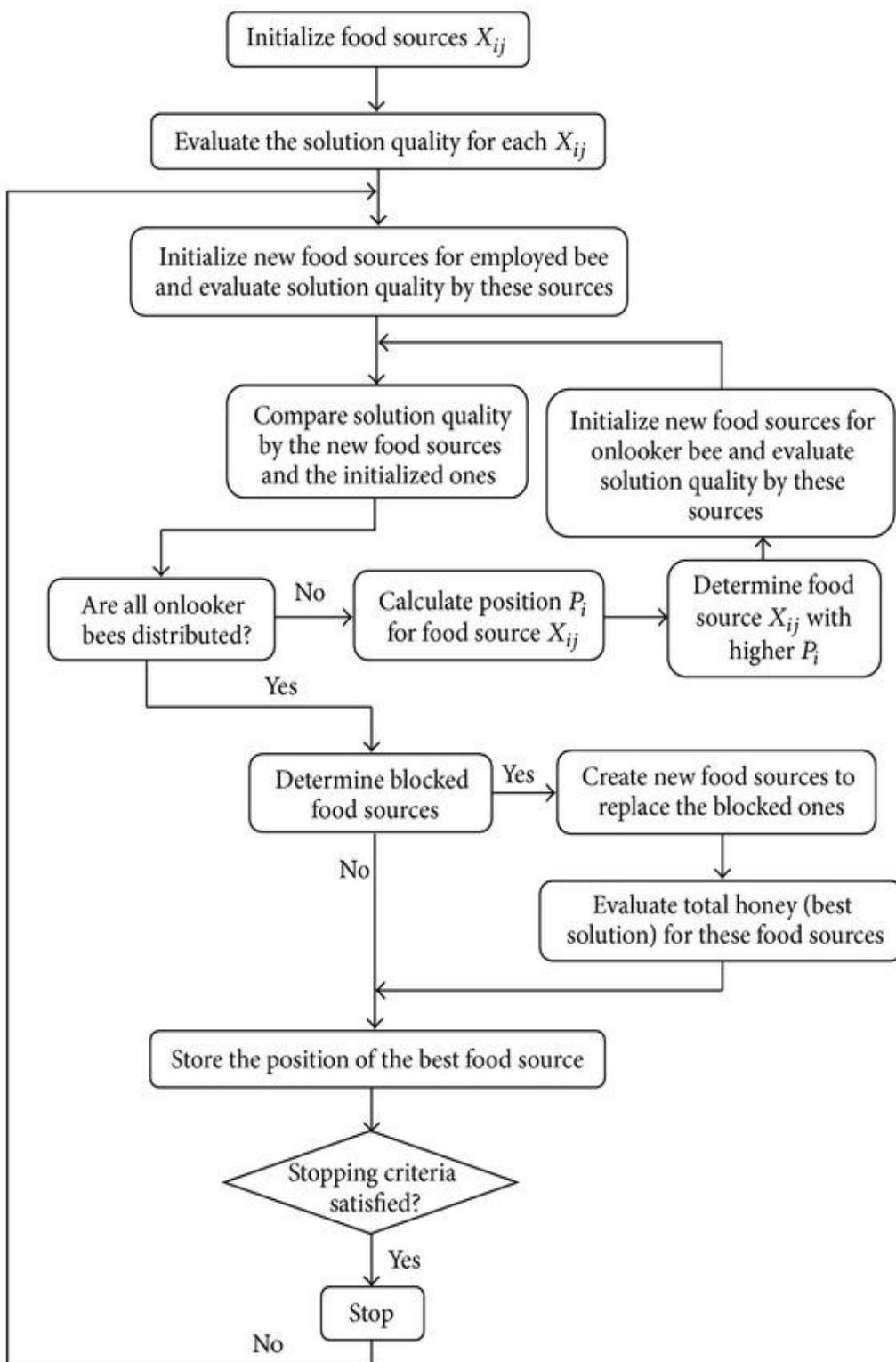
### **3.2.2 Görevli İşçi Arılar**

İşçi arılar, daha önceden keşfedilen belli kaynaklara ait nektarin kovana getirilmesinden sorumludurlar. İşçi arılar aynı zamanda ziyaret ettikleri kaynağın kalitesi ve yeri ile ilgili bilgiyi kovanda bekleyen diğer arılarla da paylaşırlar. Bundan sonar görevi belirli işçi arılar için görevli arılar ifadesi kullanılacaktır.

### **3.2.3 Görevsiz İşçi Arılar**

Bu arılar nektarını toplayabilecekleri kaynak arayışı içerisinde edirler. Görevi belirsiz iki çeşit arı bulunmaktadır: İçsel bir dürtüyle veya bir dış etmene bağlı olarak rastgele kaynak arayışında olan kaşif arılar ve kovanda bekleyen ve görevli arıları izleyerek bu arılar tarafından paylaşılan bilgiyi kullanarak yeni bir kaynağa yönelen gözcü arılar. Kaşif arıların sayısının tüm koloniye oranı %5 - %10 arasındadır.

Arılar arasında bilgi paylaşımı kolektif bilginin oluşumunda en önemli husustur. Kovan göz önüne alındığında, tüm kovanlarda ortak olan bazı parçalara ayırmak mümkündür ve bu parçalardan en önemlisi ise dans alanıdır. Yiyecek kaynağının kalitesi ve yeri ile ilgili bilgi paylaşımı dans alanında olmaktadır. Bir arı dans ederken diğer arılarda ona antenleri ile dokunarak bulduğu kaynağın tadı ve kokusu ile ilgili bilgi alırlar. Ziyaret ettikleri kaynağa daha fazla arı yönlendirebilmek için kovanındaki çeşitli alanlarda bu dansı gerçekleştirir ve kaynağına geri dönerler.



**Şekil 3.1** ABC algoritmasının akış diyagramı

Kesin çizgileri olmamakla beraber genelde kovandaki uçuş yapılan yere yakın bir yerde olan dans alnınınında yapılan danslar, taşınan bilgiye göre değişmektedir. En önemlisi nektarin tatlılığıdır. Bu üstünlük dans eşik değerini belirler. Ancak bu uyarıcı yeterli değildir. Bunun yanında nektarin çıkarılmasının kolaylığı, kovandan olan uzaklık, çiçek nektarının kıvamı, besinin genel durumu, kalitedeki göreceli değişimler, hava koşulları ve günün hangi vaktinin olduğu dansı etkileyen diğer etmenlerdir.

Yiyecek getircilerin, diğer yiyecek getircileri uzak noktadaki bir kaynağa yönlendirmek için hedefe ait yön bilgisi verilmesi gereklidir. Yön bilgisi alındıktan sonra hedefe ulaşmada güneşten faydalananlardır. Bileşik gözleri ile arılar, kendi yörüngeleri ile güneş arasındaki açıyı hesaplayabilmektedirler. Güneşin önü kapanmış olsa dahi polarize gün ışığından yine güneşin konumunu tayin edebilmektedirler.

Kaynağın kovana olan mesafesine göre çeşitli danslar mevcuttur: Dairesel dans, kuyruk dansı ve titreme dansı gibi. Daire ve kuyruk dansları yiyecek getircilerin yeniden aktivasyonunda etkilidir. Bu iki dans farklı uzaklıktaki bölgelerin ayrılmışlarında kullanılır. Daire dansı ile belirtilen yiyecek kaynağının kovana olan uzaklığını maksimum 50-100 metre civarında olduğundan bu dans yön ve uzaklık bilgisi vermemektedir.

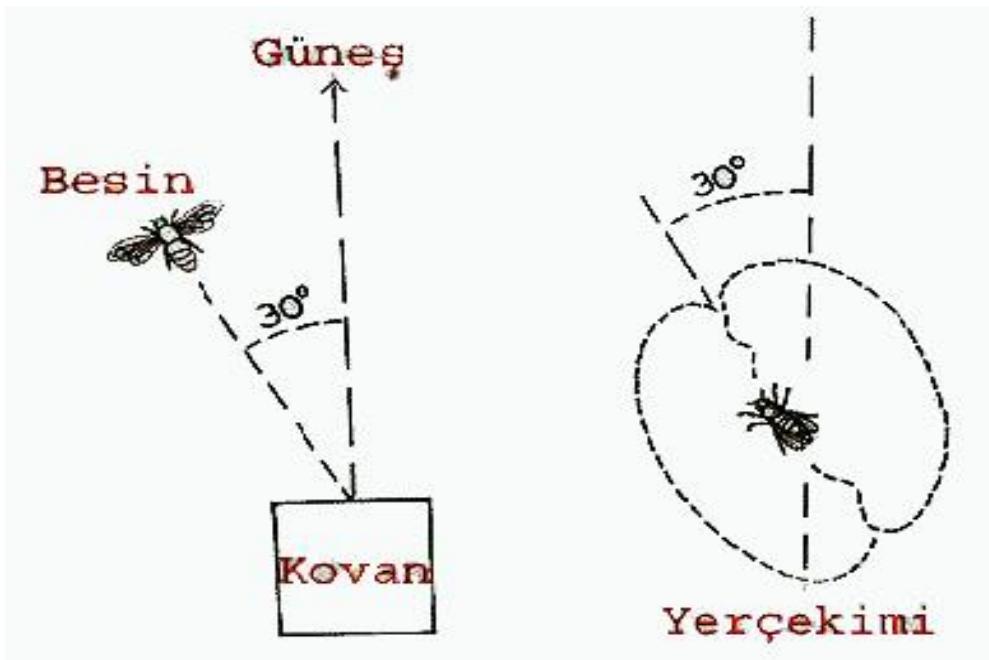
Titreme dansında, arıların petek üzerinde düzensiz tarzda ve yavaş tempoda bacaklarını titreterek ileri, geri, saga ve sola hareketleri söz konusudur. Arı, zengin bir nektar kaynağı bulduğunu, ancak kovana işlenebileceğinden fazla nektar geldiğini ve bundan dolayı nektarı işleme görevine geçmek istediği belirtmektedir. Sadece dans alanında değil kovanın başka bölümlerinde de bu dans gerçekleştirilmektedir. Bu dansın amacı kovan kapasitesi ve yiyecek getirme aktivitesi arasındaki dengeyi sağlamaktadır.

100 metreden 10 kilometreye kadar olan geniş bir alan içerisinde bulunan kaynaklarla ilgili bilgi aktarımında kuyruk dansı kullanılmaktadır. Bu dans “8” rakamına benzeyen figürlerin yapıldığı bir dans çeşididir. Dansı izleyen arıların bir titreşim oluşturması ile bu dansı yapan arı, dansına son verir. Her 15 saniyede dansın tekrarlanması sayısı, nektarin kaynağının uzaklığını hakkında bilgi vermektedir. Daha az tekrarlanma sayısı daha uzak bölgeleri ifade etmektedir. Yön bilgisi, Şekil 3.2’deki gibi 8 rakamı şeklindeki dansın açı bilgisinden elde edilir (Karaboğa, 2004).

Tüm zengin kaynaklarla ilgili bilgiler dans alanında gözcü arılarla iletiliğinde, gözcü arılar bir kaç dansı izledikten sonra hangisini tercih edeceğini karar verir. Zengin kaynaklarla ilgili daha fazla bilgi aktarımı olduğundan bu kaynakların seçilme olasılığı daha fazladır. Yiyecek arayıcıların davranışlarından daha iyi anlaşılabilmesi için Şekil 3.3’de verilen modelin incelenmesi faydalı olacaktır (Karaboğa, 2004).

A ve B ile gösterilen iki keşfedilmiş kaynak bölgesi olduğunu varsayıyalım. Araştırmmanın başlangıcında potansiyel bir yiyecek arayıcı, görevi belirsiz bir işçi arı olarak araştırmaya başlayacak ve bu arı kovan etrafındaki kaynakların yerinden haberdar değildir. Bu durumda bir arı için iki olası seçenek söz konusudur:

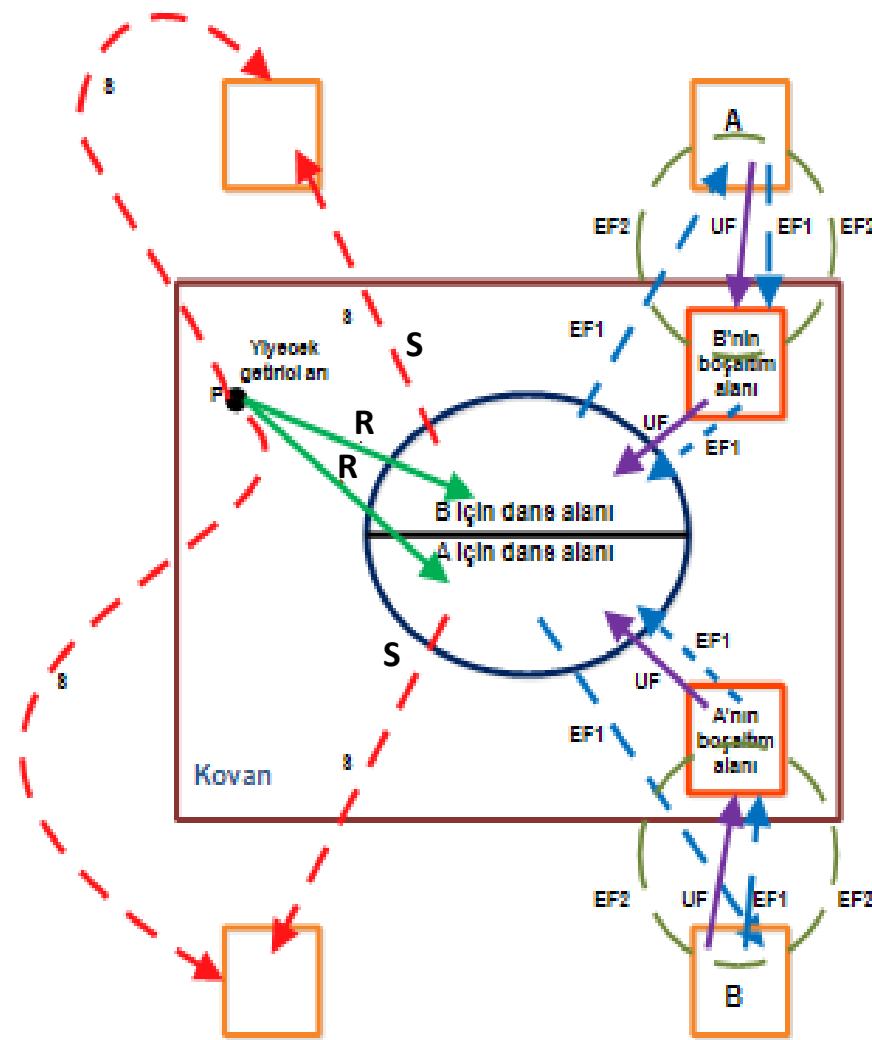
- Bu arı kaşif arı olabilir ve içsel ve dışsal etkenlere bağlı olarak yiyecek aramaya başlayabilir. Kaşif arılar Şekil 3.3’de S ile gösterilmektedir.



**Şekil 3.2** Arıların dansı

- Bu arı, kuyruk danslarını izleyen bir gözücü arı olabilir ve izlediği dansla anlatılan kaynağa gidebilir (Şekil 3.3'de R ile gösterilmektedir). Bir kaynak keşfedildikten sonar arı imkanları dahilinde bu kaynağın yerini hafızasına alır ve hemen nektar toplamaya başlar. Böylece bu arı artık nektarın zenginlik durumuna göre görevli arı haline gelmektedir. İşçi arı kaynaktan nektarı aldıktan sonar kovana döner ve bunu yiyecek depolayıcılara aktarır. Nektarın aktardıktan sonar arı için üç seçenek ortaya çıkmaktadır:
  - ❖ Gittiği kaynağı bırakarak bağımsız izleyici olabilir (Şekil 3.3'de UF ile gösterilmektedir).
  - ❖ Gittiği kaynağa dönmeden önce dans eder ve kovandaki diğer arılara da aynı kaynağı yönlendirebilir (Şekil 3.3'de EF1 ile gösterilmektedir).
  - ❖ Diğer arıları yönlendirmeden kaynağına gidebilir (Şekil 3.3'de EF2 ile gösterilmektedir).

Tüm arıların eş zamanlı olarak yiyecek arama sürecinde olmadıklarını belirtmek gerekir. Yeni arıların yiyecek aramaya katılma olasılıklarının toplam arı sayısı ile o anda yiyecek arama sürecinde olan arıların sayılarının farklıyla orantılı olduğu çalışmalarla doğrulanmıştır.



Şekil 3.3 Yiyecek arama çevrimi

Arıların davranışlarından ortaya çıkan kendi kendine organize olabilme özelliği aşağıdaki verilen temel özelliklerden kaynaklanmaktadır.

- **Pozitif geribesleme:** Kaynağın nektar miktarı arttıkça bu kaynağı seçen gözcü arı sayısı da artmaktadır.
- **Negatif geribesleme:** Tükenen kaynak terk edilmektedir.
- **Çoklu etkileşim:** Arıları yiyecek kaynakları ile ilgili olarak dans alanında bilgi paylaşımı gerçekleştirmektedirler.

### **3.3 Uygulama Alanları**

Arılar ile sürü optimizasyonu klasik mühendislik problemleri dışında günlük hayatı bir çok sorunu çözmek için kullanılmaktadır. Bunlardan bazıları aşağıda yıllara göre sıralanmıştır.

- İnsansız Hava Araçları İçin Rota Planlama (2017)
- Uydu Görüntüleri İle Tarım Alanlarının Sınıflandırılması (2015)
- Uzay Çelik Yapıların Optimum Tasarımı (2014)
- Otomatik Ders Çizelgeleme (2011)
- Uydu Görüntülerinin Bölütlenmesi (2010)

### **3.4 Seçilen Örnek Problem**

Türkiye tarım alanları bakımından zengin bir ülke konumundadır. Bu nedenle bu alanların kullanımı hakkında bilgi edinilmesi önem arz etmektedir. Uzaktan algılama ile görüntü sınıflandırma işlemleri uydu görüntülerinden bilgi çıkarımı konusunda büyük bir paya sahiptir. Tarım alanlarının uydu görüntülerini üzerinden sınıflandırılması, bu alanlar hakkında bilgi edinme açısından çeşitli kolaylıklar sunmaktadır. Uydu görüntülerini üzerinden tarimsal nitelikteki alanları belirlemek ve hangi alanda hangi ürünün yetiştiği hakkında bilgi çıkarımı yapmak mümkündür. Bu yüzden projenin gelecek dönemde üzerinde çalışacağım örnek problem S.Reis ve A.T.Torun tarafından yapılan bir çalışma esas alınmıştır (Reis ve Torun, 2015). Uzaktan algılamada görüntü sınıflandırma işlemlerinde birçok yöntem kullanılmaktadır. Bu yöntemler başlıca, en yakın mesafe, maksimum benzerlik, paralelkenar yöntemi vb. olarak sıralanabilir. Yapay zeka optimizasyon algoritmaları klasik yöntemlere alternatif olarak ortaya çıkmıştır. Çalışmamızda, çok yüksek çözünürlüklü uydu görüntülerini kullanılarak yapay arı koloni algoritması (YAKA) ile uydu görüntülerini sınıflandırılmıştır. Araziden ve hava fotoğraflarından yararlanılarak gerekli olan yer kontrol noktaları ve eğitim verileri toplanmıştır. Çalışma alanı olarak, bitki örtüsü, arazi topografyası ve tarım alanları göz önüne alınarak, Rize ili pilot bölge olarak seçilmiştir. Uygulama sonucunda Yapay Arı Koloni Algoritması ile elde edilen doğruluk değerleri klasik sınıflandırma yöntemleri ile karşılaştırılarak değerlendirilmiş ve YAKA ile elde edilen doğruluk oranlarının daha iyi olduğu sonucuna ulaşılmıştır.

### **3.5 Tarım Alanlarının Sınıflandırılması**

Türkiye tarım alanları konusunda dünyanın onde gelen ülkeleri arasında yer almaktadır. Ülkemizde çeşitli tarım ürünleri yetiştirmekte ve dünyaya sunulmaktadır. Bu ürünlerin onde gelenlerinden biri olarak çay, Türkiye tarım niteliği bakımından önem arz etmektedir. Çay ve benzer nitelikteki tarım alanları ülke kalkınma, planlama ve çiftçi destekleme çalışmaları başta olmak üzere bir çok uygulama için hızlı ve doğru bir şekilde belirlenmesi gerekmektedir. Bu çerçevede uzaktan algılama teknolojisi kullanılarak yüksek doğrulukta sonuçlara ulaşılabilir.

Uzaktan algılama, herhangi bir fiziksel temas olmaksızın sensörler yarımiyla oluşturulan uydu görüntülerini üzerinden yeryüzündeki nesnelerden bilgi çıkarımı işlemidir. Elde edilen uydu görüntülerini birçok meslek disiplininde aktif olarak kullanılmaktadır. Bu görüntüler çeşitli görüntü işleme ve zenginleştirme yöntemlerine tabii tutularak kullanılabilir hale gelmektedir. Görüntü işleme yöntemlerinden elde edilen veriler, meslek disiplinine ve kullanılacak olan hassasiyete göre değişiklikler gösterebilmektedir. Uzaktan algılamada en önemli görüntü işleme yöntemlerinden birisi hiç şüphesiz sınıflandırma işlemidir. Sınıflandırma işlemi genel olarak yeryüzündeki benzer spektral yansıtma değerine sahip objelerin aynı grup altında toplanması işlemi olarak tanımlanabilir. Diğer bir ifadeyle, görüntüyü oluşturan her bir pikselin

tüm bantlardaki değerlerinin diğer pikseller ile karşılaştırılarak benzer piksellerin kullanıcının belirlediği sınıflara ayrılması işlemidir.

Günümüzde kullanılmakta olan birçok görüntü sınıflandırma yöntemi bulunmaktadır. Fakat teknolojinin hızla ilerlemesi, hassasiyet ve doğruluk gereksinimlerinin artması gibi nedenler bilim adamlarını yeni arayışlara sürüklemeştir. Yapay zeka optimizasyon algoritmaları bu konuda kendine büyük bir yer edinmiştir. Son yirmi yılda Parçacık Sürü Optimizasyonu, Diferansiyel Gelişim ve Karınca Kolonisi Optimizasyonu gibi Yapay zeka optimizasyon algoritmaları (sezgisel algoritmalar) birçok meslek disiplinde, değişik optimizasyon problemleri için kendine geniş bir yer bulmuştur. Bu algoritmalar halihazırda kullanılan klasik sınıflandırma yöntemlerine (en çok benzerlik, en kısa mesafe, paralelkenar yöntemi vb.) bir alternatif olarak kullanılabilirlerdir.

Yapay arı koloni algoritması (YAKA) arıların yiyecek arama davranışlarından ilham alınarak geliştirilen bir yapay zeka optimizasyon algoritmasıdır (Karaboğa, 2011). Arılar yiyecek arama davranışlarını sergilemek adına çeşitli salınım ve hareketlerde bulunurlar. Bu hareketler belirli görevlere sahip arılar tarafından izlenerek besin kaynağı hakkında bilgi edinimi sağlar. Yapay zeka optimizasyon algoritmalarının kullanımı uzaktan algılama alanında da etkili olmuştur. Yapılan bazı çalışmalarda diğer sınıflandırma teknikleri ile karşılaştırılan YAKA'nın daha yüksek doğruluk verdiği görülmüştür. Bu çalışmada yapay arı koloni algoritması kullanılarak Rize ilinde özellikle çay tarım alanlarının en doğru şekilde sınıflandırılması incelenmiştir. Bu işlemler yapılrken Rize iline ait 4 bantlı multispektral QuickBird-2 uydu görüntüsü kullanılmıştır. Kullanılan görüntüler MATLAB R2017a ile işlenmiş ve sınıflandırma sonuçlarına varılmıştır.

### **3.5.1 ABC Algoritması ile Tarım Alanlarının Sınıflandırılması Probleminin Çözümü**

ABC algoritması ile Tarım Alanlarının Sınıflandırılması; araziye gitmeden arazinin uydu görüntüsünün, bilgisayar ortamında minimum sürede işlenerek maksimum karı elde etmek için arazinin hangi alanına hangi mahsulün ekilebileceğini bulmayı hedefler.

Bu çözümde daha önceden hafızada tutulan hangi toprak renginde hangi mahsul yetişir bilgisinden yola çıkılarak bu renklere karşılık gelecek RGB renk kodları ile temsil edilmiştir. Verilen bütün arazi bilgileri bir matris temsil edilmiştir. Burada;

- Her piksel  $1 \text{ km}^2$  toprak parçası ile,
- Pixellerin içerisindeki değerler toprağın renk tonu ile,
- Yiyecek kaynakları sınıflandırmada kullanılan sınıflar olarak (1. kategori, 2. kategori, 3. kategori),
- Komşuluk çözümleri kullanılan uzaklık verisi olarak,
- İşçi arılar sınıflandırmada kullanılan kategori değerleri olarak,
- Fonksiyon değerleri (nektar kalitesi) pixelin içerisindeki değer ile kategorilere atanmış değer arasındaki fark olarak temsil edilmiştir.

Algoritma Çalışlığında;

Girdi: Uydu Görüntüsü  
Çıktı: Sınıflandırılmış Matris

**Adım 1:** Haritanın oluşturulması.

**REPEAT**

**Adım 2:** YAKA temel kodlarının ve verilerinin düzenlenmesi.

**Adım 3:** Renk değerlerine göre piksellerin belirlenen sınıflara atanması.

**Adım 4:** Bir sınıfa ait bütün piksellerin atanması.

**UNTIL** (verilen iterasyon sayısı)

### 3.5.2 Örnek Çalışma

Bu örnek çalışma, MATLAB R2017a sürümü ve bu sürüm içerisinde bulunan Statistics Toolbox'ı kullanılarak yapılmıştır. (Problemin detaylı MATLAB kodu EK A.1'de gösterilmiştir.)

Burada 30x30 kilometrelük bir arazi, 30x30luk bir çözünürlük ile temsil edilmiştir. Çalışmanın amacı ise bu 900 kilometrekarelük alandaki her 1 kilometre karelük toprak alanlarını ayrı ayrı inceleyip o bölgeye ekilebilecek ve hasat sonunda en yüksek karı elde edebilecek mahsülü bulmaktadır. İterasyon sayısı 400 ve arı sayısı 100 olarak verilmiştir. Uydu görüntüsü yerine 0 ile 256 (RGB renk kodları sayısı) sayıları arasında rastgele 900 değer oluşturulur ve 30x30'luk bir matrisin içerişine bu değerler yazılır. Oluşturulan bu harita Şekil 3.4'de gösterilmiştir. Her değer içerişine yazılan 1 kilometrekarelük alanın rengini temsil eder. Önceden tanımlı 4 çeşit mahsül bulunmaktadır. Bunlar 1. tür mahsul (1.00 ile gösterilir), 2. tür mahsul (2.00 ile gösterilir), 3. tür mahsul (3.00 ile gösterilir) ve iterasyon sayısı yetmediğinden kontrol edilemeyen alanlar veya bilinmeyen türden alan için ise kategorize edilemeyen (0.00 ile gösterilir) ile temsil edilir. Algoritma çalışlığında 400 iterasyonun sonunda Şekil 3.5'de gösterilen sonuç matrisi oluşturulur ve ekrana basılır. Burada her 1 kilometrekarelük alana hangi kategoriden mahsül ekileceği rakamlar ile gösterilmiştir.

 Figure 2

PDF

— □ ×

April 10, 2018, 2:54 PM

## Matrix: map

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30					
1	160.00	242.00	147.00	146.00	39.00	102.00	217.00	143.00	251.00	166.00	49.00	255.00	181.00	124.00	151.00	97.00	12.00	177.00	65.00	211.00	224.00	155.00	244.00	239.00	165.00	218.00	220.00	196.00	231.00	11.00					
2	134.00	215.00	139.00	213.00	51.00	25.00	140.00	95.00	235.00	164.00	252.00	69.00	84.00	68.00	13.00	160.00	65.00	56.00	20.00	61.00	186.00	13.00	222.00	232.00	106.00	94.00	172.00	11.00	140.00	189.00					
3	210.00	98.00	34.00	70.00	125.00	224.00	67.00	177.00	142.00	254.00	96.00	182.00	131.00	135.00	95.00	236.00	6.00	15.00	118.00	42.00	123.00	211.00	175.00	122.00	182.00	249.00	98.00	99.00	95.00	222.00					
4	221.00	42.00	36.00	132.00	32.00	255.00	249.00	218.00	132.00	193.00	192.00	49.00	113.00	112.00	117.00	29.00	247.00	15.00	35.00	177.00	184.00	149.00	169.00	21.00	178.00	202.00	106.00	222.00	13.00	168.00					
5	211.00	137.00	128.00	40.00	122.00	250.00	200.00	76.00	224.00	235.00	164.00	153.00	186.00	121.00	6.00	228.00	120.00	290.00	62.00	73.00	117.00	245.00	55.00	157.00	84.00	153.00	66.00	222.00	180.00	123.00					
6	249.00	129.00	43.00	79.00	193.00	145.00	82.00	113.00	126.00	62.00	86.00	130.00	229.00	206.00	226.00	87.00	19.00	136.00	94.00	19.00	179.00	256.00	174.00	23.00	120.00	102.00	19.00	43.00	154.00	236.00					
7	93.00	0.00	103.00	194.00	98.00	242.00	150.00	91.00	102.00	77.00	95.00	133.00	54.00	115.00	148.00	146.00	137.00	27.00	195.00	114.00	30.00	235.00	69.00	104.00	53.00	123.00	102.00	21.00	222.00	50.00					
8	89.00	215.00	59.00	16.00	4.00	159.00	112.00	5.00	163.00	157.00	132.00	15.00	55.00	144.00	191.00	216.00	210.00	228.00	141.00	249.00	245.00	215.00	170.00	121.00	13.00	242.00	232.00	53.00	130.00	111.00					
9	108.00	253.00	62.00	64.00	39.00	11.00	33.00	70.00	154.00	248.00	156.00	183.00	242.00	256.00	238.00	126.00	157.00	164.00	190.00	119.00	244.00	172.00	147.00	195.00	224.00	110.00	248.00	30.00	209.00	233.00					
10	159.00	29.00	172.00	145.00	217.00	174.00	83.00	57.00	54.00	123.00	8.00	8.00	35.00	155.00	230.00	227.00	198.00	28.00	247.00	182.00	239.00	30.00	31.00	93.00	256.00	153.00	178.00	91.00	27.00	190.00					
11	202.00	214.00	60.00	231.00	94.00	218.00	239.00	64.00	193.00	105.00	251.00	154.00	119.00	191.00	164.00	85.00	139.00	243.00	151.00	201.00	194.00	118.00	87.00	42.00	180.00	51.00	84.00	86.00	241.00	60.00					
12	0.00	125.00	216.00	55.00	156.00	223.00	129.00	84.00	140.00	16.00	10.00	103.00	89.00	142.00	138.00	160.00	135.00	33.00	190.00	34.00	151.00	216.00	66.00	198.00	145.00	80.00	106.00	105.00	24.00	36.00	130.00				
13	54.00	193.00	105.00	174.00	129.00	84.00	104.00	16.00	241.00	251.00	156.00	126.00	103.00	183.00	6.00	155.00	48.00	109.00	111.00	42.00	112.00	183.00	159.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			
14	11.00	231.00	146.00	129.00	251.00	100.00	84.00	221.00	47.00	243.00	48.00	65.00	152.00	190.00	194.00	59.00	113.00	30.00	138.00	234.00	36.00	250.00	136.00	71.00	73.00	198.00	36.00	14.00	151.00	243.00					
15	113.00	81.00	58.00	208.00	5.00	151.00	37.00	209.00	160.00	89.00	55.00	254.00	172.00	144.00	21.00	84.00	199.00	215.00	254.00	65.00	119.00	220.00	247.00	107.00	52.00	236.00	135.00	233.00	78.00	89.00					
16	11.00	23.00	64.00	48.00	233.00	72.00	39.00	247.00	4.00	163.00	18.00	5.00	87.00	23.00	234.00	73.00	78.00	63.00	201.00	148.00	41.00	139.00	59.00	234.00	173.00	83.00	101.00	194.00	0.00	0.00					
17	155.00	20.00	153.00	31.00	247.00	39.00	245.00	129.00	90.00	218.00	142.00	20.00	176.00	238.00	177.00	53.00	231.00	29.00	39.00	194.00	131.00	197.00	139.00	125.00	7.00	226.00	8.00	112.00	61.00	36.00	52.00	135.00	128.00	161.00	
18	4.00	237.00	223.00	238.00	192.00	157.00	176.00	136.00	183.00	46.00	145.00	110.00	75.00	149.00	110.00	103.00	131.00	197.00	139.00	125.00	7.00	226.00	8.00	112.00	61.00	36.00	52.00	135.00	128.00	161.00					
19	47.00	197.00	112.00	153.00	217.00	78.00	65.00	152.00	161.00	116.00	113.00	189.00	179.00	235.00	73.00	116.00	73.00	125.00	41.00	226.00	103.00	3.00	2.00	16.00	29.00	158.00	94.00	98.00	83.00	20.00	0.00				
20	98.00	9.00	132.00	150.00	23.00	57.00	110.00	243.00	252.00	98.00	8.00	94.00	104.00	51.00	196.00	201.00	124.00	18.00	24.00	19.00	55.00	83.00	44.00	146.00	213.00	40.00	154.00	87.00	138.00	0.00	97.00				
21	102.00	171.00	54.00	171.00	79.00	39.00	4.00	175.00	195.00	105.00	98.00	185.00	106.00	47.00	47.00	72.00	164.00	161.00	76.00	4.00	123.00	160.00	200.00	169.00	207.00	237.00	227.00	81.00	27.00	0.00					
22	62.00	9.00	229.00	62.00	83.00	247.00	55.00	5.00	193.00	83.00	193.00	151.00	236.00	15.00	7.00	218.00	45.00	179.00	224.00	183.00	133.00	99.00	63.00	175.00	108.00	232.00	158.00	167.00	168.00	0.00	0.00	0.00			
23	61.00	199.00	69.00	25.00	67.00	194.00	193.00	46.00	165.00	97.00	101.00	179.00	154.00	249.00	165.00	58.00	240.00	123.00	31.00	27.00	62.00	198.00	89.00	203.00	124.00	69.00	45.00	168.00	39.00	192.00	0.00				
24	109.00	59.00	189.00	198.00	16.00	139.00	207.00	206.00	102.00	156.00	5.00	243.00	166.00	251.00	46.00	74.00	164.00	193.00	6.00	9.00	56.00	96.00	79.00	245.00	100.00	165.00	25.00	12.00	142.00	0.00	0.00	0.00			
25	227.00	87.00	58.00	217.00	90.00	14.00	66.00	200.00	128.00	104.00	11.00	7.00	130.00	147.00	104.00	168.00	169.00	223.00	1.00	122.00	158.00	79.00	171.00	41.00	109.00	84.00	173.00	221.00	243.00	49.00	0.00	0.00	0.00		
26	96.00	105.00	30.00	193.00	37.00	151.00	72.00	149.00	112.00	141.00	84.00	256.00	125.00	213.00	73.00	17.00	198.00	231.00	147.00	228.00	29.00	198.00	27.00	91.00	37.00	131.00	246.00	256.00	129.00	0.00	174.00	0.00	0.00	0.00	
27	172.00	18.00	127.00	126.00	134.00	130.00	42.00	197.00	65.00	219.00	96.00	224.00	124.00	6.00	32.00	131.00	135.00	250.00	98.00	208.00	230.00	133.00	149.00	0.00	225.00	10.00	67.00	62.00	111.00	96.00	0.00	0.00	0.00	0.00	0.00
28	72.00	117.00	97.00	227.00	74.00	157.00	158.00	16.00	39.00	36.00	12.00	213.00	180.00	136.00	31.00	130.00	196.00	49.00	213.00	116.00	205.00	164.00	130.00	255.00	9.00	169.00	106.00	115.00	89.00	143.00	211.00	0.00	211.00	183.00	
29	237.00	14.00	142.00	224.00	211.00	235.00	51.00	207.00	39.00	17.00	241.00	236.00	151.00	14.00	64.00	165.00	225.00	27.00	208.00	7.00	141.00	87.00	204.00	104.00	88.00	221.00	111.00	248.00	175.00	0.00	0.00	0.00	0.00	0.00	0.00
30	23.00	212.00	244.00	101.00	233.00	23.00	55.00	33.00	3.00	238.00	146.00	42.00	199.00	164.00	152.00	66.00	64.00	150.00	122.00	186.00	59.00	18.00	97.00	230.00	147.00	217.00	42.00	211.00	183.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 1



**Matrix: cat**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	2.00	3.00	2.00	2.00	4.00	4.00	1.00	2.00	3.00	4.00	4.00	3.00	1.00	2.00	4.00	4.00	1.00	4.00	1.00	1.00	2.00	3.00	3.00	2.00	1.00	1.00	3.00	4.00		
2	2.00	1.00	2.00	1.00	4.00	4.00	2.00	4.00	3.00	2.00	3.00	4.00	4.00	2.00	4.00	4.00	4.00	4.00	1.00	4.00	1.00	3.00	4.00	2.00	4.00	0.00	4.00	1.00		
3	1.00	4.00	4.00	4.00	2.00	1.00	4.00	1.00	2.00	3.00	4.00	1.00	2.00	4.00	4.00	2.00	4.00	4.00	4.00	2.00	4.00	0.00	1.00	0.00	4.00	0.00	4.00	1.00		
4	1.00	4.00	4.00	2.00	4.00	3.00	1.00	1.00	4.00	2.00	4.00	2.00	4.00	3.00	4.00	4.00	4.00	4.00	1.00	2.00	4.00	0.00	1.00	4.00	0.00	4.00	2.00			
5	1.00	2.00	4.00	4.00	2.00	3.00	1.00	4.00	1.00	3.00	2.00	2.00	1.00	2.00	4.00	3.00	2.00	4.00	4.00	2.00	4.00	3.00	4.00	2.00	4.00	0.00	1.00	2.00		
6	3.00	2.00	4.00	4.00	1.00	2.00	1.00	4.00	1.00	2.00	3.00	4.00	1.00	3.00	4.00	4.00	2.00	4.00	4.00	1.00	3.00	2.00	4.00	2.00	4.00	0.00	2.00	3.00		
7	4.00	4.00	4.00	1.00	4.00	3.00	2.00	3.00	1.00	4.00	4.00	2.00	4.00	2.00	2.00	4.00	1.00	2.00	4.00	3.00	4.00	4.00	4.00	2.00	4.00	0.00	4.00	4.00		
8	4.00	1.00	4.00	4.00	4.00	2.00	4.00	4.00	4.00	2.00	2.00	4.00	4.00	2.00	1.00	1.00	3.00	2.00	3.00	3.00	2.00	2.00	3.00	4.00	4.00	2.00	4.00	4.00		
9	4.00	3.00	4.00	4.00	4.00	4.00	2.00	3.00	2.00	4.00	3.00	3.00	2.00	2.00	1.00	2.00	3.00	2.00	3.00	1.00	2.00	3.00	1.00	2.00	3.00	1.00	3.00	3.00		
10	2.00	4.00	2.00	1.00	2.00	4.00	4.00	4.00	2.00	4.00	2.00	2.00	4.00	4.00	2.00	2.00	4.00	3.00	1.00	4.00	3.00	1.00	4.00	0.00	4.00	2.00	4.00	4.00		
11	1.00	1.00	4.00	3.00	4.00	1.00	3.00	4.00	1.00	4.00	3.00	2.00	2.00	1.00	2.00	2.00	2.00	1.00	2.00	2.00	2.00	2.00	4.00	4.00	2.00	3.00	4.00			
12	4.00	2.00	1.00	4.00	2.00	1.00	3.00	3.00	1.00	2.00	4.00	2.00	2.00	4.00	1.00	4.00	2.00	2.00	4.00	2.00	2.00	2.00	2.00	4.00	0.00	2.00	2.00			
13	4.00	1.00	4.00	2.00	4.00	2.00	4.00	4.00	4.00	1.00	4.00	3.00	3.00	2.00	2.00	4.00	1.00	2.00	4.00	2.00	2.00	4.00	0.00	4.00	0.00	2.00	2.00			
14	4.00	3.00	2.00	3.00	4.00	4.00	1.00	4.00	3.00	4.00	2.00	2.00	1.00	4.00	2.00	2.00	4.00	2.00	2.00	4.00	0.00	4.00	2.00	4.00	3.00	4.00	2.00			
15	2.00	4.00	4.00	1.00	4.00	2.00	4.00	1.00	2.00	4.00	4.00	2.00	1.00	4.00	1.00	2.00	4.00	2.00	2.00	3.00	4.00	2.00	2.00	3.00	4.00	0.00	4.00	4.00		
16	4.00	4.00	4.00	3.00	4.00	4.00	4.00	4.00	4.00	1.00	4.00	4.00	2.00	4.00	4.00	1.00	4.00	4.00	1.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
17	2.00	4.00	4.00	3.00	4.00	3.00	2.00	4.00	1.00	2.00	4.00	1.00	3.00	1.00	4.00	3.00	4.00	4.00	2.00	2.00	4.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00		
18	4.00	3.00	1.00	3.00	1.00	2.00	1.00	4.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	2.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
19	4.00	4.00	4.00	2.00	1.00	4.00	4.00	2.00	2.00	1.00	1.00	3.00	4.00	2.00	4.00	2.00	4.00	4.00	3.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
20	4.00	4.00	2.00	4.00	4.00	4.00	3.00	3.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
21	4.00	2.00	4.00	4.00	4.00	4.00	1.00	4.00	4.00	4.00	1.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	2.00	2.00	4.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00		
22	4.00	4.00	3.00	4.00	4.00	4.00	1.00	4.00	1.00	2.00	3.00	4.00	2.00	1.00	4.00	0.00	1.00	4.00	0.00	2.00	4.00	0.00	4.00	0.00	4.00	3.00	4.00	2.00		
23	4.00	1.00	4.00	4.00	1.00	4.00	4.00	4.00	2.00	1.00	2.00	4.00	0.00	4.00	0.00	4.00	1.00	4.00	1.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	1.00			
24	4.00	4.00	1.00	1.00	4.00	2.00	1.00	4.00	2.00	4.00	3.00	2.00	4.00	0.00	2.00	1.00	4.00	0.00	4.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	2.00			
25	3.00	4.00	4.00	1.00	4.00	4.00	1.00	2.00	4.00	4.00	2.00	2.00	4.00	4.00	4.00	4.00	2.00	2.00	4.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00		
26	4.00	4.00	1.00	4.00	2.00	4.00	2.00	4.00	3.00	2.00	4.00	1.00	4.00	0.00	3.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	2.00		
27	2.00	0.00	2.00	2.00	2.00	4.00	1.00	4.00	1.00	2.00	3.00	4.00	1.00	3.00	4.00	1.00	3.00	3.00	2.00	2.00	4.00	0.00	4.00	0.00	4.00	0.00	4.00	4.00		
28	4.00	2.00	4.00	3.00	4.00	1.00	4.00	4.00	4.00	1.00	2.00	4.00	1.00	4.00	1.00	4.00	0.00	1.00	4.00	0.00	2.00	2.00	0.00	4.00	0.00	4.00	0.00	4.00		
29	3.00	0.00	2.00	1.00	3.00	4.00	1.00	4.00	0.00	3.00	2.00	4.00	0.00	4.00	2.00	4.00	1.00	2.00	4.00	1.00	4.00	3.00	4.00	2.00	4.00	0.00	4.00	1.00		
30	4.00	1.00	3.00	4.00	3.00	1.00	4.00	4.00	4.00	3.00	2.00	4.00	4.00	2.00	2.00	4.00	1.00	4.00	4.00	3.00	2.00	1.00	4.00	4.00	1.00	4.00	1.00			

April 10, 2018, 2:54 PM

### 3.6 İş-Zaman Diyagramı

Arılar ile Süriü Algoritması

### 2017-2018 Akademik Yılı Bahar Dönemi Bitirme Projesi

#### İş-Zaman Planı

Görev Adı	Süre	Şubat 2018				Mart 2018				Nisan 2018				Mayıs 2018			
		H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12	H13	H14	H15	H16
1 <b>Seçilen örnek problem için kaynak araştırmasının yapılması</b>	4 hf																
2 <b>Seçilen örnek problem için Matlab kodu araştırması</b>	4 hf																
3 <b>Matlab uygulamasının kodlanması</b>	5 hf																
4 <b>Ara dönem sunumu hazırlanması ve sunumu</b>	4 hf																
5 <b>Matlab yardımıyla ABC'yi kullanarak basit bir problem çözümü</b>	5 hf																
6 <b>Sonuç karşılaştırması</b>	3 hf																
7 <b>Poster, final raporu ve sununu hazırlanması ve final sunumu</b>	2 hf																

Hazırlayan Ad-Soyad : Eray Altay

12.03.2018

### **3.7 Sonuç**

Doğada arıların besin arama davranışları insanlara ilham kaynağı olmuş ve bunun neticesinde ise Yapay Arı Kolonisi Algoritması (YAKA) geliştirilmiştir. YAKA'da arıların bütün davranışları bire bir modellenmemiştir ve bu model klasik mühendislik problemleri dışında günlük hayatta bir çok problemin çözümüne yardımcı olmuştur.

### **3.8 Kaynakça**

- Hahn, B. ve Daniel V. "Temel MATLAB (5. Baskı)" Nobel Akademik Yayıncılık (2016) ss. 101-140.
- Reis, S. ve A.T. Torun "Yapay Arı Koloni Algoritmasının Tarım Alanlarının Sınıflandırılmasında Kullanılabilirliğinin İrdelenmesi" Aksaray Üniversitesi Harita Mühendisliği Bölümü (2015) ss. 1-5.
- Nagwani, N. K. ve S. Verma "A Comparative Study of Bug Classification Algorithms, Volume 24, Issue 1" World Scientific Publishing Company (2014) ss. 111-138.
- Civiohlu, P. ve E. Besdok "A Conceptual Comparison of The Cuckoo-Search, Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony Algorithms, Volume 39, Issue 4" Artificial Intelligence Review (2013) ss. 315–346.
- Yuan, Z., M. A. M.Oca, M. Birattari ve T. Stützle "Continuous Optimization Algorithms for Tuning Real and Integer Parameters of Swarm Intelligence Algorithms, Volume 6, Issue 1" Springer (2012) ss. 49–75.
- Lien, L. ve M. Y. Cheng "A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization, Volume 39, Issue 10" Elsevier (2012) ss. 9642-9650.
- Derrac, J., S. Garcia, D. Molina ve F. Herrera "Swarm and Evolutionar Computation, Volume 1, Issue 1" Elsevier (2011) ss. 3-18.
- Omkar, S., J. Senthilnath, R. Khandelwal, G. N. Naik ve S. Gopalakrishnan "Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures, Volume 11, Issue 1" Elsevier (2011) ss. 489-499.
- Karaboga, D. ve C. Ozturk "A Novel Clustering Approach: Artificial Bee Colony (ABC) Algorithm, Volume 11, Issue 1" Elsevier (2011) p.p 652-657.
- Zhu, G. ve S. Kwong "Gbest-Guided Artificial Bee Colony Algorithm for Numerical Function Optimization, Volume 217, Issue 7" Elsevier (2010) ss. 3166-3173.
- Alatas, B. "Chaotic Bee Colony Algorithms for Global Numerical Optimization, Volume 37, Issue 8" Elsevier (2010) ss. 5682-5687.
- Tübitak, Rize İline (TR904) Ait Heyelan Risk Bölgeleri ve Uygun Yerleşim Alanlarının Coğrafi Bilgi Teknolojileri İle Belirlenmesi (Proje Yürüttüsü: Yrd. Doç. Selçuk Reis), 106Y018 nolu 1001 Bilimsel ve Teknolojik Araştırma Projesi, Ankara (2009), 151 s.
- Akay, B., Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi, Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi (2009).
- Karaboga, D. (2004). Yapay Zeka Optimizasyon Algoritmaları (4. Baskı). İstanbul: NOBEL Akademik Yayıncılık.

## **IV KARINCALAR İLE SÜRÜ OPTİMİZASYONU**

Karinca kolonisi optimizasyonu, gerçek karınca kolonilerinin davranışlarından esinlenerek geliştirilmiştir. Tekniğin en temel unsurlarından biri haberleşme aracı olarak kullanılan ve problemlerde çözümün kalitesini gösteren feromon kimyasalıdır. Feromon, gerçek karıncaların da bir haberleşme ve yön bulma aracı olarak kullandıkları, vücutlarından salgıladıkları kimyasaldır. Feromon izleri, karıncalar tarafından güncellenmekte ve bir bilgiyi temsil etmektedir. Bir yolda feromon izinin yoğun olması, yolu kalitesini gösterir ve tercih olasılığını artırır.

Bugüne kadar birçok karıncalarla sürü optimizasyonu için algoritması geliştirilmiştir. Bunlardan ilki TSP'ye uygulanmak üzere Dorigo ve arkadaşları tarafından geliştirilmiş olan Karınca Sistemi (Ant System-AS)'dır. Bu algoritma işlem zamanı ve çözüm kalitesi açısından diğer sezgisellerden daha geride olduğundan algoritmanın performansını artırmak ve daha iyi çözümler elde etmek amacıyla çalışmalar yapılmıştır. Gambardella ve Dorigo Ant-Q sistemini, Bullnheimer ve arkadaşları Rank Temelli Karınca Sistemi (AS<sub>rank</sub>)'ni, Dorigo ve Gambardella Karınca Koloni Sistemi (ACS)'ni, Stützle ve Hoos Max-Min Karınca Sistem (MMAS)'ni ortaya koymuşlardır. Ayrıca birden çok karınca kolonisi ile çalışmayı esas alan algoritmalar da geliştirilmiştir. Bunların dışında birçok çalışmada farklı alanlardaki uygulamalara yönelik, yöntemin performansını artırmak amaçlı değişiklikler ve eklemeler yapılmıştır.

Karıncalarla optimizasyon 5 anahtar kelime temelinde incelenbilir. Bunlar;

1. Yakınlık (Visibility): Problem boyunca sabit kalan ve sezgisel seçimliliği etkileyen sezgisel değerdir.
2. Feromon: Karıncaların gittikleri yol üzerine bıraktıkları kimyasala verilen isimdir. Algoritmalarla koku olarak da adlandırılır.
3. Tabu Listesi: Her sanal karınca gerçeğinden farklı olarak daha önce uğradığı yerlerin listesini tutar ve bu yerlere tur tamamlanıncaya kadar tekrar uğranmaz.
4. Karar verme: Hangi yolu seçileceği olasılık formülüne bağlı olarak belirlenir. En yüksek olasılığa sahip yol tercih edilir.
5. Buharlaşma: Yoldaki feromonun belli bir oranda azaltılmasını ifade eder. İlk keşifler tamamen rasgele olduğundan, bu aşamada bırakılan feromonlar herhangi bir bilgi içermezler. Dolayısıyla diğer karıncaların bu işe yaramayan değerleri kolayca unutması ve iyi çözümler üzerinden yol alabilmesi için buharlaştırma mekanizması gereklidir.

### **4.1 Ant Colony Optimization (ACO) Algoritması (Karinca Kolonisi Optimizasyon Algoritması -KKOA)**

ACO algoritması, gerçek karıncaların yuvaları ile yiyecek noktaları arasındaki en kısa yolu bulma kabiliyetlerinden esinlenerek geliştirilmiştir. Alternatif yolların söz konusu olduğu durumlarda karıncalar, öncelikle bu yollara eşit olasılıkla dağılırlarken belli bir süre sonra en kısa olan yolda yoğunlaşmaktadır. Zaman geçtikçe tüm karıncaların en kısa olan yolu kullandıkları Şekil 4.1'de görülmektedir(Randall,2002).

Marco Dorigo ve Thomas Stützle karıncaların davranışlarını inceleyerek 'Ant Colony Optimization algoritmasını geliştirdiler.

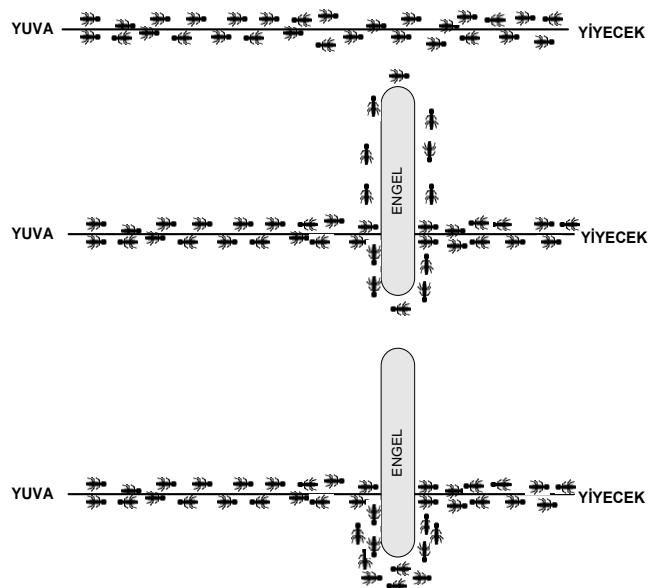
Karıncaların görme duyuları kötüdür ancak kimyasal algılayıcıları gelişmiştir. Harvard Üniversitesi profesörü Dr. Edward O. Wilson 1962 yılında karıncaların haberleşmek için kimyasal vücut salgılarını kullandıklarını tespit etmişti. Bu alanda yapılan çalışmalar karıncaların kimyasal yollar oluşturduklarını ve diğer karıncaların da bu yolları takip ederek hareket ettiklerini ortaya çıkarmıştı.

ACO algoritması bu kimyasal yolların karıncalar tarafından takibini olasılık teorisi dahilinde değerlendiriyor. Bir karınca yuvanın etrafını araştırırken besin bulduğunda yuvaya en kısa mesafeden dönüyor. Bu en kısa mesafeyi feromonlarıyla işaretliyor. Diğer işçi karıncalar feromonlarla işaretlenmiş bu yolu takip ederek besine ulaşıyorlar. Aynı yolu kat eden karıncaların sayısı arttıkça bu sanal yol daha fazla belirginleşiyor. Karıncalar daha belirgin olan bu yolu diğer yollara oranla daha çok tercih ediyorlar.

Karıncaların bu davranışlarını olasılık teorisine göre değerlendiren optimizasyon algoritması da aynı esasları baz alıyor. Bir işleminden alınan olumlu geri bildirimler arttıkça, aynı işlem daha çok tekrarlanıyor.

Aynı zamanda besin arama işlemlerinin sekteye uğramaması için bazı karıncaların keşfe devam etmeleri gerekiyor. Bu nedenle bazı karıncalar daha az belirgin yolları veya daha hiç keşfedilmemiş yolları tercih ediyorlar. Optimizasyon algoritması bu iş bölümünü olasılık hesaplarına dayandırarak modelliyor ancak karıncaların bu iş bölümünü nasıl gerçekleştirdiklerini hala bilmiyoruz.

Bu algoritma karıncaların her gün çözdükleri optimizasyon problemlerinden yalnızca birini açıklıyor. Karıncalar bunun yanında hava sıcaklığı, havanın nemlilik oranı, yuvadaki besin miktarı gibi parametreleri de değerlendiriyor. Normal şartlarda insanların stok takip yazılımlarıyla ve merkezi bir yönetimle gerçekleştiremedikleri işleri, karıncalar ortak bir akıl ile sorunsuzca gerçekleştiriyorlar. Beyinleri olmayan binlerce karınca sanki yuvaya ait bütün gelişmeleri aynı anda bütün detaylarıyla biliyorlar.

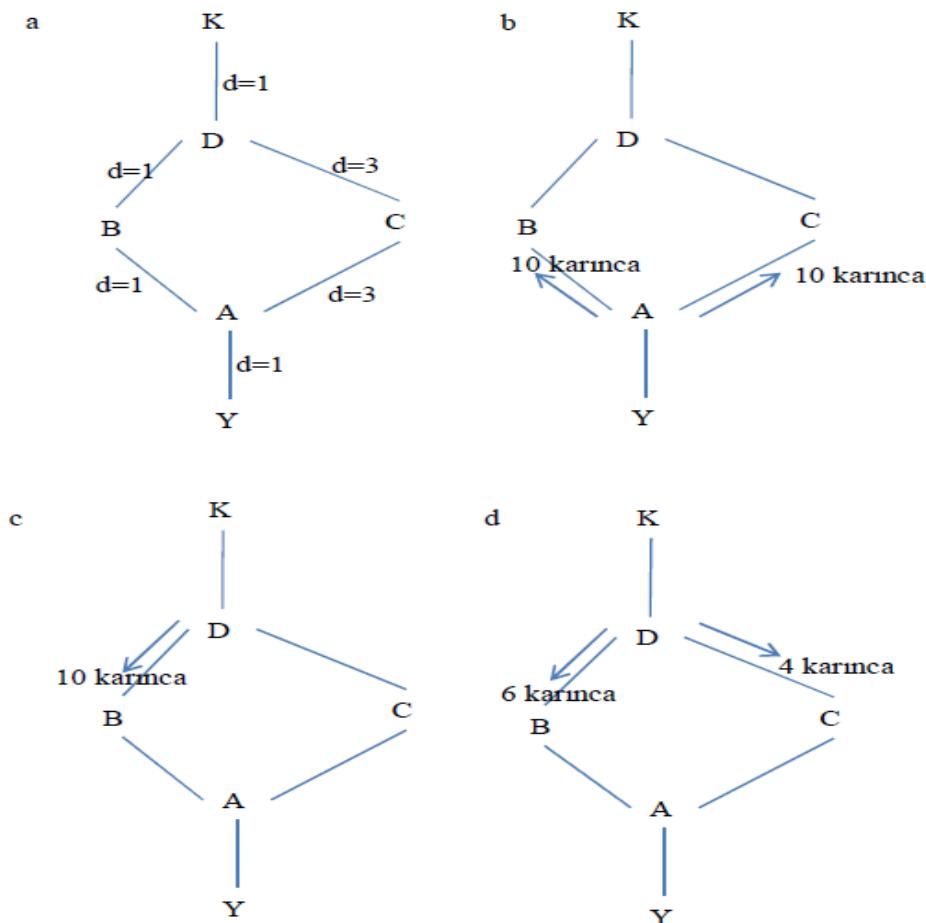


**Şekil 4.1** Gerçek karıncaların en kısa yolu bulma aşamaları

Bir karınca yiyecek bulduğunda diğer karıncaların da yiyeceğe ulaşabilmesi için dönüş yoluna feromon salgısını bırakır. Böylece diğer karıncalar da yoldaki feromon yoğunluğuna göre izleri takip ederek yiyeceğe ulaşabilirler. Bu sistem ile karıncalar diğer karıncaların deneyimlerini kullanarak optimal yolu ve yiyeceği bulurlar.

Yiyecek arayan karıncalar, üzerinde iz olmayan bir yolda engelle karşılaşıklarında, rassal bir şekilde bazıları uzun yolu, bazıları kısa yolu seçer. Kısa yolu seçen karıncalar yiyeceğe daha çabuk ulaşacak ve geldiği yoldan feromon salgılayarak geri döneceklerdir. Böylece kısa yolda daha fazla feromon birikecek ve arkadan gelen karıncalar feromon oranı fazla olan yolu daha fazla tercih edeceklerdir. Feromon uçucu bir madde olduğu için zamanla izler kaybolur. Bundan dolayı besine ulaşmayan yollardan da izler silinir ve böylece karınca yanılmaktan kurtulur.

Kısa olan yolda feromon miktarı uzun yollara nispeten daha fazla birikmektedir. Kısa olan yoldan geçiş daha hızlı gerçekleşeceğini, birim zamanda geçiş yapan karınca sayısı uzun yola göre daha fazla olacaktır. Dolayısıyla herhangi iki düğüm arasındaki yol üzerinde bulunan feromon miktarı, yoluñ uzunluğuyla ters orantılıdır. Bu durum Şekil 4.2'de görülmektedir (Randall,2002).



**Şekil 4.2** Karıncaların yol seçimleri

Şekli kısaca açıklamak gerekirse; yuvadan (Y) yiyecek aramak için çıkan karıncalar (20 tane olsun) A noktasına geldiklerinde iki yolda da feromon olmadığı için ortalama eşit miktarlarda karınca AB ve AC yollarını seçecektir (Şekil 4.2.b). AB yolunu seçen karıncalar yiyecek kaynağına varıp geri D noktasına döndüklerinde DB ve DC yollarından birini seçecektir. Bu seçim feromon maddesine göre yapıldığı için DB yolundaki feromon miktarı 10 birim (her karıncanın geçtiği yola 1 birim feromon bıraktığı varsayıldığında) iken DC yoluna henüz karıncalar tarafından feromon bırakılmamış olacaktır. Bundan dolayı yuvaya dönen 10 karınca DB yolunu seçecektir (Şekil 4.2.c). Kaynağa uzun yoldan giden 10 karınca kaynaktan D noktasına ulaştıklarında DB yolundaki feromon miktarı 20 birim iken DC yolundaki feromon miktarı 10 birim olduğundan 10 karıncadan büyük çoğunluğu DB yolunu tercih edecek bir kısmı ise DC yolunu kullanacaklardır (Şekil 4.2.d). Bu süreç tüm karıncalar kısa olan yolu (Y-A-B-D-K) seçene kadar devam etmektedir.

Yapay karıncalardan oluşan ACO algoritması, yapay feromon izlerinin güncelleştirilmesiyle tekrarlanan bir yapıya sahiptir. Algoritmanın çalışma sürecinde, karıncalar tarafından güncellenen feromon izleriyle iyi bir çözümün bulunması için bilgi oluşturulmakta ve her iterasyonda bu bilgiler güncellenmektedir. ACO algoritmasına ait döngü Şekil 4.3' teki gibidir (Dorigo,1997).

Algoritmanın çalışma sürecinde temel işlemler, yapay karıncaların turları sonunda geçmiş oldukları yolların feromon miktarlarının arttırılması, belirli bir oranda feromon buharlaşmasının gerçekleştirilmesi, en iyi çözümün bulunması, buna bağlı olarak global feromon güncellemesinin yapılması ve karıncaların yenilenen bu feromon miktarlarına bağlı olarak yeni turlarını gerçekleştirmeleridir. Anlatılan bu adımların sözde kodu Şekil 4.4' te verilmiştir. Tüm bu işlemlere ait hesaplama yöntemleri ayrıntılarıyla alt başlıklarda verilmiştir (Dorigo,1997).

**Adım 1 :** Başlangıç feromon değerleri belirlenir.

**Adım 2 :** Karıncalar her düğüme rastsal olarak yerleştirilir.

**Adım 3:** Her karınca, sonraki şehri denklemde verilen lokal arama olasılığına bağlı olarak seçmek suretiyle turunu tamamlar.

**Adım 4:** Her karınca tarafından katedilen yolların uzunluğunu hesaplanır ve lokal feromon güncellemesi yapılır.

**Adım 5:** En iyi çözüm hesaplanır ve global feromon yenilemesinde kullanılır.

**Adım 6:** Maksimum iterasyon sayısı ya da yeterlilik kriteri sağlanana kadar Adım 2' ye gidilir.

**Şekil 4.3** ACO algoritmasının adımları

Başlangıç parametrelerini ayarla ve yollara bir miktar feromon bırak

**While** (Durdurma kriteri ile karşılaşılmadıysa) **do**

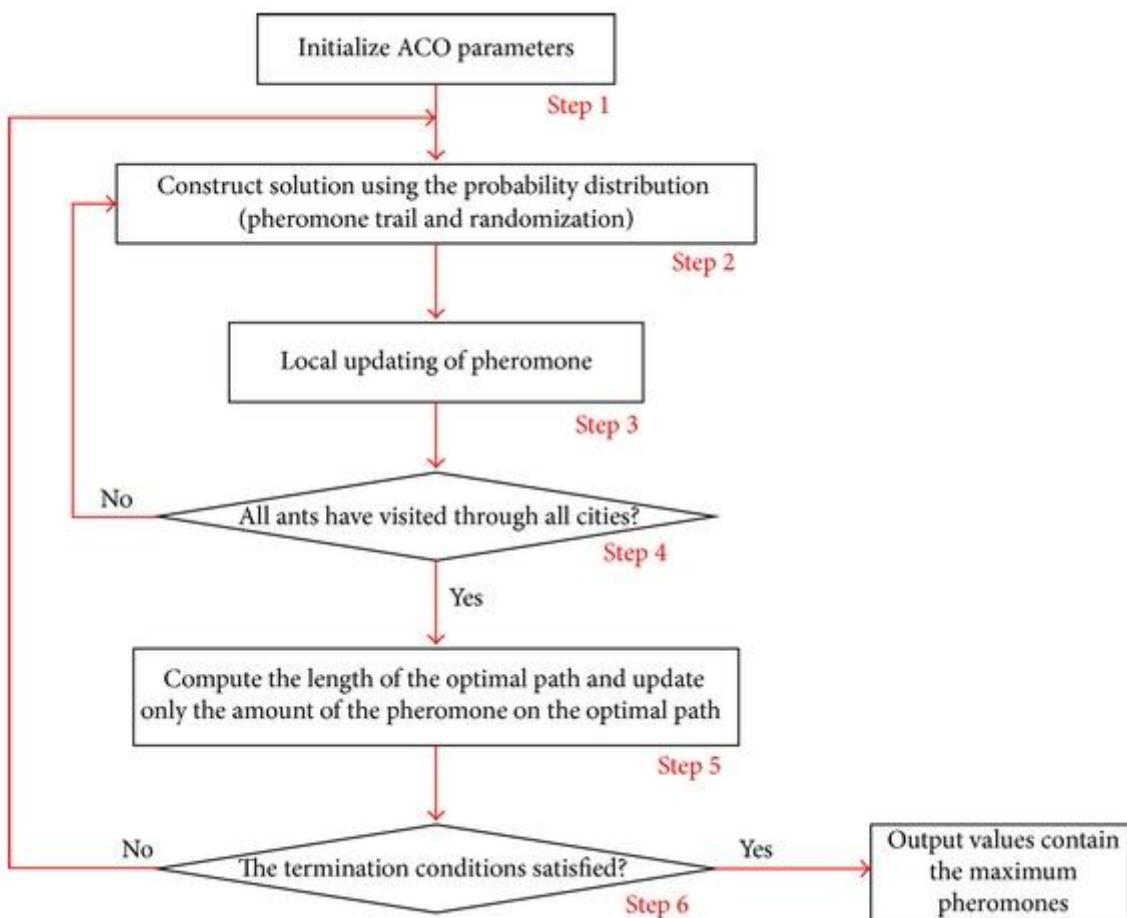
Karıncaların turlarını oluştur.

Karıncalar geçtiği yollardaki feromonu güncelleştir

**End While**

**Şekil 4.4** ACO algoritması için sözde kod

ACO algoritması için akış diyagramı Şekil 4.5'te verilmiştir (Dorigo,1997).



**Şekil 4.5** ACO algoritması için akış diyagramı

#### 4.1.1 Geçiş Kuralı

ACO algoritmasında yapay karıncalar ilk iterasyonda başlangıç noktasından gidecekleri bir sonraki noktayı eşit olasılıkla seçmekte ve bu şekilde devam ederek ilk turlarını

tamamlamaktadır. Eşit olasılıklı seçim her bir yol için eşit olarak atanan feromon izleriyle sağlanır. Daha sonra her bir karıncanın geçiş yapmış olduğu yollardaki feromon miktarları, yolu uzunluğuyla da bağlantılı olarak arttırılır. Böylelikle tüm yollar için, geçiş sayısı ve uzunlukla orantılı olarak feromon güncellemesi yapılmaktadır. Bir sonraki ve devam eden iterasyonlarda yol tercihleri güncellenen feromon miktarlarına göre belirlenir.

ACO algoritmasında yol tercihi belli bir olasılığa bağlı olarak iki şekilde gerçekleştirilir: İlk seçenek  $q_0$  olasılıkla feromonun en yoğun olduğu yolu seçilmesidir.  $q_0$  parametresi genellikle % 90 olarak belirlenir.  $\tau(i, j)$   $i$  ve  $j$  noktaları arasındaki feromon miktarı, seçilebilirlik parametresi  $\eta(i, j)$ ,  $i$  ve  $j$  noktaları arasındaki mesafenin tersi ( $1/\delta(i, j)$ ),  $\alpha$  ve  $\beta$  ayarlanabilir parametreler olmak üzere,  $i$  noktasında bulunan bir karıncanın gideceği nokta Denklem 4.1 ile seçilmektedir (Dorigo, 1992).

$$j = \max_{u \in J_k(i)} \left\{ [\tau(i, u)]^\alpha \times [\eta(i, u)]^\beta \right\} \quad \text{eğer } q \leq q_0 \quad (4.1)$$

İkinci seçenek ise gidilmesi mümkün olan yollardan birini, yollardaki feromon izleriyle orantılı olarak seçmektir. Bu şekilde yol seçimi olasılığı  $1 - q_0$  oranındadır.  $J_k(i)$ ,  $i$  noktasındaki karıncanın gidebileceği noktalar yani ziyaret edilmemiş şehirleri temsil eder. Tüm şehirler için seçilme olasılıkları Denklem 4.2 ile hesaplanmaktadır (Dorigo, 1992).

$$p_k(i, j) = \begin{cases} \frac{[\tau(i, j)]^\alpha \times [\eta(i, j)]^\beta}{\sum_{u \in J_k(i)} [\tau(i, u)]^\alpha \times [\eta(i, u)]^\beta} & \text{eğer } j \in J_k(i) \\ 0 & \text{Diğer durumlarda} \end{cases} \quad (4.2)$$

Bu olasılıklara bağlı olarak yol seçilmektedir. Feromonun yoğun olduğu yolların seçilme olasılığı yüksektir.

#### 4.1.2 Feromon Güncellemesi

Feromon güncellemesi, çözüm uzayının taranması amacıyla yapılmaktadır. Güncelleme işlemi tüm karıncalar turlarını tamamladıktan sonra yapılır. Feromon güncellemesinin iki temel elemanı vardır:

- a) Tüm yollardaki feromonların, belirlenen oranda (buharlaşma oranı) buharlaştırılması.

b) Karıncaların geçiş yapmış oldukları yollardaki feromon miktarlarının, o yolu kullanan karıncanın yol uzunluğuyla ters orantılı olarak arttırılması.

Buharlaşma oranı daha önceki çözümlerin önemini azaltılmasını sağlamaktadır. Yol uzunluğuyla ters orantılı olarak feromon artışı ise, iyi çözümlerin önemini arttırmamasını temin eder.

Karinca kolonisi optimizasyonlarının değişik versiyonlarında farklı feromon yenileme kuralları kullanılmıştır. ACO algoritmasında feromon yenilemesi lokal ve global olmak üzere iki düzeyde gerçekleşmekte ve bir yoldaki toplam feromon düzeyi; lokal ve global feromon düzeyinin toplamından oluşmaktadır.

#### 4.1.3 Lokal Feromon Güncellemesi

$\tau_{ij}(t)$ , t iterasyonuna kadar biriken feromon düzeyi,  $\Delta\tau_{ij}^k(t+1)$ , t iterasyonundaki feromon düzeyi ve  $\rho$  ( $0 \leq \rho \leq 1$ ), feromon buharlaşma parametresi olmak üzere lokal feromon düzeyi Denklem 4.3 ve Denklem 4.4'teki formüllerle hesaplanır (Dorigo,1992).

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \sum_{k=1}^m \Delta\tau_{ij}^k(t+1) \quad (4.3)$$

$\Delta\tau_{ij}^k(t+1)$ , aşağıdaki formülle hesaplanır:

$$\Delta\tau_{ij}^k(t+1) = \begin{cases} 1/L^k(t+1) & \text{k karıncası (i, j)} \\ & \text{yolunu kullanmışsa,} \\ 0 & \text{diğer durumlarda} \end{cases} \quad (4.4)$$

$L^k(t+1)$  k karıncasının toplam tur uzunluğudur. Lokal feromon güncellemesi, turları dinamik olarak değiştirerek geçiş yapılan yolları cazip hale getirir. Karıncalar değişen feromon miktarlarına bağlı olarak her iterasyonda turlarını da değiştirmektedirler. Böylelikle sürekli olarak daha kısa turları bulmak amaçlanmaktadır.

#### 4.1.4 Global Feromon Güncellemesi

ACO algoritmasında, global feromon güncellemesi, geçerli iterasyondaki en iyi sonuca sahip karıncaının izlediği yolu feromon düzeyinin arttırılmasından oluşur ve iterasyonlarda bulunan en iyi sonuçların belli bir oranda ileriki iterasyonlara aktarılmasını sağlar.

Global feromon güncellemesi lokal feromon güncellemesine benzer formüllerle hesaplanmaktadır. Bahsi geçen bu formüller Denklem 4.5 ve Denklem 4.6'da verilen formüllerdir (Dorigo,1992).

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \Delta \tau_{ij}^k(t+1) \quad (4.5)$$

$$\Delta \tau_{ij}(t+1) = \begin{cases} \frac{1}{L_{best}(t+1)} & (i, j) \text{ en iyi tura ait ise} \\ 0 & \text{diğer durumlarda} \end{cases} \quad (4.6)$$

$L_{best}(t+1)$  geçerli iterasyonda bulunan en iyi turun uzunluğudur.

#### 4.1.5 Optimum Karınca Sayısı

Karınca sayısının artırılması çözümde iyileşmeye neden olur. Fakat hesaplamaları artırdığı için karınca sayısının fazla artırılması işlem zamanlarının uzamasına neden olur.

GSP problemlerdeki yapılan denemeler sonucunda karınca sayısının şehir sayısına eşit seçilmesinin uygun olacağı sonucuna varılmıştır. Karınca sayısı, problem büyüklüğüne ve uygulama alanına bağlı olarak değişir.

#### 4.1.6 Parametre Değerleri

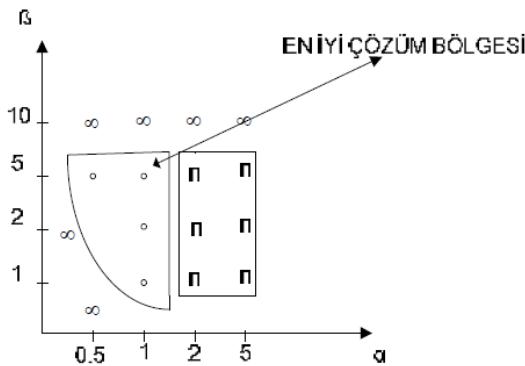
Karınca algoritmalarında ayarlanabilir temel parametreler; karınca sayısı,  $q_0$ ,  $\alpha$  ve  $\beta$ 'dır. Parametrelerin etkin olarak seçilmesi, algoritmanın performansında önemli derecede iyileşme sağlamaktadır. Parametrelerin listelendiği tablo Şekil 4.7'de gösterilmektedir (Tsai,2004).

Karınca sayısının artırılması çözümde iyileşme sağlamakla beraber, hesaplamaları artırdığı için işlem zamanlarının uzamasına neden olur. TSP problemlerdeki yapılan denemeler sonucunda karınca sayısının şehir sayısına eşit seçilmesinin uygun olacağı sonucuna varılmıştır. Karınca sayısı, problem büyüklüğüne ve uygulama alanına bağlı olarak değişmektedir.

$q_0$  değeri en iyi çözümün sonraki iterasyonlara aktarılması olasılığını belirleyen parametredir.  $q_0$ , gezgin satıcı problemlerde genellikle %90 olarak alınmıştır. Bunun anlamı, mevcut karıncaının %90 olasılıkla en iyi çözümü sağlayan yolu takip etmesi, %10 olasılıkla da feromon izlerine bağlı olarak yolunu belirlemesidir.

$\alpha$  değeri, ilgili yolun feromon miktarının önemini belirler ve önceki iterasyonların sonuçlarının ilerleyen iterasyonlara aktarılmasını temin eder.  $\alpha$  değerinin yüksek olması feromonun yoğun olduğu yolların seçilme olasılığını arttırmaktadır.  $\beta$  değeri ise yol uzunlıklarının, bir sonraki noktanın seçimindeki etkisini belirlemektedir.  $\beta$  değeri arttıkça bir sonraki yolun seçiminde rassallık artmaktadır.  $\beta$ 'nın düşük olması ise alternatif çözümlerin araştırılması ihtimalini azaltır.

$\alpha$  ve  $\beta$  parametre değerleri birbirinden bağımsız değildir. Problemin özelliğine göre değişen bu parametreler için denemeler yapılarak en etkin parametrelerin belirlenmesi gereklidir.  $\alpha$  ve  $\beta$  arasındaki ilişki Şekil 4.6'da açıklanmıştır (Middendorf, 2002).



Şekil 4.6 Parametre seçimi

Parametre	Anlamı
$n_k$	Karinca sayısı
$n_t$	En fazla iterasyon sayısı
$T_0$	Başlangıç feromon miktarı
$\rho$	Feromon buharlaşma oranı
$\alpha$	Feromon kuvvetlendirme oranı
$\beta$	Sezgisellik kuvvetlendirme oranı

Şekil 4.7 Parametre listesi

## 4.2 Karıncalar İle İlgili Diğer Algoritmalar

### 4.2.1 Maksimum-Minimum Karınca Sistemi

MMAS'yi klasik karınca optimizasyonundan ayıran iki temel özellikten ilki, her iterasyonda sadece bir karıncanın feromon yenilemesine izin verilerek çözümün sağlanmasıdır. Bu karınca, bir önceki iterasyondaki en iyi çözümü veren karıncadır. İkinci özellik ise, aramadaki dalgalanmayı önlemek için, feromon izlerinin sınırlı bir aralıktır. Alt ve üst sınırlar  $[\tau_{\min}, \tau_{\max}]$ ,  $\tau_{\min} \leq \tau \leq \tau_{\max}$  şeklinde ifade edilir. Feromon izleri üst limitten başlatılır ve bu da başlangıçta yüksek oranda iyileşme sağlar. MMAS'de alt ve üst limitler uygun seçilmediği takdirde tüm karıncalar aynı yoldan gitmekte ve bu da iyi bir çözümün bulunmasını engellemektedir. Alt ve üst sınırlar Denklem 4.7'deki formüllerle hesaplanır (Stüzle, 2000).

$$\tau_{\max} = \frac{1}{\rho} \times \frac{1}{L^{\text{eniyi}}}, \quad \tau_{\min} = \frac{\tau_{\max}}{2n} \quad (4.7)$$

$L^{\text{eniyi}}$  = bulunan en iyi turun uzunluğu

#### 4.2.2 Rank Temelli Karınca Sistemi

AS<sub>rank</sub>'da, geçerli iterasyondaki en iyi m tane karıncanın feromon güncellemesine izin verilir. Bu amaçla karıncalar tur uzunluklarına göre ( $L_1(t) \leq L_2(t) \leq \dots L_m(t)$ ) şeklinde sıralanır ve bir karıncanın feromon düzeyi karıncanın r (sıra) değerine bağlıdır. Böylece mevcut en iyi çözümün feromon güncelleme düzeyinin en yüksek olması sağlanmış olur. AS<sub>rank</sub>'da feromon güncelleme formülleri Denklem 4.8 ve Denklem 4.9'daki gibidir (Gambardella, 2009).

$$\tau_{ij}^r(t+1) = (1-p)\tau_{ij}^r(t) + \sum_{r=1}^{w-1} (w-r)\Delta\tau_{ij}^r(t) + w\Delta\tau_{ij}^{eniyi}(t) \quad (4.8)$$

$$\Delta\tau_{ij}^r(t) = 1/L^r(t), \quad \Delta\tau_{ij}^{gb}(t) = 1/L^{eniyi} \quad (4.9)$$

#### 4.2.3 Çoklu Karınca Koloni Algoritmaları

Çoklu koloni yaklaşımı, iki veya daha fazla koloninin çözümünün birleştirilmesi esasına dayanır. Bu konudaki iki temel yaklaşımından ilki olan heterojen yaklaşım tüm kolonilerin birbirinden farklı davranış sergilemesi esasına dayanmaktadır ve çok kriterli optimizasyon problemlerinin çözümünde kullanılmıştır.

İkinci yaklaşımda tüm koloniler birbirine paralel olarak çalışırlar ve her i jenerasyon sonunda koloniler arasında feromonlar vasıtası ile karşılıklı bilgi alışverişi olur. Sözü edilen ikinci yaklaşım, bilgi alışverisinin sınırlı olduğu ve çok kısa olmayan aralıklarla yapıldığı takdirde başarılı sonuçlar vermektedir.

Bilgi alışverişi için önerilen 4 yöntem vardır:

- Her bilgi alışverişinde global en iyi çözümler hesaplanır ve tüm kolonilere gönderilir. Böylece her koloni yeni bir lokal en iyi çözüme sahip olur.
- Koloniler arasında bir köprü kurulur ve her bilgi alışverişi adımda, her koloni lokal en iyi çözümleri en yakınındaki bir ya da daha fazla koloniye gönderir. Tsai ve arkadaşları ikili en yakın komşu ve çoklu en yakın komşu ile denemeler yapmışlar, ikili en yakın komşuda en iyi sonucu almışlardır .
- (b) seçenekindeki bilgiler kullanılarak her bilgi alışverişi adımda, bir kolonideki en iyi  $m_b$  karınca, en yakınındaki en iyi  $m_b$  karıncayla karşılaşılır ve  $2m_b$  karıncanın en iyi  $m_b$  tanesi alınarak feromon güncellemesinde kullanılır.
- Bu seçenek (b) ve (c) seçeneklerinin birleştirilmesinden oluşur.

#### 4.2.4 Melez ACO

ACO yaklaşımı ile diğer yaklaşımın birleştirilmesinden elde edilen yeni algoritmaların genel adıdır. Gambardella ve Dorigo çalışmalarında, ACO ile MPQ/AI algoritmasını birleştirerek ardışık sipariş problemlerine uygulamışlar ve işlem zamanından %2 ile %30 arasında kazanç sağlamışlardır. Lee kaynak atama problemlerinin çözümünde genetik algoritma ile ACO'yu birleştirerek elde ettiği yeni algoritma ile çözerek iyi sonuçlar elde etmiştir.

### **4.3 Uygulama Alanları**

ACO algoritmasının ve diğer sezgisel algoritmaların kullanım alanları en kısa zaman problemleriyle sınırlı değildir. Karıncalar yalnızca bu amaç için kullansa da matematiksel formüller değiştirilerek aynı yapı en az kaynak (para veya enerji) veya başka optimizasyonlar için kullanılabilir. Örneğin yukarıda bahsettiğimiz gibi bir kargo şirketinin dağıtım rotasını belirlemek için kullanabilir -ki kargo şirketleri için sezgisel algoritmalar kullanan bu tür yazılımlar vardır. Bunun yanında en kısa zaman yerine maksimum sözcük ilişkisi koyarsak, rahatlıkla Google benzeri bir metin tarama ve arama sistemi elde ederiz.

Google ve benzeri arama motorları arka planda sözcük ilişkisi ve sayfa değerini maksimize etmekte optimizasyon algoritmalarını kullanırlar. Google optimizasyon için kullandığı algoritmayı (ve bu algoritmanın bu yazında bahsedilen sezgisel algoritmalarдан biri olup olmadığını) açıklamamaktadır. Bununla beraber örneğin ACO algoritması bir arama motoru tarafından bu amaçla kullanılabilir.

ACO algoritması literatürde sıkılıkla optimizasyon problemlerinde uygulanmıştır. Bu problemlerden bazıları şöyledir.

- Gezgin Satıcı Problemleri
- Araç Rotalama Problemleri
- Tesis Yeri Atama Problemleri
- İş Çizelgeleme Problemleri
- Harita Renklendirme Problemleri
- Kareli Atma Problemleri
- Aritmi sınıflandırma

### **4.4 Seçilen Örnek Problem : ACO ile Aritmi Sınıflama**

Elektrokardiyogram (EKG) işaretleri, kalbin çalışma şekliyle alakalı birçok bilgiyi içermesi, taşınabilir ve düşük maliyetli cihazlar vasıtasyyla kolayca toplanabilmesi nedeniyle kalp hastalıklarının teşhisini ve tedavi yönteminin belirlenmesinde en çok kullanılan yöntemlerden biridir. EKG işaretlerinde ortaya çıkan ritim ve şekil bozuklukları (aritmiler) kalp kriz belirtilerinin önceden tespiti, kalbin çalışma bozukluklarının teşhisini gibi alanlarda yoğunlukla kullanılmaktadır.

EKG işaretinin bilgisayar ortamında işlenmesinde karşılaşılan en önemli problemlerden biri EKG dalga şeklindeki değişkenlidir. Aynı hastadan farklı zaman ve şartlarda alınan EKG işaretinde dahi şekil ve ritim farklılıklarını olabilmektedir. Geçtiğimiz on yıllarda, EKG analizine kümeleme temelli algoritmalar diğer tekniklerle birlikte kullanılarak bu güçlüklerin azaltılmasına çalışılmıştır. Bu tekniklerin kullanıldığı alanlar EKG aritmi sınıflama, EKG öznitelik seçimi, EKG karakteristik noktalarının bulunması ve EKG şekil analizlerinin yapılması olarak belirtilebilir (Nizam ve Korürek,2011).

Bu çalışmada, belirlenen aritmi tiplerinin görüntüleri üzerinde frekans düzleminde öznitelik çıkarımı yapılmış ve ACO ile aritmeler sınıflandırılmıştır. Çalışmanın görsel taslağı Şekil 4.9'daki gibidir.

#### 4.5 Problemin ACO ile Çözülmesi

Problem, MATLAB R2017a platformu ve bu platformun içindeki Statistics Toolbox'ı (plot fonksiyonu) kullanılarak çözülmüştür. (Algoritmanın genel kodu <https://uk.mathworks.com> adresinden alınmıştır.) Aynı zamanda MATLAB'ın temel fonksiyonları olan figure, display, scatter ve title fonksiyonları kullanılmıştır. (Problemin detaylı MATLAB kodu EK B.1' de gösterilmiştir.)

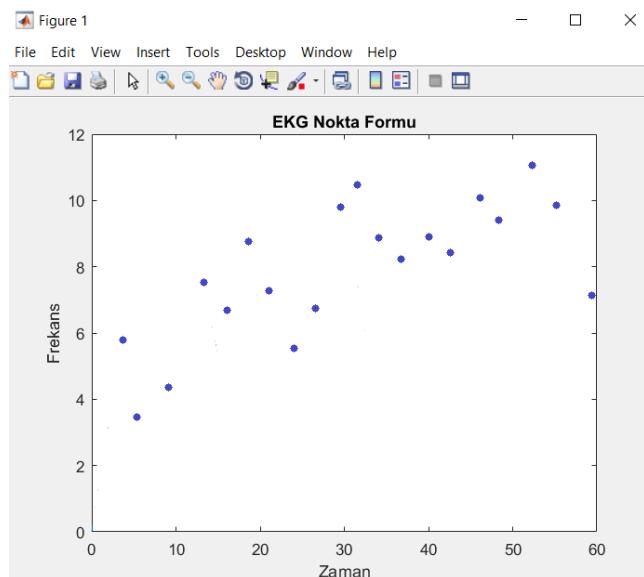
ACO algoritması kullanılarak EKG görüntüsündeki kalp atışının %59 başarı oranında hangi aritmi sınıfına ait olduğu tespit edilebilmektedir. Bu başarım oranı MATLAB yardımıyla oluşturulan 100 farklı EKG sinyali üzerinde denenmiş ve 59 tanesi doğru sınıflandırılmıştır.

Bu çalışmada, belirlenen aritmi sinyalleri üzerinde frekans düzleminde öznitelik çıkarımı yapılmış ve ACO ile aritmeler sınıflandırılmıştır. Benzesim oranı **maksimize** edilmiştir.

Problemin çözümnesine uygun zemin oluşturmak için öncelikle EKG sinyalleri dalga formundan nokta formuna geçirilmiştir. EKG sinyallerinden 3 saniyelik aralıklarla noktalar işaretlenmiş ve işaretlenen noktaların frekans değerleri belirlenmiştir. Bu işlemden sonra her EKG sinyali için 20 tane nokta elde edilmiş ve grafiğe dökülmüştür. Bu işlemin MATLAB çıktısı Şekil 4.8'de görülmektedir.

Her sinyal üzerinde belirlenen noktaların noktaların frekans değerleri 1x20 boyutunda vektörlerde tutulmuştur. Çalışmanın kalanı görüntüler üzerinde değil vektörler üzerinde devam etmektedir.

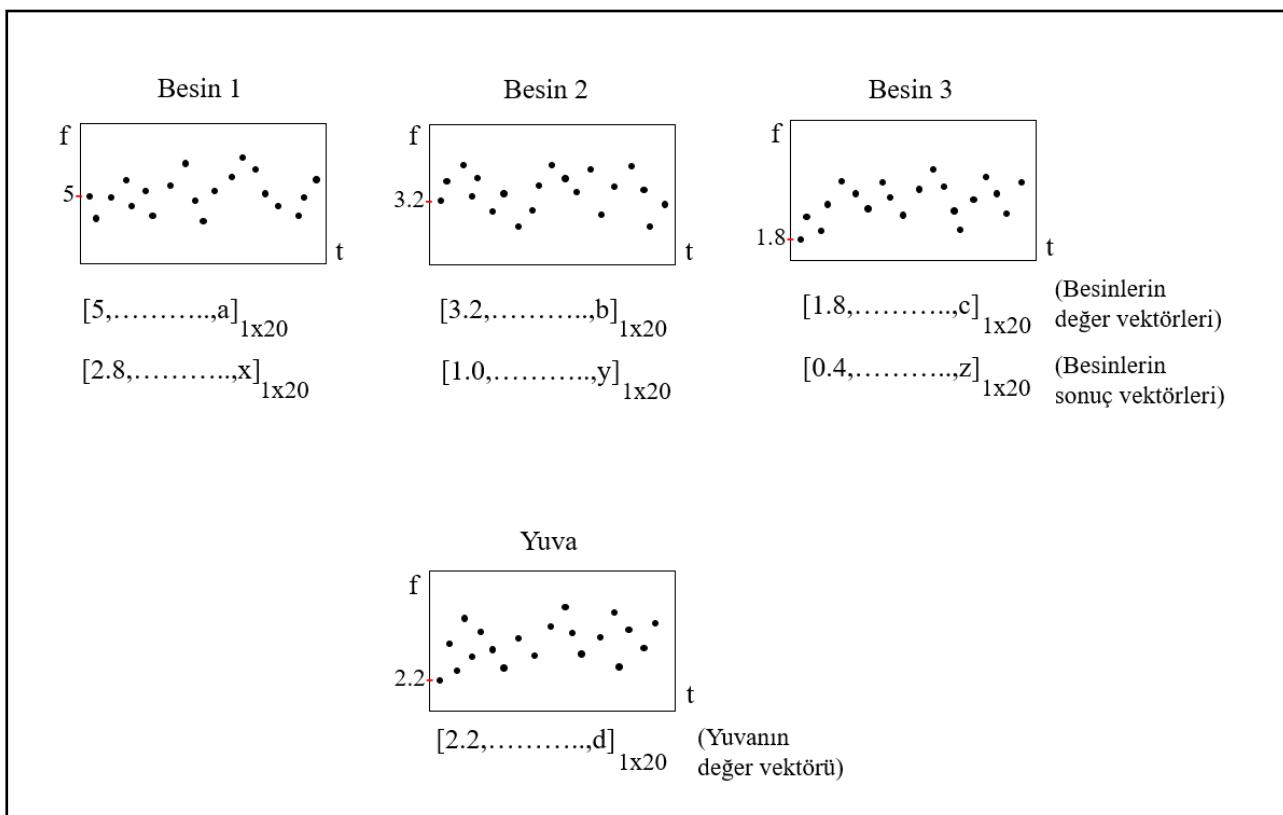
Örnek problemin çözümünde aritmi tipi bulunmak istenen EKG sinyali yuva, nokta formundaki her bir nokta bir karınca, veritabanı olarak kullanılan aritmi tipleri besin kaynakları olarak belirlenmiştir. İşlemleri kolaylaştmak için yuva için bir değer vektörü, her bir besin kaynağı için de bir değer ve bir sonuç vektörü oluşturulmuştur. Vektörler 1x20 boyutundadır. Değer vektöründe sinaldeki noktaların frekans değerleri tutulmaktadır. Sonuç vektöründe besin ve yuvanın değer vektöründeki ilgili elemanların farkları tutulur.



Şekil 4.8 EKG görüntüsü nokta formu

Aritmi tipi belirlenmek istenen görüntünün değer vektörünün ilk değeri için örnek aritmilerin vektörlerinin ilk değerleri karşılaştırılmıştır. Çünkü karıncalar başlangıçta yollarda feromon izi olmadığı için rastlantısal olarak besin kaynaklarına ve bu kaynaklara giden yollara dağılırlar. En kısa yolu bulup yuvaya geri dönen karıncanın kullandığı yolda feromon miktarı artar. Karşılaştırma sonuçlarının yer aldığı sonuç matrisleri oluşturulmuştur. Karşılaştırma sonucunda aynı veya en yakın değerin bulunduğu vektörün feromon değeri yükseltilmiştir.

Feromon değeri yükseltildiği için karşılaştırmaya bu vektörden devam edilmiştir. Elde edilen sonuçlar sonuç vektöründe tutulmuştur. Daha sonra diğer vektörler için de aynı işlemler tekrarlanmıştır. Son aşamada üç sonuç vektörü elde edilmiştir. Bu vektörlerin elemanları toplanmıştır ve birbirleriyle karşılaştırılmıştır. Toplam değeri en küçük olan vektöre sahip aritmi tipi sonuç olarak belirlenmiştir. En küçük değere sahip bir başka vektör daha varsa bu iki aritmi tipi için algoritma tekrar çalıştırılır. Problem çözümünün adımları Şekil 4.10'daki gibidir.



**Şekil 4.9** Problem çözümünün görsel taslağı

Şekil 4.9'da Besin 1, Besin 2, Besin 3 ve Yuva olarak gösterilen şekiller veritabanı olarak kullanılan ve tipi belirlenmek istenen aritmi tiplerinin üçer saniye arayla frekans değerlerinin işaretlendiği nokta formu gösterimdir. Besinlerin ve yuvanın altında çözüm adımlarında kullanılan vektörler gösterilmiştir.

**Adım 1:** Tüm EKG sinyalleri dalga formundan nokta formuna geçirilir. Her noktanın frekans eksenindeki değeri vektörüne yerleştirilir. Feromon yolun seçilme sırasını belirleyen matematiksel bir fonksiyondur. Uygulamada feromon yerine kullanılan fonksiyonun her aritmi için başlangıç değeri belirlenir. Tüm yollar için aynı değer atanır.

**Adım 2 :** Yuvaladaki karıncalar (tipi belirlenmek istenen EKG sinyalinin değer vektörünün her elemanı) besinlere(veritabanı olarak tanımlanan EKG sinyallerinin değer vektörleri) rassal olarak dağılırlar.

**Adım 3 :** Her karınca gittiği besine olan yolu hesaplar. (Besin ve yuhanın değer vektörlerindeki ilgili elemanların farkı mutlak değerle alınır ve her besin için oluşturulan sonuç vektörüne yazılır.)

**Adım 4 :** Her bir besinin sonuç vektörüne bakılır ve en küçük değere sahip olan besinden devam edilir ve lokal feromon güncellemesi yapılır. (En küçük değere sahip besinin diğer elemanları için aynı işlem yapılır ve değerler sonuç vektörüne yerleştirilir.)

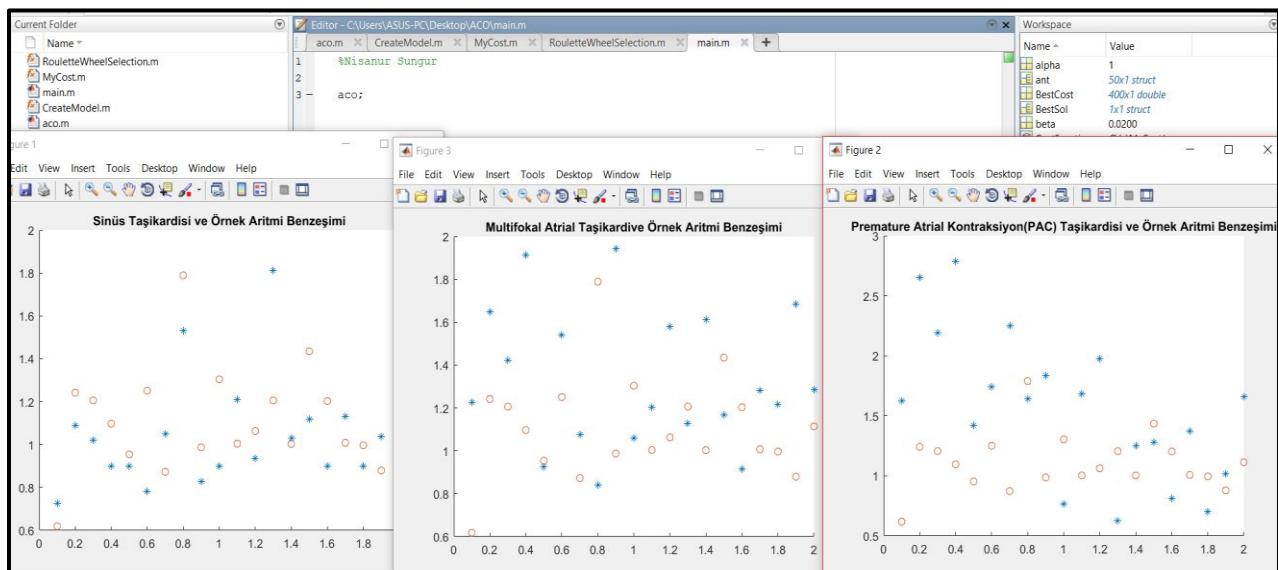
**Adım 5 :** Üzerinde çalışılan besin bittiği için diğer besinler için de aynı işlemler tekrarlanır ve sonuç vektörleri oluşturulur. Her bir besinin sonuç vektöründeki elemanlar toplanır ve en küçük değere sahip olan besin en iyi çözüm olarak belirlenir.

**Adım 6 :** Maksimum iterasyon sayısı sağlanana kadar Adım 2' ye gidilir.

**Şekil 4.10** Aritmi sınıflama probleminin ACO ile çözüm adımları

## 4.6 Örnek Çalışma

Program çalıştırıldığında veritabanı olarak tanımlanan 3 tip aritmi(sinüs taşikardi, multifokal atrial taşikardi,PAC taşikardi) ve tipi bulunmak istenen aritminin EKG sinyalinin nokta formu ve bu nokta formlarının karşılaştırılması Şekil 4.11 (a), Şekil 4.11 (b), Şekil 4.11 (c)'de gösterilmektedir.



**Şekil 4.11 (a)** Sinüs taşikardisi ve örnek aritmi benzeşimi

**Şekil 4.11 (b)** Multifokal atrial taşikardı ve örnek aritmi benzeşimi

**Şekil 4.11 (c)** PAC taşikardı ve örnek aritmi benzeşimi

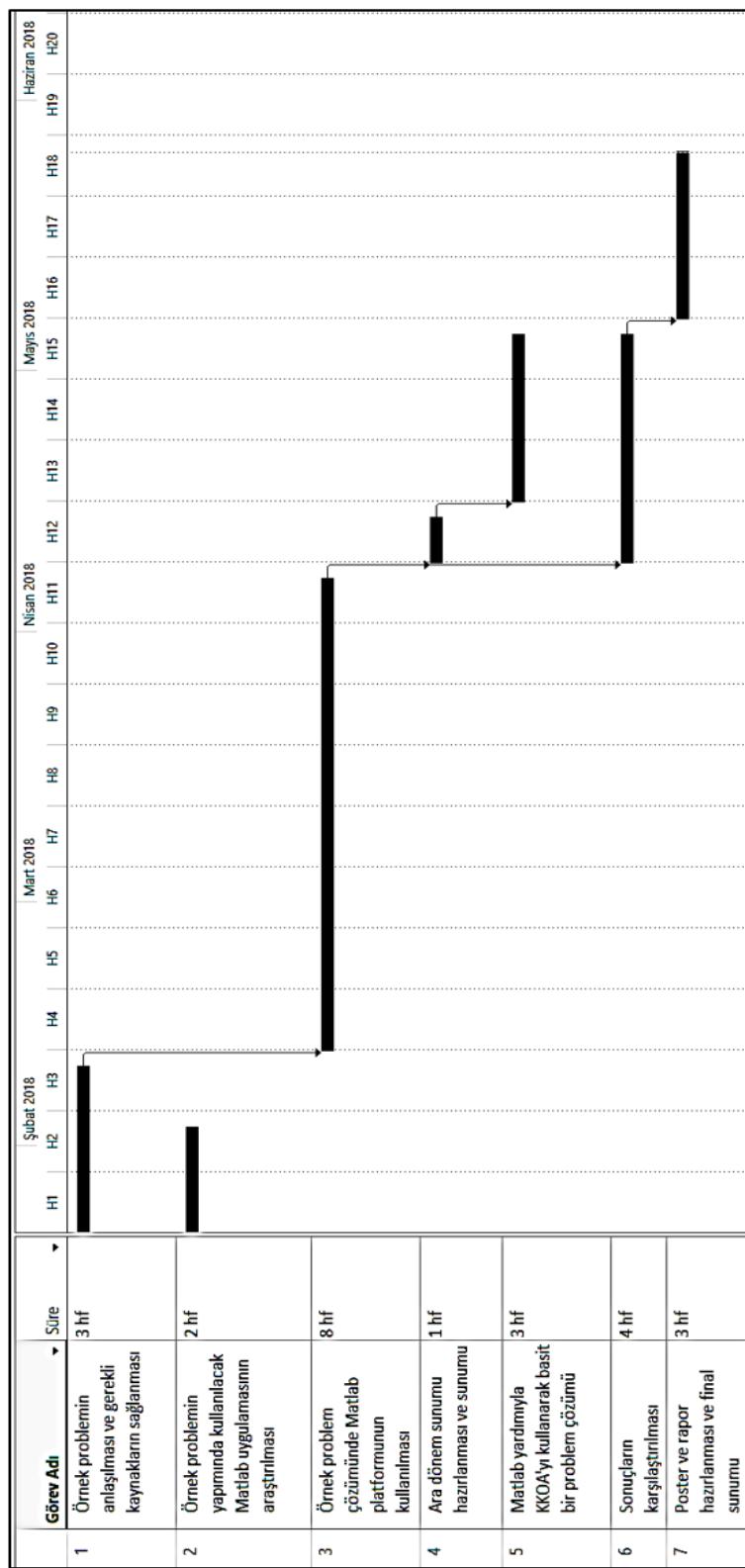
#### **4.7 Sonuç**

Doğal karınca kolonilerinin davranışlarından etkilenerek ortaya olmuş olan karınca algoritmaları, pek çok problemin çözümünde başarıyla kullanılmıştır. Bu çalışmada algoritmalarının aritmi sınıflama problemine çözüm getirirken hangi adımları izlediği belirtilmiştir. ACO algoritması sezgiselinde parametre optimizasyonu yapılarak optimuma yakın sonuçların elde edilebileceği açıklanmıştır. ACO parametrelerinin seçim aralıkları, algoritmasının optimuma ulaşması üzerinde önemli bir faktördür.

# Karıncalar ile Süre Optimalizasyonu

## 2017-2018 Akademik Yılı Bahar Dönemi

### İş-Zaman Planı



Hazırlayan : 21494986 - Nisanur Sungur

20.03.2018

#### **4.9 Kaynakça**

- Ural, B. ve M.Yüksek “ Mühendisler İçin MATLAB’ın Temelleri ve Mühendislik Matematiği Uygulamaları ” İstanbul: NOBEL Akademik Yayıncılık, (2017).
- Gonzalez, R. ve R. Woods “ Sayısal Görüntü İşleme ” İstanbul: Palme Yayıncılık, (2014).
- Nizam, A. ve M. Korürek “Karınca Koloni Optimizasyonuna Dayalı Yeni Bir Aritmi Sınıflama Tekniği” İTÜ Fen Bilimleri Enstitüsü, Biyomedikal Mühendisliği Programı, itüdergi/d mühendislik cilt:10, sayı:1, ss. 21-30,(2011).
- Gambardella, L., Taillard, E.D., Agazzi, G. “MACS-VRPT: A Multiple Ant Colony System with for Vehicle Routing Problems with Time (sec.) Windows”, New Ideas in Optimization, Ed.: Corne, D., Dorigo, M., Glover, F., Mc Graw-Hill, ss. 63-76, (2009).
- Karaboğa, D. “ Yapay Zeka Optimizasyon Algoritmaları ” İstanbul: NOBEL Akademik Yayıncılık, (2004).
- Tsai, C.F. “A New Hybrid Heuristic Approach for Solving Large Traveling Salesman Problem”, Information Sciences, 166(1-4), ss. 67-81,(2004).
- Middendorf, M. “Multi Colony Ant Algorithms”, Journal of Heuristics, 8-3, ABI/INFORM Global, ss. 305-320, (2002).
- Randall, M. “A Parallel Implementation of Ant Colony Optimization”, Journal of Parallel and Distributed Computing, 62, ss. 1421–1432, (2002).
- Stützle, T. ve Hoos, H.H. “Max Min Ant System”, Journal of Future Generation Computer Systems, 8 (16), ss. 889–914, (2000).
- Dorigo, M. ve Gambardella, L.M. “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”, IEEE Transaction on Evolutionary Computation, 1, ss. 53-66, (1997).

## V KUŞLAR İLE SÜRÜ OPTİMİZASYONU

### 5.1 Kuşlar İle Sürü Optimizasyonu - Parçacık Sürü Optimizasyonu (PSO) Algoritması

İlk olarak 1995–1996 yıllarında sosyolog-psikolog James Kennedy ve elektrik mühendisi Russel Eberhart tarafından, Sezgisel yöntemlerden biri olan Parçacık Sürü Optimizasyonu tekniği kuş ve balık sürülerinin hareketlerinden esinlenerek, doğrusal olmayan nümerik problemlere optimal sonuçlar bulmak için ortaya atılmıştır. Tabanında sosyal etkileşim yatan ve sürü zekâsını temel alan bu algoritma, biyolojik toplumlardaki toplu zekâyı araştıran sosyobilişsel bir çalışma sırasında kuş sürülerinin hareketlerinin temsilinin gerçekleştirilmesi sonucunda ortaya koyulmuştur. Araştırmalar sonucunda görülmüştür ki Sürü halinde hareket eden hayvanların yiyecek ve güvenlik gibi durumlarda, çoğu zaman rasgele ve içgüdüsel olarak sergiledikleri hareketlerin, amaçlarına daha kolay ulaşmalarını sağlamıştır. PSO, Çok parametreli ve çok değişkenli optimizasyon problemlerine çözüm bulmak için kullanılmaktadır. Parçacık sürü optimizasyonu az parametre gerektirmesi, yerel optimumlara takılma riskinin az olması ve hızlı sonuç bulması sebebiyle diğer birçok arama algoritmasının önüne geçmiştir. PSO algoritmasının senaryosu Bir alanda yiyecek arayan bir kuş sürüsünün, yiyecek kaynağına en yakın kuş takip etmesi üzerine oluşturulmaktadır. Bu Algoritmada en önemli nokta optimizasyon problemini ifade eden amaç fonksiyonunu oluşturmaktır. Ayrıca PSO nun bir artısı daha, bu algoritma çeşidinde amaç fonksiyonunun değişimini elde etmek için matematiksel işlemlere ihtiyaç duymayan bir global optimizasyon algoritma olmasıdır. Metodun ilk orijinal versiyonu sadece lineer olmayan sürekli optimizasyon problemlerin çözümünde kullanılıabiliyordu. Daha sonra metot geliştirilerek daha karmaşık mühendislik problemlerinin global optimal çözümlerinde kullanılabilir hale geldi. Bu teknığın bilinen dezavantajı, gerçek optimumu bulmadan önce erken yakınsama sergilememesidir.

PSO, parçacık adı verilen ve her biri ilgili probleme farklı bir çözüm önerisi getiren elemanlardan oluşur. Bu parçacık topluluğuna sürü adı verilir. Süredek bütün parçacıklar çözüm uzayında rastgele değerler alarak arama işlemeye başlarlar. Algoritmada bir sürü birçok parçacıkta meydana gelmekte ve her bir parçacık bir potansiyel çözümü göstermektedir. PSO bireyler arasındaki sosyal bilgi paylaşımını esas alır. Kuş ve balık sürüleri yiyecek ya da barınak bulmak için belirli bir alanı tararlar. PSO algoritması, bu sürülerin soysal davranışlarından oluşur. Bu davranışlardan ilki süredeki her bir parçanın geçmiş hatırları içerisinde en iyi konuma gitme davranışıdır. İkinci davranış sürü içerisindeki yiyeceğe en yakın parçacığı takip etme hareketidir. Son davranış ise parçacığın geniş alan taramasını sağlayan geçmiş hız değeridir. Bu davranışlar PSO algoritmasının temelini oluşturur. Her bir parçacık kendi pozisyonunu, bir önceki tecrübesinden yararlanarak süredeki en iyi pozisyonu doğru ayarlar. PSO, temel olarak sürüde bulunan bireylerin pozisyonunun, sürünen en iyi pozisyonu sahip olan bireyine yaklaşırmasına dayanır. Bu yaklaşma hızı rasgele gelişen durumdur ve çoğu zaman sürü içinde bulunan bireyler yeni hareketlerinde bir önceki konumdan daha iyi konuma gelirler ve bu süreç hedefe ulaşıcaya kadar devam eder. Eğer bir en-azlama (minimizasyon) problemiyle uğraşıyorsak bu uygunluk değeri, parçacığın ziyaret ettiği konumların uygunluk fonksiyonunu minimize etme yetenekleri ile ölçülür. Fonksiyonu en iyi minimize eden konum, en yüksek uygunluk değerine sahiptir (Erdoğan, 2016).

## 5.2 Parçacık Sürü Optimizasyonu'nda Kullanılan Terimler

Bu bölümde Parçacık Sürü Optimizasyonu'nun tanımlanmasında kullanılan terimler tanıtılacaktır. Bu terimler diğer evrimsel hesaplama yöntemlerine göre bazı farklılıklar göstermektedir.

Parçacık: Sürüdeki her bir birey, bir parçacık olarak adlandırılır. Sürüdeki her bir parçacık, bireysel olarak aynı hukmedici prensip altında hareket eder

Konum: Parçacık Sürü Optimizasyonu'nda konum, söz konusu problem için bir çözüm kümesi anlamına gelmektedir. PSO yalnızca parçacıkların hızlarına dayandığından teorik olarak sonsuz boyutlu bir problemin çözümüne ulaşmak mümkündür. Çözülmek istenen problemin "n" boyutlu olduğu varsayılsa, n-boyutlu bir uzay problemimizin çözüm uzayıdır ve bu uzaydaki herhangi bir koordinat (yani konum), incelenen problem için olası bir çözüm kümesidir. Örneğin üç boyutlu bir optimizasyon probleminde x-y-z koordinat sistemi bizim çözüm uzayımız, herhangi bir  $(x',y',z')$  konumu ise olası bir çözüm kümemizdir. Optimizasyonun amacı, problem için en uygun sonucu sağlayan  $(x',y',z')$  konumunu bulmaktır.

Sürü: Bütün parçacıkların oluşturduğu toplumdur.

Hız: Herhangi bir parçacığın, en iyi konuma ulaşmak için kullandığı, kendi bildiği ve sürünenin bildiği en iyi konuma göre değişen yönelme vektörüdür.

Uygunluk: Her evrimsel hesaplama yönteminde olduğu gibi, PSO'da da herhangi bir konumun uygunluğunu değerlendirmek için kullanılan bir fonksiyon vardır. Uygunluk fonksiyonu, konumun performansını değerlendirebilmek amacıyla o konumdan tek bir sayıdan oluşan bir değer oluşturur. Kısaca, ulaşımak istenen uygunluk fonksiyonu değerine göre konumun çözüme olan uygunluğu belirlenir.

Kişisel En İyi (pbest): Bir parçacığın kendi ulaştığı konumlar arasındaki en iyi uygunluk değerine sahip konumdur. Parçacık bir sonraki hızını belirlerken ve dolayısıyla bir sonraki konumuna ulaşırken, kendi kişisel en iyi konumundan faydalıdır. Hareketi boyunca her bir parçacık o anki konumunu kendi kişisel en iyi konumuyla karşılaştırır ve eğer o anki konumu daha önceki kişisel en iyi konumundan daha iyi bir uygunluk değerine sahipse, o anki konumu yeni kişisel en iyi konumu haline gelir.

Küresel En İyi (gbest): Bütün sürünenin ulaştığı en iyi konumdur. Sürüdeki bütün parçacıkların bu konumu bir şekilde bildikleri varsayılar ve her bir parçacık, bir sonraki konumuna doğru hareket ederken kendi en iyi konumunu ve bütün sürünenin ulaştığı en uygun konuma göre hareket eder. Kısacası bilinen en iyi uygunluk değerine sahip konumdur. Yolculuğu boyunca her bir parçacık o anki konumunu sürünenin en iyi konumuyla karşılaşır ve eğer o anki konumu sürünenin daha önceki en iyi konumundan daha iyi bir uygunluk değerine sahipse, o anki konumu sürünenin küresel en iyi konumu haline gelir.

İterasyon: Sürüdeki herhangi bir parçacığın zaman adımlarını temsil eder. Eğer sürünenin en iyi konumu arama süresini  $T$  saniye olarak belirlersek, her bir saniye bir iterasyona karşılık gelir.

Parçacık Sürü Optimizasyonu'nda bir parçacığın değişikliği yalnızca kendine ait hız vektörüne bağlıdır. Aralarındaki ilişki Denklem 5.1 ve Denklem 5.2'de ifade edilmiştir. Bu hız vektörü, parçacığın konumunu değiştirir ve yeni bir çözüm kümesi haline gelmesini sağlar. Parçacığın konumu,  $x$ , mevcut konumuna  $v$  hız vektörünün eklenmesiyle değiştirilir. Denklem 5.1'de görüldüğü üzere, Burada  $c1$  bilişsel bileşen ve  $c2$  sosyal bileşen olarak adlandırılır. Bilişsel bileşenin ( $c1$ ) yüksekliği parçacığın kendi kişisel en iyisine doğru daha fazla yönelmesini, sosyal bileşenin ( $c2$ ) yüksekliği parçacığın küresel en iyiye doğru daha fazla yönelmesi anlamına gelmektedir. Bilişsel bileşendeki bir artış, parçacığın kendi kişisel en iyisine doğru çekilmesine sebep olacağından çözüm uzayının daha iyi araştırılmasını sağlarken sosyal bileşenin artırılması ulaşılmak istenen küresel en yükseğin ortaya çıkışını hızlandırır (Özsağlam, 2008).

$$v_{n,i} = \omega v_{n,i-1} + c_1 rand_1 \ gbest_i - x_{n,i} + c_2 rand_2 \ pbest_{n,i} - x_{n,i} \quad (5.1)$$

$$x_{n,i+1} = x_{n,i} + v_{n,i} \quad (5.2)$$

Yukarıdaki denklemlerde görüldüğü üzere, Burada  $c1$  bilişsel bileşen ve  $c2$  sosyal bileşen olarak adlandırılır. Bilişsel bileşenin ( $c1$ ) yüksekliği parçacığın kendi kişisel en iyisine doğru daha fazla yönelmesini, sosyal bileşenin ( $c2$ ) yüksekliği parçacığın küresel en iyiye doğru daha fazla yönelmesi anlamına gelmektedir. Bilişsel bileşendeki bir artış, parçacığın kendi kişisel en iyisine doğru çekilmesine sebep olacağından çözüm uzayının daha iyi araştırılmasını sağlarken sosyal bileşenin artırılması ulaşılmak istenen küresel en yükseğin ortaya çıkışını hızlandırır.

Denklemdeki  $\omega$  bileşeni atalet bileşeni olarak adlandırılır ve 0 ile 1 arasında seçilmesi gereken bu bileşen, parçacığın bir önceki adıma ait olan hızına ne ölçüde sadık kalacağını belirler. Daha küçük bir atalet bileşeni o anki adımda öncekinden çok daha farklı bir hız vektörünün, daha büyük bir atalet bileşeni ise önceki hız'a yakın bir hız vektörünün parçacığı etkileyeceği anlamına gelir.

Denklemde yer alan  $rand1$  ve  $rand2$  değerleri [0,1] aralığında yer alan, isteğe bağlı olarak düzgün dağılımdan örneklenebilecek rastsal sayılardır. Bu rastsal değerler optimizasyon algoritmasına rastgelelik katmaktadır.

### 5.3 Parçacık Sürü Optimizasyonu Algoritması

Optimizasyon sırasında kullanılacak olan uygunluk fonksiyonu ve hesaplanmak istenen değişken sayısı (boyut) belirlendikten sonra algoritma, aşağıdaki adımlar kullanılarak probleme uygulanabilir. Bu adımların akış diyagramı Şekil 5.1'de verilmiştir (Eberhard, 1995).

#### Adım 1.

Başlangıç toplumu yaratılır. Eğer çözülmek istenen probleme ait kısıtlar var ise, başlangıç toplumundaki parçacıklar çözüm uzayı içerisinde yer almmalıdır. Dolayısıyla " $n$ " boyutlu bir optimizasyon probleminde " $n$ " tane boyut için en yüksek ve en düşük değerlerin sınırladığı bir başlangıç toplumu, rastgele değerler kullanılarak yaratılır. Yaratılan parçacıkların ( $x1, x2, \dots, xn$ ) koordinatları  $t=0$  anındaki konumları anlamına gelmektedir.

### Adım 2.

Her bir parçacık için uygunluk fonksiyonu hesaplanır. Her bir parçacığın başlangıç konumu, o parçacığın henüz karşılaştığı tek konum olduğundan parçacığın kişisel en iyi konumu haline gelir. İlk küresel en iyi, bu konumlar arasındaki en iyi uygunluk değerine sahip konumdur.

### Adım 3.

Her bir parçacık için "n" boyutlu bir hız vektörü, rastgele olarak yaratılır.

### Adım 4.

Her bir parçacığın hız değerine göre konumu güncellenir.

### Adım 5.

Her bir parçacık için yeni konumlarına göre uygunluk fonksiyonu hesaplanır. Eğer bir parçacığın hesaplanan yeni uygunluk değeri daha önceki kişisel en iyi uygunluk değerinden daha iyiysse, o anki konum kişisel en iyi konum olarak atanır. Benzer şekilde bütün sürü için hesaplanan uygunluk değerleri arasından bir değer daha önceki küresel en iyi değerden daha iyiysse, o anki en iyi uygunluk değerine karşılık gelen konum küresel en iyi konum olarak atanır.

### Adım 6.

Her bir parçacık için yeni hız vektörleri, PSO hız vektörü denklemi kullanılarak hesaplanır. Bu hız vektörleri kullanılarak parçacıkların konumları güncellenir. Hız vektöründe yer alan rastgele öğeler algoritma rastsallık katmaktadır. "Adım 5", bu yeni konumlar kullanılarak tekrarlanır.

### Adım 7.

Adım 5 ve Adım 6, durdurma koşulları sağlanana kadar tekrarlanır. Durdurma koşulu arzu edilen iterasyon sayısına ulaşma ya da belirli bir uygunluk fonksiyonu değerine ulaşma olabilir. Parçacık sürü optimizasyonunda genellikle belirli bir iterasyon sayısına ulaşma tercih edilir.

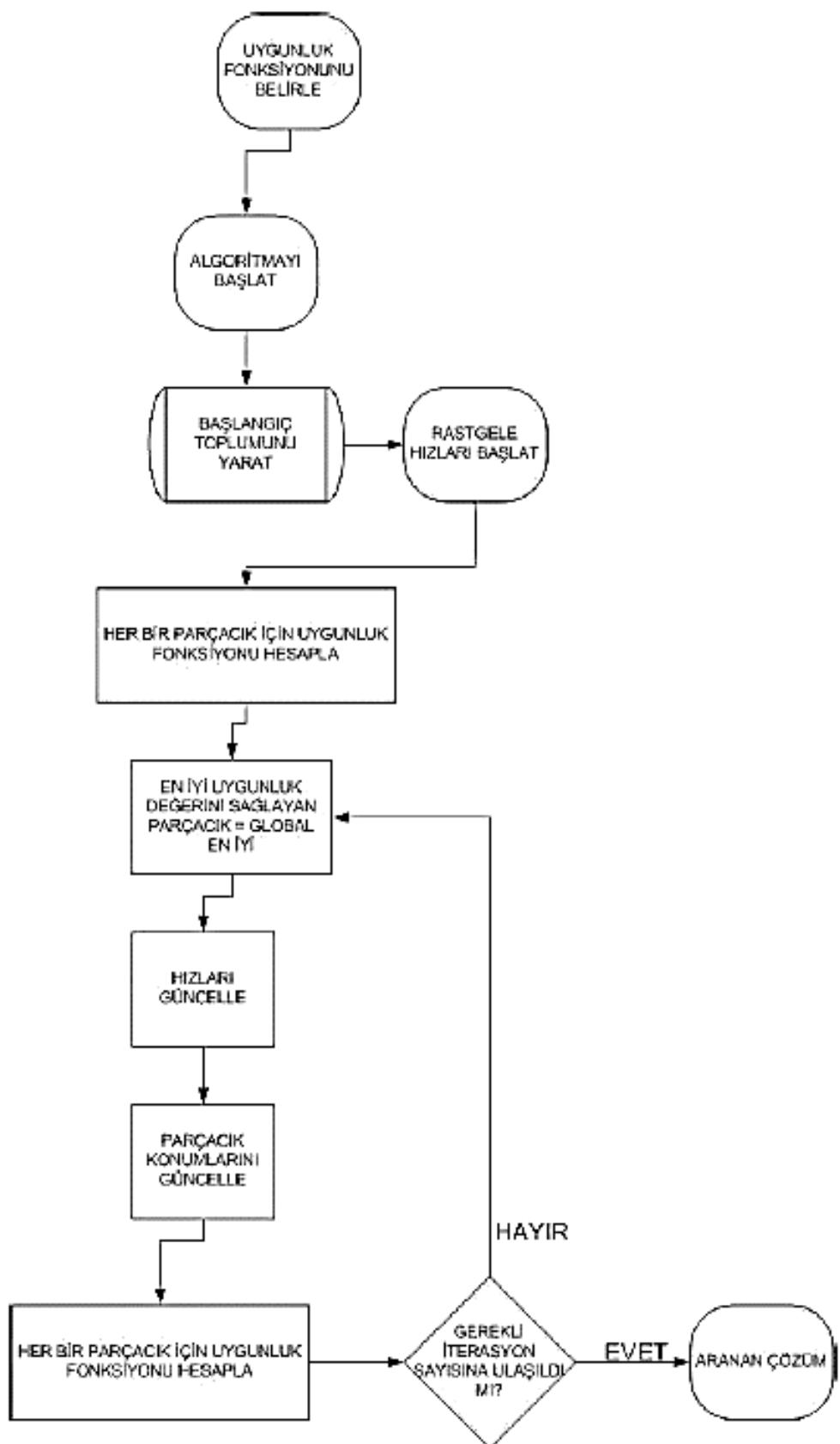
Aşağıda ise parçacık sürü optimizasyon algoritmasının sözde kodu verilmiştir. Dikkat edilirse sözde kod adımları ile akış şeması adımları (adım1,adım2..vb) birbirleri ile örtüşmektedir. Parçacık sürü optimizasyon algoritmasının sözde kodu Şekil 5.2'de görüldüğü gibidir(Akyol,2012).

```

BEGIN
    n Tane Parçacığa Hız ve Konum Değerleri Ata
REPEAT
    FOR i=1 TO n
        Uygunluk Değerini Hesapla;
        Pbest Değerini Güncelle;
        gbest Değerinin Güncelle;
        Hız ve Pozisyon Değerlerini Güncelle;
    END FOR
    UNTIL Sonlandırma Kriteri
END

```

Şekil 5.1 PSO için sözde kod



**Şekil 5.2** Parçacık sürü optimizasyonu akış diyagramı

## **5.4 Parçacık Sürü Optimizasyonu Parametreleri**

Parçacık Sürü Optimizasyonu uygulanırken seçilen parametreler hem algoritmanın işleyişini hem de bazı durumlarda sonuca ulaşma süresini etkilerler. Bu bölümde hangi parametrenin algoritmayı nasıl etkilediği yer almaktadır.

### **5.4.1 Sürünün Büyüklüğü**

Sürünün büyülüklüğü, optimizasyon sırasında seçilen sürüyü oluşturan parçacık sayısıdır. Sürüde daha fazla kuşun yer olması, sürünen başlangıçta çözüm uzayı üzerinde daha fazla konumda uygunluğu araştırdığı anlamına gelmektedir. Benzer şekilde sürü büyülüklüğü ne kadar fazla olursa, her bir iterasyonda o kadar fazla konum araştırılabilir. Ancak sürünen fazla büyük olması daha fazla uygunluk fonksiyonu çözümü gerektirdiğinden hesaplama süresinin artışına sebep olmaktadır. Kısıtlı bir optimizasyon probleminde başlangıç sürüsü, problemin sınır koşulları dahilinde oluşturulmalıdır.

### **5.4.2 İterasyon Sayısı**

PSO'da iterasyon sayısı optimizasyon probleminin boyutuna ve sürünen büyülüklüğüne göre değişmektedir. Çok sayıda iterasyon, hesaplama süresinin çok uzamasına sebep olacaktır.

### **5.4.3 Bilişsel ve Sosyal Bileşenler**

Hız vektörlerinin belirlenmesinde etkili olan bilişsel ve sosyal bileşen parametreleri  $c1$  ve  $c2$ , genellikle optimizasyon boyunca sabit parametreler olarak kabul edilirler. Bilişsel bileşen, sosyal bileşene göre daha büyük seçildiğinde parçacık, yönelmesini kendi kişisel en iyi değerine doğru belirler. Aynı şekilde sosyal bileşen bilişsel bileşene göre daha büyük seçildiğinde ise parçacıklar küresel en iyi değere doğru daha erken hareketlenmeye başlarlar. Bu iki bileşen dengelendiğinde, PSO en verimli ve tasarlandığı haliyle çalışır. Eberhart ve Kennedy'nin önerilerine göre bilişsel bileşenin değeri 1.5 ila 2 arasında değişirken, sosyal bileşenin değeri 2 ila 2.5 arasında olmalıdır.

### **5.4.4 Atalet Bileşeni**

Atalet bileşeni bir parçacığın bir önceki adımdaki hızının, bir sonraki adımda hızını ne oranda etkileyeceğini belirleyen parametredir. Orijinal algoritmada yer almayan bu parametre, Eberhart ve Shi tarafından daha sonra denkleme eklenmiştir. Atalet bileşeni genel olarak 0.9'dan başlayarak 0.4'e doğru iterasyon süresince doğrusal olarak azaltılır. Atalet bileşeninin uygun seçimi, küresel ve yerel keşif ve açığa çıkarma arasında bir denge kurarak istenen sonuca ulaşmada kullanılacak iterasyon sayısının azaltılmasını sağlar.

### **5.4.5 En Yüksek Hız**

PSO'da parçacıkların hızı, bir en yüksek hız ( $V_{max}$ ) değeriyile kısıtlanır. Bu değer bir parçacığın çözüm uzayındaki keşif ve açığa çıkarma yeteneğini kontrol eder. En yüksek hız parametresinin ortaya çıkışının en önemli sebeplerinden biri, PSO'da parçacıkların çözüm uzayı içerisinde kalmasını sağlayan herhangi bir parametrenin olmayışıdır. Eğer

herhangi bir parçacığın hız değeri  $V_{max}$ 'ın üzerine çıkarsa, o parçacığın hızı  $V_{max}$  olarak belirlenir. Ancak bu parametre de bütün parçacıkların çözüm uzayı içerisinde kalmasını sağlamaz.

#### 5.4.6 Kısıtlama Faktörü

Kısıtlama faktörü  $\kappa$ , parçacıkların hızını kısıtlayarak ve kontrol ederek hem parçacıkların çözüm uzayı içerisinde kalmasını hem de algoritmanın performansını artırmayı hedeflemektedir. Kısıtlama faktörünün kullanımı, PSO hız vektörünün Denklem 5.3'teki gibi alternatif gösterimini geliştirmiştir(Akulut,2009).

$$v_n = \kappa \left[ v_n + \varphi_1 rand_1 \times p_{best,n} - x_n + \varphi_2 rand_2 \times g_{best,n} - x_n \right] \quad (5.3)$$

Kısıtlama faktörü olan  $\kappa$ , Denklem 5.4 ve Denklem 5.5 ile belirlenmektedir (Akbulut,2009).

$$\varphi = \varphi_1 + \varphi_2; \quad \varphi > 4 \quad (5.4)$$

$$\kappa = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad (5.5)$$

### 5.5 Uygulama Alanları

Parçacık sürü optimizasyonu günümüzde birçok klasik ve modern mühendislik problemlerine çare bulmaktadır. Bu problemlerden ve uygulamalardan en önemlileri şöyledir:

- Çok Seviyeli Depo Yerleşim Düzenlemesi
- Yapay Sinir Ağları
- İletişimsel Anten Tasarımı
- Damarsal Aritmi Sınıflama Yöntemi
- Sosyal Ağ Analizi
- MR Görüntülenmesinde Tümör Tespiti
- Çevresel Ekonomik Güç Dağıtım Problemlerine Uygulanması

### 5.6 Seçilen Örnek Problem : PSO ile Sensör Yerleştirme Problemi

Bir arazi üzerine maksimum miktarda alan gözetleme amacıyla, belirli sayıda sensörün konuşlandırılması özellikle askeri operasyonlarda oldukça önem taşır. Sensörler radar, termal kamera gibi bir hedefi tespit edebilen ve belirli bir görüş mesafesi olan cihazlardır. Arazinin çok geniş ve engebeli bir yüzeye sahip olması, problemin mümkün olan en az sayıda ve en uygun tarama alanı olan sensör kullanılarak çözülmek istenmesi bu problemin tüm ihtimaller denenerek çözülmesi ile mümkün olmamaktadır.

Bu çalışmada, MATLAB R2017a programı ile statistic toolbox'ı kullanılarak sayısal yükseklik haritaları ile alan modellenmiş ve sensörler modellenmiştir. (Problemin detaylı MATLAB kodu EK C.1'de gösterilmiştir.)

Sensör yerleştirme probleminin çözümü için bir PSO (Parçacık Sürü Optimizasyonu) modeli geliştirilmiştir. Test sonuçlarında PSO algoritması ile tüm arama alanının yüzeysel olarak tarandığı ve iyi noktalarda daha detaylı arama yapıldığı gözlemlenmiştir. Sonuç olarak genelde arazi üzerinde yüksekteki noktalar üzerinde yoğun arama yapıldığı ve çözümün bu noktalardan seçildiği görülmektedir. Dolayısıyla PSO algoritmasının hem global aramayı hem de yerel aramayı yaptığı gözlemlenmiştir.

Gerçek hayatı özellikle askeri operasyonlarda radar, termal kamera, gündüz kamerası vb. sensörlerin ilgili bölgeyi en etkin gözetleyecek şekilde konuşlandırılmaları oldukça önemlidir. Bir araziye belirlenen sayıdaki sensör çeşidinin yükseltiye göre alan taramasını sağlayacak şekilde yerleştirilmesi problemin genel tanımını oluşturur. Bu kısımda arazi, sensör hesaplarından bahsedilecek ve problem bir optimizasyon problemi olarak ifade edilecektir.

### 5.6.1 Arazi

Sensör yerleştirme problemi için önerilen bir çözümün gerçekçi sonuçlar verebilmesi için, arazi verilerinin de gerçeğe yakın modellenmesi gerekmektedir. Örnek olarak seçilen bir arazinin piksel değerleri üzerinden belirlenen noktalar bir matris içerisinde tutulmuştur ve okunan yükseklik verileri ile işlemler yapılmıştır.

### 5.6.2 Sensör

Sensörler, genel olarak belirli bir algılama kapasitesi olan ve belirli bir menzil dahilinde bilgi toplayabilen cihazlardır. Problem çözümünde kullanılacak sensör modeli problemin gereksinimlerine göre farklılık gösterebilir (Termal, Elektromanyetik, Mekanik, Kimyasal, Akustik, Optik vb.). Bu çalışmada belirli bir menzile sahip, menzili içerisinde bulunan cisimleri çizgisel olarak tarayabilen genel bir sensör modeli kullanılmıştır. Bir sensörün menzilinde bulunan herhangi bir nokta çizgisel tarama alanı içerisinde ise bu noktanın sensör tarafından görüldüğü kabul edilir. Yani sensör menzili içerisinde bir cismin görülebilme ihtimali tüm uzaklıklar için aynıdır. Bu çalışmada ise iki adet sensör tipi belirlenmiştir. Bunlardan biri az yarıçaplı tarama sağlayan sensör çeşidi, diğer ise çok yarıçaplı tarama sağlayan sensör çeşididir.

### 5.6.3 Problemin Formüle Edilmesi

Problem formüle edilmesinde yine hep kullandığımız parçacık sürü algoritmasının ana fonksiyonundan yararlanılmıştır.

$$v_{n,i} = \omega v_{n,i-1} + c_1 rand_1 \ gbest_i - x_{n,i} + c_2 rand_2 \ pbest_{n,i} - x_{n,i} \quad (5.6)$$

$$x_{n,i+1} = x_{n,i} + v_{n,i} \quad (5.7)$$

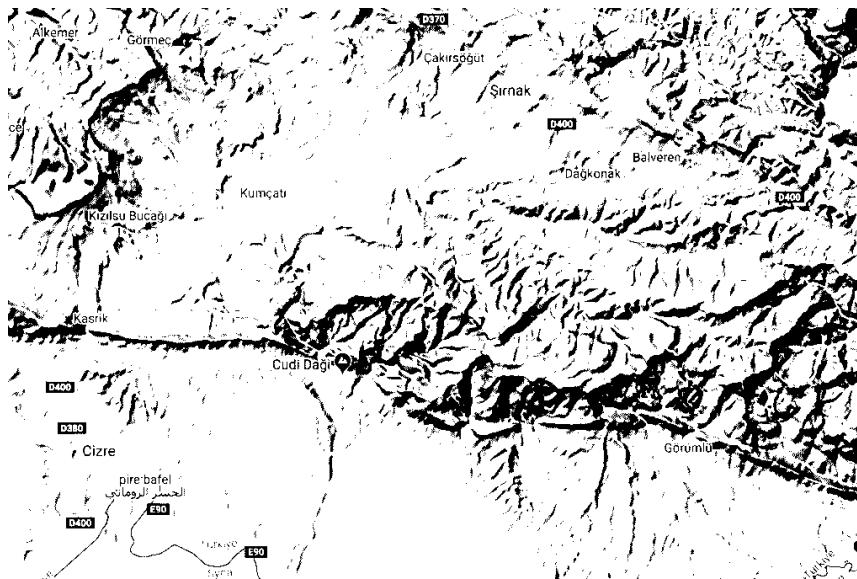
Ayrıca problemin amaç fonksiyonu ise denklem 4.3 de gösterilmiştir. Bu formüle göre Z değeri tanımlı alanı , N değeri ise taranabilecek alanı göstermektedir.

$$Z = \max \frac{\left( \sum_i 0.2N_i A_i \right)}{T} \times 100 \quad (5.8)$$

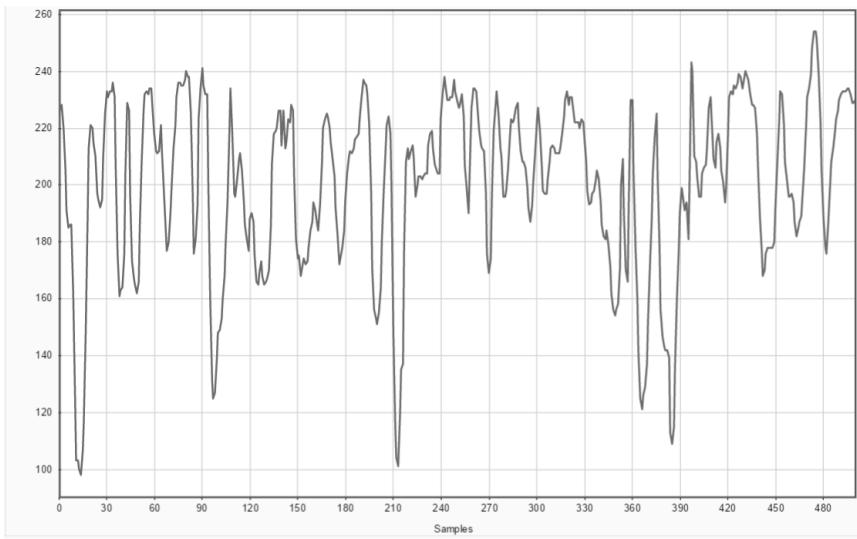
#### 5.6.4 Problemin İşleyişi

Bu projenin temel amacı kuş hareketlerinden esinlenerek sensorlerin alan taraması yapmasıdır.

Öncelikle Şekil 5.3' de görüldüğü üzere elimizdeki siyah – beyaz haldeki arazi resminin piksel değerlerini bulunur ve şekil 5.4' teki gibi grafiğe dökülür.

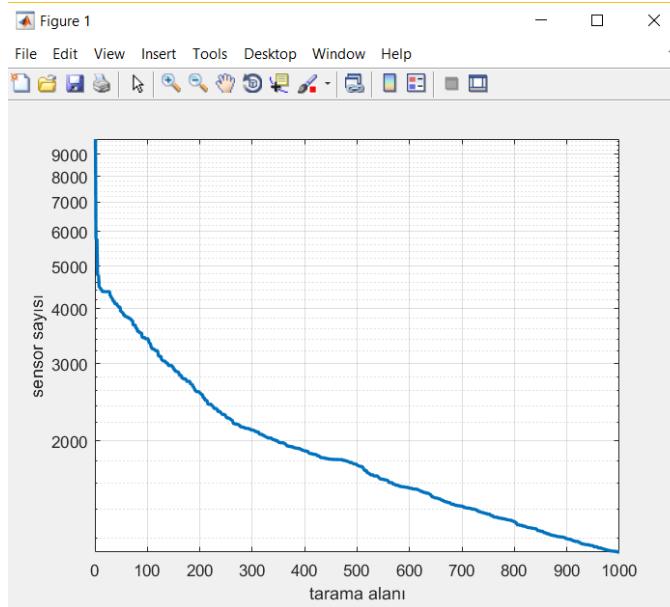


Şekil 5.3 Seçilen arazinin görünümü

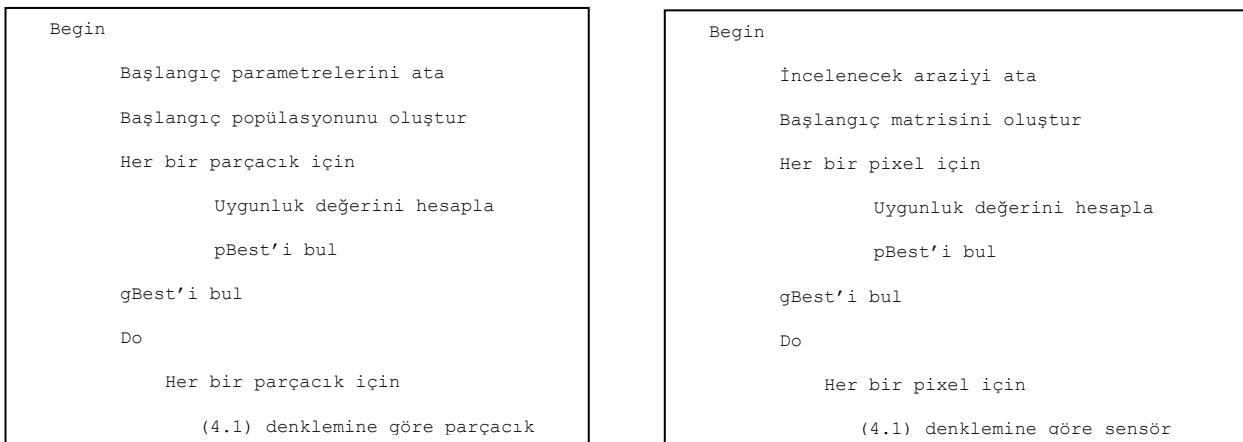


Şekil 5.4 Seçilen arazinin grafiğe dökümü

Daha sonrasında PSO ile birlikte resim matrisi belirli işlemlerden geçirilir ve sonuç olarak aşağıdaki şekil 5.5 ‘teki grafik oluşturulur. Bu sonuca göre sensörler yüksek rakımlı kısımlarda daha az tarama (dk. başına) yaparken, rakımı düşük olan kısımlarda daha fazla tarama yapmaktadır. Daha fazla bilgi için Bölüm 5.6.5 e bakınız.



**Şekil 5.5** Sensörlerin tarama grafiği



**Şekil 5.6** PSO ve Sensör Yerleştirme Problemi sözde kodları

### 5.6.5 Problemin Parçacık Sürü Optimizasyonuna Uyarlanması

Bu kısımda, problemin PSO algoritmasına nasıl uyarıldığı anlatılmaktadır. İlk olarak problemin çözüm kümesinin nasıl temsil edileceği anlatılmış ve ardından uygulanan PSO algoritması ve parametreleri gösterilmiştir. Problemin gösterim biçiminin tanımlanması sonrasında PSO algoritmasını uygulanması için başlangıç popülasyonunun belirlenmesi ve algoritmanın parametrelerinin atanması gerekmektedir. Algoritma başlangıç popülasyonu

olarak her bir parçacık için sensörlere alan içerisinde belirlenen arazi resmindeki koordinatları atayarak başlar. Parçacıkların başlangıç hızları [0,1] aralığında rasgele sayı değerleridir. Her bir iterasyonda sensörlerin koordinat değerleri (5.6) ve (5.7) numaralı denklemler kullanarak güncellenir ve böylece çözüm uzayında yeni yerler denenerek küresel en iyi değer araştırılır. PSO algoritmasında atalet sabitinin zamanla düşürülmesi ile daha iyi sonuçlar verdiği gözlemlenmiştir.  $w$  değerinin  $(t+1)$ 'inci iterasyondaki güncellemesi aşağıdaki formüle göre yapılmaktadır.

$$w_{t+1} = w_t \times 0.95 \quad (5.9)$$

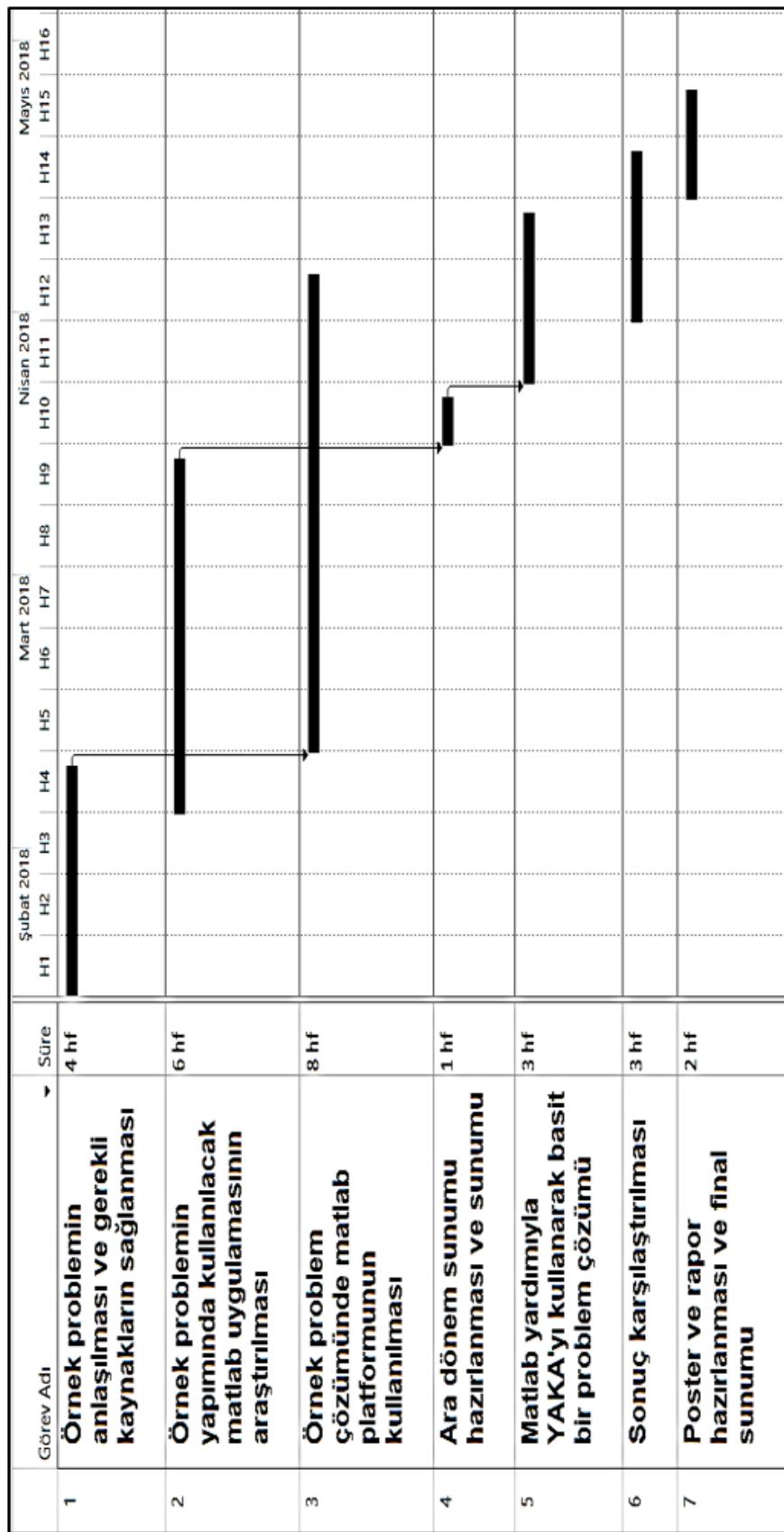
Başlangıçta  $w$  değeri 1 olarak seçilir. Durdurma kriteri olarak belirli sayıda etkinlik fonksiyonu çağrılmaması sonucunda algoritma sonlandırılır.

### 5.7 Sonuç

Bu çalışmada sensör yerleştirme problemini çözmek için bir Parçacık Sürü Optimizasyonu algoritması önerilmiş ve bu yaklaşımın performansı ve sonuçları incelenmiştir. Sonuç olarak hangi yüksekliğe nasıl bir sensör yerleştirilmesi gerektiği belirlenmiştir. Bu çalışma benzer çalışmalar için bir temel oluşturmaktır ve performans metrikleri açısından daha sonraki çalışmalar için bir altyapı sunmaktadır.

## 5.8 İş-Zaman Diyagramı

### Kuşlar ile Süre Optimizasyonu 2017-2018 Akademik Yılı Bahar Dönemi Bitirme Projesi İş-Zaman Pları



## 5.9 Kaynakça

- Erdoğmuş, P. "Doğadan Esinlenen Optimizasyon Algoritmaları ve Optimizasyon Algoritmalarının Optimizasyonu, Düzce Üniversitesi Bilim ve Teknoloji Dergisi 4", Düzce Üniversitesi ss. 293-304, (2016).
- Hahn, B. ve Daniel V. "Temel MATLAB (5. Baskı)" Nobel Akademik Yayıncılık ,(2016).
- Tozan, A., F. E. Sevilgen ve O.İnce "Sensör Yerleştirme Probleminin Parçacık Sürü Optimizasyonu ile Çözümü", Bilgisayar Mühendisliği Bölümü, Gebze Yüksek Teknoloji Enstitüsü, Kocaeli, (2014).
- Özsäglam M. Y. ve M. Çunkaş "Optimizasyon Problemlerinin Çözümü için Parçacık Sürü Optimizasyonu Algoritması, Politeknik Dergisi 11", Gazi Üniversitesi, ss. 299-305, (2008).
- Dhillon, S. S. Ve K. Chakrabarty "Sensor placement for effective coverage and surveillance in distributed sensor networks", Proc. IEEE Wireless Communications and Networking Conference, pp. 1609--1614,(2003).
- Can, T. ve V. İşler "Sensor Optimization In a Virtual Environment", in Proceedings of the 9th Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, May, (2000).
- Dhillon, S. S. ve K. Chakrabarty "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", IEEE ,(2003).
- Undeğer, C. ve Z. İpekkan "Sensor Platform Optimization and Simulation for Surveillance of Large Scale Terrains", Proceedings of I'ITSEC 2002 Conference on Modeling and Simulation, Orlando, Florida, (2002).
- National Imagery and Mapping Agency,Digital Terrain Elevation Data (DTED). Standards and Specifications Publications: MIL-PRF-89020A Amendment-1, 27, (1999).
- Marengoni, M.ve R. Sitaraman "Placing Observers to Cover a Polyhedral Terrain in Polynomial Time (sec.)", Proceedings of the Third Workshop on Applications of Computer Vision, (1996).
- Eberhard, R.C. ve J. Kennedy "New optimizer using Particle Swarm Theory", Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, (1995).
- Bresenham, J. E., "Algorithm for computer control of a digital plotter", IBM Systems J., 4(1), ss.25-30(1965)

## VI KURTLAR VE SÜRÜ OPTİMİZASYONU

Sürü optimizasyonunun ilgilendiği hayvan grupları genellikle çok bireyli hayvan sürüleridir. Bu sürüler besin zincirinin en tepesinde bulunmamakla birlikte aynı zamanda çeşitli hiyerarşik yapılara sahiptirler. Kurt sürüleri ise diğer hayvan sürülerine göre az bireye sahiptir ve besin zincirinin en tepesindedir. Kurt sürülerini bu özelliklerinden dolayı konu alan algoritmalar diğerlerine göre yeni olmak ile birlikte sayıları da azdır.

### 6.1 Kurt Sürüleri

Kurtlar, köpekgiller ailesine ait olan canlılardır. Besin zincirinin en tepesindedirler, yani bulundukları ekosistemlerde kurtları avlayabilen canlılar bulunmamaktadır. Kurtlar sürü halinde yaşamak zorunda değildirler, ama sürü halinde yaşamanın sağladığı avantajlar nedeniyle ortalama 5-12 bireyden oluşan sürüler ile yaşamayı tercih ederler.

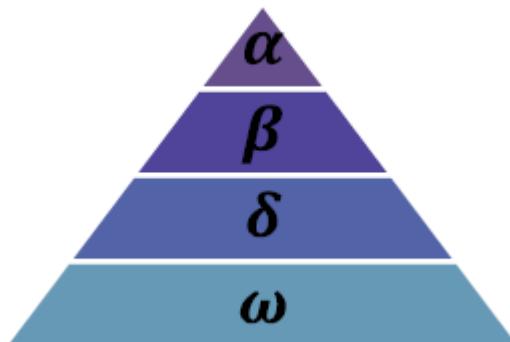
Bu sürüler çok katı bir hiyerarşije sahiptir, ve bu hiyerarşik yapıda her bireyin bir görevi vardır. Bu hiyerarşik yapı Şekil 6.1'de gösterilmiştir (Mirjalili, 2014).

Kurt sürülerinin liderleri bir erkek ve bir dişi kurttur. Bu iki kurt hiyerarşik yapıda “alfa kurt” isimi verilmiştir. Alfa kurtlar avlanma, dinlenme zamanı ve yeri hakkında kararlar almak ile sorumludur. Bu kararlar sürüün geri kalanına dikte edilir. Ancak sürü hakkında kararlar alınırken bazen demokratik özellikler gözlemlenebilir.

Sürü içinde çifteleşebilen tek hiyerarşik tabaka alfa kurtlardır, ancak bu sürü içerisinde en güçlü kurtların alfa kurtlar olduğu anlamına gelmez. Alfa kurtlar, sürüyü en iyi Şekilde yöneten kurtlar oldukları için hiyerarşik yapıda alfa kısmındadırlar. Bu özellik, kurt sürülerinde örgütlenme ve disiplinin fiziksel güçden daha değerli olduğunu göstermektedir.

Hiyerarşik yapıda ikinci kısımda yer alan kurtlara “beta kurtlar” ismi verilmiştir. Beta kurtlar alfa kurtlara karar alma ve çeşitli sürü görevlerinde destek olmak ile sorumludur. Beta kurtlar erkek veya dişi olabilir ve alfa kurtların görevlerine devam edemedikleri bir durumda, alfa kurtların yerine geçerler. Bundan dolayı sürüün geri kalani beta kurtların emirlerine uymak zorundadır. Beta kurtlar alfa kurtlara danışmanlık yapmak dışında sürüün genel disiplinini korumak ile ve aynı zamanda alpha kurtun emirlerinin gerçekleştiğinden emin olmak ile sorumludur.

Beta kurtların bir alt katmanında bulunan hiyerarşik katmandaki kurtlara “omega kurt” isimi verilmiştir. Omega kurtlar hiyerarşik yapıda düşük bir rütbeye sahip olmalarına rağmen sürüün devamlılığı için en önemli kurtlardır. Omega kurtlarını kaybeden kurt sürülerinin hiyerarşik sorunlar yaşadığı gözlemlenmiştir, bunun nedeni omega kurtların sürü içerisinde diğer bütün kurtların saldırgan davranışlarının hedefi olmalarıdır. Omega kurtlar diğer kurtların baskınlıklarını onaylayarak onları tatmin etmek ile görevlidirler bu görevi yerine getirerek de dolaylı olarak sürüün hiyerarşik yapısını korumak ile görevlidirler.



Şekil 6.1 Hiyerarşik yapı

Bu görevlerinin dışında omega kurtlar yaşılı ve yavru sürü üyelerine bakıcılık yaparlar, ve sürü yemek yerken diğer bütün kurtlardan sonra beslenebilirler.

Alfa beta ve omega kurt olmayan kurtlara “ast kurt” veya “delta kurt” isimi verilmiştir. Delta kurtlar alfa ve beta kurtların astıdır ancak omega kurtların üstüdür. Delta kurtlar sürü içerisinde çeşitli görevli kurtlar veya yaşılı kurtlar olabilirler, izcilik, nöbetçilik, avcılık veya bakıcılık görevi ile sorumlu kurtlar omega kurt olabilir. İzcilik ile sorumlu olan kurtlar sürüünün bölgesini gözlemlerek ve sürüyü tehlike durumunda uyarmak ile sorumludur. Nöbetçi kurtlar ise sürüyü korumak ile görevlidir. Yaşılı kurtlar eskiden alfa veya beta olan ve artık görevlerini yerine getiremeyen kurtlardır. Avcı kurtlar, av sırasında alfa ve beta kurtlara destek olmak ile ve av sonunda besinin sürüye dağıtımları ile sorumludur. Son olarak, bakıcı kurtlar hasta ve yaralı kurtlara bakıcılık yapmak ile sorumludur.

Bu hiyerarşik yapı kurt sürüsünün devamlılığı için büyük önem taşımaktadır, ve bu devamlılığı mümkün kılmak yapılan avlanmanın başarılı olmasına bağlıdır.

Kurtlar sürü halinde avlanırken belirli adımlar izler, bunlar;

- Avlanılan hayvanın izinin sürülmesi ve takip edilmesi.
- Avlanılan hayvanın etrafının sarılması ve hareket ettiği sürece takip edilmesi.
- Hareketini durduran hayvana saldırılması.

Bu hareketler Şekil 6.2’de fotoğraflar ile adım adım gösterilmiştir (Mirjalili, 2014).



**Şekil 6.2 Avlanma örneği**

## 6.2 Kurt Sürülerini Konu Alan Sürü Optimizasyonu Algoritmaları

Kurtları konu alan ilk sürü optimizasyonu algoritması, parçacık sürü optimizasyonu algoritması ile kurtların davranışlarını harmanlayarak ortaya Wei. B, Q. Peng ve X. Chen tarafından 2012de çıkarılmış “PSO Wolves” isimli bir optimizasyon algoritmasıdır.

Kurtları tek başına konu alan ilk algoritma ise Mirjalili S., S.M. Mirjalili ve A. Lewis tarafından 2014’de ortaya çıkarılmış “Gray wolf optimizer” isimli algoritmadır. Bu algoritma kurtları konu alan en başarılı algoritmadır. Günümüzde ortaya çıkarılan kurtlar ile ilgili çoğu optimizasyon algoritması da bu algoritmadan esinlenerek veya çalışma şeklini değiştirerek ortaya çıkartılmıştır.

Örnek olarak; Emarya E., W. Yamanyb, A.E. Hassanien ve V. Snasel tarafından 2015de ortaya çıkarılan “Multi objective grey wolf optimization” ve Kohli M. Ve S Arora tarafından 2016da ortaya çıkarılmış “Chaotic Grey wolf optimization” verilebilir.

Bu algoritmalar klasik mühendislik problemlerinin çözümünde ve bir çok farklı alanda problemlerin çözümünde kullanılmıştır, bunlardan bazıları aşağıda yayın yılına göre sıralanmıştır;

- Optik tampon tasarımı (2014)
- Görüntü segmentasyonu (2017)
- Makine arıza tehsisi (2017)

### 6.3 Grey Wolf Optimization Algorithm(GWO)

Grey wolf optimizer (GWO) kurtların hiyerarşik yapısı ve avlanma mekanizmalarından esinlenerek ortaya çıkarılmış bir algoritmadır. Bu algoritmda alfa beta delta olmak üzere dört farklı kurt temsil edilmiştir ve avlanma mekanizmaları avın aranması, etrafının sarılması ve avlanılan hayvana saldırılması olmak üzere üç adıma ayrılmıştır. Bu hiyerarşik yapı ve avlanma mekanizmaları matematiksel olarak temsil edilmiştir. Hiyerarşik yapı çözümlerin sıralanmasında kullanılırken avlanma mekanizmaları ise aramanın işleyişini göstermektedir.

#### 6.3.1 Çalışma Prensipleri

Bu algoritma kurtların avlanma davranışlarını iki aşamada işlemektedir, bunlar; keşif(exploration) ve yararlanma(exploitation). Bu iki aşamada kurtların hareketleri farklıdır ve bu fark kurtların yön vektörlerine verilen parametrelerin değişimi ile temsil edilir.

GWO çalışırken her kurtun verilen zamanda bulunduğu noktadan gideceği yön bir vektör ile belirlenir. Bu yön vektörü belirleyen denklem avın her aşamasında farklıdır. Avın etrafı sarılırken denklem 6.1,6.2,6.3 ve 6.4'den yararlanılır (Mirjalili, 2014).

$$\vec{D} = \left| \vec{C} \cdot \overrightarrow{X(t)_P} - \overrightarrow{X(t)} \right| \quad (6.1)$$

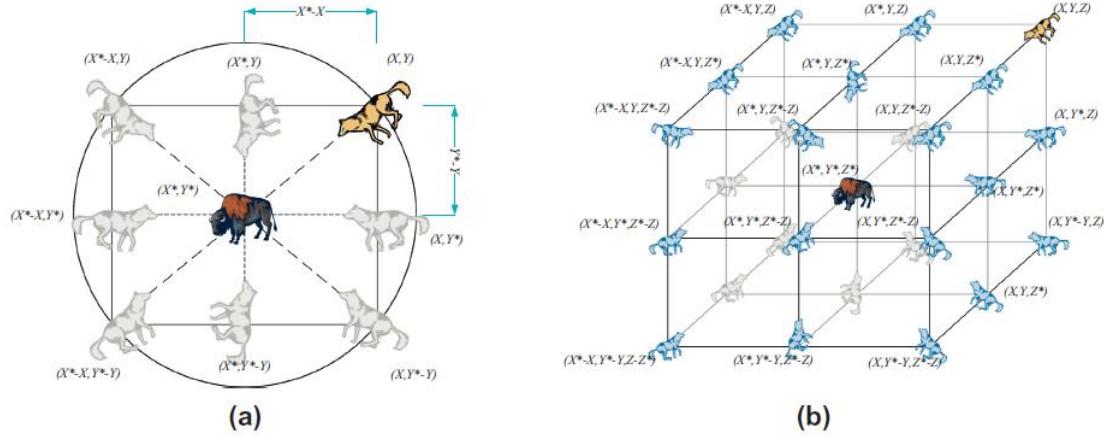
$$\overrightarrow{X(t+1)} = \overrightarrow{X(t)_P} - \vec{A} \cdot \vec{D} \quad (6.2)$$

Denklem 6.1 ve 6.2de t, verilen zamandaki iterasyonu temsil etmektedir.  $\vec{A}$  ve  $\vec{C}$  ise katsayı vektörleridir  $\overrightarrow{X(t)_P}$  avlanılan hayvanın t zamanındaki pozisyonudur.  $\overrightarrow{X(t)}$  ise t zamanında verilen kurtun pozisyonunu temsil etmektedir.  $\vec{A}$  ve  $\vec{C}$  vektörlerinin değerleri ise denklem 6.3 ve 6.4 ile hesaplanmaktadır.

$$\vec{A} = 2\vec{a} \cdot \vec{r1} - \vec{a} \quad (6.3)$$

$$\vec{C} = 2 \cdot \vec{r2} \quad (6.4)$$

$\vec{a}$  değeri doğrusal olarak 2 değerinden 0 değerine tekrarlar ile düşmektedir.  $\vec{r1}$  ve  $\vec{r2}$  değerleri 0 ve 1 arasında değişen rasgele değerlere sahip olan vektörlerdir. Bu vektörlerin aldığı değerler ile bir kurtun sonraki konumu belirlenmektedir, bu durum Şekil 6.3 (a) ve Şekil 6.3 (b) ile gözlemlenebilir.



**Şekil 6.3** Kurtların sonraki pozisyonlarının 2 ve 3 boyutlu ortamda belirlenmesi

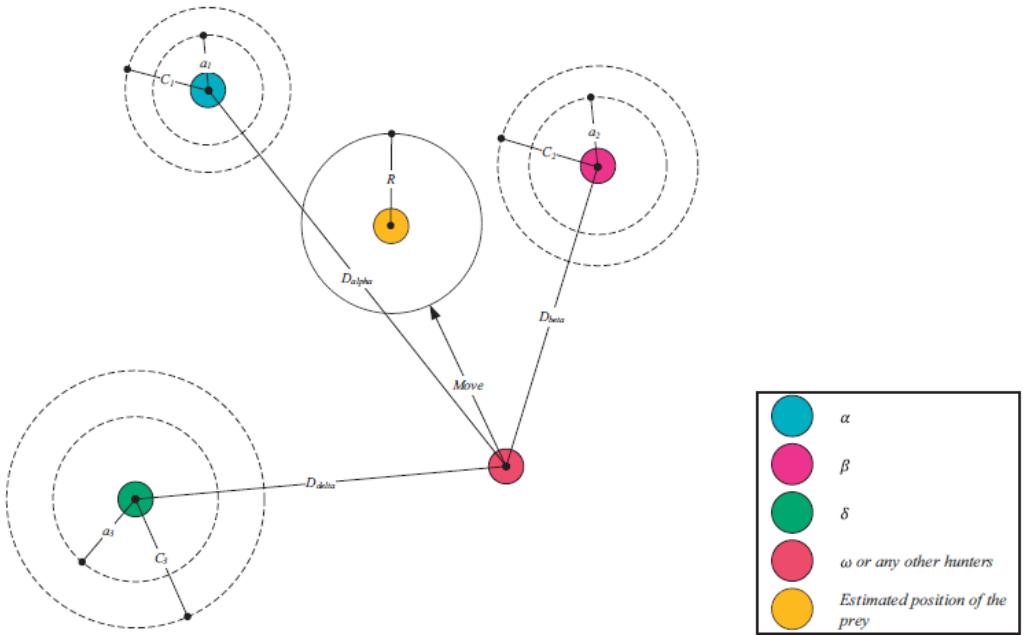
Kurtlar avlanırken avladıkları hayvanın bulunduğu pozisyonu görüp, hayvanın etrafını sarabilme yeteneğine sahiptirler. Avlanma davranışları genellikle alfa kurt tarafından yönetilir, beta ve delta kurtlar da avlara katılabılır. Ancak bir arama uzayında avlanılan hayvanın bulunduğu nokta (optimum) bilinmez. Matematiksel olarak avlanma davranışını temsil edebilmek için alfa (en iyi aday çözüm), beta ve delta kurtun avladıkları hayvanın bulunduğu nokta konusunda daha bilgili olduklarını kabul ederiz. Bu nedenden dolayı en iyi üç çözümü (kurt) kayıt edip diğer kurtların pozisyonlarını bu kurtlara (çözümlere) göre değiştiririz. Denklem 6.5, 6.6 ve 6.7 bu davranışları temsil etmektedir (Mirjalili, 2014).

$$\overrightarrow{D_\alpha} = |\overrightarrow{C1} \cdot \overrightarrow{X_\alpha - X}|, \overrightarrow{D_\beta} = |\overrightarrow{C2} \cdot \overrightarrow{X_\beta - X}|, \overrightarrow{D_\delta} = |\overrightarrow{C3} \cdot \overrightarrow{X_\delta - X}| \quad (6.5)$$

$$\overrightarrow{X_1} = |\overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot (\overrightarrow{D_\alpha})|, \overrightarrow{X_2} = |\overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot (\overrightarrow{D_\beta})|, \overrightarrow{X_3} = |\overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot (\overrightarrow{D_\delta})| \quad (6.6)$$

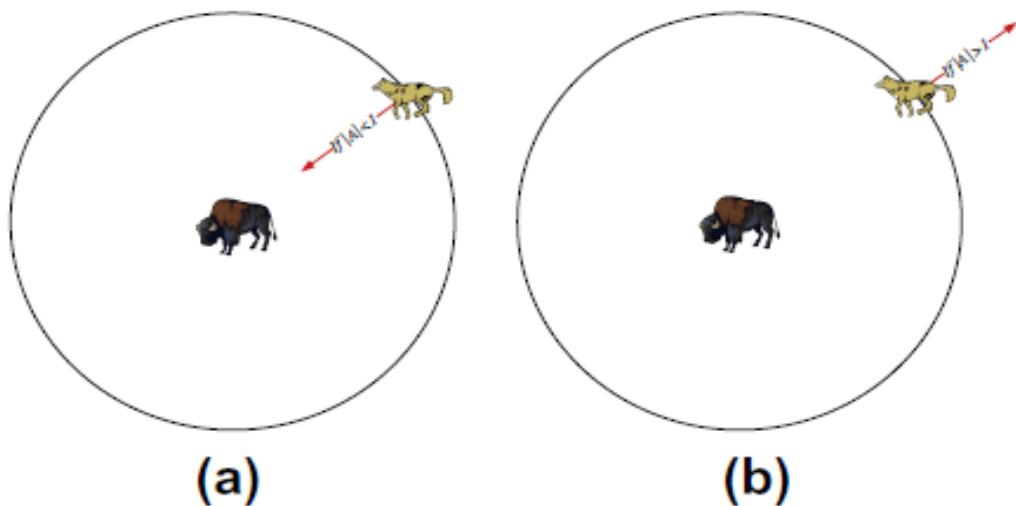
$$\overrightarrow{X(t+1)} = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (6.7)$$

Bu denklemlerde bulunan parametrelerin anlamı 6.1, 6.2, 6.3 ve 6.4'de bulunan parametreler ile aynıdır. Bu denklemlerin çalışma şekli, Şekil 6.4 ile gözlemlenebilir (Mirjalili, 2014).



**Şekil 6.4:** Kurtların pozisyon güncelleme mekanizması

Kurtlar avlanma etkinliklerini avladıkları hayvana saldırarak sonlandırırlar, bu davranış matematiksel olarak ifade etmek için  $\vec{\alpha}$  değeri düşürülür.  $\vec{\alpha}$  değerinin değişimi de  $\vec{\alpha}$  değeri tarafından azaltılır.  $\vec{\alpha}$  değeri aslında  $[-2a, 2a]$  aralığındadır ( $a$  değeri tekrarlar ile 2'den 0'a düşürülür).  $\vec{\alpha}$  değeri  $[-1, 1]$  değer aralığında bulunurken kurtların sonraki pozisyonu herhangi bir nokta olabilir. Ancak  $|A|<1$  ise kurtlar direkt olarak avladıklara hayvana saldırımaktadır. Bu durum Şekil 6.5 (a) ve (b) de gözlemlenilebilir (Mirjalili, 2014).



**Şekil 6.5** Kurtların avlarına saldırma durumları

Kurtlar avlanırken alfa, beta ve delta kurtların pozisyonlarına göre aramalarını gerçekleştirirler. Aşları ararken kurtlar birbirilerinden uzaklaşır ve aşlarına saldırırken birbirilerine yaklaşırlar. Bu davranışları matematiksel olarak ifade etmek için  $\vec{A}$  değeri 1 ile -1 arasında değiştirilir. Bu değişim keşif aşamasını vurgular ve global bir arama yapmasını sağlar.

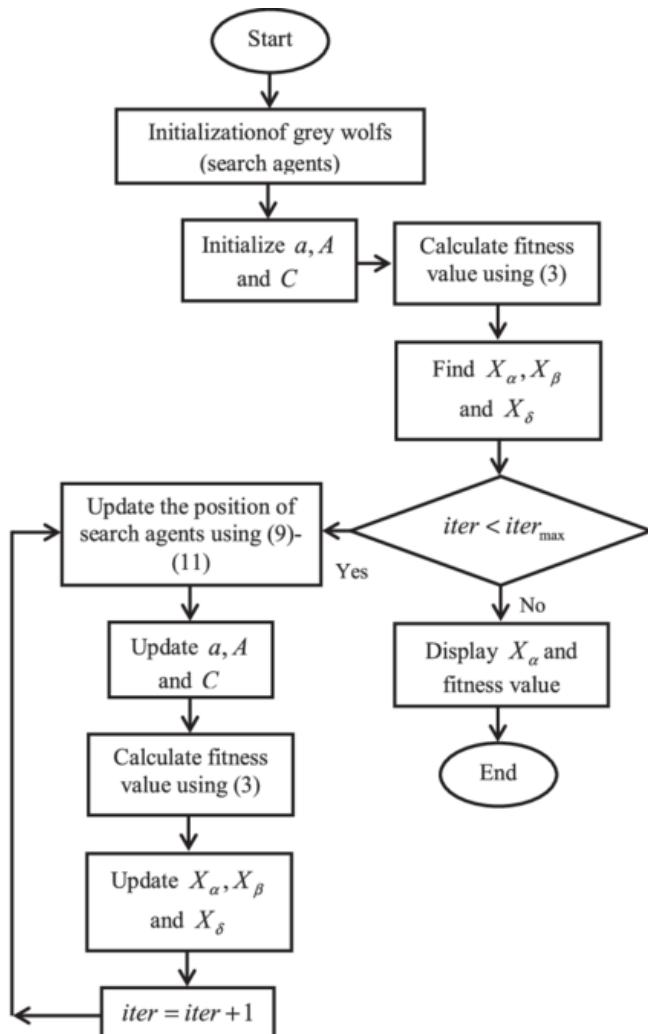
Keşif aşamasının daha etkili çalışmasını sağlayan değerlerden biri de  $\vec{C}$ dir. Değerleri 0 ve 2 arasında rastgele değiştirilerek, GWO'nun lokal optimumdan kaçınmasını mümkün kılmaktadır. Ancak  $\vec{C}$  değeri  $\vec{A}$  değeri gibi doğrusal azalmamaktadır ve bu da keşifin aşamasının daha verimli olmasını sağlamaktadır.

GWO'nun çalışmalarını kısa özetini Şekil 6.6'da verilen sözde kodda görebiliriz (Mirjalili, 2014).

```
Initialize the grey wolf population  $X_i$  ( $i = 1, 2, \dots, n$ )
Initialize  $a$ ,  $A$ , and  $C$ 
Calculate the fitness of each search agent
 $X_\alpha$ =the best search agent
 $X_\beta$ =the second best search agent
 $X_\delta$ =the third best search agent
while ( $t < \text{Max number of iterations}$ )
    for each search agent
        Update the position of the current search agent by equation (3.7)
    end for
    Update  $a$ ,  $A$ , and  $C$ 
    Calculate the fitness of all search agents
    Update  $X_\alpha$ ,  $X_\beta$ , and  $X_\delta$ 
     $t=t+1$ 
end while
return  $X_\alpha$ 
```

**Şekil 6.6** GWO için sözde kod

Aynı çalışma prensipleri Şekil 6.7'de verilen akış diyagramında daha basitleştirilmiş bir şekilde gözlemlenebilir (Mirjalili, 2014).



Şekil 6.7 Akış diyagramı

#### 6.4 Uygulama Alanları

GWO kurtlarla ilgili diğer optimizasyon algoritması gibi bir çok alanda uygulanmıştır, Algoritmayı ilk ortaya atan kişiler tarafından optik tampon tasarımlında kullanılmıştır.

Bu alanlardan biri savunma sanayii şirketlerini için büyük önem taşıyan insansız hava araçlarıdır. Sen Zhang, Yongquan Zhou, Zhiming Li ve Wei Pan tarafından ortaya çıkarılan GWO insansız hava aracı yol planlama algoritması bu alanda GWO algoritmasının başarılı bir uygulamasıdır. Bu algoritma GWO algoritmasını kullanarak bir insansız hava aracının A noktasından B noktasına mümkün olduğunda az yakıt harcayan ve tehlikeli bölgelere girmeyen bir rota planlamaktadır. Bu algoritma 2016da “advances in engineering software” dergisinde yayımlanmıştır ve testlerde başarı göstermiştir.

GWOnun bu alan dışında da bir çok uygulaması bulunmaktadır, bunlardan bazıları yayın yılına göre sıralanmıştır;

- Hyper spectral band selection (2015)
- Image segmentation (2017)
- Design of hybrid renewable energy system (2017)

## **6.5 Seçilen Örnek Problem**

Bu projede incelenenek olan örnek problem ise savunma sanayii şirketlerini için büyük önem taşıyan insansız hava araçları ile ilgilidir. Sen Zhang, Yongquan Zhou, Zhiming Li ve Wei Pan tarafından ortaya çıkarılan GWO insansız hava aracı yol planlama algoritması bu alanda GWO algoritmasının başarılı bir uygulamasıdır. Bu algoritma GWO algoritmasını kullanarak bir insansız hava aracının A noktasından B noktasına mümkün olduğunda az yakıt harcayan ve tehlikeli bölgelere girmeyen bir rota planlamaktadır. Bu algoritma 2016'da "Advances in Engineering Software" dergisinde yayımlanmıştır ve testlerde başarı göstermiştir.

## **6.6 İHA Yol Bulma Problemi**

İnsansız hava araçları, içinde pilot taşımadan kontrol edilebilen uçaklardır. Pilot taşımadan kontrol edilebilmeleri, boyut olarak küçük olmalarına ve pilotları riske atmadan tehlikeli görevler için ideal bir aday olmalarına neden olmuştur. Günümüzde bazı ülkelerin insansız hava araçları yarı otonom çalışma özelliğine sahiptir (General Atomics MQ-9 Reaper, A.B.D.); yarı otonom bir insansız hava aracı kendisine gönderilen emirleri yerine getirebilir ve emir verilmediğinde ise emir verilene kadar kendini havada tutma özelliğine sahiptir. İnsansız hava araçları bir pilotun verdiği emirlere bağlı olduğu için aynı zamanda pilotun yapabileceği hatalardan da zarar görebilir, bu nedenden dolayı tam otonom insansız hava araçları için bazı ülkeler (A.B.D, Çin Halk Cumhuriyeti, Rus Federasyonu) çalışmalara başlamıştır, otonom insansız hava araçları için çalışmalar devam etmektedir ancak bu çalışmalar basit prototipler dışında sonuç verememiştir.

Bir insansız hava aracı için yol planlama problemi, otonom ve yarı otonom insansız hava araçlarının gelişimi için bir engel olmuştur. Bu problem, bir insansız hava aracının A noktasından B noktasına minimum yakıt kullanarak ve tehlikeli alanlarda minimum zaman geçirerek nasıl gideceğini belirlemeyi amaçlar. Yani bu problemed optimum yol minimum tehlike ve minimum yakıt maliyetine sahiptir.

Bu problem bir çok farklı algoritma ile çözümlenmiştir, bu algoritmaların biri Mirjalili S., S.M. Mirjalili ve A. Lewis tarafından 2014de ortaya çıkarılmış "Gray Wolf Optimizer" isimli algoritmadır. Bu çalışmanın amacı problemi bu algoritma ile MATLAB R2017a platformunda ana modül (display, subplot, sqrt, rand, polyfit) ve statistics toolbox (plot) isimli modülü kullanılarak çözümlenmiştir. (Problemin detaylı MATLAB kodu Bölüm 8.4'de gösterilmiştir.)

## **6.7 GWO ile İHA Yol Bulma Probleminin çözümü**

GWO ile İHA yol bulma problemi bir çok farklı şekilde ve farklı başarı dereceleri ile çözümlenmiştir. Bu projenin amacı bu çözümlerden yararlanarak yeni bir çözüm oluşturmaktır.

Bu çalışmada Zhang S., Y. Zhou, Z. Li, W. Pan tarafından geliştirilmiş algoritmadan yararlanılmıştır (Zhang, 2016), geliştirilmiş bu algoritmda koordinat sisteminin üzerinde işlemlerin hızının yüksek tutulabilmesi için koordinat sistemi denklem 6.8'e göre kaydırılır. Bu

işlem algoritmanın çalışma süresini uzatmaktadır. Bu nedenden dolayı bu çalışmada kullanılmamıştır.

$$\theta = \arcsin \frac{y_2 - y_1}{|\vec{AB}|}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_1 \\ y_2 \end{bmatrix} \quad (6.8)$$

Bu çözümde verilen tehlike bilgilerine göre istenilen çözünürlüğe göre bir yol planlanabilir. Verilen bütün arazi bilgileri bir x,y düzleminde temsil edilmiştir. Verilen çözünürlük bilgisi ise bu x,y düzleminde her bir noktanın arasının uzaklığını belirlemektedir, örneğin 1x1 kilometrelik bir alan için 100x100 noktalı bir çözünürlük seçilirse iki noktanın arası 10 metreyi temsil etmektedir. Çözünürlük değerinin bu algoritmada bir parametre olarak verilmesinin nedeni ise çözünürlük yükseldikçe çözüm süresinin de doğal olarak yükselmesidir, ve farklı arazi boyutlarının da incelenmesini mümkün kılmaktır. Algoritma çalışmaya başlamadan önce başlangıç ve hedef noktasının, tehlike noktalarının ve tehlikelerden ne kadar uzak durulması gereği, iterasyon ve kurt sayısının belirtilmesi gerekmektedir. Bu bilgiler verildikten sonra algoritma çalışabilir. Algoritma çalışırken İhanın takip edebileceği farklı yollar uygunluklarına göre sıralanır ve o sıralamadaki en uygun yollara göre yeni yollar belirlenmektedir. Uygunluk hesaplanırken denklem 6.9'den yararlanılır, uygunluğu hesaplanan bir yolun üzerinde bulunduğu her nokta için  $\lambda$  ağırlık katsayısı ile o noktanın tehlike maliyeti çarpılır ve önceki yakıt maliyeti ile toplanır. Bu ağırlık katsayısı, belirlenecek yolun aranmasında hangi maliyetin daha kritik olduğunu belirlemek için kullanılır. Bu katsayı yüksek olduğunda tehlikelerden daha uzak olan yollar seçilir, düşük olduğunda ise tehlikelere daha yakın yollar seçilir.

$$\text{Uygunluk} = \sum_{k=0}^n \mathbf{T}(k) \cdot \mathbf{x} \cdot \lambda + \mathbf{F}(k) \quad (6.9)$$

Algoritma çalışlığında;

1. Hedef noktası ve başlangıç noktasını içinde bulunduran bir doğrunun denklemi hesaplanır, bu doğru tehlike bilgileri dikkate alınmadığında hedef noktasına ulaşmak için takip edilebilecek en kısa yoldur.
2. Hedef ve başlangıç noktası arasındaki doğrunun üzerindeki her x değeri için her kurda bir y değeri gwo ile atanır. Bunun nedeni ise bütün potansiyel yolları test etmek yerine en kısa yolu takip edip o yol üzerinde değişiklikler yaparak çok daha kısa sürede çözüme ulaşabilmektir. Her kurda x ve y değerleri atamak yerine sadece y değerleri atandığında problemin boyut sayısı yarıya inmektedir.
3. Bu atanmış değerlerin uygunluğu hesaplanır, bu hesap sırasında kurdun sahip olduğu y değerleri ve var olan x değerleri birleştirilir ve bir yol oluşturulur, Bu yol üzerindeki her bir noktanın arasındaki mesafe ile yakıt maliyeti hesaplanır ve bu yol üzerinde her bir nokta için tehlike noktalarına olan mesafe hesaplanır bu mesafelere göre de bir tehlike maliyeti hesaplanır, yakıt ve tehlike maliyeti toplandıktan sonra geri gönderilir.

4. Geri gönderilen maliyetlere göre GWO ile yeni değerler üretilir ve bu değerlerin tekrar uygunluğu hesaplanır. Algoritma üçüncü adımdan itibaren iterasyon sayısı bitene kadar tekrar eder.
5. İterasyon sayısı bittiğinde ise en iyi çözüm ekranaya çıktı olarak verilir.  
Bu özetlenen çalışma prensiplerini ise şekil 6.8'da sözde kod şeklinde detaylı görebiliriz.

```

Begin
    1. İlk kurt değerlerini, a,A ve C değerlerini oluştur
    2. Hedef ve başlangıç noktaları arasında olan bir doğru denklemi belirlenir
        Bu doğru denklemi üzerinde olan x noktaları alınır
    3. Her bir kurt için (for)
        Yakıt ve tehdit maliyetini hesapla
        End for
        En uygun ilk üç kurt değerini seç
    4. İterasyon sayısına ulaşılmadığı sürece (while)
        Her kurt için (for)
            Pozisyon güncelle
            End for
            a, A ve C değerlerini güncelle
            her kurt için (for)
            Yakıt ve tehdit maliyetini hesapla
            End for
            En uygun üç kurt değerini seç
            İterasyon yükselt
            End while
            En uygun kurdu döndür
    End

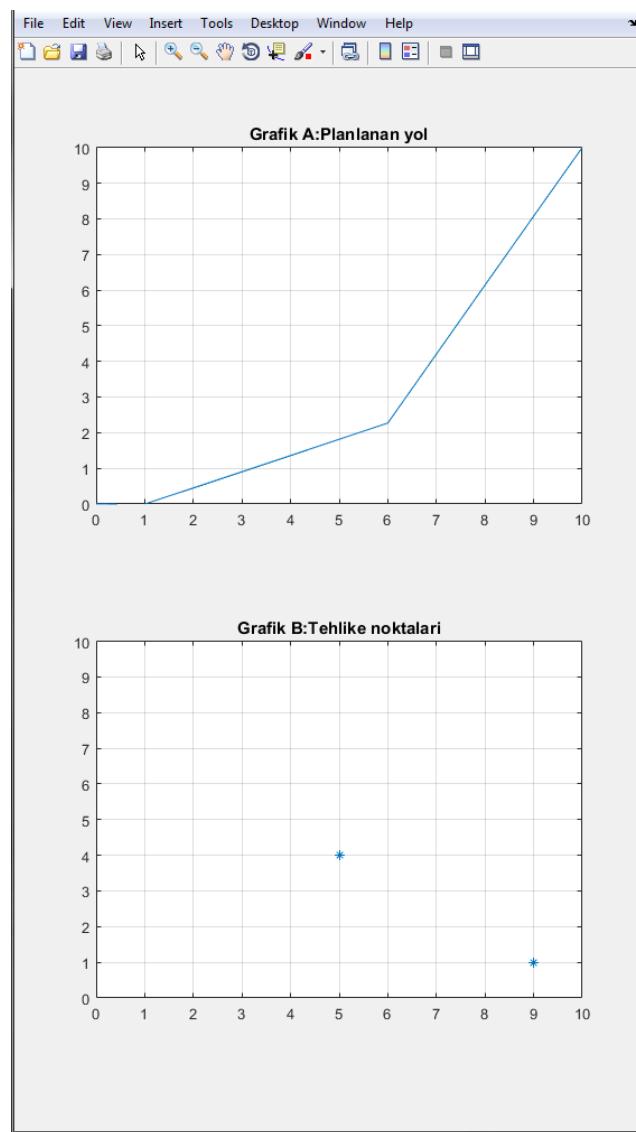
```

**Şekil 6.8 İHA GWO Sözde kod**

## 6.8 Örnek çalışma

Bu örnek çalışmada 10x10 kilometrelük bir arazi 10x10luk bir çözünürlük ile temsil edilmiştir (her noktanın arası 1 km). Çalışmanın amacı ise (0,0) noktasında bulunan bir insansız hava aracının (10,10) noktasına ulaşmak için takip edeceği yolu bulmaktır. Arazinin (5,4) ve (9,1) noktalarında ise 2 kilometre menzilli uçak savar silahları bulunmaktadır. İterasyon sayısı 5000, kurt sayısı 30 ve tehlike ağırlık katsayıısı ise 1 olarak verilmiştir. Algoritma sonlandığında 5000

iterasyonun sonunda şekil 6.9'da üstte planlanan yol ve alta ise tehlike noktaları belirtilmiş şekilde verilmiştir. Planlanan yolda İHA (5,4) noktasındaki tehlikeden uzak durmak için y değerini tehlichenin menzilinden çıkışa kadar düşürmüştür, menzilden çıktıgı an ise hedefe normal bir şekilde devam etmiştir, (9,1) noktasında bulunan tehlichenin menzilinin dışında olduğu için bu tehlikeyi dikkate almamıştır.



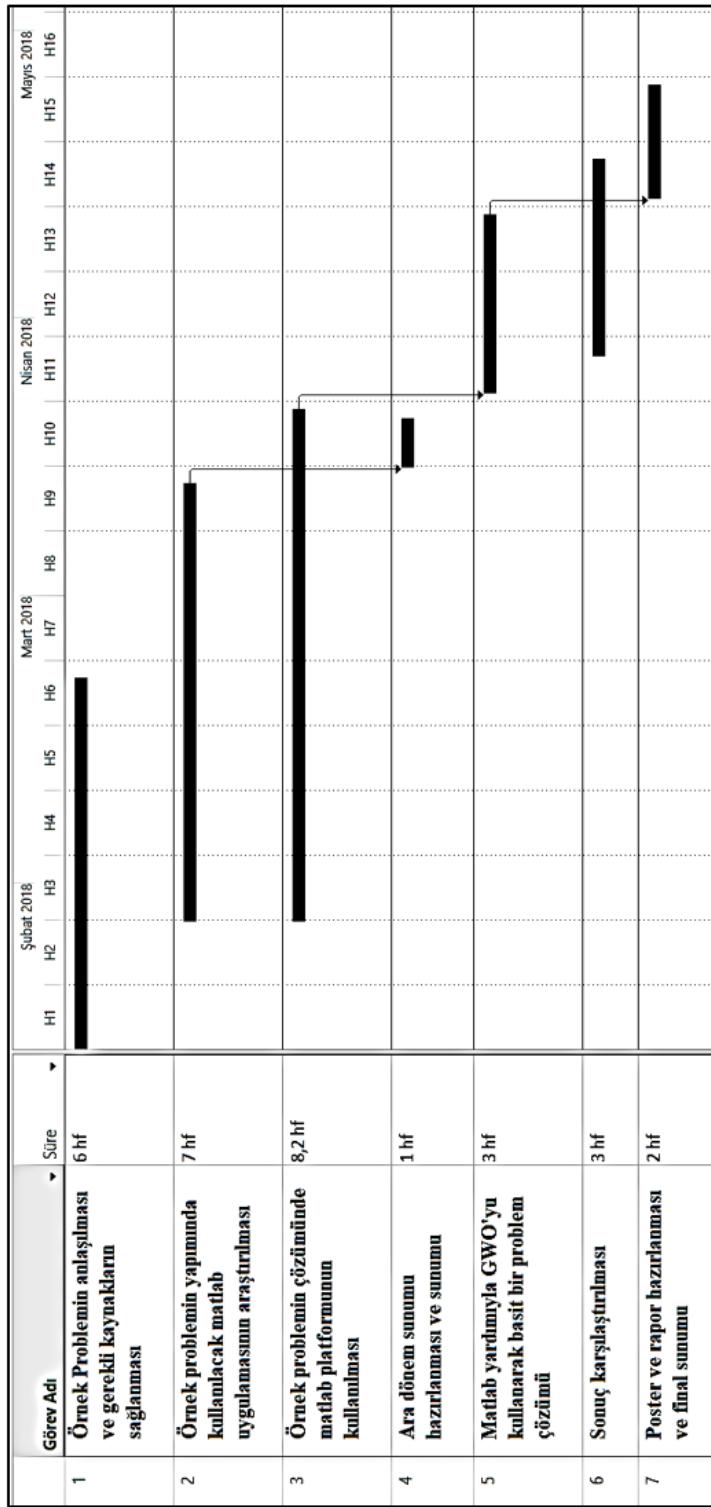
**Şekil 6.9** Örnek Çıktı

## 6.9 Sonuç

Gray wolf optimizer, sürü optimizasyonu konusu altında işlenen yeni bir algoritma olmasına rağmen bir çok alanda başarılı sonuçlar ortaya çıkarmıştır. Diğer algoritmaların daha yeni olması ve sürü içerisinde bulunan bireylerin özelliklerinin ve sayısının farklı olması nedeniyle uygulamaları daha zor olmasına rağmen hızlı bir şekilde gelişmektedir.

## 6.10 İş-Zaman Diyagramı

### Kurtlar ile Süre Optimizasyonu 2017-2018 Akademik Yılı Bahar Dönemi Bitirme Projesi İş-Zaman Planı



Hazırlayan : Mehmet Emre Yorulmaz  
20.03.2018

## **6.11 Kaynakça**

- Khairuzzaman A. K. and S. Chaudhury “Multilevel thresholding using grey wolf optimizer for image segmentation ,Expert Systems with Applications Volume 86 ” Elsevier (2017) pp. 64-76.
- Zhang X,Q. Miao, Z. Liu ve Z. He “An adaptive stochastic resonance method based on grey wolf optimizer algorithm and its application to machinery fault diagnosis ISA Transactions volume 71” (2017) pp. 206-214.
- Mason K. A., T. Duncan, G. B. Johnson, J. B. Losos and S. R. Singer “Understanding Biology 2<sup>nd</sup> Edition” McGraw-Hill Higher Education (2017) pp. 1157-1279.
- Yu H, Y. Yu, Y. Liu, Y. Wang and S. Gao “Chaotic grey wolf optimization International Conference on Progress in Informatics and Computing”, IEEE (2016) pp. 103- 113.
- Zhang, S.,Y. Zhou, Z. Li, W. Pan “Grey wolf optimizer for unmanned combat aerial vehicle path planning “Advances in Engineering Software issue 99” Elsevier (2016) ss. 121-136.
- Mirjalili S., S. Saremi, S.M Mirjalili and L. D. S. Ceolho “ Multi objective grey wolf optimizer Exper systems with applications volume 47” Elsevier (2016) pp. 106-119 .
- Breed M. D. and Janice Moore “Animal Behavior, Elsevier science and technology books 2<sup>nd</sup> Edition” Elsevier (2016) pp. 1-26.
- Mirjalili S., S. M. Mirjalili and A. Lewis “Grey Wolf Optimizer, Advances in Engineering Software issue 69” Elsevier (2014) pp. 46-61.
- Reece J. B., L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky and R. B. Jackson “Campbell Biology 10<sup>th</sup> Edition” Pearson Education (2013) pp. 409-477.
- Lieberman M. A. and R. Ricer “Molecular Biology, and Genetics 6<sup>th</sup> Edition, BRS Biochemistry” Wolters Kluver and Lippincott Williams Health (2013) pp. 1-86.
- Wei. B, Q. Peng and X. Chen “An improved particle swarm optimization based on wolves' activities circle, Proceedings of the 10th World Congress on Intelligent Control and Automation”, IEEE (2012) pp. 4557 – 4562.
- Muro C., R. Escobedo, L. Spector and R.P. Coppinger “Wolf-pack (*Canis lupus*) hunting strategies emerge from simple rules in computational simulations ,Behavioral Processes issue 88”, Elsevier (2011) pp. 192-197.

## VII KARŞILAŞTIRMA VE SONUÇ

### 7.1 Performans Ölçümü ve Karşılaştırma

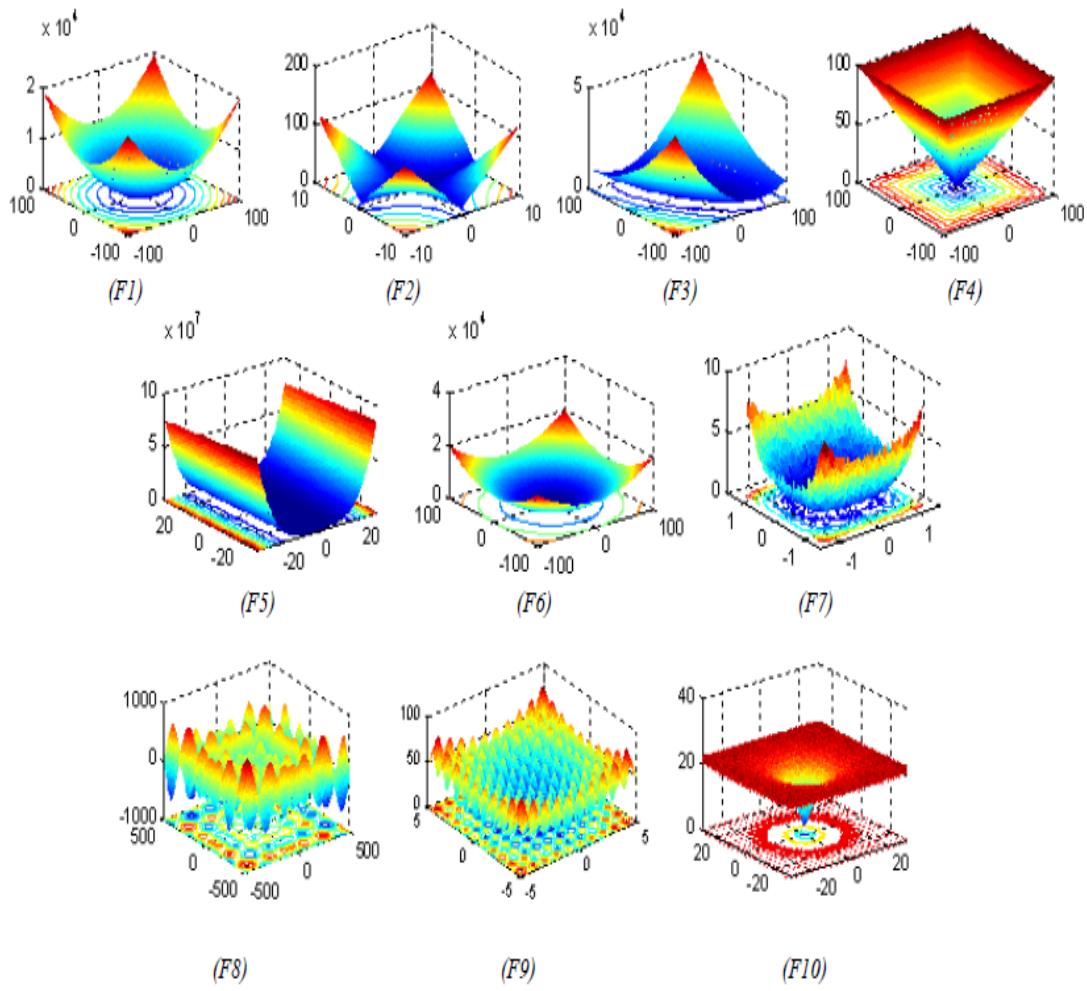
Bu bölümde projenin incelediği dört algoritmanın performansı belirli metotlar ile ölçülmesi ve karşılaştırılması amaçlanmaktadır. Sezgisel optimizasyon algoritmalarının performansının ölçümleri benchmark fonksiyonları ile yapılabilir. Bu fonksiyonlar optimizasyon algoritmalarının performans ölçümünü etkili bir şekilde gerçekleştirebilmektedir ve bir çok araştırmada kullanılmışlardır (Mirjalili,2014). Bu çalışmada toplamda on tane farklı benchmark fonksiyonu ile ABC, ACO, PSO ve GWO algoritmalarının performans ölçümü yapılmıştır. Kullanılan fonksiyonlar unimodal ve multimodal olmak üzere iki sınıfa ayrılmıştır, unimodal fonksiyonlar tek bir lokal maksimum değere sahipken multimodal fonksiyonların birden çok lokal maksimum değeri olabilir. Kullanılan benchmark fonksiyonları, boyutları (Dim), çalışma aralıkları (Range) ve min değerleri (Fmin) ile şekil 7.1'de verilmiştir. Fonksiyonların iki boyutlu görünümü şekil 7.2'de verilmiştir.

Tablolardaki Average (ortalama) değerleri her bir fonksiyonun tüm iterasyonlarının ortalamasını, Time (zaman) değerleri her bir fonksiyonun toplam çalışma sürelerini, Standart Deviation (standart sapma) değerleri ise her bir fonksiyonun standart sapma değerlerini vermektedir.

Function	Dim	Range	$f_{\text{min}}$
$f_1(x) = \sum_{i=1}^n x_i^2$	30	[-100,100]	0
$f_2(x) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	30	[-10,10]	0
$f_3(x) = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2$	30	[-100,100]	0
$f_4(x) = \max_i\{ x_i , 1 \leq i \leq n\}$	30	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
$f_6(x) = \sum_{i=1}^n [(x_i + 0.5)]^2$	30	[-100,100]	0
$f_7(x) = \sum_{i=1}^n i x_i^4 + \text{random}[0,1]$	30	[-1.28,1.28]	0

Function	Dim	Range	$f_{\text{min}}$
$F_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	[-500,500]	-418.9829×5
$F_9(x) = \sum_{i=1}^n [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
$F_{10}(x) = -20\exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0

Şekil 7.1 Benchmark fonksiyonları



**Şekil 7.2** İki boyutlu Benchmark fonksiyonları

ABC, ACO, PSO ve GWO algoritmaları bu fonksiyonlar ile bin iterasyon ile elli arama ajansı kullanarak yüz kere çalıştırılmıştır. Elde edilen sonuçların ortalaması ve standart sapması hesaplanmıştır. Bu denemeler MATLAB 2017Ra Platformunda Intel Core I7 4790k işlemciye, 16 Gigabyte DDR3 3999 MHZ frekansa sahip ram ve GEFORCE GTX 650TI BOOST grafik kartına, sahip bir bilgisayarda yapılmıştır.

## 7.2 Artificial Bee Colony Algoritmasının Benchmark Fonksiyonları ile İncelenmesi

Artificial Bee Colony algoritmasının benchmark fonksiyonları ile çalıştırılmasından elde edilen sonuçlar Ek A.2'de verilmiştir.

**Tablo 7.1** ABC Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri

Benchmarks	F1	F2	F3	F4
Average	9384,400781	477,4802174	80681,56793	73,37913891
Time (sec.)	34.577286	36.093287	58.880638	33.761224
Standard deviation	8767,07803	88,0563622	85309,94128	72,93665587

Benchmarks	F5	F6	F7	F8
Average	36230784,02	9245,407103	16,26933366	-4,96789E+14
Time (sec.)	36.398215	34.260816	39.848192	38.296503
Standard deviation	23953560,9	8536,988569	13,20342311	-1,2442E+14

Benchmarks	F9	F10
Average	306,5957096	17,71236006
Time (sec.)	35.864105	37.467774
Standard deviation	258,0979337	16,55345832

ABC algoritması çalıştığı benchmark fonksiyonlarının tamamının çalışma süresi toplamda 384.7 saniyedir. 3. benchmark fonksiyonu dışında diğerlerinin çalışma süreleri 30-40 saniye arasındadır. 3. Benchmark fonksiyonunun çalışma süresi ise 58.8 saniyedir. Bu sonuç 3. Benchmark fonksiyonunun ABC algoritması ile çalıştırıldığında işlemciyi diğer benchmark fonksiyonlarına göre daha çok zorladığı anlamına gelmektedir, çalışma süresi en kısa olan benchmark fonksiyonu ise 33.76 saniye ile 4. Benchmark fonksiyonudur. Çalışma sürelerinin en hızlıdan en yavaşa sıralamak istersek sıra; 4.,6.,1.,9.,10.,2.,5.,10.,8.,7.,3. Şeklinde olur. Tablo 7.1'de verilen standart sapma satırı algoritmaların sezgisel özelliğini tekrar öne çıkarmaktadır. Sezgisel algoritmalar her çalışmasında farklı sonuçlar verebileceğinden bu çalışmada yapıldığı gibi deneme sayısının yüksek tutulması önemlidir.

Tablo 7.1'de verilen ortalamalar önceki bölümlerde açıklanan diğer algoritmaların ortalamaları ile karşılaştırıldığında dört algoritmanın genel olarak hangi problemlerde daha etkili olacağını görebiliriz. Sonraki bölümde bu karşılaştırılmalar yapılmıştır.

### 7.3 Ant Colony Optimization Algoritmasının Benchmark Fonksiyonları ile İncelenmesi

Ant Colony Optimization algoritmasının benchmark fonksiyonları ile çalıştırılmasından elde edilen sonuçlar Ek B.2'de verilmiştir.

**Tablo 7.2** ACO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri

Benchmarks	F1	F2	F3	F4
Average	6.61E+00	6.41E+00	4.31E+00	3.43E+00
Time (sec.)	29.6698	41.482036	98.872015	42.6327
Standard deviation	3.991258112	3.106056753	1.729378739	1.671871239

Benchmarks	F5	F6	F7	F8
Average	2.51E+01	1.35E+00	1.99E-02	-5.76E+03
Time (sec.)	24.36721	43.06987	39.50791	41.96108
Standard deviation	4.38446911	0.400708866	0.011737782	621.1799698

Benchmarks	F9	F10
Average	6.39E+01	1.32E+00
Time (sec.)	62.18904	21.69207
Standard deviation	15.29510979	0.220097972

ACO ile çalışılan benchmark fonksiyonlarının tamamının çalışma süresi toplam 445.44419 saniyedir. 3. ve 9. benchmark fonksiyonları dışında diğerlerinin çalışma süreleri 20 ile 45 saniye arasındadır. 3. benchmark fonksiyonun çalışma süresi ise 98, 9. benchmark fonksiyonun çalışma süresi 62 saniyedir. Bu sonuç 3. ve 9. benchmark fonksiyonun ACO ile çalıştırıldığında işlemciyi diğer benchmark fonksiyonlarına göre daha çok zorladığı anlamına gelmektedir, çalışma süresi en kısa olan benchmark fonksiyonu ise 21.69207 saniye ile 10. benchmark fonksiyonudur. Çalışma sürelerinin en hızdan en yavaş sıralamak istersek sıra; 10., 5., 1., 7., 2., 8., 4., 6., 9., 3. şeklinde olur. Tablo 7.2'de verilen standart sapma satırı algoritmaların sezgisel özelliğini tekrar öne çıkarmaktadır. Sezgisel algoritmalar her çalışmasında farklı sonuçlar verebileceğinden bu çalışmada yapıldığı gibi deneme sayısının yüksek tutulması önemlidir.

Tablo 7.2'de verilen ortalamalar önceki bölümlerde açıklanan diğer algoritmaların ortalaması ile karşılaştırıldığında dört algoritmanın genel olarak hangi problemlerde daha etkili olacağını görebiliriz. Sonraki bölümde bu karşılaştırılmalar yapılmıştır.

#### 7.4 Particle Swarm Optimization Algoritmasının Benchmark Fonksiyonları ile İncelenmesi

Particle Swarm Optimization algoritmasının benchmark fonksiyonları ile çalıştırılmasından elde edilen sonuçlar Ek C.2'de verilmiştir.

**Tablo 7.3** PSO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri

Benchmarks	F1	F2	F3	F4
Average	7,96E-40	0,005709694	0,114699449	0,11608065
Time (sec.)	206.928.051	185.685.596	322.883.934	173.136.008
Standard deviation	7,90E-39	0,017817676	0,284024178	0,07179942

Benchmarks	F5	F6	F7	F8
Average	37,8452094	5,57E-16	0,010521514	-6577,243646
Time (sec.)	181.185.022	171.140.294	192.238.756	175.130.112
Standard deviation	28,74609513	5,52E-15	0,004697737	668,4216919

Benchmarks	F9	F10
Average	48,46437808	0,928955796
Time (sec.)	170.405.224	178.845.571
Standard deviation	15,58895856	7,94E-01

PSO çalıştığı benchmark fonksiyonlarının tamamının çalışma süresi toplamda 1956.571 saniyedir. 3. benchmark fonksiyonu dışında diğerlerinin çalışma süreleri 170-180 saniye arasındadır. 3. Benchmark fonksiyonunun çalışma süresi ise 322 saniyedir. Bu sonuç 3. Benchmark fonksiyonunun PSO ile çalıştırıldığında işlemciyi diğer benchmark fonksiyonlarına göre daha çok zorladığı anlamına gelmektedir, çalışma süresi en kısa olan benchmark fonksiyonu ise 170.4 saniye ile 9. Benchmark fonksiyonudur. Çalışma sürelerinin en hızdan en yavaşa sıralamak istersek sıra; 9.,6.,4.,8.,10.,5.,2.,7.,1.,3. Şeklinde olur. Tablo 7.3'de verilen standart sapma satırı algoritmaların sezgisel özelliğini tekrar öne çıkarmaktadır. Sezgisel algoritmalar her çalışmasında farklı sonuçlar verebileceğinden bu çalışmada yapıldığı gibi deneme sayısının yüksek tutulması önemlidir.

Tablo 7.3'de verilen ortalamalar önceki bölümlerde açıklanan diğer algoritmaların ortalamaları ile karşılaştırıldığında dört algoritmanın genel olarak hangi problemlerde daha etkili olacağını görebiliriz. Sonraki bölümde bu karşılaştırılmalar yapılmıştır.

## 7.5 Gray Wolf Optimizer Algoritmasının Benchmark Fonksiyonları ile İncelemesi

Gray wolf optimizer algoritmasının benchmark fonksiyonları ile çalıştırılmasından elde edilen sonuçlar Ek D.2'de verilmiştir.

**Tablo 7.4** GWO Benchmark Fonksiyonlarının Average, Time (sec.), Standart Deviation değerleri

Benchmarks	F1	F2	F3	F4
Average	3.20E-70	4.26E-41	3.02E-19	1.91E-17
Time (sec.)	3.12E+01	34.386021	116.939583	30.296272
Standard deviation	7.88123E-70	4.36563E-41	1.55436E-18	3.09747E-17

Benchmarks	F5	F6	F7	F8
Average	2.64E+01	3.75E-01	4.63E-04	-6.16E+03
Time (sec.)	33.959303	30.051411	48.829518	37.178855
Standard deviation	0.730322667	0.30877764	0.000254972	766.0962148

Benchmarks	F9	F10
Average	3.86E-01	1.34E-14
Time (sec.)	30.987519	33.319876
Standard deviation	1.508702583	3.23492E-15

Gwo çalıştığı benchmark fonksiyonlarının tamamının çalışma süresi toplamda 427.1 saniyedir. 3. benchmark fonksiyonu dışında diğerlerinin çalışma süreleri 30-40 saniye arasındadır. 3. Benchmark fonksiyonunun çalışma süresi ise 117 saniyedir. Bu sonuç 3. Benchmark fonksiyonunun GWO ile çalıştırıldığında işlemciyi diğer benchmark fonksiyonlarına göre daha çok zorladığı anlamına gelmektedir, çalışma süresi en kısa olan benchmark fonksiyonu ise 30.05 saniye ile 6. Benchmark fonksiyonudur. Çalışma sürelerinin en hızlıdan en yavaşa sıralamak istersek sıra; 6.,4.,9.,1.,10.,5.,2.,8.,7.,3. Şeklinde olur. Tablo 7.4'de verilen standart sapma satırı algoritmaların sezgisel özelliğini tekrar öne çıkarmaktadır. Sezgisel algoritmalar her çalışmasında farklı sonuçlar verebileceğinden bu çalışmada yapıldığı gibi deneme sayısının yüksek tutulması önemlidir.

Tablo 7.4'de verilen ortalamalar önceki bölümlerde açıklanan diğer algoritmaların ortalamaları ile karşılaştırıldığında dört algoritmanın genel olarak hangi problemlerde daha etkili olacağını görebiliriz. Sonraki bölümde bu karşılaştırılmalar yapılmıştır.

## 7.6 Sonuçların Karşılaştırılması

**Tablo 7.5** Algoritmaların karşılaştırılması

	ABC	ACO	PSO	GWO
<b>Toplam saniye</b>	384.7	445.39	1956	427
<b>En hızlı benchmark</b>	33.7 4. benchmark	21.69 10. benchmark	170.4 9. benchmark	30.05 6. benchmark
<b>En yavaş benchmark</b>	58 3. benchmark	98.87 3. benchmark	322 3. benchmark	117 saniye 3. benchmark
<b>1. fonksiyon ortalama</b>	9384.400781	6.61E+00	7.96467E-40	3.19818E-70
<b>2. fonksiyon ortalama</b>	477.4802174	6.41E+00	0.005709694	4.26438E-41
<b>3. fonksiyon ortalama</b>	80681.56793	4.31E+00	0.114699449	3.02464E-19
<b>4. fonksiyon ortalama</b>	73.37913891	3.43E+00	0.11608065	1.90909E-17
<b>5. fonksiyon ortalama</b>	36230784.02	2.51E+01	37.8452094	26.41545095
<b>6. fonksiyon ortalama</b>	9245.407103	1.35E+00	5.56776E-16	0.375351843
<b>7. fonksiyon ortalama</b>	16.26933366	1.99E-02	0.010521514	0.000463105
<b>8. fonksiyon ortalama</b>	-4.96789E+14	-5.76E+03	-6577.243646	-6158.308939
<b>9. fonksiyon ortalama</b>	306.5957096	6.39E+01	48.46437808	0.385837081
<b>10. fonksiyon ortalama</b>	17.71236006	1.32E+00	0.928955796	1.34293E-14

İncelenen bütün algoritmaların sonuçları tablo 7.5'de verilmiştir. Bu sonuçlar karşılaştırıldığında en hızlı çalışan algoritma 384 saniye ile ABC bulunmaktadır. Diğer algoritmala göre hızlı çalışan ABC algoritması matrix içeren işlemlerde başarılı olmasından dolayı olmaktadır. İkinci en hızlı algoritma ise GWO algoritmasıdır. GWO ile PSO algoritmalarının çalışma prensipleri benzerdir, ancak GWO, PSO algoritmasının zorlandığı durumlarda da başarılı olabilecek şekilde geliştirilmiştir. Bu nedenden dolayı PSO algoritmasından daha hızlıdır. Üçüncü en hızlı algoritma ise ACO algoritmasıdır. En yavaş çalışan ise PSO algoritmasıdır. PSO algoritmasının fonksiyon ortalamaları, GWO ile diğerlerine göre daha yakındır. Bu yakınlığın nedeni PSO ve GWO algoritmalarının çalışma prensiplerinin benzer olmasıdır.

## 7.7 Benchmark Fonksiyonlarının MATLAB Kodu

```
function [lb,ub,dim,fobj] = Get_Functions_details(F)
```

```
switch F
    case 'F1'
        fobj = @F1;
        lb=-100;
        ub=100;
        dim=30;

    case 'F2'
        fobj = @F2;
        lb=-10;
        ub=10;
        dim=30;

    case 'F3'
        fobj = @F3;
        lb=-100;
        ub=100;
        dim=30;

    case 'F4'
        fobj = @F4;
        lb=-100;
        ub=100;
        dim=30;

    case 'F5'
        fobj = @F5;
        lb=-30;
        ub=30;
        dim=30;

    case 'F6'
        fobj = @F6;
        lb=-100;
        ub=100;
        dim=30;

    case 'F7'
        fobj = @F7;
        lb=-1.28;
        ub=1.28;
        dim=30;

    case 'F8'
        fobj = @F8;
        lb=-500;
```

```

ub=500;
dim=30;

case 'F9'
fobj = @F9;
lb=-5.12;
ub=5.12;
dim=30;

case 'F10'
fobj = @F10;
lb=-32;
ub=32;
dim=30;

end

% F1

function o = F1(x)
o=sum(x.^2);
end

% F2

function o = F2(x)
o=sum(abs(x))+prod(abs(x));
end

% F3

function o = F3(x)
dim=size(x,2);
o=0;
for i=1:dim
    o=o+sum(x(1:i))^2;
end
end

% F4

function o = F4(x)
o=max(abs(x));
end

% F5

function o = F5(x)
dim=size(x,2);
o=sum(100*(x(2:dim)-(x(1:dim-1).^2)).^2+(x(1:dim-1)-1).^2);

```

```
end
```

```
% F6
```

```
function o = F6(x)
o=sum(abs((x+.5)).^2);
end
```

```
% F7
```

```
function o = F7(x)
dim=size(x,2);
o=sum([1:dim].*(x.^4))+rand;
end
```

```
% F8
```

```
function o = F8(x)
o=sum(-x.*sin(sqrt(abs(x))));
```

```
end
```

```
% F9
```

```
function o = F9(x)
dim=size(x,2);
o=sum(x.^2-10*cos(2*pi.*x))+10*dim;
```

```
end
```

```
% F10
```

```
function o = F10(x)
dim=size(x,2);
o=-20*exp(-.2*sqrt(sum(x.^2)/dim))-exp(sum(cos(2*pi.*x))/dim)+20+exp(1);
```

```
end
```

Optimizasyon problemleri için birçok algoritma önerilmiştir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalarıdır. Bu çalışmada sezgisel optimizasyon algoritmalarından sürü zekâsı tabanlı olanlar incelenmiş ve bu algoritmaların ABC, ACO, PSO ve GWO ayrıntılı şekilde açıklanmıştır. Doğadaki birçok süreç, olay ya da model incelenerek değişik sürü zekâsı tabanlı yöntemler geliştirilip farklı problemler için etkili çözüm bulmak amacıyla kullanılabilir. Sezgisel algoritmaların her biri tüm problemler için en iyi sonucu vermeyebilir. Bir problem için iyi çözüm veren bir yöntem, başka bir problem için kötü olabilir. Bu yüzden algoritmaların iyi çözüm verdiği problem tipleri belirlenmelidir. Sonraki çalışmalarda, bu yöntemlerin çeşitli örnek problemlere uygulanması ve sonuçlarının karşılaştırılması amaçlanmaktadır.

## VIII EKLER

### EK A

#### A.1 Tarım Alanlarının Sınıflandırılması Problemi MATLAB Kodları

##### main.m

```
clc;
clear;
close all;

% 0 = Not Categorized
% 1 = Product Category 1
% 2 = Product Category 2
% 3 = Product Category 3
% 4 = Product Category 4
```

```
global map
global cat
global iterationposition
global myx
global myy
global dist1
global dist2
global dist3
global dist4
global mindist
```

```
cat = zeros(30,30)
map = randi ([0 256], 30 , 30 );
```

```
abc;
```

##### abc.m

```
clc;
clear;
close all;

%% Problem Definition
global map
global cat
global iterationposition
global myx
global myy
global dist1
global dist2
global dist3
global dist4
global mindist
```

```

CostFunction=@(x) Sphere(x);      % Cost Function

nVar=2;           % Number of Decision Variables

VarSize=[1 nVar]; % Decision Variables Matrix Size

VarMin=0;         % Decision Variables Lower Bound
VarMax= 10;       % Decision Variables Upper Bound

%% ABC Settings

MaxIt=400;        % Maximum Number of Iterations

nPop=100;          % Population Size (Colony Size)

nOnlooker=nPop;   % Number of Onlooker Bees

L=round(0.6*nVar*nPop); % Abandonment Limit Parameter (Trial Limit)

a=1;               % Acceleration Coefficient Upper Bound

%% Initialization

% Empty Bee Structure
empty_bee.Position=[];
empty_bee.Cost=[];

% Initialize Population Array
pop=repmat(empty_bee,nPop,1);

% Initialize Best Solution Ever Found
BestSol.Cost=inf;

% Create Initial Population
for i=1:nPop
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
    pop(i).Cost=CostFunction(pop(i).Position);
    if pop(i).Cost<=BestSol.Cost
        BestSol=pop(i);
    end
end

% Abandonment Counter
C=zeros(nPop,1);

% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);

%% ABC Main Loop

```

```

for it=1:MaxIt

    % Recruited Bees
    for i=1:nPop

        % Choose k randomly, not equal to i
        K=[1:i-1 i+1:nPop];
        k=K(randi([1 numel(K)]));

        % Define Acceleration Coeff.
        phi=a*unifrnd(-1,+1,VarSize);

        % New Bee Position
        newbee.Position=pop(i).Position+phi.*(pop(i).Position-
            pop(k).Position);

        % Evaluation
        newbee.Cost=CostFunction(newbee.Position);

        % Comparision
        if newbee.Cost<=pop(i).Cost
            pop(i)=newbee;
        else
            C(i)=C(i)+1;
        end

    end

    % Calculate Fitness Values and Selection Probabilities
    F=zeros(nPop,1);
    MeanCost = mean([pop.Cost]);
    for i=1:nPop
        F(i) = exp(-pop(i).Cost/MeanCost); % Convert Cost to Fitness
    end
    P=F/sum(F);

    % Onlooker Bees
    for m=1:nOnlooker

        % Select Source Site
        i=RouletteWheelSelection(P);

        % Choose k randomly, not equal to i
        K=[1:i-1 i+1:nPop];
        k=K(randi([1 numel(K)]));

        % Define Acceleration Coeff.
        phi=a*unifrnd(-1,+1,VarSize);

        % New Bee Position
        newbee.Position=pop(i).Position+phi.*(pop(i).Position-pop(k).Position);

    end
end

```

```

% Evaluation
newbee.Cost=CostFunction(newbee.Position);

% Comparision
if newbee.Cost<=pop(i).Cost
    pop(i)=newbee;
else
    C(i)=C(i)+1;
end

end

% Scout Bees
for i=1:nPop
    if C(i)>=L
        pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
        pop(i).Cost=CostFunction(pop(i).Position);
        C(i)=0;
    end
end

% Update Best Solution Ever Found
for i=1:nPop
    if pop(i).Cost<=BestSol.Cost
        BestSol=pop(i);
    end
end

% Store Best Cost Ever Found
BestCost(it)=BestSol.Cost;

% Display Iteration Information
disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it))]);

end

%% Results
display_matrix(cat);
display_matrix(map);
%figure;
%plot(BestCost,'LineWidth',2);
%semilogy(BestCost,'LineWidth',2);
%xlabel('Iteration');
%ylabel('Best Cost');

```

## Sphere.m

```

function z=Sphere(x)
global map

```

```

global cat
global myx
global myy
global dist1
global dist2
global dist3
global dist4
global mindist

global iterationposition
iterationposition=x;
myx=x(1);
myy=x(2);
myx = round(myx,0);
myy = round(myy,0);

if(myx<=0)
    myx=1;
end
if(myy<=0)
    myy=1;
end
if(myx>30)
    myx=30;
end
if(myy>30)
    myy=30;
end
dist1 = abs(map(myx,myy)-200);
dist2 = abs(map(myx,myy)-150);
dist3 = abs(map(myx,myy)-250);
dist4 = abs(map(myx,myy)-75);

mindist=dist1;
if(dist1<dist2 && dist1<dist3 && dist1<dist4)
    mindist=dist1;
    cat(myx,myy)=1;
end
if(dist2<dist1 && dist2<dist3 && dist2<dist4)
    mindist=dist2;
    cat(myx,myy)=2;
end
if(dist3<dist1 && dist3<dist2 && dist3<dist4)
    mindist=dist3;
    cat(myx,myy)=3;
end
if(dist4<dist1 && dist4<dist2 && dist4<dist3)
    mindist=dist4;
    cat(myx,myy)=4;

```

```
end
```

```
z=mindist;  
end
```

### RouletteWheelSelection.m

```
function i=RouletteWheelSelection(P)  
  
r=rand;  
  
C=cumsum(P);  
  
i=find(r<=C,1,'first');  
  
end
```

### display\_matrix.m

```
function display_matrix(matrix,varargin)  
  
indigo = [0.2471 0.3176 0.7098];  
dark_indigo=[ 0.1882 0.2431 0.6314];  
header_gray=[ 0.9176 0.9176 0.9176];  
line_gray=[ 0.8118 0.8118 0.8118];  
divider_gray=[ 0.9020 0.9020 0.9020];  
background_gray=[ 0.9804 0.9804 0.9804];  
title_gray=[ 0.9098 0.9176 0.9647];  
status_gray=[ 0.7725 0.7922 0.9137];  
text_gray=[ 0.4588 0.4588 0.4588];  
text_darkgray=[ 0.3804 0.3804 0.3804];  
diagonal_gray=[ 0.1294 0.1294 0.1294];  
purple=[ 0.7922 0.5725 0.8510];  
  
M=size(matrix);  
  
% grab input arguments  
rowheader=[]; colheader=[]; title_text=[];  
while(~isempty(varargin))  
    if(length(varargin)<2)  
        error('Argument <strong>%s</strong> has invalid value argument.',varargin{1});  
    end  
    switch varargin{1}  
        case 'rowheader', rowheader=varargin{2};  
        case 'colheader', colheader=varargin{2};  
        case 'title', title_text=varargin{2};  
    end  
    varargin(1:2)=[];
```

```

end

% determine orientation of output paper
rows=M(1); cols=M(2);
if(rows>cols),orientation='portrait';
else orientation='landscape';
end

% generate figure
fig_width=11; fig_height=8.5;
f=figure('units','inches','dockcontrols','off','paperpositionmode','manual',...
    'paperorientation',orientation,'toolbar','none','menubar','none','paperunits','normalized',...
    'paperposition',[0 0 1 1],'renderer','painters','color','background_gray','inverthardcopy','off');
oldUnits=get(0,'units'); set(0,'units','inches');
scrsz = get(0,'ScreenSize'); set(0,'units',oldUnits);
left=(scrsz(3)-fig_width)*.5; bot=(scrsz(4)-fig_height)*.5;
set(f,'outerposition',[left bot fig_width fig_height]);

if(isempty(rowheader)), rowheader=1:cols; end
if(~ischar(rowheader)), rowheader=trim(cellstr(int2str(rowheader(:)))); end
if(isempty(colheader)), colheader=1:rows; end
if(~ischar(colheader)), colheader=trim(cellstr(int2str(colheader(:)))); end
if(isempty(title_text)),title_text=sprintf('Matrix: %s',inputname(1)); end
rowheader_p=.05;
marg=.01;

status_p=.03; % status bar height
title_p=.06; % title bar height
max_colheader_len=max(cellfun(@length,colheader));
colheader_p=min(.02+.01*max_colheader_len,.1);
row_p=(1-status_p-title_p)/rows; % row height
col_p=1/cols; % column height

axes('position',[0 0 1 1],'box','off','xtick',[],'ytick',[],'xlim',[0 1],'ylim',[0
1],'color','background_gray'); hold all;

% display status bar
patch('Xdata',[-.1 -.1 1.1 1.1],'YData',[1-title_p 1 1 1-
title_p],'FaceColor','dark_indigo','EdgeColor','none');

% display title bar
ytop=1-status_p; ybot=1-status_p-title_p;
patch('XData',[-.1 -.1 1.1 1.1],'YData',[ybot ytop ytop
ybot],'FaceColor','indigo','EdgeColor','none');

% display header row
ytop=1-status_p-title_p;
ybot=1-status_p-title_p-rowheader_p;

```

```

patch('XData',[-.1 -.1 1.1 1.1],'YData',[ybot ytop ytop
ybot],'FaceColor','header_gray','EdgeColor','none');

% draw lines
plot([0 1],[1 1]*1-status_p-title_p-rowheader_p,'linewidth',1,'color',line_gray);

X=repmat([colheader_p+marg 1 nan],1,M(1)-1);
y=linspace(0,(1-status_p-title_p-rowheader_p),M(1)+1); y([1 end])=[];
Y=reshape([y;y;nan(1,M(1)-1)],1,[]);
plot(X,Y,'linewidth',1,'color',divider_gray);

% print date on the top right of page
text(.99,1-status_p/2,datestr(now,'mmmm dd, yyyy, HH:MM
AM'),'horizontalalignment','right',...
'verticalalignment','middle','fontunits','normalized','fontsize',status_p*.6,'color',status_gray,'fo
ntangle','italic');

text(0.5,1-status_p-title_p/2,title_text,'horizontalalignment','center',...
'verticalalignment','middle','fontunits','normalized','fontsize',title_p*.8,'color',title_gray);

% convert to text
M2=matrix(:);
mat_string=reshape(arrayfun(@(x) strtrim(cellstr(sprintf('%2.2f',x))),M2),M);

X=linspace(colheader_p+marg,1,M(2)+1); X(end)=[];
X=X+(X(2)-X(1))/2;
Y=linspace(1-status_p-title_p-rowheader_p,0,M(1)+1); Y(end)=[];
Y=Y+(Y(2)-Y(1))/2;
colheader_X=colheader_p/2; rowheader_y=(1-status_p-title_p-rowheader_p)/2;
header_size=min([row_p col_p colheader_p rowheader_p .4])*.5;
% rows
for i = 1:M(1)
    y=Y(i);

    % colheader first
    text(colheader_X,y,colheader{i}, 'horizontalalignment','center','verticalalignment','middle',...
        'fontunits','normalized','fontsize',header_size,'color',text_darkgray,'fontweight','bold');

    for j = 1:M(2)
        x=X(j);
        if(i==1)

text(x,rowheader_y,rowheader{j}, 'horizontalalignment','center','verticalalignment','middle',...
    'fontunits','normalized','fontsize',header_size,'color',text_darkgray,'fontweight','bold');
        end
        if(i==j),textcolor=diagonal_gray;
        else textcolor=text_gray;
        end
    end
end

```

```

text(x,y,mat_string{i,j},'horizontalalignment','center','verticalalignment','middle','fontunits','no
rmalized','fontsize',min(row_p,col_p)*.4,'color',textcolor);
    end
end

% add print button
uicontrol('style','pushbutton','units','normalized','position',[.005 1-status_p*.9 .05
status_p*.8],...

'String','PDF','backgroundcolor',purple,'foregroundcolor',text_darkgray,'CallBack',{ @print_m
atrix,f,title_text});

function print_matrix(hObject,~,fig_handle,title_text)
set(hObject,'visible','off');
filename=sprintf("%s.pdf",strrep(title_text,' ','_'));
print(fig_handle,filename,'-dpdf','-r600');
set(hObject,'visible','on');

```

## A.2 Artificial Bee Colony Algoritmasının İki Boyutlu Benchmark Fonksiyonları Tablosu

Benchmarks	F1	F2	F3	F4
	8767,07803	88,0563622	85309,94128	72,93665587
	9529,317633	645,1031393	85358,97223	72,73195822
	11300,51933	2404,150033	88910,56621	69,57393924
	10912,0673	136,1257037	72833,27717	79,67046895
	6520,602457	151,4036278	75746,25272	75,27868899
	10006,23005	150,8499169	96084,27582	70,7202861
	7938,656143	146,6495213	148978,5181	79,922686
	8711,970629	112,2430503	75201,04999	65,48548377
	10601,81422	126,7641743	82632,74688	82,45440409
	9001,255415	168,1308254	106032,2922	78,3397266
	12031,6612	113,266317	81407,13761	74,05743271
	13586,71531	209,7937965	70770,3524	68,58234645
	10473,94153	140,5685553	63603,40585	73,87177867
	7945,486908	149,7447548	82903,3442	65,94877105
	6526,341965	194,9161007	101259,8638	78,3606697
	6931,123943	76,01847177	80208,56459	73,35103467
	10360,68304	75,35693223	79032,74566	77,75246201
	10358,23335	461,9007625	60768,72587	78,91505247
	8419,588097	679,2342739	62808,30598	72,28059528
	9437,945106	167,385483	83306,44799	75,82936609
	10727,79846	74,89581548	76211,7931	79,41875637
	10015,76599	451,4556599	79144,07572	75,55538729
	8190,755433	147,5218912	71551,96255	75,5979657
	11262,59932	116,9963704	89307,52265	74,04592591
	6503,27966	172,8011961	101528,9513	74,69511876
	8229,594319	1187,12847	112585,8419	76,41236623
	7056,057913	103,8288075	86019,66784	76,09769346
	7577,942425	164,6213163	94747,22226	71,87738733
	9909,805599	116,658479	85089,54493	77,79656255
	8815,547108	163,234097	49016,09314	74,82447594
	8222,196016	192,4524026	100713,0174	76,61986543
	9172,639309	98,98638947	78353,6152	76,3587911
	7166,772832	171,3293473	72444,59048	71,69120678
	10624,77199	108,1856773	93203,33369	75,04837556
	9440,898677	88,60284034	78851,65001	73,5216725
	5584,019972	156,4430726	59494,74349	70,72545878
	10143,03494	310,330571	73557,03061	79,63332721
	7150,786368	109,1374976	79740,30171	76,0970082
	7677,532674	362,1459708	51406,78739	81,12430099
	9201,687428	2992,860701	68983,65734	73,39018422
	11049,53899	424,2847812	97983,45665	65,27466391
	8170,991465	172,1638155	95594,05795	77,79336513

	8159,236586	1262,690727	81004,13365	61,46493971
	15770,2685	92,93225216	76669,96192	64,64162549
	7639,061307	11528,71794	69928,06176	66,96192966
	10141,47093	8124,200085	92550,53411	72,78542447
	7851,867167	118,9326546	74156,07687	71,4503122
	7656,05649	102,248887	60007,85452	70,0574589
	11294,71004	223,5619769	84188,5159	67,63389762
	11515,64251	435,2241409	86303,88711	80,17645718
	9947,419575	221,6218836	73646,61843	69,47245443
	10449,88204	395,5400799	68858,35038	74,92488641
	8754,052147	153,7734922	58304,83471	75,88045182
	12396,42493	108,7306493	91328,38916	62,8891066
	8092,213476	100,4499621	78658,04443	69,56303633
	9423,86891	339,9326907	49164,17929	65,13324907
	7944,35095	140,409373	78692,88033	72,13253891
	6874,08027	168,9592855	107510,9566	78,47206408
	8124,023548	101,9304506	72982,03004	79,22141406
	9998,44689	126,8633238	86209,13181	75,82455759
	8366,972341	180,4991942	111090,4852	76,28645602
	9561,601069	192,7910381	81107,2448	76,39072428
	10563,40331	89,70191246	77165,56129	68,21907128
	7130,077519	109,5036689	62152,96975	77,49870852
	10072,77763	219,5606911	52588,69835	76,58611115
	7783,003316	133,1292953	94902,73063	77,03161956
	11612,14644	132,1939765	96334,08407	81,92310537
	5816,694047	102,6777256	69204,89517	58,62991109
	10281,97507	179,5202346	78644,30207	73,252207
	7104,972207	132,4842298	57387,27568	64,04595402
	12323,86459	278,8171367	91685,87932	76,64045026
	12807,0765	133,3774076	91757,9586	76,2428034
	7626,143941	120,469057	54584,89241	74,52472506
	8734,654579	1090,915348	122773,1297	73,73194573
	10402,43735	121,5449498	91002,11076	78,33789844
	9245,298147	105,3963797	92799,54635	76,71028103
	10924,33981	137,4483819	89339,49074	75,39226342
	9053,857001	101,6450647	71451,17489	73,40400677
	11363,65031	404,2616516	67988,59883	66,52458458
	11089,26876	92,97213883	86221,14752	70,62052535
	10002,06522	158,2346715	65718,17119	69,4411853
	8163,030233	73,56884123	83702,3778	75,75024168
	7764,91072	155,7486987	62526,80473	70,32681964
	8288,486773	188,349007	89888,82667	62,67042695
	5201,726394	78,7337404	69422,89299	65,67255894
	9191,165661	711,7009159	97920,15675	71,28374516
	8401,957043	102,4497382	111297,7574	82,22178295

	7320,740278	2272,639674	81644,1951	72,516555
	13040,56522	91,36426471	80358,30289	71,39124764
	11188,98411	121,1122888	81591,22881	72,64273769
	11746,06584	157,7474238	88632,69634	64,41362561
	13166,80813	226,8899318	68813,43307	77,66953277
	11723,2562	77,58316471	57905,29185	77,00586163
	11682,68849	422,5187568	85881,18638	81,70603532
	9610,337103	176,3515736	84723,44765	75,86933794
	8413,820549	177,2715588	70568,5987	64,33536941
	8024,971022	97,03248151	52860,96075	65,98552258
	7705,213512	124,1095988	75490,39617	75,01961762
	8728,163121	167,7573962	62452,3906	75,4478542
	13922,58372	77,50170202	87725,35639	76,20100981
Average	<b>9384,400781</b>	<b>477,4802174</b>	<b>80681,56793</b>	<b>73,37913891</b>
Time (sec.)	34.577286	36.093287	58.880638	33.761224
Standard deviation	8767,07803	88,0563622	85309,94128	72,93665587

Benchmarks	F5	F6	F7	F8
	23953560,9	8536,988569	13,20342311	-1,2442E+14
	35031362,12	9297,869752	16,14072486	-6,91923E+12
	25128156,45	7861,552079	13,18958905	-8,77766E+13
	39647568,97	10609,31199	20,71317371	-8,85E+15
	31462991,1	9610,418887	5,18438521	-3,19251E+13
	22454847,02	8237,07972	13,51241117	-5,20145E+12
	44674916,77	4899,124582	16,20395416	-5,41107E+14
	32598358,47	7925,432192	15,84222788	-9,84237E+14
	59625240,78	13108,07612	8,565355317	-1,28056E+14
	20773328,72	10860,09302	15,39311009	-1,85531E+14
	37159062,96	6983,513775	17,07732953	-1,06718E+14
	34684291,51	9128,694887	12,64808293	-4,45022E+13
	29446336,76	9211,342364	9,7200003	-4,80216E+13
	35060889,21	6556,302547	12,99145347	-7,39785E+13
	22850925,76	9981,304649	17,59490744	-3,98449E+13
	49656859,74	12257,7104	17,0443873	-9,43726E+13
	43038069,54	9633,501778	10,74192688	-1,10566E+14
	43290049,26	8525,759217	19,36927232	-1,27669E+14
	42726196,68	9505,4278	19,36376316	-5,51216E+13
	28131423,47	11355,65801	25,35972721	-1,40876E+14
	25718405,71	11170,11577	9,400520034	-2,40486E+14
	22777810,34	7219,173447	17,17789842	-1,27E+15
	45761070,01	9728,142127	17,9170524	-4,83106E+12
	58143841,95	8958,946289	17,12688753	-3,65002E+14
	18491326,11	8840,116338	24,67287032	-1,94461E+14
	49033093,71	11224,04666	15,31955293	-1,20889E+14

	10436916,22	9774,9228	7,978159553	-1,55E+15
	57305319,79	10595,9515	17,0660736	-1,08046E+14
	46167170,79	10281,36158	19,02554464	-1,70972E+14
	10441712,95	7783,977068	25,00792338	-1,31E+16
	28316346,48	10208,70381	17,08417831	-4,44373E+13
	28534480,45	9007,605925	11,41661382	-9,03714E+13
	64204487,9	10630,97222	12,96133126	-3,70061E+14
	34096857,2	8265,131559	11,37155966	-1,43233E+13
	62251641,59	8820,927404	17,21325307	-1,27384E+14
	22534041,9	10359,78498	24,76778584	-2,28209E+13
	45887921,41	9641,65321	10,24986702	-5,49743E+12
	41010907,88	6651,826245	15,00505121	-4,15095E+13
	62368491,62	5648,719882	19,0935344	-3,31E+15
	62329786,26	9544,049228	16,91971565	-9,47095E+12
	37671608,59	9411,980272	26,19179059	-6,14373E+14
	44930524,85	7826,75265	17,82495982	-1,25983E+14
	31828029,61	7753,929132	15,680476	-2,38139E+14
	30519343,97	6147,865005	24,49165163	-5,02879E+13
	24874324,07	8302,452633	15,07028145	-1,5465E+14
	13889577,83	11902,10719	23,4392772	-9,89277E+13
	25504982,3	11219,65348	17,01492085	-1,9262E+13
	30816323,73	9378,381226	16,46807726	-1,6057E+13
	28791762,19	8975,32106	9,236805264	-9,06205E+12
	61728345,86	10205,01928	6,30414221	-8,52269E+14
	50150221,09	8000,107358	10,96996381	-8,1606E+13
	58553366,05	5199,038735	5,70801389	-9,64748E+13
	45996698,96	8452,948054	25,18452284	-5,67568E+13
	40161181,76	10048,93699	24,44968377	-8,15334E+14
	18989733,1	4559,933723	13,03728339	-4,54615E+14
	43860959,08	10350,96205	17,91173786	-5,00937E+13
	53370780,16	8288,283475	8,745161869	-2,58635E+14
	26383668,41	9206,965103	27,74535203	-1,57312E+14
	25768065,36	7657,245597	12,19414847	-2,79851E+13
	48522376,01	9332,880843	22,73183663	-6,17E+15
	28060656	12911,46781	18,94822991	-6,82958E+13
	64508000,95	7199,529964	20,77252727	-3,62902E+14
	14442963,42	8667,383577	10,5236635	-2,49526E+13
	60961830,58	8566,117225	13,07664318	-4,05789E+13
	28462309,02	8308,67569	9,162225878	-4,25464E+14
	36976315,05	13307,6056	23,28921275	-9,50977E+14
	39715963,56	8771,027097	3,052715182	-3,096E+13
	44935700,4	5716,70829	17,2591143	-8,67861E+12
	17910703,58	7514,570441	23,15458006	-5,06678E+12
	20167953,8	10899,83955	9,9410516	-3,0859E+13
	41727786,42	9265,251771	14,81827846	-4,51635E+14

	34252690,82	9488,082087	20,98582711	-6,47803E+13
	31367607,37	9113,842424	5,470561645	-9,55206E+12
	49745675,9	10007,23525	23,48882351	-7,70071E+13
	34341427,17	11961,01453	28,89235443	-5,03617E+13
	45122687,89	7198,621659	19,03781268	-1,74033E+13
	32868222,94	8171,918005	14,36710844	-2,65327E+14
	60082964,02	5579,21868	15,55162787	-4,47961E+14
	28547074,48	6537,05181	12,79263579	-1,65316E+14
	46811073,64	12995,34339	11,62033368	-1,56456E+13
	33925356,78	13684,20988	23,33262049	-2,06259E+14
	32003778,98	10716,24855	18,79826309	-2,06429E+14
	30038897,13	9631,819672	13,6861249	-8,63419E+12
	29458435,76	8363,104137	17,41506392	-1,14983E+14
	36164084,14	8192,682681	17,7421645	-1,08E+15
	31808688,34	8049,276922	22,75482118	-1,12729E+14
	23318495,26	7281,428323	9,225859511	-3,16118E+14
	26184267,64	11226,91198	29,75069999	-1,44912E+13
	28598787,58	11143,10713	15,93735968	-7,55903E+13
	31839994,93	7892,577476	14,39138176	-4,60114E+13
	16562588,83	9850,190608	13,83010083	-3,5119E+12
	27983073,56	12911,08331	16,55701584	-5,30901E+13
	43377075,96	10919,19136	16,40107509	-1,74761E+13
	48271646,41	7394,721372	26,07306219	-1,54164E+13
	30707235,83	10748,11191	9,446979493	-5,09555E+13
	26446302,97	10097,46802	10,87975118	-3,03944E+13
	31572152,79	10386,50558	21,76861168	-3,65443E+14
	39906112,46	10671,56143	19,09566834	-6,33167E+13
	34163977,69	11158,11317	13,17748752	-3,55845E+13
	20219902,55	11379,83768	13,42926298	-5,52331E+13
Average	36230784,02	9245,407103	16,26933366	-4,96789E+14
Time (sec.)	36.398215	34.260816	39.848192	38.296503
Standard deviation	23953560,9	8536,988569	13,20342311	-1,2442E+14

Benchmarks	F9	F10
	258,0979337	16,55345832
	317,7198252	17,84102643
	326,4366022	16,88671087
	271,840735	18,5576418
	313,6743297	17,08951998
	287,4329563	17,92193367
	296,8872079	17,27490165
	328,2815628	17,6757543
	338,8108206	17,59891715
	302,6644709	18,18431168

	335,8797324	16,79149644
	289,6098219	17,90152294
	325,9611314	18,04878582
	293,2759957	18,12689942
	314,0611285	18,67311845
	313,6976531	16,39078544
	317,7057116	17,50248669
	303,3321739	18,09729328
	298,2096205	17,43238348
	291,3473503	18,24622233
	303,6843037	19,12726167
	326,5192706	18,66569313
	302,4454179	16,84063742
	293,9447624	18,34730958
	321,1580166	17,22078493
	266,4718039	17,83458343
	306,9147422	19,30483466
	306,1048553	17,81851503
	304,4021882	17,7472169
	297,7152492	18,76175188
	347,8148442	19,11101576
	342,1528993	17,37574353
	305,5416099	17,58645542
	306,2214125	17,94636558
	322,8212213	17,76603786
	308,3381654	17,92208294
	317,3604612	17,72880959
	330,3369969	18,44403851
	253,4210358	18,12715372
	269,7470898	18,31808456
	301,4928344	18,34098873
	295,1571118	18,19677122
	290,9818286	18,46947301
	313,0289033	17,84581843
	301,6748609	18,08838774
	316,6822948	16,52351724
	326,0613686	17,40181709
	248,8183174	17,82169818
	304,8681093	18,54314903
	269,7644216	17,92247124
	315,2031116	17,63437283
	308,1304439	19,00991421
	289,0138046	16,98407345
	302,4765654	16,92858593
	270,4225906	18,16453804

	291,904134	18,59927877
	307,283823	16,88795289
	291,9147515	16,36313669
	319,3065724	17,61214711
	307,189535	17,65212034
	290,3627862	17,416602
	315,0766863	16,75354625
	285,1819133	17,66461275
	325,3847842	17,4672183
	287,1551052	18,5798196
	306,2661718	17,43913638
	296,4433272	17,48972843
	327,5867538	18,44360419
	327,7888989	18,94393192
	290,8266461	17,40723542
	291,3375306	17,45217658
	311,8928617	18,14198221
	323,6491421	18,31062863
	316,5731521	16,54462208
	321,4257846	17,47899082
	324,4378116	18,41449486
	346,8340911	18,40068502
	323,9464247	17,55824549
	327,8085118	18,24548105
	330,4237117	17,07884972
	301,0490613	17,85013182
	301,508097	18,04246744
	281,1181316	17,44535207
	315,52292	17,69263529
	324,6646125	17,75693836
	308,5861801	17,27646293
	297,23255	17,43883592
	314,5321636	17,82188274
	294,5231877	17,58951157
	293,8637706	17,51160273
	314,0017459	18,43182498
	312,5225447	18,78335157
	311,5052156	15,96952674
	292,0234957	17,39580928
	346,6619681	15,27785982
	336,6464781	14,60663746
	283,4625911	18,173458
	304,7776014	17,66662728
	298,1762221	17,39490025
	327,3718328	16,09883935

Average	<b>306,5957096</b>	<b>17,71236006</b>
Time (sec.)	35.864105	37.467774
Standard deviation	258,0979337	16,55345832

## EK B

### B.1 ACO İle Aritmi Sınıflandırma Problemi MATLAB Kodları

#### B.1.1 EKG Sinyalini Üreten MATLAB Kodu

**complete.m**

```
x=0.01:0.01:2;
default=input('Press 1 if u want default ecg signal else press 2:\n');
if(default==1)
    li=30/72;

    a_pwav=0.25;
    d_pwav=0.09;
    t_pwav=0.16;

    a_qwav=0.025;
    d_qwav=0.066;
    t_qwav=0.166;

    a_qrswav=1.6;
    d_qrswav=0.11;

    a_swav=0.25;
    d_swav=0.066;
    t_swav=0.09;

    a_twav=0.35;
    d_twav=0.142;
    t_twav=0.2;

    a_uwav=0.035;
    d_uwav=0.0476;
    t_uwav=0.433;
else
    rate=input('\n\nenter the heart beat rate :');
    li=30/rate;

    %p wave specifications
    fprintf('\n\np wave specifications\n');
    d=input('Enter 1 for default specification else press 2: \n');
    if(d==1)
        a_pwav=0.25;
        d_pwav=0.09;
        t_pwav=0.16;
    else
        a_pwav=input('amplitude = ');
        d_pwav=input('duration = ');
        t_pwav=input('p-r interval = ');
        d=0;
    end
```

```

%q wave specifications
fprintf('\n\nq wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
    a_qwav=0.025;
    d_qwav=0.066;
    t_qwav=0.166;
else
    a_qwav=input('amplitude = ');
    d_qwav=input('duration = ');
    t_qwav=0.166;
    d=0;
end

%qrs wave specifications
fprintf('\n\nqrs wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
    a_qrswav=1.6;
    d_qrswav=0.11
else
    a_qrswav=input('amplitude = ');
    d_qrswav=input('duration = ');
    d=0;
end

%s wave specifications
fprintf('\n\ns wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
    a_swav=0.25;
    d_swav=0.066;
    t_swav=0.09;
else
    a_swav=input('amplitude = ');
    d_swav=input('duration = ');
    t_swav=0.09;
    d=0;
end

%t wave specifications
fprintf('\n\nnt wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
    a_twav=0.35;
    d_twav=0.142;
    t_twav=0.2;
else
    a_twav=input('amplitude = ');

```

```

d_twav=input('duration = ');
t_twav=input('s-t interval = ');
d=0;
end

%u wave specifications
fprintf('\n\nu wave specifications\n');
d=input('Enter 1 for default specification else press 2: \n');
if(d==1)
    a_uwav=0.035;
    d_uwav=0.0476;
    t_uwav=0.433;
else
    a_uwav=input('amplitude = ');
    d_uwav=input('duration = ');
    t_uwav=0.433;
    d=0;
end
end

pwav=p_wav(x,a_pwav,d_pwav,t_pwav,li);

%qwav output
qwav=q_wav(x,a_qwav,d_qwav,t_qwav,li);

%qrswav output
qrswav=qrs_wav(x,a_qrswav,d_qrswav,li);

%swav output
swav=s_wav(x,a_swav,d_swav,t_swav,li);

%twav output
twav=t_wav(x,a_twav,d_twav,t_twav,li);

%uwav output
uwav=u_wav(x,a_uwav,d_uwav,t_uwav,li);

%ecg output
ecg=pwav+qrswav+twav+swav+qwav+uwav;
figure(1)
plot(x,ecg);

```

### **p\_wav.m**

```

function [pwav]=p_wav(x,a_pwav,d_pwav,t_pwav,li)
l=li;
a=a_pwav;
x=x+t_pwav;
b=(2*l)/d_pwav;
n=100;
p1=1/l;

```

```

p2=0;
for i = 1:n
    harm1=((sin((pi/(2*b))*(b-(2*i))))/(b-
(2*i))+sin((pi/(2*b))*(b+(2*i)))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    p2=p2+harm1;
end
pwave1=p1+p2;
pwave=a*pwave1;

```

### **qrs\_wav.m**

```

function [qrswav]=qrs_wav(x,a_qrswav,d_qrswav,li)
l=li;
a=a_qrswav;
b=(2*l)/d_qrswav;
n=100;
qrs1=(a/(2*b))*(2-b);
qrs2=0;

for i = 1:n
    harm=((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b))*cos((i*pi*x)/l);
    qrs2=qrs2+harm;
end
qrswav=qrs1+qrs2;

```

### **q\_wav.m**

```

function [qwave]=q_wav(x,a_qwave,d_qwave,t_qwave,li)
l=li;
x=x+t_qwave;
a=a_qwave;
b=(2*l)/d_qwave;
n=100;
q1=(a/(2*b))*(2-b);
q2=0;
for i = 1:n
    harm5=((2*b*a)/(i*i*pi*pi))*(1-cos((i*pi)/b))*cos((i*pi*x)/l);
    q2=q2+harm5;
end
qwave=-1*(q1+q2);

```

### **s\_wav.m**

```

function [swave]=s_wav(x,a_swave,d_swave,t_swave,li)
l=li;
x=x-t_swave;
a=a_swave;
b=(2*l)/d_swave;
n=100;
s1=(a/(2*b))*(2-b);
s2=0;

```

```

for i = 1:n
    harm3=((((2*b*a)/(i*pi*pi))*(1-cos((i*pi)/b)))*cos((i*pi*x)/l));
    s2=s2+harm3;
end
swav=-1*(s1+s2);

```

### **t\_wav.m**

```

function [twav]=t_wav(x,a_twav,d_twav,t_twav,li)
l=li;
a=a_twav;
x=x-t_twav-0.045;
b=(2*l)/d_twav;
n=100;
t1=1/l;
t2=0;

for i = 1:n
    harm2=(((sin((pi/(2*b))*(b-(2*i)))/(b-
(2*i))+sin((pi/(2*b))*(b+(2*i)))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    t2=t2+harm2;
end
twav1=t1+t2;
twav=a*twav1;

```

### **u\_wav.m**

```

function [uwav]=u_wav(x,a_uwav,d_uwav,t_uwav,li)
l=li;
a=a_uwav
x=x-t_uwav;
b=(2*l)/d_uwav;
n=100;
u1=1/l
u2=0
for i = 1:n
    harm4=(((sin((pi/(2*b))*(b-(2*i)))/(b-
(2*i))+sin((pi/(2*b))*(b+(2*i)))/(b+(2*i)))*(2/pi))*cos((i*pi*x)/l);
    u2=u2+harm4;
end
uwav1=u1+u2;
uwav=a*uwav1;

```

### B.1.2 ACO ile Aritmi Sınıflama Problemi MATLAB Kodu

**aco.m**

```
clc;  
clear;  
close all;
```

```
%% Problem Definition
```

```
model=CreateModel();  
CostFunction=@(x) MyCost(x,model);  
nVar=model.n;
```

```
%% ACO Parameters
```

```
MaxIt=300; % Maximum Number of Iterations  
nAnt=50; % Number of Ants (Population Size)  
Q=1;  
tau0=0.1; % Initial Phromone  
alpha=1; % Phromone Exponential Weight  
beta=0.02; % Heuristic Exponential Weight  
rho=0.1; % Evaporation Rate
```

```
%% Initialization
```

```
N=[0 1];  
eta=[model.ar1d model.ar2d model.ar3d];  
tau=tau0*ones(2,nVar); % Phromone Matrix  
BestCost=zeros(MaxIt,1); % Array to Hold Best Cost Values
```

```
% Empty Ant  
empty_ant.Tour=[];  
empty_ant.x=[];  
empty_ant.Cost=[];  
empty_ant.Sol=[];
```

```
% Ant Colony Matrix  
ant=repmat(empty_ant,nAnt,1);
```

```
% Best Ant  
BestSol.Cost=inf;
```

```
%% ACO Main Loop
```

```
for it=1:MaxIt
```

```
% Move Ants  
for k=1:nAnt
```

```

ant(k).Tour=[];
for l=1:nVar
    P=tau(:,l).^alpha.*eta(:,l).^beta;
    P=P/sum(P);
    j=RouletteWheelSelection(P);
    ant(k).Tour=[ant(k).Tour j];
end
ant(k).x=N(ant(k).Tour);
[ant(k).Cost, ant(k).Sol]=CostFunction(ant(k).x);
if ant(k).Cost<BestSol.Cost
    BestSol=ant(k);
end
% Update Phromones
for k=1:nAnt
    tour=ant(k).Tour;
    for l=1:nVar
        tau(tour(l),l)=tau(tour(l),l)+Q/ant(k).Cost;
    end
end
% Evaporation
tau=(1-rho)*tau;
% Store Best Cost
BestCost(it)=BestSol.Cost;
% Show Iteration Information
if BestSol.Sol.IsFeasible
    FeasibilityFlag = '*';
else
    FeasibilityFlag = "";
end

```

```
%disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it)) ' ' FeasiblityFlag]);  
end
```

### CreateModel.m

```
function model=CreateModel()  
  
ar1d=[0.7248 1.090 1.021 0.8999 0.8985 0.7830 1.049 1.531 0.8264 0.9004 1.211 0.935 1.813  
1.03 1.119 0.8992 1.131 0.9002 1.038 0.8999];  
ar2d=[1.625 2.652 2.190 2.786 1.421 1.739 2.250 1.640 1.837 0.768 1.681 1.972 0.624 1.250  
1.282 0.8145 1.374 0.702 1.015 1.657];  
ar3d=[1.225 1.649 1.423 1.912 0.926 1.541 1.076 0.840 1.943 1.061 1.203 1.581 1.128 1.613  
1.167 0.9146 1.282 1.215 1.683 1.284];  
orard= [0.6189 1.242 1.206 1.097 0.954 1.251 0.873 1.789 0.987 1.304 1.004 1.063 1.206 1.003  
1.435 1.203 1.008 0.9964 0.8796 1.114];  
n=size(ar1d,2);  
m=size(ar2d,2);  
o=size(ar3d,2);  
p=size(orard,2);  
ar1s=zeros(1,n);  
ar2s=zeros(1,m);  
ar3s=zeros(1,o);  
son=zeros(0,3);  
t=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0];  
model.n=n;  
model.m=m;  
model.o=o;  
model.p=p;  
model.ar1d=ar1d;  
model.ar2d=ar2d;  
model.ar3d=ar3d;  
model.orard=orard;  
model.ar1s=ar1s;  
model.ar2s=ar2s;  
model.ar3s=ar3s;  
model.son=son;  
model.t=t;  
  
figure(1)  
scatter(model.t,model.ar1d,'*');  
title('Sinüs Taşikardisi ve Örnek Aritmi Benzeşimi');  
hold on  
scatter(model.t,model.orard);  
figure(2)  
scatter(model.t,model.ar2d,'*');  
title('Premature Atrial Kontraksiyon(PAC) Taşikardisi ve Örnek Aritmi Benzeşimi');  
hold on
```

```

scatter(model.t,model.orard);
figure(3)
scatter(model.t,model.ar3d,'*');
title('Multifokal Atrial Taşikardive Örnek Aritmi Benzesimi');
hold on
scatter(model.t,model.orard);
end

```

### **MyCost.m**

```

function [z, sol]=MyCost(x,model)

ar1d=model.ar1d;
ar2d=model.ar2d;
ar3d=model.ar3d;
orard=model.orard;
ar1s=model.ar1s;
ar2s=model.ar2s;
ar3s=model.ar3s;

for i=1:20
    ar1s(1,i)=abs(ar1d(1,i)-orard(1,i));
    ar2s(1,i)=abs(ar2d(1,i)-orard(1,i));
    ar3s(1,i)=abs(ar3d(1,i)-orard(1,i));
end
X=sum(ar1s);
Y=sum(ar2s);
Z=sum(ar3s);
son=[X Y Z];

z=min(son);

Violation=0;
sol.X=X;
sol.Y=Y;
sol.Z=Z;
sol.Violation=Violation;
sol.z=z;
sol.IsFeasible=(Violation==0);

end

```

### **RouletteWheelSelection.m**

```

function j=RouletteWheelSelection(P)

r=rand;

C=cumsum(P);

j=find(r<=C,1,'first');

```

end

## B.2 Ant Colony Optimization Algoritmasının İki Boyutlu Benchmark Fonksiyonları Tablosu

Benchmarks	F1	F2	F3	F4
	4.223621	6.20E+00	3.95E+00	3.23E+00
	13.103351	4.62E+00	3.73E+00	1.02E+00
	5.460857	8.62E+00	4.89E+00	1.71E+00
	2.115738	1.01E+01	7.03E+00	5.84E-01
	5.967971	3.51E+00	1.99E+00	1.45E+00
	3.401852	8.97E+00	3.98E+00	2.36E+00
	5.657244	3.33E+00	6.37E+00	4.19E+00
	0.689972	1.71E+00	2.96E+00	1.75E+00
	3.734837	4.75E+00	3.92E+00	5.98E+00
	13.799869	1.44E+00	1.49E+00	5.11E+00
	10.879382	9.39E+00	7.16E+00	2.95E+00
	4.954616	6.06E+00	5.13E+00	5.28E+00
	1.227635	5.18E+00	6.20E+00	4.02E+00
	9.032475	4.97E+00	5.11E+00	2.00E+00
	6.325395	3.92E+00	7.15E+00	6.13E+00
	13.541512	4.00E+00	4.49E+00	1.39E+00
	4.012396	9.44E+00	5.87E+00	1.97E+00
	3.776732	3.33E+00	7.02E+00	4.50E+00
	3.998868	3.55E+00	3.73E+00	2.51E+00
	1.0906	2.90E+00	4.82E+00	4.12E+00
	6.172649	6.76E+00	7.09E+00	1.47E+00
	11.514801	1.14E+01	6.44E+00	3.10E+00
	12.006205	1.03E+01	3.72E+00	1.87E+00
	4.989093	6.08E+00	5.74E+00	2.20E+00
	3.586892	8.73E+00	2.38E+00	3.17E+00
	13.641887	2.13E+00	5.79E+00	4.05E+00
	0.307236	1.08E+01	2.09E+00	5.11E+00
	8.248673	1.15E+01	6.54E+00	1.81E+00
	3.881036	9.98E+00	1.77E+00	5.85E+00
	10.052813	8.91E+00	3.19E+00	3.25E+00
	8.161391	2.00E+00	2.21E+00	3.58E+00
	12.22223	7.20E+00	7.03E+00	6.05E+00
	10.264037	3.84E+00	3.46E+00	1.01E+00
	5.065029	8.25E+00	1.96E+00	3.15E+00
	2.234007	2.04E+00	3.26E+00	3.04E+00
	13.606537	1.88E+00	3.49E+00	2.65E+00
	6.509998	2.01E+00	3.69E+00	1.60E+00
	4.208783	1.18E+01	6.23E+00	4.02E+00
	2.417737	7.92E+00	5.14E+00	6.24E+00
	7.724977	3.33E+00	5.04E+00	5.16E+00
	6.098022	1.70E+00	6.40E+00	4.71E+00
	4.453176	5.27E+00	4.94E+00	4.56E+00
	12.840194	2.24E+00	5.39E+00	2.95E+00
	4.239333	4.99E+00	1.20E+00	1.09E+00

	6.860875	4.65E+00	2.32E+00	3.41E+00
	10.492719	9.64E+00	4.74E+00	2.48E+00
	3.395306	5.06E+00	1.70E+00	1.49E+00
	5.689538	3.85E+00	3.60E+00	5.48E+00
	14.259851	9.28E+00	6.17E+00	7.96E-01
	2.458324	1.14E+01	4.56E+00	6.16E+00
	8.389636	6.10E+00	5.17E+00	3.19E+00
	6.500834	1.10E+01	1.23E+00	2.54E+00
	3.240379	2.70E+00	5.82E+00	5.75E+00
	2.089554	4.67E+00	2.63E+00	1.01E+00
	1.323646	6.58E+00	4.61E+00	1.05E+00
	5.765911	1.08E+01	6.92E+00	3.55E+00
	10.383615	7.04E+00	3.37E+00	4.70E+00
	5.422452	3.54E+00	1.89E+00	5.35E+00
	12.810081	5.94E+00	2.98E+00	4.30E+00
	5.639351	7.45E+00	4.01E+00	3.48E+00
	11.803272	6.97E+00	2.19E+00	9.75E-01
	0.527189	1.11E+01	7.14E+00	6.19E+00
	7.646858	7.27E+00	5.30E+00	4.88E+00
	2.947545	4.89E+00	1.45E+00	5.46E+00
	7.308637	1.60E+00	3.91E+00	4.03E+00
	9.96422	3.71E+00	4.18E+00	4.11E+00
	2.340928	9.99E+00	5.77E+00	5.60E+00
	3.056648	2.10E+00	6.24E+00	1.99E+00
	9.18522	7.70E+00	3.54E+00	1.08E+00
	4.676184	6.36E+00	5.79E+00	4.72E+00
	10.614478	8.04E+00	6.18E+00	2.45E+00
	6.398712	5.56E+00	5.53E+00	5.16E+00
	12.904782	1.99E+00	4.17E+00	5.78E+00
	0.79733	5.89E+00	2.31E+00	3.08E+00
	12.16986	6.53E+00	3.27E+00	5.32E+00
	11.000269	8.49E+00	5.22E+00	2.97E+00
	5.624513	6.98E+00	2.63E+00	4.18E+00
	0.190713	1.15E+01	5.30E+00	5.04E+00
	4.162523	8.52E+00	1.78E+00	1.38E+00
	10.013099	4.04E+00	3.93E+00	4.89E+00
	3.985339	8.38E+00	4.34E+00	4.63E+00
	0.39321	9.71E+00	3.89E+00	2.16E+00
	1.752641	7.11E+00	2.88E+00	4.56E+00
	13.291883	5.01E+00	3.94E+00	5.84E-01
	5.46653	8.28E+00	6.44E+00	5.97E+00
	6.487741	1.15E+01	3.96E+00	2.96E+00
	4.278609	1.21E+01	6.43E+00	7.31E-01
	0.514969	3.84E+00	5.21E+00	9.04E-01
	2.108756	2.31E+00	1.25E+00	2.18E+00
	13.17405	5.59E+00	6.13E+00	1.99E+00
	7.929219	2.37E+00	1.29E+00	3.04E+00
	6.499524	1.20E+01	4.71E+00	2.71E+00
	12.361446	7.24E+00	6.47E+00	3.05E+00
	3.960027	4.99E+00	1.56E+00	1.13E+00

	9.957674	1.03E+01	6.39E+00	3.33E+00
	3.098981	1.34E+00	2.11E+00	5.35E+00
	8.167064	4.80E+00	3.20E+00	6.17E+00
	7.247976	1.12E+01	6.03E+00	6.02E+00
	13.816016	8.05E+00	2.63E+00	4.48E+00
	7.142363	9.11E+00	3.53E+00	3.00E+00
Average	6.61E+00	6.41E+00	4.31E+00	3.43E+00
Time (sec.)	29.6698	41.482036	98.872015	42.6327
Standard deviation	3.991258112	3.106056753	1.729378739	1.671871239

Benchmarks	F5	F6	F7	F8
	25.162712	1.36E+00	0.025067	-6037.51416
	19.090609	1.376136	0.024284	-5896.338379
	18.682522	1.56E+00	0.01677	-5930.921875
	27.2829	1.431048	0.036103	-5321.253906
	20.587637	1.004738	0.025166	-6101.985352
	19.512272	1.063805	0.027956	-4851.241699
	26.403942	1.004271	0.040277	-6072.824707
	28.225914	1.510193	0.00716	-6685.335938
	28.538553	0.793591	0.039397	-5210.363281
	19.816427	2.075322	0.006011	-4716.47998
	19.837214	1.58E+00	0.025539	-5362.978516
	31.252632	0.694461	0.023045	-6118.450684
	31.387953	1.897607	0.03364	-4949.237793
	20.248695	1.576591	0.026866	-6869.69043
	24.097103	1.348868	0.011345	-6173.730469
	29.41539	1.766446	0.027565	-6238.268066
	18.545927	1.348774	0.016269	-6180.342773
	28.720539	7.47E-01	0.038115	-5332.627441
	24.469133	0.806945	0.013284	-5200.841309
	18.942986	1.01E+00	0.012852	-5921.598145
	26.908749	0.823754	0.03638	-5295.597656
	31.557213	0.807272	0.039467	-4777.115723
	28.104588	1.267388	0.017883	-5802.044922
	31.359531	1.773964	0.001418	-4728.315918
	18.435209	1.301661	0.019239	-6134.783203
	21.936615	1.64E+00	0.012626	-6711.19043
	22.134296	1.514535	0.01522	-6623.972168
	31.513096	0.949966	0.034628	-4789.87793
	20.994452	1.05E+00	0.029401	-4839.207031
	30.857269	1.14E+00	0.030899	-5352.266602
	21.515802	1.465181	0.020522	-6669.928711
	19.035461	1.89E+00	0.031749	-6710.132324
	20.330992	1.37E+00	0.028242	-5671.449219
	29.008152	1.856751	0.016557	-5442.724609
	31.548304	8.08E-01	0.014885	-5209.768066
	24.378777	1.070482	0.013483	-5774.603516
	29.294491	0.879786	0.000727	-5572.85791

	19.686621	1.63944	0.015664	-6011.064453
	21.058931	1.280555	0.040034	-6347.836426
	31.144882	6.25E-01	0.025932	-6683.550293
	30.681225	1.06203	0.00578	-5488.681152
	24.347385	0.807785	0.011462	-5669.796387
	23.104458	2.11E+00	0.011596	-6651.545898
	30.165388	1.504216	0.015116	-6768.850586
	25.231434	8.30E-01	0.009272	-5734.598145
	30.703707	1.103447	0.02414	-5953.337891
	28.287846	1.870525	0.014004	-5991.822266
	28.547039	1.96E+00	0.003763	-4912.539063
	20.138824	1.334673	0.029134	-6597.324219
	21.985825	1.386222	0.005641	-5451.916016
	23.829853	0.854665	0.03203	-4828.296387
	24.531916	2.073221	0.007895	-5627.54248
	22.801998	1.77E+00	0.0047	-6113.755859
	32.176979	2.003321	0.033545	-6746.963867
	24.2057	1.937203	0.003962	-6425.202148
	21.589615	2.123977	0.004278	-6181.136719
	28.086773	1.325614	0.001017	-5386.056152
	26.108694	1.427686	0.028274	-5068.592773
	29.477325	0.956083	0.004688	-6730.829102
	23.097672	1.814493	0.036343	-5559.566406
	22.119873	1.771395	0.038355	-4709.933594
	21.867893	1.661152	0.025861	-6682.558594
	21.229887	1.296197	0.021588	-5573.519043
	25.419781	1.835786	0.038317	-5819.435547
	18.69355	1.394954	0.035741	-5208.511719
	31.977177	1.412044	0.027359	-4759.129883
	29.557499	0.886183	0.025417	-6740.086914
	20.244453	0.994278	0.033999	-5622.451172
	26.190567	7.54E-01	0.012021	-5747.228027
	32.297878	1.193752	0.010213	-6232.382813
	19.588629	1.57561	0.002203	-5921.532227
	23.986385	1.168118	0.005494	-6335.206543
	30.612503	1.768407	0.0175	-5490.466797
	21.638823	0.883195	0.016528	-4900.967285
	20.207123	1.35E+00	0.038878	-5594.811035
	20.015804	1.018279	0.007065	-6655.57959
	31.62763	0.770851	0.020898	-6207.983398
	19.598385	1.64565	0.0097	-5113.755371
	21.059355	9.87E-01	0.006814	-5990.037109
	26.126936	1.453134	2.26E-02	-6148.272949
	22.612377	1.371981	0.037321	-5191.121094
	23.929117	1.945515	0.023784	-6457.735352
	18.794937	0.916627	0.032963	-4712.115723
	20.09598	0.708188	0.019953	-6650.554199
	21.106018	1.873374	0.031254	-5388.568848
	25.158047	1.078513	0.012667	-6122.880859
	30.543781	1.525508	0.002614	-6167.382813

	19.147028	2.039789	0.025686	-5930.32666
	26.75264	1.45E+00	0.002148	-5452.841797
	30.251503	1.476153	0.004679	-5198.593262
	24.664692	1.024256	0.014833	-6195.948242
	32.282181	1.184741	0.0234	-5059.203125
	27.052132	1.231107	0.030414	-4975.753906
	26.247833	1.79675	0.010322	-6239.39209
	31.578423	1.065906	0.034555	-5061.583496
	28.381596	1.064599	0.027355	-6547.862793
	25.289551	1.71E+00	0.000675	-4937.070801
	30.254896	1.124133	0.004365	-5689.170898
	23.315289	1.991041	0.012349	-5532.984863
	28.240335	9.97E-01	0.01271	-5205.999023
Average	2.51E+01	1.35E+00	1.99E-02	-5.76E+03
Time (sec.)	24.36721	43.06987	39.50791	41.96108
Standard deviation	4.38446911	0.400708866	0.011737782	621.1799698

Benchmarks	F9	F10
	69.645508	1.38E+00
	81.929893	1.03E+00
	70.472183	1.62E+00
	4.85E+01	1.61E+00
	39.486187	1.39E+00
	48.274078	1.41E+00
	41.633305	1.16E+00
	81.10965	1.14E+00
	63.123749	1.28E+00
	49.88723	9.42E-01
	89.406998	1.22E+00
	76.075592	1.59E+00
	39.786945	1.07E+00
	54.580322	1.26E+00
	75.398483	9.81E-01
	73.113052	1.61E+00
	54.451656	1.58E+00
	64.577675	9.55E-01
	38.350712	1.08E+00
	5.69E+01	1.64E+00
	87.576721	1.58E+00
	45.271336	9.87E-01
	71.926109	1.48E+00
	6.68E+01	1.04E+00
	8.77E+01	1.24E+00
	8.30E+01	1.21E+00
	42.413342	1.64E+00
	48.968876	9.77E-01

	79.002747	1.01E+00
	68.590446	1.54E+00
	43.962158	1.61E+00
	78.420532	1.65E+00
	55.620911	1.06E+00
	74.435104	1.62E+00
	38.265469	1.52E+00
	76.152786	1.30E+00
	72.008133	1.47E+00
	89.260643	1.54E+00
	50.609367	1.12E+00
	7.09E+01	1.05E+00
	70.64267	1.19E+00
	67.651184	1.09E+00
	54.327816	1.31E+00
	78.102081	1.49E+00
	67.491959	1.48E+00
	6.53E+01	1.55E+00
	72.360359	1.06E+00
	65.132545	1.22E+00
	77.814194	1.03E+00
	46.46793	9.36E-01
	49.567173	1.49E+00
	77.646927	1.54E+00
	86.918915	1.05E+00
	8.39E+01	1.41E+00
	77.182114	1.38E+00
	47.013153	1.62E+00
	8.08E+01	1.38E+00
	62.81012	1.35E+00
	48.761402	1.22E+00
	38.543709	1.10E+00
	72.06282	9.77E-01
	83.012299	1.04E+00
	66.835762	1.59E+00
	39.482971	1.62E+00
	84.205673	1.50E+00
	40.092529	1.28E+00
	47.17881	1.63E+00
	46.017597	1.44E+00
	81.199715	1.38E+00
	52.534531	1.60E+00
	50.733208	1.51E+00
	75.454773	1.23E+00
	69.512016	1.46E+00
	57.528381	1.00E+00

	53.600853	1.05E+00
	72.275116	1.16E+00
	43.923557	1.20E+00
	85.370102	1.34E+00
	52.725922	1.41E+00
	72.041908	1.47E+00
	5.58E+01	1.65E+00
	61.296688	1.46E+00
	60.889782	1.46E+00
	64.830177	1.32E+00
	75.905106	1.38E+00
	6.53E+01	1.41E+00
	45.438599	1.41E+00
	8.03E+01	1.61E+00
	88.5578	1.59E+00
	56.613243	1.10E+00
	49.48354	9.90E-01
	41.400097	1.33E+00
	88.787796	1.55E+00
	46.709179	1.06E+00
	75.642944	1.09E+00
	67.141342	1.21E+00
	50.62545	1.10E+00
	42.887798	1.28E+00
	76.535568	1.27E+00
	85.997345	1.31E+00
Average	6.39E+01	1.32E+00
Time (sec.)	62.18904	21.69207
Standard deviation	15.29510979	0.220097972

## EK C

### C.1 PSO İle Sensör Yerleştirme Problemi MATLAB Kodları

#### Sphere.m

```
function z = Sphere(x)  
  
z = sum(x.^2);  
  
end
```

#### PSO.m

```
function out = PSO(problem, params)  
  
%% Problem Definition  
  
CostFunction = problem.CostFunction; % Cost Function  
  
nVar = problem.g; % Number of Unknown (Decision) Variables  
  
VarSize = [1 nVar]; % Matrix Size of Decision Variables  
  
VarMin = problem.VarMin; % Lower Bound of Decision Variables  
VarMax = problem.VarMax; % Upper Bound of Decision Variables  
  
%% Parameters of PSO  
  
MaxIt = params.MaxIt; % Maximum Number of Iterations  
  
nPop = params.nPop; % Population Size (Swarm Size)  
  
w = params.w; % Inertia Coefficient  
wdamp = params.wdamp; % Damping Ratio of Inertia Coefficient  
c1 = params.c1; % Personal Acceleration Coefficient  
c2 = params.c2; % Social Acceleration Coefficient  
  
% The Flag for Showing Iteration Information  
ShowIterInfo = params.ShowIterInfo;  
  
MaxVelocity = 0.2*(VarMax-VarMin);  
MinVelocity = -MaxVelocity;  
  
%% Initialization
```

```

% The Particle Template
empty_particle.Position = [];
empty_particle.Velocity = [];
empty_particle.Cost = [];
empty_particle.Best.Position = [];
empty_particle.Best.Cost = [];

% Create Population Array
particle = repmat(empty_particle, nPop, 1);

% Initialize Global Best
GlobalBest.Cost = inf;

% Initialize Population Members
for i=1:nPop

    % Generate Random Solution
    particle(i).Position = unifrnd(VarMin, VarMax, VarSize);

    % Initialize Velocity
    particle(i).Velocity = zeros(VarSize);

    % Evaluation
    particle(i).Cost = CostFunction(particle(i).Position);

    % Update the Personal Best
    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;

    % Update Global Best
    if particle(i).Best.Cost < GlobalBest.Cost
        GlobalBest = particle(i).Best;
    end

end

% Array to Hold Best Cost Value on Each Iteration
BestCosts = zeros(MaxIt, 1);

%% Main Loop of PSO

for it=1:MaxIt

    for i=1:nPop

        % Update Velocity

```

```

particle(i).Velocity = w*particle(i).Velocity ...
+ c1*rand(VarSize).*(particle(i).Best.Position - particle(i).Position) ...
+ c2*rand(VarSize).*(GlobalBest.Position - particle(i).Position);

% Apply Velocity Limits
particle(i).Velocity = max(particle(i).Velocity, MinVelocity);
particle(i).Velocity = min(particle(i).Velocity, MaxVelocity);

% Update Position
particle(i).Position = particle(i).Position + particle(i).Velocity;

% Apply Lower and Upper Bound Limits
particle(i).Position = max(particle(i).Position, VarMin);
particle(i).Position = min(particle(i).Position, VarMax);

% Evaluation
particle(i).Cost = CostFunction(particle(i).Position);

% Update Personal Best
if particle(i).Cost < particle(i).Best.Cost

    particle(i).Best.Position = particle(i).Position;
    particle(i).Best.Cost = particle(i).Cost;

% Update Global Best
if particle(i).Best.Cost < GlobalBest.Cost
    GlobalBest = particle(i).Best;
end

end

% Store the Best Cost Value
BestCosts(it) = GlobalBest.Cost;

% Display Iteration Information
if ShowIterInfo
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCosts(it))]);
end

% Damping Inertia Coefficient
w = w * wdamp;

end

out.pop = particle;

```

```
out.BestSol = GlobalBest;
out.BestCosts = BestCosts;

end
```

### **psol.m**

*%% Problem Definiton*

```
problem.CostFunction = @(x) Sphere(x); % Cost Function
I = imread('arazi.jpg');
g = rgb2gray(I);
problem.g = 100; % Number of Unknown (Decision)
```

#### *Variables*

```
problem.VarMin = -10; % Lower Bound of Decision Variables
problem.VarMax = 10; % Upper Bound of Decision Variables
```

*%% Parameters of PSO*

```
params.MaxIt = 1000; % Maximum Number of Iterations
params.nPop = 50; % Population Size (Swarm Size)
params.w = 1; % Inertia Coefficient
params.wdamp = 0.99; % Damping Ratio of Inertia Coefficient
params.c1 = 2; % Personal Acceleration Coefficient
params.c2 = 2; % Social Acceleration Coefficient
params.ShowIterInfo = true; % Flag for Showing Iteration Information
```

*%% Calling PSO*

```
out = PSO(problem, params);
```

```
BestSol = out.BestSol;
BestCosts = out.BestCosts;
```

*%% Results*

```
figure;
% plot(BestCosts, 'LineWidth', 2);
semilogy(BestCosts, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

### **psol2.m**

*%% Problem Definiton*

```
problem.CostFunction = @(x) Sphere(x); % Cost Function
```

```

I = imread('arazi.jpg');
g = rgb2gray(I);
problem.g = 689;                                % Number of Unknown (Decision)

Variables
problem.VarMin = -10;                            % Lower Bound of Decision Variables
problem.VarMax = 10;                             % Upper Bound of Decision Variables

%% Parameters of PSO

% Constriction Coefficients
kappa = 1;
phi1 = 2.05;
phi2 = 2.05;
phi = phi1 + phi2;
chi = 2*kappa/abs(2-phi-sqrt(phi^2-4*phi));

params.MaxIt = 1000;                            % Maximum Number of Iterations
params.nPop = 50;                             % Population Size (sensor Size)
params.w = chi;                               % Intertia Coefficient
params.wdamp = 1;                            % Damping Ratio of Inertia Coefficient
params.c1 = chi*phi1;                         % Personal Acceleration Coefficient
params.c2 = chi*phi2;                         % Social Acceleration Coefficient
params.ShowIterInfo = true;                    % Flag for Showing Iteration Information

%% Calling PSO

out = PSO(problem, params);

BestSol = out.BestSol;
BestCosts = out.BestCosts;

%% Results

figure;
% plot(BestCosts, 'LineWidth', 2);
semilogy(BestCosts, 'LineWidth', 2);
xlabel('tarama alani');
ylabel('sensor sayısı');
grid on;

```

**C.2 Particle Swarm Optimization Algoritmasının İki Boyutlu Benchmark Fonksiyonları Tablosu**

Benchmarks	F1	F2	F3	F4
	2,12E-49	0,000153781	0,115853278	0,050004436
	2,12E-49	0,000647988	0,040104495	0,059854036
	6,19E-42	0,000594431	0,30191519	0,091075263
	6,81E-46	0,002260072	0,092430342	0,049259179
	2,27E-50	0,000692786	0,030051944	0,197539774
	5,69E-48	0,015474844	0,146385444	0,067108918
	2,05E-41	0,000933859	0,031142274	0,062599763
	1,29E-48	0,003180995	0,086004932	0,035209722
	1,25E-47	0,000121762	0,044776291	0,066882556
	1,04E-48	0,000594748	0,030858501	0,169166418
	2,99E-45	0,000966899	2,738170455	0,078684002
	3,34E-45	0,000445972	0,070166077	0,183714909
	4,67E-50	9,84E-06	0,08341844	0,154191363
	6,66E-49	0,000527767	0,125121715	0,110510069
	7,72E-44	5,20E-06	0,028471204	0,125078104
	3,31E-46	0,000224604	0,037741816	0,159755071
	1,47E-45	0,001973071	0,158615032	0,070574268
	5,79E-48	2,16E-05	0,016877109	0,053077288
	2,94E-50	0,012014439	0,105479353	0,132620829
	3,39E-49	0,000618902	0,026487188	0,102564287
	4,01E-48	0,000112282	0,028233674	0,154242892
	1,88E-44	8,56E-05	0,014857908	0,090331769
	2,77E-45	0,000640503	0,122796758	0,221052587
	1,55E-45	0,001436255	0,335284445	0,158908074
	2,65E-44	0,010023865	0,151164535	0,212172659
	2,50E-44	0,014704821	0,168981146	0,030282337
	7,51E-46	0,002698794	0,097279369	0,090864356
	5,47E-47	0,001549712	0,072436887	0,118582798
	1,91E-47	0,000999092	0,065907276	0,092861922
	1,98E-44	0,000477961	0,028121009	0,440080102
	3,24E-50	0,00015473	0,039395319	0,045674139
	1,36E-45	6,10E-05	0,150958618	0,06307716
	6,35E-52	0,001506072	0,032149964	0,143815198
	1,23E-48	0,098367982	0,111811156	0,080388714
	4,58E-46	0,000919655	0,032436918	0,234056348
	4,91E-48	0,017288379	0,123163099	0,049128799
	1,11E-46	2,75E-05	0,054354363	0,167362435
	2,66E-52	0,00049201	0,050144071	0,056694094
	3,69E-43	0,086884721	0,250879529	0,135791869
	2,92E-43	0,000206234	0,056718503	0,219445085
	1,26E-40	4,01E-06	0,267094604	0,080413727

	6,59E-46	4,27E-05	0,038466068	0,170892255
	6,87E-49	0,003329356	0,053327118	0,071375669
	5,00E-51	0,000206653	0,010403385	0,108103785
	1,78E-48	0,000575214	0,015066031	0,141357173
	6,91E-51	0,008400296	0,032777457	0,144921982
	5,79E-41	4,54E-05	0,01454854	0,323806255
	1,09E-51	6,72E-05	0,051983671	0,055444242
	9,62E-45	0,000573024	0,074438161	0,035545935
	3,80E-46	0,000656915	0,057350971	0,204536222
	8,89E-47	0,003966167	0,107599441	0,145032197
	1,48E-43	0,051134087	0,050996093	0,156292477
	1,61E-46	0,009135729	0,056249119	0,130654404
	1,81E-47	0,000423368	0,01650109	0,185628165
	1,10E-44	3,31E-05	0,138664662	0,06570993
	3,58E-51	0,000145481	0,036655647	0,124267281
	1,78E-50	0,00061721	0,077212742	0,113605617
	1,61E-49	0,000233357	0,126979675	0,098533153
	3,25E-47	0,005557693	0,027606899	0,218129361
	2,33E-49	0,004509464	0,106033671	0,020672622
	6,40E-42	0,001579716	0,060576387	0,050958448
	2,92E-45	0,018461757	0,090004521	0,089212799
	5,29E-50	0,00026824	0,007703595	0,2588715
	5,28E-50	0,001116349	0,010977889	0,106754005
	1,24E-46	0,00057778	0,050882963	0,021864571
	7,94E-38	0,000218621	0,019784558	0,011043647
	3,98E-50	0,002778402	0,087847656	0,118290052
	1,90E-51	0,004095319	0,130196961	0,053571867
	2,15E-46	8,67E-07	0,050515419	0,111554112
	2,37E-51	0,003898422	0,028658105	0,039379401
	6,94E-51	0,01121029	0,022060698	0,227909869
	7,14E-48	8,52E-05	0,062696104	0,113020023
	3,48E-51	0,003896751	0,033110941	0,094056443
	2,02E-47	6,83E-06	0,157486006	0,05886083
	1,40E-46	4,24E-05	0,042517218	0,08660368
	4,08E-45	0,004982876	0,043762611	0,165040431
	2,76E-47	0,000735295	0,269506541	0,160751177
	2,31E-50	1,91E-05	0,037484114	0,044593757
	7,70E-45	2,53E-06	0,088728679	0,140416684
	2,09E-50	0,000638549	0,016234347	0,128267708
	1,49E-48	0,000251541	0,049468301	0,022402679
	1,91E-46	0,114143319	0,549984164	0,031494568
	5,75E-49	0,00328588	0,011428392	0,082270864
	1,64E-44	0,000127854	0,692424037	0,12130415
	2,37E-48	0,000206564	0,013331633	0,175260072
	1,88E-48	1,89E-05	0,343229114	0,022786473

	2,96E-48	2,84E-05	0,016923726	0,173639146
	2,97E-50	0,001105353	0,056639243	0,257489035
	3,42E-43	0,001199195	0,09425984	0,137091216
	6,84E-46	0,000537458	0,077630226	0,124898317
	2,04E-49	0,000202366	0,023769841	0,239278599
	3,88E-43	0,000188866	0,018264503	0,042081367
	1,12E-49	0,000345206	0,005037754	0,109479542
	1,55E-50	0,008072866	0,017964316	0,117309488
	2,32E-46	0,000131291	0,07496326	0,067426048
	7,98E-46	0,015263781	0,068080741	0,097850604
	2,79E-45	3,98E-05	0,294978404	0,173390902
	1,27E-46	0,001035195	0,014329761	0,023480003
	4,55E-48	7,08E-06	0,017022881	0,04824262
	6,44E-53	0,000405998	0,062378823	0,040662291
Average	7,96E-40	0,005709694	0,114699449	0,11608065
Time (sec.)	206.928.051	185.685.596	322.883.934	173.136.008
Standard deviation	7,90E-39	0,017817676	0,284024178	0,07179942

Benchmarks	F5	F6	F7	F8
	73,43466017	1,04E-24	0,006417022	-7713,23157
	18,24134998	6,13E-26	0,008884894	-6528,807424
	21,9424048	1,54E-23	0,005890579	-7020,352332
	17,78853476	9,33E-26	0,009427005	-6309,964229
	21,11131714	4,52E-24	0,014212342	-5620,222135
	146,9668414	1,24E-27	0,008468903	-7238,872061
	22,00023593	1,49E-26	0,015832878	-5895,658133
	21,26946556	4,59E-24	0,010127069	-7515,83161
	21,18975121	5,55E-14	0,009906256	-6942,949105
	27,40745502	5,95E-28	0,021960399	-5600,855602
	19,44273865	3,36E-29	0,008381356	-6191,480774
	8,286774012	6,40E-29	0,007929761	-5541,189314
	21,94196209	8,29E-31	0,00573315	-6982,78925
	71,13107168	1,12E-30	0,013325731	-6092,916438
	72,25598576	2,39E-25	0,018023464	-7496,148168
	23,9360129	1,64E-26	0,008612404	-5857,626953
	22,85489996	2,74E-29	0,00527295	-6252,423854
	20,98452336	1,07E-27	0,003989176	-7160,389117
	16,75524002	4,34E-28	0,009272316	-7614,690626
	74,80820786	2,91E-26	0,017363102	-6607,727711
	21,97175928	9,00E-30	0,018935349	-7042,189735
	78,74283602	8,65E-29	0,009668621	-6212,908515
	15,5917601	1,49E-22	0,007765474	-5502,279047
	77,96426891	5,80E-30	0,005053706	-7515,909632
	21,46181188	2,88E-30	0,01060468	-7772,438788

	3,366988223	1,04E-23	0,013073819	-5304,509936
	20,35966288	5,28E-22	0,008451458	-7357,974471
	78,90354384	1,52E-26	0,008536178	-6509,037523
	20,96903139	9,60E-28	0,00963932	-6351,210267
	22,67555264	2,38E-24	0,009000145	-5837,772538
	4,225176469	1,40E-29	0,010914356	-7711,736265
	76,1056302	2,08E-22	0,011109829	-6074,82289
	22,70390519	6,75E-29	0,003595261	-5876,81622
	73,79059868	3,26E-29	0,008517301	-6252,586954
	23,3212131	1,83E-29	0,007956451	-7139,360153
	70,68348313	1,25E-27	0,005628605	-6706,517143
	73,30273902	4,16E-22	0,006559991	-5916,835243
	22,05208444	9,71E-30	0,011358228	-6528,691082
	26,33093511	3,70E-27	0,008612345	-6073,113635
	19,94767036	2,16E-24	0,009242999	-5956,286948
	20,24494834	1,95E-25	0,010592533	-6765,672947
	21,62407105	9,28E-28	0,011159129	-7397,482444
	26,75059583	5,18E-31	0,002570271	-5561,360946
	26,82182221	1,60E-16	0,014093958	-7121,051044
	21,18914562	1,81E-30	0,005289842	-6133,966474
	14,06553354	1,03E-27	0,009290533	-5186,302116
	21,00871913	2,86E-28	0,010644453	-6291,984313
	24,47147871	2,89E-29	0,008535095	-8088,349377
	75,74537004	3,39E-31	0,007873084	-6351,03602
	11,92050029	4,11E-30	0,010827686	-5916,815658
	86,94157576	5,75E-28	0,013523995	-5363,970495
	9,80322071	1,66E-29	0,005175128	-6074,260228
	23,00599304	1,30E-29	0,016404206	-6588,106521
	14,06046286	1,07E-27	0,014771644	-6133,863449
	21,38575323	1,60E-24	0,016919337	-7989,693163
	21,5925186	1,98E-28	0,005863495	-6785,304212
	81,32355776	9,84E-30	0,013346164	-7555,460582
	21,95478838	1,69E-25	0,004492939	-6489,289376
	81,74321199	5,74E-29	0,003434754	-6647,362101
	73,76301237	1,87E-30	0,013267298	-6803,668516
	13,53559309	2,67E-30	0,010826013	-6232,475615
	20,46575689	1,50E-26	0,007943776	-5877,365024
	22,51041035	1,23E-26	0,018595447	-7160,489433
	6,140471162	2,09E-30	0,00519344	-6489,43645
	21,88558624	1,84E-27	0,005924189	-6509,126692
	21,46719327	9,06E-31	0,008827024	-5638,912319
	26,34508967	1,76E-28	0,006912813	-6923,742962
	22,93914065	4,34E-31	0,008309965	-5679,983037
	14,15670476	2,29E-30	0,01159807	-7120,848242
	21,00423048	3,30E-31	0,020225638	-6785,499116

	75,24444068	3,83E-29	0,009611125	-6114,433914
	25,91629902	6,87E-30	0,025455729	-7200,018088
	22,5786498	4,35E-24	0,013367077	-6449,692234
	20,26041284	7,21E-31	0,008955897	-8165,787117
	11,19391014	4,13E-27	0,016291934	-7061,823709
	97,9685917	1,05E-26	0,008364767	-6450,000771
	10,96803689	6,96E-24	0,013071119	-6706,634025
	74,89356192	1,66E-30	0,012241581	-7772,520592
	72,73736778	1,97E-30	0,011388373	-6607,899249
	72,09604082	1,76E-26	0,006057112	-6133,943108
	78,29059503	1,59E-25	0,024473679	-6331,413193
	25,409318	7,64E-27	0,007599846	-6803,601102
	80,95354975	2,47E-31	0,00708434	-7178,742275
	20,78730412	1,88E-26	0,005172794	-6055,065233
	21,81078768	2,31E-24	0,011610326	-7180,342775
	70,09044919	2,16E-20	0,01500661	-5699,156278
	14,46836941	1,94E-28	0,006926672	-6369,423151
	25,88674279	5,03E-28	0,02302087	-6489,43916
	21,46149067	2,84E-29	0,009203261	-6270,652232
	75,37146783	2,04E-29	0,00887098	-6548,4743
	74,45370985	1,10E-24	0,004622511	-6509,151156
	76,62580569	1,65E-25	0,012877609	-6805,144237
	74,67451212	2,60E-28	0,010585181	-6527,242535
	20,51626415	9,60E-25	0,010312975	-7276,986683
	24,34238471	9,55E-24	0,018335736	-6153,769277
	76,56657932	1,00E-25	0,017025431	-6685,221744
	68,88522633	1,11E-30	0,011730848	-5581,202043
	13,53470285	1,11E-27	0,013788304	-7042,042661
	2,271557776	2,20E-28	0,006397469	-7259,226378
	26,17627185	3,31E-26	0,007014477	-6805,313065
Average	<b>37,8452094</b>	<b>5,57E-16</b>	<b>0,010521514</b>	<b>-6577,243646</b>
Time (sec.)	181.185.022	171.140.294	192.238.756	175.130.112
Standard deviation	28,74609513	5,52E-15	0,004697737	668,4216919

Benchmarks	F9	F10
	96,5108066	2,120053362
	66,66213074	1,646223633
	32,83362369	1,340421288
	45,76807632	3,81E-12
	62,68229955	1,15E-11
	50,74284637	2,660168373
	46,76301518	1,00E-13
	30,84371062	1,340421288
	67,6570898	1,501746567

	46,76301519	1,54E-13
	37,80839375	3,95E-13
	70,64194678	2,57E-13
	29,84875156	1,155148503
	64,67225806	1,21E-11
	44,77307687	1,646223633
	34,82354181	1,340421288
	51,73779027	1,501746567
	45,76804097	1,777996987
	49,74790751	2,013315236
	44,77311727	1,340421288
	54,72265737	1,646223633
	42,78318399	4,59E-12
	61,6873405	1,899744051
	30,84371062	1,17E-07
	29,8487566	2,93E-14
	40,79326588	5,27E-12
	34,82352161	1,775071978
	34,82354181	0,931304602
	40,79328104	4,95E-13
	41,78822997	2,71E-13
	54,72265737	1,501746567
	46,76301518	1,155148503
	45,76806116	3,24E-12
	70,64192136	2,118957803
	54,72267757	1,501746567
	57,7075295	2,013315236
	55,71759623	1,501746567
	38,803373	1,57E-13
	31,83866968	0,931304602
	42,78318903	1,38E-11
	31,83864948	1,899744051
	33,82858275	2,013315236
	37,80839878	4,49E-12
	31,83864948	2,68E-12
	31,83866968	0,931304602
	27,85882841	2,23E-11
	51,73778524	0,931304602
	67,6570898	1,501746567
	59,69744762	1,155148503
	30,84371062	4,74E-10
	51,73778524	1,501746567
	35,81850087	1,899744051
	39,79829166	1,899744051
	39,79833206	1,155148503

	56,71250967	5,06E-07
	39,79833206	8,62E-14
	22,88404824	1,501746567
	41,7881997	0,931304602
	36,81343469	3,28E-13
	41,78822494	1,501746567
	89,54613863	1,501746567
	40,79328608	0,931304602
	39,79831186	6,90E-11
	33,82858779	1,340421288
	83,57633359	2,013315236
	23,87900729	1,777996987
	64,67223786	5,66E-12
	68,65200324	9,73E-12
	64,67221767	9,39E-13
	66,66213578	1,340421288
	44,77310211	4,41E-13
	36,81343973	1,155148503
	27,85881325	2,013315236
	33,82856759	1,340421288
	65,66719188	1,155148503
	49,74786712	1,501746567
	49,74790751	3,28E-13
	59,69736665	3,06E-12
	49,74788732	2,579534568
	60,69240164	1,646223633
	53,72770335	3,70E-13
	24,87396635	0,931304602
	53,72769327	0,931304602
	29,8487364	1,155148503
	73,62683407	2,97E-10
	85,56625171	1,340421288
	53,72765774	2,07E-10
	44,77310211	1,775071978
	62,68211668	1,501746567
	61,68729992	5,73E-09
	75,61670657	1,646223633
	50,74283122	0,931304602
	51,73779027	0,931304602
	68,65204382	1,155148503
	21,88909422	1,11E-09
	33,82856258	2,29E-10
	54,72267757	1,775071978
	36,81343973	1,02E-12
	70,6419569	1,340421288

	36,81344985	1,47E-13
Average	<b>48,46437808</b>	<b>0,928955796</b>
Time (sec.)	170.405.224	178.845.571
Standard deviation	15,58895856	7,94E-01

## EK D

### D.1 İHA İle Yol Bulma Problemi MATLAB Kodları

#### Main.m

```
%  
% Grey Wolf Optimizer (GWO) source codes version 1.0 %  
%  
% Developed in MATLAB R2011b(7.13) %  
%  
% Author and programmer: Seyedali Mirjalili %  
%  
% e-Mail: ali.mirjalili@gmail.com %  
% seyedali.mirjalili@griffithuni.edu.au %  
%  
% Homepage: http://www.alimirjalili.com %  
%  
% Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis %  
% Grey Wolf Optimizer, Advances in Engineering %  
% Software , in press, %  
% DOI: 10.1016/j.advengsoft.2013.12.007 %  
%  
%  
%
```

% You can simply define your cost in a seperate file and load its handle to fobj

% The initial parameters that you need are:

```
%  
% fobj = @YourCostFunction  
% dim = number of your variables  
% Max_iteration = maximum number of generations  
% SearchAgents_no = number of search agents  
% lb=[lb1,lb2,...,lbn] where lbn is the lower bound of variable n  
% ub=[ub1,ub2,...,ubn] where ubn is the upper bound of variable n  
% If all the variables have equal lower bound you can just  
% define lb and ub as two single number numbers
```

% To run GWO:

```
[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,ub,dim,fob  
j)  
%
```

clear all

clc

global Positions; %declare the positions globally so that we can examine them in our works space.

global fitness;

global xcoor;

global ycoor;

global xo;

global yo;

```

global Startpoint;
global Targetpoint;
global siz
global dim
global threats
global numberofthreats
global threatrange
global threatdanger
global xnew
global threat
%global siz;
%Declare start and end points
%change these to change the resolution of your solution
Startpoint=[0,0];
Targetpoint=[10,10];

numberofthreats=2;
threats=[5 9; 4 1];
threatrange=2;
threatdanger=10;
%Find the coeff
coeff=mathoperations( Startpoint(1),Startpoint(2),Targetpoint(1),Targetpoint(2));

%gives the xcoordinates that the UCAV will follow

xcoor=givethexcoor(coeff,Startpoint,Targetpoint);

ycoor=givetheycoor(coeff,Startpoint,Targetpoint,xcoor);

SearchAgents_no=30; % Number of search agents

Function_name='F1'; % Name of the test function that can be from F1 to F23 (Table 1,2,3 in
the paper)

Max_iteration=50000; % Maximum numbef of iterations

% Load details of the selected benchmark function
[lb,ub,dim,fobj]=Get_Functions_details(Function_name);

[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,ub,dim,fob
j);

bestoutput = [ycoor(1) Best_pos(1:end) ycoor(end)];
subplot(2,1,1);
LineH =plot(xcoor,bestoutput);
title('Grafik A:Planlanan yol')
axis([0 10 0 10]);
%set(LineH, 'YLimInclude', 'off');
grid on;
subplot(2,1,2);

```

```

Line2 =plot(threats(1,:),threats(2,:),'*');
title('Grafik B:Tehlike noktalari')
grid on;
axis([0 10 0 10]);
%set(Line2, 'YLimInclude', 'off');

display(['The best solution obtained by GWO is : ', num2str(Best_pos)]);
display(['The best optimal value of the objective funciton found by GWO is : ', num2str(Best_score)]);

```

### **MathOperations.m**

```

function [ coefficients ] = mathoperations( x1,y1,x2,y2 )
%this function finds the coefficients for the equation of the line that
%connects the starting and target point.
coefficients = polyfit([x1, x2], [y1, y2], 1);
end

```

### **initialization.m**

```

%
% _____ %
% Grey Wolf Optimizer (GWO) source codes version 1.0 %
% %
% Developed in MATLAB R2011b(7.13) %
% %
% Author and programmer: Seyedali Mirjalili %
% %
% e-Mail: ali.mirjalili@gmail.com %
% seyedali.mirjalili@griffithuni.edu.au %
% %
% Homepage: http://www.alimirjalili.com %
% %
% Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis %
% Grey Wolf Optimizer, Advances in Engineering %
% Software , in press, %
% DOI: 10.1016/j.advengsoft.2013.12.007 %
% %
% _____ %

```

```

% This function initialize the first population of search agents
function Positions=initialization(SearchAgents_no,dim,ub,lb)

```

```

Boundary_no= size(ub,2); % numnber of boundaries

```

```

% If the boundaries of all variables are equal and user enter a signle
% number for both ub and lb
if Boundary_no==1
    Positions=rand(SearchAgents_no,dim).*(ub-lb)+lb;

```

```

end

% If each variable has a different lb and ub
if Boundary_no>1
    for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        Positions(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;
    end
end

```

## GWO.m

```

%_____
% Grey Wolf Optimizer (GWO) source codes version 1.0 %
%
% Developed in MATLAB R2011b(7.13) %
%
% Author and programmer: Seyedali Mirjalili %
%
% e-Mail: ali.mirjalili@gmail.com %
% seyedali.mirjalili@griffithuni.edu.au %
%
% Homepage: http://www.alimirjalili.com %
%
% Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis %
% Grey Wolf Optimizer, Advances in Engineering %
% Software , in press, %
% DOI: 10.1016/j.advengsoft.2013.12.007 %
%
%_____
%
% Grey Wolf Optimizer
function
[Alpha_score,Alpha_pos,Convergence_curve]=GWO(SearchAgents_no,Max_iter,lb,ub,dim,obj)

% initialize alpha, beta, and delta_pos
Alpha_pos=zeros(1,dim);
Alpha_score=inf; %change this to -inf for maximization problems

Beta_pos=zeros(1,dim);
Beta_score=inf; %change this to -inf for maximization problems

Delta_pos=zeros(1,dim);
Delta_score=inf; %change this to -inf for maximization problems

%Initialize the positions of search agents
Positions=initialization(SearchAgents_no,dim,ub,lb);

```



```

r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]

A1=2*a*r1-a; % Equation (3.3)
C1=2*r2; % Equation (3.4)

D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j)); % Equation (3.5)-part 1
X1=Alpha_pos(j)-A1*D_alpha; % Equation (3.6)-part 1

r1=rand();
r2=rand();

A2=2*a*r1-a; % Equation (3.3)
C2=2*r2; % Equation (3.4)

D_beta=abs(C2*Beta_pos(j)-Positions(i,j)); % Equation (3.5)-part 2
X2=Beta_pos(j)-A2*D_beta; % Equation (3.6)-part 2

r1=rand();
r2=rand();

A3=2*a*r1-a; % Equation (3.3)
C3=2*r2; % Equation (3.4)

D_delta=abs(C3*Delta_pos(j)-Positions(i,j)); % Equation (3.5)-part 3
X3=Delta_pos(j)-A3*D_delta; % Equation (3.5)-part 3

Positions(i,j)=(X1+X2+X3)/3;% Equation (3.7)

end
end
% % % % % % % % % % % % % % % % % % % % % % % EVALUAGE J HERE F?RST
FOR ALL X? EQ2% % % % % % % % % % % % % % % % % % % % % % %
l=l+1;
Convergence_curve(l)=Alpha_score;
end

```

### **givetheycoor.m**

```

function [ ycoor ] = givetheycoor(coefficients,startpoint,targetpoint,xcoor)
%this function finds the coefficients for the x axis of the line that
%connects the starting and target point.

```

```

a=coefficients(1);
b=coefficients(2);

```

```

siz=targetpoint(1)-startpoint(1)+1;

```

```

for c = 1:siz

```

```

ycoor(c)=a*xcoor(c)+b;
end
end

```

### **givethexcoor.m**

```

function [ xcoor ] = givethexcoor(coefficients,startpoint,targetpoint)
%this function finds the coefficients for the x axis of the line that
%connects the starting and target point.
siz=targetpoint(1)-startpoint(1)+1;
a=startpoint(1);
for c = 1:siz
xcoor(c)=a;
a=a+1;
end
end

```

### **Get\_function\_details.m**

```

%
% Grey Wolf Optimizer (GWO) source codes version 1.0 %
%
% Developed in MATLAB R2011b(7.13) %
%
% Author and programmer: Seyedali Mirjalili %
%
% e-Mail: ali.mirjalili@gmail.com %
% seyedali.mirjalili@griffithuni.edu.au %
%
% Homepage: http://www.alimirjalili.com %
%
% Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis %
% Grey Wolf Optimizer, Advances in Engineering %
% Software , in press, %
% DOI: 10.1016/j.advengsoft.2013.12.007 %
%
%
```

```

% This function contains full information and implementations of the benchmark
% functions in Table 1, Table 2, and Table 3 in the paper

```

```

% lb is the lower bound: lb=[lb_1,lb_2,...,lb_d]
% ub is the upper bound: ub=[ub_1,ub_2,...,ub_d]
% dim is the number of variables (dimension of the problem)

```

```

function [lb,ub,dim,fobj] = Get_Functions_details(F)

```

```

switch F

```

```

case 'F1'%THE BOUNDS

global Startpoint
global Targetpoint
global siz
global dim
siz=Targetpoint(1)-Startpoint(1)-1;

fobj = @F1;
lb=-100;
ub=100;
dim=siz;

end

end

% F1

function o = F1(x)

global Startpoint
global Targetpoint
global xcoor;
global ycoor;
global xo;
global yo;
global siz
global xnew
%global siz;
xo=xcoor;
yo=ycoor;

%determine the amount of turning points in our path
siz=Targetpoint(1)-Startpoint(1);

cost=0;
%calculate the fuel cost for each turning point in our path
xnew = [ycoor(1) x(1:end) ycoor(end)];

%add the fuel cost for each point
for c = 1:siz
distance=CalcDistance(xcoor(c),xnew(c),xcoor(c+1),xnew(c+1));
cost=cost+distance;
end

%calculate the threat cost for each threat
global threats
global numberofthreats
global threatrange

```

```

global threatdanger
global threat
threatcost=0;
sizz=siz+1;
for c = 1:sizz
for n = 1:numberofthreats

threat=CalcDistance(xcoor(c),xnew(c),threats(1,n),threats(2,n));
if threat<=threatrange
threatcost=(threat*threatdanger)+threatcost;
end

end
end

cost=cost+threatcost;
o=cost;

end

```

### **CalcDistance.m**

```

function euclideanDistance = CalcDistance(x1, y1, x2, y2)
euclideanDistance = sqrt((x2-x1)^2+(y2-y1)^2);

```

## D.2 Gray Wolf Optimizer Algoritmasının İki Boyutlu Benchmark Fonksiyonları Tablosu

Benchmarks	F1	F2	F3	F4
	2.26E-71	1.25E-40	1.78E-24	3.08E-17
	1.86E-71	9.23E-42	3.72E-20	8.49E-18
	6.94E-71	2.96E-41	1.23E-21	3.19E-18
	5.15E-71	1.65E-41	8.06E-23	4.16E-18
	2.86E-71	1.94E-41	4.29E-22	1.92E-17
	6.32E-71	2.66E-41	5.51E-19	3.75E-18
	2.37E-69	1.45E-40	5.82E-22	3.14E-17
	1.18E-72	4.54E-41	3.11E-21	8.81E-18
	2.60E-72	9.69E-41	9.21E-21	7.91E-18
	2.23E-69	7.48E-41	2.23E-22	1.11E-17
	9.98E-71	1.41E-41	1.18E-19	4.53E-19
	1.07E-70	4.44E-41	7.04E-20	8.56E-18
	2.84E-71	2.83E-41	5.05E-24	4.62E-18
	5.04E-71	4.83E-42	2.35E-23	1.10E-17
	2.14E-70	1.84E-41	2.27E-21	2.33E-17
	7.60E-72	6.96E-41	2.36E-22	4.57E-17
	1.07E-71	8.07E-41	1.41E-20	4.94E-17
	3.43E-71	4.22E-41	1.52E-21	8.20E-18
	7.01E-70	8.36E-42	3.51E-20	4.08E-17
	4.40E-71	3.13E-41	1.21E-22	8.86E-18
	9.50E-71	9.22E-42	5.04E-20	3.96E-18
	2.14E-72	1.04E-40	4.37E-19	1.01E-17
	1.46E-70	5.06E-42	2.30E-22	2.23E-17
	4.93E-72	6.32E-42	5.29E-21	2.32E-18
	3.07E-69	2.45E-41	1.63E-22	4.90E-18
	6.91E-73	1.25E-40	1.42E-23	6.98E-17
	1.88E-70	7.60E-42	2.63E-21	3.35E-18
	1.55E-70	5.01E-42	8.43E-24	2.00E-17
	2.03E-71	6.87E-41	1.26E-22	1.67E-18
	1.42E-69	2.29E-41	1.02E-17	2.15E-17
	3.35E-71	7.09E-41	6.08E-19	6.45E-18
	5.47E-70	1.45E-41	1.08E-22	1.56E-18
	6.87E-71	2.51E-41	4.83E-23	7.44E-19
	5.53E-72	7.70E-41	4.61E-24	1.09E-17
	3.71E-70	1.64E-41	1.02E-20	2.18E-17
	2.85E-71	6.10E-41	8.47E-24	6.10E-18
	1.20E-70	3.16E-41	1.60E-23	2.07E-17
	1.17E-71	3.60E-41	2.47E-23	1.96E-17
	4.90E-72	1.53E-41	9.71E-24	1.13E-16

	9.33E-72	7.11E-41	2.79E-22	1.14E-16
	7.42E-71	2.88E-41	9.43E-22	4.43E-18
	6.17E-71	4.20E-41	1.14E-21	1.01E-17
	3.26E-71	3.30E-41	7.03E-20	2.08E-17
	3.90E-71	1.56E-41	1.57E-23	4.51E-18
	2.32E-70	3.39E-41	1.19E-19	1.85E-17
	4.60E-71	1.17E-40	8.21E-22	2.53E-18
	1.32E-70	6.79E-41	6.62E-21	1.25E-17
	5.89E-71	2.99E-41	7.06E-19	2.90E-18
	7.67E-72	3.13E-41	4.41E-23	3.16E-18
	9.11E-71	8.77E-42	1.96E-21	8.10E-18
	2.53E-70	3.07E-42	2.96E-19	6.21E-18
	1.01E-71	3.44E-41	3.65E-20	4.10E-17
	1.66E-70	3.98E-42	9.09E-22	2.46E-18
	7.79E-71	6.76E-42	6.48E-20	1.10E-17
	1.43E-71	1.34E-41	7.84E-21	7.13E-18
	3.33E-72	2.90E-41	1.68E-21	4.55E-17
	1.79E-72	2.29E-41	1.57E-23	1.48E-18
	3.87E-71	1.84E-41	8.02E-22	2.55E-17
	8.35E-72	1.54E-41	1.45E-21	5.27E-18
	6.00E-72	3.45E-41	3.57E-19	1.25E-17
	4.98E-71	4.82E-41	1.82E-19	3.29E-18
	4.20E-71	2.64E-41	4.56E-23	2.19E-18
	1.96E-70	4.48E-41	6.46E-22	2.49E-18
	7.72E-70	4.54E-41	7.27E-19	3.17E-17
	1.82E-71	1.41E-41	7.16E-22	3.29E-17
	2.50E-71	1.34E-41	1.26E-23	3.77E-19
	7.88E-72	7.13E-42	4.27E-19	1.04E-18
	2.99E-69	6.05E-42	1.00E-18	1.37E-18
	1.93E-71	1.33E-41	1.80E-23	1.58E-18
	7.38E-70	1.14E-40	1.42E-22	1.28E-18
	9.66E-71	6.38E-42	2.62E-22	4.06E-17
	3.65E-71	2.48E-41	3.14E-23	4.33E-18
	7.11E-70	1.52E-40	1.26E-21	1.18E-17
	4.10E-70	8.25E-41	3.40E-19	2.47E-18
	1.30E-71	2.87E-40	1.11E-21	7.87E-18
	1.88E-71	1.90E-41	1.76E-21	5.71E-17
	4.10E-70	9.69E-41	1.07E-24	2.11E-18
	1.35E-71	1.04E-40	3.72E-21	1.51E-18
	1.73E-70	1.31E-40	1.98E-25	3.33E-17
	4.96E-69	4.91E-42	1.10E-24	3.46E-18
	1.46E-70	8.84E-41	9.04E-20	2.39E-16
	5.17E-73	1.74E-41	1.37E-24	3.16E-18
	7.18E-71	8.89E-41	2.17E-20	3.44E-17
	4.84E-72	2.18E-41	8.43E-20	4.50E-19

	1.09E-71	8.22E-42	5.23E-20	9.17E-18
	2.23E-70	6.18E-41	1.39E-24	9.25E-18
	4.90E-71	2.39E-41	1.86E-23	3.57E-18
	3.37E-72	1.30E-41	1.03E-20	5.71E-18
	8.66E-71	1.54E-41	2.89E-19	3.67E-17
	1.25E-70	4.99E-41	1.03E-20	3.44E-17
	9.61E-72	5.31E-41	8.39E-22	6.77E-18
	1.26E-69	9.19E-42	5.43E-20	2.35E-18
	5.83E-70	5.59E-42	1.19E-17	6.38E-18
	7.14E-71	8.16E-41	9.23E-22	1.03E-16
	8.68E-72	3.36E-41	1.32E-23	1.64E-17
	1.35E-72	2.04E-41	1.06E-18	3.30E-17
	4.18E-70	8.00E-41	1.40E-21	2.65E-17
	1.98E-71	6.86E-42	1.20E-19	1.73E-17
	3.80E-71	8.77E-42	9.36E-21	1.53E-17
	3.34E-69	1.76E-41	2.45E-21	1.37E-17
Average	3.20E-70	4.26E-41	3.02E-19	1.91E-17
Time (sec.)	3.12E+01	34.386021	116.939583	30.296272
Standard deviation	7.88123E-70	4.36563E-41	1.55436E-18	3.09747E-17

Benchmarks	F5	F6	F7	F8
	25.20935368	1.56E-05	0.000343119	-6139.824902
	27.93230236	0.500605791	0.000308748	-5466.808414
	27.8723658	1.59E-05	0.00032411	-5705.918231
	25.20744744	0.500775882	0.000277729	-4123.635052
	26.15298071	0.248654712	0.001212299	-6560.81688
	27.87899194	1.254866291	0.000323317	-6250.51903
	26.20987318	0.255104437	0.000498778	-5809.281642
	25.72549645	0.373498035	0.000297922	-5129.005756
	26.20168489	0.745438128	0.00026215	-5929.187395
	26.05941472	0.249328007	0.000485866	-5907.044194
	27.11182479	1.35E-05	0.000215179	-3775.761901
	27.1234256	1.253112808	0.000318169	-7014.946977
	27.11502293	0.250185994	0.00028807	-5748.64727
	26.16702182	0.09866497	0.000309547	-6458.675296
	26.2093988	0.501142789	0.000348699	-6453.985824
	26.17176266	0.497464722	0.000254912	-6343.394645
	27.07422417	0.250072973	0.000166784	-6358.645314
	27.08193173	1.44E-05	0.000312058	-6483.657849
	26.20777852	0.254655698	0.000763046	-6710.966466
	26.17473739	1.08E-05	0.000144007	-6581.868156
	27.14786266	0.489380069	0.000311315	-7233.497883
	25.28019761	0.247081223	0.00012413	-6425.345159
	27.11697617	0.498377885	0.000135125	-5485.513992

	27.92613966	0.739903676	0.000273535	-5787.150964
	26.05948009	0.250067816	0.000686345	-5730.347169
	25.20802377	1.75E-05	0.000410172	-6387.633927
	27.09429253	0.501631532	0.000203735	-5353.973169
	26.11747873	0.50055357	0.00112149	-7694.838488
	25.2113645	1.48E-05	0.000837885	-6641.705477
	26.19336557	1.49E-05	0.000253113	-7171.691766
	26.20344116	0.495711028	0.000548761	-6918.022052
	27.10024128	1.11E-05	0.000459773	-6233.753466
	26.22868711	9.00E-06	0.000434227	-6461.612078
	26.21337517	0.735535832	0.000683567	-7592.898085
	26.05287899	7.99E-06	0.000918135	-5863.562876
	26.12439517	1.004381173	0.00032114	-5641.106184
	27.11847943	0.485909134	0.000393821	-5785.298146
	27.13103026	0.754158046	0.000979557	-5255.900795
	26.15748471	0.493808752	0.000173887	-6673.327774
	27.90970944	1.30E-05	0.000834285	-6535.668333
	26.19582209	0.497736937	0.000458001	-7189.058332
	26.19965551	0.506807391	0.00130929	-5943.856169
	27.08883458	7.96E-06	0.00067324	-6385.715609
	27.09210146	0.499785804	0.000255633	-6322.925506
	25.76866188	1.80E-05	0.000422925	-5796.264115
	25.83288408	0.750367367	0.000342379	-5180.726488
	26.14538802	0.250486024	0.000505805	-6067.124824
	26.21802238	1.63E-05	0.000724403	-5465.111368
	26.19865535	1.004465183	0.000463968	-3956.163362
	26.1672561	0.252114226	0.000305878	-7064.55806
	25.29560785	0.494984074	0.000744913	-6671.876386
	26.18471762	0.502244433	0.000824962	-6248.303263
	27.12917789	9.77E-06	0.000252782	-6165.075028
	25.22928635	0.249958292	0.000248913	-6531.369071
	25.24963908	0.250394962	0.000938685	-6586.931318
	26.16478648	0.2513804	0.000169406	-6718.966052
	26.18641952	0.450922603	0.000202738	-6784.119331
	27.08574602	0.679836967	0.000449823	-6490.564351
	26.22931069	0.503822132	0.000403607	-5755.591921
	26.18168981	0.503427259	0.000279329	-6597.804798
	25.68784041	0.499545083	0.000324143	-6034.193001
	25.24218072	0.250656481	0.000463886	-5983.937023
	26.19716851	0.244684583	0.000815064	-6146.0497
	26.13056103	0.500849501	0.000388945	-7140.776236
	27.93606755	0.499302262	0.000813255	-6522.897715
	26.17701019	1.504674669	0.000522312	-6294.747274
	25.95496526	0.251198877	0.000312601	-4957.050819
	26.07071707	0.503573185	0.000275054	-6780.845262

	27.89034352	8.24E-06	0.000103837	-5665.765539
	27.14626983	0.491103823	0.000555691	-5324.465041
	26.16325357	0.502963278	0.0002426	-5616.34988
	26.16850704	0.251233747	0.000421894	-6643.616528
	27.23791011	0.504882544	0.000331193	-5743.654108
	26.11798411	0.250688566	0.000671773	-6397.87529
	26.208972	1.05E-05	0.00055039	-6460.986576
	26.19574577	0.255519166	0.000633864	-6955.228659
	27.16316809	0.502423498	0.000304676	-5250.131051
	27.08886138	0.445464695	0.00078215	-6497.632914
	25.12442723	1.33E-05	0.000635789	-6323.651646
	27.08501324	1.004648728	9.19E-05	-6236.33833
	26.247092	0.254229759	0.000732673	-5916.046623
	26.07501587	0.499580124	0.000522189	-6452.97105
	25.35672543	0.527291123	0.000604525	-6586.362801
	26.19904418	0.708956851	0.000719557	-6622.453936
	26.16293905	0.746847862	0.000287248	-7423.605776
	26.17999839	0.497269169	0.000737232	-6246.289374
	26.1293885	0.246461677	0.000462768	-5729.499682
	27.12673988	0.25331122	0.000101204	-6977.804196
	26.21462212	1.02E-05	0.000137034	-4042.577443
	26.23490004	0.248757047	0.000321961	-4494.937072
	27.14483222	0.937980913	0.000372735	-6829.077415
	26.16634348	0.249870776	0.000367178	-7817.693018
	26.25158215	0.252219743	0.000196471	-6734.647116
	25.21076133	0.250034305	0.000928308	-5325.055234
	26.20087855	0.50027978	0.000489816	-5454.358437
	27.89102627	0.06711311	0.000597788	-4887.179264
	26.18697817	1.35E-05	0.0004072	-5782.729389
	26.17606852	0.251230601	0.000674097	-6513.703604
	27.12559459	0.501257226	0.0005459	-7024.90913
	27.95199148	9.17E-06	0.000328387	-6245.256394
Average	2.64E+01	3.75E-01	4.63E-04	-6.16E+03
Time (sec.)	33.959303	30.051411	48.829518	37.178855
Standard deviation	0.730322667	0.30877764	0.000254972	766.0962148

Benchmarks	F9	F10
	0	1.15E-14
	0	1.51E-14
	0	7.99E-15
	5.68E-14	1.51E-14
	0	1.15E-14
	0	7.99E-15
	0	1.15E-14

	0	1.15E-14
	0	1.51E-14
	3.343330295	1.51E-14
	0	7.99E-15
	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
	0	1.51E-14
	0	1.51E-14
	0	2.22E-14
	0	1.51E-14
	5.68E-14	1.51E-14
	0	1.15E-14
	0	7.99E-15
	0	2.22E-14
	1.71E-13	1.51E-14
	5.68E-14	1.51E-14
	5.68E-14	1.15E-14
	0	1.51E-14
	9.598428781	7.99E-15
	0	1.51E-14
	0	7.99E-15
	0	1.15E-14
	0	1.15E-14
	0	1.51E-14
	0	1.51E-14
	0	7.99E-15
	0	7.99E-15
	0	1.51E-14
	0	1.51E-14
	0	1.51E-14
	5.68E-14	1.51E-14
	0	1.15E-14
	0	1.51E-14
	4.442785865	1.51E-14
	0	1.51E-14
	0	2.22E-14
	5.68E-14	1.51E-14
	0	1.51E-14
	0	7.99E-15
	5.345002925	1.51E-14
	0	7.99E-15
	0	1.51E-14
	0	1.15E-14

	3.160392685	2.22E-14
	5.68E-14	1.15E-14
	0	1.51E-14
	0	1.51E-14
	5.68E-14	1.15E-14
	0	1.51E-14
	0	7.99E-15
	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
	0	1.15E-14
	0	7.99E-15
	0	1.51E-14
	0	1.87E-14
	0	1.51E-14
	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
	0	1.51E-14
	0	1.51E-14
	0	7.99E-15
	0	1.51E-14
	0	1.51E-14
	0	1.87E-14
	0	1.51E-14
	0	7.99E-15
	0	1.15E-14
	5.68E-14	1.15E-14
	0	1.15E-14
	0	1.51E-14
	0	1.15E-14
	0	1.15E-14
	5.68E-14	1.15E-14
	6.756962549	1.51E-14
	5.68E-14	1.15E-14
	0	1.51E-14
	5.936805009	1.15E-14
	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
	0	1.51E-14
	0	7.99E-15
	0	1.51E-14
	0	1.51E-14

	0	1.51E-14
	0	1.15E-14
	0	1.51E-14
Average	3.86E-01	1.34E-14
Time (sec.)	30.987519	33.319876
Standard deviation	1.508702583	3.23492E-15