T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BALİNA OPTİMİZASYON ALGORİTMASI VE GRİ KURT OPTİMİZASYONU ALGORİTMALARI KULLANILARAK YENİ HİBRİT OPTİMİZASYON ALGORİTMALARININ GELİŞTİRİLMESİ

Hazırlayan
Cansu DOĞAN

Danışman
Dr. Öğr. Üyesi Mustafa DANACI

Yüksek Lisans Tezi

Mayıs 2019
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BALİNA OPTİMİZASYON ALGORİTMASI VE GRİ KURT OPTİMİZASYONU ALGORİTMALARI KULLANILARAK YENİ HİBRİT OPTİMİZASYON ALGORİTMALARININ GELİŞTİRİLMESİ

(Yüksek Lisans Tezi)

Hazırlayan
Cansu DOĞAN

Danışman
Dr. Öğr. Üyesi Mustafa DANACI

Mayıs 2019
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Cansu DOĞAN

"Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonu Algoritmaları Kullanılarak Yeni Hibrit Optimizasyon Algoritmalarının Geliştirilmesi" adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi' ne uygun olarak hazırlanmıştır.

Hazırlayan

Cansu DOĞAN

Danışman

Dr. Öğr. Üyesi Mustafa DANACI

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Veysel ASLANTAŞ

Dr. Öğr. Üyesi Mustafa DANACI danışmanlığında **Cansu DOĞAN** tarafından hazırlanan "Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonu Algoritmaları Kullanılarak Yeni Hibrit Optimizasyon Algoritmalarının Geliştirilmesi" adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği** Anabilim Dalında **yüksek lisans** tezi olarak kabul edilmiştir.

20 / 05/ 2019

JÜRİ:

Danışman : Dr. Öğr. Üyesi Mustafa DANACI .....

Üye : Doç. Dr. Celal ÖZTÜRK .....

Üye : Dr. Öğr. Üyesi Hasan BADEM .....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun 21/05/2019 tarih ve 2019/31-06 sayılı kararı ile onaylanmıştır.

21/05/2019

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

ÖNSÖZ/TEŞEKKÜR

“Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonu Algoritmaları Kullanılarak Yeni Hibrit Optimizasyon Algoritmalarının Geliştirilmesi” isimli yüksek lisans tez çalışmam kapsamında bilgi, tecrübe ve desteğini esirgemeyen kıymetli danışman hocam sayın Dr. Öğr. Üyesi Mustafa DANACI’ya teşekkür ederim.

Ayrıca her zaman olduğu gibi yüksek lisans çalışmalarımda da beni destekleyen sevgili aileme en içten teşekkürlerimi sunarım.

Cansu DOĞAN

Mayıs 2019, KAYSERİ

BALİNA OPTİMİZASYON ALGORİTMASI VE GRİ KURT OPTİMİZASYONU ALGORİTMALARI KULLANILARAK YENİ HİBRİT OPTİMİZASYON ALGORİTMALARININ GELİŞTİRİLMESİ

Cansu DOĞAN

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Mayıs 2019
Danışman: Dr. Öğr. Üyesi Mustafa DANACI

ÖZET

Metasezgisel optimizasyon algoritmaları sağladığı çeşitli avantajlardan dolayı son zamanlarda oldukça popüler olmuştur.

Bu çalışmada sürü zekasına dayanan biyolojik tabanlı algoritmalardan, Balina Optimizasyon Algoritması, Gri Kurt Optimizasyonu, Parçacık Sürü Optimizasyonu ve Diferansiyel Gelişim Algoritmasının açıklaması yer almaktadır.

Bu tez çalışmasında kambur balinaların avlanma davranışlarını simüle eden Balina Optimizasyon Algoritması ile gri kurtların sosyal hiyerarşi ve avlanma davranışını simüle eden Gri Kurt Optimizasyonu kullanılarak hibrit bir optimizasyon algoritması oluşturulmuştur.

Oluşturulan hibrit optimizasyon algoritmasını iyileştirebilmek maksadıyla modifikasyon eklenerek İyileştirilmiş Hibrit Balina Optimizasyon Algoritması (İHBOA) önerilmiştir. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması, balinaların kuyruklarını suya vurma şiddetinin balinaların konumlarının değiştirilmesinde etkili olmasından esinlenilerek geliştirilmiştir.

Önerilen İHBOA 23 adet benchmark fonksiyonu ile test edilmiş ve sonuçları Balina Optimizasyon Algoritması, Gri Kurt Optimizasyonu, Parçacık Sürü Optimizasyonu, Diferansiyel Gelişim Algoritması ile karşılaştırılmıştır. Karşılaştırma sonucunda ise İHBOA'nın oldukça başarılı ve rekabetçi sonuçlar ürettiği görülmüştür.

Anahtar Kelimeler: Balina Optimizasyon Algoritması, Gri Kurt Optimizasyon Algoritması, Parçacık Sürü Optimizasyonu, Benchmark Problemleri

IMPROVEMENT OF NEW HYBRID OPTIMIZATION ALGORITHMS USING WHALE OPTIMIZATION ALGORITHM AND GREY WOLF OPTIMIZATION ALGORITHMS

Cansu DOĞAN

Erciyes University, Graduate School of Natural and Applied Sciences
Master Thesis, May 2019
Supervisor: Dr. Mustafa DANACI

ABSTRACT

Metaheuristic optimization algorithms have become quite popular recently due to the various advantages it provides.

In this study, Whale Optimization Algorithm, Grey Wolf Optimization, Particle Swarm Optimization and Differential Evolution Algorithm which are the biological intelligence based on herd intelligence algorithms are explained.

In this thesis, a hybrid optimization algorithm has been created by using Whale Optimization Algorithm simulating the hunting behaviors of humpback whales and Grey Wolf Optimization simulating the social hierarchy and hunting behaviors of grey wolves.

The Improved Hybrid Whale Optimization Algorithm (IHWOA) was proposed by adding modification to improve the hybrid optimization algorithm. Improved Hybrid Whale Optimization Algorithm was developed with the inspiration of the effects of intensity of splashing movements of whale tails on the position change of whales

The proposed IHWOA was tested with 23 benchmarks and the results were compared with the Whale Optimization Algorithm, Grey Wolf Optimization, Particle Swarm Optimization, Differential Evolution Algorithm. As a result of the comparison, it is seen that IHWOA produces highly successful and competitive results.

Keywords: Whale Optimization Algorithm, Grey Wolf Optimization, Particle Swarm Optimization, Benchmark Problems

İÇİNDEKİLER

OPTİMİZASYON ALGORİTMALARININ TEST PROBLEMLERİNE UYGULANMASI VE PERFORMANS ANALİZLERİNİN YAPILMASI

BİLİMSEL ETİĞE UYGUNLUK i
YÖNERGEYE UYGUNLUK SAYFASI ii
KABUL VE ONAY iii
ÖNSÖZ/TEŞEKKÜR iv
ÖZET v
ABSTRACT vi
İÇİNDEKİLER vii
KISALTMALAR x
TABLOLAR LİSTESİ xi
ŞEKİLLER LİSTESİ xii

1. BÖLÜM

GİRİŞ

1.1. Literatür Çalışması 1

2. BÖLÜM

GENEL BİLGİLER

2.1. Optimizasyon 5

3. BÖLÜM

POPÜLASYON TABANLI OPTİMİZASYON ALGORİTMALARI

3.1. Parçacık Sürü Optimizasyonu (PSO) 12

3.1.1. Parçacığın Konumunun Hesaplanması 12

3.1.2. Parçacığın Hızının Hesaplanması 13

3.1.3. Parçacık Sürü Optimizasyon Algoritması Akış Diyagramı 14

3.2. Diferansiyel Gelişim Algoritması (DGA) 16

3.2.1. Başlangıç Popülasyonunun Oluşturulması 17

3.2.2. Mutasyon 17

3.2.3. Çaprazlama 18

3.2.4. Seçim 19

3.2.5. Diferansiyel Gelişim Algoritması Akış Diyagramı 20

3.3. Balina Optimizasyon Algoritması (BOA) 21

3.3.1. Avın Kuşatılması 21

3.3.2. Kabarcık-Ağ Metodu 22

3.3.2.1. Daralan Kuşatma Mekanizması 22

3.3.2.2. Spiral Hareket 23

3.3.3. Av Arama 24

3.3.4 Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı 25

3.4. Gri Kurt Optimizasyonu (GKO) 27

3.4.1 Sosyal Hiyerarşi 27

3.4.2. Avın Kuşatılması 28

3.4.3. Avlama 29

3.4.4. Ava Saldırma 29

3.4.5. Av Arama 30

3.4.6 Gri Kurt Optimizasyonunun Adımları ve Akış Diyagramı 30

4. BÖLÜM

HİBRİT BALİNA OPTİMİZASYON ALGORİTMASI (HBOA)

4.1. Hibrit Balina Optimizasyon Algoritmasının Oluşturulması 32

4.2. Hibrit Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı 34

4.3 Hibrit Balina Optimizasyon Algoritmasının Test Edilmesi 36

5. BÖLÜM

İYİLEŞTİRİLMİŞ HİBRİT BALİNA OPTİMİZASYON ALGORİTMASI (İHBOA)

5.1. İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasının Oluşturulması 42

5.2. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı 44

5.3. İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasının Test Edilmesi.....46

6. BÖLÜM

SONUÇ.....48

KAYNAKÇA 51

ÖZGEÇMİŞ.....55

KISALTMALAR

BOA **:** Balina Optimizasyon Algoritması

DGA **:** Diferansiyel Gelişim Algoritması

FIFO **:** First In First Out

GKO **:** Gri Kurt Optimizasyonu

HBOA **:** Hibrit Balina Optimizasyon Algoritması

İHBOA **:** İyileştirilmiş Hibrit Balina Optimizasyon Algoritması

LIFO **:** Last In First Out

PSO **:** Parçacık Sürü Optimizasyonu

TABLOLAR LİSTESİ

Tablo 4.1 Tek Modlu Benchmark Fonksiyonları 37
Tablo 4.2. Çok Modlu ve yüksek boyutlu Benchmark Fonksiyonları 38
Tablo 4.3. Çok Modlu ve düşük boyutlu Benchmark Fonksiyonları 39
Tablo 4.4. HBOA, BOA, GKO, PSO VE DG algoritmalarının Benchmark fonksiyonları ile elde edilen sonuçları 41
Tablo 5.1. İHBOA, HBOA, BOA, GKO, PSO VE DG algoritmalarının Benchmark fonksiyonları ile elde edilen sonuçları 47

ŞEKİLLER LİSTESİ

Şekil 1.1. Optimizasyon Algoritmalarının Sınıflandırılması 1
Şekil 1.2. Metasezgisel Optimizasyon Algoritmalarının Sınıflandırılması 2
Şekil 2.1. Optimizasyon Problemlerinin Sınıflandırılması 7
Şekil 2.2. Tek Modlu Sphere Fonksiyonunun Grafiği 9
Şekil 2.3. Çok Modlu Rastrigin Fonksiyonu 10
Şekil 3.1. Parçacık Sürü Optimizasyon Algoritmasında Konum Güncelleme İşlemi 14
Şekil 3.2. Parçacık Sürü Optimizasyon Algoritması Akış Diyagramı 15
Şekil 3.3. NP tane D boyutlu kromozomdan oluşan popülasyon 17
Şekil 3.4. Mutasyon İşlemi 18
Şekil 3.5. Çaprazlama İşlemi 19
Şekil 3.6. Diferansiyel Gelişim Algoritması Akış Diyagramı 20
Şekil 3.7. Kambur Balinaların Kabarcık-ağ Metodu İle Avlanması 21
Şekil 3.8. Spiral Hareket 23
Şekil 3.9. Balina Optimizasyon Algoritması Akış Diyagramı 26
Şekil 3.10. Gri Kurt Hiyerarşisi 27
Şekil 3.11. Gri Kurt Optimizasyonu Algoritması Akış Diyagramı 31
Şekil 4.1. Hibrit Balina Optimizasyon Algoritması Akış Diyagramı 35
Şekil 5.1. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması Akış Diyagramı 45

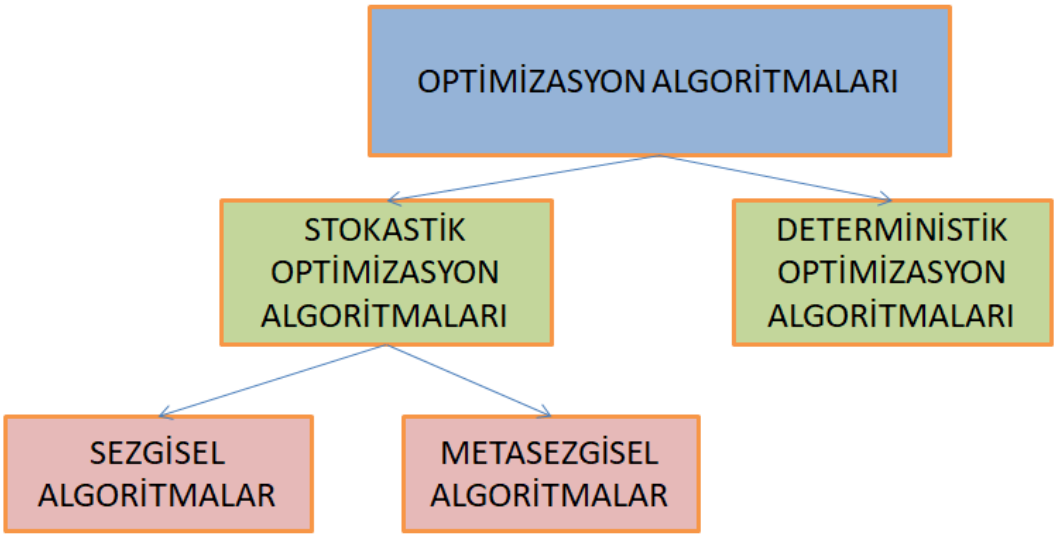
1. BÖLÜM

GİRİŞ

Optimizasyon teknolojisi, karar verme işlemlerini hızlandırma ve karar kalitesini arttırma süreçlerinde kullanılarak gerçek hayatta karşılaşılan problemlerin etkin, doğru ve gerçek zamanlı olarak çözümlenmesini sağlar [1]. Bu gibi sebeplerden dolayı optimizasyon algoritmaları gerçek hayatta karşılaşılan birçok problemin çözümü için sıklıkla kullanılmakta ve popülerliği sürekli artmaktadır.

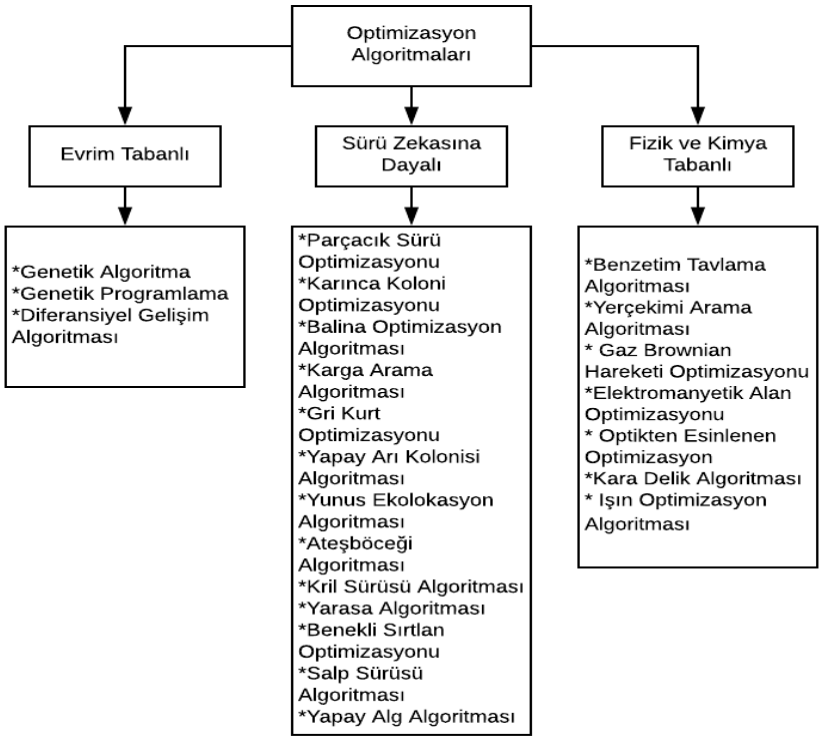
1.1. Literatür Çalışması

Optimizasyon problemleri Şekil 1.1'de görüldüğü gibi deterministik optimizasyon algoritmaları ve deterministik optimizasyon algoritmalarına rastgelelik eklenerek üretilen stokastik optimizasyon algoritmaları olarak ikiye ayrılır. Stokastik optimizasyon algoritmaları ise sezgisel algoritmalar ve metasezgisel algoritmalar olarak ikiye ayrılır.



Şekil 1.1. Optimizasyon Algoritmalarının Sınıflandırılması

Metasezgisel optimizasyon algoritmaları genellikle Şekil 1.1'de görüldüğü gibi evrim tabanlı algoritmalar, sürü zekasına dayalı algoritmalar, fizik ve kimya tabanlı algoritmalar olmak üzere üç ana kategoriye ayrılır.



Şekil 1.2. Metasezgisel Optimizasyon Algoritmalarının Sınıflandırılması

Evrim tabanlı algoritmalar, doğal sistemlerde var olan kalıtım prensibini temel alan algoritmalardır. 1975 yılında Holland tarafından önerilen Genetik Algoritma'da [2], bir başlangıç popülasyonu ile çözüme başlanır ve seleksiyon, çaprazlama, mutasyon işlemleri ile bir sonraki popülasyon üretilir [3]. 1992 yılında John Koza tarafından önerilen Genetik Programlama'da [4], Genetik Algoritma'da olduğu gibi seleksiyon, çaprazlama, mutasyon işlemleri mevcuttur. Genetik Algoritmadan farklı olarak çözümlerin gösterim şekli dizi değil ağaç yapısı şeklindedir. 1995 yılında K. Price tarafından önerilen Diferansiyel Gelişim Algoritması [5], kalıtım prensibine dayanan popüler evrim tabanlı optimizasyon algoritmalarındandır.

Sürü zekasına dayalı algoritmalar, kendi problemlerini çözmek için sınırlı yetenekleri olan fakat bir araya geldiklerinde muazzam bir şekilde organize olarak zor problemlerin çözümünü kolaylıkla gerçekleştirebilen canlı topluluğundan ilham alınarak geliştirilmiş algoritmalardır. Russell Eberhart ve James Kennedy 1995 yılında Kuş ve balık sürülerinin yiyecek arama davranışlarından esinlenerek Parçacık Sürü

Optimizasyonu'nu [6] önermişlerdir. Dorigo ve arkadaşları tarafından 1991'de karınca kolonisinde yer alan karıncaların sosyal davranışından esinlenen ve karınca kolonilerinin yiyecek arama davranışlarının matematiksel modeline dayanan Karınca Koloni Optimizasyonu [7] geliştirilmiştir. 2016 yılında Seyedali Mirjalili ve Andrew Lewis tarafından önerilen Balina Optimizasyon Algoritması [8], kambur balinaların sosyal davranışını modeller ve kabarcık-ağ avlama stratejisinden esinlenmiştir. 2016 yılında Alireza Askarzadeh tarafından önerilen Karga Arama Algoritması [9], kargaların fazla olan yiyeceklerini gizli bir yerde depolaması ve tekrar yiyecek lazım olduğunda buradan almaları fikrine dayanır. 2014 yılında Seyedali Mirjalili, Seyed Mohammad Mirjalili ve Andrew Lewis tarafından önerilen Gri Kurt Optimizasyonu [10], doğadaki gri kurtların liderlik hiyerarşisini ve avlanma mekanizmasını taklit eder. Derviş Karaboğa tarafından 2005 yılında bal arılarının yiyecek arama davranışını simüle eden Yapay Arı Kolonisi algoritması [11] önerilmiştir. 2013 yılında A. Kaveh ve N. Farhoudi tarafından yunusların avlanma ve navigasyon amacı ile kullandıkları ekolokasyon yeteneği simüle eden Yunus Ekolokasyon algoritması [12] önerilmiştir. 2009 yılında Xin-She Yang tarafından önerilen Ateşböceği Algoritması [13], tropikal iklim bölgelerindeki ateş böceklerinin sosyal davranışlarından esinlenilerek geliştirilmiştir. 2012 yılında Amir Hossein Gandomi ve Amir Hossein Alavi tarafından önerilen Kril Sürüsü algoritması [14], krillerin sürü halindeki davranışına dayanır. Xin-She Yang tarafından 2010 yılında yarasaların ekolokasyon davranışına dayanan Yarasa Algoritması [15] önerilmiştir. 2017 yılında Gaurav Dhiman ve Vijay Kumar tarafından önerilen Benekli Sırtlan Optimizasyonu algoritması [16], benekli sırtlanların işbirlikçi davranışlarına ve aralarındaki sosyal ilişkiye dayanır. Seyedali Mirjalili ve arkadaşları tarafından 2017 yılında önerilen Salp Sürüsü Algoritması [17], salpların okyanusta yön ve yiyecek arama davranışlarından esinlenilerek geliştirilmiştir. 2015 yılında Sait Ali Uymaz, Gülay Tezel ve Esra Yel tarafından önerilen Yapay Alg Algoritması [18], mikroalglerin evim süreci, adaptasyon süreci ve hareketine dayanır.

Fizik tabanlı algoritmalar, fizik kanunlarından esinlenilerek geliştirilmiş algoritmalardır. 1983 yılında Kirkpatrick tarafından önerilen Benzetim Tavlama algoritması [19], demir tavlama işleminden esinlenerek geliştirilmiştir. 2009 yılında Esmat Rashedi, Hossein Nezamadi-pour ve Saeid Saryazi tarafından geliştirilen Yerçekimi Arama Algoritması'nda [20], Newton'un hareket ve yerçekimi kanunlarından esinlenilmiştir.

2013 yılında Marjan Abdechiri, Mohammad Reza Meybodi ve Helena Bahrami tarafından önerilen Gaz Brownian Hareketi Optimizasyonu [21], gaz brownian hareketi ve rotasyonel gaz hareketlerine dayanır. 2016 yılında Hosein Abedinpourshotorban, Siti Mariyam Shamsuddin ve Zahra Beheshti tarafından önerilen Elektromanyetik Alan Optimizasyonu [22], elektromıknatısların davranışlarından esinlenilerek geliştirilmiştir. 2015 yılında Ali Husseinzadeh Kashan tarafından önerilen Optikten Esinlenen Optimizasyon [23] algoritması, iç bükey ve dış bükey ayna özelliklerinden esinlenilerek geliştirilmiştir. 2013 yılında Abdolreza Hatamlou tarafından önerilen Kara Delik algoritması [24], kara delik olayına dayanan bir algoritmadır. 2012 yılında A. Kaveh ve M. Khayatazad tarafından önerilen Işın Optimizasyon algoritması [25], Snell'in ışık kırılma yasasına dayanan fizik tabanlı algoritmalardandır.

2. BÖLÜM

GENEL BİLGİLER

2.1. Optimizasyon

Optimizasyon, bir problemin belli koşullar altında en optimum çözüm ya da çözümlerinin araştırılması ve bulunması işlemidir.

Günümüzde; maliyetin azaltılıp karın artırılması, zamanın faydalı kullanılması, enerji verimliliğinin artırılması, seri üretimi gerçekleştirilen herhangi bir ürünün sayısının artırılması ya da elektronik olarak çalışan bir cihazın boyutunun küçülmesi gibi çok farklı alanlarda optimizasyon işlemine ihtiyaç duyulmaktadır. Bu nedenle optimizasyon algoritmaları ile ilgili çalışmalar yıllar önce başlamış ve popülerliği giderek artacak şekilde devam etmektedir.

Bir problemin optimum çözümünü bulmak için gerçekleştireceğimiz optimizasyon işleminde öncelikle karar değişkenlerini (karar parametreleri/tasarım parametreleri) tanımlamak gerekir. Karar değişkenleri, optimizasyon algoritması sayesinde değerini belirlemeye çalıştığımız parametrelerdir.

Sonrasında ise söz konusu karar değişkenleri kullanılarak alternatif çözümlerin birbirleriyle kıyaslanmasına yarayan, çözümün kalitesinin ya da uygunluğunun ölçülmesini sağlayan ve optimizasyon algoritmasının uzayda yön bulmasını sağlayan amaç fonksiyonunun tanımlanması gerekir. Optimizasyon problemlerinde amaç fonksiyonu problemin türüne göre seçilir ve minimize edilecek maliyet fonksiyonu veya maksimize edilecek kar fonksiyonu olacak şekilde 2 çeşittir. Eğer amaç fonksiyonu bir maliyet fonksiyonu ise alternatif çözümlerden en düşük sonucu üreten çözüm en iyi çözüm iken eğer amaç fonksiyonu bir kar fonksiyonu ise alternatif çözümlerden en yüksek sonucu üreten çözüm en iyi çözümdür.

Tanımlanması gereken bir diğer şey ise optimizasyon problemi ile ilgili kısıtlama fonksiyonlarıdır ve bu fonksiyonlar parametrelerin alamayacağı değerleri belirtirler.

Karar değişkenlerini, amaç fonksiyonunu ve kısıtlama fonksiyonlarını örneklemek amacıyla " Bir fabrikada 4 kg ham madde kullanılarak A ürünü, 3 kg ham madde kullanılarak B ürünü üretilmektedir ve A ürünü satıldığında K TL kar B ürünü satıldığında L TL kar elde edilmektedir. Maksimum 40 kg ham madde kullanılarak ve en az 2 tane A ürünü üreterek en fazla ne kadar kar elde edilir." problemini ele alalım. Bu problemde A ürünü sayısına x_1, B ürünü sayısına ise x_2 dersek; karar değişkenleri x_1 ve x_2, kısıtlar Denkem (2.1) ve Denklem (2.2), amaç fonksiyonu ise Denklem (2.3)'te yer almaktadır.

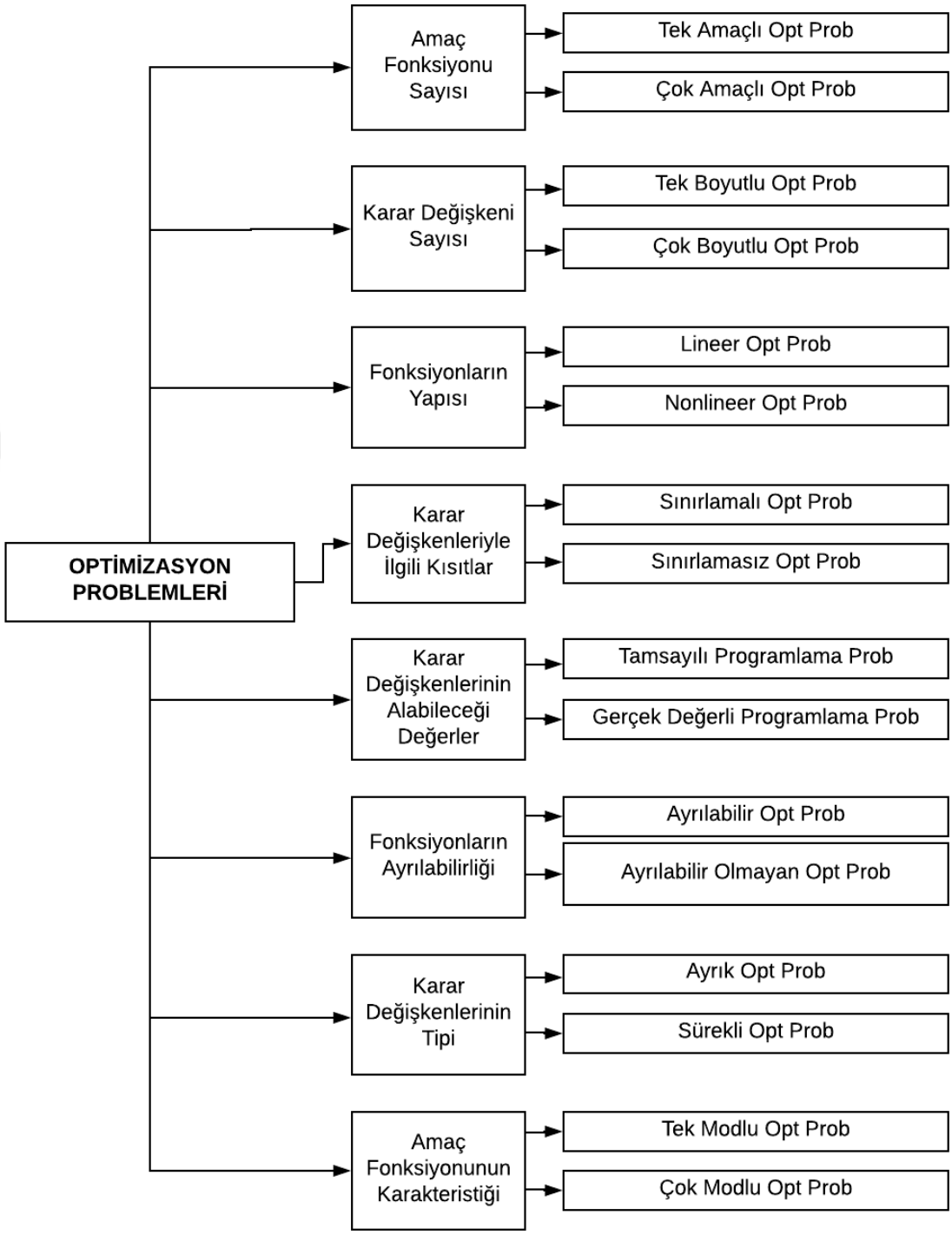
$$4.x_1 + 3.x_2 \leq 40 \quad (2.1)$$

$$x_1 \geq 2 \quad (2.2)$$

$$f(x)=K.x_1 + L.\ x_2 \quad (2.3)$$

2.2. Optimizasyon Problemlerinin Sınıflandırılması

Optimizasyon problemleri amaç fonksiyonlarının, kısıtlama fonksiyonlarının ve karar değişkenlerinin çeşitli özelliklerine göre Şekil 2.1'de görüldüğü gibi sınıflandırılır.



Şekil 2.1. Optimizasyon Problemlerinin Sınıflandırılması

Optimizasyon problemleri; sadece bir tane amaç fonksiyonu olduğunda tek amaçlı, birden fazla amaç fonksiyonu olduğunda ise çok amaçlı optimizasyon problemi olarak isimlendirilir. Bazı optimizasyon problemlerinde çözümün kalitesini değerlendirmek için birden fazla kriter söz konusu olup bu kriterlerin tek bir amaç fonksiyonunda

toplanması olanaklı olmayabilir. Bu gibi durumlarda birden fazla amaç fonksiyonu tanımlanır. Tanımlanan bu amaç fonksiyonlarının birinin minimize edilmesi gerekirken diğerinin maksimize edilmesinin gerekmesi gibi durumlarda ise çok amaçlı optimizasyon problemlerinin çözülmesi tek amaçlı optimizasyon problemlerine göre daha zor bir hal alır.

Optimizasyon problemlerinde karar değişkenlerinin sayısı problemin boyutunu verir. Eğer karar değişkeni sadece bir tane ise tek boyutlu optimizasyon problemi olarak adlandırılırken karar değişkeninin sayısının birden fazla olmasında ise çok boyutlu optimizasyon problemi olarak adlandırılır.

Bir optimizasyon probleminde amaç ve sınırlama fonksiyonları lineer ise lineer optimizasyon problemi, amaç ve sınırlama fonksiyonlarından herhangi birisi nonlieer ise nonlineer optimizasyon problemi olarak isimlendirilir.

Optimizasyon problemleri karar değişkenleri ile ilgili bir sınırlaması olup olmamasına göre de sınıflandırılabilir. Eğer karar değişkenleri ile ilgili bir sınırlama varsa sınırlama fonksiyonları ile tanımlanır ve bu tür optimizasyon problemleri sınırlamalı optimizasyon problemleri olarak adlandırılırken, karar değişkenleri ile ilgili bir sınırlama yoksa sınırlamasız optimizasyon problemleri olarak adlandırılır.

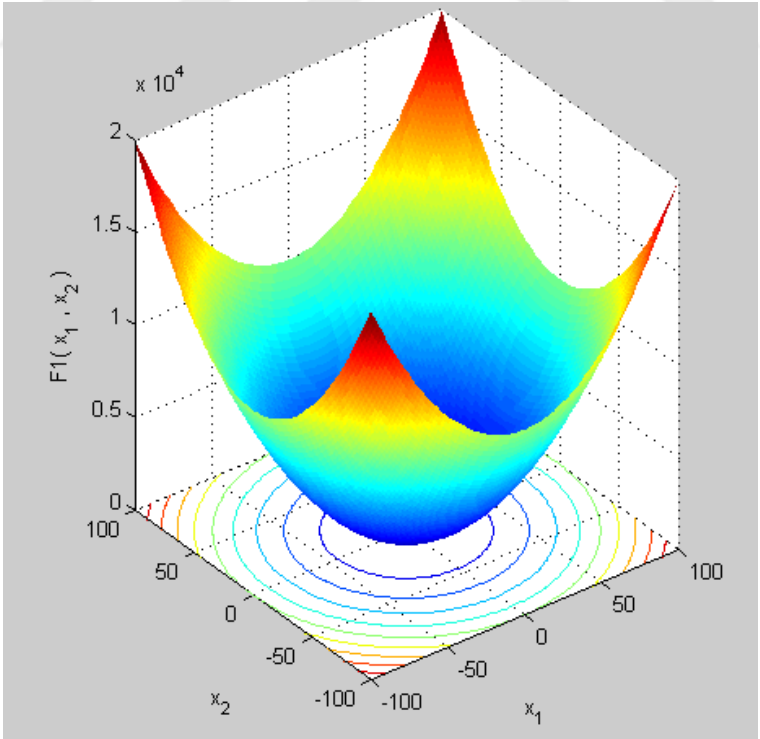
Bir optimizasyon probleminde amaç fonksiyonunu oluşturan karar değişkenlerinin tamamı ya da bir kısmı tamsayı değerler alması ile kısıtlandırılıyorsa bu tür problemler tamsayılı programlama problemi olarak adlandırılır. Eğer optimizasyon probleminde karar değişkenleri gerçek değerler alabiliyorsa gerçek değerli programlama problemi olarak adlandırılır.

Optimizasyon problemleri amaç ve kısıt fonksiyonlarının ayrılabilirliğine göre de sınıflandırılabilir. Eğer fonksiyonlar alt fonksiyonlara ayrılabiliyorsa bunlar ayrılabilir optimizasyon problemleri olarak isimlendirilirken, fonksiyonların alt fonksiyonlara ayrılamadığı problemlerde ise ayrılabilir olmayan optimizasyon problemleri olarak isimlendirilir.

Optimizasyon problemleri ayrık optimizasyon problemleri ve sürekli optimizasyon problemleri olarak da sınıflandırılabilir. Yapılacak işlerin olduğu bir listede en uygun

sıralamanın bulunması işlemi gibi, ayrık niceliklerin (objelerin) en optimum şekilde düzenlenmesi, gruplandırılması, sıraya konulması ve seçilmesi problemlerine ayrık optimizasyon problemleri denir. Bu tür problemlerde optimum çözüm belirli bir diziliş ile elde edildiği için bu tür problemler kombinasyonel optimizasyon problemi olarak da adlandırılır. Karar değişkenlerinin alacağı değerlerin sürekli değerler olduğu optimizasyon problemlerine ise sürekli optimizasyon problemleri denir.

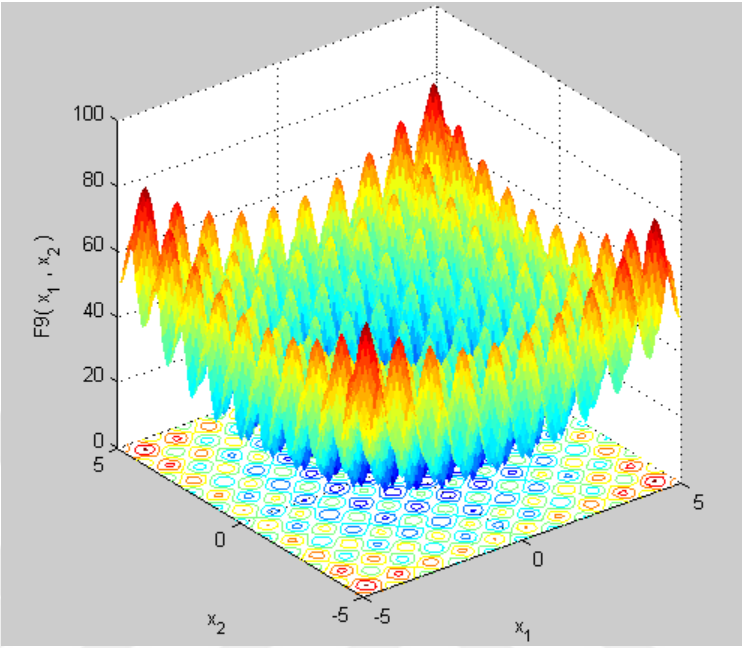
Optimizasyon problemlerini amaç fonksiyonunun karakteristiğine göre tek modlu optimizasyon problemleri ve çok modlu optimizasyon problemleri olarak sınıflandırılabilir [26]. Tek modlu fonksiyonlar sadece bir adet vadi veya tepeye sahiptirler. Bu fonksiyonlardaki lokal optimum değeri, fonksiyon sadece bir tane vadi veya tepe değerine sahip olduğu için aynı zamanda global optimumdur. Tek modlu fonksiyonlar konveks (dış bükey) yapıdadırlar ve amaç fonksiyonunun iyileşme yönü daima global optimum değere doğrudur. Şekil 2.2'de tek modlu benchmark fonksiyonlarından, $f(x)=\sum_{i=1}^{n} x_i^2$ ile ifade edilen Sphere yer almaktadır.



Şekil 2.2. Tek Modlu Sphere Fonksiyonunun Grafiği

Çok modlu fonksiyonlarda ise birden fazla vadi ve tepe bulunur. Bu fonksiyonlarda birden fazla olacak şekilde bulunan lokal optimum noktalardan biri global optimum

noktadır. Şekil 2.3'te çok modlu benchmark fonksiyonlarından, $f(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$ ile ifade edilen Rastrigin yer almaktadır.



Şekil 2.3. Çok Modlu Rastrigin Fonksiyonu

2.3. Optimizasyon Algoritmalarının Sınıflandırılması

Bölüm 2.2'de tanımlanan optimizasyon problemlerinin çözümü için geliştirilmiş algoritmalar optimizasyon algoritmaları olarak isimlendirilir.

Genel olarak optimizasyon algoritmaları deterministik optimizasyon algoritmaları ve stokastik optimizasyon algoritmaları olarak iki ana kategoriye ayrılır.

Deterministik algoritmalar aynı başlangıç değerleri kullanılarak gerçekleştirilen her koşturmada aynı sonuçları üretir ve problemin kesin çözümünü garanti ederler. Stokastik algoritmalar ise rastgelelik içerdiğinden dolayı aynı başlangıç değerleri kullanılarak koşturulsalar bile genellikle birbirlerine çok yakın fakat farklı sonuçlar üretirler ve kesin çözümü garanti etmeyip kesin çözüm yakınındaki çözümü garanti ederler [27]. Deterministik algoritmalar kolay problemlerin çözümünde ve lokal optimumu bulma konusunda oldukça başarılıdırlar fakat bu algoritmaların lokal optimuma yakalanıp global optimuma ulaşamama riskleri mevcuttur. Deterministik algoritmaların lokal optimuma yakalanmasını önlemek amacıyla algoritmaya rastgelelik eklenerek stokastik yetenekler kazandırılır [28].

Stokastik algoritmalar sezgisel algoritmalar ve metasezgisel algoritmalar olarak ikiye ayrılır. Sezgisel optimizasyon algoritmaları belli bir probleme göre özel olarak geliştirilen ve doğru çalıştığı mantıksal olarak doğrulanabilen algoritmalardır. FIFO (First In First Out- İlk Giren İlk Çıkar), LIFO (Last In First Out- Son Giren İlk Çıkar) ve sıralama algoritmaları sezgisel algoritmalara örnek olarak verilebilir. Metasezgisel optimizasyon algoritmaları ise uygulama alanı sezgisel algoritmalara göre daha geniş olan algoritmalardır. Metasezgisel algoritmalar gerçekleştirilecek küçük değişikler ile çeşitli problemlere uygulanabilir hale getirilebilen ve kendi içerisinde sezgisel algoritmaları kullanan algoritmalardır. Metasezgisel algoritmalar zor ve karmaşık problemler için makul süre içerisinde kabul edilebilir çözümler sunarlar [29].

Metasezgisel optimizasyon algoritmalarının son zamanlarda optimizasyon problemlerinin çözümü için sıklıkla tercih edilmesi

- Kullanımının gerçek hayat problemleri için uygun olması,
- Çok çeşitli problemlere kolaylıkla uyarlanabilir olması,
- Stokastik yetenekleri sayesinde lokal optimizasyona yakalanma riskinin az olması,
- Optimumu bulmak için herhangi bir türev bilgisinin gerekmemesi,
- Genellikle doğadan ilham alınarak geliştirildiği için anlaşılması ve analiz edilmesinin kolaylığı

sebeplerine dayanmaktadır [8,10].

3. BÖLÜM

POPÜLASYON TABANLI OPTİMİZASYON ALGORİTMALARI

3.1. Parçacık Sürü Optimizasyonu (PSO)

Parçacık Sürü Optimizasyonu Russell Eberhart ve James Kennedy tarafından 1995 yılında önerilmiş popülasyon temelli sezgisel bir optimizasyon algoritmasıdır.

PSO, kuş ve balık sürülerinin yiyecek arama davranışlarından esinlenerek geliştirilmiştir ve her bir birey parçacık, parçacıkların oluşturduğu popülasyon ise sürü olarak isimlendirilmiştir [30, 31].

Her bir parçacığın bir sonraki konumu hesaplanırken iki en iyi parçacığa göre işlem yapılır. Bunlardan ilki sürüdeki bir parçacığın o ana kadar sahip olduğu en iyi değerdir ve buna yerel en iyi denir. İkincisi ise sürüdeki her bir parçacığın en iyi değerlerinin kıyaslanmasıyla bulunmuş sürünün en iyi değeridir ve buna da global en iyi denir. Kısacası herhangi bir parçacığın bir sonraki konumu hesaplanırken hem parçacığın kendi deneysel bilgisi (bilişsel bileşen) hem de sosyal olarak komşularından elde etmiş olduğu bilgi (sosyal bileşen) kullanılır.

3.1.1. Parçacığın Konumunun Hesaplanması

Parçacık Sürü Optimizasyonu algoritmasında sürüdeki her bir parçacığın bir sonraki konumunun bulunması Denklem (3.1)'de ifade edildiği gibi mevcut konumuna V hız vektörü eklenmesiyle elde edilir.

$$X_i(t+1)=X_i(t)+V_i(t+1) \tag{3.1}$$

Denklem (3.1)'de i indeks değeri sürüdeki i. parçacığı gösterir ve 1 ile sürünün boyutu arasında herhangi bir değeri alabilir. $X_i(t)$ değeri t anında i. parçacığın konumunu, $V_i(t+1)$ ise t+1 anında i. parçacığın hızını ifade eder.

3.1.2. Parçacığın Hızının Hesaplanması

Parçacık Sürü Optimizasyon algoritmasında hız hesabı gerçekleştirilirken hem bilişsel bileşenlerden hem de sosyal bileşenlerden istifade edilir. Hız hesabının gerçekleştirilmesindeki bahse konu bilişsel bileşen her bir parçacığın sahip olduğu en iyi konum bilgisini ifade eder ve p_i olarak gösterilir; sosyal bileşen ise her bir parçacığın en iyi konum bilgilerinin kıyaslanmasıyla elde edilen sürüdeki en iyi konumu ifade eder ve p_{best} olarak gösterilir.

PSO'da parçacığın hızı Denklem (3.2)'de belirtildiği gibi hesaplanır.

$$V_{i,j}(t+1)= V_{i,j}(t)+c_1.r_{1,j}(t).\,[p_{i,j}(t)-X_{i,j}(t)] + c_2.r_{2,j}(t).\,[p_{bestj}(t)-X_{i,j}(t)] \tag{3.2}$$

Denklem (3.2)'deki c_1 ve c_2 değerleri formülde yer alan bilişsel bileşen ve sosyal bileşenlerin katkısını ölçeklemek amacıyla kullanılan pozitif değerlerdir. $r_{1,j}(t)$ ve $r_{2,j}(t)$ ise [0 1] aralığında rastgele değer alan sayılardır ve bu sayılar algoritmanın stokastik özelliğini etkilemek için kullanılmaktadır.

Denklem (3.2)'de yer alan ve yukarıda açıklanan p_i ve p_{best} değerleri hesaplanırken ise herhangi bir i parçacığının konumlarının *f* uygunluk fonksiyonundaki değeri hesaplanır ve böylece parçacığın (çözümün) kalitesi hesaplanmış olur. Minimizasyon problemlerinde söz konusu i parçacığının f uygunluk fonksiyonundaki değerinin küçük olması o parçacığın kalitesinin daha iyi olduğu anlamına gelmektedir. Minimizasyon problemleri dikkate alındığında p_i değeri Denklem (3.3)'te, p_{best} değeri ise Denklem (3.4)'te ifade edildiği gibidir.

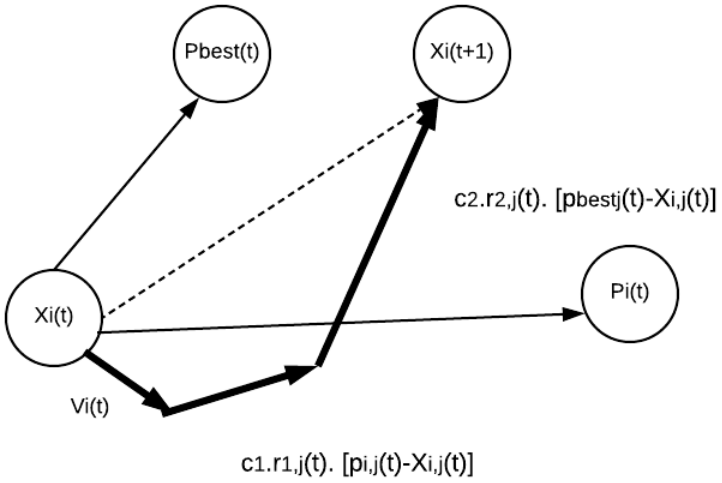
$$p_i(t+1)=\begin{cases} p_i(t) \quad , f(X_i(t+1)) \geq f(X_i(t)) \\ X_i(t+1)), f(X_i(t+1)) < f(X_i(t)) \end{cases} \tag{3.3}$$

$$p_{best}(t) \in \{ p_0(t), \ldots, p_n(t)\} \mid f(p_{best}(t))= \min\{ f(p_0(t)), \ldots, f(p_n(t))\} \tag{3.4}$$

Denklem (3.4)'te yer alan n değeri sürüdeki toplam parçacık sayısını ifade etmektedir.

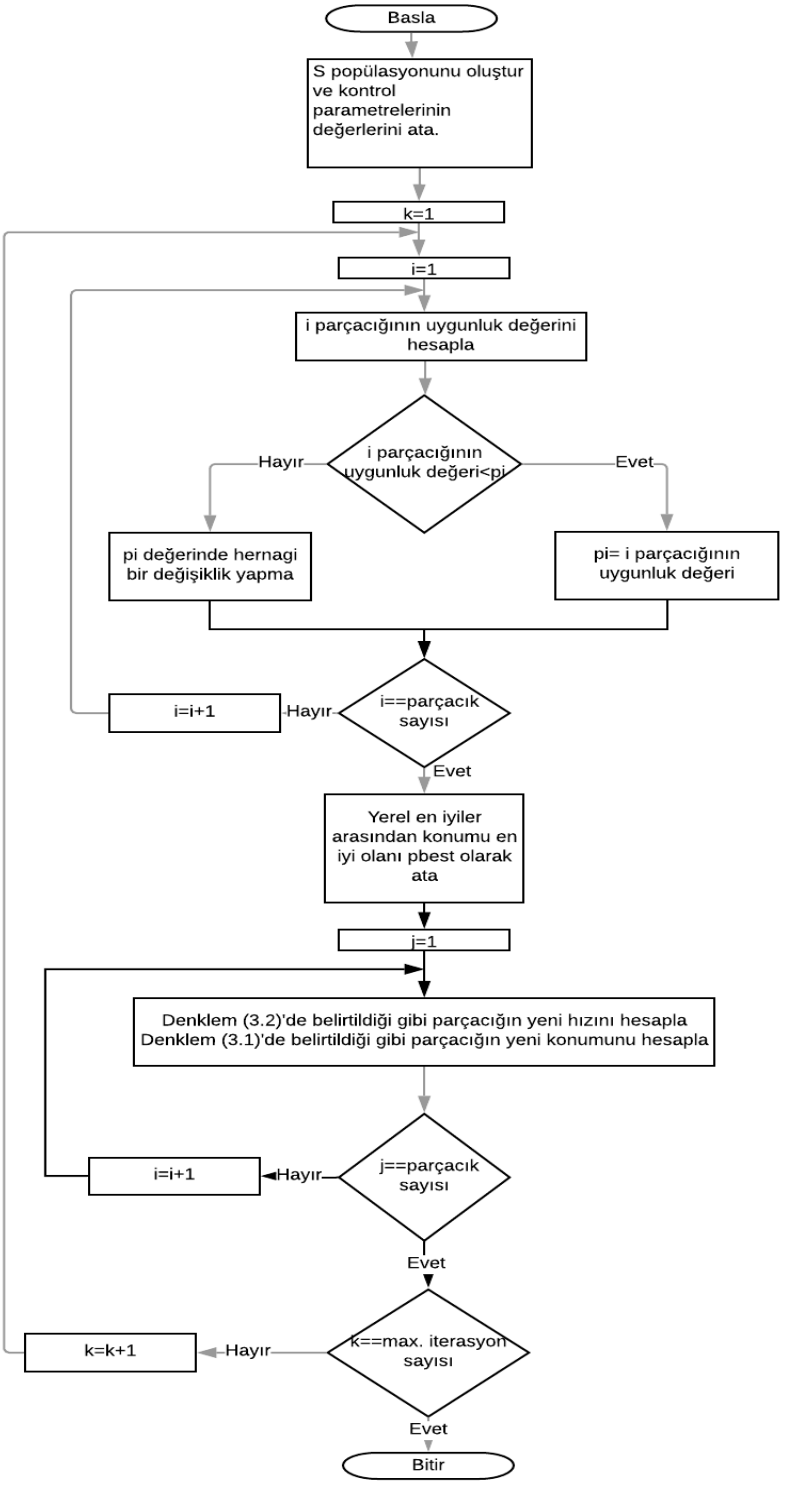
Parçacık Sürü Optimizasyon algoritmasına göre konum güncelle işlemi Şekil 3.1'de özetlenmiştir. Söz konusu şekilde parçacığın o ana kadar sahip olduğu en iyi konum (p_i)

ve sürüdeki en iyi konum (p_{best}) kullanılarak hesaplanan hız vektörünün, parçacığın mevcut konumuna eklenmesiyle elde edilen yeni konumu görselleştirilmiştir.



Şekil 3.1. Parçacık Sürü Optimizasyon Algoritmasında Konum Güncelleme İşlemi

3.1.3. Parçacık Sürü Optimizasyon Algoritması Akış Diyagramı



Şekil 3.2. Parçacık Sürü Optimizasyon Algoritması Akış Diyagramı

3.2. Diferansiyel Gelişim Algoritması (DGA)

Diferansiyel Gelişim Algoritması, K. Price tarafından 1995 yılında önerilmiştir. Diferansiyel Gelişim Algoritması popülasyon tabanlı sezgisel bir optimizasyon algoritması olup doğal sistemlerde var olan ve kalıtım prensibini temel alan evrimsel algoritmalardan bir tanesidir.

Doğadaki bir popülasyonun gelişimi 3 operasyona bağlıdır. Diferansiyel Gelişim Algoritması ise bu 3 operasyonu modelleyen algoritmalardan bir tanesidir. Söz konusu operasyonlar aşağıda verilmiştir;

a) Değişim

   Popülasyonda yer alan bireyler gelişmek amacıyla yarışsa da yardımlaşa da bireylerin gelişimi belli bir noktaya kadar iyileşme göstermesine rağmen bir noktadan sonra gelişim olmaz. Gelişimin devamlı olabilmesi için değişim şarttır. Bu değişim popülasyondaki bireylerin gelişmesine katkı sağlayacak kadar çok ve var olan bilginin kaybolmasına engel olacak kadar az olmalıdır.

b) Yardımlaşma

   Popülasyondaki bireylerin gelişiminin hızlanması için bilgi alışverişinin olması gerekir. Bu bilgi alışverişi bireylerin gelişimini hızlandıracak kadar fazla ve bireylerin birbirine benzemesine engel olacak kadar az olmalıdır.

c) Yarışma

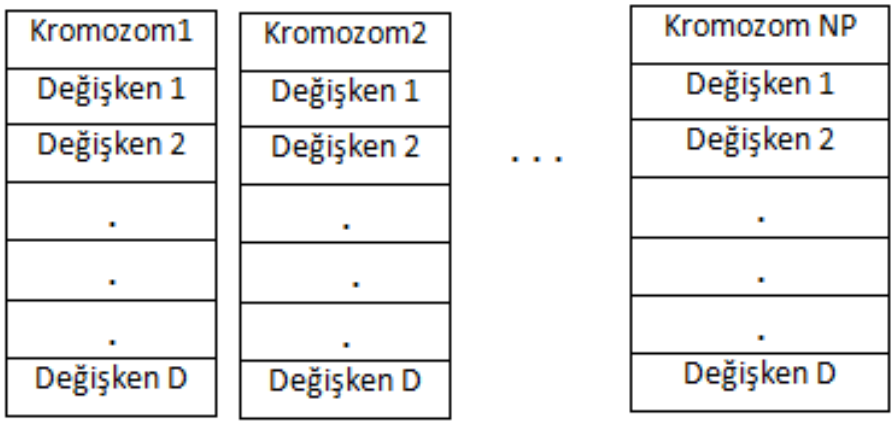
   Bu operasyon popülasyondaki bireyler arasındaki rekabeti artırarak bireylerin gelişimini ve çevreye adaptasyon hızını artırır. Eğer yarışma olmazsa popülasyonda yer alan bireyler pasifleşir.

Yukarıda bahsedilen 3 operasyonun evrimsel algoritmalarda ve Diferansiyel Gelişim Algoritmasındaki modellenmesi mutasyon, çaprazlama ve seleksiyon işlemleri ile gerçekleştirilir.

3.2.1. Başlangıç Popülasyonunun Oluşturulması

Optimizasyon algoritmalarında çözümler arasından en optimumu bulmak ve söz konusu çözümleri amaç fonksiyonuna entegre edebilmek için çözümlerin gösterimi büyük önem taşımaktadır.

Diferansiyel Gelişim Algoritmasında popülasyondaki her bir birey (çözüm) kromozomdur. Diferansiyel Gelişim Algoritmasında NP, popülasyondaki kromozom sayısını ve D, kromozomun boyutunu temsil etmek üzere NP tane D boyutlu kayan-noktalı diziden oluşur. Söz konusu popülasyon Şekil 3.3'teki gibidir.



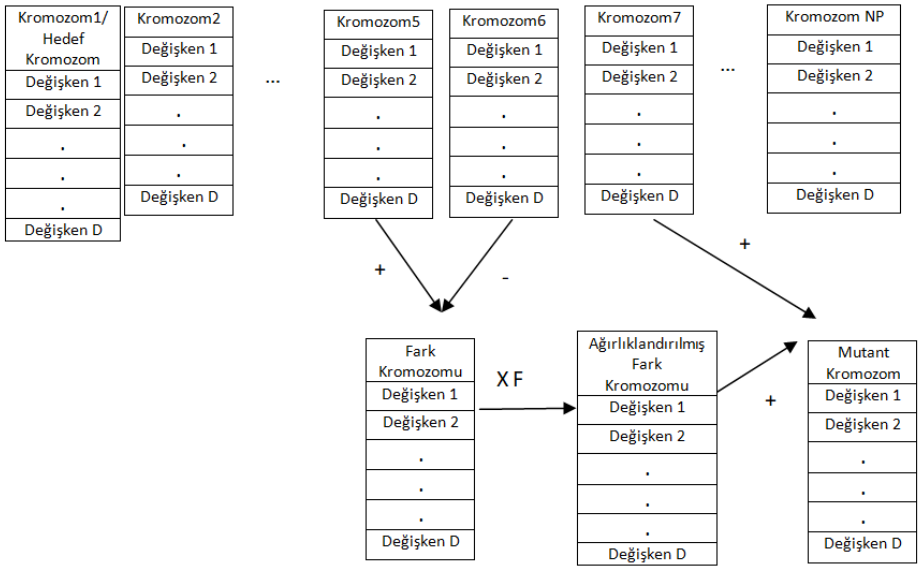
Şekil 3.3. NP tane D boyutlu kromozomdan oluşan popülasyon

3.2.2. Mutasyon

Diferansiyel Gelişim Algoritmasında mutasyon, kromozomların üzerindeki rastgele değişikliklerdir. İyi bir mutasyon işleminin gerçekleştirmesi gereken şey, mutasyona uğrayacak olan kromozomun doğru zamanda doğru miktar ile doğru yönde hareket etmesini sağlayacak adımları üretmektedir [27].

Diferansiyel Gelişim Algoritmasında herhangi bir çözümün (kromozomun) mutasyona uğraması için söz konusu çözüm haricinde rastgele 3 tane çözüm seçilir ve bu rastgele seçilmiş 3 adet çözümün içinden yine rastgele seçilen 2 tane çözümün farkı alınarak 0 ile 2 arasında değer alabilen F (Ölçeklendirme Faktörü) ile çarpılır. Çarpım neticesinde elde edilen sonuç kalan diğer çözüm ile toplanır ve mutant ya da mutasyona uğramış kromozom elde edilir. Elde edilen bu mutant vektör çaprazlamada kullanılacak olan çözümdür.

Bahse konu mutasyon işleminin gerçekleşebilmesi için herhangi bir çözümün mutasyona uğrama sırasında kendisi haricinde 3 tane çözüme ihtiyaç duyulması sebebi ile Diferansiyel Gelişim Algoritmasında popülasyon büyüklüğünün 3'ten büyük olması gerekmektedir.



Şekil 3.4. Mutasyon İşlemi

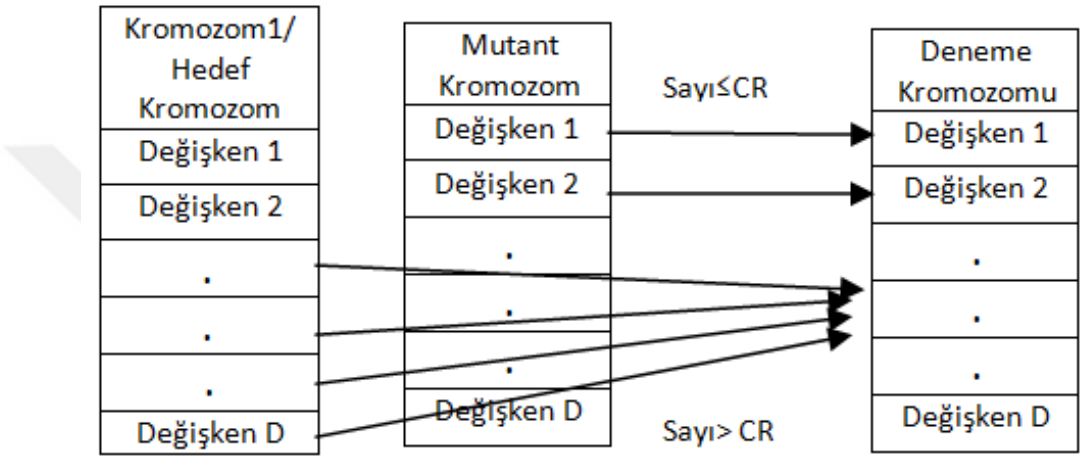
Şekil 3.4'te NP adet kromozomdan oluşan popülasyonun içinden Kromozom 5, Kromozom 6 ve Kromozom 7 olmak üzere 3 tane rastgele çözüm seçilmiştir. Söz konusu bu çözümlerin içerisinden tekrar rastgele Kromozom 5 ve Kromozom 6 seçilip farkı alınarak Fark Kromozomu oluşturulmuştur. Oluşturulan Fark Kromozomu F ile çarpılarak Ağırlıklandırılmış Fark Kromozomu oluşturulmuş ve bu kromozomun kalan diğer kromozom ile toplanmasının ardından Mutant Kromozom elde edilmiştir. Elde edilen Mutant Kromozom, Hedef Kromozom olan Kromozom 1 ile çaprazlanacak olan çözümdür.

3.2.3. Çaprazlama

Çaprazlama işlemi popülasyonda bulunan mevcut kromozom çeşitliliğini artırmak için kullanılır ve çaprazlama işlemi sonunda elde edilen vektöre deneme vektörü denir. Bu deneme vektörü hedef kromozom (ebeveyn) ile değiştirilme ihtimali olan kromozomdur.

Çaprazlama işlemi hedef kromozom ile mutant kromozom arasında gerçekleştirilir ve çaprazlama işlemi neticesinde oluşan deneme vektörünün bazı elemanları hedef

kromozodan (ebeveyden) bazıları ise mutant kromozomdan meydana gelir. Bahse konu deneme vektörünün hangi elemanını hangi vektörden alacağı ise CR (Crossover Rate) olarak isimlendirilen, 0 ile 1 arasında değer alan kontrol parametresi ile gerçekleştirilir. Bu işlem için 0 ile 1 arasında rastgele olarak üretilen sayının CR'den küçük veya eşit olması durumunda deneme vektörü mutant vektörden eleman alırken sayının CR'den büyük olması durumunda ise deneme vektörü elemanını ebeveynden alır.



Şekil 3.5. Çaprazlama İşlemi

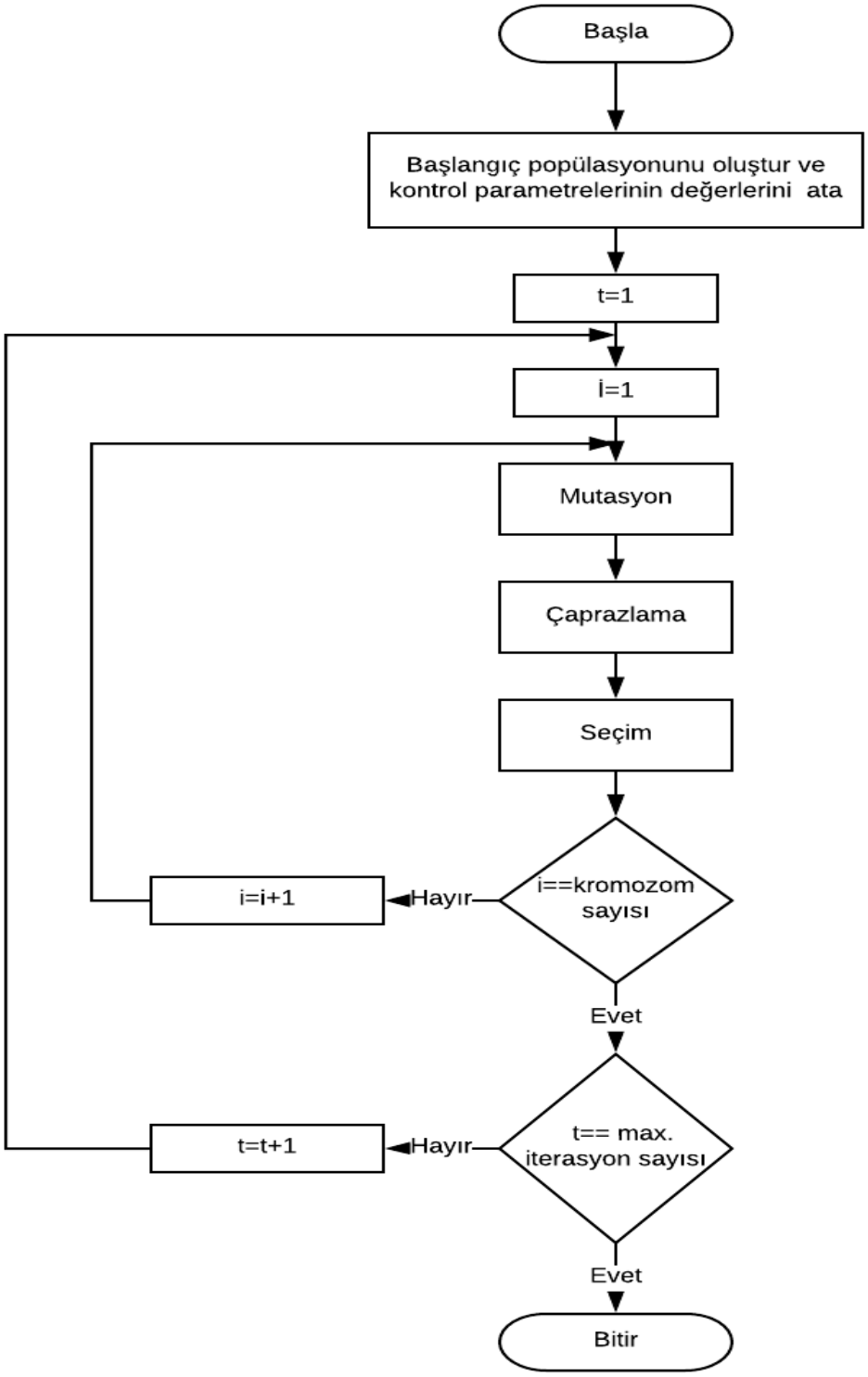
Şekil 3.5'te Hedef Kromozom ile Mutant Kromozomun çaprazlanma işlemi gösterilmiştir Söz konusu şekilde Değişken 1 ve Değişken 2 için rastgele üretilen sayının CR değerinden küçük olması nedeniyle Deneme Kromozomuna Mutant Kromozomun elemanının seçildiği, diğer değişkenler için ise rastgele üretilen sayının CR değerinden büyük olması nedeniyle Deneme Kromozomuna Hedef Kromozomun elemanının seçildiği simüle edilmiştir.

3.2.4. Seçim

Diferansiyel Gelişim Algoritmasında hangi kromozomun seçilerek popülasyonda yer alacağını belirlemek için yarıştırılacak 2 kromozom için amaç fonksiyonu kullanılır ve kalite değeri yüksek olan kromozom popülasyonda yerini alır.

Diferansiyel Gelişim Algoritmasında seçim için yarıştırılacak kromozomlar Hedef Kromozom ve çaprazlama neticesinde oluşmuş olan Deneme Kromozomudur.

3.2.5. Diferansiyel Gelişim Algoritması Akış Diyagramı



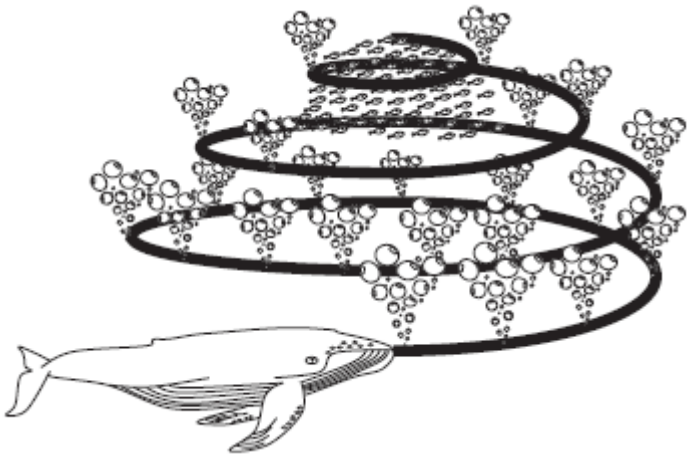
Şekil 3.6. Diferansiyel Gelişim Algoritması Akış Diyagramı

3.3. Balina Optimizasyon Algoritması (BOA)

Balina Optimizasyon Algoritması Seyedali Mirjalili ve Andrew Lewis tarafından 2016 yılında önerilmiştir. Balina Optimizasyon Algoritması kambur balinaların avlanma davranışlarını simüle eden metasezgisel bir algoritmadır.

Balina Optimizasyon Algoritmasının esin kaynağı olan kambur balinaların en çok tercih ettiği av, kril ve küçük balık sürüleridir. Kambur balinalar ile ilgili en ilginç şey ise 'Kabarcık-ağ' adı verilen, kendilerine özgü olan ve sadece kambur balinalarda görülen avlanma metodudur [32].

Şekil 3.7.'de görüldüğü gibi kambur balinalar avlanma sırasında yukarı doğru daralan spiral hareket gerçekleştirirler.



Şekil 3.7. Kambur Balinaların Kabarcık-ağ Metodu İle Avlanması [8]

3.3.1. Avın Kuşatılması

Kambur balinalar avının konumunu belirleyebilir ve avını kuşatabilir. Balina Optimizasyon Algoritması mevcut en iyi çözümü av olarak ya da ava en yakın çözüm olarak kabul eder. En iyi arama ajanı belirlendikten sonra ise diğer arama ajanlarının konumları en iyi arama ajanına göre güncellenir. Bu davranış için kullanılan denklemler aşağıda yer almaktadır.

$$\vec{D} = \left| \vec{C}\vec{X^*}(t) - \vec{X}(t) \right| \tag{3.5}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X^*}(t) - \overrightarrow{A}.\overrightarrow{D} \quad (3.6)$$

Denklem (3.5)'te ve Denklem (3.6)'da bulunan t iterasyon sayısını, $\vec{A}$ ve $\vec{C}$ katsayı vektörünü, X* mevcut en iyi konum vektörünü ve $\vec{X}$ ilgili arama ajanının konum vektörünü ifade eder.

Her bir iterasyon sonunda X* değerinden daha iyi bir çözüm olup olmadığına bakılır. Eğer daha iyi bir çözüm varsa X* değeri güncellenir.

$\vec{A}$ ve $\vec{C}$ katsayı vektörünün hesaplanabilmesi için kullanılan denklemler ise aşağıdaki gibidir.

$$\overrightarrow{A} = 2\overrightarrow{a}.\overrightarrow{r} - \overrightarrow{a} \quad (3.7)$$

$$\overrightarrow{c} = 2.\overrightarrow{r} \quad (3.8)$$

Denklem (3.7)'de ve Denklem (3.8)'de bulunan $\vec{r}$ [0 1] aralığında rastgele değer alır. Denklem (3.7)'de yer alan $\vec{a}$ ise iterasyon boyunca 2'den 0'a kadar lineer olarak azalır.

3.3.2. Kabarcık-Ağ Metodu

Kambur balinaların kabarcık-ağ davranışı aşağıda yer alan iki yaklaşımla matematiksel olarak modellenmiştir.

3.3.2.1. Daralan Kuşatma Mekanizması

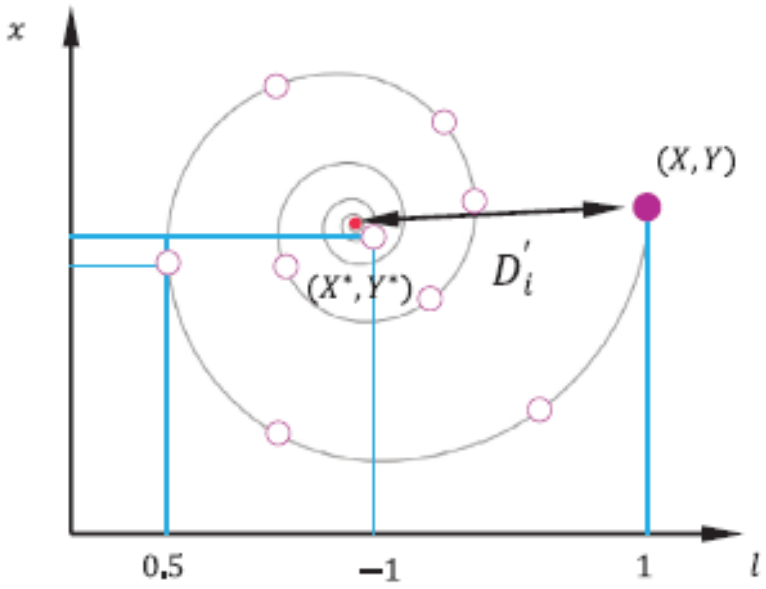
Bu mekanizma Denklem (3.7)'de yer alan $\vec{a}$ değerinin azalması sayesinde olur. Aynı zamanda $\vec{A}$'nın alabileceği değer aralığındaki dalgalanma da a değerinin iterasyon boyunca 2'den 0'a kadar azalması ve $\vec{A}$'nın [-a,a] aralığında değer alabilmesi sebebiyle a tarafından azaltılır.

3.3.2.2. Spiral Hareket

Bu yaklaşıma göre Şekil 3.8.'de görüldüğü gibi öncelikle (X,Y) konumundaki balina ile (X*,Y*) konumundaki av arasındaki mesafe hesaplanır. Sonrasında kambur balinaların sarmal şeklindeki hareketini modelleyebilmek için Denklem (3.9) kullanılır.

$$\vec{X}(t+1)=\vec{D'}e^{bl}.\cos(2\pi l)+\vec{X^*}(t) \tag{3.9}$$

Denklem (3.9)'da b, spiralin modelini belirleyebilmek için kullanılan bir sabit; l, [-1,1] aralığında rastgele bir sayı; $\vec{D}$ ise balina ile av (o ana kadar elde edilmiş en iyi çözüm) arasındaki mesafedir ve $\vec{D'}=\left|\vec{X^*}(t)-\vec{X}(t)\right|$ formülü ile hesaplanır.



Şekil 3.8. Spiral Hareket [8]

Kambur balinalar avının etrafında eş zamanlı olarak hem daralan kuşatma mekanizmasına göre hem de spiral harekete göre yüzerler. Balina Optimizasyon Algoritmasında bu eş zamanlı hareketi modelleyebilmek için daralan kuşatma mekanizmasının ya da spiral hareketin %50 olasılık oranı ile seçilebileceği farz edilmiştir ve matematiksel modeli Denklem (3.10)'daki gibidir.

$$\vec{X}(t+1)=\begin{cases}\vec{X^*}(t)-\vec{A}.\vec{D} & p<0.5\\ \vec{D'}e^{bl}.\cos(2\pi l)+\vec{X^*}(t) & p\geq 0.5\end{cases} \tag{3.10}$$

Denklem (3.10)'da yer alan p, [0-1] aralığında rastgele bir sayıdır.

3.3.3. Av Arama

Kambur balinalar Kabarcık-Ağ metodu dışında avlarını rastgele olarak da avlayabilirler. Rastgele av aramada referans balinadan uzaklaşabilmek için $\vec{A}$ vektörünün 1'den daha büyük ya da -1'den daha küçük rastgele aldığı değerlerinden faydalanılır. Av Arama mekanizmasında Kabarcık-Ağ mekanizmasından farklı olarak arama ajanının konumu güncellenirken şimdiye kadar bulunmuş en iyi arama ajanı yerine rastgele seçilen bir arama ajanı kullanılır. $|\vec{A}|>1$ olması ve rastgele bir arama ajanının kullanılması Balina Optimizasyon Algoritmasının global arama yapmasına neden olur.

Bu mekanizmada kullanılan matematiksel model ise Denklem (3.11) ve Denklem (3.12) ile gösterilmiştir.

$$\overrightarrow{D} = \left| \overrightarrow{C}.\overrightarrow{Xrand} - \overrightarrow{X} \right| \tag{3.11}$$

$$\overrightarrow{X}\left(t+1\right) = \overrightarrow{Xrand} - \overrightarrow{A}.\overrightarrow{D} \tag{3.12}$$

3.3.4 Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı

Balina Optimizasyon Algoritmasının sahte kodu:

a, A, C, l, p, maksimum iterasyon sayısı, popülasyon büyüklüğü değerlerinin ilk atamasını yap ve popülasyonu oluştur.

for: her bir arama ajanı için

Uygunluk değerini hesapla

end

X*=Bilinen en iyi arama ajanı

t=0

while (t< maksimum iterasyon sayısı)

for: her bir arama ajanı için

Güncelle a, A, C, l, p

if (p<0.5)

if (|A| <1)

Denklem(3.6)'ya göre arama ajanının konumunu güncelle

else if (|A| >=1)

Rastgele bir arama ajanı seç (Xrand)

Denklem(3.12)'ye göre arama ajanının konumunu güncelle.

end if

else if (p>=0.5)

Denklem (3.9)'e göre arama ajanının konumunu güncelle.

end if

end for

Arama ajanının kısıt dışına çıkıp çıkmadığına bak çıkıyorsa sınır değerini ver

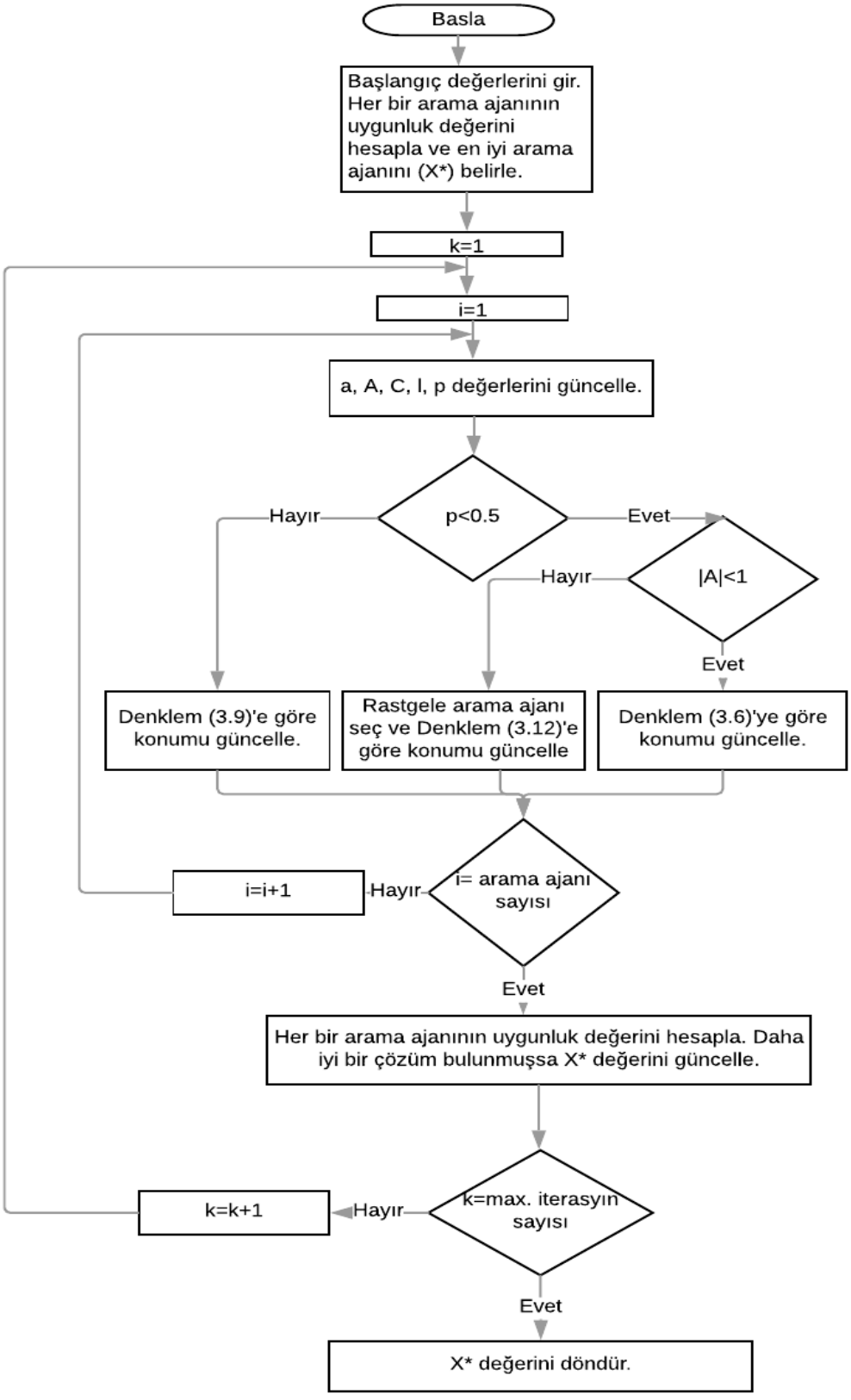
Her bir arama ajanının uygunluk değerini hesapla, daha iyi bir çözüm varsa X* değerini güncelle.

t=t+1

end while

X* değerini döndür.

Balina Optimizasyon Algoritması akış diyagramı:



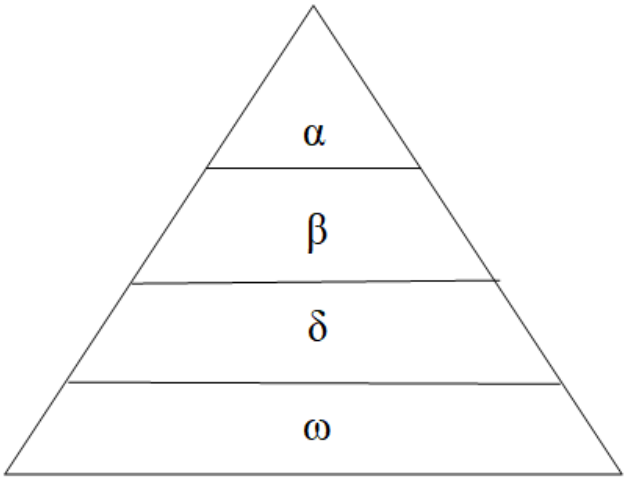
Şekil 3.9. Balina Optimizasyon Algoritması Akış Diyagramı

3.4. Gri Kurt Optimizasyonu (GKO)

Gri Kurt Optimizasyonu Seyedali Mirjalili, Seyed Mohammad Mirjalili ve Andrew Lewis tarafından 2014 yılında önerilen gri kurtların sosyal hiyerarşi ve avlanma davranışından esinlenerek oluşturulmuş popülasyon tabanlı bir algoritmadır.

3.4.1 Sosyal Hiyerarşi

Gri kurtlar sürü halinde yaşarlar ve Şekil 3.10.'da görüldüğü gibi yukarıdan aşağıya doğru azalan güçlü bir sosyal hiyerarşiye sahiptirler.



Şekil 3.10. Gri Kurt Hiyerarşisi

Bu hiyerarşide liderler alfa olarak isimlendirilir. Alfa; avlanma, uyuma yeri, kalkma zamanı gibi kararlar almakla ve bu kararları sürüye iletmekle sorumludur. Sürü, alfanın emirlerini uyguladığı için alfa aynı zamanda baskın kurt olarak isimlendirilir [33]. Alfa sürüyü yönetmek açısından en iyi birey olmasına rağmen sürünün en güçlü bireyi olmasına gerek yoktur. Bu da sürünün organizasyon yapısının ve disiplininin gücünden daha önemli olduğu anlamına gelmektedir.

Hiyerarşinin ikinci seviyesi beta olarak isimlendirilir. Beta, karar verme ya da diğer sürü ile ilgili faaliyetlerde alfaya yardım eder. Beta, alfanın ölmesi veya çok yaşlanması gibi durumlarda alfanın yerine geçecek en muhtemel adaydır. Beta, alfadan aldığı emirleri uygularken hiyerarşi piramidindeki düşük seviyedeki kurtlara ise komutlar vermesinin yanı sıra alfanın komutlarının sürüye aktarılmasına yardımcı olur ve alfaya çeşitli geri beslemeler verir.

Hiyerarşinin üçüncü seviyesinde yer alan kurt ise deltadır. Delta, alfa ve betadan emir alırken hiyerarşi piramidinde kendinden aşağıda bulunanlara ise emir verir. İzciler, gözcüler, yaşlılar, avcılar ve bekçiler deltayı oluşturur. İzciler, bölgenin sınırlarını gözetlemek ve herhangi bir tehlike durumunda sürüyü uyarmakla sorumludur. Gözcüler, sürünün güvenliğini sağlar. Yaşlılar, daha önceden alfa ya da betaya ait olan tecrübeli kurtlardır. Avcılar, avlanma sırasında alfa ve betaya yardım eder ve sürüye yiyecek sağlar. Bekçiler ise sürüdeki hasta, yaralı ve zayıf kurtlara bakmakla sorumludur.

Hiyerarşi piramidinde en alt seviyede omega yer alır ve omega, üst seviyede bulunan kurtlardan gelen emirleri alır. Omega en alt seviyede bulunmasından dolayı önemsiz gibi düşünülemez çünkü omeganın olmaması durumunda sürü iç çatışmalar ve problemler ile karşı karşıya kalır.

Gri Kurt Optimizasyonu, kurtların sosyal hiyerarşisi dışında gri kurtların avlama stretejilerini de simüle eder.

Gri Kurt Optimizasyonunda sosyal hiyerarşinin matematiksel olarak modellenmesi amacıyla en iyi birey alfa (α) olarak, ikinci en iyi birey beta (β) olarak ve üçüncü en iyi birey delta (δ) olarak alınır. Kalan diğer çözümler ise omega (ω) olarak farzedilir.

3.4.2. Avın Kuşatılması

Gri Kurt Optimizasyonun oluşturulmasında sosyal hiyerarşinin dışında gri kurtlara özgü avlanma davranışı da bir diğer önemli sosyal davranıştır. Muro et al. [34] gri kurtların avlanma davranışını; takip etme, avın çevrelenmesi ve ava doğru saldırı olarak açıklamıştır.

Avlama sırasında gri kurtların avını kuşatması matematiksel olarak aşağıdaki gibi modellenmiştir.

$$\overrightarrow{D} = \left| \overrightarrow{C}.\overrightarrow{Xp}(t) - \overrightarrow{X}(t) \right| \tag{3.13}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{Xp}(t) - \overrightarrow{A}.\overrightarrow{D} \tag{3.14}$$

Denklem (3.13)'te ve Denklem (3.14)'de yer alan t iterasyon sayısını, $\vec{A}$ ve $\vec{C}$ katsayı vektörünü, Xp avın konum vektörünü ve $\vec{X}$ gri kurdun konum vektörünü ifade eder.

$\vec{A}$ ve $\vec{C}$ ise aşağıdaki gibi hesaplanır.

$$\overrightarrow{A} = 2\overrightarrow{a}.\overrightarrow{r_1} - \overrightarrow{a} \tag{3.15}$$

$$\overrightarrow{C} = 2\overrightarrow{r_2} \tag{3.16}$$

Denklem (3.15)'te ve Denklem (3.16)'da yer alan $\overrightarrow{r1}$ ve $\overrightarrow{r2}$ [0 1] aralığında rastgele vektörlerdir. Denklem (3.15)'te yer alan alan $\vec{a}$ ise iterasyon boyunca 2'den 0'a kadar lineer olarak azalmaktadır.

3.4.3. Avlama

Gri kurtlar avını bulabilme ve onun etrafı kuşatabilme yeteneğine sahiptir. Bu avlamada bazen beta ve delta da yer alsa da genellikle alfanın yol göstermesi neticesinde gerçekleşir. Bunu matematiksel olarak modelleyebilmek için alfanın, betanın ve deltanın avın konumunu bulmak açısından diğerlerinden daha iyi çözümler olduğu varsayılmıştır. Bu sebeple omega dahil olmak üzere arama ajanlarının yeni konumları en iyi üç çözüme göre güncellenir.

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \overrightarrow{X}\right|,\ \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}.\overrightarrow{X_\beta} - \overrightarrow{X}\right|,\ \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}.\overrightarrow{X_\delta} - \overrightarrow{X}\right| \tag{3.17}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\left(\overrightarrow{D_\alpha}\right),\ \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\left(\overrightarrow{D_\beta}\right),\ \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\left(\overrightarrow{D_\delta}\right) \tag{3.18}$$

$$\overrightarrow{X}\left(t+1\right) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{3.19}$$

3.4.4. Ava Saldırma

Gri kurtlar avlamalarını ava saldırarak sona erdirirler. Bu yaklaşımın matematiksel olarak modellenebilmesi için ise $\vec{a}$ değeri azaltılır. $\vec{a}$ değerinin iterasyon boyunca 2'den 0'a kadar azalması ile aynı zamanda Denklem (3.15) ile ifade edilen $\vec{A}$'nın da alabileceği değer aralığındaki dalgalanma $\vec{a}$ tarafından azaltılır.

$\vec{A}$'nın alacağı rastgele değer [-1,1] aralığında olduğunda arama ajanının bir sonraki konumu mevcut konumu ile avının konumu arasında bir yerde olur. Başka bir ifade ile |A| <1 olduğunda ava saldırma işlemi gerçekleşir.

3.4.5. Av Arama

Gri kurtların birbirlerinden uzaklaşarak av arama davranışını matematiksel olarak modellemek için $\vec{A}$'nın alabileceği 1'den daha büyük ve -1'den daha küçük değerlerden faydalanılır.

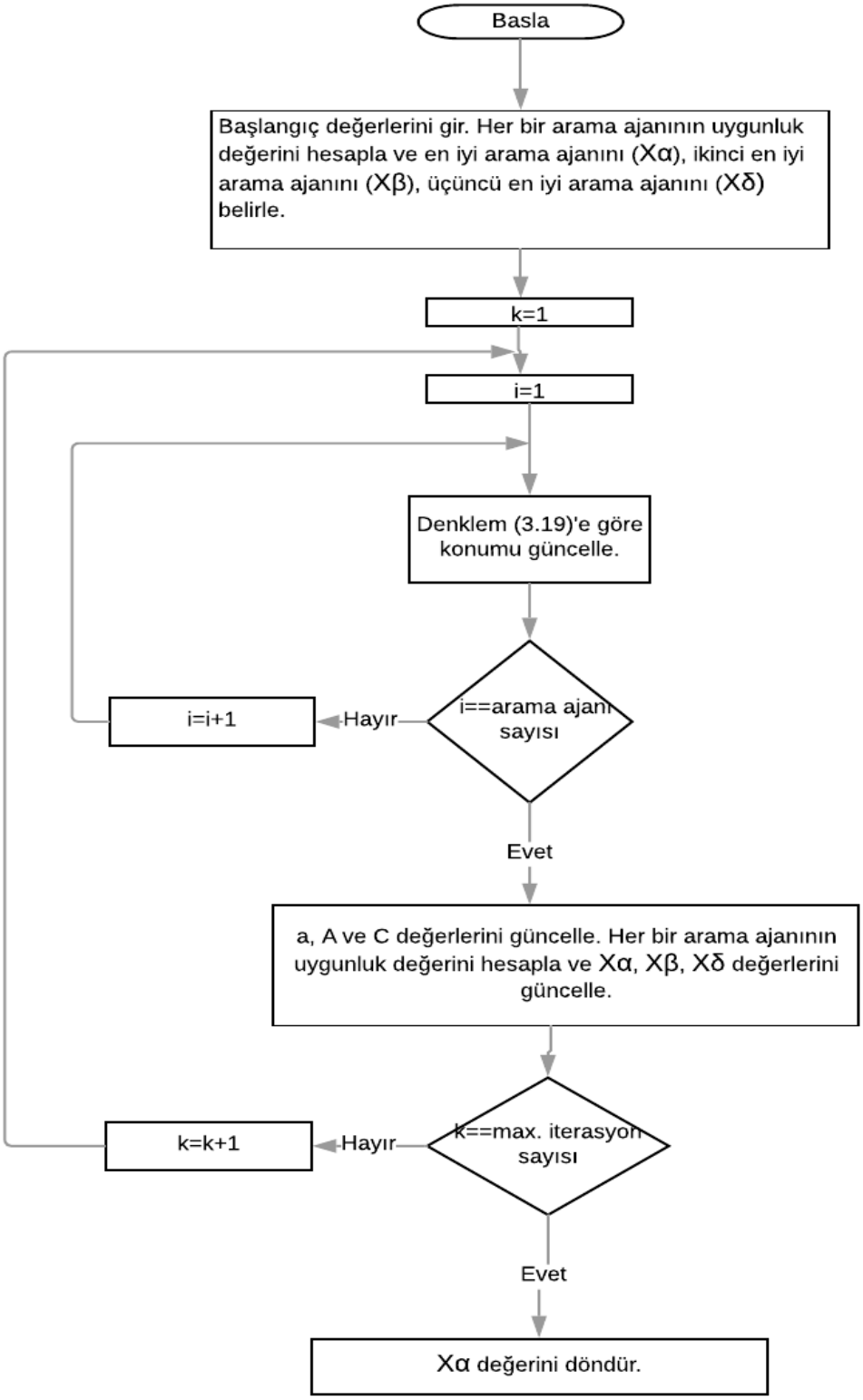
|A| >1 olması gri kurdun avından uzaklaşarak daha iyi bir av aramasını sağlar ve bu sayede Gri Kurt Optimizasyon algoritmasının global araştırma yeteneği iyileşir.

3.4.6 Gri Kurt Optimizasyonunun Adımları ve Akış Diyagramı

Gri Kurt Optimizasyonunun sahte kodu:

a, A, C, maksimum iterasyon sayısı, popülasyon büyüklüğü değerlerinin ilk atamasını yap ve popülasyonu oluştur.
for: her bir arama ajanı için
    Uygunluk değerini hesapla
end
Xα= en iyi arama ajanı
Xβ= ikinci en iyi arama ajanı
Xδ= üçüncü en iyi arama ajanı
t=0
while (t< maksimum iterasyon sayısı)
  for: her bir arama ajanı için
      Denklem(3.19)'a göre arama ajanının konumunu güncelle
  end for
 Güncelle a, A, C
Her bir arama ajanının uygunluk değerini hesapla; Xα, Xβ, Xδ değerini güncelle
t=t+1
end while
Xα değerini döndür.

Gri Kurt Optimizasyonu akış diyagramı:



Şekil 3.11. Gri Kurt Optimizasyonu Algoritması Akış Diyagramı

4. BÖLÜM

HİBRİT BALİNA OPTİMİZASYON ALGORİTMASI (HBOA)

4.1. Hibrit Balina Optimizasyon Algoritmasının Oluşturulması

Bu bölümde Balina Optimizasyon Algoritmasına Gri Kurt Optimizasyonunda bulunan birtakım özellikler entegre edilip hibrit bir yaklaşım oluşturulmuştur. Bu işlem gerçekleştirilirken Balina Optimizasyon Algoritmasının sadeliğinin korunması hedeflenmiştir.

Bölüm 3'te belirtildiği gibi Balina Optimizasyon Algoritması'nda Denklem (3.6) ve Denklem (3.9) kapsamında konum güncelleme işlemi yaparken sadece bir tane en iyi arama ajanının bilgisinden istifade edilmekteyken Gri Kurt Optimizasyonunda ise Denklem (3.19) kapsamında konum güncelleme işlemi için en iyi üç arama ajanının (alfa, beta ve delta) bilgisinden istifade edilmektedir. Bu kapsamda Balina Optimizasyon Algoritmasının tüm özelliklerine ek olarak gerçekleştirilmiş iterasyonlar boyunca elde edilmiş en iyi arama ajanı yerine en iyi üç arama ajanının bilgisinden istifade edilen hibrit bir yaklaşım oluşturulmuştur.

Oluşturulan hibrit balina optimizasyon algoritmasında konum güncelleme işlemleri, Balina Optimizasyon Algoritmasında en iyi bireyi kullanan konum güncelleme formüllerinin en iyi üç bireyi kullanan konum güncelleme formülleri haline getirilmesi ile gerçekleştirilmiştir. Oluşturulan yeni denklemler aşağıda yer almaktadır.

$$\overrightarrow{D_\alpha} = \left| \vec{C}.\overrightarrow{X_\alpha}(t) - \overrightarrow{X}(t) \right| \tag{4.1}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha}(t) - \vec{A}.\overrightarrow{D_\alpha} \tag{4.2}$$

$$\overrightarrow{D_\beta} = \left| \vec{C}.\overrightarrow{X_\beta}(t) - \vec{X}(t) \right| \tag{4.3}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta}(t) - \vec{A}.\overrightarrow{D_\beta} \tag{4.4}$$

$$\overrightarrow{D_\delta} = \left| \vec{C}.\overrightarrow{X_\delta}(t) - \vec{X}(t) \right| \tag{4.5}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta}(t) - \vec{A}.\overrightarrow{D_\delta} \tag{4.6}$$

$$X(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{4.7}$$

$$\overrightarrow{X_1} = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X_\alpha} \tag{4.8}$$

$$\overrightarrow{X_2} = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X_\beta} \tag{4.9}$$

$$\overrightarrow{X_3} = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X_\delta} \tag{4.10}$$

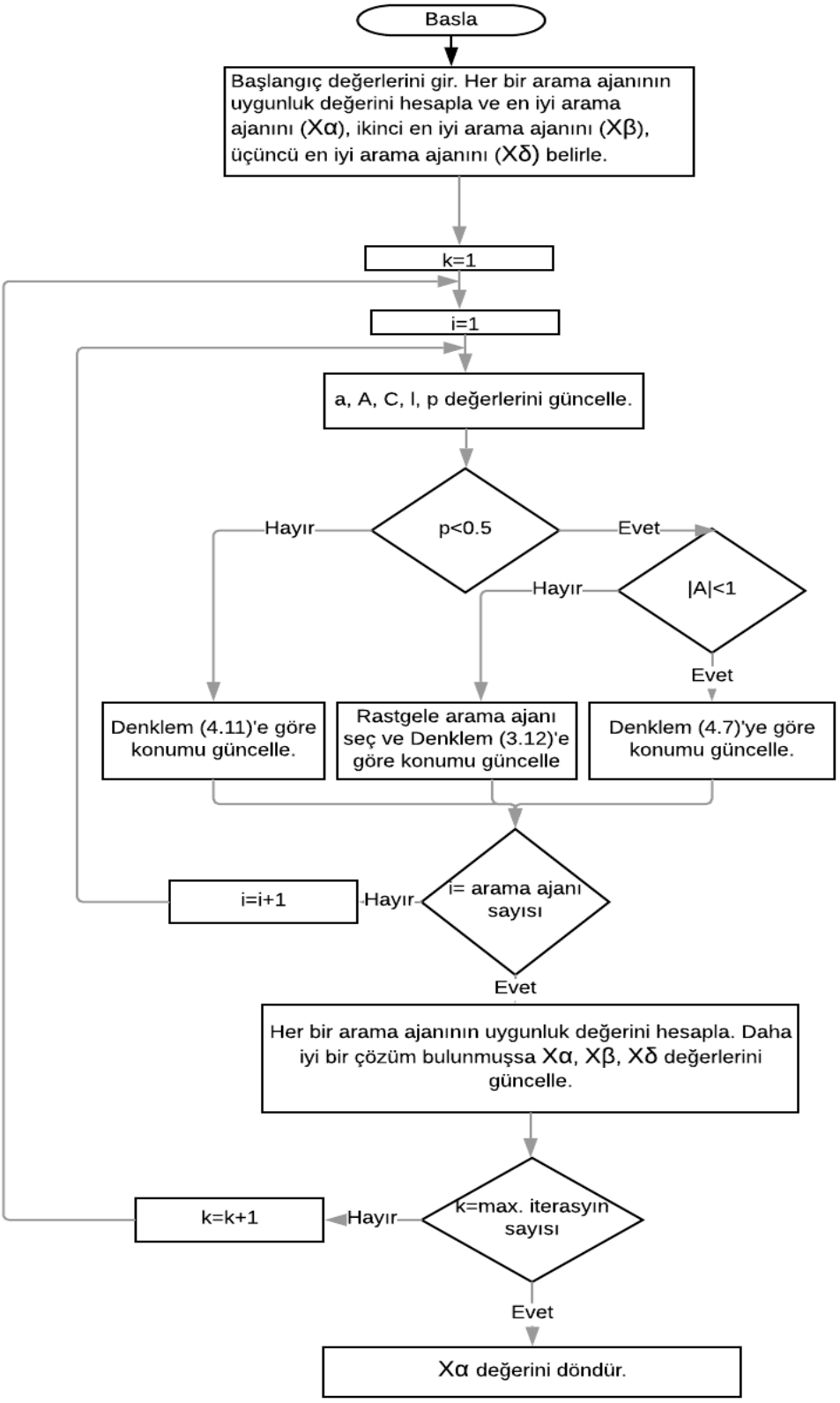
$$X(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{4.11}$$

4.2. Hibrit Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı

Hibrit Balina Optimizasyon Algoritması sahte kodu:

a, A, C, l, p, maksimum iterasyon sayısı, popülasyon büyüklüğü değerlerinin ilk atamasını yap ve popülasyonu oluştur.
for: her bir arama ajanı için
    Uygunluk değerini hesapla
end
Xα= en iyi arama ajanı
Xβ= ikinci en iyi arama ajanı
Xδ= üçüncü en iyi arama ajanı
t=0
while (t< maksimum iterasyon sayısı)
  for: her bir arama ajanı için
      Güncelle a, A, C, l, p
        if (p<0.5)
            if (|A| <1)
                Denklem(4.7)'ye göre arama ajanının konumunu güncelle
            else if (|A| >=1)
                Rastgele bir arama ajanı seç (Xrand)
                Denklem(3.12)'ye göre arama ajanının konumunu güncelle.
            end if
        else if (p>=0.5)
            Denklem (4.11)'e göre arama ajanının konumunu güncelle.
        end if
  end for
Arama ajanının kısıt dışına çıkıp çıkmadığına bak çıkıyorsa sınır değerini ver
Her bir arama ajanının uygunluk değerini hesapla
Daha iyi bir çözüm varsa Xα, Xβ, Xδ değerini güncelle
t=t+1
end while
Xα değerini döndür.

Hibrit Balina Optimizasyon Algoritması akış diyagramı:



Şekil 4.1. Hibrit Balina Optimizasyon Algoritması Akış Diyagramı

4.3 Hibrit Balina Optimizasyon Algoritmasının Test Edilmesi

Oluşturulan hibrit balina optimizasyon algoritmasının etkinliği 23 adet benchmark fonksiyonu ile ölçülmüştür. Kullanılan benchmark fonksiyonları; problemlerin tek modlu (unimodal), çok modlu (multimodal) ve yüksek boyutlu (high dimensional), çok modlu ve düşük boyutlu (low dimensional) olmasına göre sınıflandırılmış olup Tablo 4.1'de tek modlu benchmark problemleri, Tablo 4.2'de çok modlu ve yüksek boyutlu benchmark problemleri, Tablo 4.3'te çok modlu ve düşük boyutlu benchmark problemleri yer almaktadır [35]. Tablo 4.1'de, Tablo 4.2'de ve Tablo 4.3'te bu çalışma kapsamında kullanılan benchmark fonksiyonlarının matematiksel ifadesi, fonksiyonun literatürde yer alan ismi, fonksiyonun boyutu yani problemin karar değişkeni sayısı, fonksiyonun karar değişkeninin hangi aralıkta değer alabileceğini gösteren araştırma uzayı, fonksiyonun belirlenen araştırma uzayı dahilinde alabileceği minimum değer ve fonksiyonun karakteristiği yer almaktadır.

Tablo 4.1 Tek Modlu Benchmark Fonksiyonları

Matematiksel İfadesi	İsmi	Boyutu	Araştırma Uzayı	Minimum Değeri	K
$f_1(x)=\sum_{i=1}^{n} x_i^2$	Sphere	30	[-100, 100]	0	T, A
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	Schwefel2.22	30	[-10, 10]	0	T, AO
$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{i} x_j\right)^2$	Schwefel1.2	30	[-100, 100]	0	T, AO
$f_4(x)=\max_i\left\{\lvert x_i\rvert, 1\le i\le n\right\}$	Schwefel2.21	30	[-100, 100]	0	T, A
$f_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^{\,2}\right)^2+\left(x_i-1\right)^2\right]$	Rosenbrock	30	[-30, 30]	0	T, AO
$f_6(x)=\sum_{i=1}^{n}\left(\left[x_i+0.5\right]\right)^2$	Step	30	[-100, 100]	0	T, A
$f_7(x)=\sum_{i=1}^{n} i x_i^4+random[0,1)$	Quartic	30	[-1.28, 1.28]	0	T, A

T: Tek modlu, Ç: Çok modlu, A: Ayrılabilir, AO: Ayrılabilir Olmayan, K:Karakteristiği

Tablo 4.2. Çok Modlu ve yüksek boyutlu Benchmark Fonksiyonları

Matematiksel İfadesi	İsmi	Boyutu	Araştırma Uzayı	Minimum Değeri	K
$f_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	Schwefel	30	[-500, 500]	-418.9829x5	Ç, A
$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	Rastrigin	30	[-5.12, 5.12]	0	Ç, A
$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	Ackley	30	[-32, 32]	0	Ç, AO
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	Griewank	30	[-600, 600]	0	Ç, AO
$f_{12}(x)=\frac{\pi}{\Pi}\left\{(10\sin(\pi y_1))+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	Penalized	30	[-50, 50]	0	Ç, AO
$f_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i5,100,4)$	Penalized2	30	[-50, 50]	0	Ç, AO

T: Tek modlu, Ç: Çok modlu, A: Ayrılabilir, AO: Ayrılabilir Olmayan, K:Karakteristiği

Tablo 4.3. Çok Modlu ve düşük boyutlu Benchmark Fonksiyonları

Matematiksel İfadesi	İsmi	Boyutu	Araştırma Uzayı	Minimum Değeri	K
$f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	Foxholes	2	[-65, 65]	1	Ç, A
$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_i x_2\right)}{b_i^2+b_i x_3+x_4}\right]^2$	Kowalik	4	[-5, 5]	0.00030	Ç, AO
$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	Six Hump Camel Back	2	[-5, 5]	-1.0316	Ç, AO
$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	Branin	2	[-5,5]	0.398	Ç, A
$f_{18}(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]*\left[30+\left(2x_1-3x_2\right)^2*\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	GoldStein-Price	2	[-2,2]	3	Ç,A
$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	Hartman3	3	[1,3]	-3.86	Ç, AO
$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	Hartman	6	[0,1]	-3.32	Ç, AO
$f_{21}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	Shekel5	4	[0,10]	-10.1532	Ç, AO
$f_{22}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	Shekel7	4	[0,10]	-10.4028	Ç, AO
$f_{23}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	Shekel10	4	[0,10]	-10.5363	Ç, AO

T: Tek modlu, Ç: Çok modlu, A: Ayrılabilir, AO: Ayrılabilir Olmayan, K:Karakteristiği

Hibrit Balina Optimizasyon Algoritmasının etkinliği Tablo 4.1'de, Tablo 4.2'de, Tablo 4.3'te yer alan 23 adet benchmark fonksiyonu ile test edilmiştir ve elde edilen sonuçlar Balina Optimizasyon Algoritması, Gri Kurt Optimizasyonu, Parçacık Sürü Optimizasyonu, Diferansiyel Gelişim algoritmalarından elde edilen sonuçlar ile karşılaştırılmıştır. Karşılaştırmalı sonuçlar Tablo 4.4'te yer almaktadır.

Tablo 4.4'te yer alan algoritmaların tamamında popülasyon büyüklüğü 30, maksimum iterasyon sayısı ise 500 olarak belirlenmiş [8] olup algoritmaların tamamı Tablo 4.1'de, Tablo 4.2'de, Tablo 4.3'te yer alan benchmark fonksiyonlarının her biri için 30 kez çalıştırılmıştır.

Tablo 4.4' te yer alan ve Hibrit Balina Optimizasyon Algoritmasının kıyaslanmasında kullanılan BOA, GKO, PSO ve DG algoritmalarına ait sonuçlar [8,10] 'dan alınmıştır.

Tablo 4.4'te görüldüğü gibi Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonunun birlikte kullanıldığı Hibrit Balina Optimizasyon Algoritması diğer algoritmalara kıyasla 23 benchmark fonksiyonundan 3 benchmark fonksiyonunda en iyi sonucu üretmiştir. Diğer taraftan, tez çalışması kapsamında karşılaştırılan diğer algoritmalara kıyasla Balina Optimizasyon Algoritması 4 benchmark fonksiyonunda, Gri Kurt Optimizasyonu 2 benchmark fonksiyonunda, Parçacık Sürü Optimizasyonu 4 benchmark fonksiyonunda ve Diferansiyel Gelişim Algoritması 16 benchmark fonksiyonunda en iyi sonuçları üretmiştir.

Tablo 4.4. HBOA, BOA, GKO, PSO VE DG algoritmalarının Benchmark fonksiyonları ile elde edilen sonuçları

F	HBOA		BOA [8]		GKO [10]		PSO [8,10]		DG [8,10]	
	ortalama	std	ortalama	std	ortalama	std	ortalama	std	ortalama	std
F1	**7.4825E-51**	2.6539E-50	1 .41E −30	4 .91E −30	6.59E-28	6.34E-05	0.000136	0.000202	8 .2E −14	5 .9E −14
F2	**1.1463E-38**	5.0604E-38	1 .06E −21	2 .39E −21	7.18E-17	0.029014	0.042144	0 .045421	1 .5E −09	9 .9E −10
F3	2.4012E+04	8.0961E+03	5 .39E −07	2 .93E −06	3.29E-06	79.14958	70 .12562	22 .11924	**6 .8E −11**	7 .4E −11
F4	21.6031	11.9353	0 .072581	0 .39747	5.61E-07	1.315088	1 .086481	0 .317039	**0**	0
F5	28.2644	0.4173	27 .86558	0 .763626	26.81258	69.90499	96 .71832	60 .11559	**0**	0
F6	0.9028	0.3195	3 .116266	0 .532429	0.816579	0.000126	0 .000102	8 .28E −05	**0**	0
F7	0.0046	0.0038	**0 .001425**	0 .001149	0.002213	0.100286	0 .122854	0 .044957	0 .00463	0 .0012
F8	-1.0970E+04	1.6206E+03	−5080.76	695.7968	-6123.1	-4087.44	−4841.29	1152.814	**−11080.1**	574.7
F9	2.4632E-14	3.8592E-14	**0**	0	0.310521	47.35612	46 .70423	11 .62938	69 .2	38 .8
F10	**8.5857E-15**	3.8562E-15	7 .4043	9 .897572	1.06E-13	0.077835	0 .276015	0 .50901	9 .7E −08	4 .2E −08
F11	0.0091	0.0499	0 .000289	0 .001586	0.004485	0.006659	0 .009215	0 .007724	**0**	0
F12	0.0789	0.1288	0 .339676	0 .214864	0.053438	0.020734	0 .006917	0 .026301	**7 .9E −15**	8E −15
F13	0.9161	0.2644	1 .889015	0 .266088	0.654464	0.004474	0 .006675	0 .008907	**5 .1E −14**	4 .8E −14
F14	2.2066	2.9542	2 .111973	2 .498594	4.042493	4.252799	3 .627168	2 .560828	**0 .998004**	3 .3E −16
F15	9.5471E-04	7.9180E-04	0 .000572	0 .000324	0.000337	0.000625	0 .000577	0 .000222	**4 .5E −14**	0 .00033
F16	-1.0316	6.7752E-16	**−1 .03163**	4 .2E −07	**-1.03163**	-1.03163	**−1 .03163**	6 .25E −16	**−1 .03163**	3 .1E −13
F17	0.3979	5.2281E-05	0 .397914	2 .7E −05	0.397889	0.397887	**0 .397887**	0	**0 .397887**	9 .9E −09
F18	3.0005	0.0027	**3**	4 .22E −15	3.000028	3	**3**	1 .33E −15	**3**	2E −15
F19	-3.8145	0.0673	−3 .85616	0 .002706	-3.86263	-3.86278	**−3 .86278**	2 .58E −15	N/A	N/A
F20	-3.1886	0.1067	−2 .98105	0 .376653	**-3.28654**	-3.25056	−3 .26634	0 .060516	N/A	N/A
F21	-9.1935	1.9703	−7 .04918	3 .629551	-10.1514	-9.14015	−6 .8651	3 .019644	**−10 .1532**	0 .0000025
F22	-9.0700	2.4947	−8 .18178	3 .829202	-10.4015	-8.58441	−8 .45653	3 .087094	**−10 .4029**	3 .9E −07
F23	-9.4722	2.0255	−9 .34238	2 .414737	-10.5343	-8.55899	−9 .95291	1 .782786	**−10 .5364**	1 .9E −07

5. BÖLÜM

İYİLEŞTİRİLMİŞ HİBRİT BALİNA OPTİMİZASYON ALGORİTMASI (İHBOA)

5.1. İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasının Oluşturulması

Bu bölümde Bölüm 4'te yer alan Hibrit Balina Optimizasyon Algoritmasının iyileştirilmesi için bir çözüm öne sürülmüştür.

Bölüm 4 kapsamında oluşturulan hibrit yaklaşımda algoritmanın arama uzayının tamamını düzgün bir şekilde araştırabilme yeteneğinin iyileştirilmesi maksadıyla suda yaşayan balık türü canlıların kuyruklarını suya vurma şiddetinin balıklarının konumunu değiştirmesinde etkili olmasından yola çıkılarak Hibrit Balina Optimizasyon Algoritmasında konum güncelleme işlemi için kullanılan denklemlere [0 1] aralığında rastgele değer alan ve balığın hareketi sırasında kuyruğunun suya vuruş şiddetini temsil eden q katsayısı çarpan olarak eklenmiş ve İyileştirilmiş Hibrit Balina Optimizasyon Algoritması oluşturulmuştur [36].

Önerilen İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasında konum güncelleme işlemleri aşağıdaki denklemler ile gerçekleştirilmektedir.

$$\vec{D}\alpha = |\vec{C} \,.\, \overrightarrow{X\alpha(t)} - \overrightarrow{X(t)}| \tag{5.1}$$

$$\overrightarrow{X_1} = [\overrightarrow{X_\alpha}(t) - \vec{A}.\overrightarrow{D_\alpha}].q \tag{5.2}$$

$$\vec{D}\beta = |\vec{C} \,.\, \overrightarrow{X\beta(t)} - \overrightarrow{X(t)}| \tag{5.3}$$

$$\overrightarrow{X_2} = [\overrightarrow{X_\beta}(t) - \vec{A}.\overrightarrow{D_\beta}].q \tag{5.4}$$

$$\vec{D}\delta = |\vec{C} \,.\, \overrightarrow{X\delta(t)} - \overrightarrow{X(t)}\,| \tag{5.5}$$

$$\overrightarrow{X_3} = [\overrightarrow{X_\delta}(t) - \vec{A}.\overrightarrow{D_\delta}].q \tag{5.6}$$

$$\overrightarrow{X(t+1)} = (\,\overrightarrow{X1} + \overrightarrow{X2} + \overrightarrow{X3}\,)\,/3 \tag{5.7}$$

$$\overrightarrow{X_1} = [\overrightarrow{D'}e^{bl}.\cos(2\pi l) + \overrightarrow{X_\alpha}].q \tag{5.8}$$

$$\overrightarrow{X_2} = [\overrightarrow{D'}e^{bl}.\cos(2\pi l) + \overrightarrow{X_\beta}].q \tag{5.9}$$

$$\overrightarrow{X_3} = [\overrightarrow{D'}e^{bl}.\cos(2\pi l) + \overrightarrow{X_\delta}].q \tag{5.10}$$

$$\overrightarrow{X(t+1)} = (\,\overrightarrow{X1} + \overrightarrow{X2} + \overrightarrow{X3}\,)\,/3 \tag{5.11}$$

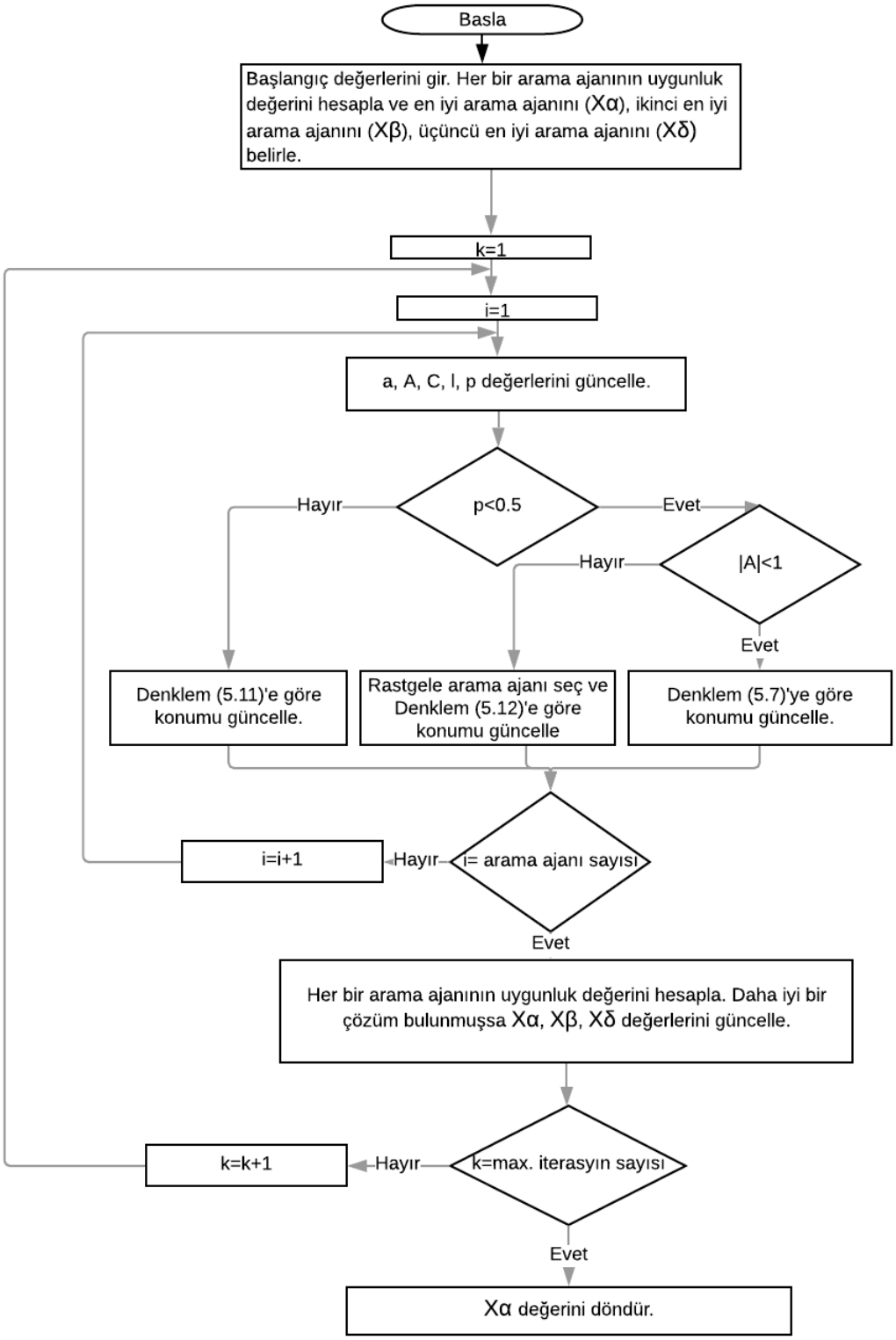
$$\overrightarrow{X}(t+1) = [\overrightarrow{Xrand} - \vec{A}.\vec{D}].q \tag{5.12}$$

5.2. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması Adımları ve Akış Diyagramı

İyileştirilmiş Hibrit Balina Optimizasyon Algoritması sahte kodu:

a, A, C, l, p, maksimum iterasyon sayısı, popülasyon büyüklüğü değerlerinin ilk atamasını yap ve popülasyonu oluştur.
for: her bir arama ajanı için
    Uygunluk değerini hesapla
end
Xα= en iyi arama ajanı
Xβ= ikinci en iyi arama ajanı
Xδ= üçüncü en iyi arama ajanı
t=0
while (t< maksimum iterasyon sayısı)
  for: her bir arama ajanı için
      Güncelle a, A, C, l, p
       if (p<0.5)
            if (|A| <1)
               Denklem(5.7)'ye göre arama ajanının konumunu güncelle
            else if (|A| >=1)
               Rastgele bir arama ajanı seç (Xrand)
               Denklem(5.12)'ye göre arama ajanının konumunu güncelle.
            end if
       else if (p>=0.5)
            Denklem (5.11)'e göre arama ajanının konumunu güncelle.
       end if
  end for
Arama ajanının kısıt dışına çıkıp çıkmadığına bak çıkıyorsa sınır değerini ver
Her bir arama ajanının uygunluk değerini hesapla
Daha iyi bir çözüm varsa Xα, Xβ, Xδ değerini güncelle
t=t+1
end while
Xα değerini döndür.

İyileştirilmiş Hibrit Balina Optimizasyon Algoritması akış diyagramı:



Şekil 5.1. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması Akış Diyagramı

5.3. İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasının Test Edilmesi

İyileştirilmiş Hibrit Balina Optimizasyon Algoritması'nın (İHBOA) etkinliği Tablo 4.1'de, Tablo 4.2'de ve Tablo4.3'te yer alan 23 adet benchmark fonksiyonu ile test edilmiştir. Elde edilen sonuçlar Hibrit Balina Optimizasyon Algoritması, Balina Optimizasyon Algoritması, Gri Kurt Optimizasyonu, Parçacık Sürü Optimizasyonu, Diferansiyel Gelişim algoritmalarından elde edilen sonuçlar ile karşılaştırılmış olup karşılaştırmalı sonuçlar Tablo 5.1'de yer almaktadır.

Tablo 5.1'de yer alan algoritmaların tamamında popülasyon büyüklüğü 30, maksimum iterasyon sayısı ise 500 olarak belirlenmiş [8] olup algoritmaların tamamı Tablo 4.1'de, Tablo 4.2'de, Tablo 4.3'te yer alan benchmark fonksiyonlarının her biri için 30 kez çalıştırılmıştır.

Tablo 5.1'de yer alan ve İyileştirilmiş Hibrit Balina Optimizasyon Algoritmasının kıyaslanmasında kullanılan BOA, GKO, PSO, DG algoritmalarına ait sonuçlar [8,10]'dan alınmıştır.

Tablo 5.1'de görüldüğü gibi Balina Optimizasyon Algoritması ve Gri Kurt Optimizasyonunun birlikte kullanıldığı Hibrit Balina Optimizasyon Algoritmasının iyileştirilmesi maksadıyla önerilen İyileştirilmiş Hibrit Balina Optimizasyon Algoritması diğer algoritmalara kıyasla 23 benchmark fonksiyonundan 9 benchmark fonksiyonunda en iyi sonucu üretmiştir. Diğer taraftan, tez çalışması kapsamında karşılaştırılan diğer algoritmalara kıyasla Hibrit Balina Optimizasyon Algoritmasının en iyi ürettiği sonuç yokken Balina Optimizasyon Algoritması 3 benchmark fonksiyonunda, Gri Kurt Optimizasyonu 2 benchmark fonksiyonunda, Parçacık Sürü Optimizasyonu 4 benchmark fonksiyonunda ve Diferansiyel Gelişim Algoritması 14 benchmark fonksiyonunda en iyi sonuçları üretmiştir.

Tablo 5.1. İHBOA, HBOA, BOA, GKO, PSO VE DG algoritmalarının Benchmark fonksiyonları ile elde edilen sonuçları

F	İHBOA		HBOA		BOA [8]		GKO [10]		PSO [8,10]		DG [8,10]	
	ortalama	std	ortalama	std	ortalama	std	ortalama	std	ortalama	std	ortalama	std
F1	**0**	0	7.4825E-51	2.6539E-50	1 .41E −30	4 .91E −30	6.59E-28	6.34E-05	0.000136	0.000202	8 .2E −14	5 .9E −14
F2	**0**	0	1.1463E-38	5.0604E-38	1 .06E −21	2 .39E −21	7.18E-17	0.029014	0.042144	0 .045421	1 .5E −09	9 .9E −10
F3	**0**	0	2.4012E+04	8.0961E+03	5 .39E −07	2 .93E −06	3.29E-06	79.14958	70 .12562	22 .11924	6 .8E −11	7 .4E −11
F4	**0**	0	21.6031	11.9353	0 .072581	0 .39747	5.61E-07	1.315088	1.086481	0.317039	**0**	0
F5	28.2906	0.3417	28.2644	0.4173	27 .86558	0 .763626	26.81258	69.90499	96 .71832	60.11559	**0**	0
F6	0.4649	0.1649	0.9028	0.3195	3 .116266	0 .532429	0.816579	0.000126	0 .000102	8.28E −05	**0**	0
F7	**6.5646E-05**	8.3630E-05	0.0046	0.0038	0 .001425	0 .001149	0.002213	0.100286	0 .122854	0.044957	0 .00463	0 .0012
F8	**-1.1547E+04**	1.3019E+03	-1.0970E+04	1.6206E+03	−5080.76	695.7968	-6123.1	-4087.44	−4841.29	1152.814	−11080.1	574.7
F9	**0**	0	2.4632E-14	3.8592E-14	**0**	0	0.310521	47.35612	46 .70423	11 .62938	69 .2	38 .8
F10	**8.8818E-16**	4.0117E-31	8.5857E-15	3.8562E-15	7 .4043	9 .897572	1.06E-13	0.077835	0 .276015	0.50901	9 .7E −08	4 .2E −08
F11	**0**	0	0.0091	0.0499	0.000289	0.001586	0.004485	0.006659	0.009215	0.007724	**0**	0
F12	0.0282	0.0165	0.0789	0.1288	0 .339676	0 .214864	0.053438	0.020734	0.006917	0.026301	**7 .9E −15**	8E −15
F13	0.3136	0.1215	0.9161	0.2644	1 .889015	0 .266088	0.654464	0.004474	0.006675	0.008907	**5 .1E −14**	4 .8E −14
F14	8.5971	5.0969	2.2066	2.9542	2 .111973	2 .498594	4.042493	4.252799	3.627168	2.560828	**0 .998004**	3 .3E −16
F15	4.0680E-04	8.3682E-05	9.5471E-04	7.9180E-04	0 .000572	0 .000324	0.000337	0.000625	0.000577	0 .000222	**4 .5E −14**	0 .00033
F16	-1.0106	0.0102	-1.0316	6.7752E-16	**−1 .03163**	4 .2E −07	**-1.03163**	-1.03163	**−1.03163**	6 .25E −16	**−1 .03163**	3 .1E −13
F17	0.3987	0.0020	0.3979	5.2281E-05	0 .397914	2 .7E −05	0.397889	0.397887	**0.397887**	0	**0 .397887**	9 .9E −09
F18	3.0133	0.0195	3.0005	0.0027	**3**	4 .22E −15	3.000028	3	**3**	1 .33E −15	**3**	2E −15
F19	-3.8208	0.0462	-3.8145	0.0673	−3 .85616	0 .002706	-3.86263	-3.86278	**−3.86278**	2 .58E −15	N/A	N/A
F20	-3.1262	0.1080	-3.1886	0.1067	−2 .98105	0 .376653	**-3.28654**	-3.25056	−3.26634	0 .060516	N/A	N/A
F21	-5.3993	1.3176	-9.1935	1.9703	−7 .04918	3 .629551	-10.1514	-9.14015	−6.8651	3 .019644	**−10 .1532**	0 .0000025
F22	-5.1276	0.4280	-9.0700	2.4947	−8 .18178	3 .829202	-10.4015	-8.58441	−8.45653	3 .087094	**−10 .4029**	3 .9E −07
F23	-5.3044	0.9008	-9.4722	2.0255	−9 .34238	2 .414737	-10.5343	-8.55899	−9.95291	1 .782786	**−10 .5364**	1 .9E −07

6. BÖLÜM

SONUÇ

Optimizasyon son zamanlarda oldukça popüler olan zor bir çalışma alanıdır. Çok çeşitli mühendislik problemlerinin çözümü için bir çok gradient tabanlı optimizasyon yöntemleri geliştirilmiştir. Gradient tabanlı optimizasyon yöntemleri başlagıç çözümü oluşturmak için gradient bilgisine ihtiyaç duymaktadır. Bu yöntemler karmaşık gerçek hayat problemlerinin ve birden fazla yerel optimuma sahip problemlerin çözümü için yetersiz kalmaktadır [37]. Türev bilgisine dayanan gradient tabanlı optimizasyon yöntemlerinin bu gibi çeşitli dezavantajlarından dolayı araştırmacılar daha kullanışlı metotları geliştirmek için çalışmalar yapmıştır ve bu çalışmalar sonucunda ortaya çıkan metasezgisel algoritmalar mühendislik problemlerinin, gerçek hayat problemlerinin optimize edilmesinde oldukça başarılı sonuçlar üretmiştir [38].

Tez çalışmasında literatürde yer alan ve sıklıkla kullanılan sezgisel optimizasyon algoritmaların matematiksel modellerinin, ilham kaynaklarının incelenme çalışması yapılmıştır. İnceleme çalışması sonunda popülasyondaki en iyi bireyi kullanarak kambur balinaların avlama davranışını modelleyen Balina Optimizasyon Algoritması, popülasyondaki en iyi üç bireyi kullanarak gri kurtların sosyal hiyerarşi ve avlama davranışını modelleyen Gri Kurt Optimizasyonunda olduğu gibi en iyi üç bireyin kullanılması ile gerçekleştirilerek Hibrit Balina Optimizasyon Algoritması oluşturulmuştur. Hibrit Balina Optimizasyon Algoritmasının 23 adet benchmark fonksiyonu kullanılarak etkinliği test edilmiştir.

Oluşturulan Hibrit Balina Optimizasyon Algoritması, kambur balinaların kuyruklarını suya vurma şiddetinin kambur balinaların konumunu değiştirmesinde etkili olmasından esinlenilerek modifiye edilmiştir. Hibrit Balina Optimizasyon Algoritmasının konum güncelleme denklemlerine kambur balinaların kuyruklarının suya vuruş şiddeti [0 1]

arasında rastgele bir değer alan çarpan olarak eklenmiştir ve İyileştirilmiş Hibrit Balina Optimizasyon Algoritması önerilmiştir. İyileştirilmiş Hibrit Balina Optimizasyon Algoritması 23 adet benchmark fonksiyonu ile test edilmiştir ve elde edilen sonuçlar Hibrit Balina Optimizasyon Algoritması, Balina Optimizasyon Algoritması, Gri Kurt Optimizasyonu, Parçacık Sürüsü Optimizasyonu, Diferansiyel Gelişim Algoritması ile karşılaştırılmıştır.

İHBOA'nın tek modlu benchmark problemlerindeki etkinliğinin ölçülmesi için Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Rosenbrock, Step, Quartic fonksiyonları kullanılmıştır. İHBOA; Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21 fonksiyonları için fonksiyonun minimum değerine ulaşmış, Rosenbrock ve Step fonksiyonları için ise rekabetçi sonuçlar üretmiş, Quartic fonksiyonu için ise karşılaştırılan diğer algoritmalardan daha iyi sonuç vermiştir.

İHBOA'nın çok modlu ve yüksek boyutlu benchmark problemlerindeki etkinliğinin ölçülmesi için Schwefel, Rastrigin, Ackley, Griewank, Penalized, Penalized2 fonksiyonları kullanılmıştır. İHBOA; Schwefel ve Ackley fonksiyonları için en iyi sonucu üretmişken Rastrigin ve Griewank fonksiyonu için fonksiyonun minimum değerine ulaşmıştır. Penalized, Penalized2 fonksiyonlarında ise en iyi üçüncü algoritmadır.

İHBOA'nın çok modlu ve düşük boyutlu benchmark problemlerindeki etkinliğinin ölçülmesi için Foxholes, Kowalik, Six Hump Camel Back, Branin, GoldStein-Price, Hartman3, Hartman, Shekel5, Shekel7, Shekel10 fonksiyonları ulanılmıştır ve İHBOA başarılı ve rekabetçi sonuçlar üretmiştir.

Tablo 5.1'de görüldüğü gibi İHBOA, tez kapsamında kıyaslama yapılan diğer algoritmalara göre 7 adet tek modlu benchmark probleminden 5 adedinde, 6 adet çok modlu ve yüksek boyutlu benchmark probleminden 4 adedinde daha iyi sonuçlar üretmiştir. Diğer taraftan 10 adet çok modlu ve düşük boyutlu benchmark problemlerinde tez kapsamında kıyaslama yapılan diğer algoritmalara göre iyi sonuçlar üretememiştir.

İyileştirilmiş Hibrit Balina Optimizasyon Algoritması'nın etkinliğinin ölçülebilmesi için kullanılan benchmark problemlerinde, İyileştirilmiş Hibrit Balina Optimizasyon Algoritması karşılaştırılan diğer optimizasyon algoritmalarına göre oldukça başarılı ve rekabetçi sonuçlar üretmiştir.

KAYNAKÇA

1. Winston, Wayne L. and Goldberg, Jeffrey B. (2004). Operations research: applications and algorithms. Belmont: Thomson Brooks/Cole.
2. Holland, John H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.
3. Erdoğmuş, Pakize. (2016). Doğadan esinlenen optimizasyon algoritmaları ve optimizasyon algoritmalarının optimizasyonu. **Düzce Üniversitesi Bilim ve Teknoloji Dergisi**, 4 (1).
4. Koza, John R. Genetic programming II, automatic discovery of reusable subprograms. MIT Press, Cambridge, MA, 1992.
5. Storn, Rainer; Price, Kenneth. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, **11** (4): 341-359.
6. Kennedy, J. and Eberhart R.C. (1995). "Particle swarm optimization," *in Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, Perth, Australia, IEEE Service Center, Piscataway, NJ, pp. 1942.
7. Dorigo, M., M.V., Colorni A. (1991). The ant system: An autocatalytic optimizing process.
8. Mirjalili, S. and Lewis A. (2016). The whale optimization algorithm. **Advances in Engineering Software**, **95**: 51-67.
9. Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. **Computers & Structures**, **169**: 1-12.
10. Mirjalili, S., Mirjalili S.M. and Lewis A. (2014). Grey wolf optimizer. **Advances in engineering software**, **69**: 46-61.
11. Karaboğa D., Baştürk B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. **Journal of global optimization**, **39** (3): 459-471.

12. Kaveh A., Farhoudi N. (2013). A new optimization method: dolphin echolocation. **Advances in Engineering Software**, **59**: 53-70.

13. Yang, X.S. (2009). Firefly algorithms for multimodal optimization. In: *International symposium on stochastic algorithms*. Springer, Berlin, Heidelberg, p. 169-178.

14. Gandomi A.H., Alavi A.H. (2012). Krill herd: a new bio-inspired optimization algorithm. **Communications in Nonlinear Science and Numerical Simulation**, **17** (12): 4831-4845

15. Yang X-S. (2010). A new metaheuristic bat-inspired algorithm. In: Proceedings of the workshop on nature inspired cooperative strategies for optimization (NICSO 2010). Springer; 2010. p. 65–74.

16. Dhiman G. Kumar V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. **Advances in Engineering Software**, **114**: 48-70.

17. Mirjalili Seyedali et al. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. **Advances in Engineering Software**, **114**: 163-191.

18. Uymaz S.A., Tezel G. and Yel E. (2015). Artificial algae algorithm (AAA) for nonlinear global optimization. **Applied Soft Computing**, **31**: 153-171.

19. Kirkpatrick S., Gelatt C.D., Vecchi M.P., (1983). Optimization by simulated annealing. **Science**, **220**: 671-680.

20. Rashedi, E., Nezamabadi-Pour H., Saryazdı Saeid. (2009). GSA: a gravitational search algorithm. **Information sciences**, **179** (13): 2232-2248.

21. Abdechiri, M., Meybodi M.R., Bahrami H. (2013). Gases Brownian motion optimization: an algorithm for optimization (GBMO). **Applied Soft Computing**, **13** (5): 2932-2946.

22. Abedınpourshotorban, H. et al. (2016). Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. **Swarm and Evolutionary Computation**, **26**: 8-22.

23. Kashan A.H. (2015). A new metaheuristic for optimization: optics inspired optimization (OIO). **Computers & Operations Research**, **55**: 99-125.

24. Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. **Information sciences**, **222**: 175-184.

25. Kaveh, A., Khayatazad M. (2012). A new meta-heuristic method: ray optimization. **Computers & structures**, **112**: 283-294.

26. Yang, X.-S. (2010). Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons.

27. Karaboğa, D. (2014). Yapay Zeka Optimizasyon Algoritmalari. Nobel Akademik Yayıncılık, 245.

28. Yang, X.S., (2010) 'Firefly Algorithm, Stochastic Test Functions and Design Optimisation', **Int. J. Bio-Inspired Computation, 2** (2):78–84.

29. Talbi E., (2009). Metaheuristics from design to implementation. Hoboken: John Wiley & Sons.

30. Eberhart R., Kennedy J. (1995). A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science.* IEEE, 39-43.

31. Eberhart, R., Kennedy, J. (1995). Particle swarm optimization. In: *Proceedings of the IEEE international conference on neural networks.* 1942-1948.

32. Watkins, William A.; Schevill, William E. (1979). Aerial observation of feeding behavior in four baleen whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaeangliae, and Balaenoptera physalus. **Journal of Mammalogy**, **60** (1): 155-163.

33. Mech, L.D. (1999). Alpha status, dominance, and division of labor in wolf packs. **Canadian Journal of Zoology**, **77** (8): 1196-1203.

34. Muro, C, et al. (2011). Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. **Behavioural processes**, **88** (3): 192-197.

35. Jamil M, Yang X-S. (2013). A literature survey of benchmark functions for global optimization problems. arXiv preprint arXiv:1308.4008.

36. Danacı M., Doğan C. (2019). *Improved Hybrid Whale Optimization Algorithm.*Uluslararası Erciyes Bilimsel Araştırmalar Kongresi, Kayseri, 26-28 Nisan 2019.

37. Lee, K.S., Geem Z.W. (2004). A new structural optimization method based on the harmony search algorithm. **Computers & structures**, **82**(9-10):781-798.

38. Cheng, M-Y., Prayogo, Doddy (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. **Computers & Structures**, **139**:98-112.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Cansu DOĞAN
Uyruğu: Türkiye (T.C)
Doğum Tarihi ve Yeri: 15.09.1993 - Melikgazi
Medeni Durum: Bekar
e-mail: cansudogan00@hotmail.com
Yazışma Adresi: Göksu Mahallesi Selçuklular Caddesi Göksupark Konutları No:47/23 Etimesgut/ANKARA

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Yüksek Lisans	Erciyes Üniversitesi, Bilgisayar Mühendisliği	2019
Lisans	Erciyes Üniversitesi, Harita Mühendisliği	2017
Lisans	Erciyes Üniversitesi, Bilgisayar Mühendisliği	2016
Lise	Akıncı Anadolu Lisesi, Ankara	2011

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2018-Halen	Savunma Sanayii Başkanlığı	Proje Mühendisi

YABANCI DİL

İngilizce