T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÖZNİTELİK SEÇİMİ PROBLEMLERİ İÇİN KARŞITLIK TABANLI ORMAN OPTİMİZASYONU ALGORİTMASI

Büşra YILDIRIM

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Ocak-2023
KONYA

TEZ KABUL VE ONAYI

Büşra YILDIRIM tarafından hazırlanan "Öznitelik Seçimi Problemleri İçin Karşıtlık Tabanlı Orman Optimizasyonu Algoritması" adlı tez çalışması 27/01/2023 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Dr. Öğr. Üyesi Levent CİVCİK	……..
Danışman Prof. Dr. Adem Alpaslan ALTUN	……..
Üye Dr. Öğr. Üyesi Tahir SAĞ	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Sait GEZGİN
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza

Büşra YILDIRIM

Tarih: 27/01/2023

ÖZET

YÜKSEK LİSANS TEZİ

ÖZNİTELİK SEÇİMİ PROBLEMLERİ İÇİN KARŞITLIK TABANLI ORMAN OPTİMİZASYONU ALGORİTMASI

Büşra YILDIRIM

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Adem Alpaslan ALTUN

2023, 77 Sayfa

Jüri
Prof. Dr. Adem Alpaslan ALTUN
Dr. Öğr. Üyesi Tahir SAĞ
Dr. Öğr. Üyesi Levent CİVCİK

Bu çalışmada popülasyon tabanlı bir meta-sezgisel algoritma olan Orman Optimizasyonu Algoritmasının (FOA) performansını iyileştirebilmek için algoritmaya Karşıtlık Tabanlı Öğrenme (OBL) yöntemi uygulanmıştır. Önerilen algoritma OFOA olarak isimlendirilmiştir. OFOA, ilk popülasyonun başlatılmasında ve iteratif sürecinde OBL yöntemini kullanır. Başlangıç popülasyonunu oluşturulması aşamasında OBL yönteminin kullanılmasıyla popülasyon çeşitliliği artırılarak probleme daha iyi aday çözümlerle başlama olasılığı artırılır. İteratif süreçte kullanım ile bazı iterasyonlarda mevcut popülasyonun uygunluğunu iyileştirmek ve daha iyi arama kabiliyeti ile yerel optimallerden kurtulmak amaçlanmıştır.

İlk olarak OFOA algoritmasının performansı çeşitli kıyaslama fonksiyonları üzerinde FOA algoritması ile karşılaştırılarak incelenmiştir. Değerlendirme ölçütü olarak ortalama uygunluk, en iyi uygunluk, en kötü uygunluk, standart sapma ve başarı oranı kullanılmıştır. Yapılan deneysel testlerde OFOA algoritması daha iyi bir performans göstermiştir ve elde edilen sonuçlar OBL yönteminin arama uzayını keşfetme yeteneğini artırdığını böylece yerel minimumlara tuzaklanma eğilimini azalttığını göstermiştir.

İkinci olarak ayrık bir optimizasyon problemi olan öznitelik seçiminde performansın incelenebilmesi için İkili Karşıtlık Tabanlı Orman Optimizasyonu Algoritması (B-OFOA) olarak adlandırılan OFOA algoritmasının ikili versiyonu sunulmuştur. B-OFOA algoritmasının performansı farklı ölçeklerdeki veri setleri üzerinde literatürden seçilen üç algoritma ile sınıflandırma doğruluğu ve seçilen öznitelik sayısı açısından karşılaştırılmıştır. Elde edilen öznitelik seçimi sonuçları OBL yönteminin getirilerini destekler niteliktedir.

Çalışmada FOA algoritmasına OBL yönteminin entegrasyonu ile algoritmanın arama uzayını keşfetme yeteneğini artırılabileceği tespit edilmiştir. Keşif aşamasının iyileştirilmesiyle OFOA algoritması, orijinal algoritmaya kıyasla daha az yerel optimallere tuzaklanma eğilimi göstermektedir. Bunun yanı sıra daha erken iterasyonlarda arama uzayının umut verici bölgeleri keşfedebilmekte ve optimal çözüme daha hızlı yakınsama sağlayabilmektedir.

Anahtar Kelimeler: Karşıtlık tabanlı öğrenme, Meta-sezgisel algoritmalar, Optimizasyon problemleri, Öznitelik seçimi, Popülasyon tabanlı algoritmalar, Sürü zekası.

ABSTRACT

MS THESIS

OPPOSITION-BASED FOREST OPTIMIZATION ALGORITHM FOR FEATURE SELECTION PROBLEMS

Büşra YILDIRIM

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Prof. Dr. Adem Alpaslan ALTUN

2023, 77 Pages

Jury
Prof. Dr. Adem Alpaslan ALTUN
Assist. Prof. Dr. Üyesi Tahir SAĞ
Assist. Prof. Dr. Levent CİVCİK

In this study, Opposition-Based Learning (OBL) method was applied to the algorithm in order to improve the performance of Forest Optimization Algorithm (FOA), which is a population-based meta-heuristic algorithm. The proposed algorithm is called as Opposition-Based Forest Optimization Algorithm (OFOA). OFOA uses the OBL method for initial population and its iterative process. By using the OBL method at the stage of creating the initial population, the population diversity is increased and the probability of starting the problem with better candidate solutions is increased. With the use of the method in the iterative process, it is aimed to improve the fitness of the current population in some iterations and to get rid of local optima with better search capability.

Firstly, the performance of OFOA algorithm is examined by comparing it with the FOA algorithm on various benchmark functions. Average fitness, best fitness, worst fitness, standard deviation and success rate were used as evaluation criteria. In the experimental tests, the OFOA algorithm performed better and the results showed that the OBL method increased the ability to explore the search space, thus reducing the tendency to be trapped in local minima.

Secondly, a binary version of OFOA algorithm called as Binary Opposition-Based Forest Optimization Algorithm (B-OFOA) is introduced to examine the performance in feature selection, which is a discrete optimization problem. The performance of the B-OFOA algorithm was compared with three algorithms selected from the literature on data sets of different scales in terms of classification accuracy and the number of selected features. The feature selection results obtained support the benefits of the OBL method.

In the study, the ability of the algorithm to explore the search space has been increased with the integration of the OBL method into the FOA algorithms. With the improvement of exploration stage, the OFOA algorithm tends to be trapped in less local optima compared to the original algorithm. In addition, in earlier iterations, promising regions of the search space can be discovered and convergence to the optimal solution can be achieved faster.

Keywords: Feature selection, Meta-heuristic algorithms, Opposition-based learning, Optimization problems, Population-based algorithms, Swarm intelligence.

ÖNSÖZ

Yüksek lisans eğitimim ve bu tez çalışması sürecinde yol gösterici katkılarından ve desteğinden dolayı değerli danışman hocam Prof. Dr. Adem Alpaslan ALTUN'a sonsuz teşekkürlerimi sunarım.

Hayatım boyunca desteklerini benden hiçbir zaman esirgemeyen, aldığım her kararda yanımda olduklarını ve bana olan sevgilerini en içten şekilde hissettiren başta kıymetli annem ve babama, kız kardeşlerime ve tüm sevdiklerime sonsuz sevgi, saygı ve teşekkürlerimi belirtmek isterim.

Büşra YILDIRIM

KONYA-2023

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

SİMGELER VE KISALTMALAR

Simgeler

α, β	:	[0,1] aralığında iki rastgele değer
age	:	ağaçların yaş bilgisi
$area\_limit$	:	popülasyon boyutu
Acc	:	sınıflandırma doğruluğu
AVG_F	:	uygunluk değerlerinin ortalaması
$BEST_F$	:	en iyi uygunluk değeri
c_1, c_2	:	ivme katsayıları
$\|C\|$	:	toplam öznitelik sayısı
CR	:	BDE algoritmasında çaprazlama oranı
dx	:	yerel tohumlamada değişkenlere eklenen değer
f	:	uygunluk fonksiyonu
F	:	başlangıç popülasyonu
F_i^*	:	algoritmaların elde ettiği en iyi uygunluk değeri
g, h	:	kısıt fonksiyonu
$Gbest$	:	küresel optimal çözüm
GSC	:	küresel tohumlama değişiklikleri
J_r	:	atlama oranı
$life\_time$	:	bir ağacın izin verilen maksimum yaşı
LSC	:	yerel tohumlama değişiklikleri
N	:	problem boyutu
N_r	:	farklı çalıştırmaların sayısı
OF	:	karşıt başlangıç popülasyonu
$Pbest$	:	BPSO algoritmasında bir parçacığın optimal konumu
$\|R\|$	:	seçilen öznitelik sayısı
SR	:	başarı oranı
STD_F	:	standart sapma
$transfer\_rate$	:	aday popülasyondan seçilecek ağaçların yüzdesi
T	:	OFOA algoritmasında aday çözüm
$\tilde{T}$	:	karşıt aday çözüm
v	:	parçacığın hızı
w	:	eylemsizlik ağırlığı
$WORST_F$	:	en kötü uygunluk değeri
x	:	parçacığın konumu
x_i	:	karar değişkenleri

Kısaltmalar

ALO	:	Karınca Aslanı Optimizasyonu
ACO	:	Karınca Kolonisi Optimizasyonu
BALO	:	İkili Karınca Aslanı Optimizasyonu
BDE	:	İkili Diferansiyel Evrim
B-OFOA	:	İkili Karşıtlık Tabanlı Orman Optimizasyonu Algoritması
BPSO	:	İkili Parçacık Sürü Optimizasyonu
CSA	:	Karga Arama Algoritması
DE	:	Diferansiyel Evrim

DM	:	Veri Madenciliği
EJaya	:	Gelişmiş Jaya Algoritması
FOA	:	Orman Optimizasyonu Algoritması
FSFOA	:	Orman Optimizasyonu Algoritması Kullanarak Öznitelik Seçimi
FSIFOA	:	İyileştirilmiş Orman Optimizasyonu Algoritması
GA	:	Genetik Algoritma
GOA	:	Çekirge Optimizasyonu Algoritması
GWO	:	Karşıtlık Tabanlı Gri Kurt Optimizasyonu
HHO	:	Harris Şahini Optimizasyonu
HS	:	Harmoni Arama
KDD	:	Veri Tabanlarında Bilgi Keşfi
m-EO	:	Değiştirilmiş Denge Optimizasyonu
MOEA/D	:	Ayrıştırma Tabanlı Çok Amaçlı Evrimsel Algoritma
OBL	:	Karşıtlık Tabanlı Öğrenme
ODE	:	Karşıtlık Tabanlı Diferansiyel Evrim
OFOA	:	Karşıtlık Tabanlı Orman Optimizasyonu Algoritması
PSO	:	Parçacık Sürü Optimizasyonu
SA	:	Benzetilmiş Tavlama
SCA	:	Sinüs Kosinüs Algoritması
SDE	:	Karıştırılmış Diferansiyel Evrim
SSA	:	Salp Sürü Algoritması
SVM	:	Destek Vektör Makineleri
TLBO	:	Öğretme Tabanlı Öğrenme Optimizasyon Algoritması
TS	:	Tabu Arama
WOA	:	Balina Optimizasyonu Algoritması

1. GİRİŞ

Optimizasyon, belirli bir problem için mümkün olan en iyi çözümü bulmayı amaçlar. Optimizasyon problemlerinin çözümünde kullanılan farklı teknikler mevcuttur. Çok sayıda gerçek dünya optimizasyon problemi giderek daha karmaşık bir hal almıştır. Bu nedenle klasik teknikler problemlerin çözümü için yetersiz kalmaya başlamıştır. Problemlerin boyutu arttıkça klasik teknikler için optimizasyon problemlerini çözmek maliyetli bir hale gelirken çeşitli modern meta-sezgisel algoritmalar önerilmiştir ve zorlu optimizasyon problemlerini ele alırken güçlü performanslarını göstermeye başlamışlardır (Wang ve ark., 2018). Günümüzde optimizasyon problemlerinin çözümünde meta-sezgisel algoritmalar yaygın olarak tercih edilmektedir. Meta-sezgisel algoritmalara örnek olarak; Genetik Algoritma (GA), Parçacık Sürü Optimizasyonu (PSO), Diferansiyel Evrim (DE), Benzetilmiş Tavlama (SA) ve Orman Optimizasyonu Algoritması (FOA) verilebilir.

FOA algoritması, 2014 yılında Ghaemi ve Feizi-Derakhshi (Ghaemi ve Feizi-Derakhshi, 2014) tarafından önerilen popülasyon tabanlı meta-sezgisel bir algoritmadır. FOA algoritması, ağaçların doğadaki süreçlerini simüle eder. Algoritmada aday çözümler, ağaçlar ile temsil edilir. Ağaçların yerel tohumlanması, popülasyon sınırlaması ve ağaçların küresel tohumlanması olmak üzere üç temel aşama bulunmaktadır. Küresel tohumlama aşaması arama uzayında keşif yapar ve yerel tohumlama aşamasında popülasyondaki aday çözümler yerel olarak optimize edilir. Yerel ve küresel tohumlama aşamaları, meta-sezgisel algoritmalarda çok önemli iki kavram olan keşif ve kullanım fazlarını karşılar.

FOA stokastik yapıya sahip bir algoritmadır ve yapısı gereği başlangıç popülasyonu rastgele oluşturulur, çözüm süreci bu popülasyon üzerinden devam eder, yerel ve küresel arama işlemleri gerçekleştirilir. Başlangıç popülasyonun optimalden oldukça uzak olan kötü aday çözümlerle başladığı durumlarda algoritmanın optimal çözüme ulaşma süresi uzayabilir ya da algoritma optimal çözüme ulaşamayabilir. Ayrıca meta-sezgisel algoritmaların keşif ve kullanım fazları arasında iyi bir denge kurabilmesi oldukça önemlidir. Bu dengenin kurulması doğrudan algoritmanın performansını etkiler ve kurulamadığı durumlarda istenmeyen sonuçlarla karşılaşılır. Literatürde bu gibi durumların önüne geçebilmek, algoritmaların zayıf yönlerini iyileştirebilmek ve performanslarını yükseltebilmek için önerilen birçok yöntem bulunmaktadır. Bu yöntemlerden biri 2005 yılında Tizhoosh (Tizhoosh, 2005) tarafından önerilen Karşıtlık

Tabanlı Öğrenme (OBL) yöntemidir. OBL, daha iyi bir aday çözüm elde edebilmek amacıyla mevcut bir aday çözümün, problem uzayı içerisindeki karşıtını da eş zamanlı olarak değerlendirmeye alıp daha uygun olan durumla devam edilmesini önerir.

Öznitelik seçimi, optimizasyon teknikleriyle çözülebilen veri madenciliği alanındaki zorlu gerçek dünya problemlerdendir. Öznitelik seçiminde amaç, bir veri setindeki sınıflandırmayı etkilemeyen gereksiz ve alakasız özniteliklerin saptanması ve veri setinden kaldırılmasıdır. Bununla birlikte veri setinde bırakılan özniteliklerle sınıflandırma doğruluğunun korunması beklenir.

Optimizasyon algoritmalarının güvenilirlik, verimlilik ve geçerlilik testi sıklıkla literatürden seçilen bir dizi standart kıyaslama fonksiyonu kullanılarak gerçekleştirilir. Kullanılan test fonksiyonları çeşitli ve tarafsız olmalıdır, ancak literatürde anlaşmaya varılmış bir kıyaslama fonksiyonu seti yoktur. Bir algoritmayı değerlendirmek için kıyaslama fonksiyon setinin tek modlu, çok modlu, düzenli, düzensiz, ayrılabilir, ayrılamaz ve çok boyutlu problemler gibi çeşitli problemleri içermesi gerekir (Jamil ve Yang, 2013).

1.1. Çalışmanın Önemi

Meta-sezgisel algoritmalar, optimizasyon problemlerini çözme konusunda oldukça başarılıdır ve yetenekleriyle tercih sebebidir. Çeşitli optimizasyon problemlerinin çözümü için sürekli yeni bir meta-sezgisel algoritma önerilmektedir. Bunun yanı sıra literatürde meta-sezgisellerin performansını artırmak ve zayıf yönlerini iyileştirebilmek için önerilen çeşitli yöntemler bulunmaktadır. Algoritmaların performanslarını artırabilmek için yapılan bu iyileştirme çalışmaları oldukça fazladır ve ilgi çeken bir araştırma alanıdır.

FOA algoritması ilk olarak sürekli optimizasyon problemleri için önerilmiştir. Daha sonra algoritmanın ayrık optimizasyon problemi olarak ele alınabilen öznitelik seçimi için ikili versiyonu FSFOA tanıtılmıştır. FOA algoritması başarılı sonuçlar elde edebilmektedir ancak nispeten arama verimliliği düşük bir algoritmadır (Ma ve ark., 2019). Xie ve ark. FSFOA algoritmasının bazı eksikleri olduğunu bildirmiştir ve bu eksikliklerin rastgele başlatma, aday popülasyonu oluşturma ve en iyi ağacı güncelleme aşamalarında olduğunu belirtmiştir (Xie ve ark., 2020). Bu nedenle arama yeteneğini geliştirebilecek yöntemler kullanılarak algoritmanın performansı iyileştirilebilir.

FOA ve FSFOA algoritmalarının performansını iyileştirmek için yapılan çalışmaların sayısı sınırlıdır. Bu çalışmada FOA algoritmasının rastgele popülasyon başlatması ve düşük arama verimliliği ele alınmıştır. Bu alandaki sınırlı çalışmalar nedeniyle FOA algoritmasının arama yeteneği ve başlatma aşaması geliştirilerek literatüre daha verimli bir algoritma olarak kazandırılması önemlidir. OBL yönteminin 2005'te önerilmesinden sonra yöntem birçok meta-sezgisel algoritmaya uygulanmıştır ve algoritmaların arama yeteneğini ve popülasyon çeşitliliğini artırdığı birçok çalışmada bildirilmiştir. Bu nedenle çalışmada OBL yöntemi belirtilen amaçlar doğrultusunda tercih edilmiştir.

1.2. Çalışmanın Amacı ve Yöntemi

Bu çalışmada OBL yöntemi kullanılarak FOA algoritmasına arama uzayını daha iyi bir şekilde keşfetme yeteneği kazandırmak ve böylece algoritmanın arama verimliliğin iyileştirilmesiyle performansının artırılması amaçlanmıştır. Böylelikle algoritmanın yerel optimallere tuzaklanma eğiliminin azalması, daha iyi aday çözümleri bulabilmesi ve yakınsama hızının artması beklenir. Ayrıca algoritmanın optimizasyon sürecine daha iyi bir başlangıç popülasyonu ile başlayabilmesini sağlamak da çalışmanın amaçları kapsamındadır. Bu amaçlarla önerilen algoritma Karşıtlık Tabanlı Orman Optimizasyonu Algoritması (OFOA) olarak isimlendirilmiştir. Yöntem algoritmaya başlangıç popülasyonun başlatılması ve iteratif süreçte uygulanmıştır.

Öznitelik seçimi, bir ayrık optimizasyon problemidir. Bu çalışmada öznitelik seçiminde önerilen algoritmanın performansını değerlendirmek için algoritmanın ikili versiyonu (B-OFOA) tanıtılmıştır. Öznitelik seçiminde algoritmanın yapılan modifikasyonlarla arama uzayını daha iyi araştırıp daha az öznitelik sayıları ile daha iyi sınıflandırma doğruluğu elde edebilmesi beklenir.

1.3. Tezin Organizasyonu

Bu çalışmada sürekli optimizasyon problemleri için OBL yöntemini kullanan OFOA algoritması önerilmiştir. Ayrıca öznitelik seçimi için önerilen algoritmanın ikili versiyonu B-OFOA geliştirilmiştir. Geliştirilen algoritmaların optimizasyon ve öznitelik seçimi problemleri üzerindeki performansı incelenmiştir. Çalışma 8 bölümden oluşmaktadır.

İlk olarak, OFOA algoritmasının optimizasyon problemlerindeki performansı, farklı özelliklere sahip 15 kıyaslama fonksiyonu üzerinde FOA algoritması ile karşılaştırmalı olarak incelenmiştir. Ayrıca popülasyon büyüklüğünün ve problem boyutunun algoritmaların performanslarına etkisini incelemek için de deneysel testler yapılmıştır.

İkinci olarak, öznitelik seçimi probleminin çözümünde B-OFOA algoritmasının performansı incelenmiştir. Deneysel testlerde 9 farklı veri seti üzerinde B-OFOA, FSFOA, İkili Diferansiyel Evrim (BDE) ve İkili Parçacık Sürü Optimizasyonu (BPSO) algoritmalarının performansları karşılaştırılmıştır.

Bu çalışma 8 bölümden oluşmaktadır. Birinci bölüm giriş bölümüdür ve bu bölümde bu tez çalışmasında kullanılan kavramların ve yöntemlerin genel tanımları yapılmıştır ve genel olarak anlatılmıştır. Çalışmanın önemi belirtilip amacı ve yöntemi açıklanmıştır.

İkinci bölümde; FOA ve FSFOA algoritmalarının performansını geliştirmek için yapılan çalışmalar, OBL yönteminin meta-sezgisel algoritmalara uygulanması ve meta-sezgisellerle öznitelik seçimi probleminin çözümü ile alakalı literatür araştırması sunulmuştur.

Üçüncü bölümde; ilk olarak optimizasyon problemlerinin genel bir açıklaması ve matematiksel modeli verilmiştir. Ardından meta-sezgisel algoritmaların optimizasyon problemlerinin çözümündeki rolü anlatılmıştır. Son olarak bu çalışmada kullanılan meta-sezgisel bir algoritma olan FOA algoritmasının genel prosedürü açıklanmıştır.

Dördüncü bölümde; OBL yönteminin temel tanımları, optimizasyonda kullanımı ve algoritmaların performanslarına etkileri açıklanmıştır.

Beşinci bölümde; OBL yönteminin FOA algoritmasına hangi amaçlarla ve ne şekillerde uygulandığı anlatılmıştır. Ardından geliştirilen OFOA algoritmasının genel prosedürü sunulmuştur.

Altıncı bölümde; öznitelik seçimi problemlerini açıklamak için öncelikle veri madenciliği ve bilgi keşfi kavramları anlatılmıştır. Öznitelik seçiminde meta-sezgisel algoritmaların kullanımı açıklanmıştır ve son olarak OFOA algoritmasının öznitelik seçimi problemlerinin çözümü için uyarlanan ikili versiyonu B-OFOA algoritmasıyla birlikte FSFOA, BDE ve BPSO algoritmaları sunulmuştur.

Yedinci bölümde; geliştirilen OFOA algoritması ve ikili versiyonu B-OFOA algoritmalarının performanslarının değerlendirilmesi için yapılan deneysel testlere ve sonuçların analizine yer verilmiştir.

Sekizinci ve son bölümde; bu tez çalışmasında elde edilen sonuçlar değerlendirilmiştir ve araştırmacılara çalışmanın geliştirilmesi için çeşitli önerilerde bulunulmuştur.

2. KAYNAK ARAŞTIRMASI

FOA algoritması sürekli optimizasyon problemleri ve FSFOA algoritması öznitelik seçimi problemleri için önerilmiş meta-sezgisel algoritmalardır. Bu algoritmalar başarılı sonuçlar elde edebilse de performanslarını düşüren zayıf yönleri bulunmaktadır. Bu algoritmaların performansını artırabilmek için yapılan çalışmalardan bazıları şunlardır;

2018 yılında Chu ve ark. yaptıkları çalışmada yeni bir başlatma stratejisi, yeni bir güncelleme mekanizması ve yerel tohumlama aşamasında açgözlü bir strateji ekleyerek yeni bir öznitelik seçimi algoritması önermiştir. Elde edilen sonuçlar, FSFOA algoritmasına kıyasla önerilen algoritmanın sınıflandırma doğruluğu ve boyut azaltmada önemli bir iyileştirmeye sahip olduğu göstermiştir (Chu ve ark., 2018).

Ma ve ark. tarafından 2018 yılında katkı derecesine dayalı olarak FOA algoritmasını kullanan yeni bir öznitelik seçimi yöntemi önerilmiştir. Katkı derecesinin amacı, FOA algoritmasının arama sürecini, öznitelikler arasında yüksek sınıf korelasyonuna ve düşük artıklığa göre öznitelikleri seçmek için yönlendirmektedir. Önerilen yöntemin sınıflandırma doğruluğunu geliştirdiği belirtilmiştir (Ma ve ark., 2018).

2019 yılında Ma ve ark. tarafından yerel tohumlama aşaması için yeni bir yerel arama stratejisi önerilmiştir (Ma ve ark., 2019). Yerel optimumlara tuzaklanmanın önüne geçmek için, aday kümenin altındaki ağaçları seçen ve algoritmaya daha fazla çeşitlilik veren yeni bir küresel tohumlama yöntemi denenmiştir. Önerilen algoritma dört öznitelik seçim yöntemi ile karşılaştırılmıştır. Sonuçların çoğu önerilen algoritmanın daha üstün olduğunu göstermiştir.

2020 yılında Xie ve ark. FSFOA algoritmasının rastgele başlatma, aday popülasyonu oluşturma ve en iyi ağacı güncelleme aşamalarındaki sorunlarını çözmeyi amaçlayan İyileştirilmiş Orman Optimizasyonu Algoritmasını (FSIFOA) önermiştir (Xie ve ark., 2020). Çalışmada üç iyileştirme önerilmiştir; ilk öneri, başlatma aşamasında rastgele başlatma yerine Pearson korelasyon katsayısı ve L1 düzenlemesinin kullanılmasıdır. İkinci öneri, iyi ağaçlar ve kötü ağaçların ayrılmasıdır. Son öneri, güncelleme aşamasında, ormana aynı hassasiyette ancak farklı boyutlarda ağaçların eklenmesidir. Algoritma küçük, orta ve büyük boyutlu veri setlerinde test edilmiştir. Elde edilen sonuçlar algoritmanın orijinali ve literatürden birkaç algoritma ile karşılaştırılmıştır. FSIFOA algoritmasının orta ve büyük boyutlu veri setlerinde FSFOA

algoritmasından daha iyi olduğu ve 10 veri setinden 6'sında diğer algoritmalara göre avantajlara sahip olduğu bildirilmiştir.

Baliarsingh ve ark. 2020'de mikro dizi verilerinin öznitelik seçimi ve sınıflandırılması için yeni bir yöntem sunmuştur (Baliarsingh ve ark., 2020). Çalışmada önceden seçilen genlerden en uygun gen setini bulmak için Gelişmiş Jaya Algoritması (EJaya) ve FOA algoritması ilkelerini kullanan evrimsel sarmal tabanlı bir yaklaşım önerilmiştir. EJaya, FOA algoritmasının yerel tohumlama değişiklikleri ve küresel tohumlama değişiklikleri parametrelerini ayarlamak için kullanılmıştır. Önerilen yöntem hem ikili sınıf hem de çok sınıflı mikro dizi veri setleri üzerinde test edilmiştir ve başarılı sonuçlar elde edildiği belirtilmiştir.

Porkodi ve ark. 2022'de çalışan yıpranmasını tahmin etmek için iki aşamalı bir tahmin modeli sunmuştur. Birinci aşamada, sınıflandırma doğruluğunu iyileştirmek amacıyla önemli öznitelikleri seçmek için gelişmiş bir ağırlıklı FOA algoritması önerilmiş ve kullanılmıştır (Porkodi ve ark., 2022). Başarılı sonuçlar elde edilebildiği bildirilmiştir.

OBL yöntemi algoritmaların performansını artırmak, daha hızlı bir yakınsama sağlamak, yerel optimumlardan kaçınmaya yardımcı olmak gibi çeşitli amaçlarla ve farklı şekillerde meta-sezgisel algoritmalarda uygulama alanı bulmuştur. Literatürde bunu konu eden birçok çalışma bulunmaktadır. Bu çalışmalardan başlıcaları şunlardır;

Karşıtlık Tabanlı Diferansiyel Evrim (ODE), 2008 yılında Rahnamayan ve ark. (Rahnamayan ve ark., 2008) tarafından önerilmiştir. Çalışmada DE algoritmasının yakınsama oranını iyileştirmek için OBL kullanılmıştır. ODE, OBL yöntemini popülasyon başlatma ve nesil atlamada kullanır. Deneysel sonuçlara göre ODE algoritmasının yakınsama hızı ve çözüm doğruluğu açısından klasik DE algoritmasına göre daha iyi performans gösterdiği belirtilmiştir. Karşıtlık tabanlı nesil atlamada kullanılan Jr parametresinin [0.1 , 0.4] aralığında kullanılması önerilmiştir.

2008 yılında Omran ve Al-Sharhan (Omran ve Al-Sharhan, 2008) OBL yöntemini farklı şekillerde PSO algoritmasına uygulamış ve sonuçları karşılaştırmıştır. Çalışmada üç PSO varyantı önerilmiştir. Versiyonlardan ilki karşıtlık tabanlı popülasyon başlatmayı kullanır ve aynı zamanda uygunluğu en kötü olan parçacığı karşıt parçacık ile değiştirir. İkinci versiyon en kötü parçacığı karşıtı ile değiştirir. Son versiyon ise karşıtlık tabanlı popülasyon başlatmayı kullanır. Sonuç olarak birinci ve ikinci versiyonun diğer yaklaşımlardan daha iyi sonuç verdiği ve buna ek olarak OBL kullanımını ek parametreler gerektirmeden PSO algoritmasının performansını artırdığı belirtilmiştir.

Gao ve ark. 2012'de Harmoni Arama (HS) ve OBL yöntemini birleştiren yeni bir optimizasyon yöntemi önermiştir. OBL yöntemi, klasik HS algoritmasının mutasyon işlemini geliştirmek için kullanılmıştır (Gao ve ark., 2012).

2012 yılında Ahandani ve Alavi-Rad tarafından OBL ve Karıştırılmış Diferansiyel Evrim (SDE) stratejilerine dayalı olarak DE algoritmasının dört versiyonu önerilmiştir. Versiyonların hepsinde daha uygun başlangıç bireylerinin elde edilmesi için karşıtlık tabanlı popülasyon başlatma kullanılmıştır. Versiyonlar arasındaki fark OBL yönteminin nesil atlamasında uygulanışıdır (Ahandani ve Alavi-Rad, 2012).

2014'te Liu ve ark. tarafından (Liu ve ark., 2014) Ayrıştırma Tabanlı Çok Amaçlı Evrimsel Algoritmasına (MOEA/D) yakınsama hızını artırmak için OBL uygulanmıştır. Önerilen yöntem evrimsel süreç sırasında yeni nesiller oluşturmak için karşıtlık tabanlı başlangıç popülasyonu ve karşıtlık tabanlı öğrenme stratejisi kullanır.

2016 yılında Fang ve ark. tarafından yeni bir PSO versiyonu önerilmiştir. Bu versiyon rekabetçi öğrenme ve karşıtlık tabanlı öğrenme yöntemlerini kullanır. Önerilen algoritmada orta uygunluktaki parçacık, OBL yöntemini kullanarak kendini günceller. OBL arama uzayını hızlı bir şekilde arayabilmek için kullanılmıştır (Fang ve ark., 2016).

2017 yılında Xiong ve ark. küresel optimizasyon için Sinüs Kosinüs Algoritmasında (SCA) OBL yöntemini uyguladılar (Xiong ve ark., 2017). OBL daha doğru çözümler üreten arama uzayının daha iyi keşfedilmesi amacı ile kullanılmıştır. Sonuçlar, önerilen yaklaşımın karmaşık arama uzaylarında en uygun çözümleri bulma etkinliğini desteklemektedir.

OBL yöntemine dayanan Çekirge Optimizasyon Algoritması bu alandaki başka bir çalışmadır. 2018 yılında Ewees ve ark. tarafından (Ewees ve ark., 2018) önerilmiştir. Bu çalışmada, OBL Çekirge Optimizasyonu Algoritmasına (GOA) iki aşamada uygulanmıştır ve bu aşamalar şunlardır: başlangıç popülasyonu ve her yinelemede GOA popülasyonunu güncelleme. Ancak zaman karmaşıklığını azaltmak için OBL yönteminin çözümlerin yalnızca yarısına uygulanması önerilmiştir. Önerilen algoritmanın performansı 23 kıyaslama fonksiyonunda ve 4 mühendislik tasarım probleminde değerlendirilmiştir ve başarılı sonuçlar elde edilmiştir.

Bu konuda yapılan bir diğer çalışma, 2018 yılında Wang ve ark. (Wang ve ark., 2018) tarafından önerilen Gauss pertürbasyonu ile karşıtlık tabanlı kral kelebek optimizasyonudur. Çalışmada OBL yöntemi evrimin son aşamasında popülasyonun yarısına uygulanır ve OBL yönteminin daha yüksek yakınsama hızını garanti ettiği belirtilmiştir.

Kang ve ark. (Kang ve ark., 2018) tarafından 2018'de Parçacık Sürü Optimizasyonu (PSO) algoritmasının gürültüye uğramış optimizasyon problemlerinde performansının artırılması için OBL yöntemi PSO varyantlarına entegre edilmiştir. Çalışmada olasılıksal OBL kullanılmıştır ve başarılı sonuçlar elde edilmiştir. Elde edilen bu sonuçların nedenleri şu şekilde belirtilmiştir: OBL yönteminin PSO algoritmasının erken yakınsamadan kaçmasına yardımcı olması, sürü ve karşıt sürüden en uygun parçacıkların seçilmesiyle gürültülü ortamlarda sürü uygunluğunun iyileştirilmesi.

2019 yılında Gupta ve Deep (Gupta ve Deep, 2019) tarafından adaptif bileşen ve OBL kullanan SCA önerilmiştir. Önerilen algoritmada yerel optimumdan dışarı atlamak için pertürbasyon oranına dayalı karşıt sayılar kullanılarak zıt popülasyon oluşturulur.

Karga Arama Algoritmasına (CSA) 2020 yılında Shekhawat ve Saxena tarafından (Shekhawat ve Saxena, 2020) kosinüs fonksiyonu tabanlı ivme faktörü ve başlatma aşamasında OBL uygulanmıştır. Keşif sürecini hızlandırmak ve iteratif süreç boyunca sömürü süresini geciktirmek için bir kosinüs fonksiyonu önerilmiştir ve OBL, CSA algoritmasının keşif davranışını geliştirmek için dahil edilmiştir.

Moayedi ve ark. 2020'de dört farklı yöntem ile Harris Şahini Optimizasyonu (HHO) algoritmasının performansını geliştirmek için bir çalışma yapmıştır. Uygulanan dört yöntemden biri OBL yöntemidir ve dengeli bir şekilde uygulanmıştır. Dengeli bir şekilde uygulamadan anlaşılması gereken iterasyonlar boyunca karşıt çözümlerin sayısının azalmasıdır. Karşıt sayıların optimumlara sıkışıp kalmayı engellediği ve yakınsama oranını artırdığı belirtilmiştir (Moayedi ve ark., 2020).

2021 yılında Ewees ve ark. tarafından (Ewees ve ark., 2021) DE ve OBL yöntemini birleştirerek geliştirilmiş Balina Optimizasyonu Algoritmasına dayanan yeni bir çok amaçlı optimizasyon yöntemi sunulmuştur. Çalışmada, Balina Optimizasyonu Algoritmasının (WOA) keşif fazını, DE algoritmasının sömürü fazını ve OBL yönteminin ise karşıt değerler üreterek keşif ve sömürüyü geliştirmek için uygulandığı belirtilmiştir.

OBL ve diğer başka üç yöntem birleştirilerek değiştirilmiş bir Denge Optimizasyonu (m-EO), Huang ve ark. tarafından (Huang ve ark., 2021) 2021 yılında önerilmiştir. Çalışmada OBL yöntemi popülasyon çeşitliliğini artırmak amacıyla kullanılmıştır. Önerilen m-EO algoritmasının kararlı arama performansı, hızlı yakınsama hızı ve yüksek yakınsama doğruluğu sunduğu belirtilmiştir.

2021'de Yu ve ark. tarafından (Yu ve ark., 2021) küresel optimizasyon için karşıtlık tabanlı gri kurt optimizasyon algoritması (GWO) önerilmiştir. OBL yöntemi, GWO algoritmasına bir atlama oranı ile dahil edilmiştir; bu, algoritmanın yerel

optimumdan atlamasına yardımcı olur ve hesaplama karmaşıklığını artırmaz. Bunun dışında $\vec{\alpha}$ katsayısında da bir değişiklik uygulanmıştır. Sonuçlar, iki iyileştirmenin GWO algoritmasının optimizasyon ilerlemesini, hesaplama karmaşıklığını koruduğunu ve algoritmanın yere optimalin dışına çıkmasına yardımcı olduğunu göstermiştir.

2022'de Hussien tarafından OBL yöntemine dayanan yeni bir Salp Sürü Algoritması (SSA) versiyonu tanıtılmıştır. Önerilen algoritma başlatma aşamasını geliştirmek ve her iterasyonda popülasyonun güncellenmesi için OBL yöntemini kullanır. Algoritmanın performansı, IEEE Evrimsel Hesaplama Kongresi'nden 30 fonksiyon ve 6 mühendislik problemi üzerinde test edilmiştir. Algoritma literatürde iyi bilinen 8 algoritma ile karşılaştırılmıştır ve başarılı sonuçlar elde edilmiştir (Hussien, 2022).

Öznitelik seçimi probleminin çözümünde meta-sezgisel algoritmalar sarmal yöntemler kategorisinde yer alır ve meta-sezgisel algoritmalarla bu problemin çözümünde başarılı sonuçlar elde edilmiştir. Bu alandaki çalışmalardan bazıları şunlardır;

Öznitelik seçimi problemlerinde meta-sezgisel algoritmaya dayalı çözüm yaklaşımı ilk olarak 1997 yılında Yang ve Honavar tarafından (Yang ve Honavar, 1998) önerilmiştir. Çalışma bir genetik algoritma kullanarak öznitelik altkümesi seçiminin çok kriterli optimizasyon problemine bir yaklaşım sunar.

2016 yılında Emary ve ark. tarafından (Emary ve ark., 2016a) gri kurt optimizasyonunun yeni bir ikili versiyonu önerilmiş ve sınıflandırma amacıyla en uygun öznitelik alt kümesini seçmek için kullanılmıştır. Orijinal versiyondan iki farklı yaklaşım önerilmiştir. Önerilen versiyonlar, 18 farklı veri seti üzerinde PSO ve GA ile karşılaştırılmıştır. Sonuçlar önerilen ikili versiyonun başarısını kanıtlamıştır.

2016'da Emary ve ark. Karınca Aslanı Optimizasyonu (ALO) algoritmasının ikili varyantlarını önermiştir ve sarmal tabanlı sınıflandırma amacıyla en uygun öznitelik alt kümesini seçmek için kullanmıştır (Emary ve ark., 2016b). İkili Karınca Aslanı Optimizasyonu (BALO) için önerilen yaklaşımlar, seçilen özniteliklerin sayısını en aza indirirken sınıflandırma performansını en üst düzeye çıkarak en iyi öznitelik alt kümesini bulmaya çalışır. Sonuçlar, BALO algoritmasının, öznitelikler uzayını uyarlamalı olarak iyi bir şekilde arayabildiğini ve optimal çözüme yakınsayabildiğini göstermiştir.

Ghaemi ve Feizi-Derakhshi tarafından 2016'da FOA algoritması öznitelik seçiminde kullanılmak üzere uyarlanmıştır. 11 gerçek dünya veri seti literatürden birkaç yöntem ile karşılaştırılmıştır. Sonuçlar FOA algoritmasının öznitelik seçimi problemleri için etkili bir arama tekniği olduğunu göstermiştir (Ghaemi ve Feizi-Derakhshi, 2016).

2017 yılında Mafarja ve Mirjalili (Mafarja ve Mirjalili, 2017) WOA algoritmasına dayalı farklı öznitelik seçimi teknikleri tasarlamak için iki melezleme modeli kullanmıştır. Birinci modelde SA algoritması WOA algoritmasına entegre edilir, ikinci modelde SA algoritması WOA algoritmasının her iterasyonundan sonra bulunan en iyi çözümü iyileştirmek için kullanılır. Deneysel sonuçların, WOA algoritmasının öznitelik uzayını arama ve sınıflandırma görevleri için en bilgilendirici öznitelikleri seçme yeteneğini garanti ettiği ifade edilmiştir.

2018 yılında İbrahim ve ark. Destek Vektör Makineleri (SVM) faktörlerini optimize ederek ve GOA algoritmasını kullanarak öznitelik seçimi çalışması yapmıştır (Ibrahim ve ark., 2019). Önerilen algoritma yüksek sınıflandırma doğrulukları elde edebilmiştir.

Taghian ve Nadimi-Shahraki 2019 yılında (Taghian ve Nadimi-Shahraki, 2019) öznitelik seçimi problemleri için SCA algoritmasının sarmal tabanlı ikili bir versiyonunu önermiştir ve başarılı sonuçlar elde edilmiştir.

2.1. Kaynak Araştırmasının Değerlendirmesi

Yapılan kaynak araştırması FOA ve FSFOA algoritmalarının başarılı sonuçlar elde edebilmesine rağmen bazı zayıf yönlerinin bulunduğunu göstermektedir. Başlangıç popülasyonunun rastgele başlatılması ve arama verimsizliği birkaç çalışmada algoritmanın zayıf yönleri olarak gösterilmiştir. Bu algoritmaların performansını iyileştirmeye yönelik bazı çalışmalar yapılmıştır ancak yapılan bu iyileştirme çalışmaları sınırlı sayıdadır.

Meta-sezgisel algoritmalar genellikle optimizasyon problemlerinin çözümlerinde oldukça başarılıdır. Literatürde meta-sezgisel algoritmaların performanslarını daha da artırabilmek ve zayıf yönlerini güçlendirebilmek amacıyla önerilen yöntemlerden biri olan OBL, önerildiği zamandan günümüze kadar meta-sezgisel algoritmalara farklı şekillerde uygulanmıştır.

OBL yöntemine dayanan başlangıç popülasyonu ve nesil atlama, en yoğun ilgi gören ve başarılı sonuçlar veren uygulama şekilleridir. İlgili çalışmalarda OBL yönteminin meta-sezgisel algoritmaların yakınsama oranını, popülasyon çeşitliliğini, yakınsama hızını artırabildiği ve yerel optimallerden kaçmayı başarılı bir şekilde sağladığı tespit edilmiştir. Bu nedenle OBL yönteminin FOA algoritmasının performansına etkisi araştırılmayı bekleyen bir çalışma konusudur.

Öznitelik seçimi problemleri de ayrık optimizasyon problemi olarak çözülebilmektedir. Meta-sezgisel algoritmalar ile öznitelik seçiminde başarılı sonuçlar elde edilebilmektedir ve bu problemlerin çözümü için algoritmaların ikili versiyonları geliştirilmektedir.

3. MATERYAL VE YÖNTEM

Optimizasyon, çoğu mühendislik faaliyeti için yaygın olarak kullanılan bir tekniktir ve en genel şekilde en iyi çözüme ulaşma süreci şeklinde açıklanır. En iyi çözüm kavramı, birden fazla çözüm olduğu ve çözümlerin eşit değerde olmadığı anlamına gelir (Haupt ve Haupt, 2004).

Optimizasyon problemlerinde, problemin maksimizasyon ve minimizasyon problemi olmasına bağlı olarak, optimal noktalar olarak bilinen çeşitli minimum veya maksimum çözümlerin bulunması hedeflenir. Minimizasyon problemlerinde optimal minimumlar, maksimizasyon problemlerinde ise optimal maksimumlar bulunmak istenir. Matematiksel olarak optimizasyon bir veya daha fazla değişkene sahip bir fonksiyonun kısıtlayıcı şartlar altında en iyi çözümünü arama süreci olarak tanımlanabilir (Erdoğmuş, 2016). Bir optimizasyon problemi matematiksel olarak aşağıdaki gibi ifade edilebilir.

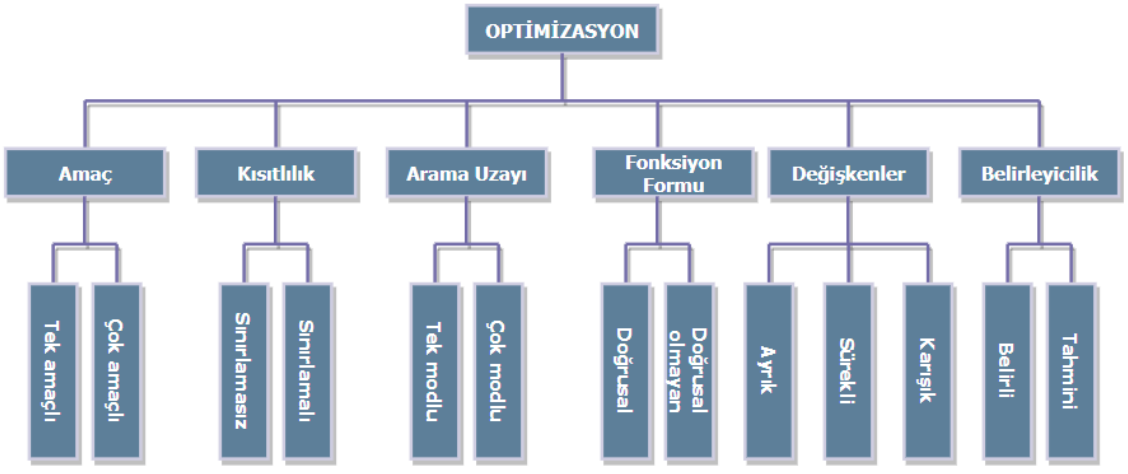
$$minimize\ \boldsymbol{x} \in R^n\ f_i\ (x), \qquad (i = 1,2,\dots,M) \tag{3.1}$$

$$h_j(x) = 0, \qquad (j = 1,2,\dots,J) \tag{3.2}$$

$$g_k(x) \leq 0, \qquad (k = 1,2,\dots,K) \tag{3.3}$$

$$x = (x_1, x_2, \dots, x_n)^T \tag{3.4}$$

Burada x_i'ler karar değişkenleridir. $f_i(x)$ amaç fonksiyonlarını temsil eder. R problemin arama uzayıdır. $h_j(x)$ eşitliği ve $g_k(x)$ eşitsizliği kısıtlamaları gösterir. $g_k(x)$ eşitsizliği " $\geq$ " şeklinde yazılırsa problem maksimizasyon problemi olur (Yang, 2010).



Şekil 3.1. Optimizasyon problemlerinin sınıflandırılması (Yang, 2010)

Şekil 3.1'de görüldüğü gibi optimizasyon problemleri amaçlarına, kısıtlılıklarına, arama uzaylarına, fonksiyon formlarına, değişkenlerine ve belirleyiciliklerine göre çeşitli şekillerde sınıflandırılabilir. Bir optimizasyon problemi bu sınıflandırmaların bir çoğuna aynı anda sahiptir. Denklem 3.1'de $M = 1$ olduğu durumlarda problem tek amaçlı, $M > 1$ olduğunda ise çok amaçlı optimizasyon problemidir. $J = K = 0$ olduğunda kısıtsız optimizasyon problemidir.

Bu çalışmada geliştirilen FOA algoritmasının optimizasyon problemleri üzerindeki performansı değerlendirilirken çeşitli sınıflandırmalara sahip olan farklı kıyaslama fonksiyonları kullanılmıştır. Öznitelik seçimi problemi ayrık bir optimizasyon problemi olarak ele alınmıştır.

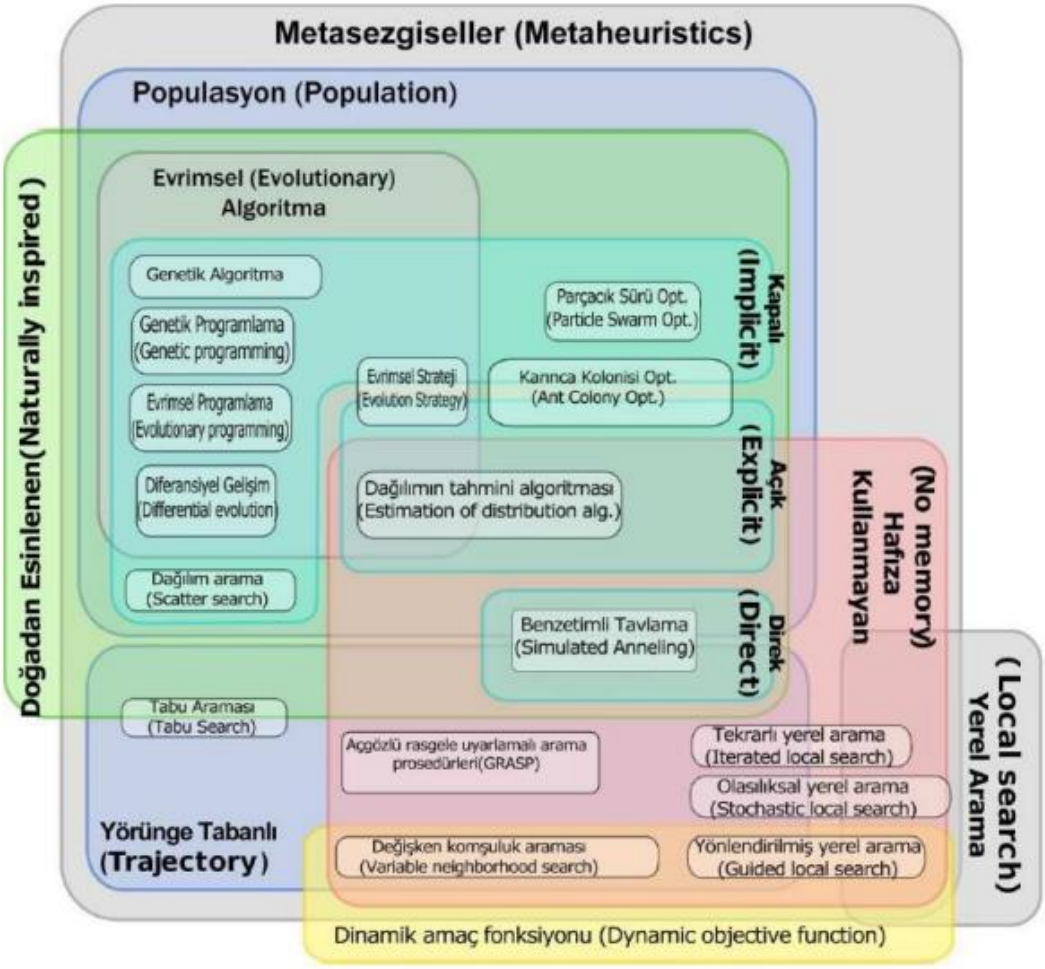
Çalışmanın bu bölümünde optimizasyon problemleri ve meta-sezgisel algoritmalar anlatılmıştır. Ayrıca çalışmada kullanılan FOA algoritması detaylı bir şekilde açıklanmıştır.

3.1. Optimizasyonda Meta-Sezgisel Algoritmalar

Günümüze kadar optimizasyon problemlerinin çözümü için çeşitli optimizasyon yöntemleri tanıtılmıştır. Zamanla problemlerin karmaşıklıkları arttıkça klasik teknikler bu problemleri çözme konusunda çeşitli zorluklarla karşılaşmaya başlamıştır (Luo ve ark., 2020). Meta-sezgisel algoritmalar, optimizasyon problemlerinin optimale yakın çözümünü elde eden optimizasyon yöntemlerindendir (Agrawal ve ark., 2021). Meta-sezgisel algoritmalara olan ilgi; basitlikleri, esneklikleri ve yerel optimallerden kaçma yetenekleri nedeniyle sürekli artış göstermiştir. Esneklik özelliği bu algoritmaların yapılarında herhangi bir değişikliğe ihtiyaç duyulmadan farklı optimizasyon problemlerine uygulanabilirliğini ifade eder (Moayedi ve ark., 2020).

Meta-sezgisel algoritmalar tek çözümlü ve popülasyon tabanlı olmak üzere iki ana kategoriye ayrılabilirler. Tek çözümlü meta-sezgisel algoritmalar, problem çözümüne bir aday çözüm ile başlar ve problem çözümü sürecinde bu çözüm güncellenir. Popülasyon tabanlı algoritmalar ise birçok aday çözümden oluşan bir popülasyon ile problemin çözümüne başlar. Çözüm sürecinde popülasyondaki bu aday çözümler güncellenerek optimal çözüme ulaşmak hedeflenir. Birçok aday çözüm, arama uzayının daha iyi bir şekilde keşfedilmesine ve yerel optimallerden kaçınmaya yardımcı olur. Aynı zamanda popülasyon tabanlı algoritmalar arama uzayının daha iyi bölgelerine doğru sıçrama özelliğine de sahiptirler (Agrawal ve ark., 2021).

Literatürde meta-sezgisel algoritmalar davranışlarına göre farklı kategorilerde sınıflandırılır. Evrimsel, sürü zekası, fizik tabanlı ve insan ile ilişkili teknikler bu kategorilerdendir (Mohamed ve ark., 2020). Evrimsel teknikleri kullanan algoritmalar, biyolojiden ilham alır. Bu kategoride iyi bilinen algoritmalar şunlardır; GA (Holland, 1992), Tabu Arama (TS) (Cvijović ve Klinowski, 1995) ve DE (Price, 2013). Sürü zekası tekniklerini kullanan algoritmalar, doğadaki canlıların sosyal davranışlarından esinlenir. PSO (Kennedy ve Eberhart, 1995), Karınca Kolonisi Optimizasyonu (ACO) (Dorigo ve ark., 1996) ve FOA (Ghaemi ve Feizi-Derakhshi, 2014) bu kategoride yer alan algoritmalardandır. Fizik tabanlı teknikleri barındıran algoritmalar, evrenin fizik kurallarından ilham alır ve SA (Bertsimas ve Tsitsiklis, 1993), HS (Geem ve ark., 2001) bu kategoride yer alır. İnsan ile ilişkili teknikleri temel alan algoritmalar ise insan davranışlarından esinlenilerek geliştirilir. Öğretme Tabanlı Öğrenme Optimizasyon Algoritması (TLBO) bu kategoriye örnek olarak verilebilir (Mohamed ve ark., 2020). Şekil 3.2'de meta-sezgisel algoritmaların sınıflandırılması görülmektedir.



Şekil 3.2. Meta-sezgisel algoritmaların sınıflandırılması (Çelik ve ark., 2019)

Meta-sezgisel algoritmalar stokastik yapılıdır; rastgele aday çözümlerle optimizasyon sürecine başlarlar. Algoritmanın her iterasyonunda arama stratejileri aracılığıyla rastgele üretilen bu aday çözümler uygunluk fonksiyonuna göre iyileştirilir ve sonuç olarak optimale en yakın çözüm elde edilmeye çalışılır. Algoritmaların arama stratejilerinde arama uzayını keşfetmek ve en iyi çözümü kullanmak konuları oldukça önemlidir (Lin ve Gen, 2009). Keşif kavramı, algoritmanın arama uzayındaki umut verici bölgelerini kapsamlı bir şekilde araştırmasını ifade eder. Keşif fazında bulunan umut verici bölgelerin yerel olarak aranması ise kullanım kavramını açıklar (Talbi, 2009). Meta-sezgisellerde başarılı bir performans için arama uzayının keşfi ve kullanımı arasında çok iyi bir denge kurulması gerekir. Bu dengenin kurulamaması durumunda algoritmalar yerel optimallere tuzaklanmaya meyilli hale gelebilirler ve optimal çözüme yakınsayamama gibi problemlerle karşılaşılabilir.

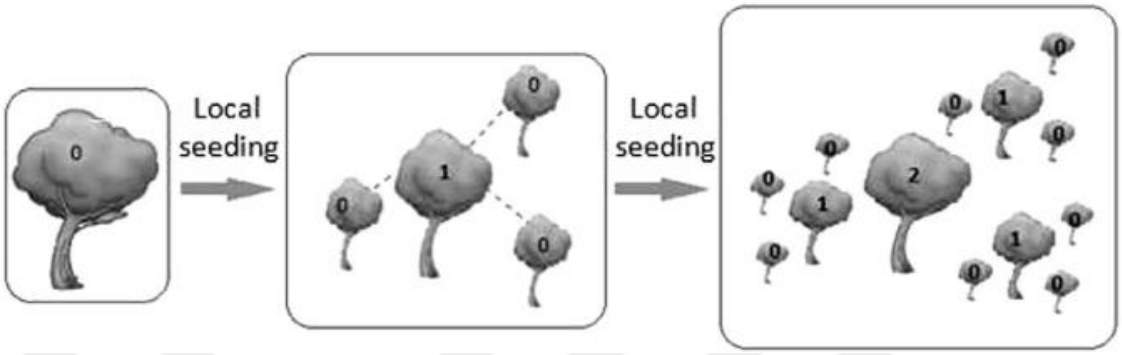
3.2. Orman Optimizasyonu Algoritması (FOA)

FOA algoritması, 2014 yılında Ghaemi ve Feizi-Derakhshi tarafından popülasyon tabanlı meta-sezgisel bir algoritma olarak önerilmiştir (Ghaemi ve Feizi-Derakhshi, 2014). FOA, ağaçların ormanlardaki yaşam süreçlerinden ilham alınarak geliştirilmiştir. Aday çözümler, ağaçlar ile temsil edilir. Ağaçlar, popülasyonu temsil eden ormanı oluşturur. Doğada ağaçların tohumlama süreçleri iki şekilde gerçekleşir; ilki tohumların ağaçların yakınlarına düşerek orada filizlenmesidir. FOA algoritmasında bu süreç yerel tohumlama aşamasını temsil eder ve bu aşama ile yerel arama gerçekleştirilir. Küresel tohumlama aşamasını oluşturan ikinci tohumlama süreci, farklı doğal etkenlerle tohumların ormanın farklı bölgelerine taşınıp bu bölgelerde filizlenmesidir. Bu aşama problem uzayında küresel aramayı gerçekleştiren adımları içerir. Bu iki aşama, bir önceki bölümde açıklanan meta-sezgisel algoritmalarda çok önemli iki konu olan keşif ve kullanım fazlarını karşılar.

3.2.1. FOA algoritması genel prosedürü

FOA algoritmasının genel prosedürü şu şekildedir; çözüme rastgele ilk popülasyon oluşturularak başlanır. Başlangıçta ve tohumlama adımlarında yeni üretilen tüm ağaçlar 0 yaşındadır. FOA algoritmasında ana adımlardan ilki yerel tohumlamadır ve sadece 0 yaşındaki ağaçlar bu aşamaya tabii tutulur. Bu aşamada amaç, popülasyondaki

ağaçların yakın komşularını popülasyona eklemektir. Bu şekilde algoritma popülasyondaki aday çözümleri yerel olarak optimize eder. Bu aşamadan sonra yeni üretilen ağaçlar dışındaki tüm ağaçların yaşları 1 artırılır. Şekil 3.3'te yerel tohumlama örneği verilmiştir.



Şekil 3.3. FOA algoritmasında yerel tohumlama örneği (Ghaemi ve Feizi-Derakhshi, 2014)

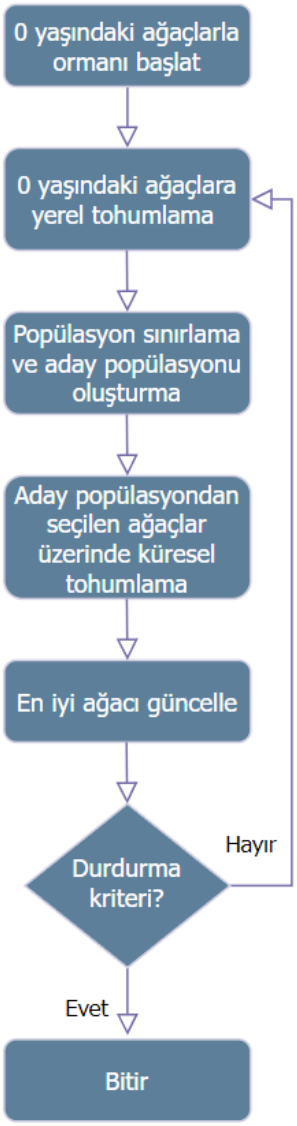
Daha sonra popülasyondaki ağaç sayısının kontrol altında tutulabilmesi için popülasyon sınırlama uygulanır. Bu aşamada ağaçların yaşları ve popülasyon alanı bazında sınırlama yapılarak bazı ağaçlar ormandan çıkarılır ve bu ağaçlar aday popülasyonu oluşturur. Bu aşamanın sonunda popülasyonda başlangıçtaki ağaç sayısı kadar ağaç kalır.

Bir sonraki aşama küresel tohumlama aşamasıdır. Bir önceki aşamada aday popülasyona alınan ağaçların belirli bir yüzdesi üzerinde gerçekleştirilir ve üretilen yeni ağaçlar popülasyona eklenir. Bu şekilde arama uzayında küresel keşif yapılır.

Son olarak popülasyondaki en iyi ağaç belirlenir ve yaşı sıfır olarak ayarlanır. Ağaçlar üzerindeki yaş kontrolünün amacı şu şekilde açıklanabilir; en iyi ağaçların yaşlarının sıfır olarak ayarlanması ile yerel tohumlama aşamasında bu ağaçların komşularını popülasyona eklemek mümkün olur ve en iyi çözümler yerelleştirilir. Aynı zamanda ağacın yaşlanması önlenerek popülasyonda tutulması sağlanır. Aksi durumda, iyi uygunluk değerine sahip olmayan kötü ağaçlar algoritmanın her aşamasında yaşlanır ve nihayetinde yaş sınırını aşarak popülasyondan çıkarılır.

FOA algoritmasının akış şeması Şekil 3.4'te ve sözde kodu Şekil 3.5'te verilmiştir. Algoritmada başlangıçta tanımlanması gereken 5 parametre bulunmaktadır; $life\_time$ parametresi, bir ağacın izin verilen maksimum yaş değeridir. $area\_limit$, popülasyondaki ağaç sayısıdır. $transfer\_rate$, aday popülasyondan seçilecek ağaçların yüzdesini belirtir. LSC, bir ağacın yerel tohumlamada kaç komşusunun ekleneceğini

gösterir. GSC parametresi, küresel tohumlama aşamasında aday popülasyondan ağaçlar seçildikten sonra GSC kadar değişkenlerinin değiştirilip ormana eklenmesini kontrol eder.



Şekil 3.4. FOA algoritması akış şeması (Ghaemi ve Feizi-Derakhshi, 2014)

FOA algoritmasında bir ağaç, değişken değerlerini ve ilgili ağacın yaş bilgisini içerir. Yerel tohumlama aşamasından sonra yeni üretilen ağaçlar dışındaki tüm ağaçların yaş değerleri 1 artırılır. Bir ağaç Denklem 3.5'teki gibi $1 * (N + 1)$ boyutunda bir vektör olarak düşünülebilir. Burada N problem boyutudur, $x_1, x_2, \dots, x_N$ değişken değerlerini temsil eder ve age ilgili ağacın yaşını gösterir (Ghaemi ve Feizi-Derakhshi, 2014).

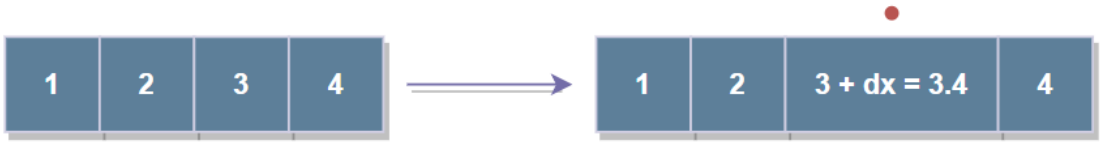
$$T = [x_1, x_2, \dots, x_N, age] \tag{3.5}$$

FOA Algoritması (life_time, LSC, GSC, transfer_rate, area_limit)

```
Rastgele ağaçlarla ormanı başlat (her ağacın yaşı başlangıçta sıfırdır)
while (durdurma kriteri sağlanana kadar)
    Yerel tohumlama (sıfır yaşındaki ağaçlara)
        for i = 1 : "LSC"
            Seçilen ağacın rastgele bir değişkenini seç
            Seçilen değişkenin değeri ile dx'i topla, dx∈[-Δx, Δx]
        end for
        Yeni oluşturulan ağaçlar hariç tüm ağaçların yaşını 1 artır
    Popülasyon sınırlama
        Yaşı "life_time" parametresini geçen ağaçları kaldır ve aday popülasyona al
        Ağaçları uygunluk değerlerine göre sırala
        "area_limit" parametresini aşan ağaçları aday popülasyona ekle
    Küresel tohumlama
        Aday popülasyonun "transfer_rate" yüzdesini seç
        for (seçilen her ağaç için)
            Seçilen ağacın "GSC" kadar değişkenini rastgele seç
            Her değişkenin değerini rastgele üretilmiş değerle değiştir ve
            Ormana 0 yaşında yeni bir ağaç olarak ekle
        end for
    Şimdiye kadarki en iyi ağacı güncelle
        Ağaçları uygunluk değerlerine göre sırala
        En iyi ağacın yaşını 0'a ayarla
end while
En iyi ağacı sonuç olarak döndür
```

Şekil 3.5. Orman optimizasyonu algoritmasının sözde kodu (Ghaemi ve Feizi-Derakhshi, 2014)

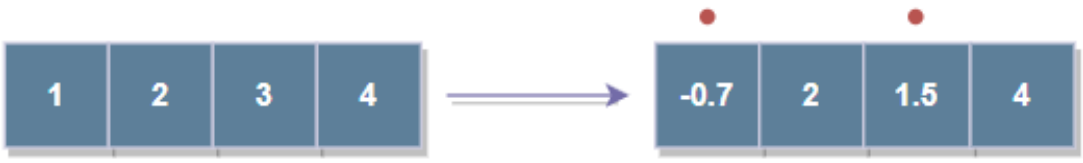
Ağaçların yakın bölgelerinde filizlenen ve komşuları olarak popülasyona eklenen tohumların sayısı LSC parametresi ile kontrol edilir. Yerel tohumlamanın uygulanması için sıfır yaşındaki ağaçların bir değişkeni rastgele seçilir. Seçilen değişkenin değeri rastgele bir değer $dx \in [-\Delta x, \Delta x]$ ile toplanır. Ağaç, güncellemiş değişken değeri ile 0 yaşında yeni bir ağaç olarak ormana eklenir. Bu işlemler LSC parametresi kadar tekrarlanır. Δx, küçük bir değerdir ve ilgili değişkenin üst limitinden küçüktür. Bu şekilde arama işlemi sınırlı bir aralıkta yapılır ve yerel arama gerçekleştirilebilir. Şekil 3.6'da bir yerel tohumlama örneği verilmiştir. Bu örnekte LSC parametresi ve Δx, 1 olarak kabul edilir ve $dx = 0.4 \in [-1,1]$ (Ghaemi ve Feizi-Derakhshi, 2014).



Şekil 3.6. Bir ağaç üzerinde yerel tohumlamanın sayısal bir örneği (Ghaemi ve Feizi-Derakhshi, 2014)

Popülasyon sınırlaması, $life\_time$ ve $area\_limit$ olmak üzere iki parametre üzerinden kontrol edilir. Öncelikle yaş değeri $life\_time$ parametresini aşan ağaçlar popülasyondan çıkarılır ve bu ağaçlarla aday popülasyon oluşturulur. Ardından popülasyonda kalan ağaçlar uygunluk değerlerine göre sıralanır ve $area\_limit$ parametresini aşan ağaçlar ormandan çıkarılır ve aday popülasyona eklenir.

Küresel tohumlamada mevcut popülasyona aday popülasyondan yeni ağaçlar eklenir. Bunun için aday popülasyondan $tranfer\_rate$ oranında seçilen ağaçların GSC kadar değişkeni rastgele bir değer ile değiştirilir. Şekil 3.7'de GSC parametresinin 2 olduğu küresel tohumlamanın sayısal bir örneği verilmiştir.



Şekil 3.7. Bir ağaç üzerinde küresel tohumlamanın sayısal bir örneği (Ghaemi ve Feizi-Derakhshi, 2014)

3.3. Bölüm Değerlendirmesi

Bu bölümde optimizasyon problemlerinin matematiksel tanımı ile sınıflandırılmaları verilmiştir. Meta-sezgisel algoritmaların optimizasyon problemlerinin çözümündeki rolü açıklanmıştır. Bununla birlikte meta-sezgisel algoritmaların sınıflandırmalarına değinilmiştir. Algoritmalar için çok önemli iki kavram olan keşif ve kullanım fazları açıklanmıştır ve son olarak bu çalışmada kullanılan sürekli optimizasyon problemlerinin çözümü için geliştirilen FOA algoritması detaylı bir şekilde anlatılmıştır.

4. KARŞITLIK TABANLI ÖĞRENME (OBL)

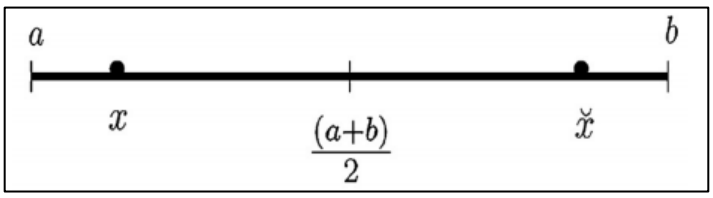
OBL yöntemi, 2005 yılında Tizhoosh tarafından algoritmaların performansını iyileştirmek için önerilen yöntemlerden biridir (Tizhoosh, 2005). Literatürdeki iyileştirme çalışmalarında OBL yöntemi oldukça ilgi görmüştür. Popülasyon çeşitliliğini artırma, yerel optimallerden kaçma ve yakınsama oranını artırma gibi başarılar sağlayabilmektedir. Bunun yanı sıra algoritmaya entegre edilme şekline bağlı olarak hesaplama maliyetini yükseltebilmesi söz konusudur.

Çalışmanın bu bölümünde OBL yönteminde açıklanması gereken tanımlar ve kavramlar üzerinde durulmuştur.

4.1. Karşıt Sayı ve Karşıt Nokta Tanımları

OBL yöntemi mevcut bir aday çözümün problem uzayındaki karşıtını da değerlendirmeye almayı önerir. Bu şekilde arama zıt yönde genişletilir. Böylece elde bulunan aday çözümlerden daha iyi aday çözümler bulma olasılığı artırılır. Şekil 4.1'de tek boyutlu bir uzayda karşıt nokta örneği verilmiştir. $x \in [a, b]$ aralığında tanımlı bir sayının karşıtı $\tilde{x}$, Denklem 4.1'deki hesaplanır (Tizhoosh, 2005).

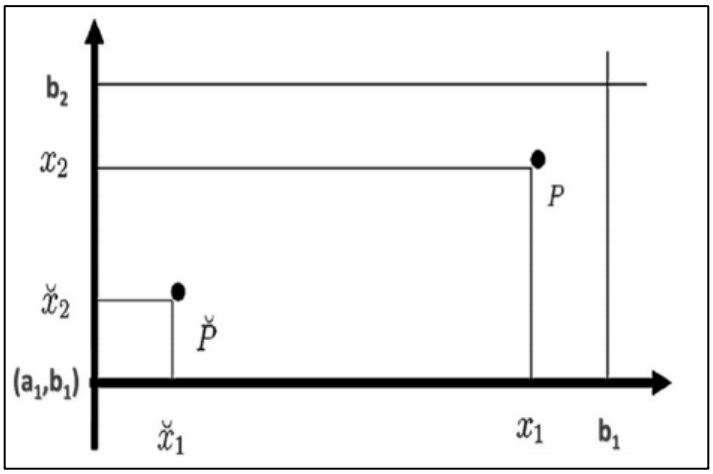
$$\tilde{x} = a + b - x \tag{4.1}$$



Şekil 4.1. Tek boyutlu bir uzayda karşıt nokta (Tizhoosh, 2005)

Çok boyutlu bir uzayda karşıt nokta, karşıtı hesaplanacak noktanın her eksendeki karşıt sayısının hesaplanması ile bulunur. Şekil 4.2'de iki boyutlu bir uzayda karşıt nokta örneği verilmiştir. $x_i \in [a_i, b_i]$ ve $x_1, x_2, \dots, x_n \in R$ olmak üzere $P(x_1, x_2, \dots, x_n)$, n boyutlu bir koordinat sisteminde tanımlı bir nokta olarak kabul edilirse karşıt $\tilde{P}$ noktası $\widetilde{x_1}, \widetilde{x_2}, \dots, \widetilde{x_n}$ karşıt sayıları ile tanımlanır ve Denklem 4.2'deki gibi hesaplanır (Tizhoosh, 2005).

$$\widetilde{x_i} = a + b - x_i, \quad i = 1, 2, \dots, n \tag{4.2}$$



Şekil 4.2. İki boyutlu bir uzayda karşıt nokta (Tizhoosh, 2005)

4.2. Karşıtlık Tabanlı Optimizasyon

Karşıtlık tabanlı optimizasyonda mevcut aday çözüm ve karşıtı aynı anda değerlendirilir ve ikisi arasından uygunluğu daha iyi olan popülasyonda tutulur. $T = (x_1, x_2, \dots, x_N)$, N boyutlu bir problem uzayında tanımlı aday çözüm ve $f(\cdot)$ uygunluk fonksiyonu olarak kabul edilirse karşıtlık tabanlı optimizasyon için öncelikle Denklem 4.2 ile aday çözümün karşıtı hesaplanır. İki aday çözümün $f(\cdot)$ fonksiyonunda uygunluk değerleri hesaplanır. Eğer $f(\tilde{T}) \leq f(T)$ (minimizasyon problemleri için) ise karşıt $\tilde{T}$ aday çözümü, aksi durumda T aday çözümü popülasyonda tutulur (Tizhoosh, 2005).

4.3. Bölüm Değerlendirmesi

Bu bölümde algoritmaların performansını artırabilmek için yapılan modifikasyon çalışmalarında uygulama alanı bulan OBL yöntemi açıklanmıştır. Karşıt sayı ve karşıt nokta tanımları verilmiştir ve karşıtlık tabanlı optimizasyonun nasıl uygulandığı açıklanmıştır. Bu yöntem algoritmalara genellikle popülasyon başlatma aşamasında ve nesil atlama şeklinde uygulanmaktadır. Bunun yanı sıra algoritmaların operatörlerini geliştirmek için kullanıldığı çalışmalar da mevcuttur.

Meta-sezgisellerde rastgelelik söz konusudur. Örnek olarak başlangıç popülasyonu rastgele üretilir. Bu problemin çözümüne sıfırdan başlamak demektir. Rastgele aday çözüm, optimal çözümden çok uzak değilse hızlı bir yakınsama ile sonuçlanabilir. Ancak, mevcut çözümden çok uzak olan rastgele bir aday çözümle başlanırsa, optimizasyon süreci daha uzun sürecektir.

Algoritma rastgele aday çözümlerle başlatılırken probleme en iyi aday çözümlerle başlamak düşük bir ihtimaldir. FOA algoritması da rastgele oluşturulan ağaçlarla başlangıç popülasyonunu başlatır. Ayrıca yerel ve küresel tohumlama aşamalarında da aday çözümlerin güncellenecek değişkenlerinin seçilmesi ve bir değerle değiştirilmesi noktasında da rastgelelik söz konusudur. Bu yöntem, arama uzayında daha iyi bir aday çözüm keşfedebilme olasılığına dayanarak zıt bölgenin de değerlendirilmesini önerir.

5. KARŞITLIK TABANLI ORMAN OPTİMİZASYONU (OFOA)

FOA algoritmasının performansının iyileştirilmesi bu çalışmanın temel amacıdır. Bu amaç doğrultusunda algoritmanın başlangıç popülasyonun başlatılması ve iteratif sürecinde OBL yöntemi kullanılmıştır. Geliştirilen algoritma OFOA olarak adlandırılmıştır. Çalışmanın bu bölümünde önerilen OFOA algoritmasının OBL yöntemine dayalı adımları detaylı bir şekilde anlatılmıştır ve yapılan modifikasyonlarla birlikte algoritmanın genel prosedürü sunulmuştur.

5.1. Karşıtlık Tabanlı Popülasyon Başlatma

OFOA algoritmasında, popülasyon çeşitliliğini artırabilmek ve böylece problemin çözümüne daha iyi aday çözümlerle başlayabilmek amaçlarıyla popülasyon başlatma aşamasında OBL yöntemi kullanılmıştır. OFOA algoritmasında karşıtlık tabanlı popülasyon başlatma sözde kodu Şekil 5.1'de verilmiştir.

OFOA Karşıtlık Tabanlı Popülasyon Başlatma	
1:	Başlangıç popülasyonunu (F) rastgele üretilen ağaçlarla oluştur
2:	Karşıt popülasyonu (OF) Denklem 5.1 ile hesapla
3:	Başlangıç popülasyonu ve karşıt popülasyonu birleştir ($F \cup OF$)
4:	İki popülasyonun birleşiminden "area_limit" kadar en uygun ağaçları seç
5:	Seçilen ağaçları başlangıç popülasyonu olarak ayarla

Şekil 5.1. OFOA algoritmasında karşıtlık tabanlı popülasyon başlatma sözde kodu

OFOA, ilk olarak başlangıç popülasyonunu rastgele üretilen ağaçlarla oluşturur ve bu popülasyonun problem uzayı içerisindeki karşıtını Denklem 5.1 ile hesaplar. Rastgele oluşturulan başlangıç popülasyonu ve karşıt popülasyon birleştirilir. İki popülasyonun birleşiminden en uygun ağaçlar seçilip başlangıç popülasyonu olarak ayarlanır.

$$OF_{i,j} = a_j + b_j - F_{i,j},\ i = 1,2,\dots,area\_limit,\ j = 1,2,\dots,N \tag{5.1}$$

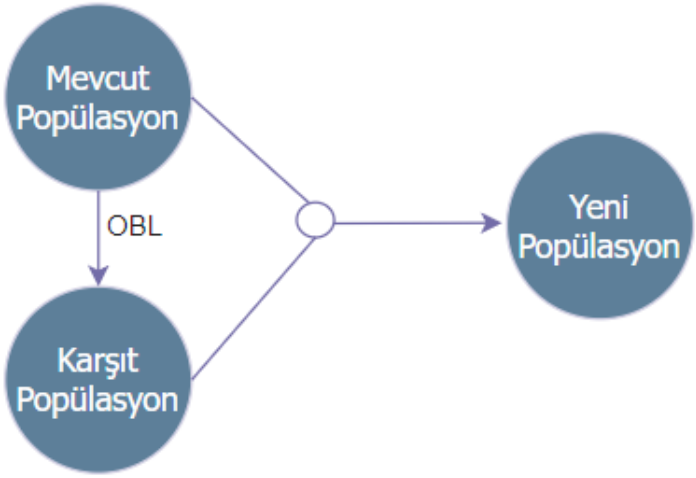
Burada $area\ limit$ popülasyon büyüklüğü yani popülasyondaki ağaç sayısıdır, N problem boyutunu ifade eder. $[a_j, b_j]$, j. değişkenin sınırlarını gösterir. $OF_{i,j}$ ve $F_{i,j}$

sırasıyla karşıt popülasyonun ve başlangıç popülasyonunun i. ağacının j. değişkenini temsil eder.

Rastgele üretilmiş bir ağacın karşıtının da değerlendirilmeye alınmasıyla arama alanı genişletilir ve bununla birlikte daha iyi bir ağacı başlangıç popülasyonuna dahil etme olasılığı ortaya çıkar. Böylelikle popülasyon çeşitliliği artırılmış olur. Daha iyi aday çözümlerden oluşan başlangıç popülasyonları elde edilebilir. Problemin çözümüne daha iyi aday çözümlerle başlamak aynı zamanda optimal çözüme daha hızlı yakınsama ihtimalini artırır.

5.2. Karşıtlık Tabanlı Nesil Atlama

İkinci olarak OFOA algoritmasında bazı adımlarda popülasyon uygunluğunu artırabilmek ve yerel optimal çözümlere takılma eğiliminin önüne geçebilmek için OBL yöntemi algoritmanın iteratif sürecinde kullanılmıştır. Nesil atlama adımlarının her iterasyonda uygulanması hesaplama maliyetini çok fazla artırır. Bu nedenle bir atlama oranı kontrolü eklenir. Şekil 5.3'te OFOA algoritmasında karşıtlık tabanlı nesil atlama sözde kodu verilmiştir. Şekil 5.2 OBL yönteminin uygulanmasının temsili bir diyagramını gösterir.



Şekil 5.2. OBL yönteminin uygulanmasının temsili diyagramı (Liu ve ark., 2019)

OFOA algoritmasına karşıtlık tabanlı nesil atlamanın uygulanması için öncelikle algoritmanın başlangıcında bir atlama oranı J_r tanımlanır. Her iterasyonda [0,1] aralığında üretilen rastgele bir değer, atlama oranı ile karşıtlaştırılır. Üretilen rastgele

değer atlama oranından küçük olduğunda karşıtlık tabanlı nesil atlama gerçekleştirilir. Algoritmanın ilgili iterasyonundaki mevcut popülasyonun karşıtı Denklem 5.2 ile hesaplanır. İki popülasyon birleştirilerek en iyi uygunluk değerine sahip olan ağaçlar mevcut popülasyon olarak ayarlanır.

$$OF_{i,j} = min_j + max_j - F_{i,j},\ i = 1,2,\dots,area\_limit,\ j = 1,2,\dots,N \quad (5.2)$$

Burada min_j ve max_j mevcut popülasyonun j. değişkeninin minimum maksimum değerleridir.

OFOA Karşıtlık Tabanlı Nesil Atlama

```
if rand(0,1) < Jr
        Mevcut popülasyonun (F) karşıtını (OF) Denklem 5.2 ile hesapla
        Mevcut popülasyon ile karşıt popülasyonu birleştir ( F ∪ OF )
        İki popülasyonun birleşiminden en iyi ağaçları seç
        Seçilen ağaçları mevcut popülasyon olarak ayarla
        En iyi ağacı güncelle
end if
```

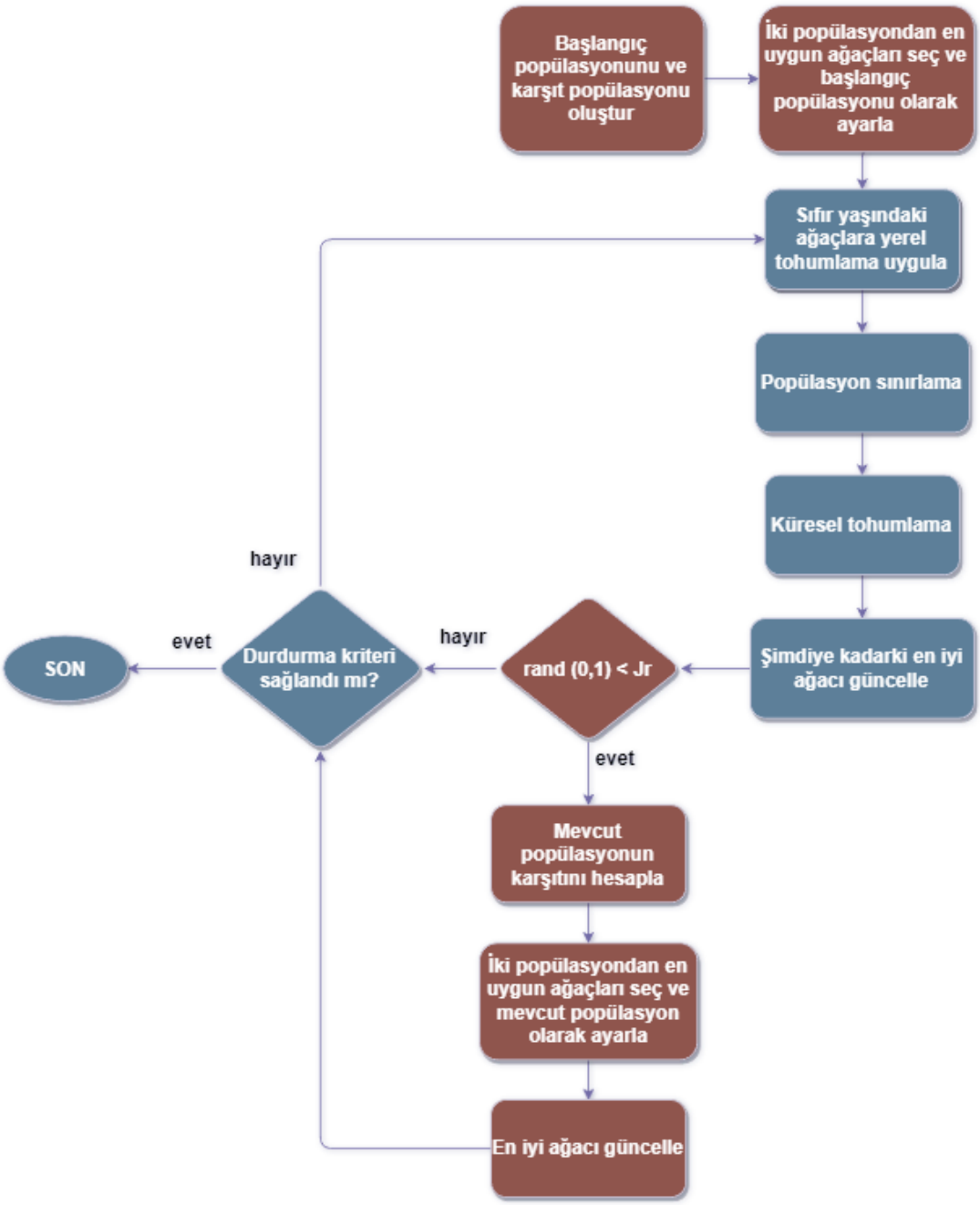
Şekil 5.3. OFOA algoritmasında karşıtlık tabanlı nesil atlama sözde kodu

Yöntemin başlangıç popülasyonunda ve iteratif süreçte uygulanma biçimi arasında alt ve üst sınırlar noktasında bir fark bulunmaktadır. Başlangıç popülasyonunda karşıt aday çözümler hesaplanırken değişkenlerin statik alt ve üst sınırları kullanılır. Ancak nesil atlamada dinamik alt ve üst sınırlar kullanılır. Değişkenlerin statik alt ve üst sınırları kullanılırsa algoritmanın ilgili iterasyona kadar daraltılmış arama uzayının dışına atlanır. Bu yüzden ilgili adımda popülasyondaki mevcut değişkenlerin aralıkları kullanılır. Dinamik sınırların kullanılmasındaki amaç zaten yakınsanmış arama uzayını kaybetmemektir (Rahnamayan ve ark., 2008). Bu şekilde yakınsanmış arama uzayı içerisinde daha iyi bir aday çözüm bulmak amaçlanır.

5.3. OFOA Algoritmasının Genel Prosedürü ve Çalışma Prensipleri

FOA algoritmasının performansını iyileştirmek amacıyla yapılan bu çalışmada, aynı zamanda karşıtlık tabanlı öğrenme yönteminin algoritmanın performansına etkisini net bir şekilde inceleyebilmek için algoritmanın orijinal versiyonunun temel prosedürü değiştirilmemeye çalışılmıştır. Algoritmanın başlangıç popülasyonunda ve iteratif

sürecinde modifikasyonlar söz konusudur. Karşıtlık tabanlı nesil atlama modifikasyonu ile algoritmanın kullandığı parametrelere ek bir Jr parametresi eklenir. OBL yönteminin entegre edildiği adımlar Şekil 5.4' teki akış şemasında kahverengi arka plan rengi ve Şekil 5.5'teki sözde kodda italik yazılarak ile vurgulanmıştır.



Şekil 5.4. OFOA algoritmasının akış şeması

Şekil 5.4'te görüldüğü gibi OFOA algoritması başlangıç popülasyonu ile birlikte karşıt popülasyonun hesaplanmasıyla başlar. İki popülasyondan popülasyon boyutu kadar

en uygun ağaçlar başlangıç popülasyonu olarak ayarlanır. İteratif süreçte nesil atlama aşamasına kadar yerel tohumlama, popülasyon sınırlama, küresel tohumlama ve en iyi ağacı güncelleme olmak üzere FOA algoritmasının temel adımları değişikliğe uğratılmadan olduğu gibi uygulanır. Her iterasyonun sonunda atlama oranı kontrolü ile karşıtlık tabanlı nesil atlama adımları gerçekleştirilir.

OFOA Algoritması (life_time, LSC, GSC, transfer_rate, area_limit, Jr)

```
1:   Karşıtlık tabanlı popülasyon başlatmayı Şekil 5.1'deki sözde koda göre uygula
2:   while (durdurma kriteri sağlanana kadar)
3:       Yerel tohumlama (sıfır yaşındaki ağaçlara)
4:           for i = 1 : "LSC"
5:               Seçilen ağacın rastgele bir değişkenini seç
6:               Seçilen değişkenin değeri ile dx'i topla, dx∈[-Δx, Δx]
7:               Yeni üretilen ağaç olarak (F)'e ekle (0 yaşında)
7:           end for
8:           Yeni oluşturulan ağaçlar hariç tüm ağaçların yaşını 1 artır
9:       Popülasyon sınırlama
10:          Yaşı "life_time" parametresini geçen ağaçları aday popülasyona al
11:          Ağaçları uygunluk değerlerine göre sırala
12:          "area_limit" parametresini aşan ağaçları aday popülasyona ekle
13:      Küresel tohumlama
14:          Aday popülasyonun "transfer_rate" yüzdesini seç
15:          for (seçilen her ağaç için)
16:              Seçilen ağacın "GSC" kadar değişkenini rastgele seç
17:              Seçilen her değişkenin değerini rastgele üretilmiş değerle değiştir
18:              (F)'e 0 yaşında yeni bir ağaç olarak ekle
19:          end for
20:      Şimdiye kadarki en iyi ağacı güncelle
21:          Ağaçları uygunluk değerlerine göre sırala
22:          En iyi ağacın yaşını 0'a ayarla
23:      Karşıtlık tabanlı nesil atlamayı Şekil 5.3'teki sözde koda göre uygula
23:  end while
24:  En iyi ağacı sonuç olarak döndür
```

Şekil 5.5. OFOA algoritmasının sözde kodu

5.4. Bölüm Değerlendirmesi

Bu bölümde, geliştirilen OFOA algoritması detaylı bir şekilde sunulmuştur. Öncelikle algoritmada OBL yöntemine dayanarak yapılan modifikasyonların hangi şekillerde uygulandığı açıklanmıştır. Ardından bu modifikasyonlarla birlikte algoritmanın genel prosedürü verilmiştir.

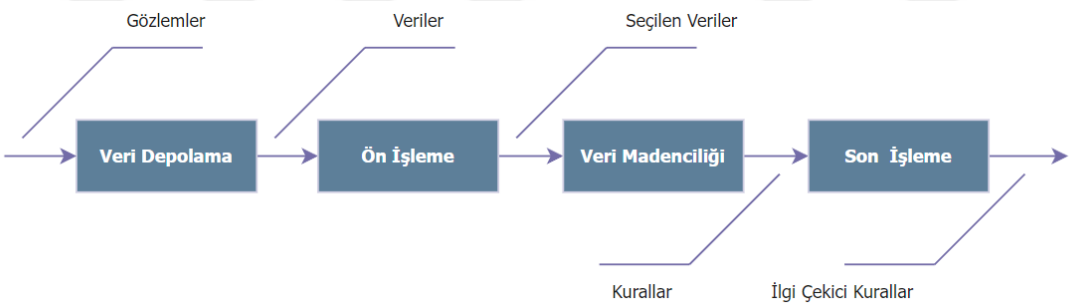
OBL yöntemi, karşıt durum değerlendirmeleri ile algoritmanın arama alanını zıt yönde genişletir. Bu şekilde arama uzayını keşfetme yeteneği artırılır ve mevcut aday

çözümlerden daha iyi aday çözümler keşfetme ihtimali artmış olur. Daha iyi aday çözümler bulunabilirse bu aday çözümler mevcut popülasyona eklenerek popülasyonun uygunluğu artırılır. Başlangıç popülasyonunda uygulanmasıyla algoritma daha iyi aday çözümlerle probleme başlama şansı yakalar. Nesil atlamada ise bazı iterasyonlarda mevcut popülasyondaki aday çözümler iyileştirilir. Bu aşama algoritmanın yerel minimumlara sıkışıp kaldığı durumlardan kurtulması sağlayabilir.

6. VERİ MADENCİLİĞİ VE BİLGİ KEŞFİ

Gerçek dünyada gün geçtikçe biriken çok büyük miktarlarda veri söz konusudur ve bir noktadan sonra bu verilerin analizi bir zorunluluk haline gelmiştir. Çok büyük miktarlardaki veri artışı, yararlı ve organize bilgilerin otomatik olarak anlaşılması veya çıkarılmasına izin vermez. Bu durum giderek daha fazla ön plana çıkan Veri Madenciliği (DM) disiplinin başlamasına yol açmıştır (García ve ark., 2015).

Veri Tabanlarında Bilgi Keşfi (KDD), veri tabanlarından anlamlı bilgilerin çıkarılması ile ilgilenir. KDD sürecindeki adımlar şu şekilde açıklanabilir: a) veri depolama: birden fazla kaynaktan farklı formatlarda toplanan verilerin tek bir uygulama için kullanılmasına olanak tanır, b) hedef veri seçimi: bir veri seti oluşturur, c) temizleme: gürültülü ve/veya aykırı değerleri ortadan kaldırır, d) planlama ve azaltma: verileri uygulamaya göre uygun kullanım için dönüştürür, e) model seçimi: uygun algoritmaları seçer, f) veri madenciliği: ilgi ve kullanışlılık kalıplarını araştırır, g) değerlendirme ve yorumlama: sonuçları doğrular ve açıklar, h) keşfedilen bilgilerin birleştirilmesi: keşfedilen bilgileri kontrol edilebilecek ve yeniden kullanılabilecek bir forma dönüştürür (Liu ve Motoda, 2012).



Şekil 6.1. Veri madenciliği ve bilgi keşfinde genel bir model (Liu ve Motoda, 2012)

Öznitelik seçimi düzeyinde bu adımlar dört temel adımda genelleştirilebilir. Şekil 6.1'de bu adımlar verilmiştir ve aşağıda açıklanmıştır.

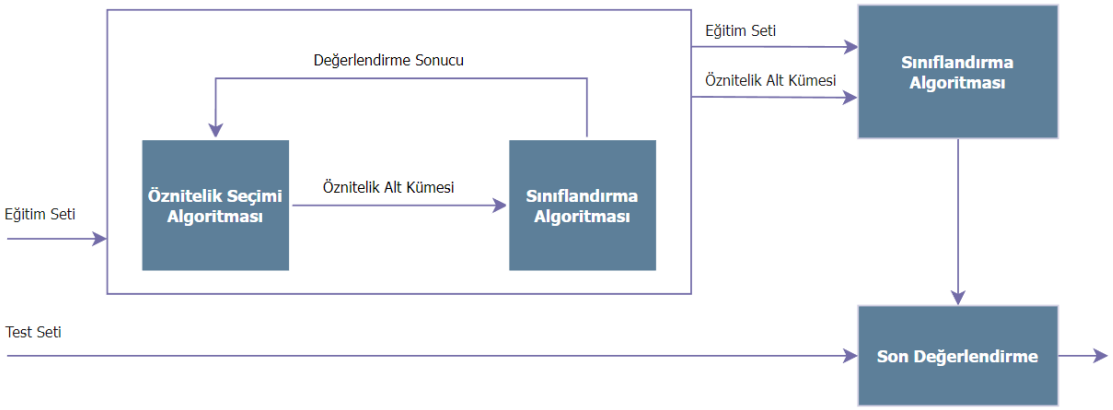
- Veri Depolama: farklı kaynaklardan verilerin toplanması,
- Ön İşleme: verilerin daha iyi kullanımı için temizleme ve indirgeme,
- Veri Madenciliği: verilerden ilgi çekici kalıpların ve bilgilerin çıkarılması,
- Son İşleme: elde edilen kalıpların ve bilgilerin değerlendirilmesi ve yorumlanması, veri özetleme ve görselleştirme.

KDD sürecinde verileri analiz etmek için kullanılan temel ön işleme adımlarından biri öznitelik seçimidir. Bu bölümde öznitelik seçimi problemleri ve bu problemlerin meta-sezgisel algoritmalar ile çözümü anlatılmıştır.

6.1. Öznitelik Seçimi

Veri setlerinden anlamlı bilgiler çıkarılabilmesi için bir veri setindeki özniteliklerin hepsi gerekli olmayabilir. Öznitelik seçimi, veri setindeki gereksiz ve alakasız öznitelikleri belirlemeyi ve bu öznitelikleri veri setinden kaldırarak veri setinin boyutunu azaltmayı amaçlar. Bu nedenle performans doğruluğu korunurken orijinal veri setlerinin boyutunu küçültmek öznitelik seçimi probleminin temel amacıdır (Agrawal ve ark., 2021).

Öznitelik seçme yöntemleri yaygın olarak sarmal ve filtreleme olmak üzere iki gruba ayrılır (Mafarja ve Mirjalili, 2017) (Zhu ve ark., 2007). Filtreleme yöntemlerinde, öznitelik altkümesinin seçilmesi ve sınıflandırma ayrı ayrı gerçekleştirilir. Öznitelikler istatistiksel yöntemlere göre sıralanır ve seçilir. Sınıflandırma algoritmasından bağımsızlık söz konusudur. Bu yöntemler basittir ve hesaplama maliyeti düşüktür (Talbi, 2009) (Tubishat ve ark., 2020).



Şekil 6.2. Sarmal yöntemlerin çalışma şeması

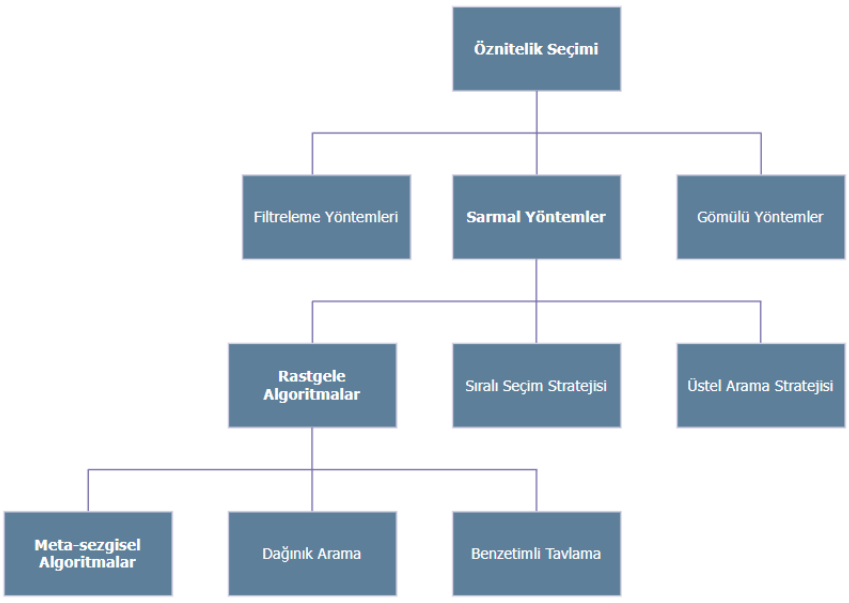
Sarmal yöntemler öznitelik alt kümelerini değerlendirmek için doğrudan sınıflandırma algoritmasını kullanır (Zhu ve ark., 2007). Sınıflandırma algoritması, seçilen öznitelik altkümelerini genellikle eğitim ve test olarak bölünmüş veri setleri üzerinde değerlendirir. En yüksek değerlendirmeye sahip öznitelik altkümesi, sınıflandırma algoritmasının çalıştırılacağı son küme olarak seçilir. Şekil 6.2'de sarmal

yöntemlerin çalışma şeması görülmektedir. Öznitelikler, kullanılan sınıflandırma algoritmasında değerlendirilip optimize edilerek seçildiğinden, sarmal yöntemler daha avantajlıdır. Fakat bununla birlikte hesaplama maliyeti yüksektir (Talbi, 2009).

6.2. Öznitelik Seçimi Problemleri ve Meta-sezgisel Algoritmalar

Orijinal kümeden bir optimal alt küme aranması zorlu bir problemdir. Mühendislik tasarımı, makine tasarımı, makine öğrenmesi, veri madenciliği vb. gibi çeşitli optimizasyon problemlerini çözmede meta-sezgisel algoritmalar çok güvenilirdir (Mafarja ve Mirjalili, 2017). Şekil 6.3'de görüldüğü gibi meta-sezgisel algoritmalar sarmal yöntemler kategorisinde yer alır.

Sarmal tabanlı öznitelik seçiminde meta-sezgisel algoritmalar, Şekil 6.2'de verilen şemada öznitelik seçimi algoritması görevini üstlenirler. Seçilen öznitelik alt kümeleri bir sınıflandırma algoritmasına gönderilir. Burada sınıflandırma algoritması uygunluk fonksiyonu rolünü üstlenmiş olur. Sınıflandırma algoritması veri setini eğitim ve test setlerine bölerek öznitelik alt kümelerini değerlendirir. Algoritma, sınıflandırma algoritmasından gelen değerlendirme sonucu ile öznitelik alt kümelerini optimize ederek en iyi öznitelik alt kümesini bulmaya çalışır. Öznitelik seçiminde uygunluk fonksiyonu olarak sınıflandırma algoritmasının değerlendirme sonucu doğrudan kullanılabildiği gibi birden fazla kriteri aynı anda kontrol eden uygunluk fonksiyonlarının da kullanılması mümkündür.



Şekil 6.3. Öznitelik seçimi yöntemlerinin sınıflandırılması (Agrawal ve ark., 2021)

Bir problemin değişkenlerinin türü, çözüm yöntemi ve optimizasyon algoritmalarının seçiminde belirleyici bir durumdur. Problemler sürekli ve ayrık uzaylarda çözülebilirler. Bu bağlamda, optimizasyon problemleri sürekli ve ayrık olmak üzere iki kategoriye ayrılır. Ayrık optimizasyon problemleri, parametrelerinin sonlu bir tamsayılar kümesine ait olduğu kombinatoryal optimizasyon olarak bilinir (Papadimitriou ve Steiglitz, 1998). İkili optimizasyon, değişkenlerinin yalnızca 0 veya 1 olabildiği ayrık optimizasyonun bir alt kümesidir (Taghian ve ark., 2018).

$S = [x_1, x_2, \dots, x_d]$, d adet öznitelik içeren bir S veri seti olarak kabul edilirse bu veri setinden seçilen öznitelikler Denklem 6.1'deki gibi ifade edilebilir. Burada $x_1, x_2, \dots, x_n$ veri setinden seçilen öznitelikleri temsil eder.

$$D = [x_1, x_2, \dots, x_n], \qquad n < d \tag{6.1}$$

Meta-sezgisellerle öznitelik seçimi problemlerinde aday çözümler, veri setindeki öznitelik sayısı boyutunda ikili vektörler (110011101…) ile temsil edilir. Vektörlerdeki 1'ler ilgili özniteliğin seçildiğini, 0'lar ise seçilmediğini gösterir (Taghian ve Nadimi-Shahraki, 2019).

6.3. İkili Arama Uzayında Meta-sezgiseller

İkili optimizasyon kategorisinde sınıflandırılan öznitelik seçimi probleminin çözümü için literatürde ikili versiyonu geliştirilen birçok meta-sezgisel algoritma yer almaktadır. Bu bölümde çalışmada performans karşılaştırmalarında kullanılan ikili algoritmalar sunulmuştur.

6.3.1. Orman optimizasyonu algoritmasını kullanarak öznitelik seçimi (FSFOA)

Ghaemi ve Feizi-Derakhshi 2016 yılında FOA algoritmasının öznitelik seçimi problemlerine uyarlanmış versiyonunu tanıtmıştır ve FSFOA olarak adlandırmıştır (Ghaemi ve Feizi-Derakhshi, 2016). Bu algoritma öznitelik seçimi problemleri için uyarlandığından ikili bir versiyondur. Algoritmaların ikili versiyonları yalnızca 0 ve 1 değerleri ile çalışır. FSFOA algoritmasının bazı adımlarında yapılan değişiklikler bu bölümde anlatılmıştır.

Başlangıç popülasyonu 0 ve 1 değerleri ile üretilen rastgele ağaçlar ile başlatılır. Bir veri setinin N özniteliği varsa, bir ağacın boyutu $1 * (N + 1)$ olacaktır, burada değişkenlerden biri ilgili ağacın yaşını temsil eder. Bir ağaçtaki her 1, karşılık gelen özniteliğin seçildiğini ve her 0 seçilmediğini gösterir (Ghaemi ve Feizi-Derakhshi, 2016).

FSFOA Algoritması (life_time, LSC, GSC, transfer_rate, area_limit)

```
Rastgele 0/1 ağaçlarla ormanı başlat
Her ağaç (N+1) boyutlu bir T vektörüdür (N tüm özniteliklerin sayısıdır)
Her ağacın yaşı başlangıçta sıfırdır
while (durdurma kriteri sağlanana kadar)
        Sıfır yaşındaki ağaçlara yerel tohumlama uygula
                for i=1: "LSC"
                        Seçilen ağacın bir değişkenini rastgele seç
                        0'dan 1'e ya da tam tersi olarak değiştir
                end for
        Bütün ağaçların yaşını 1 artır
        Popülasyon sınırlama
        Küresel tohumlama
                Aday popülasyondan "transfer rate" yüzdesini seç
                for (seçilen her ağaç)
                        Seçilen ağacın rastgele "GSC" kadar değişkenini seç
                        0'dan 1'e ya da tam tersi olarak değiştir
                 end for
        Şimdiye kadarki en iyi ağacı güncelle
                Ağaçları uygunluk değerlerine göre sırala
                En iyi ağacın yaşını 0 olarak ayarla
end while
Seçilen en iyi öznitelik alt kümesini gösteren en iyi ağacı sonuç olarak döndür
```

Şekil 6.4. FSFOA algoritmasının sözde kodu (Ghaemi ve Feizi-Derakhshi, 2016)

Yerel tohumlama aşamasında yaşı 0 olan her bir ağaç için bazı değişkenler rastgele seçilir. Seçilen değişkenlerin değerleri 0'dan 1'e veya tam tersi olarak değiştirilir. Küresel tohumlama aşamasında yerel tohumlamada olduğu gibi ilgili ağaçların seçilen her bir değişkeni terslenir (Ghaemi ve Feizi-Derakhshi, 2016). Popülasyon sınırlaması ve en iyi ağacı güncelleme aşamalarında herhangi bir uyarlama yapılmamıştır. Şekil 6.4'te FSFOA algoritmasının sözde kodu verilmiştir.

6.3.2. İkili parçacık sürü optimizasyonu (BPSO)

BPSO algoritması, ilk olarak Kennedy ve Eberhart tarafından ikili optimizasyon problemlerini çözmek amacıyla önerilmiştir (Kennedy ve Eberhart, 1997). BPSO

algoritmasında popülasyon, arama uzayında dolaşan N parçacıktan oluşan bir sürüdür. Parçacık aday çözümü temsil eder ve optimal çözümü bulmak amacıyla arama uzayında dolaşır. Her parçacık, kendi deneyim ve bilgisine göre küresel maksimum ya da minimumu bulmaya çalışır (Too ve ark., 2019).

D boyutlu bir problem için, parçacığın hızı $V = (v_{i1}, v_{i2}, \dots, v_{iD})$ ve konumu $X = (x_{i1}, x_{i2}, \dots, x_{iD})$ olarak temsil edilir, burada i, popülasyondaki parçacığın sırasını gösterir. Her parçacığın optimal konumu $Pbest$ ve popülasyondaki küresel optimum $Gbest$ olarak adlandırılır. Her t iterasyonunda, bir parçacık hızını Denklem 6.2'deki gibi günceller.

$$v_i^d(t+1) = w(t) \times v_i^d(t) + c_1 \times r_1 \times \left(Pbest_i^d(t) - x_i^d(t)\right) + c_2 \times r_2 \times (Gbest^d(t) - x_i^d(t)) \quad (6.2)$$

Burada x parçacığın konumu, v parçacığın hızı, i parçacığın popülasyondaki sırası, d arama uzayının boyutu, w eylemsizlik ağırlığı, c_1 ve c_2 ivme katsayılarıdır, r_1 ve r_2 ise 0 ile 1 arasında düzgün dağılmış iki bağımsız rastgele sayıdır. Hız, Denklem 6.3'teki gibi sigmoid fonksiyonu kullanılarak bir olasılık değerine dönüştürülür.

$$S\left(v_i^d(t+1)\right) = \frac{1}{1+e^{-v_i^d(t+1)}} \quad (6.3)$$

Sonrasında parçacığın konumu Denklem 6.4'e göre güncellenir.

$$x_i^d(t+1) = \begin{cases} 1, & If\ \delta < S(v_i^d(t+1)) \\ 0, & Otherwise \end{cases} \quad (6.4)$$

Burada δ, 0 ile 1 arasında düzgün bir şekilde dağıtılmış rastgele bir sayıdır. Algoritmada küresel ve yerel keşif arasında iyi ve istikrarlı bir denge sağlamak için eylemsizlik ağırlığı daha yüksek bir değerden daha düşük bir değere kademeli olarak düşürülür. Her iterasyonda eylemsizlik ağırlığı Denklem 6.5 ile hesaplanır.

$$w(t) = w_{max} - (w_{max} - w_{min})\frac{t}{T} \quad (6.5)$$

Burada w_{max} ve w_{min} eylemsizlik ağırlığının sınırlarıdır, t geçerli iterasyondur ve T maksimum iterasyon sayısıdır.

6.3.3. İkili diferansiyel evrim (BDE)

DE algoritması, Storn ve Price tarafından önerilen evrimsel bir meta-sezgisel algoritmadır (Storn ve Price, 1997). Başlangıçta DE algoritması sürekli optimizasyon için tasarlanmıştır ve 2016 yılında Zorarpacı ve Özel tarafından (Zorarpacı ve Özel, 2016) öznitelik seçimi için DE algoritması, BDE algoritması olarak uyarlanmıştır (Too ve ark., 2019). BDE algoritması basit, doğrudan kullanımlı ve verimli bir öznitelik seçme yöntemidir. Mutasyon, çaprazlama ve seçim olmak üzere üç ana operatörden oluşur.

İlk olarak BDE algoritması, D boyutlu bir problem için rastgele bir başlangıç popülasyonu üretir; burada D, optimize edilmesi gereken öznitelik sayısıdır. Mutasyon aşamasında, x_i vektörü için popülasyondan rastgele x_{r1}, x_{r2}, x_{r3} vektörleri seçilir ve $r_1 \neq r_2 \neq r_3 \neq i$'dir. Daha sonra fark vektörü Denklem 6.6 ile hesaplanır.

$$difference\ vector_i^d = \begin{cases} 0, & If\ x_{r1}^d = x_{r2}^d \\ x_{r1}^d, & Otherwise \end{cases} \tag{6.6}$$

Burada i vektörün popülasyondaki sırası ve d vektörün boyutudur. x_{r1}'in d'inci boyutu x_{r2}'ye eşitse, fark vektörü 0 olur. Aksi takdirde x_{r1} ile aynı olacaktır. Daha sonra, Denklem 6.7'ye göre mutasyon gerçekleştirilir.

$$mutant\ vector_i^d = \begin{cases} 1, & If\ difference\ vector_i^d = 1 \\ x_{r3}^d, & Otherwise \end{cases} \tag{6.7}$$

Mutasyondan sonra çaprazlama işlemi Denklem 6.8'deki gibi yürütülür.

$$u_i^d = \begin{cases} mutant\ vector_i^d, & If\ \delta \leq CR(t) || d = d_{rand} \\ x_i^d, & Otherwise \end{cases} \tag{6.8}$$

Burada u deneme vektörüdür, x vektördür, d arama uzayının boyutudur, $CR \in (0,1)$ geçiş oranıdır, d_{rand} 1 ile D arasında dağıtılan bir rastgele öznitelik indeksidir ve δ 0 ve 1 arasında rastgele bir sayıdır.

Seçim işlemi için, deneme vektörünün uygunluk değeri daha iyi ise mevcut vektör değiştirilecektir. Aksi takdirde, mevcut vektör bir sonraki nesil için korunur.

6.4. İkili Karşıtlık Tabanlı Orman Optimizasyonu (B-OFOA)

Bu çalışmada öznitelik seçimi problemlerinde performans analizi yapmak için OFOA algoritmasının ikili versiyonu hazırlanmıştır. OFOA algoritması bu çalışma prensibini karşılayacak şekilde uyarlanmıştır ve B-OFOA olarak isimlendirilmiştir.

6.4.1. B-OFOA algoritmasının genel prosedürü

B-OFOA algoritması, OFOA algoritmasının tüm prosedürlerini aynen uygular. OFOA algoritmasının ikili arama uzayına uyarlanabilmesi için kullanılan yöntemin tanımlarında ve algoritmanın tohumlama adımlarında bazı değişiklikler söz konusudur.

İkili arama uzayında çözülen problemlerde aday çözümlerin değişkenleri sadece 0 ve 1 değerlerini alabilir. Karşıt noktaların hesaplanmasında kullanılan değişkenlerin alt ve üst sınırları da sırasıyla 0 ve 1 olacaktır. Bu durumda ikili bir uzayda tanımlı x'in karşıtı Denklem 6.9'daki gibi hesaplanır. Denklem 6.10 ikili arama uzayında bir popülasyonun karşıtının hesaplanmasını verir (Tizhoosh, 2005).

$$\tilde{x} = 1 - x \tag{6.9}$$

$$OF_{i,j} = 1 - F_{i,j},\ i = 1,2,\dots,area\_limit,\ j = 1,2,\dots,N \tag{6.10}$$

Burada $area\_limit$ popülasyon büyüklüğüdür, N problem boyutunu ifade eder. $OF_{i,j}$ ve $F_{i,j}$ sırasıyla karşıt popülasyonun ve başlangıç popülasyonunun i. ağacının j. değişkenini temsil eder.

OFOA algoritmasının tohumlama aşamalarında ağaçların değişken değerleri rastgele üretilen bir sayı ile değiştirilmektedir. İkili arama uzayında değişkenler sadece 0 ve 1 değerlerini alabildikleri için B-OFOA algoritmasında ilgili değişkenler tohumlama aşamalarında terslenirler. Değişkenin değeri 0 ise 1 olarak ya da tam tersi şekilde ayarlanır. Şekil 6.5'te verilen B-OFOA algoritmasının sözde kodunda sözü geçen modifikasyonlar görülmektedir. Değişiklik yapılan adımlar italik yazılarak vurgulanmıştır.

B-OFOA Algoritması (life_time, LSC, GSC, transfer_rate, area_limit, Jr)

```
1:  0 ve 1 değerleriyle rastgele üretilen ağaçlarla başlangıç popülasyonunu başlat (F) (her ağaç 0 yaşında)
2:  Karşıt popülasyonu (OF) Denklem 6.10 ile hesapla
3:  Başlangıç popülasyonu ve karşıt popülasyonu birleştir ( F ∪ OF )
4:  İki popülasyonun birleşiminden "area_limit" kadar en uygun ağaçları seç
5:  Seçilen ağaçları başlangıç popülasyonu olarak ayarla
6:  while (durdurma kriteri sağlanana kadar)
7:      Yerel tohumlama (sıfır yaşındaki ağaçlara)
8:          for i = 1 : "LSC"
9:              Ağacın bir değişkenini rastgele seç
10:             Değişkenin değerini tersle (0 → 1 , 1 → 0)
11:             Yeni üretilen ağaç olarak (F)'e ekle (0 yaşında)
12:         end for
13:         Yeni oluşturulan ağaçlar hariç tüm ağaçların yaşını 1 artır
14:     Popülasyon sınırlama
15:     Küresel tohumlama
16:         Aday popülasyonun "transfer_rate" yüzdesini seç
17:         for (seçilen her ağaç için)
18:             Seçilen ağacın "GSC" kadar değişkenini rastgele seç
19:             Seçilen her değişkenin değerini tersle (0 → 1 , 1 → 0)
20:             (F)'e 0 yaşında yeni bir ağaç olarak ekle
21:         end for
22:     Şimdiye kadarki en iyi ağacı güncelle
23:     if rand(0,1) < Jr
24:         Mevcut popülasyonun (F) karşıtını (OF) Denklem 6.10 ile hesapla
25:         Mevcut popülasyon ile karşıt popülasyonu birleştir ( F ∪ OF )
26:         İki popülasyonun birleşiminden en iyi ağaçları seç
27:         Seçilen ağaçları mevcut popülasyon olarak ayarla
28:         En iyi ağacı güncelle
29:     end if
30: end while
31: En iyi ağacı sonuç olarak döndür
```

Şekil 6.5. BOFOA algoritmasının sözde kodu

6.5. Bölüm Değerlendirmesi

Bu bölümde öncelikle DM ve KDD sürecinde önemli ön işleme adımlarından biri olan öznitelik seçimi açıklanmıştır. Ardından öznitelik seçimi yöntemlerine ve öznitelik seçimi problemlerinin meta-sezgisel algoritmalar ile çözümüne yer verilmiştir.

Bu çalışmada öznitelik seçimi problemlerinde performans kıyaslamasında kullanılan FSFOA, BPSO ve BDE algoritmaları genel olarak açıklanmıştır. Son olarak OFOA algoritmasının öznitelik seçimi için uyarlanmış versiyonu B-OFOA algoritması sunulmuştur.

Öznitelik seçimi ikili optimizasyon problemi olarak bilinir ve bu problemlerinin çözümünde algoritmaların ikili versiyonları kullanılmaktadır. FSFOA bu amaç için uyarlanan algoritmalardan biridir ve öznitelik seçiminde başarılı sonuçlar elde

edebilmektedir. Çalışmada OBL yöntemini kullanan OFOA algoritmasının öznitelik seçimi problemlerini çözebilecek şekilde uyarlanmıştır ve algoritmanın öznitelik seçimindeki performansı incelenmiştir.

7. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu çalışmada OBL yöntemi FOA algoritmasına popülasyon başlatma aşamasında ve iteratif süreçte uygulanmıştır. Uygulanan yöntemin algoritmanın performansına etkisi analiz edilmiştir. Bunun için öncelikle algoritmanın performansı çeşitli kıyaslama fonksiyonları üzerinde, ikinci olarak çeşitli veri setleri üzerinde öznitelik seçimi problemlerinde incelenmiştir.

Bu çalışmadaki tüm deneysel testler 2.40 GHz 4 çekirdek 10. nesil Intel Core i5 işlemciye sahip, Windows 10 işletim sisteminde Matlab' da geliştirilmiştir. Bu bölümde ilk olarak deneysel testlerde kullanılan veri setleri ve kıyaslama fonksiyonları sunulmuştur. Gerçekleştirilen deneylerin ortamı, sonuçları ve analizlerine yer verilmiştir.

7.1. Optimizasyon Sonuçları

Geliştirilen algoritmanın performansının değerlendirilmesi için literatürden seçilen çeşitli kıyaslama fonksiyonları kullanılmıştır. Elde edilen sonuçlar algoritmanın orijinal versiyonu ile karşılaştırılmıştır. Ayrıca her iki algoritma üzerinde popülasyon büyüklüğü ve problem boyutunun etkisinin analiz edilebilmesi için de deneysel testler yapılmıştır.

Bu bölümde FOA ve OFOA algoritmalarının kıyaslama fonksiyonları üzerindeki performanslarının karşılaştırmalı sonuçları sunulmuştur. Sunulan tüm sonuçlar, 50 farklı çalıştırmadan elde edilen değerlerdir. Her farklı çalıştırmada algoritmalar aynı başlangıç popülasyonu ile başlatılmıştır. Öncelikle kullanılan kıyaslama fonksiyonları, algoritmaların parametre ayarları ve performansların değerlendirilmesinde kullanılan ölçütler açıklanmıştır.

7.1.1. Kıyaslama fonksiyonları

Algoritmaları doğrulamak ve karşılaştırmak için literatürde birçok kıyaslama fonksiyonu bulunmaktadır. Fakat standart bir fonksiyon seti bulunmamaktadır. Algoritmaların doğru bir şekilde değerlendirilebilmeleri için farklı kategorilerde yer alan fonksiyonların kullanılması önerilir. Bu çalışmada ölçeklenebilir ve ölçeklenemeyen olarak iki kategoriye ayrılmak üzere toplam 15 kıyaslama fonksiyonu kullanılmıştır.

F1, F3, F5-F8 fonksiyonları ölçeklenemeyen fonksiyonlardır. F9-F15 fonksiyonları ölçeklenebilir fonksiyonlardır. F2 ve F4 fonksiyonları ölçeklenebilir fonksiyonlardır ancak bu çalışmada sırasıyla 2 ve 4 boyuta sabitlenerek kullanılmıştır.

Bu fonksiyonlar aynı zamanda tek modlu ve çok modlu olarak da sınıflandırılır. F3-F6, F10, F11 ve F15 fonksiyonları tek modludur. Diğer fonksiyonlar çok modlu fonksiyonlardır. Tüm kıyaslama fonksiyonlarının formülleri Çizelge 7.1'de verilmiştir. Çizelgede "Mod" sütunu fonksiyonun çok modlu (M) ve tek modlu (U) olduğunu gösterir.

Çizelge 7.1. Deneysel testlerde kullanılan kıyaslama fonksiyonları

Fonksiyon	Formül	Mod	Boyut
F1 (SHUBERT)	$f(x) = \prod_{i=1}^{d} \left(\sum_{j=1}^{5} \cos\left((j+1)x_i + j\right) \right)$	**M**	**2**
F2 (LANGERMANN)	$f(x) = \sum_{i=1}^{m} c_i \, exp\left(-\frac{1}{\pi} \sum_{j=1}^{d} (x_j - A_{ij})^2 \right) cos\left(\pi \sum_{j=1}^{d} (x_j - A_{ij})^2 \right)$	**M**	**2**
F3 (BEALE)	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$	**U**	**2**
F4 (POWER SUM)	$f(x) = \sum_{i=1}^{d} \left[\left(\sum_{j=1}^{d} x_j^i \right) - b_i \right]^2$	**U**	**4**
F5 (COLVILLE)	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	**U**	**4**
F6 (BOOTH)	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	**U**	**2**
F7 (SHEKEL)	$f(x) = -\sum_{i=1}^{m} \left(\sum_{j=1}^{4} \left(x_j - C_{ji}\right)^2 + \beta_i \right)^{-1}$	**M**	**4**
F8 (LEVY N.13)	$f(x) = sin^2(3\pi x_1) + (x_1 - 1)^2[1 + sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + sin^2(2\pi x_2)]$	**M**	**2**
F9 (STYBLINSKI-TANG)	$f(x) = \frac{1}{2} \sum_{i=1}^{d} (x_i^4 - 16x_i^2 + 5x_i)$	**M**	*
F10 (DIXON-PRICE)	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$	**U**	*
F11 (ZAKHAROV)	$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5 i x_i \right)^2 + \left(\sum_{i=1}^{d} 0.5 i x_i \right)^4$	**U**	*
F12 (GRIEWANK)	$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	**M**	*
F13 (LEVY)	$f(x) = sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + sin^2(2\pi w_d)]$	**M**	*
F14 (RASTRIGIN)	$f(x) = -a \, exp\left(-b \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos(c x_i) \right) + a + \exp(1)$	**M**	*
F15 (SPHERE)	$f(x) = \sum_{i=1}^{d} x_i^2$	**U**	*

7.1.2. Parametre ayarları

Bu çalışmada yapılan deneysel testlerde kullanılan parametre değerleri Çizelge 7.2'de sunulmuştur. Bütün testlerde iterasyon sayısı 2000'e sabitlenmiştir. Her çalıştırmada algoritmalar aynı başlangıç popülasyonu ile başlatılmıştır. OBL yönteminin algoritmanın performansına etkisini net bir şekilde inceleyebilmek için iki algoritmanın ortak parametreleri eşit ayarlanmıştır.

Çizelge 7.2. Optimizasyon deneysel testlerinde algoritmaların parametre değerleri

Algoritma	Parametreler	Değerler
FOA (Orman Optimizasyonu Algoritması)	Popülasyon büyüklüğü, $area\_limit$	30
	Yaşam süresi, $life\_time$	6
	Transfer oranı, $transfer\_rate$	10
	Küresel tohumlama, GSC	Problem boyutu/2
	Yerel tohumlama, LSC	Problem boyutu/5
OFOA (Karşıtlık Tabanlı Orman Optimizasyonu Algoritması)	Popülasyon büyüklüğü, $area\_limit$	30
	Yaşam süresi, $life\_time$	6
	Transfer oranı, $transfer\_rate$	10
	Küresel tohumlama, GSC	Problem boyutu/2
	Yerel tohumlama, LSC	Problem boyutu/5
	Atlama oranı, Jr	0.3

7.1.3. Değerlendirme ölçütleri

Bu çalışmada algoritmaların kıyaslama fonksiyonları üzerindeki performanslarını değerlendirmek için ortalama uygunluk değeri, en iyi uygunluk değeri, en kötü uygunluk değeri, standart sapma ve doğruluk oranı kullanılmıştır. Kullanılan değerlendirme ölçütleri matematiksel tanımları ile bu bölümde sunulmuştur.

Ortalama ölçütü, farklı çalıştırmalar için algoritmanın elde ettiği en iyi uygunluk değerlerinin ortalamasını gösterir ve Denklem 7.1 ile hesaplanır. Burada N_r farklı çalıştırmaların sayısıdır ve bu çalışmada 50 olarak ayarlıdır. F_i^* elde edilen en iyi uygunluk değeridir.

$$AVG_F = \frac{1}{N_r}\sum_{i=1}^{N_r} F_i^* \tag{7.1}$$

En kötü ölçütü, algoritmanın farklı çalıştırmalarda elde ettiği en kötü uygunluk değeridir ve Denklem 7.2'deki gibi hesaplanır.

$$WORST_F = \min_{1 \le i \le N_r} F_i^* \tag{7.2}$$

En iyi ölçütü, algoritmanın farklı çalıştırmalarda elde ettiği en iyi uygunluk değeridir ve Denklem 7.3 ile hesaplanır.

$$BEST_F = \max_{1 \le i \le N_r} F_i^* \tag{7.3}$$

Standart sapma, algoritmanın farklı çalıştırmalarda aynı en iyi uygunluk değerine ulaşıp ulaşamayacağını test etmek ve bir algoritmanın sonuçlarının tekrarlanabilirliğini test etmek için kullanılır ve Denklem 7.4'teki gibi hesaplanır.

$$STD_F = \sqrt{\frac{1}{N_r - 1} \sum_{i=1}^{N_r} (F_i - ortalama)^2} \tag{7.4}$$

Başarı oranı, algoritmanın gerekli ulaşılacak değere (VTR) ulaşma sayısının oranını hesaplayarak bir algoritmanın güvenilirliğini kontrol eder ve Denklem 7.5 kullanılarak hesaplanır. Bu çalışmada VTR tüm fonksiyonlar için 0,01 olarak ayarlanmıştır.

$$SR = \left(\frac{VTR' ye\ ulaşılan\ çalıştırma\ sayısı}{N_r} \right) \times 100 \tag{7.5}$$

7.1.4. Sabit fonksiyonlar üzerinde elde edilen sonuçlar

Bu çalışmada öncelikle algoritmanın performansı F1-F8 fonksiyonları üzerinde değerlendirilmiştir. Elde edilen sonuçlar FOA algoritması ile ortalama, en iyi, en kötü uygunluk değerleri ve standart sapmalar açısından karşılaştırılmıştır ve karşılaştırmalı sonuçlar Çizelge 7.3'te sunulmuştur.

Ortalama uygunluk değerleri ve standart sapma değerlerine göre OFOA algoritması FOA algoritmasına kıyasla tüm fonksiyonlarda daha iyi sonuçlar elde etmiştir. OFOA algoritmasının 6 fonksiyonda en iyi uygunluk değeri ve 7 fonksiyonda en kötü uygunluk değeri daha iyidir.

Çizelge 7.3. Sabit fonksiyonlar üzerinde FOA ve OFOA algoritmalarının uygunluk değerleri açısından karşılaştırmalı sonuçları

	FOA				OFOA			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F1	-1,86714E+02	2,12E-02	-1,86731E+02	-1,86656E+02	-1,86723E+02	1,49E-02	-1,86731E+02	-1,86631E+02
F2	-4,06178E+00	3,86E-01	-4,15580E+00	-2,19346E+00	-4,15206E+00	8,61E-03	-4,15581E+00	-4,12735E+00
F3	1,53170E-02	1,08E-01	2,31777E-07	7,62202E-01	3,23163E-05	4,45E-05	6,08448E-08	1,75478E-04
F4	2,78353E-02	3,32E-02	5,36345E-04	1,93218E-01	1,63189E-02	1,52E-02	6,03720E-04	8,44528E-02
F5	6,99100E-01	6,13E-01	6,07213E-03	2,41040E+00	3,19193E-01	3,46E-01	1,48369E-03	1,61889E+00
F6	1,54675E-04	1,77E-04	2,36496E-07	6,08024E-04	7,36495E-05	8,40E-05	3,85842E-08	4,09978E-04
F7	-8,78273E+00	2,82E+00	-1,05352E+01	-2,78975E+00	-1,05300E+01	6,37E-03	-1,05358E+01	-1,05051E+01
F8	1,66798E-04	2,64E-04	1,30413E-07	1,41145E-03	9,48802E-05	1,66E-04	4,16143E-08	1,02387E-03

Bu deneysel sonuçlar OBL yöntemini kullanan OFOA algoritmasının FOA algoritmasına kıyasla arama uzayını daha iyi keşfedebildiğini gösterir. Ayrıca entegre edilen yöntem algoritmanın yerel optimallere tuzaklanma eğilimini azaltabilmektedir. Bununla birlikte daha iyi ortalama uygunluk değerleri ile birlikte standart sapma değerlerinin tüm fonksiyonlarda OFOA algoritmasında daha iyi olmasından OFOA algoritmasının birbirine daha yakın sonuçlar elde ederek daha kararlı bir performans gösterdiği söylenebilir.

7.1.5. Popülasyon büyüklüğünün etkisi

Bu bölümde popülasyon büyüklüğünün FOA ve OFOA algoritmalarının performansını nasıl etkilediği incelemiştir. Bu deneysel test, tüm fonksiyonlar üzerinde gerçekleştirilmiştir. Algoritmaların ortak parametreleri eşit ve Çizelge 7.2'teki gibi ayarlanmıştır. Bu parametre değerleriyle algoritmalar 20, 30 ve 50 popülasyon büyüklükleriyle çalıştırılmıştır. Ölçeklenebilir fonksiyonların boyutu 5 olarak sabitlenmiştir. Çizelge 7.4'te ortalama uygunluk değerleri açısından karşılaştırmalı sonuçları sunulmuştur. OFOA algoritması tüm popülasyon büyüklüklerinde FOA algoritmasına kıyasla daha iyi performans göstermiştir.

Karşıt aday çözümlerin de değerlendirilmesi ile OFOA algoritması daha fazla aday çözümü değerlendirir ve bu şekilde popülasyon çeşitliliği artırılmış olur. Böylece OFOA algoritması düşük popülasyon büyüklüklerinde de daha iyi sonuçlar verebilmiştir. Karşıtlık tabanlı popülasyon başlatma ile OFOA algoritmasının, problem çözümüne daha iyi aday çözümler ile başlama ihtimali yüksektir. Ayrıca algoritmanın iteratif sürecinde OBL yönteminin kullanılmasıyla bazı adımlarda, mevcut popülasyonun uygunluğu

iyileştirilebilmektedir. Popülasyon büyüklüğü 50 olarak ayarladığında iki algoritmanın performansları birbirine yaklaşmıştır.

Çizelge 7.4. Popülasyon büyüklüğünün etkisi deneyinin karşılaştırmalı sonuçları

	$area\_limit = 20$		$areaLimit = 30$		$area\_limit = 50$	
	FOA	**OFOA**	**FOA**	**OFOA**	**FOA**	**OFOA**
F1	-1,86713E+02	-1,86718E+02	-1,86711E+02	-1,86716E+02	-1,86716E+02	-1,86722E+02
F2	-3,88904E+00	-4,15411E+00	-4,13749E+00	-4,15164E+00	-4,14183E+00	-4,14852E+00
F3	9,15821E-02	1,52804E-02	1,23755E-04	4,62403E-05	6,83438E-05	3,17694E-05
F4	3,66378E-02	1,46309E-02	2,98364E-02	1,55219E-02	1,82010E-02	1,82730E-02
F5	5,25166E-01	3,96761E-01	4,03678E-01	3,32967E-01	4,70570E-01	3,66204E-01
F6	2,13414E-04	8,98761E-05	2,01764E-04	7,13101E-05	1,20139E-04	7,43701E-05
F7	-7,91647E+00	-1,00514E+01	-8,83738E+00	-1,05262E+01	-1,00848E+01	-1,03101E+01
F8	4,45150E-04	7,55575E-05	1,54265E-04	9,31415E-05	1,39662E-04	1,34360E-04
F9	-1,75473E+02	-1,78583E+02	-1,74625E+02	-1,91835E+02	-1,95830E+02	-1,95830E+02
F10	9,51206E-04	1,16669E-04	6,47801E-04	1,14909E-04	3,14183E-04	6,58595E-05
F11	6,80868E-03	5,39555E-03	5,53161E-03	4,87737E-03	5,14378E-03	5,52172E-03
F12	5,48445E-02	5,69441E-02	5,14546E-02	5,65384E-02	6,24750E-02	5,55420E-02
F13	1,05292E-04	5,64964E-02	1,07800E-04	8,42018E-05	8,53456E-05	8,89380E-05
F14	7,16579E-03	8,18712E-03	8,37985E-03	9,16335E-03	8,55272E-03	7,78608E-03
F15	5,15668E-05	4,90888E-05	3,83170E-02	4,51735E-05	4,20982E-05	3,81267E-05

20, 30 ve 50 popülasyon büyüklüklerinin detaylı sonuçları sırasıyla Çizelge 7.5, Çizelge 7.6 ve Çizelge 7.7'de verilmiştir.

Çizelge 7.5. 20 popülasyonda büyüklüğünde deneysel testlerden elde edilen detaylı sonuçlar

	FOA				**OFOA**			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F1	-1,86713E+02	2,61E-02	-1,86731E+02	-1,86586E+02	-1,86718E+02	2,11E-02	-1,86731E+02	-1,86613E+02
F2	-3,88904E+00	6,43E-01	-4,15581E+00	-2,19334E+00	-4,15411E+00	5,71E-03	-4,15581E+00	-4,12575E+00
F3	9,15821E-02	2,50E-01	5,16751E-08	7,62538E-01	1,52804E-02	1,08E-01	1,86024E-07	7,62104E-01
F4	3,66378E-02	4,42E-02	3,39825E-04	2,31551E-01	1,46309E-02	1,29E-02	1,13072E-03	5,92512E-02
F5	5,25166E-01	4,70E-01	1,48473E-02	1,89182E+00	3,96761E-01	3,94E-01	5,21835E-03	1,54310E+00
F6	2,13414E-04	2,81E-04	8,25006E-07	1,28884E-03	8,98761E-05	9,69E-05	6,29787E-07	4,18959E-04
F7	-7,91647E+00	3,48E+00	-1,05361E+01	-1,85410E+00	-1,00514E+01	1,66E+00	-1,05358E+01	-2,78949E+00
F8	4,45150E-04	1,00E-03	1,86757E-06	6,20787E-03	7,55575E-05	9,87E-05	3,66689E-07	3,69110E-04
F9	-1,75473E+02	1,18E+01	-1,95831E+02	-1,39283E+02	-1,78583E+02	9,16E+00	-1,95831E+02	-1,67555E+02
F10	9,51206E-04	9,56E-04	1,45758E-05	3,52710E-03	1,16669E-04	1,19E-04	2,11245E-08	5,28851E-04
F11	6,80868E-03	5,96E-03	3,86499E-05	2,29567E-02	5,39555E-03	7,21E-03	2,71137E-04	3,60836E-02
F12	5,48445E-02	2,22E-02	1,59921E-02	1,16417E-01	5,69441E-02	2,26E-02	1,94680E-02	1,28111E-01
F13	1,05292E-04	1,15E-04	8,42834E-06	6,62916E-04	5,64964E-02	3,99E-01	8,56081E-06	2,81838E+00
F14	7,16579E-03	6,64E-03	2,40058E-04	2,95003E-02	8,18712E-03	7,02E-03	6,13651E-04	3,42029E-02
F15	5,15668E-05	5,73E-05	5,08444E-06	3,35378E-04	4,90888E-05	4,48E-05	1,12607E-06	1,82843E-04

Çizelge 7.6. 30 popülasyonda büyüklüğünde deneysel testlerden elde edilen detaylı sonuçlar

	FOA				OFOA			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F1	-1,86711E+02	3,38E-02	-1,86731E+02	-1,86538E+02	-1,86716E+02	3,61E-02	-1,86731E+02	-1,86506E+02
F2	-4,13749E+00	2,26E-02	-4,15580E+00	-4,07007E+00	-4,15164E+00	9,19E-03	-4,15580E+00	-4,12669E+00
F3	1,23755E-04	1,75E-04	1,59821E-07	9,46793E-04	4,62403E-05	6,55E-05	1,02760E-07	3,55702E-04
F4	2,98364E-02	3,60E-02	4,75443E-04	2,02114E-01	1,55219E-02	1,24E-02	2,37127E-04	5,60580E-02
F5	4,03678E-01	4,47E-01	8,85145E-03	1,83152E+00	3,32967E-01	3,81E-01	3,86814E-03	1,89573E+00
F6	2,01764E-04	2,75E-04	3,00466E-07	1,30781E-03	7,13101E-05	7,73E-05	1,61503E-08	3,19713E-04
F7	-8,83738E+00	2,88E+00	-1,05360E+01	-2,78946E+00	-1,05262E+01	1,03E-02	-1,05362E+01	-1,04926E+01
F8	1,54265E-04	3,55E-04	8,46270E-08	2,41324E-03	9,31415E-05	1,49E-04	3,51143E-07	6,85745E-04
F9	-1,74625E+02	1,19E+01	-1,95831E+02	-1,53420E+02	-1,91835E+02	6,39E+00	-1,95831E+02	-1,81689E+02
F10	6,47801E-04	1,07E-03	5,41651E-07	6,65770E-03	1,14909E-04	1,48E-04	1,42460E-07	6,43414E-04
F11	5,53161E-03	6,29E-03	4,65502E-04	2,65395E-02	4,87737E-03	4,65E-03	2,51696E-04	1,81230E-02
F12	5,14546E-02	2,86E-02	5,44041E-03	1,37626E-01	5,65384E-02	2,25E-02	1,28716E-02	1,15452E-01
F13	1,07800E-04	1,09E-04	6,84184E-06	4,90794E-04	8,42018E-05	6,98E-05	5,82543E-06	3,32056E-04
F14	8,37985E-03	5,84E-03	1,25782E-03	2,87340E-02	9,16335E-03	7,32E-03	7,65176E-04	3,83170E-02
F15	3,83170E-02	4,23E-05	2,26689E-06	1,90262E-04	4,51735E-05	4,46E-05	2,58738E-06	1,86320E-04

Çizelge 7.7. 50 popülasyonda büyüklüğünde deneysel testlerden elde edilen detaylı sonuçlar

	FOA				OFOA			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F1	-1,86716E+02	1,95E-02	-1,86731E+02	-1,86652E+02	-1,86722E+02	1,31E-02	-1,86731E+02	-1,86672E+02
F2	-4,14183E+00	1,72E-02	-4,15579E+00	-4,08348E+00	-4,14852E+00	1,17E-02	-4,15580E+00	-4,12620E+00
F3	6,83438E-05	1,08E-04	5,22096E-07	6,58517E-04	3,17694E-05	3,94E-05	2,16859E-08	1,54746E-04
F4	1,82010E-02	1,53E-02	7,23225E-05	6,40283E-02	1,82730E-02	2,36E-02	9,03858E-04	1,54223E-01
F5	4,70570E-01	3,67E-01	1,94750E-02	1,43245E+00	3,66204E-01	2,90E-01	4,19412E-03	1,13067E+00
F6	1,20139E-04	1,62E-04	7,31464E-07	7,66786E-04	7,43701E-05	7,70E-05	2,67967E-08	3,64482E-04
F7	-1,00848E+01	1,47E+00	-1,05362E+01	-5,12698E+00	-1,03101E+01	1,07E+00	-1,05363E+01	-5,12781E+00
F8	1,39662E-04	2,73E-04	4,23898E-10	1,23201E-03	1,34360E-04	2,00E-04	6,63214E-07	8,93254E-04
F9	-1,95830E+02	4,65E-04	-1,95831E+02	-1,95829E+02	-1,95830E+02	5,63E-04	-1,95831E+02	-1,95828E+02
F10	3,14183E-04	3,51E-04	2,09248E-06	1,11701E-03	6,58595E-05	8,71E-05	1,60875E-06	3,46456E-04
F11	5,14378E-03	4,79E-03	6,20950E-04	2,94521E-02	5,52172E-03	5,92E-03	1,49203E-04	3,22297E-02
F12	6,24750E-02	2,10E-02	2,42561E-02	1,09981E-01	5,55420E-02	2,33E-02	1,63983E-02	1,22447E-01
F13	8,53456E-05	9,42E-05	3,94496E-06	5,49673E-04	8,89380E-05	1,03E-04	3,67681E-06	5,02941E-04
F14	8,55272E-03	7,90E-03	2,19692E-04	3,29875E-02	7,78608E-03	7,18E-03	3,837977E-04	3,15542E-02
F15	4,20982E-05	3,26E-05	5,12327E-06	1,80683E-04	3,81267E-05	3,86E-05	3,86539E-06	1,97742E-04

Her popülasyon büyüklüğündeki detaylı sonuçlar incelendiğinde, OFOA algoritmasının FOA algoritmasına kıyasla, en kötü ortalama uygunluk değerleri ve en iyi ortalama uygunluk değerleri açısından daha iyi sonuçlar ortaya koyduğu görülmektedir. Bu sonuçlar, karşıtlık tabanlı nesil atlamanın sağladığı bir durumdur. Algoritmanın bazı adımlarında mevcut popülasyonun karşıtı da değerlendirmeye alınarak mevcut

popülasyonun uygunluğu iyileştirilebilmektedir ve algoritma arama uzayının daha umut verici bir bölgesini keşfedebilmekte ve bu bölgeye sıçrayarak yerel minimumlardan kaçabilmektedir. Bu deneysel sonuçlar OBL yönteminin algoritmanın arama uzayındaki keşif yeteneğini artırabildiğini destekler niteliktedir.

Çizelge 7.8. Farklı popülasyon büyüklüklerinde başarı oranlarının karşılaştırılması

	$area\ limit = 20$		$area\ limit = 30$		$area\ limit = 50$	
	FOA	**OFOA**	**FOA**	**OFOA**	**FOA**	**OFOA**
F1	0%	0%	58%	72%	62%	72%
F2	48%	96%	56%	88%	58%	76%
F3	88%	98%	100%	100%	100%	100%
F4	32%	48%	26%	34%	36%	38%
F5	0%	4%	2%	2%	0%	2%
F6	100%	100%	100%	100%	100%	100%
F7	13%	64%	42%	66%	56%	58%
F8	100%	100%	100%	100%	100%	100%
F9	0%	0%	5%	60%	100%	100%
F10	100%	100%	100%	100%	100%	100%
F11	78%	88%	84%	84%	92%	84%
F12	0%	0%	2%	0%	0%	0%
F13	100%	98%	100%	100%	100%	100%
F14	72%	78%	26%	38%	74%	72%
F15	100%	100%	100%	100%	100%	100%

Çizelge 7.8 FOA ve OFOA algoritmalarının Çizelge 7.4'te sonuçları verilen deneysel testte elde ettiği başarı oranlarını gösterir. Sonuçlar incelendiğinde OFOA algoritmasının tüm popülasyon büyüklüklerinde başarı oranının daha yüksek olduğu görülmektedir. Daha düşük popülasyon büyüklüklerinde OFOA algoritmasının popülasyon çeşitliliğinin artırılmasıyla fonksiyonların çoğunu FOA algoritmasından daha iyi çözebildiği başarı oranları ile desteklenmiştir. Popülasyon büyüklüğü arttıkça algoritmaların performanslarının birbirine yaklaştığı görülmektedir.

7.1.6. Problem boyutunun etkisi

Bu bölümde FOA ve OFOA algoritmalarının farklı boyutlardaki fonksiyonlar üzerindeki performansı incelenmiştir. Ölçeklenebilir fonksiyonların boyutları 5, 10 ve 20 olarak ayarlanıp iki algoritmanın performansları bu boyutlarda kıyaslanmıştır. Algoritmaların parametre değerleri Çizelge 7.2'deki gibi kullanılmıştır. Çizelge 7.9, Çizelge 7.10 ve Çizelge 7.11'de sırasıyla 5, 10 ve 20 boyutlu fonksiyonlardaki ortalama,

en iyi, en kötü uygunluk değerleri ve standart sapma değerleri açısından elde edilen sonuçlar verilmiştir.

Çizelge 7.9. 5 boyutlu ölçeklenebilir fonksiyonlarda FOA ve OFOA sonuçlarının karşılaştırılması

	FOA				**OFOA**			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F9	-1,74625E+02	1,19E+01	-1,95831E+02	-1,53420E+02	-1,91835E+02	6,39E+00	-1,95831E+02	-1,81689E+02
F10	6,47801E-04	1,07E-03	5,41651E-07	6,65770E-03	1,14909E-04	1,48E-04	1,42460E-07	6,43414E-04
F11	5,53161E-03	6,29E-03	4,65502E-04	2,65395E-02	4,87737E-03	4,65E-03	2,51696E-04	1,81230E-02
F12	5,14546E-02	2,86E-02	5,44041E-03	1,37626E-01	5,65384E-02	2,25E-02	1,28716E-02	1,15452E-01
F13	1,07800E-04	1,09E-04	6,84184E-06	4,90794E-04	8,42018E-05	6,98E-05	5,82543E-06	3,32056E-04
F14	8,37985E-03	5,84E-03	1,25782E-03	2,87340E-02	9,16335E-03	7,32E-03	7,65176E-04	3,83170E-02
F15	3,83170E-02	4,23E-05	2,26689E-06	1,90262E-04	4,51735E-05	4,46E-05	2,58738E-06	1,86320E-04

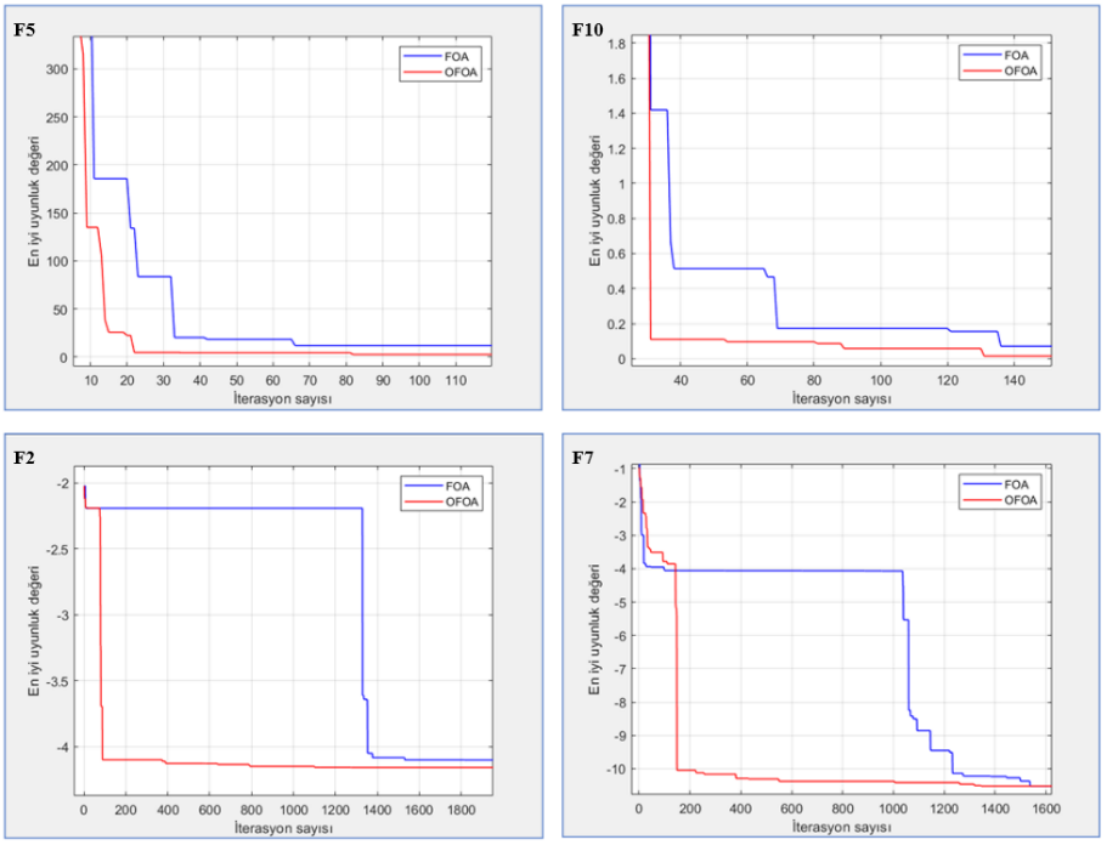
Çizelge 7.10. 10 boyutlu ölçeklenebilir fonksiyonlarda FOA ve OFOA sonuçlarının karşılaştırılması

	FOA				**OFOA**			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F9	-3,85149E+02	7,67E+00	-3,91661E+02	-3,63386E+02	-3,87938E+02	6,32E+00	-3,91661E+02	-3,75758E+02
F10	1,73363E-04	1,68E-04	2,18719E-06	7,21744E-04	1,07107E-04	1,24E-04	9,43541E-07	4,36789E-04
F11	7,37463E-02	4,75E-02	7,79372E-03	2,21318E-01	7,32909E-02	4,81E-02	1,75876E-02	2,48962E-01
F12	1,33851E-01	3,82E-02	6,56562E-02	2,21628E-01	1,22291E-01	3,91E-02	5,91987E-02	2,10948E-01
F13	5,51157E-04	1,15E-03	4,46642E-05	6,68421E-03	2,39362E-04	1,73E-04	4,37928E-05	6,73637E-04
F14	2,07412E-02	1,46E-02	2,16091E-03	6,38608E-02	1,93144E-02	1,18E-02	4,02667E-03	5,39621E-02
F15	1,18431E-04	6,84E-05	3,08979E-05	2,85392E-04	1,09382E-04	7,49E-05	1,81203E-05	3,48193E-04

Çizelge 7.11. 20 boyutlu ölçeklenebilir fonksiyonlarda FOA ve OFOA sonuçlarının karşılaştırılması

	FOA				**OFOA**			
	AVG_F	STD_F	$BEST_F$	$WORST_F$	AVG_F	STD_F	$BEST_F$	$WORST_F$
F9	-7,18284E+02	2,25E+01	-7,55046E+02	-6,56086E+02	-7,23654E+02	2,38E+01	-7,69174E+02	-6,70224E+02
F10	9,65447E-05	1,04E-04	7,28348E-07	5,49238E-04	1,18351E-04	1,24E-04	6,61431E-07	4,33694E-04
F11	2,24380E+00	1,23E+00	4,43390E-01	6,39184E+00	2,13846E+00	1,37E+00	4,31288E-01	5,52989E+00
F12	2,02432E-01	5,73E-02	8,86875E-02	3,32562E-01	2,01600E-01	6,93E-02	9,06401E-02	3,75091E-01
F13	4,97085E+00	2,81E+00	4,05105E-04	9,92651E+00	4,59062E+00	2,80E+00	6,89273E-04	1,08200E+01
F14	5,80441E-02	3,10E-02	1,64277E-02	1,56372E-01	4,95031E-02	2,26E-02	1,37647E-02	1,33689E-01
F15	2,39061E-04	1,31E-04	7,92887E-05	8,30451E-04	2,66609E-04	1,41E-04	9,94039E-05	9,01197E-04

Elde edilen sonuçlara göre tüm boyutlarda OFOA algoritması FOA algoritmasına kıyasla daha başarılı sonuçlar elde edebilmektedir. Ancak boyut arttıkça iki algoritmanın elde edebildiği sonuçlar birbirine yaklaşır. Sonuçlar incelendiğinde FOA algoritmasına kıyasla OFOA algoritmasının, düşük boyutlardaki arama uzayını daha iyi keşfetme yeteneğini yüksek boyutlarda da devam ettirebildiği görülmektedir.



Şekil 7.1. F2, F5,F7 ve F10 fonksiyonlarının yakınsama grafikleri

Şekil 7.1'de F2, F5, F7 ve F10 olmak üzere dört fonksiyonun yakınsama grafikleri verilmiştir. Şekilden OFOA algoritmasının FOA algoritmasından daha erken iterasyonlarda yakınsama sağlayabildiği görülmektedir. Yakınsama grafiklerinden OBL yönteminin nesil atlama şeklinde uygulanması ile daha iyi aday çözümlerin daha erken iterasyonlarda elde edilmesini sağladığı söylenebilir. Ayrıca grafiklerden OFOA algoritmasının bazı iterasyonlarda arama uzayının daha iyi bölgelerini keşfedip o bölgelere sıçrayarak yakınsama hızının artırılabildiği ve yerel minimumlardan kaçtığı yorumu yapılabilir.

7.2. Öznitelik Seçimi Sonuçları

Öznitelik seçiminde performans değerlendirmesi için çeşitli veri setleri üzerinde OFOA algoritmasının ikili versiyonu olan B-OFOA algoritması ile FSFOA, BDE ve BPSO algoritmalarının sonuçları sınıflandırma doğruluğu ve seçilen öznitelik sayısı ölçütleri ile karşılaştırılmıştır.

Bu bölümde algoritmaların veri setleri üzerindeki karşılaştırmalı sonuçları sunulmuştur. Sunulan sonuçlar 10 farklı çalıştırmanın ortalama değerleridir. Her farklı

çalıştırmada algoritmalar aynı başlangıç popülasyonu ile başlatılmıştır. Testlerde kullanılan veri setleri ve algoritmaların parametre değerleri sunulmuştur.

7.2.1. Veri setleri

Öznitelik seçimi problemlerinde algoritmaların performanslarını değerlendirmek için UCI (University of California, Irvine machine learning repository) veri deposundan 9 adet veri seti kullanılmıştır (Dua, 2017). Kullanılan veri setlerinin detayları Çizelge 7.12'de verilmiştir.

Çizelge 7.12. Öznitelik seçiminde kullanılan veri setleri

Veri Seti	Öznitelik Sayısı	Sınıf Sayısı	Örnek Sayısı
GLASS	9	7	214
HEART-STATLOG	13	2	270
WINE	13	3	178
HEPATITIS	19	2	155
BREASTEW	31	2	569
IONOSPHERE	34	2	351
WAVEFORM2	40	3	5000
SONAR	60	2	208
LIBRASMOVEMENT	90	15	360

Öznitelik seçimi probleminde veri setleri küçük ölçekli, orta ölçekli ve büyük ölçekli olarak sınıflandırılabilir. Küçük, orta ve büyük ölçekli veri setleri sırasıyla [0,19], [20,49] ve [50,∞] öznitelik içerir (Ghaemi ve Feizi-Derakhshi, 2016). Bu çalışmada kullanılan veri setlerinin 4 tanesi küçük ölçekli, 3 tanesi orta ölçekli ve 2 tanesi büyük ölçekli veri setidir.

7.2.2. Test ortamı ve parametre ayarları

Testler gerçekleştirilirken kullanılan parametre değerleri Çizelge 7.13'te verilmiştir. Yapılan testlerde iterasyon sayısı 70 ve popülasyon büyüklüğü 30 olarak sabitlenmiştir.

FSFOA algoritmasının parametreleri popülasyon büyüklüğü haricinde önerildiği orijinal makaledeki gibi ayarlanmıştır. B-OFOA algoritmasında ise OBL yönteminin etkisinin incelenebilmesi için FSFOA ile aynı parametre değerleri kullanılmıştır. BDE ve

BPSO algoritmaları için Too ve ark. tarafından (Too ve ark., 2019) 2019'da yapılan çalışmadaki parametreler kullanılmıştır.

Çizelge 7.13. Öznitelik seçiminde algoritmaların parametre değerleri

Algoritma	Parametreler	Değerler
FSFOA (Orman Optimizasyonu Algoritması Kullanılarak Öznitelik Seçimi)	Yaşam süresi, $life\_time$ Yerel tohumlama, LSC Küresel tohumlama, GSC Transfer oranı, $transfer\_rate$	15 Problem boyutu/5 Problem boyutu/2 5
B-OFOA (İkili Karşıtlık Tabanlı Orman Optimizasyonu Algoritması)	Yaşam süresi, $life\_time$ Yerel tohumlama, LSC Küresel tohumlama, GSC Transfer oranı, $transfer\_rate$ Atlama oranı, Jr	15 Problem boyutu/5 Problem boyutu/2 5 0.4
BDE (İkili Diferansiyel Evrim)	Çaprazlama oranı, CR	1
BPSO (İkili Parçacık Sürü Optimizasyonu)	Hızlanma katsayısı, $c1$ ve $c2$ Eylemsizlik ağırlığı, w Hız sınırları	2 0.9-0.4 (-6,6)

Öznitelik seçiminde minimum sınıflandırma hata oranı ve minimum seçilen öznitelik sayısı ile en iyi öznitelik alt kümesinin bulunması hedeflenir. Bu doğrultuda yapılan testlerde kullanılan k-NN sınıflandırıcısına bağımlı uygunluk fonksiyonu Denklem 7.6'da verilmiştir.

$$f(x) = \alpha * (1 - ACC) + \beta * \frac{|R|}{|C|}, \quad \beta = 1 - \alpha \tag{7.6}$$

Burada ACC seçilen öznitelik alt kümesi ile k-NN sınıflandırıcısında elde edilen sınıflandırma doğruluğunu, $|R|$ veri setinden seçilen öznitelik sayısını ve $|C|$ veri setinde bulunan toplam öznitelik sayısını gösterir. α ve β sınıflandırma doğruluğu ile seçilen öznitelik sayısı arasında bir denge kurmak için kullanılan iki parametredir ve $\alpha \in [0,1]$'dir.

Sınıflandırma doğruluğunun değerlendirilmesi için k-NN (k=5) sınıflandırıcısı kullanılmıştır ve veri setleri Hold-Out (0.2) yöntemi ile %80 eğitim seti ve %20 test seti olarak ayrılmıştır.

7.2.3. Değerlendirme ölçütleri

Bu çalışmada algoritmaların veri setleri üzerindeki performanslarını değerlendirmek için ortalama sınıflandırma doğruluğu ve ortalama seçilen öznitelik sayısı ölçütleri kullanılmıştır. Kullanılan ölçütler matematiksel tanımları ile bu bölümde sunulmuştur.

Sınıflandırma doğruluğu, bulunan en iyi öznitelik alt kümesi ile k-NN sınıflandırıcısında elde edilen doğru sınıflandırmaların veri setindeki tüm örnek sayısına oranıdır. Ortalama sınıflandırma doğruluğu, algoritmanın farklı çalıştırmalar için elde ettiği sınıflandırma doğruluklarının ortalamasıdır. Denklem 7.7 ile hesaplanır. Burada R_C doğru sınıflandırmaların sayısıdır ve T_C veri setindeki örnek sayısıdır. N_r, farklı çalıştırmaların sayısıdır.

$$AVG_{ACC} = \frac{1}{N_r}\sum_{i=1}^{N_r}\frac{R\_C}{T\_C} \tag{7.7}$$

Ortalama seçilen öznitelik sayısı, algoritmanın farklı çalıştırmalar için elde ettiği en iyi öznitelik alt kümesindeki öznitelik sayılarının ortalamasıdır. Burada $|R|^*$, bulunan en iyi öznitelik alt kümesinden seçilen öznitelik sayısıdır.

$$AVG_{FC} = \frac{1}{N_r}\sum_{i=1}^{Nr}|R|^* \tag{7.8}$$

B-OFOA ve FSFOA algoritmaların karşılaştırılmasında ayrıca en iyi sınıflandırma doğruluğu, en kötü sınıflandırma doğruluğu ve standart sapma ölçütleri kullanılmıştır. Bu değerlendirme ölçütlerinin matematiksel tanımları aşağıdaki verilmiştir.

En kötü ölçütü, algoritmanın farklı çalıştırmalarda elde ettiği en kötü sınıflandırma doğruluğudur ve Denklem 7.9'daki gibi hesaplanır.

$$WORST_{ACC} = \min_{1\le i\le N_r}\frac{R\_C}{T\_C} \tag{7.9.}$$

En iyi ölçütü, algoritmanın farklı çalıştırmalarda elde ettiği en iyi sınıflandırma doğruluğudur ve Denklem 7.10'daki gibi hesaplanır.

$$BEST_{ACC} = \max_{1\le i\le N_r}\frac{R\_C}{T\_C} \tag{7.10.}$$

Standart sapma, algoritmanın tutarlılığını değerlendirebilmek için kullanılan bir ölçüttür ve Denklem 7.11 ile hesaplanır.

$$STD_{ACC} = \sqrt{\frac{1}{N_r - 1}\sum_{i=1}^{N_r}\left(\frac{R\_C}{T\_C} - ortalama\right)^2} \tag{7.11}$$

7.2.3. Elde edilen sonuçlar

Öznitelik seçimi değerlendirmesinin dört algoritma için sınıflandırma doğruluğu ve seçilen öznitelik sayısı açısından karşılaştırmalı sonuçları Çizelge 7.14'te sunulmuştur. Daha iyi doğruluk değerleri ve seçilen öznitelik sayıları gri arka plan rengi ile vurgulanmıştır. Doğruluk değerlerinin aynı olduğu durumlarda seçilen öznitelik sayısı daha küçük olan daha iyi kabul edilmiştir.

Çizelge 7.14. Öznitelik seçiminde elde edilen karşılaştırmalı sonuçlar

	B-OFOA		**FSFOA**		**BDE**		**BPSO**	
	AVG_{ACC}	AVG_{FC}	AVG_{ACC}	AVG_{FC}	AVG_{ACC}	AVG_{FC}	AVG_{ACC}	AVG_{FC}
GLASS	0,793	4,10	0,788	3,70	0,786	5,00	0,786	3,50
HEART-STATLOG	0,889	4,20	0,891	4,30	0,841	6,70	0,883	4,00
WINE	0,989	3,90	0,989	4,00	0,977	6,90	0,989	4,00
HEPATITIS	0,752	3,00	0,735	2,60	0,713	8,90	0,755	3,30
BREASTEW	0,987	8,30	0,983	8,40	0,956	16,40	0,973	9,20
IONOSPHERE	0,973	5,50	0,970	5,60	0,901	15,40	0,937	7,80
WAVEFORM2	0,858	13,50	0,859	14,50	0,832	26,60	0,843	17,60
SONAR	0,961	11,70	0,944	13,10	0,885	37,20	0,934	22,10
LIBRAS	0,756	40,90	0,746	40,30	0,583	61,20	0,607	42,40

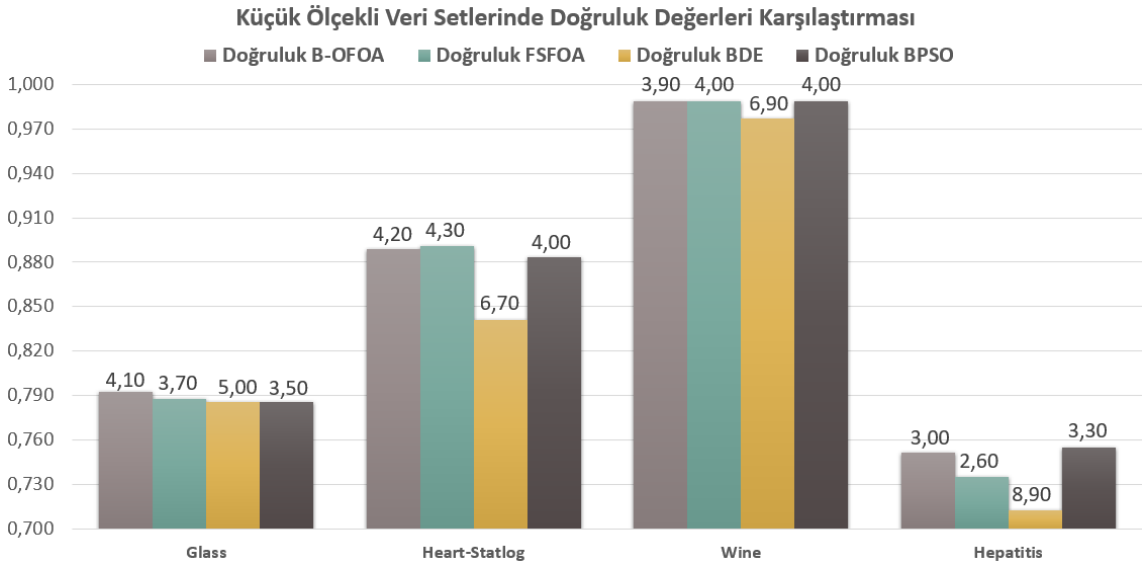
Farklı ölçeklerdeki 9 veri seti üzerinden B-OFOA algoritması diğer algoritmalara kıyasla hem sınıflandırma doğruluğu hem de seçilen öznitelik sayısı açısından çoğu veri setinde daha iyi sonuçlar vermiştir. Sınıflandırma doğruluğunun eşit ya da aynı olduğu durumlarda B-OFOA algoritması daha az öznitelik seçimi ile daha başarılı sonuçlar elde etmiştir. Küçük ölçekli veri setlerinden ziyade orta ve büyük ölçekli veri setlerinde OFOA algoritmasının başarısı daha belirgindir. Problem zorlaştığında B-OFOA algoritması performansını koruyabilmiştir. Çizelge 7.15'te B-OFOA ve FSFOA algoritmalarının ortalama, en iyi, en kötü sınıflandırma doğruluğu ve standart sapma değerleri açısından karşılaştırmalı sonuçları sunulmuştur. Sonuçlar B-OFOA algoritmasının daha iyi performansını gösterir.

Çizelge 7.15. Öznitelik seçiminde B-OFOA ve FSFOA algoritmalarının karşılaştırmalı sonuçları

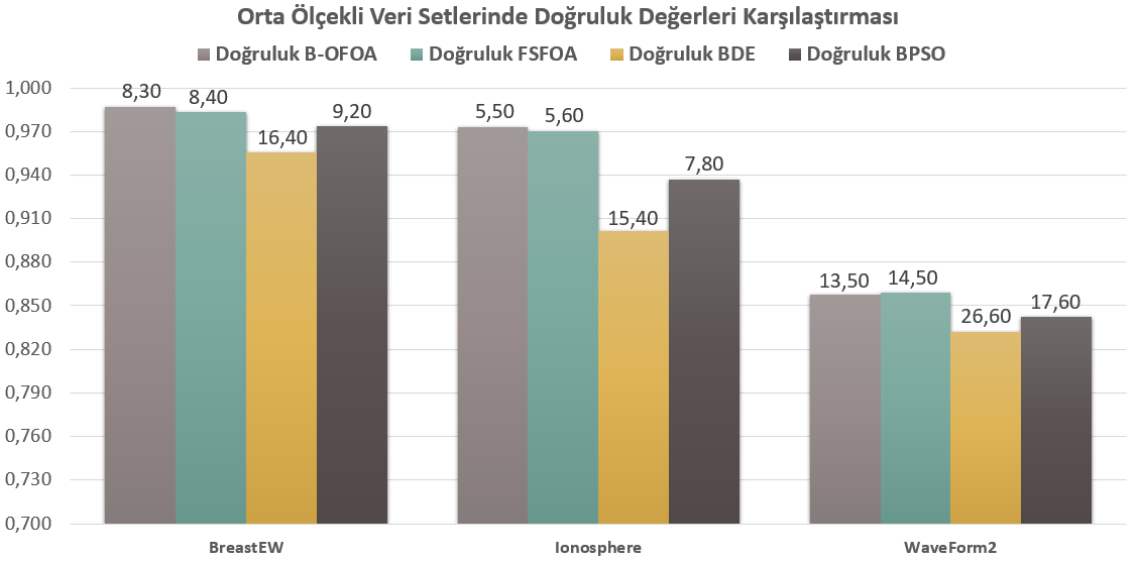
	B-OFOA				FSFOA			
	AVG_{ACC}	STD_{ACC}	$BEST_{ACC}$	$WORST_{ACC}$	AVG_{ACC}	STD_{ACC}	$BEST_{ACC}$	$WORST_{ACC}$
GLASS	0,793	0,056	0,881	0,714	0,788	0,061	0,881	0,690
HEART-STATLOG	0,889	0,025	0,926	0,833	0,891	0,020	0,926	0,852
WINE	0,989	0,020	1,000	0,943	0,989	0,020	1,000	0,943
HEPATITIS	0,752	0,083	0,839	0,548	0,735	0,071	0,806	0,548
BREASTEW	0,987	0,010	1,000	0,973	0,983	0,012	1,000	0,965
IONOSPHERE	0,973	0,016	1,000	0,943	0,970	0,016	1,000	0,943
WAVEFORM2	0,858	0,008	0,871	0,847	0,859	0,009	0,872	0,848
SONAR	0,961	0,024	1,000	0,927	0,944	0,043	1,000	0,854
LIBRAS	0,756	0,062	0,861	0,667	0,746	0,061	0,833	0,639

Elde edilen en kötü sınıflandırma doğruluğu açısından 3 veri setinde iki algoritmada aynı sonucu elde etmiştir. 4 veri setinde B-OFOA algoritmasını daha iyi sonuçlara sahiptir. En iyi sınıflandırma doğrulukları dikkate alınırsa 6 veri setinde iki algoritma aynı sonuçları elde etmiştir, 2 veri setinde B-OFOA algoritması FSFOA algoritmasının bulamadığı optimale daha yakın öznitelik alt kümelerini bulmayı başarmıştır. Bu durum karşıtlık tabanlı nesil atlama uygulamasının getirdiği avantajları doğrular niteliktedir.

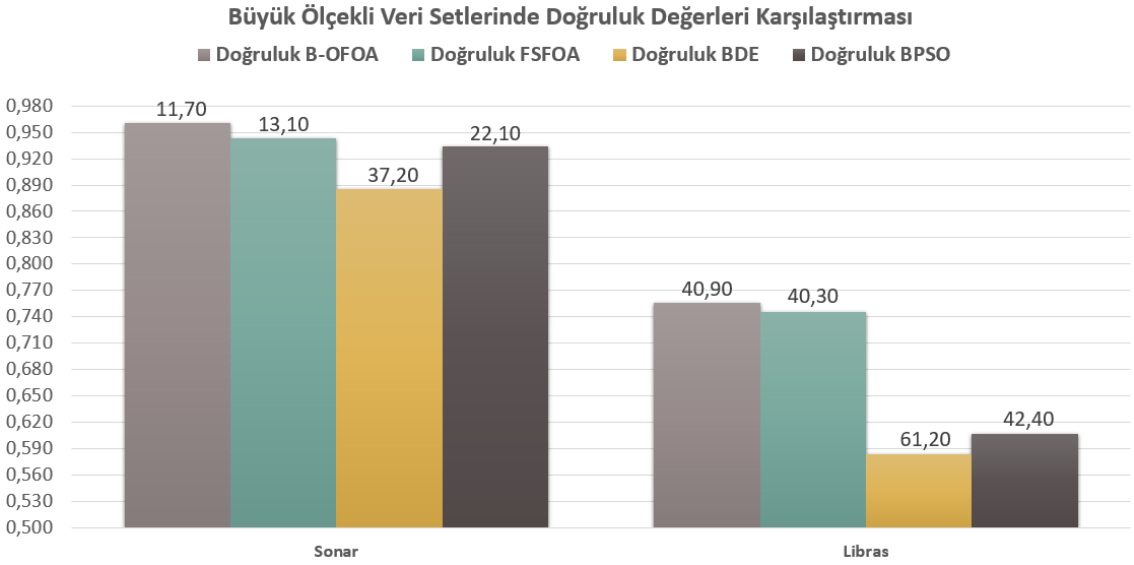
Şekil 7.2, 7.3 ve 7.4'te sırasıyla küçük, orta ve büyük ölçekli veri setleri üzerinde elde edilen sonuçların grafikleri sunulmuştur. Grafiklerde sütunlar sınıflandırma doğruluklarını ve sütunların etiketlerindeki değerler seçilen öznitelik sayılarını gösterir.



Şekil 7.2. Küçük ölçekli veri setlerinde elde edilen sonuçların grafiksel gösterimi



Şekil 7.3. Orta ölçekli veri setlerinde elde edilen sonuçların grafiksel gösterimi



Şekil 7.4. Büyük ölçekli veri setlerinde elde edilen sonuçların grafiksel gösterimi

Wine ve Heart-Statlog veri setlerinde B-OFOA algoritması BPSO ve FSFOA algoritmaları ve Hepatitis veri setinde BPSO algoritması ile birbirine yakın sonuçlar elde etmiştir. Wine veri setinde bu üç algoritma aynı sınıflandırma doğruluğuna sahiptir ancak B-OFOA seçilen öznitelik sayısını diğerlerinden daha fazla azaltabilmiştir. Glass veri seti, sınıf sayısının fazlalığı açısından diğer küçük ölçekli veri setlerinden ayrılır. Glass veri setinde B-OFOA algoritması en yüksek sınıflandırma doğruluğuna sahiptir.

Şekil 7.3'te görüldüğü gibi orta ölçekli veri setlerinde B-OFOA ve FSFOA algoritmaları sınıflandırma doğruluğu ve öznitelik sayısı bakımından diğer iki algoritmadan daha iyi sonuçlar elde etmiştir. B-OFOA ve FSFOA algoritmalarının ikili

karşılaştırmasına göre, B-OFOA BreastEW ve Ionosphere veri setlerinde daha yüksek sınıflandırma doğruluğu ve daha düşük öznitelik sayısı ile daha başarılıdır. WaveForm2 veri setinde iki algoritmanın sınıflandırma doğrulukları çok yakındır ve seçilen öznitelik sayısını azaltmada B-OFOA daha başarılı olmuştur.

Büyük ölçekli veri setlerinde ise B-OFOA algoritması diğer algoritmalara kıyasla daha iyi performansını devam ettirebilmiştir. Kullanılan iki büyük ölçekli veri setinde B-OFOA algoritması daha yüksek sınıflandırma doğruluğuna sahiptir.

8. SONUÇLAR VE ÖNERİLER

Optimizasyon problemlerinin çözümünde meta-sezgisel algoritmaların başarısı bilinmektedir. Literatüre sürekli yeni bir meta-sezgiselin eklenmesinin yanı sıra, mevcut meta-sezgiselleri iyileştirme çalışmaları da oldukça yaygındır. Literatürde bu araştırma alanı oldukça ilgi görmektedir. Çeşitli yöntemler uygulanarak algoritmaların performansları artırılmaya ve zayıf yönleri güçlendirilmeye çalışılmaktadır.

FOA algoritmasında, ağaçların doğadaki tohumlama ve yaşam süreçleri simüle edilir. Rastgele oluşturulan başlangıç popülasyonundaki ağaçlar yaşları sıfır olarak üretilir ve sırasıyla yerel tohumlama, popülasyon sınırlama, küresel tohumlama ve en iyi ağacı güncelleme adımlarından geçerler. Yerel tohumlama sadece sıfır yaşındaki ağaçlara uygulanır ve ilgili ağaçların yakın komşularını popülasyona ekleyerek algoritmanın yerel arama sürecini gerçekleştirir. Popülasyon sınırlama ile popülasyonun büyümesi kontrol altına alınır. Küresel tohumlama problem uzayının küresel olarak aranması sürecini gerçekleştiren adımıdır. En iyi ağaç belirlendikten sonra yaşı sıfır olarak güncellenir. Böylelikle iyi ağaçlar yerel tohumlama aşamasında yerel olarak optimize edilir. FSFOA algoritması, FOA algoritmasının öznitelik seçimi problemleri için uyarlanmış versiyonudur.

FOA ve FSFOA algoritmaları optimizasyon problemlerinde başarılı sonuçlar elde edebilmektedir fakat algoritmalar bazı zayıf yönlere sahiptir. Bu çalışmada ele alınan noktalardan ilki yerel tohumlama ve küresel tohumlama operatörleri ile gerçekleştirilen çözüm uzayının aranma verimliliğinin nispeten zayıf olmasıdır. Aynı zamanda başlangıç popülasyonunun rastgele başlatılması da dikkate alınmıştır.

OBL, 2005 yılında algoritmaların yakınsama hızlarını artırabilecek ve problem uzayını arama yeteneğini geliştirebilecek bir yöntem olarak önerilmiştir. Bu yöntemin temel fikri mevcut bir durumun karşıtını da değerlendirmeye almaktır. Karşıtın değerlendirilmesiyle daha iyi bir durumu keşfedebilme ihtimali üzerinde durur.

Çalışmada önerilen OFOA algoritması ilk popülasyonun başlatılmasında ve iteratif süreçte nesil atlama şeklinde OBL yöntemini kullanır. Karşıtlık tabanlı popülasyon başlatma ile OFOA algoritmasının daha iyi bir uygunluğa sahip popülasyon ile probleme başlaması olasıdır. Nesil atlama, algoritmanın bazı iterasyonlarda popülasyon uygunluğunu iyileştirme ve arama uzayının daha umut verici bölgelerini keşfedip yerel optimallerden kaçabilme ihtimalini artırır. Ayrıca önerilen algoritma ikili

arama uzaylarına uyarlanarak (B-OFOA) algoritmanın öznitelik seçimi problemlerinde performans analizi yapılmıştır.

Kıyaslama fonksiyonlarında gerçekleştirilen deneysel testlerden elde edilen sonuçlar, OBL yönteminin algoritmanın arama uzayını keşfetme yeteneğini artırabildiğini göstermiştir. OFOA algoritması FOA algoritmasına kıyasla daha az yerel optimallere tuzaklanma eğilimi göstermiştir. Yöntem popülasyon çeşitliliğini artırabildiği için algoritmanın daha küçük popülasyon büyüklüklerinde de daha iyi bir performans elde edebilmesini sağlamıştır. Ayrıca daha büyük problem boyutlarında performansını koruyabilmiştir. Dört fonksiyona ait paylaşılan yakınsama grafikleriyle OFOA algoritmasının daha erken iterasyonlarda daha iyi aday çözümleri bulabildiği gösterilmiştir. Paylaşılan başarı oranları da elde edilen sonuçları destekler.

Öznitelik seçimi testlerinde B-OFOA algoritması literatürden alınan FSFOA, BDE ve BPSO algoritmalarından daha iyi bir performans göstermiştir. Algoritmaların performansı küçük veri setlerinde birbirine daha yakındır ancak orta ve büyük ölçekli veri setlerinde B-OFOA algoritması diğer algoritmalara kıyasla daha iyi sonuçlar elde edebilmiştir. Öznitelik seçiminden elde edilen sonuçlar OBL yönteminin arama uzayını keşfetme yeteneğini artırdığını destekler niteliktedir.

Bu tez çalışmasında 2014 yılında önerilen popülasyon tabanlı bir meta-sezgisel algoritma olan FOA algoritmasının performansının iyileştirilmesi amaçlanmıştır. Çalışmada önerilen karşıtlık tabanlı iki algoritmanın orijinal versiyondan daha iyi sonuçlar elde ettiği tespit edilmiştir. Bu doğrultuda sürekli optimizasyon problemleri için OFOA algoritması ve öznitelik seçimi problemleri için B-OFOA algoritması literatüre kazandırılmıştır.

Gelecek çalışmalar için OBL yöntemi, FOA algoritmasının operatörlerini iyileştirmek için uygulanabilir. Örneğin; yerel tohumlama operatörüne uygulanarak algoritmanın kullanım fazı iyileştirilebilir. Bunun yanı sıra OBL yönteminin literatürde farklı versiyonları bulunmaktadır. Öznitelik seçimine yönelik olarak OBL yönteminin ayrık problemler için önerilmiş versiyonları kullanılabilir.

KAYNAKLAR

Agrawal, P., Abutarboush, H. F., Ganesh, T. ve Mohamed, A. W. J. I. A., 2021, Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019), 9, 26766-26791.

Ahandani, M. A. ve Alavi-Rad, H. J. S. c., 2012, Opposition-based learning in the shuffled differential evolution algorithm, 16 (8), 1303-1337.

Baliarsingh, S. K., Vipsita, S., Dash, B. J. N. C. ve Applications, 2020, A new optimal gene selection approach for cancer classification using enhanced Jaya-based forest optimization algorithm, 32 (12), 8599-8616.

Bertsimas, D. ve Tsitsiklis, J. J. S. s., 1993, Simulated annealing, 8 (1), 10-15.

Chu, B., Li, Z.-S., Zhang, M.-L. ve Yu, H.-H. J. J. o. S., 2018, Research on improvements of feature selection using forest optimization algorithm, 29 (9), 2547-2558.

Cvijović, D. ve Klinowski, J. J. S., 1995, Taboo search: an approach to the multiple minima problem, 267 (5198), 664-666.

Çelik, Y., YILDIZ, İ. ve Karadeniz, A. T. J. A. B. v. T. D., 2019, Son Üç Yılda Geliştirilen Metasezgisel Algoritmalar Hakkında Kısa Bir İnceleme, 463-477.

Dorigo, M., Maniezzo, V., Colorni, A. J. I. T. o. S., Man, ve Cybernetics, P. B., 1996, Ant system: optimization by a colony of cooperating agents, 26 (1), 29-41.

Emary, E., Zawbaa, H. M. ve Hassanien, A. E. J. N., 2016a, Binary grey wolf optimization approaches for feature selection, 172, 371-381.

Emary, E., Zawbaa, H. M. ve Hassanien, A. E. J. N., 2016b, Binary ant lion approaches for feature selection, 213, 54-65.

Erdoğmuş, P. J. D. ü. b. v. t. d., 2016, Doğadan esinlenen optimizasyon algoritmaları ve optimizasyon algoritmalarının optimizasyonu, 4 (1), 293-304.

Ewees, A. A., Abd Elaziz, M. ve Houssein, E. H. J. E. S. w. A., 2018, Improved grasshopper optimization algorithm using opposition-based learning, 112, 156-172.

Ewees, A. A., Abd Elaziz, M. ve Oliva, D. J. E. S. w. A., 2021, A new multi-objective optimization algorithm combined with opposition-based learning, 165, 113844.

Fang, W., Zhou, J., Wu, X., Sun, J. ve Cheng, S., 2016, An opposition-based learning competitive particle swarm optimizer, *2016 IEEE Congress on Evolutionary Computation (CEC)*, 515-521.

Gao, X., Wang, X., Ovaska, S. ve Zenger, K. J. E. O., 2012, A hybrid optimization method of harmony search and opposition-based learning, 44 (8), 895-914.

García, S., Luengo, J. ve Herrera, F., 2015, Data preprocessing in data mining, Springer, p.

Geem, Z. W., Kim, J. H. ve Loganathan, G. V. J. s., 2001, A new heuristic optimization algorithm: harmony search, 76 (2), 60-68.

Ghaemi, M. ve Feizi-Derakhshi, M.-R. J. E. S. w. A., 2014, Forest optimization algorithm, 41 (15), 6676-6687.

Ghaemi, M. ve Feizi-Derakhshi, M.-R. J. P. R., 2016, Feature selection using forest optimization algorithm, 60, 121-129.

Gupta, S. ve Deep, K. J. E. S. w. A., 2019, A hybrid self-adaptive sine cosine algorithm with opposition based learning, 119, 210-230.

Haupt, R. L. ve Haupt, S. E., 2004, Practical genetic algorithms, John Wiley & Sons, p.
Holland, J. H. J. S. a., 1992, Genetic algorithms, 267 (1), 66-73.

Huang, H., Fan, Q., Yang, K., Zhang, S., Yao, L. ve Xiong, Q. J. E. S. w. A., 2021, A modified equilibrium optimizer using opposition-based learning and novel update rules, 170, 114575.

Hussien, A. G. J. J. o. A. I., 2022, An enhanced opposition-based salp swarm algorithm for global optimization and engineering problems, 13 (1), 129-150.

Ibrahim, H. T., Mazher, W. J., Ucan, O. N., Bayat, O. J. N. C. ve Applications, 2019, A grasshopper optimizer approach for feature selection and optimizing SVM parameters utilizing real biomedical data sets, 31 (10), 5965-5974.

Jamil, M. ve Yang, X.-S. J. a. p. a., 2013, A literature survey of benchmark functions for global optimization problems.

Kang, Q., Xiong, C., Zhou, M. ve Meng, L. J. I. A., 2018, Opposition-based hybrid strategy for particle swarm optimization in noisy environments, 6, 21888-21900.

Kennedy, J. ve Eberhart, R., 1995, Particle swarm optimization, *Proceedings of ICNN'95-international conference on neural networks*, 1942-1948.

Kennedy, J. ve Eberhart, R. C., 1997, A discrete binary version of the particle swarm algorithm, *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, 4104-4108.

Lin, L. ve Gen, M. J. S. C., 2009, Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation, 13 (2), 157-168.

Liu, F., Ma, X., Qi, Y., Gong, M., Yin, M., Li, L., Jiao, L. ve Wu, J. J. N., 2014, MOEA/D with opposition-based learning for multiobjective optimization problem, 146, 48-64.

Liu, H. ve Motoda, H., 2012, Feature selection for knowledge discovery and data mining, Springer Science & Business Media, p.

Liu, K., Wang, X. ve Qu, Z. J. E., 2019, Train Operation Strategy Optimization Based on a Double-Population Genetic Particle Swarm Optimization Algorithm, 12 (13), 2518.

Luo, J., He, F. ve Yong, J. J. I. D. A., 2020, An efficient and robust bat algorithm with fusion of opposition-based learning and whale optimization algorithm, 24 (3), 581-606.

Ma, T., Jia, D., Zhou, H., Xue, Y. ve Cao, J. J. I. D. A., 2018, Feature selection using forest optimization algorithm based on contribution degree, 22 (6), 1189-1207.

Ma, T., Zhou, H., Jia, D., Al-Dhelaan, A., Al-Dhelaan, M., Tian, Y. J. C. M. i. E. ve Sciences, 2019, Feature selection with a local search strategy based on the forest optimization algorithm, 121 (2), 569-592.

Mafarja, M. M. ve Mirjalili, S. J. N., 2017, Hybrid whale optimization algorithm with simulated annealing for feature selection, 260, 302-312.

Moayedi, H., Gupta, S., Deep, K., Heidari, A. A. ve Wang, M. J. E. S. w. A., 2020, Opposition-based learning Harris hawks optimization with advanced transition rules: Principles and analysis, 158, 113510.

Mohamed, A. W., Hadi, A. A., Mohamed, A. K. J. I. J. o. M. L. ve Cybernetics, 2020, Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm, 11 (7), 1501-1529.

Omran, M. G. ve Al-Sharhan, S., 2008, Using opposition-based learning to improve the performance of particle swarm optimization, *2008 IEEE Swarm Intelligence Symposium*, 1-6.

Papadimitriou, C. H. ve Steiglitz, K., 1998, Combinatorial optimization: algorithms and complexity, Courier Corporation, p.

Porkodi, S., Srihari, S. ve Vijayakumar, N., 2022, Talent management by predicting employee attrition using enhanced weighted forest optimization algorithm with improved random forest classifier.

Price, K. V., 2013, Differential evolution, In: Handbook of optimization, Eds: Springer, p. 187-214.

Rahnamayan, S., Tizhoosh, H. R. ve Salama, M. M. J. I. T. o. E. c., 2008, Opposition-based differential evolution, 12 (1), 64-79.

Shekhawat, S. ve Saxena, A. J. I. t., 2020, Development and applications of an intelligent crow search algorithm based on opposition based learning, 99, 210-230.

Storn, R. ve Price, K. J. J. o. g. o., 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, 11 (4), 341-359.

Taghian, S., Nadimi-Shahraki, M. H. ve Zamani, H., 2018, Comparative analysis of transfer function-based binary Metaheuristic algorithms for feature selection, *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 1-6.

Taghian, S. ve Nadimi-Shahraki, M. H. J. I. J. C. S. E., 2019, A binary metaheuristic algorithm for wrapper feature selection, 8, 168-172.

Talbi, E.-G., 2009, Metaheuristics: from design to implementation, John Wiley & Sons, p.

Tizhoosh, H. R., 2005, Opposition-based learning: a new scheme for machine intelligence, *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, 695-701.

Too, J., Abdullah, A. R. ve Mohd Saad, N. J. A., 2019, Hybrid binary particle swarm optimization differential evolution-based feature selection for EMG signals classification, 8 (3), 79.

Tubishat, M., Idris, N., Shuib, L., Abushariah, M. A. ve Mirjalili, S. J. E. S. w. A., 2020, Improved Salp Swarm Algorithm based on opposition based learning and novel local search algorithm for feature selection, 145, 113122.

Wang, G.-G., Feng, Y., Dong, J., Wang, L. J. C. ve Engineering, E., 2018, Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem, 67, 454-468.

Xie, Q., Cheng, G., Zhang, X., Peng, L. J. I. T. ve Control, 2020, Feature selection using improved forest optimization algorithm, 49 (2), 289-301.

Xiong, S. J. E. S. w. A., Abd Elaziz, M. ve Oliva, D., 2017, An improved opposition-based sine cosine algorithm for global optimization, 90, 484-500.

Yang, J. ve Honavar, V., 1998, Feature subset selection using a genetic algorithm, In: Feature extraction, construction and selection, Eds: Springer, p. 117-136.

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

Yu, X., Xu, W. ve Li, C. J. K.-B. S., 2021, Opposition-based learning grey wolf optimizer for global optimization, 226, 107139.

Zhu, Z., Ong, Y.-S., Dash, M. J. I. T. o. S., Man, ve Cybernetics, P. B., 2007, Wrapper–filter feature selection algorithm using a memetic framework, 37 (1), 70-76.

Zorarpacı, E. ve Özel, S. A. J. E. S. w. A., 2016, A hybrid approach of differential evolution and artificial bee colony for feature selection, 62, 91-103.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Büşra YILDIRIM
Uyruğu : T.C.
Doğum Yeri ve Tarihi :
e-mail :

EĞİTİM

Derece	Adı, İlçe, İl	Bitirme Yılı

UZMANLIK ALANI

Optimizasyon, Meta-Sezgisel Algoritmalar, Makine Öğrenmesi

YABANCI DİLLER

İngilizce

YAYINLAR

Yıldırım B. ve Altun A. A., 2022, Öznitelik Seçimi Problemleri İçin Karşıtlık Tabanlı Orman Optimizasyonu Algoritması, *15. Uluslararası Bilimsel Araştırmalar Kongresi, UBAK-2022,* Ankara, 101.