KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

YAPAY ÇEKİRGE SÜRÜ OPTİMİZASYONU

DOKTORA TEZİ

Eda ÖZKUL

MART 2022
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK VE BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

YAPAY ÇEKİRGE SÜRÜ OPTİMİZASYONU

Eda ÖZKUL

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"DOKTOR (İSTATİSTİK)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 16 / 02 /2022
Tezin Savunma Tarihi : 18 / 03 /2022

Tez Danışmanı : Doç. Dr. Orhan KESEMEN

Trabzon 2022

ÖNSÖZ

"Yapay Çekirge Sürü Optimizasyonu" isimli bu tez Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü İstatistik ve Bilgisayar Bilimleri Anabilim Dalı, Doktora Programı'nda hazırlanmıştır.

Tez çalışmam boyunca hiçbir desteğini esirgemeyen ve bilim insanı olma yolundaki ilk adımlarımı atarken kıymetli bilgileriyle yolumu aydınlatan danışman hocam Sayın Doç. Dr. Orhan KESEMEN'e teşekkürü bir borç bilirim. Ayrıca, yapıcı eleştirileri ve önerileri ile tezime büyük katkıda bulunan saygıdeğer hocalarım Sayın Prof. Dr. Mualla YALÇINKAYA'ya ve Sayın Dr. Öğr. Üyesi Uğur ŞEVİK'e, eğitim öğretim hayatımda katkısı olan tüm hocalarıma ve ortak olarak çalışmalar yürüttüğümüz ve tez çalışması sürecinde de hiçbir yardımdan kaçınmayan arkadaşlarım Arş. Gör. Özge TEZEL'e ve Arş. Gör. Dr. Buğra Kaan TİRYAKİ'ye teşekkür ederim.

Son olarak, hayatım boyunca aldığım tüm kararlarda bilgi ve tecrübelerinden faydalandığım, yaşadığım en zor ve sıkıntılı zamanlarda dahi beni bir an olsun yalnız bırakmayan, maddi ve manevi desteklerini hiçbir zaman esirgemeyen ve bugünlere gelmemde en büyük paya sahip olan annem ve babam Gönül-Ahmet ÖZKUL'a, kardeşim İsmail ÖZKUL'a teşekkür eder, minnettarlığımı sunarım.

Bu tez çalışmasının bundan sonraki çalışmalara katkı sağlamasını temenni ederim.

Eda ÖZKUL

Trabzon 2022

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum “Yapay Çekirge Sürü Optimizasyonu” başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. Orhan KESEMEN’in sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 18/03/2022

Eda ÖZKUL

İÇİNDEKİLER

Sayfa No

Doktora Tezi

ÖZET

YAPAY ÇEKİRGE SÜRÜ OPTİMİZASYONU

Eda ÖZKUL

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
İstatistik ve Bilgisayar Bilimleri Anabilim Dalı
Danışman: Doç. Dr. Orhan KESEMEN
2022, 108 Sayfa

Optimizasyon genel olarak en iyiyi bulma sürecidir ve belirli bir arama uzayında tanımlanan bir probleme kabul edilebilir bir çözüm bulmayı amaçlar. Yapay zekânın gelişmesiyle birlikte, optimizasyon problemlerinin çözümü için meta-sezgisel algoritmalar popüler hale gelmiştir. Meta-sezgisel algoritmalar, özellikle doğadaki biyolojik sistemlerden ve sürü davranışından esinlenir. Sadece bir yöntemin bütün optimizasyon problemlerini çözmesi mümkün olmadığından, literatürde birçok meta-sezgisel algoritma geliştirilmiş ve geliştirilmeye devam etmektedir. Bu bağlamda, bu tez çalışmasında, zor optimizasyon problemlerinin çözümü için çekirge sürülerinin rastgele zıplama ve bitkileri istila etme davranışlarından esinlenerek yapay çekirge sürü optimizasyonu (ALSO) adı verilen sürü zekâsı tabanlı meta-sezgisel bir algoritma önerilmiştir. Çekirgeler yiyecek arama için aile ve sosyal olma üzere iki fazda birbirleriyle etkileşimde bulunurlar. Ailesel fazda, yerel bir alanda çekirgeler küçük gruplar halinde yiyecek arar ve sosyal fazda topladıkları bilgileri paylaşırlar. Önerilen algoritma, 22 test fonksiyonu ve 3 mühendislik tasarım problemi üzerinde test edilerek yaygın ve yeni geliştirilen optimizasyon algoritmaları ile karşılaştırılmıştır. Simülasyon sonuçları, diğer algoritmalarla kıyaslandığında önerilen ALSO algoritmasının oldukça rekabetçi olduğunu kanıtlamaktadır. Ayrıca, aynı koşullar altında, ALSO algoritması daha az çalışma zamanı ve bellek alanı gerektirmektedir.

Anahtar Kelimeler: Optimizasyon, Sürü Zekâsı, Meta-sezgisel Algoritmalar, Yapay Çekirge Sürü Optimizasyonu

PhD. Thesis

SUMMARY

ARTIFICIAL LOCUST SWARM OPTIMIZATION

Eda ÖZKUL

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Statistics and Computer Sciences Graduate Program
Supervisor: Assoc. Prof. Orhan KESEMEN
2022, 108 Pages

Optimization is generally the process of finding the best and aims to find an acceptable solution to a problem defined over a given search space. As the artificial intelligence has developed, meta-heuristic algorithms have become popular for solving optimization problems. Meta-heuristic algorithms are particularly inspired by biological systems and swarm behavior in nature. Since it is not possible for just one method to solve all optimization problems, many meta-heuristic algorithms have been developed and continue to be developed in the literature. In this regard, in this thesis, a new swarm intelligence based meta-heuristic algorithm called Artificial Locust Swarm Optimization (ALSO) is proposed inspiring random jumping and plant invasion behavior of locust swarms. Locusts interact in two different ways of searching for food: social and familial. In the familial phase, small locust groups search foods in a local area and the locusts share their information in the social phase. The proposed algorithm is tested on 22 benchmark functions and 3 engineering design problems and compared with other recent and well-known optimization algorithms. Simulation results prove that the proposed ALSO algorithm is quite competitive when compared to the other algorithms. Moreover, it even requires the less runtime and memory space under the same conditions.

Key Words: Optimization, Swarm intelligence, Metaheuristic, Artificial Locust Swarm Optimization

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

ALSO : Yapay Çekirge Sürü Optimizasyonu
GA : Genetik Algoritma
PSO : Parçacık Sürü Optimizasyonu
ABC : Yapay Arı Koloni Algoritması
BAT : Yarasa Algoritması
GWO : Gri Kurt Optimizasyon Algoritması
WOA : Balina Optimizasyon Algoritması
HHO : Harris Şahini Optimizasyonu
RSO : Fare Sürüsü Optimizasyonu
BRO : Battle Royal Optimizasyonu
CSA : Karga Arama Algoritması
NFL : No-Free-Lunch

1. GENEL BİLGİLER

Optimizasyon, verilen koşullar altında, belirli bir problem için mevcut çözümler arasından en iyi olanının belirlenmesi sürecidir. Optimizasyonun amacı, uygun bir uzayda tanımlı değişkenlere karşılık gelen bir amaç fonksiyonunun maksimum veya minimum değerini aramaktır. Matematik, fizik, bilgisayar bilimleri, mühendislik, tıp, ekonomi, işletme, endüstri gibi çeşitli alanlarda optimizasyon uygulanmaktadır (Hassanien ve Emary, 2016; Jain vd., 2019; Vasuki, 2020).

Genel olarak optimizasyon, belirli kısıtlar altında, amaç fonksiyonuna en iyi değeri veren değişkenlerin kümesinin bulunması olarak ifade edilebilir.

Optimizasyon problemleri matematiksel olarak,

$$\min_{x} f(\vec{x}) \quad \vec{x} = (x_1, \dots, x_d) \in F \subseteq S \subseteq \mathbb{R}^d$$
$$g_i(\vec{x}) \leq 0 \qquad i = 1, .., q$$
$$h_j(\vec{x}) = 0 \qquad j = 1, \dots, m$$

biçiminde tanımlanabilir. Burada, $f(\vec{x})$, problemin amaç fonksiyonunu; $\vec{x} = (x_1, \dots, x_d)$, çözüm vektörünü; F, uygun çözüm alanını; S, arama uzayını; $g_i(\vec{x})$ ve $h_j(\vec{x})$ sırasıyla eşitsizlik ve eşitlik kısıtlarını ifade etmektedir. $\vec{x}$ vektörü, bütün kısıtları sağlıyorsa uygun çözüm olarak adlandırılır ve tüm uygun çözümler uygun çözüm alanını oluşturur. Yani, problemin kısıtlarını sağlayan karar değişkenlerinin kümesi uygun çözüm alanı olarak ifade edilir. Hem eşitlik hem de eşitsizlik kısıtları doğrusal veya doğrusal olmayan bir yapıda olabilir.

Bir optimizasyon probleminin çözümü, açıkça belirtilmiş hedeflerle problemin tanımlanmasıyla başlar. Problem ile ilgili kısıtlar ve parametrelerin açık bir şekilde belirtilmesi gerekir. Ayrıca, fonksiyonların başlangıç değerleri, parametrelerin sınırları, problemin amacı ve kısıtları matematiksel olarak ifade edilmelidir.

Optimizasyon problemlerinde, genellikle maliyetin, enerjinin ve kullanılan kaynakların minimizasyonu veya kar ve kalitenin maksimizasyonu amaçlanmaktadır. Kısaca, amaç fonksiyonu ulaşılan çözümün kalitesinin bir göstergesidir ve amaç fonksiyonunun en iyi değeri tüm kısıtları sağlayan karar değişkenleri seçilerek elde edilir. Bu bağlamda, problemin amaç fonksiyonu minimizasyon veya maksimizasyon fonksiyonu olarak belirlenir. Eğer amaç fonksiyonu maksimize edilecekse buna kalite fonksiyonu,

minimize edilecekse buna maliyet fonksiyonu adı verilir. Ayrıca, problemin mümkün çözümleri, değişkenler kümesine karşılık gelen amaç fonksiyonunun değeri ile ölçülür.

Karar değişkenleri amaç fonksiyonunun değerini belirler ve her optimizasyon probleminde amaç fonksiyonuna en iyi veya optimal değeri veren karar değişkenleri belirlenmeye çalışılır.

Optimizasyon problemleri karar değişkenlerinin türüne göre sürekli ve kesikli optimizasyon problemi olmak üzere ikiye ayrılır. Karar değişkenlerinin sonlu bir aralıkta sonsuz farklı değer aldığı problemlere sürekli, sonlu bir aralıkta sonlu farklı değer aldığı problemlere ise kesikli optimizasyon problemleri adı verilir (Bozorg-Haddad vd., 2017).

Optimizasyon problemlerinin boyutu karar değişkenlerinin sayısı ile ifade edilir. Sadece bir tane karar değişkeni varsa optimizasyon probleminin tek boyutlu, birden fazla karar değişkeninin olduğu durumda ise optimizasyon probleminin çok boyutlu olduğu söylenebilir (Vasuki, 2020).

Amaç fonksiyonunun sayısına göre optimizasyon problemleri ikiye ayrılır. Optimizasyon problemi, sadece bir tane amaç fonksiyonuna sahipse tek amaçlı, birden fazla amaç fonksiyonuna sahipse çok amaçlı optimizasyon problemi olarak adlandırılır. Tek amaçlı optimizasyon problemlerinde, amaç fonksiyonuna en iyi değeri veren karar değişkenlerine ait tek bir çözüm vardır. Çok amaçlı optimizasyon problemlerinde ise, bir çözüm kümesi bulunmaya çalışılır. Gerçek hayatta çoğu optimizasyon problemi çok amaçlı optimizasyon problemidir (Yang, 2014; Vasuki, 2020).

Optimizasyon problemleri, içerdiği kısıt sayısına göre kısıtsız ve kısıtlı olmak üzere ikiye ayrılır. Eğer optimizasyon probleminde herhangi bir kısıt yoksa problem kısıtsız optimizasyon problemi olarak adlandırılır. Eğer problemle ilişkili kısıtlar varsa problem kısıtlı optimizasyon problemi olarak tanımlanır (Vasuki, 2020).

Optimizasyon problemlerinde amaç fonksiyonları ve kısıtlar lineer veya lineer olmayan bir yapıda olabilir. Bu durumda, optimizasyon problemi, amaç fonksiyonu ve kısıtlar lineer ise lineer optimizasyon problemi, bunlardan herhangi biri lineer değil ise lineer olmayan optimizasyon problemi olarak adlandırılır (Yang, 2014).

Optimizasyon yöntemleri, genel olarak deterministik ve stokastik algoritmalar olmak üzere iki grupta incelenir. Deterministik optimizasyon algoritmaları, herhangi bir rastgelelik içermeyen yöntemlerdir. Başka bir ifadeyle, bu yöntemler, aynı başlangıç değerleri kullanılarak algoritma çalıştırıldığında her zaman aynı sonucu üretir. Stokastik optimizasyon algoritmaları ise rastgelelik içeren yöntemlerdir ve algoritmada aynı başlangıç değerleri

kullanılsa bile her zaman farklı sonuçlar elde edilir. Genellikle meta-sezgisel yöntemler olarak adlandırılan stokastik optimizasyon algoritmalarının amacı, başlangıçta üretilen rastgele çözümler yardımıyla arama uzayı keşfedilerek global çözüme ulaşmaktır (Yang, 2010; Nanda ve Panda, 2014; Hassanien ve Emary, 2016; Vasuki, 2020; Mishra ve Shinde, 2021).

1.1. Deterministik Algoritmalar

Deterministik algoritmalar, tutarlı optimizasyon algoritmalarıdır. Yani, algoritma aynı girdi parametreleriyle her çalıştırıldığında her zaman aynı sonucu üretir. Çoğu deterministik algoritma, türevlenebilir fonksiyonlara uygulanabilir. Bu nedenle, amaç fonksiyonunun türevlenebilir ve sürekli olması gerekir. Problemin maksimum veya minimum değerini bulmak için amaç fonksiyonunun birinci ve ikinci dereceden türevleri hesaplanır. Rastgelelik içermeyen matematiksel bir formülasyona dayalı olduğundan deterministik optimizasyon sürecinde elde edilen sonuçlar kesindir ve tekrarlanabilir (Cavazzuti, 2013; Sharma ve Kaushik, 2021).

Deterministik yöntemler, genellikle gradyan tabanlı yöntemlerdir ve sürekli uzaydaki optimizasyon fonksiyonlarına kesin çözümler sunarlar. Her ne kadar problemlere kesin çözüm sağlasalar da deterministik algoritmaların ulaştığı çözüm global optimum değil yerel bir optimum olabilir.

Deterministik algoritmaların etkili olabilmesi için, amaç fonksiyonunun ikinci türevinin alınabilmesi ve tek tepeli olması gerekir. Dolayısıyla, bu yöntemler sadece birkaç tür optimizasyon problemi için uygulanabilir (Cuevas vd., 2018).

1.2. Meta-sezgisel Algoritmalar

Günümüzde bilgisayar teknolojilerindeki hızlı gelişmeler sayesinde makineler, insanların yaptığı birçok işi daha hızlı yapabilmektedir. Bununla birlikte, insanlara özgü olan düşünme, karar verme, öğrenme, akıl yürütme, algılama gibi özelliklerin makinelere aktarılabilmesine ilişkin çalışmaların başlaması yapay zekâ kavramını ortaya çıkarmıştır. Buna paralel olarak bilim adamları, insanın düşünme yeteneğinin yanı sıra doğadaki canlıların davranış biçimlerinden esinlenerek de yeni yöntemler geliştirmektedir. Uzman sistemler, yapay sinir ağları, makine öğrenmesi, optimizasyon yapay zekânın dallarını

oluşturmaktadır. Özellikle son yıllarda, karmaşık problemlerin çözümü için doğal yaşamdan esinlenerek geliştirilen yöntemler, meta-sezgisel olarak adlandırılan zeki optimizasyon algoritmalarına olan ilgiyi arttırmıştır. Glover (1986) tarafından ortaya atılan meta-sezgisel terimi, üst düzey anlamına gelen yunanca ön ek "meta" ile keşfetme, bulma anlamına gelen "heuriskein veya euriskein" kelimelerinin birleşmesinden oluşmaktadır.

Meta-sezgisel algoritmalar çok çeşitli problemlere uygulanabilen problemden bağımsız yöntemlerdir. Bu algoritmalar, optimizasyon problemlerinin karar değişkenlerine yaklaşık değer bulmayı amaçlamaktadır. Böylece, amaç fonksiyonunu en iyilemeye çalışır (Bozorg-Haddad vd., 2017). Son yıllarda, meta-sezgisel algoritmalar, optimizasyon problemlerinin çözümü için sıklıkla kullanılmaktadır. Basit kavramlara dayanması, yerel optimal değerlere takılmasının az olması ve farklı disiplinlerdeki birçok probleme kolaylıkla uygulanabilmesi meta-sezgisel algoritmalara olan ilgiyi artırmıştır (Darwish, 2018).

Meta-sezgisel yöntemler, genel olarak biyolojik, doğal veya sosyal bir sistemi soyut olarak simüle eden algoritmalardır. Daha açık bir şekilde ifade etmek gerekirse, meta-sezgisel algoritmalar, bir veya daha fazla aday çözümü temsil eden ajan (birey) veya ajanların (bireylerin) oluşturduğu bir popülasyon içerir ve bu popülasyon yeni aday çözümlerin üretilmesiyle dinamik olarak değişir. Meta-sezgisel optimizasyon algoritmaları, başlangıçtan itibaren değerlendirilen aday çözümlerden elde edilen bilgileri kullanarak optimizasyon probleminin yapısı hakkında bilgi edinir ve bu bilgiyi daha iyi aday çözümler oluşturmak için kullanır. Bir çözümün hayatta kalma ve üreme olasılığı ise uygunluk fonksiyonu ile belirlenir. Aday çözümler, nesiller boyunca uygunluklarını, yani optimizasyon problemlerini çözme yeteneklerini geliştirir. Böylece, yinelemeli yaklaşımlarla rastgele seçilen bir veya daha fazla başlangıç çözümü için daha iyi bir çözüm elde etmeye çalışır. Dolayısıyla, optimize edilecek amaç fonksiyonu ile ilişkili olan aday çözümlerin iyileştirilmesi sayesinde global çözüme yaklaşılır (Cuevas vd., 2018).

Deterministik algoritmalarla karşılaştırıldığında, meta-sezgisel algoritmaların hem avantajları hem de dezavantajları vardır. Meta-sezgisel algoritmalar, matematiksel olarak daha az karmaşıktır ve arama stratejisi rastgelelik içerir. Optimal çözüme yaklaştıkça daha yavaş bir yakınsamaya sahiptirler. Bununla birlikte, arama alanında daha kapsamlı bir araştırma yapabilirler ve böylece yerel optimumlara takılmadan global bir çözüme ulaşma imkânı sağlarlar. Aynı zamanda, yerel optimum noktaların üstesinden gelme yetenekleri global optimumu bulma olasılıklarını arttırarak, yöntemin sağlamlığını geliştirirler

(Cavazzuti, 2013). Ayrıca, deterministik optimizasyon algoritmaları problemin yapısına bağlı iken, meta-sezgisel algoritmalar problemden bağımsızdır (Kumar vd., 2020).

Geliştirilen her meta-sezgisel algoritma, belirli bir optimizasyon problemini çözerken başarıya ulaşmak için keşif ve sömürü ile adlandırılan iki ana bileşen üzerinde durur. Keşif, bir arama algoritmasının, arama uzayının farklı bölgelerine yayılmış farklı çözümleri keşfetme yeteneğini ifade eder. Sömürü, daha iyi çözümler bulmak veya mevcut çözümleri iyileştirmek amacıyla, iyi bir çözümün etrafında aramaya yoğunlaşmaktır. Keşif, algoritmanın mevcut çözümlerden uzakta yeni çözümler üreterek global bir arama yapmasını sağlar. Bunun avantajı, algoritmanın yerel bölgede takılma olasılığının düşük ve global optimal çözümün daha ulaşılabilir olmasıdır. Ancak, üretilen yeni çözümlerin global optimum değerden daha uzak olması da söz konusu olabilir. Bu durum ise, algoritmanın optimal çözüme yakınsamasını yavaşlatabilir. Sömürü, algoritmanın yerel bir arama yapmasını sağlar. Genellikle, sömürü optimal çözüme yüksek yakınsama oranı ile ulaşılmasına yardımcı olur. Ancak, elde edilen son çözüm, başlangıç noktasına bağlı olduğu için algoritmanın yerel çözümlere takılmasına neden olabilir (Yang, 2014; Morales-Castañeda vd., 2020). Bu durumda, algoritmanın performansını arttırmak için geliştirilen her meta-sezgisel algoritma keşif ve sömürü arasında bir denge kurmalıdır. Keşif işlemine çok az sömürü işlemine çok fazla odaklanıldığı zaman, algoritma daha hızlı yakınsar. Ancak, bu durumda doğru global optimumu bulma olasılığı daha düşük olabilir. Benzer şekilde, keşif işlemine daha çok, sömürü işlemine daha az yoğunlaşıldığında, daha yavaş bir yakınsamaya neden olabilir. Dolayısıyla, keşif ve sömürü arasındaki denge oldukça önemlidir (Yang vd., 2014). Bununla birlikte, meta-sezgisel bir algoritmanın optimal çözümü bulma yeteneği, arama uzayında keşif ve sömürü süreçlerini dengeleme kapasitesine bağlıdır (Cuevas ve Rodríguez, 2020). Keşif ve sömürü arasında uygun bir denge kurulması çok önemli olmasına rağmen, bu denge için pratik bir kılavuz yoktur. Bununla birlikte, keşif ve sömürü arasındaki dengeyi sağlamanın yolunu bulmak hala açık bir problemdir. Bu nedenle, meta-sezgisel algoritmaların ortak bir çerçevesi olan keşif ve sömürü dengesini her algoritma genellikle farklı operatörler kullanarak sağlamaya çalışır (Rashedi vd., 2009; Yang, 2014; Cuevas ve Rodríguez, 2020).

Optimizasyon sürecinde, probleme uygun algoritmanın seçimi oldukça önemli bir konudur. Bununla birlikte, tüm mümkün çözümlerin bulunması zor olan bazı karmaşık optimizasyon problemleri vardır (Darwish, 2018). Literatürde, karmaşık ve zor optimizasyon problemlerinin çözümü için son yıllarda birçok algoritma geliştirilmiştir. Bu

algoritmalar, doğal yaşamdaki canlıların evrimsel süreçleri veya yaşam tarzları temel alınarak modellenmiştir (Meetei, 2014; Yang, 2014; Hassanien ve Emary, 2016). Bu bağlamda, meta-sezgisel algoritmalar, farklı hayvan türlerinin davranışlarından, doğadaki canlıların fiziksel, kimyasal ve biyolojik süreçlerinden esinlenerek geliştirilmiştir (Fister Jr ve Fister, 2021). Ayrıca, meta-sezgisel algoritmalar, optimizasyon problemlerinin çözümünde basit, hızlı ve etkin sonuçlar ürettiği için araştırmacılar tarafından geniş bir uygulama yelpazesine sahiptir. Doğal yaşamdan esinlenerek geliştirilen optimizasyon algoritmaları görüntü işleme, veri analizi, makine öğrenmesi, tıp, finans, veri madenciliği gibi birçok alana uygulanmaktadır (Martens vd., 2011; Ahmed ve Glasgow, 2012).

İnsan beyni, zor ve karmaşık problemleri çözmek için müthiş bir sezgisel yapıya sahiptir. Tarih boyunca insanoğlu, birçok problemi çözmek için bu sezgisellikten yararlanmıştır. Son yıllarda, özellikle zor ve karmaşık optimizasyon problemlerine çözüm üretmek için meta-sezgisel yöntemlerin kullanılması yaygınlaşmış ve bu kavramların ortaya çıkmasından bu yana çok büyük ilerlemeler kaydedilmiştir. Buna rağmen, fizik, kimya ve matematik gibi çalışma alanlarıyla karşılaştırıldığında meta-sezgisel yöntemlerin henüz tam olarak olgunlaştığı söylenemez (Sörensen, 2018).

Meta-sezgisel yöntemlerin ilk olarak ne zaman kullanıldığını kestirmek zordur. Ancak, bu yöntemlerin kullanıldığı ilk çalışmaların 1940'lı yıllara dayandığı söylenebilir (Nayyar ve Nguyen, 2018; Sörensen, 2018). İkinci Dünya Savaşı sırasında, Matematikçi Alan Turing Enigma şifrelerini kırarken kullandığı yöntemi sezgisel arama olarak adlandırmış ve daha sonra otomatik hesaplama motorunu ortaya atmıştır (Yang, 2014).

1960 ve 1970'li yıllar meta-sezgisel yöntemlerin gelişiminde büyük bir rol oynamıştır. Bu dönemde, Ingo Rechenberg ve Hans-Paul Schwefel uzay mühendisliğindeki optimizasyon problemlerini çözmek için evrimsel strateji adı verilen bir arama yöntemi geliştirmiştir (Yang, 2014; Osaba ve Yang, 2021). Bununla birlikte, bir diğer evrimsel çalışma olan genetik algoritmalar John Holland ve arkadaşları tarafından önerilmiştir (Holland, 1975). Genetik algoritmalar, literatürde büyük bir etki yaratmış ve doğadan esinlenerek geliştirilen yöntemlerin temelini oluşturmuştur (Yang, 2014; Sörensen, 2018; Osaba ve Yang, 2021).

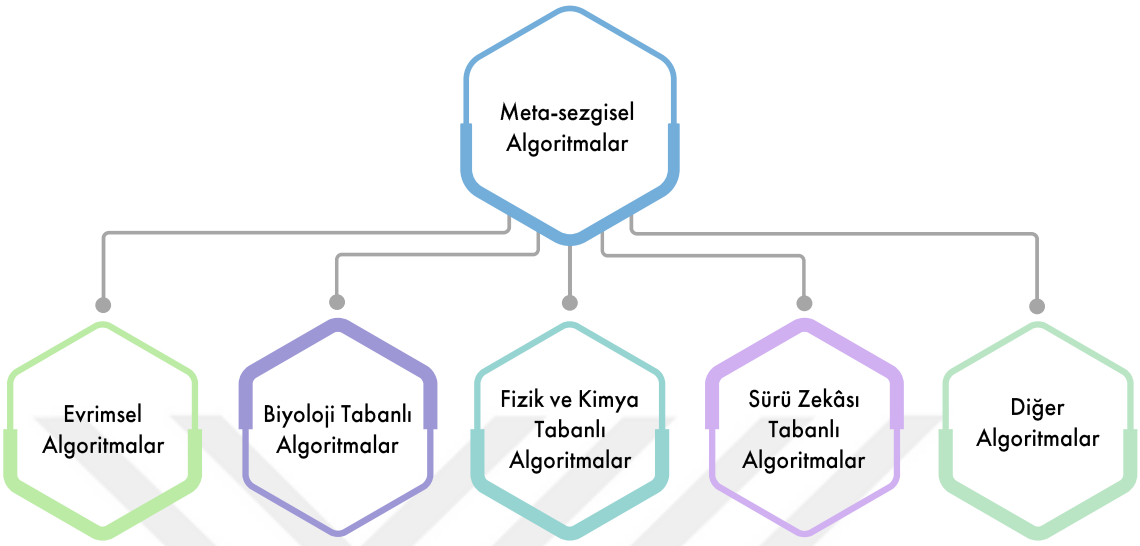
Metallerin tavlanması sürecinden esinlenerek ortaya atılan benzetimli tavlama algoritması (Kirkpatrick vd., 1983) meta-sezgisel algoritmalara öncü bir katkı sağlamıştır (Yang, 2014; Boussaïd, 2016). Daha sonra, 1986'da tabu araması (Glover, 1986) ve yapay bağışıklık sistemi (Farmer vd., 1986) önerilmiştir. Dorigo (1992) karıncaların birbirleriyle

iletişim kurmak ve yuvalarıyla besin kaynağı arasındaki en uygun güzergahı bulmak için feromon salgılama biçiminden ilham alarak karınca koloni optimizasyonunu geliştirmiştir. Walker vd. (1993) arı kolonilerini temel alan ilk çalışmayı yapmıştır. Eberhart ve Kennedy (1995), kuş veya balık sürülerinin davranışlarını modelleyen parçacık sürü optimizasyonunu önermiştir. Karınca koloni ve parçacık sürü optimizasyonunun ortaya çıkması, meta-sezgisel yöntemlere olan ilginin artmasını sağlamış ve böylece bu çalışmalar meta-sezgisel yöntemler için bir dönüm noktası olmuştur (Yang, 2014; Boussaïd, 2016). Storn ve Price (1997) diferansiyel gelişim adı verilen vektör tabanlı bir evrimsel algoritma geliştirmiştir. Bunların yanı sıra, müzisyenlerin doğaçlama yaparak en iyi melodiyi bulma süreçlerinden esinlenerek harmoni arama algoritması önerilmiştir (Geem vd., 2001). Passino (2002) Escherichia Coli (E. coli) bakterilerinin sosyal yiyecek arama davranışına dayanan bakteriyel besin arama optimizasyonunu ortaya koymuştur. Nakrani ve Tovey (2004), bal arılarından ilham alarak bal arısı algoritmasını geliştirmiştir (Yang, 2014). Daha sonra, Karaboğa (2005), bal arılarının yiyecek arama davranışlarını temel alan yapay arı koloni algoritmasını önermiştir. Rashedi vd. (2009), yer çekimi kanununa dayanan yerçekimsel arama algoritmasını sunmuştur. Aynı dönemde, Yang (2009) ateş böceklerinin parlaklıklarını ve hareket yönlerini modelleyerek ateş böceği algoritmasını geliştirmiştir. Bununla birlikte, Yang ve Deb (2009), guguk kuşlarının yumurtalarını başka yuvalara bırakmasından esinlenerek guguk kuşu arama algoritmasını ortaya atmıştır. Ayrıca, Yang (2010) yarasaların ekolokasyon davranışına dayanan yarasa algoritmasını geliştirmiştir. Bunların yanı sıra, Yang (2012) çiçeklerin tozlaşma mekanizmalarına dayanan çiçek tozlaşma algoritmasını önermiştir. Daha sonra, kimyasal reaksiyonların türlerinden ve oluşmasından ilham alınarak yapay kimyasal reaksiyon optimizasyon algoritması geliştirilmiştir (Alatas, 2011). Eskandar (2012) tarafından önerilen su döngüsü algoritması, suyun yağmurla atmosferden yeryüzüne, nehir ve akarsularla deniz veya okyanuslara daha sonra buharlaşarak tekrar atmosfere ulaşma sürecine dayanmaktadır. Gandomi ve Alavi (2012) belirli biyolojik veya çevresel süreçlere tepki olarak kril sürülerinin toplanmasından esinlenerek kril sürüsü optimizasyon algoritmasını geliştirmiştir. Cuevas vd. (2013) biyolojik yasalara dayanarak birbirleriyle etkileşime giren sosyal örümceklerin iş birlikçi davranışlarını modelleyerek sosyal örümcek algoritmasını önermiştir. Kaveh ve Farhoudi (2013) tarafından önerilen yunus ekolokasyon algoritması, yunusların avlanma süreçlerinde kullandıkları ekolokasyonu taklit etmektedir. Mirjalili vd. (2014) gri kurtların avlanma mekanizmasını modelleyen gri kurt optimizasyonunu geliştirmiştir. Ayrıca, Mirjalili (2015) tarafından önerilen karınca aslanı

algoritması, doğadaki karınca aslanlarının avlanma mekanizmasını ilham almıştır. Bunların yanı sıra, Mirjalili ve Lewis (2016) balinaların avlanma stratejisine dayanan balina optimizasyon algoritmasını sunmuştur. Askarzadeh (2016) kargaların fazla yiyeceklerini saklama ve gerektiğinde geri alma fikrinden yola çıkarak karga sürüsü algoritmasını önermiştir. Dhiman ve Kumar (2017) benekli sırtlanların avlanma davranışlarından esinlenerek benekli sırtlan optimizasyonunu geliştirmiştir. Saremi vd. (2017) çekirge sürülerinin davranışını matematiksel olarak modelleyen ve taklit eden çekirge sürü optimizasyonunu önermiştir. Mirjalili vd. (2017) tarafından geliştirilen salp sürüsü algoritması, salp sürülerinin okyanusta yiyecek arama davranışlarını taklit etmektedir. Jain vd. (2018) baykuşların, karanlıkta görme duyuları yerine işitme yeteneklerine dayanan avlanma mekanizmasını modelleyen baykuş arama algoritmasını geliştirmiştir. İmparator penguen optimizasyonu (Dhiman ve Kumar, 2018) ve imparator penguen kolonisi (Harifi vd., 2019), imparator penguenlerin toplanma davranışlarını taklit eden algoritmalardır. İmparator penguen optimizasyonu, penguenlerin toplanma etrafındaki sıcaklık profilini modellerken, imparator penguen kolonisi her penguenin vücut ısısını ve çekiciliklerini temel almıştır (Dhiman ve Kumar, 2018; Harifi vd., 2019). Jain vd. (2019), uçan sincapların yiyecek arama ve süzülerek uçma davranışlarını modelleyerek sincap arama algoritmasını önermiştir. Xue ve Shen (2020), serçelerin, yiyecek arama davranışlarından ve düşmanlarını tespit ettiklerinde sergiledikleri hareketlerden ilham alarak serçe arama algoritmasını ortaya atmıştır. Chou ve Truong (2021) okyanustaki denizanalarının davranışlarını temel alarak yapay denizanası arama optimizasyonunu geliştirmiştir. Hashim vd. (2021) Arşimet prensibinden yola çıkarak Arşimet optimizasyon algoritmasını tasarlamıştır. Dhiman vd. (2021) ortaya atılan fare sürüsü optimizasyonu, farelerin doğadaki kovalama ve saldırma davranışlarını matematiksel olarak modellemiştir. Farshi (2021) savaş ve strateji oyunlarından esinlenerek Battle Royal optimizasyon algoritmasını geliştirmiştir. Abualigah vd. (2022) tarafından önerilen sürüngen arama algoritması, timsahların avlanma davranışları ilham alınarak modellenmiştir. Hashim vd. (2022), bal porsuğu algoritmasını, bal porsuklarının tünel kazma ve bal arama davranışının matematiksel modeline dayanarak ortaya atmıştır.

Esinlenme kaynaklarına göre meta-sezgisel optimizasyon algoritmaları evrimsel algoritmalar, biyoloji tabanlı algoritmalar, sürü zekâsı tabanlı algoritmalar, fizik ve kimya tabanlı algoritmalar ve diğer algoritmalar olmak üzere beş kategoride incelenebilir (Fister Jr

vd., 2013; Dhiman ve Kumar, 2017; Kumar ve Bawa, 2020). Şekil 1'de meta-sezgisel algoritmaların sınıflandırılması verilmiştir.



Şekil 1. Meta-sezgisel algoritmaların sınıflandırılması

1.2.1. Evrimsel Algoritmalar

Darwin'in evrim teorisini ilham alan evrimsel algoritmalar üreme, mutasyon, çaprazlama, doğal seçilim gibi evrimsel süreçleri modellemektedir. Bu algoritmalar, belirli bir popülasyondaki en uygun bireyin hayatta kalmasını temel almaktadır (Dhiman vd., 2021). Evrimsel hesaplamaların tarihi 1960'lara dayanmaktadır. Bu dönemde yapılan çalışmalar, şu an kullanılan birçok algoritmaya yol göstermiştir (de Castro, 2007; Vasuki, 2020). Evrimsel algoritmalar, uygunluk değerine göre hayatta kalmaya çalışan bir popülasyon ile başlar. Popülasyondaki ebeveynler, çaprazlama ve mutasyon gibi genetik operatörler yardımıyla bulundukları ortama uyum sağlaması için yavrularına özelliklerini aktarırlar. Bu süreç, tatmin edici bir çözüm bulunana kadar birkaç nesil boyunca devam eder (Nanda ve Panda, 2014). En yaygın evrimsel algoritmalara genetik algoritma (Holland, 1975), genetik programlama (Koza, 1990) ve diferansiyel gelişim algoritması (Storn ve Price, 1997) örnek verilebilir.

1.2.1.1. Genetik Algoritma

Genetik algoritma, 1975 yılında John Holland ve arkadaşları tarafından geliştirilmiştir (Holland, 1975). Darwin'in doğal seçilim teorisine dayanan genetik algoritma, biyolojik evrimin bir modelidir. Çaprazlama, mutasyon, üreme, seçilim gibi genetik operatörler, problem çözme stratejisi olarak genetik algoritmanın temelini oluşturmaktadır (Vasuki, 2020; Yang, 2014).

Genetik algoritmalar problemlere birden fazla rastgele çözüm üretir ve bu çözümler arasından en iyi çözümü bulmayı hedefler. Çözümlerin oluşturduğu kümeye popülasyon denir. Popülasyon, bireylerden (kromozom) oluşur. Bireyi oluşturan her bir elemana ise gen adı verilir. Popülasyon içindeki her birey bir çözümdür ve her bireyin kendisine ait bir amaç değeri vardır. Bu amaç değeri yardımıyla bir uygunluk değeri hesaplanır ve popülasyon içindeki her bireyin soyunu sürdürme olasılığı bu uygunluk değerinin büyüklüğü ile doğru orantılıdır.

Genetik algoritma, mümkün çözümlerden oluşan rastgele bir popülasyon üretilmesiyle başlar. Popülasyonun boyutu genel olarak probleme bağlıdır. Daha sonra, popülasyondaki her bir kromozomun uygunluk değeri hesaplanır ve uygunluk değerine göre seçim işlemi yapılır. Seçim operatörü, sonraki nesillerde popülasyon oluşturmak için hayatta kalan kromozomları tanımlayan üreme operatörünü belirtir. Darwin'in doğal seçilim teorisine dayanarak, uygunluk değeri yüksek olan kromozomlar daha uzun süre hayatta kalır. Yeni popülasyon oluştururken hangi kromozomların seçileceği rulet çarkı seçimi, sıralı seçim ve turnuva seçimi gibi yöntemlerle yapılır (Bozorg-Haddad vd., 2017; Feng vd., 2019).

Rulet çarkı seçiminde, her birey uygunluk değerine göre seçilir. Kromozomların uygunluk değeri ne kadar yüksek olursa seçilme şansları da o kadar yüksek olur. Bir kromozomun seçilme olasılığı,

$$P(x_k) = \frac{f(x_k)}{\sum_{i=1}^{N} f(x_i)} \tag{1}$$

biçiminde hesaplanır. Burada, $f(x_k)$, x_k çözümünün uygunluk fonksiyonunu ve $P(x_k)$, k. kromozomun seçilme olasılığını ifade eder.

Sıralı seçimde, kromozomlar uygunluk değerlerine göre sıralanır. En iyi uygunluk değerine sahip kromozom ilk sırada yer alırken, en kötü uygunluk değerine sahip olan kromozom son sırada yer alır (Bozorg-Haddad vd., 2017).

Turnuva seçimi yaklaşımında ise popülasyondan iki veya daha fazla kromozom rastgele olarak çekilir. Bu kromozomların uygunluk değerleri karşılaştırılır ve daha yüksek uygunluk değerine sahip olan kromozom seçilir. Bu süreç istenilen sayıda kromozom seçilene kadar devam eder. Bununla birlikte, karşılaştırılan kromozomlar seçim yapılan popülasyona geri gönderilip, tekrar seçilebilir (Feng vd., 2019).

Uygunluk değerine göre seçilen kromozomlar bir ebeveyn havuzu oluşturur. Ebeveynler, çaprazlama operatörüyle bir sonraki popülasyonun tamamını veya bir kısmını oluşturan yavrular üretir. Bu nedenle, bir sonraki popülasyon ebeveyn ve yavru popülasyonunun karışımı olabilir (Bozorg-Haddad vd., 2017). Çaprazlama işleminde rastgele seçilen ebeveynler arasında gen değişimi yapılarak yeni yavrular üretilir. Seçilen her iki ebeveynden iki yavru oluşur ve yavrular kendisini oluşturan ebeveynlerin özelliklerini taşır. Her yeni nesil bir önceki neslin en iyi bireylerinin çaprazlanmasıyla meydana gelir. Bu yüzden, yeni neslin bir önceki nesle göre daha iyi genleri taşıması beklenmektedir. Bununla birlikte, iterasyon ilerledikçe popülasyon içindeki bireyler birbirlerine benzemeye başlar. Bu durumda, popülasyonun çeşitliliğini artırmak için mutasyon işlemi uygulanır. Mutasyon işlemi, bir kromozomun bir veya daha fazla geninin değiştirilmesi ile gerçekleştirilir. Böylece, popülasyonun çeşitliliği sağlanır ve algoritmanın yerel çözümlere takılma olasılığı azalır (Feng vd., 2019). Tüm iterasyonlar sonucunda değişen nesil içindeki en iyi amaç değerine sahip kromozom çözüm olarak kabul edilir (Lim, 2014; Kesemen ve Özkul, 2018).

1.2.1.2. Diferansiyel Gelişim Algoritması

Diferansiyel gelişim algoritması, ilk olarak Storn ve Price tarafından geliştirilen, vektör tabanlı evrimsel bir algoritmadır (Storn ve Price, 1997). İşleyiş ve operatörleri açısından genetik algoritmayı temel alan diferansiyel gelişim algoritması, genetik algoritma gibi çaprazlama, mutasyon ve seçilim operatörlerini kullanır. Her ne kadar genetik algoritmaya benzese de iki algoritma arasında birtakım farklılıklar vardır. Bu algoritmada genetik operatörler kromozomlara tek tek uygulanır (Xia vd., 2021). Ayrıca, diferansiyel gelişim algoritması, gerçek değerli parametreler kullanır ve böylece gerçek yaşam problemlerine kolaylıkla uygulanabilir (Yang, 2014; Zorarpacı ve Özel, 2016).

1.2.2. Biyoloji Tabanlı Algoritmalar

Biyoloji tabanlı algoritmalar, herhangi bir biyolojik olaydan veya canlı organizmaların davranışlarından esinlenerek geliştirilen meta-sezgisel optimizasyon algoritmalarıdır. Bu kategorideki başlıca algoritmalar arasında yapay bağışıklık sistemleri (Farmer vd., 1986), bakteriyel besin arama optimizasyonu (Passino, 2002) ve çiçek tozlaşma algoritmaları (Yang, 2012) sayılabilir (Nanda ve Panda, 2014).

1.2.2.1. Yapay Bağışıklık Sistemleri

Bağışıklık sistemi, vücudu yabancı saldırılara karşı savunmak için birlikte çalışan bir hücre, doku ve organ ağıdır. Bununla birlikte, bakteri, virüs gibi zararlı mikro organizmaların yapıları hakkında önceden bilgi sahibi olmadan vücudu korur. Yapay bağışıklık sistemleri, problemleri çözmek için bağışıklık sisteminin ilkelerinden ve çalışma mekanizmasından ilham alan hesaplamalı sistemler olarak tanımlanabilir. Mühendislik ve bilim alanlarındaki problemlere çözüm sunabilecek birçok yapay bağışıklık sistemi modeli geliştirilmiştir. Yapay bağışıklık sistemlerindeki gelişmelerin büyük çoğunluğu klonal seçim, bağışıklık ağları ve negatif seçim olmak üzere bağışıklık bilimindeki üç ana teoriye odaklanmıştır (Farmer vd., 1986; Dasgupta, 2006; Timmis, 2007)

1.2.2.2. Bakteriyel Besin Arama Optimizasyonu

Escherichia Coli (E. coli) bakterisinin besin arama davranışlarından esinlenerek geliştirilen bakteriyel besin arama optimizasyonu, kemotaksis, üreme, eliminasyon-dağılma olmak üzere üç aşamada optimizasyon problemlerine uygulanmaktadır. Kemotaksis, E. coli bakterilerinin besin açısından zengin bölgelere toplanmalarını sağlamak amacıyla salgıladıkları kimyasal bir maddedir. Algoritmada, kemotaksis aşamasında bakterilerin ne kadar hareket edeceği belirlenir. Üreme aşamasında, belirli sayıda bakteri popülasyondan çıkarılır. Kalan bakteriler amaç değerlerine göre sıralanarak popülasyonun ilk yarısı kopyalanırken, son yarısı popülasyondan atılır. Eliminasyon-dağılma aşamasında ise bakterilerin popülasyondan elenmesi gerçekleştirilir. Bu aşamada, önceden belirlenen bir değer ile her bakteri için rastgele üretilen bir sayı karşılaştırılır. Eğer bu sayı, önceden

belirlenen değerden küçük ise bakteriler popülasyondan atılır ve bunların yerine yeni bakteriler üretilir (Passino, 2002).

1.2.3. Fizik ve Kimya Tabanlı Algoritmalar

Bazı algoritmalar, elektrik yükleri, yerçekimi, nehir sistemleri gibi belirli fiziksel veya kimyasal kanunları taklit etmektedir (Fister Jr vd., 2013). Bu algoritmalardan en çok kullanılanları benzetimli tavlama (Kirkpatrick vd., 1983), harmoni arama (Geem vd., 2001), yerçekimsel arama (Rashedi vd., 2009), yapay kimyasal reaksiyon (Alatas, 2011) ve su döngüsü (Eskandar vd., 2012) algoritmalarıdır.

1.2.3.1. Benzetimli Tavlama

Benzetimli tavlama algoritması, doğal tavlama işleminin fiziksel sürecine dayanan Kirkpatrick vd. (1983) tarafından geliştirilen sezgisel bir arama algoritmasıdır. Tavlama, metallerin yüksek sıcaklıklarda ısıtılması ve sonra katılaşana kadar yavaş yavaş soğutulması işlemidir. Eğer soğutma işlemi çok hızlı gerçekleşirse düzensiz yapılar ortaya çıkar. Yüksek sıcaklıklarda, malzemelerin atomları yüksek enerji seviyelerindedir. Dolayısıyla, sıcaklık azaldıkça da enerji seviyeleri azalır ve böylece madde kararlı hale gelir. Benzetimli tavlama algoritmasında, katı durumlar optimizasyon probleminin mümkün olan çözümlerini temsil ederken, bu durumların enerjileri çözümlerin amaç değerlerine karşılık gelir. Optimal çözüm ise minimum enerji durumunda elde edilir (Kirkpatrick vd., 1983).

1.2.3.2. Harmoni Arama Algoritması

Harmoni arama algoritması, caz orkestralarındaki müzisyenlerin harmonik açıdan en iyi melodiyi elde edebilmek için yaptıkları doğaçlama çalışmalardan esinlenerek geliştirilmiştir (Geem vd., 2001). Cazda, bir müzisyen doğaçlamaya başlayınca diğerleri de doğaçlama yaparak ona eşlik ederler. Böylece, yeni melodiler elde edilir. Bu bağlamda, harmoni arama algoritması, doğaçlama ile optimizasyon problemlerine çözüm üretmeyi amaçlar. Orkestrada harmoninin başarısı estetik anlayışa bağlı iken, algoritmada amaç fonksiyonuna bağlıdır. Karar değişkenleri, müzik aletlerinde kullanılan notalar ile temsil

edilir ve elde edilen harmoni ise optimizasyon probleminin çözümüne karşılık gelir (Geem vd., 2001; Manjarres vd., 2013).

1.2.3.3. Yerçekimsel Arama Algoritması

Yerçekimsel arama algoritması, Newton'un yerçekimi ve hareket kanununu temel alan meta-sezgisel bir optimizasyon algoritmasıdır. Algoritmada, her nesne kütlesi oranında diğer nesnelere bir yerçekimi kuvveti uygular. Kütlesi hafif olan nesneler, kütle bakımından ağır olan nesnelere doğru hareket eder. Nesneler arası etkileşim kurularak, bütün nesnelerin en ağır nesneye yönelmesi sağlanır. En ağır kütleye sahip nesne diğerlerine göre daha yavaş hareket eder ve böylece diğerlerini kendine doğru çeker. Dolayısıyla, algoritmada, her bir nesne bir çözümü ifade etmektedir ve bunların ağırlıkları ise performanslarını göstermektedir (Rashedi vd., 2009).

1.2.3.4. Su Döngüsü Optimizasyon Algoritması

Doğadan esinlenerek geliştirilen su döngüsü optimizasyon algoritması, suyun yağmurla atmosferden yeryüzüne, nehir ve akarsularla deniz veya okyanuslara daha sonra buharlaşarak tekrar atmosfere ulaşma sürecine dayanmaktadır. Algoritmada, popülasyondaki bireyler yağmur damlalarıyla ifade edilir ve en iyi birey deniz olarak seçilir. İyi yağmur damlaları nehir, kalan yağmur damlaları ise nehirden denize su taşıyan akarsu olarak kabul edilir. Akış debisine bağlı olarak her nehir akarsulardan su alır ve her akarsudan denize veya nehirlere akan su miktarı farklıdır (Eskandar vd., 2012).

1.2.4. Sürü Zekâsı Tabanlı Algoritmalar

Sürü kelimesi, kuş, balık ve böcek gibi uyumlu hareket eden canlıların oluşturduğu topluluk anlamına gelir. Başka bir ifadeyle sürü, aynı türdeki canlıların birbirleri ile etkileşimleri sonucu oluşturdukları sosyal grup olarak tanımlanır (Bansal ve Pal, 2019). Bir sürüdeki bireyler aktif, dinamik ve basit olmalıdır. Bununla birlikte, sürüdeki bireyler iş birlikçi davranışları sayesinde, rastgele aramadan daha iyi bir arama stratejisi oluşturur. Bu şekilde elde edilen akıllı arama stratejisi genellikle sürü zekâsı olarak adlandırılır (Bonabeau vd., 1999; Bansal ve Pal, 2019). Bu bağlamda, kuş, balık, karınca, arı ve diğer küme veya

koloni oluşturduğu bilinen canlıların merkezi olmayan ve kendi kendini organize eden davranışlarından esinlenerek geliştirilen algoritmalara sürü zekâsı algoritmaları adı verilmektedir (Bonabeau vd., 1999; Blum ve Li, 2008). Yapay zekânın gelişmesiyle birlikte en çok çalışılan alanlardan biri olan sürü zekâsı, meta-sezgisel optimizasyon algoritmaları arasında en çok gelişen daldır (Osaba ve Yang, 2021). Özellikle, böcek kolonileri ve hayvan grupları, sürü zekâsı modellerinin tasarlanması için zengin bir kaynak oluşturur (Cuevas vd., 2018).

Bir sürü modeli geliştirmek için belirli prensiplerin sağlanması gerekmektedir. Bunlar,

- Çoklu birimler arasında bilgi paylaşımı,
- Birimlerin kendi kendini organize etmesi ve evrim geçirmesi,
- Birimlerin eş öğrenmesi açısından etkili olması,
- Uygulamalı ve gerçek zamanlı problemlere kolayca uyarlanabilmesi

olarak verilebilir (Thakare, 2013; Meetei, 2014; Yu vd., 2015; Hassanien ve Emary, 2016).

Biyolojik sistemlerden esinlenerek geliştirilen algoritmaların çoğu direkt olarak sürü zekâsını kullanmaktadır (Fister Jr vd., 2013). Literatürde sürü zekâsına dayalı algoritmalara örnek olarak karınca koloni optimizasyonu (Dorigo ve Di Caro, 1999), parçacık sürü optimizasyonu (Eberhart ve Kennedy, 1995), yapay arı koloni algoritması (Karaboğa, 2005), guguk kuşu algoritması (Yang ve Deb, 2009), ateş böceği algoritması (Yang, 2009), yarasa algoritması (Yang, 2010), kril sürüsü optimizasyon algoritması (Gandomi ve Alavi, 2012), sosyal örümcek algoritması (Cuevas vd., 2013), yunus ekolokasyon algoritması (Kaveh ve Farhoudi, 2013), gri kurt optimizasyonu (Mirjalili vd., 2014), karınca aslanı optimizasyonu (Mirjalili, 2015), karga arama algoritması (Askarzadeh, 2016), balina optimizasyon algoritması (Mirjalili ve Lewis, 2016), benekli sırtlan optimizasyonu (Dhiman ve Kumar, 2017), çekirge sürü optimizasyonu (Saremi vd., 2017), salp sürüsü algoritması (Mirjalili vd., 2017), baykuş arama algoritması (Jain vd., 2018), imparator penguen optimizasyonu (Dhiman ve Kumar, 2018), imparator penguen kolonisi (Harifi vd., 2019), sincap arama algoritması (Jain vd., 2019), kelebek optimizasyon algoritması (Arora ve Singh, 2019), serçe arama algoritması (Xue ve Shen, 2020), yapay denizanası arama optimizasyonu (Chou ve Truong, 2021), fare sürüsü optimizasyonu (Dhiman vd., 2021), sürüngen arama algoritması (Abualigah vd., 2022) ve bal porsuğu algoritması (Hashim vd., 2022) verilebilir.

1.2.4.1. Karınca Koloni Optimizasyonu

Karınca koloni optimizasyonu, hesaplaması zor olan kombinasyonel problemlerin çözümü için geliştirilen sürü zekâsına dayalı meta-sezgisel bir algoritmadır (Dorigo ve Di Caro, 1999). Bu algoritma, gerçek karıncaların yiyecek arama davranışlarından esinlenerek geliştirilmiştir. Karıncalar, yiyecek ararken yuvalarının çevresinde rastgele araştırma yaparlar. Yiyecek kaynağı bulan karıncalar, kaynağın kalitesini ve miktarını ölçüp değerlendirir ve yiyeceğin bir kısmını yuvaya götürür. Yuva ile yiyecek kaynağı arasındaki güzergaha ise feromen adı verilen bir koku bırakır. Yiyeceğin kalitesi ve miktarı bu kokunun yoğunluğuna etki etmektedir. Bununla birlikte, feromen salgılandıktan belirli bir süre sonra yok olmaktadır. Yuvadan yiyecek kaynağına feromen sayesinde ulaşan karıncalar, feromen yoğunluğuna göre rastgele bir seçim yapar. Böylece, karıncalar yuva ile yiyecek kaynağı arasındaki en kısa yolu belirlerler. Algoritmada, her olası çözüm feromen değeri ile ilişkilidir. Ayrıca, algoritmadaki yapay karıncalar gerçek karıncalardan farklı olarak tamamen kör değildir ve belirli bir hafızaya sahiptir (Dorigo vd., 1991; Dorigo ve Di Caro, 1999; Dorigo vd., 2006).

1.2.4.2. Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu, balık ve kuş gibi canlıların doğadaki sosyal davranışlarından esinlenerek geliştirilen meta-sezgisel bir optimizasyon algoritmasıdır (Eberhart ve Kennedy, 1995). Bir kuş sürüsü birbirine çarpmadan birlikte uçar, yön değiştirir ve bunları eşzamanlı olarak gerçekleştirir. Sürüdeki bir kuş yiyecek bulduğunda, diğer kuşlarla iletişim kurarak onların da yiyeceğe ulaşmasını sağlar. Aynı şekilde, balık sürülerinde de bir balık bir tehlike sezdiğinde sürüdeki diğer balıklarla iletişime geçerek onların tehlikeden haberdar olmasını sağlar. Özellikle kuş ve balık sürülerinin iş birlikçi davranışlarından ilham alan parçacık sürü optimizasyonu, yapay parçacıkların arama uzayında kuş ve balık sürülerinin hız ve konum gibi fiziksel özelliklerini taklit etmesiyle optimal çözümü bulmayı amaçlar. Algoritmada, her parçacık muhtemel bir çözümü temsil etmektedir (Eberhart ve Kennedy, 1995; Kaewkamnerdpong ve Bentley, 2005; Krause vd., 2013; Vasuki, 2020).

Parçacıkların arama uzayındaki konumlarının değişimi bireylerin sosyo-bilişsel eğilimine bağlıdır ve parçacıkların mevcut konumlarının güncellenmesi için hız vektörü

kullanılır. Bir parçacığın en iyi konumu *"pbest"* olarak adlandırılır ve bu değer hız vektörünün güncellenmesi için hafızada tutulur. Bununla birlikte, hız vektörünün hesaplanmasında popülasyondaki herhangi bir parçacık tarafından elde edilen en iyi konum da kullanılmaktadır. Bu konum ise *"gbest"* olarak adlandırılmaktadır. *"pbest"* ve *"gbest"* değerleri bulunduktan sonra parçacıkların konumu ve hızı güncellenir (Eberhart ve Kennedy, 1995). Bir parçacığın hızı,

$$v_i^{t+1} = wv_i^t + c_1 r_1(pbest_i - x_i^t) + c_2 r_2(gbest - x_i^t) \quad (2)$$

biçiminde hesaplanır. Burada, v_i^{t+1}, i. parçacığın $(t+1)$. iterasyondaki hızını; v_i^t, i. parçacığın t. iterasyondaki hızını ve x_i^t, i. parçacığın t. iterasyondaki konumunu ifade etmektedir. Ayrıca, w, atalet ağırlığıdır ve genellikle $[0, 1.2]$ aralığından seçilir. Öğrenme katsayıları olarak adlandırılan c_1 ve c_2 genellikle $[0, 2]$ aralığında belirlenir. c_1, parçacığın kendi deneyimlerine, c_2 ise parçacığın sürüdeki diğer parçacıkların deneyimlerine göre hareket etmesini sağlar. r_1 ve r_2 ise $[0, 1]$ aralığında düzgün dağılımdan rastgele üretilen değerlerdir.

Parçacıkların konumu ise,

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3)$$

biçiminde güncellenir. Burada, x_i^{t+1}, i. parçacığın $(t+1)$. iterasyondaki konumunu ifade etmektedir (Eberhart ve Kennedy, 1995; Shi ve Eberhart, 1998; Kashani vd., 2021).

1.2.4.3. Yapay Arı Koloni Algoritması

Yapay arı koloni algoritması, bal arılarının yiyecek arama davranışlarını ilham alan sürü zekâsı tabanlı meta-sezgisel bir yöntemdir (Karaboğa, 2005). Bal arıları kovanlarının çevresindeki geniş alanlardan nektar toplar. Bunun yanı sıra, arı kolonilerinin, bir bölgede bulunan yiyecek miktarına göre, nektar toplamak için diğer bölgelerden o bölgeye arı gönderdiği gözlenmiştir. Zengin yiyecek kaynağı arayan arılar, yiyecek kaynaklarının yönü, mesafesi ve kalitesi hakkındaki bilgileri birbirleriyle dans yolu ile paylaşırlar (Hassanien ve Emary, 2016). Ayrıca, kovana aktarılan nektar miktarını en üst düzeyde tutmak ve daha kaliteli yiyecek kaynakları bulmak amacıyla bir görev paylaşımı yaparlar. Bu görevlerini

yerine getirmek için işçi, gözcü ve kâşif arı olmak üzere üç gruba ayrılırlar. İşçi arılar, buldukları çiçeklerden elde ettikleri nektarı kovana getirirler ve kovanda bekleyen gözcü arılara kaynak ve konum bilgisi vermek için dans ederler. Gözcü arılar, işçi arıların danslarını izler ve danslardan edindikleri bilgilere göre potansiyel olarak iyi olan çiçeklere uçarlar. Kâşif arılar ise, kovanın etrafında yeni yiyecek kaynakları bulmak için çevreyi araştırırlar (Karaboga ve Basturk, 2007a).

Bal arılarının bu görev paylaşımından ve kendi kendine organize olma davranışından esinlenerek geliştirilen yapay arı koloni algoritması işçi, gözcü ve kâşif arı olmak üzere üç grup arı içerir. İşçi arı sayısı gözcü arı sayısına eşittir ve her işçi arı bir yiyecek kaynağından sorumludur. İşçi arılar, yiyecek kaynaklarındaki nektar miktarlarını araştırırlar ve elde ettikleri bilgileri gözcü arılarla paylaşırlar. Gözcü arılar, işçi arılardan edindikleri bilgiler yardımıyla en iyi yiyecek kaynağını belirlemek için yiyecek bölgelerinin komşuluklarında arama yaparlar. Böylece, gözcü arılar zengin yiyecek kaynaklarına gitme eğiliminde olurlar. Kâşif arı ise bir yiyecek kaynağı tükendiğinde yeni bir yiyecek kaynağı araştırır. Yeni yiyecek kaynağı bulan kâşif arı, işçi veya gözcü arıya dönüşür (Karaboğa, 2005; Karaboga ve Basturk, 2007a; Karaboga ve Basturk, 2007b).

Algoritmanın ilk aşamasında, yiyecek kaynağı sayısı ve yiyecek kaynaklarının konumları rastgele olarak belirlenir. $X_i = \{x_{i1}, x_{i2}, \dots, x_{iD}\}$ ile tanımlanan yiyecek kaynakları,

$$x_{ij} = x_j^{min} + U(0,1)\left(x_j^{max} - x_j^{min}\right), \qquad \begin{matrix} i = 1,2,\dots,SN \\ j = 1,2,\dots,D \end{matrix} \tag{4}$$

biçiminde oluşturulur. Burada, SN, yiyecek kaynağı sayısını; D, optimize edilecek problemin boyutunu; x_j^{min} ve x_j^{max} j. boyuttaki alt ve üst sınırlarını; x_{ij}, j. boyuttaki i. yiyecek kaynağını ifade etmektedir. $U(0,1)$ ise $[0,1)$ aralığında düzgün dağılımdan seçilmiş rastgele bir sayıdır (Karaboğa, 2005; Hancer vd., 2015).

Her işçi arı, bir yiyecek kaynağı ile ilişkilendirilir ve mevcut konumunun komşuluğunda daha zengin bir yiyecek kaynağı bulmak için konumlarını değiştirir. Bu durumda, $V_i = \{v_{i1}, v_{i2}, \dots, v_{iD}\}$ ile tanımlanan yeni yiyecek kaynaklarının konumu,

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right), \quad k \in [1,2,\dots,SN] \tag{5}$$

şeklinde belirlenir. Burada, v_{ij}, j. boyuttaki i. yiyecek kaynağına karşılık gelir. ϕ_{ij}, $[-1,1]$ aralığında rastgele bir sayıdır. Ayrıca, k herhangi bir yiyecek kaynağını ifade etmektedir ve $k \neq i$ koşulunu sağlamalıdır.

Her yiyecek kaynağı problemin muhtemel bir çözümüne karşılık gelmektedir. Aynı şekilde, bir yiyecek kaynağındaki nektar miktarı, amaç fonksiyonu açısından bu çözümün kalitesini ifade etmektedir. Bununla birlikte, işçi arıların sağladığı bilgiler yardımıyla gözcü arılar, yiyecek kaynaklarındaki nektar miktarlarını değerlendirir ve nektar miktarına bağlı olarak bir yiyecek kaynağı seçerler. Bu seçim işlemi, her çözümün uygunluk değerine dayanan olasılıksal bir süreçtir. Diğerlerine göre daha yüksek olasılığa sahip yiyecek kaynaklarına yönelen gözcü arılar, yiyecek kaynağını,

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{6}$$

olasılık değerini kullanarak belirlerler (Karaboğa, 2005). Burada, p_i, i. yiyecek kaynağının olasılık değerini ve fit_i, i. çözümün uygunluk değerini göstermektedir. $f(x_i)$, X_i çözümünün amaç değeri olmak üzere, bir minimum problemi için uygunluk değeri,

$$fit_i = \begin{cases} \dfrac{1}{1+f(X_i)} & f(X_i) \geq 0 \\ 1+|f(X_i)| & f(X_i) < 0 \end{cases} \tag{7}$$

şeklinde belirlenir (Karaboğa, 2005).

Yapay arı koloni algoritmasında, herhangi bir ilerleme göstermeyen yiyecek kaynağının tükendiği varsayılır. Bu durumda, bu yiyecek kaynağındaki işçi arı kâşif arıya dönüşür. Kâşif arı, yeni bir yiyecek kaynağı aramaya başlar ve tükenen yiyecek kaynağının konumunu Eşitlik (4)'e göre günceller (Karaboğa, 2005).

1.2.4.4. Ateş Böceği Algoritması

Ateş böceği algoritması, ateş böceklerinin parlaklıklarını ve hareket yönlerini modelleyen meta-sezgisel bir algoritmadır. Algoritmada, tek cins olduğu kabul edilen ateş böceklerinin çekicilikleri parlaklıklarıyla doğru orantılıdır. Yani, yanıp sönen herhangi iki

ateş böceğinden daha az parlak olan diğerine doğru ilerler. Ayrıca, uzaklık arttıkça hem parlaklık hem de çekicilik azalır. Eğer belirli bir ateş böceğinden daha parlak bir ateş böceği yoksa, o ateş böceği rastgele bir konuma hareket eder. Bununla birlikte, ateş böceklerinin parlaklıkları amaç fonksiyonuna bağlı olarak belirlenir (Yang, 2009; Fister vd., 2013).

1.2.4.5. Guguk Kuşu Arama Algoritması

Guguk kuşu arama algoritması, bazı guguk kuşu türlerinin zorunlu kuluçka paraziti davranışına dayanan ve Levy uçuşunu kullanan meta-sezgisel bir algoritmadır (Yang ve Deb, 2009). Bazı guguk kuşu türleri, farklı bir kuşun yuvasını gözetleyerek, bu kuş yuvadan uzaklaştığında, kendi yumurtasını ev sahibi kuşun yumurtalarının olduğu yuvaya bırakır. Bıraktığı yumurtanın fark edilmemesi için kendi yumurtasına benzer yumurtaların olduğu yuvayı tercih eder ve ev sahibi yuvadaki yumurtalardan birini alır. Ev sahibi kuş, yumurtaların kendisine ait olmadığını anlarsa ya yuvadaki yabancı yumurtaları atar ya da yuvayı terk ederek yeni bir yuva kurar (Yang ve Deb, 2009; Vasuki, 2020).

Guguk kuşu araması, basitleştirilmiş üç temel kurala dayanır.

- Her guguk kuşu bir seferde tek yumurta bırakır ve yuvayı rastgele seçer.
- Yüksek kaliteli yumurtalara sahip en iyi yuvalar bir sonraki nesle aktarılır.
- Yuva sayısı sabittir ve ev sahibi kuş yuvasına bırakılan yumurtayı $p_a \in (0,1)$ olasılıkla fark edebilir. Bu durumda, ev sahibi kuş yumurtayı atabilir veya yuvayı terk edip yeni bir yuva kurabilir.

Guguk kuşu aramasında, bir yuvadaki her yumurta bir çözümü temsil eder. Her guguk kuşu sadece bir yumurta bırakır ve bu yumurta yeni bir çözüme karşılık gelir. Buradaki amaç, yeni ve potansiyel olarak daha iyi çözümleri, yeterince iyi olmayan çözümlerle değiştirmektir (Yang ve Deb, 2009).

Algoritmada, yeni çözümler üretilirken Levy uçuşu kullanılır. $x_i^{(t)}$ i. guguk kuşuna ait mevcut çözüm olmak üzere, yeni çözüm,

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Levy(\lambda) \tag{8}$$

biçiminde üretilir. Burada, α adım büyüklüğü olarak adlandırılır ve $\alpha > 0$ olmak üzere, çoğu durumda $\alpha = 1$ olarak alınır. $\oplus$ işareti ise eleman eleman çarpma işlemini ifade etmektedir.

Levy uçuşu, rastgele adım büyüklüğü Levy dağılımından gelen bir rastgele yürüyüş sağlar. Bununla birlikte, Levy dağılımı,

$$Levy{\sim}u = t^{-\lambda}, \quad (1 < \lambda < 3) \tag{9}$$

biçiminde tanımlanır.

Yeni çözümler üretildikten sonra, ev sahibi yuvadaki yumurta ile guguk kuşunun bıraktığı yumurtanın uygunlukları karşılaştırılır. Eğer guguk kuşunun yumurtasının uygunluk değeri daha yüksek ise, ev sahibi yumurta ile yer değiştirilir. Bununla birlikte, en kötü uygunluk değerine sahip yuva terk edilir. Son olarak, maksimum iterasyon sayısına ulaşıldığında, en yüksek uygunluk değerine sahip yuva optimal çözüm kabul edilir (Yang ve Deb, 2009; Vasuki, 2020).

1.2.4.6. Yarasa Algoritması

Yarasa algoritması, yarasaların ekolokasyon yani sesin yankılanmasından yararlanarak yön ve uzaklık saptama davranışını modelleyen meta-sezgisel bir yöntemdir (Yang, 2010). Yarasalar belirli bir frekansta, genellikle insanların algılamayacağı sınırlarda, ekolokasyon adı verilen bir sinyal yayarlar. Bu sinyal sayesinde hem birbirleriyle iletişimi sağlarlar hem de besinlerin konumlarını belirleyebilirler. Ayrıca, ortam tamamen karanlık olsa dahi herhangi bir nesneyi algılayıp o nesneye çarpmadan kolaylıkla hareket edebilirler (Yang ve Karamanoglu, 2013; Okwu ve Tartibu, 2021).

Yarasa algoritmasında, yarasaların ekolokasyon davranışı üç temel kurala dayanmaktadır.

- Bütün yarasalar yön bulma ve rota belirleme esnasında, av(yiyecek), avcı veya herhangi bir engeli algılamak veya ayırt etmek ve mesafeyi ölçmek için ekolokasyon kullanır.
- Yarasalar, yiyecek ararken, x_i konumunda v_i hızında, sabit f_i frekansı, değişen λ dalga boyu ve A_0 ses şiddeti ile rastgele uçarlar. Hedeflerinin yakınlığına bağlı olarak, sinyal yayılım oranlarını ($r \in [0,1]$), yaydıkları sinyallerin dalga boylarını ve frekanslarını otomatik olarak ayarlayabilirler.
- Gerçekte farklı şekillerde değişiklik göstermesine rağmen, algoritmada ses şiddeti, pozitif bir A_0 değeri ile minimum A_{min} değeri arasında değişir.

Algoritmada, yarasalar arama uzayına rastgele yerleştikten sonra, bulundukları konumların amaç değerleri hesaplanır. Her yarasa, frekansına ve popülasyondaki en iyi bireye ait çözüme karşılık olarak bir hız değeri belirler. Bu değerlere göre bir yarasanın yeni konumu,

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{10}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i \tag{11}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{12}$$

biçiminde belirlenir. Burada, β, $[0,1]$ aralığında düzgün dağılımdan rastgele üretilen bir değerdir. f_i, i. yarasaya ait frekans değerini; f_{min} ve f_{max} sırasıyla minimum ve maksimum frekans değerlerini; x_i^t, i. yarasanın t. iterasyondaki konumunu; v_i^t, i. yarasanın t. iterasyondaki hızını ve x_*, t. iterasyona kadar popülasyondaki en iyi çözümü ifade etmektedir (Yang, 2010).

Algoritmanın yerel arama kabiliyetini artırmak amacıyla, her yarasa uygunluk değerine bağlı olarak popülasyonda başka bir çözüm seçer ve bu çözüm etrafında yeni bir kaynağa yönlendirilir. Bu durumda, bir yarasanın yeni konumu,

$$x_{new} = x_{old} + \epsilon A^t \tag{13}$$

şeklinde güncellenir. Burada, ϵ, $[-1,1]$ aralığında rastgele bir sayı ve A^t, t. iterasyondaki tüm yarasaların ortalama ses şiddetini ifade etmektedir.

İterasyonlar ilerledikçe, yarasaların ekolokasyon ile ürettikleri ses şiddetinin ve sinyal yayılım oranlarının güncellenmesi gerekir. Yarasa avına yaklaştıkça ses şiddeti (A_i) azalırken, sinyal yayılım oranı (r_i) artar. Algoritmada, yarasaların ses şiddeti ve sinyal yayılım oranı sırasıyla,

$$A_i^{t+1} = \alpha A_i^t \tag{14}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{15}$$

biçiminde güncellenir (Yang, 2010).

1.2.4.7. Gri Kurt Optimizasyon Algoritması

Gri kurt optimizasyon algoritması, gri kurtların sosyal hiyerarşisini ve avlanma davranışlarını ilham alarak geliştirilen meta-sezgisel bir optimizasyon algoritmasıdır (Mirjalili vd., 2014). Gri kurtlar, doğada sürü halinde yaşar ve her sürü 5-12 kurttan oluşur. Sürü içerisinde gri kurtlar, alfa, beta, delta ve omega olmak üzere dört sınıftan oluşan hiyerarşik bir yapıya sahiptir.

Hiyerarşinin ilk basamağında sürü lideri olarak görev yapan alfa kurtlar vardır. Alfa kurtlar, avlanma, uyuma yeri, uyanma zamanı gibi konularda karar vermeden sorumludur. Sürü, baskın olan bu kurtların emirlerini yerine getirmek zorundadır. Bununla birlikte, bir av yakalandığında, her zaman ilk olarak alfa kurtlar yer (Mirjalili vd., 2014; Vasuki, 2020).

Hiyerarşinin ikinci basamağında alfa kurtlara danışmanlık ve yardımcılık yapan ve sürüdeki geri bildirimleri alfa kurtlara ileten beta kurtlar yer alır. Alfa kurt öldüğünde veya yaşlandığında, beta kurt alfanın yerine geçebilecek en güçlü adaydır. Bunun yanı sıra, beta kurt, alt seviyelerdeki kurtlara emir verebilir ancak, alfa kurda saygı duymak zorundadır (Mirjalili vd., 2014).

Üçüncü basamakta yer alan delta kurtlar, sınırları izleyerek herhangi bir tehlike durumunda sürüyü uyarır. Alfa ve beta kurtlara itaat etmesi gereken delta kurtlar, izcilerden, nöbetçilerden, yaşlılardan, avcılardan ve bakıcılardan oluşur. İzciler, bölgenin sınırlarını gözetlemek ve herhangi bir tehlike durumunda sürüyü uyarmakla görevlidir. Nöbetçiler, sürünün güvenliğinden sorumludur. Yaşlılar, eskiden alfa veya beta olan deneyimli kurtlardan oluşur. Avcılar, avlanmada ve sürüye yiyecek sağlarken alfa ve betalara yardım ederler. Son olarak, bakıcılar sürüdeki zayıf, hasta ve yaralı kurtların bakımını üstlenir (Mirjalili vd., 2014).

Hiyerarşinin son basamağındaki omega kurtlar ise diğer baskın kurtlara her zaman boyun eğmek zorundadır. Avı yemesi izin verilen son kurtlardır. Sürü için önemli bir birey gibi görülmemesine rağmen, omega kurtların olmaması durumunda sürüde iç karışıklıkların meydana geldiği gözlenmiştir. Ayrıca, omega kurtlar sürüde bebek bakıcılığı görevini de üstlenir (Mirjalili vd., 2014).

Gri kurt optimizasyonu, gri kurtların avlanma ve sosyal davranışlarını, sosyal hiyerarşi, avı çevreleme, avlanma, ava saldırma ve arama olmak üzere beş aşamada

modellemiştir. Algoritmada, doğada olduğu gibi, gri kurtlar alfa, beta, delta ve omega olmak üzere sınıflandırılır. Gri kurtların sosyal hiyerarşisini matematiksel olarak modellemek için en uygun çözüm alfa (α) olarak ele alınır. İkinci ve üçüncü en iyi çözümler sırasıyla beta (β) ve delta (δ) olarak adlandırılır. Kalan diğer aday çözümler ise omega (ω) olarak kabul edilir. Algoritmada avlanma α, β ve δ kurtlar tarafından yapılırken, ω kurtlar, baskın olan bu üç kurdu takip eder (Mirjalili vd., 2014).

Gri kurtların, avını çevreleme davranışı matematiksel olarak,

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \tag{16}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{17}$$

biçiminde modellenmiştir. Burada, t, mevcut iterasyonu; $\vec{A}$ ve $\vec{C}$ katsayı vektörlerini; $\vec{X}_p$, avın konum vektörünü; $\vec{X}$, gri kurdun konum vektörünü belirtmektedir. $\vec{A}$ ve $\vec{C}$ katsayı vektörleri,

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{18}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{19}$$

şeklinde hesaplanmaktadır. $\vec{a}$ iterasyonlar boyunca 2'den 0'a doğrusal olarak azalırken, r_1 ve r_2 [0,1] aralığından rastgele vektörlerdir (Mirjalili vd., 2014).

Beta ve delta kurtlar, alfa kurda yardım etmek için ara sıra avlanmaya katılabilir. Algoritmada, alfa en iyi aday çözüm, beta ve delta avın potansiyel konumu hakkında en iyi bilgiye sahip olduğu varsayıldığından, en iyi ilk üç çözüm kaydedilir ve en iyi arama ajanlarının konumuna göre, omegalar dahil olmak üzere diğer arama ajanlarının konumları,

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \quad \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \quad \vec{D}_\delta = \left|\vec{C}_\delta \cdot \vec{X}_\alpha - \vec{X}\right| \tag{20}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \tag{21}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{22}$$

biçiminde belirlenir.

Av hareket etmeyi bıraktığında, gri kurtlar ava saldırarak avlanmayı bitirir. Matematiksel olarak, $\vec{a}$ iterasyon boyunca 2'den 0'a azalır ve dolayısıyla $\vec{a}$ azaldıkça $\vec{A}$ da azalır. $|A| < 1$ olduğunda, gri kurtlar ava saldırmaya zorlanır (Mirjalili vd., 2014; Vasuki, 2020).

Gri kurtlar, alfa, beta ve delta kurtların yönlendirmesine göre avlanırlar. Av aranırken birbirlerinden ayrılır ve ava saldırırken bir araya gelirler. Arama ajanını avdan ayrılmaya zorlamak için 1'den büyük veya -1'den küçük rastgele bir A değeri kullanılır. Böylece, gri kurtların daha uygun bir av bulma umuduyla avdan uzaklaşması sağlanarak algoritmanın global arama yeteneği geliştirilir (Mirjalili vd., 2014; Vasuki, 2020; Okwu ve Tartibu, 2021).

1.2.4.8. Karınca Aslanı Optimizasyon Algoritması

Karınca aslanı optimizasyon algoritması, doğadaki karınca aslanlarının avlanma mekanizmasından esinlenerek geliştirilmiş meta-sezgisel bir algoritmadır. Adlarını en sevdikleri av olan karıncalardan alan karınca aslanlarının doğal yaşam döngüsü larva ve yetişkinlik olmak üzere iki aşamadan oluşur. Larva dönemlerinde avlanan karınca aslanları, yetişkinlik dönemlerinde çoğalırlar. Bununla birlikte, larva dönemlerinde karıncaların bulunduğu bölgelere kum çukurları oluşturarak tuzak kurarlar ve çukurun dibinde saklanarak tuzağa düşecek karıncaları beklerler. Karıncaların çukurdan çıkmasını engellemek ve onları çukurun dibine kaydırmak için kum fırlatırlar. Çukurun dibine düşen karıncaları çeneleriyle yakalayarak beslenirler. Daha sonra, tuzaklarını yeniden düzenleyip, yeni av için hazırlık yaparlar. Karınca aslanlarının bu avlanma stratejilerini modelleyen karınca aslanı algoritması, karıncaların rastgele yürüyüşü, tuzak kurma, karıncaların tuzaklara düşmesi, avı yakalama ve tuzakları yeniden kurma gibi avlanmanın beş ana aşamasını uygulamaktadır (Mirjalili, 2015).

1.2.4.9. Balina Optimizasyon Algoritması

Balina optimizasyon algoritması, dünyanın en büyük memeli hayvanı olarak kabul edilen balinaların sosyal davranışlarını modelleyen meta-sezgisel bir yöntemdir. Algoritma, yedi farklı türü olan balinalardan, kambur balinaların avlanma stratejisinden esinlenmiştir. Hava kabarcığı adı verilen kendilerine özgü bir avlanma stratejisine sahip kambur balinalar, genellikle yüzeye yakın küçük balık sürüleri ile beslenirler. Kambur balinalar, suyun altında soluk verip hava kabarcığı bulutları oluşturarak yüzeye doğru yükselmeye başlar. Yüzeye doğru yükselirken sarmal bir yol izler ve bu esnada kabarcık oluşturmaya devam ederek avlanma alanını daraltır (Mirjalili ve Lewis, 2016).

Algoritmada, kambur balinaların avlanma stratejisi avı çevreleme, hava kabarcığı yöntemi (ava yaklaşma) ve av arama olmak üzere üç aşamadan meydana gelir. Kambur balinalar avın konumunu belirleyip, avı çevreleyebilir. Arama uzayındaki en iyi konum önceden bilinmediğinden, algoritma mevcut en iyi çözümü hedef av olarak kabul eder ve bunun optimale yakın olduğunu varsayar. En iyi arama ajanı belirlendikten sonra, diğer arama ajanları, en iyi arama ajanına göre konumlarını günceller. Matematiksel olarak, arama ajanlarının konumları,

$$\vec{D} = \left|\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)\right| \tag{23}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D} \tag{24}$$

biçiminde güncellenir. Burada, t, mevcut iterasyonu; $\vec{A}$ ve $\vec{C}$ katsayı vektörlerini; $\vec{X}^*$, t. iterasyona kadar elde edilen en iyi çözümün konum vektörünü ve $\vec{X}$, balinanın konum vektörünü belirtmektedir. $\vec{A}$ ve $\vec{C}$ katsayı vektörleri,

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{25}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{26}$$

şeklinde hesaplanmaktadır. $\vec{a}$ iterasyonlar boyunca 2'den 0'a doğrusal olarak azalırken, r [0,1] aralığından rastgele bir vektördür (Mirjalili ve Lewis, 2016).

Kambur balinaların hava kabarcığı davranışının matematiksel modellenmesi için daralan çevreleme mekanizması ve konumu sarmal güncelleme olmak üzere iki yaklaşım geliştirilmiştir (Mirjalili ve Lewis, 2016).

Daralan çevreleme mekanizmasında, a değeri azaltılır. Bu nedenle, $\vec{A}$'nın değişim aralığı da azalır. $\vec{A}$ için $[-1,1]$ aralığında rastgele değerler kullanılarak, bir arama ajanının yeni konumu, arama ajanının mevcut konumu ile en iyi ajanın konumu arasında belirlenir.

Konumu sarmal güncelleme yaklaşımında, (X, Y) konumundaki balina ile (X^*, Y^*) konumundaki av arasındaki mesafe hesaplanır. Daha sonra, kambur balinaların sarmal şeklindeki hareketini taklit etmek için, balina ve avın konumu arasında,

$$\vec{X}(t+1) = \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{27}$$

biçiminde sarmal bir denklem tanımlanır. Burada, $\vec{D}' = \left|\vec{X}^*(t) - \vec{X}(t)\right|$ olmak üzere i. balinanın elde edilen en iyi çözüme olan uzaklığını ifade eder. b, logaritmik sarmalın şeklini tanımlamada kullanılan bir sabit ve l, $[-1,1]$ aralığında rastgele bir sayıdır (Mirjalili ve Lewis, 2016).

Algoritmada, kambur balinaların daralan çevreleme mekanizması ve konumu sarmal güncelleme yaklaşımlarından birini seçme olasılığı %50 olarak kabul edilmiştir. Bu durumda, balinaların yeni konumları,

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & p < 0.5 \\ \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & p \geq 0.5 \end{cases} \tag{28}$$

biçiminde belirlenir. Burada, p, $[0,1]$ aralığında rastgele bir sayıdır (Mirjalili ve Lewis, 2016).

Hava kabarcığı yöntemine ek olarak, kambur balinalar rastgele av ararlar. $\vec{A}$ vektörünün değişimine dayanan yaklaşım, av arama aşamasında da kullanılabilir. Aslında, kambur balinalar, birbirlerinin konumlarına göre rastgele arama yaparlar. Bu nedenle, balina optimizasyon algoritmasında, referans bir balinadan daha uzağa gitmek için, arama ajanları $\vec{A}$'nın -1'den küçük ve 1'den büyük rastgele değerini kullanır. $\left|\vec{A}\right| > 1$ yaklaşımı yardımıyla, algoritmanın global arama yapması sağlanır. Bu aşamada, balinanın konumu,

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_{rand} - \vec{X}\right| \quad (29)$$

$$\vec{X}(t+1) = \vec{X}_{rand} - \vec{A} \cdot \vec{D} \quad (30)$$

şeklinde güncellenir. Burada, $\vec{X}_{rand}$ mevcut popülasyondan seçilen rastgele bir konum vektörüdür (Mirjalili ve Lewis, 2016).

1.2.4.10. Karga Arama Algoritması

Karga arama algoritması, kargaların yiyeceklerini saklama ve gerektiğinde sakladıkları yerden geri alma fikrinden yola çıkılarak geliştirilen meta-sezgisel bir optimizasyon algoritmasıdır. Doğadaki en zeki kuş olan kargaların beyni, bedenlerine göre daha büyüktür. Hatta, beden-beyin oranı baz alındığında, beyinleri insanlarınkinden biraz küçüktür. Gördükleri yüzleri unutmayan kargalar, herhangi bir tehlike karşısında birbirlerini uyarır. Ayrıca, karmaşık yollarla iletişim kurabilirler ve sakladıkları yiyeceklerin yerlerini birkaç ay sonra bile hatırlayabilirler. Kargalar diğer kuşları izleyerek, onların yiyeceklerini nereye sakladıklarını gözlemler ve onlar gittikten sonra yiyecekleri çalarlar. Hırsızlık yapan bir karga, kendi saklama yerini de değiştirerek önlem alır. Aslında, kendi hırsızlık deneyimlerinden yola çıkarak başka bir hırsızlık davranışını tahmin edebilirler. Böylece, gizli yerlerindeki yiyeceklerinin çalınmasını önlemek için güvenli bir yol belirleyebilirler (Askarzadeh, 2016).

Kargaların zeki davranışlarına dayanan karga arama algoritması dört kurala göre geliştirilmiştir. Bu kurallar aşağıdaki gibi verilebilir:

- Kargalar sürü halinde yaşar.
- Saklama yerlerinin konumlarını ezberlerler.
- Hırsızlık yapmak için birbirlerini takip ederler.
- Saklama yerlerindeki yiyecekleri çalınmaktan korurlar.

N karga sayısı, D problemin boyutu ve T_{max} maksimum iterasyon sayısı olmak üzere, t. iterasyondaki i. karganın konumu,

$$x^{i,t} = \left[x_1^{i,t}, x_2^{i,t}, \dots, x_d^{i,t}\right], \;\; t = 1, 2, \dots, T_{max} \quad (31)$$

biçiminde tanımlanır. Her karga saklama yerinin konumunu hafızasında tutar ve i. karganın saklama yerinin konumu t. iterasyonda $m^{i,t}$ ile ifade edilir. Bu konum, i. karganın şimdiye kadar elde ettiği en iyi konumdur. Her iterasyonda, j. karganın saklama yerini ($m^{j,t}$) ziyaret ettiği varsayılır. Aynı zamanda, i. karga da j. kargayı takip ederek saklama yerini öğrenmeye çalışır. Bu durumda, iki ihtimal söz konusu olabilir. Birincisi, j. karga i. karganın kendisini takip ettiğini fark etmez ve i. karga j. karganın saklama yerine yaklaşır. İkincisi ise, j. karga i. karganın kendisini takip ettiğini fark eder ve saklama yerini korumak için, i. kargayı arama uzayında başka bir konuma yönlendirerek oyalar. Bu ihtimaller düşünüldüğünde, kargaların yeni konumları,

$$x^{i,t+1} = \begin{cases} x^{i,t} + r_i \times fl^{i,t} \times \left(m^{j,t} - x^{i,t}\right) & r_j \geq AP^{j,t} \\ rastgele\ bir\ konum & r_j < AP^{j,t} \end{cases} \tag{32}$$

şeklinde belirlenir. Burada, r_i, [0,1] aralığında düzgün dağılımdan rastgele bir sayıdır. $fl^{i,t}$, t. iterasyonda i. karganın uçuş uzunluğunu; $AP^{j,t}$, t. iterasyonda j. karganın farkındalık olasılığını ifade eder (Askarzadeh, 2016).

Kargaların yeni konumları kontrol edilerek, eğer uygun bir çözümse, eski konumları güncellenir. Aksi takdirde, kargalar mevcut konumlarında kalırlar. Kargaların yeni konumlarının uygunluk değerleri hesaplanır ve her karganın hafızası,

$$m^{i,t+1} = \begin{cases} x^{i,t+1} & f\left(x^{i,t+1}\right) > f(m^{i,t}) \\ m^{i,t} & f\left(x^{i,t+1}\right) \leq f(m^{i,t}) \end{cases} \tag{33}$$

biçiminde güncellenir. Burada, $f(.)$, optimize edilen amaç fonksiyonu değerini ifade etmektedir (Askarzadeh, 2016).

Algoritmada, maksimum iterasyon sayısına ulaşıldığında, amaç değeri açısından hafızadaki en iyi konum problemin çözümü olarak kabul edilir.

1.2.4.11. Harris Şahini Optimizasyonu

Harris şahini optimizasyonu, Harris şahinlerinin iş birlikçi davranışı, sürpriz pençe ve kovalama karakteristiğinden esinlenerek geliştirilmiş meta-sezgisel bir algoritmadır. Harris şahinleri, tavşan avlama sürecinde sürü halinde hareket ederler. Avlarını yakalamadaki ana

taktikleri, yedi öldürme olarak da bilinen sürpriz pençe stratejisidir. Bu akıllı stratejide birkaç şahin, kaçan bir tavşana iş birliği içinde farklı yönlerden saldırmaya ve yaklaşmaya çalışır. Saldırı birkaç saniye içinde avı yakalayarak tamamlanabilir. Ancak, bazen avın kaçma yeteneğine bağlı olarak, sürpriz pençe stratejisi birkaç dakika boyunca avın etrafında, birden fazla, kısa uzunlukta, hızlı dalışlar içerebilir. Harris şahinleri, koşullara ve avın kaçma durumuna göre farklı kovalama stratejileri sergileyebilir. Lider şahin, avın üzerine eğilip kaybolduğunda bir yer değiştirme taktiği meydana gelir ve kovalamaya sürüdeki diğer şahinler devam eder. Bu yer değiştirme aktiviteleri, kaçan tavşanın kafasını karıştırmak için etkili olduğundan farklı durumlarda gözlenebilir. Bu iş birlikçi davranışların temel avantajı, Harris şahinlerinin kaçan tavşanı yoruluncaya kadar takip etmesidir. Böylece, yorulan tavşan daha savunmasız olur. Ayrıca, kaçan avı şaşırtarak, av, savunma yeteneklerini geri kazanamaz ve karşı karşıya kaldığı ekip kuşatmasından kaçamaz. Çünkü, en güçlü ve deneyimli şahinlerden biri yorulan tavşanı yakalar ve diğer sürü üyeleriyle paylaşır (Heidari vd., 2019).

Harris şahinlerinin avı keşfetme, sürpriz pençe ve farklı saldırma stratejilerinden esinlenerek, Harris şahinleri optimizasyonunun keşif ve sömürü aşamaları modellenmiştir (Heidari vd., 2019).

Harris şahinleri, güçlü gözleriyle avlarını izler ve daha sonra onları tespit eder. Ancak bazı durumlarda, avlarını kolayca göremeyebilir ve bu yüzden bekleyip avlarını gözlemlerler. Bu bekleme saatlerce sürebilir. Avlarını gözlemleyen şahinler, daha sonra avlarına saldırırlar. Algoritmada, Harris şahinleri aday çözümlerdir ve her adımdaki en iyi aday çözüm istenilen av veya optimum çözüm olarak kabul edilir. Bazı konumlarda, Harris şahinleri, rastgele tüner ve iki stratejiye dayanarak avını tespit etmek için bekler. Bu stratejiler,

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)| & q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - r_3\left(LB + r_4(UB - LB)\right) & q < 0.5 \end{cases} \quad (34)$$

biçiminde modellenmiştir. Burada, t, mevcut iterasyonu; $X(t+1)$, şahinin yeni konum vektörünü; $X_{rabbit}(t)$, tavşanın konumunu; $X(t)$, şahinlerin mevcut konum vektörlerini; r_1, r_2, r_3, r_4 ve q, her iterasyonda güncellenen $(0,1)$ aralığından seçilen rastgele sayıları; LB ve UB, değişkenlerin alt ve üst sınırlarını; $X_{rand}(t)$, mevcut popülasyondan rastgele seçilen bir

şahinin konumunu; X_m, mevcut popülasyondaki şahinlerin ortalama konumlarını ifade etmektedir. Şahinlerin ortalama konumu,

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{35}$$

şeklinde hesaplanır. Burada, $X_i(t)$, t. iterasyondaki i. şahinin konumunu; N ise toplam şahin sayısını göstermektedir (Heidari vd., 2019).

Keşif aşaması tamamlandıktan sonra, Harris şahinleri avın enerjisine göre farklı saldırı biçimleri geliştirebilir. Avın enerjisi, kaçma davranışı sırasında önemli derecede azalabilir. Bu durumu göstermek için, avın enerjisi,

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{36}$$

biçiminde modellenmiştir. Burada, E, avın kaçma enerjisini; E_0, avın başlangıç enerjisini; T maksimum iterasyon sayısını göstermektedir. Ayrıca, E_0, her iterasyonda $(-1,1)$ aralığında rastgele olarak değişmektedir (Heidari vd., 2019).

Algoritmanın sömürü aşamasında, Harris şahinleri, tespit edilen ava saldırarak sürpriz pençe davranışını gerçekleştirir. Saldırı aşamasını modellemek için, avın kaçma davranışı ve Harris şahinlerinin farklı kovalama taktiklerine göre, yumuşak kuşatma, sert kuşatma, aşamalı hızlı dalışlarla yumuşak kuşatma ve aşamalı hızlı dalışlarla sert kuşatma olmak üzere dört muhtemel strateji geliştirilmiştir. Bu bağlamda, $|E| \geq 0.5$ olduğunda yumuşak kuşatma meydana gelirken, $|E| < 0.5$ olduğunda ise sert kuşatma gerçekleşir (Heidari vd., 2019).

r, avın kaçma şansı olmak üzere, $r \geq 0.5$ ve $|E| \geq 0.5$ olduğunda, tavşan yeterli enerjiye sahiptir ve rastgele zıplamalar yaparak kaçmaya çalışır. Ancak, sonunda başarısız olur. Bu girişimler sırasında, Harris şahinleri, tavşanı daha fazla yormak için çevreler ve sürpriz pençe davranışını gerçekleştirir. Bu davranış,

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \tag{37}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{38}$$

biçiminde modellenmiştir. Burada, $\Delta X(t)$, t. iterasyondaki mevcut konum ile tavşanın konum vektörü arasındaki farkı ifade etmektedir. J tavşanın doğal hareketini simüle etmek için her iterasyonda rasgele değişen bir değerdir ve r_5, $(0,1)$ aralığında rastgele sayı olmak üzere $J = 2(1 - r_5)$ olarak tanımlanmıştır (Heidari vd., 2019).

$r \geq 0.5$ ve $|E| < 0.5$ olduğunda, av çok yorgundur ve çok az kaçma enerjisi vardır. Böylece, Harris şahinleri avı sert bir şekilde çevreler ve sürpriz pençe davranışını gerçekleştirir. Bu durumda, mevcut konumlar,

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \tag{39}$$

şeklinde güncellenir (Heidari vd., 2019).

$r < 0.5$ ve $|E| \geq 0.5$ olduğunda, tavşanın başarılı olarak kaçması için yeterli enerjisi vardır ve sürpriz pençe gerçekleşmeden önce yumuşak kuşatma oluşturulur. Şahinlerin gerçek davranışlarından esinlenerek, rekabetçi durumlarda avı yakalamak istediklerinde, ava doğru en iyi dalışı aşamalı olarak seçecekleri varsayılmıştır. Bu nedenle, yumuşak bir kuşatma gerçekleştirmek için, şahinler, bir sonraki hamlesine,

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \tag{40}$$

kuralına göre karar verir. Bu hamlenin, daha iyi bir dalış olup olmadığını belirlemek için önceki dalış ile karşılaştırılır. Eğer bu hamle mantıklı değilse, tavşana yaklaşırken düzensiz, ani ve hızlı dalışlar gerçekleştirirler. Buna göre, şahinlerin Levy uçuşu tabanlı bir dalış yapacağı varsayılır ve bu dalış,

$$Z = Y + S \times LF(D) \tag{41}$$

kuralına göre yapılır. Burada, D, problemin boyutunu; S, $1 \times D$ boyutunda rastgele bir vektörü ifade etmektedir. Levy uçuşu fonksiyonu olan LF,

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \qquad \sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{42}$$

biçiminde tanımlanır. Burada, u ve v, $(0,1)$ aralığında rastgele değerlerdir ve β, 1.5 olarak alınan bir sabittir. Dolayısıyla, yumuşak kuşatma aşamasında, son strateji için şahinlerin konumlarının güncellenmesi,

$$X(t+1) = \begin{cases} Y & F(Y) < F(X(t)) \\ Z & F(Z) < F(X(t)) \end{cases} \quad (43)$$

ile gerçekleştirilir (Heidari vd., 2019).

$r < 0.5$ ve $|E| < 0.5$ olduğunda, tavşanın kaçmak için yeterli enerjisi yoktur ve avı yakalamak ve öldürmek için sürpriz pençeden önce sert kuşatma oluşturulur. Bu durum av açısından yumuşak kuşatmaya benzer. Ancak, bu aşamada, şahinler kaçan av ile ortalama konumlarını azaltmaya çalışır. Bu nedenle, sert kuşatma durumunda,

$$X(t+1) = \begin{cases} Y & F(Y) < F(X(t)) \\ Z & F(Z) < F(X(t)) \end{cases} \quad (44)$$

kuralı uygulanır. Burada, Y ve Z,

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \quad (45)$$

$$Z = Y + S \times LF(D) \quad (46)$$

biçiminde elde edilir (Heidari vd., 2019).

1.2.4.12.Fare Sürüsü Optimizasyonu

Fare sürüsü optimizasyonu, farelerin doğadaki kovalama ve dövüşme davranışlarını matematiksel olarak modelleyen sürü zekâsı temelli meta-sezgisel bir optimizasyon algoritmasıdır. Fareler, büyüklük ve ağırlık bakımından farklı olan, uzun kuyruklu orta boy kemirgenlerdir. Siyah ve kahverengi olmak üzere iki tür fare vardır. Agresif bir yapıya sahip olan fareler hem erkek hem de dişi gruplarla yaşayan bölgesel canlılardır. Hırçın davranışlarıyla bir grup halinde avlarını kovalarlar ve daha sonra avlarıyla dövüşürler. Fare sürüsü optimizasyonunda, farelerin avlarını kovalama ve avlarıyla dövüşme davranışları matematiksel olarak modellenmiştir (Dhiman vd., 2021).

Genel olarak fareler, sosyal agonistik davranışlarıyla bir grup halinde avını kovalayan sosyal havyanlardır. Bu davranışı matematiksel olarak tanımlamak için, algoritmada, en iyi arama ajanının, avın konumuyla ilgili bilgiye sahip olduğu varsayılmıştır. Elde edilen en iyi arama ajanına göre diğer arama ajanlarının konumları,

$$\vec{P} = A \cdot \vec{P}_i(x) + C \cdot \left(\vec{P}_r(x) - \vec{P}_i(x)\right) \tag{47}$$

biçiminde güncellenir. Burada, $\vec{P}_i(x)$, farelerin konumlarını tanımlarken, $\vec{P}_r(x)$, en iyi çözümü ifade etmektedir. Bununla birlikte, A ve C parametreleri,

$$A = R - x \times \left(\frac{R}{Max_{iteration}}\right), \quad x = 0,1,2, \dots, Max_{iteration} \tag{48}$$

$$C = 2 \cdot rand() \tag{49}$$

biçiminde hesaplanır. R ve C, sırasıyla $[1,5]$ ve $[0,2]$ aralığından seçilen rastgele sayılardır. Bu parametreler iterasyon boyunca, keşif ve sömürüden sorumludur (Dhiman vd., 2021).

Farelerin avlarıyla dövüşme davranışı ise matematiksel olarak,

$$\vec{P}_i(x+1) = \left|\vec{P}_r(x) - \vec{P}\right| \tag{50}$$

şeklinde modellenmiştir. Burada, $\vec{P}_i(x+1)$ farenin güncellenmiş yeni konumunu tanımlamaktadır. Daha sonra, en iyi çözüm kaydedilir ve en iyi arama ajanına göre diğer arama ajanlarının konumları güncellenir. Bu şekilde, durdurma kriteri sağlanana ya da maksimum iterasyona ulaşılıncaya kadar en iyi çözüm aranır (Dhiman vd., 2021).

1.2.4.13. Yapay Denizanası Arama Optimizasyonu

Yapay denizanası arama optimizasyonu, okyanustaki denizanalarının davranışlarını modellemiştir. Farklı derinliklerde ve sıcaklıklarda yaşayan denizanaları, çeşitli renk, boyut ve şekillere sahiptir. Bazı denizanaları 1 cm'den daha küçükken bazıları çok daha büyük olabilir. Denizanaları dokunaçları ile sokarak avlarını hareketsiz hale getirirler. Yapay denizanası arama optimizasyonu ise, denizanalarının okyanus akıntısını takip etme, sürü

içerisindeki aktif ve pasif hareketlerini, bu hareketler arasında geçiş yapmak için zaman kontrol mekanizması ve çiçeklenme yakınsamalarını matematiksel olarak modellemiştir (Chou ve Truong, 2021).

1.2.5. Diğer Algoritmalar

Meta-sezgisel algoritmalar geliştirilirken ilham kaynağı her zaman doğa olmayabilir. Video oyunları, antik geçmiş, spor yarışmaları gibi farklı karakteristiklerden esinlenerek geliştirilen meta-sezgisel algoritmalar da vardır. Bu algoritmalar, evrimsel, biyoloji, sürü, fizik veya kimya tabanlı algoritmaların olduğu kategorilerde yer almazlar. Bu sebeple, bu algoritmaları ayrı bir kategoride incelemek gerekir. Bu kategorideki algoritmalara örnek olarak lig şampiyonası algoritması (Kashan, 2009), Giza piramitleri inşaatı (Harifi vd., 2021), Battle Royal optimizasyon algoritması (Farshi, 2021) verilebilir.

1.2.5.1. Battle Royal Optimizasyon Algoritması

Battle Royal optimizasyon algoritması, Battle Royal video oyunlarından esinlenerek geliştirilen meta-sezgisel bir yöntemdir. Dijital platformda birçok kişinin oynadığı Battle Royal oyunlarında, genellikle oyun alanı sınırlandırılarak, oyuncuların hayatta kalmak için çevrelerini keşfetmeleri ve kullanmaları gerekir. Genellikle, savaşlar, oyuncular tarafından seçilen belirli bir savaş alanında yapılır. Bireysel veya takım halinde oynanabilen bu oyunların zorluğu diğer oyuncuları yenmekle kalmayıp, güvenli bölge olarak adlandırılan oyun alanının, oyun boyunca küçülmesini de içerir. Güvenli bölgenin dışında kalan oyuncular ya yaralanır ya da oyundan atılma riski ile karşı karşıya kalır. Ayrıca, oyuncular güvenli bölgenin dışında kaldıkları her saniye zarar görürler. Dolayısıyla, sakatlanan bu oyuncular daha küçük bir güvenli bölgede diğer oyuncularla mücadele etmek zorunda kalır. Battle Royal oyunlarının çoğu eşit güç ve kaynaklara sahip oyuncularla başlar. Genellikle oyuncular oyun alanına rastgele yerleştirilir ve oyunculara çok fazla kaynak veya ekipman sağlanmaz. Rakipleri tarafından öldürülmekten kaçarken, oyuncuların, hayatta kalmaya yardımcı olacak araçları araması gerekir. Bu yüzden, keşif, oyunun önemli bir bileşenidir. Bir diğer önemli bileşen ise, bazı oyunlarda karakterlerin yeniden canlanmasıdır. Eğer bir oyuncu öldürülürse, savaş alanının rastgele bir bölgesinde yeniden doğar. Son olarak, en fazla öldürme sayısına sahip oyuncu, oyunun kazananı olur (Farshi, 2021).

Battle Royal optimizasyon algoritmasında, her oyuncu kendisine en yakın askeri (oyuncuyu) öldürmeye çalışır. Bir asker yaralandığında, onun hasar seviyesi artar. Ayrıca, yaralandıktan sonra askerler konum değiştirmek isterler ve böylece rakiplerine başka bir taraftan saldırabilirler. Bu nedenle, yaralanan bir asker, bir önceki konumu ve en iyi konum arasında bir konuma yerleşir. Bu etkileşimler matematiksel olarak,

$$x_{dam,d} = x_{dam,d} + r(x_{best,d} - x_{dam,d}) \tag{51}$$

şeklinde modellenmiştir. Burada, $x_{dam,d}$, d. boyutta yaralanmış askerin konumunu; $x_{best,d}$, bulunan en iyi çözümü ifade etmektedir. r, $[0,1]$ arasında düzgün dağılımdan rastgele bir sayıdır. Bununla birlikte, yaralanan asker, bir sonraki iterasyonda rakiplerini yaralarsa hasarı sıfırlanır. Ayrıca, bir askerin hasar seviyesi önceden tanımlanmış eşik değerini aşarsa, o asker ölür ve uygun çözüm alanında sıfır hasarlı olarak yeniden canlanır. Bu durumda, ölen asker arama alanına,

$$x_{dam,d} = r(ub_d - lb_d) + lb_d \tag{52}$$

biçiminde yerleştirilir. Burada, lb_d ve ub_d, d boyutlu uzayın sırasıyla alt ve üst sınırlarını göstermektedir. Ayrıca, her Δ iterasyonda bir uygun çözüm alanı en iyi çözüme doğru daralmaya başlar. $MaxCicle$, maksimum iterasyon sayısı olmak üzere, başlangıçta $\Delta = \log_{10}(MaxCicle)$ olarak alınır ve daha sonra, $\Delta = \Delta + round(\Delta/2)$ ile güncellenir. Bu durumda, uygun çözüm alanının alt ve üst sınırları,

$$\begin{aligned} lb_d &= x_{best,d} - SD(\bar{x}_d) \\ ub_d &= x_{best,d} + SD(\bar{x}_d) \end{aligned} \tag{53}$$

şeklinde güncellenir. Burada, $SD(\bar{x}_d)$, d. boyutta tüm popülasyonun standart sapmasını ifade etmektedir (Farshi, 2021).

1.3. No-Free-Lunch (NFL) Teoremi

Literatürde birçok optimizasyon algoritması geliştirilmiş olmasına rağmen, her algoritmanın farklı avantajları ve dezavantajları vardır. Bazı algoritmalar bazı problemleri

çözmede diğerlerine göre daha tutarlı olabilir. Bununla birlikte, gerçek yaşamdaki problemler karmaşık ve çeşitlidir. Ayrıca, bazı problemlerin çözümü kolayken bazılarının çözümü diğerlerine göre oldukça zor olabilir. Bu nedenle, bütün problemlerin çözümünü sağlayacak tek bir yöntemin olması mümkün değildir. Bu amaçla, Wolpert ve Macready (1997) tarafından ortaya atılan No-Free-Lunch teoremi, bütün optimizasyon algoritmalarının çok çeşitli problemlere uygulandığında performans açısından aynı olduğunu ifade etmektedir. Yani, bu teoreme göre hiçbir algoritma diğerinden üstün değildir ve her biri belirli bir optimizasyon problemi için uygundur. Bir problem için en iyi sonucu veren bir optimizasyon algoritması, başka bir probleme uygulandığında en iyi sonucu elde edemeyebilir. Ancak, başka bir optimizasyon algoritması, diğer probleme en iyi sonucu üretebilir. Ayrıca, bir optimizasyon algoritmasının bir problem üzerindeki iyi performansı, başka bir problem üzerindeki performansı ile dengelenir. Yani, her algoritma belirli bir problem türü için uygun olabilir, ancak tüm problem türleri için aynı derecede iyi performans göstermeyebilir. Kısaca, NFL teoremi bir problem türü üzerinden çeşitli optimizasyon algoritmalarının performanslarının karşılaştırılamayacağını ve bütün algoritmaların aslında performans açısından eşit olduğunu belirtmektedir. Ayrıca, algoritmalar iyi veya kötü olarak sınıflandırılamaz ve aynı şekilde en iyi veya en kötü algoritma yoktur. (Du ve Swamy, 2016; Yang, 2014; Vasuki, 2020).

1.4. Çekirge Sürüleri ile ilgili Yapılan Çalışmalar

Bu tez çalışmasında, önerilen algoritma çekirge sürülerinin davranışlarına dayanan sürü zekâsı tabanlı meta-sezgisel bir yöntemdir. Literatürde daha önce çekirge sürülerinin davranışlarını modelleyen birkaç algoritma olmasına rağmen, bunlar çekirgelerin farklı davranışlarını model almıştır. Topaz vd. (2008) ‘sosyal’ fazda olan çekirgeler üzerinde yoğunlaşmıştır ve bunun için matematiksel bir model kurmuştur. Daha sonra, Topaz vd. (2012) tarafından yapılan bir diğer çalışmada, polifenik çekirgelerin ‘bireysel’ ve ‘sosyal’ fazlardaki sosyal etkileşimleri ve hareketleri modellenmiştir. Ancak, bu çalışmalarda, çekirge sürülerinin davranışlarının sadece matematiksel modeli kurulmuştur. Daha sonraki çalışmalarda, bu modeller temel alınarak çekirge sürülerinin sosyal etkileşimlerine dayanan optimizasyon algoritmaları geliştirilmiştir. Chen (2009a) parçacık sürü optimizasyonuna dayanan bir çekirge sürüsü algoritması sunmuş ve daha sonra geliştirdiği bu algoritmayı büyük ölçekli optimizasyon problemlerinde analiz etmiştir (Chen, 2009b). Lewis (2009)

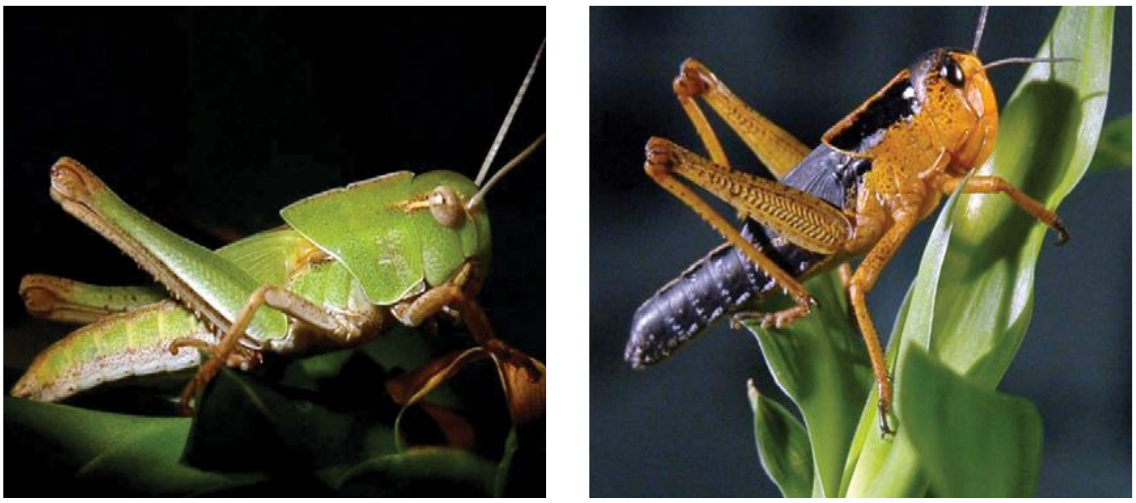
çekirge ve cırcır böceklerinin davranış modelini kullanarak, çok amaçlı optimizasyon problemlerini çözmek için uzamsal sosyal ağ kavramlarının genişletilmesine dayanan parçacık sürü optimizasyonu temelli bir yaklaşım önermiştir. Cuevas vd. (2015) tarafından önerilen çekirge araması algoritması, 'bireysel' ve 'sosyal' olmak üzere iki aşamadan oluşmaktadır. Birinci aşamada, çekirgeler, birbirlerinden kaçınarak, birbirlerini itme eğilimindedir. İkinci aşamada ise var olan çözümler komşuluk ilişkilerine göre geliştirilir. Bu çalışmada, Topaz vd. (2008; 2012) tarafından geliştirilen model temel alınmıştır. Ancak, bu modelden farklı olarak, Cuevas vd. (2015), bireysel aşamadaki çekirgelerin birbiriyle etkileşim içinde olduklarını kabul etmiştir. Bunun yanı sıra, Topaz vd. (2008) tarafından geliştirilen modeli kullanarak, Saremi vd. (2017) tek amaçlı optimizasyon problemlerinin çözümü için, çekirgelerin sosyal etkileşimlerini temel alan çekirge sürü optimizasyonunu önermiştir.

Bu tez çalışmasında, çekirge sürülerinin davranışlarını modelleyen Yapay Çekirge Sürü Optimizasyonu (ALSO) algoritması, biyolojik arka planı, motivasyonu ve arama davranışı bakımından, çekirge sürülerini temel alan literatürdeki diğer çalışmalardan tamamen farklıdır.

2. YAPILAN ÇALIŞMALAR

2.1. Çekirge Sürülerinin Davranışları

Çekirgeler, Acrididae familyasından filogenetik olarak heterojen bir gruptur. Nüfus yoğunluğuna bağlı olarak çekirgeler, *'bireysel'* fazdan (Şekil 2(a)) *'sosyal'* faza (Şekil 2(b)) geçerler (Topaz vd., 2012). Faz değişimi davranış, renk, morfometri, üreme gelişimi, doğurganlık, endokrin fizyolojisi, metabolizma, moleküler biyoloji gibi özelliklerle ilişkilidir (Collett vd., 1998; Simpson vd., 1999; Kang vd., 2004; Simpson vd., 2011; Ernst vd., 2015). *'Bireysel'* faz popülasyon yoğunluğunun az olduğu evredir. Bu evredeki çekirgeler birbirlerinden kaçınırlar. *'Sosyal'* fazda ise çekirgeler sürü halinde yaşarlar. Böylece, büyük göçler başlatabilirler (Simpson ve Sword, 2008; Simpson vd., 2011). Çekirgeler, *'bireysel'* fazdan *'sosyal'* faza geçtiklerinde, yüzlerce kilometrelik bir alana yayılan sayısız bireyden oluşan muazzam gruplar oluştururlar. Faz geçişinin en belirgin etkileri, vücut boyutu ve rengi dahil olmak üzere morfolojik görünümdeki değişikliklerdir. *'Bireysel'* fazdan *'sosyal'* faza geçiş saatler içerisinde meydana gelirken, renk ve morfolojik özelliklerdeki değişiklikler daha uzun bir sürede gerçekleşir (Simpson vd., 2011; Ernst vd., 2015).



(a) (b)

Şekil 2. Çekirge (a) *'bireysel'* fazdaki çekirge (b) *'sosyal'* fazdaki çekirge (Simpson ve Sword, 2008)

Popülasyon yoğunluğu ve sürünün oluşumu sıcaklık ve rüzgârdan etkilenmektedir. Sıcaklığın artmasıyla birlikte bitkiler gelişir ve böylece çekirgeler beslenme ihtiyaçlarını daha kolay karşılayabilirler. Rüzgâr çekirgelerin uçmalarını sağlamaktadır ve rüzgârın yönü ile sürünün hareket yönü aynıdır. Ayrıca, çekirgeler havadayken kanatlarını kullanmadıkça, hareket yörüngesi üzerinde hiçbir kontrolleri yoktur (Hoyle, 1958). Sıcaklık ve rüzgâr aynı zamanda sürünün hızını da etkilemektedir. Sıcak mevsimin sonunda dişi çekirgeler bulundukları yere yumurta bırakır. Yumurtalar nemli toprakta gelişir ve ilk yağmurlardan hemen sonra yavru çekirgeler yumurtadan çıkar. Yavru çekirgelerin kanatları yoktur. Ancak, yavrular hızla gelişip erginleşince kanatları oluşur. Böylece, sürünün de hızı artar (Simpson ve Sword, 2008). Göç halinde yaklaşık 100 km yol alabilen çekirgeler, bulundukları bölgelerdeki bitkileri istila ederler. Bu yüzden, tarım alanlarına önemli derecede zarar verebilirler (Inglis vd., 2007).

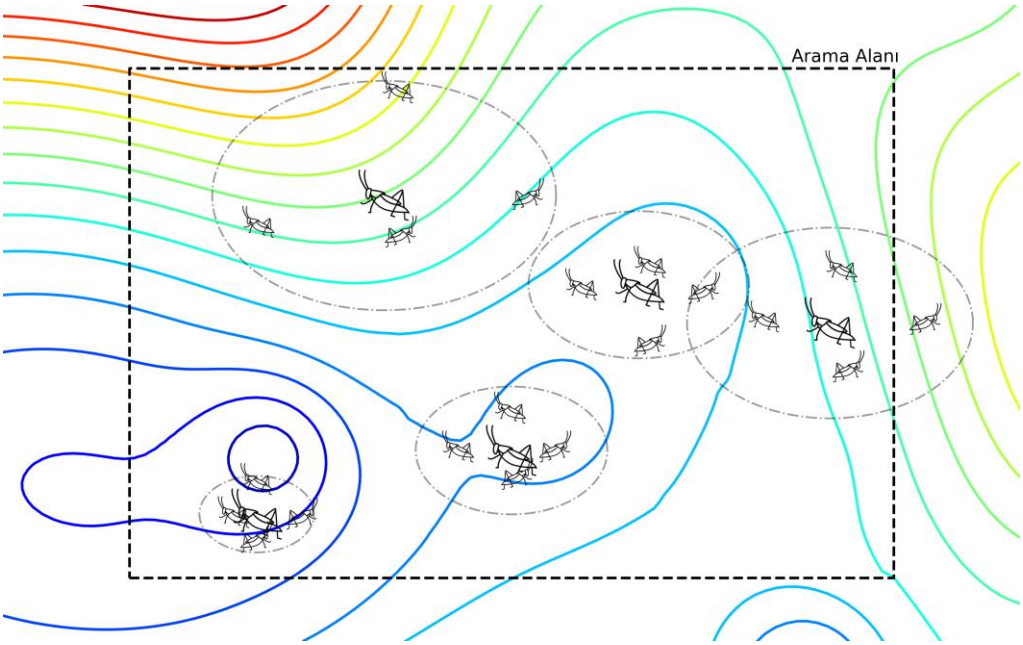
Çekirgeler müthiş bir sıçrama yeteneğine sahiptir ve bulundukları bölgede rastgele sıçramalar yaparlar. Beden uzunluklarının yaklaşık 10 katı kadar dikey ve yaklaşık 20 katı kadar yatay olarak sıçrayabilen çekirge türleri vardır (Hoyle, 1958; Scott, 2005; Simpson vd., 2011; Topaz vd., 2012). Bu sıçrama gözle takip edilemeyecek kadar hızlıdır (Hoyle, 1958).

Çekirgelerin gözlem yapma veya çevreyi koklama gibi özellikleri yoktur. Bu nedenle, iyi besin kaynaklarını bulmak için rastgele yönlere sıçrayarak yiyecek ararlar. Sıçrama yetenekleri besin kalitesine bağlı olarak değişebilir. İyi (bol) besin kaynaklarında daha az sıçrama yaparken, kötü (kıt) besin kaynaklarında daha fazla sıçrama yapabilirler. Bununla birlikte, çekirge sürüleri, uçarak bölgeleri keşfederler ve iyi besin kaynakları bulmaya çalışırlar. Uçuş sırasında, bölgelerdeki besin kalitesi hakkında bilgi sahibi olabilirler.

Bu çalışmada, çekirge sürülerinin yiyecek arama davranışlarından esinlenerek, sürekli optimizasyon problemlerinin çözümü için meta-sezgisel bir algoritma geliştirilmiştir. Sürü davranışı gösteren böcek türlerine göre akıllı ve sosyal davranış gösterme yaklaşımından uzak olan çekirgeler, rastgele davranış gösteren istilacı türlerden biridir. Önerilen algoritma çekirgelerin sıçrama ve istila etme davranışına dayanmaktadır. Bununla birlikte, çekirgelerin bu davranışları sıçrama miktarı ve duyarlılık fonksiyonu ile modellenmiştir. Çekirgelerin sıçrama davranışı ile arama uzayı keşfedilir, istila etme davranışı ile bulunan iyi çözümün etrafında global çözüm iyileştirilir. Böylece, önerilen algoritmanın keşif ve sömürü bileşenleri arasında bir denge kurması sağlanır.

2.2. Yapay Çekirge Sürü Optimizasyonu (ALSO)

Bu çalışmada, çekirge sürülerinin yiyecek arama davranışlarından esinlenerek yapay çekirge sürü optimizasyonu (ALSO) ile adlandırılan meta-sezgisel bir optimizasyon algoritması önerilmiştir. Doğada, çekirgeler bulundukları yerel bölgelerde yiyecek ararken, rastgele yönlere ve uzaklıklara sıçramaktadır. Algoritma, optimizasyon problemlerini çözmek için çekirge sürülerinin davranışını aile ve sosyal olmak üzere iki aşamada modellemiştir. Aile fazında küçük çekirge grupları yerel bir alanda yiyecek ararken, sosyal fazda sürü halinde yiyecek arama davranışı gösterirler. Aile fazındaki çekirge gruplarında, ana çekirge etrafında arama yapılır (Şekil 3). Her nesilde ana çekirge sürüden ayrılır ve belirli bir alana yumurta bırakır. Böylece, bu bölgede kendi ailesini oluşturur ve aile üyeleriyle birlikte yiyecek arar. Yavru çekirgeler arama bölgesinde denemeler gerçekleştirerek en iyi çözümü ararken, ana çekirge kendi konumunu korumaktadır. Ana çekirge dahil olmak üzere ailedeki tüm bireyler arasında en iyi yiyecek kaynağını bulan çekirge sosyal faza geçerek sürüye katılırken, diğer çekirgeler yok olur. Dolayısıyla, sürüdeki çekirgeler birbirleriyle etkileşime girerek besin kaynakları hakkında bilgi sahibi olmaya çalışır. Daha sonra, çekirgeler kendi nesillerini oluşturmak için yerel bölgelerine giderek ailesel faza geçerler. Yani, sürüdeki çekirgeler bir sonraki nesilde bulundukları konumdan aynı süreci devam ettirerek genel çözüme yaklaşırlar.



Şekil 3. Arama bölgesindeki aile fazındaki çekirgelerin temsili gösterimi

2.2.1. Çekirgelerin Başlangıç Konumlarının Oluşturulması

ALSO algoritması, arama uzayındaki çözümlere karşılık gelen çekirgelerin başlangıç konumlarını rastgele üretmekle başlar. Başlangıç konumları, arama uzayının aralığında rastgele olarak,

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right)\ i = 1, \dots, NF, j = 1, \dots, D \tag{54}$$

biçiminde oluşturulur. Burada, NF, aile sayısını; D, amaç fonksiyonunun boyutunu ve x_{ij}, j. boyuttaki i. çekirgenin konumunu ifade etmektedir. x_j^{min} ve x_j^{max} sırasıyla problem uzayındaki j. boyutun alt ve üst sınırlarını göstermektedir.

2.2.2. Maksimum Yer Değiştirme Miktarının Belirlenmesi

ALSO algoritması, yerel alanlarda yiyecek arayan çekirgelerin rastgele yönlerde ve mesafelerde sıçramasına dayanır. Aile fazında çekirgeler uçmak yerine sıçramayı tercih ederler. Bir çekirgenin ne kadar uzağa sıçrayacağı, o çekirgenin bulunduğu bölgedeki yiyeceğin verimine bağlıdır. Bölgenin yiyecek verimi ise uygunluk değeri ile ölçülür. Böylece, yiyecek veriminin yüksek olduğu bölgelerde bir çekirgenin maksimum yer değiştirme miktarı azalırken, düşük olduğu bölgelerde maksimum yer değiştirme miktarı artar. Başka bir ifadeyle, uygunluk değeri ile maksimum yer değiştirme miktarı ters orantılıdır.

Minimizasyon problemleri için uygunluk fonksiyonu,

$$u_i = \frac{f\left(\mathbf{x}_i^{(t)}\right) - \min f\left(\mathbf{x}^{(t)}\right)}{\max f(\mathbf{x}^{(t)}) - \min f(\mathbf{x}^{(t)})}, \qquad i = 1,2, \dots, NF \tag{55}$$

şeklinde hesaplanmaktadır. Burada, $u_i \in [0,1]$, $i.$ bölgenin uygunluk değerini; $\mathbf{x}_i^{(t)}$, $t.$ iterasyondaki $i.$ çözüm vektörünü; $\mathbf{x}^{(t)}$, $t.$ iterasyondaki bütün çözüm vektörlerini; $f(\mathbf{x})$, problemin amaç fonksiyonunu; $f\left(\mathbf{x}_i^{(t)}\right)$, $t.$ iterasyondaki $i.$ bölgenin amaç değerini; $\min f\left(\mathbf{x}^{(t)}\right)$, $t.$ iterasyondaki minimum amaç değerini; $\max f\left(\mathbf{x}^{(t)}\right)$, $t.$ iterasyondaki maksimum amaç değerini ve NF aile sayısını ifade etmektedir. Maksimizasyon problemleri

için uygunluk değeri ise $1-u_i$ ile tanımlanmaktadır. Uygunluk değeri ile ters orantılı maksimum yer değiştirme miktarı,

$$\mathbf{r_i} = \mathbf{r_0}\big(u_i + Precision(t,T,\epsilon,\lambda)\big), \qquad i = 1,2,\dots,NF \tag{56}$$

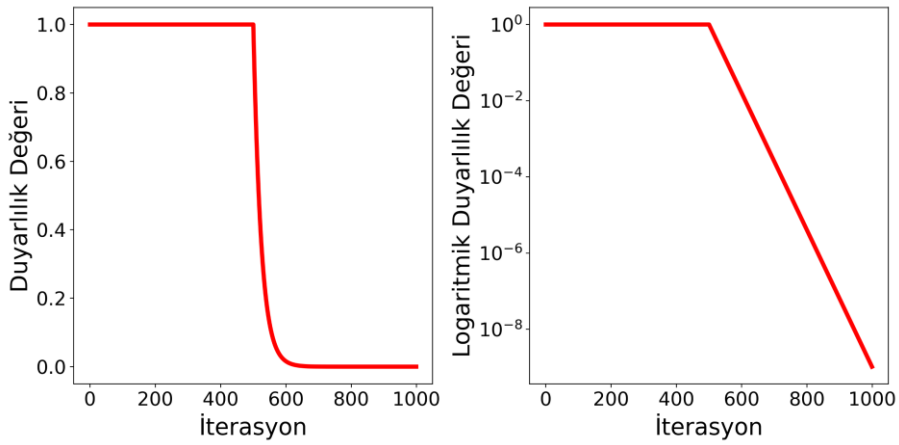
biçiminde hesaplanır. Burada, $\mathbf{r_i}$, maksimum yer değiştirme miktarını; $\mathbf{r_0}$, her boyut için arama uzayının çeyrek genişliğini ifade eder ve

$$\mathbf{r_0} = \frac{(\mathbf{x_{max}} - \mathbf{x_{min}})}{4}, \qquad \mathbf{r_0}, \mathbf{x_{max}}, \mathbf{x_{min}} \in \mathbb{R}^{\boldsymbol{D}} \tag{57}$$

şeklinde hesaplanır. Burada, $\mathbf{x_{max}}$ ve $\mathbf{x_{min}}$, sırasıyla arama uzayının üst ve alt sınır vektörlerini temsil etmektedir. Ayrıca, Eşitlik (56)'da yer alan $Precision(t,T,\epsilon,\lambda)$, iki parçalı duyarlılık fonksiyonunu ifade etmektedir. Duyarlılık fonksiyonu,

$$Precision(t\,,T,\epsilon,\lambda = 0.5) = \begin{cases} 1 & t \le \lambda T \\ \epsilon^{\left(\frac{t-\lambda T}{T-\lambda T}\right)} & t > \lambda T \end{cases}, \qquad (t = 1,2,\dots,T) \tag{58}$$

biçiminde tanımlanmaktadır. Burada, T, maksimum iterasyon sayısını; t, mevcut iterasyonu; ϵ, beklenen minimum duyarlılığı ifade etmektedir. λ, $(0,1)$ aralığında değişen sabit bir değerdir ve λT ile tanımlanan duyarlılık fonksiyonunun kesim noktasını belirlemektedir. Şekil 4'te duyarlılık fonksiyonu verilmektedir.



Şekil 4. Duyarlılık fonksiyonu

Duyarlılık fonksiyonu, λT noktasına kadar sabitken, λT noktasından sonra üstel olarak azalmaktadır. Bu çalışmada, $\lambda = 0.5$ alındığından, λT maksimum iterasyon sayısının yarısını ifade etmektedir. Bununla birlikte, λT, λ'ya bağlı olarak değişkenlik gösterebilmektedir.

Duyarlılık fonksiyonunun kesim noktasına kadar genel çözüm aranır. Daha sonra, genel çözümün hassasiyeti (duyarlılığı) iyileştirilir. Meta-sezgisel optimizasyon algoritmalarında, çözümün hassasiyeti bireylerin etkileşimine bağlıdır. Bu nedenle, çözümün duyarlılığına müdahale edilememektedir. Ancak, bu çalışmada kullanılan duyarlılık fonksiyonu sayesinde istenilen duyarlılığa ϵ değeri kullanılarak ulaşılabilir.

2.2.3. Ana Çekirgenin Yeni Yavrularının Üretilmesi

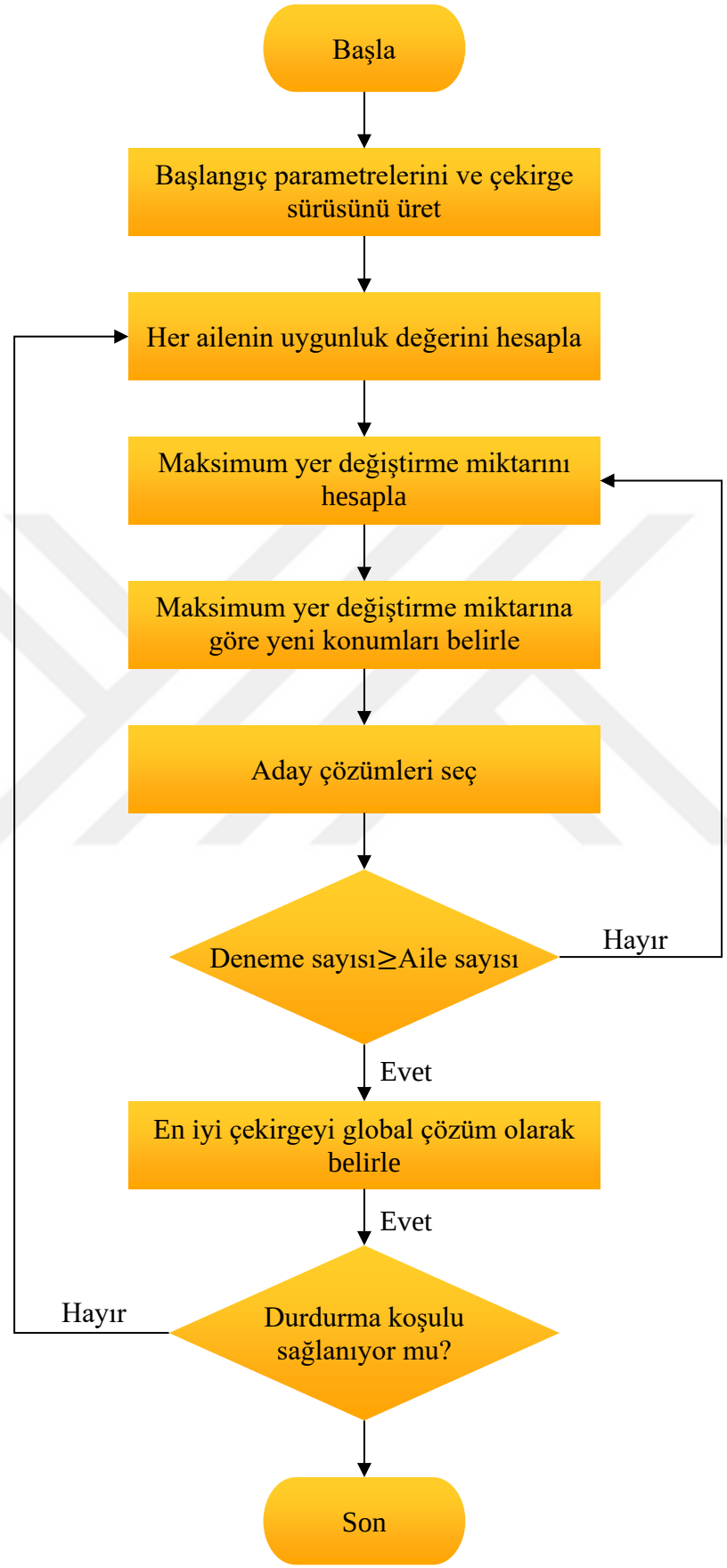
Bir yavrunun konumu maksimum yer değiştirme miktarı ve ana çekirgenin mevcut konumuna bağlı olarak,

$$v_{k,J} = x_{i,J} + rand(-\mathbf{r}_i, \mathbf{r}_i) \tag{59}$$

biçiminde tanımlanmaktadır. Burada, $v_{k,J}$ ve $x_{i,J}$ sırasıyla k. yavrunun ve i. ana çekirgenin konumlarını ifade etmektedir. J ise problem uzayının rastgele bir boyutudur.

Çekirgeler maksimum yer değiştirme miktarına göre yeni konumlarına yerleşir ve bulundukları bölgelerin uygunluk değerleri hesaplanır. Daha sonra, en iyi bölgenin konumu güncellenir. Böylece, belirli bir iterasyon sayısı ya da durdurma koşulu sağlanana kadar global çözüm aranmaya devam eder.

Önerilen ALSO algoritmasının akış diyagramı Şekil 5'te ve sözde kodu Algoritma 1'de verilmiştir.



Şekil 5. ALSO algoritmasının akış diyagramı

Algoritma 1. ALSO algoritmasının sözde kodu

```
function also(f, N, D, T, eps,xmin,xmax)
" f(.): Amaç fonksiyonu
" N   : Toplam çekirge sayısı
" D   : Amaç fonksiyonunun boyutu
" T   : Maksimum iterasyon sayısı
" eps : Beklenen minimum duyarlılık değeri
" xmin,xmax: Arama uzayının alt ve üst sınırları
" precision(.)   : Duyarlılık fonksiyonu
"   D-boyutlu uzayda tüm çekirgelerin konumlarının
   oluşturulması
FS = D / 2  "Ailedeki çekirge sayısı
NF = N / FS        " Aile sayısı

r0_j = (xmax_j − xmin_j) / 4 → (j = 1,2, … , D)
for family_i in (1 to NF) do
    x_{i,j} = rand(xmin_j , xmax_j) → (j = 1,2, … , D)
    f_i = f(x_{i,:})

" iterasyonu başlatılması
for t in (1 to T) do

    " sosyal faz
    for family_i in (1 to NF) do
        " uygunluk değerlerini güncellenmesi
        u_i = (f_i − min(f)) / (max(f) − min(f))

    " ailesel faz
    for family_i in (1 to NF) do
        xNext_j = x_{i,j} → (j = 1,2, … , D) " güncellenen çekirge

        " yeni konumların denenmesi
        for trial_k in (1 to FS) do
            v_j = x_{i,j} → (j = 1,2, … , D)
            J = irand(1, D) " işlenecek boyutun seçilmesi

            " maksimum yer değiştirme miktarının gücellenmesi
            r = r0_J ∗ (u_i + Precision(t, T, eps))

            " aday çözümlerin güncellenmesi
            v_J = x_{i,J} + rand(−r, r)

            " J.boyutun sınırlarının kontrol edilmesi
            if is not xmin_J < v_J < xmax_J
                v_J = rand(xmin_J, xmax_J)

            " aday çözümün seçilmesi
            if f(v) < f_i
                xNext_j = v_j → (j = 1,2, … , D)
                f_i = f(v)
        " i. en iyi konumun güncellenmesi
        x_{i,j} = xNext_j → (j = 1,2, … , D)
return x
```

Algoritma 1'de `rand(a,b)` fonksiyonu $[a, b)$ aralığında düzgün dağılımdan rastgele bir sayı üretir ve `irand(a,b)` fonksiyonu ise $[a, b]$ aralığında tamsayılar kümesinden rastgele örnekleme yapmaktadır.

3. BULGULAR VE İRDELEME

Bu bölümde, önerilen ALSO algoritmasının performansını ölçmek için yaygın ve güncel optimizasyon algoritmaları ile kapsamlı bir değerlendirme ve deneysel bir çalışma yapılmıştır. Yeni geliştirilen bir algoritmanın performansının doğrulanması ve mevcut algoritmalarla karşılaştırılması gerekir. Bu amaçla, bir optimizasyon algoritmasının başarımı literatürde tanımlanmış test fonksiyonlarına uygulanarak ölçülmektedir. Sayısal optimizasyon problemleri üzerinde iyi performans gösteren algoritmalar gerçek yaşam problemlerini çözmek için etkili yöntemler olarak kabul edilir. Bu çalışmada, literatürdeki standart test fonksiyonları üzerinde çalışılmıştır. Tüm işlemler, Intel Core i5-6400 CPU 2.70 GHz işlemcili, 8 GB RAM özellikli bir bilgisayar ortamında ve MATLAB R2019a yazılımı kullanılarak gerçekleştirilmiştir.

Simülasyon çalışmasında, ALSO algoritması, tek tepeli, çok tepeli ve birleşik test fonksiyonları olmak üzere üç kategoride incelenmiştir. Tek tepeli test fonksiyonları, arama uzayında sadece bir tane optimal çözümü olan fonksiyonlardır ve optimizasyon algoritmasının sömürü yeteneğini anlamasına yardımcı olduğundan, algoritmanın yerel arama performansını ölçmek için kullanılır. Tablo 1, tek tepeli test fonksiyonlarının isimlerini, tanım aralıklarını ve minimum değerlerini göstermektedir. Şekil 6'da tek tepeli test fonksiyonlarının grafiksel gösterimi verilmiştir. Çok tepeli test fonksiyonları birden fazla yerel ve global optimuma sahiptir. Bu yüzden, global çözüme yerel optimumlara takılmadan ulaşmak için optimizasyon algoritmasının keşif yeteneğine odaklanmasını sağlar. Tablo 2'de çok tepeli test fonksiyonlarının isimleri, tanım aralıkları ve minimum değerleri, Şekil 7'de ise grafiksel gösterimleri verilmiştir. Birleşik test fonksiyonları CEC2017 test kümesinde (Awad vd., 2016) tanımlanan fonksiyonlar olup, temel fonksiyonlarının birleşimiyle elde edilen karmaşık fonksiyonlardır. Zor optimizasyon problemlerini temsil eden bu fonksiyonlar, tek ve çok tepeli fonksiyonların kaydırılması, döndürülmesi ve genişletilmesi ile oluşturulur. Birleşik test fonksiyonları, bir optimizasyon algoritmasının en iyi çözüme ulaşmak için keşif ve sömürü arasında kurduğu dengeyi anlamaya yardımcı olur. Tablo 3'te birleşik test fonksiyonlarının özellikleri ve Şekil 8'de bu fonksiyonların grafiksel gösterimleri verilmiştir.

Tablo 1. Tek tepeli test fonksiyonları

F	İsim	Fonksiyon	Tanım Aralığı	Global Minimum
F1	*Sphere*	$f_1(\vec{x}) = \sum_{j=1}^{D} x_j^2$	$-100 \le x_j \le 100$	$f_1(\vec{0}) = 0$
F2	*Sphere M1*	$f_2(\vec{x}) = \sum_{j=1}^{D} (x_j - 1)^2$	$-100 \le x_j \le 100$	$f_2(\vec{1}) = 0$
F3	*Exponential M1*	$f_3(\vec{x}) = 1 - \exp\left(-0.5 \sum_{j=1}^{D} (x_j - 1)^2\right)$	$0 \le x_j \le 2$	$f_3(\vec{1}) = 0$
F4	*Sum of Different Powers M1*	$f_4(x) = \sum_{j=1}^{D} \lvert x_j - 1\rvert^{j+1}$	$-10 \le x_j \le 10$	$f_4(\vec{1}) = 0$
F5	*Schwefel 2.22 M1*	$f_5(\vec{x}) = \sum_{j=1}^{D} \lvert x_j - 1\rvert + \prod_{j=1}^{D} \lvert x_j - 1\rvert$	$-100 \le x_j \le 100$	$f_5(\vec{1}) = 0$
F6	*Sum Squares M1*	$f_6(x) = \sum_{j=1}^{D} j(x_j - 1)^2$	$-10 \le x_j \le 10$	$f_6(\vec{1}) = 0$

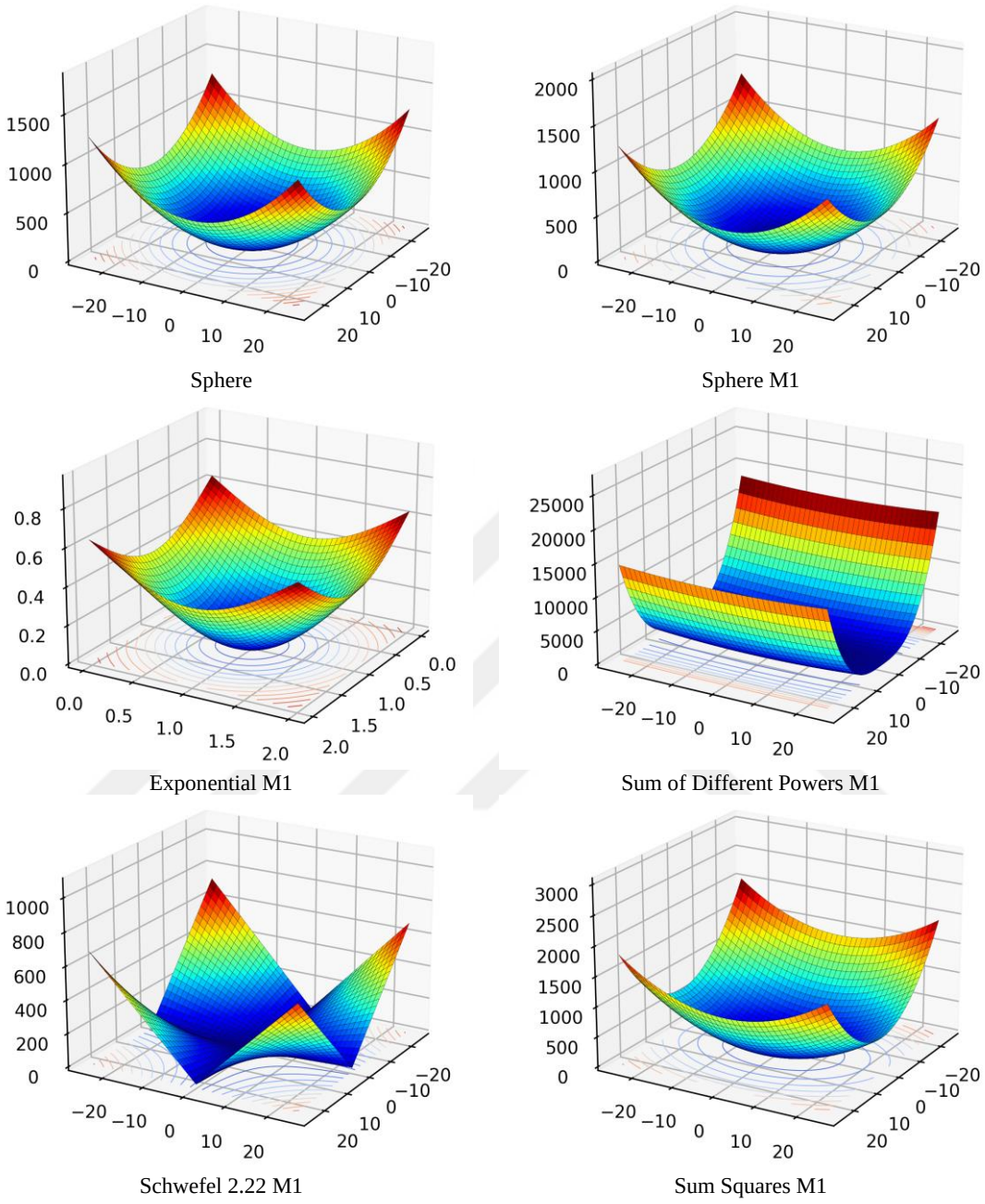
Tablo 2. Çok tepeli test fonksiyonları

F	İsim	Fonksiyon	Tanım Aralığı	Global Minimum
F7	*Ackley*	$f_7(\vec{x}) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D}\sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D}\sum_{j=1}^{D} \cos(2\pi x_j)\right) + 20 + \exp(1)$	$-32.768 \le x_j \le 32.768$	$f_7(\vec{0}) = 0$
F8	*Rastrigin*	$f_8(\vec{x}) = \sum_{j=1}^{D} (x_j^2 - 10\cos(2\pi x_j) + 10)$	$-5.12 \le x_j \le 5.12$	$f_9(\vec{0}) = 0$
F9	*Rosenbrock*	$f_9(\vec{x}) = \sum_{j=1}^{D-1} 100(x_{j+1} - x_j^2)^2 + (1 - x_j)^2$	$-5 \le x_j \le 5$	$f_{10}(\vec{1}) = 0$
F10	*Griewank*	$f_{10}(\vec{x}) = 1 + \sum_{j=1}^{D} \frac{x_j^2}{4000} - \prod_{j=1}^{D} \cos\left(\frac{x_j}{\sqrt{j}}\right)$	$-600 \le x_j \le 600$	$f_8(\vec{0}) = 0$
F11	*Schwefel*	$f_{11}(\vec{x}) = 418.9829 \cdot D - \sum_{j=1}^{D} x_j \sin\left(\sqrt{\lvert x_j\rvert}\right)$	$-500 \le x_j \le 500$	$f_{11}(\overrightarrow{420.9687}) \approx 0$
F12	*Weierstrass*	$f_{12}(\mathbf{x}) = \sum_{j=1}^{D} \left(\sum_{k=0}^{20} \frac{1}{2^k} \cos\left(2\pi 3^k \left(x_j + \frac{1}{2}\right)\right)\right) - D \sum_{k=0}^{20} \frac{1}{2^k} \cos(\pi 3^k)$	$-5 \le x_j \le 5$	$f_{12}(\vec{0}) = 0$

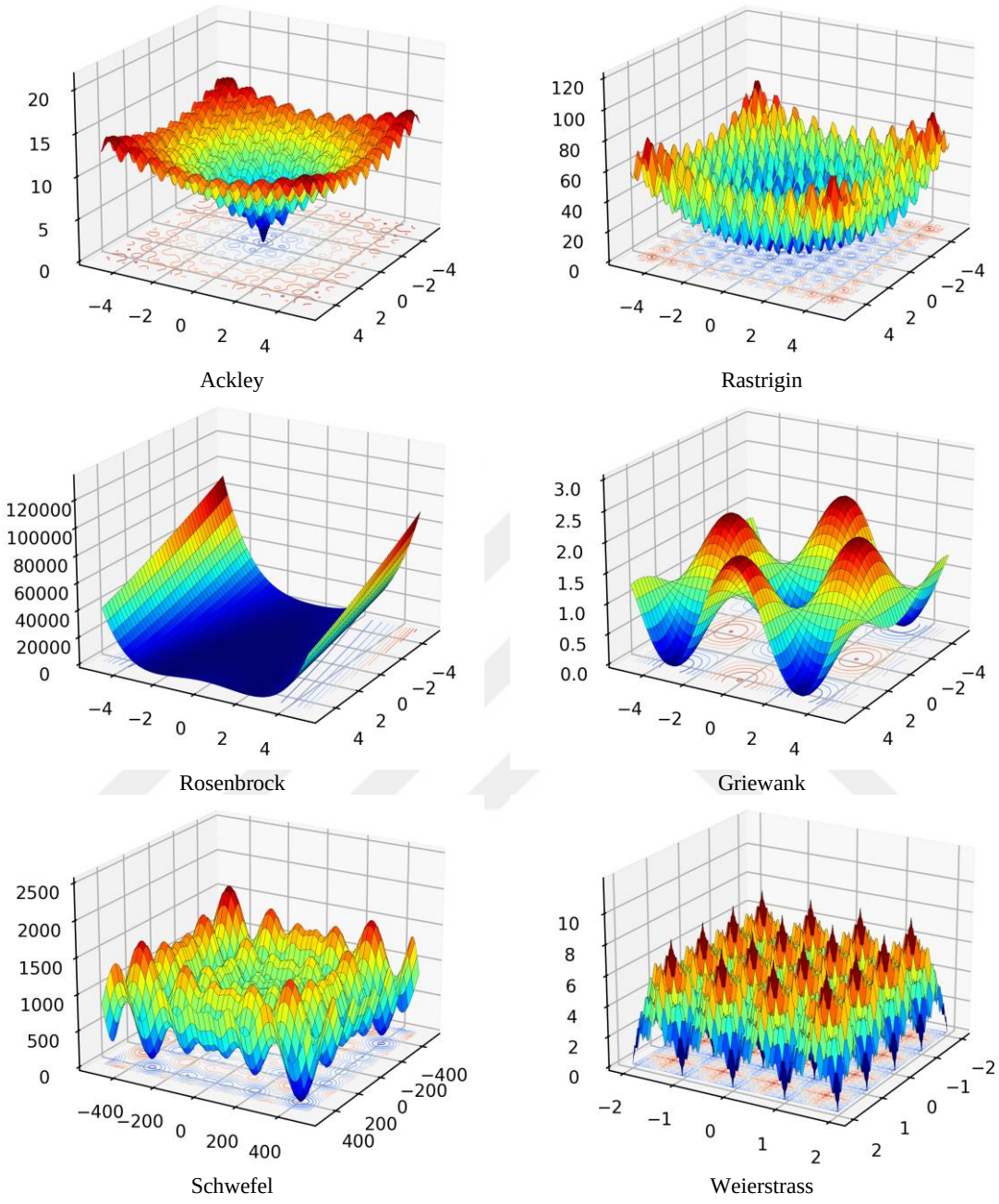
Tablo 3. Birleşik test fonksiyonları

F	İsim	Birleştirilen Fonksiyonlar	Parametreler
F13	Birleşik Fonksiyon 1	$g_1(x)$: $Rosenbrock$ $g_2(x)$: $High\ Conditioned\ Elliptic$ $g_3(x)$: $Rastrigin$	$\sigma = [10,20,30]$ $\lambda = [1,1e-6,1]$ $bias = [0,100,200]$
F14	Birleşik Fonksiyon 2	$g_1(x)$: $Rastrigin$ $g_2(x)$: $Griewank$ $g_3(x)$: $Modifed\ Schwefel$	$\sigma = [10,20,30]$ $\lambda = [1,10,1]$ $bias = [0,100,200]$
F15	Birleşik Fonksiyon 3	$g_1(x)$: $Rosenbrock$ $g_2(x)$: $Ackley$ $g_3(x)$: $Modified\ Schwefel$ $g_4(x)$: $Rastrigin$	$\sigma = [10,20,30,40]$ $\lambda = [1,10,1,1]$ $bias = [0,100,200,300]$
F16	Birleşik Fonksiyon 4	$g_1(x)$: $Ackley$ $g_2(x)$: $High\ Conditioned\ Elliptic$ $g_3(x)$: $Girewank$ $g_4(x)$: $Rastrigin$	$\sigma = [10,20,30,40]$ $\lambda = [10,10,1e-6,10,1]$ $bias = [0,100,200,300]$
F17	Birleşik Fonksiyon 5	$g_1(x)$: $Rastrigin$ $g_2(x)$: $Happycat$ $g_3(x)$: $Ackley$ $g_4(x)$: $Discus$ $g_5(x)$: $Rosenbrock$	$\sigma = [10,20,30,40,50]$ $\lambda = [10,1,10,1e-6,1]$ $bias = [0,100,200,300,400]$
F18	Birleşik Fonksiyon 6	$g_1(x)$: $Expanded\ Scaffer$ $g_2(x)$: $Modified\ Schwefel$ $g_3(x)$: $Griewank$ $g_4(x)$: $Rosenbrock$ $g_5(x)$: $Rastrigin$	$\sigma = [10,20,30,40,50]$ $\lambda = [1e-26,10,1e-6,10,5e-4]$ $bias = [0,100,200,300,400]$
F19	Birleşik Fonksiyon 7	$g_1(x)$: $HGBat$ $g_2(x)$: $Rastrigin$ $g_3(x)$: $Modified\ Schwefel$ $g_4(x)$: $Bent-Cigar$ $g_5(x)$: $High\ Conditioned\ Elliptic$ $g_6(x)$: $Expanded\ Scaffer$	$\sigma = [10,20,30,40,50,60]$ $\lambda = [10,10,2.5,1e-26,1e-6,5e-4]$ $bias = [0,100,200,300,400,500]$
F20	Birleşik Fonksiyon 8	$g_1(x)$: $Ackley$ $g_2(x)$: $Griewank$ $g_3(x)$: $Discus$ $g_4(x)$: $Rosenbrock$ $g_5(x)$: $HappyCat$ $g_6(x)$: $Expanded\ Scaffer$	$\sigma = [10,20,30,40,50,60]$ $\lambda = [10,10,1e-6,1,1,5e-4]$ $bias = [0,100,200,300,400,500]$
F21	Birleşik Fonksiyon 9	$g_1(x)$: $Hybrid\ Fonksiyon\ 5$ $g_2(x)$: $Hybrid\ Fonksiyon\ 8$ $g_3(x)$: $Hybrid\ Fonksiyon\ 9$	$\sigma = [10,30,50]$ $\lambda = [1,1,1]$ $bias = [0,100,200]$
F22	Birleşik Fonksiyon 10	$g_1(x)$: $Hybrid\ Fonksiyon\ 5$ $g_2(x)$: $Hybrid\ Fonksiyon\ 6$ $g_3(x)$: $Hybrid\ Fonksiyon\ 7$	$\sigma = [10,30,50]$ $\lambda = [1,1,1]$ $bias = [0,100,200]$

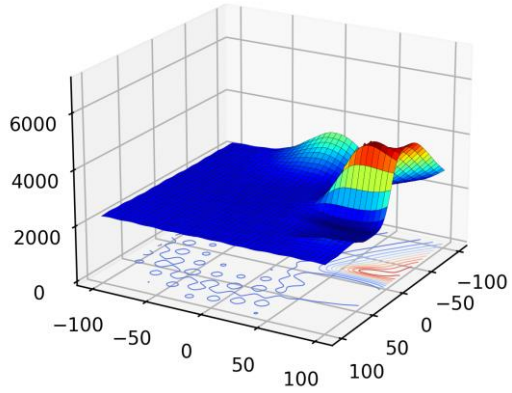
Tablo 3'te, $g(x)$ birleştirme için kullanılan fonksiyonları göstermektedir. σ ve λ sırasıyla her fonksiyonun yayılım aralığını ve basıklığını ifade etmektedir. $bias$ ise birleştirme için kullanılan fonksiyonların global optimum değerini belirtmektedir.



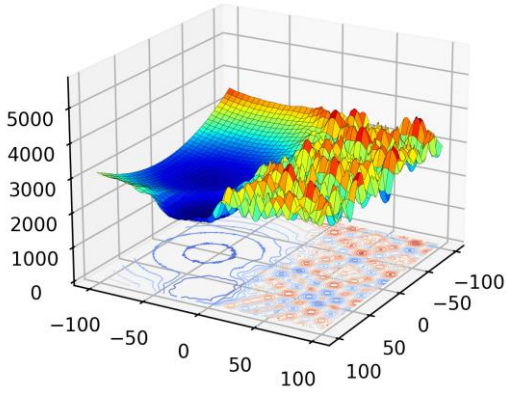
Şekil 6. Tek tepeli test fonksiyonları



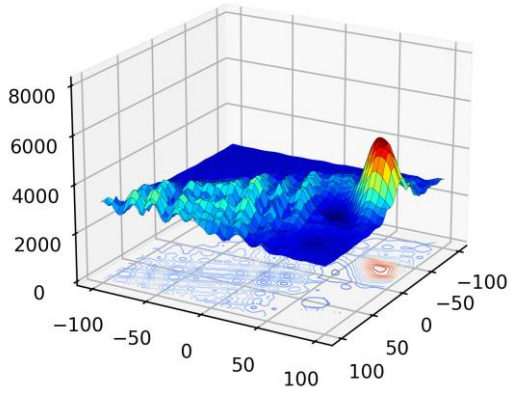
Şekil 7. Çok tepeli test fonksiyonları



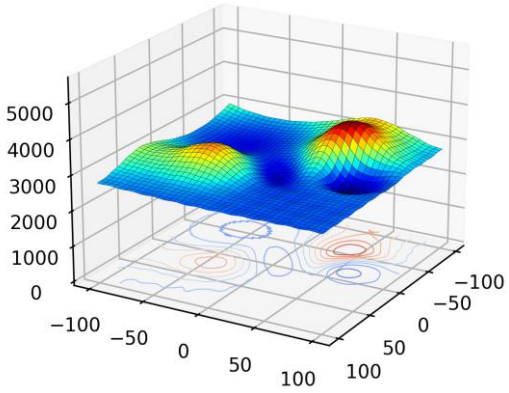
Birleşik Fonksiyon 1



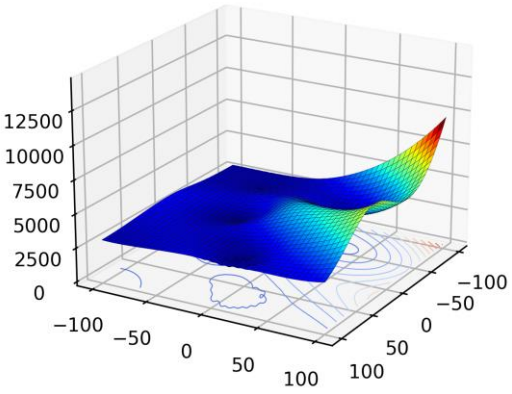
Birleşik Fonksiyon 2



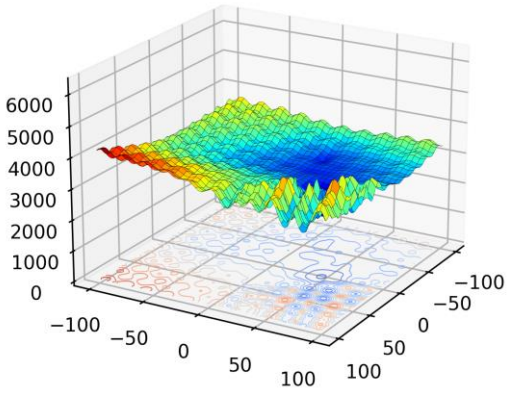
Birleşik Fonksiyon 3



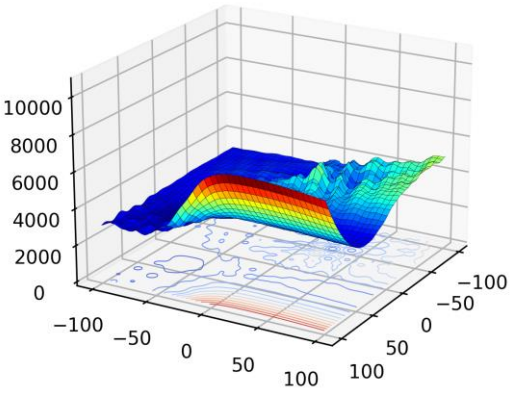
Birleşik Fonksiyon 4



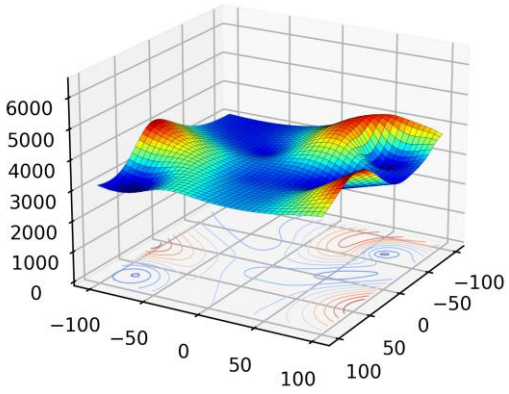
Birleşik Fonksiyon 5



Birleşik Fonksiyon 6



Birleşik Fonksiyon 7



Birleşik Fonksiyon 8

Şekil 8. Birleşik test fonksiyonları

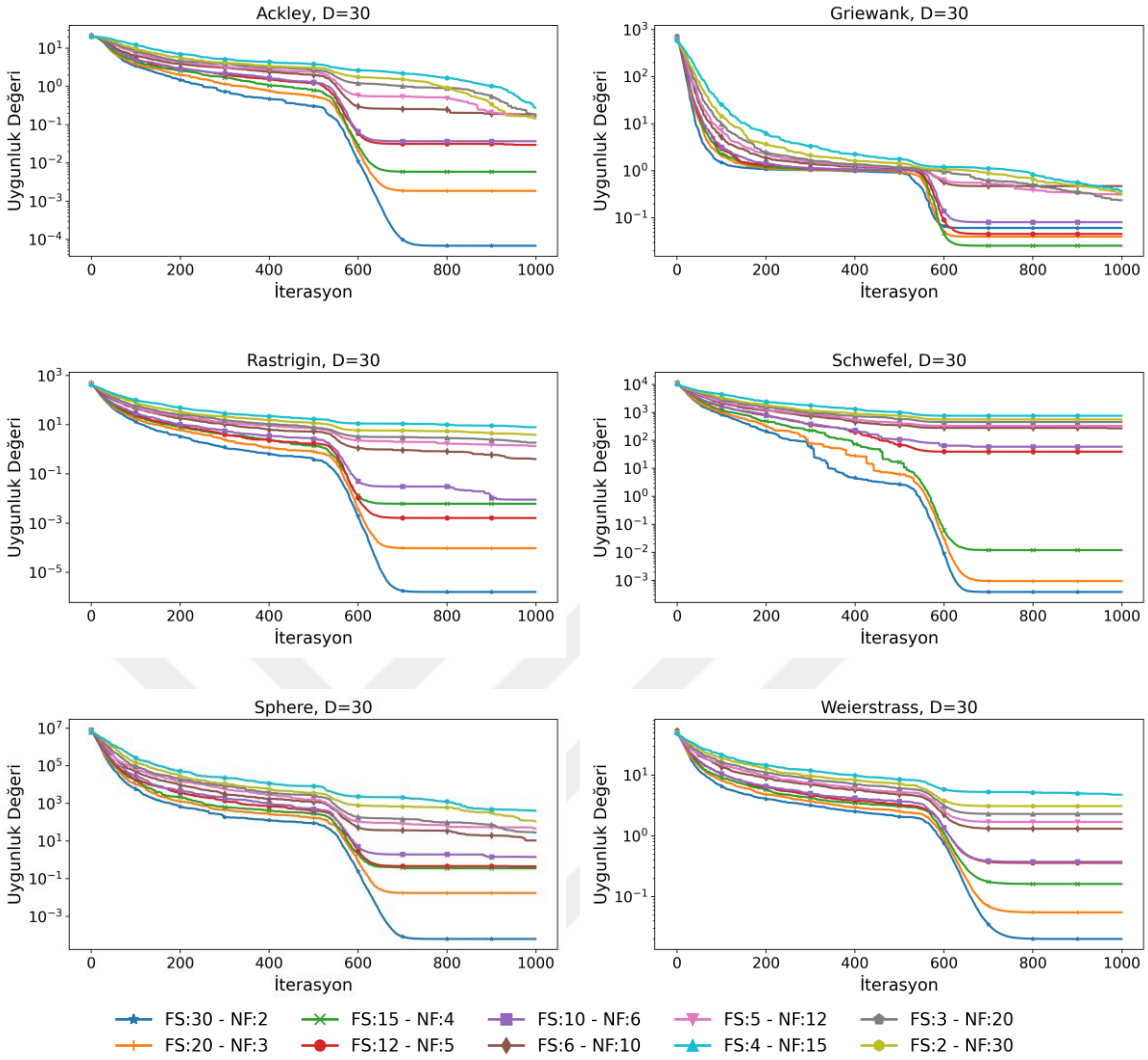
Bu çalışmada, ilk olarak, simülasyon çalışması için ALSO algoritmasının parametreleri belirlenmiştir. Daha sonra, literatürde yaygın olarak kullanılan ve yeni geliştirilen optimizasyon algoritmaları ile önerilen algoritma karşılaştırılmıştır.

3.1. ALSO Algoritmasının Parametrelerine Göre Değerlendirilmesi

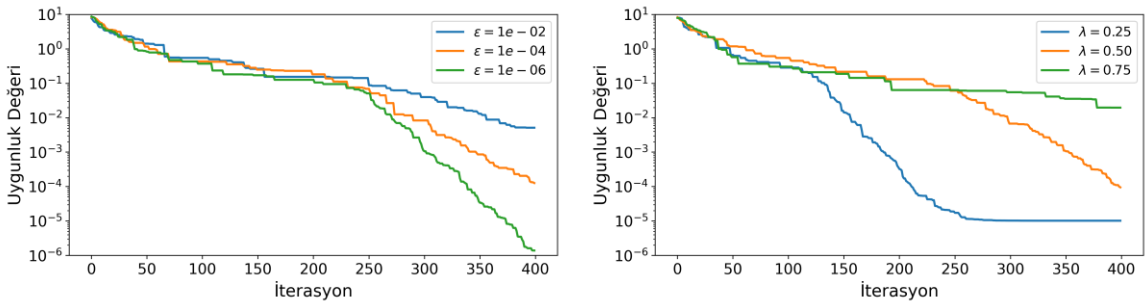
Önerilen algoritmada, en uygun aile sayısı (NF) ve her ailedeki çekirge sayısının (FS) belirlenmesi için bazı test fonksiyonları üzerinde simülasyon çalışması yapılmıştır. Aile sayısı $2, 3, 4, 5, 6, 10, 12, 15, 20$ ve 30 olarak denenmiş ve her ailede sırasıyla $30, 20, 15, 12, 10, 6, 5, 4, 3$ ve 2 çekirge kullanılmıştır. Böylece, toplam çekirge sayısı 60 olarak alınmıştır. Önerilen algoritma, $D = 30$ için 30 kez koşturulmuş ve elde edilen sonuçlar Şekil 9'da verilmiştir.

Şekil 9'a göre, önerilen ALSO algoritması, aile sayısına göre değerlendirildiğinde, genel çözümün hassasiyeti aile sayısı azaldığında artmaktadır. Bununla birlikte, her ailedeki çekirge sayısı arttığında da genel çözümün hassasiyeti artmaktadır. Griewank fonksiyonu için, $FS = 15$ olduğunda, ALSO algoritması en iyi sonucu elde etmiştir. Diğer fonksiyonlarda, önerilen algoritma en iyi sonucu $FS = 20$ ve $FS = 30$ olduğunda sağlamıştır. Özellikle Schwefel fonksiyonu için, $FS = 30$, $FS = 20$ ve $FS = 15$ olduğu durumlarda, algoritmanın yakınsama eğrileri birbirlerine yakın sonuçlar vermektedir. Dolayısıyla, keşif ve sömürü arasındaki denge göz önüne alındığında, simülasyon çalışmasında $FS = 15$ olmasının daha uygun olacağı çıkarımı yapılabilir. Bu nedenle, bu çalışmada, bir ailedeki çekirge sayısı boyutun yarısı olarak belirlenmiştir.

Önerilen algoritmada, maksimum yer değiştirme miktarının belirlenmesinde kullanılan duyarlılık fonksiyonundaki ϵ ve λ parametrelerinin etkisini incelemek amacıyla, Ackley fonksiyonu için en iyi uygunluk değerlerindeki değişim Şekil 10'da gösterilmiştir.



Şekil 9. ALSO algoritmasının parametrelerine göre değerlendirilmesi



(a) (b)

Şekil 10. ϵ ve λ parametrelerine göre en iyi uygunluk değerleri (a) $\lambda = 0.5$ olduğunda ϵ parametresine göre en iyi uygunluk değerleri, (b) $\epsilon = 1e - 04$ olduğunda λ parametresine göre en iyi uygunluk değerleri

Şekil 10(a)'dan önerilen ALSO algoritmasının, toplam iterasyon sayısının yarısına ($t < 200$) kadar arama alanını keşfetmeye, diğer yarısında ise sömürmeye odaklandığı sonucu çıkarılabilir. Bu nedenle, yakınsama eğrileri toplam iterasyon sayısının yarısında benzer bir azalma eğilimi göstermektedir. İterasyonun son yarısındaki ϵ değerlerindeki fark, sömürünün ne kadar geliştiğini ortaya koymaktadır.

Şekil 10(b)'de, $\lambda = 0.25$ ve $\epsilon = 1e-04$ olduğunda yakınsama eğrileri, toplam iterasyonun ilk çeyreğinde, algoritmanın arama alanını keşfettiğini gösterirken, diğer çeyreklerinde sömürüye odaklandığını işaret etmektedir. Bu durumda, erken yakınsamalar meydana gelebilir ve algoritmanın yerel çözümlere takılmasına sebep olabilir. Bununla birlikte, $\lambda = 0.75$ olduğunda yakınsama eğrileri, üçüncü çeyrekte ($t > 300$) algoritmanın sömürüye odaklandığını ortaya koymaktadır. Bu durumda, genel çözüme yakınsama yeterli derecede tatmin edici olmayabilir. Sonuç olarak, simülasyon çalışmasında $\lambda = 0.5$ alınarak denemeler gerçekleştirilmiştir.

3.2. ALSO Algoritmasının Diğer Algoritmalarla Karşılaştırılması

ALSO algoritması, performansının ölçülmesi amacıyla, tek tepeli, çok tepeli ve birleşik test fonksiyonları üzerinde denenerek, yaygın ve yeni geliştirilen optimizasyon algoritmaları ile karşılaştırılmıştır. Bütün denemeler aynı koşullar altında gerçekleştirilmiş ve maksimum iterasyon sayısı $T = 3000$ olarak belirlenmiştir. Ayrıca, algoritma 30, 50 ve 100 boyutlarında her fonksiyon için test edilmiştir. Bununla birlikte, simülasyon çalışmasında, ALSO algoritması, PSO (parçacık sürü optimizasyonu) (Lynn ve Suganthan, 2015), ABC (yapay arı koloni algoritması) (Karaboga ve Basturk, 2007a), CS (guguk kuşu araması) (Yang ve Deb, 2009), BAT (yarasa algoritması) (Yang, 2010), GWO (gri kurt optimizasyonu) (Mirjalili vd., 2014), WOA (balina optimizasyon algoritması) (Mirjalili ve Lewis, 2016), HHO (harris şahini optimizasyonu) (Heidari vd., 2019), BRO (battle royal optimizasyon algoritması) (Farshi, 2021), ve RSO (fare sürüsü optimizasyonu) (Dhiman vd., 2021) algoritmalarıyla karşılaştırılmıştır. Ayrıca, algoritmaların kontrol parametreleri orijinal makalelere göre ayarlanmış olup, bütün algoritmalarda arama ajanlarının sayısı 150 olarak belirlenmiştir.

Meta-sezgisel optimizasyon algoritmalarının performansını analiz etmek için bazı doğrulama kriterleri göz önünde bulundurulur. Bunlar, genellikle belirli bir iterasyon sonucunda elde edilen amaç fonksiyonunun ortalama, standart sapma ve algoritmanın

hesaplama zamanıdır. Hesaplama zamanı, bir algoritmanın çalışma süresini; ortalama, doğruluğunu ve standart sapma ise, tutarlılığını göstermektedir. Bu nedenle, ALSO ve karşılaştırılan diğer algoritmalar 30 kez koşturulmuş ve fonksiyonların en iyi değerlerine göre elde edilen en iyi ortalama ve standart sapma değerleriyle birlikte çalışma zamanları da hesaplanmıştır. Karşılaştırma sonuçlarında, makine epsilon değerinden $(2.22E-16)$ küçük değerlerin kullanılması anlamsız olacağından, bu değerden küçük olanlar makine epsilon değerine eşitlenmiştir.

Tablo 4, 30 boyut için tek tepeli test fonksiyonlarından elde edilen sonuçları göstermektedir. Buna göre, ALSO, PSO, GWO, WOA, HHO, BRO ve RSO, F1 fonksiyonu için en iyi değeri vermiştir. F2 fonksiyonu, F1 fonksiyonunun 1 birim sağa kaydırılmış şeklidir. Bu fonksiyonda ise en iyi değerleri ALSO, PSO ve BRO sağlamıştır. F2 fonksiyonu, F1 fonksiyonunun kaydırılmış hali olmasına rağmen, GWO, WOA, HHO ve RSO F1 fonksiyonundaki kadar etkili olamamıştır. Bununla birlikte, ALSO, PSO, ABC, CS ve BRO, F3 fonksiyonunda güzel sonuçlar elde etmiştir. F4 ve F5 fonksiyonlarında, ALSO diğer algoritmalarla kıyaslandığında daha iyi bir performans göstermiştir. F6 fonksiyonunda ise, ALSO ve BRO iyi bir performans sergilemiştir. Bunların yanı sıra, diğer algoritmalarla karşılaştırıldığında, çoğu durumda ALSO algoritmasının daha iyi bir standart sapma sağladığı gözlenmektedir. Ayrıca, 30 boyut için 6 tek tepeli test fonksiyonunda, çalışma zamanı açısından değerlendirildiğinde, ALSO algoritması karşılaştırılan diğer optimizasyon algoritmalarına göre daha hızlı çalışmaktadır.

50 boyut için tek tepeli test fonksiyonlarından elde edilen sonuçlar Tablo 5'te verilmektedir. Bu sonuçlara göre, F1 fonksiyonu için ALSO, PSO, GWO, WOA, HHO ve RSO en iyi sonuçları elde etmiştir. F2 fonksiyonunda, en iyi değerleri ALSO ve PSO sağlamıştır. F2 fonksiyonu F1 fonksiyonunun kaydırılmış hali olmasına rağmen GWO, WOA, HHO ve RSO, F1 fonksiyonundaki kadar etkili çalışmamıştır. ALSO, PSO ve BRO, F3 fonksiyonunda en iyi performansı sergilemiştir. F4, F5 ve F6 fonksiyonlarında, ALSO diğer algoritmalarla kıyaslandığında daha iyi sonuçlar ortaya koymuştur. Bununla birlikte, ALSO, çoğu problemde yakın doğrulukla global optimuma ulaşmıştır. Ayrıca, yapılan denemeler, ALSO algoritmasının genel olarak daha hızlı sonuçlar verdiğini göstermektedir.

Tablo 6, 100 boyutta tek tepeli test fonksiyonlarından elde edilen sonuçları göstermektedir. Tablo 6'daki sonuçlara göre, GWO, WOA, HHO ve RSO, F1 fonksiyonunda en iyi değerleri elde etmiştir. Bununla birlikte, F1 fonksiyonunda, ALSO iyi bir performans göstermiştir. Ayrıca, ALSO ve HHO, F2 fonksiyonunda diğer algoritmalara

göre daha iyi sonuçlar ortaya koymuştur. Bununla birlikte, F2 fonksiyonu, F1 fonksiyonunun kaydırılmış hali olmasına rağmen, GWO, WOA ve RSO, F1 fonksiyonundaki kadar etkili olamamıştır. F3 fonksiyonunda, karşılaştırılan diğer algoritmalara göre, ALSO ve PSO daha iyi sonuçlar elde etmiştir. F4 ve F5 fonksiyonlarında, ALSO diğer tüm algoritmalardan daha iyi bir performans sergilemiştir. ALSO ve HHO, F6 fonksiyonu için en iyi sonuçları sunmuştur. Boyut 100 olduğunda, 6 tek tepeli test fonksiyonundan elde edilen standart sapmalara dayanarak, diğer algoritmalarla karşılaştırıldığında, ALSO algoritmasının daha tutarlı ve doğru çalıştığı gözlenmiştir. Bununla birlikte, çalışma zamanına göre kıyaslama yapıldığında ise, ALSO algoritmasının daha hızlı olduğu görülmüştür.

Tablo 4-6'dan görülebileceği gibi, ALSO, 30, 50 ve 100 boyutlarında tek tepeli test fonksiyonları için rekabetçi sonuçlar elde etmiştir. Buna ek olarak, GWO, WOA, HHO ve RSO algoritmalarının, çözüm 0 noktasında olduğunda daha iyi çalıştığı gözlenmiştir. Ayrıca, bu algoritmalar, çözüm 0 noktasından uzaklaştıkça optimal değerden uzaklaşmaktadır.

Tablo 4. 30 boyut için tek tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	2.22E-16	2.22E-16	3.93E-15	1.12E-11	3.25E+01	2.22E-16	2.22E-16	2.22E-16	2.22E-16	2.22E-16
F1	std	2.47E-32	2.47E-32	5.53E-15	3.53E-12	9.75E+01	2.47E-32	2.47E-32	2.47E-32	2.47E-32	2.47E-32
	süre(sn)	1.16	16.50	12.02	8.51	3.38	3.96	2.15	7.03	13.47	2.28
	ort	2.22E-16	2.22E-16	3.18E-15	1.01E-11	4.22E+01	2.97E-01	4.11E-06	2.30E-07	2.22E-16	2.47E+01
F2	std	2.47E-32	2.47E-32	3.34E-15	3.60E-12	1.40E+02	5.23E-01	1.61E-06	3.10E-07	2.47E-32	1.11E+00
	süre(sn)	1.16	16.36	12.02	8.51	3.37	3.96	1.99	6.87	13.47	2.17
	ort	5.74E-16	2.22E-16	2.22E-16	6.96E-16	4.68E-06	1.25E-06	2.18E-06	9.30E-07	2.83E-16	9.41E-01
F3	std	1.38E-16	8.01E-21	2.47E-32	1.95E-16	5.63E-07	2.73E-07	1.17E-06	1.34E-06	5.88E-17	1.61E-02
	süre(sn)	1.18	16.29	12.14	8.53	3.39	3.98	2.01	7.04	13.49	2.24
	ort	2.80E-14	1.05E+03	1.26E-01	1.00E+10	2.46E-04	3.34E-02	6.18E-04	1.23E-05	3.74E-08	2.38E+01
F4	std	2.13E-13	7.55E+03	1.52E-01	0.00E+00	3.77E-05	1.80E-01	3.68E-04	2.61E-05	2.06E-08	1.29E+00
	süre(sn)	1.68	16.87	12.78	9.84	3.92	4.50	2.53	8.21	14.02	2.68
	ort	9.24E-15	7.66E+01	9.50E+01	1.00E+10	3.11E+27	9.13E-01	3.04E-02	3.71E-03	5.40E+02	2.52E+01
F5	std	1.68E-15	8.01E+01	7.48E+01	0.00E+00	2.30E+28	9.12E-01	2.21E-02	2.81E-03	1.24E+02	7.39E-01
	süre(sn)	1.16	16.40	12.18	8.82	3.33	3.96	1.99	6.86	13.45	2.18
	ort	2.22E-16	8.77E+00	7.54E-16	1.17E-12	1.55E-04	8.49E+00	2.17E-03	1.72E-06	2.22E-16	3.61E+02
F6	std	2.47E-32	3.05E+01	1.75E-15	4.96E-13	3.15E-05	7.33E+00	2.32E-03	3.20E-06	2.47E-32	1.54E+01
	süre(sn)	1.17	16.35	12.03	8.52	3.36	3.96	1.99	6.93	13.49	2.17

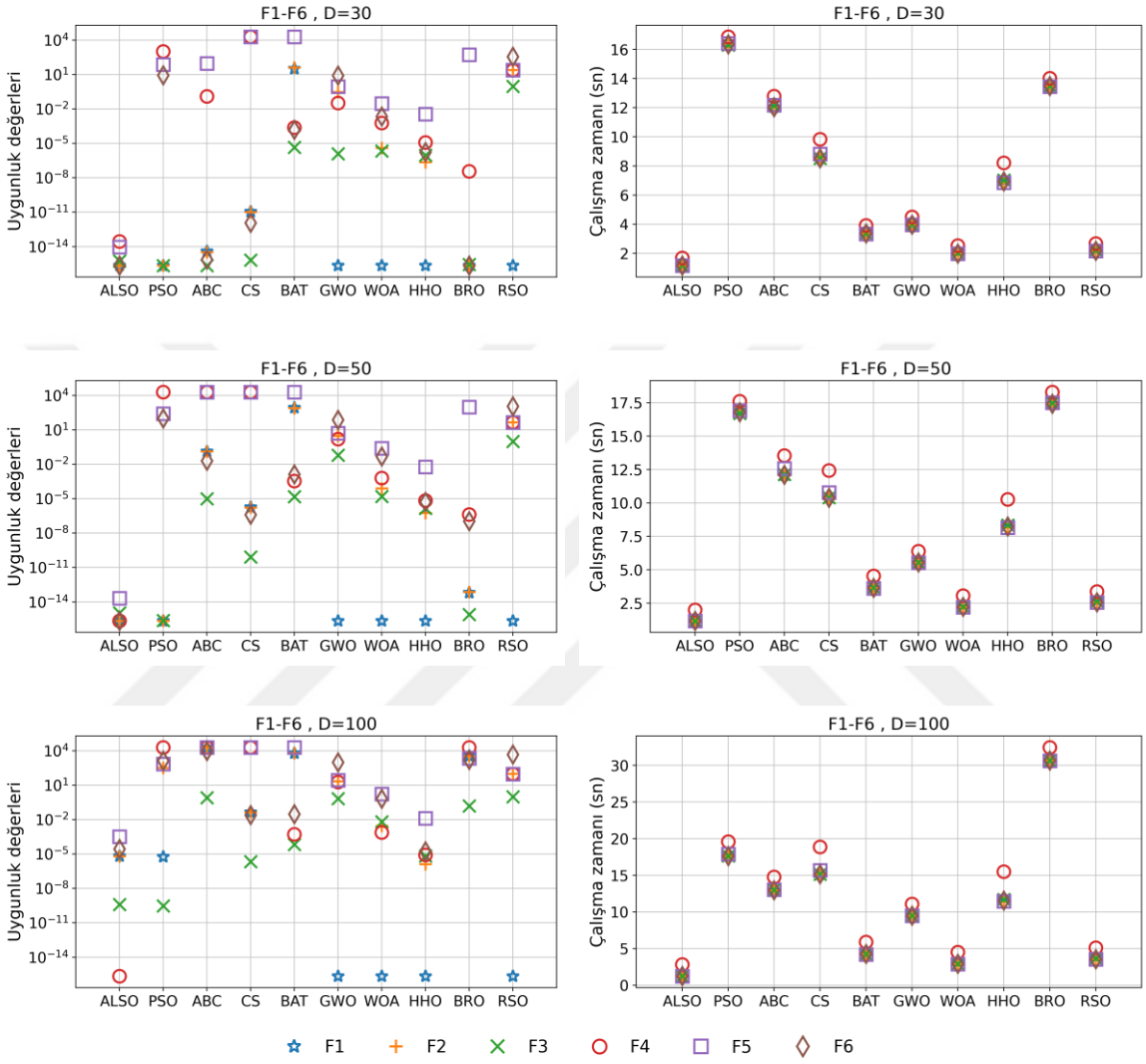
Tablo 5. 50 boyut için tek tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	2,22E-16	2,22E-16	1,30E-01	1,82E-06	8,76E+02	2,22E-16	2,22E-16	2,22E-16	6,58E-14	2,22E-16
F1	std	2,47E-32	2,47E-32	3,41E-02	4,14E-07	6,68E+02	2,47E-32	2,47E-32	2,47E-32	1,28E-13	2,47E-32
	süre(sn)	1,18	16,84	12,07	10,38	3,67	5,55	2,43	8,46	17,42	2,73
	ort	2,22E-16	2,22E-16	1,32E-01	1,67E-06	8,04E+02	3,08E+00	8,02E-05	5,88E-07	7,52E-14	4,54E+01
F2	std	2,47E-32	2,47E-32	3,89E-02	3,87E-07	7,26E+02	1,48E+00	2,77E-05	8,64E-07	2,60E-13	1,21E+00
	süre(sn)	1,17	16,83	12,07	10,39	3,65	5,54	2,22	8,16	17,40	2,56
	ort	1,13E-15	2,46E-16	1,01E-05	8,41E-11	1,53E-05	6,27E-02	1,62E-05	1,47E-06	8,05E-16	9,95E-01
F3	std	1,80E-16	4,57E-17	3,54E-06	2,02E-11	1,32E-06	1,52E-01	7,53E-06	2,67E-06	1,59E-16	3,31E-03
	süre(sn)	1,19	16,70	12,15	10,41	3,68	5,56	2,24	8,36	17,49	2,65
	ort	2,22E-16	3,16E+13	8,14E+11	1,00E+10	3,42E-04	1,57E+00	6,45E-04	7,06E-06	4,27E-07	4,49E+01
F4	std	2,47E-32	2,37E+14	1,47E+12	0,00E+00	5,96E-05	1,04E+00	3,26E-04	1,18E-05	1,84E-07	1,03E+00
	süre(sn)	2,01	17,60	13,55	12,43	4,53	6,40	3,07	10,27	18,30	3,38
	ort	2,12E-14	2,65E+02	3,97E+28	1,00E+10	1,36E+49	5,31E+00	2,49E-01	6,17E-03	1,01E+03	4,56E+01
F5	std	2,55E-15	1,21E+02	1,68E+29	0,00E+00	1,02E+50	1,81E+00	1,39E-01	3,95E-03	1,34E+02	1,04E+00
	süre(sn)	1,17	16,91	12,58	10,79	3,61	5,53	2,22	8,16	17,52	2,56
	ort	2,22E-16	1,02E+02	1,97E-02	4,04E-07	1,33E-03	7,42E+01	4,74E-02	5,24E-06	1,10E-07	1,12E+03
F6	std	2,47E-32	2,34E+02	6,06E-03	1,14E-07	2,77E-04	3,28E+01	5,95E-02	7,43E-06	6,61E-07	2,86E+01
	süre(sn)	1,18	16,79	12,10	10,40	3,64	5,55	2,22	8,24	17,43	2,57

Tablo 6. 100 boyut için tek tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	6,73E-06	5,69E-06	3,59E+04	4,38E-02	6,17E+03	2,22E-16	2,22E-16	2,22E-16	3,61E+03	2,22E-16
F1	std	6,78E-06	1,77E-05	2,51E+03	9,93E-03	3,01E+03	2,47E-32	2,47E-32	2,47E-32	7,91E+02	2,47E-32
	süre(sn)	1,24	17,66	12,95	15,13	4,30	9,55	3,22	11,94	30,64	3,91
	ort	6,47E-06	3,27E+02	3,49E+04	4,34E-02	6,31E+03	2,16E+01	2,88E-03	1,35E-06	3,45E+03	9,76E+01
F2	std	5,59E-06	1,76E+03	2,75E+03	1,13E-02	2,73E+03	2,83E+00	5,17E-04	2,66E-06	7,06E+02	5,49E+00
	süre(sn)	1,23	17,63	12,93	15,12	4,28	9,51	2,91	11,41	30,70	3,56
	ort	4,01E-10	3,10E-10	8,37E-01	2,24E-06	7,09E-05	6,82E-01	6,67E-03	6,85E-06	1,57E-01	1,00E+00
F3	std	3,08E-10	1,26E-09	2,06E-02	4,96E-07	4,61E-06	2,14E-01	5,04E-02	1,35E-05	2,64E-02	3,45E-06
	süre(sn)	1,25	17,66	13,02	15,15	4,29	9,52	2,94	11,71	30,66	3,69
	ort	2,22E-16	9,64E+44	2,92E+52	1,00E+10	5,30E-04	1,94E+01	7,81E-04	8,54E-06	7,04E+18	9,60E+01
F4	std	2,47E-32	7,33E+45	9,84E+52	0,00E+00	1,05E-04	3,38E+00	3,76E-04	1,46E-05	3,48E+19	2,50E+00
	süre(sn)	2,84	19,59	14,79	18,86	5,90	11,10	4,53	15,48	32,45	5,11
	ort	3,38E-04	7,30E+02	1,75E+98	1,00E+10	6,78E+105	2,81E+01	1,70E+00	1,32E-02	2,24E+03	9,57E+01
F5	std	2,73E-04	2,47E+02	1,20E+99	0,00E+00	5,20E+106	3,48E+00	4,79E-01	8,83E-03	2,40E+02	3,04E+00
	süre(sn)	1,24	17,89	13,07	15,70	4,21	9,50	2,91	11,47	30,64	3,54
	ort	2,83E-05	1,39E+03	9,48E+03	2,48E-02	3,03E-02	9,98E+02	6,56E-01	1,76E-05	1,65E+03	4,82E+03
F6	std	2,19E-05	1,29E+03	7,93E+02	5,62E-03	8,44E-03	1,64E+02	3,77E-01	3,27E-05	2,91E+02	2,12E+02
	süre(sn)	1,26	17,63	12,96	15,15	4,27	9,51	2,92	11,55	30,67	3,55

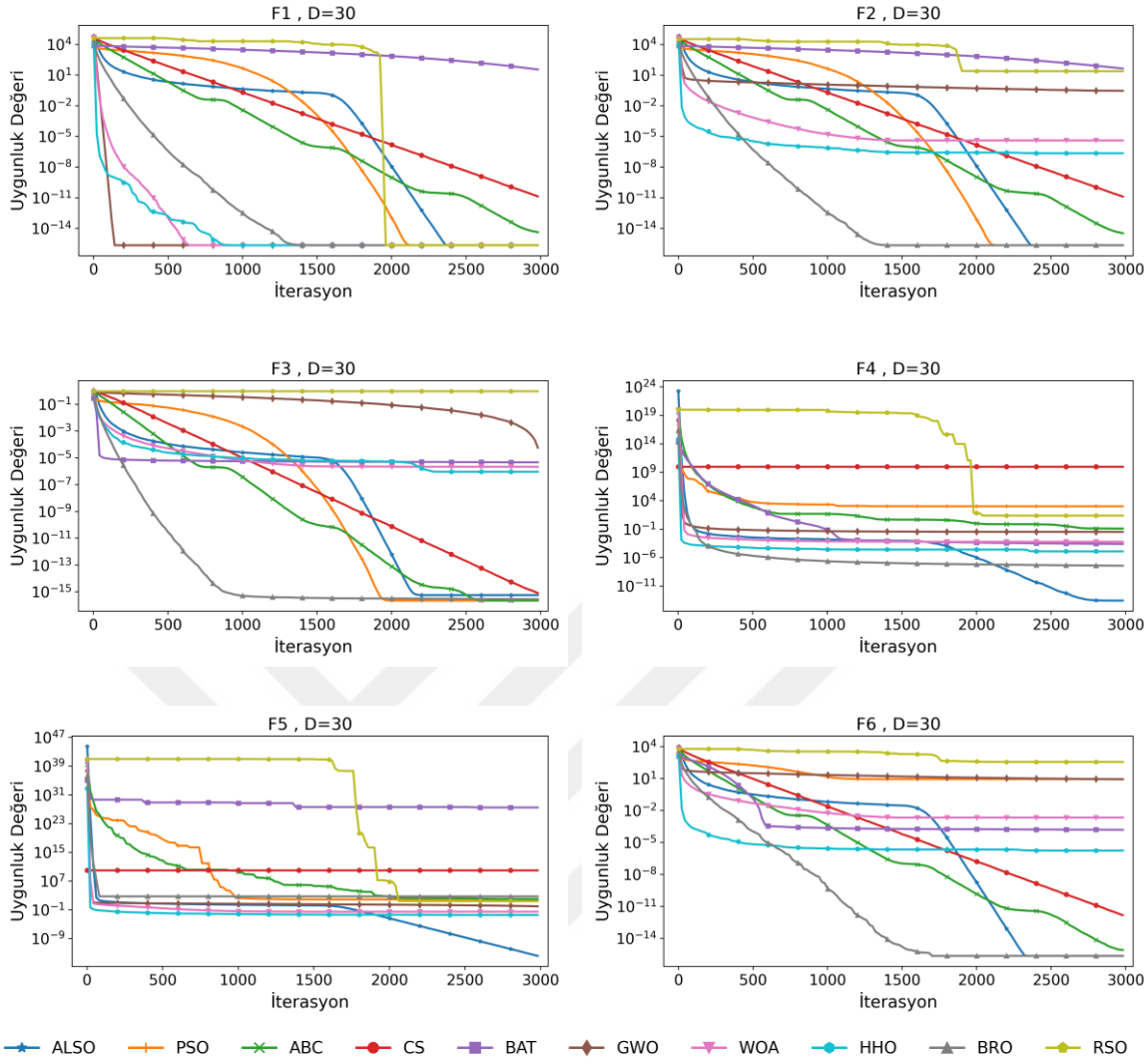
30, 50 ve 100 boyut için tek tepeli test fonksiyonlarından elde edilen en iyi uygunluk değerleri ve çalışma zamanı değerleri Şekil 11'de verilmiştir. Grafiklerde değişiklikleri daha iyi göstermek için 10^4'ten büyük olan değerler 10^4 olarak alınmıştır.



Şekil 11. Tek tepeli test fonksiyonları için elde edilen en iyi uygunluk değerleri ve çalışma zamanı değerleri

Şekil 11, önerilen ALSO algoritmasının tüm boyutlar için, 6 tek tepeli test fonksiyonunda, en iyi uygunluk değeri açısından en iyi sonuçları sağladığını göstermektedir. Sonuçlarda da görüldüğü gibi, ALSO global çözüme en kısa sürede ulaşan algoritmadır.

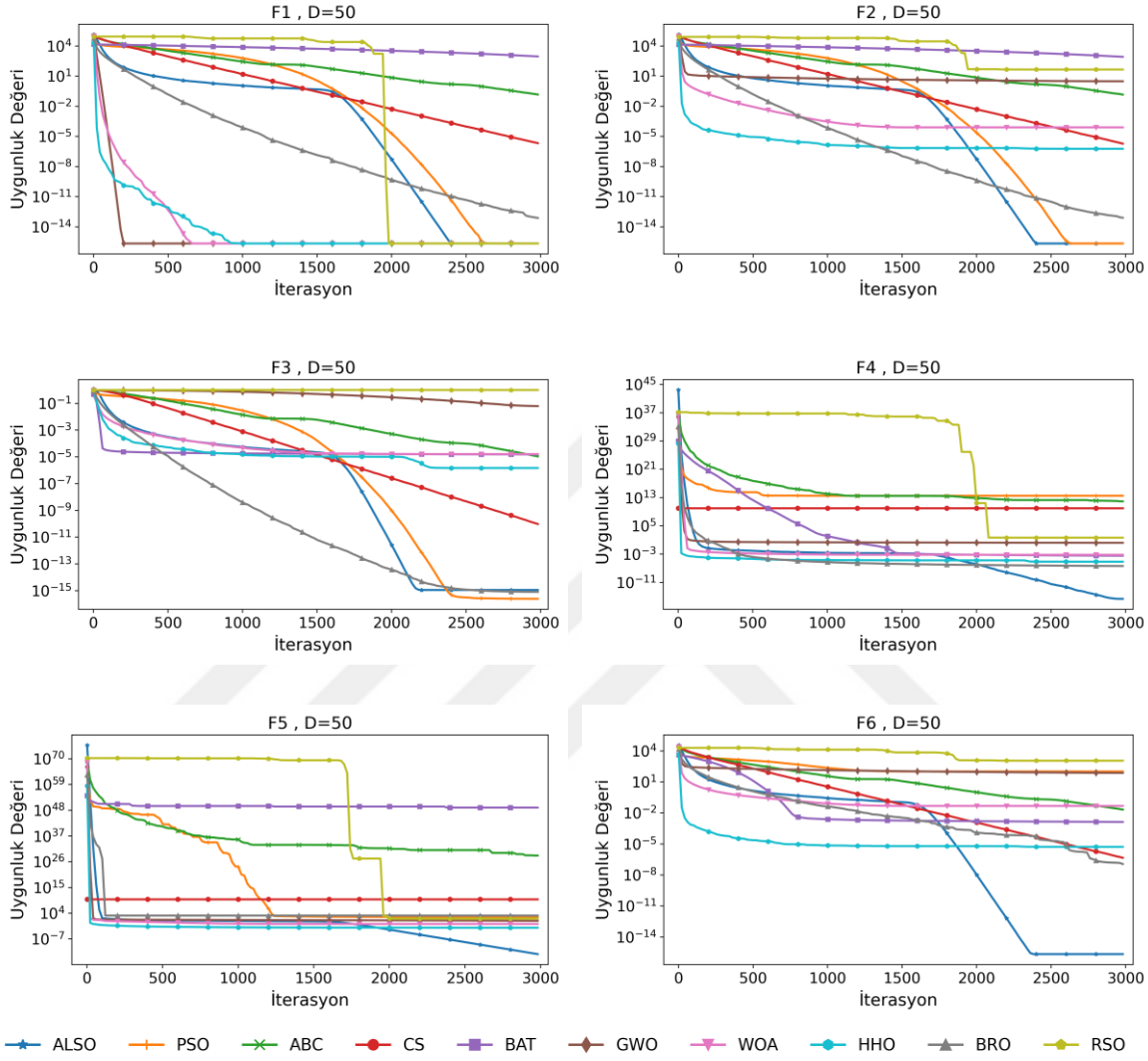
Şekil 12, 30 boyut için tek tepeli test fonksiyonlarında 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer algoritmaların yakınsama eğrilerini göstermektedir.



Şekil 12. 30 boyut için tek tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

Şekil 12'deki yakınsama eğrileri incelendiğinde, ALSO algoritmasının, karşılaştırılan diğer meta-sezgisel algoritmalarla yarışabilecek düzeyde olduğu söylenebilir. Bununla birlikte, ALSO algoritmasının yakınsama eğrisinin parçalı bir fonksiyon olduğu ve duyarlılık fonksiyonuna uygun çalıştığı açıkça görülmektedir. $\lambda = 0.5$ olduğundan birinci kısım maksimum iterasyon sayısının yarısını ifade etmektedir ve λ'ya göre bu durum değiştirilebilir. Ayrıca, birinci kısım keşif işlevini gerçekleştirirken, ikinci kısım sömürü işlevini yürütmektedir. Boyut 30 olduğunda, ALSO algoritması, F4 ve F5 fonksiyonlarında hızlı bir yakınsama göstermektedir. Bununla birlikte, yakınsama eğrilerinin ikinci yarısında, ALSO algoritmasının yakınsama hızının arttığı görülmüştür. Bu nedenle, ALSO algoritmasının keşif ve sömürü arasında etkili bir denge sağladığı söylenebilir.

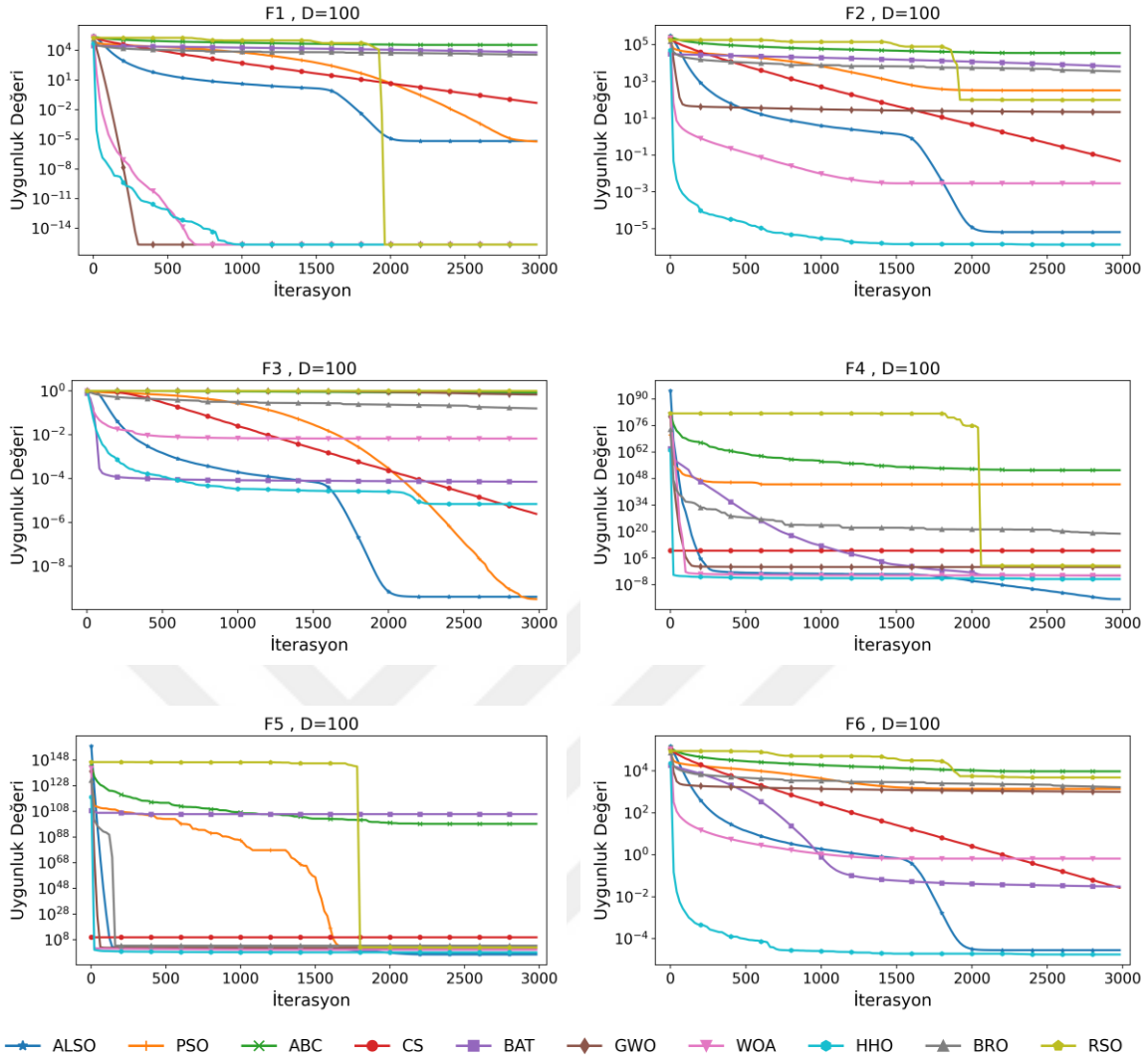
50 boyut için tek tepeli test fonksiyonlarında 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer algoritmaların yakınsama eğrileri Şekil 13'te verilmiştir.



Şekil 13. 50 boyut için tek tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

Şekil 13'e göre, ALSO algoritması 30 boyutta olduğu gibi 50 boyutta da F4 ve F5 fonksiyonlarında hızlı bir yakınsama sağlamıştır. Bununla birlikte, yakınsama eğrileri duyarlılık fonksiyonunun karakteristiğini göstermiştir. Ayrıca, ALSO algoritması, global çözüme yaklaşmak için keşif ve sömürü arasında uygun bir denge kurmuştur.

Şekil 14'te 100 boyut için 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer meta-sezgisel algoritmalar için yakınsama eğrileri gösterilmektedir.



Şekil 14. 100 boyut için tek tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

100 boyut için, Şekil 14'te verilen yakınsama eğrileri incelendiğinde, ALSO algoritması keşif ve sömürü arasında uygun bir denge sağlayarak global çözüme yaklaşmaya çalışmaktadır. Önerilen algoritmanın duyarlılık fonksiyonuna bağlı olarak parçalı bir yakınsama eğrisine sahip olduğu açıkça görülmektedir. Bununla birlikte, F4 ve F5 fonksiyonlarında ALSO algoritması hızlı bir yakınsama göstermiştir.

Tablo 7'de, 30 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçlar verilmektedir. Bu sonuçlara göre, F7 fonksiyonunda, HHO en iyi sonucu ortaya koymuştur. Ayrıca, ALSO algoritması da kayda değer bir sonuç üretmiştir. F8 fonksiyonu için, en iyi değerleri ALSO, GWO, WOA, HHO ve RSO algoritmaları sağlamıştır. HHO, F9 fonksiyonu için en iyi sonucu elde etmiştir. Bununla birlikte, F10 fonksiyonunda, GWO, HHO ve RSO en iyi sonucu üretmiştir. ALSO ve HHO, F11 fonksiyonunda en iyi performansı göstermiştir.

ALSO, PSO, CS, BAT, WOA ve HHO, F12 fonksiyonunda diğer algoritmalara göre daha iyi sonuçlar elde etmiştir. Ayrıca, ALSO çoğu problemde yakın doğrulukla global optimuma yaklaşmıştır. Çalışma zamanına göre ise, ALSO algoritmasının diğer algoritmalardan daha hızlı çalıştığı görülmektedir.

Tablo 8, 50 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçları göstermektedir. Tablo 8'e göre, ALSO, GWO, WOA, HHO ve RSO, F7 ve F8 fonksiyonlarında diğer algoritmalara göre daha iyi sonuçlar üretmiştir. F9 fonksiyonunda, HHO en iyi değeri vermiştir. F10 fonksiyonu için HHO ve RSO karşılaştırılan diğer algoritmalara göre daha iyi bir performans sergilemiştir. Bununla birlikte, ALSO, F11 fonksiyonunda en iyi sonucu üretmiştir. F12 fonksiyonu için en iyi değeri PSO, CS, BAT, WOA ve HHO algoritmaları sağlamıştır. Bunların yanı sıra, çoğu durumda ALSO algoritması, diğer algoritmalara kıyasla daha iyi bir standart sapmaya sahiptir. Ayrıca, yapılan denemeler, ALSO algoritmasının daha hızlı çalıştığını göstermektedir.

100 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçlar Tablo 9'da verilmektedir. Elde edilen sonuçlara göre, F7, F8 ve F10 fonksiyonlarında en iyi değerleri GWO, WOA, HHO ve RSO üretmiştir. HHO, F9 fonksiyonunda en iyi sonucu elde etmiştir. F11 fonksiyonunda, ALSO ve HHO en iyi performansı sergilemiştir. PSO, CS, BAT, WOA ve HHO, F12 fonksiyonunda en iyi sonucu sağlamıştır. 100 boyutta 6 çok tepeli test fonksiyonu üzerinde gerçekleştirilen uygulamalardan elde edilen standart sapmalara göre, diğer algoritmalarla karşılaştırıldığında, ALSO algoritmasının tutarlı çalıştığı gözlenmiştir. Ayrıca, çalışma zamanına göre, ALSO algoritmasının diğer algoritmalardan daha hızlı çalıştığı görülmektedir.

Tablo 7. 30 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	5,23E-14	1,04E-14	1,21E-07	5,39E-01	1,25E+01	6,69E-15	2,47E-15	2,22E-16	1,16E-13	2,16E-15
F7	std	1,23E-14	3,57E-15	1,06E-07	3,50E-01	9,11E-01	1,14E-15	2,30E-15	2,47E-32	6,40E-14	1,64E-15
	süre(sn)	1,26	16,50	12,25	8,95	3,53	4,03	2,08	7,39	13,67	2,30
	ort	2,22E-16	2,19E+01	1,91E+02	5,94E+01	6,65E+01	2,22E-16	2,22E-16	2,22E-16	3,74E+01	2,22E-16
F8	std	2,47E-32	7,14E+00	1,00E+01	4,04E+00	3,02E+01	2,47E-32	2,47E-32	2,47E-32	8,80E+00	2,47E-32
	süre(sn)	1,24	16,56	12,58	9,00	3,52	4,03	2,06	7,37	13,69	2,29
	ort	6,13E+00	3,84E+01	2,48E+01	1,39E+01	2,97E+00	2,58E+01	2,36E+01	4,62E-06	2,87E+01	2,83E+01
F9	std	7,83E+00	6,39E+01	1,30E-01	2,84E+00	6,25E+00	7,28E-01	2,45E-01	7,74E-06	1,35E+01	3,60E-01
	süre(sn)	1,18	16,28	11,95	8,50	3,39	3,97	1,99	7,11	13,46	2,18
	ort	6,06E-03	1,07E-02	1,68E-05	1,19E-06	4,56E+01	2,22E-16	5,07E-04	2,22E-16	1,76E-02	2,22E-16
F10	std	7,67E-03	1,27E-02	1,28E-04	1,23E-06	1,39E+01	2,47E-32	2,35E-03	2,47E-32	1,98E-02	2,47E-32
	süre(sn)	1,34	16,58	12,28	8,93	3,81	4,09	2,14	7,48	13,82	2,40
	ort	3,82E-04	2,29E+03	6,82E+03	2,86E+03	6,84E+03	5,95E+03	2,32E+02	5,75E-04	5,66E+03	6,51E+03
F11	std	4,04E-12	4,20E+02	3,07E+02	1,59E+02	5,42E+02	6,61E+02	5,59E+02	3,39E-04	6,23E+02	8,21E+02
	süre(sn)	2,56	17,25	13,49	10,98	4,38	4,93	2,83	9,19	14,47	3,10
	ort	2,21E-14	2,22E-16	5,71E+00	2,22E-16	2,22E-16	1,21E+01	2,22E-16	2,22E-16	1,30E+01	1,09E+00
F12	std	1,47E-14	2,47E-32	1,13E+00	2,47E-32	2,47E-32	5,61E+00	2,47E-32	2,47E-32	3,76E+00	2,40E+00
	süre(sn)	27,22	41,98	38,77	60,86	29,93	30,01	27,62	74,14	39,85	27,94

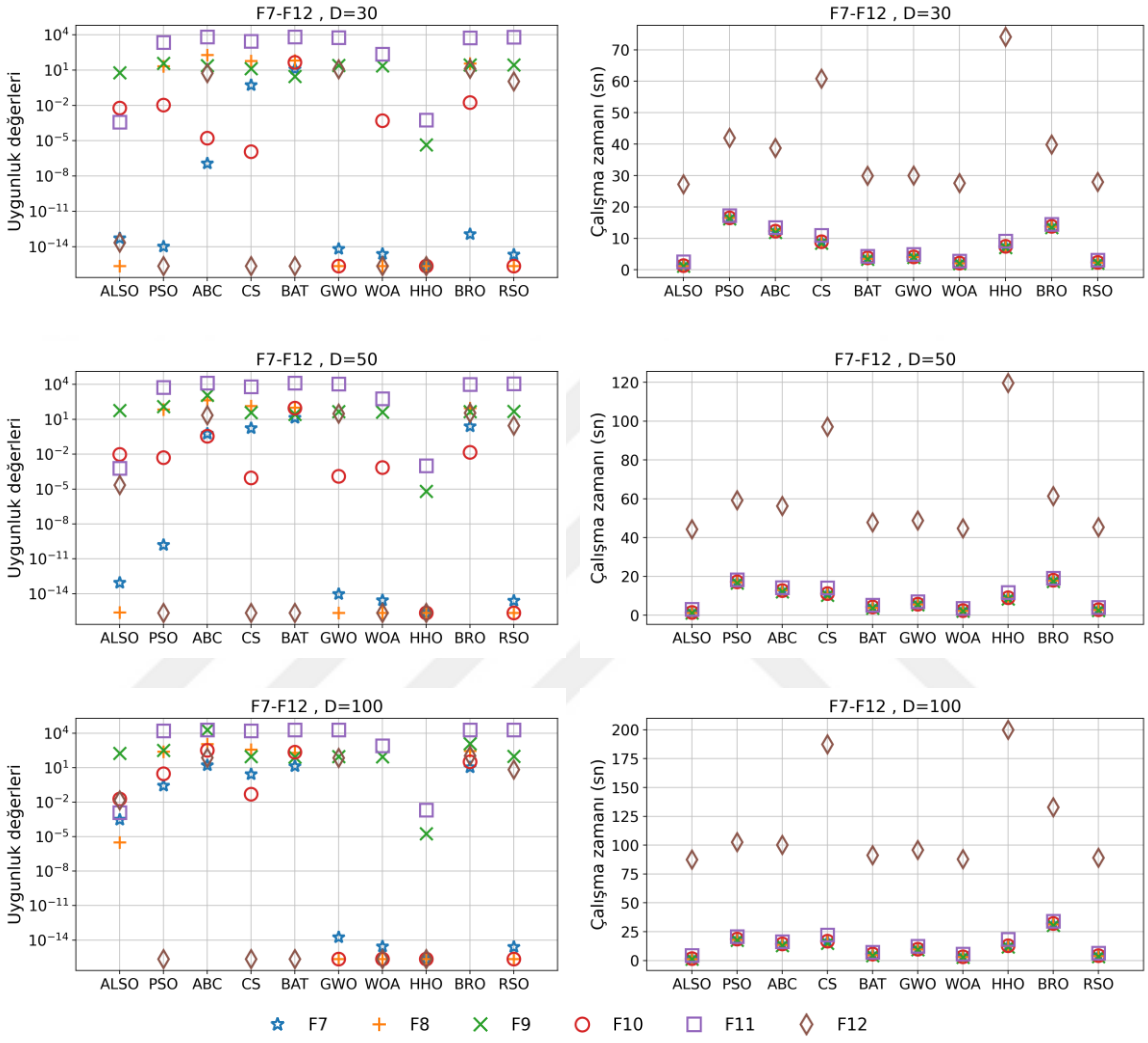
Tablo 8. 50 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	8,73E-14	1,59E-10	5,82E-01	1,75E+00	1,34E+01	9,53E-15	2,75E-15	2,22E-16	2,50E+00	2,55E-15
F7	std	1,50E-14	7,54E-10	3,11E-01	2,24E-01	1,06E+00	2,94E-15	2,21E-15	2,47E-32	8,96E-01	1,53E-15
	süre(sn)	1,30	17,06	12,51	11,06	3,89	5,61	2,33	8,84	17,74	2,74
	ort	2,48E-16	7,05E+01	4,36E+02	1,37E+02	1,04E+02	2,22E-16	2,22E-16	2,22E-16	9,18E+01	2,22E-16
F8	std	1,99E-16	1,87E+01	1,56E+01	1,26E+01	3,29E+01	2,47E-32	2,47E-32	2,47E-32	1,79E+01	2,47E-32
	süre(sn)	1,28	17,05	13,00	11,22	3,89	5,62	2,31	8,81	17,83	2,73
	ort	5,77E+01	1,25E+02	1,19E+03	3,97E+01	2,71E+01	4,58E+01	4,32E+01	6,59E-06	4,76E+01	4,85E+01
F9	std	2,86E+01	3,33E+02	3,03E+02	2,25E+00	7,00E+00	7,80E-01	5,63E+00	9,95E-06	5,72E+00	3,68E-01
	süre(sn)	1,19	16,66	12,20	10,38	3,69	5,55	2,22	8,46	17,46	2,58
	ort	9,73E-03	5,05E-03	3,50E-01	9,05E-05	9,28E+01	1,25E-04	7,11E-04	2,22E-16	1,48E-02	2,22E-16
F10	std	9,00E-03	5,52E-03	9,51E-02	4,41E-05	2,60E+01	9,62E-04	3,57E-03	2,47E-32	1,90E-02	2,47E-32
	süre(sn)	1,44	17,20	12,70	11,15	4,37	5,72	2,44	9,03	18,06	2,92
	ort	6,36E-04	5,70E+03	1,35E+04	6,34E+03	1,32E+04	1,14E+04	5,96E+02	1,03E-03	1,04E+04	1,17E+04
F11	std	9,75E-12	6,17E+02	4,12E+02	2,64E+02	8,87E+02	1,02E+03	1,42E+03	9,19E-04	8,75E+02	1,41E+03
	süre(sn)	3,14	18,27	14,27	14,17	5,27	7,09	3,59	11,85	19,18	4,05
	ort	2,25E-05	2,22E-16	2,26E+01	2,22E-16	2,22E-16	3,09E+01	2,22E-16	2,22E-16	3,63E+01	2,85E+00
F12	std	1,35E-04	2,47E-32	1,65E+00	2,47E-32	2,47E-32	1,35E+01	2,47E-32	2,47E-32	5,53E+00	5,84E+00
	süre(sn)	44,38	59,33	56,30	97,10	47,88	48,84	44,81	119,71	61,46	45,40

Tablo 9. 100 boyut için çok tepeli test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	3,06E-04	2,72E-01	1,66E+01	2,74E+00	1,37E+01	1,81E-14	2,86E-15	2,22E-16	1,08E+01	2,55E-15
F7	std	1,56E-04	1,33E+00	2,46E-01	1,44E-01	7,14E-01	3,53E-15	2,34E-15	2,47E-32	7,74E-01	1,53E-15
	süre(sn)	1,45	18,13	13,73	16,36	4,69	9,60	3,10	12,48	31,31	3,86
	ort	3,13E-06	2,51E+02	1,14E+03	3,86E+02	1,96E+02	2,22E-16	2,22E-16	2,22E-16	2,84E+02	2,22E-16
F8	std	3,50E-06	5,01E+01	3,90E+01	2,73E+01	5,78E+01	2,47E-32	2,47E-32	2,47E-32	3,36E+01	2,47E-32
	süre(sn)	1,41	18,14	13,75	16,70	4,68	9,64	3,09	12,49	31,30	3,86
	ort	1,78E+02	3,32E+02	1,60E+05	9,56E+01	8,83E+01	9,62E+01	9,12E+01	1,84E-05	1,23E+03	9,89E+01
F9	std	4,37E+01	6,18E+02	2,02E+04	5,76E-01	8,92E+00	7,80E-01	1,69E+01	2,88E-05	2,93E+02	2,45E-01
	süre(sn)	1,26	17,57	13,07	15,14	4,29	9,53	2,93	11,87	30,68	3,57
	ort	1,88E-02	3,03E+00	3,29E+02	4,97E-02	2,28E+02	2,22E-16	2,22E-16	2,22E-16	3,36E+01	2,22E-16
F10	std	1,76E-02	1,63E+01	2,49E+01	1,97E-02	4,87E+01	2,47E-32	2,47E-32	2,47E-32	7,36E+00	2,47E-32
	süre(sn)	1,73	18,65	14,44	16,93	5,67	9,84	3,33	12,96	32,14	4,33
	ort	1,30E-03	1,68E+04	3,09E+04	1,67E+04	3,06E+04	2,48E+04	8,34E+02	2,06E-03	2,36E+04	2,34E+04
F11	std	2,44E-05	1,37E+03	1,20E+03	4,01E+02	2,00E+03	1,68E+03	1,67E+03	1,50E-03	1,52E+03	2,14E+03
	süre(sn)	4,57	20,80	16,36	22,16	7,31	12,51	5,61	18,49	34,10	6,46
	ort	1,45E-02	2,22E-16	7,57E+01	2,22E-16	2,22E-16	7,15E+01	2,22E-16	2,22E-16	1,11E+02	6,84E+00
F12	std	9,80E-03	2,47E-32	2,79E+00	2,47E-32	2,47E-32	4,63E+01	2,47E-32	2,47E-32	6,87E+00	1,53E+01
	süre(sn)	87,42	102,70	100,16	187,52	91,23	95,77	87,90	233,46	132,91	89,20

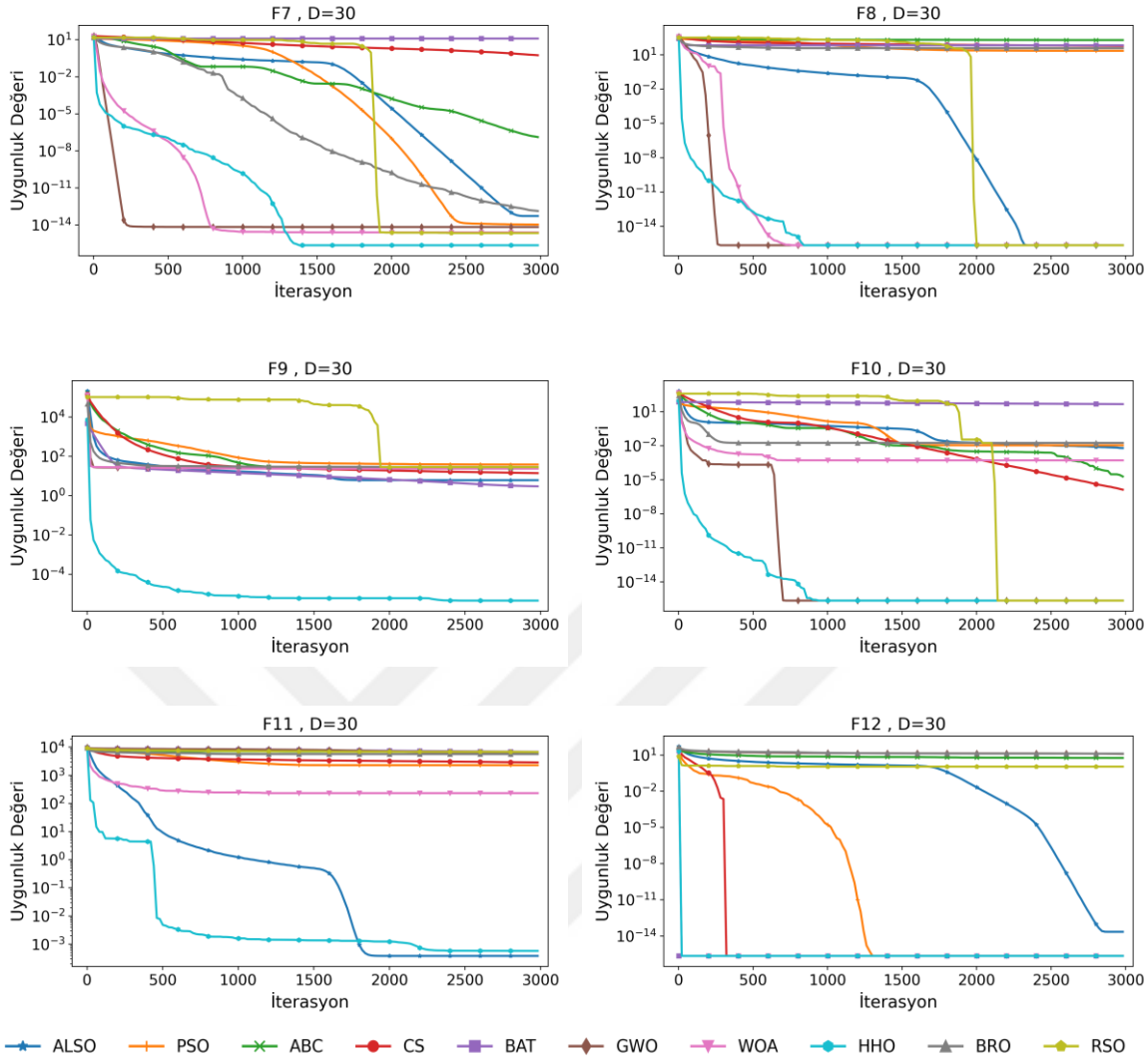
30, 50 ve 100 boyut için çok tepeli test fonksiyonlarından elde edilen en iyi uygunluk değerleri ve çalışma zamanı değerleri Şekil 15'te verilmiştir. Grafiklerde değişiklikleri daha iyi göstermek için 10^4'ten büyük olan değerler 10^4 olarak alınmıştır.



Şekil 15. Çok tepeli test fonksiyonları için elde edilen en iyi uygunluk değerleri ve çalışma zamanı değerleri

Şekil 15, önerilen ALSO algoritmasının tüm boyutlarda, 6 çok tepeli test fonksiyonu için, en iyi uygunluk değeri açısından en iyi sonuçları ürettiğini ortaya koymaktadır. Ayrıca, ortalama çalışma zamanı bakımından, ALSO algoritmasının karşılaştırılan diğer algoritmalara göre daha hızlı olduğu gözlenmiştir.

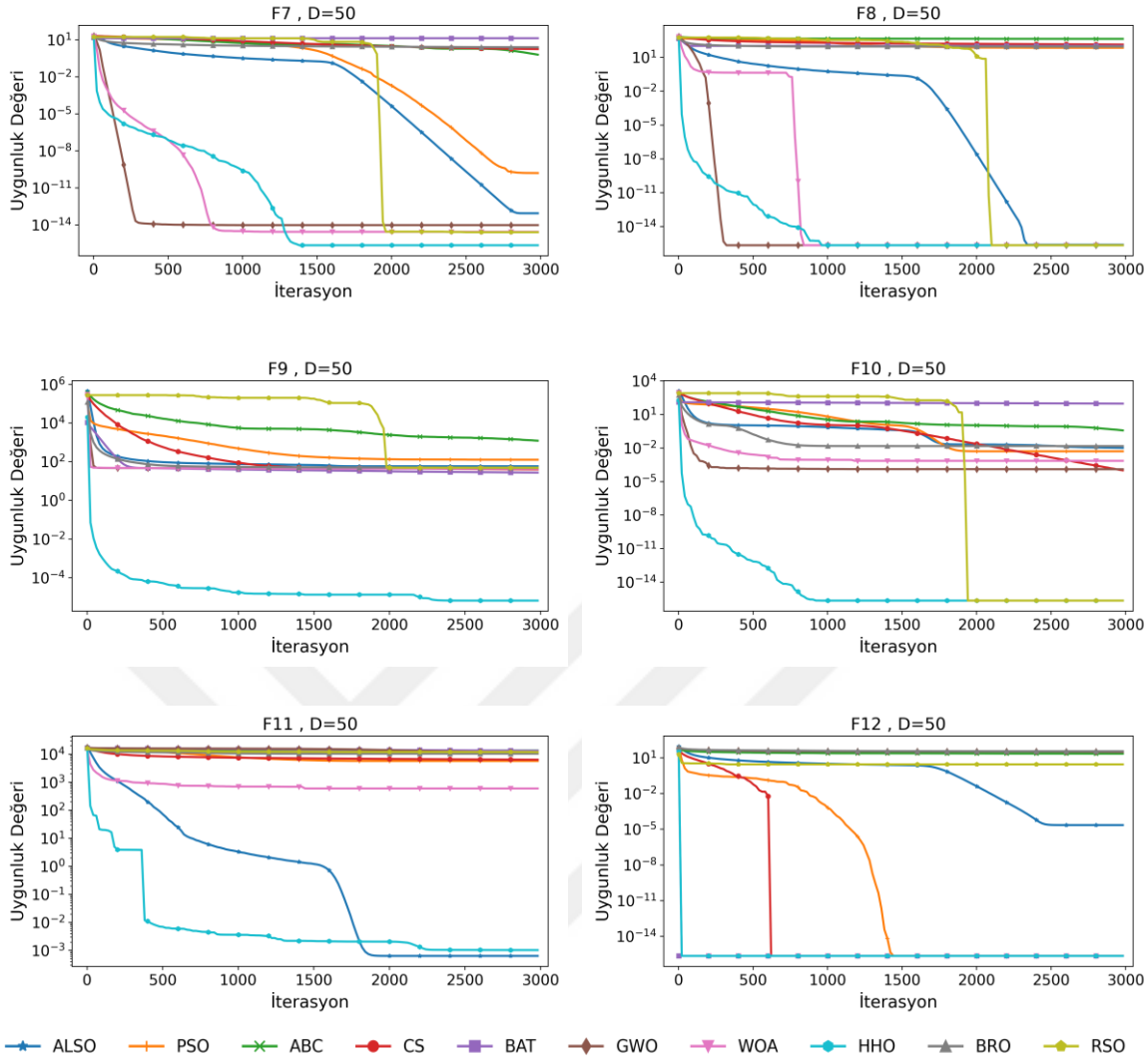
30 boyutta çok tepeli test fonksiyonları için, 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer algoritmaların yakınsama eğrileri Şekil 16'da verilmektedir.



Şekil 16. 30 boyut için çok tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

Şekil 16'ya göre, ALSO algoritmasının yakınsama eğrilerinden, iterasyon sayısının yarısına kadar algoritmanın arama uzayını keşfettiği ve böylece yerel optimumdan kaçındığı söylenebilir. Daha sonra, algoritmanın yakınsama hızı artmıştır. Böylece, önerilen algoritma keşif ve sömürü arasında düzgün bir denge sağlayarak optimal sonuca ulaşmaktadır.

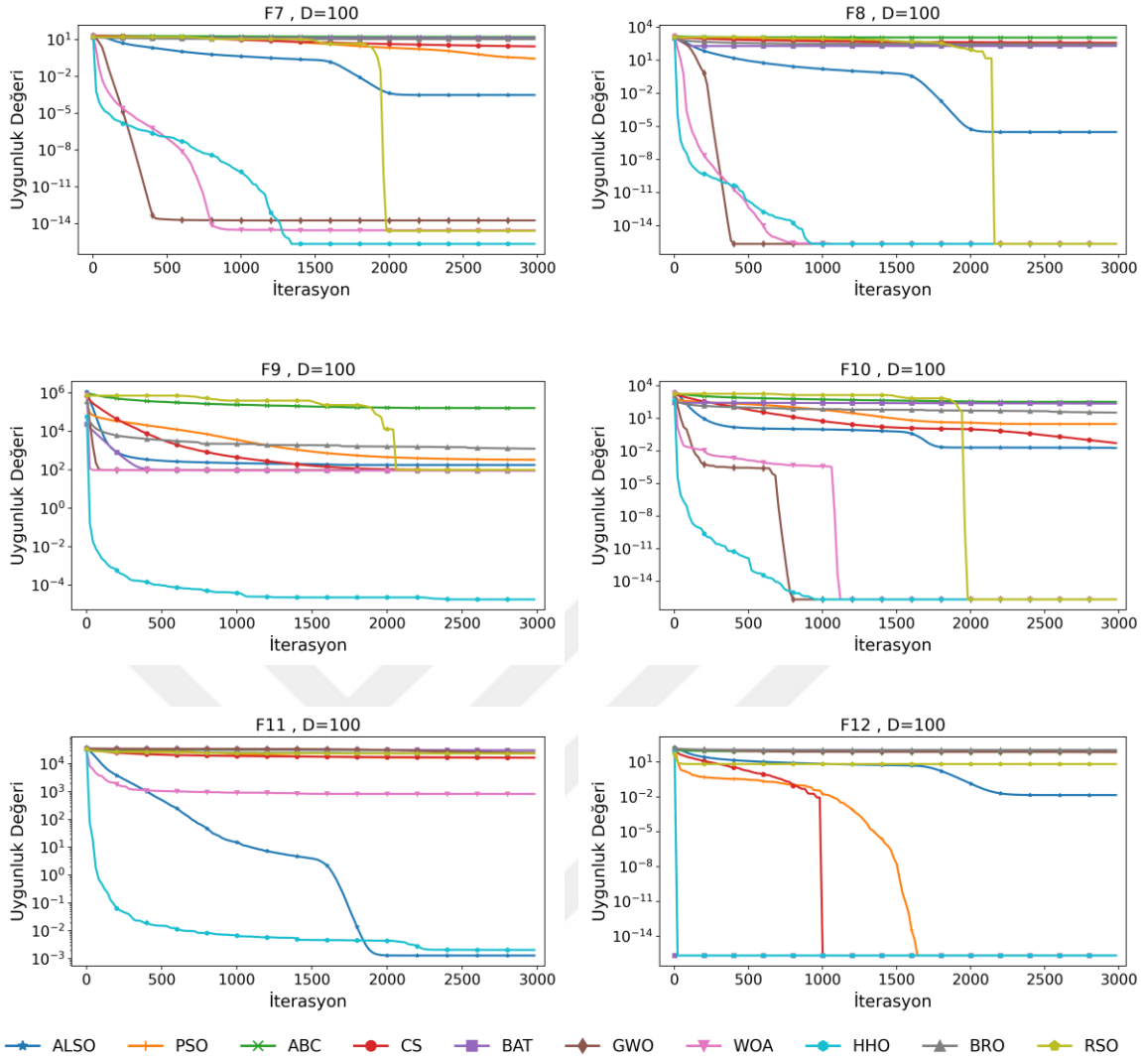
Şekil 17, 50 boyut için çok tepeli test fonksiyonlarında, 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer algoritmaların yakınsama eğrilerini göstermektedir.



Şekil 17. 50 boyut için çok tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

Şekil 17'deki yakınsama eğrileri incelendiğinde, önerilen ALSO algoritmasının F9 ve F10 fonksiyonlarında, başlangıçta hızlı bir yakınsama yaptığı ve daha sonra optimuma yakınsadığı gözlenmiştir. Ayrıca, elde edilen sonuçlar, ALSO algoritmasının, global çözüme ulaşmak için keşif ve sömürü arasındaki dengeyi koruduğunu ortaya koymaktadır.

Şekil 18'de 100 boyut için 3000 iterasyon sonucunda, ALSO ve karşılaştırılan diğer meta-sezgisel algoritmalar için yakınsama eğrileri gösterilmektedir.



Şekil 18. 100 boyut için çok tepeli test fonksiyonlarından elde edilen yakınsama eğrileri

Şekil 18'de verilen 100 boyuttaki çok tepeli test fonksiyonlarından elde edilen yakınsama eğrileri, ALSO algoritmasının arama uzayındaki iyi bölgeleri keşfettiğini ve en iyi bölgede arama yaptığını ortaya koymaktadır. Böylece, önerilen algoritma keşif ve sömürü arasında uygun bir denge sağlamış olur.

Tablo 10, 30 boyut için birleşik test fonksiyonlarından elde edilen sonuçları içermektedir. Tablo 10'dan görüldüğü üzere, ALSO algoritması, diğer algoritmalarla kıyaslandığında F13, F14, F16, F17 ve F19 fonksiyonlarında rekabetçi bir performans göstermiştir. Bununla birlikte, F15, F18, F20, F21 ve F22 fonksiyonlarında da başarılı sonuçlar elde etmiştir. Ayrıca, standart sapma değerlerine dayanarak, ALSO algoritmasının karşılaştırılan diğer algoritmalara göre daha doğru ve tutarlı çalıştığı söylenebilir. Çalışma

zamanına göre ise, elde edilen sonuçlar, ALSO algoritmasının global çözüme daha kısa sürede ulaştığını göstermektedir.

Tablo 11'de 50 boyutta birleşik test fonksiyonlarından elde edilen sonuçlar verilmektedir. Bu sonuçlar incelendiğinde, ALSO algoritmasının birçok fonksiyonda rekabetçi sonuçlar elde ettiği ve F14, F18, F21 ve F22 fonksiyonlarında diğer algoritmalara göre daha iyi bir performans sergilediği görülmektedir. Bununla birlikte, diğer algoritmalarla karşılaştırıldığında, ALSO algoritmasının çoğu durumda daha iyi bir standart sapmaya sahip olduğu gözlenmiştir. Ayrıca, çalışma zamanı açısından, önerilen ALSO algoritması, diğer algoritmalarla kıyaslandığında, genel olarak daha hızlı sonuçlar elde etmiştir.

100 boyut için birleşik test fonksiyonlarından elde edilen sonuçlar, Tablo 12'de verilmektedir. Bu sonuçlar, ALSO algoritmasının F13 ve F18 fonksiyonları için rekabetçi sonuçlar elde edebildiğini ve F14, F15, F16, F17, F19, F20, F21 ve F22 fonksiyonlarında da karşılaştırılan diğer algoritmalardan daha iyi performans gösterdiğini belirtmektedir. Buna ek olarak, karşılaştırılan diğer algoritmalara göre, çoğu durumda ALSO algoritması daha iyi bir standart sapmaya sahiptir ve çalışma zamanı açısından daha hızlı bir performans ortaya koymuştur.

Tablo 10. 30 boyut için birleşik test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	2,38E+03	2,37E+03	2,49E+03	2,40E+03	2,54E+03	2,37E+03	2,57E+03	2,51E+03	2,47E+03	2,63E+03
F13	std	4,57E+01	1,61E+01	1,24E+01	6,04E+01	4,46E+01	1,93E+01	5,97E+01	3,98E+01	3,29E+01	2,39E+01
	süre(sn)	4,77	19,93	16,23	15,85	7,00	7,39	5,39	15,11	17,08	5,64
	ort	3,67E+03	3,71E+03	3,33E+03	2,51E+03	6,99E+03	4,10E+03	6,42E+03	5,53E+03	3,82E+03	7,61E+03
F14	std	1,39E+03	1,46E+03	5,95E+02	7,37E+02	1,48E+03	1,62E+03	2,17E+03	2,15E+03	2,08E+03	1,40E+03
	süre(sn)	5,88	20,85	17,56	18,23	8,13	8,47	6,45	17,86	18,07	6,74
	ort	2,72E+03	2,79E+03	2,85E+03	2,77E+03	3,18E+03	2,73E+03	3,05E+03	3,02E+03	3,04E+03	3,13E+03
F15	std	2,38E+01	3,49E+01	9,48E+00	2,42E+01	1,20E+02	3,05E+01	1,05E+02	8,94E+01	7,05E+01	6,66E+01
	süre(sn)	6,55	21,51	18,23	19,55	8,80	9,13	7,12	19,10	18,78	7,40
	ort	3,01E+03	2,98E+03	3,01E+03	2,88E+03	3,38E+03	2,90E+03	3,18E+03	3,28E+03	3,18E+03	3,36E+03
F16	std	1,63E+02	5,10E+01	1,02E+01	1,16E+02	1,65E+02	4,46E+01	9,72E+01	1,44E+02	7,71E+01	6,60E+01
	süre(sn)	6,35	21,28	17,86	18,88	8,52	8,85	6,85	18,19	18,54	7,11
	ort	2,89E+03	2,91E+03	2,89E+03	2,88E+03	3,26E+03	2,94E+03	2,94E+03	2,91E+03	2,93E+03	4,18E+03
F17	std	1,76E+00	3,89E+01	7,21E-02	4,26E-02	1,58E+02	2,57E+01	2,57E+01	1,82E+01	2,19E+01	3,86E+02
	süre(sn)	6,12	21,23	17,45	18,25	8,44	8,84	6,84	18,45	18,56	7,09
	ort	3,02E+03	4,65E+03	5,59E+03	3,13E+03	8,26E+03	4,39E+03	7,63E+03	6,65E+03	6,23E+03	7,94E+03
F18	std	5,25E+02	4,97E+02	9,15E+01	1,73E+02	1,06E+03	3,27E+02	9,05E+02	1,46E+03	1,74E+03	4,74E+02
	süre(sn)	7,91	23,01	19,71	22,46	10,19	10,59	8,56	22,65	20,19	8,84
	ort	3,23E+03	3,25E+03	3,21E+03	3,22E+03	3,52E+03	3,23E+03	3,34E+03	3,30E+03	3,53E+03	3,78E+03
F19	std	8,88E+00	3,16E+01	3,56E+00	5,22E+00	1,64E+02	1,35E+01	5,52E+01	4,05E+01	9,70E+01	2,18E+02
	süre(sn)	9,14	23,81	20,78	24,49	11,24	11,58	9,53	24,46	21,18	9,84
	ort	3,20E+03	3,31E+03	3,22E+03	3,20E+03	4,16E+03	3,33E+03	3,28E+03	3,24E+03	3,26E+03	5,22E+03
F20	std	7,83E+00	7,98E+01	9,67E+00	1,01E+01	4,26E+02	4,08E+01	2,49E+01	2,90E+01	2,55E+01	4,07E+02
	süre(sn)	7,63	22,81	19,12	21,54	9,96	10,31	8,32	22,14	20,10	8,56
	ort	3,56E+03	3,57E+03	4,13E+03	3,74E+03	5,61E+03	3,64E+03	4,78E+03	4,25E+03	4,19E+03	4,71E+03
F21	std	1,18E+02	1,17E+02	1,16E+02	6,97E+01	6,49E+02	1,35E+02	3,41E+02	3,08E+02	2,60E+02	2,95E+02
	süre(sn)	7,33	22,40	19,18	21,01	9,57	9,88	7,89	21,30	19,60	8,17
	ort	8,41E+03	2,41E+05	4,18E+04	9,99E+03	2,07E+07	4,59E+06	8,04E+06	7,13E+05	6,07E+04	2,91E+08
F22	std	1,18E+03	7,74E+05	2,69E+04	1,30E+03	2,88E+07	3,46E+06	6,65E+06	3,91E+05	3,08E+04	2,56E+08
	süre(sn)	10,28	25,25	22,02	26,59	12,41	12,70	10,73	28,21	22,50	10,96

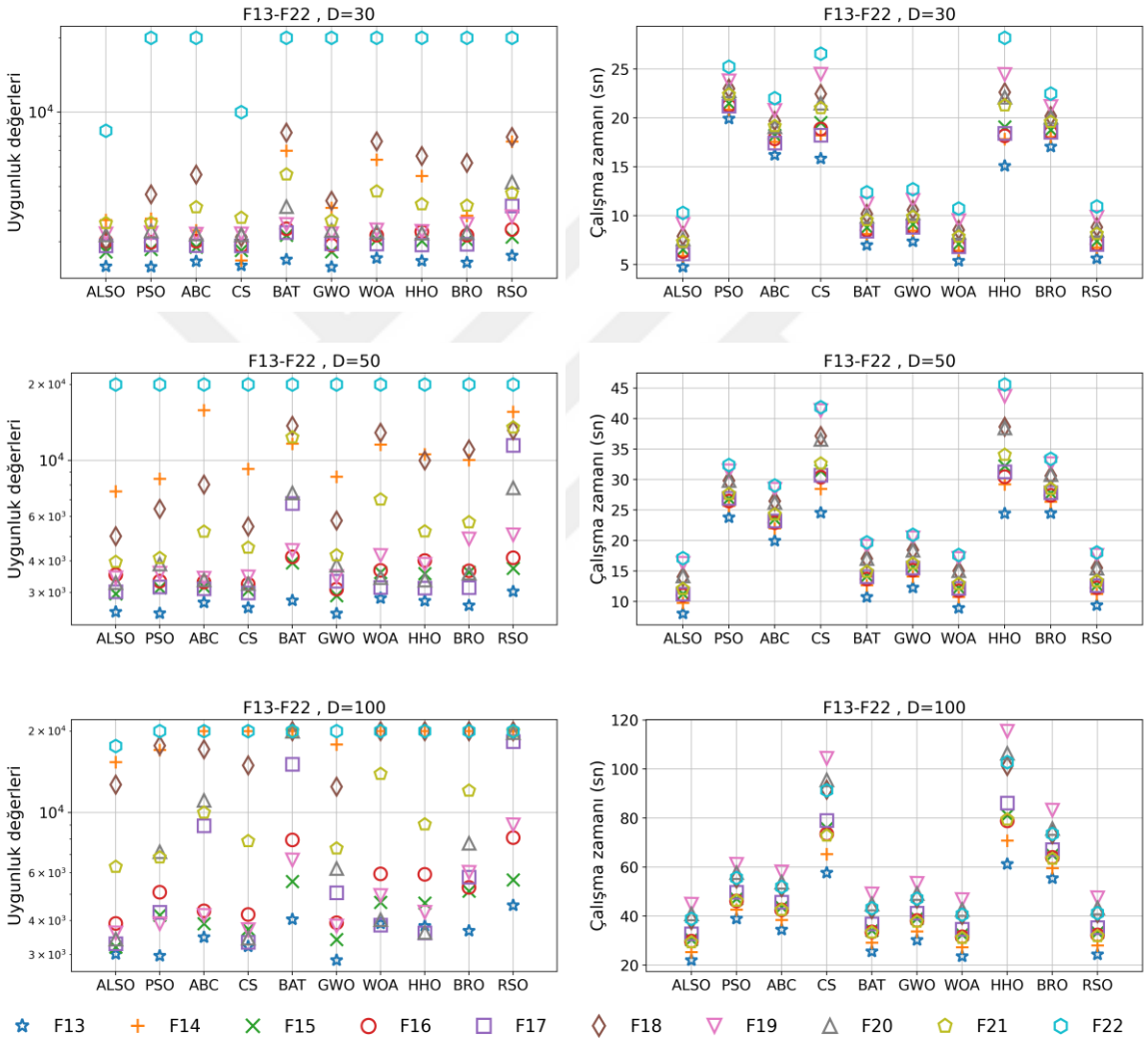
Tablo 11. 50 boyut için birleşik test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	2,51E+03	2,48E+03	2,74E+03	2,61E+03	2,79E+03	2,48E+03	2,85E+03	2,78E+03	2,66E+03	3,03E+03
F13	std	5,79E+01	4,50E+01	1,35E+01	3,01E+01	8,23E+01	3,87E+01	8,22E+01	7,61E+01	5,91E+01	4,62E+01
	süre(sn)	8,03	23,80	19,99	24,55	10,77	12,30	8,94	24,45	24,48	9,39
	ort	7,54E+03	8,46E+03	1,58E+04	9,25E+03	1,16E+04	8,63E+03	1,16E+04	1,06E+04	1,00E+04	1,56E+04
F14	std	7,83E+02	1,28E+03	5,91E+02	1,26E+03	1,48E+03	1,95E+03	1,24E+03	1,12E+03	1,12E+03	8,68E+02
	süre(sn)	9,79	25,68	21,99	28,49	12,68	14,10	10,75	29,21	26,38	11,25
	ort	2,96E+03	3,13E+03	3,16E+03	3,06E+03	3,93E+03	2,91E+03	3,61E+03	3,57E+03	3,61E+03	3,74E+03
F15	std	4,18E+01	9,28E+01	1,46E+01	3,34E+01	2,33E+02	5,26E+01	1,61E+02	1,23E+02	1,80E+02	1,03E+02
	süre(sn)	11,27	26,89	23,55	31,44	14,23	15,64	12,26	32,24	27,81	12,72
	ort	3,53E+03	3,33E+03	3,30E+03	3,24E+03	4,16E+03	3,09E+03	3,67E+03	4,02E+03	3,66E+03	4,12E+03
F16	std	1,88E+02	1,08E+02	1,33E+01	3,13E+01	2,24E+02	8,96E+01	1,55E+02	1,63E+02	1,43E+02	1,65E+02
	süre(sn)	11,04	26,48	22,94	30,38	13,71	15,15	11,78	30,48	27,40	12,26
	ort	3,02E+03	3,16E+03	3,10E+03	2,99E+03	6,76E+03	3,32E+03	3,15E+03	3,13E+03	3,14E+03	1,15E+04
F17	std	2,61E+01	1,22E+02	1,90E+01	1,58E+01	1,13E+03	1,58E+02	3,92E+01	3,38E+01	2,14E+01	9,97E+02
	süre(sn)	11,24	26,94	23,25	30,75	14,10	15,55	12,20	31,31	27,88	12,63
	ort	5,01E+03	6,43E+03	8,02E+03	5,48E+03	1,37E+04	5,79E+03	1,29E+04	9,99E+03	1,11E+04	1,32E+04
F18	std	1,51E+03	8,78E+02	1,25E+02	1,19E+03	1,51E+03	4,34E+02	1,49E+03	2,42E+03	1,13E+03	8,33E+02
	süre(sn)	14,08	29,85	26,46	37,16	17,05	18,48	15,05	38,66	30,61	15,52
	ort	3,43E+03	3,58E+03	3,42E+03	3,46E+03	4,40E+03	3,47E+03	4,20E+03	3,89E+03	4,89E+03	5,08E+03
F19	std	5,64E+01	1,38E+02	3,31E+01	3,93E+01	3,04E+02	6,68E+01	3,63E+02	1,83E+02	3,15E+02	4,78E+02
	süre(sn)	16,27	31,46	28,44	41,34	19,07	20,46	17,06	43,68	32,60	17,60
	ort	3,29E+03	3,89E+03	3,37E+03	3,26E+03	7,44E+03	3,85E+03	3,44E+03	3,36E+03	3,58E+03	7,80E+03
F20	std	1,86E+01	6,55E+02	1,51E+01	1,54E+00	1,14E+03	2,78E+02	6,97E+01	3,42E+01	3,91E+01	6,30E+02
	süre(sn)	14,22	29,86	26,29	36,62	17,06	18,41	15,09	38,50	30,86	15,49
	ort	3,97E+03	4,11E+03	5,23E+03	4,51E+03	1,24E+04	4,22E+03	7,01E+03	5,24E+03	5,69E+03	1,35E+04
F21	std	2,13E+02	3,42E+02	1,62E+02	1,28E+02	4,14E+03	2,37E+02	8,05E+02	5,21E+02	4,06E+02	2,46E+03
	süre(sn)	12,07	27,63	24,31	32,65	14,90	16,25	12,90	34,10	28,49	13,36
	ort	8,16E+05	4,83E+06	6,79E+06	4,05E+06	2,58E+08	7,09E+07	8,25E+07	1,28E+07	2,19E+07	2,33E+09
F22	std	7,64E+04	4,78E+06	4,71E+06	9,74E+05	1,37E+08	2,46E+07	2,83E+07	3,34E+06	2,21E+06	1,04E+09
	süre(sn)	17,13	32,40	29,04	41,87	19,71	20,96	17,63	45,60	33,39	18,05

Tablo 12. 100 boyut için birleşik test fonksiyonlarından elde edilen sonuçlar

		ALSO	PSO	ABC	CS	BAT	GWO	WOA	HHO	BRO	RSO
	ort	3,01E+03	2,96E+03	3,47E+03	3,21E+03	4,04E+03	2,86E+03	3,90E+03	3,82E+03	3,67E+03	4,55E+03
F13	std	6,53E+01	7,90E+01	3,25E+01	6,72E+01	2,35E+02	1,14E+02	1,69E+02	1,55E+02	1,26E+02	1,22E+02
	süre(sn)	22,03	39,01	34,54	57,69	25,59	30,26	23,60	61,25	55,50	24,43
	ort	1,54E+04	1,71E+04	3,36E+04	2,24E+04	2,31E+04	1,79E+04	2,52E+04	2,30E+04	2,17E+04	3,26E+04
F14	std	9,06E+02	1,34E+03	5,56E+02	4,46E+02	3,33E+03	2,80E+03	2,17E+03	1,58E+03	1,50E+03	1,56E+03
	süre(sn)	25,32	42,65	38,45	65,33	29,17	33,79	27,27	70,82	59,64	28,02
	ort	3,19E+03	4,19E+03	3,90E+03	3,72E+03	5,58E+03	3,40E+03	4,67E+03	4,64E+03	5,14E+03	5,65E+03
F15	std	4,89E+01	2,15E+02	2,48E+01	5,44E+01	3,36E+02	8,79E+01	2,14E+02	2,10E+02	2,73E+02	3,11E+02
	süre(sn)	30,44	47,15	43,71	75,88	34,26	39,14	32,56	81,23	65,06	33,36
	ort	3,90E+03	5,09E+03	4,35E+03	4,21E+03	7,93E+03	3,93E+03	5,95E+03	5,92E+03	5,30E+03	8,09E+03
F16	std	6,91E+01	3,01E+02	2,63E+01	5,96E+01	8,45E+02	9,96E+01	4,40E+02	3,73E+02	2,54E+02	4,82E+02
	süre(sn)	29,85	46,19	42,59	73,57	33,57	38,22	31,58	78,91	64,00	32,39
	ort	3,29E+03	4,29E+03	8,96E+03	3,32E+03	1,51E+04	5,07E+03	3,85E+03	3,60E+03	5,79E+03	1,83E+04
F17	std	5,06E+01	7,37E+02	5,51E+02	3,20E+01	2,74E+03	3,43E+02	1,12E+02	6,41E+01	4,00E+02	1,25E+03
	süre(sn)	32,81	49,80	45,70	79,05	36,88	41,26	34,65	86,22	67,20	35,32
	ort	1,27E+04	1,77E+04	1,72E+04	1,49E+04	3,96E+04	1,25E+04	3,23E+04	2,49E+04	3,04E+04	3,71E+04
F18	std	2,42E+03	2,84E+03	2,45E+02	6,29E+02	4,85E+03	8,82E+02	3,56E+03	1,64E+03	1,92E+03	3,66E+03
	süre(sn)	38,47	55,36	51,82	91,64	42,84	47,16	40,51	101,23	73,46	41,28
	ort	3,60E+03	3,89E+03	4,18E+03	3,70E+03	6,70E+03	3,82E+03	4,95E+03	4,29E+03	6,02E+03	9,03E+03
F19	std	6,43E+01	2,31E+02	7,53E+01	5,92E+01	9,76E+02	9,51E+01	5,27E+02	3,06E+02	7,86E+02	7,31E+02
	süre(sn)	44,87	61,21	58,11	104,47	49,07	53,41	46,76	115,51	83,31	47,53
	ort	3,42E+03	7,18E+03	1,11E+04	3,45E+03	2,15E+04	6,23E+03	4,04E+03	3,62E+03	7,72E+03	1,98E+04
F20	std	2,90E+01	2,62E+03	6,32E+02	4,35E+01	2,81E+03	8,69E+02	1,28E+02	4,61E+01	6,37E+02	2,22E+03
	süre(sn)	40,98	57,83	53,86	95,67	45,01	49,30	42,72	106,47	75,81	43,36
	ort	6,32E+03	6,85E+03	1,00E+04	7,85E+03	2,32E+04	7,38E+03	1,39E+04	9,06E+03	1,21E+04	7,63E+04
F21	std	4,63E+02	6,25E+02	2,53E+02	1,84E+02	5,77E+03	4,65E+02	1,56E+03	5,72E+02	9,70E+02	3,96E+04
	süre(sn)	29,39	46,27	42,55	72,90	33,40	37,78	31,15	79,42	63,48	31,89
	ort	1,76E+04	9,07E+08	1,84E+06	1,00E+10	7,42E+09	3,03E+08	2,86E+08	2,97E+07	3,00E+08	3,01E+10
F22	std	6,65E+03	9,56E+08	6,61E+05	0,00E+00	3,52E+09	2,52E+08	1,32E+08	8,59E+06	8,65E+07	2,94E+09
	süre(sn)	38,87	55,61	51,89	91,33	43,18	47,16	40,60	102,87	73,51	41,29

30, 50 ve 100 boyut için birleşik test fonksiyonlarından elde edilen en iyi uygunluk ve çalışma zamanı değerleri Şekil 19'da verilmiştir. Grafiklerde değişiklikleri daha iyi göstermek için 2×10^4'ten büyük olan değerler 2×10^4 olarak alınmıştır. Şekil 19 incelendiğinde, birleşik test fonksiyonları için en iyi uygunluk değeri açısından, ALSO algoritmasının tatmin edici sonuçlar elde ettiği görülmektedir. Ayrıca, çalışma zamanı bakımından, ALSO algoritması diğer algoritmalara göre daha hızlı çalışmaktadır.



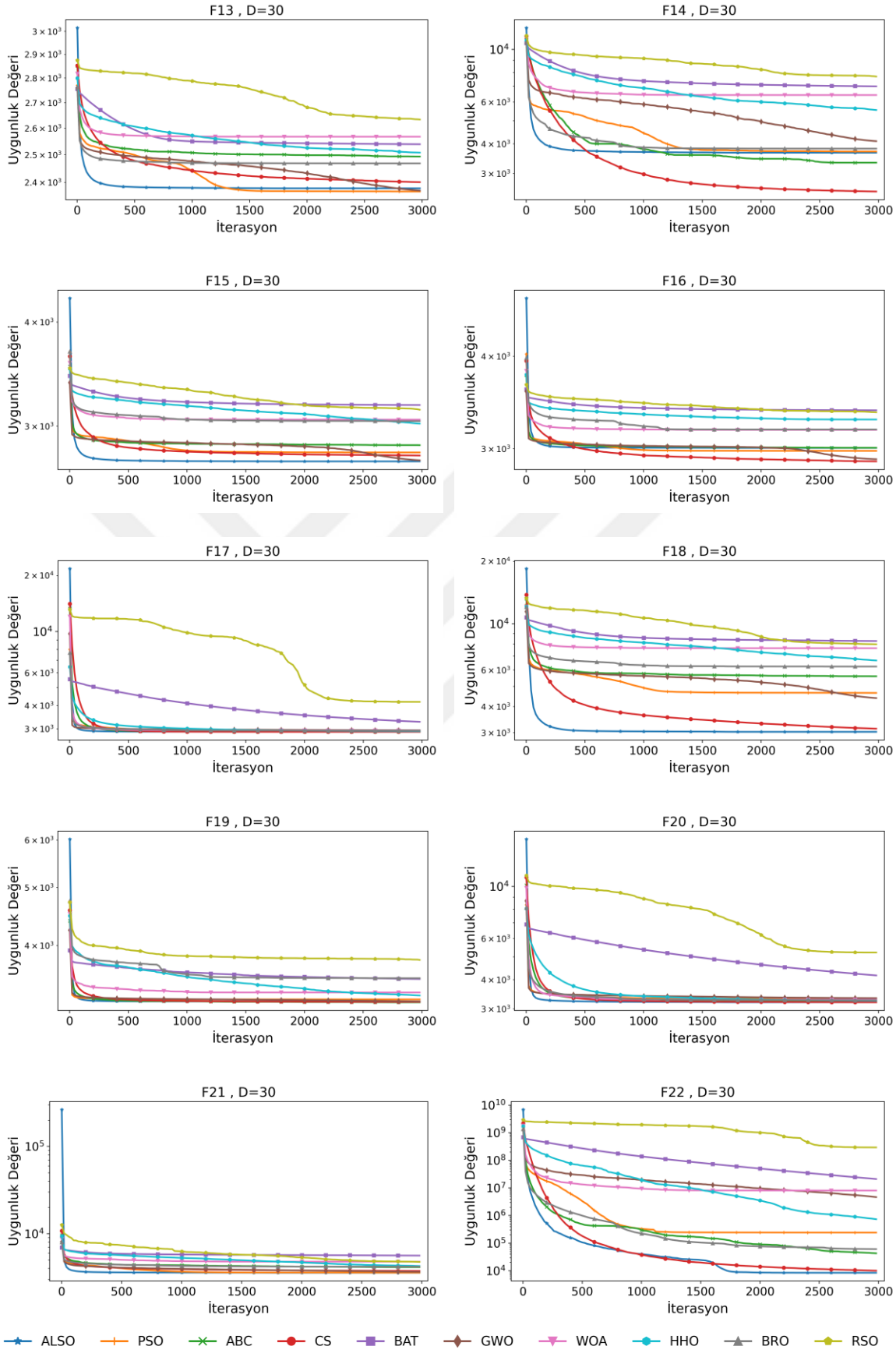
Şekil 19. Birleşik test fonksiyonları için elde edilen en iyi uygunluk değerleri ve çalışma zamanı değerleri

Şekil 20'de 30 boyut için birleşik test fonksiyonlarında 3000 iterasyon sonucunda ALSO ve karşılaştırılan diğer algoritmaların yakınsama eğrileri verilmektedir. Buna göre, ALSO algoritması, 30 boyut için birleşik test fonksiyonlarında hızlı bir yakınsama eğilimi

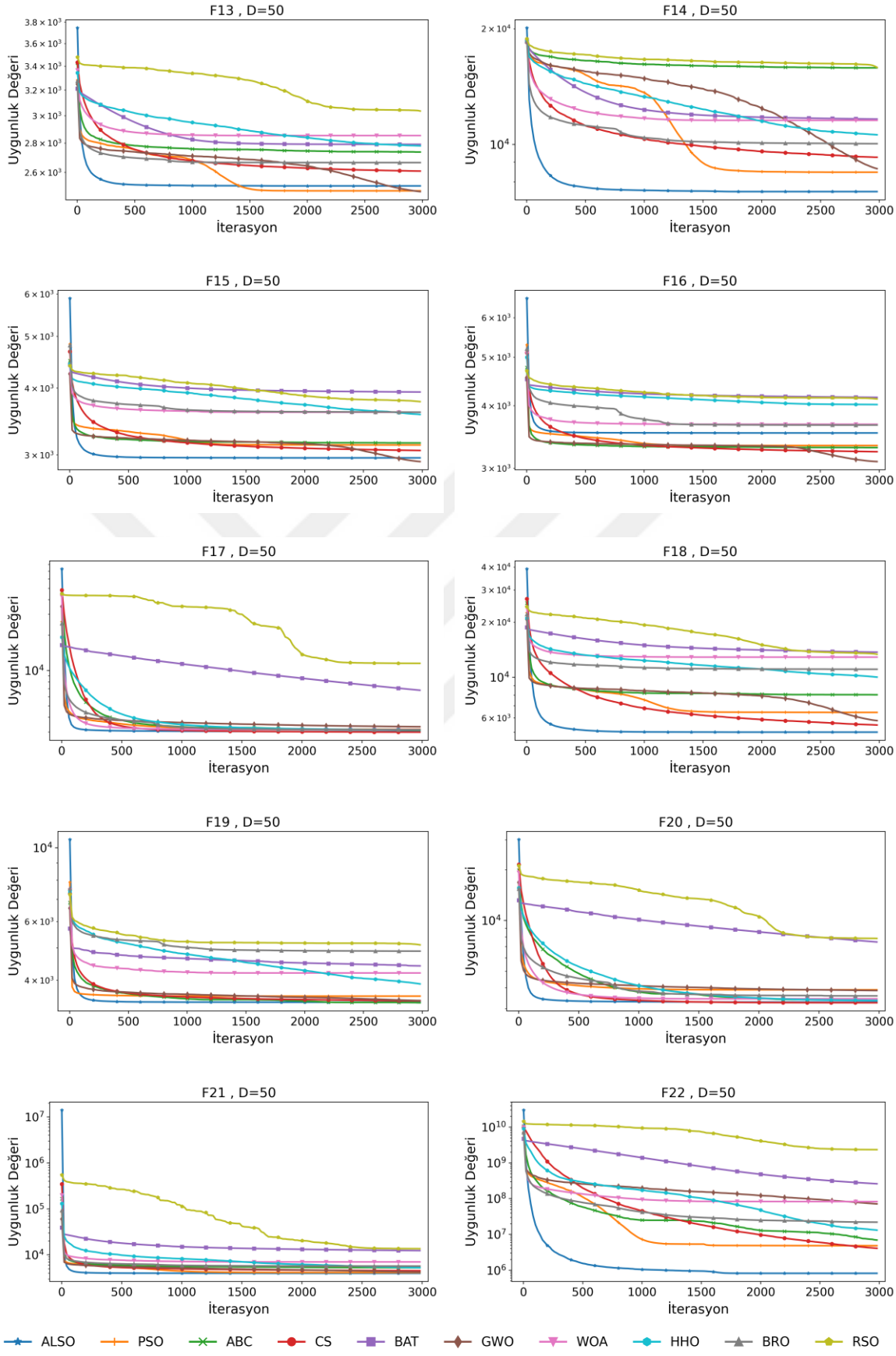
göstermiştir. Ayrıca, önerilen ALSO algoritmasının bu tür zor problemlerin çözümünde keşif ve sömürü arasında uygun bir denge kurduğu söylenebilir.

Şekil 21, 50 boyut için 3000 iterasyon sonucunda, ALSO ve karşılaştırılan diğer meta-sezgisel algoritmalardan elde edilen yakınsama eğrilerini göstermektedir. Şekil 21 incelendiğinde, ALSO algoritmasının rekabetçi sonuçlar ürettiği görülmektedir. Bununla birlikte, ALSO algoritmasının yakınsama hızı yüksektir. Dolayısıyla, bu durum algoritmanın yerel optimumdan kaçınmasını sağlar.

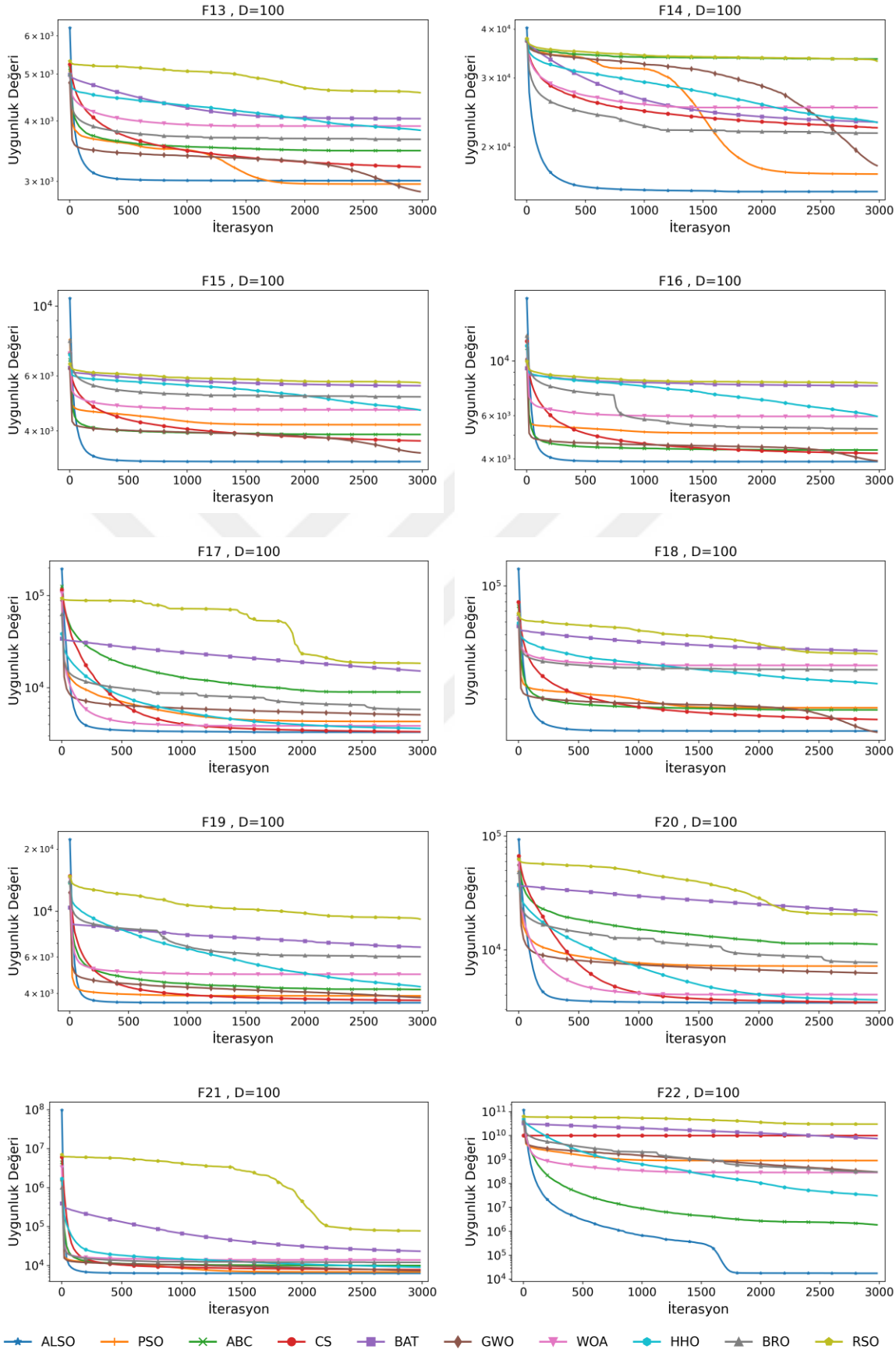
100 boyutta 3000 iterasyon sonucunda, ALSO ve karşılaştırılan diğer meta-sezgisel algoritmaların yakınsama eğrileri Şekil 22'de verilmiştir. Elde edilen yakınsama eğrileri, ALSO algoritmasının yakınsama hızının yüksek olduğunu göstermekle birlikte, keşif ve sömürü arasında uygun bir denge sağladığını da ortaya koymaktadır.



Şekil 20. 30 boyut için birleşik test fonksiyonlarından elde edilen yakınsama eğrileri



Şekil 21. 50 boyut için birleşik test fonksiyonlarından elde edilen yakınsama eğrileri



Şekil 22. 100 boyut için birleşik test fonksiyonlarından elde edilen yakınsama eğrileri

3.3. Simülasyon Sonuçlarının İstatistiksel Analizi

Optimizasyon algoritmalarının performansını karşılaştırmak ve ölçmek için hesaplama zamanı, ortalama ve standart sapma yeterli değildir. Bu nedenle, geliştirilen her yeni optimizasyon algoritmasının performans değerlendirmesini yapmak amacıyla istatistiksel testler kullanılmaktadır. Bununla birlikte, belirli bir optimizasyon probleminin çözümünde, önerilen yeni bir meta-sezgisel algoritmanın mevcut diğer meta-sezgisel algoritmalara kıyasla daha iyi olduğunu ortaya koymak için istatistiksel bir testin yapılması gerekir (Derrac vd., 2011). Böylece, istatistiksel testler yardımıyla elde edilen deneysel sonuçların bilimsel olarak güvenirliliği tahmin edilebilir.

Bu çalışmada, önerilen ALSO algoritması uygulanarak test fonksiyonlarından elde edilen sonuçların, karşılaştırılan diğer algoritmaların sonuçlarından istatistiksel olarak anlamlı bir farklılık gösterip göstermediği incelenmiştir. Bu amaçla, aşağıdaki hipotezler kurulmuştur:

H_0: ALSO ve karşılaştırılan algoritmalardan elde edilen sonuçlar arasındaki fark istatistiksel olarak anlamlı değildir.

H_1: ALSO ve karşılaştırılan algoritmalardan elde edilen sonuçlar arasındaki fark istatistiksel olarak anlamlıdır.

ALSO ve karşılaştırılan algoritmalardan elde edilen sonuçlar, parametrik varsayımları sağlamadığından, parametrik olmayan bir test olan Wilcoxon sıra toplam (Mann-Whitney U) testi, %5 anlamlılık düzeyinde uygulanmıştır. Bununla birlikte, her bir boyut için Wilcoxon sıra toplam testinden elde edilen p değerleri hesaplanmıştır. Önerilen ALSO algoritmasının karşılaştırılan algoritmalardan daha iyi olduğu durum "1+", daha kötü olduğu durum da "1-" ile ifade edilmiştir. Eğer karşılaştırılan iki algoritmanın performansı eşit ise "0" olarak gösterilmiştir. Bu durumda, ALSO algoritmasının performansını "0" ve "1+" durumlarının toplam sayısı ifade etmektedir.

Tablo 13'te 30 boyuttaki test fonksiyonlarından elde edilen sonuçlara Wilcoxon sıra toplam testi uygulanarak, hesaplanan p değerleri verilmiştir. Tablo 13 incelendiğinde, Wilcoxon sıra toplam testine dayanarak, ALSO algoritmasının istatistiksel olarak karşılaştırılan diğer algoritmalardan farklı olduğu ve daha iyi sonuçlar ürettiği söylenebilir. Bununla birlikte, önerilen algoritmanın performansı %86 olarak hesaplanmıştır.

Tablo 14, 50 boyuttaki test fonksiyonlarından elde edilen sonuçlara Wilcoxon sıra toplam testinin uygulanmasıyla elde edilen p değerlerini göstermektedir. Tablo 14'teki p değerleri, ALSO algoritmasının istatistiksel olarak diğer algoritmalardan farklı olduğunu ve daha başarılı sonuçlar sağladığını doğrulamıştır. Bunun yanı sıra, önerilen ALSO algoritmasının 50 boyuttaki test fonksiyonlarındaki performansı %84 olarak hesaplanmıştır.

100 boyuttaki test fonksiyonlarının sonuçlarına uygulanan Wilcoxon sıra toplam testinden elde edilen p değerleri Tablo 15'te yer almaktadır. p değerleri incelendiğinde, ALSO ve karşılaştırılan algoritmalardan elde edilen sonuçlar arasındaki farkın istatistiksel olarak anlamlı ve ALSO algoritmasının daha başarılı olduğu görülmüştür. Ayrıca, 100 boyut için önerilen ALSO algoritmasının performansı %83 olarak hesaplanmıştır.

Tablo 13. $D = 30$ için Wilcoxon sıra toplam testi ile ALSO ve diğer algoritmalar arasındaki istatistiksel karşılaştırma

	PSO	**ABC**	**CS**	**BAT**	**GWO**	**WOA**	**HHO**	**BRO**	**RSO**
	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h
F1	1.00e+00: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.00e+00: 0	1.00e+00: 0	1.00e+00: 0	1.00e+00: 0	1.00e+00: 0
F2	1.00e+00: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.00e+00: 0	3.47e-21: 1+
F3	3.47e-21: 1-	3.47e-21: 1-	3.02e-04: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.31e-19: 1-	3.47e-21: 1+
F4	8.95e-03: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F5	2.08e-01: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F6	4.31e-01: 0	2.18e-09: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.00e+00: 0	3.47e-21: 1+
F7	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	8.29e-17: 1+	3.47e-21: 1-
F8	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.00e+00: 0	1.00e+00: 0	1.00e+00: 0	3.47e-21: 1+	1.00e+00: 0
F9	1.94e-17: 1+	6.69e-20: 1+	1.09e-08: 1+	5.02e-01: 0	4.68e-21: 1+	8.55e-18: 1+	3.47e-21: 1-	4.76e-20: 1+	3.47e-21: 1+
F10	1.97e-01: 0	9.58e-12: 1-	4.59e-12: 1-	3.47e-21: 1+	2.66e-16: 1-	4.81e-14: 1-	2.66e-16: 1-	4.41e-02: 1+	2.66e-16: 1-
F11	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F12	1.54e-20: 1-	3.47e-21: 1+	1.54e-20: 1-	1.54e-20: 1-	3.47e-21: 1+	1.54e-20: 1-	1.54e-20: 1-	3.47e-21: 1+	4.77e-09: 1+
F13	1.27e-08: 1-	3.47e-21: 1+	2.55e-07: 1+	3.47e-21: 1+	2.34e-07: 1-	3.47e-21: 1+	5.72e-21: 1+	5.00e-20: 1+	3.47e-21: 1+
F14	4.12e-02: 1+	9.37e-01: 0	6.44e-01: 0	3.93e-16: 1+	1.64e-03: 1+	4.26e-14: 1+	4.82e-11: 1+	5.63e-03: 1+	4.68e-21: 1+
F15	1.03e-19: 1+	3.47e-21: 1+	1.54e-14: 1+	3.47e-21: 1+	9.79e-01: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F16	1.24e-08: 1-	8.79e-09: 1+	1.14e-12: 1-	1.47e-20: 1+	2.63e-12: 1-	9.23e-12: 1+	7.65e-19: 1+	1.96e-16: 1+	3.47e-21: 1+
F17	3.47e-21: 1+	1.68e-05: 1+	3.03e-16: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	2.70e-18: 1+	3.47e-21: 1+	3.47e-21: 1+
F18	1.32e-16: 1+	3.47e-21: 1+	1.16e-13: 1+	3.47e-21: 1+	8.54e-16: 1+	3.47e-21: 1+	1.08e-19: 1+	5.20e-17: 1+	3.47e-21: 1+
F19	2.01e-08: 1+	1.93e-10: 1-	8.09e-05: 1-	3.47e-21: 1+	9.50e-01: 0	3.47e-21: 1+	4.46e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F20	3.47e-21: 1+	3.47e-21: 1+	6.67e-01: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	7.37e-20: 1+	9.38e-20: 1+	3.47e-21: 1+
F21	7.33e-01: 0	4.46e-21: 1+	1.02e-14: 1+	3.47e-21: 1+	1.92e-03: 1+	3.47e-21: 1+	2.92e-20: 1+	7.33e-21: 1+	3.47e-21: 1+
F22	9.12e-14: 1+	3.47e-21: 1+	1.63e-09: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+

Tablo 14. $D = 50$ için Wilcoxon sıra toplam testi ile ALSO ve diğer algoritmalar arasındaki istatistiksel karşılaştırma

	PSO	**ABC**	**CS**	**BAT**	**GWO**	**WOA**	**HHO**	**BRO**	**RSO**
	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h
F1	1.00e+00: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.00e+00: 0	1.00e+00: 0	1.00e+00: 0	3.47e-21: 1+	1.00e+00: 0
F2	1.00e+00: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F3	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.06e-14: 1-	3.47e-21: 1+
F4	2.66e-16: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F5	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F6	4.59e-03: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F7	6.76e-14: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-
F8	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	8.75e-01: 0	8.75e-01: 0	8.75e-01: 0	3.47e-21: 1+	8.75e-01: 0
F9	1.29e-01: 0	3.47e-21: 1+	5.32e-04: 1-	1.58e-09: 1-	2.40e-02: 1-	8.42e-03: 1-	3.47e-21: 1-	2.38e-01: 0	5.29e-01: 0
F10	1.35e-03: 1-	3.47e-21: 1+	9.22e-05: 1-	3.47e-21: 1+	2.99e-12: 1-	4.11e-11: 1-	1.38e-12: 1-	2.94e-01: 0	1.38e-12: 1-
F11	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.17e-18: 1+	3.47e-21: 1+	3.47e-21: 1+
F12	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	8.62e-08: 1+
F13	1.88e-06: 1-	3.47e-21: 1+	2.92e-20: 1+	3.47e-21: 1+	5.54e-11: 1-	3.47e-21: 1+	3.47e-21: 1+	1.47e-20: 1+	3.47e-21: 1+
F14	3.01e-10: 1+	3.47e-21: 1+	1.62e-17: 1+	3.83e-21: 1+	1.20e-05: 1+	3.47e-21: 1+	5.44e-21: 1+	1.93e-19: 1+	3.47e-21: 1+
F15	5.00e-19: 1+	3.47e-21: 1+	8.02e-19: 1+	3.47e-21: 1+	1.57e-10: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F16	1.43e-15: 1-	1.17e-18: 1-	1.17e-18: 1-	1.20e-20: 1+	1.28e-18: 1-	2.70e-06: 1+	5.44e-21: 1+	8.64e-05: 1+	3.47e-21: 1+
F17	4.93e-18: 1+	4.46e-21: 1+	1.74e-12: 1-	3.47e-21: 1+	3.65e-21: 1+	6.01e-21: 1+	1.15e-20: 1+	3.65e-21: 1+	3.47e-21: 1+
F18	2.21e-07: 1+	3.47e-21: 1+	1.08e-01: 0	3.47e-21: 1+	5.02e-01: 0	3.47e-21: 1+	5.56e-16: 1+	2.29e-20: 1+	3.47e-21: 1+
F19	9.93e-12: 1+	4.62e-01: 0	1.01e-04: 1+	3.47e-21: 1+	7.53e-04: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F20	1.27e-20: 1+	3.47e-21: 1+	4.10e-18: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	5.24e-20: 1+	3.47e-21: 1+	3.47e-21: 1+
F21	2.57e-02: 1+	3.47e-21: 1+	1.38e-19: 1+	3.47e-21: 1+	1.22e-07: 1+	3.47e-21: 1+	1.97e-20: 1+	3.47e-21: 1+	3.47e-21: 1+
F22	4.46e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+

Tablo 15. $D = 100$ için Wilcoxon sıra toplam testi ile ALSO ve diğer algoritmalar arasındaki istatistiksel karşılaştırma

	PSO	**ABC**	**CS**	**BAT**	**GWO**	**WOA**	**HHO**	**BRO**	**RSO**
	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h	p-değeri: h
F1	1.58e-08: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-
F2	2.28e-07: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	5.42e-14: 1-	3.47e-21: 1+	3.47e-21: 1+
F3	4.38e-13: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	4.24e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F4	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F5	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F6	2.54e-13: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.18e-07: 1-	3.47e-21: 1+	3.47e-21: 1+
F7	8.81e-19: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-
F8	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-
F9	3.06e-01: 0	3.47e-21: 1+	1.77e-17: 1-	2.25e-18: 1-	1.77e-17: 1-	2.14e-18: 1-	3.47e-21: 1-	3.47e-21: 1+	2.91e-17: 1-
F10	1.08e-10: 1+	3.47e-21: 1+	1.74e-14: 1+	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-
F11	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.24e-10: 1+	3.47e-21: 1+	3.47e-21: 1+
F12	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	3.47e-21: 1-	3.47e-21: 1-	3.47e-21: 1+	8.62e-08: 1+
F13	5.86e-04: 1-	3.47e-21: 1+	1.59e-19: 1+	3.47e-21: 1+	3.46e-15: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F14	2.37e-10: 1+	3.47e-21: 1+	3.47e-21: 1+	4.68e-21: 1+	2.81e-15: 1+	3.47e-21: 1+	3.47e-21: 1+	3.65e-21: 1+	3.47e-21: 1+
F15	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	5.44e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F16	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	1.32e-01: 0	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F17	3.47e-21: 1+	3.47e-21: 1+	1.49e-05: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F18	1.77e-17: 1+	3.47e-21: 1+	2.66e-17: 1+	3.47e-21: 1+	2.93e-04: 1-	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F19	1.61e-14: 1+	3.47e-21: 1+	4.11e-12: 1+	3.47e-21: 1+	3.42e-19: 1+	3.47e-21: 1+	3.65e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F20	3.47e-21: 1+	3.47e-21: 1+	4.29e-04: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F21	5.91e-06: 1+	3.47e-21: 1+	1.04e-20: 1+	3.47e-21: 1+	4.97e-17: 1+	3.47e-21: 1+	3.65e-21: 1+	3.47e-21: 1+	3.47e-21: 1+
F22	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+	3.47e-21: 1+

3.4. ALSO Algoritmasının Mühendislik Tasarım Problemlerine Uygulanması

Mühendislik tasarımı, belirlenmiş ihtiyaçları karşılayan ürünler oluşturmak için karar verme süreci olarak tanımlanmaktadır. Mühendislik tasarım problemlerinin çoğu, çok fazla karar değişkenine ve karmaşık amaç fonksiyonuna sahiptir (Yang ve Deb, 2010; Eskandar vd., 2012; Askarzadeh, 2016). Ayrıca, mühendislik tasarım problemleri büyük ölçekli, lineer olmayan ve kısıtlı optimizasyon problemleridir. Uygun çözümler, tasarım parametrelerinin olası tüm değerleri ile karakterize edilen tüm tasarımların kümesidir. Tüm uygun çözümler içindeki en iyi çözümü bulmak için optimizasyon yöntemleri kullanılabilir (Akay ve Karaboga, 2012; Askarzadeh, 2016).

Bu bölümde, gerilim/sıkıştırma yay tasarım problemi, kaynaklı kiriş tasarım problemi ve basınçlı kap tasarım problemi olmak üzere üç tane mühendislik tasarım problemi ele alınmıştır. Bu problemler, arama uzayındaki bazı bölgelerin yasaklı olmasına neden olan birçok eşitlik ve eşitsizlik kısıtı içermektedir. Dolayısıyla, ALSO algoritmasının, kısıtlı optimizasyon problemlerini çözebilmesi için uyarlanması gerekmektedir.

Minimum amaç değerini bulmaya çalışan arama ajanları yasaklı bölgelere düştüklerinde arama algoritmalarının iki seçeneği bulunur. Birincisi, bu bölgeden çıkmak için arama ajanlarının rastgele bir konuma yönlendirilmesidir. Ancak, bu işlem, genelde uygun bölge bulunamadığında fazla miktarda döngüye sebep olur. İkincisi ise yasaklı bölgede bulunan arama ajanlarının ceza fonksiyonu kullanılarak cezalandırılmasıdır (Datta ve Deb, 2014).

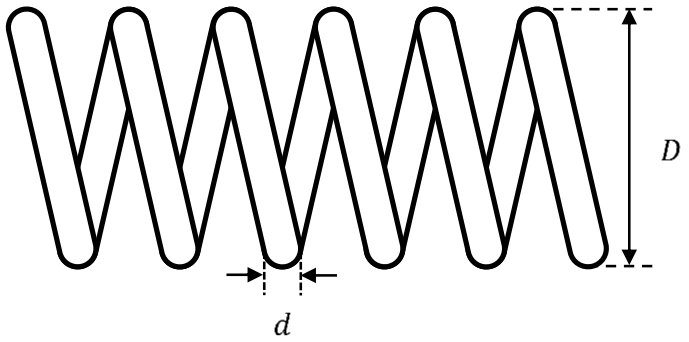
Ceza fonksiyonları, kısıtlı optimizasyon problemlerinin kısıtsız optimizasyon problemlerine dönüştürmek için kullanılan yöntemlerden biridir. Ceza fonksiyonları, ilk olarak 1940'lı yıllarda Courant (1943) tarafından ortaya atılmıştır. Daha sonraları, Carroll (1961), Fiacco ve McCormick (1966) ceza fonksiyonlarını geliştirmişlerdir. Ceza fonksiyonunun amacı, belirli bir çözümde sağlanmayan kısıtlara bağlı olarak elde edilen bir değeri, amaç fonksiyonuna ekleyerek veya amaç fonksiyonundan çıkartarak kısıtlı bir optimizasyon problemini, kısıtsız bir optimizasyon problemine dönüştürmektir (Coello, 2002; Yeniay, 2005; Mezura-Montes ve Coello, 2011).

Bu çalışmada, ALSO algoritmasının kısıtlı optimizasyon problemlerine uygulanabilmesi için ceza fonksiyonu yaklaşımı kullanılmıştır. Her ceza fonksiyonu bir sabite eşitlenerek, arama ajanları her yasaklı bölgeye düştüklerinde amaç değerine bu ceza değeri eklenmektedir. Sabit bir ceza değerinin amaç fonksiyonu eklenmesi sonucunda elde

edilen fonksiyon, birçok bölgede süreksizliklere sahip olmaktadır. Ayrıca, uygun bölge çoğu durumda bir yarıkta olabilmekte ve tek yönlü aramaların zorlanmasına neden olmaktadır. Dolasıyla, kısıtlı problemler için çok yönlü bileşke aramaları kullanılmıştır.

3.4.1. Gerilim/Sıkıştırma Yay Tasarım Problemi

Gerilim/sıkıştırma yay tasarım problemi, Şekil 23'te gösterilen bir gerilim/sıkıştırma yayının minimum ağırlığını bulmayı amaçlamaktadır (Coello, 2000; Faramarzi vd., 2020).



Şekil 23. Gerilim/sıkıştırma yayı

Gerilim/sıkıştırma yay tasarım problemi, tel çapı (d), ortalama bobin çapı (D) bobin sayısı (N) olmak üzere 3 tasarım değişkeninden oluşmaktadır (Coello, 2000). Bu problemin matematiksel modeli,

$$\mathbf{x} = [x_1 \quad x_2 \quad x_3] = [d \quad D \quad N]$$

$$\min f(\mathbf{x}) = (x_3 + 2)x_2x_1^2$$

$$g_1(\mathbf{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$$

$$g_2(\mathbf{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leq 0$$

$$g_3(\mathbf{x}) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0$$

$$g_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

$$x_1 \in [0.05, 2.00]$$

$$x_2 \in [0.25, 1.30]$$

$$x_1 \in [2.00, 15.0]$$

biçiminde tanımlanmaktadır.

ALSO algoritması, gerilim/sıkıştırma yay tasarım problemine uygulanmış ve GA (Coello, 2000), PSO (He ve Wang, 2007), BAT (Gandomi vd., 2013), GWO (Mirjalili vd., 2014), WOA (Mirjalili ve Lewis, 2016) ve CSA (Askarzadeh, 2016) algoritmaları ile karşılaştırılmıştır. Tablo 16, gerilim/sıkıştırma yay tasarımı probleminden elde edilen sonuçları göstermektedir.

Tablo 16. Gerilim/sıkıştırma yay tasarım problemi için elde edilen sonuçlar

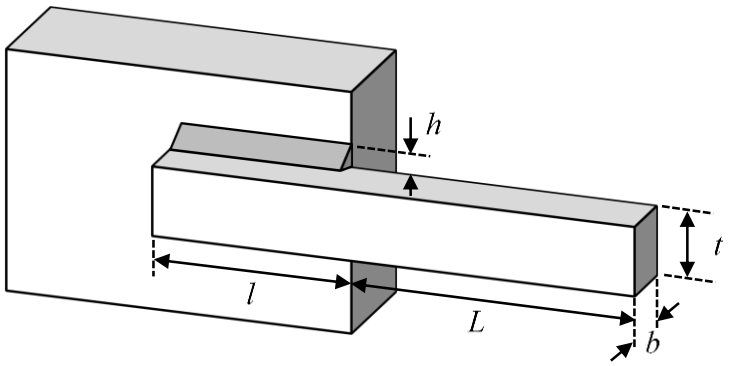
	ALSO	**GA**	**PSO**	**BAT**	**GWO**	**WOA**	**CSA**
$x_1(d)$	0.0517372	0.05148	0.051728	0.05169	0.05169	0.051207	0.051689028
$x_2(D)$	0.357876	0.351661	0.357644	0.35673	0.356737	0.345215	0.356716954
$x_3(N)$	11.2215	11.632201	11.244543	11.2885	11.28885	12.004032	11.2890118
$g_1(\mathbf{x})$	-7.48E-06	-3.34E-03	-8.25E-04	***1.08E-05**	-7.91E-05	-5.64E-04	-2.95E-09
$g_2(\mathbf{x})$	-2.12E-06	-1.10E-04	-2.53E-05	-2.33E-05	-7.51E-06	-3.70E-05	***1.94E-09**
$g_3(\mathbf{x})$	-4.06E+00	-4.03E+00	-4.05E+00	-4.05E+00	-4.05E+00	-4.03E+00	-4.05E+00
$g_4(\mathbf{x})$	-7.27E-01	-7.31E-01	-7.27E-01	-7.28E-01	-7.28E-01	-7.36E-01	-7.28E-01
$f(\mathbf{x})$	**0.012665407**	0.0127047834	0.0126747	**0.01266522**	0.012666	0.0126763	**0.0126652328**

Tablo 16'ya göre, ALSO, BAT ve CSA algoritmaları, gerilim/sıkıştırma yay problemi için en iyi sonuçları vermiştir. Ancak, BAT algoritması problemin ilk kısıtını, CSA

algoritması ise ikinci kısıtını sağlamamaktadır. Bununla birlikte, ALSO algoritması ise tüm kısıtları sağlamaktadır.

3.4.2. Kaynaklı Kiriş Tasarım Problemi

Kaynaklı kiriş tasarım probleminin amacı, bir kaynaklı kirişin (Şekil 24) maliyetini minimize etmektir. Problemin, kaynak inceliği (h), kaynak bağlantı uzunluğu (l), kiriş genişliği (t), kiriş inceliği (b) olmak üzere dört tasarım değişkeni vardır (Coello, 2000; Heidari vd., 2019).



Şekil 24. Kaynaklı kiriş

Kaynaklı kiriş tasarım probleminin matematiksel modeli,

$$\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4] = [h \quad l \quad t \quad b]$$

$$\min f(\mathbf{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \leq 0$$

$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \leq 0$$

$$g_3(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \leq 0$$

$$g_4(\mathbf{x}) = x_1 - x_4 \leq 0$$

$$g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0$$

$$g_6(\mathbf{x}) = 0.125 - x_1 \leq 0$$

$$g_7(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) - 5.00 \leq 0$$

$$x_1 \in [0.1, 2]$$

$$x_2 \in [0.1, 10]$$

$$x_3 \in [0.1, 10]$$

$$x_4 \in [0.1, 2]$$

$$\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \qquad \tau'' = \frac{MR}{J}, \qquad M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\mathbf{x}) = \frac{6PL}{x_4x_3^2}, \qquad \delta(\mathbf{x}) = \frac{6PL^3}{Ex_3^2x_4}$$

$$P_c(\mathbf{x}) = \frac{4.013E\sqrt{x_3^2x_4^6}}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000\ lb, \qquad L = 14\ in., \qquad \delta_{max} = 0.25\ in.,$$
$$E = 30 \times 10^6\ psi,$$

$$G = 12 \times 10^6\ psi, \qquad \tau_{max} = 13\,600\ psi, \qquad \sigma_{max} = 30\,000\ psi$$

biçiminde tanımlanmaktadır. Burada, $\tau(\mathbf{x})$, kaynaktaki kesme gerilimini; $\sigma(\mathbf{x})$, kiriş malzemedeki normal gerilimi; $\delta(\mathbf{x})$, kiriş ucundaki sapmayı ve $P_c(\mathbf{x})$, çubuktaki burkulma yükünü ifade etmektedir.

ALSO algoritması, kaynaklı kiriş tasarım problemine uygulanmış ve GA (Coello, 2000), PSO (He ve Wang, 2007), BAT (Gandomi vd., 2013), GWO (Mirjalili vd., 2014), WOA (Mirjalili ve Lewis, 2016) ve CSA (Askarzadeh, 2016) algoritmaları ile karşılaştırılmıştır. Elde edilen sonuçlar Tablo 17'de gösterilmiştir.

Tablo 17. Kaynaklı kiriş tasarım problemi için elde edilen sonuçlar

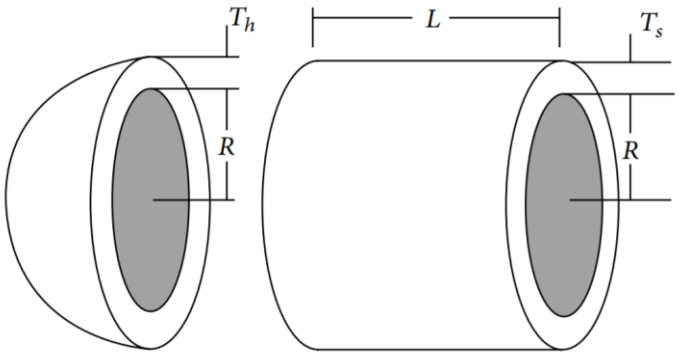
	ALSO	**GA**	**PSO**	**BAT**	**GWO**	**WOA**	**CSA**
$x_1(h)$	0.205163	0.2088	0.202369	0.2015	0.205676	0.205396	0.20572964
$x_2(l)$	3.25999	3.4205	3.544214	3.562	3.478377	3.484293	3.470488666
$x_3(t)$	9.0496	8.9975	9.04821	9.0414	9.03681	9.037426	9.03662391
$x_4(b)$	0.205666	0.21	0.205723	0.2057	0.205778	0.206276	0.20572964
$g_1(\mathbf{x})$	-1.06E+00	-7.58E+02	-8.11E+02	-8.07E+02	-7.94E+02	-7.97E+02	-7.71E+02
$g_2(\mathbf{x})$	-7.67E+01	-3.54E+02	-7.58E+01	-2.74E+01	-8.29E+00	-8.48E+01	-2.31E-06
$g_3(\mathbf{x})$	-2.36E-01	-2.36E-01	-2.36E-01	-2.36E-01	-2.36E-01	-2.36E-01	-2.36E-01
$g_4(\mathbf{x})$	-5.03E-04	-1.20E-03	-3.35E-03	-4.20E-03	-1.02E-04	-8.80E-04	0.00E+00
$g_5(\mathbf{x})$	-8.86E-02	-3.63E+02	-4.47E+00	***5.09E-01**	-4.31E+00	-4.83E+01	-1.24E-06
$g_6(\mathbf{x})$	-8.02E-02	-8.38E-02	-7.74E-02	-7.65E-02	-8.07E-02	-8.04E-02	-8.07E-02
$g_7(\mathbf{x})$	-3.45E+00	-3.41E+00	-3.42E+00	-3.42E+00	-3.43E+00	-3.43E+00	-3.43E+00
$f(\mathbf{x})$	**1.697082867**	1.74830941	1.728024	1.7312	1.72624	1.730499	1.724852309

Tablo 17'ye göre, ALSO algoritmasının en iyi sonucu verdiği görülmektedir. Bununla birlikte, BAT algoritması problemin beşinci kısıtını sağlamamaktadır.

3.4.3. Basınçlı Kap Tasarım Problemi

Basınçlı kap tasarım problemi, mühendislik uygulamalarında sıklıkla kullanılan bir problemdir. Basınçlı kap, iki ucu yarı küresel başlıklarla kapalı olan bir silindirdir.

Şekil 25'te şematik olarak gösterilen basınçlı kap probleminin amacı, malzeme, şekillendirme ve kaynak maliyetlerini minimize etmektir (Coello, 2000; Cagnina vd., 2008; Sadollah vd., 2013).



Şekil 25. Basınçlı kap

Basınçlı kap tasarım probleminde, kabuk kalınlığı (T_s), başlık kalınlığı (T_h), iç yarıçap (R) ve silindirik kısmın uzunluğu (L) olmak üzere dört karar değişkeni vardır. Bu değişkenlerden, x_1 ve x_2 ayrık, x_3 ve x_4 sürekli değişkenlerdir (Askarzadeh, 2016). Basınçlı kap tasarım probleminin matematiksel modeli,

$$\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4] = [T_s \quad T_h \quad R \quad L]$$

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(x) = -x_3 + 0.00954x_3 \leq 0$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0$$

$$g_4(x) = x_4 - 240 \leq 0$$

$$x_1 \in [0, 99] \times 0.0625$$

$$x_2 \in [0, 99] \times 0.0625$$

$$x_3 \in [10, 200]$$

$$x_4 \in [10, 200]$$

biçiminde tanımlanmaktadır.

ALSO algoritması, basınçlı kap tasarım problemine uygulanmış ve GA (Coello, 2000), PSO (He ve Wang, 2007), BAT (Gandomi vd., 2013), GWO (Mirjalili vd., 2014), WOA (Mirjalili ve Lewis, 2016) ve CSA (Askarzadeh, 2016) algoritmaları ile karşılaştırılmıştır. Tablo 18'de, basınçlı kap tasarım probleminden elde edilen sonuçlar verilmektedir.

Tablo 18. Basınçlı kap tasarım problemi için elde edilen sonuçlar

	ALSO	**GA**	**PSO**	**BAT**	**GWO**	**WOA**	**CSA**
$x_1(T_s)$	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125
$x_2(T_h)$	0.4375	0.4375	0.4375	0.4375	***0.4345**	0.4375	0.4375
$x_3(R)$	42.098437123378	40.3239	42.091266	42.0984456	42.089181	42.0982699	42.09844539
$x_4(L)$	176.6367864059546	200	176.7465	176.6365958	176.758731	176.638998	176.6365986
$g_1(\mathbf{x})$	-1.635188e-07	-3.42E-02	-1.39E-04	***8.00E-11**	-1.79E-04	-3.39E-06	-3.97E-09
$g_2(\mathbf{x})$	-0.03588	-5.28E-02	-3.59E-02	-3.59E-02	-3.30E-02	-3.59E-02	-3.59E-02
$g_3(\mathbf{x})$	-0.476468	-3.04E+02	-1.16E+02	-4.97E-05	-4.06E+01	-1.25E+00	-8.72E-04
$g_4(\mathbf{x})$	-63.3632136	-4.00E+01	-6.33E+01	-6.34E+01	-6.32E+01	-6.34E+01	-6.34E+01
$f(\mathbf{x})$	**6059.717367**	6288.7445	6061.0777	**6059.714335**	**6051.5639**	6059.741	**6059.714363**

Tablo 18'den görüldüğü üzere, ALSO ve CSA algoritmaları, diğer algoritmalara göre yüksek bir performans sergilemiştir. Ayrıca, BAT algoritması problemin birinci kısıtını ve GWO algoritması, x_2 değişkenine ait sınır kısıtını sağlamamaktadır.

4. SONUÇLAR

Optimizasyon, verilen bir problem için en iyi çözümün bulunması işlemidir. Mühendislik, ekonomi, işletme, matematik, tıp gibi birçok alandaki problemlerin çözümünde optimizasyon kullanılır. Son yıllarda, bilgisayar bilimlerindeki gelişmelere paralel olarak, biyolojik evrim, sürü davranışı, fiziksel ve kimyasal süreçlerden temel alınarak birçok meta-sezgisel optimizasyon algoritması önerilmiştir. Meta-sezgisel algoritmalar, verilen optimizasyon problemi için en iyi çözümün bulunmasını amaçlamaktadır. Bütün problemlerin çözümünü sağlayacak tek bir yöntemin olması mümkün değildir. Bu sebeple, literatürde birçok meta-sezgisel algoritma geliştirilmiştir ve geliştirilmeye de devam edecektir.

Bu tez çalışması kapsamında, sürekli optimizasyon problemlerini çözmek için çekirge sürülerinin davranışını modelleyen sürü zekâsı tabanlı bir optimizasyon algoritması önerilmiştir. Önerilen ALSO algoritması, basit ve uygulanması kolay bir yöntemdir. Diğer yöntemlerden farklı olarak önerilen algoritma, arama uzayındaki detayları daha iyi ortaya çıkarmak için bir duyarlılık fonksiyonu kullanmaktadır. Duyarlılık fonksiyonu, iki parçalı bir fonksiyondur. Birinci kısımda algoritma keşif işlemini gerçekleştirirken, ikinci kısımda ise sömürü işlemine odaklanmaktadır. Yani, duyarlılık fonksiyonun birinci kısmında, algoritmada genel çözüm aranırken, ikinci kısmında genel çözümün hassasiyeti iyileştirilir. Böylece, ALSO algoritması arama uzayında keşif ve sömürü arasında uygun bir denge kurar. Bununla birlikte, duyarlılık fonksiyonu λ parametresine göre ayarlanabilir. Ayrıca, literatürdeki yöntemler, tüm iterasyonlar boyunca, elde edilen en iyi çözüm etrafında kümelenir ve bu çözümün duyarlılığını iyileştirmeye çalışır. Bu algoritmalardan farklı olarak, ALSO algoritması mevcut en iyi çözümün duyarlılığını iyileştirirken, aynı zamanda bu çözümden daha iyi bir çözüm aramaya da devam eder. Böylece, yerel çözüme takılma olasılığını azaltmaya çalışır.

Önerilen algoritmanın yakınsama ve yerel optimumdan kaçınma performansını ölçmek için yaygın ve güncel optimizasyon algoritmaları ile kapsamlı bir değerlendirme ve deneysel bir çalışma yapılmıştır. Simülasyon çalışmasında, tek tepeli, çok tepeli ve birleşik olmak üzere toplamda 22 test fonksiyonu üzerinde algoritma analiz edilmiştir. Ayrıca, PSO, ABC, CS, BAT, GWO, WOA, HHO, BRO ve RSO olmak üzere literatürde yaygın olarak kullanılan ve yeni geliştirilen optimizasyon algoritmaları ile önerilen algoritma

karşılaştırılmıştır. Elde edilen sonuçlar, ALSO algoritmasının rekabetçi bir algoritma olduğunu göstermiştir. Bununla birlikte, tek tepeli test fonksiyonları, ALSO algoritmasının yakınsama üstünlüğünü kanıtlamıştır. Ayrıca, ALSO algoritmasının yerel optimumdan kaçınma yeteneği, çok tepeli test fonksiyonlarından elde edilen sonuçlarla doğrulanmıştır. Bunların yanı sıra, CEC2017 birleşik test fonksiyonları üzerinde yapılan uygulamalar, ALSO algoritmasının keşif ve sömürü arasında uygun bir denge kurduğunu göstermiştir.

Çalışma zamanı açısından ele alındığında, ALSO algoritması karşılaştırılan diğer algoritmalara göre daha kısa sürede optimal çözüme ulaşmaktadır. Ayrıca, ALSO algoritması, yüksek boyutlu problemlerde daha az bellek alanına ihtiyaç duymaktadır.

Önerilen ALSO algoritması ve karşılaştırılan diğer algoritmalar arasında istatistiksel olarak anlamlı bir fark olup olmadığını doğrulamak amacıyla Wilcoxon sıra toplam testi kullanılmıştır. 30, 50 ve 100 boyutları için %5 anlamlılık düzeyinde Wilcoxon sıra toplam sıra toplam testi uygulanmış ve her boyutta ALSO algoritması ve diğer algoritmalar arasında istatistiksel olarak anlamlı bir fark olduğu ortaya konulmuştur. Bununla birlikte, önerilen algoritmanın performansı 30 boyut için %86, 50 boyut için %84 ve 100 boyut için %83 olarak hesaplanmıştır.

Önerilen algoritmanın gerçek yaşam problemlerine uygulanabilirliği göstermek için ALSO algoritması, gerilim/sıkıştırma yay tasarım problemi, kaynaklı kiriş tasarım problemi ve basınçlı kap tasarım problemi olmak üzere üç mühendislik tasarım problemine uygulanmıştır. Elde edilen sonuçlar, ALSO algoritmasının yüksek bir performansa sahip olduğunu ortaya koymuştur.

Sonuç olarak, önerilen ALSO algoritmasının literatürdeki optimizasyon algoritmaları ile rekabet edebilir olduğu söylenebilir. Bununla birlikte, ALSO algoritmasının sayısal test fonksiyonlarını çözmedeki etkisinin yanı sıra, gerçek yaşam problemleri için de tatmin edici sonuçlar sağladığı görülmüştür.

5. ÖNERİLER

Doktora tezi olarak sunulan bu çalışmada, Yapay Çekirge Sürü Optimizasyonu (ALSO) adı verilen ve çekirge sürülerinin sıçrama ve istila etme davranışlarını ilham alan sürü zekâsı tabanlı meta-sezgisel bir algoritma geliştirilmiştir. Bu tez konusu ile ilgili çalışma yapmak isteyecek araştırmacı ve uygulayıcılara iletilmek istenilen öneriler aşağıdaki gibi belirtilmiştir:

- Önerilen algoritma, farklı optimizasyon problemleri üzerinde uygulanarak, algoritmanın farklı alanlardaki uygulamaları yapılabilir.
- Bu tez çalışmasında önerilen algoritma, 3 mühendislik tasarım problemine uygulanmıştır. Daha fazla mühendislik tasarım problemine uygulanarak, algoritmanın diğer problemler üzerindeki performansı değerlendirilebilir.
- Önerilen algoritmanın kısıtlı optimizasyon problemlerine uygulanmasında, ceza fonksiyonu yaklaşımı kullanılmıştır. Farklı yaklaşımlar geliştirilerek algoritmanın kısıtlı optimizasyon problemlerindeki başarımı ölçülebilir.
- Önerilen algoritmada, ailedeki çekirge sayısının boyuta bağlı olarak değerlendirilmesine ilişkin çalışmalar gerçekleştirilebilir.

6. KAYNAKLAR

Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z. W. ve Gandomi, A. H., 2022. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer, Expert Systems with Applications, 191, 116158.

Ahmed, H. ve Glasgow, J., 2012. Swarm intelligence: concepts, models and applications, Technical Report, Queen's University, School of Computing, Kingston, Ontario, Canada.

Akay, B. ve Karaboga, D., 2012. Artificial bee colony algorithm for large-scale problems and engineering design optimization, Journal of intelligent manufacturing, 23, 4, 1001-1014.

Alatas, B., 2011. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization, Expert Systems with Applications, 38, 13170-13180.

Arora, S. ve Singh, S., 2019. Butterfly optimization algorithm: a novel approach for global optimization, Soft Computing, 23, 3, 715-734.

Askarzadeh, A., 2016. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Computers & Structures, 169, 1-12.

Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J. ve Qu, B. Y., 2016. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Technical Report.

Bansal, J. C. ve Pal, N. R. 2019. "Swarm and Evolutionary Computation." 231-246 in *Evolutionary and Swarm Intelligence Algorithms. Studies in Computational Intelligence*, edited by Bansal, J. C., Singh, P. K. ve Pal, N. R. Cham: Springer.

Blum, C. ve Li, X. 2008. "Swarm Intelligence in Optimization." 43-85 in *Swarm Intelligence*, edited by Blum, C. ve Merkle, D. Natural Computing Series ed. Berlin: Springer.

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

Boussaïd, I. 2016. "Some Other Metaheuristics." 229-262 in *Metaheuristics*, edited by Siarry, P. Cham: Springer International Publishing.

Bozorg-Haddad, O., Solgi, M. ve Loáiciga, H. A., 2017. Meta-heuristic and evolutionary algorithms for engineering optimization, John Wiley & Sons.

Cagnina, L. C., Esquivel, S. C. ve Coello, C. A. C., 2008. Solving engineering optimization problems with the simple constrained particle swarm optimizer, Informatica, 32, 3, 2592-2612.

Carroll, C. W., 1961. The created response surface technique for optimizing nonlinear, restrained systems, Operations Research, 9, 2, 169-184.

Cavazzuti, M., 2013. Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics, Springer Science & Business Media.

Chen, S., 2009a. Locust Swarms - A new multi-optima search technique, 2009 IEEE Congress on Evolutionary Computation, May, Trondheim, Norway, 1745-1752.

Chen, S., 2009b. An analysis of locust swarms on large scale global optimization problems, 4th Australian Conference on Artificial Life, December, Melbourne, Australia, 211-220.

Chou, J.-S. ve Truong, D.-N., 2021. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Applied Mathematics and Computation, 389, 125535.

Coello, C. A. C., 2000. Use of a self-adaptive penalty approach for engineering optimization problems, Computers in Industry, 41, 2, 113-127.

Coello, C. A. C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer methods in applied mechanics and engineering, 191, 11-12, 1245-1287.

Collett, M., Despland, E., Simpson, S. J. ve Krakauer, D. C., 1998. The spatial scales of locust gregarisation, Proc. Natl. Acad. Sci., 95, 13052-13055.

Courant, R., 1943. Variational methods for the solution of problems of equilibrium and vibrations, Bull. Amer. Math. Soc., 49, 1, 1-23.

Cuevas, E., Cienfuegos, M., Zaldívar, D. ve Pérez-Cisneros, M., 2013. A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Systems with Applications, 40, 6374-6384.

Cuevas, E., González, A., Zaldívar, D. ve Pérez-Cisneros, M., 2015. An optimisation algorithm based on the behaviour of locust swarms, International Journal of Bio-Inspired Computation, 7, 6, 402-407.

Cuevas, E. ve Rodríguez, A., 2020. Metaheuristic Computation with MATLAB, 1st ed., CRC Press, Boca Raton, FL, USA.

Cuevas, E., Zaldívar, D. ve Pérez-Cisneros, M., 2018. Advances in Metaheuristics Algorithms: Methods and Applications, Springer.

Darwish, A., 2018. Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications, Future Computing and Informatics Journal, 3, 2, 231-246.

Dasgupta, D., 2006. Advances in artificial immune systems, IEEE computational intelligence magazine, 1, 4, 40-49.

Datta, R. ve Deb, K., 2014. Evolutionary Constrained Optimization, Springer India.

de Castro, L. N., 2007. Fundamentals of natural computing: an overview, Physics of Life Reviews, 4, 1, 1-36.

Derrac, J., García, S., Molina, D. ve Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, 1, 3-18.

Dhiman, G., Garg, M., Nagar, A., Kumar, V. ve Dehghani, M., 2021. A novel algorithm for global optimization: Rat Swarm Optimizer, Journal of Ambient Intelligence and Humanized Computing, 12, 8457-8482.

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

Dhiman, G. ve Kumar, V., 2018. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems, Knowledge-Based Systems, 159, 20-50.

Dorigo, M., 1992. Optimization, Learning and Natural Algorithms, PhD Thesis, Politecnico di Milano, Milano, Italy.

Dorigo, M., Birattari, M. ve Stutzle, T., 2006. Ant colony optimization, IEEE computational intelligence magazine, 1, 4, 28-39.

Dorigo, M. ve Di Caro, G., 1999. Ant colony optimization: a new meta-heuristic, Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), July, Washington, DC, USA, 2, 1470-1477.

Dorigo, M., Maniezzo, V., Colorni, A. ve Maniezzo, V., 1991. Positive feedback as a search strategy.

Du, K.-L. ve Swamy, M., 2016. Search and Optimization by Metaheuristics Techniques and Algorithms Inspired by Nature, Birkhäuser, Cham.

Eberhart, R. C. ve Kennedy, J., 1995. A new optimizer using particle swarm theory, Proceedings of the 6th international symposium on micro machine and human science, October, Nagoya, Japan, 1, 39-43.

Ernst, U. R., Van Hiel, M. B., Depuydt, G., Boerjan, B., De Loof, A. ve Schoofs, L., 2015. Epigenetics and locust life phase transitions, Journal of Experimental Biology, 218, 1, 88-99.

Eskandar, H., Sadollah, A., Bahreininejad, A. ve Hamdi, M., 2012. Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems, Computers Structures, 110, 151-166.

Faramarzi, A., Heidarinejad, M., Mirjalili, S. ve Gandomi, A. H., 2020. Marine predators algorithm: A nature-inspired Metaheuristic, Expert Systems with Applications, 113377.

Farmer, J. D., Packard, N. H. ve Perelson, A. S., 1986. The immune system, adaptation, and machine learning, Physica D: Nonlinear Phenomena, 22, 1-3, 187-204.

Farshi, T. R., 2021. Battle royale optimization algorithm, Neural Computing and Applications, 33, 1139-1157.

Feng, L., Ong, Y.-S. ve Gupta, A. 2019. "Genetic Algorithm and Its Advances in Embracing Memetics." 61-84 in *Evolutionary and Swarm Intelligence Algorithms*, edited by Feng, L., Ong, Y. S. ve Gupta, A. Studies in Computational Intelligence ed. Cham: Springer.

Fiacco, A. V. ve McCormick, G. P., 1966. Extensions of SUMT for nonlinear programming: equality constraints and extrapolation, Management science, 12, 11, 816-828.

Fister Jr, I. ve Fister, I. 2021. "A Brief Overview of Swarm Intelligence-Based Algorithms for Numerical Association Rule Mining." 47-59 in *Applied Optimization and Swarm Intelligence*, edited by Osaba, E. ve Yang, X. Springer Tracts in Nature-Inspired Computing ed. Singapore: Springer.

Fister Jr, I., Yang, X.-S., Fister, I., Brest, J. ve Fister, D., 2013. A Brief Review of Nature-Inspired Algorithms for Optimization, arXiv preprint arXiv:1307.4186.

Fister, I., Yang, X.-S. ve Brest, J., 2013. A comprehensive review of firefly algorithms, Swarm and Evolutionary Computation, 13, 34-46.

Gandomi, A. H. ve Alavi, A. H., 2012. Krill herd: A new bio-inspired optimization algorithm, Communications in Nonlinear Science and Numerical Simulation, 17, 4831-4845.

Gandomi, A. H., Yang, X.-S., Alavi, A. H. ve Talatahari, S., 2013. Bat algorithm for constrained optimization tasks, Neural Computing and Applications, 22, 6, 1239-1255.

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

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence, Computers & operations research, 13, 5, 533-549.

Hancer, E., Xue, B., Karaboga, D. ve Zhang, M., 2015. A binary ABC algorithm based on advanced similarity scheme for feature selection, Applied Soft Computing, 36, 334-348.

Harifi, S., Khalilian, M., Mohammadzadeh, J. ve Ebrahimnejad, S., 2019. Emperor Penguins Colony: a new metaheuristic algorithm for optimization, Evolutionary Intelligence, 12, 2, 211-226.

Harifi, S., Mohammadzadeh, J., Khalilian, M. ve Ebrahimnejad, S., 2021. Giza Pyramids Construction: an ancient-inspired metaheuristic algorithm for optimization, Evolutionary Intelligence, 14, 4, 1743-1761.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. ve Al-Atabany, W., 2022. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems, Mathematics and Computers in Simulation, 192, 84-110.

Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S. ve Al-Atabany, W., 2021. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems, Applied Intelligence, 51, 1531-1551.

Hassanien, A. E. ve Emary, E., 2016. Swarm Intelligence: Principles, Advances, and Applications, CRC Press.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. ve Chen, H., 2019. Harris hawks optimization: Algorithm and applications, Future Generation Computer Systems, 97, 849-872.

He, Q. ve Wang, L., 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Engineering applications of artificial intelligence, 20, 1, 89-99.

Holland, J. H., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence., U Michigan Press.

Hoyle, G., 1958. The leap of the grasshopper, Scientific American, 198, 1, 30-35.

Inglis, G. D., Goettel, M. S., Erlandson, M. A. ve Weaver, D. K. 2007. "Grasshoppers and locusts." 627-654 in *Field Manual of Techniques in Invertebrate Pathology*, edited by Lacey, L. A. ve Kaya, H. K. Dordrecht: Springer.

Jain, M., Maurya, S., Rani, A. ve Singh, V., 2018. Owl search algorithm: a novel nature-inspired heuristic paradigm for global optimization, Journal of Intelligent & Fuzzy Systems, 34, 3, 1573-1582.

Jain, M., Singh, V. ve Rani, A., 2019. A novel nature-inspired algorithm for optimization: Squirrel search algorithm, Swarm and evolutionary computation, 44, 2, 148-175.

Kaewkamnerdpong, B. ve Bentley, P. J., 2005. Perceptive particle swarm optimisation: An investigation, Proceedings 2005 IEEE Swarm Intelligence Symposium (SIS 2005), June, Pasadena, CA, USA, 169-176.

Kang, L., Chen, X., Zhou, Y., Liu, B., Zheng, W., Li, R., Wang, J. ve Yu, J., 2004. The analysis of large-scale gene expression correlated to the phase changes of the migratory locust, Proceedings of the National Academy of Sciences of the United States of America, 101, 51, 17611-17615.

Karaboga, D. ve Basturk, B., 2007a. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization, 39, 459-471.

Karaboga, D. ve Basturk, B. 2007b. "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems." 789-798 in *Foundations of Fuzzy Logic and Soft Computing. IFSA 2007. Lecture Notes in Computer Science*, edited by Melin, P., Castillo, O., Aguilar, L. T., Kacprzyk , J. ve Pedrycz , W. Berlin, Heidelberg: Springer.

Karaboğa, D., 2005. An idea based on honey bee swarm for numerical optimization, Technical Report, Erciyes University, Engineering Faculty, Computer Engineering Department , Kayseri.

Kashan, A. H., 2009. League championship algorithm: a new algorithm for numerical function optimization, 2009 international conference of soft computing and pattern recognition, 43-48.

Kashani, A. R., Chiong, R., Mirjalili, S. ve Gandomi, A. H., 2021. Particle Swarm Optimization Variants for Solving Geotechnical Problems: Review and Comparative Analysis, Archives of Computational Methods in Engineering, 1871-1927.

Kaveh, A. ve Farhoudi, N., 2013. A new optimization method: Dolphin echolocation, Advances in Engineering Software, 59, 53-70.

Kesemen, O. ve Özkul, E., 2018. Solving cross-matching puzzles using intelligent genetic algorithms, Artificial Intelligence Review, 49, 2, 211-225.

Kirkpatrick, S., Gelatt, C. D. ve Vecchi, M. P., 1983. Optimization by simmulated annealing, science, 220, 4598, 671-680.

Koza, J. R., 1990. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Technical Report, Stanford University, Computer Science Department, Stanford, CA.

Krause, J., Cordeiro, J., Parpinelli, R. S. ve Lopes, H. S., 2013. A survey of swarm algorithms applied to discrete optimization problems, Swarm Intelligence and Bio-Inspired Computation, 4, 9, 169-191.

Kumar, A. ve Bawa, S., 2020. A comparative review of meta-heuristic approaches to optimize the SLA violation costs for dynamic execution of cloud services, Soft Computing, 24, 6, 3909-3922.

Kumar, D., Gandhi, B. R. ve Bhattacharjya, R. K. 2020. "Firefly Algorithm and Its Applications in Engineering Optimization." 93-103 in *Nature-Inspired Methods for Metaheuristics Optimization*, edited by Bennis, F. ve Bhattacharjya, R. K. Modeling and Optimization in Science and Technologies ed. Cham: Springer.

Lewis, A., 2009. LoCost: a spatial social network algorithm for multi-objective optimisation, 2009 IEEE Congress on Evolutionary Computation, May, Trondheim, Norway, 2866-2870.

Lim, T. Y., 2014. Structured population genetic algorithms: a literature survey, Artificial Intelligence Review, 41, 3, 385-399.

Lynn, N. ve Suganthan, P. N., 2015. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm and Evolutionary Computation, 24, 11-24.

Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M. N., Salcedo-Sanz, S. ve Geem, Z. W., 2013. A survey on applications of the harmony search algorithm, Engineering Applications of Artificial Intelligence, 26, 8, 1818-1831.

Martens, D., Baesens, B. ve Fawcett, T., 2011. Editorial survey: swarm intelligence for data mining, Machine Learning, 82, 1, 1-42.

Meetei, K. T., 2014. A Survey: Swarm Intelligence vs. Genetic Algorithm, International Journal of Science and Research, 3, 231-235.

Mezura-Montes, E. ve Coello, C. A. C., 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm and Evolutionary Computation, 1, 4, 173-194.

Mirjalili, S., 2015. The Ant Lion Optimizer, Advances in Engineering Software, 83, 80-98.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H. ve Mirjalili, S. M., 2017. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Advances in Engineering Software, 114, 163-191.

Mirjalili, S. ve Lewis, A., 2016. The Whale Optimization Algorithm, Advances in Engineering Software, 95, 51-67.

Mirjalili, S., Mirjalili, S. M. ve Lewis, A., 2014. Grey Wolf Optimizer, Advances in engineering software, 69, 46-61.

Mishra, D. ve Shinde, V. 2021. "A review of global optimization problems using meta-heuristic algorithm." 87-106 in *Nature-Inspired Optimization Algorithms: Recent Advances in Natural Computing and Biomedical Applications*, edited by Khamparia, A., Khanna, A., N, N. ve Le Nguyen, B. Berlin, Boston: De Gruyter.

Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F. ve Rodríguez, A., 2020. A better balance in metaheuristic algorithms: Does it exist?, Swarm and Evolutionary Computation, 54, 100671.

Nakrani, S. ve Tovey, C., 2004. On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers, Adaptive Behavior, 12, 3-4, 223-240.

Nanda, S. J. ve Panda, G., 2014. A survey on nature inspired metaheuristic algorithms for partitional clustering, Swarm and Evolutionary computation, 16, 6, 1-18.

Nayyar, A. ve Nguyen, N. G. 2018. "Introduction to Swarm Intelligence." 53-78 in *Advances in Swarm Intelligence for Optimizing Problems in Computer Science*, edited by Nayyar, A., Le, D.-N. ve Nguyen, N. G. Cham: Chapman and Hall/CRC.

Okwu, M. O. ve Tartibu, L. K., 2021. Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications, Studies in Computational Intelligence ed., Springer, Cham.

Osaba, E. ve Yang, X.-S. 2021. "Applied Optimization and Swarm Intelligence: A Systematic Review and Prospect Opportunities." 1-23 in *Applied Optimization and Swarm Intelligence*, edited by Osaba, E. ve Yang, X. Springer Tracts in Nature-Inspired Computing ed. Singapore: Springer.

Osaba, E. ve Yang, X.-S. 2021. "Soccer-Inspired Metaheuristics: Systematic Review of Recent Research and Applications." 81-102 in *Applied Optimization and Swarm Intelligence*, edited by Osaba, E. ve Yang, X.-S. Springer Tracts in Nature-Inspired Computing ed. Singapore: Springer.

Passino, K. M., 2002. Biomimicry of bacterial foraging for distributed optimization and control, IEEE control systems magazine, 22, 3, 52-67.

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S., 2009. GSA: a gravitational search algorithm, Information sciences, 179, 13, 2232-2248.

Sadollah, A., Bahreininejad, A., Eskandar, H. ve Hamdi, M., 2013. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Applied Soft Computing, 13, 5, 2592-2612.

Saremi, S., Mirjalili, S. ve Lewis, A., 2017. Grasshopper optimisation algorithm: Theory and application, Advances in Engineering Software, 105, 30-47.

Scott, J., 2005. The locust jump: an integrated laboratory investigation, Advances in Physiology Education, 29, 1, 21-26.

Sharma, S. ve Kaushik, B., 2021. A survey on nature-inspired algorithms and its applications in the Internet of Vehicles, International Journal of Communication Systems, 34.

Shi, Y. ve Eberhart, R., 1998. A modified particle swarm optimizer, 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), May, Anchorage, AK, USA: BoD-Books on Demand, 69-73.

Simpson, S. J., McCaffery, A. R. ve Hägele, B. F., 1999. A behavioural analysis of phase change in the desert locust, Biological Reviews, 74, 4, 461-480.

Simpson, S. J. ve Sword, G. A., 2008. Locusts, Current Biology, 18, 9, R364-R366.

Simpson, S. J., Sword, G. A. ve Lo, N., 2011. Polyphenism in insects, Current Biology, 21, 18, R738-R749.

Sörensen, K. S. 2018. "A History of Metaheuristics." 791--808 in *Handbook of Heuristics*, edited by Mart, Pardalos, P. M. ve Resende, M. G. C. Cham: Springer International Publishing.

Storn, R. ve Price, K., 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization, 11, 4, 341-359.

Thakare, S., 2013. Comparison of Swarm Intelligence Techniques, International Journal of Computer Science and Business Informatics, 1, 1.

Timmis, J., 2007. Artificial immune systems—today and tomorrow, Natural computing, 6, 1, 1-18.

Topaz, C. M., Bernoff, A. J., Logan, S. ve Toolson, W., 2008. A model for rolling swarms of locusts, The European Physical Journal Special Topics, 157, 93-109.

Topaz, C. M., D'Orsogna, M. R., Edelstein-Keshet, L. ve Bernoff, A. J., 2012. Locust Dynamics: Behavioral Phase Change and Swarming, Plos Computation Biology, 8, 8, 1-11.

Vasuki, A., 2020. Nature-Inspired Optimization Algorithms, CRC Press.

Walker, A., Hallam, J. ve Willshaw, D., 1993. Bee-havior in a mobile robot: The construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment, IEEE International Conference on Neural Networks, March-April, San Francisco, CA, USA, 1, 1-3, 1451-1456.

Wolpert, D. H. ve Macready, W. G., 1997. No free lunch theorems for optimization, IEEE transactions on evolutionary computation, 1, 1, 67-82.

Xia, X., Gui, L., Zhang, Y., Xu, X., Yu, F., Wu, H., Wei, B., He, G., Li, Y. ve Li, K., 2021. A fitness-based adaptive differential evolution algorithm, Information Sciences, 549, 116-141.

Xue, J. ve Shen, B., 2020. A novel swarm intelligence optimization approach: sparrow search algorithm, Systems Science & Control Engineering, 8, 1, 22-34.

Yang, X.-S., 2009. Firefly algorithms for multimodal optimization, International Symposium on Stochastic Algorithms, 169-178.

Yang, X.-S. 2010. "A New Metaheuristic Bat-Inspired Algorithm." 65-74 in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, edited by González, J. R., Pelta, D. A., Cruz, C., Terrazas, G. ve Krasnogor, N. Studies in Computational Intelligence ed. Berlin, Heidelberg: Springer.

Yang, X.-S., 2010. Nature-Inspired Metaheuristic Algorithms, 2nd ed., Luniver Press, Frome. United Kingdom.

Yang, X.-S., 2012. Flower pollination algorithm for global optimization, 11th International Conference on Unconventional Computing and Natural Computation, September, Orléan, France, 240-249.

Yang, X.-S., 2014. Nature-Inspired Optimization Algorithms, 1st ed., Elsevier, London.

Yang, X.-S., 2014. Swarm intelligence based algorithms: a critical analysis, Evolutionary Intelligence, 7, 1, 17-28.

Yang, X.-S. ve Deb, S., 2009. Cuckoo search via Lévy flights, 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), December, Coimbatore, India, 210-214.

Yang, X.-S. ve Deb, S., 2010. Engineering optimisation by cuckoo search, arXiv preprint arXiv:1005.2908, 23, 4, 1001-1014.

Yang, X.-S., Deb, S. ve Fong, S., 2014. Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification, Applied Mathematics & Information Sciences, 8, 3, 977-983.

Yang, X.-S. ve Karamanoglu, M. 2013. "Swarm Intelligence and Bio-Inspired Computation: An Overview." 3-23 in *Swarm Intelligence and Bio-Inspired Computation*, edited by Yang, X.-S., Cui, Z., Xiao, R., Gandomi, A. H. ve Karamanoglu, M. Oxford: Elsevier.

Yeniay, Ö., 2005. Penalty function methods for constrained optimization with genetic algorithms, Mathematical and Computational Applications, 10, 1, 45-56.

Yu, T., Wang, L., Han, X., Liu, Y. ve Zhang, L., 2015. Swarm Intelligence Optimization Algorithms and Their Application.

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

ÖZGEÇMİŞ

Eda ÖZKUL, İlkokulu Canik 100. Yıl İlköğretim okulunda, ortaokulu ise Dikbıyık İlköğretim okulunda tamamladı. 2008 yılında İlkadım 19 Mayıs YDA lisesini bitirdi. 2008-2009 öğretim yılında Karadeniz Teknik Üniversitesi Fen Fakültesi İstatistik ve Bilgisayar Bilimleri bölümünde lisans eğitimine başladı. 2012 yılında bu bölümden mezun oldu. Aynı yıl Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstatistik ve Bilgisayar Bilimleri Anabilim dalında tezli yüksek lisans programına başladı. 2015 yılında "Sayısal Görüntülerde İki Boyutlu Nesne Analizi ile Beton Boşluk Oranının Kestirimi" başlıklı yüksek lisans tezini sunmasının ardından doktora eğitimine başladı. 2013 yılından itibaren Karadeniz Teknik Üniversitesi İstatistik ve Bilgisayar Bilimleri Bölümü'nde araştırma görevlisi olarak görev yapmaktadır. İyi derece İngilizce bilmektedir.