**T.C.**

**BATMAN ÜNİVERSİTESİ**

**LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# META-SEZGİSEL ALGORİTMALARIN SİSTEM TANIMLAMA PROBLEMLERİNE UYGULANMASI

**Metin ZALOĞLU**

**YÜKSEK LİSANS**
**Elektrik Elektronik Mühendisliği Anabilim Dalı**

**Haziran-2023**
**BATMAN**

## TEZ KABUL VE ONAYI

Metin ZALOĞLU tarafından hazırlanan "Meta-sezgisel algoritmaların sistem tanımlama problemlerine uygulanması" adlı tez çalışması 09/06/2023 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Batman Üniversitesi Lisansüstü Eğitimi Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

| **Jüri Üyeleri** | **İmza** |
|---|---|
| **Başkan**<br>Dr. Öğr. Üye Cafer BUDAK | …….. |
| **Danışman**<br>Dr. Öğretim Üye Şehmus FİDAN | …….. |
| **Üye**<br>Dr. Öğr. Üye Ömer Ali KARAMAN | …….. |

Yukarıdaki sonucu onaylarım.

Prof. Dr. Osman PAKMA
Lisansüstü Eğitim Enstitüsü MüdürV.

## TEZ BİLDİRİMİ

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

## DECLARATION PAGE

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

Metin ZALOĞLU

Tarih:

# ÖZET

## YÜKSEK LİSANS TEZİ

## META-SEZGİSEL ALGORİTMALARIN SİSTEM TANIMLAMA PROBLEMLERİNE UYGULANMASI

**Metin ZALOĞLU**

**Batman Üniversitesi**
**Lisansüstü Eğitim Enstitüsü**

**Elektrik Elektronik Mühendisliği Anabilim Dalı**

**Danışman: Dr. Öğr. Üye. Şehmus FİDAN**

**2023, 99 Sayfa**

**Jüri**
**Dr. Öğr. Üye. Şehmus FİDAN**
**Dr. Öğr. Üye. Ömer Ali KARAMAN**
**Dr. Öğr. Üye. Cafer BUDAK**

Son yıllarda çeşitli optimizasyon problemlerini çözmek için kullanılan meta-sezgisel optimizasyon algoritmalarının sayısında önemli artışlar yaşanmıştır. Bu tür algoritmalar, biyolojik evrim, sürü davranışı, bitki büyüme süreçleri vb. gibi fenomenlerden ilham alarak tasarlanmıştır. Çok çeşitli algoritmalar olmakla birlikte genetik algoritma, parçacık sürü optimizasyon algoritmaları oldukça popülerdir. Meta-heuristik algoritmalar, makine öğrenmesinde hiperparametre hesaplama, kontrolör tasarımı, finans vb. çok çeşitli uygulama alanlarında etkin bir şekilde kullanılmaktadır. Yapılan literatür araştırmalarında meta-heuristik algoritmaların sistem tanımlama problemlerine uygulanmasında önemli eksiklikler olduğu belirlenmiştir. Sistem tanımlama yöntemleri, sistemin giriş ve çıkış verilerini kullanarak sistemin matematiksel modelini belirlemeye çalışır. Bir sistemin matematiksel modelini elde etmek genellikle zahmetli ve karmaşık bir süreç olabilir. Bu sürecin giriş/çıkış verilerinin analiz edilmesi yoluyla sistem tanımlama yöntemleri kullanarak aşmak mümkündür. Bu şekilde, sistemin davranışını anlamak ve optimize etmek için kullanılabilecek bir model elde edilebilir. Bu çalışmada, Meta-sezgisel algoritmalar, saç kurutma(hair-dryer) deney setinden alınan giriş/çıkış verileri kullanılarak sistemin modelini elde etmek için kullanılmıştır. Tezde; yapay ekosistem (AEO), çiçek tozlaşma (FPA), karınca aslanı (ALO), güve-alev (MFO), halat çekme (TWO), atom arama (ASO), beyin fırtınası (BSO), su döngüsü (WCA), mercan resifleri (CRO) ve yaşam seçimi tabanlı algoritma (LCO) gibi çeşitli meta-sezgisel optimizasyon algoritmaları ele alınmış ve sistem tanımlama problemine uygulanmıştır. Belirtilen algoritmaların zaman, maksimum jenerasyon, erken durdurma ve fonksiyon hesaplama sınırlılıkları ele alınmış ve performansları incelenmiştir. Algoritmaların performanslar karşılaştırıldığında AEO algoritması, diğer algoritmalara göre yüksek bir performans göstermiştir. Yapılan analizler sonucunda önerilen meta-sezgisel algoritmaların sistem tanımlama problemlerine kolaylıkla ve başarıyla uygulanabileceği görülmüştür.

**Anahtar Kelimeler:** *Meta-Sezgisel Algoritmalar, Sistem Tanımlama, Yapay ekosistem algoritma*

# ABSTRACT

## MS THESIS

## APPLICATION OF META-HEURISTIC ALGORITHMS TO SYSTEM IDENTIFICATION PROBLEMS

**Metin ZALOĞLU**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCEOF BATMAN UNIVERSITY**

**THE DEGREE OF MASTEROF SCIENCE IN ELECTRICALAND ELECTRONICS ENGINEERING**

**Advisor: Dr. Şehmus FİDAN**

**2023, 99 Pages**

**Jury**
**Dr. Şehmus FİDAN**
**Dr. Ömer Ali KARAMAN**
**Dr. Cafer BUDAK**

In recent years, there has been a significant increase in the number of meta-heuristic optimization algorithms used to solve various optimization problems. These algorithms are designed by taking inspiration from biological evolution, swarm behavior, plant growth processes, and other phenomena. While there are a wide variety of algorithms, genetic algorithms and particle swarm optimization algorithms are quite popular. Meta-heuristic algorithms are effectively used in a wide range of applications such as hyperparameter optimization in machine learning, controller design, finance, and more. Literature research has identified important shortcomings in the application of meta-heuristic algorithms to system identification problems. System identification methods aim to determine the mathematical model of a system using its input and output data. Obtaining the mathematical model of a system can often be a tedious and complex process. However, this process can be overcome by using system identification methods based on the analysis of input-output data. In this way, a model that can be used to understand and optimize the behavior of the system can be obtained. In this study, various meta-heuristic algorithms were employed, including Artificial Ecosystem Optimization (AEO), Flower Pollination Algorithm (FPA), Ant Lion Optimizer (ALO), Moth Flame Optimization (MFO), Tug of War Optimization (TWO), Atom Search Optimization (ASO), Brain Storm Optimization (BSO), Water Cycle Algorithm (WCA), Coral Reefs Optimization (CRO), and Life Choice-Based Optimization Algorithm (LCO). These algorithms were applied to obtain the model of the system using the input/output data obtained from a hair dryer experiment. Factors such as time, maximum generation, early termination, and function computation limitations were considered, and the performance of the algorithms was examined. When comparing the performance of the algorithms, the AEO algorithm exhibited higher performance compared to other algorithms. The conducted analyses revealed that the proposed meta-heuristic algorithms can be easily and successfully applied to system identification problems.



**Keywords:**, *Artificial Ecosystem Alghorithm, Meta-Heuristic Alghorithm, System Identification*

# ÖNSÖZ

Bu çalışmanın gerçekleştirilmesinde bana büyük yardımları dokunan tez danışmanım sayın Dr. Öğr. Üyesi Şehmus FİDAN'a teşekkürlerimi sunar, tez konusunu araştırma ve yazma sürecinde beni motive eden, tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen sevgili aileme ve bu süreçte hep yanımda olan dostlarıma teşekkürlerimi bir borç bilirim.

Metin ZALOĞLU
BATMAN-2023

## İÇİNDEKİLER

# ÇİZELGE LİSTESİ

## ŞEKİL LİSTESİ

## SİMGELER VE KISALTMALAR

**Simgeler**

| | |
|---|---|
| $N$ | Bir popülasyonun büyüklüğü |
| $T$ | Maksimum iterasyon sayısı |
| $L, U$ | Alt limit ve üst limit |
| $a$ | Lineer ağırlık katsayısı |
| $x_{rand}$ | Arama uzayında rastgele üretilen bir bireyin konumunu temsil eder |
| $C, D$ | Tüketim ve Ayrışma Faktörü |
| $e, h$ | Ağırlık katsayıları |
| $x_i$ | Bireyin konumu |
| $L$ | Levy dağılımının matematiksel ifadesini |
| $\Gamma(\lambda)$ | Standart gama fonksiyonu |
| $s$ | Adım büyüklüğü |
| $x^{t+1}$ | Çözüm vektörü |
| $\gamma$ | Adım boyutunu ayarlama faktörü |
| $n$ | Maksimum iterasyon sayısı |
| $t$ | Rastgele yürüyüş adımları |
| $cumsum$ | Kümülatif toplam |
| $r(t)$ | Bir stokastik fonksiyon |
| $M_i$ | Güvenin mevcut konumu |
| $F_j$ | Yaklaşılmak istenen j. alevi ifade eder |
| $D_i$ | Güve ile alev arasındaki mesafeyi temsil eder |
| $B$ | Sarmalın şeklini tanımlayan bir sabittir |
| $\boldsymbol{\mu}$s | Statik sürtünme katsayısı |
| $\boldsymbol{\mu}$k | Kinematik sürtünme katsayısı |
| $F_i$ | Bir etkileşim kuvvetini ifade eder |
| $G_i$ | Bir kısıt kuvvetini ifade eder |
| $m_i$ | Kütle |
| $a_i$ | İvme |
| $u$ | Giriş sinyali |
| $y$ | Çıkış sinyali |
| $r_1,r_2,r_3,r_4$ | [0, 1] aralığında rastgele oluşturulan vektörler |
| $v_1, v_2,$ | Ortalama değeri 0 ve standart sapması 1 olan normal dağılım |
| $y(k)$ | Bağımlı değişkenin (zaman serisi) mevcut değerini temsil eder |
| $e(k)$ | Dışsal değişkenin mevcut değerini temsil eder |
| $u(k)$ | Mevcut giriş değerini temsil eder |

**Kısaltmalar**

| | |
|---|---|
| AEO | Artificial Ecosystem-based Optimizer (Yapay Ekosistem Tabanlı Optimizasyon) |
| AEONM | Novel Hybrid Metaheuristic Optimization Algorithm (Yeni Hibrit Metasezgisel Optimizasyon Algoritması) |
| FPA | The flower pollination algorithm (Çiçek Tozlaşma Algoritması) |
| ALO | Ant Lion Optimizer (Karınca Aslanı Algoritması) |
| MFO | Moth-Flame Optimizasyon (Güve-Alev Optimizasyon Algoritması) |
| TWO | Tug of War Optimization (Halat Çekme Optimizasyonu (TWO) |
| ASO | Atom Search Optimization (Atom Arama Optimizasyonu) |
| BSO | Brain Storm Optimization (Beyin Fırtınası Optimizasyonu) |
| WCA | Water Cycle Algorithm (Su Döngüsü Algoritması) |
| CRO | Coral Reefs Optimization (Mercan Resifleri Optimizasyonu) |
| LCO | Life Choice Based Optimization (Yaşam Seçimi Tabanlı Optimizasyon) |
| ACO | Ant Colony Optimization (Karınca kolonisi optimizasyonu) |
| SA | Simuliating Annehalition (Benzetilmiş tavlama) |
| DE | Differantial Evolution (Diferansiyel evrim) |
| SI | System Identification (Sistem Tanımlama) |
| LPSP | Loss of Power Supply Probability (Kayıp güç arzı olasılığı) |
| REF | Renewable Energy Fraction (Yenilenebilir enerji oranı) |
| DSM | Demand Side Management (Talep Tarafı Yönetimi) |
| HHO | Harris Hawk Optimization (Harris Hawk Optimizasyonu) |
| FSA | Future Search Algorithm (Gelecek Arama Algoritması) |
| COE | Cost of Energy (Enerji maliyeti) |
| CSA | Cuckoo Search Algorithm (Guguk kuşu arama algoritması) |
| HSI | Hyperspectral imaging (Hhiperspektral görüntüleme) |
| PSO | Particle Swarm Optimization (Parçacık Sürüsü Optimizasyonu) |
| PV | Photovoltaik |
| GA | Genetic Algorithm (Genetik Algoritma) |
| BFO | Bacterial Foraging Optimization (Bakteriyel Toplayıcı Optimizasyondan) |
| MRG | Manyetik rezonans görüntüleme |
| HELMCO | Holoentropy Life Choice Optimization (Holoentropi Yaşam Seçimi Optimizasyonu) |
| DR | Diyabetik Retinopati |
| FBSO | Fuzzy brain storming optimization (Bulanık beyin fırtınası optimizasyonu) |

# 1. GİRİŞ

En uygun çözümü bulmaya çalışan optimizasyon yöntemleri uzun yıllardır araştırmacıların ilgisini çekmektedir. Çok sayıda bilim insanı, çeşitli gerçek dünya optimizasyon problemlerini çözmek için farklı yöntemler araştırmakla birlikte en sık kullanılan yaklaşımlar genellikle basit ve ideal matematiksel modelleri benimseyen sayısal yöntemlerdir.

Kullanılan bu yöntemler başlangıç noktalarına çok duyarlıdırlar; özellikle ele alınan problemler birden fazla veya keskin tepe noktalarına sahip olduğunda, başlangıç noktalarının yanlış seçimi, global optimumu aramayı zor ve kararsız hale getirmektedir. Son zamanlarda, çok sayıda karmaşık optimizasyon problemi ortaya konmuştur. Bu problemler genellikle çeşitli alanlarda, çoklu karar değişkenleri, karmaşık doğrusal olmayan kısıtlamalar ve amaç fonksiyonları içeren zorluklarla karşılaşmaktadır. Bu tür zorluklar giderek artan bir şekilde ortaya çıkmaktadır. Bu nedenle, bu karmaşık problemler geleneksel sayısal yöntemlerle genellikle kabul edilebilir bir zaman ve çözüm kesinliğiyle iyi bir şekilde çözülememektedir. Ancak doğa, bu karmaşık sorunların üstesinden gelmek için düşünceler, ilham ve kavramlar dahil olmak üzere bize bol miktarda kaynak sunmaktadır.

Canlı organizmalar, nesillerini devam ettirebilmek adına bulundukları ortama uyum sağlamaya çalışmaktadır. Bu adaptasyon sürecinde, organizmalar kendilerine çevresel avantaj sağlayacak olumlu özellikleri ekleyerek, çevresel olarak dezavantajlı olan özelliklerden kurtulmaya çalışırlar. Organizmalar, sürekli adaptasyon süreçleriyle çevrelerinde en avantajlı özellikleri öğrenmeye başlar. Bu süreç, organizmaların çevresel koşullara uyum sağlayarak daha dayanıklı hale gelmesini sağlar. Sonuçta doğadan ilham alan algoritmalar optimizasyon problemlerinin çözümü için makine öğrenmesi algoritmalarında yaygın olarak kullanılmaya başlanmıştır. Meta-sezgisel algoritmaları, çözümü zor olan veya karmaşık yapıya sahip problemleri çözmek için kullanırız. Aşağıda meta-sezgisel algoritmaların önemli kullanım nedenleri gösterilmiştir.

1. Genel Uygulanabilirlik: Meta-sezgisel algoritmalar, farklı türde optimizasyon problemlerine uygulanabilir. Bu algoritmalar, problemin özelliklerine bağlı olarak esnek bir şekilde uyarlanabilirler.

2. Karmaşık Problemlerin Çözümü: Karmaşık optimizasyon problemleri, çoklu değişkenler, doğrusal olmayan kısıtlamalar veya büyük arama alanları gibi zorluklar içerir. Meta-sezgisel algoritmalar, bu tür karmaşık problemlerin çözümünde etkilidir.

3. Yakınsama Garantisi Olmaması: Bazı problemler için optimal çözümü garanti etmek zor veya hatta imkânsız olabilir. Meta-sezgisel algoritmalar, optimal çözüm garantisi olmadan iyi sonuçlar.

4. Heuristik Arama: Meta-sezgisel algoritmalar, bir çözüm arama alanında keşif yaparak, heuristik veya sezgisel bilgileri kullanır. Bu bilgiler, algoritmanın daha hızlı bir şekilde çözüme yaklaşmasına yardımcı olur.

5. Paralel ve Dağıtık Uygulanabilirlik: Meta-sezgisel algoritmalar, paralel ve dağıtık hesaplama yapılarına uyarlanabilir. Bu sayede büyük ölçekli problemlerin çözümünde daha hızlı sonuçlar elde edilebilir.

Meta-sezgisel algoritmalar, optimizasyon problemlerinin pratik ve etkili bir şekilde çözümünde kullanılırken, problem özellikleri ve kullanım senaryosuna bağlı olarak farklı meta-sezgisel yaklaşımlar tercih edilebilir.

Şekil 1.1'de bir karınca kolonisinin bir engel karşısında yeni duruma alışma aşamaları gösterilmiştir. Karıncalar bir engelle karşılaştıklarında belli bir duraksamadan sonra engeli aşmayı binlerce yıldır doğal olarak başarabilmektedirler. Bu tür doğadan alınan örnekler araştırmacılar için ilgi odağı olmuş ve doğadan ilham alan meta-sezgisel algoritmalar ortaya çıkmaya devam etmiştir. Doğadan ilham alan algoritmaların yanı sıra, birçok farklı algoritma türü ve yaklaşımı kullanılmaktadır. Örneğin:

1. Matematiksel Programlama Algoritmaları: Lineer programlama, tam-sayılı programlama, karma tam-sayılı programlama gibi matematiksel programlama yöntemleri optimizasyon problemlerinin çözümünde yaygın olarak kullanılır.

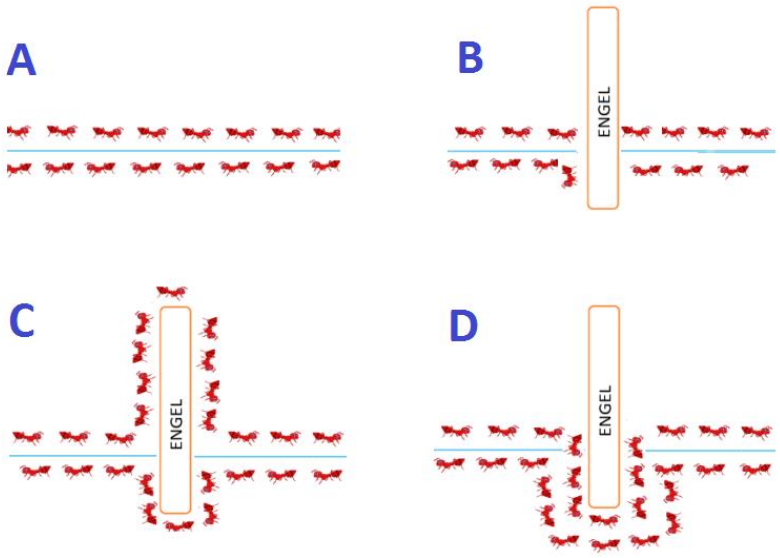
2. Yerel Arama Algoritmaları: Simüle Edilen Tavlama (Simulated Annealing), Tabu Arama (Tabu Search) gibi yerel arama algoritmaları, bir başlangıç çözümünden başlayarak aday çözümleri iteratif olarak iyileştirme yolunu izler.

3. Evrimsel Algoritmalar: Genetik Algoritmalar (Genetic Algorithms), Evrim Stratejileri (Evolution Strategies), Genetik Programlama (Genetic Programming) gibi evrimsel algoritmalar, popülasyon tabanlı optimizasyon yaklaşımlarıdır ve doğal seleksiyon ve çaprazlama gibi evrimsel operatörleri kullanır.

4. Swarm Zeka Algoritmaları: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization), Yarasaların Sürü Davranışı (Bat Algorithm) gibi swarm zeka algoritmaları, sürü tabanlı yöntemlerdir ve bireylerin bir araya gelerek problem çözümünde işbirliği yapmasına dayanır.

5. Bulanık Mantık: Bulanık mantık, belirsizlik ve kesin olmayan verilerin işlendiği ve kararların verildiği bir yöntemdir. Bulanık mantık, optimizasyon problemlerinde kullanılabilir ve belirsizlikleri ele almak için etkilidir.

6. Yapay Sinir Ağları: Yapay Sinir Ağları (Artificial Neural Networks), örneklerden veya verilerden öğrenme yeteneği olan algoritmalar olup, derin öğrenme ve sinir ağı tabanlı optimizasyon yöntemleri içerir.




Şekil 1.1. Karınca Kolonisi Yeni Duruma Alışma Evresi

Bireylerin (örneğin; kuşlar, karıncalar) popülasyon hareketi; Dorigo ve ark., tarafından tanıtılan karınca kolonisi optimizasyonu (ACO-Ant Colony Optimization), bir karınca kolonisinin yiyecek arama sürecinden etkilenerek ortaya çıkmıştır (Dorigo, Maniezzo, ve Colorni, 1996, s.29) Kirkpatrick ve diğerleri tarafından önerilen benzetilmiş tavlama (SA-Simuliating Annehalition), fiziksel malzemede kullanılan tavlama işleminden kaynaklanır (Kirkpatrick, Gelatt ve Vecchi, 1983, s.671). Stern tarafından sunulan diferansiyel evrim (DE-Differantial Evolution), genetik mirasın biyolojik süreçlerini ve en uygun olanın hayatta kalmasını simüle eder (Storn ve Price, 1997, s.341). Bu yaklaşımlar, özellikle türevlenemeyen, sürekli olmayan, çok modlu ve çok boyutlu problemler için daha iyi bir optimizasyon performansı sağladıklarından dolayı akıllı hesaplamanın bilimsel alanlarında çok popülerdirler (Zhao, Wang, ve Zhang, 2020, s.9383). Dikkatleri ve uygulamaları ile çok sayıda başka optimizasyon algoritması geliştirilmiş ve bir dizi alana başarıyla uygulanmıştır. Bu algoritmalar kabaca üç sınıfa ayrılır. Bunlar evrim tabanlı (EB), fizik tabanlı (PB) ve sürü tabanlı (SB) algoritmalardır. Evrim tabanlı algoritmalar seleksiyon, çaprazlama, mutasyon, kemotaksis ve göç gibi doğal evrimi taklit eder. Evrim tabanlı algoritmaların aksine, sürü tabanlı algoritmalar genetik operatörleri benimsemez. Araştırılan sorunlara daha iyi çözümler sunmak için her zaman akıllı türlerin toplu davranışlarını teşvik ederler. Fizik tabanlı algoritmalar, evrim tabanlı ve sürü tabanlı algoritmalarından farklı olarak, doğadaki fiziksel yasalardan motive olurlar (Zhao, Wang, ve Zhang, 2020, s.9383).

Tüm meta-sezgisel optimizasyon algoritmaları, aşağıdaki özellikleri paylaşırlar:

1. Bazı temel teorilere ve matematiksel modellere dayanırlar.
2. Basittirler ve uygulanması kolaydır.
3. Var olan meta-sezgisellere dayalı olarak varyantlarının geliştirilmesi kolaydır.
4. Bir dizi girdi verildiğinde bir dizi çıktının kolayca elde edilebildiği kara kutular olarak görülebilirler.
5. Çeşitli optimizasyon problemleriyle baş etmede çok yönlü ve esnektirler.

Bununla birlikte, mevcut birçok algoritmaya rağmen neden hala yeni optimizasyon yöntemlerinin geliştirildiği sorusu sorulabilir. Bunun cevabı, belirli

gerçek dünya sorunlarının üstesinden gelmek için sürüden ilham alan yeni ve etkili optimize ediciler geliştirmek, hızlandırmak, kolaylaştırmaktır. Ek olarak, çoğu optimizasyon algoritmasının birkaç kontrol parametresi vardır. Belirli bir algoritma için, algoritmanın kullanılabilirliğini büyük ölçüde sınırlayan, farklı problemlere iyi uyan bir dizi farklı parametre bulmak zaman alıcıdır ve zordur. Bu nedenle, daha az parametreli yeni bir optimize edici geliştirmek, yapılan çalışmaların bir diğer motive edici tarafıdır.

Yapay ekosistem tabanlı optimizasyon (AEO), canlı organizmaların üretim, tüketim ve ayrışma davranışlarını taklit etmektedir. Yapılan bir çalışmada AEO, 31 matematik problemi ve 8 gerçek dünya mühendislik problemi üzerinde değerlendirmektedir. Karşılaştırmalı test, önerilen yaklaşımın bu popüler meta-sezgisel yöntemlerden daha üstün olduğunu ortaya koymaktadır. Sonuç olarak, AEO'nun hidrojeolojik parametrelerin tanımlanması alanındaki uygulamaları, diğer mühendislik alanlarındaki potansiyelini göstermektedir (Zhao, Wang, ve Zhang, 2020, s.9383).

Matematiksel yöntemler bir sistemi kontrol etmek için detaylı bir matematiksel modele ihtiyaç duyar. Matematiksel modeller, bir sistemi tanımlayan denklemlerden oluşmaktadır. Sistemi analiz etmek veya simüle etmek için kullanılmaktadır. Ancak bazen elimizde sistemi tam olarak ifade eden bir matematiksel model bulunmayabilir veya doğrusal olmayan bir model gerekebilir. Böyle durumlarda, gerçek sistemden veya simülasyondan elde edilen verileri kullanarak sistemin matematiksel modelini oluşturmak veya sistem tanımlama (System Identification) olarak adlandırılan bir süreçle sistemi analiz etmek gerekebilir. Sistem tanımlama, sistemin dinamik modellerini, sistem verilerinden elde etmeye çalışmaktadır. Kontrol teorisi; istatistik, analiz ve sinyal işleme alanları gibi pek çok uygulamada kullanılmaktadır. Tam bir denklem yerine elimizde sistemden elde edilen giriş ve çıkış değerleri vardır. Bu giriş-çıkış verileri kullanılarak sistem için matematiksel model yerine geçebilecek bir yapı oluşturulmaya çalışılmaktadır. Sistem tanımlama için günümüze kadar kontrol sistemlerinde kullanılması için çeşitli yöntemler geliştirilmiştir. Yapılan literatür taramalarında 80'den fazla meta-sezgisel algoritmanın olduğu belirlenmiştir. Ancak önerilen meta-sezgisel algoritmaların hybrid veya geliştirilmiş versiyonları dikkate alındığında çok daha fazla algoritmanın olduğu söylenebilir. Meta-sezgisel algoritmaları

evrimsel, sürü, fizik, insan, biyolojik, sistem ve matematik tabanlı olarak sınıflandırmak mümkündür (Zaloğlu, Fidan ve Erkan, 2023, s.510) .

Tez özet olarak;

1. AEO algoritması, orta düzey bir bilgisayar kullanılarak çözüm karakteristik değerleri ve performansları incelenerek, diğer 9 farklı algoritma ile kıyaslanmıştır.
2. Temel AEO algoritması ve diğer algoritmalar ilk defa bir sistemi tanımlamak ve sürekli zaman transfer fonksiyonları üretmek için kullanılmıştır.
3. Önerilen meta-sezgisel algoritmalar içinde AEO algoritmasının daha iyi performans sunduğu görülmüştür.
4. LCO ve WCA algoritmalarıda aykırı değerlere rağmen %98 başarılı performans gösterdiği söylenebilmektedir.
5. Performansı incelemek için hair dryer deney seti verileri kullanılmıştır.
6. Python programlama dili kullanılmıştır.
7. Erken durdurma, maksimum jenerasyon, zaman, maksimum fonksiyon hesaplama sınırlıklarına ait kıyaslamaları eklenmiştir.
8. Ele alınan sistem için parametre değişimleri ve güvenilirlikleri incelenmiştir.

Tezin İçeriğinde;

Bölüm 1`de, matematiksel modelleme ile ilgili sorunları çözümlemek için bu tezde ele alınan meta-sezgisel optimizasyon yöntemleri ile ilgili literatür bilgisi verilmiştir. Bölüm 2`de, seçilen 10 adet meta-sezgisel algoritmaya ilişkin bilgilere kısaca yer verilmiştir. Bölüm 3`te, Yapay ekosistem algoritması ve diğer 9 algoritmanın özellikleri ve matematiksel denklemlerle ilgili olarak bilgiler sunulmuştur. Bölüm 4`te, kullanılan algoritmalarla ilgili olarak farklı sınırlamalar altında performanslar ve analizler yer almaktadır. Transfer fonksiyonları çıkarılarak grafik ve tablolarla ilgili araştırma sonuçlarına yer verilmiştir. Bölüm 5`te, sonuç ve önerilere yer verilmiştir.

## 2. KAYNAK ARAŞTIRMASI

Bu bölümde, Yapay Ekosistem Tabanlı Optimizasyon Algoritması (AEO) ile diğer dokuz algoritmadan esinlenilerek ortaya konan yayınlarla ilgili literatür araştırmaları sunulmaktadır.

### 2.1. Yapay Ekosistem Optimizasyonu (AEO – Artificial Ecosystem Optimizer)

El-Dabah ve ark., güneş pilleri ve fotovoltaik modüllerin üçlü diyot modelinin bilinmeyen parametrelerini belirlemek için modern bir optimizasyon algoritması önermektedir. Bunun için, deneysel verilere uymayı ve genel bir PV modeli geliştirmeyi amaçlayan, modelin dokuz bilinmeyen parametresini belirlemek için Yapay Ekosistem Tabanlı Optimize Edici (AEO) adlı bir meta-sezgisel optimizasyon algoritması kullanmaktadır. Yapılan çalışmada AEO, üç farklı ticari PV hücresine/modülüne uygulanmış ve performansı, literatürdeki diğer köklü optimizasyon algoritmalarıyla karşılaştırılmıştır. Sonuçlar, AEO'nun çoklu PV modellerini tanımlamak için yüksek hassasiyet ve hızlı yanıt elde ettiğini göstermektedir. (El-Dabah ve ark., 2021, s.13043)

Omotoso ve ark., yenilenebilir enerjiyle akıllı mikro-şebekeler, uzak kırsal bölgelerin elektrifikasyonunu sağlamak için optimal boyutlandırma problemini çözmek için AEO tabanlı meta-sezgisel algoritması önermişlerdir. Bu çalışmada, AEO algoritması Harris Hawk Optimizasyonu (HHO) ve Gelecek Arama Algoritması (FSA) ile karşılaştırılmış ve elde edilen sonuçlar AEO'nun en düşük enerji maliyetine (COE) sahip, en yüksek tutarlılık seviyesine HHO ve FSA'ya kıyasla minimal standart sapmaya sahip Hibrit Enerji Sisteminin en uygun boyutlandırmasını sağlamak için etkili bir algoritma olduğunu göstermiştir (Omotoso ve ark., 2022, s.204).

Dağıtım ağının çalışmasında güç kaybı azaltma üzerine çalışmaları olan Nguyen bu sorunun çözümünün önemli olduğunu belirtmektedir. Bu amaçla, yapay ekosistem tabanlı optimizasyon (AEO) yöntemine dayanan bir ağ yeniden yapılandırma (NR-Network Reconfiguration) yöntemi sunulmaktadır. NR problemlerinin çözümü güç kaybını azaltmaktadır. Önerilen AEO yönteminin performansı, 14 düğümlü, 69 düğümlü ve 136 düğümlü dağıtım ağları üzerinde denenmiştir. Ek olarak, AEO yöntemine dayalı NR yaklaşımı, karşılaştırmak için guguk kuşu arama algoritması (CSA) kullanılarak uygulanmıştır. İstatistiksel sonuç karşılaştırmaları, AEO

yönteminin, CSA yöntemine göre daha yüksek başarı oranına sahip olduğu, optimal ağ yapılandırmasının kalitesi ile performansı açısından daha iyi olduğunu göstermektedir. Ayrıca, AEO, önceki literatürdeki bazı yaklaşımlardan daha iyi bir ağ yapılandırması elde etme açısından da daha iyi sonuç vermektedir. Bu nedenle, AEO yöntemi, NR problemi için çok etkili bir yaklaşımdır ( Nguyen, 2021, s.14729).

Zhao ve ark., AEO'nun üstün yakınsama oranları ve hesaplama verimliliği sunarak gerçek dünya mühendislik problemleri için etkili olduğunu göstermiştir. Çalışma ayrıca, AEO'nun hidrojeolojik parametre tanımlamasına uygulanmasını ele alarak, bilinmeyen arama alanıyla ilgili zorlu sorunların üstesinden gelmedeki etkinliğini daha da öne çıkarmaktadır (Zhao, Wang, ve Zhang, 2020, s.9383).

Davut İzci ve ark., bir DC-DC düşürücü tip dönüştürücünün çıkış voltajını düzenlemek için en uygun PID denetleyiciyi tasarlamak için AEONM adlı yeni bir hibrit optimizasyon algoritması sunmaktadır. AEONM algoritması, PID denetleyici parametrelerini verimli bir şekilde ayarlamak için yapay ekosistem tabanlı optimizasyon (AEO) algoritmasını tek yönlü Nelder-Mead yöntemiyle birleştirir. Önerilen algoritmanın, dönüştürücünün maksimum verimliliği için PID denetleyicisini etkili bir şekilde optimize edebilmektedir (Izci, Hekimoğlu, ve Ekinci, 2022, s.2030).

### 2.2. Çiçek Tozlaşma Algoritması (FPA - The flower pollination algorithm)

Mergos ve Yang, basitliği ve yüksek hesaplama performansı ile karakterize edilen bir meta-sezgisel optimizasyon algoritması olan çiçek tozlaşma algoritmasının (FPA) parametrelerinin optimizasyonunu araştırmıştır. Gerçek parametreli tek amaçlı optimizasyon problemleri için 28 fonksiyona iteratif olmayan bir ayarlama yöntemi uygulanmıştır. Sonuçlar, optimal FPA parametrelerinin amaç fonksiyonlarına, problem boyutlarına ve hesaplama maliyetine bağlı olarak değiştiğini göstermiştir. Çalışma ayrıca, ortalama tahmin hatalarını en aza indiren FPA parametrelerinin her zaman en sağlam tahminleri sağlamayabileceğini göstermiştir (Mergos ve Yang, 2021, s.14429).

Mohamed ve ark., yaptığı çalışmada biyolojik ilhamı, uygulaması, varyantları, hibritleri ve uygulamaları dahil olmak üzere Çiçek Tozlaşma Algoritmasının (FPA) kapsamlı bir incelemesini sunmaktadır. Çalışma ayrıca kısıtlı bir mühendislik optimizasyon problemini çözmede FPA'yı diğer altı meta-sezgisel algoritma ile

karşılaştırır ve sonuçlar FPA'nın rakiplerine kıyasla üstün olduğunu gösterir (Mohamed Abdel-Basset ve Shawky, 2018, s.2533).

Rodrigues ve ark., çalışmasında parametre ayarlarını sürekli olarak değiştirebilen ve en iyi çözümleri izleyebilen, Flower Pollination Algorithm'ın geliştirilmiş bir uyarlanabilir versiyonunu sunmuştur. Önerilen yaklaşım, Yarasa Algoritması, Parçacık Sürü Optimizasyonu ve çiçek tozlaşma algoritmasının öncül versiyonu ile karşılaştırılmıştır. Deneysel sonuçlar, literatürde bulunan dokuz benchmark fonksiyonunda yapılmıştır ve diğer tekniklerden daha hızlı bir çözüm yeteneğine sahip olduğunu göstermiştir (Rodrigues ve ark., 2020, s.1).

### 2.3. Karınca Aslanı Algoritması (ALO - Ant Lion Optimizer)

Fotovoltaik modülün düzensiz güneş ışığı seviyesi altındaki maksimum güç noktası takibine odaklanan Engel ve ark. ALO algoritması ile ilgili problemin çözümüne odaklanmıştır. Çalışmasında ALO algoritmasını kısmen gölgeli bir fotovoltaik modülün küresel maksimum güç noktasını bulmak için kullanılır. Simülasyon sonuçları, ALO algoritmasının, perturbasyon ve gözlem algoritmasından daha iyi performans gösterdiği anlaşılmaktadır (Engel ve Kovalev, 2016, s.451). GUO ve ark., sarmal bir karmaşık yol arama modeline dayalı geliştirilmiş bir Karınca aslanı algoritması önermektedir. Algoritma, karınca aslanlarının doğadaki avlanma mekanizmasını simüle eder ve sekiz farklı sarmal yol arama stratejisi kullanarak keşif ve sömürü arasında denge kurmaktadır. Geliştirilen algoritma, simülasyon deneyleri ile test edilmiş ve fonksiyon optimizasyonunda, kısıtlamalı mühendislik problemlerinde ve çok amaçlı fonksiyon optimizasyon problemlerinde hızlı yakınsama ve yüksek hassasiyetle üstün performansa sahip olduğunu kanıtlamıştır (GUO ve ark., 2020, s.22094).

Liu ve ark., mühendislik optimizasyon sorunları ve yapay zeka uygulamaları için ALO algoritmasını evrimsel algoritmalar ile birlikte kullanan hibrit bir yöntem önermiştir. Karınca aslanı optimizasyonunu, çok amaçlı optimizasyon alanına genişletmek için Pareto arşivini nasıl güncelleyeceği ve arşivdeki elit ve karınca aslanlarının nasıl seçeceği sorunlarının çözülmesi gerekir. Pareto egemenliği ve bireylerin mesafe bilgilerinin birleştirilmesiyle oluşan yeni bir ölçüm sunulmuş ve ilk

sorunu çözmek için kullanılmıştır. Zaman ağırlığı kavramı, ikinci sorunu ele almak için geliştirilmiştir. Ayrıca, arşivin ortasındaki çözümlere mutasyon işlemi uygulanmıştır ve performans daha da iyileştirilmiştir. Yeni metodun performansı, 11 fonksiyon, diğer dört algoritma ve dört gösterge ile değerlendirilmiş ve daha iyi performans ve daha düşük zaman karmaşıklığına sahip olduğu gösterilmiştir (Liu ve ark., 2021, s.901).

### 2.4. Güve-Alev Optimizasyon Algoritması (MFO - Moth-Flame Optimizasyon)

MFO, yapay ışık kaynakları etrafında güvelerin uçuşundan ilham alan bir meta-sezgisel optimizasyon algoritmasıdır. MFO algoritması, arama aracısı olarak güvelerle arama uzayını araştırır ve keşif tamamlanana kadar her adımda en iyi çözümü günceller. Güve-Alev Optimizasyon Algoritması, optimizasyon problemlerini çözmek için yaygın olarak kullanılır, ancak alevlerin erken toplanması nedeniyle erken olgunlaşma sorunu vardır. Jiang ve ark., keşif davranışını geliştirmek için "alev füzyon mekanizmasını" entegre ederek algoritmayı geliştirmiştir. Mekanizma, dağıtıma dayalı olarak alev toplama durumunu değerlendirir, konsantrasyon yüksek olduğunda daha iyi alevler kaynaştırılır ve keşif ile kullanımı dengelemek için yineleme sırasında füzyon hızını değiştirir. 10 kıyaslama işlevinden elde edilen sonuçlar, diğer algoritmalara kıyasla gelişmiş optimizasyon yeteneği ve daha yüksek kararlılık gösterdiği belirtilmiştir (Jiang ve ark., 2021, s.1). MFO Algoritması: güç ve enerji sistemleri, ekonomik dağıtım, mühendislik tasarımı, görüntü işleme ve tıp gibi alanlarda farklı optimizasyon problemlerine başarıyla uygulanmış ve umut verici meta-sezgisel algoritmalardan biridir. Yapılan çalışmada MFO'nun mevcut literatürünü, varyantlarını ve hibrit hali, MFO yayınlarının büyümesini, MFO uygulama alanlarını, teorik analizlerini ve diğer algoritmalarla karşılaştırmalarını içermektedir. Sonuçlar, MFO üzerine yapılan şu anki çalışmaları odak alır, zayıflıklarını vurgular ve muhtemel gelecek araştırma yönlerini önerir (Shehab ve ark., 2020, s.9859).

Ethan ve ark., hiperspektral görüntüleme (HSI) bant seçimi için Güve alevi optimizasyonu (Moth-Flame Optimization – MFO) kullanmayı önermektedir. Çalışmada, Indian Pines HSI veri kümesi için MFO tabanlı bant seçimine ilişkin bir pilot çalışma sunulmakta ve sonuçları Particle Swarm Optimization (PSO) ve Genetic Algorithm (GA) ile karşılaştırmaktadır. Sonuçlar, MFO'nun PSO ve GA ile

karşılaştırıldığında HSI bant seçimi için umut verici bir strateji olduğunu göstermektedir ( Worch ve ark., 2020, s.1271).

### 2.5. Halat Çekme Optimizasyonu (TWO - Tug of War Optimization)

TWO algoritmasında her aday çözüm için bir ip çekme yarışmasında bir takım olarak kabul edilir ve çözüm kalitelerine göre takımların yeni konumlarını belirlemek için Newton mekanik yasalarını kullanmaktadır. Kaveh ve ark., açıklıkları ve kirişleri olan mazgallı kirişlerin tasarımını optimize etmek için Halat Çekme Optimizasyonu algoritmasını uygulamaktadır. Çalışmada, altıgen açıklı ve dairesel açıklı olmak üzere iki tip mazgallı kiriş ele alınmıştır. Bu fonksiyondaki amaç minimum maliyetle literatürdeki algoritmalarla kıyaslama yaparak problemleri çözmek için kullanılır (A. Kaveh ve Shokohi, 2016, s.533). Algoritmayı ilk defa öneren Kaveh ve Zolghadr, iki süreksiz, çok modlu, düzgün olmayan ve dışbükey olmayan fonksiyonları optimize etmek için kullanmıştır. Yapılan çalışmada, matematiksel fonksiyonları kıyaslanarak, mühendislik tasarım problemlerinde etkinliği gösterilmiştir (Kaveh ve Zolghadr, 2016).

Ghelichi ve ark., sivil yapılar için Halat Çekme optimizasyon yöntemine dayalı yeni bir aktif kontrol algoritması sunmaktadır. Önerilen yöntem, hidrolik aktüatörler ve doğrusal olmayan elemanlar içeren doğrusal olmayan bir karayolu köprüsünün sonlu eleman modeli üzerinde test edilmiştir. Çalışmada stabilite, Lyapunov teorisi kullanılarak sağlanmaktadır. Algoritmanın performansı altı farklı depremde test edilmiştir. Doğrusal olmama durumlarının basit tanımına ve sınırlı yapısal bilgilere rağmen yapının performans indekslerini etkili bir şekilde azaltabildiği gösterilmiştir (Ghelichi ve ark., 2021, s.333).

### 2.6. Atom Arama Optimizasyonu (ASO - Atom Search Optimization)

Atom Arama Optimizasyonu (ASO) algoritması, doğadaki atomik hareket modelinden ilham alan popülasyon tabanlı bir optimizasyon tekniğidir. ASO, Lennard-Jones ve bağ uzunluğu potansiyellerini kullanarak atom etkileşimlerini modeller ve çok çeşitli optimizasyon problemlerine uygulanabilir. Yapılan bir çalışmada moleküler temelli dinamiklerden ilham alınarak geliştirilmiş olan Atom Arama Optimizasyonu (ASO) algoritmasının karşılaştığı yerel optimuma takılma ve önceden yakınsama sorunlarını çözmek için üç strateji sunmaktadır: Bunlar secant faktörü ile küresel

topoloji, doğrusal olmayan atalet ağırlıkları ve güncelleme öğrenimi olarak tanımlanabilir. Bu stratejiler, her birey için en iyi çözümü sağlamaya, popülasyonun bilgi alışverişini zenginleştirmeye, erken yakınsamayı önlemeye, keşif ve sömürü arasında denge sağlamaya ve yerel optimumlardan çıkmak için daha fazla fırsat sağlamaya yardımcı olur (Bi ve Zhang, 2023, s.10). Sun ve ark., ASO algoritmasının sorunlarını çözmek için algoritmanın geliştirilmiş bir versiyonunu önermiştir. İyileştirmeler arasında keşif ve kullanımı dengelemek için doğrusal olmayan bir atalet ağırlık faktörü, atomlar arasındaki bilgi alışverişini artırmak için bir komşu öğrenme bileşeni ve atomların yerel optimumdan dışarı kurtulmasına yardımcı olmak için mevcut popülasyonun en iyi ve en kötü atomları için bir güncelleme mekanizması yer almaktadır (Sun ve ark., 2021, s.837). Zhao ve ark., ASO'nun Parçacık Sürüsü Optimizasyonu (PSO), Genetik Algoritma (GA) ve Bakteriyel Toplayıcı Optimizasyondan (BFO) daha iyi performans gösterdiğini ve rakipleriyle rekabet ettiğini belirtmiştir. ASO algoritmasının uygulanması basittir ve kaynak kodu indirilebilir (Zhao, Wang ve Zhang, 2019, s.601)

### 2.7. Beyin Fırtınası Optimizasyonu (BSO – Brain Storm Optimization)

Beyin Fırtınası Optimizasyonu (BSO) algoritması, insan beyin fırtınası konferansını taklit ederek çok amaçlı optimizasyon problemlerini çözmek için geliştirilmiştir. Narmantha ve ark., yaptıkları çalışmada beyin tümörlerini, özellikle yüksek hastalık ve ölüm oranlarına sahip birincil bir intrakraniyal tümör olan gliomu tartışmaktadır. Beyin tümörü segmentasyonu ve sınıflandırması süreci zorludur ve sürü zekası teknikleriyle daha verimli bir şekilde çözülebilir. Yazarlar, tıbbi görüntü segmentasyonu ve sınıflandırması için FBSO (Fuzzy BSO) adı verilen bulanık ve beyin fırtınası optimizasyon tekniklerinin bir kombinasyonunu önermektedir. Narmatha ve ark., yaptığı çalışmada BRATs 2018 veri kümesi kullanıldı ve önerilen FBSO (Bulanık beyin fırtınası optimizasyonu) yöntemi, yüksek verimlilik ve sağlamlık sergiledi. Optimizasyon algoritmasının segmentasyon süresini önemli ölçüde azaltırken, %93.85 doğruluk, %94.77 hassasiyet, %95.77 duyarlılık ve %95.42 F1 puanı (hassasiyet ve duyarlılığın ağırlıklı ortalaması) ile sonuçlandı (Narmatha ve ark., 2020, s.7).

Tuba ve ark., önerdiği beyin fırtınası optimizasyon yöntemini termal ve görsel görüntüler üzerinde test etmiştir. Çalışmada elde edilen sonuçlar karşılaştırılarak,

önerilen beyin fırtınası optimizasyon yönteminin, parçacık sürüsü optimizasyon yöntemi gibi diğer optimizasyon yöntemlerinin yanı sıra gradyan piramidi, Laplacian piramidi, Laplacian piramit oranı ve kaydırmalı değişmeyen ayrık dalgacık dönüşümü gibi çeşitli standart görüntü füzyon yöntemlerini de geride bıraktığı sonucuna varılmıştır. Karşılaştırma, entropi, QABF, LABF ve NABF dahil olmak üzere dört kalite ölçütü açısından değerlendirilmiştir (Tuba ve ark., 2019, s.234). Cheng ve ark.,, yüzlerce bireyin simüle edilmesi gereken bir bilgi yayılma probleminde birden fazla etmen arasındaki işbirliği stratejilerini simüle etmek için evrimsel oyun teorisi ve beyin fırtınası optimizasyonu (BSO) algoritmasının kullanımını tartışmaktadır. Bunun için değiştirilmiş BSO algoritması kullanılmış ve farklı sürü optimizasyon algoritmalarının özellikleri araştırılmıştır (Cheng ve ark., 2021, s.5)

WU ve ark., karar değişkeni kümeleme yöntemini kullanarak değişkenleri yakınsama ve çeşitlilik ile ilgili değişkenlere bölen BSO algoritmasının yeni bir versiyonunu önermiştir. Algoritma, yakınsama ile ilgili değişkenler ve seçim baskısını artırmak için bir ayrıştırma stratejisi kullanırken; popülasyonu güncellemek, çeşitliliği artırmak ve çeşitlilikle ilgili değişkenler için bir referans noktası stratejisi kullanmaktadır. Deneysel sonuçlar, yeni algoritmanın çok amaçlı optimizasyon problemlerini çözmek için umut verici bir yaklaşım olduğunu göstermektedir (Wu ve ark., 2019, s.186572).

### 2.8. Su Döngüsü Algoritması (WCA - Water Cycle Algorithm)

Heidari ve ark., doğal çevrede gözlemlenen idealize hidrolojik döngüden esinlenerek geliştirilen bir nüfus tabanlı meta-sezgisel teknik olan Su Döngüsü Algoritmasını (WCA) tanıtmışlardır. WCA, kısıtlanmış ve kısıtsız problemleri çözmek için diğer iyi tanınmış tekniklerle karşılaştırıldığında daha iyi bir performans gösterebilmektedir. Ancak, diğer meta-sezgisel tekniklerde olduğu gibi, bazı optimizasyon görevleriyle uğraşırken yerel optimuma erken yakalanma sorunu hala oluşabilmektedir. Bu makalede, gerçek su döngüsü davranışındaki kaos gibi, WCA'nın stokastik işlemlerine kaotik desenler dahil edilerek, geleneksel algoritmanın performansını geliştirmek ve erkenden lokal minimuma takılma sorununu azaltılması amaçlanmaktadır. Farklı kaotik sinyal fonksiyonları ve çeşitli kaotik güçlendirilmiş WCA stratejileri (toplam 39 meta-sezgisel) uygulanmış ve en iyi sonuç, WCA'nın

modifikasyonu içeren bir yöntemle elde edilmiştir. Ardından, kaotik algoritma, çeşitli örnek problemlerle ve ayrıca sinir ağlarının eğitimiyle başa çıkmak için kullanılmıştır. Yeni tekniklerin karşılaştırmalı istatistiksel sonuçları, erken yakalanma sorununun önemli ölçüde giderildiğini göstermektedir (Heidari, Abbaspour ve Jordehi, 2017, s.57).

Su çevrimi algoritmasını (WCA) ele alan diğer bir çalışmada Zhou ve ark., bu algoritmanın bilimsel hesaplama ve mühendislik optimizasyonunda yaygın bir şekilde kullanıldığını belirtmiştir. WCA algoritmasının keşif ve sömürü yeteneklerini artırmak, yakınsama hızını hızlandırmak ve hesaplama doğruluğunu artırmak isteyen yazarlar kompleks değerli kodlama WCA (CWCA) adlı yeni bir algoritma önermiştir. CWCA'nın performansını değerlendirmek için 12 benchmark fonksiyonu ve dört mühendislik örneği kullanılmıştır. Deneysel sonuçlar, CWCA'nın gerçek değerli WCA ve diğer iyi bilinen meta-sezgisel algoritmalarından daha yüksek hassasiyet ve yakınsama hızına sahip olduğunu göstermiştir (Zhou ve ark., 2021, s.5836).

Roeva ve ark., nehir hareketlerinden ilham alan bir meta-sezgisel olan Su Döngüsü Algoritmasını (WCA), fermantasyon süreci modellerinde zorlu parametre tanımlama işlemine ilk kez uygulamaktadır. Çalışma, WCA'yı Genetik Algoritma (GA) ile karşılaştırır ve bu gerçek dünya görevinde her bir algoritmanın güçlü ve zayıf yönlerini belirlemek için sonuçları analiz etmektedirler. Çalışma, temsili modeller olarak bakteri ve mayayı kullanır. Yeni uygulanan WCA'nın model doğruluğu açısından GA'dan daha iyi performans gösterdiğini ve model tanımlama sorunlarını gidermek için potansiyelini ortaya koyduğu belirtilmektedir (Roeva ve ark., 2020, s.920).

### 2.9. Mercan Resifleri Optimizasyonu (CRO - Coral Reefs Optimization )

CRO algoritmasını temel alan bir çalışmada Xu ve ark., mercan resifleri algoritmasına dayalı bulut bilgi işlem için yeni bir görev planlama yöntemi sunmaktadır. Çizelgeleme modeli, yük dengeleme, kaynak kullanımı ve kararlılığı ölçmek için önerilen bir amaç fonksiyonu ile resmi olarak tanımlanır. Mercan resifi temsil yöntemi ve polip kodlama şeması, varyasyonu geliştirmek için bir matris rastgele eşleme ile tasarlanmıştır. Yapılan çalışmada mercan resifi algoritması, Karınca Kolonisi Optimizasyonu, Genetik Algoritma ve Round Robin algoritmaları ile karşılaştırılmış ve tamamlama süresini azalttığı ve kaynak kullanımını artırdığı belirlenmiştir (Chen,

2019, s.27). Diğer bir çalışmada aracı tabanlı modellerin otomatik kalibrasyonu için mercan resifleri optimizasyon algoritmalarının kullanımı önerilmiş ve performansı diğer evrimsel algoritmalara ve bunların melezlerine karşı değerlendirilmiştir. Sonuçlar, memetik mercan resifleri optimizasyonunun, artan sayıda karar değişkeni ile 12 problem örneğinin bir kıyaslamasına dayalı olarak kalibrasyon doğruluğu açısından mükemmel performansa sahip olduğunu göstermektedir (Moya ve ark., 2021, s.104170).

Salcedo ve ark., elektromanyetik kirliliğin kontrolünde önemli bir rol oynayan Mobil Ağ Dağıtım Problemi' nin (MNDP) çözümü için yeni bir meta-sezgisel yaklaşım olan Mercan Resifleri Optimizasyonu (CRO) algoritması uygulanmıştır. Bu çalışmada iki şey ele alınmıştır. İlk olarak, CRO yaklaşımının gerçek ve zor bir optimizasyon probleminde performansı incelemekte ve ikinci olarakta, telekomünikasyon alanında önemli bir problemi çözmektedir (Salcedo-Sanz ve ark., 2014, s.239).

## 2.10. Yaşam Seçimi Tabanlı Optimizasyon (LCO – Life Choice Based Optimization)

LCO, insanların hedeflerine ulaşmak için diğer üyelerin öğrenme yeteneğine dayanan meta-sezgisel algoritmalardan biridir. Ahammed ve ark., optimizasyon tabanlı bir destek vektör makinesi (SVM) kullanarak otomatik bir görsel duygu analizi (VSA) modeli gerçekleştirmiştir. Analizler, performans parametreleri (Doğruluk, Hassasiyet ve Özgünlük gibi) temel alınarak Emotion-6 ve Abstract Art_photo veri kümeleri üzerinde yapılmış olup önerilen modelin doğruluğu, Emotion-6 veri kümesinde %70,7 ve Art_photo veri kümesinde %76,8 olarak ölçülmüştür (Afzal, 2021, s.171). Diğer bir çalışmada LCO, altı CEC-2005 fonksiyonunu içeren 29 popüler benchmark fonksiyonu üzerinde incelenmiş ve performansı, son zamanlarda geliştirilen yedi optimizasyon tekniği ile karşılaştırılmıştır (Khatri, ve ark., 2019, s.9121). LCO algoritması akıllı telefonların kameralarını kullanalarak Diyabetik Retinopati (DR) için otomatik bir tespit sisteminin geliştirilmesinde kullanılmıştır. Önerilen şema, APTOS-2019-Blindness-Detection ve EyePacs gibi veri kümeleri üzerinde test edilmiştir ve doğruluk açısından diğer mevcut yaklaşımlardan daha iyi performans göstermektedir. Sistem, düşük maliyetli ve taşınabilir retinal görüntüleme sistemleri kullanarak DR'nin erken tespitini iyileştirmeyi amaçlamaktadır (S. Gupta , Thakur ve A. Gupta, 2022, s.14475).

# 3. MATERYAL VE YÖNTEM

Bu bölüm boyunca çeşitli meta-sezgisel algoritmalara ve sistem tanımlama kavramı ile ilgili açıklamalara yer verilmiştir. Ayrıca analiz ile ilgili verilere ulaşmak için kullanılan donanım ve yazılım ile ilgili teknik bilgilere değinilmektedir.

## 3.1. Meta-sezgisel Algoritmalar

Meta-sezgisel algoritmalar, karmaşık optimizasyon problemlerini çözmek için kullanılan sezgisel yöntemlerdir. Bunlar, genellikle matematiksel model gerektirmeyen ve doğal olaylardan esinlenen arama stratejileridir. Yaygın olarak kullanılan meta-sezgisel algoritmalar arasında genetik algoritma, karınca kolonisi optimizasyonu ve parçacık sürü optimizasyonu algoritmaları bulunur. Ancak bu konuda ön planda olan ve daha az bilinen farklı algoritmalarda bulunmaktadır. Bölüm-3' te bu tez çalışmasının ana konusu olan algoritmalardan bahsedilmiştir.

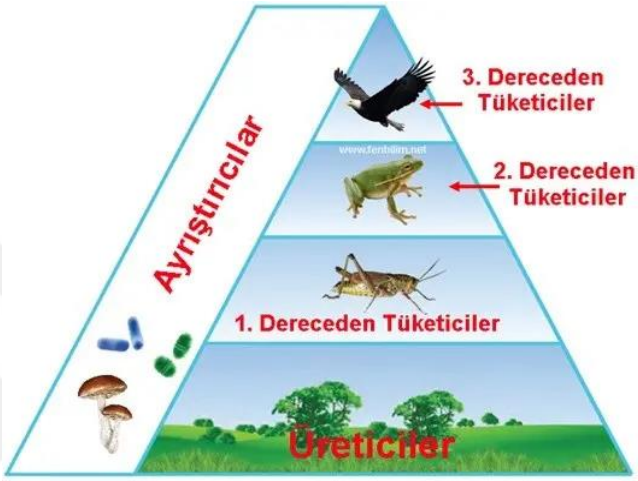
### 3.1.1.Yapay Ekosistem Tabanlı Optimizasyon (AEO)

Yapay ekosistem tabanlı optimizasyon (AEO) algoritması, doğal ekosistemlerden ilham alarak karmaşık optimizasyon problemlerini çözmek için kullanılan bir yöntemdir. AEO, dünyadaki bir ekosistemdeki enerji akışından motive edilen popülasyona dayalı bir optimize edicidir ve bu algoritma, üretim, tüketim ve ayrışma dahil olmak üzere canlı organizmaların üç benzersiz davranışını taklit etmektedir (Zhao, Wang, ve Zhang, 2020, s.9383).

#### 3.1.1.1. Esin Kaynağı

Çoğu ekolojist için sezgisel olarak çekici bir kavram olan 'Ekosistem' ilk olarak 1955'te AG Tansley tarafından tanıtıldı. Bu çekicilik sayesinde ekosistem kavramı giderek daha önemli ve popüler hale geldi (Zhao, Wang, ve Zhang, 2020, s.9383). Ekosistem, canlı organizmaların, fiziksel çevrelerinin ve belirli bir uzay birimindeki tüm karşılıklı ilişkilerinin birleşimi olarak tanımlanabilir. Bir ekosistem, abiyotik ve biyotik bileşenler olarak sınıflandırılmaktadır. Abiyotik bileşenler güneş ışığı, su, hava ve diğer cansız elementleri içerir. Biyotik bileşenler tüm canlı organizmaları içerir. Enerji akışı ve besin döngüsü, tüm canlı organizmaların önemli bir rol oynadığı bir ekosistemde normal ekolojik düzeni koruyan başlıca itici güçlerdir. Canlı organizmalar

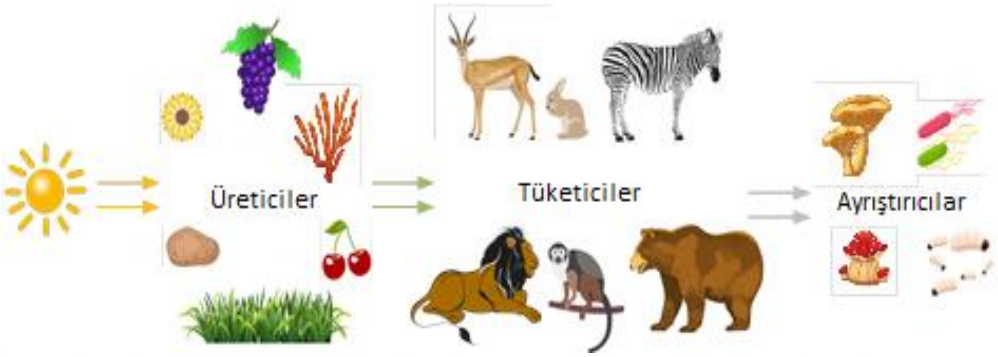
üretici, tüketici ve ayrıştırıcı olarak sınıflandırılmaktadır. Üreticilerin diğer organizmalardan enerji elde etmesine gerek yoktur ve çoğu üretici, besin enerjisini doğrudan fotosentez süreci yoluyla alan herhangi bir yeşil bitki türündedir. Fotosentez sırasında, karbondioksit ve su, güneş ışığının varlığında birlikte reaksiyona girerek oksijen ve glikoz adı verilen bir şeker üretmektedir. Bitkiler bu şekeri; yaprak, meyve, ağaç ve kök gibi birçok şeyi yapmak için kullanmaktadır. Şekil 3.1' de üreticiler, tüketiciler ve ayrıştırıcılara ilişkin ekosistem piramidi gösterilmektedir.




Şekil 3.1. Üreticiler, Tüketiciler ve Ayrıştırıcılar

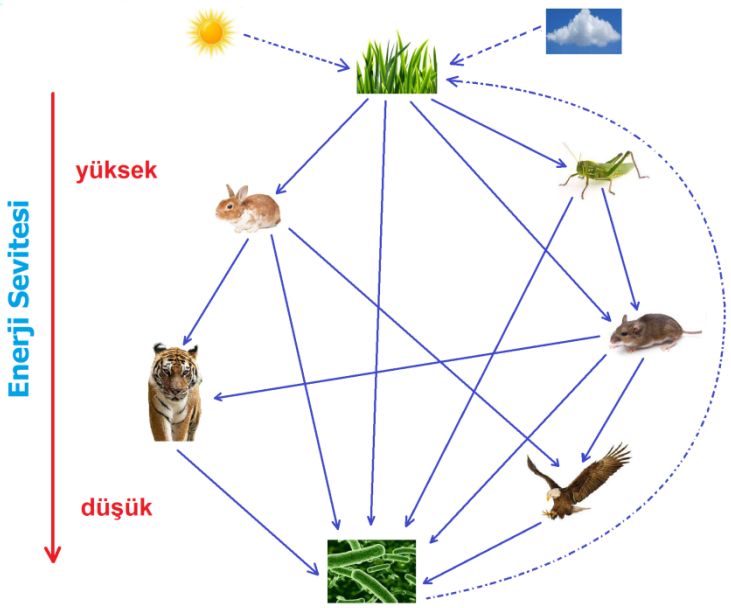
Üreticiler, genellikle otobur ve omnivor tüketiciler için temel gıda sağlamaktadır. Tüketiciler, yiyeceklerini yapamayan hayvanlardır; bu nedenle enerji ve besin elde etmek için üreticilerden veya diğer tüketicilerden beslenmek zorundadırlar. Tüketiciler etobur, otobur ve omnivor olarak sınıflandırılmaktadır. Sadece üreticileri (bitkileri) yiyen hayvanlara otobur denir. Hem üreticileri hem de diğer hayvanları yiyen hayvanlara omnivorlar denir. Sadece diğer hayvanları yiyen hayvanlara etobur denir. Ayrıştırıcılar, hem ölü bitkiler (üreticiler) hem de hayvanlar (tüketiciler) veya canlı organizmaların atıklarıyla beslenen bir organizmadır. Ayrıştırıcılar çoğu, bakteri ve mantarı içerir. Bir organizma öldüğünde, ayrıştırıcılar kalıntıları parçalar ve onları karbondioksit, su ve mineraller gibi basit moleküllere dönüştürmektedir. Daha sonra, bu enerji formları, üreticiler tarafından tekrar fotosentez yoluyla şeker ve oksijen yapmak için absorbe edilmekte ve döngü baştan başlamaktadır. Birbirleriyle etkileşime giren üreticiler, tüketiciler ve ayrıştırıcılar bir besin zinciri oluşturur. Besin zinciri, besinlerin hareketini sağlamak için her canlının besin enerjisi alması gereken bir ekosistemde kimin kimi beslediğini tanımlamaktadır. Besin zinciri, enerji ve besinlerin bir ekosistem

boyunca izleyebileceği olası bir bağlantılı yoludur ve bir ekosistem içindeki farklı yeme düzeylerini göstermektedir. Üreticiler, genellikle yem ağlarının ve çoğu besin zincirinin başında yer almaktadır. Tüketiciler, üç tür organizmanın en karmaşık olanıdır. AEO'ya dayalı bir ekosistemin temsili Şekil 3.2' de gösterilmektedir.




Şekil 3.2. AEO'ya dayalı bir ekosistemin temsili görünümü

Şekil-3.3'te bir ekosistemdeki enerji akışı gösterilmektedir. Mavi oklar, enerji transfer yolunu temsil eder. Kırmızı ok, üreticilerden ayrıştırıcılara azalan farklı enerji seviyelerini temsil eder. Enerji her zaman yüksek enerjili canlılardan düşük enerjili canlılara doğru akar. Tipik olarak, ekosistem, türlerin istikrarını korumak için bir strateji olarak bu enerji aktarım mekanizmasını geliştirmekte ve ekolojik dengeyi uzun vadede koruyabilmektedir. Bu nedenle, bu enerji aktarım mekanizmasının sağlam ve istikrarlı bir ekosistemi şekillendirebileceğine ve sürdürebileceğine inanmak için nedenler vardır.




Şekil 3.3. Bir ekosistemde enerji akışı

### 3.1.1.2. Yapay ekosistem operatörleri

Yapay ekosistem tabanlı optimizasyon algoritması üretim, tüketim ve ayrıştırma dahil olmak üzere üç operatör kullanır. İlk operatör, keşif ve işletme arasındaki dengeyi geliştirir. İkinci operatör, algoritmanın keşfini geliştirmek için kullanılır. Üçüncü operatör ise, algoritmanın sömürülmesini teşvik etmesi için öneri sunar. AEO, bir çözüm aramak için genellikle aşağıdaki birkaç kuralı takip eder.

1. Bir popülasyon sisteminde ekosistem üç tür organizma içerir: üretici, tüketici ve ayrıştırıcı.
2. Bir popülasyonda birey olarak yalnızca bir üretici vardır.
3. Bir popülasyonda birey olarak yalnızca bir ayrıştırıcı vardır.
4. Bir popülasyonun diğer bireyleri, her biri aynı olasılıkla etobur, otobur veya omnivor olarak seçilen tüketicilerdir.
5. Bir popülasyondaki her bireyin enerji seviyesi, fonksiyon uygunluk değeri ile değerlendirilir.

Popülasyon, fonksiyon uygunluk değerinin azalan düzeninde sıralanır, bu nedenle daha yüksek fonksiyon uygunluk değeri, bir minimizasyon problemi için daha yüksek enerji seviyesini gösterir.

### 3.1.1.3. Üretim

Bir ekosistemde üretici, karbondioksit, su ve güneş ışığının yanı sıra ayrıştırıcı tarafından sağlanan beslenme ile gıda enerjisi üretebilmektedir. Aynı şekilde, AEO'da, bir popülasyondaki üreticinin (en düşük seviyedeki birey), arama uzayının alt/üst limitleri ve ayrıştırıcı (en iyi birey) tarafından güncellenmesi gerekir ve bu güncellenen birey, otobur ve omnivor dahil olmak üzere diğer bireylere rehberlik etmektedir. Üretim operatörü, AEO'ya en iyi birey ($x_n$) ve arama uzayında rastgele üretilen bir birey ($x_{rand}$) arasında önceki bireyi değiştirerek yeni bir birey rastgele üretme imkânı sağlar. Üretim operatörünün matematiksel modeli Denklem (3.1 – 3.3)' te temsil edilir:

$$x_1(t+1) = (1-a)x_n(t) + ax_{rand}(t) \tag{3.1}$$

$$a = \left(1 - {}^{t}/_{T}\right) r_1 \tag{3.2}$$

$$x_{rand} = r(U - L) + L \tag{3.3}$$

Burada;

$N$ : Bir populasyonun büyüklüğünü

$T$ : Maksimum iterasyon sayısını

$L, U$ : Alt ve üst limitleri

$r_1$ : *[0, 1]* aralığında rastgele bir sayıyı

$r$ : *[0, 1]* aralığında rastgele bir vektörü

a : Lineer bir ağırlık katsayıyı

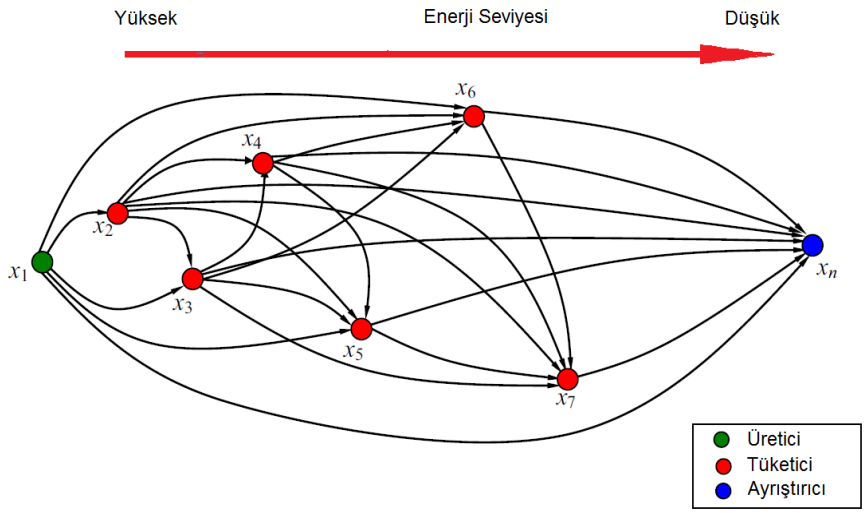
$x_{rand}$ : Bir bireyin konumunu temsil eder.

Denklem (3.1)'de, $a$ ağırlık katsayısı, iterasyonlar ilerledikçe bireyin rastgele üretilen bir konumdan en iyi bireyin konumuna doğru lineer bir şekilde hareket etmesini sağlamak için kullanılır. Denklem (3.1) gösteriyor ki erken iterasyonlarda $x_1(t+1)$, diğer bireylerin arama uzayında geniş bir keşif yapmasını sağlayabilirken, sonraki iterasyonlarda $x_1(t+1)$, diğer bireyleri $x_n$ etrafında yoğun bir şekilde kullanılan bölgesel keşfe yönlendirebilir.

### 3.1.1.4. Tüketim

Üretici, üretim operatörünü tamamladıktan sonra, tüm tüketiciler tüketim operatörünü gerçekleştirebilmektedir. Gıda enerjisini elde etmek için, her tüketici daha düşük enerji düzeyine sahip rastgele seçilmiş bir tüketiciyi ya da bir üreticiyi yiyebilmektedir. Ya da her ikisini de birden yiyebilmektedir. Şekil 3.4'te enerji seviyesindeki değişim görülebilir.

Şekil 3.4'te rastgele ilerlemenin genellikle merkezi bir nokta etrafında kalabalıklaştığı ve bazen bir önceki noktadan uzağa uzun bir adım attığı görülmektedir. AEO'nun yerel aşırılıklardan kaçınma ve tüm arama alanını keşfetme fırsatına sahip bir durumdadır. Bu nedenle, bu tüketim faktörü, her tüketicinin yiyecek aramasına yardımcı olabilirken, farklı tüketici türleri ile farklı tüketim stratejileri benimser. Örnek olarak bir tüketici otobur olarak rastgele seçilirse, yalnızca üreticiyi yer. Tüketim faktörü olarak adlandırılan Levy uçuşu önerilmiştir.




Şekil 3.4. AEO'da bir ekosistem (Zhao, Wang, ve Zhang, 2020, s.9386).

Keşif yeteneğini artırmak için, tüketim faktörü (C) olarak adlandırılan bir Levy uçuşu benzeri, rastgele yürüyüşün matematiksel modellemesi Denklem (3.4)' te ifade edilmiştir:

$$C = \frac{1}{2}\frac{v_1}{|v_2|} \tag{3.4}$$

$$v_1 \sim N(0,1), \qquad v_2 \sim N(0,1) \tag{3.5}$$

Burada;

$v_1$ ve $v_2$, $N(0,1)$ için ortalama değeri 1 ve standart sapması 1 olan normal dağılımdır.

Şekil 3.4'de gösterildiği gibi, hem tüketici $x_2$ hem de $x_5$, sadece üretici $x_1$'i tüketen otçullardır. Otçulların bu tüketim davranışını matematiksel olarak modellemek için aşağıdaki Denklem (3.6)' da ifade edilmiştir:

$$x_i(t+1) = x_i(t) + C\big(x_i(t) - x_1(t)\big), \quad i \in [2, \ldots\ldots, n] \tag{3.6}$$

Etobur: Eğer bir tüketici rastgele olarak bir etçil olarak seçilirse, sadece enerji seviyesi daha yüksek olan bir tüketiciyi rastgele seçerek yiyebilir. Şekil 3.4 'te, tüketici $x_6$ bir etçil olduğundan dolayı, $x_5$ tüketicinden daha yüksek enerji seviyelerine sahip olan $x_2$ ile $x_5$ bireyleri arasından rastgele bir tüketiciyi yiyecek olarak seçmesi

gerekmektedir (Zhao, Wang, ve Zhang, 2020, s.9386). Tüketimi modelleyen matematiksel ifadesi Denklem (3.7) gibidir:

$$x_i(t+1) = x_i(t) + C\left(x_i(t) - x_j(t)\right), \qquad i \in [3, \ldots\ldots, n] \tag{3.7}$$

$$j = randi([2i-1])$$

Omnivore: Bir tüketici omnivor olarak rastgele seçilirse, hem enerji düzeyi yüksek olan tüketiciyi hem de üreticiyi rastgele yiyebilir. Şekil-3.4' te Tüketici $x_7$ bir omnivordur, bu yüzden hem üreticiyi hem de tüketiciyi yemesi gerekir. Rastgele seçilen bir tüketici, $x_2$ ile $x_6$ ya göre daha yüksek enerji seviyelerine sahiptir. Bir omnivorun tüketim davranışını modelleyen matematiksel ifade Denklem (3.8) de gösterilmiştir.

$$x_i(t+1) = x_i(t) + C\left(r_2\left(x_i(t) - x_1(t)\right) + (1-r_2)\left(x_i(t) - x_j(t)\right)\right), \tag{3.8}$$

$$i \in [3, \ldots\ldots, n]$$

$$j = randi([2i-1])$$

Randi [0,1] aralığında rastgele bir sayıdır. Bu tüketim operatörü için, AEO, bir popülasyondaki en kötü bireye ve rastgele seçilen bir bireye veya her ikisine göre arama yapan bireyin konumunu güncellemektedir. Bu davranış, keşfi vurgulama eğiliminde olduğundan AEO' nun küresel bir arama yapmasına izin vermektedir.

#### 3.1.1.5. Ayrışma

Ayrışma, bir ekosistemin işleyişi açısından çok hayati bir süreçtir ve üreticinin büyümesi için gerekli besin maddelerini sağlamaktadır. Ayrışma sırasında, popülasyondaki her birey öldüğünde ayrıştırıcı, kalıntıları kimyasal olarak bozacak veya parçalayacaktır. Bu davranışı matematiksel olarak modellemek için ayrışma faktörü *D* ve ağırlık katsayıları *e* ve *h* tasarlanmıştır. Bu nedenle, bir popülasyondaki *i.* bireyin konumu $x_i$, bu parametreler *D, e* ve *h* aracılığıyla ayrıştırıcı $x_n$' in konumu tarafından güncellenebilir. Bu, her bireyin bir sonraki konumunun (ayrıştırıcı olan en iyi bireyin) etrafında yayılmasına ve kısmen sömürüye işaret etmesine olanak tanır (Zhao, Wang, ve Zhang, 2020, s.9387). Bu ayrışma davranışını ifade eden matematiksel ifadeler Denklem (3.9 - 3.12) de ifade edilmiştir:

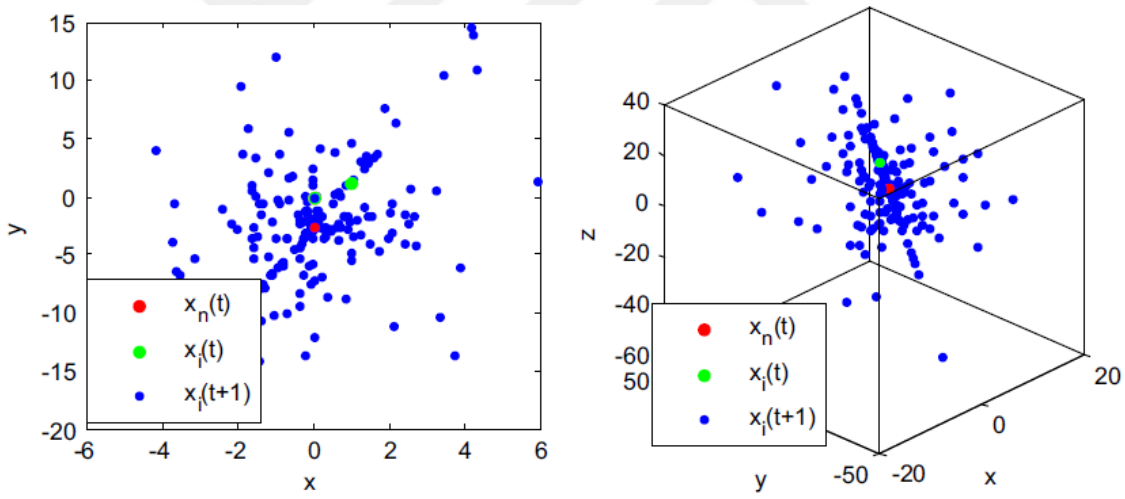
$$x_i(t+1) = x_n(t) + D\left(ex_n(t) - hx_i(t)\right), \quad i = 1, \ldots, n \tag{3.9}$$

$$D = 3u, \quad u \sim N(0,1) \tag{3.10}$$

$$e = r_3 randi([1\ 2]) - 1 \tag{3.11}$$

$$h = 2r_3 - 1 \tag{3.12}$$

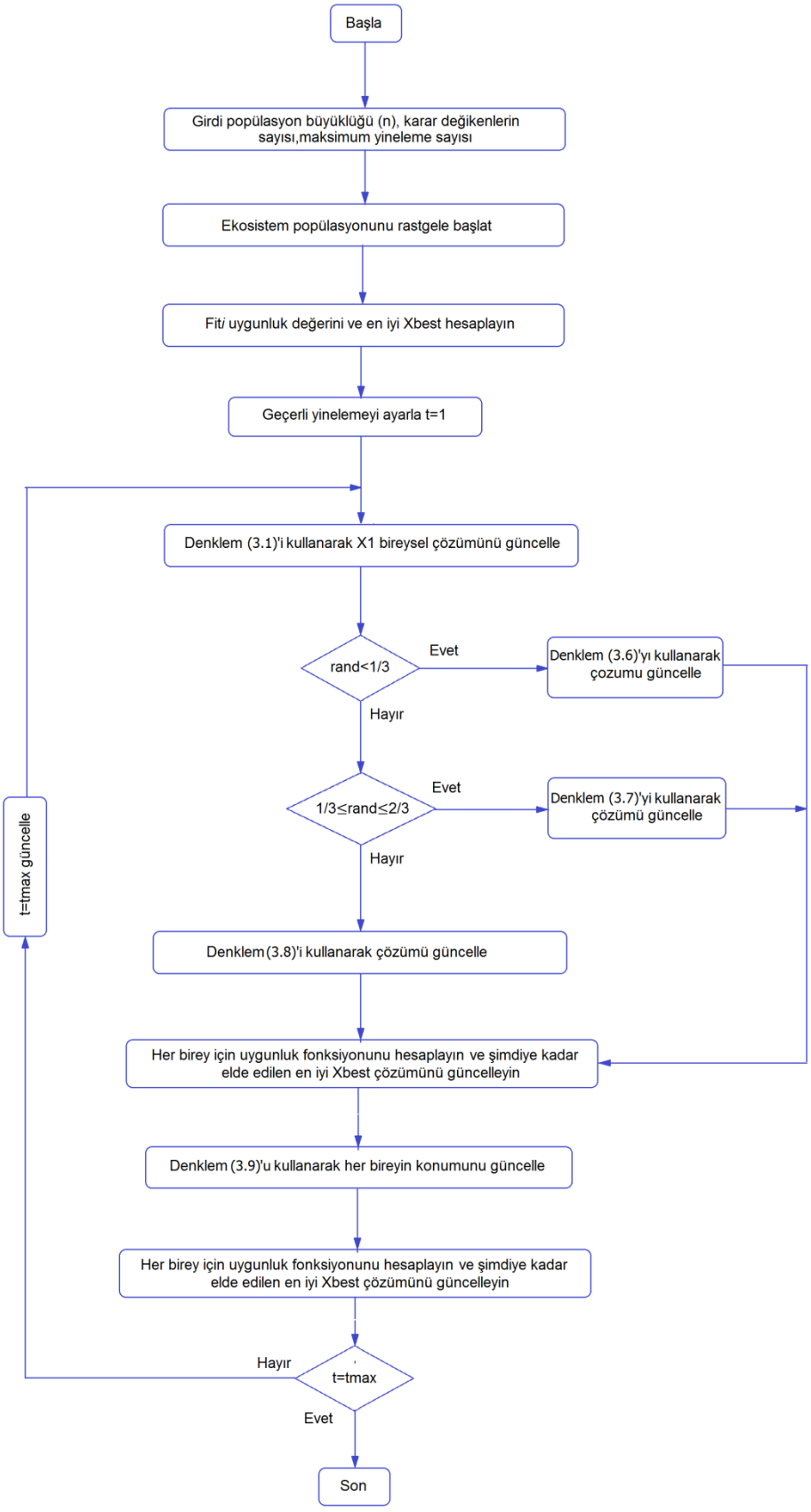
Şekil 3.5'te Denklem (3.2)' ye göre sırasıyla 2 boyutlu ve 3 boyutlu uzayda bireylerin davranışı gösterilmektedir. Örneklenen noktaların çoğu, mevcut birey arasındaki mesafede rastgele dağılmıştır. Algoritmayı kısaca özetlemek gerekirse, ilk başta rastgele bir popülasyon oluşturulur ve her yineleme boyunca ilk bireyin konumu denklem kullanılarak güncellenir. Diğer bireylerin konumları, denklemler arasından seçim yapılarak gerçekleşir. Her birey konumunu denklem kullanarak daha iyi bir fonksiyon değerine sahip bir birey olması durumunda aramada rastgele bir birey oluşturmaktadır. AEO algoritmasının akış diyagramı altta gösterilmiş olup genel çözüm süreci ayrıca Şekil 3.6' da gösterilmektedir.




Şekil 3.5. 2000 yineleme boyunca 2 boyutlu ve 3 boyutlu alanlarda tüketim davranışları (Zhao, Wang, ve Zhang, 2020, s.9388)

***AEO Akış Diyagramı***

1. *Adım* — *Başla*
2. *Adım* — *Popülasyon büyüklüğünü(n), karar değişken sayısını ve maksimum yineleme sayısını (tmax) gir*
3. *Adım* — *Ekosistem popülasyonunu rastgele başlat*
4. *Adım* — *En iyi çözüm için Fiti uygunluk değerini ve $X_{best}$ değerini hesaplayın.*
5. *Adım* — *Geçerli yinelemeyi t=1 olarak ayarla*
6. *Adım* — *Denklem(3.1) i kullanarak X1 bireysel çözümünü güncelle*
7. *Adım* — *rand<1/3 ise Denklem (3.6) yı kullanarak çözümü güncelle*
8. *Adım* — *1/3<rand<2/3 aralığında ise Denklem(3.7) yi kullanarak çözümü güncelle*
9. *Adım* — *1/3<rand<2/3 aralığında değil ise Denklem(3.8) i kullanarak çözümü güncelle*
10. *Adım* — *Her birey için uygunluk fonksiyonunu hesaplayın ve şimdiye kadar elde edilen en iyi $X_{best}$ çözümünü güncelleyin*
11. *Adım* — *Denklem(3.9) kullanarak bireyin konumunu güncelleyin*
12. *Adım* — *Her birey için uygunluk fonksiyonunu hesaplayın ve şimdiye kadar elde edilen en iyi $X_{best}$ çözümünü güncelleyin*
13. *Adım* — *$t = t_{max}$ ise döngü bitsin*
14. *Adım* — *$t = t_{max}$ değil ise eşitlik oluşana kadar $X_{best}$ e dönün*




Şekil 3.6. Temel AEO algoritmasının akış şeması

### 3.1.2. Çiçek Tozlaşma Algoritması (FPA)

Çiçek tozlaşma algoritması (FPA), bitkilerin üreme sürecinden ilham alarak geliştirilmiş bir optimizasyon yöntemidir (Yang, 2012, s.240). Bu yöntem, doğada gözlenen biyotik ve abiyotik tozlaşma şekillerinden ilham alır (Korkmaz ve Akgüngör, 2018, s.41) Biyotik tozlaşmada, sinekler ve arılar gibi canlılar polenleri taşıyarak bitkiler arasında dolaşır ve tozlaşmayı gerçekleştirir. Abiyotik tozlaşmada ise su ve rüzgar gibi doğal faktörler, polenlerin doğal yollarla yayılmasını sağlar ve tozlaşmayı gerçekleştirir. Çiçekli bitkilerde genellikle %90 oranında biyotik tozlaşma ve %10 oranında abiyotik tozlaşma gerçekleşir (Kızılkaplan, Tolga ve Yalçınkaya, 2020, s.587).

Çiçek tozlaşma algoritması (FPA), bu doğal tozlaşma süreçlerinden esinlenerek optimizasyon problemlerini çözmek için kullanılır. FPA'nın 4 temel kuralı şunlardır:

1- Çiçeklerin yerleşimi: Algoritma, çiçeklerin rastgele dağılımını taklit ederek çözüm adaylarını oluşturur.
2- Polinasyon: Çiçeklerin polenleri, en iyi çözüm adaylarına doğru taşınır. Bu, çözümlerin birbirleriyle etkileşimini simgeler.
3- Üreme: Yeni çözüm adayları, polenlerin taşınması ve kombinasyonu yoluyla oluşturulur. Bu, yeni nesillerin üretilmesini sağlar.
4- Adaptasyon: Çözüm adayları, objektif fonksiyon değerlerine göre değerlendirilir ve adaptasyon süreci gerçekleştirilir. En iyi çözümler, gelecek nesillere aktarılırken diğerleri elenir.

Bu kurallar, FPA'nın algoritmasının problemleri çözmek için doğal tozlaşma süreçlerini taklit eder. Bu algoritma, özellikle karmaşık optimizasyon problemlerinde önemli sonuçlar elde etmek için kullanılabilmektedir.

Böcekler uzun süre uçabildikleri için polenler de uzun mesafelerde havada farklı yerlere gidebilmektedir. Bu şekilde bitkiler arasında uzun mesafelere rağmen üreme oluşabilmektedir. Çiçek sabitesinin matematiksel ifadesi Denklem (3.13)‘te gösterilmiştir.

$$x^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t) \quad (3.13)$$

Burada $x^{t+1}$ çözüm vektörüdür, $\gamma$ adım boyutunu ayarlama faktörüdür, $g_*$ mevcut en iyidir. Lévy dağılımı, tozlaşmayı gerçekleştirmek için kullanılan bir dağılım türüdür. Özellikle böceklerin uzun mesafelere taşındıklarında, hareketleri Lévy dağılımını oluşturmaktadır. Lévy dağılımının matematiksel ifadesi Denklem (3.14)'te ifade edilmiştir.

$$L \sim \frac{\lambda \Gamma(\lambda) sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0) \quad (3.14)$$

Burada;

*L* :Levy dağılımının matematiksel ifadesini

*Γ (λ)* : Standart gama fonksiyonunu sembolize eder

*s* :Adım büyüklüğünü oluşturur.

Bu dağılım *s> 0* parametresine uyan büyük adımlar için geçerli olmaktadır. Teoride, $s_0 \gg 0$ olması gerekir, ancak uygulamada $s_0$ , *0.1* kadar küçük olabilmektedir. Yerel tozlanma için matematiksel tozlanma sürecini nicel olarak analiz eder. Matematiksel ifadesi Denklem (3.15)'te hesaplanmaktadır.

$$x^{t+1} = x_i^t + \epsilon\left(x_j^t - x_k^t\right) \quad (3.15)$$

Burada $x_j^t$ *ve* $x_k^t$ aynı bitki türlerinin farklı çiçeklerinden gelen polen türüdür. Bu algoritmanın optimizasyon için en önemli özelliği Levy dağılımını kullanarak arama uzayında birçok çözüm noktasının araştırılmasını sağlamaktadır.

### 3.1.3. Karınca Aslanı Algoritması (ALO)

Karınca aslanları, Myrmeleontidae ailesine ait olan ve larva evresinde ilginç beslenme davranışlarıyla bilinen yırtıcı bir böcek türüdür. Bu böcekler, karıncaların bulunduğu bölgelere tuzaklarını dairesel yol çizerek kurarlar. Kendilerini tuzakların dibine, yani koninin sivri ucunun alt kısmına gömerler ve karıncaları tuzağa düşürmek için beklerler. Karıncalar tuzağa girdiğinde karınca aslanları onları tuzağın dibine çekmek ve çıkışlarını engellemek için kum fırlatmaya başlarlar. Büyük çeneleriyle tuzağa düşen karıncaları yakalar ve yutarlar. Avlanmak için sürekli bunu tekrar ederler

(Yüzgeç ve Kılıç, 2018, s.13; Jama ve Baykan, 2020, s.404). Bu avlanma davranışları Şekil 3.7'de gösterilmektedir.

Şekil 3.7. Karınca aslanı avlanma stratejisi (Yüzgeç ve Kılıç, 2018, s.14).

Bu ilginç avlanma mekanizmasına ait matematiksel model Denklem (3.16) da ifade edilmektedir.

$$X(t) = \begin{bmatrix} 0 \\ cumsum(2r(t_1) - 1) \\ cumsum(2r(t_2) - 1) \\ \dots \\ cumsum(2r(t_n) - 1) \end{bmatrix} \tag{3.16}$$

Burada;

*n* : maksimum iterasyon sayısı

t : rastgele yürüyüş adımları

*cumsum*: kümülatif toplam

r(t) : Bir stokastik fonksiyondur. Denklem (3.17) de matematiksel ifadesi belirtilmiştir.

$$r(t) = \begin{cases} 1, & if\ rand > 0.5 \\ 0, & if\ rand \leq 0.5 \end{cases} \tag{3.17}$$

Rastgele yürüyüşe başlayan karıncaları arama uzayında tutmak için matematiksel ifadesi Denklem (3.18) de ifade edilmiştir.

$$X_i^t = (X_i^t - a_i)(d_i^t - c_i^t)(b_i - a_i)^{-1} + c_i^t \tag{3.18}$$

Burada;

*i* : değişken sayısını

*t* : iterasyon sayısını

*a* : minimum rastgele yürüyüşünü

*b* : maksimum rastgele yürüyüşünü

*c, d* : her bir iterasyonda güncellenen karınca aslanı pozisyonlarının sırasıyla minimum ve maksimum değerlerini göstermektedir.

Karınca tuzağa girdiğinde karınca aslanı onları tuzağın dibine çekmek için kum fırlatmaya başlar (Yüzgeç ve Kılıç, 2018, s.13). Buna ait matematiksel ifade Denklem (3.19 - 3.22) de gösterilmektedir.

$$c_i^t = Antlion_i^t + c^t \tag{3.19}$$

$$d_i^t = Antlion_i^t + d^t \tag{3.20}$$

$$c^t = c^t . I^{-1} \tag{3.21}$$

$$d^t = d^t . I^{-1} \tag{3.22}$$

Burada $I$ kaydırma oranını göstermektedir ve optimizasyon sırasında belirli oranlarda arttırılır. Bu mekanizmanın ayrıntıları Mirjalili'nin çalışmasında bulunabilir. (Mirjalili, 2015, s.80). Karıncaların oluşturdukları yeni pozisyonlarının matematiksel ifadesi Denklem (3.23) gibi olmaktadır (Yüzgeç ve Kılıç, 2018, s.13).

$$Ant_i^t = 0.5(R_A^t + R_E^t) \tag{3.23}$$

Burada $Ant_i^t$ ifadesinde *t.* iterasyonu, *i*. karıncayı ifade etmektedir. Karınca aslanı tuzağın dibine kaydırdığı karıncaları yediğinde kendi pozisyonunu Denklem (3.24)' e göre hesaplar:

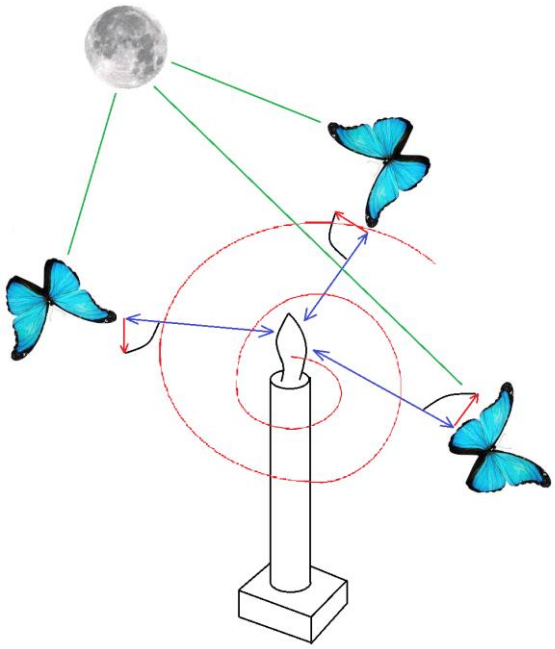
$$if\ f(Ant_i^t) < f(Antlion_i^t), \quad Antlion_i^t = Ant_i^t \tag{3.24}$$

Burada $Antlion_i^t$ ifadesinde *t.* iterasyonu i. karınca aslanını, $Ant_i^t$ ifadesinde ise *t.* iterasyonu, *i.* karıncayı ifade eder (Yüzgeç ve Kılıç, 2018, s.13).

### 3.1.4. Güve-Alev Optimizasyon Algoritması (MFO)

Güvelerin uçuş mekanizması doğadaki yatay yönelimli hareketlerden oluşmaktadır. Güveler, uçuşlarında ay ışığından faydalanarak gece uçmaktadırlar. Bu çalışma mekanizmalarında güveler, sabit bir açıyı koruyarak düz bir yolda uzun mesafelere gidebilmektedirler. Ay ile güve arasındaki mesafe, güvelerin düz bir şekilde

hareket etmelerini sağlar. Ancak enine yönelime rağmen güveler, genellikle ışıklar etrafında spiral şeklinde uçma eğiliminde olup yapay bir ışık kaynağında da aynı şekilde düz bir çizgide uçmaya çalışırlar. Ancak yapay bir ışık kaynağı, ay ile karşılaştırıldığında çok daha yakın olduğu için güvelerin uçuş yollarındaki hareketlerinde sapmalara neden olabilmektedir. MFO algoritması, güvelerin hareket mekanizmasından esinlenerek çalışır. Bu algoritma, matematiksel çözümleri güvelerle temsil eder ve problemin değişkenlerini güvelerin uzaydaki konumu olarak belirler. Güveler, konumlarını değiştirerek problem boyutunda havada hareket edebilirler (Gürgen, 2022, s.1508). Yapay bir ışık kaynağında güvelerin hareketi Şekil 3.8 gibidir.



Şekil 3.8. Güverlerin spiral uçuş yolu

MFO' nun temel çalışması, her güvenin karşılık gelen alevine göre konumunu güncellediğini belirtir. Her yinelemede güvelere atanan alev dizisi vardır. İlk güve en iyi aleve doğru hareket edecek ve sonuncusu ise en kötü aleve doğru hareket edecektir. Bu, güvelerin ortamdaki gerçek davranışı değildir. Bununla birlikte, her güveye bir alev atamak, keşfi teşvik etmesiyle beraber ancak sömürü kabiliyetini de düşürebilmektedir.

Önerilen yaklaşımda tüm güveler için en iyi alevin etkisi alınmıştır. Burada tüm güveler, ilgili alevlerine ve en iyi aleve göre konumlarını güncelleyecektir. Güveler, en iyinin ortalama konumuna ve buna karşılık gelen aleve doğru hareket edecektir. Güveler en iyi aleve doğru hareket ettiğinden, bu değişiklik sömürüyü teşvik eder ve yakınsama

hızını artırmaya yardımcı olur. En iyi alevin etkisi, yalnızca daha fazla sömürünün gerekli olduğu yinelemelerin ikinci yarısında alınır (Kaur, Singh ve Salgotra, 2020, s.2315). Yeni güncellenen matematiksel ifade Denklem gibi olmaktadır (3.25-3.26):

$$S(M_i, F_j) = D_i . e^{bt} . Cos(2\pi t) + \frac{a * F_j + (1 - a) * en\ iyi\ alev}{2} \quad (3.25)$$

$$a = 0.6 - 0.4 * \frac{L}{T} \quad (3.26)$$

Bu güncelleme denkleminde $M_i$, *i.* güvenin mevcut konumunu kullanır. $F_j$, yaklaşılmak istenen j. alevi ifade eder. $D_i$, güve ile alev arasındaki mesafeyi temsil eder. *b*, sarmalın şeklini tanımlayan bir sabittir. t, -1 ile 1 arasında rastgele sayıdır. L mevcut iterasyonu, T ise maksimum iterasyonu temsil eder.

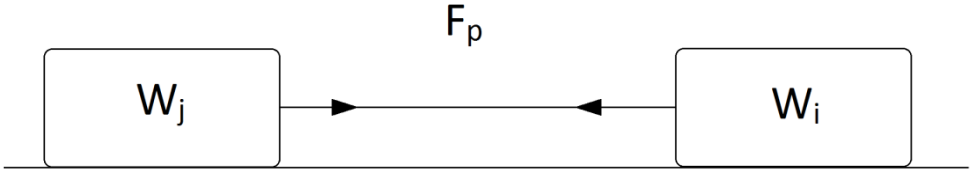
Burada "a" parametresi lineer azalan fonksiyondur ve en iyiye karşılık gelen alevin etkisini kontrol eden kontrol faktörü olarak kullanılır. İterasyonlara göre "*a*" değeri 0,2 ile 0,4 arasında değişir. İlk aşamada, bir parametrenin değeri yüksek tutulur, bu da ilgili alevin etkisinin daha fazla olduğunu gösterir. Böylece güve arama alanını verimli bir şekilde keşfedebilir.

### 3.1.5. Halat Çekme Optimizasyonu (TWO)

Halat çekme, rakip takımın çekme kuvvetine göre karşıdaki kişileri kendi tarafına getirmek için bir halatın zıt uçlarını çeken iki rakip takımın bir güç yarışmasıdır. Faaliyet çok eski zamanlara dayanmaktadır ve o zamandan beri farklı şekillerde varlığını sürdürmektedir. Oyun için çok çeşitli kurallar ve düzenlemeler vardır, ancak temel kısım neredeyse hiç değişmeden kalmaktadır. Doğal olarak, her iki takım da ipi sıkıca kavradığı sürece, ipin hareketi kaybeden takımın yer değiştirmesine dayanır. Şekil 3.9' da çekişmede yarışan takımlardan birini göstermektedir.

Şekil 3.9. Halat çekmede yarışan bir takım

Gerçek bir halat çekme oyununda zafer genellikle birçok faktöre bağlıdır ve analiz edilmesi zor olabilir. Ancak bu çalışmada iki takımın ağırlıklarının olduğu idealize edilmiş bir çerçeve kullanılmıştır. *Wi* ve *Wj* Şekil 3.10'da gösterildiği gibi düz bir yüzey üzerinde yatan iki nesne olarak kabul edilir.




Şekil 3.10. İdealleştirilmiş bir halat çekme çerçevesi (Kaveh ve Zolghadr, 2016)

$\boldsymbol{\mu}$s : Statik sürtünme katsayısı, $\boldsymbol{\mu}$k : Kinematik sürtünme katsayısı

İpin çekilmesi sonucunda takımlar eşit ve zıt iki kuvvet *(Fp)* yaşarlar. Nesne i için çekme kuvveti, maksimum statik sürtünme kuvveti (Wi $\boldsymbol{\mu}$s) kadar küçük olduğu sürece, nesne başlangıç yerinde durur (Kaveh ve Zolghadr, 2016, s.471). Aksi takdirde sıfır olmayan bileşke kuvvet Denklem (3.27) de hesaplanmaktadır:

$$F_r = F_p - W_i \mu_k \tag{3.27}$$

Sonuç olarak, Newton'un ikinci yasasına göre, i nesnesi, j nesnesine doğru ivmelenir. Matematiksel ifade Denklem (3.28) deki gibidir:

$$a = \frac{F_r}{\left(W_i/g\right)} \tag{3.28}$$

*i* nesnesi sıfır hızdan başladığı için, yeni konumu Denklem (3.29) da belirlenir:

$$X_i^{yeni} = \frac{1}{2}at^2 + X_i^{eski} \tag{3.29}$$

### 3.1.6. Atom Arama Optimizasyonu (ASO)

ASO'da, arama uzayındaki her bir atomun konumu, kütlesi ile ölçülen bir çözümü temsil eder, daha iyi bir çözüm daha ağır bir kütleyi gösterir ve bunun tersi de geçerli olmaktadır. Popülasyondaki tüm atomlar, aralarındaki mesafeye göre birbirlerini çekecek veya itecek, bu da daha hafif atomların daha ağır olanlara doğru hareket etmesini teşvik edecektir. Atomlar, molekülleri oluşturmak için kovalent bağlarla bağlanır. Atomların kütlesi ve hacmi vardır. Atomlar arasındaki etkileşim kuvvetleri, atomlar arasındaki farklı mesafelere göre itme veya çekme kuvveti olarak gösterilir. İtme bölgesinde, mesafeler azaldıkça atomlar arasındaki itme kuvvetleri keskin bir şekilde artar (Fu ve ark., 2020). Çekici bölgede mesafeler belli bir oranda arttığında çekim kuvvetleri maksimuma ulaşır. Mesafeler artmaya devam ettikçe çekim kuvvetleri kademeli olarak sıfıra düşer. İki atom denge mesafesinde olduğunda, aralarındaki etkileşim kuvveti sıfırdır. Moleküllerde, atomik kuvvetlerin geometrik kısıtlamaların ve atomların iç hareketlerinin etkilerini de dikkate alması gerekir. Geometrik kısıtlamalar ve atomların iç hareketleri bağlama kuvveti olarak adlandırılır (Erdal ve ark., 2019, s.841; Hekimoglu, 2019, s.38100). Newton'un ikinci yasasına göre, $F_i$ bir etkileşim kuvvetini ve $G_i$ bir kısıt kuvvetini ifade eder. Bunlar, atom sistemindeki i. atoma birlikte uygulanır. Bu durumda, kütlesi ($m_i$) ile ilişkili olarak ivmesinin ($a_i$) matematiksel ifadesi basitçe Denklem (3.30) gibi yazılabilir:

$$F_i + G_i = m_i a_i \tag{3.30}$$

ASO algoritması, moleküler dinamikten esinlenen popülasyon tabanlı sezgisel-üstü bir algoritmadır (Erdal ve ark., 2019, s.841). ASO'daki her bir atomun arama uzayındaki konumu, kendisinden daha ağır ya da daha hafif bir kütleye işaret eden bir çözüme sahiptir. Popülasyondaki tüm atomlar ağır olanların hafif olanları kendine

çekmesi gibi birbirini ya çekecek ya da itecektir. Daha ağır atomların hız değişimi daha yavaş olduğundan yerel uzayda daha iyi çözümler üretmelerine neden olurken hafif atomlar daha hızlı olduğundan yeni ve ümit verici alanları bulmak için tüm arama uzayında tarama yaparlar (Eker, Kayri ve Ekinci, 2019, s.845). ASO'da her bir atom en iyi atom ile bir kovalent bağa sahiptir. Yani her bir atom en iyi atoma doğru hareket etmeye zorlanır ve *i*. atomun kısıdı ise Denklem (3.31) de ifade edilmiştir.

$$Q_i(t) = \left[|x_i(t) - x_{best}(t)|^2 - b_{i,best}^2\right] \tag{3.31}$$

Burada $x_{best}(t)$ ifadesi en iyi atomun *t*. iterasyondaki konumunu temsil eder. $b_{i,best}^2$ en iyi atom ile *i*. atom arasındaki sabit bağ uzunluğunu ifade eder. Dolayısıyla kısıtlama kuvveti ise Denklem (3.32) de hesaplanmaktadır (Eker, Kayri ve Ekinci, 2019, s.845).

$$G_i^d(t) = \lambda(t)\left(x_{best}^d(t) - x_i^d(t)\right) \tag{3.32}$$

Burada $\lambda(t) = \beta e^{\frac{20t}{T}}$ Langragian çarpanıdır ve $\beta$ ise çarpan ağırlığı olarak ifade edilir. Bu çalışmada $\beta$, 0.2 olarak alınmıştır. *i*. atomun *t*. iterasyondaki kütlesi aşağıdaki Denklem (3.33 – 3.34) te hesaplanır (Eker, Kayri ve Ekinci, 2019, s.845).

$$M_i(t) = e^{\frac{Fit_i(t) - Fit_{best}(t)}{Fit_{worst}(t) - Fit_{best}(t)}} \tag{3.33}$$

$$m_i(t) = \frac{M_i(t)}{\sum_{j=1}^{N} M_j(t)} \tag{3.34}$$

Burada $Fit_{best}(t)$ ve $Fit_{worst}(t)$ sırasıyla *t*. iterasyondaki minimum ve maksimum uygunluk değerine sahip atomlardır. $Fit_i(t)$ ise *i*. atomun *t*. iterasyondaki uygunluk değer fonksiyonudur. Algoritmayı basitleştirmek için *i*. atomun *(t+1)*. İterasyondaki konum ve hız vektörleri de Denklem (3.35) ve Denklem (3.36) gibi ifade edilir (Eker, Kayri ve Ekinci, 2019, s.845).

$$v_i^d(t+1) = rand_i^d v_i^d(t) + a_i^d(t) \tag{3.35}$$

$$x_i^d(t+1) = x_i^d + v_i^d(t+1) \tag{3.36}$$

ASO algoritmasında, keşif ve sömürü aşamalarında atomların etkileşime girmesi önemli olmaktadır. Keşif aşamasında, her bir atomun *K* komşusuyla etkileşime girmesi

gerekmektedir. Bu komşular, daha iyi uygunluk değerlerine sahip atomlardır. Bu etkileşimler, keşfi geliştirmek için yapılan bir adımdır. Öte yandan, sömürü aşamasında da atomlar, *K* komşusuyla etkileşime girmelidir. Bu etkileşimler, sömürüyü artırmak için gerçekleştirilir. Zamanla, *K* değeri azalır. *K* değeri, iterasyonların artmasıyla birlikte yavaş yavaş azalır ve Denklem (3.37)'de hesaplanır (Eker, Kayri ve Ekinci, 2019, s.845).

$$K(t) = N - (N-2) * \sqrt{\frac{t}{T}} \tag{3.37}$$

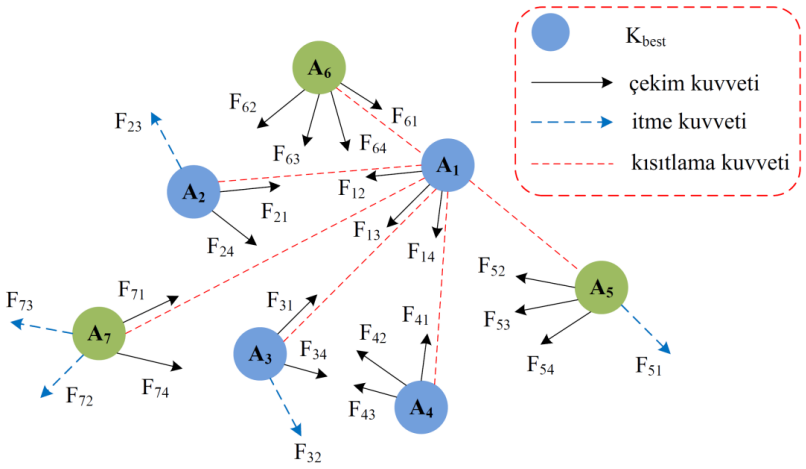
Burada;

*N* :sistemdeki toplam atom sayısı

*t* : mevcut iterasyon sayısı

*T* :maksimum iterasyon sayısıdır.

Bir atom popülasyonunun kuvvetleri Şekil 3.11'de gösterilmiştir. Burada, en iyi uygunluk değerine sahip ilk 5 atom $K_{best}$ olarak kabul edilir (Eker, Kayri ve Ekinci, 2019, s.845; Hekimoglu, 2019, s.38100).




Şekil 3.11. K=5 için $K_{best}$ ile Temsil Edilen Atom Sisteminin Etkileşimleri (Eker, Kayri ve Ekinci, 2019, s.841)

### 3.1.7. Beyin Fırtınası Optimizasyonu (BSO)

BSO, insanların beyin fırtınası süreci olarak adlandırılan kolektif davranışı simüle eder. BSO'nun üç ana bileşeni vardır: kümeleme, seçme ve üretim (Cheng ve ark., 2014, s.86). BSO, benzer bireyleri kümelemek için k-ortalama kullanır ve her kategorinin en iyi bireyini merkez olarak belirler. Rastgele bir birey, merkezlerden

birinin yerine olasılıksak olarak yerleştirilebilir. Yeni bireylerin üretimini yönlendirmek için dört tür birey seçilir: bir kümelerin merkezi, bir kümelerin rastgele bireyi, iki kümeye ait iki merkezin kombinasyonu ve iki kümeye ait iki rastgele bireyin kombinasyonu olarak ifade edilir (Cai ve ark., 2022, s.10896).

Beyin fırtınası optimizasyon algoritması, bir tür arama alanı azaltma algoritmasıdır; tüm çözümler sonunda birkaç kümeye girecektir. Bu kümeler, bir sorunun yerel optimumunu gösterir. Bir alanın bilgisi, iyi uygunluk değerlerine sahip çözümler içerir ve bir kümeden diğerine yayılır. Bu algoritma ilk önce karar uzayında keşif yapacak ve iterasyonlardan sonra keşif ve sömürü, bir denge durumuna geçecektir. Yeni bireyler Denklem (3.38) ile hesaplanır.

$$X' = X_{seçilmiş} + \xi(t).N(0,1) \tag{3.38}$$

$X^1$, yen bir bireyi temsil ettiği yerde $X_{seçilmiş}$ seçim sonucu olur. *N(0, 1)* standart normal dağılımı temsil eder. $\xi(t)$ Denklem (3.39)' da gösterildiği gibi adım boyutunu ayarlamak için bir işlevdir.

$$\xi(t) = logsig\left(\frac{0.5.T - t}{c}\right).U(0,1) \tag{3.39}$$

*logsig(),* log-sigmoid aktarım işlevi olduğu yerde *T* ve *t* sırasıyla maksimum yineleme sayısı ve geçerli yineleme sayısıdır. *c, logsig()*'in eğimini kontrol eden bir katsayıdır. *U(0, 1)*, 0'dan 1'e düzgün dağılım temsil eder. Algoritma 2'de BSO süreci sunulmuştur.

***Algoritma 2: Beyin fırtınası optimizasyon algoritmasının prosedürü***

*1 Başlatma: Rastgele n potansiyel çözüm (birey) üret ve n bireyi değerlendir;*

*2 Eğer"yeterince iyi" bir çözüm bulamamış veya önceden belirlenmiş maksimum sayıya ulaşmamışsa;*

*3 Kümeleme: Bir kümeleme algoritması ile n kişiyi m küme halinde kümeleyin;*

*4 Yeni bireylerin üretimi: yeni birey oluşturmak için rastgele bir veya iki küme seçin;*

*5 Seçim: Yeni oluşturulan birey, mevcut bireyle karşılaştırılır. Aynı bireysel indeks, daha iyi olanı yeni birey olarak tutulur ve kaydedilir.*

### 3.1.8. Su Döngüsü Algoritması (WCA)

WCA, su döngüsünün gözlemlenmesine dayanan popülasyon tabanlı metasezgisel bir algoritmadır. Gerçek dünyada suların ve derelerin yokuş aşağı denize doğru aktığında nehirleri oluşur. Çoğu nehir kar veya buzulların eridiği yüksek rakımlarda oluşur. Akarsular ve nehirler sonunda bir deniz olur. Suyun buharlaşıp atmosfere karışmasıyla beraber daha soğuk atmosferde yoğunlaşarak suyu yağmur veya yağış şeklinde yeryüzüne geri salan bulutlar oluşturur. Rastgele oluşturulan ilk popülasyona yağmur damlaları denir (Hanafi ve ark., 2021, s.012005, 7). Bu nedenle, bir yağmur damlası bir dizi tasarım ve karar değişkeni içeren bireysel veya tek çözümden oluşmaktadır. En iyi birey (en iyi yağmur damlası), belirli bir soruna en uygun çözümü temsil etmesi için deniz olarak seçilir. WCA algoritması içerisinde çalışma mekanizmaları matematiksel olarak 12 denklem ile ifade edilmiştir. WCA algoritması içinde en küçük değere "Yağmur Damlası" denilmektedir. Popülasyon tabanlı metasezgisel yöntemlerde optimizasyon için değişken değerlerine ait bir dizi oluşturulmaktadır (Gür, 2020, s.16). Bir $N_{var}$ boyutlu optimizasyon denkleminde, bir yağmur damlası $N_{var}$ olarak sembolize edilir ve şu şekilde hesaplanır. Denklem (3.40):

$$Yağmur\ Damlası = N_{var} = [x_1, x_2, x_3, \dots, x_N,] \tag{3.40}$$

Algoritmanın başlayabilmesi için $X = N_{pop} x N_{var}$ boyutunda bir yağmur damlası popülasyonu oluşturulur. Bu nedenle, rastgele üretilen $X$ matrisi Denklem (3.41) de verilmiştir:

$$X = \begin{bmatrix} YD_1 \\ YD_2 \\ YD_3 \\ . \\ . \\ YD_{Npop} \end{bmatrix} = \begin{bmatrix} X_1^1 & X_2^1 & X_3^1 & . & X_{Nvar}^1 \\ X_1^2 & X_2^2 & X_3^2 & . & X_{Nvar}^2 \\ . & . & . & . & . \\ X_1^{Npop} & X_2^{Npop} & X_3^{Npop} & . & X_{Nvar}^{Npop} \end{bmatrix} \tag{3.41}$$

Karar değişkeni değerlerinin ($X_1$, $X_2$, … , $X_{var}$) her biri başlangıç noktası numarası veya sürekli ve ayrık problemler için dizi olarak gösterilebilir. Bir damlanın maliyeti en düşük değeri temsil eder. Maliyet fonksiyonu, deniz ve nehirler olarak belirlenen bireyler üzerinde ekonomik olarak tanımlanır. Denizin mutlak hedef olduğu düşünülerek tek bir değeri bir olarak kabul edilir. Buna göre, verilen denklemde *N_sr*, nehir sayısını ve denizin toplam değerini ifade eder. Geri kalan değerler ise yağmur

damlalarının nehirlerde veya doğrudan denizde biriktirebileceği değerlerden oluşur (Gür, 2020, s.17).

Sonuç olarak su zerreciklerin birleşmesi suyu büyütür. Buharlaşma ve Erozyon ise suyun küçülmesine sebep olur (Gür, 2020, s.17). Denklem (3.42).

$$C_{i=} = Maliyet_i = f\left(x_1^i, x_2^i, x_3^i, \dots, x_{Npop}^i\right) \tag{3.42}$$

Burada *i=1,2,3,…,N_pop*

$$N_{sr} = Nehir\ Sayısı + \underbrace{1}_{Deniz} \tag{3.43}$$

$$N_{yd} = N_{pop} - N_{sr} \tag{3.44}$$

Akışkanlık hızına bağlı olarak damlaların nehirlere ve denize akışını modelleyen Denklem (3.45) te $NS_n$, belirli nehirlere veya denize akan akarsuların sayısını oluşturur (Gür, 2020, s.17).

$$NS_n = yuvarla\left\{\left|\frac{Maliyet_n}{\sum_{i=1}^{N_{ST}} Maliyet_i}\right| XN_{yd}\right\} \tag{3.45}$$

Burada *n=1,2,…$N_{ST}$*

Bir akışın sağlanabilmesi için yağmur damlalarının birleşerek nehir ve deniz gibi suları oluşturması gerekir. Nehirler ve akarsular en düşük seviye olan denize doğru akarlar (Gür, 2020, s.17). Nehre doğru olan akış koşulu matematiksel ifadesi Denklem (3.46) gibidir:

$$X \in (0, C \times d), \quad C > 1 \tag{3.46}$$

Burada *C*, *1* ile *2* arasında ve 2' ye yakındır. Dolayısıyla *C=2* şeklinde hesaplanır. Akış ve nehir arasındaki uzaklık değeri, (*0 ve* $C \times d$)'nin arasındaki bir değer gelecektir. $C > 1$ olduğu için akarsuların nehirlere doğru akışını gerçekleştirmektedir. Matematiksel ifade Denklem (3.47) ve Denklem (3.48)' de belirtilmektedir (Gür, 2020, s.18).

$$X_{akış}^{i+1} = X_{akış}^i + rand\ x\ C\ x\left(X_{nehir}^i - X_{akış}^i\right) \tag{3.47}$$

$$X_{nehir}^{i+1} = X_{akış}^{i} + rand\ x\ C\ x\ \left(X_{deniz}^{i} - X_{nehir}^{i}\right) \quad (3.48)$$

Denklem (3.49)'daki $d_{max}$ değeri sıfıra yakındır. Eğer, bir nehir ve deniz arasındaki mesafe yani $dmax$'tan az ise, nehrin denize ulaşacağı anlaşılır. Denizde de buharlaşma işlemi gerçekleşir. Buharın biriken ve yoğuşarak yağmur damlası haline dönüşmesi sonucunda, yağmur damlaları tekrar yeryüzüne düşer. Bu süreçte, damlaların seyahat ettiği yolun en az maliyetli olması hedeflenir. Bu amaçla, damlaların hareketi $dmax$ değeri en yüksek olan noktadan başlar ve denize yaklaşırken yoğunluk azalarak en düşük deniz yoğunluğuna ulaşır. Bu şekilde, en az maliyetli yolun bulunması amaçlanır (Gür, 2020, s.18).

$$d_{max}^{i+1} = d_{max}^{i} - \frac{d_{max}^{i}}{max\ \ iterasyon} \quad (3.49)$$

Buharlaşma uygun sıcaklıkta yoğunlaşarak yağmur olarak yağmaktadır. Yeni yağmur damlaları birleşerek akarsuları meydana getirir (Gür, 2020, s.18). Yeni akarsuların konumlarını oluşturmak için Denklem (3.50) de hesaplanır:

$$X_{akış}^{yeni} = LB + rand\ X\ (UB - LB) \quad (3.50)$$

*LB* ve *UB* değerleri sırasıyla alt ve üst sınırlardır. Doğrudan denize akan akarsular için algoritmanın daha hızlı yakınsama yapması için Denklem (3.51) ile hesaplanır (Gür, 2020, s.18).

$$X_{akış}^{yeni} = X_{deniz} + \sqrt{\mu}\ X\ randn\ (1, N_{var}) \quad (3.51)$$

Burada;

$\mu$ : denize yakın arama bölgesi aralığını gösteren bir katsayıdır.

*randn* :normal olarak dağıtılan rasgele sayıdır.

$\mu$ değeri daha büyük seçilebilir, dolayısıyla uygulanan alan büyür ve "doğrudan denize akan dere" bulunma ihtimalini artırır. $\mu$'ya 0,1 değeri ön tanımlama olarak atanmıştır. Aynı zamanda matematiksel ifade olarak $\mu$ terimi standart sapmayı ve buna bağlı olarak varyans kavramını da ifade etmektedir (Gür, 2020, s.18).

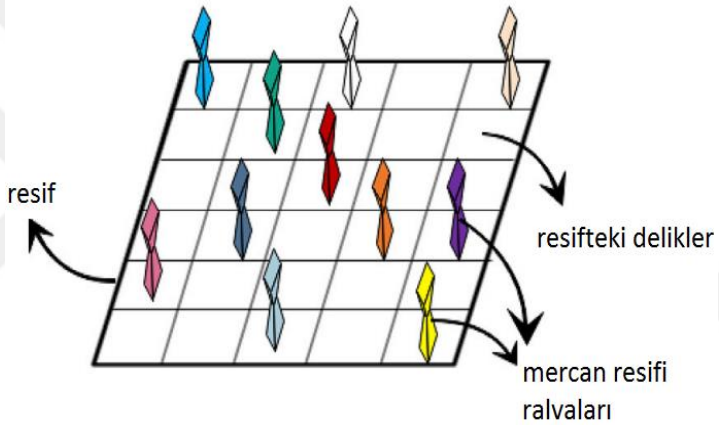
### 3.1.9. Mercan Resifleri Optimizasyonu (CRO)

Mercan Resifleri Optimizasyonu (CRO) algoritması, mercan resiflerindeki süreçlerin simülasyonuna dayanan yeni bir evrimsel biyo-ilhamlı yaklaşımdır. CRO algoritması, belirli bir doğal ekosistemin davranışının yapay simülasyonunu deneyen, biyo-esinli algoritmalar ailesi içinde sınıflandırılmaktadır. CRO algoritmasının literatürdeki alternatif optimizasyon algoritmalarından daha iyi çözümler elde ederek birçok tek amaçlı optimizasyon probleminde etkili olduğu kanıtlanmıştır. Temel olarak, CRO algoritması, belirli bir optimizasyon problemine farklı çözümleri kodlayan bir bireyler popülasyonundan başlar. Bu çözümler, algoritmanın başında boş alanların da bulunduğu kare bir ızgarada (resif) bulunur. Algoritmanın mercan üreme sürecini (cinsel ve eşeysiz üreme operatörleri uygulanır) ve uzay için bir mücadelenin meydana geldiği mercan resifi oluşum sürecini simüle ettiği düşünülmektedir. Böylece CRO algoritmasının her adımında mercan larva oluşumu gerçekleşir ve her larva resifte kendine yer edinmeye çalışır. Larvanın ne kadar güçlü olduğuna (optimizasyon probleminin çözümünün ne kadar iyi olduğuna) veya resifte boş bir yer bulacak kadar şanslı olup olmadığına bağlıdır (Salcedo-Sanz, ve ark. 2016, s.966).

Temel olarak CRO, bir $N \times M$ kare ızgaradan oluşan bir mercan resifinin yapay modellemesine dayanır. Her karenin(i,j) bir mercanı (veya mercan kolonisini), tahsis edebileceği varsayılır, bu da belirli bir alfabede bir sayı dizisi olarak kodlanan belirli bir optimizasyon problemine bir çözümü temsil etmektedir. CRO algoritması ilk olarak, bazı karelerin mercanlar tarafından işgal edilmesini ve ızgaradaki diğer bazı karelerin boş olması yani resifte yeni mercanların gelecekte serbestçe yerleşebileceği ve büyüyebileceği delikler atanarak rastgele başlatılmaktadır. Algoritmanın başlangıcındaki serbest/dolu kareler arasındaki oran, CRO algoritmasının önemli bir parametresidir ve $\rho$ ile gösterilir. $0<\rho 0<1$ eşitlikte olmasına dikkat edilmesi gerekmektedir. Her mercan, problemin amaç fonksiyonunu temsil eden ilişkili bir sağlık fonksiyonu $f(i,j): I \rightarrow R$ ile etiketlenmiştir. CRO, daha sağlıklı (daha güçlü) mercanlar hayatta kaldığı sürece resifin ilerleyeceği, daha az sağlıklı mercanların ise yok olacağı gerçeğine dayanır (Salcedo-Sanz ve ark. 2016, s.966).

Yukarıda açıklanan resifin başlatılmasından sonra, resifin ikinci aşaması oluşumu ile beraber CRO algoritmasında yapay olarak simüle edilir. Resifte mercan

üremesinin bir simülasyonu, farklı operatörlerin sırayla uygulanmasıyla yapılır. Bu sıralı operatörler seti daha sonra belirli bir durdurma kriteri karşılanana kadar uygulanmaktadır. Mercan üremesini taklit etmek için çeşitli operatörler tanımlanmıştır, bunların arasında: mercan eşeyli üreme modellemesi (yayın yumurtlama ve kuluçka), eşeysiz üreme modeli (tomurcuklanma) ve resifteki bazı felaket olayları, yani polip öncesi oluşumu yer alır. Eşeyli ve eşeysiz üremeden sonra oluşan larva seti, resifte büyüyecek bir yer bulmaya çalışır. Boş bir alanda veya işgal edilmiş bir alanda olabilmektedir. Aslında o yerde bulunan mercanlara karşı savaşarak ta olabilir. Larvalar, belirli sayıda deneme ile büyüyecek bir yer bulmayı başaramazlarsa, bu aşamada yok olmaktadırlar. Şekil 3.12'de resifin başlatılmasını gösteren 5-6 ızgaraya sahip model mercanlar ve mercan kolonileri resifi gösterilmektedir.




Şekil 3.12. Mercan resifi modeli (Yan ve ark., 2019, s.103)

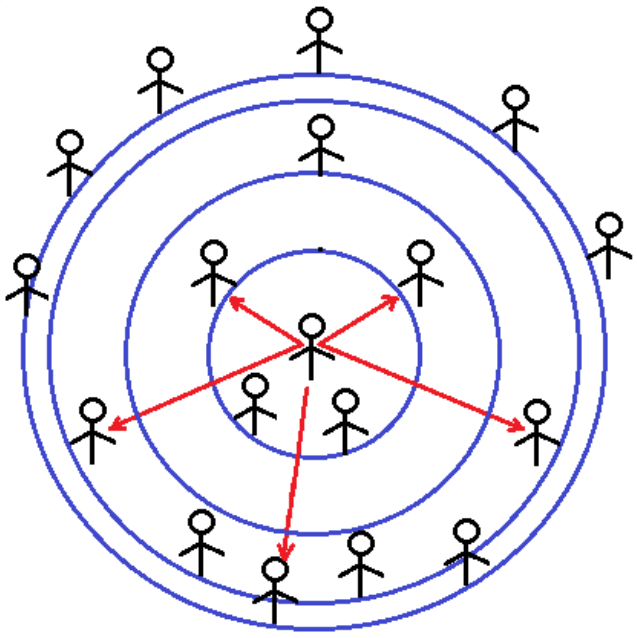
### 3.1.10. Yaşam Seçimi Tabanlı Optimizasyon (LCO)

LCO algoritması, insanın yaşam döngüsünü ve çalışma etiğini dikkatlice gözlemleyerek ilham almaktadır. İnsanlar doğadan ilham alarak yeni şeyler öğrenmişlerdir. LCO algoritması da insan davranışlarına dayanan yeni bir teknolojidir. İnsanlar diğer türlerden öğrenme kabiliyetine sahip olmuşlardır ve bu, hayvanların korunmasına ve hayatta kalmasına odaklanmalarına yardımcı olmuştur. İnsanlar hayvanları evcilleştirmiş, evcil hayvan olarak benimsemiş ve böylece karşılıklı hayatta kalma odaklı çalışmalar yürütmüşlerdir. Bu çalışmada önerilen algoritma, seçici etkiden yararlanan Jaya optimizasyon tekniğinden de ilham almaktadır (Khatri ve ark., 2020, s. 9121). İnsan her zaman bir şeylerden ilham alır, ya kendisi gibi birini, bir ünlüyü ya da arkadaşlarını takip eder. Bir kişinin hedefi varsa, o kişi strateji oluşturmak için o

alandaki en iyi insanların nasıl çalıştığını düşünür ve inceler. Kişi, hedefi gerçekleştirmek için en iyi kişilerden faydalı bir şeyler almaya çalışır ve üstün kişinin verimliliğini gözlemleyerek bir desen veya parametre türetir. Böylece, hedefi gerçekleştirmek için bazı beceriler geliştirebilir veya ele alınan problemleri çözebilir. Sıralanmış uyum değerlerine ve satın alma fonksiyonlarına sahip bir veri kümesi *X* için, Denklem (3.52), LCO algoritmasının en iyi özelliklerinden öğrenmeyi temsil eder (Khatri ve ark., 2020, s. 9122).

$$X_j' = \sum_{k=1}^{n} [rand(k) * X_k]/_n \tag{3.52}$$

Burada toplamda, *k, 1*'den *n*'ye kadar değişir, *n* algoritmadaki bir parametredir ve sorunu çözmek için dikkate alınan popülasyonun karekökünün üst tamsayıya eşittir. Parametre $X_j$, *j*. veya mevcut arama ajanını temsil eder ve $X_j'$, $X_J$'den daha iyi bir uygunluğa sahipse yalnızca $X_j$ 'nin güncelleneceğini belirtir. Şekil 3.13' te algoritmanın bu özelliği gösterilmektedir. Dairelerin ortasındaki arama aracısı, süreçteki geçerli arama aracısını temsil eder. Rasgele sayılarla değişken uzunlukların etki düzeyi belirlenmektedir.



Şekil 3.13. LCO da ortak en iyi gruptan öğrenme

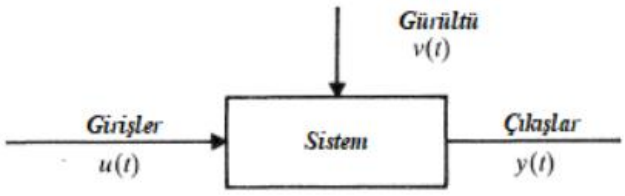
## 3.2. Sistem Tanımlama

### 3.2.1.Sistem Tanımlamanın Temel Anlamı

Sistem tanılama, dinamik bir sistemin modelinin gerçek sistemden alınan giriş-çıkış ölçümleriyle bulunması anlamına gelir. Sistem tanılamanın amacı, belirli bir sisteme veri giriş-çıkışı ilişkisinden, daha sonra sistem üzerinde tekrar yapılacak çalışmalarda kullanılabilecek güvenilir bir matematiksel model oluşturmaktır. Sistem tanımlama uygulamaları kontrol mühendisliği alanının önemli bir alt dalı olarak kabul edilir (Fissore, 2021). Birçok tanımı olan sistemin tanımları şöyle yapılabilir;

1. Sistem birbiriyle etkileşim halinde olan bileşenler kümesidir.
2. Sistem, öğelerin veya birleşenlerin ilişkilerinin bir amacını gerçekleştirmek için organize edilmesidir.
3. Sistem, bir amaca yönelik olarak bir araya gelen ve aralarında düzenli ilişkiler bulunan öğelerin oluşturmuş oldukları bir bütündür.
4. Bir amacı gerçekleştirmek ve sonuca ulaşmak için, tasarlanan bağımlı ve/veya bağımsız birimlerin bir plana göre organize edilen ve bütünü oluşturan parçalarıdır.

Şekil 3.14'te belirtilen sistemde giriş *u(t)*, gürültü *v(t)* ve bu işlemlerin sonucunda bir çıkış değeri *y(t)* elde etmektedir. Sistemlerde girişler, kontrol edilmektedir. Ancak gürültüler tahmin edilemediğinden kontrol edilememektedir. Sistem üzerinde kontrolleri yapabilmek için sistemin bir modelinin olması gerekmektedir. Bu sistemi işleyişini gösteren matematiksel ifade gerçek sistemi temsil etmektedir.




Şekil 3.14. Temel sistem tasarımı

Sistemi modelleyen kısım gerçek sisteme ne kadar benzer tasarlanırsa ardından kontrol algoritmaları da gerçek sisteme uyarlandığında o kadar iyi sonuçlar verebilmektedir.

Sistem Tanımlama (SI), deneysel verilerden dinamik sistemlerin matematiksel modellerini oluşturmaya yönelik bir metodolojidir, yani belirli bir model yapısında ayarlanabilir parametrelerin değerlerini tahmin etmek için sistem giriş/çıkış (IO) sinyallerinin ölçümleri kullanılmaktadır (Fissore, 2021). Sistem tanımlama süreçleri ile ilgili gerekli adımlar;

1. Sistemin IO sinyallerinin ölçülmesi,
2. Aday model yapısının seçimi,
3. Aday model yapısındaki ayarlanabilir parametrelerin değerini tahmin etmek için bir yöntemin seçimi ve uygulanması,
4. Tercihen farklı bir veri seti ile yapılması gereken modelin uygulama ihtiyaçları için doğru olup olmadığını görmek için tahmin edilen modelin doğrulanması ve değerlendirilmesi,

Sistemin matematiksel olarak ifade edilmesi, sistemin girişleri, çıkışları ve içyapısı arasındaki ilişkilerin matematiksel denklemler veya formüller kullanılarak gösterilmesidir. Matematiksel ifadeler, sistemin dinamik veya statik özelliklerini açıklayabilir, sistem davranışını modelleyebilir ve analiz edilebilecek hale getirebilir. Matematiksel ifadeler genellikle denklem veya formül şeklinde ifade edilir. Dinamik sistemlerde diferansiyel denklemler, integral denklemler veya diferansiyel ve integral denklem kombinasyonları kullanılabilir. Statik sistemlerde ise genellikle denklem veya formülde sabit oranlar kullanılır.

Örneğin, bir elektrik devresi matematiksel olarak Ohm Kanunu veya Kirchoff Yasaları kullanılarak ifade edilebilir. Bir mekanik sistem, Newton'un hareket yasaları veya Euler-Lagrange denklemleri ile matematiksel olarak modellenebilir.

Matematiksel ifadeler sayesinde sistem davranışı analiz edilebilir, sistem parametreleri optimize edilebilir ve kontrol tasarımları yapılabilir. Ancak matematiksel modelin doğru bir şekilde oluşturulması, sistemin tam olarak anlaşılması ve sisteme ait parametrelerin doğru bir şekilde belirlenmesi gerekmektedir (Fidan, Sevim ve Erkan, 2022, s.25).

### 3.2.2. Matematiksel model ve sistem tanımlama

Matematiksel model; bir sistemin davranışını, giriş ve çıkışları arasındaki ilişkileri matematiksel denklemler veya formüller kullanarak ifade eden bir yapıdır. Matematiksel modelleme, gerçek dünyadaki sistemleri anlamak, analiz etmek, tasarlamak ve kontrol etmek için kullanılan bir yöntemdir.

Matematiksel modelleme daha sağlam ve kesin bir yöntemdir, çünkü sistemin giriş ve çıkışları arasındaki ilişkiyi tam olarak ifade eder. Matematiksel modeller, sistemin matematiksel denklemler veya formüller aracılığıyla analiz edilmesine ve kontrol tasarımına olanak sağlar. Ancak sistemin matematiksel modelini tam olarak belirlemek bazen zor olabilir ve deneysel verilere dayalı yaklaşımlar kullanmak gerekebilir.

Sistem tanımlama ise, matematiksel modelini tam olarak çıkaramadığımız sistemlerde, deneysel verileri kullanarak sistemi modelleme işlemidir. Bu durumda, sistemin matematiksel ifadesini belirlemek zor olabilir veya mümkün olmayabilir. Bu nedenle, deneysel veriler kullanılarak sistemin davranışı anlaşılmaya çalışılır ve bu verilere dayanarak sistemi modellemek ve analiz etmek için çeşitli yöntemler kullanılır. Sistem tanımlama, gerçek dünyadaki sistemlerin karmaşıklığını ele alırken matematiksel modellemenin sınırlamalarını aşmayı amaçlar (Fidan, Sevim ve Erkan, 2022, s.26).

### 3.2.3. Hata fonksiyonları

Sistem parametrelerini belirlemek ve performansları incelemek için karesel hatanın integrali (Integral Square Error-ISE), hatanın mutlak değerinin integrali (Integral absolute error - IAE), zaman ağırlıklı karesel hatanın integrali (The Integral Time Square Error -ITSE) , zaman ağırlıklı mutlak hatanın integrali (The Integral Time Absolute Error - ITAE) ve zamansal ve karesel hatanın integrali (Integral square time error - ISTE) gibi yöntemler mevcuttur (Jagatheesan, ve ark., 2018, s.895). Sistemin performansını ölçmek ve iyileştirmelerin yapılabileceği alanları belirlemek için kullanılan mutlak hata ile çarpılan zamanın integrali (ITAE), geri beslemeli bir sistemde belirli bir süre boyunca biriken hatanın PSCAD simülasyon ortamında tasarlanan bu beş

farklı tipteki fonksiyonlar simpleks algoritması ile kullanılmaktadır. Beş fonksiyonun matematiksel ifadesi aşağıda Denklem (3.53 - 3.57) de gösterilmektedir:

Hatanın mutlak değerinin integrali

$$IAE = \int |e| dt \tag{3.53}$$

Karesel hatanın integrali

$$ISE = \int e^2 dt \tag{3.54}$$

Zaman ağırlıklı mutlak hatanın integrali

$$ITAE = \int t|e| dt \tag{3.55}$$

Zaman ağırlıklı karesel hatanın integrali

$$ITSE = \int te^2 dt \tag{3.56}$$

Zamansal ve karesel hatanın integrali

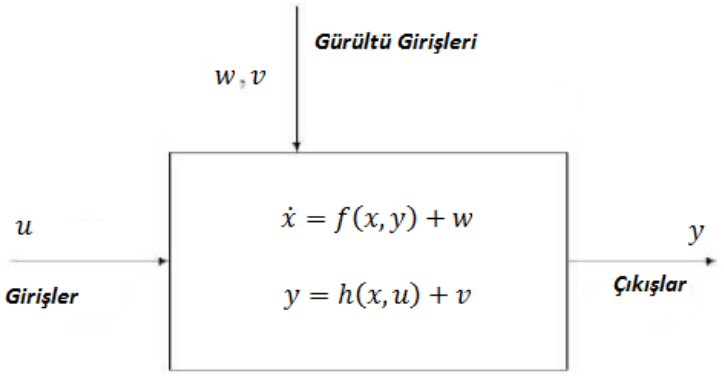
$$ISTE = \int t^2 e^2 dt \tag{3.57}$$

### 3.2.4. Sistem Tanımlama Tipleri

Sistem tanımlamada üç ayrı modelden oluştuğu söylenebilir. Bunlar beyaz kutu, gri kutu ve siyah kutu sistem tanımlama tipleridir (Sanatel, 2020, s.8). Bu üç yarı model aşağıda açıklanmaktadır.

#### 3.2.4.1. Beyaz kutu sistem tanımlama

Beyaz kutu sistem tanımlama, bir sistemin iç yapısını ve çalışma prensiplerini tam olarak bildiğimiz durumlarda kullanılan bir yaklaşımdır. Beyaz kutu sistem tanımlama, sistemi oluşturan her bir bileşenin davranışını ve etkileşimini anlamak için matematiksel denklemleri veya fiziksel yasaları kullanır. Bu şekilde, sistemin giriş ve çıkışları arasındaki ilişkiyi tam olarak ifade eden bir matematiksel model elde edilir. (Fissore, 2021). Şekil 3.15'te beyaz kutu sistem tanımlama modeli gösterilmiştir.




Şekil 3.15. Beyaz kutu sistem parametreleri (Fissore, 2021)

### 3.2.4.2.Gri kutu sistem tanımlama

Gri kutu sistem tanımlama, sistemin iç yapısı ve çalışma prensipleri hakkında sınırlı bilgiye sahip olduğumuz durumlarda kullanılan bir model yaklaşımdır. Bu yöntemde, sistemin iç bileşenleri ve işleyişi hakkında tam bir bilgiye sahip olunmaz, ancak sistemin giriş ve çıkışları arasındaki ilişkiyi belirlemek için deneysel veriler veya gözlemlemelere dayanmaktadır. Gri kutu sistem tanımlama, beyaz kutu sistem tanımlamasına kıyasla daha kısıtlı bir bilgiye dayanır ve sistemin iç yapısı hakkında eksiklikleri olabilir. Ancak, gerçek dünya sistemlerinde beyaz kutu sistem tanımlamasının zor veya mümkün olmadığı durumlarda, gri kutu sistem tanımlama yöntemi değerli bir yöntem olabilmektedir (Fissore, 2021).

### 3.2.4.3.Siyah kutu sistem tanımlama

Siyah kutu sistem tanımlama, sistemin iç yapısı, bileşenleri veya işleyişi hakkında hiçbir bilgiye sahip olunmadığı durumlarda kullanılan bir model yaklaşımdır. Siyah kutu sistem tanımlaması, gerçek dünya sistemlerinin karmaşık olduğu, iç bileşenlerinin veya yapısındaki ticari sırlarının patentlerle korunduğu yada sistemin çalışmasının teknik olarak karmaşık olduğu durumlarda kullanışlı olabilir. Bu yaklaşım, gerçek sistemin iç detaylarına dair bilgiye sahip olmaksızın sistemi analiz etmek ve yönetmek için pratik bir yol sunar (Fissore, 202; Fidan ve Erkan, 2023, s. 210). Sekil 3.16'da siyah kutu sistem tanımlama modeli verilmiştir. Bu tezde bu tanımlama yöntemi temel olarak ele alınmaktadır.


Şekil 3.16. Siyah kutu sistem tanımlama grafiği

Yukarıda belirtilen 3 sistem tanımlama yaklaşılarından en fazla siyah kutu sistem tanımlama modeli yaklaşımı tercih edilmektedir. Siyah kutu sistem tanımlamada derin algoritmalar kullanılmıştır (Fissore, 2021; Fidan ve Erkan, 2023, s. 210).

**Siyah Kutu Modelleri Değişkenleri:** Tanımlanması gereken model Denklem (3.58)'de verilmiştir.

$$\begin{aligned} y(t) = f[&y_1(t-1), \dots, y_1(t-n_{y_1}), \dots, y_r(t-1), \dots, y_r(t-n_{y_r}), \\ &u_1(t-1), \dots, u_1(t-n_{u_1}), \dots, u_m(t-1), \dots, u_m(t-n_{u_m}), \\ &e_1(t-1), \dots, e_1(t-n_{e_1}), \dots, e_r(t-1), \dots, e_r(t-n_{e_r})] \end{aligned} \tag{3.58}$$

**Siyah Kutu Modelleri (Regressors):** Ø dizisini tanımlayalım, regressors olarak adlandırılmış bileşenler:

$$\begin{aligned} &\varphi(t) = [y_1(t-1), \dots, y_1(t-n_{y_1}), \dots, y_r(t-1), \dots, y_r(t-n_{y_r}), \\ &u_1(t-1), \dots, u_1(t-n_{u_1}), \dots, u_m(t-1), \dots, u_m(t-n_{u_m}), \\ &e_1(t-1), \dots, e_1(t-n_{e_1}), \dots, e_r(t-1), \dots, e_r(t-n_{e_r})]^T \end{aligned} \tag{3.59}$$

Eğer *d* sistemin toplam değişken sayısı, ardından boyut *I* dizinin olarak *Ø* genel sipariş toplamı olarak tanımlanır.

$$\ell = \sum_{i=1}^{d} n_i \tag{3.60}$$

Girdi değişkenlerinin çıktı değişkenleri üzerinde gecikmeli bir etkiye sahip olabileceği göz önüne alındığında, gecikme süreleri, $n_{ki}$, süreç modelinde her girdi değişkeni için:

$$\varphi(t) = [y_1(t-1), \dots, y_1(t-n_{y_1}), \dots, y_r(t-1), \dots, y_r(t-n_{y_r}), u_1(t-n_{k_1} - 1), \dots, u_1(t-n_{k_1}-n_{u_1}), \dots, u_m(t-n_{k_m}-1), \dots, u_m(t-n_{k_m} - n_{u_m})\ e_1(t-1), \dots, e_1(t-n_{e_1}), \dots, e_r(t-1), \dots, e_r(t-n_{e_r})]^T \quad (3.61)$$

**Siyah Kutu Modelleri İşlevi:**

$p$ parametrelerini kullanarak $y$ çıkış değişkenlerine $f$ işlev regresörlerini $\emptyset$ dizisini eşleyebilir. Denklem (3.62) de $y$ çıkış ın matematiksel ifadesi verilmiştir.

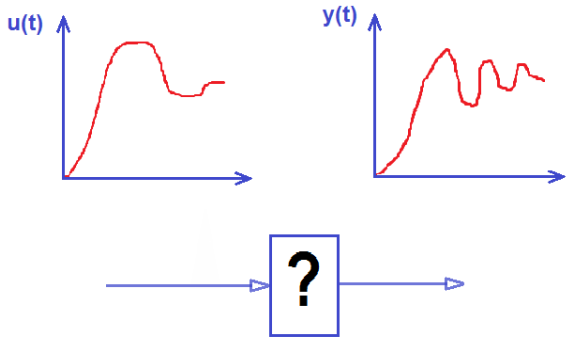
$$y(t) = f[\varphi(t), p] \quad (3.62)$$

Doğrusal ve doğrusal olmayan fonksiyonlardan her ikisini de kullanmak mümkündür. Fonksiyon için en basit modeli Denklem (3.63) te ifade edilmektedir:

$$y(t) = p * \varphi(t) \quad (3.63)$$

### 3.2.5. Deterministik ve Ampirik Modeller

Deterministik modeller, sürecin derin bilgisinden elde edilmektedir. Birinci ilkeye dayalı modeller olarak kütle, enerji, momentum (kuvvet) korunum yasalarından oluşmaktadır. Bu modeller genellikle diferansiyel denklemler, fark denklemleri, cebirsel denklemler ve mantıksal ilişkiler aracılığıyla ifade edilir. Başlangıç ve sınır koşulları göz önüne alındığında, tahmin edilmesi gelecekte sistemin evrimi açısından önem arz etmektedir (Fissore, 2021, s.2). Ampirik modeller, bir sistemde meydana gelen ısı, kütle ve momentum transfer süreçlerini dikkate almazlar, ancak süreç hakkında bilgi almak için deneyler ve ölçümler yapılmaktadır. Çoğu durumda sistemin girdisi ve çıktısı arasında matematiksel bir ilişkiden oluşmaktadır. Şekil 3.17' de modelin girdi ve çıktı grafiği gösterilmektedir.

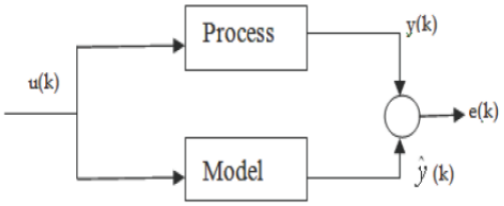



Şekil 3.17. Modelin girdi ve çıktı grafiği

$$y(k) = ay(k-1) = bu(k-1) \tag{3.64}$$

İşlem, bir giriş sinyalinden *(u)* ve bir çıkış sinyalinden *(y)* oluşur. Burada iki bilinmeyen olarak *a* ve *b* parametreleri mevcuttur. Deklem 3.64 te matematiksel ifadesi belirtilmiştir.

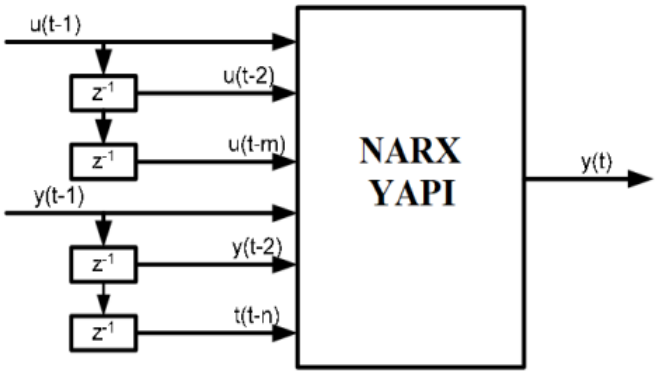
Şekil-3.18'de Model çıktısının mümkün olduğu kadar iyi süreç çıktısı gibi görünmesini istiyoruz. Süreç ve model çıktıları arasındaki fark hata *e*, parametrelerin değerlerini bulmak ve minimize etmek için *a* ve *b* ölçüleri olacaktır.




Şekil 3.18. Model akış diyagramı (Fissore, 2021).

### 3.2.6. Sistem Tanımlamada NARX Modeli

NARX modeli, sistem tanımlama için kullanılan bir matematiksel modeldir. Bu model, nonlineer sistemlerin modellenmesinde etkili bir araç olarak kullanılır. NARX modeli, özellikle tahmin, kontrol ve sistem tanımlama gibi uygulamalarda kullanılır. Sistemin dinamik yapısını ve giriş-çıkış ilişkilerini anlamak için NARX modeli kullanılarak sistemin matematiksel bir temsili elde edilebilir. Bu temsil, sistem analizi, tahmin, simülasyon veya kontrol amaçlarıyla kullanılabilir. Şekil 3.19'da NARX sistem için bir üretici örnegi verilmiştir (Fissore, 2021, s.7).




Şekil 3.19. NARX işaret akış diyagramı (Fissore, 2021).

NARX modeli yaklaşımıda bir sistem tasarımındaki *t+1* anındaki çıkışı şekilde belirtilen modelde açıklanmaktadır (Fissore, 2021, s.8). Örnek bir NARX modelinin matematiksel ifadesi Denklem (3.65)' te verilmiştir:

$$y(t+1) = F(u(t), u(t-1), u(t-2), \dots, y(t), y(t-1), y(t-2), \dots.) \tag{3.65}$$

NARX modelleri kullanılarak sistemin daha önceki değerlerinden sonraki değerleri bulunabilmektedir. Bu çalışma şeklinden ötürü NARX modelleri genellikle makine ögrenmesi algoritmalarında kullanılması oldukça elverişli olmaktadır.

### 3.2.7. Arx ve Armax Modeli:

ARX modeli, doğrusal olmayan sistemlerin özelliklerini tanımlayamaz. Bir tahmin elde etmek için gereken süre çok kısadır.

$$\begin{aligned} y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_y} y\left(t - n_y\right) \\ = b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_u} y(t - n_u) \end{aligned} \tag{3.66}$$

Armax Modeli, hata terimlerinin varlığı sayesinde e, bu modelin doğruluğunun bir ARX modelinden daha yüksek olması beklenmektedir. Ayrıca, belirli bir dereceye kadar, hata terimi prosesin doğrusal olmama durumunu, ölçülemeyen bozulmaları, gürültülü ölçümlerini hesaba katabilmektedir.

$$\begin{aligned} y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_y} y\left(t - n_y\right) \\ = b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_u} y(t - n_u) \\ + d_1 e(t-1) + d_2 e(t-2) + \cdots + d_{n_e} y(t - n_e) \end{aligned} \tag{3.67}$$

### 3.2.8. Box-Jenkins modeli:

Toplamsal bozulmaya sahip doğrusal bir sistem *v(k)* şu şekilde tanımlanabilir:

$$y(k) = G(z)u(k) + v(k) \tag{3.68}$$

*G(z)* bir transfer fonksiyonu;

$$G(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a}} \tag{3.69}$$

$Z^{-m}\, x(k) = x(k-m)$, dikkate alınarak sistem bir fark denklemi kullanılarak tanımlanabilir:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \cdots - a_{n_a} y(k-n_a) + b_0 u(k) + b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + v(k) \quad (3.70)$$

*v(k)* beyaz bir gürültü veya renkli bir gürültü olabilir:

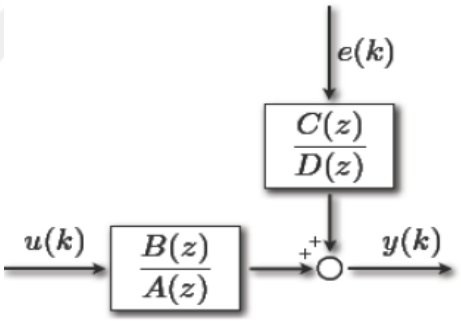
$$v(k) = H(z)e(k) \quad (3.71)$$

*e(k)* beyaz bir gürültü ve *H(z)* başka bir aktarım işlevinin matematiksel ifadesi:

$$H(z) = \frac{C(z)}{D(z)} = \frac{1 + c_1 z^{-1} + \cdots + c_{n_c} z^{-n_c}}{1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{n_d} z^{-n_d}} \quad (3.72)$$

Nihai olarak Box-Jenkins modeli genel olarak Denklem (3.73)'teki gibi gösterilir. Şekil 3.20 modelin genel görünümünü vermektedir.

$$y(k) = \frac{B(z)}{A(z)} u(k) + \frac{C(z)}{D(z)} e(k) \quad (3.73)$$



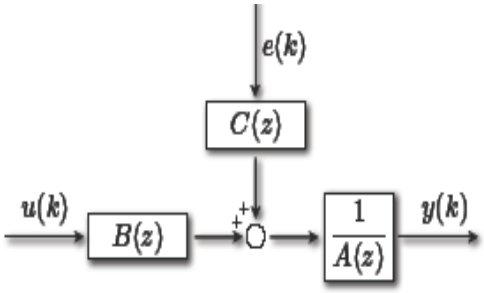

Şekil 3.20. Box-jenkins Modeli (Fissore, 2021).

Şekil 3.20' de belirtilen Box-Jenkins modeli, zaman serisi verilerinin analizinde yaygın olarak kullanılan bir yöntemdir. Bu model, gelecekteki değerleri tahmin etmek, mevcut verileri kontrol etmek veya gelecekteki trendleri belirlemek için kullanılabilir. Şekil 3.21' de ARMAX modeli, sistem tanımlama ve kontrol teorisi alanında yaygın olarak kullanılan bir lineer regresyon modelidir. Bu model, bir sistemin davranışını girdi ve çıktı verilerine dayanarak açıklamak için kullanılır.

*G(z)* ve *H(z)* aynı paydaya sahip ise Denklem (3.74) uygulanır.

*(A(z) =D(z))*

$$A(z)y(k) = B(z)u(k) + C(z)e(k) \tag{3.74}$$




Şekil 3.21. Armax Modeli (Fissore, 2021).

Burada;

*y(k)*: Bağımlı değişkenin (zaman serisi) mevcut değerini temsil eder

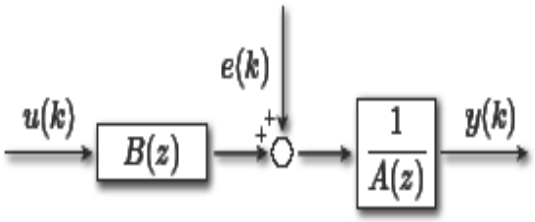
*e(k)*: Dışsal değişkenin mevcut değerini temsil eder.

*u(k)*: Mevcut giriş değerini temsil eder

ARMAX modelinde, bir sistemin mevcut çıktısı, geçmiş çıktı değerleri ile mevcut ve geçmiş girdi değerlerinin lineer kombinasyonu olarak modelleme yapılır. Model, giriş ve çıktı değişkenleri arasında lineer bir ilişki olduğunu varsayar. ARMAX modeli, sistem tanımlama için yaygın olarak kullanılırken, amaç giriş-çıktı verilerine dayanarak modelin bilinmeyen parametrelerini (katsayılarını) tahmin etmektedir. Model parametreleri belirlendikten sonra, ARMAX modeli tahmin, kontrol veya sistem davranışının analizi için kullanılabilir. Şekil 3.22 de ARX modeli, sistem tanımlama ve kontrol teorilerinde yaygın olarak kullanılan bir lineer regresyon modelidir. Bu model, bir sistemin davranışını giriş ve çıkış verilerine dayanarak açıklamak için kullanılır.

*C(z) = I* durumunda Denklem (3.75) uygulanır.

$$A(z)y(k) = B(z)u(k) + e(k) \tag{3.75}$$




Şekil 3.22. Arx Modeli (Fissore, 2021).

Burada;

*y(k*): Bağımlı değişkenin (zaman serisi) mevcut değerini temsil eder

*e(k):* Dışsal değişkenin mevcut değerini temsil eder.

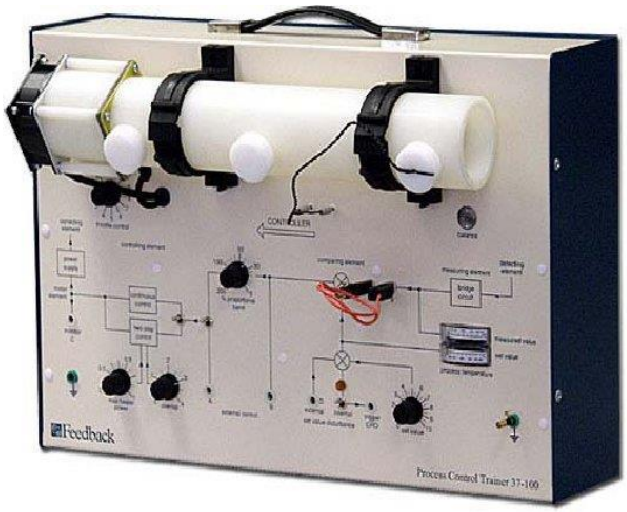
*u(k):* Mevcut giriş değerini temsil eder

ARX modeli genellikle sistem tanımlama için kullanılır, burada hedef, giriş-çıkış verilerine dayanarak modelin bilinmeyen parametrelerini (katsayıları) tahmin etmektir. Model parametreleri belirlendikten sonra, ARX modeli tahmin, kontrol veya sistem davranışının analizi için kullanılabilir. ARX modeli, giriş ve çıkış değişkenleri arasında lineer bir ilişki varsaydığı için karmaşık olmayan lineer sistemlerin modellenmesi için uygundur. ARX modeli karmaşık ve doğrusal olmayan dinamikleri yakalayamaz. Bu durumlarda, nonlineer ARX (NARX) modeli veya diğer nonlineer sistem tanımlama teknikleri gibi daha gelişmiş modeller kullanılabilir (Fissore, 2021, s.14).

### 3.3. Donanımsal Özelikler

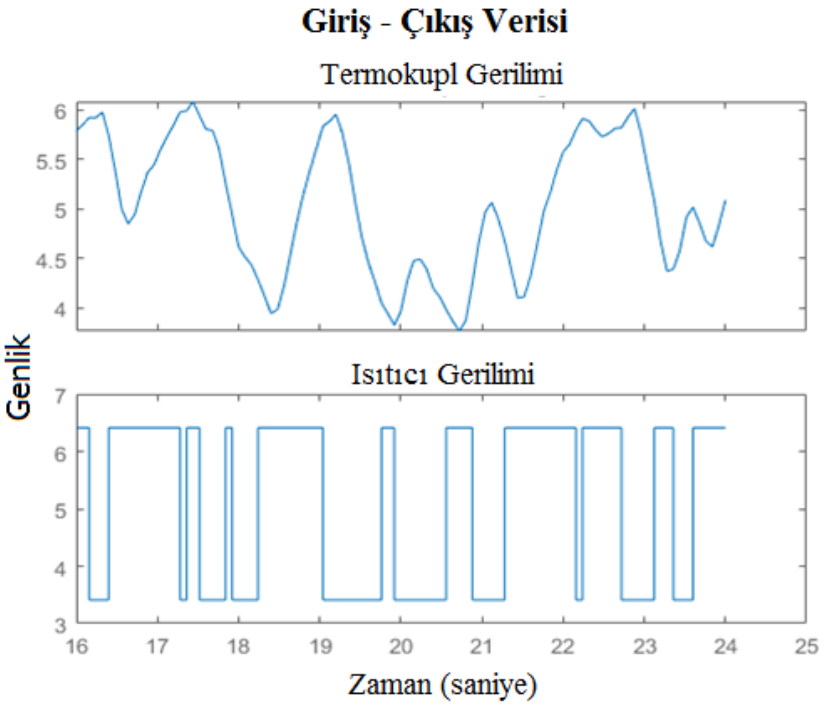
Sistem modelinin elde edilmesi ve AEO algoritmalarının geliştirilmesi aşamasında 2.5 GHz hızında 6 çekirdekli Intel Core i5-3210M CPU, 6 Gb RAM ve 1 TB SSD özelliklerine sahip bir PC kullanılmıştır. Yazılımların geliştirme aşamasında Python 3.8 sürümü ile birlikte mealpy 2.5.0 (Van Thieu ve Mirjalili, 2023, s.102871), pandas 1.4.2, control 0.9.2 kütüphaneleri kullanılmıştır. IDE olarak Spyder ve Jupyter Notebook programları kullanılmıştır. Çizimler için kulanılan temel Jupyter Notebook kodları Ek 1'de verilmiştir.

Deney setinin verileri Matlab içinde bulunan bir veri seti kullanılarak temin edilmiştir. Bu veriler; Şekil 3.23'te gösterilen laboratuar ölçeğinde "hairdryer" deney setinden (Feedback's Process Trainer PT326) elde edilmiştir. Bu processde bir tüp içinden geçirilen hava, ağız kısmında ısıtılır ve tüp çıkışına bağlanan bir termokupl sayesinde hava sıcaklığı ölçülür. Girişe konumlandırılan ısıtıcıya uygulanan gerilim giriş sinyali olarak tanımlanır. Çıkış ise termokupl aracılığıyla ölçülen düşük seviyeli voltaja karşılık gelen hava sıcaklığıdır.




Şekil 3.23. Feedback's process trainer pt326 deney seti görünümü

Çıkış verisi, çıkış hava akımındaki sıcaklıkla orantılı olan termokupl voltajının 1000 adet ölçümünü içerir. Giriş gerilimi bir seviyeden diğerine rastgele olarak geçecek şekilde örnekleme yapılmıştır. Ölçüm düzeneğinin örnekleme süresi 0.08 saniyedir. Elde edilen verilerin %80'i modelin eğitiminde %20'si ise test aşamasında kullanılmıştır. Şekil 3.24'te ısıtıcı girişine uygulanan gerilime karşılık çıkışta ölçülen termokupl gerilimini göstermektedir.




Şekil 3.24. Isıtıcı girişine uygulanan gerilime karşılık çıkışta ölçülen termokupl gerilimi

# 4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu tez çalışmasında belirlenen meta-sezgisel algoritmalar sistem tanımlama problemlerinin çözümü için uygulanmıştır. Öncelikle $R^2$ performans değerleri hesaplanmış ve karşılaştırmalar yapılmıştır. Ancak meta-sezgisel algoritmaların testleri sırasında rastgele başlangıç değerlerinden kaynaklı olarak $R^2$ performans değerlerinde sürekli değişiklikler olduğu görülmüştür. Bu sebeple her algoritma 100'er defa çalıştırılmış ve $R^2$ değerlerinin değişimi gözlemlenmiştir. Bu sayede her algoritmanın ne kadar tutarlı sonuçlar verdiği ile ilgili yorumlar yapılmıştır.

Meta-sezgisel algoritmalar temelde bir problemi çok sayıdaki döngülerden oluşan kodlar sayesinde çözmektedir. Bu tez çalışmasında önerilen algoritmaların çeşitli sınırlılıklar altında çalışması durumunda da performansı incelenmiştir. Bu sınırlılıklar zaman sınırlılığı performansı, maksimum jenerasyon sınırlılığı performansı, fonksiyon hesaplama sınırlılığı ve erken durdurma sınırlılığı kriterlerine bağlı olarak ifade edilebilir. Bu yapılan analizler sayesinde bir algoritmanın tek bir sefer çalıştırılması durumunda ne kadar güvenli sonuç verdiği bu bölümde sunulmuştur.

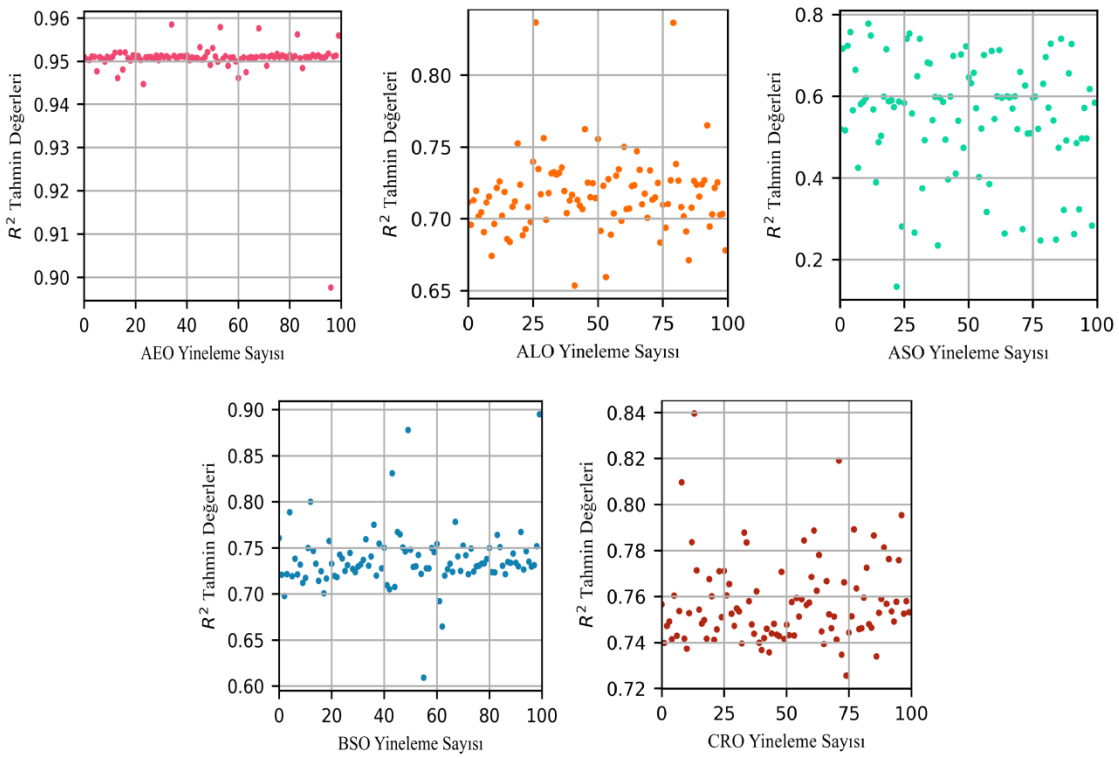
## 4.1. Güvenilirlik Analizi ve Sınırlılıklar

Performansa odaklanan birçok çalışmada durdurma kriterlerine çok önem verilmemiş olmasına rağmen durdurma kriterlerini belirleyen sınır değerlerinin uygun seçilmesi, sistem modelinin tahmininde oldukça kritik bir öneme sahiptir. Özellikle kısıtlı kaynaklara sahip gömülü donanımlar için minimum sürede çözüme ulaşmak oldukça önemlidir (Ravber ve ark., 2022, s.109478). Bu tez de meta-sezgisel algoritmaların performansının incelenmesinin yanında farklı durdurma kriterlerine bağlı olarak elde edilen sonuçları da karşılaştırılmıştır. Bu çalışmada algoritmaların güvenilirliğini elde etmek için her iterasyon sonucunda elde edilen $R^2$ değerinin 0.8 ve üstünde olması durumda performansın yeterli olduğu anlaşılmaktadır.

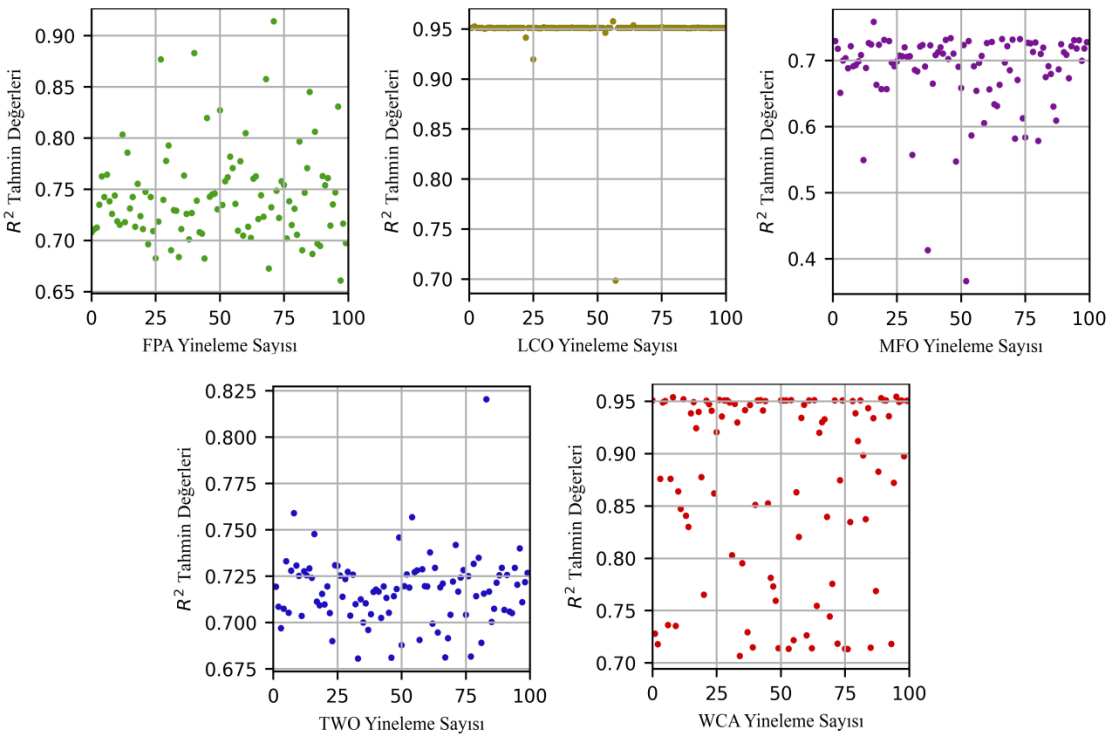
### 4.1.1. Zaman Sınırlılığı Performansı

Ele alınan algoritmalar maksimum zaman sınırlılığı belirlenerek (45 saniye) test edilmiştir. Burada temel amaç her bir sonuç elde edilirken 45 saniye içinde nasıl bir $R^2$ değerine ulaştığını göstermektir. Elbette bu test algoritmanın 100 defa çalıştırılmasına yani güvenilirliğinin belirlenmesine engel değildir. Bu test sonucunda çıkan performans

değerleri Şekil 4.1 ve Şekil 4.2 de verilmiştir. AEO algoritması bu karşılaştırma sonucunda 0,94 – 0,96 değerleri arasında en yüksek ve kararlı $R^2$ değerlerine ulaştığı görülmüştür. Bu da AEO algoritmasının performansının başarılı olduğunu elde edilen $R^2$ değerlerinin birbiriyle tutarlı olduğunu göstermiştir. Bu durumda bu algoritmanın güvenilir olduğu ortadadır. LCO algoritmasında, çoğunluk $R^2$ değerleri 0,95 etrafında toplanmış ise de bir aykırı değerin 0,70 civarında görülmesinden dolayı %98 oranında güvenli olduğu belirlenmiştir. Diğer algoritmalar ise belirlenen kriterler dikkate alındığında başarısız olmuştur. Bu başarısızlığın sebebi özellikle başlangıç değerlerinin iyi seçilmemesi veya algoritmalara özgü giriş değerlerinin doğru girilmemiş olması olabilir. Bunun araştırılması bu tez çalışmasında yapılmamıştır. Çünkü başlangıç değerlerinin veya optimal giriş değerlerinin doğru belirlenmesi başlı başına bir çalışma konusudur. Sunulan şekiller algoritma sayısının fazla olması, 100 iterasyon boyunca performans belirlenmesi gibi sebeplerden dolayı iki kısma ayrılarak sunulmuştur.



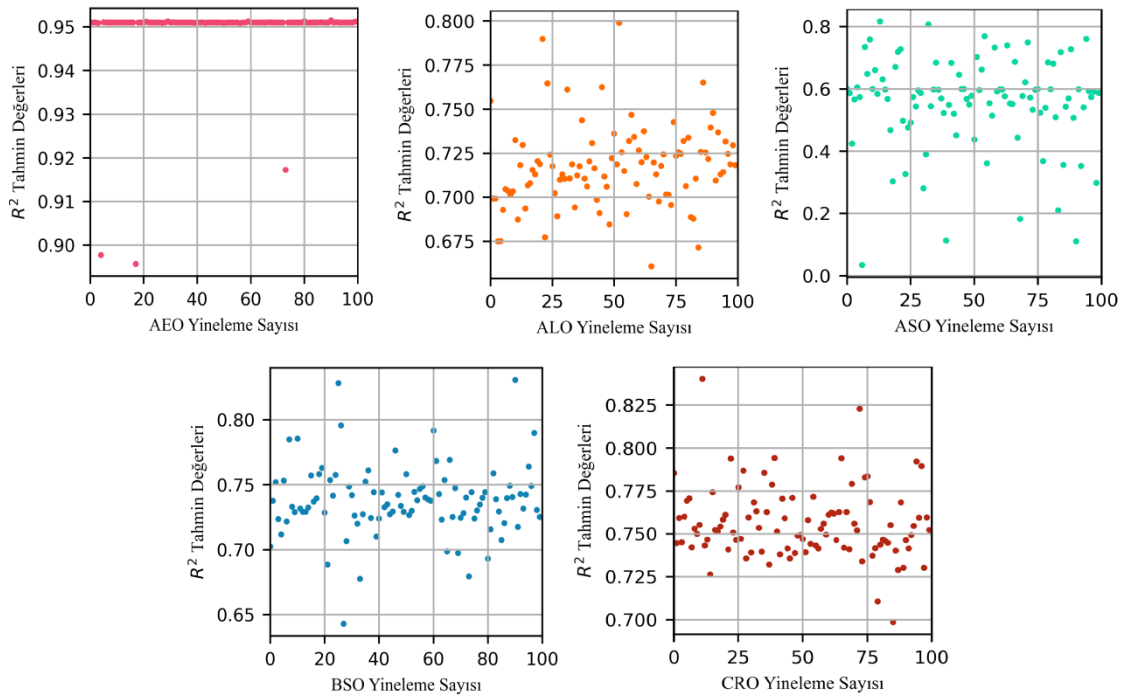

Şekil 4.1. Zaman sınırlılığı altında $R^2$ performans grafikleri-1



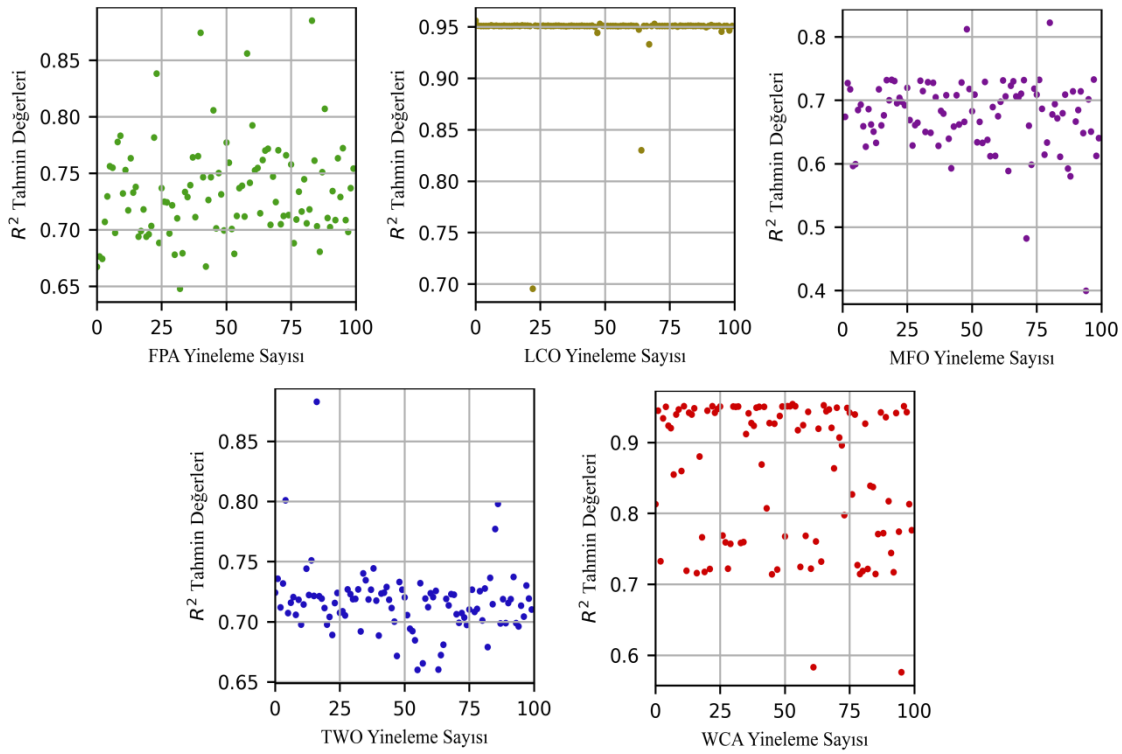

Şekil 4.2. Zaman sınırlılığı altında $R^2$ performans grafikleri -2

### 4.1.2. Maksimum Jenerasyon Sınırlılığı Performansı

Maksimum jenerasyon sayısı, algoritmanın kaç kez popülasyonun yeniden oluşturulacağını ve geliştirileceğini belirler. Her jenerasyonda, çeşitli meta-sezgisel operatörler (seçim, çaprazlama, mutasyon vb.) kullanılarak popülasyon güncellenir ve yeni çözümler üretilir. Bu süreç, algoritmanın potansiyel çözümleri keşfetmesini ve daha iyi çözümler bulması için iteratif olarak çalışmasını sağlar. Maksimum jenerasyon sayısı (30) dikkate alınarak hesaplanan $R^2$ değerleri Şekil 4.3 ve Şekil 4.4' te belirtilmiştir. AEO algoritması bu sınırlılık altında yine 0,91 – 0,95 değerleri arasında $R^2$ değerlerine ulaştığı tespit edilmiştir. Bu durumda yine AEO algoritmasının güvenilir olduğu söylenebilir. LCO algoritması yaklaşık 0,88 – 0,94 aralığında $R^2$ değerlerine ulaştığı görülmüştür. Bu durumda LCO algoritması da aykırı değere rağmen %98 oranla güvenilir olarak ifade edilebilir. %98 oran belirtilmesinin sebebi 100 iterasyonda sadece 2 değer 0,8 in altında değer almıştır. Diğer bir algoritma ise WCA kısmen başarılı olarak kabul edilebilir. Diğer algoritmalar ise sıklıkla 0.8 değerinin altına düştüğü için güvenilir bulunmamıştır.



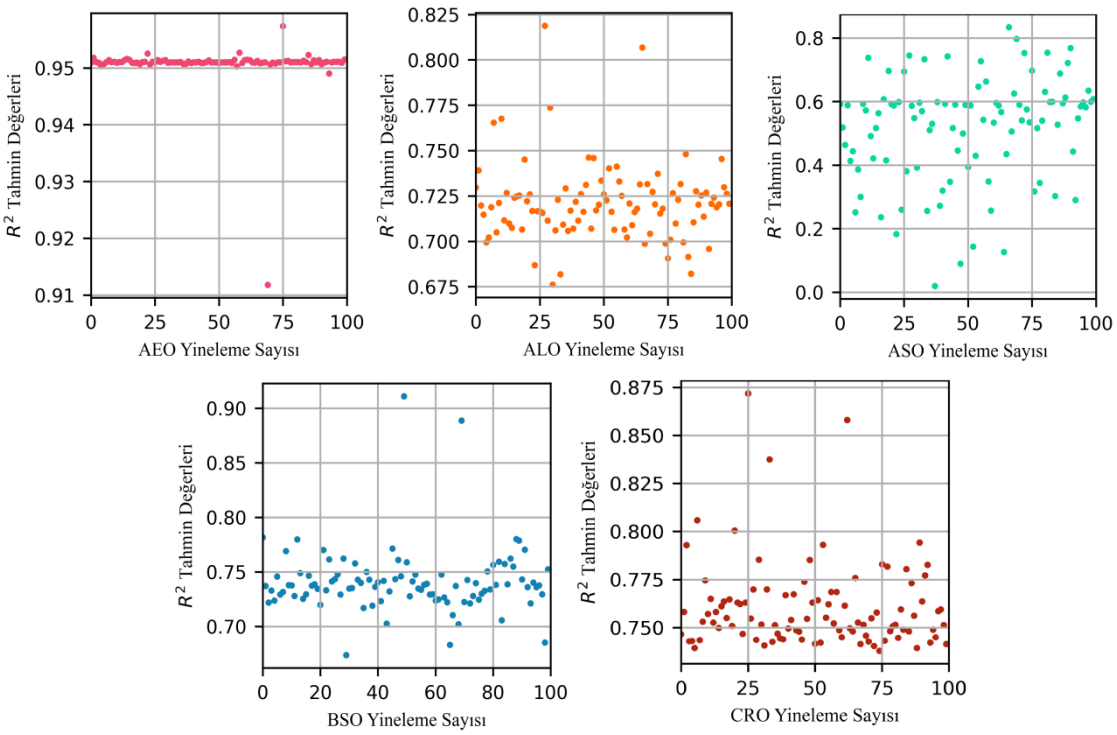

Şekil 4.3. Maksimum jenerasyon sınırlılıkları altında $R^2$ performans grafikleri -1



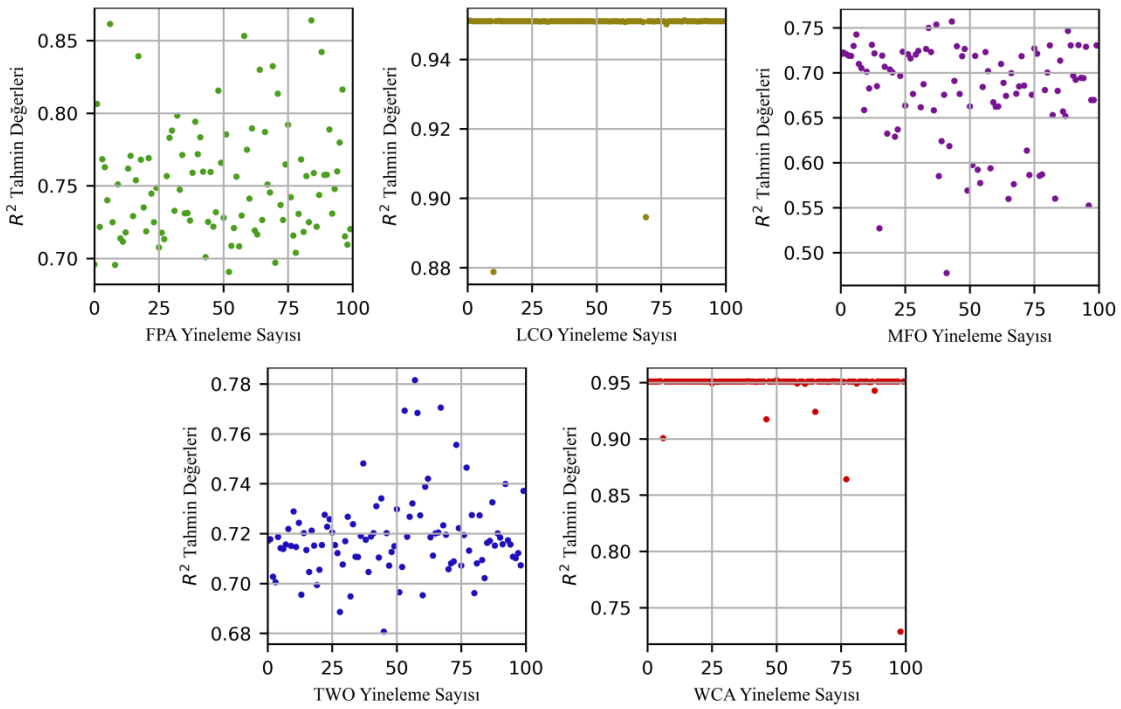

Şekil 4.4. Maksimum jenerasyon sınırlılıkları altında $R^2$ performans grafikleri -2

### 4.1.3. Fonksiyon Hesaplama Sınırlılığı

Meta-sezgisel algoritmalar, genellikle büyük ve karmaşık problemlerde çalıştığından, fonksiyonların kaç defa çalıştırılacağı hesaplama açısından maliyetli olabilir. Fonksiyon değerlendirme sayısı, algoritmanın performansını, çözüm kalitesini ve çalışma süresini etkiler. Algoritmanın amaçlarına ulaşmak için yeterli sayıda fonksiyon değerlendirmesi yapılması önemlidir. Bu sebeple fonksiyon hesaplama sınırlılığı dikkate alınarak yapılan hesaplamalarda algoritmalar 4000 iterasyonla çalıştırılarak performans sonuçları incelenmiştir. Maksimum fonksiyon hesaplama kriteri dikkate alınarak elde edilen $R^2$ değerleri Şekil 4.5 ve Şekil 4.6' da verilmiştir. Güvenilirlik belirlenirken algoritmaların 100 iterasyon boyunca 0.8 $R^2$ değerini geçmesi dikkate alınmıştır. Bu durumda AEO algoritması 100 çalıştırmanın tamamında 0.8 üzeri $R^2$ performansına sahip olmuştur. LCO ve WSA algoritmalarını da bu sınırlılık altında güvenli olarak ifade etmek mümkündür. ASO algoritması diğer sınırlamalar altında olduğu gibi halen en düşük performanslara sahip algoritma olarak karşımıza çıkmaktadır.



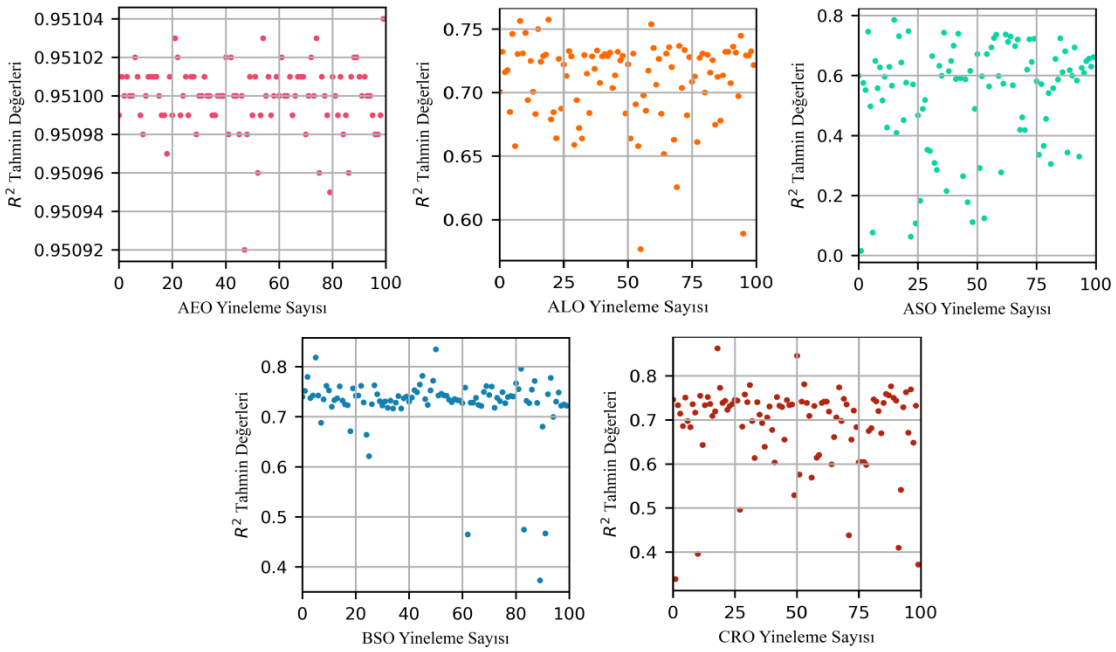

Şekil 4.5. Fonksiyon hesaplama sınırlılığı altında $R^2$ performans grafikleri-1



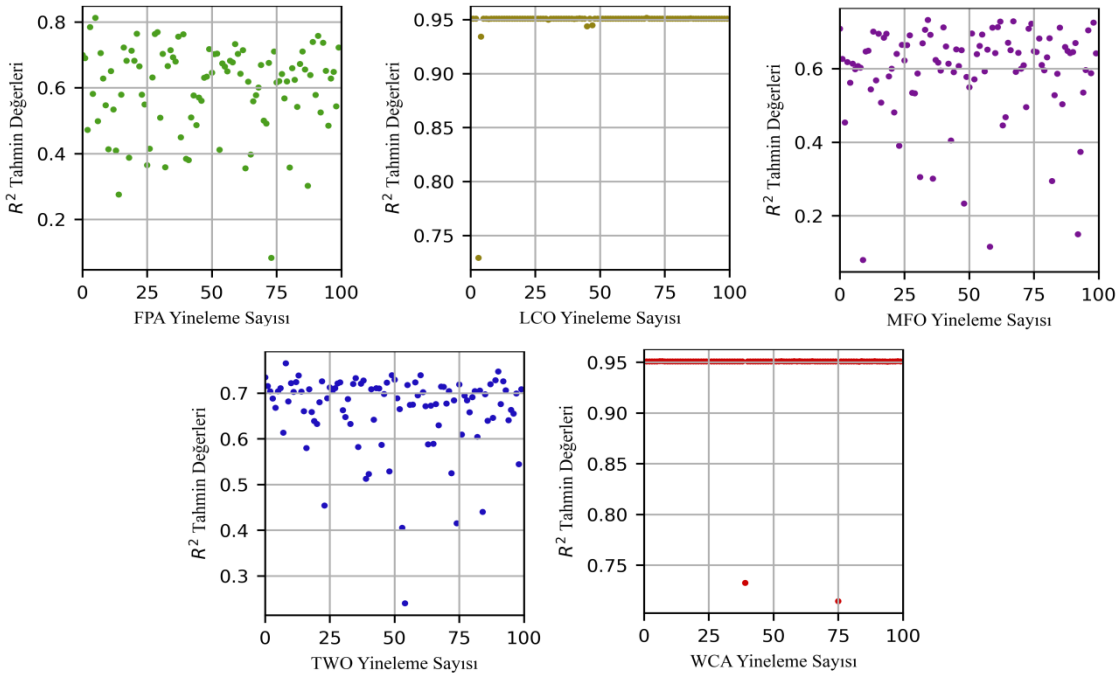

Şekil 4.6. Fonksiyon Hesaplama sınırlılığı altında $R^2$ performans grafikleri-2

### 4.1.4. Erken Durdurma Sınırlılığı

Erken durdurma, algoritmaların optimizasyon sürecini sonlandırma kriterlerine dayanarak durmasını sağlar. Bu kriterler, belirli bir durumu, hedef değeri veya performansı temsil edebilir. Örneğin, algoritmanın en iyi çözüme ulaşması durumunda, belirli bir süre veya iterasyon sayısı aşıldığında, belirli bir hedef değeri elde edildiğinde veya performans artık iyileşmediğinde erken durdurma uygulanabilir. Maksimum erken durdurma (Erken durdurma-Early stopping) kriteri, meta-sezgisel algoritmalarının hesaplanan minimum hata değerinin 3 döngü boyunca hiç değişmemesi sonunda araştırma sürecinin bitirilmesi olarak tanımlanabilir. Bu noktada hata değerinin hiç değişmemesi hatanın 10e-6 değerinin altına 3 defa inmesi erken durdurma olarak ifade edilebilir. Bu kriter algoritmalara çözümü bulmak için oldukça uzun süre vermektedir. Şekil 4.7 ve Şekil 4.8 erken durdurma sınırlılığı altında önerilen algoritmaların performansını göstermektedir. Belirtilen 10 algoritma için bu şekiller incelendiğinde AEO algoritmasının 100 iterasyon sonunda global çözüm noktasını her seferinde belirlemeyi yüksek bir performansla başarmıştır. Bu sebeple bu algoritma bir kez daha güvenilir olarak ifade edilebilmektedir. Diğer sınırlılıklarda olduğu gibi LCO ve WCA algoritmaları da her ne kadar 1-2 aykırı değerin 0,8 eşik değerinin altına inmişse de %98 oranla başarılı kabul edilmektedir.




Şekil 4.7. Erken durdurma sınırlılığı altında $R^2$ performans grafikleri-1




Şekil 4.8. Erken durdurma sınırlılığı altında $R^2$ performans grafikleri -2

Erken durdurma kriteri algoritmalara bir problemin çözümü için gerektiği kadar süre vermektedir. Bu durumda algoritmaların problem çözümlerinde ne kadar oyalandığını görmek performansları incelemek adına diğer önemli bir kriterdir. Elde edilen çözüm süreleri: AEO algoritması için 138 saniye, ALO için 13 sn, ASO için 8 sn, BSO için 41 sn, CRO için 7 sn, FPA için 10 sn, LCO için 79 sn, MFO için 8 sn, TWO

için 13 sn ve WCA için 215 sn olarak hesaplanmıştır. Karşılaştırmalar sonucunda - AEO ve WCA algoritmaları yüksek çözüm sürelerine ulaşmıştır. Ancak aynı algoritmalar zaman sınırlılıkları altında da oldukça iyi performans göstermiştir. Özellikle AEO algoritması her sınırlılık altında yüksek bir başarı göstermiştir.

### 4.2. Farklı Sınırlamalar Altında Performans İndekslerinin Karşılaştırılması

Zaman sınırlılığı temel olarak performans sonucuna bakılmaksızın verilen süre sonunda algoritmanın durdurulmasını temel almaktadır. Bu çalışmada verinin doğası, algoritmaların yapısı ve donanımsal kaynaklar dikkate alınarak süre sınırlaması 45 sn olarak belirlenmiştir. Bu çizelgelerde 10 farklı algoritma 100 iterasyon çalışması sonunda 10. döngüde çıkan sonuçları sunulmuştur. Burada döngü değeri 10 rastgele olarak seçilmiştir. Bu durumda Çizelge 4.1'de verilen transfer fonksiyonları ve performans değerleri elde edilmiştir. AEO ve LCO algoritmaları ile hesaplanan transfer fonksiyonları performans açısından birbirlerine oldukça yakındır.

Çizelge 4.1. Zaman sınırlılıkları altında performansların karşılaştırılması

| Algoritmalar | MAPE | MAE | MSE | $R^2$ | Süre | Transfer Fonksiyonları |
|---|---|---|---|---|---|---|
| AEO | 0,02756 | 0,13242 | 0,02758 | 0,95066 | 45,84 | $T_f(s) = \frac{-1.47s + 12.54}{s^2 + 6.08s + 12.84}$ |
| ALO | 0,06955 | 0,33607 | 0,16969 | 0,6964 | 46,71 | $T_f(s) = \frac{-9.54s + 85.50}{s^2 + 54.19s + 100}$ |
| ASO | 0,0779 | 0,37715 | 0,22828 | 0,59156 | 45,18 | $T_f(s) = \frac{-5.61s + 96.89}{s^2 + 42.43s + 98.98}$ |
| BSO | 0,06738 | 0,32542 | 0,15798 | 0,71734 | 45,64 | $T_f(s) = \frac{-7.31s + 59.83}{s^2 + 35.27s + 61.57}$ |
| CRO | 0,06742 | 0,3258 | 0,15915 | 0,71525 | 45,42 | $T_f(s) = \frac{-6.69s + 49.94}{s^2 + 32.19s + 51.38}$ |
| FPA | 0,06715 | 0,32299 | 0,15725 | 0,71865 | 45,22 | $T_f(s) = \frac{-8.13s + 51.11}{s^2 + 25.64s + 53.43}$ |
| LCO | 0,02748 | 0,13216 | 0,02744 | 0,95091 | 45,99 | $T_f(s) = \frac{-1.50s + 12.64}{s^2 + 6.13s + 12.94}$ |
| MFO | 0,06848 | 0,32848 | 0,16827 | 0,69894 | 46,50 | $T_f(s) = \frac{-14.40s + 95.75}{s^2 + 56.95s + 97.02}$ |
| TWO | 0,06679 | 0,32051 | 0,15363 | 0,72514 | 46,89 | $T_f(s) = \frac{-10.99s + 97.93}{s^2 + 60.13s + 99.55}$ |
| WCA | 0,04547 | 0,22646 | 0,07607 | 0,8639 | 45,17 | $T_f(s) = \frac{-1.42s + 12.38}{s^2 + 5.89s + 12.67}$ |

Maksimum jenerasyon sınırlandırma kriterinde, çalışmakta olan algoritma, verilen maksimum nesil sayısına ulaştığında durdurulur. Çizelge 4.2'de maksimum jenerasyon sınırlılığı 30 olarak seçildikten sonra hesaplanan parametreler görülmektedir. Bu çizelgede 10 farklı algoritma 100 iterasyon çalışması sonunda 10. döngüde çıkan sonuçları sunulmuştur. AEO ve LCO algoritmaları ile hesaplanan transfer fonksiyonları birbirlerine oldukça yakın olmakla birlikte MAPE, MAE, MSE ve $R^2$ performans göstergeleri de birbirine oldukça yakındır. Çizelgelerde son sütunda performanslar haricinde elde edilen transfer fonksiyonları da sunulmuştur.

Çizelge 4.2. Jenerasyon sınırlılıkları altında performansların karşılaştırılması

| Algoritmalar | MAPE | MAE | MSE | $R^2$ | Süre | Transfer Fonksiyonları |
|---|---|---|---|---|---|---|
| AEO | 0,02746 | 0,13207 | 0,0274 | 0,95097 | 84,43 | $T_f(s) = \frac{-1.49s + 12.59}{s^2 + 6.11s + 12.90}$ |
| ALO | 0,06537 | 0,31334 | 0,14953 | 0,73247 | 50,80 | $T_f(s) = \frac{-11.24s + 72.81}{s^2 + 44.67s + 74.15}$ |
| ASO | 0,07774 | 0,3748 | 0,22521 | 0,59707 | 53,73 | $T_f(s) = \frac{-6.20s + 82.48}{s^2 + 78.25s + 83.72}$ |
| BSO | 0,05846 | 0,28123 | 0,11983 | 0,78561 | 43,16 | $T_f(s) = \frac{-8.58s + 66.96}{s^2 + 39.62s + 68.70}$ |
| CRO | 0,06675 | 0,3208 | 0,1586 | 0,71624 | 23,11 | $T_f(s) = \frac{-6.57s + 84.63}{s^2 + 51.25s + 85.43}$ |
| FPA | 0,06523 | 0,31444 | 0,14966 | 0,73223 | 42,27 | $T_f(s) = \frac{-9.01s + 88.97}{s^2 + 50.53s + 91.49}$ |
| LCO | 0,02747 | 0,13211 | 0,02741 | 0,95096 | 42,46 | $T_f(s) = \frac{-1.58s + 12.96}{s^2 + 6.30s + 13.29}$ |
| MFO | 0,07004 | 0,33677 | 0,17552 | 0,68596 | 42,53 | $T_f(s) = \frac{-8.08s + 100}{s^2 + 56.97s + 100}$ |
| TWO | 0,06975 | 0,33695 | 0,16892 | 0,69778 | 48,00 | $T_f(s) = \frac{-12.43s + 92.57}{s^2 + 55.36s + 95.87}$ |
| WCA | 0,04664 | 0,22546 | 0,07829 | 0,85992 | 42,07 | $T_f(s) = \frac{-1.43s + 12.93}{s^2 + 6.87s + 13.27}$ |

Fonksiyon hesaplama sınırlandırma kriterinde, çalışmakta olan algoritma, verilen maksimum fonksiyon değerine ulaştığında durdurulmaktadır. Bu durdurma kriteri maksimum jenerasyon durdurma kriterine oldukça benzemektedir. Yapılan çalışma boyunca veri setinin doğası gereği maksimum fonksiyon sınırlandırma değeri 4000 olarak belirlenmiştir. Bu şartlar altında hesaplanan performans göstergeleri

Çizelge 4.3'te verilmiştir. Çizelge 4.3'te 10 farklı algoritma için hesaplanan transfer fonksiyonları ve performans göstergeleri sunulmuştur. Diğer kısıtlamalar altında olduğu gibi AEO, LCO ve WCA algoritmaları da benzer performanslara sahiptir.

Çizelge 4.3. Fonksiyon hesaplama sınırlılığı altında performansların karşılaştırılması

| Algoritmalar | MAPE | MAE | MSE | $R^2$ | Süre | Transfer Fonksiyonları |
|---|---|---|---|---|---|---|
| AEO | 0,02747 | 0,13212 | 0,02742 | 0,95094 | 55,93 | $T_f(s) = \frac{-1.52s + 12.72}{s^2 + 6.19s + 13.02}$ |
| ALO | 0,06012 | 0,29109 | 0,12988 | 0,76762 | 67,15 | $T_f(s) = \frac{-12.05s + 97.35}{s^2 + 57.36s + 100}$ |
| ASO | 0,08063 | 0,39265 | 0,24611 | 0,55966 | 71,21 | $T_f(s) = \frac{-8.67s + 96.100}{s^2 + 67.91s + 96.51}$ |
| BSO | 0,06522 | 0,31382 | 0,14662 | 0,73767 | 56,50 | $T_f(s) = \frac{-5.88s + 43.68}{s^2 + 25.42s + 44.89}$ |
| CRO | 0,0664 | 0,32072 | 0,15305 | 0,72617 | 56,83 | $T_f(s) = \frac{-6.96s + 48.24}{s^2 + 27.71s + 49.89}$ |
| FPA | 0,06732 | 0,32909 | 0,15991 | 0,7139 | 56,21 | $T_f(s) = \frac{-3.84s + 35.40}{s^2 + 25.17s + 36.46}$ |
| LCO | 0,02746 | 0,13204 | 0,02739 | 0,95099 | 56,36 | $T_f(s) = \frac{-1.49s + 12.60}{s^2 + 6.11s + 12.90}$ |
| MFO | 0,06882 | 0,32945 | 0,16706 | 0,7011 | 55,38 | $T_f(s) = \frac{-13.91s + 100}{s^2 + 62.14s + 100}$ |
| TWO | 0,06596 | 0,31763 | 0,15154 | 0,72887 | 64,12 | $T_f(s) = \frac{-11.78s + 93.99}{s^2 + 55.10s + 96.25}$ |
| WCA | 0,02747 | 0,1321 | 0,02741 | 0,95096 | 254,19 | $T_f(s) = \frac{-1.49s + 12.59}{s^2 + 6.11s + 12.89}$ |

Erken durdurma sınırlılığı temel olarak global en iyi çözümünün artık hiç azalmadığı 3. nesil için algoritmanın durdurulmasını sağlar. Erken durdurma kriter değeri 3 seçildiği zaman tüm AEO algoritmalarının performansı oldukça iyi olmakla birlikte çözüm süreleri oldukça artmaktadır. Çizelge 4.4'te 10 farklı algoritma için hesaplanan transfer fonksiyonları MAPE, MAE, MSE, $R^2$ değerleri sunulmuştur. AEO, LCO ve WCA algoritmaları ile hesaplanan transfer fonksiyonları ve çözüm süreleri birbirine oldukça yakın olmakla birlikte özellikle AEO ve WCA algoritmalarının çözüm süresi çok uzun sürmüştür. AEO, LCO ve WCA algoritmaları $R^2$ performans değerleri 0,8 üstü olduğundan başarılı kabul edilmektedir.

Çizelgelerde sunulan MAPE, MAE, MSE, $R^2$ değerleri Bölüm 3.2.3'te tanıtılmıştır. Performans incelenmek istediğinde literatürde yaygın olarak kullanılmaktadır. Özellikle $R^2$ değerinin yüksek olması performansı güçlü bir sistemin elde edildiğinin bir göstergesidir. Bu noktada elde edilen transfer fonksiyonlarının cevabı ve gerçek verilerin karşılaştırılması ile bulunmaktadır.

Çizelge 4.4. Erken durdurma sınırlılıkları altında performansların karşılaştırılması

| Algoritmalar | MAPE | MAE | MSE | $R^2$ | Süre | Transfer Fonksiyonları |
|---|---|---|---|---|---|---|
| AEO | 0,02745 | 0,13203 | 0,02739 | 0,951 | 125,19 | $T_f(s) = \frac{-1.49s + 12.59}{s^2 + 6.11s + 12.89}$ |
| ALO | 0,06318 | 0,3022 | 0,14137 | 0,74707 | 17,42 | $T_f(s) = \frac{-10.55s + 99.83}{s^2 + 63.63s + 100}$ |
| ASO | 0,08824 | 0,41352 | 0,27678 | 0,50479 | 7,2 | $T_f(s) = \frac{-16.98s + 81.01}{s^2 + 82.47s + 88.48}$ |
| BSO | 0,06312 | 0,30446 | 0,13824 | 0,75267 | 9,84 | $T_f(s) = \frac{-4.20s + 59.34}{s^2 + 76.13s + 60.68}$ |
| CRO | 0,10454 | 0,49296 | 0,37162 | 0,33511 | 14,75 | $T_f(s) = \frac{-8.78s + 71.32}{s^2 + 48.60s + 73.66}$ |
| FPA | 0,0961 | 0,48415 | 0,32809 | 0,41299 | 13,06 | $T_f(s) = \frac{-7.71s + 73.73}{s^2 + 50.49s + 74.39}$ |
| LCO | 0,02745 | 0,13203 | 0,02739 | 0,951 | 91,17 | $T_f(s) = \frac{-1.49s + 12.59}{s^2 + 6.11s + 12.89}$ |
| MFO | 0,07422 | 0,36052 | 0,19727 | 0,64706 | 13,94 | $T_f(s) = \frac{-4.66s + 100}{s^2 + 80.08s + 97.84}$ |
| TWO | 0,0669 | 0,31681 | 0,15556 | 0,72167 | 20,63 | $T_f(s) = \frac{-11.82s + 100}{s^2 + 68.94s + 100}$ |
| WCA | 0,02745 | 0,13202 | 0,02739 | 0,951 | 136,36 | $T_f(s) = \frac{-1.49s + 12.58}{s^2 + 6.10s + 12.88}$ |

## 4.3. Algoritmaların Geçici Durum Değerleri

Çizelge-4.5, 4.6, 4.7 ve 4.8 de farklı algoritmalar kullanılarak elde edilen transfer fonksiyonları ile ilgili olarak yükselme süresi, yerleşme süresi, aşım miktarı, düşüm miktarı, tepe noktası ve tepe zamanı gibi geçici durum değerleri sunulmuştur.

Çizelge 4.5' te 45 sn zaman sınırlılığı altında AEO Algoritmasında yükselme süresi (0,729 sn), yerleşme süresi (1,277 sn), aşım miktarı (0,645sn), düşüm miktarı (6,069sn), tepe noktası (0,982sn) ve tepe zamanı olarak (1,756 sn) değerleri bulunmuştur. Çizelge 4.5 incelendiğinde AEO algoritması diğer algoritmalarla

karşılaştırıldığında bütün parametrelerde daha düşük değerler bulunmuş olup LCO Algoritması da AEO Algoritmasına yakın değerler almıştır. Bu çizelgede 10 algoritma kullanılmış olup, 100 iterasyon sonunda rastgele 10. döngüdeki değerler baz alınmıştır. Bu çizelgede dikkat edilen durumlardan bir tanesi de AEO ve LCO algoritmasının dışında diğer algoritmalarda aşım değeri tespit edilememiştir.

Çizelge 4.5. Algoritmaların zaman sınırlılığında (45 sn) geçici duruma cevapları

| Algoritmalar | Yükselme Süresi | Yerleşme Süresi | Aşım Miktarı | Düşüm Miktarı | Tepe noktası | Tepe zamanı |
|---|---|---|---|---|---|---|
| AEO | 0,729791 | 1,277134 | 0,645476 | 6,069158 | 0,982553 | 1,756059 |
| ALO | 1,180223 | 2,247309 | 0 | 17,74053 | 0,957853 | 3,710374 |
| ASO | 1,497444 | 2,70506 | 0 | 0 | 0,976193 | 4,78216 |
| BSO | 1,566425 | 2,899934 | 0 | 8,595102 | 0,978978 | 4,924513 |
| CRO | 1,227713 | 2,319288 | 0 | 16,01996 | 0,971095 | 3,853929 |
| FPA | 1,187139 | 2,231984 | 0 | 8,746113 | 0,979014 | 3,732171 |
| LCO | 0,730066 | 1,277615 | 0,647488 | 6,06151 | 0,982548 | 1,756721 |
| MFO | 1,44766 | 2,623884 | 0 | 1,728577 | 0,998904 | 4,478698 |
| TWO | 1,242869 | 2,346555 | 0 | 15,10117 | 0,99035 | 3,906856 |
| WCA | 1,276967 | 2,394746 | 0 | 10,27175 | 0,973017 | 4,010854 |

Çizelge 4.6’ da maksimum jenerasyon sınırlılığı (MG 30) altında AEO Algoritmasında yükselme süresi (0,730 sn), yerleşme süresi (1,278 sn), aşım miktarı(0,653sn), düşüm miktarı (6,051sn), tepe noktası (0,982sn) ve tepe zamanı olarak (1,757 sn) değerleri bulunmuştur. Çizelge incelendiğinde önceki tabloda olduğu gibi AEO algoritması diğer algoritmalarla karşılaştırıldığında düşük değer alan algoritmalardan olmuştur. LCO ve WCA Algoritmaları da AEO Algoritması ile benzer değerler almıştır. Çizelgede 100 iterasyon sonunda rastgele 10. döngüdeki değerler baz alınmıştır. Bu çizelgede AEO, LCO ve WCA algoritmaların dışında diğer

algoritmalarda aşım değeri olmamıştır. WCA Algoritması da yükselme süresi (0,686 sn), yerleşme süresi (1,208 sn), aşım miktarı (1,149sn), düşüm miktarı (6,334 sn), tepe noktası (0,982sn) ve tepe zamanı olarak (1,634 sn) değerleri ile en düşük değerleri almıştır.

Çizelge 4.6. Algoritmaların maksimum jenerasyon sınırlılıkları altında (MG=30) geçici duruma cevapları

| Algoritmalar | Yükselme Süresi | Yerleşme Süresi | Aşım Miktarı | Düşüm Miktarı | Tepe Noktası | Tepe Zamanı |
|---|---|---|---|---|---|---|
| AEO | 0,730536 | 1,278439 | 0,653924 | 6,05095 | 0,982594 | 1,757853 |
| ALO | 1,078391 | 2,055497 | 0 | 16,25964 | 0,976029 | 3,390081 |
| ASO | 1,061885 | 2,109224 | 0 | 27,21129 | 0,907235 | 3,340817 |
| BSO | 1,211576 | 2,29025 | 0 | 13,37398 | 0,97394 | 3,810902 |
| CRO | 1,267524 | 2,376218 | 0 | 10,89341 | 0,992016 | 3,983202 |
| FPA | 1,013803 | 2,007057 | 0 | 18,03729 | 0,988689 | 3,19896 |
| LCO | 0,730139 | 1,277743 | 0,649968 | 6,05666 | 0,982556 | 1,756897 |
| MFO | 1,194541 | 2,303387 | 0 | 23,83762 | 0,950297 | 3,752418 |
| TWO | 1,258586 | 2,384562 | 0 | 16,85766 | 0,975758 | 3,956589 |
| WCA | 0,686912 | 1,208018 | 1,149833 | 6,334358 | 0,982571 | 1,634378 |

Çizelge 4.7’ de fonksiyon hesaplama sınırlılığı (FE 4000) altında 100 iterasyonda 10. döngüde AEO Algoritmasın yükselme süresi (0,732 sn), yerleşme süresi (1,281 sn), aşım miktarı (0,669sn), düşüm miktarı (6,020sn), tepe noktası (0,982sn) ve tepe zamanı olarak (1,738 sn) değerleri almıştır. Çizelge incelendiğinde önceki tablolarda olduğu gibi AEO algoritması diğer algoritmalarla karşılaştırıldığında en düşük değer alan algoritmalardan olmuştur. LCO ve WCA Algoritmaları da AEO Algoritmasına yakın düşük değerler almıştır. Bu çizelgede AEO, LCO ve WCA algoritmaların dışında diğer algoritmalarda aşım değeri sıfır bulunmuştur.

Çizelge 4.7. Algoritmaların fonksiyon hesaplama sınırlılığı altında (FE=4000) geçici duruma cevapları

| Algoritmalar | Yükselme Süresi | Yerleşme Süresi | Aşım Miktarı | Düşüm Miktarı | Tepe Noktası | Tepe Zamanı |
|---|---|---|---|---|---|---|
| AEO | 0,732108 | 1,281189 | 0,669131 | 6,020112 | 0,982782 | 1,738756 |
| ALO | 1,238631 | 2,332837 | 0 | 13,93482 | 0,976484 | 3,897037 |
| ASO | 2,210254 | 4,006085 | 0 | 0,68498 | 0,99892 | 6,837973 |
| BSO | 1,167799 | 2,233874 | 0 | 14,55343 | 0,969009 | 3,678364 |
| CRO | 1,141891 | 2,179973 | 0 | 12,57048 | 0,972392 | 3,592668 |
| FPA | 1,343207 | 2,585124 | 0 | 18,99938 | 0,977773 | 4,214589 |
| LCO | 0,730396 | 1,278192 | 0,652627 | 6,052953 | 0,982581 | 1,757514 |
| MFO | 1,708722 | 3,196964 | 0 | 0 | 0,985448 | 5,456887 |
| TWO | 1,239212 | 2,33638 | 0 | 14,51355 | 0,972433 | 3,896416 |
| WCA | 0,730607 | 1,278562 | 0,654809 | 6,049868 | 0,98261 | 1,758022 |

Çizelge 4.8’ de erken durdurma sınırlılığı altında kriter olarak 3 sayısı girilerek 10 adet algoritma seçilmiştir. Burada erken durdurma kriterinin 3 girilmesinin nedeni 3 başarılı sonuç sonrası sadece bir iterasyon için işlemi bitirmektedir. Ancak bu kriter de işlem süresi çok uzun sürebilmektedir. İterasyon sonucunda gerçeğe yakın başarılı sonuçlar verebilmektedir. 100 iterasyon çalıştırmada 10. döngüdeki değerler baz alınmıştır. Çizelgede AEO algoritması yükselme süresi (0,729 sn), yerleşme süresi (1,277 sn), aşım miktarı (0,647sn), düşüm miktarı (6,060sn), tepe noktası (0,982sn) ve tepe zamanı olarak (1,756 sn) gibi geçici durum değerleri ortaya çıkmıştır. Algoritmalar birbiriyle kıyaslandığında AEO, LCO ve WCA algoritmaları birbirine yakın değerler alarak bütün parametrelerde diğerlerinden daha düşük zaman değerlerini almıştır. Bu kriter altında da diğer sınırlılıklar altında olduğu gibi çizelgedeki değerler AEO algoritması ön plana çıkarmaktadır. Bu da bu algoritmayı başarılı kılmaktadır. LCO ve WCA algoritmalarında aykırı değerleri olmasına rağmen %98 oranla $R^2$ değerleri 0,9 etrafında olduğundan başarılı kabul edilmektedir.

Çizelge 4.8. Algoritmaların erken durdurma sınırlılıkları altında (ES=3) geçici duruma cevapları

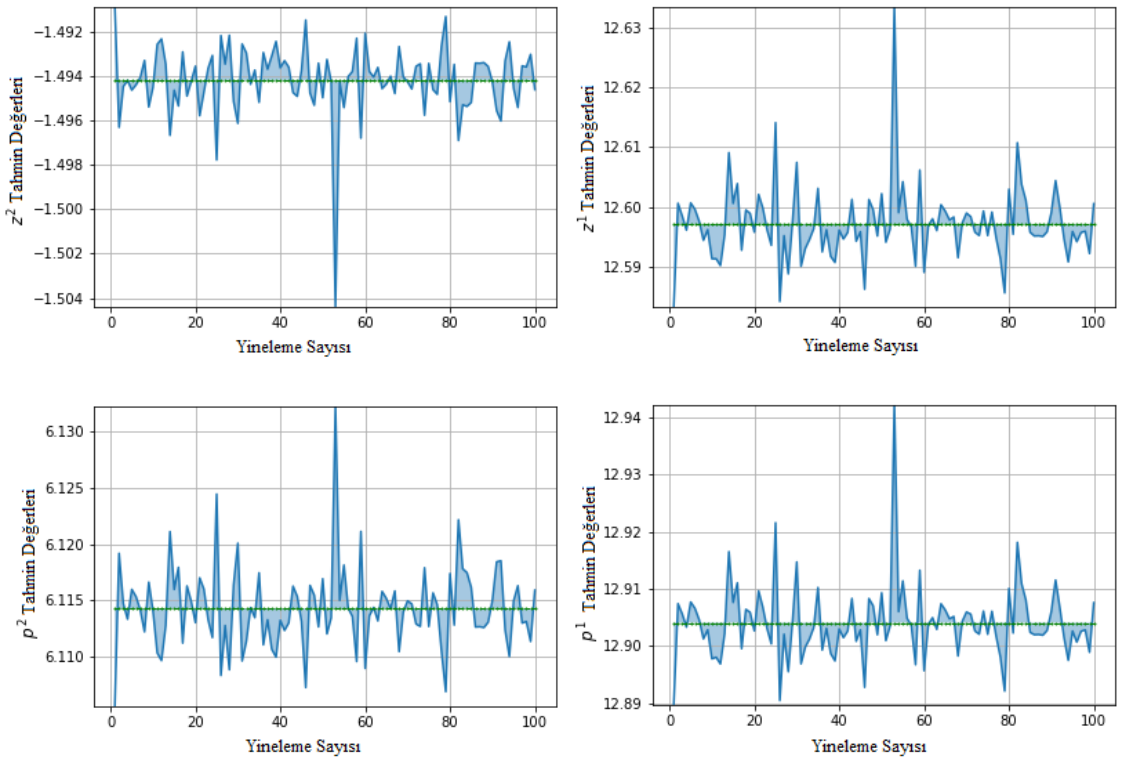
| Algoritmalar | Yükselme Süresi | Yerleşme Süresi | Aşım Miktarı | Düşüm Miktarı | Tepe Noktası | Tepe Zamanı |
|---|---|---|---|---|---|---|
| AEO | 0,729864 | 1,277261 | 0,647388 | 6,060724 | 0,98253 | 1,756234 |
| ALO | 1,060335 | 2,024673 | 0 | 12,50767 | 1,004781 | 3,333728 |
| ASO | 1,701271 | 3,331251 | 0 | 32,47562 | 1,010654 | 5,350093 |
| BSO | 1,147521 | 2,202588 | 0 | 12,61748 | 0,967589 | 3,611157 |
| CRO | 1,516303 | 2,908995 | 0 | 22,61467 | 1,004732 | 4,771411 |
| FPA | 1,415892 | 2,477812 | 0 | 0 | 0,986779 | 4,380417 |
| LCO | 0,730264 | 1,277962 | 0,651339 | 6,054587 | 0,98257 | 1,757198 |
| MFO | 3,578008 | 6,261513 | 0 | 0 | 0,981312 | 11,06946 |
| TWO | 1,121064 | 2,166308 | 0 | 23,8927 | 0,982871 | 3,522958 |
| WCA | 0,730649 | 1,278635 | 0,655017 | 6,04889 | 0,982609 | 1,758123 |

## 4.4. Model Parametrelerinin Değişim Analizi

Sistemin modeli geliştirilirken pay ve payda kısmında 2 adet katsayı olacak şekilde bir transfer fonksiyonu önerilmiştir. Önerilen transfer fonksiyonu aşağıdaki Denklem (4.1)'de sunulmuştur.

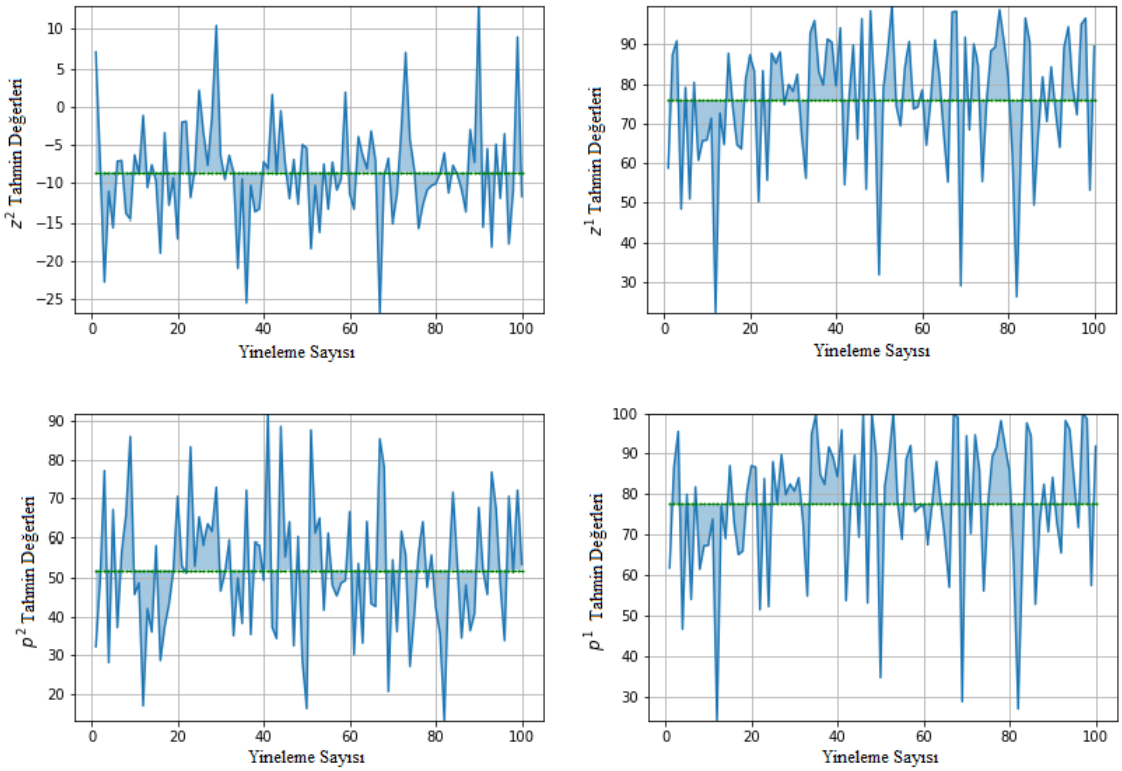
$$T_f(s) = \frac{z_2 s + z_1}{s^2 + p_2 s + p_1} \tag{4.1}$$

Bu denklem, sistemin transfer fonksiyonunu ifade etmektedir. Transfer fonksiyonu, sistemin giriş sinyalini, çıktı sinyaline $T_f(s)$ dönüştüren matematiksel bir işlevdir. Denklemdeki $z_1$, $z_2$, $p_1$ ve $p_2$, sistemdeki özellikleri ve davranışı ifade eden parametrelerdir. $z_1$ ve $z_2$, sistemin pay kısmında, $p_1$ ve $p_2$ ise payda kısmında yer alan katsayılardır. Bu katsayılar, sistemin frekans tepkisini ve istenen davranışı nasıl gerçekleştireceğini belirler.




Şekil 4.9. Erken durdurma sınırlılıkları altında AEO algoritmasının pay ve kök parametrelerinin değişimi

AEO algoritmasının ES (erken durdurma) kriteri kullanılarak 100 iterasyon boyunca hesaplanan "$z_1$, $z_2$, $p_1$, $p_2$" değerleri Şekil-4.9'da verilmiştir. Her sınırlılık altında her seferinde en iyi sonucu sadece AEO algoritması elde ettiği için özellikle sunulmuştur. Ayrıca, şekiller incelenecek olursa, bu parametrelerin sırasıyla $z_1$ değeri (12.583 ile 12.634), $z_2$ değeri ( -1.504 ile -1.493), $p_1$ değeri (12.89 ile 12.94) ve $p_2$ değeri (6.105 ile 6.113) arasında değerler almıştır. Bu veriler, algoritmanın performansını ve optimize edilen değerlerin aralığını belirtmektedir. AEO algoritmasında parametre değişimleri çok sınırlı kalırken performansı kötü olan CRO algoritmasının kök ve sıfır değerlerinin değişimi çok daha geniş aralıkta olmaktadır. Bu değişimde performansı ciddi oranda etkilemektedir. Şekil 4.10' da CRO algoritmasının ES (erken durdurma) kriteri kullanılarak 100 iterasyon boyunca hesaplanan "$z_1$, $z_2$, $p_1$, $p_2$" değerleri verilmiştir. Şekil 4.10 incelenecek olursa, bu parametrelerin $z_1$ değerinin (22 ile 98) değerleri arasında, $z_2$ değerinin (-26 ile 13) değerleri arasında, $p_1$ değerinin (24 ile 100) değerleri arasında ve $p_2$ değerinin de (13 ile 91) değerleri arasında olduğu görülmektedir. Bu değerlere göre CRO algoritması dengesiz bir dağılım gösterdiğinden performansının başarısız olduğu söylenebilir.

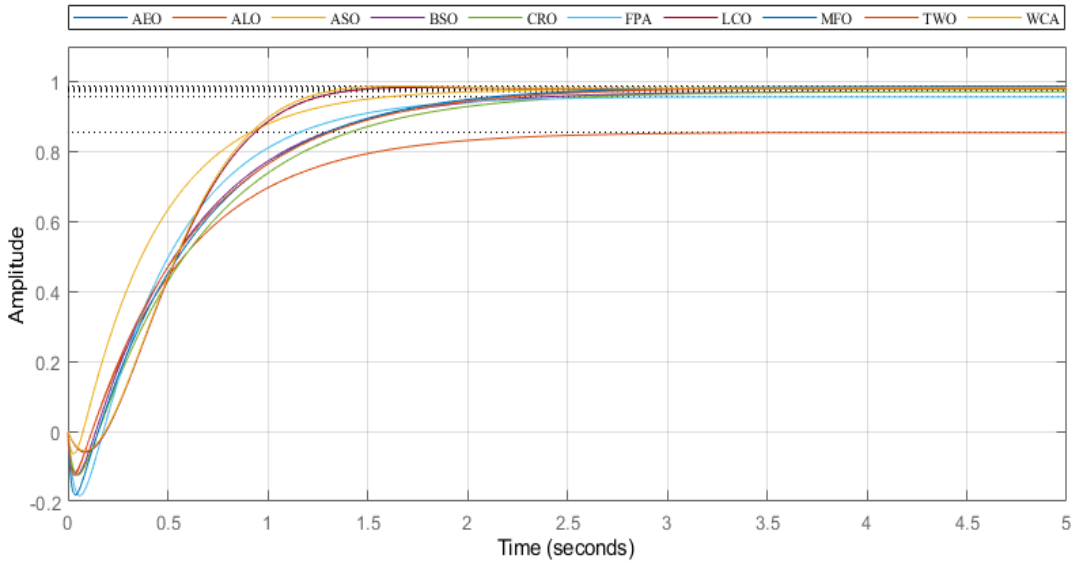



Şekil 4.10. Erken durdurma sınırlılığı altında CRO algoritmasının pay ve kök parametrelerinin değişimi

Sonuç olarak önerilen AEO algoritmasının sistem tanımlama problemlerini çözmekte ki durumu yukarıdaki grafikler incelendiğinde hem başarılı hem de tutarlı olduğu net bir şekilde söylenebilmektedir.

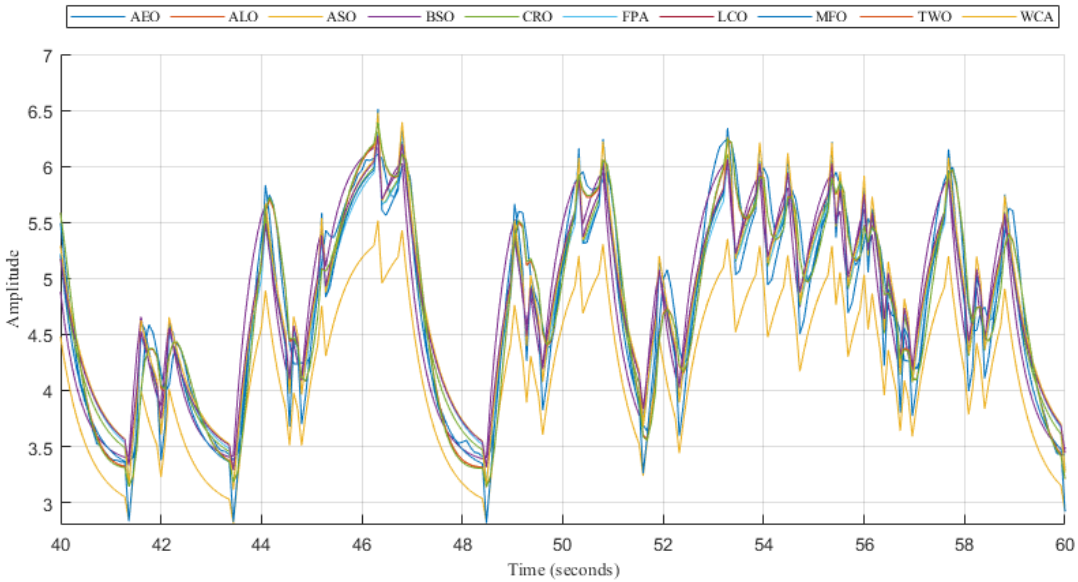
## 4.6 Zaman Cevabı Analizi

Elde edilen transfer fonksiyonlarının zaman cevabı sistem adına bir yorum yapabilmek için önemlidir. Bu noktada zaman sınırlılığı temel alınarak ilgili transfer fonksiyonlarının basamak girişine karşılık sistem cevabı Şekil 4.11'te sunulmuştur. Bu şekilde meta-sezgisel algoritmaların biri dışında neredeyse benzer zaman cevabına sahip olduğu görülmektedir. Şekil 4.11'de hair dryer veri setinin (koyu mavi) ve meta-sezgisel algoritmaların zamana karşılık genlik grafikleri verilmiştir. AEO algoritması ile elde edilen transfer fonksiyonu orijinal veriyi diğer algoritmalara kıyasla daha küçük hatayla takip ederken diğer algoritmalar takip işlemini benzer veya daha düşük bir başarıyla gerçekleştirmektedir.




Şekil 4.11. Meta-sezgisel algoritmaların zamana karşılık genlik grafiği

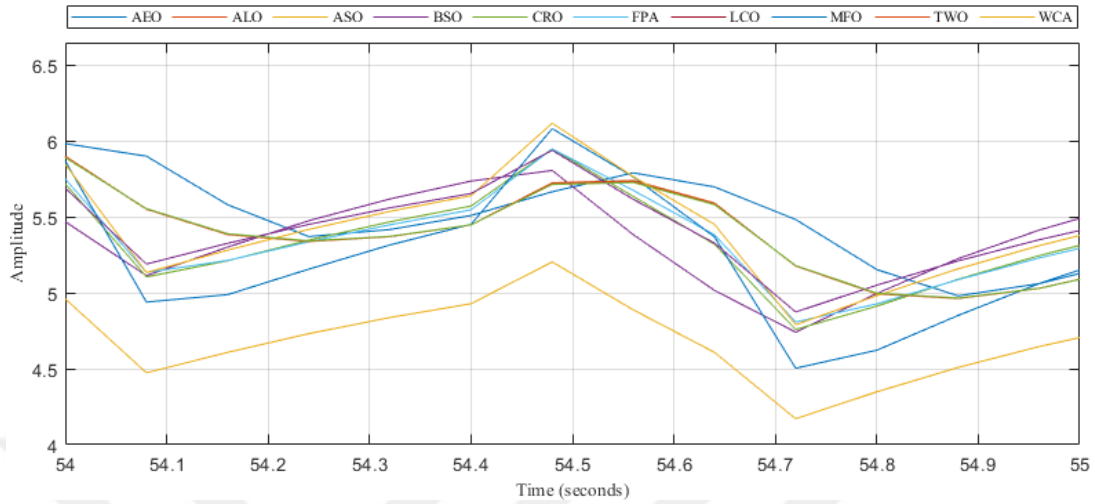
Şekil 4.12'de gösterilen zaman cevabı 40 ile 60 saniye aralığında algoritmaların ürettiği transfer fonksiyonlarının cevabını göstermektedir.




Şekil 4.12. Meta-sezgisel algoritmaların zaman cevabı

Şekil 4.12 grafiği incelendiğinde algoritma sayısı fazla olduğundan dolayı algoritmaların detaylı çizimi Şekil 4.13'te verilmiştir. Bu şekilde 54 ve 55 saniye aralığında çizim yaptırılmıştır. Bu şekilden sonuçların birbirlerine benzerliğini görmek

mümkündür. Ancak en iyi performans güvenilirlik analizleri sonucunda ortaya çıkmaktadır.




Şekil 4.13 Meta-sezgisel algoritmaların detaylı zaman cevabı

# 5. SONUÇLAR VE ÖNERİLER

## 5.1 Sonuçlar

Meta-sezgisel algoritmalar son yıllarda oldukça popüler hale gelmiş ve çok çeşitli problemlerin çözümü için kullanılmıştır. Bu tez çalışmasında meta-sezgisel algoritmalar sistem tanımlama problemlerine uygulanmış ve sonuçları incelenmiştir. Elde edilen performans kriterleri incelendiğinde, AEO, ALO, ASO, BSO, CRO, FPA, LCO, MFO, TWO ve WCA Algoritmaları birbiriyle karşılaştırıldığında AEO, LCO VE WCA algoritmalarının gerçek dünyadaki problemlerle başa çıkmak için daha umut verici olduğunu göstermektedir. Sistem tanımlama uygulamaları, giriş-çıkış verilerinin analiz edilerek matematiksel bir modelin oluşturulma süreci olarak tanımlanmaktadır. Bu tez siyah-kutu (black-box) sistem tanımlama yöntemini temel alarak farklı algoritmaların transfer fonksiyonunu araştırmaya odaklanmıştır. Özet olarak aşağıdaki bulgular elde edilmiştir.

1- Orta düzeyde bir PC kullanılmış ile 10 farklı meta-sezgisel algoritmasının sistem tanımlama problemlerinde kolaylıkla kullanılabileceği gösterilmiştir.

2- Bu çalışmada saç kurutma (hair dryer) deney setinden elde edilen veriler için sistem tanımlama çalışması yapılmıştır.

3- Python programlama dili kullanılmıştır.

4- AEO algoritması ve diğer 9 algoritma 100 defa çalıştırılarak performansları incelenmiştir.

5- Algoritmaların güvenilirliği ile ilgili sonuçların sunulduğu bölümde AEO algoritması 100 iterasyon sonunda yüksek bir performans değerine ulaştığı için en güvenilir algoritma olmuştur.

6- LCO ve WCA algoritmalarında aykırı değerler olmasına rağmen aldıkları değerlerin 0,8 eşik değerinin üstünde olmasından dolayı %98 oranla başarılı kabul edilmektedir.

7-Zaman, maksimum jenerasyon, fonksiyon hesaplama ve erken durdurma sınırlılıkları dikkate alınarak algoritmalar çalıştırılmıştır. Erken durdurma sınırlılığı, algoritmalara problem çözümü için yeterince süre vererek performansını belirlemek için önemlidir. Bu noktada özellikle AEO ve WCA algoritmaları global çözüme çok uzun sürelerde

ulaşmıştır. Ancak özellikle AEO algoritması zaman sınırlıkları altında da başarıyla çalıştığı için oldukça güvenilir ve yüksek performansla çalıştığı tekrar teyit edilmiştir. Özellikle zaman sınırlaması altında çıkan $R^2$ değerlerinin yüksek ve tutarlı çıkması oldukça önem kazanmaktadır.

## 5.2 Öneriler

1- AEO algoritması, doğal ekosistemlerin prensiplerinden esinlenerek geliştirildiği için karmaşık ayrık sistem tanımlama problemleri için uygulanabilir.

2- AEO'nun çözüm süresi performansını iyileştirmek için yeni varyasyonlar geliştirilebilir veya özelleştirmeler yapılabilir.

3- Meta-heuristik algoritmaların girişinde probleme özgü belirlenen sınırların daha uygun seçilmesine dönük çalışmalar yapılabilir.

## KAYNAKLAR


Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(1), 29-41.

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Simüle edilmiştavlama ile optimizasyon. Bilim 220(4598):671–680

Storn R, Price K (1997) Diferansiyel evrim: sürekli uzaylar üzerinde global optimizasyon için basit ve verimli bir buluşsal yöntem. *J Glob Optimum* 11:341–359

Zhao, W., Wang, L., & Zhang, Z. (2020). Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. *Neural Computing and Applications*, *32*, 9383-9425.

ZALOĞLU, M., Fidan, Ş., & ERKAN, E. (2023, April). Meta-Heuristik Optimizasyon Algoritmalarının Sistem Tanımlama Problemine Uygulanması ve Performans Karşılaştırması. In *International Conference on Engineering, Natural and Social Sciences* (Vol. 1, pp. 510-515).

El-Dabah, M. A., El-Sehiemy, R. A., Becherif, M., & Ebrahim, M. A. (2021). Parameter estimation of triple diode photovoltaic model using an artificial ecosystem-based optimizer. *International Transactions on Electrical Energy Systems*, *31*(11), e13043.

Omotoso, H. O., Al-Shaalan, A. M., Farh, H. M., & Al-Shamma'a, A. A. (2022). Techno-economic evaluation of hybrid energy systems using artificial ecosystem-based optimization with demand side management. *Electronics*, *11*(2), 204.

Nguyen, T. T. (2021). A novel metaheuristic method based on artificial ecosystem-based optimization for optimization of network reconfiguration to reduce power loss. *Soft Computing*, *25*(23), 14729-14740.

Izci, D., Hekimoğlu, B., & Ekinci, S. (2022). A new artificial ecosystem-based optimization integrated with Nelder-Mead method for PID controller design of buck converter. *Alexandria Engineering Journal*, *61*(3), 2030-2044.

Mergos, P. E.,& Yang, X. S. (2021). Flower pollination algorithm parameters tuning. *Soft computing*, *25*(22), 14429-14447.

Abdel-Basset, M.,& Shawky, L. A. (2019). Flower pollination algorithm: a comprehensive review. *Artificial Intelligence Review*, *52*, 2533-2557.

Rodrigues, D., de Rosa, G. H., Passos, L. A., & Papa, J. P. (2020). Adaptive improved flower pollination algorithm for global optimization. *Nature-inspired computation in data mining and machine learning*, 1-21.

Engel, E. A.,& Kovalev, I. V. (2016). MPPT of a partially shaded photovoltaic module by ant lion optimizer. In *Advances in Swarm Intelligence: 7th International Conference, ICSI 2016, Bali, Indonesia, June 25-30, 2016, Proceedings, Part I 7* (pp. 451-457). Springer International Publishing.

Guo, M. W., Wang, J. S., Zhu, L. F., Guo, S. S., & Xie, W. (2020). Improved ant lion optimizer based on spiral complex path searching patterns. *IEEE Access*, *8*, 22094-22126.

Liu, Y., Qin, W., Zhang, J., Li, M., Zheng, Q., & Wang, J. (2021). Multi-Objective Ant Lion Optimizer Based on Time Weight. *IEICE TRANSACTIONS on Information and Systems*, *104*(6), 901-904.

Jiang, L., Hao, K., Tang, X. S., Wang, T., & Liu, X. (2021, October). An improved moth-flame optimization algorithm based on fusion mechanism. In *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society* (pp. 1-6). IEEE.

Shehab, M., Abualigah, L., Al Hamad, H., Alabool, H., Alshinwan, M., & Khasawneh, A. M. (2020). Moth–flame optimization algorithm: variants and applications. *Neural Computing and Applications*, *32*, 9859-9884.

Worch, E., Samiappan, S., Zhou, M., & Ball, J. E. (2020, September). Hyperspectral band selection using moth-flame metaheuristic optimization. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium* (pp. 1271-1274). IEEE.

Kaveh, A.,& Shokohi, F. (2016). Optimum design of laterally supported castellated beams using tug of war optimization algorithm. *Structural Engineering and Mechanics*, *3*(58), 533-553.

Kaveh, A.,& Zolghadr, A. (2016). A novel meta-heuristic algorithm: tug of war optimization.

Fu, Y., Li, Z., Qu, C., & Chen, H. (2020). Modified atom search optimization based on immunologic mechanism and reinforcement learning. *Mathematical Problems in Engineering*, *2020*.

Ghelichi, M., Goltabar, A. M., Tavakoli, H. R., & Karamodin, A. (2021). Neuro-fuzzy active control optimized by Tug of war optimization method for seismically excited benchmark highway bridge. *Numerical Algebra, Control and Optimization*, *11*(3), 333-351.

Bi, J.,& Zhang, Y. (2022). An improved atom search optimization for optimization tasks. *Multimedia Tools and Applications*, 1-55.

Sun, P., Liu, H., Zhang, Y., Tu, L., & Meng, Q. (2021). An intensify atom search optimization for engineering design problems. *Applied Mathematical Modelling*, *89*, 837-859.

Zhao, W., Wang, L., & Zhang, Z. (2019). A novel atom search optimization for dispersion coefficient estimation in groundwater. *Future Generation Computer Systems*, *91*, 601-610.

Narmatha, C., Eljack, S. M., Tuka, A. A. R. M., Manimurugan, S., & Mustafa, M. (2020). A hybrid fuzzy brain-storm optimization algorithm for the classification of brain tumor MRI images. *Journal of ambient intelligence and humanized computing*, 1-9.

Tuba, E., Strumberger, I., Bacanin, N., Zivkovic, D., & Tuba, M. (2019, June). Brain Storm Optimization Algorithm for Thermal Image Fusion using DCT Coefficients. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (pp. 234-241). IEEE.

Cheng, S., Zhang, M., Ma, L., Lu, H., Wang, R., & Shi, Y. (2021). Brain storm optimization algorithm for solving knowledge spillover problems. *Neural Computing and Applications*, 1-14.

Wu, Y., Wang, X., Fu, Y., & Li, G. (2019). Many-objective brain storm optimization algorithm. *Ieee Access*, *7*, 186572-186586.

Heidari, A. A., Ali Abbaspour, R., & Rezaee Jordehi, A. (2017). An efficient chaotic water cycle algorithm for optimization tasks. *Neural Computing and Applications*, *28*, 57-85.

Zhou, G., Zhou, Y., Tang, Z., & Luo, Q. (2021). CWCA: Complex-valued encoding water cycle algorithm. *Mathematical Biosciences and Engineering*, *18*(5), 5836-5864.

Roeva, O., Angelova, M., Zoteva, D., & Pencheva, T. (2020). Water Cycle Algorithm for Modelling of Fermentation Processes. *Processes*, *8*(8), 920.

Chen, W. (2019, December). Cloud computing task scheduling method based on a coral reefs optimization algorithm. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 27-34). IEEE.

Moya, I., Bermejo, E., Chica, M., & Cordon, O. (2021). Coral reefs optimization algorithms for agent-based model calibration. *Engineering Applications of Artificial Intelligence*, *100*, 104170.

Salcedo-Sanz, S., García-Díaz, P., Portilla-Figueras, J. A., Del Ser, J., & Gil-Lopez, S. (2014). A coral reefs optimization algorithm for optimal mobile network deployment with electromagnetic pollution control criterion. *Applied Soft Computing*, *24*, 239-248.

Afzal, A. M. S. (2021, May). Optimized support vector machine model for visual sentiment analysis. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)* (pp. 171-175). IEEE.

Khatri, A., Gaba, A., Rana, K. P. S., & Kumar, V. (2020). A novel life choice-based optimizer. *Soft computing*, *24*, 9121-9141.

Gupta, S., Thakur, S., & Gupta, A. (2022). Optimized hybrid machine learning approach for smartphone based diabetic retinopathy detection. *Multimedia Tools and Applications*, *81*(10), 14475-14501.

Yang, X. S. (2012). Flower pollination algorithm for global optimization. In *Unconventional Computation and Natural Computation: 11th International Conference, UCNC 2012, Orléan, France, September 3-7, 2012. Proceedings 11* (pp. 240-249). Springer Berlin Heidelberg.

KORKMAZ, E.,& AKGÜNGÖR, A. P. (2018). Türkiye'deki Araç Sahipliğinin Çiçek Tozlaşma Algoritması ile Tahmini. *Gazi Mühendislik Bilimleri Dergisi*, *4*(1), 39-45.

KIZILKAPLAN, E., Tolga, E. R. E. N., & YALÇINKAYA, F. (2020). Kablosuz Sensör Ağlarında Konum Belirlemede Sezgisel Algoritmaların Kuantum Davranışları ile Karşılaştırılması. *International Journal of Engineering Research and Development*, *12*(2), 587-602.

YÜZGEÇ, U., & KILIÇ, H. Kutulama Problemi için Geliştirilmiş Karınca Aslanı Optimizasyonu Algoritması. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, *11*(2), 13-19.

Jama, B. S. A., & Baykan, N. (2020). Modified Region Growing Method For Image Segmentation Using Ant Lion Optimization Algorithm. *Avrupa Bilim ve Teknoloji Dergisi*, 404-411.

Mirjalili, S. (2015). The ant lion optimizer. *Advances in engineering software*, *83*, 80-98.

GÜRGEN, A. Güve-Alev Optimizasyon Algoritması Kullanarak Pleurotus cornucopiae var. citrinopileatus Mantarı Ekstraksiyon Koşullarının Optimizasyonu. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, *10*(3), 1508-1523.

Kaur, K., Singh, U., & Salgotra, R. (2020). An enhanced moth flame optimization. *Neural Computing and Applications*, *32*, 2315-2349.

Erdal, E. K. E. R., KAYRİ, M., & EKİNCİ, S. (2019). Kısıtlı optimizasyon problemlerinin çözümü için atom arama optimizasyon algoritması. *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, *10*(3), 841-851.

Hekimoğlu, B. (2019). Optimal tuning of fractional order PID controller for DC motor speed control via chaotic atom search optimization algorithm. *IEEE Access*, *7*, 38100-38114.

Cheng, S., Shi, Y., Qin, Q., Zhang, Q., & Bai, R. (2014). Population diversity maintenance in brain storm optimization algorithm. *Journal of Artificial Intelligence and Soft Computing Research*, *4*(2), 83-97.

Cai, Z., Gao, S., Yang, X., Yang, G., Cheng, S., & Shi, Y. (2022). Alternate search pattern-based brain storm optimization. *Knowledge-Based Systems*, *238*, 107896.

Hanafi, M. F. I. M., Bahreininejad, A., & Uddin, N. (2021, August). Optimization of shell and tube heat exchanger using the water cycle algorithm. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1173, No. 1, p. 012005). IOP Publishing.

Gür, H. (2020). *Su döngüsü algoritması ile PID ve kayan kipli kontrol parametrelerinin optimizasyonu* (Master's thesis, İskenderun Teknik Üniversitesi/Mühendislik ve Fen Bilimleri Enstitüsü/Elektrik-Elektronik Mühendisliği Anabilim Dalı).

Salcedo-Sanz, S., Pastor-Sánchez, A., Portilla-Figueras, J. A., & Prieto, L. (2016). Effective multi-objective optimization with the coral reefs optimization algorithm. *Engineering Optimization*, *48*(6), 966-984.

Yan, C., Ma, J., Luo, H., & Patel, A. (2019). Hybrid binary coral reefs optimization algorithm with simulated annealing for feature selection in high-dimensional biomedical datasets. *Chemometrics and Intelligent Laboratory Systems*, *184*, 102-111.

Khatri, A., Gaba, A., Rana, K. P. S., & Kumar, V. (2020). A novel life choice-based optimizer. *Soft computing*, *24*, 9121-9141.

Fissore, D. (2021). 2.3 System Identification: Linear methods.

FİdAn, Ş., Sevim, D., & Erkan, E. (2022, October). System Identification and Control of High Voltage Boost Converter. In *2022 Global Energy Conference (GEC)* (pp. 25-31). IEEE.

Jagatheesan, K., Anand, B., Dey, K. N., Ashour, A. S., & Satapathy, S. C. (2018). Performance evaluation of objective functions in automatic generation control of thermal power system using ant colony optimization technique-designed proportional–integral–derivative controller. *Electrical Engineering*, *100*, 895-911.

FİDAN, Ş., & ERKAN, E. (2023, May). Boost Konvertörün Black-Box Sistem Tanımlama Yöntemi ile Transfer Fonksiyonunun Elde Edilmesi ve Parçacık Sürü Algoritması Tabanlı PI Kontrolör Tasarımı. In *International Conference on Recent Academic Studies* (Vol. 1, No. 1, pp. 210-217).

Sanatel, Ç. (2020). *Uzun kısa süreli bellek tabanlı sistem tanıma ve uyarlamalı kontrol* (Doctoral dissertation, Fen Bilimleri Enstitüsü).

Van Thieu, N., & Mirjalili, S. (2023). MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*, *139*, 102871.

Ravber, M., Liu, S. H., Mernik, M., & Črepinšek, M. (2022). Maximum number of generations as a stopping criterion considered harmful. *Applied Soft Computing*, *128*, 109478.

# EKLER

## EK-1 Grafik Çizimleri için Python Kodları

```
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
font = FontProperties()
font.set_family('serif')
font.set_name('Times New Roman')

font.set_size(8)

plt.rc('xtick', labelsize=8)    # onay etiketlerinin yazı tipi boyutu
plt.rc('ytick', labelsize=8)    # onay etiketlerinin yazı tipi boyutu

fig, ax = plt.subplots(figsize=(2.25, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalAEOTB45_1['R2'],c='#ef476f',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('AEO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent/Images/TB/OriginalAEOTB45_1", bbox_inches='tight', dpi=600)
plt.show()

#Y label'in kırpılmasını önlemek için aralığı düzeltin
fig.tight_layout()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed:   0'],y=OriginalALOTB45_1['R2'],c='#ff6900',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('ALO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalALOTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()


fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalASOTB45_1['R2s'],c='#06d6a0',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('ASO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalASOTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()

fig, ax = plt.subplots(figsize=(2.25, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalBSOTB45_1['R2'],c='#1181b2',s=3)
```

```
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('BSO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalBSOTB45_1", bbox_inches='tight', dpi=600)
plt.show()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalCROTB45_1['R2s'],c='#b22611',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('CRO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalCROTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalFPATB45_1['R2'],c='#4b9e1e',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('FPA Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalFPATB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalLCOTB45_1['R2'],c='#918213',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('LCO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalLCOTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalMFOTB45_1['R2'],c='#781391',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('MFO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalMFOTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()
```

```
fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalTWOTB45_1['R2'],c='#1e0fbd',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('TWO Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalTWOTB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()

fig, ax = plt.subplots(figsize=(2, 2.2))
plt.scatter(x=OriginalAEOTB45_1['Unnamed: 0'],y=OriginalWCATB45_1['R2'],c='#cc0202',s=3)
ax.set_ylabel('Estimated $R^{2}$ Values',fontproperties=font)
ax.set_xlabel('WCA Iter. Number',fontproperties=font)
ax.grid(axis='y')
ax.grid(True)
ax.set_facecolor('#ffffff')
plt.xlim(0.0, 100)
plt.savefig("SysIdent\Images\TB\OriginalWCATB45_1", bbox_inches='tight', dpi=600,pad_inches = 0)
plt.show()
```