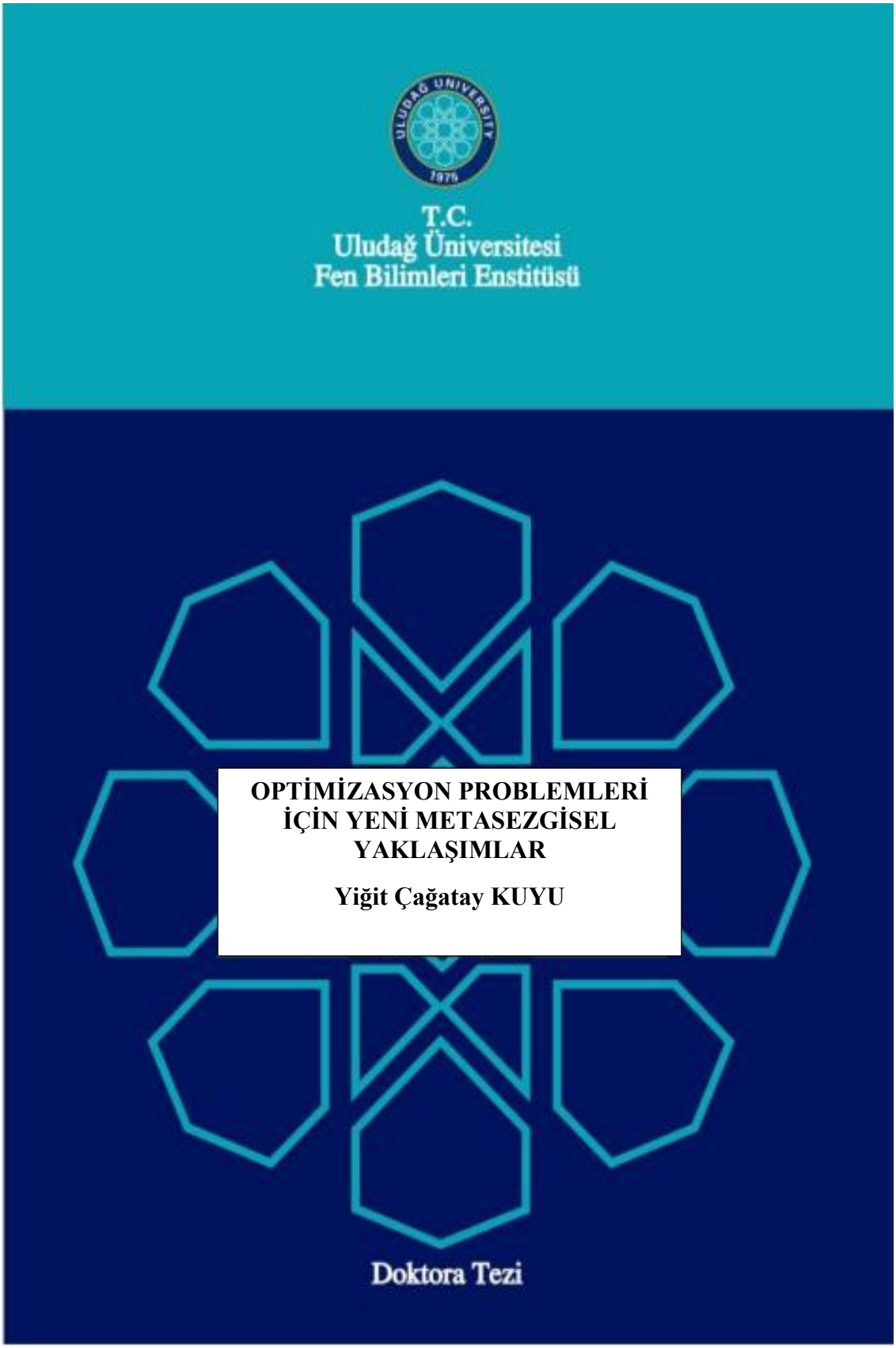

T.C.
Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
OPTİMİZASYON PROBLEMLERİ İÇİN YENİ METASEZGİSEL YAKLAŞIMLAR
Yiğit Çağatay KUYU
Doktora Tezi

OPTİMİZASYON PROBLEMLERİ İÇİN YENİ METASEZGİSEL YAKLAŞIMLAR

Yiğit Çağatay KUYU

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OPTİMİZASYON PROBLEMLERİ İÇİN YENİ METASEZGİSEL YAKLAŞIMLAR

Yiğit Çağatay KUYU
0000-0002-7054-3102

Prof. Dr. Fahri VATANSEVER
0000-0002-3885-8622
(Danışman)

DOKTORA TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2023

TEZ ONAYI

Yiğit Çağatay Kuyu tarafından hazırlanan "OPTİMİZASYON PROBLEMLERİ İÇİN YENİ METASEZGİSEL YAKLAŞIMLAR" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Elektronik Anabilim Dalı'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Fahri VATANSEVER

Başkan : Prof. Dr. Fahri VATANSEVER İmza
0000-0002-3885-8622
Bursa Uludağ Üniversitesi
Mühendislik Fakültesi
Devreler ve Sistemler Anabilim Dalı

Üye : Doç. Dr. Cemal HANİLÇİ İmza
0000-0002-9174-0367
Bursa Teknik Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi
Telekomünikasyon Anabilim Dalı

Üye : Doç. Dr. Neyir ÖZCAN SEMERCİ İmza
0000-0002-5513-9072
Bursa Uludağ Üniversitesi
Mühendislik Fakültesi
Devreler ve Sistemler Anabilim Dalı

Üye : Doç. Dr. Sait Eser KARLIK İmza
0000-0001-5985-210X
Bursa Uludağ Üniversitesi
Mühendislik Fakültesi
Telekomünikasyon Anabilim Dalı

Üye : Dr. Öğr. Üyesi Ömer ZOR İmza
0000-0001-6461-9812
Bursa Teknik Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi
Elektromanyetik Alanlar ve Mikrodalga Tekniği Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
..../..../…

B.U.Ü. Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

12/02/2023

Yiğit Çağatay KUYU

imza

TEZ YAYINLANMA

FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezin/raporun tamamını veya herhangi bir kısmını, basılı (kâğıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma izni Bursa Uludağ Üniversitesi'ne aittir. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları ile tezin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları tarafımıza ait olacaktır. Tezde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alınarak kullandığını ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederiz.

Yükseköğretim Kurulu tarafından yayınlanan "**Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge**" kapsamında, yönerge tarafından belirtilen kısıtlamalar olmadığı takdirde tezin YÖK Ulusal Tez Merkezi / B.U.Ü. Kütüphanesi Açık Erişim Sistemi ve üye olunan diğer veri tabanlarının (Proquest veri tabanı gibi) erişimine açılması uygundur.

Danışman Adı-Soyadı Tarih	Öğrencinin Adı-Soyadı Tarih
İmza Bu bölüme kişinin kendi el yazısı ile okudum anladım yazmalı ve imzalanmalıdır.	İmza Bu bölüme kişinin kendi el yazısı ile okudum anladım yazmalı ve imzalanmalıdır.

ÖZET

Doktora Tezi

OPTİMİZASYON PROBLEMLERİ İÇİN YENİ METASEZGİSEL YAKLAŞIMLAR

Yiğit Çağatay KUYU

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Fahri VATANSEVER

Optimizasyon işlemleri, mühendislik problemlerinde en sık karşılaşılan uygulamalardır. Bu tip problemlerin çözümünde klasik yöntemlerin yanında metasezgisel algoritmalardan da yoğun şekilde yararlanılmaktadır. Bu amaç doğrultusunda da birçok metasezgisel algoritma geliştirilmiş ve geliştirilmeye de devam edilmektedir.

Bu tez çalışmasında; geometrik sekizli bölge arama algoritması (GSBA), modifiye adli tıp temelli soruşturma algoritması (modFBI) ve modifiye hiyerarşik yığın tabanlı optimizasyon algoritması (HBO-CO) olarak adlandırılan üç adet metasezgisel algoritma önerilmiştir. GSBA, böl ve yönet prensibinden ilham alırken, Cauchy tabanlı mutasyon ve karşıtlık temelli öğrenme diğer iki algoritmaya adapte edilmiştir. Önerilen metasezgisel algoritmalar, iki problem setine uygulanmış ve karşılaştırmalı biçimde test edilerek ayrıntılı performans değerlendirmeleri (Wilcoxon, Friedman testleri vb.) gerçekleştirilmiştir: birincisi sayısal (tek modlu, çok modlu ve yüksek boyutlu) fonksiyonlar ve ikincisi de 22 tane gerçek dünya problemleridir. Karşılaştırmalarda 5'i klasik ve 7'si güncel olmak üzere 12 tane algoritma kullanılmıştır. Gerçek dünya problemlerinde özellikle Elektrik-Elektronik mühendisliği uygulama alanına ait problemler (filtre tasarımları, harmonik eliminasyonları, dinamik ve statik ekonomik yük dağıtımları, parametre kestirimi, anten dizisi tasarımı vb.) ilgili algoritmalara uyarlanmış ve uygulanmıştır.

Gerçekleştirilen testlerde önerilen GSBA algoritması, her iki problem setinde istatistikî olarak başarılı sonuçlara ulaşmıştır. Önerilen diğer iki algoritmanın; HBO-CO ve modFBI, nümerik sonuçlara göre kıyaslanan algoritmalardan daha iyi olduğu gözlemlenmiştir. Diğer yandan, gerçek dünya problemlerinde, bu üç algoritma arasında baskın bir algoritma olmayıp, bir problemde en iyi performansı gösteren algoritma diğer bir problemde en iyi olmayabilmektedir.

Anahtar Kelimeler: Optimizasyon, metasezgisel algoritma, gerçek dünya problemi.

2023, ix + 140 sayfa.

ABSTRACT

PhD Thesis

NOVEL METAHEURISTIC APPROACHES FOR OPTIMIZATION PROBLEMS

Yiğit Çağatay KUYU

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Prof. Dr. Fahri VATANSEVER

Optimization techniques are among the prevalent methods utilized in resolving engineering challenges. In addition to traditional techniques, the application of metaheuristic algorithms has become widespread in addressing complex problem scenarios. To this end, numerous metaheuristic algorithms have been devised and ongoing efforts are being made to enhance their efficacy.

In this thesis, three metaheuristic algorithms called geometric octal zone distance estimation algorithm (GSBA), modified forensic-based investigation algorithm (modFBI) and modified hierarchical heap-based optimization algorithm (HBO-CO) are proposed. The GSBA is inspired by the divide-and-conquer principle, while Cauchy-based mutation and opposition-based learning are adapted to the other two algorithms. The proposed algorithms are implemented and tested on two sets of problems, with a subsequent comparison of their performance. A comprehensive assessment of their performance is conducted (Wilcoxon, Friedman tests, etc.) through two distinct evaluation approaches. The first is numerical (unimodal, multimodal and high-dimensional) functions and the second is 22 real-world problems by comparing with 12 algorithms in total, 5 classical and 7 up-to-date. In real world problems, this study also incorporates problems relevant to the domain of Electrical and Electronics Engineering applications (filter designs, harmonic eliminations, dynamic and static economic load distributions, parameter estimation, antenna array design, etc.).

From the experimental analysis, GSBA has achieved the most successful results for both numerical problem sets according to statictical analyzes. The other two proposed algorithms, which are HBO-CO and modFBI, were observed to be better than or at least comparable to the compared algorithms according to numerical results. On the other hand, in real-world problems, there is no absolute leader among the developed three algorithms. It can be observed that the algorithm that performs best in one problem cannot be the best in another problem.

Key words: Parameter estimation, metaheuristic algorithm, real-world problem.

2023, ix + 140 pages.

ÖNSÖZ VE TEŞEKKÜR

Bu çalışmada bilgisi ve tecrübesi ile her türlü konuda yardımlarını benden esirgemeyen değerli danışmanım Sayın Prof. Dr. Fahri VATANSEVER'e teşekkürlerimi sunarım. Ayrıca tez çalışmam süresince maddi manevi desteklerini benden esirgemeyen başta değerli eşim Sayın Gözde KUYU olmak üzere tüm aile fertlerime teşekkürü bir borç bilirim.

Yiğit Çağatay Kuyu

12/02/2023

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
BWO	Karadul Optimizasyon Algoritması (Black Widow Optimization Algorithm)
ChOA	Şempanze Optimizasyon Algoritması (Chimp Optimization Algorithm)
CS	Guguk Kuşu Arama Algoritması (Cuckoo Search Algorithm)
D	Problemin Boyutu
DE	Diferansiyel Geliş Algoritması (Differential Evolution Algorithm)
FBI	Adli Tıp Temelli Soruşturma Algoritması (Forensic-based Investigation Algorithm)
GA	Genetik Algoritma (Genetic Algorithm)
GSBA	Geometrik Sekizli Bölge Arama Algoritması (Geometric Octal Zone Distance Estimation)
HBO	Hiyerarşik Yığın Tabanlı Optimizasyon Algoritması (Heap-Based Optimizer)
HBO-CO	Modifiye Hiyerarşik Yığın Tabanlı Optimizasyon Algoritması (Modified Heap-Based Optimizer)
HHO	Harris Şahini Optimizasyon Algoritması (Harris Hawks Optimization Algorithm)
HS	Harmoni Arama Algoritması (Harmony Search Algorithm)
lb	Problemin Alt Sınırı
modFBI	Modifiye Adli Tıp Temelli Soruşturma Algoritması (Modified Forensic-based Investigation Algorithm)
MVO	Çoklu Evren Optimizasyon Algoritması (Multi-verse Optimizer)
Ort	Ortalama
PSO	Parçacık Sürü Optimizasyon Algoritması (Particle Swarm Optimization Algorithm)
$rand$	0 ile 1 arasında üretilmiş rastgele sayı
Std	Standart Sapma
SSA	Salp Sürüsü Algoritması (Salp Swarm Algorithm)
ub	Problemin Üst Sınırı
X_{best}	En İyi Birey

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

1.1. Optimizasyon

Optimizasyon kavramı, genel olarak, belirli kriter veya kriterler altında, bu kriterleri sağlayan bir çözüm kümesi elemanlarından en uygun değer veya değerlerin seçilimi olarak tanımlanır (Simon, 2013). Gradient-tabanlı (Ruder, 2016), Newton-tabanlı (Meza, 2011) ve lineer programlama (Dantzig ve Thapa, 2003) gibi klasik optimizasyon teknikleri yerel minimum noktalarda sıkışıp kalabilir ve hızlı yakınsama sorunuyla karşılaşıldığında başarısız olabilmektedir (Deb, 2001; Haupt, 1994). Metasezgisel algoritmalar, geleneksel yolların başarısız olduğu veya yetersiz kaldığı, çözüme giden yeni sonuçların geliştirilmesine ihtiyaç duyulduğu noktalarda sıklıkla başvurulan optimizasyon yöntemlerinden biri olmaktadır. Günümüzde minimum kaynak kullanımı ile maksimum fayda sağlama ihtiyacı, doğal ve sosyal bilimlerde, fizik, matematik, mühendislik ve ekonomi gibi birçok alanda metasezgisel algoritmaların kullanımını yaygınlaştırmaktadır (Sharma ve Kaur, 2020; Meng, Li, Wang, Sait, S ve Yıldız, 2021). Metasezgisel algoritmalarının amacı, tek amaçlı optimizasyonda, problemin arama uzayında bir global bir çözüm noktası bulmaktır. Global çözüm noktası bulunamadığında, çözüm kümesindeki global çözüme en yakın birey veya vektör çözüm olarak kabul edilmektedir. Gerçek dünya problemlerinde, çözüm uzayının global noktası çoğu zaman bilinmemektedir. Bu durumda, mevcut aday çözümler arasında kısıtları ve amaç fonksiyonunu sağlayan birey veya vektör, problemin en yakın çözümü olarak kabul edilmektedir (Bozorg-Haddad, Solgi,Loáiciga ve Loáiciga, 2017).

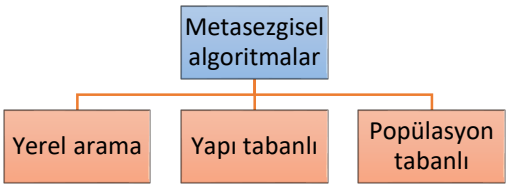
1.2. Metasezgisel Algoritmalar

Optimizasyonda birçok problem zorludur ve aralarından bazıları polinomsal zamanda çözülemez problemler olarak sınıflandırılır. Bunun anlamı, algoritmaların problemin boyutuna polinomik olarak sınırlanmış adımların sayısı dâhilinde, problemin her bir örneğini çözememesidir (Garey ve Johnson, 1979). Diğer yandan matematiksel modeller daha çok küçük boyutlu problemlerde verimli olmaktadır.

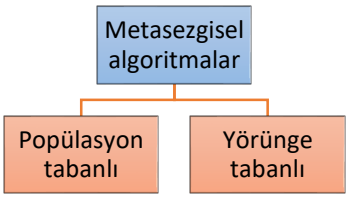
Bunun sonucu olarak yüksek boyutlardaki problemlerin çözümlerinde metasezgisel algoritmalardan da faydalanılmaktadır. Bu algoritmalar, sorunun optimal çözümünü tek

seferde bulamasa bile, optimal çözüme belirli bir çalışma zamanında yakınsayabilmektedirler. Klasik yöntemlerin önemli bir dezavantajı, ilk yerel optimumla karşılaştıktan sonra hareket etmeye devam edememeleridir. Metasezgiseller bu soruna bir çözüm geliştirmek için uyguladıkları işlemlerde rastgelelik barındırmaktadırlar. Bu durum, aday çözümlerin kalitesini daha iyi yapsa da, asla optimal çözüme ulaşılacağını garanti etmemektedir. Fred Glover, metasezgiselleri problem bazlı olmayan algoritma diye tanımlamaktadır (Glover, 1986). Bunun altında yatan sebep metasezgisellerin çeşitlendirme (diversification) ve yoğunlaştırma (intensification) özellikleridir (Blum ve Roli, 2003). Bunun anlamı, metasezgisel algoritma önce olası optimal çözümün bulunduğu alanı keşfeder (çeşitlendirme), daha sonra bu keşfedilen alan üzerinde optimal çözüme yakınsamaya çalışır (yoğunlaştırma). Buna ek olarak, "Bedava Yemek Yok" (No Free Lunch - NFL) teoremine göre, bir metasezgisel algoritma tüm optimizasyon problemlerinde en iyi çözümü bulamaz, yani en iyi tek bir metasezgisel algoritma yoktur (Wolpert ve Macready, 1997). Ayrıca, sürekli, ayrık, çok-amaçlı gibi farklı tipte metasezgiseller vardır. Metasezgisellerin temel karakteristikleri aşağıdaki gibi özetlenebilmektedir (Boussaïd, Lepagnot ve Siarry, 2013).

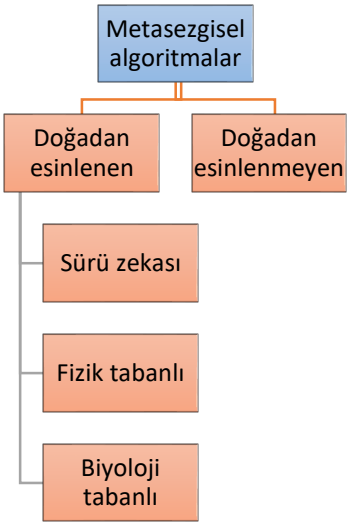
- Metasezgiseller spesifik problemlere entegre edilebilse bile genel olarak tüm problemleri kapsar ve entegre edilebilir.
- Metasezgiseller genellikle yakınsama yapar.
- Metasezgiseller en iyiye en yakın optimal çözümü bulmayı amaçlar.
- Metasezgiseller genellikle paralel kullanıma izin verir.
- Metasezgiseller yerel (local) aramadan genel (global) aramaya kadar geniş bir konsepti kapsar.
- Metasezgisellerin performansı çeşitli arama mekanizmalarıyla arttırılabilir.
- Metasezgiseller genelde geçmiş olası çözümlerden yararlanarak yeni çözümler üretir.



(a)



(b)



(c)

Şekil 1.1 Metasezgisel algoritmaların bazı sınıflandırmaları

Yukarıda da bahsedildiği üzere, metasezgisellerde çeşitlendirme ve yoğunlaştırma arasında bir denge kurmak çok önemlidir. Osman (2003), metasezgiselleri, yerel arama, yapı temelli ve popülasyon temelli olarak sınıflandırmıştır (Şekil 1.1a). Yapı temelli metasezgiseller, eksik bir çözüme her seferinde bir parça ekleyerek, bileşen parçalarından çözümler oluşturmaktadır. Popülasyon temelli metazesgiseller ise olası çözümler kümesini tek bir popülasyonda toplar ve bu popülasyonu iteratif olarak yeni çözümler üretmek için kullanır. Gendreau and Potvin (2005) ise metasezgiselleri ikiye ayırmaktadır; bunlar yörünge tabanlı ve popülasyon tabanlı metasezgiseller olarak adlandırılırlar (Şekil 1.1b). Yörünge tabanlı bir algoritmada, başlangıçta tek bir çözüm

vardır ve her yinelemede mevcut en iyi çözüm yenisiyle değiştirilir. Popülasyon tabanlı algoritma, rastgele bir ilk çözüm popülasyonu oluşturarak başlamaktadır. Ardından ilk popülasyon, arama yinelemeleri yoluyla aşamalı olarak geliştirilmektedir. Her yinelemede, yeni oluşturulan en iyi çözümler tüm popülasyonun veya popülasyonun bir kısmının yerini alır. Genel olarak, yörünge tabanlı algoritmalar daha çok yoğunlaştırmaya yönelirken, popülasyon tabanlı algoritmalar ise daha çok çeşitlendirmeye yönelmektedir. Fisher ve arkadaşları (Fister, Yang, Fister, Brest and Fister, 2013) tüm metasezgisel algoritmaları iki sınıfa ayırmıştır; bunlar doğadan esinlenen metasezgiseller ve doğadan esinlenmeyen metasezgiseller olarak adlandırılmaktadır. Doğadan esinlenen metasezgisel algoritmalar ise kendi içinde, sürü zekası, fizik tabanlı, biyoloji tabanlı ve belirtilen bu üç kategoriye girmeyen algoritmalar olarak sınıflandırmıştır (Şekil 1.1c). Ruiz-Vanoye ve arkadaşları (Ruiz-Vanoye, Díaz-Parra, Cocón, Soto, Ángeles, Arias, Verduzco-Reyes ve Alberto-Lira, 2012) "sürü zekası tabanlı değil" adıyla yeni bir metasezgiselleri sınıflandırma yaklaşımı sunmuştur. Buna rağmen literatürde en çok kullanılan sınıflandırma "doğadan esinlenen" ve "doğadan esinlenmeyen" metasezgiseller olmaktadır.

Biyoloji tabanlı algoritmalar, evrimsel biyolojik prensiplere dayanmaktadır. Özellikle temsil şemalarının doğası (yapı, bileşenler, vb.) bakımından farklılık gösteren çeşitli biyolojik metaforları simüle etmekle ilgilenirler. Üç ana paradigma vardır: evrimsel, sürü ve bağışıklık sistemleri. Evrimsel algoritmalar (EA'lar), daha iyi aday çözümler (kromozomlar) üretmek için seçim, çaprazlama, mutasyon ve üreme operatörlerini kullanarak hücresel düzeyde evrimin biyolojik ilerlemesini simüle etmektedirler. Evrimsel hesaplama için dört temel paradigma vardır; evrimsel programlama (Fogel, Owens ve Walsh, 1966), evrimsel stratejiler (Rechenberg, 1962), genetik algoritma (Goldberg, 1989; Holland, 1975) ve genetik programlama (Koza, 1990). Sürü zekâsı (SI), bir topluluktaki kuşlar ve böcekler gibi canlıların toplu davranışlarını taklit etmektedirler. SI yönteminde aday çözümler birbirleriyle ve çevreleriyle yerel etkileşim yoluyla güncellenmektedir. Literatürde sıklıkla rastlanan SI algoritmaları arasında parçacık sürüsü optimizasyonu (Particle Swarm Optimization - PSO) (Eberhart ve Kennedy, 1995) ve karınca kolonisi optimizasyonu (Ant Colony Optimization - ACO) (Dorigo, 1992) bulunmaktadır. Yapay bağışıklık sistemleri (Artificial Immune Systems - AIS) ilhamını teorik immünolojiden ve gözlemlenen bağışıklık fonksiyonları, ilkeleri ve modellerinden

almaktadır (De-Castro, 2002). AIS tabanlı metasezgiseller, negatif seçim algoritmaları (Dasgupta, 2007), klonal seçim algoritması (Clonal Selection Algorithm - CLONALG) (De-Castro ve Zuben, 2000), yapay bağışıklık ağının optimizasyon versiyonu (Castro, De-Castro ve Timmis, 2002) ve B-hücre algoritması (Kelsey ve Timmis, 2003) olarak gösterilebilmektedir.

Darwin'in "en güçlü olanın hayatta kalması" ilkesi, biyolojik evrim mekanizmasını tanıtmanın başlangıç noktasıdır. Genetik algoritma (Genetic algorithm - GA) (Goldberg, 1989; Holland, 1975), seçim, çaprazlama ve mutasyon gibi süreçleri tanımlayarak kromozomların biyolojik evrim sürecini taklit etmektedir. Kromozomlar belirli bir problem için aday çözümler olarak ele alınır ve uygunluklarına göre değerlendirilir. Yetiştirme için ebeveyn seçimi, yeni çözümler üretmek için etkili bir süreçtir. Rulet çarkı seçimi (Roulette Wheel Selection - RWS), turnuva seçimi (Tournament Selection-TOS), doğrusal sıra seçimi gibi farklı seçim şemaları bulunmaktadır. Çaprazlama aşaması sırasında seçilen iki kromozomun bazı kısımları değiştirilir. Kromozom parçaları; bir, iki ve tek tip geçiş gibi farklı şekillerde değiştirilebilir. Mutasyonda bazı kromozomların parçaları, yerel optimalden kaçmak için rastgele değiştirilir. Bununla birlikte, yeni kromozomlar oluşturulurken en iyi kromozomlar kaybedilebilmektedir. Bir başka biyoloji tabanlı algoritma olan klonal seçim algoritmasında (CLONALG) (De-Castro ve Zuben, 2000) rastgele bir antikor popülasyonu (aday çözümler) oluşturulur ve uygunluk değerleri hesaplanır. Sonra daha yüksek afiniteli antikorlar, antijenlere karşı daha fazla antikor üretmek için klonlanır. Klonlanmayan antikorlar yenileri ile değiştirilir. Bağışıklık kazanmak için en iyi çözümler bir hafıza hücresinde tutulur.

Harmoni arama algoritması (Harmony Search Algorithm - HS) (Gem, Kim ve Loganathan, 2001), müzikal doğaçlamayı temel alan bir algoritmadır. Müzikal doğaçlamada; her enstrümantalist, birlikte bir armoni vektörü oluşturan olası aralıktaki herhangi bir tonu çalar. Tüm ton muhteşem bir armoni yaratırsa, bu deneyim her oyuncunun hafızasında saklanır ve bir dahaki sefere iyi bir armoni çalma olasılığı artar. HS'de her aday çözüm bir vektördür. Bir çözüm vektörünün tüm değerleri iyi bir uyuma (yüksek uygunluk) sahipse, bunlara bağlı her değişken bellekte korunur ve bir dahaki sefere iyi bir çözüm yapma olasılığı da artar. Diğer bir müzikten ilham alan algoritma, müzik kompozisyonu yöntemi (Method of Musical Composition - MMC) (Mora-Gutiérrez, Ramírez-Rodríguez ve Rincón-García, 2014), bestecilerin bir müzik bestesi

oluşturmak için kendi aralarında ve toplulukları arasında bilgi alışverişinde bulundukları bir müzik topluluğunun simülasyonudur. HS gibi, ana fikir bestecinin melodilerinin katılımıyla iyi bir müzik eseri yaratmaktır. MMC'de her bir aday çözüm, karar değişkenlerinin n boyutlu bir vektörü olan bir besteci ayarıyla sembolize edilir. Başlatma aşamasında, her besteci bir skor matrisinde saklanan bir dizi rastgele melodi üretir. Daha sonra etkileşim kurallarına göre bilgi alışverişi yapılır. Başka bir deyişle, değişim ancak ve ancak değiştiriciler arasında bir bağlantı varsa ve ilk bestecinin en kötü melodisi, ikinci bestecinin en kötü melodisinden daha iyi olduğunda yapılır. Bu aşama iki alt aşamaya ayrılmıştır ve bağlantı güncelleme ve bilgi alışverişi olarak adlandırılır. Bu fazlarda, her besteciye, puan matrisinden ve çevreden alınan bilgilerden oluşan bilgi matrisini (Knowledge Matrix - KM) oluşturmak için çevreleri hakkında bilgi verilir. Daha sonra KM'ye göre yeni bir melodi oluşturulur ve her besteci için puan matrisi güncellenmektedir.

Temel optimizasyon algoritması (Base Optimization Algorithm - BOA) (Salem, 2012), matematik temelli bir algoritmadır ve aritmetik operatörlerin kombinasyonuna dayanmaktadır. BOA'da rastgele bir çözüm popülasyonu oluşturulur ve yer değiştirme parametresi tanımlanır. Daha sonra, her çözüm değerlendirilir ve dört olası çözüm bulunur. Bunlar $[+,-,\times,\div]$ ile ifade edilir. Sinüs kosinüs algoritması (Sine Cosine Algorithm - SCA) (Mirjalili, 2016), bir diğer matematik temelli algoritma olup sinüs ve kosinüs denklemlerinin kullanımından yararlanmaktadır. Popülasyon temelli metasezgisellere benzer şekilde, bu algoritma, popülasyonu rastgele çözümlerin dağılımıyla oluşturur ve mevcut en iyi çözümü korur. Algoritmada sinüs ve kosinüs fonksiyonlarının aralığının değiştirilmesi, bir çözüm konumunun güncellenmesi şeklinde yansımaktadır. Başka bir deyişle; çeşitlendirme ve yoğunlaştırma arasındaki denge, sinüs ve kosinüs işlevlerinin aralığını değiştirerek elde edilmektedir.

Fizik tabanlı algoritmalara örnek olarak Kirkpatrick ve arkadaşlarının 1983 yılında geliştirdiği tavlama benzetimi (Simulated Annealing - SA) algoritması örnek gösterilebilir (Kirkpatrick, Gelatt ve Vecchi, 1983). SA adını fiziksel tavlama sürecinden almaktadır. Böylece tavlamanın ilk durumundaki gibi algoritma yumuşaktır ve daha kötü bir çözüme geçebilmektedir. Bu algoritma, çalışma prensibi itibariyle yerel çözümlerden atlama yeteneğine sahiptir ve rastgelelik özelliğini içinde bulundurduğu Markov zinciri

vasıtasıyla barındırmaktadır. Yerçekimsel arama algoritması (Gravitational Search Algorithm - GSA) (Rashedi, Nezamabadi-Pour ve Saryazdi, 2009), Newton'un yerçekimi ve hareket kanunlarından esinlenmektedir. GSA'da her bir aday çözüm, dört özelliğe sahiptir. Bunlar; konum, eylemsizlik kütlesi, aktif yerçekimi kütlesi ve pasif yerçekimi kütlesi olarak kabul edilmektedir. Nesnelerin konumları, çözümleri temsil eder ve uygunlukları kütleleri ile ölçülür. Daha büyük bir yerçekimi kütlesine sahip daha yüksek performanslı nesneler, büyük bir etkili çekim yarıçapına ve dolayısıyla büyük bir çekim yoğunluğuna sahiptir. Dolayısıyla nesneler en iyi çözümlere doğru hareket etme eğilimindedir.

Öğretme-öğrenmeye dayalı optimizasyon algoritması (Teaching–Learning-Based Optimization - TLBO) (Rao, Savsani ve Vakharia, 2011), sosyal tabanlı bir algoritmadır. Bu algoritma; klasik bir okul öğrenme sürecinde, öğretmenlerin öğrenciler üzerindeki etkisini simüle etmektedir; yani öğretmen (en iyi çözüm) bilgilerini öğrencilerle (çözüm popülasyonu) paylaşır ve öğretim kalitesinin öğrenci notları üzerindeki etkisi uygunluk değerlerini temsil eder. TLBO öğrenme süreci iki ana aşamaya ayrılmıştır; öğretmen aşaması ve öğrenci aşaması. Bu aşamalarda öğretmen olmak için en iyi çözüm seçilmekte ve öğrencilerin pozisyonlarının ortalaması hesaplanarak öğretmenin konumuna doğru kaydırılmaktadır. Diğer bir sosyal tabanlı algoritma lig şampiyonası algoritmasıdır (League Championship Algorithm - LCA) (Kashan, 2014). LCA; birkaç takımın bir ligde birkaç sezon oynadığı bir spor şampiyonasını taklit etmektedir; yani gerçek bir spor şampiyonası gibi bir lige (çözüm popülasyonu) birkaç takım (çözüm) katılır ve çiftler halinde rekabet eder. Ardından maç, güçlü/zayıf yönler/fırsatlar/tehditler (SWOT) şeklinde analiz edilir. Kazanan takım, takım oyuncularının iyi dağılımına ve oynama gücüne (uygunluk değeri) göre belirlenmektedir. Her takım, oyuncularını yeniden düzenlemekte ve oyun tarzını değiştirerek performansını artırmaya çalışmaktadır. LCA, başlangıç popülasyonunun başlatılmasıyla başlar. Ardından her sezonda bir lig planlanır, yani her takımın başka bir katılımcıyla yarıştığı tek bir "round-robin" programı kullanılır. Kazanan/kaybeden takımların belirlenmesi rastgele yapılır. Sonraki maç için en iyi dizilişin formülasyonu, mevcut en iyi diziliş ve önceki hafta SWOT analizi vasıtasıyla yapılır.

Tabu arama algoritması (Tabu Search - TS) (Glover ve McMillan, 1989), yinelemeli araması nedeniyle EA olarak sınıflandırılabilmektedir. TS'de yasaklanmanın ana fikri, çeşitlendirmeyi teşvik etmek için daha önce ziyaret edilen arama alanının tekrar ziyaret edilmesinde, "tabu veya tabu olmamasından" kaynaklanmaktadır. Yani çözüm alanını keşfetmek ve yerel optimumda sıkışıp kalmamak için TS, herhangi bir yerel arama prosedürünü yoğun bir şekilde uygular. TS'de iki ana özellik vardır; uyarlanabilir bellek ve duyarlı keşif. İlki (tabu listesi olarak adlandırılır), döngü içindeki kuşatmayı önlemek için arama işlemi sırasında gerçekleştirilen eylemlerin geçmişini saklar. İkincisi, arama sürecinin daha fazla yoğunlaştırma için iyi bölgelere ve en iyi çözüme odaklanmasını sağlamak ve daha fazla keşif için gelecek vaat eden yeni bölgeleri keşfetmek için birincisini kullanır. Değişken komşuluk arama algoritması (Variable Neighborhood Search - VNS) (De-Castro, 2002), bir diğer yinelemeli algoritma olup EA sınıfı içerisindedir. Bu algoritmada ana fikir, en uygun (veya optimuma yakın) bir çözüm arayışı sırasında, birkaç komşuluğu sistematik veya rastgele olarak keşfetmektir. VNS, dönüşümlü olarak iki iyileştirme prosedürünün uygulanmasına dayanmaktadır. Birincisi, yerel minimumlardan çıkmak için yapılan "sallama"dır. Ayrıca, bu komşulardan yerel olarak en uygun olanı veya birkaç komşuluk yapısını araştıran daha gelişmiş prosedürleri elde etmek için, basit bir yerel arama yöntemi uygulanmaktadır. İkinci adım, VNS'nin mevcut çözümün rastgele bir şekilde uzaktaki komşuluklarını araştırdığı ve ancak bir iyileştirme yapılırsa oradan yenisine geçtiği "komşuluk değişimi"dir. Bu şekilde, iyi değişkenler korunur ve gelecek vaat eden komşular elde edilir.

Metasezgisel algoritmalar, kendi içlerinde farklı kümelere ayrılabilmektedir. Örneğin, uyarlanabilir (adaptif) metasezgisellerde rastgele adım boyutu veya arama aralığı, erken yakınsamayı önlemek için algoritma ilerlemesine göre otomatik olarak ayarlanabilir. Max–min karınca sistemi (Stützle ve Hoos, 2000), uyarlanabilir parçacık sürüsü optimizasyon algoritması (Chunxia ve Youhong, 2008), uyarlanabilir benzetilmiş tavlama (Oliveira, Rwmbold, Petraglia, Ingber ve Augusta 2012), uyarlanabilir ateşböceği algoritması (Cheung, Ding, Shen, 2014), değiştirilmiş guguk kuşu arama (Walton, Hassan, Morgan ve Brown, 2011), balina optimizasyonu algoritması (Trivedi, Pradeep, Narottam, Arvin ve Dilip, 2016), geliştirilmiş gri kurt algoritması (Kumar, Kumar and Chhabra, 2016) bunlara örnektir. Diğer bir örnekse kaotik metasezgiseller olarak adlandırılmaktadır. Kaos haritaları farklı zamana sahip başlangıç durumuna bağlı

olarak deterministtik sınırlı rastgele sayılar dizisi üreten evrim işlevleridir. Ayrıca kaos haritaları; rastgele tabanlı, periyodik olmayan ve parametre uyarlaması için yakınsak olmadıkları için metasezgisel algoritmaların rastgele sayı oluşturucularının yerini alabilirler. Son zamanlarda, daha fazla rastgelelik ve yüksek yakınsama oranı kullandıklarından kaos temelli metasezgiseller yaygındır. Bunlara örnek olarak, kaotik parçacık sürüsü optimizasyonu (Hefny ve Azab, 2010), kaotik uyum araması (Alatas, 2010), kaotik genetik algoritma (Snaselova ve Zboril, 2015), kaotik lig şampiyonası algoritması (Bingol ve Alatas, 2016) gösterilebilmektedir. Gauss dağılım tabanlı metasezgiseller de literatürde sıklıkla yer alan algoritmalardandır. Bunlar arasında çıplak kemik parçacık sürüleri (Kennedy, 2003), Gauss ateşböceği algoritması (Farahani, Abshouri, Nasiri ve Meybodi, 2011), Gauss çıplak kemiklerin diferansiyel evrimi (Wang, Rahnamayan, Sun, ve Omran, 2013), çıplak kemikler öğretme-öğrenmeye dayalı optimizasyon (Zou, Wang, Hei, Chen, Jiang ve Li, 2014) algoritmaları gösterilebilmektedir. Genel sınıflandırmalarıyla birlikte bazı metasezgisel algoritmalar Çizelge 1.1-1.4'te verilmektedir.

Çizelge 1.1. Biyoloji tabanlı metasezgisel algoritmalar

Biyoloji Tabanlı Metasezgiseller	
Algoritma	**Referans**
Genetik Algoritma (GA)	(Goldberg, 1989)
Karınca Kolonisi Optimizasyonu (ACO)	(Dorigo, 1992)
Parçacık Sürüsü Optimizasyonu (PSO)	(Eberhart ve Kennedy, 1995)
Diferansiyel Gelişim Algoritması (DE)	(Storn ve Price, 1997)
Yapay Arı Kolonisi Algoritması (ABC)	(Dervis, Basturk, 2007)
Ateşböceği Algoritması (FF)	(Yang, 2008)
Biyocoğrafya Tabanlı Optimizasyon (BBO)	(Simon, 2008)
Guguk Kuşu Arama Algoritması (FF)	(Yang ve Deb, 2009)
Yarasa Algoritması (BA)	(Yang, 2010)
Krill Sürü Algoritması (KSA)	(Gandomi ve Alavi, 2012)
Çiçek Tozlaşma Algoritması (FPA)	(Yang, 2012)
Geri-izleme Arama Optimizasyon (BSA)	(Civicioglu, 2013)
Simbiyotik Organizmalar Arama Algoritması (SOS)	(Cheng ve Prayogo, 2014)
Gri Kurt Optimize Edici (GWO)	(Mirjalili, Mirjalili ve Lewis, 2014)
Sosyal Örümcek Optimizasyonu (SSA)	(James ve Victor, 2015)
Aslan Optimizasyon Algoritması (LOA)	(Yazdani ve Jolai, 2015)
Güve-Alev Optimizasyon Algoritması (GOA)	(Mirjalili, 2015)
Kelebek Algoritması (KOA)	(Arora ve Singh, 2015)

Virüs Kolonisi Arama Algoritması (VKAA)	(Li, Zhao, Weng ve Han, 2016)
Balina Optimizasyon Algoritması (BOA)	(Mirjalili ve Lewis, 2016)
Grasshopper Optimizasyon Algoritması (GOA)	(Saremi, Mirjalili ve Lewis, 2017)
Karadul Optimizasyon Algoritması (BWO)	(Vahideh ve Kazem, 2020)
Harris Şahinleri Optimizasyonu (HHO)	(Heidari, Mirjalili, Faris, Aljarah, Mafarja ve Chen, 2020)
Salp Sürü Algoritması (SSA)	(Mirjalili, Gandomi, Mirjalili, Saremia, Faris ve Saremia, 2020)

Çizelge 1.2. Fizik tabanlı metasezgisel algoritmalar

Fizik Tabanlı Metasezgiseller	
Algoritma	**Referans**
Tavlama Benzetimi (SA)	(Kirkpatrick, Gelatt ve Vecchi,1983)
Elektromanyetizma Benzeri Optimizasyon Algoritması (EO)	(Birbil ve Fanf , 2003)
Büyük Patlama Büyük Çöküş Optimizasyon Algoritması (BBO)	(Erol ve Eksin, 2006)
Yerçekimsel Arama Algoritması (GSA)	(Rashedi ve diğerleri, 2009)
Kara Delik Algoritması (BOA)	(Hatamlou, 2013).
Girdap Arama Algoritması (VS)	(Dogan ve Ölmez, 2015)
Isı Transferi Araması Algoritması (HTA)	(Patel ve Savsani, 2015)
Optiklerden Esinlenen Optimizasyon Algoritması (OA)	(Kashan, 2015)
Yıldırım Arama Algoritması (ESA)	(Shareef, Ibrahim ve Mutlag, 2015)
Çoklu Evren Optimize Edici (MVO)	(Mirjalili, Mirjalili ve Hatamlou, 2016)
Elektromanyetik Alan Optimizasyonu (EFO)	(Abedinpourshotorban, 2016)
Su Buharlaşma Optimizasyonu (WEA)	(Kaveh ve Bakhshpoori, 2016)
Galaktik Sürü Optimizasyonu (GSA)	(Muthiah-Nakarajan ve Noel, 2016)

Çizelge 1.3. Sosyal tabanlı metasezgisel algoritmalar

Sosyal Tabanlı Metasezgiseller	
Algoritma	**Referans**
Kültürel Algoritma (CA)	(Reynolds, 1994)
Emperyalist Rekabet Algoritması (ECA)	(Atashpaz-Gargari ve Lucas, 2007)
Öğretme-Öğrenmeye Dayalı Optimizasyon (TLBO)	(Rao ve diğerleri, 2011)
Beyin Fırtınası Optimizasyon Algoritması (BSA)	(Shi, 2011)
Yapay Kabile Algoritması (ACA)	(Chen, Wang, ve Li, 2012)
İç Arama Algoritması (ISA)	(Gandomi, 2014)
Lig Şampiyonası Algoritması (LCA)	(Kashan, 2014)
Borsa Piyasası Algoritması (EA)	(Ghorbani ve Babaei, 2014)
Yapay Bulaşıcı Hastalık Optimizasyon Algoritması (YBHA)	(Huang, 2016)
Yin-Yang-çifti Optimizasyonu	(Punnathanam ve Kotecha, 2016)
İnsan Zihinsel Arama Algoritması (IZA)	(Mousavirad ve Ebrahimpour-Komleh, 2017)
Kohort Zeka Algoritması (KZA)	(Kale ve Kulkarni, 2018)
Şempanze Optimizasyon Algoritması (ChOA)	(Khishe ve Mosavi, 2020)

Çizelge 1.4. Matematik tabanlı metasezgisel algoritmalar

Matematik Tabanlı Metasezgiseller	
Algoritma	**Referans**
Matheuristic	(Boschetti, M. A., Maniezzo, V., Roffilli, M., ve Röhler, 2009)
Temel Optimizasyon Algoritması (TOA)	(Salem, 2012)
Sinüs Kosinüs Algoritması (SKA)	(Mirjalili, 2016)
Simule Edilmiş Kalman Filter Algorithm (SKFA)	(Ibrahim, Aziz, Aziz, Razali, ve Mohamad , 2016)
Altın Sinüs Algoritması (ASA)	(Tanyildizi ve Demir, 2017)

1.3. Metasezgisel Algoritmaların Uygulama Alanları

Metasezgisel algoritmalar, nümerik problemlerin çözümünün yanı sıra, birçok gerçek dünya optimizasyon probleminde kullanılmaktadır. Metasezgisellerin uygulama alanları arasında, iletişim, görüntü ve sinyal işleme, çizelgeleme problemleri, çok büyük ölçekli entegrasyon tasarımı ve finansal planlama yer almaktadır. Literatürde sıklıkla kullanılan algoritmalardan genetik algoritma, planlama (Datta, Amaral ve Figueira, 2011), dizilim (Lee, 2018), kontrol (Lee, 2018), yapay zeka (Chen, Liang, Hong ve Gu D-X, 2015), görüntü işleme (Chouhan, Kaul ve Singh, 2018), video işleme (Alkhafaji, Salih, Nabat ve

Shnain, 2020), medikal imgeleme (Kavitha ve Chellamuthu, 2016), oyun (Junru ve Lan, 2014), kablosuz ağ (Lorenzo ve Glisic, 2013), yük dengeleme (Ekbatanifard, Monsefi, Akbarzadeh-T, ve Yaghmaee, 2010), konumlandırma (Yun, Lee, Chung, Kim ve Kim, 2009) ve band genişliği yerleşimi (Kandavanam, Botvich, Balasubramaniam ve Jennings, 2010) gibi bir çok farklı uygulamalarda kullanılmıştır. Bir diğer popüler algoritmalardan olan diferansiyel gelişim algoritması, yakıt etanol üretiminde bulanık karar verme problemleri (Wang, Jing ve Tsao, 1998), bulanık mantık denetleyicisinin tasarımı (Sastry, Behera ve Nagrath, 1998), toplu mayalanma süreci (Wang ve Cheng, 1999), çoklu sensör füzyonu (Joshi ve Sanderson, 1999), sürekli polimer reaktörün dinamik optimizasyonu (Lee, Han ve Chang, 1999), damlama yataklı reaktörde ısı transfer parametrelerinin tahmini (Babu ve Sastry, 1999), alkilasyon reaksiyonunun optimizasyonu (Babu ve Chaturvedi, 2000), eşanjörlerin optimum tasarımı (Babu ve Munawar, 2000) ve yapay zeka (Ilonen, Kamarainen ve Lampinen, 2003) gibi problemler üzerinde uygulanmış ve hala birçok probleme uygulanmaya devam etmektedir (Osman ve Laporte, 1996).

Çizelge 1.5. Bazı metasezgisel algoritmaların uygulamaları

SA	Üretim sisteminde atölye planlaması, nakliye ve lojistik yönetiminde araç rotası, iletişim: mobil ağ tasarımı, yönlendirme, kanal tahsisi.
TS	Kaynak tahsisi, mühendislik teknolojisi: hücre yerleştirme, güç dağıtımı, yapısal tasarım, yapay zeka: örüntü tanıma, veri madenciliği, kümeleme.
GA	Finansal planlama, stok tahminleri, görüntü işleme: esnek yönetim sistemlerinde sıkıştırma, bölümleme, sıralama.
DE	İşaret ve görüntü işleme, ürün tasarımı: aerodinamik, kimya mühendisliği.
PSO	Telekomünikasyon: ağ tasarımı, yönlendirme, sinir ağı eğitimi, sistem simülasyonu ve tanımlama, karar verme ve planlama, sinyal işleme.
ACO	Rotalama çözümleri, planlama, endüstriyel uygulamalar, öznitelik seçimi, bilgisayar ağları, optimal tasarım/atama problemleri.
HS	Endüstri, güç sistemleri, sinyal ve görüntü işleme, kümeleme, robotik.
GSA	Güç akışı, enerji yönetim sistemleri, kümeleme, öznitelik seçimi, sınıflandırma.

Çizelge 1.5'te Metasezgisellerin geniş uygulama alanları verilmiştir. Metasezgisel algoritmaların elektrik-elektronik mühendisliği alanında da geniş kapsamlı uygulamaları bulunmaktadır. Yapay arı kolonisi algoritması, karınca kolonisi optimizasyonu, guguk kuşu arama algoritması (CS), ateşböceği algoritması, genetik algoritma, harmoni arama algoritması (HS), parçacık sürü optimizasyon algoritması ve tavlama benzetimi algoritması, 14. dereceden sonlu darbe cevaplı (FIR) alçak geçiren filtre tasarımında kullanılmıştır. Yapay arı kolonisi algoritmasının diğer karşılaştırılan metasezgisel algoritmalara kıyasla en az hatayla filtre katsayılarını bulabildiği görülmektedir (Kuyu ve Vatansever, 2016). Sonsuz dürtü cevaplı (IIR) filtre 12. dereceden alçak ve yüksek geçiren filtre tasarımları, geri-izleme optimizasyon algoritması (BS), elektromanyetik alan optimizasyon algoritması (EFO) ve girdap arama algoritması ile gerçeklenmektedir. Çalışmanın nümerik sonuçlarına göre her iki filtre tasarımında EFO karşılaştırılan algoritmalar arasından en iyi sonucu vermektedir (Kuyu ve Vatansever, 2017). Mühendislikte kontrolör tasarımlarında metasezgisel algoritmaların uygulamaları mevcuttur. PSO, alkali yakıt hücresi PID kontrollerinin entegrasyonunu optimize etmek için kullanılmıştır (Dwivedi ve Kumar, 2017). Bat algoritması önerilen PID sisteminin optimal tasarımında kullanılmış ve performansı bu tasarımda ateş böceği algoritması ile kıyaslanmıştır (Chaib, Choucha ve Arif, 2017). ABC, GA ve PSO, tank reaktörü için, uyarlanabilir PID tasarımında kullanılmış ve karşılaştırmalı performans analizleri yapılmıştır (Goud ve Swarnkar, 2019). Güneş enerjisi panellerinde sıklıkla kullanılan fotovoltaik hücre modellemesinde metasezgisel algoritmalar bir çözüm yöntemi olmaktadır. Bu modellemede doğru parametre tanımlaması önem taşımaktadır. GA, bu probleme çözüm getirmek amacıyla kullanılan temel algoritmalar arasında yer almaktadır (Zagrouba, Sellami, Bounaicha ve Ksouri, 2010; Dizqah, Maheri ve Busawon, 2014). Diğer yandan DE, ABC ve PSO algoritmaları da bu problemler üzerinde literatürde çözüm getiren algoritmalar olmaktadır (Ishaque ve Salam, 2011; Wang, Zhan ve Zhou, 2015; . Ye, Wang ve Xu, 2009). Anten dizisi, tek bir ışıma elemanı oluşturmak için topluca çalışan bir anten elemanları topluluğudur. GA, lineer anten dizilerinin ağırlıklarını optimize etmek için kullanılmıştır (Goswami ve Mandal, 2013). DE ile bu alanda yeniden yapılandırılabilir anten dizisinin optimizasyonu üzerinde çalışılmıştır (Rao ve Sarma, 2017). ACO ve GSA algoritmaları anten dizilerinin optimizasyonunda

kullanılan, literatürde karşımıza çıkan algoritmalardandır (Aelterman, Goossens, Declercq ve Rogier, 2009; Sharma ve Mathur, 2018).

Reaktif güç sistem optimizasyonunda da metasezgisel algoritmalar kendine yer bulmuştur. Yin-Yang-çifti optimizasyon algoritması (YYPO), balina optimizasyon algoritması ve Salp sürü algoritması IEEE 13 ve 30 baralı sistemlerde reaktif güç analizi yapmak için kullanılmaktadır (Kuyu ve Vatansever, 2018). Güç elektroniği alanında sıklıkla kullanılan eviriciler/invertörler, DC-AC gerilim dönüşümü yapabilen devre türlerinin başında gelmektedir. Harmonikler güç kalitesini, enerji/güç sistemlerinde belirleyen parametrelerin başında yer almaktadır. Dolayısıyla evirici devrelerinin çıkışındaki harmoniklerin elenmesi/bastırılması önem arz etmektedir. YYPO, CS, EFO, HS, iç arama ve Harris şahinleri optimizasyon algoritmaları seçilen harmonikleri elemek için en uygun anahtarlama açılarını bulmada kullanılabilmektedir (Vatansever ve Kuyu, 2019).

1.4. Sunulan Bilimsel Katkılar

Bu tez çalışmasında, yeni geometrik sekizli bölge arama algoritması (Geometric Octal Zone Distance Estimation Algorithm - GSBA) (Kuyu ve Vatansever, 2022a) geliştirilmiştir. Ayrıca mevcut iki algoritma üzerinde değişiklikler/iyileştirmeler yapılarak modifiye adli tıp temelli soruşturma algoritması (Modified Forensic-Based Investigation Algorithm - modFBI) (Kuyu ve Vatansever, 2021) ile modifiye hiyerarşik yığın tabanlı optimizasyon algoritması (Modified Hierarchical Heap-Based Optimization Algorithm - HBO-CO) (Kuyu ve Vatansever, 2022b) önerilmiştir. Geliştirilen bu algoritmalar, sayısal fonksiyonlar üzerinde test edilmiş, gerçek dünya ve özellikle de elektrik-elektronik mühendisliği problemlerine uygulanarak karşılaştırmalı performans analizleri gerçekleştirilmiştir. Gerçekleştirilen tezin katkıları aşağıdaki gibi özetlenebilmektedir.

- Bu tez çalışması hem yeni bir metasezgisel algoritma hem de var olan iki algoritmanın farklı yaklaşımlarla orijinallerinden daha iyi performanslı versiyonlarını literatüre kazandırmıştır.

- GSBA algoritması, bilindiği kadarıyla ilk defa, sekiz farklı alt popülasyonun (bölgenin) simültane olarak kullanıldığı algoritmadır.
- GSBA algoritmasının çalışma prensibinde dinamik popülasyon değişimi önerilmiştir ve bu bir alt popülasyonun her bir iterasyonda farklı metotlarla geliştirilmesine imkan sağlamıştır.
- Cauchy tabanlı mutasyon (CBM) ve karşıtlık temelli öğrenme (OBL) metotlarının, FBI algoritmasının A ve B fazlarında birbirine paralel olarak kullanılması yaklaşımı ilk defa bu tezdeki modFBI algoritmasıyla önerilmiştir.
- HBO algoritmasına başlangıç popülasyonun geliştirilmesinde OBL metodunun uygulanması ilk defa sunulmuştur.
- HBO algoritmasının çözümleri iyileştirme sürecine, CBM ilk defa entegre edilmiştir.
- Sunulan üç algoritmanın gerek nümerik fonksiyonlar gerekse gerçek dünya problemleri üzerinde analizleri yapılmıştır. GSBA algoritması, kullanılan nümerik problem setlerinde istatistiki olarak 3.217 ve 3.950 rank değerleri daha başarılı sonuçlara ulaşmıştır. Diğer iki modifiye algoritma ise analog filtre tasarım problemlerinde, 0.008, 0.012 ve 0.002 hata değerleri ile en iyi hatalara ulaşmışlardır.

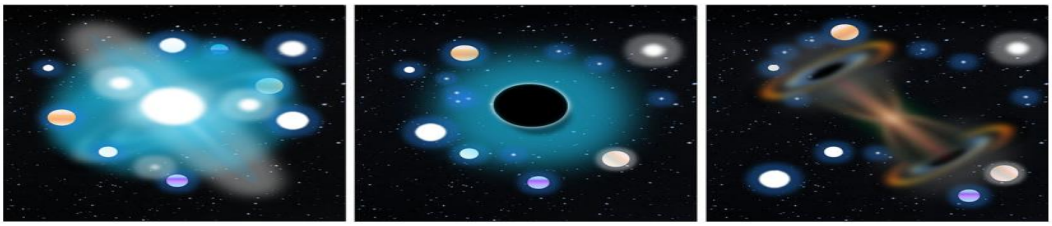
2. METASEZGİSEL ALGORİTMALAR

Metasezgisel algoritmalar, herhangi bir amacı gerçekleştirmek üzere çoğunlukla doğal olgulardan ilham alan algoritmalardır. Bu bölümde, tezde karşılaştırma amaçlı kullanılan popülasyon tabanlı metasezgisel algoritmaların çalışma prensipleri özetlenecektir.

2.1. Çoklu Evren Optimizasyon Algoritması

Çoklu evren optimizasyon algoritmasının (Multiverse Optimization Algorithm - MVO) (Mirjalili, Mirjalili, ve Hatamlou, 2016) ana ilham kaynakları, kozmolojideki üç konsepte dayanmaktadır: beyaz delik, kara delik ve solucan deliği. Bu kavramların sembolik modelleri Şekil 2.1'de verilmektedir. Bu üç kavramın matematiksel modelleri sırasıyla keşif, kullanma ve yerel arama yapmak üzere geliştirilmiştir. Spesifik olarak, beyaz delik ve kara delik kavramları global aramadan sorumlu iken, solucan deliği kavramı ise yerel aramaya yardımcı olmaktadır. Bu algoritmada her çözümün bir evrene benzediği ve çözümdeki her değişkenin o evrendeki bir nesne olduğu varsayılmaktadır. Buna ek olarak, her çözüme, çözümün karşılık gelen uygunluk fonksiyonu değeriyle orantılı olan bir enflasyon oranı atanmaktadır. MVO algoritmasında yukarıda belirtilen üç evren tipi arasında aşağıdaki kurallar uygulanmaktadır:

- Yüksek enflasyon oranı, daha yüksek bir olasılık ile beyaz deliğe sahip olmak anlamına gelmektedir.
- Yüksek enflasyon oranı, daha düşük bir olasılıkla siyah deliğe sahip olmak anlamına gelmektedir.
- Daha yüksek enflasyon oranına sahip evrenler, beyaz deliğe nesneler gönderme eğilimindedir.
- Daha düşük enflasyon oranına sahip evrenler, kara deliklerden daha fazla nesne alma eğilimindedir.
- Tüm evrenlerdeki nesneler, enflasyon oranından bağımsız olarak solucan delikleri vasıtasıyla en iyi evrene doğru rastgele bir hareketle karşılaşabilmektedirler.



Şekil 2.1. Beyaz delik, siyah delik ve solucan deliği (Mirjalili, Mirjalili, ve Hatamlou, 2016)

İki evren arasında beyaz/siyah bir tünel kurulduğunda, daha yüksek enflasyon oranına sahip evrenin beyaz deliğe sahip olduğu kabul edilirken, daha düşük enflasyon oranına sahip evrenin kendi kara deliklerine sahip olduğu varsayılmaktadır. Nesneler daha sonra kaynak evrenin beyaz deliklerinden hedef evrenin kara deliklerine aktarılmaktadır. Bu mekanizma, evrenlerin nesneleri kolayca değiş tokuş etmesini sağlamaktadır. Evrenlerin tüm enflasyon oranını iyileştirmek için, yüksek enflasyon oranlarına sahip evrenlerin beyaz deliklere sahip olma ihtimalinin yüksek olduğu varsayılmaktadır. Diğer yandan, düşük enflasyon oranlarına sahip evrenlerin kara deliklere sahip olma olasılığı yüksek olmaktadır. Bu nedenle, nesneleri yüksek enflasyon oranına sahip bir evrenden düşük enflasyon oranına sahip bir evrene taşıma olasılığı her zaman yüksek olmaktadır. Bu, tüm evrenlerin ortalama enflasyon oranlarının yinelemelere göre iyileştirilmesini garanti edebilmektedir. Her iterasyonda evrenler enflasyon oranına bağlı olarak sıralanmakta ve beyaz deliğe sahip olmak için bir evren seçilmektedir.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_M \end{bmatrix} \qquad X_i = \left[X_{i,1}, X_{i,2}, \dots, X_{i,D}\right] \qquad i = 1,2, \dots M \tag{2.1}$$

Denklem (2.1)'de X bir popülasyonu belirtmektedir, M evren sayısını ve D ise problemin boyutunu ifade etmektedir.

$$x_i^j = \begin{cases} x_k^j, & r1 < NI(Ui) \\ x_i^j, & r1 \geq NI(Ui) \end{cases} \tag{2.2}$$

Denklem (2.2)'de, x_i^j $i.$ evrenin $j.$ parametresini belirtirken, Ui, i evreni ve $NI(Ui)$ $i.$ evrenin normalize enflasyon oranını ifade etmektedir. $r1$, 0 ile 1 arasında bir rastsal sayıdır. x_k^j rulet çemberi vasıtasıyla seçilen $k.$ evrenin $j.$ parametresini ifade etmektedir.

Enflasyon oranı ne kadar az olursa, nesneleri beyaz/kara delik tünellerinden gönderme olasılığı o kadar yüksek olmaktadır. Evrenlerin nesneleri değiş tokuş etmesi ve arama alanını keşfetmek için ani değişikliklerle karşılaşması gerektiğinden, keşif bu mekanizma kullanılarak garanti edilebilmektedir. Yukarıdaki mekanizma ile evrenler, nesneleri değiş tokuş etmeye devam etmektedirler. Evrenlerin çeşitliliğini korumak ve sömürüyü gerçekleştirmek için, her evrenin nesnelerini rastgele uzayda taşımak için solucan deliklerine sahip olmaktadırlar. Lokal değişiklikler sağlamak amacıyla, solucan delikleri kullanılmaktadır. Enflasyon oranını iyileştirme olasılığının yüksek olması için, solucan deliği tünellerinin her zaman bir evren ile şimdiye kadar oluşan en iyi evren arasında kurulduğu varsayılmaktadır. Bu mekanizmanın formülasyonu Denklem (2.3)'teki gibidir:

$$\begin{cases} \begin{cases} X_j + TDR\left(\left(ub_j - lb_j\right)r4 + lb_j\right) & r3 < 0.5 \\ X_j - \ TDR\left(\left(ub_j - lb_j\right)r4 + lb_j\right) & r3 \geq 0.5 \end{cases} \\ x_i^j \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r2 \geq WEP \end{cases} \tag{2.3}$$

Burada X_j en iyi bulunan evrenin $j.$ parametresini temsil etmektedir. TDR ve WEP denklemin katsayıları olmaktadırlar. $r2, r3\ ve\ r4$ ise 0-1 aralığında rastgele sayılardır. WEP katsayısı solucan deliğinin evrenlerde var olma oranını ifade etmekte ve iterasyonlar boyunca lineer olarak artmaktadır. Adaptif WEP formulasyonu Denklem (2.4)'te verilmektedir:

$$WEP = min + l(\frac{max-min}{L}) \tag{2.4}$$

Yukarıdaki denklemde min değeri 0.2 , max değeri ise 1 olarak alınan katsayılardır. l iterasyon indeksi ve L maksimum iterasyon sayısıdır. TDR katsayısı ise daha çok yerel aramaya odaklanmış olup Denklem (2.5)'teki gibi hesaplanmaktadır.

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{2.5}$$

p burada yerel arama doğruluğudur ve 6 olarak kabul edilmektedir. Daha yüksek p değeri demek daha fazla yerel aramaya odaklanmak anlamına gelmektedir.

MVO algoritmasında, optimizasyon süreci, bir dizi rastgele evren oluşturmakla başlar. Her yinelemede, yüksek enflasyon oranlarına sahip evrenlerdeki nesneler, beyaz/kara delikler aracılığıyla düşük enflasyon oranlarına sahip evrenlere hareket etme eğiliminde olmaktadır. Buna ek olarak, her bir evren, nesnelerinde solucan delikleri aracılığıyla en iyi evrene doğru rastgele ışınlanmalarla karşı karşıya kalmaktadır. Bu süreç, bir sonlandırma kriteri sağlanıncaya kadar yinelenerek devam etmektedir. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Çoklu Evren Optimizasyon Algoritması
1. Başlangıç parametrelerini tanımla. 2. Evrenleri üret. 3. Her bir evren için enflasyon oranını tanımla. 4. Evrenleri enflasyon oranına göre sırala. 5. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi çözümü seç. b. Sağlanmıyor: - Evrenleri güncelle. - 3. Adıma geç.

2.2. Diferansiyel Gelişim Algoritması

Diferansiyel gelişim (Differential Evolution Algorithm - DE) algoritması (Storn ve Price, 1995), popülasyon temelli bir algoritma olup sürekli zaman problemlerinin çözümü için tasarlanmıştır. Bu algoritma; gerek yerel minimuma hızlı bir şekilde yakınsamada, gerekse küresel minimuma ulaşmadaki efektifliği açısından, kendini birçok farklı problemde kanıtlamıştır (Yang 2010). DE algoritmasında, ileride bahsedilecek olan genetik algoritmaya benzer olarak, çaprazlama, mutasyon ve seçilim işlemleri mevcuttur. Fakat bu işlemler, genetik algoritmadakinden birçok açıdan farklı şekilde ele alınmaktadır. DE bahsedilen işlemleri tüm popülasyona uygulayarak yeni aday çözümler üretmektedir. Popülasyon içerisinde bulunan her bir bireyin uygunluk değeri hesaplanmakta ve bu uygunluk değerlerine göre optimal çözümü arama süreci yönlendirilmektedir.

Tüm popülasyon tabanlı metasezgisel algoritmalara benzer olarak, aday çözüm kümesi olarak tanımlanan bir başlangıç popülasyonu öncelikle rastgele işlemler vasıtasıyla üretilmektedir. Bu üretim, problemin türünün sınırlarına bağlı olarak, tasarım değişkenlerine üst ve alt sınır değerleri atanarak, Denklem (2.6)'daki gibi başlangıç popülasyonu oluşturulmaktadır:

$$X_{i,j}^{0} = L_j + rand * \left(U_j - L_j\right),\ i = 1, \dots, NP,\ j = 1, \dots, ND \tag{2.6}$$

Burada, NP popülasyondaki toplam kromozom sayısını, ND değişken sayısını, X^0 başlangıç popülasyonunu, $U\ ve\ L$ aday çözümlerin değişkenlerinin alabileceği maksimum ve minimum sınırları, $rand$ ise sıfır ile bir aralığında üretilen rastgele sayıyı belirtmektedir. Başlangıç popülasyonu oluşturulduktan sonra uygunluk değerleri hesaplanır ve popülasyon sırasıyla mutasyon, çaprazlama ve seçilim işlemlerine tabi tutularak yeni çözümler üretilmektedir.

2.2.1. Mutasyon işlemi

Diferansiyel gelişim algoritmasında popülasyonu oluşturan aday çözümler kromozomları ve kromozomların içerisindeki her bir değişken ise genleri ifade etmektedir. Mutasyon işlemi bir kısım veya tüm genler üzerinde değişiklik yapıp yeni mutant bireyler elde edilmesi olarak tanımlanabilmektedir. Bu işlem sonucu, popülasyonda yeni üretilen mutant bireylerin eski ebeveyn bireylerinden daha iyi olması olasılığını doğurmaktadır. Mutasyon işlemine tabi tutulacak mevcut kromozom dışında üç adet daha kromozom seçilerek (r_1, r_2, r_3) yeni kromozom oluşturma işlemi yapılmaktadır. Mutasyon işlemi için yaygın olarak kullanılan operatörler Çizelge 2.1'de özetlenmektedir (Opara ve Arabas, 2018).

Çizelge 2.1. Mutasyon işleminde kullanılan operatörler

Mutasyon işlemi	Formülizasyonu
DE/rand/1	$V_i = X_{r_3} + F(X_{r_1} - X_{r_2})$
DE/best/1	$V_i = X_i + F * (X_{best} - X_i) + F(X_{r_1} - X_{r_2})$
DE/rand-to-best/1	$V_i = X_{best} + F(X_{r_1} - X_{r_2})$
DE/best/2	$V_i = X_{best} + F * (X_{r_1} - X_{r_2}) + F(X_{r_3} - X_{r_4})$
DE/rand/2	$V_i = X_{r_1} + F(X_{r_2} - X_{r_3}) + F(X_{r_4} - X_{r_5})$
DE/rand-to-best/2	$V_i = X_i + F(X_{best} - X_i) + F(X_{r_1} - X_{r_2}) + F(X_{r_3} - X_{r_4})$

Çizelge 2.1'de X_r *ve* X_{best} popülasyondan seçilen bireyi belirtmektedir. Burada X_{best} popülasyondaki en iyi kromozomu, X_r rastgele seçilen kromozomları, X_i mevcut kromozomu ifade eder. Denklemlerde ifade edilen F ise ölçekleme faktörünü belirtmektedir.

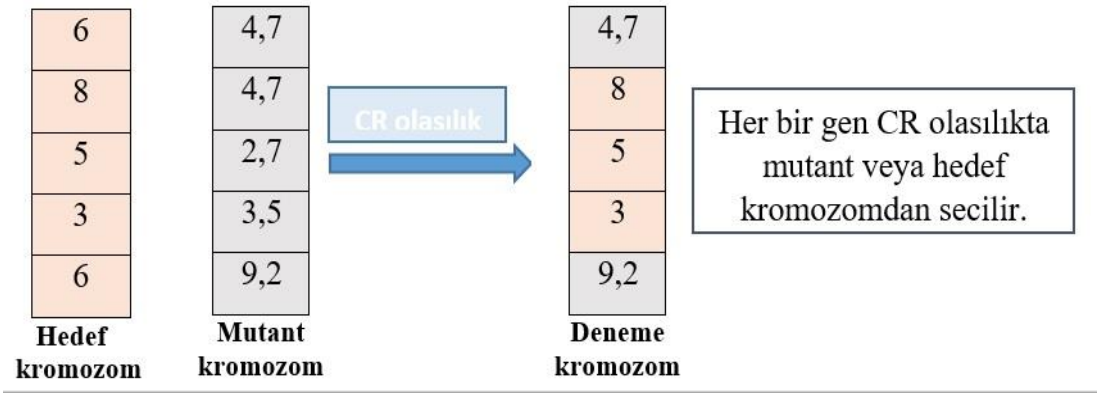
2.2.2. Çaprazlama işlemi

Diferansiyel gelişim algoritmasında yeni kromozomlar çaprazlama işlemi vasıtasıyla mevcut kromozomlardan üretilebilmektedir. Bu amaçla, mutant vektör/kromozom mutasyon sonucunda elde edilerek, çaprazlama işlemi, önceki nesilden elde edilen kromozom ve mutant vektör/kromozom arasında olmaktadır. Çaprazlama işlemi sonucunda yeni deneme kromozomu elde edilmektedir. Klasik diferansiyel gelişim algoritmasında sıklıkla kullanılan binom çaprazlama Denklem (2.7)'de verilmektedir (Storn ve Price, 1997).

$$U_{i,j}^{k+1} = \begin{cases} V_{i,j}^{k+1} \; eğer \; rand_j(0,1) \leq CR \; ya \; da \; j = j_{rand} \\ X_{i,j}^{k} \; diğer \; durumda \end{cases} \tag{2.7}$$

Burada, CR, çaprazlama olasılığını ifade etmektedir. U_i deneme kromozomuna ait her bir gen, 0 ile 1 arasında üretilen rastgele sayıya bağlı olarak, CR olasılıkla mutant kromozomdan (V_i), $1 - CR$ olasılıkla mevcut kromozomdan (hedef kromozom, X_i) seçilmektedir. Buradaki temel amaç, belirli bir oranda genin alınması vasıtasıyla yeni kromozomlar üretilmesidir. j_{rand} parametresi en az bir tane genin mutant kromozomdan alınmasını garanti etmek amacıyla kullanılmaktadır. Rastgele seçilen j_{rand} noktasındaki

gen CR olasılığına bakılmaksızın mutant kromozomdan seçilmektedir (Storn ve Price 1995). Şekil 2.2'de temsili bir çaprazlama operasyonu gösterilmektedir (Kuyu, 2016). Görüldüğü üzere, hedef kromozomdan 3 gen ve mutant kromozomdan iki gen paylaşılarak yeni bir deneme kromozomu oluşturulmaktadır.



Şekil 2.2. Çaprazlama işlemi örnek gösterimi

2.2.3. Seçilim işlemi

Seçilim, sonraki nesile katılacak bireyleri belirlemek amacıyla tasarlanmış bir işlemdir. Popülasyonun o anki bireyi olan hedef kromozom ile bu kromozomun bilinen uygunluk değeri, mutasyon ve çaprazlama işlemleri vasıtasıyla elde edilen deneme kromozomu ve bu deneme kromozomunun hesaplanan uygunluk değeri, yeni nesile katılacak bireyin seçiminde rol oynamaktadır. Hedef kromozom ve deneme kromozomu uygunluk değerleri arasındaki en kaliteli uygunluk değeri, bu minimizasyon problemi için daha düşük uygunluk değeri olurken, maksimizasyon problemi için daha büyük uygunluk değeri olmaktadır. Böylece yeni nesile katılacak olan kromozomun seçiminde kullanılmaktadır. Bu işlem Denklem (2.8)'de ifade edilmektedir (Storn ve Price, 1997).

$$\forall\, i \leq NP\colon\ X_i^{k+1} = \begin{cases} U_i^{k+1}\ eğer\ f(U_i^{k+1}) \leq f(X_i^k) \\ X_i^k\ diğer\ durumda \end{cases} \tag{2.8}$$

Burada; NP popülasyon sayısını, X_i^k hedef kromozomu, X_i^{k+1} yeni nesilin i. kromozomunu, U_i^{k+1} deneme kromozomunu, $f(U_i^{k+1})$ deneme kromozomunun uygunluk değerini ve $f(X_i^k)$ ise X_i^k kromozomun uyguluk değerini göstermektedir. Bu

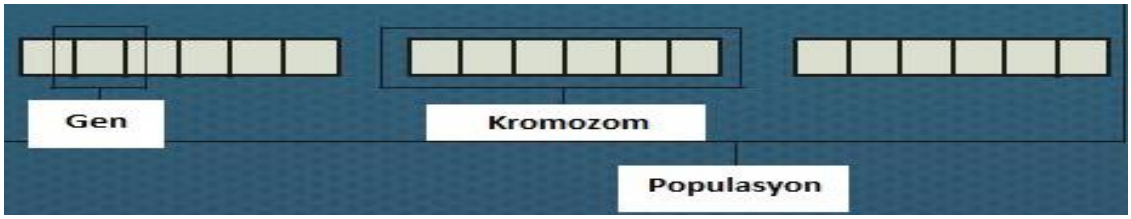
süreçler algoritmanın sonlandırma kriteri sağlanıyorsa durdurulmakta; sağlanmıyorsa mutasyon, çaprazlama, seçilim işlemlerine devam edilmektedir. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Diferansiyel Gelişim Algoritması
1. Başlangıç parametrelerini tanımla. 2. Başlangıç popülasyonunu üret. 3. Kromozomlardaki uygunluk değerini hesapla. 4. Kromozomları çaprazlama ve mutasyon kriterlerine göre güncelle. 5. Güncellenen kromozomların uygunluk değerini hesapla 6. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi çözümü seç. b. Sağlanmıyor: - 4. Adıma geç.

2.3. Genetik Algoritma

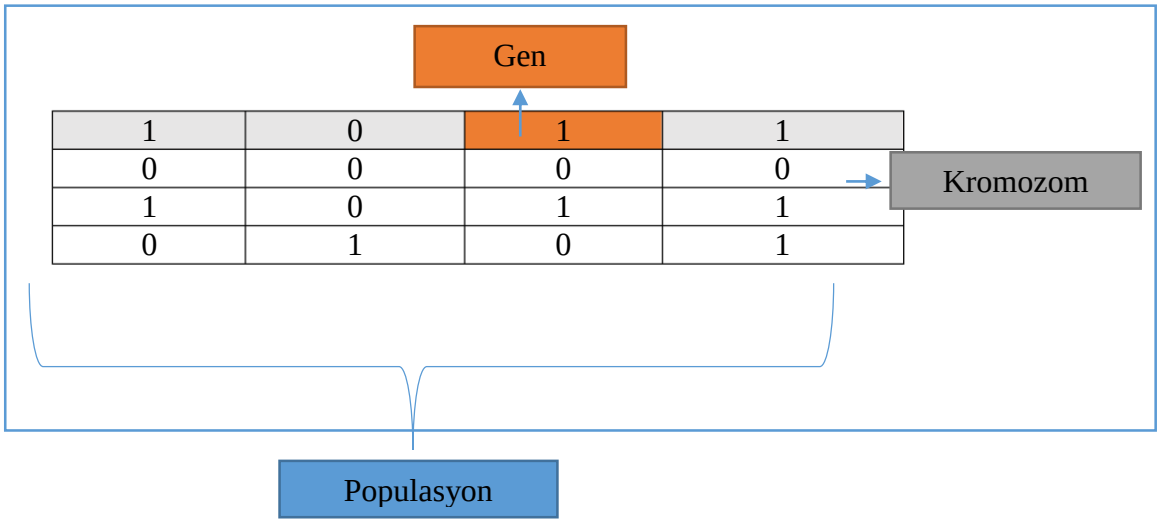
Genetik algoritma (Genetic Algorithm - GA); doğada gözlemlenen evrimsel süreçten esinlenen, bir optimizasyon problemine en uygun çözümü (optimal) arayan bir metasezgisel optimizasyon tekniğidir (Holland, 1975). Bu algoritma, birçok optimizasyon uygulamalarında, hem çok boyutlu nümerik fonksiyonlarda hem de karmaşık gerçek dünya problemlerinde kullanılmakta ve başarılı sonuçlar elde edilmektedir (Sivanandam ve Deepa, 2008).

Genetik algoritmalar, diğer popülasyon temelli algoritmalara benzer olarak çözümü arama sürecini başlatmak için bir başlangıç popülasyonuna ihtiyaç duymaktadırlar. Bu popülasyon ele alınan problemin boyutuna bağlı olarak rastgele aday çözümlerden oluşturulmaktadır. Genetik algoritmalar evrim teorisine dayanarak güçlü olanın kazanma şansının daha fazla olması prensibine dayanarak geliştirilmiştir (Goldberg 1989). Evrim teorisinden esinlenilen bu durumda; probleminin çözüm kümesini oluşturan bu aday çözümler, kromozom adı verilen, popülasyonun bir bireyini belirten vektörlerden oluşmaktadır. Kromozomların içindeki her bir element gen olarak adlandırılmaktadır. Kromozomların, genetik algoritma operatörleri vasıtasıyla daha iyiye geliştirilmesi amaçlanmaktadır. GA popülasyon yapısı Şekil 2.3'te verilmektedir (Kuyu, 2016).



Şekil 2.3. Genetik algoritmadaki popülasyon yapısı

Sıfır bir rakamları ile ifade edilen ikili sayı sistemi GA'da sıklıkla kullanılmaktadır. Oluşturulacak başlangıç popülasyonu, problemin sınırları gözetilerek, rastgele genler kodlanarak kromozomları oluşturmaktadır. Kromozomların ikili sayı sistemi ile kodlanmasıyla dört elemanlı bir popülasyonun elde edilmesine ait şema Şekil 2.4'te verilmektedir.



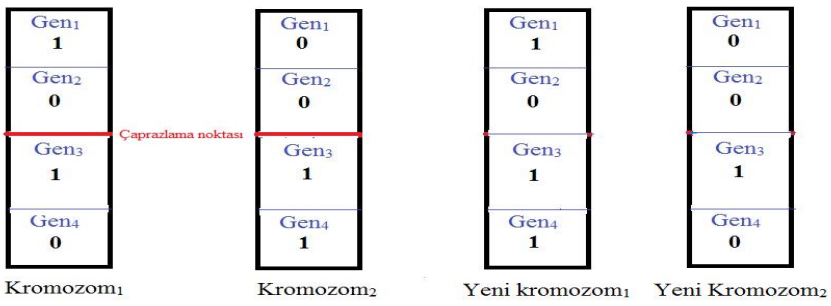
Şekil 2.4. Genetik algoritmadaki ikili kodlama gösterimi

GA; başlangıç popülasyonunu oluşturduktan sonra, popülasyonu oluşturan bütün kromozomlar için amaç fonksiyonunda (objective function) uygunluk değerleri (fitness value) hesaplanmaktadır. Bu değerler, bir sonraki nesilde popülasyon bireylerinin yaşama şanslarını belirlemektedir; çünkü uygunluk değerleri, popülasyon içerisindeki bireylerin ne oranda çözüm sağladıklarını aynı zamanda bireylerin kalitesini gösteren sayısal değerlerdir. En iyi birey daima saklanacağından, sonraki nesil için ilk hayatta kalan olacaktır. GA, başlangıç popülasyonunu oluşturup bu popülasyona ait uygunluk değerlerini hesapladıktan sonra, yeni çözümler geliştirmek için çaprazlama mutasyon ve seçilim işlemlerini uygulamaktadır.

2.3.1. Çaprazlama işlemi

Çaprazlama işlemi, ebeveynlerin rastgele seçilmiş bir konumdan sonraki genlerinin yeni bireyler oluşturmak için değiş tokuş edilmesi olarak tanımlanabilmektedir. Buradaki amaç, uygun bir çaprazlama noktası seçildiğinde, ebeveynlerdeki iyi genleri çocuk bireylerde birleştirmek ve daha iyi kromozomlu çocuk bireyler elde etmektir. Bu işlemde, genellikle iki ebeveyn kromozomdan alınan genler ve bu genlerin değişim noktası belirlenerek, genler arası bir değiş tokuş sağlamakta ve uygunluk değeri daha kötü olan ebeveyn kromozomlar yerine, uygunluk değeri daha iyi kromozomlardan çaprazlama yöntemiyle yeni çocuk kromozomlar oluşturulmaktadır.

Genetik algoritmada, problemin türüne bağlı olarak çaprazlama noktası genellikle rastgele olarak seçilmektedir. Tek noktalı çaprazlama literatürde sıklıkla kullanılmaktadır (Holland 1975). Bu çaprazlama türünde, kromozomun uzunluğu boyunca rastgele olarak bir tam sayı üretilmekte ve bu sayı, çaprazlama noktası olarak seçilmektedir. Çaprazlama noktasının belirlenmesinden sonra, ebeveyn kromozom çiftleri karşılıklı olarak gen değiş tokuşu yapmaktadırlar. Değiştirilen genler genetik olarak bireylerin iyi özelliklerini taşıyorsa, bu genlerden oluşan çocuk bireyler daha iyi uygunluk değerlerine sahip olacaktır. Tek noktalı çaprazlama yöntemi Şekil 2.5'te gösterilmektedir (Kuyu, 2016).

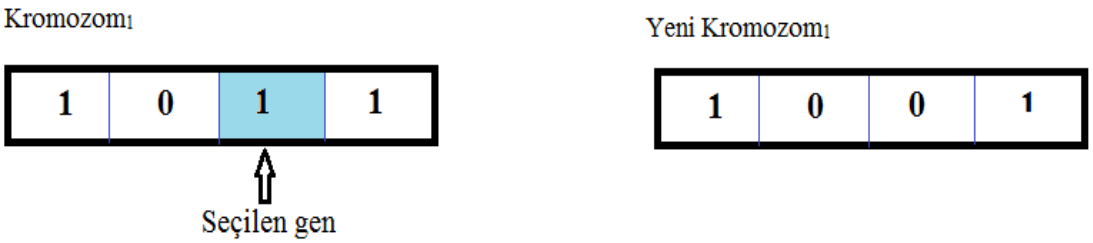


Şekil 2.5. Tek noktalı çaprazlama yöntemi

Şekil 2.5'te, ebeveyn kromozomların genleri, yeni çocuk kromozomlar oluşturmak amacıyla değiştirilmektedir.

2.3.2. Mutasyon işlemi

Çaprazlama işleminden sonra gerçekleşen mutasyon işlemi, evrimsel süreçte belirli bir genin doğal olmayan yollarla değişimini ifade etmektedir. Burada amaç, popülasyonu oluşturan kromozomlar arasında çeşitlilik yaratmak ve bu çeşitlilik vasıtasıyla kromozomların yerel minimuma takılmasını engelleyebilmektedir (Spears 1993). Bu işlemde, başlangıçta bir mutasyon oranı tanımlanmakta ve bu oran dahilinde kromozomlara mutasyon işleminin hangi sıklıkla yapılacağı belirlenmektedir. Eğer bu oran 1'e eşit olursa, tüm popülasyona mutasyon uygulanacağı anlamını taşır ve o jenerasyondaki tüm kromozomlar değişir. Bundan dolayı yüksek mutasyon oranının daha iyi çözümlere hızla yakınsama sağlaması olasılığı barındırdığı gibi optimal çözümlerden uzak noktalarda yeni aday çözümleri oluşturulmasını da mümkün kılmaktadır. Mutasyon işleminin sonucunda oluşan yeni kromozomun gen sayısı, mutasyon işlemi uygulanan kromozomun gen sayısıyla aynı kalmaktadır. Mutasyon sayesinde, çaprazlama işlemi ile olası kaybedilen iyi özelliklerin geri kazanılabileceği düşünülmektedir (Goldberg 1989). Bu işlem, ikili kodlamada sıfır olan genin bir veya bir olan genin sıfır olması ile ifade edilir. İkili sayı sistemi ile örnek mutasyon işlemi Şekil 2.6'da gösterilmektedir (Kuyu, 2016).



Şekil 2.6. Mutasyon işlemi

2.3.3. Seçilim işlemi

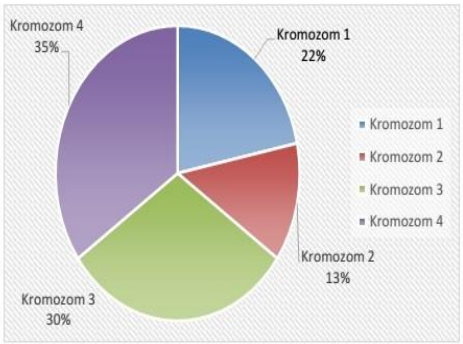
Genetik algoritmada, iterasyonlar boyunca birbirini izleyen yeni nesillerin oluşmasında kromozomların seçimi önemli rol oynamaktadır. Bu seçilim işleminde daha iyi aday çözümlere ulaşmak amacıyla yeni nesillere aktarılacak kromozomlar belirlenmekte ve bu gelecek nesillerin en uygun çözümü arama sürecine yön vermektedir. Genetik

algoritmada turnuva seçimi ve rulet çemberi metotları sıklıkla seçilim işleminde kullanılmaktadır.

Turnuva seçilimi yönteminde; popülasyon içinden önceden belirlenen sayıda, rastgele bir grup kromozom ebeveyn olarak atanmaktadır. Belirlenen olasılığa göre en iyi kromozom seçilmekte ve gruptaki bireylerin sayısı turnuva büyüklüğünü ifade etmektedir. Seçilen olasılık arttıkça, en iyi kromozom seçiliminin şansı artmakta; olasılık değerinin bir olması, en iyi kromozomun sürekli seçiliminin yapılacağı anlamına gelmektedir. Bu yöntem ile önceden seçilen bir kromozomun tekrar seçilmesi engellenerek ve popülasyondaki çeşitliliği arttırarak, olası kromozomların çözüm arayışında yerel minimum noktasına düşmesinin önüne geçilmeye çalışılmaktadır. Diğer bir deyişle, oluşturulan yeni bireyler vasıtasıyla nesildeki kötü bireyler elimine edilmeye çalışılmaktadır.

Bir diğer yöntem olan rulet çemberi seçilimi yönteminde, öncelikle popülasyondaki tüm kromozomların uygunluk değerleri hesaplanmaktadır. Bireylerin uygunluk değerleri toplanarak popülasyonun toplam uygunluk değeri elde edilmektedir. Her bireyin uygunluk değerleri, popülasyonun toplam uygunluk değerine bölünmekte ve bireyin seçilme olasılıkları elde edilmektedir. Bu oran yüzdesel olarak kromozomların rulet çemberinde kapsayacağı alanı ifade etmektedir. Oluşturulan rulet çemberinden sonra sıfır ile yüz arasında bir sayı seçilir ve bu sayıya oluşturulan çember üzerinde olan kromozom seçilir. Yapılan bu seçilim işlemi, rulet çemberinin döndürülmesi olarak adlandırılmaktadır. Bu işlemde, uygunluk değeri iyileştikçe kromozomların kaplayacağı alan çemberde daha fazla olacağından, seçilim olasılıkları daha yüksektir. Şekil 2.7'de rulet çemberinin oluşumu ve ilgili kromozomların rulet çemberine yerleşimi gösterilmektedir (Kuyu, 2016).

Kromozom	Uygunluk Değeri	Rulet Oranı
Kromozom 1	25	22%
Kromozom 2	15	13%
Kromozom 3	35	30%
Kromozom 4	40	35%
Toplam	115	100%



Şekil 2.7. Rulet çemberi yöntemi

Şekil 2.7'de dört kromozomdan oluşan bir popülasyon, her bir kromozomun uygunluk değeri ve rulet çemberinde kapladığı alan yüzdesel oran olarak verilmektedir. Şekilden görülebileceği üzere, uygunluk değeri en yüksek olan "Kromozom 4", rulet çemberinde en çok alanı kapsamaktadır ve seçilim olasılığı diğer kromozomlara göre daha yüksek olmaktadır. Diğer yandan rulet çemberinde daha az yer kaplayan kromozomların seçilim olasılığı bulunmaktadır.

Genetik algoritmada, çaprazlama, mutasyon, ve seçilim işlemleri sonrasında nesillerdeki en iyi uygunluk değerine sahip kromozom, bir sonraki kuşağa aktarılmayabilir. Diğer bir ifadeyle en iyi uygunluk değerine sahip kromozom, bu işlemler sonucunda bozulmaya ve uygunluk değeri orijinal değerinden daha kötü hale gelmeye başlayabilmektedir. Daha önce de bahsedildiği gibi, GA evrimden esinlenip en güçlünün hayatta kalması ilkesinden yola çıktığından, en iyi kromozomun kaybolmaması için genetik algoritma elitizm yöntemine başvurmaktadır. Bu yöntemde belirlenen bir elitizm oranınca, popülasyondaki kromozomların en iyi uygunluk değerleriyle korunması ve sonraki nesile aktarılması amaçlanmaktadır. Bu yöntem vasıtasıyla, yüksek kaliteli kromozomlar korunarak, sonraki nesillerde bozulmasının önüne geçilmesi öngörülmektedir. Buna ek olarak, popülasyonun elit kromozom veya kromozomlar dışında kalan kromozomları; seçilim, çaprazlama ve mutasyon operatörlerine tabi tutularak, evrimsel süreçte kromozomların kalitesi arttırılmaya devam edilmektedir. Geliştirilen yeni kromozomlar, uygunluk değeri vasıtasıyla seçilen elit kromozomla karşılaştırılarak, daha kaliteli olmaması durumunda seçilen elit kromozom bozulmadan sonraki nesile aktarılarak, en kaliteli kromozomun devamlılığı sağlanmaktadır. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Genetik Algoritma
1. Başlangıç parametrelerini tanımla. 2. Başlangıç popülasyonunu üret. 3. Kromozomlardaki uygunluk değerini hesapla. 4. Kromozomları çaprazlama ve mutasyon kriterlerine göre güncelle. 5. Güncellenen kromozomların uygunluk değerini hesapla. 6. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi kromozomu çözüm olarak seç. b. Sağlanmıyor: - 4. Adıma geç.

2.4. Guguk Kuşu Arama Algoritması

Bu algoritma, guguk kuşu türlerinden ilham alınarak geliştirilen popülasyon tabanlı bir optimizasyon yöntemidir (Yang ve Deb, 2009). Guguk kuşları, diğer kuşların yuvalarına yumurta bırakmakta ve kendi yavrularını beslemek için diğer kuşlara güvenmektedirler. Diğer yandan, guguk kuşunun üreme ve daha fazla yumurta bırakma konusunda daha fazla zaman harcaması gerekmektedir. Bununla birlikte, yuvaları istila edilen kuşlar, istilacı yumurtaları tespit etmek için karşı stratejiler ve giderek daha karmaşık yollar geliştirmektedirler. Guguk kuşu algoritması, arama alanını dolaşmak ve en uygun çözümleri bulmak için bahsedilen bu davranışları modellemektedir.

İçinde yumurta bulunan bir dizi yuva, yumurtaların her birinin bir aday çözümü temsil ettiği arama uzayında rastgele konumlara yerleştirilmektedir. Bir dizi guguk kuşu, karşılaşılan farklı aday çözümler için en yüksek uygunluk değerlerini kaydederek arama uzayını geçmekle görevlendirilmektedir. Guguk kuşları, gerçek hayatta kuşlar ve böcekler tarafından da kullanılan Lévy uçuşu adlı bir arama yöntemi kullanmaktadır. Guguk kuşu arama algoritması aşağıdaki ön tanımları kabul etmektedir:

- Her guguk kuşu, tek seferde bir yumurtayı rastgele seçilen yuvaya bırakır.
- Kaliteli yumurtaları olan en iyi yuvalar gelecek nesillere taşınır.
- Mevcut konak yuvalarının sayısı sabittir ve yumurta bir guguk kuşu tarafından atılan bir ev sahibi kuş tarafından $p_a \epsilon\ (0,1)$ olasılıkla keşfedilir. Bu durumda ev sahibi kuş, ya yumurtadan kurtulur ya da yuvayı terk ederek tamamen yeni bir yuva inşa eder.

Yukarıda bahsedilen tanımlamalar şöyle gerçekleşebilmektedir: n konak yuvalarının bir p_a kesri, yeni yuvalarla (yeni rastgele çözümlerle) yer değiştirmektedir ve problemde, bir çözümün kalitesi sadece uygunluk değeriyle orantılı olmaktadır. Uygulama açısından, bir yuvadaki her yumurtanın bir çözümü temsil ettiği ve her guguk kuşunun yalnızca bir yumurta bırakabileceği (böylece bir çözümü temsil ettiği) basit temsilleri kullanabilmektedir. Buradaki amaç, yuvalarda çok iyi olmayan bir çözümün yerine, yeni ve potansiyel olarak daha iyi çözümleri kullanmaktır. Guguk kuşları, kendi yumurtalarının yaşama olasılığını artırmak için yuva sahibi kuşun yumurtalarını atarlar

ve kendi yumurtalarını bırakırlar. Bu algoritma, her yuvanın bir dizi çözümü temsil eden birden fazla yumurtaya sahip olduğu daha karmaşık durumlarda da genişletilebilmektedir. Fakat burada, her yuvanın yalnızca tek bir yumurtaya sahip olduğu en basit yaklaşım kullanılmaktadır. Bu durumda, yumurta, yuva veya guguk kuşu arasında bir ayrım yoktur. Çünkü her yuva aynı zamanda bir guguk kuşunu temsil eden bir yumurtaya karşılık gelmektedir.

Bu algoritma rastsal ve global keşif yürüyüşlerinin p_a parametresiyle dengelenmiş bir şeklini kullanmaktadır. Rastgele yürüyüş Denklem (2.9)'da ifade edilmektedir (Yang ve Deb, 2014).

$$X_i^{t+1} = X_i^t + \alpha s H\,(p_a - \epsilon)(X_i^t - \mathrm{X_k^t}) \tag{2.9}$$

Burada $X_i^t\ ve\ X_k^t$ permutasyon vasıtasıyla rastsal olarak seçilen iki farklı çözümdür, $H(u)$ "Heaviside" fonksiyonudur, ϵ düzgün dağılımdan elde edilen bir katsayıdır, s ise adım büyüklüğünü göstermektedir. Global rastsal yürüyüş ise Lévy uçuşu vasıtasıyla Denklem (2.10)'da ifade edilmektedir.

$$X_i^{t+1} = X_i^t + \alpha L(s, \lambda) \tag{2.10}$$

Burada $L(s, \lambda)$, guguk kuşunun Lévy uçuşunu ve $\alpha\ (> 0)$ adım büyüklüğü ölçeklendirme katsayısını/faktörünü ifade etmektedir. Algoritmanın yeni çözümler üretmesi sürecinde, guguk kuşları $p_a \epsilon\ (0,1)$ olasılığında en kötü çözümleri terk ederek yerine yenilerini inşa etmektedirler. Bu algoritma iki arama özelliğine sahiptir: yerel (local) arama ve küresel (global) arama. Bu denge bir anahtarlama keşif olasılığı ile kontrol edilmektedir. Yerel arama, arama süresinin yaklaşık 1/4' ünde (p_a = 0.25 için) çok yoğunken, genel arama ise toplam arama süresinin yaklaşık 3/4' ünü kapsamaktadır. Bu, arama alanının küresel ölçekte daha verimli şekilde keşfedilebilmesini ve dolayısıyla küresel optimalliğin daha yüksek olasılıkla bulunabilmesini sağlamaktadır. Guguk kuşu arama algoritmasının bir başka avantajı da genel aramanın standart rastsal yürüyüş yerine Lévy uçuşunu kullanmasıdır. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Guguk Kuşu Arama Algoritması
1. Başlangıç parametrelerini tanımla. 2. Başlangıç guguk kuşu popülasyonunu oluştur. 3. Popülasyonun uygunluk değerini hesapla. 4. Guguk kuşlarını Levy uçuşu prensibine göre güncelle. 5. p_a oranında kötü çözümleri elimine et ve yeni çözümler oluştur. 6. Guguk kuşlarının uygunluk değerini hesapla. 7. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi guguk kuşunu çözüm olarak seç. b. Sağlanmıyor: - 4. Adıma geç.

2.5. Harmoni Arama Algoritması

Harmoni arama algoritması (Harmoni Search - HS); problemin ve çözüm parametrelerinin belirlenmesi, harmoni belleğinin oluşturulması, yeni harmoni üretilmesi, harmoni belleğinin güncellenmesi ve durdurma kriteri kontrolü temel adımlarını kullanarak yeni çözümler elde etmektedir (Geem, Kim ve Loganathan, 2001; Lee ve Geem, 2004). Bu algoritma birden çok yönlü arama yapabilme özelliğine sahip olup yerel minimumdan kurtulmayı amaçlarken, genetik algoritmanın da yapısında olan 0-1 değişkenlerini de kullanma özelliğine sahiptir. Bunun yanında karar değişkenleri için özel bir çözüme ihtiyaç duymamaktadır, bu da algoritmaya esneklik sağlamaktadır (Lee ve Geem 2004; Yang, 2010). HS algoritmasında da diğer algoritmalarda olduğu gibi probleme ait matematiksel modelleme yapılmaktadır (bkz. Denklem (2.11)).

$$F = \min\{f(x)\} \quad x_i \in X_i = 1,2,\dots M \tag{2.11}$$

Burada $f(x)$ uygunluk değerini verecek hedef (objective) fonksiyonu, x_i problemin değişkenini, X_i her bir değişken için kullanılacak çözüm uzayını ve M ise toplam değişken sayısını ifade etmektedir. Harmoni arama algoritması popülasyon büyüklüğü ve durdurma kriteri dışında, başlangıçta ön tanımlı olarak belirtilmesi gereken kendi özel parametrelerine sahiptir. Bu parametreler; harmoni belleği kapasitesi (HMS), harmoni belleği dikkate alma oranı ($HMCR$) ve ton ayarlama oranıdır (PAR). Çözüm uzayı içerisinde, D boyutlu bir problemin çözümü için Denklem (2.12)'de verilen matrise benzer olarak, rastsal bir şekilde, olası çözümleri içeren bir harmoni belleği oluşturulmaktadır.

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ \vdots & \ddots & \vdots \\ x_{HMS,1} & \cdots & x_{HMS,D} \end{bmatrix} \quad (2.12)$$

HS algoritmasında; bir karar değişkeni, mevcut harmoni belleğinden 0 ile 1 arasında değişen $HMCR$ olasılıkla seçilebilirken, $(1 - HMCR)$ olasılıkla mevcut çözüm uzayı içerisinden rastsal olarak da seçilebilmektedir. Seçilim işlemi Denklem (2.13)'te ifade edilmektedir.

$$x_i^{'} = \begin{cases} x_i^{'} \in \{x_{1,i}, x_{2,i}, \ldots . x_{HMS,i}\} & HMCR\ olasılığında \\ x_i^{'} \in X_i & Diğer\ durum \end{cases} \quad (2.13)$$

Denklem (2.13)'te, yeni harmoni vektörü $x' = (x_1', x_2', \ldots x_D')$ harmoni belleğinde bulunan tonlardan rastsal olarak seçilmektedir. Değişkenlerin harmoni belleğinden seçilimi $0 - 1$ arasında belirlenen $HMCR$ olasılığına göre olmaktadır. Bu işlemden sonra PAR parametresine bağlı olarak ton ayarlama işleminin gerekli olup olmadığı Denklem (2.14)'e göre belirlenebilmektedir.

$$x_i^{'} = \begin{cases} x_i^{'} \pm Rand(0,1) * bw & PAR\ olasılığında \\ x_i^{'} & Diğer\ durum \end{cases} \quad (2.14)$$

Denklem (2.14)'te, bw parametresi bant genişliğini, $Rand(0,1)$ 0 ile 1 arasında üretilmiş rastgele sayıyı ifade etmektedir. Eşitlikten görüleceği üzere $1 - PAR$ olasılığında x_i' karar değişkeni aynı kalmaktadır. Denklem (2.14) vasıtasıyla oluşturulan yeni harmoni vektörü ile uygunluk değerine göre, harmoni hafızasında bulunan en kötü uygunluk değerine sahip harmoni vektörü arasında bir karşılaştırma ve bunun sonucunda bir seçilim işlemi uygulanmaktadır. Uygunluk fonksiyonu değerlerine göre yeni oluşturulan harmoni vektörü, en kötü harmoniden iyiyse, en kötü harmoni vektörü hafızadan çıkarılır ve yeni harmoni vektörü onun yerine yerleştirilmektedir. Bu algoritma ön tanımlı olarak belirlenen bir durdurma kriterine ulaşınca kadar devam etmekte, en iyi uygunluk değerine sahip harmoniyi çözüm olarak sunmaktadır. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Harmoni Arama Algoritması
1. Başlangıç parametrelerini tanımla ve başlangıç hafızayı rastsal olarak oluştur. 2. Başlangıç harmonilerini üret. 3. Başlangıç harmonilerin uygunluk değerini hesapla. 4. Harmonileri güncelle. 6. Harmonilerin güncel uygunluk değerini hesapla. 7. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi harmoniye çözüm olarak seç. b. Sağlanmıyor: - 4. Adıma geç.

2.6. Harris Şahini Optimizasyon Algoritması

Adından da anlaşılabileceği üzere, bu algoritma Harris şahinlerinden esinlenerek geliştirilen popülasyon tabanlı bir algoritmadır (Heidari, Mirjalili, Faris, Aljarah, Mafarja ve Chen, 2019). Harris şahinlerinin avı yakalamak için ana taktiği "sürpriz saldırı" olmaktadır. Bu " yedi öldürme" stratejisi olarak da bilinmektedir. Bu akıllı stratejide, birkaç şahin farklı yönlerden işbirliği içinde saldırmaya çalışmakta ve aynı anda tespit edilip kaçan bir tavşana yakınlaşmaktadırlar. Saldırı, avın birkaç saniye içinde yakalanmasıyla hızlı bir şekilde tamamlanabilmektedir. Ancak bazen avın kaçma yetenekleri ve davranışları ile ilgili olarak birkaç dakika boyunca avın yakınında birden fazla, kısa uzunlukta, hızlı dalışlar (yedi öldürme) içerebilmektedir. Harris şahinleri, koşulların dinamik doğasına ve avın kaçış düzenlerine bağlı olarak çeşitli kovalamaca stilleri gösterebilmektedir. Bir geçiş taktiği, en iyi şahin (lider) avına çarptığında ve kaybolduğunda ortaya çıkmakta ve kovalamaca parti üyelerinden biri tarafından devam ettirilmektedir. Bu geçiş aktiviteleri farklı durumlarda gözlemlenebilmektedir; çünkü kaçan tavşanın kafasını karıştırmak için farklı durumlar da faydalıdır. Bu işbirlikçi taktiklerin ana avantajı, Harris şahinlerinin, tespit edilen tavşanı çok yoruluncaya kadar takip edebilmesidir. Bu da, avın savunmasızlığını artırmaktadır. Üstelik kaçan avı şaşırtarak, avın savunma yetenekleri azaltır ve nihayet karşı karşıya kaldığı ekip kuşatmasından kurtulamaz; çünkü çoğu zaman en güçlü ve deneyimli şahinlerden biri yorulmadan tavşanı yakalamakta ve diğer parti üyeleri ile tavşanı paylaşmaktadır.

Harris Şahini Optimizasyon algoritmasının aşamaları, ayrıntılı olarak aşağıdaki alt başlıklarda özetlenmektedir.

2.6.1. Keşif aşaması

Bu bölümde, algoritmanın keşif mekanizması modellenmektedir. Eğer Harris şahinlerinin doğası göz önünde bulundurulursa; bu şahinler avlarını güçlü gözleriyle izleyebilmekte ve tespit edebilmektedir. Ancak bazen av kolayca görülememektedir. Harris şahini optimizasyon algoritmasında, Harris şahinleri aday çözümleri temsil etmekte ve her adımdaki en iyi aday çözüm, amaçlanan av veya neredeyse optimum olarak kabul edilmektedir. Algoritma modelinde, Harris şahinleri bazı yerlere rastgele tünemekte ve iki stratejiye dayalı bir av tespit etmeyi beklemektedir. Her tüneme stratejisi için eşit bir q şansı göz önüne alınırsa diğer aile üyelerinin konumlarına (saldırırken onlara yeterince yakın olmak için) ve buna bağlı olarak Denklem (2.15)'te modellenen tavşana göre tünemektedirler.

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3\big(LB + r_4(UB - LB)\big) & q < 0.5 \end{cases} \quad (2.15)$$

Denklem (2.15)'te, $X(t+1)$ şahinlerin sonraki iterasyon t deki pozisyonlarını belirten vektör, $X_{rabbit}(t)$ tavşanın pozisyonu, $X(t)$ şahinlerin o anki pozisyonu, $r1, r2, r3, r4\ ve\ q$, $0-1$ arasında belirlenen rastsal sayılar olmaktadır ve bu sayılar her iterasyonda değişmektedir. Denklem (2.15)'te, $LB\ ve\ UB$ ise problemin alt sınır ve üst sınır değerlerini ifade etmekte, X_m ise o anki şahin popülasyonunun değişkenlerinin ortalama değeri olmaktadır. Burada, rastgele konumlar oluşturmak için, (LB, UB) aralığında bir model önerilmektedir. İlk kural, rastgele bir yere ve diğer şahinlere dayalı çözümler üretilmesidir. İkinci kural ise, şimdiye kadarki en iyi konumun ve grubun ortalama konumunun ve değişken aralığına göre rastgele ölçeklendirilmiş bir bileşenin farkına sahipken, $r3$, kuralın rastgele doğasını daha da artırmak için bir ölçeklendirme katsayısıdır. $r4$, 1'e yakın değerler almakta ve benzer dağılım modelleri oluşabilmektedir.

2.6.2. Keşiften saldırıya geçiş safhası

Harris şahini optimizasyon algoritması keşiften sömürüye geçebilmekte ve daha sonra kaçan avın enerjisine dayanan farklı sömürücü davranışlar sergileyebilmektedir. Kaçma

sırasında bir avın enerjisi önemli ölçüde azalmaktadır. Bundan yola çıkarak, bir avın enerjisi Denklem (2.16)'da modellenmektedir.

$$E = 2E_0(1 - \frac{t}{T}) \tag{2.16}$$

Denklem (2.16)'da, E avın kaybolan enerjisini, T toplam iterasyon sayısını ve E_0 avın başlangıç enerji durumunu vermektedir. E_0 rastsal olarak -1 ile 1 aralığında her iterasyonda değişmektedir. E_0'ın değeri 0 dan -1' e düştüğünde tavşan fiziksel olarak zayıf düşerken, E_0'ın değeri 0'dan 1'e çıktığında, tavşan güçlenmektedir. Dinamik kaçış enerjisi E, yinelemeler sırasında azalan bir eğilime sahip olmaktadır.

Avlar, her zaman tehdit edici durumlardan kaçmaya çalışmaktadırlar. Eğer r bir avın başarılı bir şekilde kaçma olasılığı olarak varsayılırsa, sürpriz saldırıdan önce $r < 0.5$ durumunda av başarılı bir şekilde kaçarken; $r \geq 0.5$ durumunda ise, av başarılı bir şekilde kaçamamaktadır. Avın hareketine/davranışına göre şahinler, avı yakalamak için sert veya yumuşak kuşatma yapacaklardır. Bu durum - avın tutulan enerjisine bağlı olarak - avı, farklı yönlerden yumuşak veya sert bir şekilde kuşatacakları anlamına gelmektedir.

2.6.3. Yumuşak kuşatma

$r \geq 0.5$ ve $|E| \geq 0.5$ durumunda, tavşan hala yeterli enerjiye sahip olmakta ve bazı rastgele yanıltıcı sıçramalarla kaçmaya çalışmakta, fakat sonunda bunu başaramamaktadır. Bu girişimler sırasında, Harris şahinleri avı kuşatmakta, daha fazla yormakta ve sonunda sürpriz saldırı gerçekleştirmektedir. Bu davranış Denklem (2.17) ve Denklem (2.18)'de modellenmektedir.

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \tag{2.17}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{2.18}$$

Denklem (2.17) ve (2.18)'de, $\Delta X(t)$ tavşanın t. iterasyondaki pozisyon vektörüdür ve o anki pozisyonun farkını ifade etmektedir, $r5$, $0-1$ arasında bir rastsal sayıdır ve $J = 2(1 - r5)$ olarak ifade edilmektedir. J değeri tavşan hareketlerini modellemek amacıyla, her iterasyonda rastsal olarak değişmektedir.

2.6.4. Sert kuşatma

$r \geq 0.5$ ve $|E| < 0.5$ durumunda, av çok yorulmuştur ve düşük bir kaçış enerjisine sahiptir. Buna ek olarak Harris şahinleri, sonunda sürpriz saldırıyı gerçekleştirmek için amaçlanan avı kuşatmaktadırlar. Bu durum Denklem (2.19)'daki gibi modellenebilmektedir.

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \tag{2.19}$$

2.6.5. Aşamalı hızlı dalışlarla yumuşak kuşatma

Hala $|E| \geq 0.5$ ve $r < 0.5$ durumunda, tavşan başarılı bir şekilde kaçmak için yeterli enerjiye sahip olmakta ve sürpriz saldırıdan önce, hala yumuşak bir kuşatma inşa edilmektedir. Lévy uçuşu kavramı, Harris şahini optimizasyon algoritmasında da kullanılmaktadır. Bu uçuş, kaçan avların (özellikle tavşanların) gerçek zikzak-aldatıcı hareketlerini ve kaçan avın etrafındaki şahinlerin düzensiz, ani ve hızlı dalışlarını taklit etmek için kullanılmaktadır. Aslında, şahinler tavşanın etrafında birkaç takım hızlı dalış yapmakta ve avın aldatıcı hareketleriyle ilgili konumlarını ve yönlerini aşamalı olarak düzeltmeye çalışmaktadırlar. Bu mekanizma, doğadaki diğer rekabetçi durumlardaki gerçek gözlemlerle de desteklenmektedir. Şahinlerin gerçek davranışlarından esinlenerek, rekabetçi durumlarda avı yakalamak istediklerinde, ava doğru mümkün olan en iyi dalışı aşamalı olarak seçebilecekleri varsayılmaktadır.

Bu nedenle, yumuşak bir kuşatma gerçekleştirmek için, şahinlerin bir sonraki hamleleri Denklem (2.20)'ye dayanarak gösterilmektedir.

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \tag{2.20}$$

Daha sonra, iyi bir dalış olup olmayacağını tespit etmek için böyle bir hareketin olası sonucu önceki dalışla karşılaştırılmaktadır. Makul değilse (avın daha aldatıcı hareketler yaptığını gördüklerinde), tavşana yaklaşırken düzensiz, ani ve hızlı dalışlar yapmaya başlarlar. Denklem (2.21)'i kullanarak Lévy uçuşu tabanlı dalış modellenebilmektedir.

$$Z = Y + S \times LF(D) \tag{2.21}$$

Denklem (2.21)'de, D problemin boyutu, s bir rastsal vekör ve LF, Lévy uçuşu fonksiyonudur. Yumuşak kuşatma aşamasında şahinlerin konumlarını güncellemek için Denklem (2.22) kullanılmaktadır.

$$X(t+1) = \begin{cases} Y \ \ eğer \ \ F(Y) < F(X(t)) \\ Z \ eğer \ F(Z) < F(Xt)) \end{cases} \tag{2.22}$$

Her adımda, sadece daha iyi konum Y veya daha iyi konum Z bir sonraki konum olarak seçilmektedir.

2.6.6. Aşamalı hızlı dalışlarla sert kuşatma

Bu adımda, $|E| < 0.5$ ve $r < 0.5$ olduğunda, tavşanın kaçmak için yeterli enerjisi olmamaktadır ve sürpriz saldırıdan önce avını yakalamak ve öldürmek için sert bir kuşatma inşa edilmektedir. Bu adımın av tarafı yumuşak kuşatmadakine benzemektedir; ancak şahinler, kaçan av ile ortalama konumlarının mesafesini azaltmaya çalışmaktadırlar. Bu durum Denklem (2.23)'te modellenmektedir.

$$X(t+1) = \begin{cases} Y \ \ eğer \ \ F(Y) < F(X(t)) \\ Z \ eğer \ F(Z) < F(Xt)) \end{cases} \tag{2.23}$$

Bu adımda, $Y \ ve \ Z$ aşağıdaki yeni kurallara dayanarak elde edilmektedir.

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \tag{2.24}$$

$$Z = Y + S \times LF(D) \tag{2.25}$$

Denklem (2.24) ve Denklem (2.25)'te, $X_m(t)$ şahinlerin pozisyonlarının ortalamasını belirtmektedir. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Harris Şahini Optimizasyon Algoritması
1. Başlangıç parametrelerini tanımla 2. Başlangıç şahinlerini üret. 3. Şahinlerin uygunluk değerini hesapla. 4. Algoritma parametrelerini ve şahinlerin konumlarını güncelle. 6. Güncel uygunluk değerini hesapla. 7. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi şahini çözüm olarak seç. b. Sağlanmıyor: - 4. Adıma geç.

2.7. Karadul Optimizasyon Algoritması

Karadul optimizasyon algoritması (Black Widow Optimization Algorithm – BWO), örümceklerden ilham alınarak geliştirilmiş popülasyon tabanlı bir algoritmadır (Hayyolalam ve Kazem, 2020). Diğer metasezgisel algoritmalar gibi, önerilen algoritma da başlangıçtaki bir örümcek popülasyonuyla başlamakta, böylece her örümcek potansiyel bir çözümü temsil etmektedir. Bu ilk örümceklerden- çiftler halinde - yeni nesli yeniden üretmeye çalışmaktadırlar. Dişi karadul, çiftleşme sırasında veya sonrasında erkek örümceği yemektedir. Daha sonra depolanmış spermleri taşımakta ve yumurta keselerine bırakmaktadırlar. Yumurtlamadan 11 gün sonra, örümcek yavruları yumurta keselerinden çıkmaktadır. Anne ağında birkaç gün ila bir hafta birlikte yaşarlar, bu süre zarfında kardeş yamyamlığı gözlenebilmektedir. Algoritma, temel olarak bu gözlemlerden yola çıkmaktadır.

2.7.1. Başlangıç popülasyonu

Bir optimizasyon problemini çözmek için problem değişkenlerinin değerlerinin mevcut sorunun çözümü için uygun bir yapıda oluşması gerekmektedir. Karadul optimizasyon algoritmasında karadul örümceklerinin her biri potansiyel bir çözümü ifade etmektedir.

d boyutlu bir problemde bir karadul $[x_1, x_2..x_d]$ şeklinde ifade edilmektedir. Bu karadulun uygunluk fonksiyonu değeri hesaplaması ise Denklem (2.26)'daki gibidir.

$$Uygunluk\ değeri = f(karadul) = f(x_1, x_2..x_d) \tag{2.26}$$

Optimizasyon algoritmasını başlatmak için $Npop \times d$ boyutunda bir aday karadul matrisi (ilk örümcek popülasyonu) üretilmektedir. Daha sonra, çiftleşme yoluyla üreme adımını gerçekleştirmek için rastgele ebeveyn çiftleri seçilmektedir. Burada erkek kara dul, dişi tarafından sonrasında yenmektedir.

2.7.2. Üreme

Çiftler birbirlerinden bağımsız olduklarından, yeni nesli üretmek için çiftleşmeye başlarlar. Bu süreçte - doğada olduğu gibi - her bir çift kendi ağında, diğerlerinden ayrı olarak çiftleşmektedirler. Gerçek dünyada; her çiftleşmede yaklaşık 1000 yumurta üretilmekte, ancak daha güçlü olan örümcek bebeklerin bir kısmı hayatta kalmaktadırlar. Bu algoritmada, çoğaltmak (üreme için) için, *alfa* (α) adı verilen bir dizi, rastgele sayılar içeren karadulları temsil eden dizi oluşturulmaktadır. Daha sonra, $x1\ ve\ x2$'nin ebeveyn olduğu Denklem (2.27) kullanılarak, $y1\ ve\ y2$ yavruları α değeri vasıtasıyla üretilmektedir.

$$\begin{cases} y1 = \alpha x1 + (1-\alpha)x2 \\ y2 = \alpha x2 + (1-\alpha)x1 \end{cases} \tag{2.27}$$

Yukarıdaki işlem $d/2$ kez tekrarlanırken, aynı rastgele seçilen sayılar kullanılmamalıdır. Son olarak, çocuklar ve anne bir diziye eklenir ve uygunluk değerlerine göre sıralanır. Yamyamlık derecesine göre, yeni oluşturulan nüfusa en iyi bireylerden bazıları eklenmektedir. Bu adımlar tüm çiftler için geçerli olmaktadır.

2.7.3. Yamyamlık

Burada üç çeşit yamyamlık vardır. İlki cinsel yamyamlık olup burada dişi karadul çiftleşme sırasında veya sonrasında erkeği yemektedir. Diğer bir yamyamlık türü, güçlü örümcek yavrularının zayıf kardeşlerini yediği kardeş yamyamlığıdır. Bu algoritmada, hayatta kalanların sayısının belirlendiği bir yamyamlık derecesi (CR) belirtilmektedir. Bazı durumlarda, bebek örümceklerin annelerini yediği üçüncü tür yamyamlık da sıklıkla görülmektedir. Güçlü veya zayıf örümcekleri belirlemek için uygunluk değerlerinden yararlanılmaktadır.

2.7.4. Mutasyon

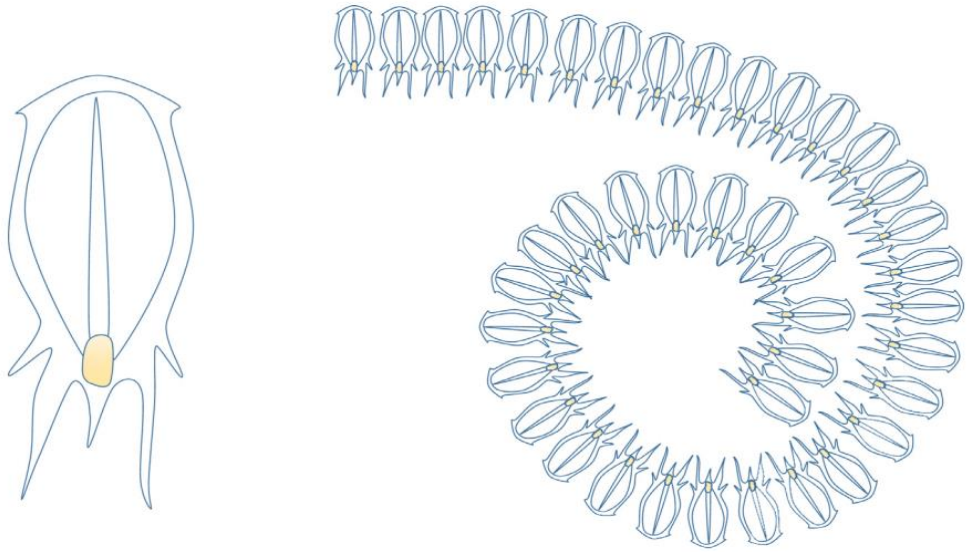
Bu algoritmada mutasyon, belirlenen sayıda örümceğin rastsal olarak popülasyondan seçilmesiyle başlamaktadır. Her bir seçilen örümceğin iki elementi rastsal olarak yer değiştirmektedir. Bu algoritmada mutasyon oranı vasıtasıyla hesaplanan bir mutasyon katsayısı bulunmaktadır. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Karadul Optimizasyon Algoritması
1. Başlangıç parametrelerini tanımla 2. Başlangıç örümcek popülasyonunu üret. 3. Popülasyonun uygunluk değerini hesapla. 4. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi örümceği çözüm olarak seç. b. Sağlanmıyor: - Popülasyonu güncelle. - 2. Adıma geç.

2.8. Salp Sürüsü Algoritması

2017 yılında önerilmiş olan salp sürüsü optimizasyon algoritması (Salp Swarm Algorithm – SSA), deniz canlısı salplardan ilham alan popülasyon tabanlı bir algoritmadır (Mirjalili, Gandomi, Mirjalili, Saremi, Faris, ve Mirjalili, 2017). Salplar şeffafımsı bir görünüme sahip olup, suyu beslenme filtrelerinden geçirerek ilerleyen ve planktonları yiyen okyanus canlılarıdır.

Algoritma, salpların okyanustaki düzenli hareket etme ve besin arama süreçlerini modellemektedir. Büyük gruplar halinde yaşayan salplar, vücutlarını bir zincire benzer şekilde birbirine bağlamakta ve beslenme sürecini bu şekilde sürdürmektedirler. Şekil (2.8)'de salpların bireysel ve zincirleme görüntüsü verilmektedir.



Şekil 2.8. Bireysel (tek) salp canlısı ve salp sürüsü (zinciri) gösterimi (Mirjalili ve diğerleri, 2017)

Salp zincirinde, lider salp, zincirin en başında yer almakta ve diğer salplar onu takip etmektedirler. SSA algoritmasında, bu mantıktan yola çıkılarak, popülasyon, lider ve takipçileri olarak iki kısma ayrılmaktadır. Takipçiler lideri takip eden salplar olarak adlandırılabilmektedirler.

Popülasyon tabanlı metasezgisellere benzer olarak, problem d boyutlu bir uzay içeriyorsa, bu algoritmada da her bir salp d tane karar değişkeni bulundurmaktadır. Liderin pozisyonu Denklem (2.28)'de güncellenmektedir.

$$x_j^1 = \begin{cases} F_j + c1\left(c2\left(ub_j - lb_j\right) + lb_j\right) & c3 \geq 0 \\ F_j - c1\left(c2\left(ub_j - lb_j\right) + lb_j\right) & c3 < 0 \end{cases} \tag{2.28}$$

Burada x_j^1 lider salpın pozisyonunu, F_j besin kaynağının pozisyonunu, $c2\ ve\ c3$ parametreleri ise $0-1$ aralığında oluşturulan rastsal sayılar olmaktadır. $ub\ ve\ lb$ değerleri ise problemin üst ve alt sınırlarını göstermektedir. $c1$ parametresi yerel arama ile global arama arasındaki geçişleri kontrol eden parametre olmaktadır ve Denklem (2.29)'daki gibi hesaplanmaktadır.

$$c1 = 2e^{-\left(\frac{4k}{maksimum\ iterasyon\ sayısı}\right)^2} \tag{2.29}$$

k burada o anki iterasyon sayısını belirtmektedir. Takipçilerin konum güncellemesi Denklem (2.30)'daki gibi yapılmaktadır.

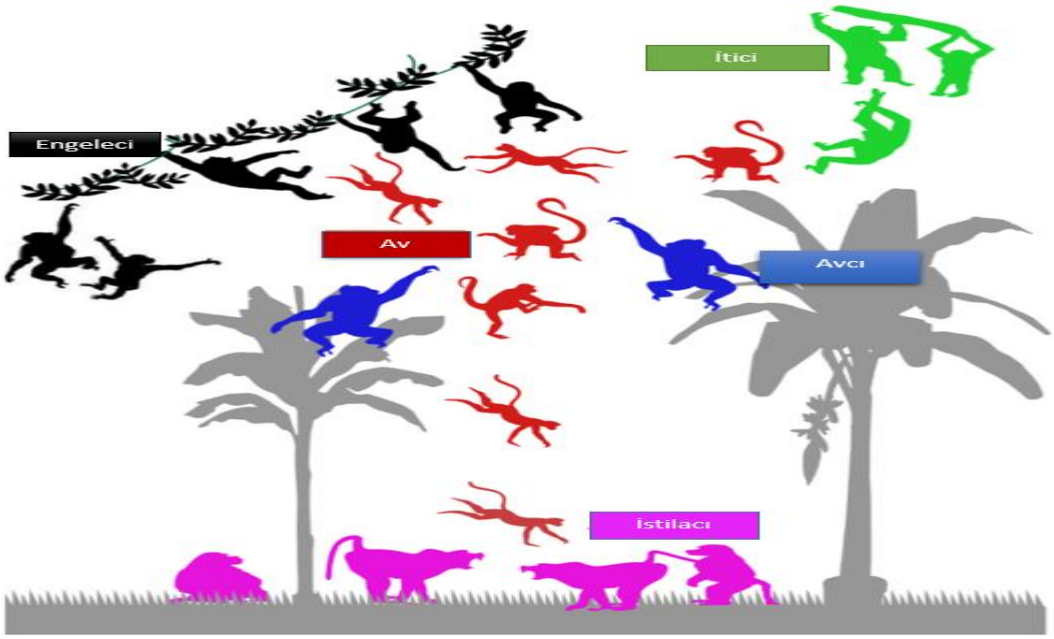
$$x_j{}^i = \frac{1}{2}(x_j{}^i + x_j{}^{i-1}) \tag{2.30}$$

Burada x popülasyondaki salpların konumunu, j boyut indeksini, i takipçinin indeksini belirtmektedir. Algoritma bulunan en iyi çözümü kaydetmekte ve bunu yiyecek kaynağı olarak tanımlamaktadır. Takipçiler kademeli olarak pozisyonlarını lider salpa yakınsamak amacıyla güncellemektedirler. Salpların konumları güncellendikten sonra, tekrar uygunluk değerleri hesaplanmakta ve yukarıda belirtilen işlemler maksimum iterasyon sayısına ulaşıncaya kadar devam etmektedirler. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Salp Sürüsü Algoritması
1. Başlangıç parametrelerini tanımla 2. Başlangıç salp popülasyonunu üret. 3. Başlangıç popülasyonun uygunluk değerini hesapla. 4. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi bireyi çözüm olarak seç. b. Sağlanmıyor: - Parametreleri ve popülasyonu güncelle. - Güncel popülasyonun uygunluk değerini hesapla. - 4. Adıma geç.

2.9. Şempanze Optimizasyon Algoritması

Bu algoritma, bir şempanze kolonisi yapısı ilham alınarak geliştirilmiştir (Khishe ve Mosavi, 2020). Bir şempanze kolonisinde, dört tür şempanze bulunmaktadır. Bunlar; itici, engelleyici, avcı ve istilacı şempanzelerdir. Hepsinin farklı yetenekleri vardır ve bu farklılıklar başarılı bir av için gereklidir. İticiler, avını yakalamaya çalışmadan avını takip etmektedirler. Engelleyiciler, avın ilerlemesi boyunca bir baraj/blokaj inşa etmek için kendilerini bir ağaca yerleştirmektedirler. Avcılar, av yakalandıktan sonra hızla hareket etmektedirler. Son olarak, istilacılar, avın olası yolunu avcılara doğru ya da alt koloniye doğru sürüklemektedirler. Avlanma sürecinde şempanzelerin adımları Şekil (2.9)'da gösterilmektedir.



Şekil 2.9. Şempanze kolonisinde avlanma süreci (Khishe ve Mosavi, 2020)

Avcıların, avın sonraki hareketlerini tahmin etmede çok daha fazla bilişsel çabaya ihtiyaç duydukları düşünülmektedir ve bu nedenle başarılı bir avdan sonra daha büyük bir et parçası ile ödüllendirilirler. Bu önemli rol (saldırı) yaş, akıllılık ve fiziksel yetenek ile pozitif ilişkilendirilmektedir. Dahası, şempanzeler aynı av sırasında görevlerini değiştirebilirler veya tüm süreç boyunca aynı görevlerini sürdürebilirler

2.9.1. Avı sürmek ve kovalamak

Av, keşif ve sömürü fazları sırasında olmakta ve Denklem (2.31) ve Denklem (2.32)'deki gibi modellenebilmektedir.

$$d = |cx_p(t) - mx_c(t)| \tag{2.31}$$

$$x_c(t+1) = x_p - ad \tag{2.32}$$

Burada t o anki iterasyon sayısını, a, m ve c katsayı vektörlerini, x_p avın pozisyonunu ve x_c de şempanzenin pozisyonunu belirtmektedir. Katsayı vektörleri Denklem (2.33) - (2-35)'teki gibi hesaplanmaktadır.

$$a = 2fr1 - f \tag{2.33}$$

$$c = 2r2 \tag{2.34}$$

$$m = kaotik\ değer \tag{2.35}$$

Burada yineleme işlemi boyunca (hem sömürü hem de keşif aşamasında), f doğrusal olmayan bir şekilde 2.5'tan 0'a düşürülmektedir. $r1$ ve $r2$, [0, 1] aralığındaki rastgele vektörler olup m de çeşitli kaotik haritalara dayanarak hesaplanan kaotik bir vektördür. Aynı zamanda bu vektör, şempanzelerin avlanma sürecindeki cinsel motivasyonunun etkisini temsil etmektedir. Şempanze optimizasyon algoritması bağımsız popülasyon grupları kullanmaktadır. Bu bağımsız gruplarının etkinliği aşağıdaki şekilde belirtilebilmektedir:

- Bağımsız grupların f 'i güncellemek için farklı stratejileri vardır. Bu nedenle şempanzeler arama alanını farklı yeteneklerle keşfedebilmektedirler.
- Algoritma, farklı ve dinamik stratejiler arasında küresel ve yerel arama arasında daha iyi bir denge sağlayabilmektedir.
- Bağımsız gruplar, f için logaritmik ve üstel fonksiyonlar gibi doğrusal olmayan stratejiler içermektedir. Böylece karmaşık optimizasyon problemlerini çözmede etkili olabilmektedir.
- Bağımsız gruplara sahip bu algoritma, daha geniş kapsamlı optimizasyon problemlerinin çözümünde uyarlanabilmektedir.

2.9.2. Saldırı yöntemi (sömürü aşaması)

Şempanzelerin saldırgan davranışlarını matematiksel olarak modellemek için algoritmada iki yaklaşım tasarlanmıştır: Şempanzeler, avın konumunu (sürerek, engelleyerek ve kovalayarak) keşfedebilmekte ve ardından onu kuşatabilmektedir. Avlanma süreci genellikle saldırgan şempanzeler tarafından yürütülmektedir. İtici, engelleyici ve istilacı şempanzeler ara sıra avlanma sürecine katılmaktadır. Optimum konuma (av) ait bilgi, arama uzayında bulunmamaktadır. Şempanzelerin davranışını matematiksel olarak simüle etmek için ilk saldırganın (mevcut en iyi çözüm), iticinin, engelleyicinin ve istilacının potansiyel avın yeri hakkında daha iyi bilgi sahibi olduğu varsayılmaktadır. Böylece, elde edilen en iyi çözümlerden dördü saklanır ve diğer

şempanzeler konumlarını en iyi şempanze konumlarına göre güncellemeye zorlanmaktadır. Bu ilişki aşağıdaki denklemlerde ifade edilmektedir.

$$d_a = |c1x_a - m_1x| \qquad d_b = |c2x_b - m_2x| \tag{2.36}$$

$$d_c = |c3x_c - m_3x| \qquad d_d = |c4x_d - m4x| \tag{2.37}$$

$$x_1 = x_a - a1(d_a) \qquad x_2 = x_b - a2(d_b) \tag{2.38}$$

$$x_3 = x_c - a3(d_c) \qquad x_4 = x_d - a4(d_d) \tag{2.39}$$

$$x(t+1) = \frac{x_1+x_2+x_3+x_4}{4} \tag{2.40}$$

2.9.3. Av saldırısı

Bu aşamada, şempanzeler ava saldırmakta ve avın hareketi durur durmaz avı avlamaktadırlar. Saldırı sürecini matematiksel olarak modellemek için, f 'nin değeri azaltılmaktadır. a'nın varyasyon aralığının da f ile azaldığına dikkat edilmelidir. Başka bir deyişle, $a, [-2f, 2f]$ aralığında rastgele bir değişkendir. Oysa f değeri yinelemeler periyodunda 2,5'dan 0'a düşmektedir. Bir şempanzenin rastgele değerleri $[-1, 1]$ aralığında olduğunda, bu şempanzenin bir sonraki konumu, mevcut konumu ile avın konumu arasındaki herhangi bir konumda olabilmektedir. Önerilen itme, engelleme ve takip mekanizması bu şekilde keşif sürecini gösterse de, algoritmanın arama aşamasını vurgulamak için daha fazla operatör gerekmektedir.

2.9.4. Avı arama (keşif aşaması)

Şempanzeler arasındaki keşif süreci, esas olarak istilacı, engelleyici, avcı ve itici şempanzelerin konumu dikkate alınarak yapılmaktadır. Şempanzeler avı aramak için ayrılmakta ve avına saldırmak için toplanmaktadırlar. Sapma davranışını matematiksel olarak modellemek için, 1'den büyük veya -1' den küçük rastgele bir değere sahip bir vektör kullanılmaktadır. Böylece arama bireylerini avdan uzaklaşmaya zorlamaktadır. Bu prosedür keşif sürecini göstermekte ve algoritmanın global olarak arama yapmasına izin vermektedir.

Bir diğer parametre olan c vektörü, şempanzelerin doğada ava yaklaşmasını zorlaştıran engellerin etkisi olarak da düşünülebilmektedir. Genel olarak, şempanzelerin yolundaki doğal engeller, avlarına uygun hızda yaklaşmalarını engellemektedir. Şempanzenin konumuna bağlı olarak, c vektörü avı daha zor veya daha kolay hale getirmek için ava rastgele bir ağırlık atayabilmektedir.

2.9.5. Sosyal teşvik

Son aşamada, tanışma ve ardından sosyal motivasyon edinme, şempanzelerin avlanma sorumluluklarını bırakmasına neden olmaktadır. Bu nedenle, etleri (avları) zorla kaotik bir şekilde elde etmeye çalışmaktadırlar. Bu aşamadaki kaotik davranış, şempanzelerin yüksek boyutlu problemlerin çözümünde yerel minimumun içinde kalma ve yavaş yakınsama hızı gibi iki sorunu daha da hafifletmesine yardımcı olmaktadır. Kaotik haritalar algoritmanın performansının arttırılmasına yardım edebilmektedirler. Önerilen algoritmada 0,7 değeri tüm haritaların ana noktası olarak kabul edilmektedir. Bu eşzamanlı davranışı modellemek için, optimizasyon sırasında şempanzelerin konumunu güncellemek için, normal güncelleme konumu mekanizması veya kaotik model arasında seçim yapma olasılığının %50 olduğu varsayılmaktadır ve matematiksel modeli Denklem (2.41)'de verilmektedir.

$$x_{c(t+1)} = \begin{cases} x_p(t) - ad & eğer\ \partial < 0.5 \\ Kaotik\ değer\ eğer\ \partial \geq 0.5 \end{cases} \tag{2.41}$$

Burada ∂, 0 ile 1 arasında bir rastsal sayıyı ifade etmektedir. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Şempanze Optimizasyon Algoritması
1. Başlangıç parametrelerini tanımla. 2. Şempanzelerin başlangıç pozisyonlarını tanımla. 3. Popülasyonun uygunluk değerini hesapla. 4. Popülasyonu itici, engelleyici, avcı ve istilacı şempanzeler olarak sınıflandır. 5. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi şempanzeyi çözüm olarak seç. b. Sağlanmıyor: - Parametreleri ve popülasyonu güncelle. - Güncel popülasyonun uygunluk değerini hesapla. - 4. Adıma geç.

2.10. Parçacık Sürü Optimizasyon Algoritması

Parçacık sürü optimizasyonu, popülasyon temelli algoritma olup kuş sürülerinin yiyecek arama davranışlarından esinlenerek oluşturulmuş bir optimizasyon metodudur (Kennedy ve Eberhart, 1995). Bu algoritmada popülasyon sürü olarak ifade edilmekte ve bu sürüde bulunan kuşlar/parçacıklar problemin olası çözümlerini temsil eden popülasyonun bireylerini oluşturmaktadır. Algoritma, modellemede kuşların sürü davranışlarından yola çıkarak, kuşların yiyecek arama davranışlarını temel almaktadır. Bu amaçla, en iyi çözüme yakınsamak için sürüde bulunan her bir parçacık kendi pozisyonunu, sürüdeki uygunluk değerlerine göre belirlenen en iyi parçacığın pozisyonuna benzetmeye çalışmaktadır. Parçacıklar, bunun için aralarında bilgi paylaşımı yapabilmekte, aynı zamanda bir önceki tecrübeleri yeni rotalarına/pozisyonlarına yol göstermektedir. Genetik ve diferansiyel gelişim algoritmalarında bulunan çaprazlama ve mutasyon işlemleri parçacık sürü optimizasyon algoritmasında bulunmamaktadır. Daha önce bahsedilen algoritmalara benzer olarak, parçacık sürü algoritmasında bir başlangıç popülasyonu oluşturulmaktadır. Rastgele oluşturulan bu popülasyon rastgele hızlara sahip olan bir sürüyü temsil etmektedir. d boyutlu bir problemde n adet parçacığa sahip sürü, Denklem (2.42) ile ifade edilebilmektedir.

$$X = \begin{bmatrix} X_{1,1} & \cdots & X_{1,d} \\ \vdots & \ddots & \vdots \\ X_{n,1} & \cdots & X_{n,d} \end{bmatrix} \tag{2.42}$$

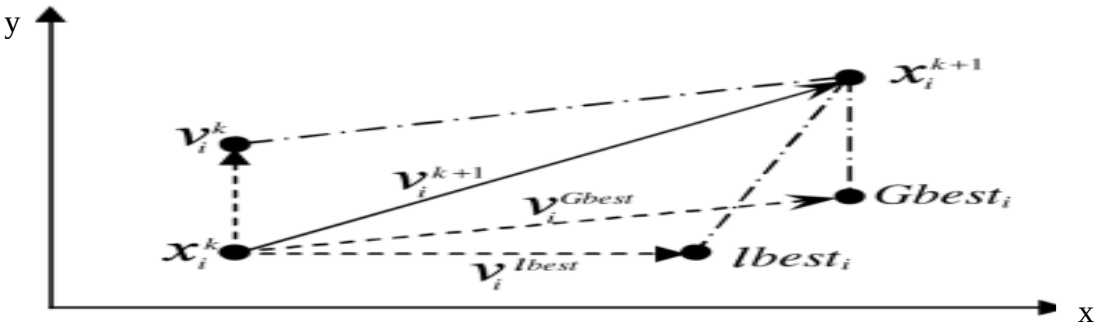
Burada matris bir sürüyü temsil etmektedir ve matrisin satırlarında bu sürüyü oluşturan i. parçacık X_i ile ifade edilmektedir. Yerel ve global arama arasında denge kurmak için, çözüm uzayında tüm populasyon bireyleri sadece en iyi pozisyonunu kullanarak hareket etmek yerine, belirli bir komşuluk tanımı vasıtasıyla yereldeki en iyi pozisyonunu kullanarak da hareket edebilmektedir. Nesiller boyunca sürüdeki parçacıklar, yerel en iyi ve global en iyi değerlerini güncelleyerek yeni pozisyonlarını oluşturmaktadırlar. Yerel en iyi o ana kadarki en iyi uygunluk değerine sahip olan parçacık olmaktadır. Bu değerle ilişkili parçacık, algoritmada *lbest* olarak adlandırılmaktadır. Global en iyi ise, nesiller boyunca oluşturulan sayısız parçacıklar arasından sürüdeki en iyi uygunluk değerine sahip parçacıktır ve *Gbest* olarak isimlendirilmektedir. Bu parçacıkların vektör gösterimi Denklem (2.43) ve Denklem (2.44)'te verilmektedir.

$$lbest_i = [\mathrm{lbest}_{i,1}, \mathrm{lbest}_{i,2}, \dots, \mathrm{lbest}_{i,D}] \quad (2.43)$$

$$Gbest = [\mathrm{l}_1, \mathrm{l}_2, \dots, \mathrm{l}_D] \quad (2.44)$$

Burada o ana kadarki en iyi uygunluk değerini veren $i.$ parçacığın pozisyonu $lbest_i$, nesiller boyu üretilen en iyi parçacık ise $Gbest$ olarak ifade edilmektedir. Parçacıklar, bu iki en iyiden yararlanarak en iyi çözüme yakınsamayı amaçlamaktadırlar.

Çözüm uzayında yerel minimum noktalarına takılmamak ve çözüm uzayını verimli bir şekilde taramak için farklı pozisyonlara sahip parçacıkların bulunması gerekmektedir. Parçacık sürü optimizasyon algoritmasında, hız vektörü parçacığın mevcut konumlarına eklenerek, ilgili parçacığın son konumlarının üretilmesinde yardımcı olmaktadır. Şekil (2.10)'dan görüldüğü üzere, parçacığın yeni pozisyon bulma stratejisi $Gbest$ ve $lbest$'e bağlı olarak ilişkilendirilmektedir. Popülasyon hafızasında bulunan $Gbest$ ve $lbest$'e parçacıkları ve $i.$ parçacığın her konumdaki değişimini ifade eden V_i hız vektörü vasıtasıyla $i.$ parçacık yeni pozisyonunu elde etmektedir (Kuyu, 2016).



Şekil 2.10. Konum bulma stratejisi (Allaoua Laoufı, Gasbaoui ve Abderrahmani, 2009)

Hız vektörü, parçacıkların, üretilen çözüm uzayında hızlı veya yavaş hareket etmelerini belirlemektedir. Hız vektörünün büyük olması durumunda, parçacıklar çok hızlı hareket eder ve en iyi çözüme yaklaşmak yerine rastgele noktalara doğru yol alabilmektedirler. Bunu engellemek için, algoritmada hız vektörüne bir limit değeri tanımlanarak, hız vektörü sınırlandırılmaktadır. İterasyonlar boyunca, hız vektörü güncellenebilmektedir. Bu durum yeni potansiyel çözümler veya daha iyi çözümler üretmeye yardım etmektedir. Ayrıca bu çözümlere dayanarak - eğer üretilen çözüm daha iyiyse - *Gbest* ve *lbest*

değerleri de güncellenmektedir. Parçacık sürü optimizasyon algoritmasının kullandığı parametreler genel olarak aşağıdaki gibi özetlenebilmektedir (Tamer ve Karakuzu 2006).

- Parçacık sayısı (n): Probleme bağlı olarak değişmektedir. Problemin zorluğu arttıkça yüksek değerler alınabilmektedir.
- Parçacık boyutu (d): Genel anlamda, ele alınan problemin kompleksliğini belirtmektedir.
- Parçacık değer aralığı: Ön tanımlı olarak verilmekte, problemden probleme değişkenlik göstermekte ve problem boyutu bazında değer alabilmektedir.
- $Vmax$: Parçacığın ulaşabileceği maksimum hızı (değişim) belirtmektedir. Örnek olarak, bir parçacığın değer aralığı [-50,50] ise $Vmax$=100 ile sınırlandırılabilmektedir.
- Öğrenme faktörleri (c_1, c_2): Arama uzayında, c_1 parçacığın $lbest$, c_2 ise $Gbest$ değeri etrafında arama yapmasını sağlar. Eberhart ve Shi (2000) yaptığı çalışmada bu katsayıların birbirine eşit olmasını ve 1.5'a yakın değer almasını önermektedir.
- Rastgele sayılar (r_1, r_2): [0 1] aralığında değer alabilen sayıları ifade etmektedir.
- Eylemsizlik sabiti (w): Parçacık hızı için kullanılabilen katsayı olmaktadır.

2.10.1. Hız ve konum güncelleme

Sürüdeki parçacıklar, çözüm arama sürecinde, arama uzayındaki konumlarını değiştirerek daha iyi çözümler bulmayı ve optimal çözüme yakınsamayı amaçlamaktadırlar. Bir hız vektörü ile problemin olası çözümlerini temsil eden her bir parçacık, parçacığın yeni konumunu Denklem (2.45)'te belirlemektedir.

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2.45}$$

Burada, t o anki nesli, X_i o anki parçacığı, V_i parçacığa ait hız vektörünü temsil etmektedir. Yeni parçacık pozisyonu bulunduktan sonra, bu yeni parçacığa ait yeni pozisyonu belirlemede kullanılacak olan hız vektörü, Denklem (2.46)'da hesaplanmaktadır. Parçacık, kendisinin ve sürünün o ana kadarki en iyi pozisyonunu kullanarak yeni bir hız formülü oluşturmaktadır.

$$V_i(t+1) = wV_i(t) + c_1 r_1(t)\big(lbest_i(t) - X_i(t)\big) + c_2 r_2(t)\big(Gbest_i(t) - X_i(t)\big) \quad (2.46)$$

Denklemden görüldüğü üzere, yerel ve global en iyi vasıtasıyla, algoritmanın parametrelerini kullanarak yeni bir hız vektörü elde edilmektedir. Her bir parçacık için bu işlem tekrarlanmaktadır. Bu denklemde, öğrenme faktörleri (c_1, c_2), algoritmanın global ve yerel arama yetenekleri arasındaki dengeyi gözetmek amacıyla ve eylemsizlik sabiti (w), önceki ve sonraki hız vektörlerinin birbiri üzerindeki etkisini arttırmak veya azaltmak için kullanılmaktadır. Algoritmada durdurma kriteri olarak, kullanıcı tarafından belirtilen maksimum iterasyon sayısı, istenen minimum değerden daha iyi bir değeri sağlama (tolerans) veya belirli bir iterasyon sayısı kadar gelişim göstermeme gibi özellikler şeklinde belirlenebilmektedir. Algoritmanın adımları aşağıdaki gibi özetlenebilmektedir.

Parçacık Sürü Optimizasyon Algoritması
1. Başlangıç parametrelerini tanımla. 2. Başlangıç sürüsünü üret. 3. Sürünün uygunluk değerini hesapla. 4. Sürünün konum ve hız bilgisini güncelle. 5. Durdurma kriteri sağlanıyor mu? a. Sağlanıyor: - En iyi çözümü seç. b. Sağlanmıyor: - 3. Adıma geç.

3. ÖNERİLEN OPTİMİZASYON ALGORİTMALARI

Bu bölümde doktora tezinde önerilen; geometrik sekizli bölge arama algoritması ve geliştirilen, modifiye adli tıp temelli soruşturma algoritması ile modifiye yığın tabanlı hiyerarşik optimizasyon algoritması açıklanacaktır.

3.1. Geometrik Sekizli Bölge Arama Algoritması (GSBA)

Bu algoritmanın temel prensibi, Julius Sezar'ın "Divide et Impera", yani böl ve fethet stratejisinden (Holmes, 1911; Kahn, 2000) ilham alınarak oluşturulmuştur. Bu strateji ile tarih boyunca rakiplerini bölerek ya da onları bölünmüş vaziyette tutarak, bütüncül durumdaki hallerinden daha zayıf durumda bırakması amaçlanmıştır. "Böl" ilkesinden yararlanarak, problemin arama alanı, bireylerin uygunluk değerlerine göre daha küçük ayırt edici bölgelere ayrılmaktadır. İlham alınan "fethet" metodolojisinde, en iyi bireyler, tüm arama alanında bulunan en iyi tek çözümü fethetmek için en umut verici bölgelerde yeniden birleştirilmektedirler.

Bu felsefeye benzer şekilde, GSBA'da, tüm çözümler (bireyler) uygunluk değerlerine göre en iyiden en kötüye doğru sıralanır. Buna bağlı olarak, popülasyonda bireylerin yerleşimi en iyi birey ilk birey ve en kötü birey son birey olarak yerleştirilmektedir. Daha sonra, sıralanmış bireyleri içeren popülasyon sekiz bölgeye ayrılmaktadır. Bu bölgelerde, en iyi çözüm (X_{best}), ve iyi çözümleri içeren $zone_1$ merkez ve referans bölge olarak gözetilmektedir. Diğer bölgelerin $zone_1$'e olan mesafeleri, ilgili bölgelerin uygunluk değerlerinin medyanlarına göre hesaplanmaktadır.

3.1.1. Başlatma

Diğer metasezgisel algoritmalara benzer olarak, başlangıç çözümleri, tekdüze dağılım vasıtasıyla, verilen çözüm uzayında Denklem (3.1)'deki gibi oluşturulmaktadır.

$$X = lb + rand(ub - lb) \tag{3.1}$$

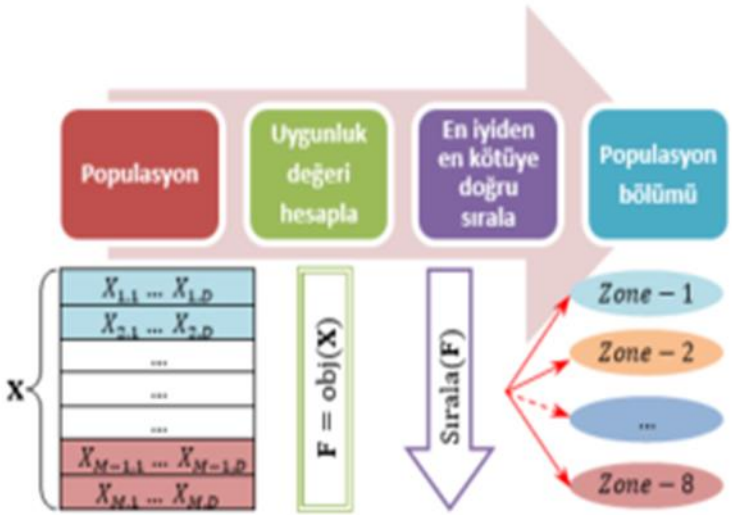
Denklem (3.1)'de, lb ve ub problemin alt ve üst sınır değerleridir. $rand$ ise [0,1] aralığında bir rastsal vektördür. Başlangıç popülasyonu Denklem (3.2)'deki gibi gösterilebilmektedir.

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_M \end{bmatrix} \quad , \quad X_i = \left[X_{i,1}, X_{i,2}, \dots, X_{i,D}\right] \quad , \quad i = 1, \dots, M \tag{3.2}$$

Burada *M* popülasyon büyüklüğünü ve *D* ise problemin boyutunu ifade etmektedir.

3.1.2. Popülasyon Bölümü

Popülasyon üyeleri, uygunluk değerlerine göre en iyiden en kötüye doğru sıralanır ve daha sonra sekiz bölgeye ($zone_i, i = 1, \dots, 8$) ayrılır.



Şekil 3.1. Popülasyon bölümü işlemi (Kuyu ve Vatansever, 2022a)

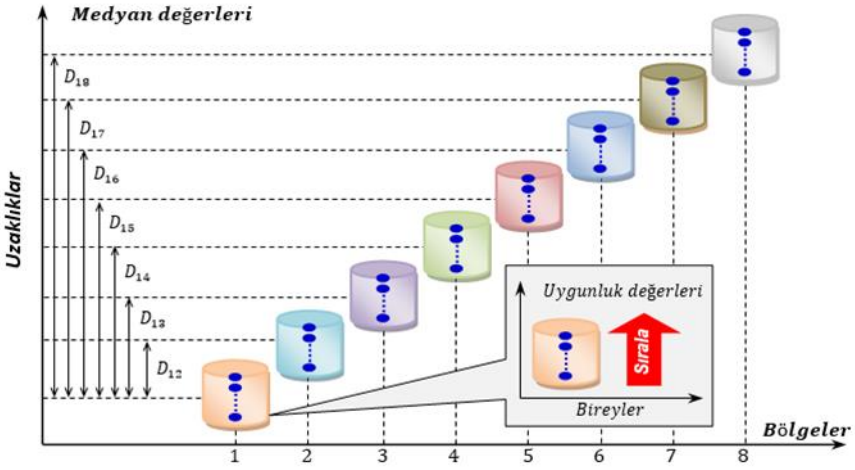
Şekil (3.1)'de, M elemanlı popülasyon bireylerinin uygunluk değerleri hesaplandıktan sonra sıralanarak, sekiz farklı bölgeyi temsilen ayrılmaktadır.

3.1.3. Bölgelerin Güncelleme Mekanizmaları

Önerilen algoritmadaki "geometrik" terimi, mesafenin sembolik bir şekilde bulunmasını ifade etmektedir. İlk olarak, her bir bölgenin birbirine olan uzaklığı, Denklem (3.3)'te gösterilen uygunluk değerlerinin medyan değerleri (med) kullanılarak hesaplanmaktadır.

$$D_{1j} = abs\{med[F(zone_1)] - med[F(zone_j)]\} \quad (3.3)$$

Burada, D_{1j} birinci ve j. bölgeler arası uzaklık, F hedef fonksiyon, $j = 2, \ldots, 8$. Bölgelerin uzaklıklara göre temsili yerleşimi Şekil 3.2'de ifade edilmektedir. Belirtilen şekilde popülasyon büyüklüğü 16 ve problemin boyutu 1 olarak kabul edilmektedir, x ekseni içinde koyu maviyle gösterilen bireylerin bulunduğu, birden sekize kadar olan bölgeleri; y ekseni ise uygunluk değerlerinin medyanlarına bağlı olarak değişen uzaklıkları belirtmektedir.



Şekil 3.2. Bölgelerin uzaklıklara göre gösterimi (Kuyu ve Vatansever, 2022a)

Temel felsefe olarak, referans bölge $zone_1$'i, kullanarak, ilk iki bölgenin ($zone_1$ ve $zone_2$) bireyleri oluşturulmaktadır. Merkeze (X_{best}) en yakın olan bölgeler katsayı vektöründeki en büyük katsayılara sahipken, merkeze en uzak bölgeler en düşük değerli katsayılara sahiptir. Her bir bölge $[-1, 0]$ aralığında rastsal olarak üretilen katsayılara (c_{zone_i}) sahiptir. Bu büyük katsayılar $zone_1$ ve $zone_2$`nin adım büyüklüğünün bir parçası olarak, c_{zone_i} ve D_{12} arasında denge oluşturmak üzere kullanılmaktadır. Başlangıç iterasyonlarında, ters orantılı D_{12} büyük olmaktadır; çünkü X_{zone_1} ve X_{zone_2} popülasyonun bireyleri arama uzayında iyi dağılmamış olabilmekte, bu da birbirlerinden uzakta kalabilme olasılığını doğurmaktadır. D_{12} uzaklığının paydada yer alması, diğer bir deyişle bölüm şeklinde kullanılması büyük adım büyüklüklerini engelleyebilecektir. Buradan yola çıkarak, ilk iki bölgenin ($zone_1$ ve $zone_2$) güncellemesi Denklem (3.4)'teki gibi olmaktadır:

$$\left.\begin{aligned} X_{zone_1} &= \left(X_{zone_1} - X_{zone_2}\right) + \left(\frac{kt}{D_{12}}\right) c_{zone_1} + X_{zone_1} \\ X_{zone_2} &= \left(X_{zone_1} - X_{zone_2}\right) + \left(\frac{kt}{D_{12}}\right) c_{zone_2} + X_{zone_2} \end{aligned}\right\} \tag{3.4}$$

Burada t o bölgeye ait popülasyonun indeksi, k rastsal bir sayı, c_{zone_1} ve c_{zone_2} ilişkili bölgelere ait katsayılardır. Denklemden görüldüğü üzere, $zone_1$ ve $zone_2$ arama yönleri, $zone_1$ ve $zone_2$ farkından yararlanarak güncellenmektedir. Bu iki bölge, arama uzayında algoritmanın o ana kadar bulduğu en iyi bireylere sahip bölgeler olduğundan, bölgelerdeki bu bireyler, en iyi çözümlere dayanarak, bu çözümler etrafında daha iyi çözümler aramayı amaçlamaktadır. Diğer yandan, $zone_3$ (3. bölge) ve $zone_4$ (4. bölge), "if –else" seçilim kuralına dayanarak, en iyi çözüm X_{best} aracılığıyla Denklem (3.5)'teki gibi güncellenmektedir.

$$\begin{cases} \{X_{zone_3}(t,d) = X_{best}(t,d) \quad X_{zone_4}(t,d) = X_{best}(t,d) \quad X_{zone_5}(t,d) = X_{best}(t,d) \quad X_{zone_6}(t,d) = X_{best}(t,d) \quad X_{zone_7}(t,d) = X_{best}(t,d) & if\ \left(rand > \frac{1}{2}\right) \\ else \begin{cases} X_{zone_3}(t,:) = X_{best}(t,:) + \left(\frac{k}{D_{13}i}\right) * \left(\frac{1}{3}\right) * c_{zone_3} \\ X_{zone_4}(t,:) = X_{best}(t,:) + \left(\frac{k}{D_{14}i}\right) * \left(\frac{1}{3}\right) * c_{zone_4} \\ X_{zone_{5_t}}(t,d) = X_{zone_5}(t,d), \quad X_{zone_5}(t,:) = \{X_{zone_1}(t,:) - X_{zone_m}(t,:)\}\left(\frac{ki}{D_{15}}\right) * \left(\frac{1}{5}\right) * c_{zone_5}, \ X_{zone_5}(t,d) = X_{zone_{5_t}}(t,d) + X_{zone_5}(t,d) \\ X_{zone_{6_t}}(t,d) = X_{zone_6}(t,d), \quad X_{zone_6}(t,:) = \{X_{zone_1}(t,:) - X_{zone_m}(t,:)\}\left(\frac{ki}{D_{16}}\right) * \left(\frac{1}{5}\right) * c_{zone_6}, \ X_{zone_6}(t,d) = X_{zone_{6_t}}(t,d) + X_{zone_6}(t,d) \\ X_{zone_{7_t}}(t,d) = X_{zone_7}(t,d), \quad X_{zone_7}(t,:) = \{X_{zone_1}(t,:) - X_{zone_m}(t,:)\} + X_{best}(t,:), \ X_{zone_7}(t,d) = \left\{X_{zone_{7_t}}(t,d) + X_{zone_7}(t,d)\right\} * \left(\frac{1}{D_{17}}\right) * c_{zone_7} \end{cases} \end{cases} \tag{3.5}$$

$d = randi(D)$ bir rastsal tamsayı, D problemin boyutu, X_{best} o ana kadarki bulunan en iyi çözüm, $X_{zone_j}(t,:)$ $zone_j$ popülasyonunun bir üyesi ve $X_{zone_j}(t,d)$ X_{zone_j}'nin *t.* bireyinin *d.* değişkenidir. Denklem (3.5)'ten görülebileceği üzere, X_{zone_3}ve X_{zone_4}, X_{best} etrafında hareketlerini sürdürmektedir ve X_{best}'in bilgilerinden yararlanabilmektedir. Bu bölgelerdeki bireyler (X_{zone_3}ve X_{zone_4}), $zone_1$ ve $zone_2$ bölgelerine en iyi çözüm vasıtasıyla yakın arama yapmaktadırlar. Ayrıca farklı adım büyüklüğü gelebilme ihtimali sebebiyle uzun adım büyüklüklerine de sahip olabilmektedirler. Denklem (3.5)'te hesaplanan X_{zone_5} ve X_{zone_6}, değişkenleri arasında bilgi değişimi yaparak X_{zone_m} vasıtasıyla kendini güncellemektedir. Burada X_{zone_m} rastsal olarak $zone_m$'den seçilen bir bireydir. m ise rastsal bir tam sayı değeridir ve $m = 1$ durumu en iyi çözümü ifade etmektedir. Diğer yandan, eğer en iyi çözüm X_{best} yerel minimuma sıkışırsa, X_{zone_m}vasıtasıyla, X_{zone_5} and X_{zone_6} çözümleri geliştirebilmektedir. Bu durum ayrıca

popülasyon çeşitliliğine katkı sağlamaktadır. Denklemden görüleceği üzere, $zone_7$ bireylerinin güncellemesi, $\{X_{zone_1} - X_{zone_m}\}$ fark vektörünün, X_{best}'e eklenmesiyle oluşmaktadır. Bu, $zone_7$ bireylerinin, hem $\{X_{zone_1} - X_{zone_m}\}$ farkının büyük gelmesi durumunda komşuluk bölgeleri keşfetmesini sağlarken, hem de $m = 1$ olması durumunda X_{best} çevresinde yeni çözümler elde etme olanağı sağlamaktadır. En kötü bireyleri içeren $zone_8$ (8. bölge), Denklem (3.6)'daki gibi güncellenmektedir.

$$\left.\begin{array}{c} d_1 = rand_{perm}(pn_8)\,,\; d_2 = rand_{perm}(pn_8)\,,\; V1_{zone_{8t}} = pop_{zone_8}(d_1,:)\,,\; V2_{zone_{8t}} = pop_{zone_8}(d_2,:) \\ X1_{zone_{8t}} = \{k\,V1_{zone_{8t}}(n,:) + (1-k)\,V1_{zone_{8t}}(n+1,:) + k\,V2_{zone_{8t}}(n,:) + (1-k)\,V2_{zone_{8t}}(n+1,:)\}\left(\frac{c_{zone_8}}{D_{18}}\right) \\ X2_{zone_{8t}} = \{k\,V1_{zone_{8t}}(n+1,:) + (1-k)\,V1_{zone_{8t}}(n,:) + k\,V2_{zone_{8t}}(n+1,:) + (1-k)\,V2_{zone_{8t}}(n,:)\}\left(\frac{c_{zone_8}}{D_{18}}\right) \end{array}\right\} \quad (3.6)$$

Burada pn_8 8. bölgenin ($zone_8$) popülasyon büyüklüğü ve $rand_{perm}$ rastsal permütasyon fonksiyonudur ($n = 1, \dots, pn_8 - 1$). $X1_{zone_{8t}}$ ve $X2_{zone_{8t}}$ bireyleri arasındaki en iyi uygunluk değerine sahip birey, $zone_8$ popülasyonunu oluştururken; önceki nesilde bulunan en iyi birey, $zone_8$ popülasyonundaki en kötü birey ile yer değiştirmektedir. En iyi bireyin seçiliminde, en kötü bölge olan $zone_8$ (8. bölge), gelecek vadeden bölgeleri keşfetmeyi amaçlarken; en iyi bireylere sahip $zone_1$ (1. bölge) ve $zone_2$ (2. bölge), en iyi bireyin komşuluğunda keşif yapmaktadır. İterasyonun sonunda farklı bölgelere ait alt popülasyonlar, tekrardan tek bir popülasyonda birleştirilmektedir. Sıralama işlemi bireylerin uygunluk değerlerine göre tekrar uygulanmakta ve tüm popülasyon bölgelere ayrılmaktadır. Bu strateji, bireylerin farklı bölgelerde bulunma ihtimalini oluşturacağından, popülasyondaki bir birey, farklı iterasyonlarda, farklı denklemler aracılığıyla geliştirilebilmekte ve böylece yerel minimumdan kaçma olasılığını arttırabilmektedir.

Farklı özelliklere sahip problemlerin optimizasyonunda (tek-modlu, çok-modlu, hibrid gibi), birden farklı yöntem (denklem) ile olası çözümleri geliştirmek, tek bir yöntemle geliştirmeye nazaran daha yararlı olabilmektedir. Bunun anlamı, eğer bir yöntem olası çözümü geliştirmez ise, çözüm diğer bir yöntem ile geliştirilebilmekte, yerel minimumlardan atlayabilmekte, bu da algoritmanın çözüm uzayında arama kapasitesini hızlandırmaktadır. Diğer yandan, geliştirilen yöntemler birbiriyle paralel olarak, işbirliği içinde hareket ettiğinden, uygulanan problemler değişkenlik gösterdiğinden, analitik olarak hangi yöntemin olası çözümlerin geliştirilmesinde katkı sağladığını bulmak, bu gibi çoklu yöntemlerin kullanıldığı algoritmalarda zorlaşmaktadır. Bu çalışmada,

önerilen geometrik sekizli bölge arama algoritmasının efektifliğini göstermek için, algoritma, diğer bölgelerdeki yöntemler göz ardı edilerek, 2 bölge ve 4 bölge olarak da çalıştırılmıştır. Ortalama (*Ort*) ve standart sapma (*Std*) değerleri, 30 bağımsız çalıştırılma üzerinden, literatürde sıklıkla kullanılan, EK 1'de verilen, 23 nümerik fonksiyon kullanılarak hesaplanmıştır. Çizelge (3.1) ve Çizelge (3.2)'de algoritmalar hem 50 popülasyonlu hem 100 popülasyonlu olarak, 2 ve 4 bölgelere ayırarak kıyaslanmaktadır. Çizelgelerden görülebileceği üzere, önerilen algoritmada bölge sayısını 8 olarak seçmek, 2 veya 4 bölge seçmeye göre, ortalama sonuç değerleri (*Ort*) kıyaslandığında daha çok en iyi çözüme ulaşmaktadır. Buradan yola çıkarak, bölge sayısını 8 olarak seçmek, 2 ve 4 olarak seçmeye göre daha avantajlı olmaktadır.

Çizelge 3.1. Önerilen algoritmanın bölge sayılarına göre nümerik sonuçları (80 popülasyon)

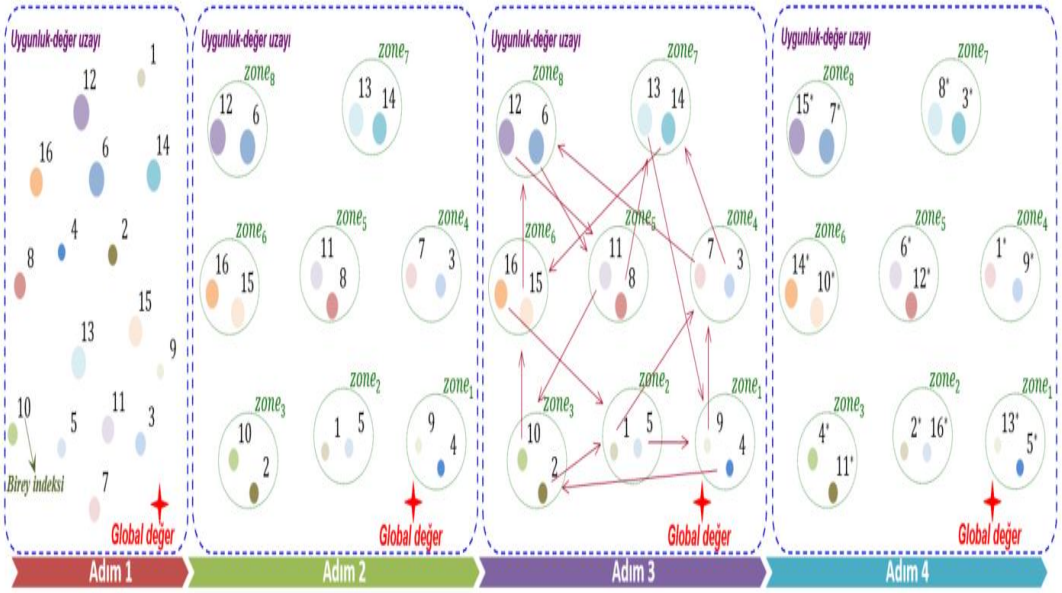
Bölge sayısı		***2***	***4***			***2***	***4***
F1	*Ort*	4.310e+03	1.379e+04	**F13**	*Ort*	2.041e+9	6.832e+8
	Std	4.864e+03	8.540e+03		*Std*	2.972e+8	3.105e+8
F2	*Ort*	1.737e+22	2.638e+16	**F14**	*Ort*	1.301	0.998
	Std	6.176e+22	1.088e+17		*Std*	1.255	1.523e-15
F3	*Ort*	1.302e+05	9.668e+04	**F15**	*Ort*	0.001	7.873e-04
	Std	9.446e+04	3.767e+04		*Std*	0.001	2.102e-04
F4	*Ort*	15.038	42.978	**F16**	*Ort*	-1.030	-1.031
	Std	10.135	8.400		*Std*	0.001	5.709e-07
F5	*Ort*	3.894e+08	1.542e+08	**F17**	*Ort*	0.464	0.397
	Std	9.199e+07	5.250e+07		*Std*	0.212	3.364e-07
F6	*Ort*	5.015e+03	1.116e+04	**F18**	*Ort*	3.018	3.000
	Std	1.314e+04	5.106e+03		*Std*	0.060	1.642e-07
F7	*Ort*	48.153	1.673	**F19**	*Ort*	-3.861	-3.862
	Std	29.186	1.768		*Std*	0.001	9.887e-06
F8	*Ort*	-5.505e+3	-7.652e+03	**F20**	*Ort*	-3.160	-3.269
	Std	5.879e+02	1.080e+03		*Std*	0.094	0.059
F9	*Ort*	5.735e+02	4.281e+02	**F21**	*Ort*	-5.152	-10.153
	Std	54.863	55.689		*Std*	2.087	1.135e-04
F10	*Ort*	13.379	15.749	**F22**	*Ort*	-4.930	-8.367
	Std	4.729	1.260		*Std*	2.203	3.167
F11	*Ort*	3.357e+02	1.112e+02	**F23**	*Ort*	-5.731	-9.127
	Std	1.032e+02	48.793		*Std*	2.386	2.874
F12	*Ort*	9.582e+08	2.938e+08				
	Std	1.932e+08	1.316e+08				

Çizelge 3.2. Önerilen algoritmanın bölge sayılarına göre nümerik sonuçları (100 popülasyon)

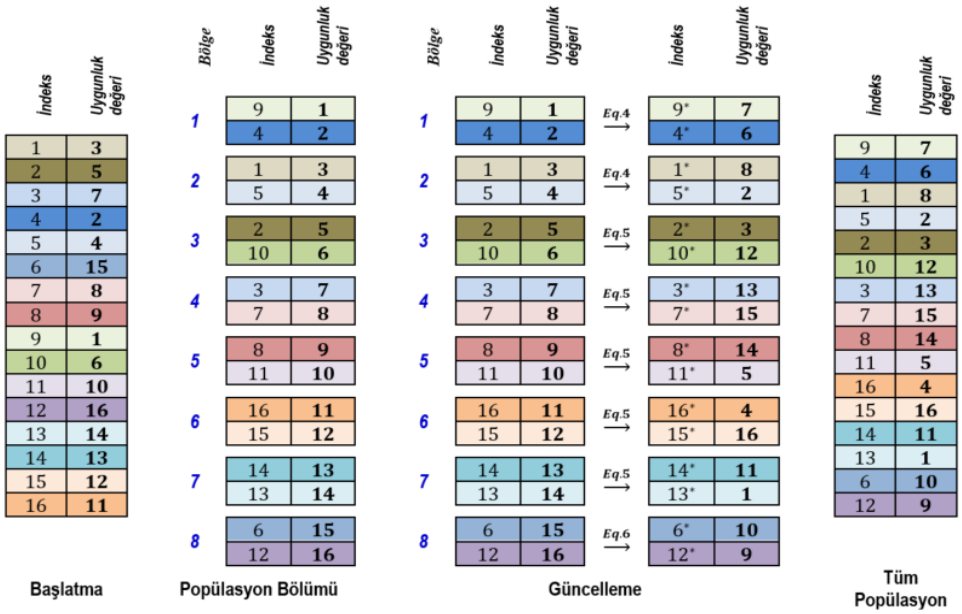
Bölge sayısı		***2***	***4***	***8***			***2***	***4***	***8***
F1	*Ort*	2.541e+03	1.537e+04	6.9e-10	**F13**	*Ort*	1.915e+9	5.657e+8	0.032
	Std	3.288e+03	1.031e+04	8.8e-10		*Std*	2.020e+8	1.904e+8	0.056
F2	*Ort*	5.843e+20	1.994e+15	1.7e-10	**F14**	*Ort*	1.090	0.998	0.998
	Std	1.050e+21	4.743e+15	1.2e-10		*Std*	0.282	1.308e-15	8.36e-15
F3	*Ort*	1.294e+05	8.047e+04	3.8e-07	**F15**	*Ort*	0.002	8.843e-04	3.483e-4
	Std	1.218e+05	1.809e+04	5.8e-07		*Std*	0.001	3.245e-04	3.46e-05
F4	*Ort*	10.665	42.967	4.6e-05	**F16**	*Ort*	-1.031	-1.031	-1.031
	Std	3.166	9.893	4.1e-05		*Std*	4.281e-04	2.130e-07	1.40e-07
F5	*Ort*	3.517e+08	1.370e+08	9.70	**F17**	*Ort*	0.053	0.397	0.397
	Std	7.813e+07	4.085e+07	20.14		*Std*	0.212	2.599e-07	1.26e-07
F6	*Ort*	1.598e+03	1.112e+04	0.02	**F18**	*Ort*	3.026	3.000	3.000
	Std	7.846e+02	6.802e+03	0.016		*Std*	0.058	1.118e-07	9.36e-11
F7	*Ort*	53.358	0.980	0.001	**F19**	*Ort*	-3.860	-3.862	-3.862
	Std	30.186	0.857	0.001		*Std*	0.001	1.945e-05	7.43e-07
F8	*Ort*	-5.5681e+3	-7.812e+03	-1.3e+4	**F20**	*Ort*	-3.146	-3.249	-3.274
	Std	3.74e+02	9.593e+03	1.0e+03		*Std*	0.093	0.061	0.061
F9	*Ort*	5.628e+02	4.309e+02	9.64e-8	**F21**	*Ort*	-4.366	-10.153	-10.153
	Std	41.117	90.945	1.89e-07		*Std*	1.050	1.159e-04	9.01e-06
F10	*Ort*	14.504	15.565	1.04e-4	**F22**	*Ort*	-4.592	-9.735	-9.734
	Std	4.387	1.106	1.09e-04		*Std*	1.175	2.111	2.111
F11	*Ort*	3.331e+02	1.00e+02	4.04e-9	**F23**	*Ort*	-4.565	-9.866	-9.196
	Std	1.032e+02	46.124	1.05e-08		*Std*	0.803	2.119	2.825
F12	*Ort*	8.998e+08	2.575e+08	0.001					
	Std	1.062e+08	1.319e+08	0.001					

Algoritmanın adımlarını daha ayrıntılı açıklamak gerekirse, toplam popülasyon büyüklüğünün 16 ve bireylerde hesaplanan uygunluk değerlerinin $[1, 16]$ arasında olduğunu farzedelim. Önerilen algoritma temel olarak üç ana adıma sahiptir; başlangıç popülasyonu oluşturma, popülasyon bölümü ve popülasyonun güncellenmesi. Şekil 3.3 adım 1'den görüleceği üzere, başlangıç popülasyonunda olan bireyler rastsal olarak arama uzayına dağılmışlardır. Şekil 3.4'te, Şekil 3.3'ün popülasyon gösterimi yer almaktadır. Her iki şekil gösteriminde de, başlangıç popülasyonu oluşturulduktan sonra, popülasyon bölümü işlemine geçilmektedir. Bu işlemde, başlangıç popülasyonu, uygunluk değerlerinden yararlanılarak, 8 bölgeye bölünmektedir (Şekil 3.3 adım 2 ve Şekil 3.4). Algoritmanın üçüncü adımı popülasyonun güncellenmesi olmaktadır. Bu adımda, farklı arama mekanizmaları, her bir bölgeye uygulanarak, yeni bireylerin oluşturulması sağlanmaktadır. Şekil 3.4'te indeksler, $1*, 2*, \ldots, 16*$ yeni bireylerin indekslerini belirtmektedir. Şekil 3.3 adım 3, bireylerin güncellendikten sonraki hareket yönlerini ifade etmektedir. Örneğin**,** $zone_1$ (1. bölge) ve $zone_7$'yi (7. bölge) göz önüne alırsak, ilgili şekilde, $zone_7$'de 13 indeksine sahip bireye ait çözüm iyileşmiştir. Bunun anlamı $zone_7$'ye uygulanan geliştirme mekanizmasının bu bireye pozitif etki ettiğidir.

Diğer yandan, $zone_1$'deki bireylere uygulanan mekanizma, $zone_7$ 'nin tam tersi olarak, $zone_1$ bireyleri üzerinde bir gelişim sağlayamamaktadır. Fakat gelişim gösteremeyen $zone_1$ bireyleri sonraki iterasyonda farklı mekanizmalara maruz kalacağından, bu bireylerin de gelişim şansı bulunmaktadır. Bu hiyerarşi, bölgeler arasında gelişim gösteremeyen bireylere de gelişim şansı tanımaktadır. Algoritma farklı bölgelerdeki gelişim süreçlerini kullanarak, Şekil 3.4'te görüldüğü üzere, yeni popülasyonu oluşturmaktadır. Yukarıda bahsi geçen tüm süreçler (Şekil 3.3 adım 2-4), durdurma kriteri sağlanıncaya kadar devam etmektedir. Bu hiyerarşi, aynı zamanda keşif ile sömürü arasında bir denge kurmayı amaçlamaktadır.



Şekil 3.3. Önerilen algoritmanın adımlarının uygunluk değer dağılımlarına göre gösterimi (Kuyu ve Vatansever, 2022a)



Şekil 3.4. Önerilen algoritmanın adımlarının popülasyon biçiminde gösterimi (Kuyu ve Vatansever, 2022a)

Geometrik Sekizli Bölge Arama Algoritması (GSBA)

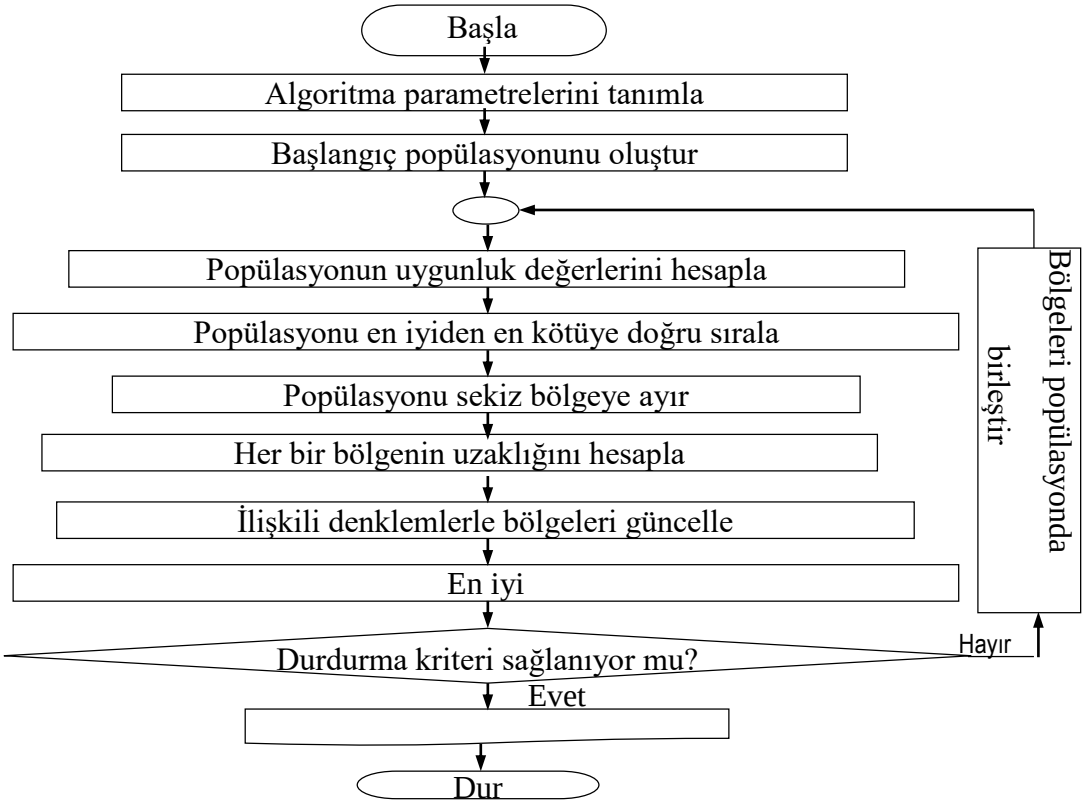
1. Popülasyon büyüklüğü, durdurma kriteri, katsayı vektörü tanımla
2. Başlangıç popülasyonu oluştur
3. Popülasyonun uygunluk değerlerini hesapla
4. Bireyleri uygunluk değerlerine göre sırala
5. Bireyleri sekizli bölgeye grupla
6. **While** (Durdurma kriteri)
- **For** her bir bölge
  - Bölgelerin uzaklıklarını denklem (3.3)'ü kullanarak hesapla
  - $zone_1$ ve $zone_2$'yi denklem (3.4)'ü kullanarak güncelle
  - Bölgeler 3-7'yi denklem (3.5)'i kullanarak güncelle
  - $zone_8$ 'i denklem (3.6)'yı kullanarak güncelle

end For
- Her bir bölgedeki bireylerin uygunluk değerini hesapla
- Sekizli bölgeyi tek bir bölgede birleştir
- Bireyleri sırala ve bölgelere ayır
- Bölgelerin uzaklıklarını hesapla

end while

7. Bulunan en iyi çözümü al

Şekil 3.5. Önerilen algoritmanın sözde kodu



Şekil 3.6. Önerilen algoritmanın akış diyagramı

3.1.4. Parametre Analizi

Parametre yapılandırılması, metasezgisel algoritmaların çözüm süreçlerine etki eden ve uygun seçimlerde algoritmanın performansını arttırabilen önemli bir süreçtir. GSBA algoritmasının tek parametresi katsayı vektörü olmaktadır. Bu vektör ön tanımlı bir aralık içinde oluşturulmakta ve vektördeki her bir katsayı bir bölgeye atanmaktadır. Ön tanımlı katsayı vektörünün optimal sınırlarını bulmak amacıyla, bu vektör çeşitli sınır varyasyonlarında örneklenmiş ve 23 nümerik fonksiyon üzerinden (Saremi, Mirjalili ve Lewis, 2017; Mirjalili ve Lewis, 2016; Mirjalili, 2015a; 2015b) ortalama değerlerine göre en uygun aralık bulunmuştur. Bu fonksiyonlar; F1-F7; tek modlu, F8-F13; çok modlu, F14-F23; sabit boyutlu-çok modlu fonksiyonlardır. Fonksiyonların özellikleri ayrıca EK 1'de verilmektedir. En iyi aralık, farklı aralıklarda, algoritmanın hesapladığı en iyi ortalama değer (No. best) sayısına göre belirlenmektedir. Diğer yandan, tüm olası aralıkları test etme imkanı olmadığından, bu hesaplama bize mümkün olan en optimal vektör aralığını sağlamaktadır. Nümerik sonuçlar, Çizelge 3.3-3.8'de görülmektedir.

"Parametre Yok" sütunu, algoritmanın parametre vektörü olmadan kullanımından bulunan sonuçları ifade etmektedir. Elde edilen sonuçlardan yola çıkarak, en iyi ortalama değer (No. best) sayı $[-1, 0]$ aralığında olduğundan, bu çalışmada parametre katsayı vektörünün optimal sınırları $[-1, 0]$ arası alınmaktadır.

Çizelge 3.3. GSBA algoritması parametre analizi (1/6)

			Parametre Yok	[0 1]	[0 2]	[0 3]	[0 4]	[0 5]
Tek Modlu	**F1**	Ort	1.076e-05	1.882e-05	5.020e-05	2.461e-04	0.139	0.011
	F2	Ort	1.954e-10	2.806e-11	4.246e-11	4.226e-11	7.654e-11	8.281e-11
	F3	Ort	2.535e-06	1.999e-05	0.013	0.032	0.023	0.017
	F4	Ort	0.002	0.006	0.022	0.026	0.031	0.042
	F5	Ort	11.171	0.291	0.232	0.917	0.751	0.859
	F6	Ort	0.116	0.105	0.074	0.077	0.092	0.420
	F7	Ort	0.002	1.152e-04	3.007e-04	7.734e-04	0.001	0.002
Çok Modlu	**F8**	Ort	-1.110e+04	-1.156+04	-1.244e+04	-1.228e+04	-1.207e+04	-1.110e+04
	F9	Ort	4.565e-06	8.207e-05	7.159e-05	1.898e-04	4.932e-04	1.0897
	F10	Ort	0.001	0.003	0.031	0.017	0.219	0.561
	F11	Ort	3.278e-05	6.6352e-05	6.193e-04	0.002	0.009	0.011
	F12	Ort	0.007	0.001	0.001	0.001	0.001	0.001
	F13	Ort	0.038	0.003	0.002	0.004	0.003	0.003
Sabit boyutlu çok modlu	**F14**	Ort	0.998	0.998	0.998	0.998	0.998	0.998
	F15	Ort	5.688e-04	5.194e-04	6.674e-04	3.902e-04	4.378-04	4.477e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.398	0.398	0.398	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.235	-3.245	-3.183	-3.260	-3.244	-3.212
	F21	Ort	-10.151	-10.152	-9.400	-10.151	-10.149	-10.148
	F22	Ort	-9.065	-9.639	-9.734	-10.401	-8.396	-9.724
	F23	Ort	-8.523	-9.866	-10.535	-10.531	-10.534	-10.529
No. best			2	3	1	0	0	0

Çizelge 3.4. GSBA algoritması parametre analizi (2/6)

			[-1 1]	[-1 0]	[-1 2]	[-1 3]	[-1 4]	[-1 5]
Tek Modlu	**F1**	Ort	3.861e-06	6.378e-06	2.850e-05	7.230e-05	9.076e-05	1.980e-04
	F2	Ort	1.422e-10	1.606e-11	9.827e-11	1.243e-10	7.428e-11	4.847e-11
	F3	Ort	2.106e-05	4.567e-05	0.002	0.002	0.026	0.025
	F4	Ort	0.002	0.002	0.004	0.014	0.025	0.058
	F5	Ort	3.862	4.242	0.640	0.450	0.717	2.141
	F6	Ort	0.103	0.055	0.135	0.124	0.540	0.233
	F7	Ort	0.001	0.001	0.001	0.001	0.001	0.001
Çok Modlu	**F8**	Ort	-1.179e+04	-1.248e+04	-1.181e+04	-1.241e+04	-1.175e+04	-1.130e+04
	F9	Ort	8.994e-05	5.129e-05	8.550e-05	2.454e-04	4.184e-04	0.001
	F10	Ort	0.001	0.001	0.01	0.026	0.139	0.253
	F11	Ort	2.727e-04	1.398e-05	4.531e-05	6.985e-05	0.002	0.037
	F12	Ort	0.002	0.001	0.001	0.001	0.002	0.001
	F13	Ort	0.012	0.010	0.006	0.001	0.009	0.005
Sabit boyutlu çok modlu	**F14**	Ort	1.031	0.998	0.998	0.998	0.998	0.998
	F15	Ort	3.994e-04	3.659e-04	4.311e-04	4.744e-04	4.510e-04	4.740e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.397	0.397	0.397	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.246	-3.266	-3.250	-3.243	-3.248	-3.232
	F21	Ort	-10.151	-10.153	-9.400	-10.151	-10.007	-10.146
	F22	Ort	-9.335	-9.954	-9.402	-10.398	-10.399	-10.393
	F23	Ort	-10.087	-10.089	-10.535	-10.532	-9.862	-10.532
No. best			2	9	0	3	1	1

Çizelge 3.5. GSBA algoritması parametre analizi (3/6)

			[-2 0]	[-2 1]	[-2 2]	[-2 3]	[-2 4]	[-2 5]
Tek Modlu	**F1**	Ort	5.471e-06	4.513e-06	7.136e-05	1.545e-05	3.533e-04	0.011
	F2	Ort	1.954e-10	2.197e-10	3.342e-10	2.952e-10	1.844e-10	8.586e-11
	F3	Ort	0.065	0.015	0.066	0.019	0.014	0.068
	F4	Ort	0.004	0.004	0.005	0.006	0.042	0.011
	F5	Ort	0.857	2.310	1.530	1.506	1.438	1.220
	F6	Ort	0.072	0.092	0.181	0.156	0.089	0.101
	F7	Ort	0.002	0.001	0.001	0.003	0.003	0.006
Çok Modlu	**F8**	Ort	-1.226e+04	-1.206+04	-1.202e+04	-1.206e+04	-1.208e+04	-1.218e+04
	F9	Ort	7.216e-05	4.791e-04	1.641e-04	8.656e-05	0.694	0.003
	F10	Ort	0.004	0.004	0.005	0.004	0.069	0.161
	F11	Ort	1.374e-04	1.966e-05	5.462e-05	6.215e-05	0.001	0.007
	F12	Ort	0.003	0.004	0.003	0.004	0.004	0.002
	F13	Ort	0.018	0.030	0.022	0.019	0.026	0.014
Sabit boyutlu çok modlu	**F14**	Ort	0.998	0.998	0.998	0.998	0.998	0.998
	F15	Ort	4.733e-04	4.702-04	3.856e-04	4.689e-04	5.337e-04	7.538e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.398	0.398	0.398	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.242	-3.202	-3.250	-3.250	-3.237	-3.248
	F21	Ort	-10.153	-10.152	-10.129	-10.151	-10.151	-10.147
	F22	Ort	-9.067	-10.401	-9.734	-9.734	-9.830	-9.727
	F23	Ort	-10.536	-9.865	-9.865	-10.534	-10.533	-9.856
No. best			0	0	0	0	0	0

Çizelge 3.6. GSBA algoritması parametre analizi (4/6)

			[-3 0]	[-3 1]	[-3 2]	[-3 3]	[-3 4]	[-3 5]
Tek Modlu	**F1**	Ort	3.169e-04	2.080e-04	1.255e-04	1.630e-04	6.383e-05	0.023
	F2	Ort	7.031e-10	4.445e-10	6.060e-10	2.395e-10	1.732e-10	2.727e-10
	F3	Ort	0.137	0.072	0.030	0.091	0.021	0.105
	F4	Ort	0.004	0.015	0.004	0.008	0.018	0.032
	F5	Ort	2.377	1.022	1.178	1.542	0.599	1.432
	F6	Ort	0.090	0.077	0.080	0.122	0.122	0.164
	F7	Ort	0.003	0.003	0.003	0.005	0.008	0.005
Çok Modlu	**F8**	Ort	-1.310e+04	-1.340e+04	-1.494e+04	-1.344e+04	-1.358e+04	-1.256e+04
	F9	Ort	0.225	0.005	0.704	0.591	0.097	0.074
	F10	Ort	0.010	0.004	0.003	0.015	0.014	0.092
	F11	Ort	8.780e-04	8.294e-05	4.831e-04	5.684e-05	8.778e-04	0.002
	F12	Ort	0.005	0.004	0.006	0.006	0.004	0.001
	F13	Ort	0.023	0.034	0.022	0.057	0.015	0.023
Sabit boyutlu çok modlu	**F14**	Ort	0.998	0.998	0.998	0.998	0.998	0.998
	F15	Ort	4.990e-04	5.414e-04	4.336e-04	4.429e-04	4.615e-04	4.842e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.398	0.398	0.398	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.280	-3.202	-3.201	-3.203	-3.211	-3.241
	F21	Ort	-10.153	-9.400	-10.152	-9.392	-10.151	-10.145
	F22	Ort	-10.402	-9.734	-9.734	-9.734	-9.734	-10.388
	F23	Ort	-9.999	-10.536	-10.536	-10.534	-10.535	-10.429
No. best			2	0	0	0	0	0

Çizelge 3.7. GSBA algoritması parametre analizi (5/6)

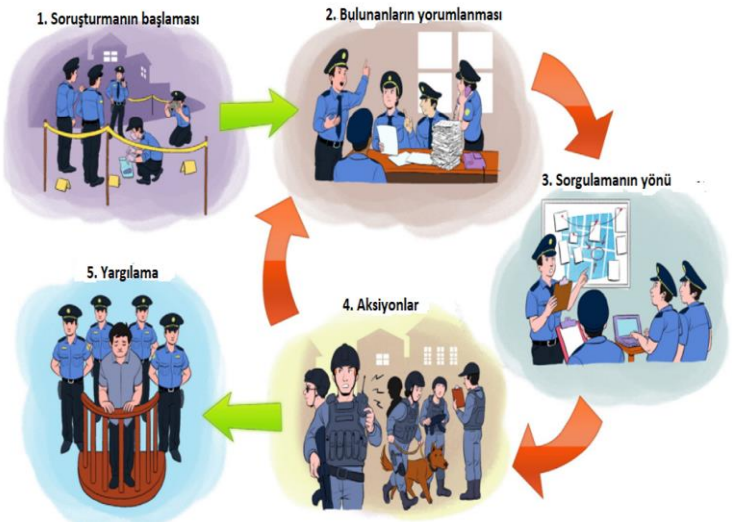
			[-4 0]	[-4 1]	[-4 2]	[-4 3]	[-4 4]	[-4 5]
Tek Modlu	**F1**	Ort	0.007	1.020e-04	8.813e-04	3.833e-04	0.016	5.637e-04
	F2	Ort	5.678e-10	4.369e-10	5.366e-10	5.074e-10	3.581e-10	3.477e-10
	F3	Ort	0.368	0.012	0.187	0.159	0.388	0.081
	F4	Ort	0.011	0.008	0.008	0.009	0.011	0.022
	F5	Ort	2.806	2.181	3.030	3.434	2.198	3.315
	F6	Ort	0.097	0.093	0.092	0.104	0.172	0.519
	F7	Ort	0.009	0.007	0.007	0.008	0.010	0.010
Çok Modlu	**F8**	Ort	-1.646e+04	-1.356e+04	-1.410e+04	-1.279e+04	-1.377e+04	-1.541e+04
	F9	Ort	6.394	1.913	1.633	3.577	0.424	0.983
	F10	Ort	0.029	0.011	0.022	0.013	0.044	0.040
	F11	Ort	4.005e-04	1.158e-04	1.353e-04	0.001	0.001	0.004
	F12	Ort	0.003	0.004	0.006	0.007	0.006	0.007
	F13	Ort	0.058	0.051	0.111	0.060	0.033	0.040
Sabit boyutlu çok modlu	**F14**	Ort	0.998	0.998	0.998	0.998	0.998	0.998
	F15	Ort	4.231e-04	4.487e-04	5.147e-04	4.814e-04	7.070e-04	5.819e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.398	0.398	0.398	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.230	-3.238	-3.250	-3.213	-3.216	-3.249
	F21	Ort	-10.152	-10.152	-10.151	-10.152	-10.150	-10.147
	F22	Ort	-9.735	-9.734	-10.402	-9.734	-10.402	-8.966
	F23	Ort	-10.536	-10.536	-9.195	-9.865	-10.110	-10.534
No. best			1	0	0	0	0	0

Çizelge 3.8. GSBA algoritması parametre analizi (6/6)

			[-5 0]	[-5 1]	[-5 2]	[-5 3]	[-5 4]	[-5 5]
Tek Modlu	**F1**	Ort	2.509e-04	8.810e-04	4.882e-04	9.029e-04	0.005	0.001
	F2	Ort	5.356e-10	7.148e-10	5.318e-10	5.550e-10	1.642e-10	1.566e-10
	F3	Ort	0.299	0.279	0.388	0.327	0.505	0.109
	F4	Ort	0.014	0.017	0.022	0.027	0.047	0.028
	F5	Ort	1.832	5.328	5.630	1.890	7.322	4.107
	F6	Ort	0.080	0.087	0.078	0.085	0.115	0.114
	F7	Ort	0.008	0.008	0.009	0.012	0.012	0.009
Çok Modlu	**F8**	Ort	-1.419e+04	-1.558e+04	-1.557e+04	-1.427e+04	-1.379e+04	-1.479e+04
	F9	Ort	3.214	5.073	4.487	0.440	4.068	2.037
	F10	Ort	0.075	0.019	0.008	0.009	0.101	0.111
	F11	Ort	6.977e-05	0.009	1.170e-04	3.495e-04	0.001	0.001
	F12	Ort	0.006	0.005	0.005	0.005	0.007	0.005
	F13	Ort	0.081	0.064	0.048	0.096	0.099	0.037
Sabit boyutlu çok modlu	**F14**	Ort	0.998	0.998	0.998	0.998	0.998	0.998
	F15	Ort	5.602e-04	6.196e-04	6.386e-04	6.771e-04	7.218e-04	7.514e-04
	F16	Ort	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	F17	Ort	0.398	0.398	0.398	0.398	0.398	0.398
	F18	Ort	3.000	3.000	3.000	3.000	3.000	3.000
	F19	Ort	-3.862	-3.862	-3.862	-3.862	-3.862	-3.862
	F20	Ort	-3.213	-3.233	-3.255	-3.250	-3.214	-3.220
	F21	Ort	-10.149	-10.147	-10.146	-10.141	-10.141	-10.140
	F22	Ort	-10.302	-9.734	-8.970	-10.396	-9.734	-10.390
	F23	Ort	-10.430	-10.531	-10.532	-9.865	-10.509	-10.522
No. best			0	0	0	0	0	0

3.2. Adli Tıp Temelli Soruşturma Algoritması (FBI)

Bu alt bölümde, adli tıp temelli soruşturma algoritması (FBI) tanımlanacak (Chou ve Nguyen, 2020), daha sonra algoritmaya önerilen yenilikler detaylı bir şekilde açıklanacaktır. Adli yarılama süreci Şekil 3.7'de ifade edilmektedir.



Şekil 3.7. Adli yargılama işlemleri (Chou ve Nguyen, 2020)

3.2.1. Adım A1

Bu adım, ekibin bilgileri analiz ettiği ve başlangıçta olası şüpheli konumunu belirlediği "bulguların yorumlanması" olarak adlandırılmaktadır. Bu adımda, X_{A_i} yeni şüpheli konum, diğer şüpheli konumlardan gelen bilgilere bağlı olmaktadır. Her bir bireyin yönü Denklem (3.7)'ye göre elde edilmektedir.

$$X_{A1_{ij}} = X_{A1_{ij}} + \left(2(rand_1 - 0.5)\right)(X_{A_{ij}} - (X_{A_{kj}} + X_{A_{hj}})/2 \qquad (3.7)$$

Burada D problemin boyutu ($j = 1,2,\dots,D$), $\left(2(rand_1 - 0.5)\right)$ [-1,1] aralığında bir rastsal sayı, $rand_1$ rastsal bir sayı, $k, h\ ve\ i$, 3 şüpheli şüpheli konumu ($\{k, h, i\} \in \{1, 2, \dots, NP\}$), NP popülasyon sayısını belirtmektedir. Burada $k\ ve\ h$ ise rastsal olarak seçilmektedir.

3.2.2. Adım A2

Bu adıma "soruşturma yönü" denmektedir. Dedektifler, en muhtemel şüpheli yeri bulmak için, her şüpheli yerin olasılığını diğer konumlarla karşılaştırırlar. Bir minimizasyon probleminde, P_{worst} en kötü uygunluk değeri (en düşük olasılık), P_{best} en iyi uygunluk değeri (en yüksek olasılık) ve X_{best} de en iyi konum olmaktadır. Her bir konumun olasılığı $Prob(X_{A_i})$ Denklem (3.8)'deki gibi hesaplanmaktadır.

$$Prob(X_{A1_i}) = (P_{worst} - P_{A_i})/(P_{worst} - P_{best}) \tag{3.8}$$

X_{A_i}'nın yönü en iyi birey ve diğer bireyler tarafından yönlendirilmektedir. Adım A1'e benzer olarak yeni şüpheli konum $X_{A2_{ij}}$ Denklem (3.9) vasıtasıyla bulunur ve uygunluk değeri (olasılık) hesaplanır.

$$X_{A2_{ij}} = X_{best} + X_{A_{dj}} + rand_5(X_{A_{ej}} + X_{A_{fj}}) \tag{3.9}$$

Burada X_{best} Adım A1'de güncellenmiş en iyi konum, $rand_5$ [0,1] arasındaki rastsal sayı, d, e, f ve i, dört tane şüpheli konumdur. Burada $\{d, e, f, i\} \in \{1, 2, \ldots, NP\}$ ve d, e ve f rastsal seçilmektedir.

3.2.3. Adım B1

Bu adım "aksiyon" olarak adlandırılmaktadır. Bu adımda her bir birey B_i, en düşük uygunluk değerine sahip en iyi konuma (en iyi olasılık), aşağıdaki denklem vasıtasıyla ulaşmayı amaçlamaktadır. Eğer bu birey, P_{B_i}'dekinden daha iyi bir uygunluk değerine sahip olursa, yeni bulunan konum güncellenmektedir.

$$X_{B1_{ij}} = rand_6 X_{B_{ij}} + rand_7(X_{best} - X_{B_{ij}}) \tag{3.10}$$

Burada X_{best} dedektifin (bireyin) bulduğu en iyi çözüm, $rand_6$ ve $rand_7$ [0, 1] aralığında üretilen rastsal sayılar olup $j = 1, 2, \ldots, D$ dir.

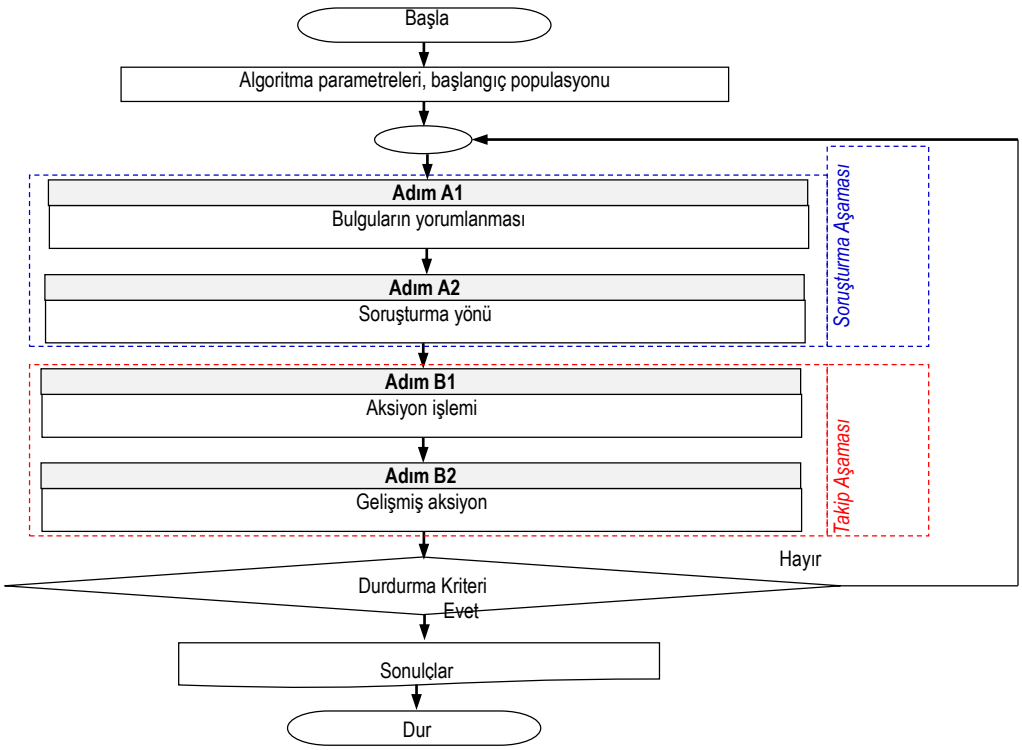
3.2.4. Adım B2

Bu adım "aksiyon" adımının genişletilmiş versiyonudur. Adım B2'de, her bir B_i bireyi yeni arama konumunu belirlemek amacıyla, diğer bireyler ile sıkı bir ilişkiye sahip olmaktadır. Birey B_i, en iyi konuma hareket etmekte ve takımın diğer üyelerinin etkisi altında kalmaktadır. Eğer P_{B_r} (B_r bireyinin olasılığı), P_{B_i}'den daha iyi ise, B_i bireyinin yeni konumu Denklem (3.11)'de, Denklem 3.12'de hesaplanmaktadır.

$$X_{B2_{ij}} = X_{B_{rj}} + rand_8\left(X_{B_{rj}} - X_{B_{ij}}\right) + rand_9\left(X_{best} - X_{B_{rj}}\right) \quad (3.11)$$

$$X_{B2_{ij}} = X_{B_{ij}} + rand_{10}\left(X_{B_{ij}} - X_{B_{rj}}\right) + rand_{11}\left(X_{best} - X_{B_{ij}}\right) \quad (3.12)$$

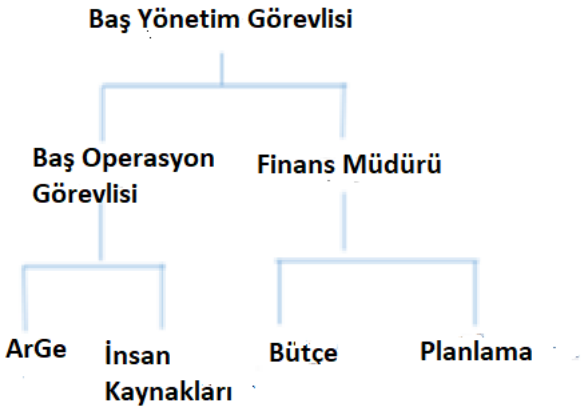
Yukarıdaki denklemde, X_{best}, en iyi çözümü ifade etmektedir. $rand_8, rand_9, rand_{10}\ ve\ rand_{11}$ $[0, 1]$ aralığında rastsal sayılar, r ile i, iki polis konumu olup, $\{r, i\} \in \{1, 2, \ldots, NP\}$ dir. Burada r rastsal olarak seçilen bir sayıyı ifade etmektedir $(j = 1, 2, \ldots, D)$. Algoritma ön tanımlı bir durdurma kriteriyle karşılaşana kadar çözüm arama sürecini sürdürmekte, karşılaştığında bulduğu en iyi çözümü sunmaktadır. Algoritmanın akış diyagramı Şekil 3.8'de sunulmaktadır.



Şekil 3.8. FBI algoritmasının akış diyagramı

3.3. Hiyararşik Yığın Tabanlı Optimizasyon Algoritması (HBO)

Bu alt başlıkta, yığın tabanlı optimizasyon algoritmasının (HBO) (Askari, Saeed ve Younas, 2020) adımları anlatılacaktır. Çalışanlar arasındaki hiyerarşinin temsili gösterimi Şekil 3.9'da verilmektedir.



Şekil 3.9. Çalışanlar hiyerarşisi

3.3.1. Başlatma

HBO algoritması diğer metasezgisel algoritmalara benzer şekilde aşağıdaki gibi başlatılmaktadır.

$$x_i = lb_i + rand(ub_i - lb_i),\ \ i = 1,2,\dots,N \tag{3.13}$$

Burada x_i N bireyli bir popülasyonun i. bireyini olmaktadır. $ub_i\ ve\ lb_i$ alt ve üst sınırları ifade etmektedir.

3.3.2. Anında Patronla Etkileşim Modeli

Her bir ebeveyn düğümün, çocukları (çalışan) için anında bir patron olduğunu varsayalım. Her bir arama bireyinin (x_i) pozisyonunun güncellenmesi, ebevyn düğüm B referans alınarak Denklem (3.14)'teki gibi modellenebilmektedir.

$$x_i^k(t+1) = B^k + \gamma\lambda^k\left|B^k - x_i^k(t)\right| \tag{3.14}$$

Burada t o anki iterasyonu, x vektörün k. elementi iken, δ^k rastsal olarak üretilen bir sayı olup Denklem (3.15)'teki gibi tanımlanmaktadır.

$$\lambda^k = 2r - 1 \tag{3.15}$$

Yukarıdaki denklemde, r $[0, 1]$ aralığında bir rastsal sayıdır. γ parametresi ise Denklem (3.16)'daki gibi ifade edilmektedir.

$$\gamma = \left| 2 - \frac{(t mod (\frac{T}{C}))}{(\frac{T}{4C})} \right| \tag{3.16}$$

Burada T toplam iterasyon sayısını ve C algoritmanın parametresini ifade etmektedir.

3.3.3. Meslektaşlar Arasında Etkileşim Modeli

Her bir meslektaşın yığında aynı seviyede olduğunu kabul ederek, $\overrightarrow{x_i}$ rastsal olarak seçilen meslektaş $\overrightarrow{S_r}$ 'ye göre Denklem (3.17)'deki gibi güncellenmektedir.

$$x_i^k(t+1) = \begin{cases} S_r^k + \gamma\delta^k \left| S_r^k - x_i^k(t) \right| & \text{f}(\overrightarrow{S_r}) < f(\overrightarrow{x_i}(t)) \\ x_i^k + \gamma\delta^k \left| S_r^k - x_i^k(t) \right| & \text{f}(\overrightarrow{S_r}) \geq f(\overrightarrow{x_i}(t)) \end{cases} \tag{3.17}$$

Burada f hedef fonksiyonunu ifade etmektedir.

3.3.4. Çalışanın Kendi Katkısının Modeli

Bu fazda çalışanın kendi katkısı gerçekleşmektedir. Bu davranış, sonraki yinelemede çalışanların önceki konumunu koruyarak Denklem (3.18)'deki gibi modellenmektedir.

$$x_i^k(t+1) = x_i^k(t) \tag{3.18}$$

3.3.5. Bir Araya Getirme

Algoritmada, rulet tekerleği mekanizması, olasılıkları p_1, p_2 ve p_3 olarak üçe bölmek için kullanılmaktadır. Bireyin güncellenmesi için, p_1 seçilim olasılığı Denklem (3.19)'daki gibi hesaplanmaktadır.

$$p_1 = 1 - \frac{t}{T} \tag{3.19}$$

Burada T maksimum iterasyon sayısını ifade etmektedir. p_2 ve p_3'ün olasılıkları Denklem (3.20) ve Denklem (3.21)'de verilmektedir.

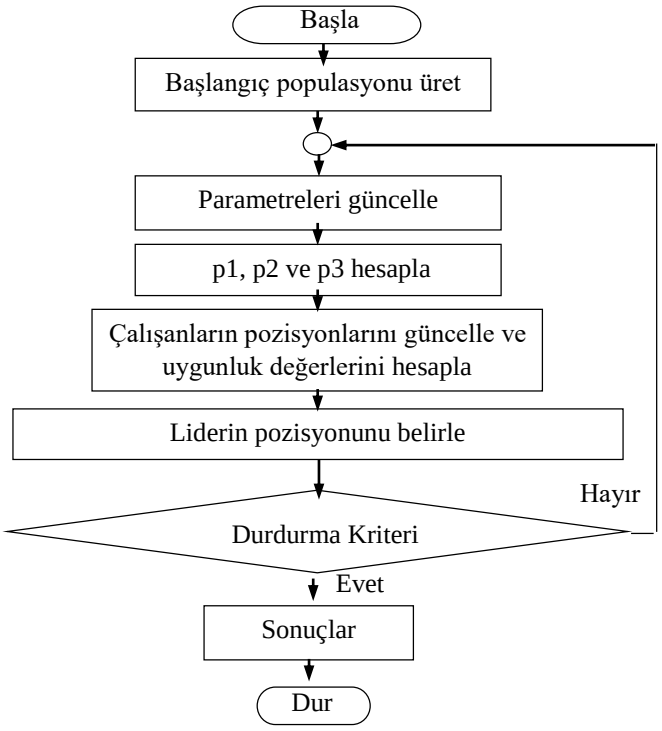
$$p_2 = p_1 - \frac{1-p_1}{2} \tag{3.20}$$

$$p_3 = p_2 - \frac{1-p_1}{2} \tag{3.21}$$

HBO algoritmasının genel konum güncellemeleri Denklem (3.22)'ye bağlı olarak gerçekleşmektedir.

$$x_i^k(t+1) = \begin{cases} x_i^k(k) & p \le p_1 \\ B^k + \gamma\delta^k\left|B^k - x_i^k(t)\right| & p > p_1 \text{ ve } p \le p_2 \\ S_r^k + \gamma\delta^k\left|S_r^k - x_i^k(t)\right| & p > p_2 \text{ ve } p \le p_3 \text{ ve } f(\overrightarrow{S_r}) < f(\overrightarrow{x_i}(t)) \\ x_i^k(t+k) = B^k + \gamma\delta^k\left|B^k - x_i^k(t)\right| & p > p_2 \text{ ve } p \le p_3 \text{ ve } f(\overrightarrow{S_r}) \ge f(\overrightarrow{x_i}(t)) \end{cases} \tag{3.22}$$

HBO algoritmasının akış şeması Şekil 3.2'de verilmektedir.



Şekil 3.10. HBO algoritmasının akış diyagramı

3.4. HBO ve FBI Algoritmalarına Önerilen Geliştirmeler

Karşıtlık temelli öğrenme (OBL), arama uzayındaki farklı bölgelerdeki çözümleri araştırmaktadır (Tizhoosh, 2005; Mahdavi, Rahnamayan, ve Deb, 2018). Bu yöntem, varolan çözümlerden karşıt aday çözümler elde etmeyi sağlamaktadır. Karşıt adayların popülasyona dahil edilmesi, arama sürecinde daha fazla bölgenin ziyaret edilmesi ile arama sürecinde çözümlerin daha iyi yayılmasına izin verebilmektedir. Asıl amaç, büyük boyutlu ve karmaşık problemlerde daha kaliteli çözümlere ulaşmayı sağlamaktır. $\boldsymbol{X} = (x_1\, x_2\, , \ldots , x_D)$ D boyutlu arama uzayında bir vektör ve $x_i \in R$ ise X'in karşıt vektörü $X_o = (x_{o1},\ x_{02}\, , \ldots , x_{oD})$, Denklem (3.23)'te ifade edilmektedir.

$$\begin{cases} x'_{oi} = lb_{oi} + rand(ub_{oi} - lb_{oi}) \\ x_{oi} = ub_{oi} + lb_{oi} - x'_{oi} \end{cases} \tag{3.23}$$

Burada $rand$, $[0, 1]$ aralığında bir rastsal sayı, ub_{oi} x_{oi}'nin üst sınırı ve lb_{oi} de x_{oi}'nin alt sınırı olmaktadır. OBL'nin FBI algoritmasına entegrasyon adımları aşağıdaki gibi verilmektedir.

Adım 1. Denklem (3.7) ve (3.9) vasıtasıyla X_{A1} ve X_{A2} üret.
Adım 2. Karşıtlık bireylerini, X_{A1_i} ve X_{A2_i}için X_{Ao1_i} , X_{A02_i} olarak Denklem (3.23) vasıtasıyla üret.
Adım 3. X_{A1_i}, X_{A2_i}, X_{Ao1_i} ve X_{Ao2_i} uygunluk değerlerini hesapla.
Adım4. Eğer X_{Ao1_i} bireyi X_{A1_i} bireyinden daha iyi ise, X_{A1_i} ile X_{Ao1_i}'i yer değiştir; diğer durumda X_{A1_i} ile devam et.
Adım 5. Eğer X_{Ao2_i} bireyi X_{A2_i} bireyinden daha iyi ise, X_{A2_i} ile X_{Ao2_i} yer değiştir, diğer durumda X_{A2_i} ile devam et.

OBL'nin HBO algoritmasına entegrasyon adımları aşağıdaki gibi verilmektedir.

Adım 1. Denklem (3.13)' e göre x_i başlangıç bireyini üret.
Adım 2. Denklem (3.23) vasıtasıyla x_i'nin karşıt bireyi, x_{oi}'yi üret.
Adım 3. x_i and x_{oi} 'nin uygunluk değerlerini hesapla.
Adım 4. Eğer x_{oi}, x_i'den iyi ise x_i'i x_{oi} ile yer değiştir, diğer durumda x_i ile devam et.
Adım 5. Yukarıdaki adımları başlangıç popülasyonun her bir bireyine uygula.

Önerilen değişiklikte kullanılan diğer yöntem, Cauchy tabanlı mutasyondur (CBM). Cauchy olasılık dağılımına dayanan, Cauchy mutasyon operatörü ilk olarak hızlı evrimsel programlama için tanıtılmıştır (Xin, Yong ve Guangming, L. 1999). Gauss olasılık dağılımı ile karşılaştırıldığında, Cauchy olasılık dağılımı daha uzun iki kuyruklu bir dağılıma sahiptir. Dolayısıyla, Cauchy mutasyon operatörü tarafından oluşturulan popülasyonun, ebeveynlerinden önemli ölçüde farklı olduğu anlamına gelmektedir (Yang ve Huang, 2011). Başka bir deyişle bunun anlamı, rastgele değerin sınırı genişlemekte ve daha sonra bu değişiklik, çözümlerin yerel minimumdan kaçması için daha fazla şans verebilmektedir. Bir boyutlu Cauchy yoğunluk fonksiyonu, Denklem (3.24) ile verilmektedir (Paiva, Silva, Leite, Marcone, ve Costa, 2017).

$$m(x) = \frac{1}{\pi}\frac{k}{k^2+x^2} \tag{3.24}$$

Burada $x \in [-\infty, +\infty]$ ve skalalama faktörü $k > 0$ dır. Cauchy dağılım fonksiyonu Denklem (3.25) ile ifade edilmektedir. Bu çalışmada $k = 1$ alınmaktadır.

$$m_i(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x}{k}) \tag{3.25}$$

CBM yöntemi, FBI algoritmasında, X_{A1} bireylerinden, X_{B1} ve X_{B2} bireylerini oluşturmada, Denklem (3.26) ve Denklem (3.27) aracılığıyla kullanılmaktadır.

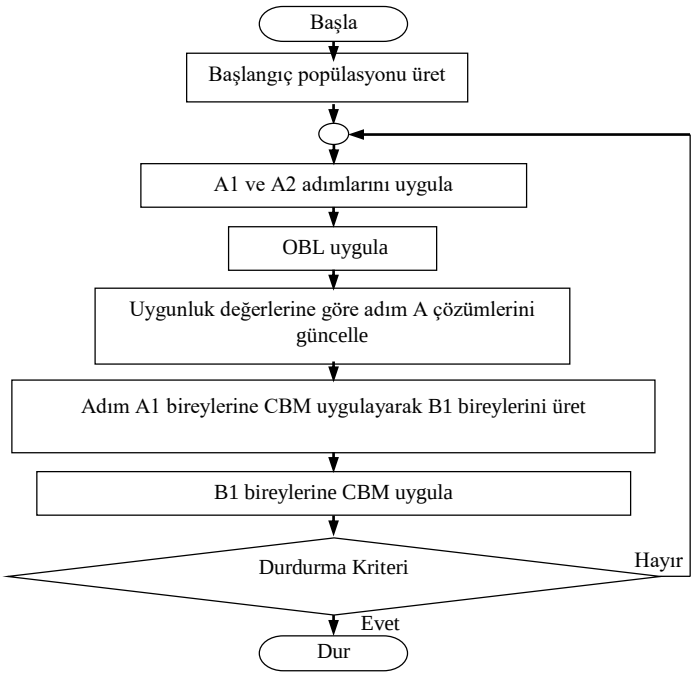
$$X_{B1_{ij}} = X_{best} + X_{A1_{ij}} Cauchy(0,1) \tag{3.26}$$

$$X_{B2_{ij}} = X_{best} + X_{B1_{ij}} Cauchy(0,1) \tag{3.27}$$

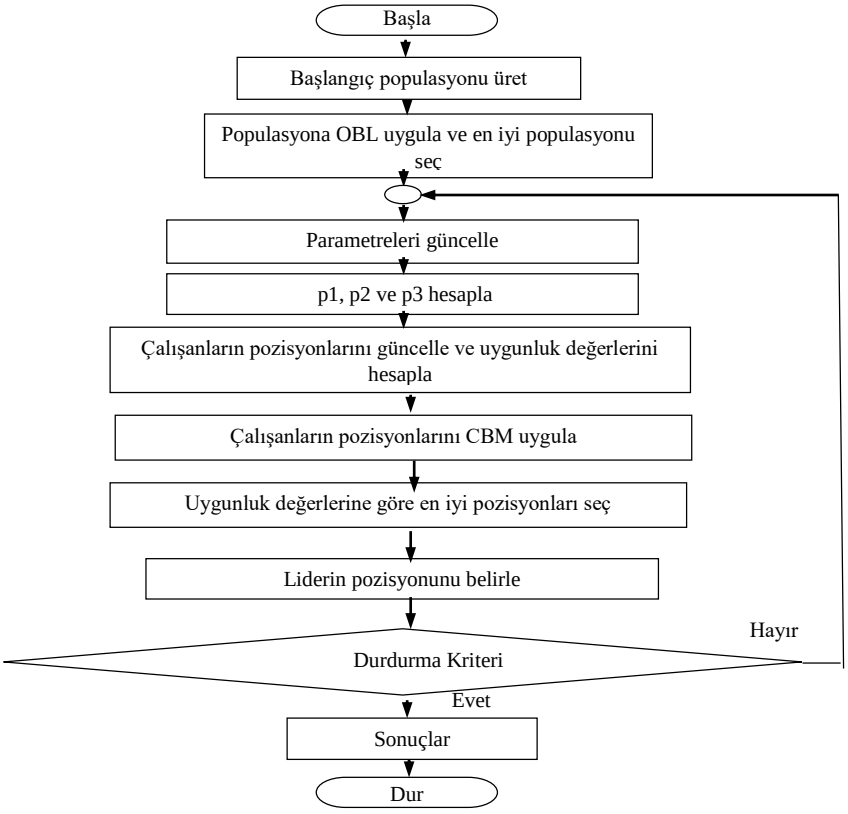
HBO algoritmasında, CBM yöntemi aracılığıyla, her bir birey x_i için, x_{ci} yeni bireyi Denklem (4.6)'da gibi oluşturulmaktadır.

$$x_{ci}(\mathrm{t}) = X_{best} + x_i(\mathrm{t}) Cauchy(0,1) \tag{4.6}$$

Denklem (4.6)'da X_{best}, o ana kadarki bulunan en iyi çözüm olmaktadır. HBO algoritmasının popülasyonuna katılacak yeni birey, x_i ve x_{ci} bireylerinden en iyisi olarak seçilmektedir. Yukarıdaki işlemlere göre, önerilen modifiye edilmiş HBO (HBO-CO) ve modifiye FBI (modFBI) algoritmasının akış diyagramları, Şekil (3.11) ve Şekil (3.12)'de verilmektedir.



Şekil 3.11. modFBI algoritmasının akış diyagramı



Şekil 3.12. HBO-CO algoritmasının akış diyagramı

4. UYGULAMALAR

4.1. Sayısal Test Sonuçları

Geliştirilen algoritmalar (GSBA, modFBI, HBO-CO), 5 klasik algoritma olan GA (Goldberg, 1989), PSO (Storn and Price, 1997), DE (Storn ve Price, 1997), CS (Yang and Deb, 2009), HS (Geem, Kim ve Loganathan, 2001; Lee ve Geem, 2004) ve 7 güncel algoritma olan BWO (Hayyolalam ve Kazem, 2020), SSA (Mirjalili, Gandomi, Mirjalili, Saremi, Faris, ve Mirjalili, 2017), MVO (Mirjalili, Mirjalili, ve Hatamlou, 2016), HHO (Heidari, Mirjalili, Faris, Aljarah, Mafarja ve Chen, 2019), ChOA (Khishe ve Mosavi, 2020), FBI (Chou ve Nguyen, 2020), HBO (Askari, Saeed ve Younas, 2020) ile karşılaştırılmıştır. Eşit kıyaslama için, tüm algoritmaların popülasyon sayısı 80, maksimum iterasyon sayısı 300 ve maksimum fonksiyon hesaplama sayısı 50000 olarak belirlenmiştir. Algoritmalar literatürde sıklıkla kullanılan tek modlu ve çok modlu 23 fonksiyon (Saremi, Mirjalili ve Lewis, 2017; Mirjalili ve Lewis, 2016; Mirjalili, 2015a; 2015b) üzerinde test edilmiştir (Ekte özellikleri verilmektedir). Yüksek boyuttaki problemler üzerinde performanslarını test etmek için, 20 tane 100D CEC (Wu, Mallipeddi, ve Suganthan, 2016) problemleri kullanılmıştır. Algoritmaların performanslarının istatistiki analizi için, Wilcoxon rank-sum (Nikolic-Dorić, Čobanović ve Lozanov-Crvenković, 2006) ve Friedman (Derrac, García, Molina ve Herrera, 2011) testlerinden %5 güven seviyesinde yararlanılmıştır. Wilcoxon rank-sum testi, referans algoritmamız ile (GSBA), diğer algoritmaları her bir fonksiyon için ikili kıyaslayan bir testtir. İlgili tabloların son sütunlarında bulunan "+/=/-", GSBA'nın kıyaslanan algoritmadan kaç fonkiyonda iyi (+), kaç fonsiyonda eşit (=) ve kaç fonksiyonda kötü (-) performans gösterdiğini Wilcoxon rank-sum testine göre belirtmektedir. Fridman testi ise tüm algoritmaların çoklu karşılaştırılmasında kullanılmaktadır. Bu test sonucundan tüm algoritmalara bir rank değeri atanmış olup, rankı küçük olan algoritma, Friedman testine göre daha iyi performans göstermektedir. Buna ek olarak, her bir fonksiyon için verilen Whisker-box grafikleri algoritmaların bulduğu hata değerlerinin yayılımını göstermektedir. İlgili tablolarda bulunan sonuçların ortalamaları (Ort) ve standart sapma (Std) değerleri verilmektedir. Karşılaştırılan algoritmaların parametreleri Çizelge 4.1'de verilmektedir.

Çizelge 4.1. Karşılaştırılan algoritmaların parametreleri

GA	HHO	PSO	ChOA	MVO	FBI
Mutasyon oranı: 0.2 Çaprazlama oranı: 0.5	Adaptif parametreler	Bilişsel sabit: 1 Sosyal sabit: 1 Yerel sabit: 0.3	Dinamik katsayı vektörü Kaos haritası temelli kaotik vektör	Solucan deliği varlığı olasılığı max.: 1 Solucan deliği varlığı olasılığı min.: 0.2	Parametresiz
DE	**SSA**	**BWO**	**CS**	**HS**	**HBO**
Çaprazlama sabiti: 0.3 Skalalama faktörü: 0.9	Adaptif parametreler	Üreme oranı: 0.6 Yamyamlık oranı: 0.44 Mutasyon oranı: 0.4	Yabancı yumurtaların keşif oranı: 0.25	Harmoni bellek oranı: 0.9 Perde ayarlama hızı max.: 0.9 Perde ayarlama hızı min.: 0.0 Bantgenişliği max.: 1 Bantgenişliği min.: 1e-6	Adaptif parametreler

Çizelge 4.2'de klasik algoritmaların, Çizelge 4.3 ve 4.4'de güncel gelişmiş algoritmaların sonuçlarına yer verilmiştir. Klasik algoritmaların sonuçlarından, GSBA algoritmasının Wilcoxon testine göre en çok sayıda doğrulukla sonuçlara (en az 16 fonksiyonda) ulaştığı görülmektedir. Diğer yandan, Friedman testine bağlı olarak, ortalama rank değeri kıyaslanan klasik algoritmalar arasında en az olmaktadır. Bu geliştirilen GSBA algoritmasının, 23 nümerik fonksiyon içeren deney setinde üstünlüğünü göstermektedir.

Çizelge 4.3 ve 4.4'ten yola çıkarak, GSBA algoritmasının, Wilcoxon testine göre kıyaslanan güncel algoritmalardan daha çok sayıda iyi fonksiyona ulaştığı görülmektedir. Buna ek olarak FBI ve HBO algoritmalarının modifiye edilmiş versiyonları olan HBO – CO ve modFBI algoritmaları da, GSBA hariç, kıyaslanan algoritmalardan üstün performans göstermektedir. HBO-CO algoritması, 9 fonksiyonda GSBA'dan üstün performans gösterirken, modFBI algoritması 11 fonksiyonda üstünlüğünü sürdürmektedir. Tabloların son sütunundaki Friedman testi sonuçları değerlendirildiğinde, modFBI algoritmasının GSBA algoritmasına yakın performans verdiği görülmektedir. Bu durum, verilen fonksiyon seti için modFBI algoritmasının GSBA'nın iyi bir alternatifi olduğunu göstermektedir. Çizelge 4.5, güncel algoritmaların 30 bağımsız çalıştırma üzerinden dağılımını göstermektedir. GSBA algoritmasının F8 ve F20 fonksiyonları hariç, dar bir aralıkta konumlandığı ve bu da geliştirilen algoritmanın farklı bağımsız çalıştırmalarda benzer performanslar gösterdiği anlamını taşımaktadır.

Çizelge 4.6'de klasik algoritmaların, Çizelge 4.7 ve 4.8'de güncel algoritmaların 20 fonksiyon üzerindeki sonuçlarına yer verilmiştir. GSBA algoritması, Wilcoxon testine göre, GA'da 1 fonksiyonda, PSO, ve DE'de 3 fonksiyonda daha kötü performans

sergilemiştir. Bu algoritma, Wilcoxon test sonuçlarına göre, diğer tüm fonksiyonlarda benzer veya daha iyi performans göstermiştir. Diğer yandan, karşılaştırılan klasik algoritmalar arasında, GSBA algoritması en küçük rank değerine sahip olmaktadır. Nümerik ve istatistiki sonuçlar, geliştirilen algoritmanın, klasik algoritmalara göre, birçok fonksiyon türü için daha iyi performans verdiğini göstermektedir.

Güncel algoritmaların kıyaslanmasında da, geliştirilen HBO-CO, modFBI ve GSBA algoritmalarının, diğer algoritmalardan birçok fonksiyonda daha iyi sonuçlar verdiği gözlemlenmektedir. Geliştirilen algoritmalar, 20 nümerik fonksiyon üzerinden, kendi içlerinde kıyaslandığında; daha önceki deney setine benzer şekilde, Friedman testinden yola çıkarak, GSBA algoritması, en iyi performansı göstermektedir ve en iyi ikinci algoritma HBO-CO olmaktadır. Çizelge 4.9'da güncel algoritmalar için hataların dağılımı verilmektedir. GSBA algoritması F14 ve F17 fonksiyonunda sapmalı bir dağılım gösterse bile, diğer fonksiyonlarda daha istikrarlı bir davranış sergilemektedir.

Çizelge 4.2. Klasik algoritmaların 23 fonksiyon için test sonuçları

			GA	PSO	DE	CS	HS	GSBA
Tek modlu	F1	Ort	9.106e+02 +	42.061 +	9.306e+04 +	2.805e+03 +	2.984e+03 +	6.378e-06
		Std	1.928e+02	22.830	1.168e+04	604.740	3.500e+02	1.119e-05
	F2	Ort	12.473 +	13.244 +	1.782e+16 +	7.426e+09 +	12.220 +	1.606e-11
		Std	1.819	4.797	4.647e+16	4.366e+09	1.139	1.046e-10
	F3	Ort	9.429e+03 +	7.457e+03 +	1.568e+05 +	3.495e+04 +	9.100e+04 +	4.567e-05
		Std	2.531e+03	2.432e+03	1.275e+04	5.195e+03	1.043e+04	6.183e-05
	F4	Ort	18.891 +	14.593 +	90.426 +	36.226 +	40.925 +	0.002
		Std	2.353	2.264	2.376	3.528	1.644	0.002
	F5	Ort	6.028e+04+	3.000e+03 +	4.008e+08 +	7.217e+05 +	1.349e+06 +	4.242
		Std	2.626e+04	2.024e+03	6.888e+07	2.625e+05	2.591e+05	12.059
	F6	Ort	8.376e+02 +	38.737 +	9.308e+04 +	2.826+03 +	2.955e+03 +	0.055
		Std	2.530e+02	23.221	1.010e+04	534.433	3.955e+02	0.058
	F7	Ort	0.357 +	0.186 +	2.847e+02 +	1.129 +	1.377 +	0.001
		Std	0.124	0.039	43.948	0.359	0.195	0.001
Çok modlu	F8	Ort	-1.214e+04 =	-8.491e+03 +	-7.250e+03 +	-4.163e+03 +	-1.194e+04 =	-1.248e+04
		Std	6.982e+02	6.781e+02	5.126e+02	2.628e+02	1.966e+02	1.493e+03
	F9	Ort	1.014e+02 +	3.165e+02 +	6.963e+02+	2.983e+02 +	52.814 +	5.129e-05
		Std	16.911	26.350	25.523	20.520	5.431	1.075e-04
	F10	Ort	6.147 +	3.718 +	19.958 +	16.612 +	9.213 +	0.001
		Std	0.647	0.460	0.001	1.114	0.449	0.002
	F11	Ort	8.736 +	1.344 +	8.387e+02 +	26.931 +	28.334 +	1.398e-05
		Std	1.556	0.157	94.565	5.508	3.882	2.099e-05
	F12	Ort	4.336 +	9.502 +	9.765e+08 +	2.051e+03+	6.943e+04 +	0.001
		Std	1.360	2.534	2.096e+08	2.664e+03	2.713e+04	0.001
	F13	Ort	3.210e+02 +	76.166 +	1.797e+09 +	3.774e+05 +	1.522e+06 +	0.010
		Std	5.135e+02	34.984	3.814e+09	1.903e+05	3.987e+05	0.016
Sabit boyutlu çok modlu	F14	Ort	0.998 =	0.998 =	0.998 +	0.998 +	0.998 +	0.998
		Std	2.620e-09	0.001	9.737e-11	4.504e-10	1.382e-10	9.313e-11
	F15	Ort	0.002 +	0.001 +	6.737e-04 =	5.549e-04 +	8.178e-04 +	3.659e-04
		Std	0.003	2.717e-04	4.562e-04	9.184e-05	9.309e-05	8.859e-05
	F16	Ort	-1.031 -	-1.031 -	-1.031 +	-1.031 +	-1.031 +	-1.031
		Std	1.045e-16	1.295e-15	6.387e-06	4.222e-06	2.188e-15	7.430e-11
	F17	Ort	0.397 +	0.397 -	0.397 -	0.397 +	0.397 -	0.397
		Std	3.710e-05	1.789e-13	0.000	1.305e-5	1.272e-14	3.342e-07
	F18	Ort	3.000 +	3.000 -	3.000 -	3.000 +	3.000 +	3.000
		Std	1.635e-04	2.059e-15	7.691e-16	3.984e-3	3.373e-3	4.963e-08

F19	Ort	-3.862 +	-3.862 +	-3.862 -	-3.862 +	-3.862 +	-3.862
	Std	2.210e-06	2.385e-5	2.668e-15	1.108e-6	6.284e-4	1.030e-06
F20	Ort	-3.235 +	-3.216 +	-3.211 +	-3.198 +	-3.203 +	-3.266
	Std	0.051	0.033	0.030	0.064	3.486e-07	0.060
F21	Ort	-6.326 +	-10.137 +	-8.769 +	-10.141 +	-5.200 +	-10.153
	Std	3.688	0.047	0.721	0.013	3.564	5.914e-05
F22	Ort	-8.970 +	-9.526 +	-9.021+	-10.396 -	-7.853 +	-9.954
	Std	2.922	1.994	0.756	0.006	3.366	1.693
F23	Ort	-8.572 +	-9.717 +	-9.188 +	-10.534 -	-10.376 -	-10.089
	Std	3.321	1.682	0.916	0.001	0.878	1.7002
+/=/-		20/2/1	19/1/3	19/1/3	21/0/2	20/1/2	
Ort. Rank		9.326	8.739	11.304	9.043	10.087	3.217

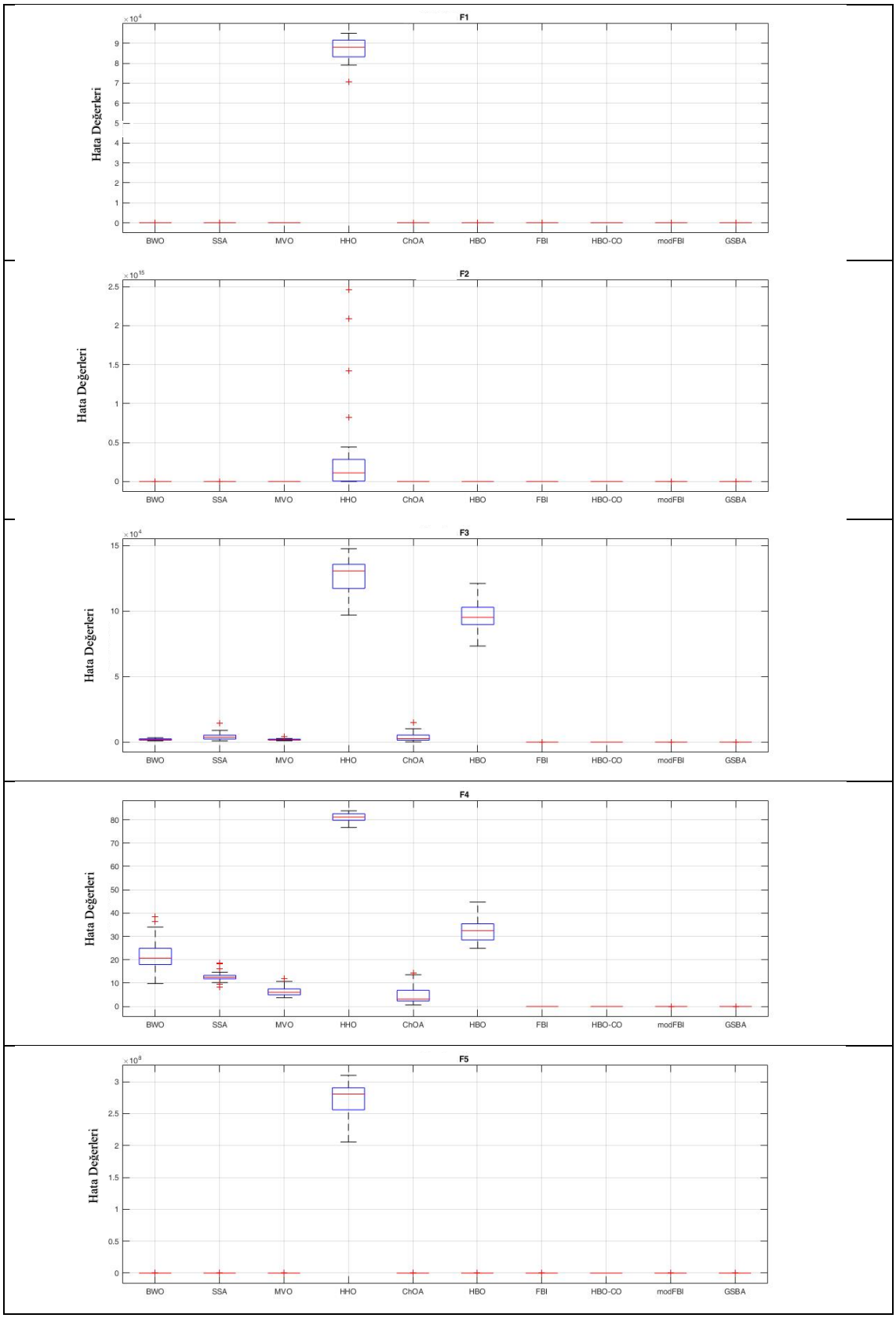
Çizelge 4.3. Güncel algoritmaların 23 fonksiyon üzerinden test sonuçları (1/2)

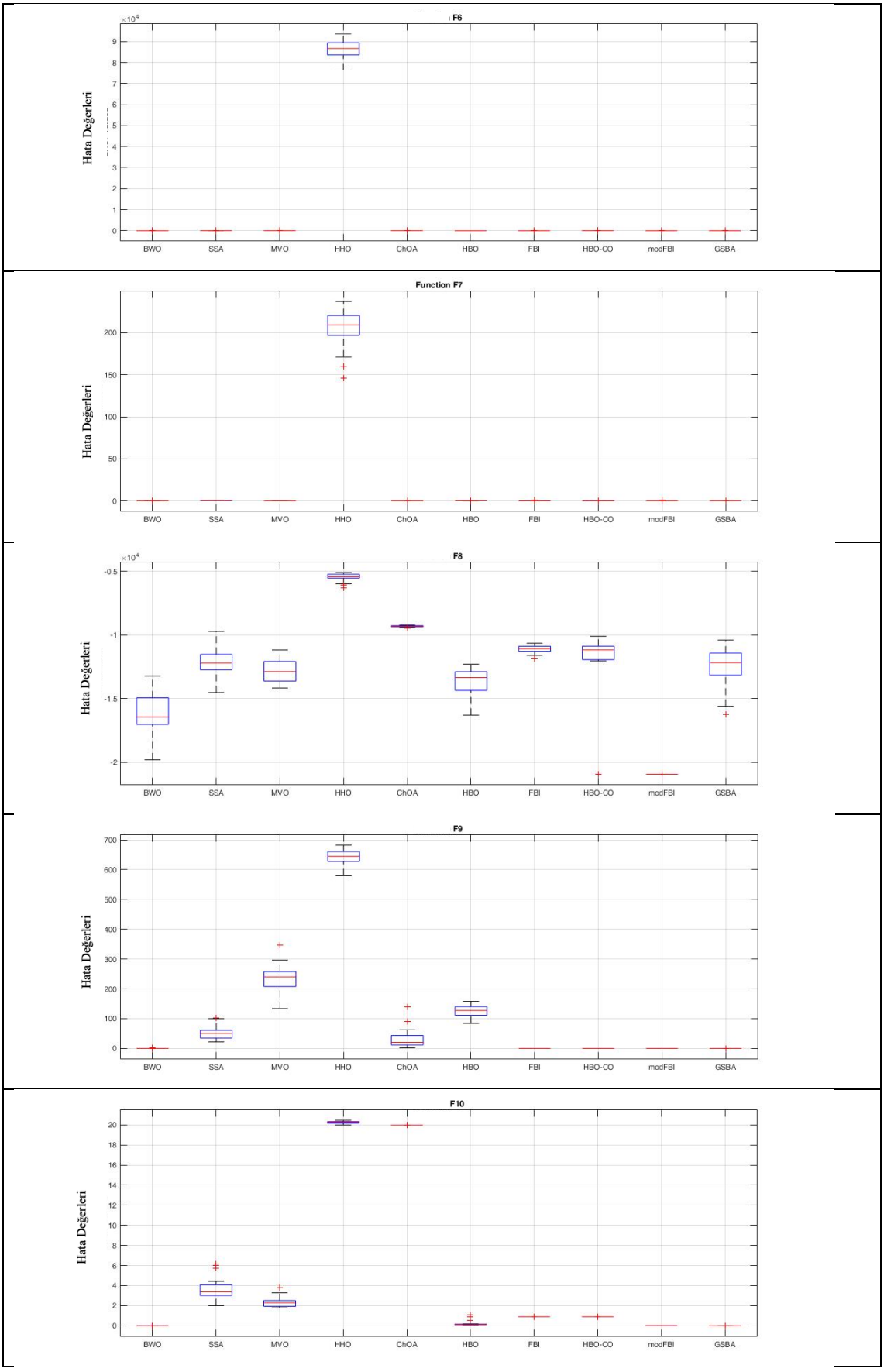
			BWO	**SSA**	**MVO**	**HHO**	**GSBA**
Tek modlu	**F1**	Ort	8.587e-04 +	1.844 +	3.407 +	8.725e+04 +	6.378e-06
		Std	0.002	1.413	0.597	5.362e+03	1.119e-05
	F2	Ort	0.001 +	5.971 +	61.567 +	3.344e+14 +	1.606e-11
		Std	0.001	2.792	70.877	6.07e+14	1.046e-10
	F3	Ort	1.949e+03+	4.324e+03 +	1.972e+03 +	1.266e+05 +	4.567e-05
		Std	6.199e+02	2.785e+03	5.694e+02	1.417e+04	6.183e-05
	F4	Ort	22.592 +	12.818 +	6.480 +	80.964 +	0.002
		Std	7.115	2.253	2.141	1.885	0.002
	F5	Ort	7.404e+03 +	6.699e+02+	5.216e+02 +	2.690e+08 +	4.242
		Std	1.327e+04	5.221e+02	7.023e+02	2.900e+07	12.059
	F6	Ort	2.445e-04 -	2.515 +	3.794 +	8.657e+04 +	0.055
		Std	2.944e-04	2.145	0.783	4.187e+03	0.058
	F7	Ort	0.050 +	0.257+	0.056 +	2.052e+02 +	0.001
		Std	0.026	0.092	0.021	22.287	0.001
Çok modlu	**F8**	Ort	-1.616e+04 -	-1.217e+04 =	-1.282e+04 =	-5.447e+03 +	-1.248e+04
		Std	1.737e+03	1.012e+03	8.717e+02	2.961e+02	1.493e+03
	F9	Ort	0.051 +	51.366 +	2.359e+02 +	6.414e+02 +	5.129e-05
		Std	0.242	20.239	38.201	23.795	1.075e-04
	F10	Ort	0.002 +	3.668 +	2.293 +	20.237 +	0.001
		Std	0.001	0.979	0.434	0.115	0.002
	F11	Ort	0.105 +	0.827 +	0.999 +	7.977e+02 +	1.398e-05
		Std	0.193	0.205	0.041	35.348	2.099e-05
	F12	Ort	0.003 +	6.401 +	3.395 +	5.189e+08 +	0.001
		Std	0.021	2.720	1.263	1.059e+08	0.001
	F13	Ort	0.019 +	58.144 +	0.543 +	1.146e+09 +	0.010
		Std	0.062	14.707	0.474	1.086e+08	0.016
Sabit boyutlu çok modlu	**F14**	Ort	4.999 +	1.196 +	0.998 =	1.045 +	0.998
		Std	2.669	0.546	1.777e-11	0.111	9.313e-11
	F15	Ort	0.001 +	8.329e-04 +	0.003 +	0.002 +	3.659e-04
		Std	9.915e-04	2.614e-04	0.007	8.949e-04	8.859e-05
	F16	Ort	-1.030 +	-1.031 =	-1.031 +	-1.029 +	-1.031
		Std	0.001	2.618e-10	1.696e-07	0.002	7.430e-11
	F17	Ort	0.398 =	0.397 +	0.397 +	0.400 +	0.397
		Std	0.002	4.632e-4	3.694e-07	0.002	3.342e-07
	F18	Ort	4.187 +	3.000 =	3.000 +	3.101 +	3.000
		Std	4.992	1.693e-08	1.629e-06	0.124	4.963e-08
	F19	Ort	-3.851 +	-3.862 =	-3.862 =	-3.856 +	-3.862
		Std	0.011	2.197e-07	2.444e-07	0.003	1.030e-06
	F20	Ort	-3.248 +	-3.209 +	-3.254 +	-3.074 +	-3.266
		Std	0.077	0.038	0.060	0.062	0.060
	F21	Ort	-7.663+	-8.140 +	-7.036 +	-3.630 +	-10.153
		Std	3.581	2.978	2.862	1.242	5.914e-05
	F22	Ort	-8.469 +	-10.004 -	-8.836 +	-3.915 +	-9.954
		Std	3.276	1.527	2.695	0.907	1.693
	F23	Ort	-10.024 +	-9.531 +	-9.124 +	-3.771 +	-10.089
		Std	1.944	2.612	2.654	1.088	1.7002
	+/=/-		20/1/2	18/4/1	20/3/0	23/0/0	
	Ort. Rank		7.673	7.782	7.652	13.891	3.217

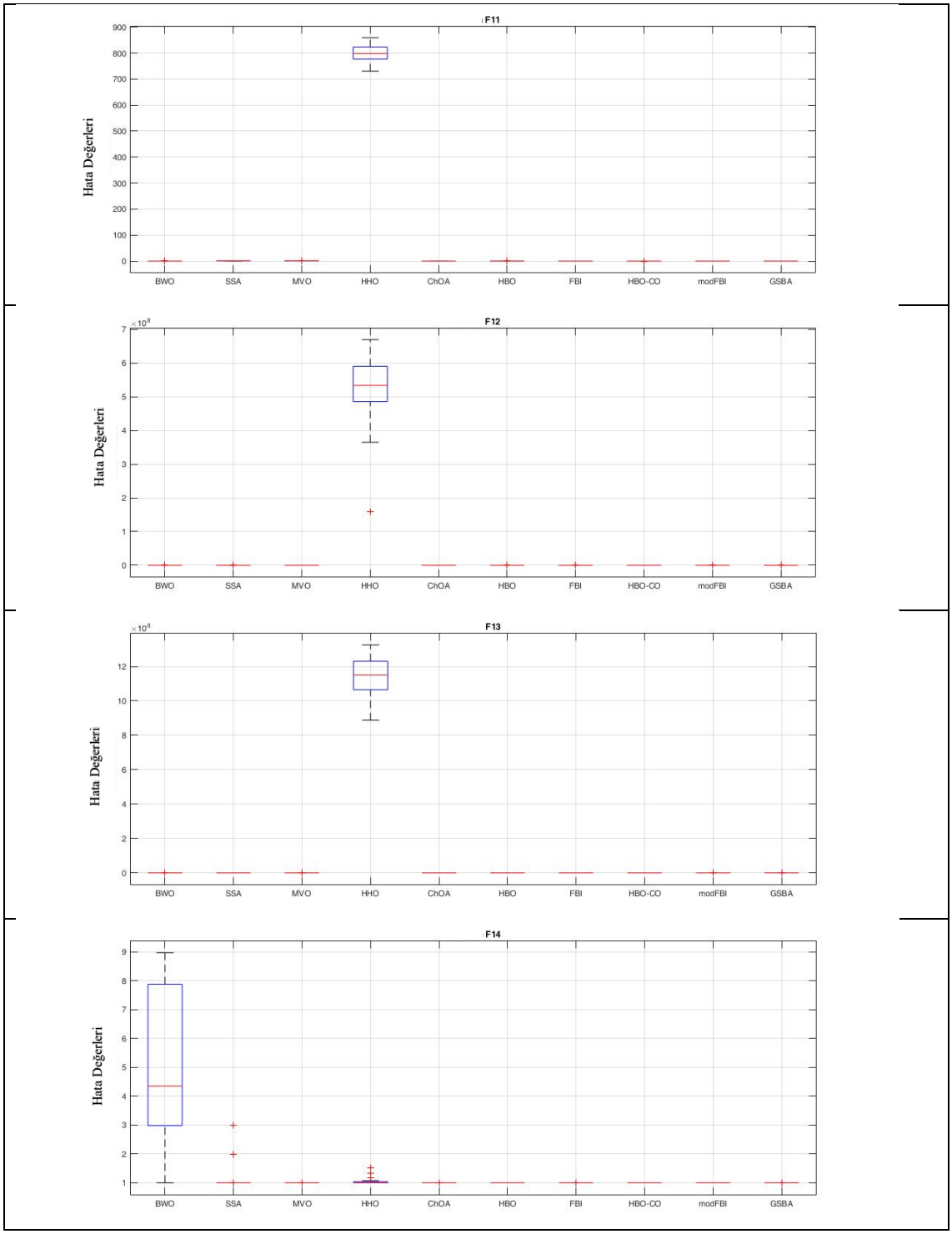
Çizelge 4.4. Güncel algoritmaların 23 fonksiyon için test sonuçları (2/2)

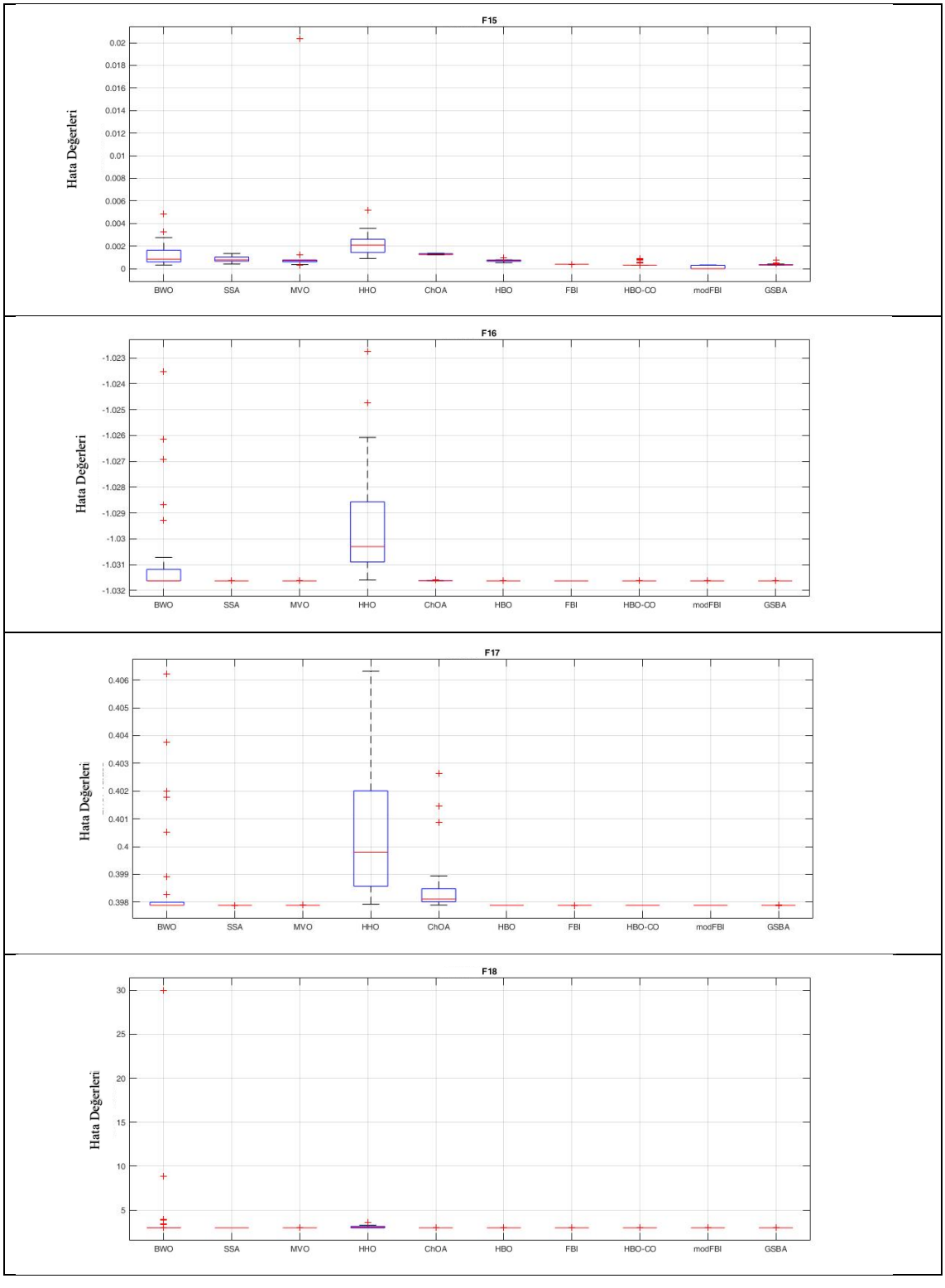
			ChOA	HBO	FBI	HBO-CO	modFBI
Tek modlu	F1	Ort	0.016 +	0.355 +	0.000 -	0.000 -	0.000 -
		Std	0.018	0.170	0.000	0.000	0.000
	F2	Ort	0.006 +	0.049 +	5.537e-08 +	6.901e-09 +	2.045e-09 +
		Std	0.005	0.013	3.114e-08	1.297e-09	2.576e-09
	F3	Ort	3.637e+03 +	9.596e+04 +	0.000 -	0.000 -	0.000 -
		Std	3.327e+03	1.110e+04	0.000	0.000	0.000
	F4	Ort	4.915 +	32.371 +	3.938e-06 -	0.000 -	4.250e-39 -
		Std	3.686	4.4061	2.431e-06	0.000	1.219e-38
	F5	Ort	49.630 +	5.453e+02 +	10.857 +	35.758 +	9.011 +
		Std	0.786	3.449e+2	20.018	18.799	15.62
	F6	Ort	6.765 +	0.334 +	0.074 +	9.983 +	0.062 +
		Std	0.902	0.113	0.140	5.077	0.176
	F7	Ort	0.004 +	0.108 +	0.106 +	0.075 +	0.064 +
		Std	0.004	0.028	0.208	0.108	0.176
Çok moldu	F8	Ort	-9.306e+03 +	-1.364e+04 -	-1.112e+04 +	-1.156e+04 +	-2.094e+04 -
		Std	57.570	9.836e+03	3.035e+02	1.855e+03	0.007
	F9	Ort	30.719 +	1.261e+02 +	7.263e-04 +	2.947e-04 +	1.261e-04 +
		Std	29.224	18.285	3.395e-04	3.169e-04	1.828e-05
	F10	Ort	19.964 +	0.200 +	0.882 +	0.882 +	0.012 +
		Std	9.734e-04	0.229	2.020e-16	1.129e-16	0.001
	F11	Ort	0.087 +	0.350 +	0.043 +	3.139e-04 +	0.035 +
		Std	0.111	0.108	0.029	3.280e-05	0.023
	F12	Ort	0.627 +	0.839 +	0.033 +	0.019 +	0.016 +
		Std	0.174	0.505	0.013	0.032	0.048
	F13	Ort	4.686 +	4.646 +	0.214 =	1.298 +	0.166 +
		Std	0.129	2.443	0.293	2.195	0.006
Sabit boyulu çok modlu	F14	Ort	0.998 +	0.998 -	0.988 +	0.988 -	0.988 -
		Std	2.774e-05	0.000	3.257e-05	0.000	0.000
	F15	Ort	0.001 +	7.007e-04 +	3.974e-04 +	3.769e-04 +	1.134e-04 -
		Std	4.409e-05	8.817e-05	2.013e-20	1.638e-04	1.517e-04
	F16	Ort	-1.031 +	-1.031 -	-1.031 -	-1.031 -	-1.031 -
		Std	6.847e-06	6.648e-16	6.775e-16	8.450e-16	6.712e-16
	F17	Ort	0.398 +	0.397 -	0.397 -	0.397 -	0.397 -
		Std	0.001	0.000	1.056e-16	0.000	0.000
	F18	Ort	3.000 +	3.000 +	3.000 +	3.000 -	3.000 -
		Std	1.204e-04	1.423e-04	3.073e-04	6.059e-16	0.000
	F19	Ort	-3.855 +	-3.862 -	-3.862 -	-3.862 -	-3.862 -
		Std	0.002	2.710e-15	2.710e-15	1.110e-15	2.668e-15
	F20	Ort	-2.903 +	-2.924 +	-2.942 +	-3.001 +	-3.182 +
		Std	0.234	0.324	0.303	0.269	0.219
	F21	Ort	-4.380 +	-10.093 +	-10.092 +	-10.100 +	-10.103 +
		Std	1.422	0.064	0.077	0.0524	0.025
	F22	Ort	-4.865 +	-8.070 +	-8.811 +	-9.716 +	-9.453 +
		Std	0.748	0.936	0.043	-9.716	1.092e-06
	F23	Ort	-5.052 +	-10.535 -	-6.622 +	-10.536 -	-10.536 -
		Std	0.032	0.007	0.013	3.7600e-15	1.205e-06
	+/=/-		23/0/0	17/6/0	16/5/2	14/0/9	12/011
	Ort. Rank		8.978	8.000	5.760	4.869	3.673

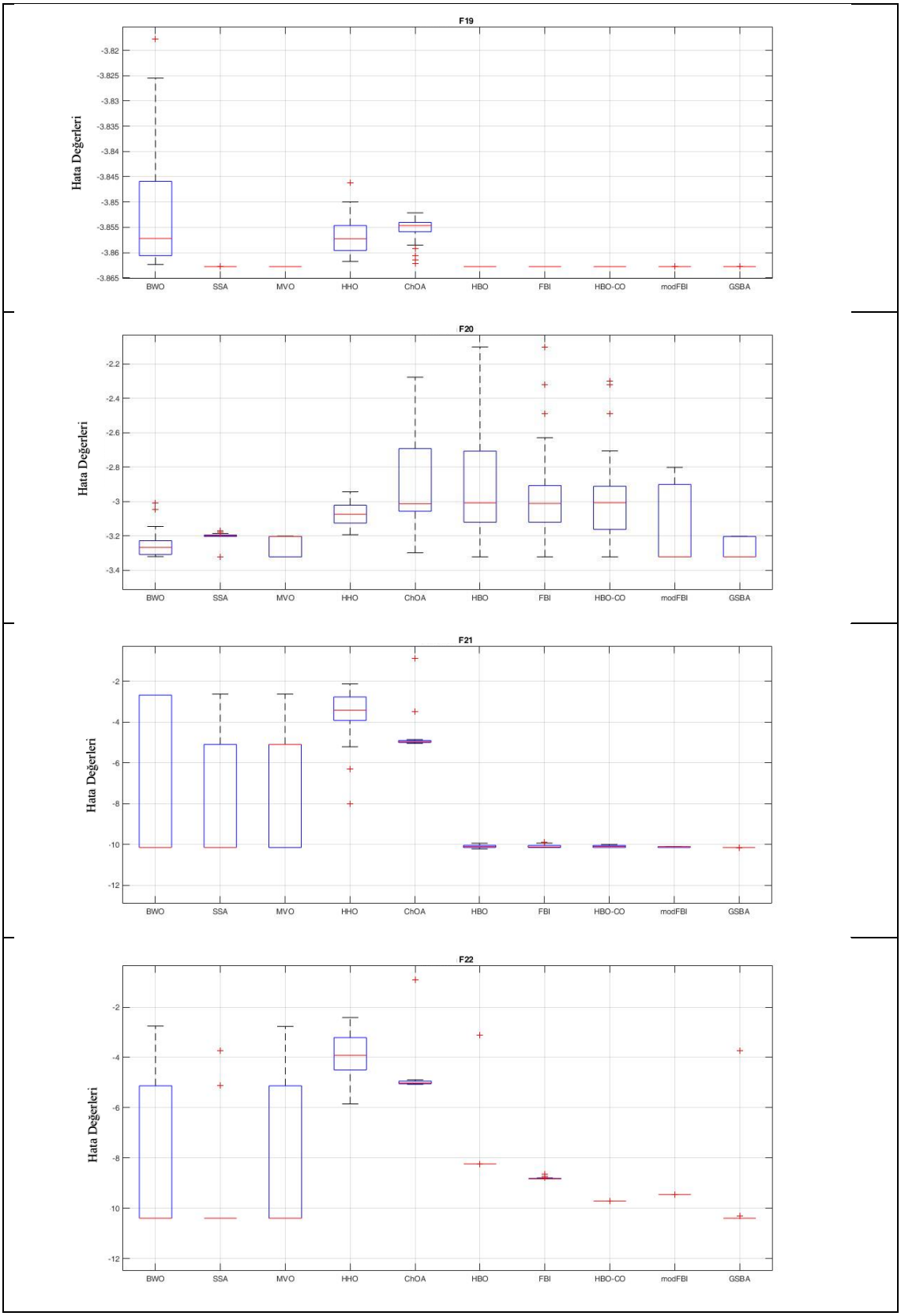
Çizelge 4.5. Güncel algoritmaların 23 fonksiyon için hata dağılımları

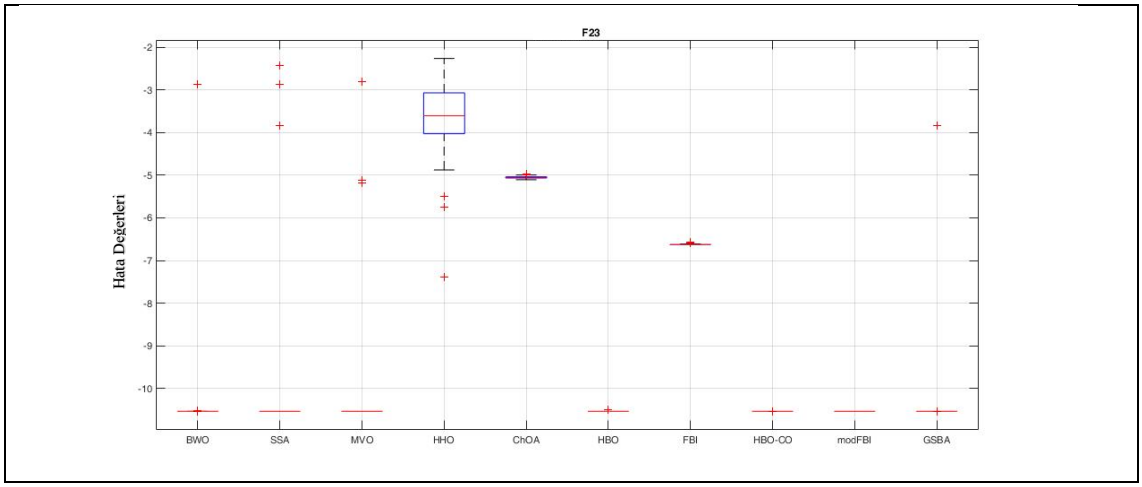



F6
Hata Değerleri
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
Function F7
F8
F9
F10


F11
Hata Değerleri
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F12
Hata Değerleri
F13
Hata Değerleri
F14
Hata Değerleri


F15
Hata Değerleri
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F16
Hata Değerleri
-1.023
-1.024
-1.025
-1.026
-1.027
-1.028
-1.029
-1.03
-1.031
-1.032
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F17
Hata Değerleri
0.406
0.405
0.404
0.403
0.402
0.401
0.4
0.399
0.398
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F18
Hata Değerleri
30
25
20
15
10
5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA


F19
Hata Değerleri
-3.82
-3.825
-3.83
-3.835
-3.84
-3.845
-3.85
-3.855
-3.86
-3.865
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F20
Hata Değerleri
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F21
Hata Değerleri
-2
-4
-6
-8
-10
-12
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F22
Hata Değerleri
-2
-4
-6
-8
10
-12
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA



Çizelge 4.6. Klasik algoritmaların 20 fonksiyon için test sonuçları

			GA	PSO	DE	CS	HS	GSBA
Hibrid	**F1**	Ort	1.376e+05+	3.856e+05+	2.081e+05+	3.479e+05+	6.017e+05+	2.317e+04
		Std	2.113e+04	8.923e+04	3.002e+04	5.035e+04	7.977e+04	1.353e+04
	F2	Ort	1.670e+10+	2.599e+10+	2.902e+10+	1.000e+10+	1.306e+10+	7.754e+08
		Std	9.248e+09	7.045e+09	2.693e+09	0.000	1.775e+09	2.430e+08
	F3	Ort	6.441e+08+	3.272e+09+	1.199e+09+	1.000e+10+	3.537e+08+	1.014e+05
		Std	7.043e+08	2.300e+09	2.617e+08	0.000	8.362e+07	4.342e+04
	F4	Ort	1.073e+07+	3.888e+07+	3.868e+07+	5.157e+07+	1.619e+08+	4.134e+06
		Std	4.736e+06	1.277e+07	1.215e+07	1.627e+07	6.143e+07	1.947e+06
	F5	Ort	6.867e+07+	1.595e+09+	2.048e+08+	1.000e+10+	1.930e+07+	1.010e+05
		Std	1.842e+08	7.188e+08	4.245e+07	0.00	9.011e+06	6.473e+04
	F6	Ort	8.434e+03-	1.135e+04=	1.221e+04=	1.961e+04+	1.079e+04-	1.195e+04
		Std	7.887e+02	8.418e+02	6.521e+02	2.080e+03	4.010e+02	1.147e+03
	F7	Ort	7.061e+03+	1.034e+04+	1.003e+04+	1.849e+06+	8.044e+03+	5.359e+03
		Std	5.210e+02	1.079e+03	5.506e+02	1.059e+06	3.553e+02	3.263e+02
	F8	Ort	2.409e+07=	4.984e+07+	7.544e+07+	7.469e+07+	3.040e+08+	1.728e+07
		Std	1.182e+07	1.590e+07	2.187e+07	1.520e+07	8.303e+07	9.184e+06
	F9	Ort	1.227e+08+	1.450e+09+	3.744e+08+	1.000e+10+	4.017e+07+	2.371e+07
		Std	1.444e+08	6.591e+08	8.352e+07	0.000	1.839e+07	1.042e+07
	F10	Ort	6.118e+03+	7.771e+03+	7.293e+03+	6.494e+03+	8.106e+03+	5.066e+03
		Std	7.038e+02	4.651e+02	2.679e+02	1.410e+02	1.381e+02	6.306e+02
Kompozisyon	**F11**	Ort	3.559e+03+	3.754e+03+	3.811e+03+	4.856e+03+	3.483e+03+	3.150e+03
		Std	97.999	82.122	35.371	1.092e+02	21.000	1.081e+02
	F12	Ort	2.700e+04+	3.463e+04+	3.270e+04+	2.893e+04+	3.518e+04+	2.154e+04
		Std	1.895e+03	1.049e+03	8.702e+02	5.388e+02	5.354e+02	8.189e+02
	F13	Ort	3.938e+03+	3.969e+03+	3.980e+03+	6.435e+03+	3.849e+03+	3.720e+03
		Std	1.100e+02	62.786	28.478	1.449e+02	48.782	1.189e+02
	F14	Ort	4.385e+03-	4.454e+03-	4.615e+03-	9.430e+03+	4.551e+03+	6.278e+03
		Std	1.037e+02	38.749	35.187	4.868e+02	58.551	5.072e+02
	F15	Ort	1.582e+04+	3.509e+04+	2.930e+04+	4.998e+04+	9.997e+03=	1.044e+04
		Std	2.319e+03	8.450e+03	4.779e+03	2.468e+03	8.541e+02	8.945e+02
	F16	Ort	1.901e+04+	1.907e+04+	2.030e+04+	5.386e+04+	1.919e+04+	1.479e+04
		Std	1.411e+03	7.513e+02	3.670e+02	2.376e+03	3.583e+02	1.049e+03
	F17	Ort	3.795e+03-	3.905e+03-	4.734e+03-	9.930e+03+	4.601e+03-	6.293e+03
		Std	1.324e+02	1.788e+02	1.600e+02	6.814e+02	2.125e+02	7.428e+02
	F18	Ort	1.576e+04+	1.858e+04+	2.196e+04+	3.921e+04+	1.664e+04+	3.656e+03
		Std	2.189e+03	8.414e+02	8.549e+02	2.057e+03	9.705e+02	48.884
	F19	Ort	1.024e+04+	1.878e+04+	1.348e+04+	2.381e+05+	1.097e+04+	8.992e+03
		Std	9.653e+02	1.787e+03	7.823e+02	1.130e+05	4.336e+02	6.333e+02
	F20	Ort	4.618e+08=	2.743e+09+	3.816e+08=	1.000e+10+	2.062e+08=	2.214e+08
		Std	3.678e+08	9.151e+08	5.721e+07	0.000	6.722e+07	7.689e+07
	+/=/-		15/2/3	18/1/2	16/2/2	20/0/0	18/2/2	
	Ort. Rank		6.250	9.850	9.550	12.250	8.250	

Çizelge 4.7. Güncel algoritmaların 20 fonksiyon için test sonuçları (1/2)

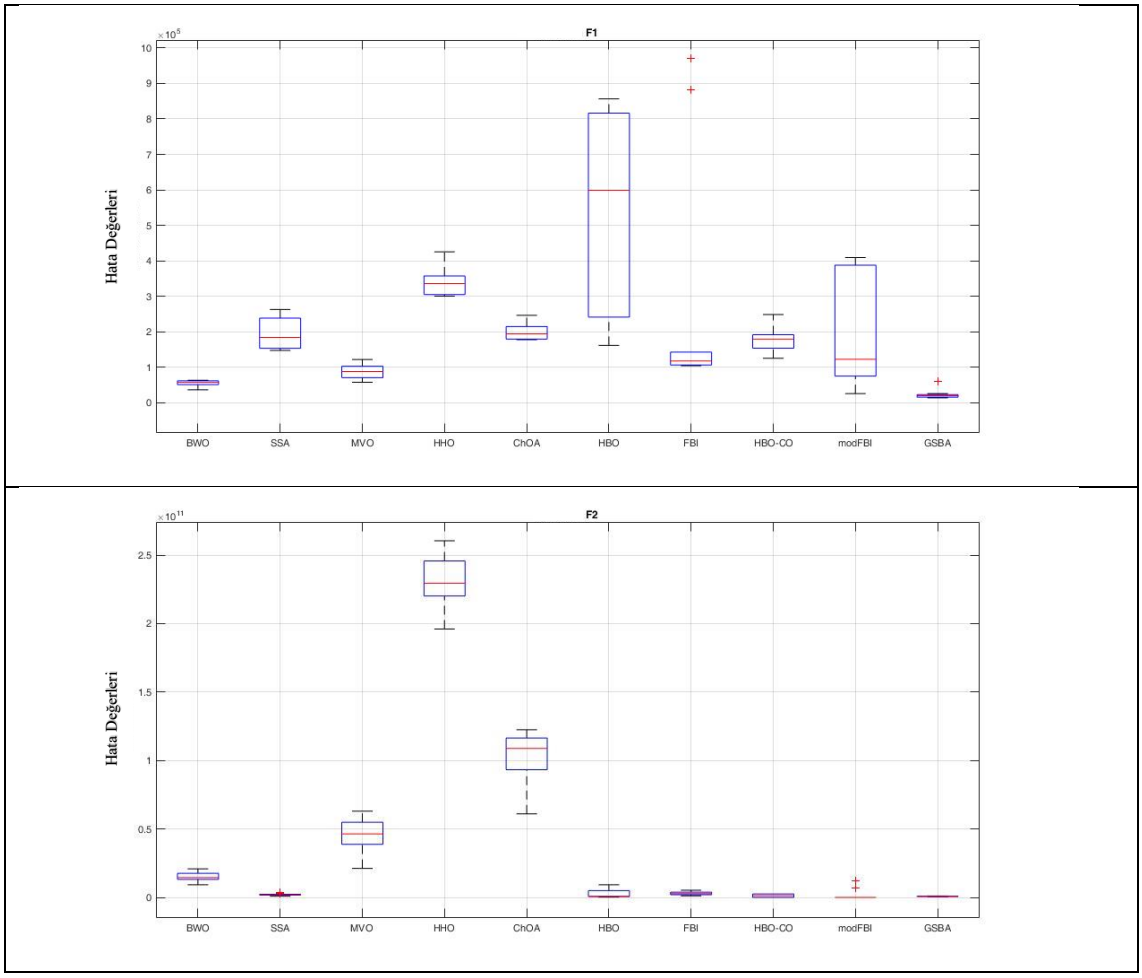
			BWO	SSA	MVO	HHO	GSBA
Hibrid	**F1**	Ort	5.448e+04+	1.958e+05+	8.820e+04+	3.396e+05+	2.317e+04
		Std	9.023e+03	4.634e+04	2.051e+04	3.804e+04	1.353e+04
	F2	Ort	1.497e+10+	2.040e+09+	4.554e+10+	2.297e+11+	7.754e+08
		Std	3.938e+09	8.576e+08	1.304e+10	2.067e+10	2.430e+08
	F3	Ort	2.736e+09+	4.792e+09+	1.675e+06+	4.920e+10+	1.014e+05
		Std	1.221e+09	1.667e+09	4.376e+05	5.695e+09	4.342e+04
	F4	Ort	8.902e+06+	8.210e+06+	1.525e+07+	1.853e+08+	4.134e+06
		Std	4.578e+06	3.595e+06	5.588e+06	5.219e+07	1.947e+06
	F5	Ort	1.026e+09+	1.118e+09+	4.321e+05+	2.161e+10+	1.010e+05
		Std	6.322e+08	1.251e+09	1.451e+05	2.659e+09	6.473e+04
	F6	Ort	7.008e+03-	7.676e+03-	6.823e+03-	2.323e+04+	1.195e+04
		Std	8.290e+02	9.056e+02	1.070e+03	1.586e+03	1.147e+03
	F7	Ort	7.377e+03+	6.047e+03+	9.764e+03+	3.165e+06+	5.359e+03
		Std	2.714e+03	7.505e+02	3.631e+03	2.052e+06	3.263e+02
	F8	Ort	5.580e+06-	6.623e+06-	7.174e+06=	3.344e+08+	1.728e+07
		Std	2.902e+06	4.398e+06	3.880e+06	1.002e+08	9.184e+06
	F9	Ort	1.161e+09+	3.258e+07+	4.962e+08+	2.310e+10+	2.371e+07
		Std	7.313e+08	2.040e+07	4.573e+08	2.524e+09	1.042e+07
	F10	Ort	6.197e+03+	5.776e+03=	5.435e+03=	8.250e+03+	5.066e+03
		Std	4.585e+02	8.208e+02	2.973e+02	2.882e+02	6.306e+02
Kompozisyon	**F11**	Ort	3.260e+03=	3.453e+03+	3.909e+03+	4.795e+03+	3.150e+03
		Std	1.132e+02	1.868e+02	1.386e+02	71.909	1.081e+02
	F12	Ort	3.010e+04+	2.312e+04+	2.233e+04=	3.566e+04+	2.154e+04
		Std	1.676e+03	1.162e+03	1.927e+03	6.036e+02	8.189e+02
	F13	Ort	4.254e+03+	3.908e+03+	4.556e+03+	6.539e+03+	3.720e+03
		Std	1.666e+02	1.949e+02	1.945e+02	2.103e+02	1.189e+02
	F14	Ort	6.003e+03=	4.552e+03-	4.302e+03-	1.060e+04+	6.278e+03
		Std	3.238e+02	1.782e+02	1.724e+02	4.458e+02	5.072e+02
	F15	Ort	6.588e+03-	5.854e+03-	3.681e+03-	8.495e+04+	1.044e+04
		Std	6.504e+02	3.383e+02	82.619	7.316e+03	8.945e+02
	F16	Ort	2.527e+04+	2.204e+04+	3.244e+04+	6.542e+04+	1.479e+04
		Std	1.070e+03	3.375e+03	1.214e+03	2.734e+03	1.049e+03
	F17	Ort	6.173e+03=	4.191e+03-	3.698e+03-	1.215e+04+	6.293e+03
		Std	3.099e+02	1.467e+02	1.121e+02	6.301e+02	7.428e+02
	F18	Ort	8.883e+03+	8.586e+03+	1.426e+04+	5.300e+04+	3.656e+03
		Std	5.056e+02	1.436e+03	1.461e+03	1.961e+03	48.884
	F19	Ort	1.362e+04+	1.147e+04+	9.216e+03=	6.123e+05+	8.992e+03
		Std	1.798e+03	1.463e+03	1.008e+03	4.249e+05	6.333e+02
	F20	Ort	2.655e+09+	5.708e+08+	6.988e+09+	3.805e+10+	2.214e+08
		Std	1.105e+09	2.824e+08	3.540e+09	2.718e+09	7.689e+07
	+/=/-		14/3/3	14/1/5	12/4/4	20/0/0	
	Ort. Rank		7.350	6.200	6.300	14.300	3.950

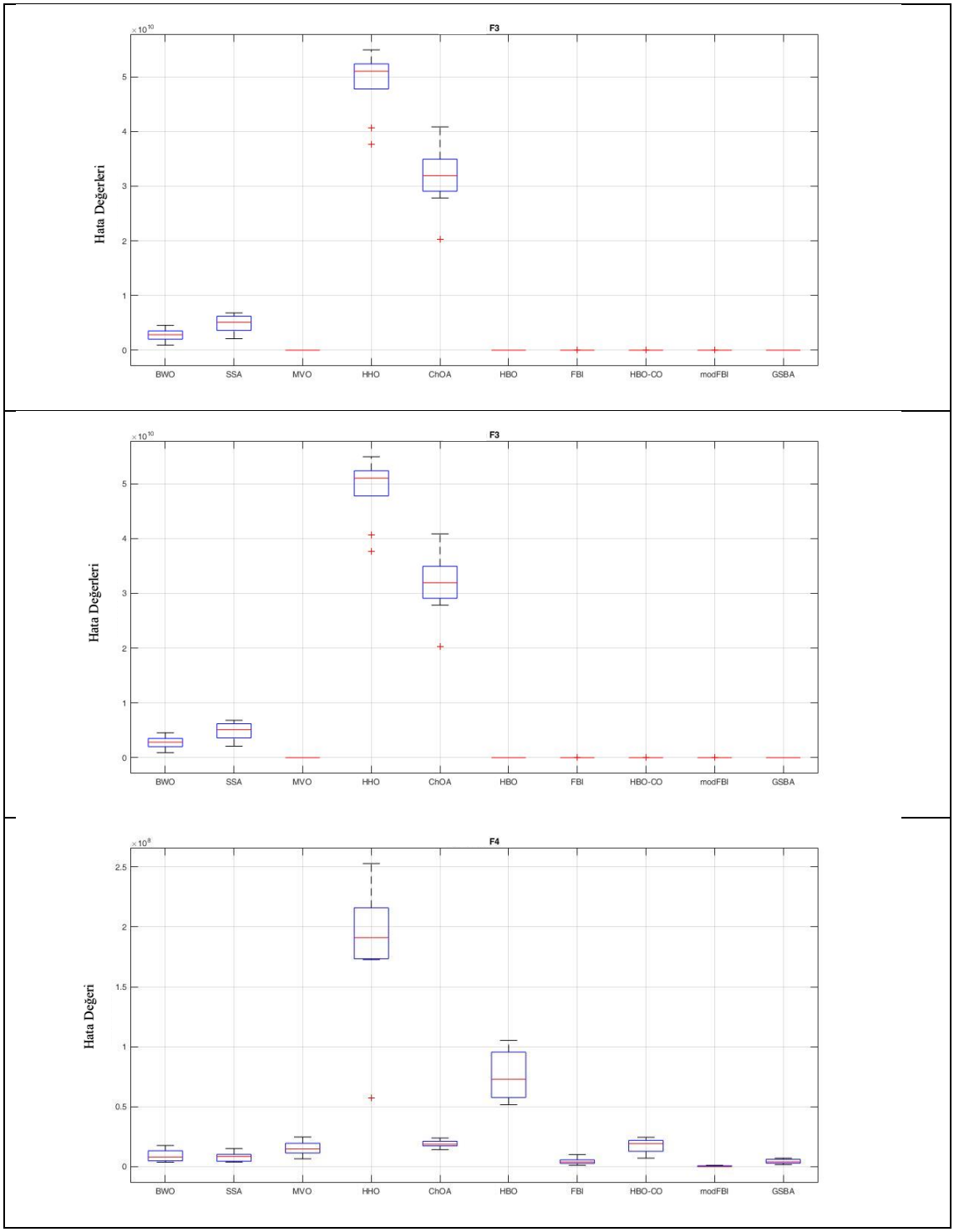
Çizelge 4.8. Güncel algoritmaların 20 fonksiyon için test sonuçları (2/2)

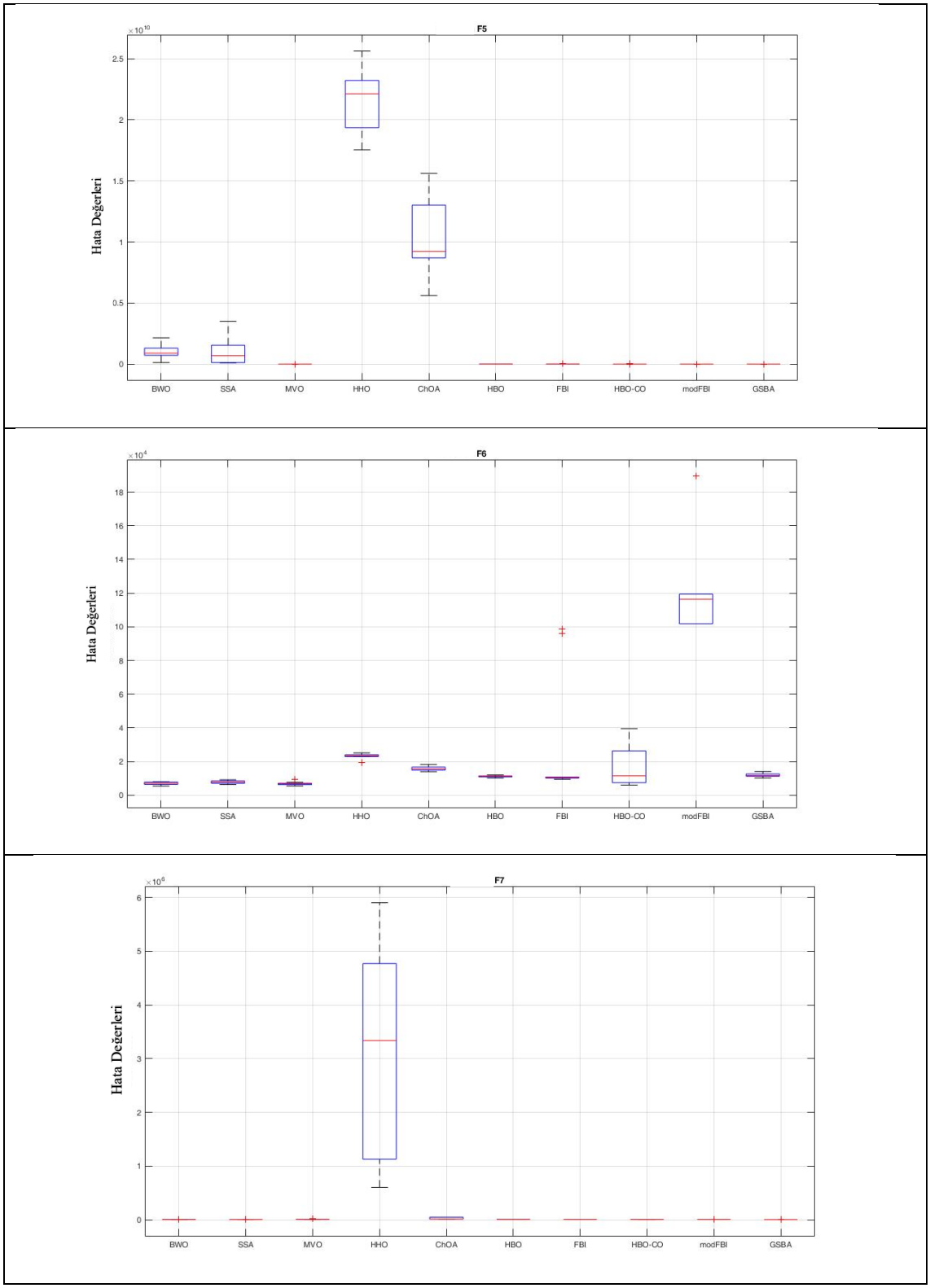
			ChOA	HBO	FBI	HBO-CO	modFBI
Hibrid	**F1**	Ort	2.015e+05+	2.509e+05+	2.776e+05+	1.771e+05+	1.801e+05+
		Std	2.503e+04	2.723+05	3.424e+05	3.383e+04	1.547e+05
	F2	Ort	1.022e+11+	2.787e+09+	3.030e+09+	1.329e+09+	1.859e+09+
		Std	1.951e+10	3.461e+09	1.432e+09	1.330e+09	4.111e+09
	F3	Ort	3.172e+10+	4.439+05+	2.628e+05+	6.579e+05+	3.520e+06+
		Std	5.504e+09	3.218e+05	1.750e+05	2.0797e+06	7.590e+06
	F4	Ort	1.906e+07+	7.617+07+	4.639e+06=	1.729e+07+	3.087+05-
		Std	2.888e+06	1.966e+07	2.970e+06	6.124e+06	4.330e+05
	F5	Ort	1.027e+10+	5.379e+06+	8.961e+06+	2.996e+06+	3.192e+06+
		Std	3.291e+09	1.460e+03	5.558e+06	5.791e+06	1.337e+06
	F6	Ort	1.580e+04+	1.116e+04=	8.961e+06+	1.818e+04+	1.187e+05+
		Std	1.376e+03	4.901e+02	3.671e+04	1.259e+04	2.621e+04
	F7	Ort	2.722e+04+	8.830e+03+	8.110e+03+	7.983e+03+	8.027e+03+
		Std	1.719e+04	6.798e+02	3.609e+02	4.398e+02	3.234e+02
	F8	Ort	3.004e+07+	1.319e+08+	6.928e+07+	4.454e+06-	1.803e+06-
		Std	1.185e+07	2.651e+07	3.276e+07	1.360e+06	4.598e+06
	F9	Ort	9.032e+09+	6.407e+03-	1.645e+04-	1.186e+03-	1.163e+03-
		Std	4.918e+09	5.345e+03	1.098e+04	2.292e+03	6.199e+02

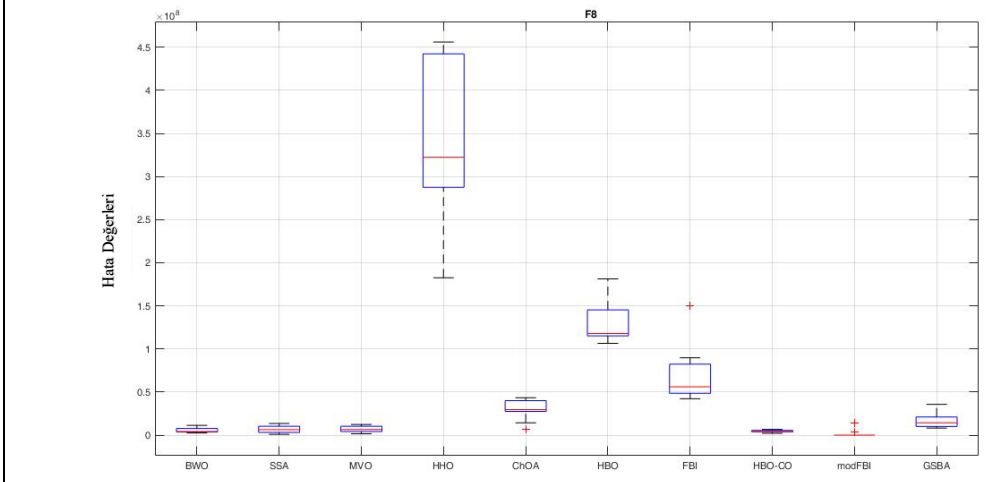
Komposizyon						
F10	Ort	8.216e+03+	2.908+04+	2.470e+04+	6.779e+03+	1.724e+03=
	Std	4.025e+02	3.373e+04	2.886e+04	8.061e+02	9.425e+04
F11	Ort	4.268e+03+	3.401e+03+	3.510e+03+	3.358e+03+	3.461+03+
	Std	1.072e+02	42.550	40.270	35.802	1.563e+03
F12	Ort	3.546e+04+	1.300e+05+	1.208e+05+	3.274e+04+	4.343e+03-
	Std	3.872e+02	1.534e+05	1.216e+05	8.053e+02	9.163e+03
F13	Ort	5.577e+03+	1.377e+04+	4.133e+04+	3.719e+03=	2.719e+04+
	Std	1.667e+02	1.610e+04	6.267e+02	1.198e+02	3.321e+04
F14	Ort	7.555e+03+	4.308+03-	5.096e+03-	4.320e+03-	3.469e+04+
	Std	4.319e+02	55.649	91.908	34.067	3.6254e+04
F15	Ort	1.725e+04+	3.985e+03-	5.684e+03-	3.105e+03-	3.932e+03-
	Std	1.195e+03	1.072e+02	2.747e+02	6.002e+03	2.157e+03
F16	Ort	3.144e+04+	4.603e+04+	2.200e+04+	1.581e+04=	2.140e+04+
	Std	7.820e+02	6.255e+04	1.906e+03	5.451e+03	2.837e+03
F17	Ort	7.615e+03+	4.253e+03-	5.009e+03-	4.057e+03-	3.270e+03-
	Std	5.806e+02	3.958e+02	1.370e+02	2.370e+02	1.629e+03
F18	Ort	1.764e+04+	8.487e+03+	8.085e+03+	4.500e+03+	7.279e+03+
	Std	9.571e+02	1.066e+03	7.040e+02	3.135e+02	1.557e+03
F19	Ort	7.632+04+	1.032e+04+	1.118e+04+	3.555e+06+	1.730e+05+
	Std	4.769e+04	5.926e+02	7.211e+02	4.577e+06	7.884e+04
F20	Ort	1.974e+10+	3.317+06-	1.593e+07-	9.425e+06-	3.655e+07-
	Std	1.906e+09	2.984e+06	8.267e+06	1.707e+07	4.994e+07
+/=/-		20/0/0	14/1/5	14/1/5	12/2/6	12/1/7
Ort. Rank		11.250	6.150	8.250	4.850	5.250

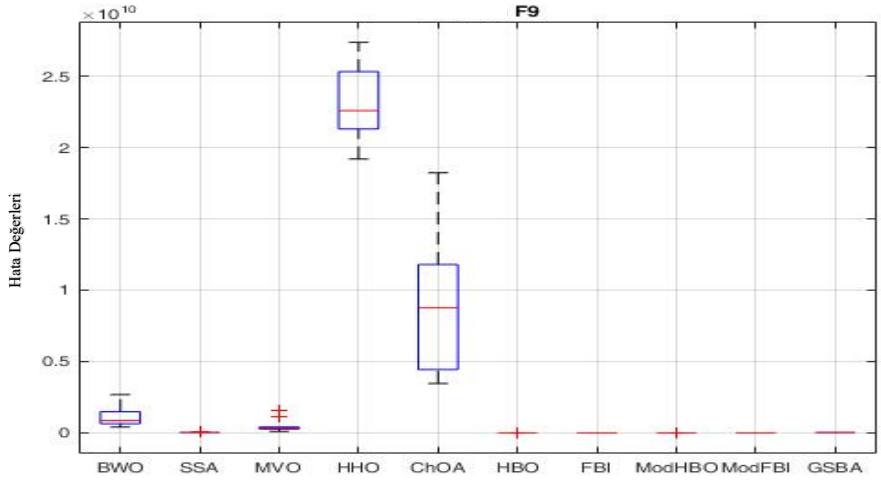
Çizelge 4.9. Güncel algoritmaların 20 fonksiyon üzerindeki hata dağılımları

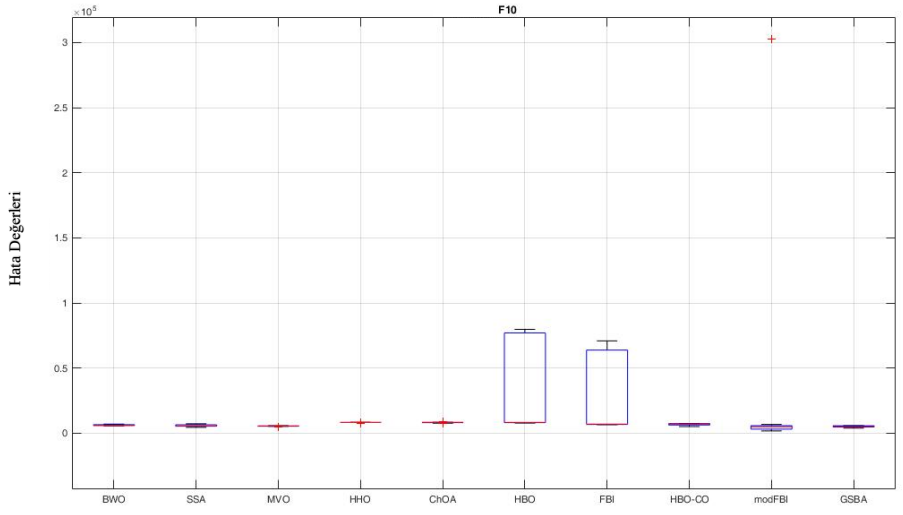


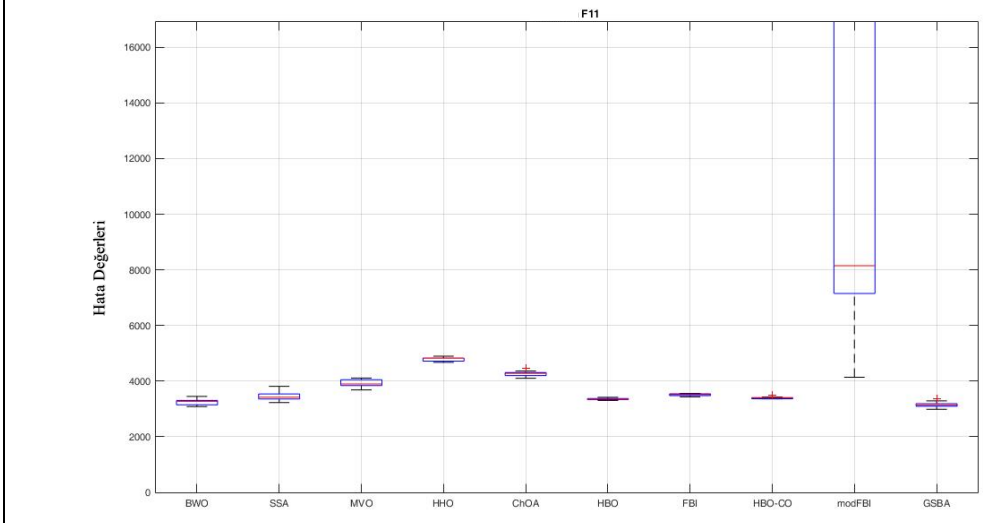

F3
×10^10
Hata Değerleri
0
1
2
3
4
5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F3
×10^10
Hata Değerleri
0
1
2
3
4
5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F4
×10^8
Hata Değeri
0
0.5
1
1.5
2
2.5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

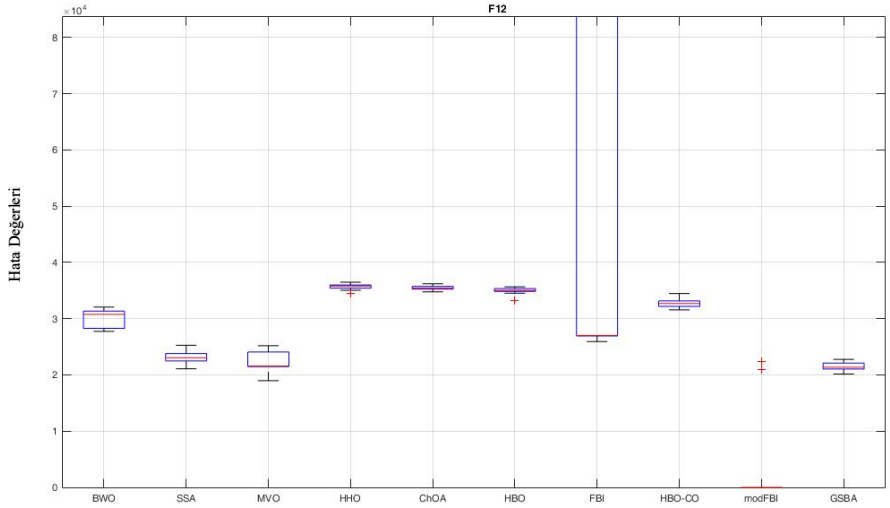

F5
×10^10
Hata Değerleri
2.5
2
1.5
1
0.5
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F6
×10^4
Hata Değerleri
18
16
14
12
10
8
6
4
2
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA
F7
×10^6
Hata Değerleri
6
5
4
3
2
1
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

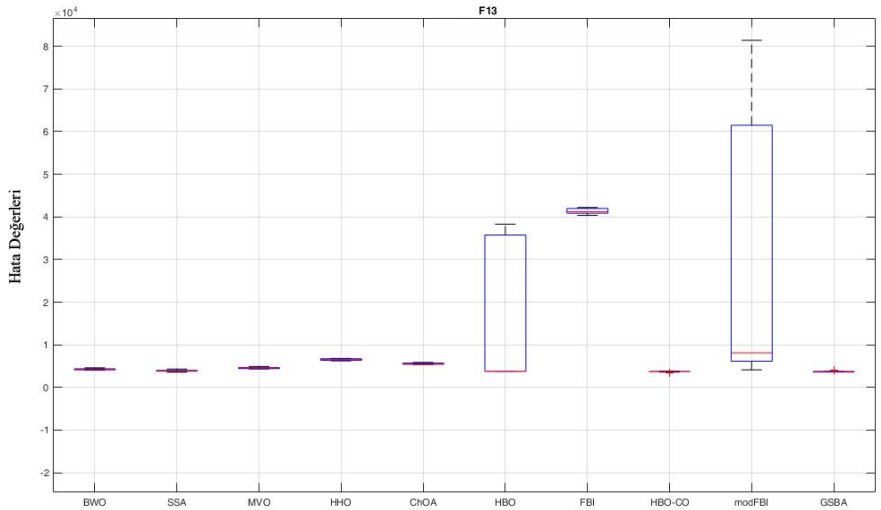

F8
×10^9
Hata Değerleri
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

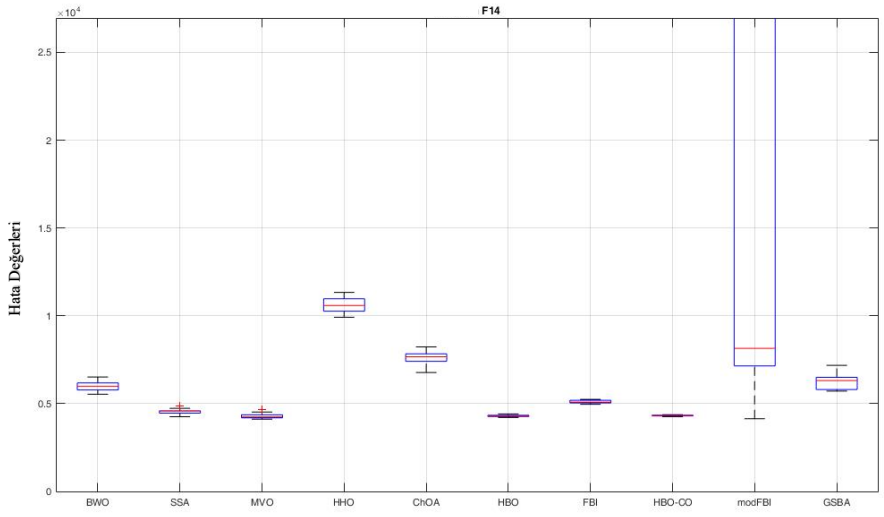

F9
×10^10
Hata Değerleri
2.5
2
1.5
1
0.5
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
ModHBO
ModFBI
GSBA

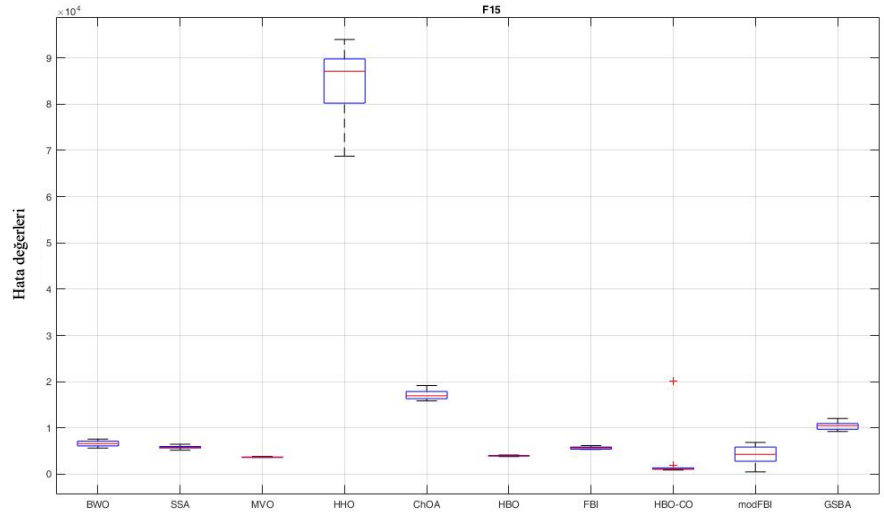

F10
×10^5
Hata Değerleri
3
2.5
2
1.5
1
0.5
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

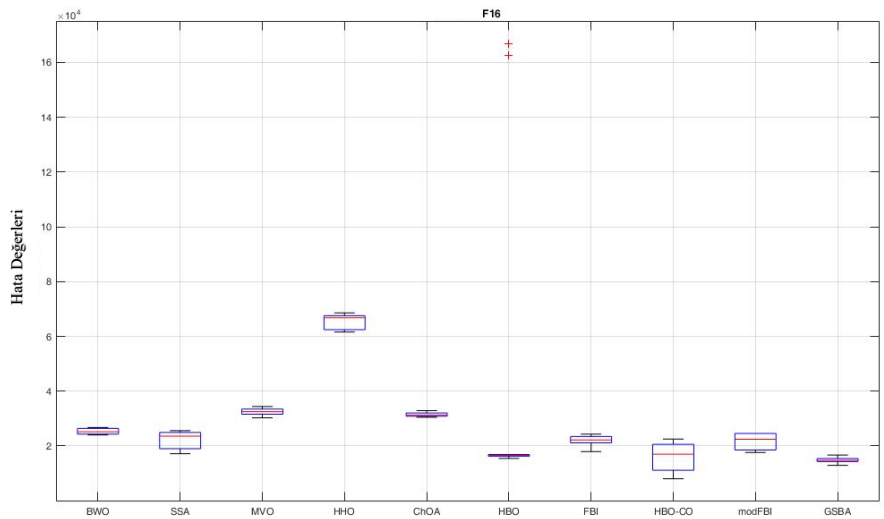

F11
Hata Değerleri
16000
14000
12000
10000
8000
6000
4000
2000
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

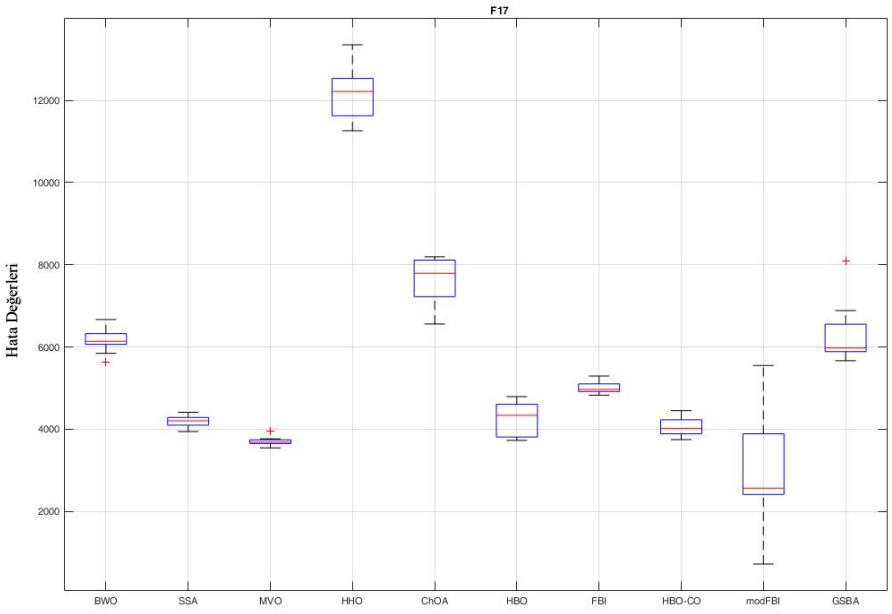

F12
×10^4
Hata Değerleri
8
7
6
5
4
3
2
1
0
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

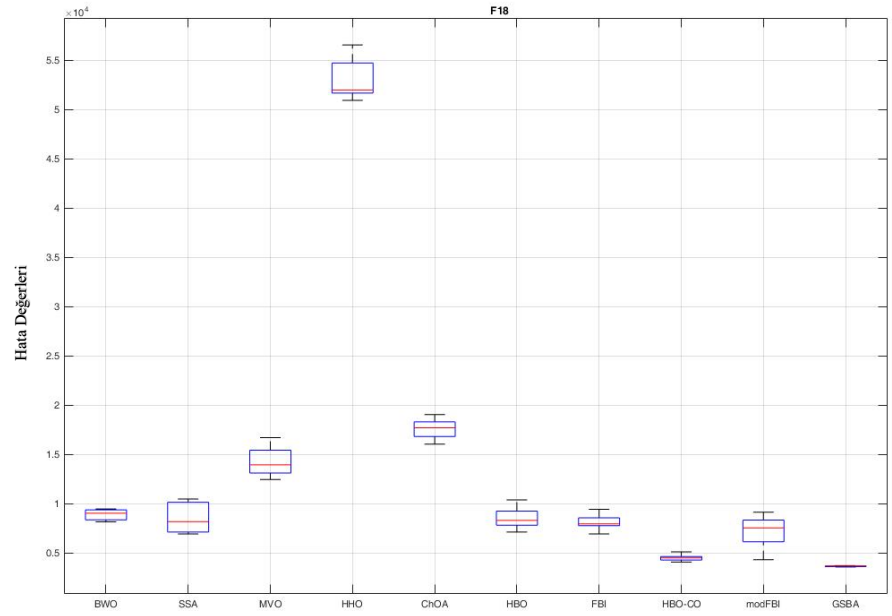

F13
×10^4
Hata Değerleri
8
7
6
5
4
3
2
1
0
-1
-2
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA

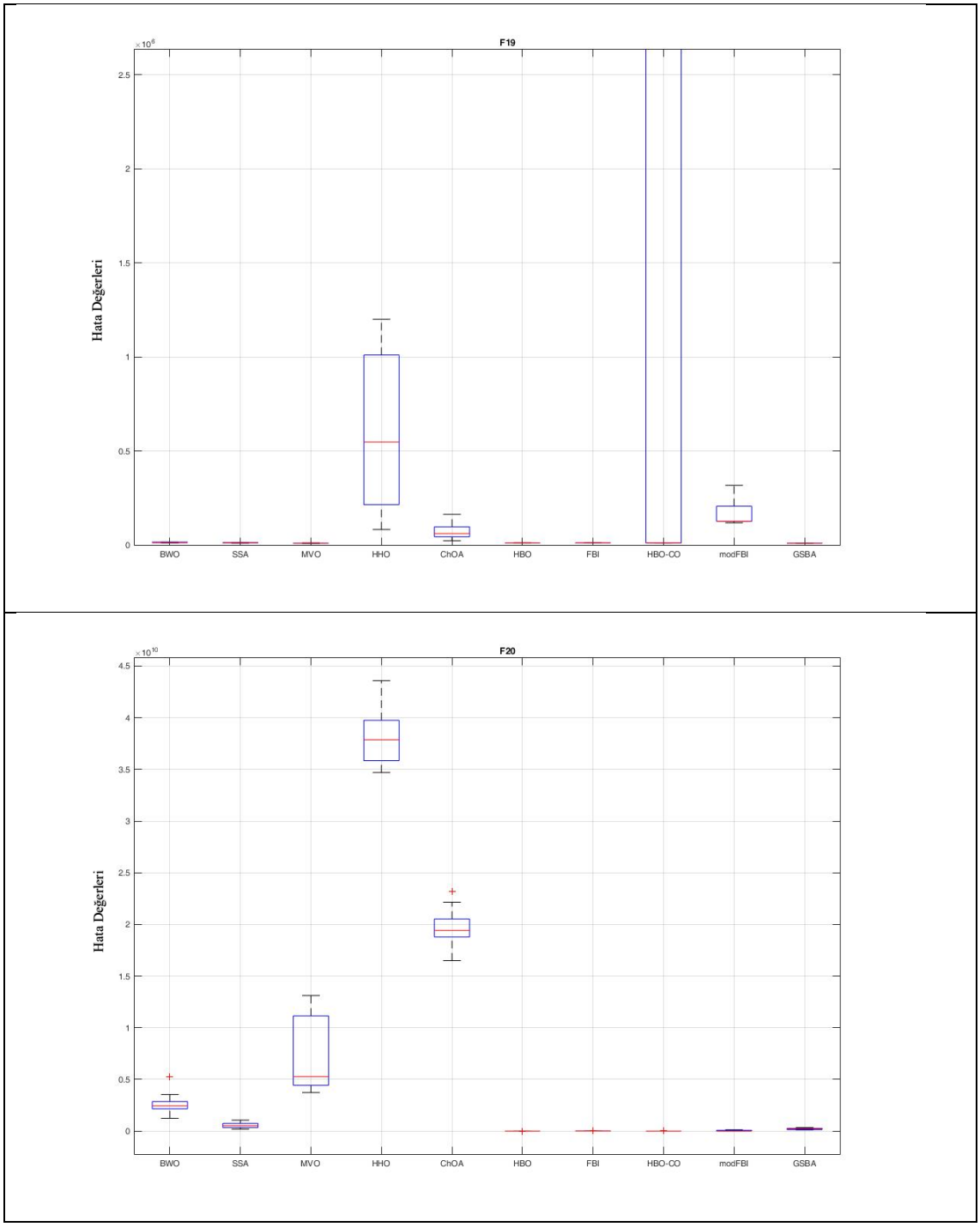

F14
×10^4
Hata Değerleri
0
0.5
1
1.5
2
2.5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA


F15
×10^4
Hata değerleri
0
1
2
3
4
5
6
7
8
9
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA


F16
×10^4
Hata Değerleri
2
4
6
8
10
12
14
16
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA


F17
Hata Değerleri
12000
10000
8000
6000
4000
2000
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA


F18
×10^4
Hata Değerleri
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
BWO
SSA
MVO
HHO
ChOA
HBO
FBI
HBO-CO
modFBI
GSBA



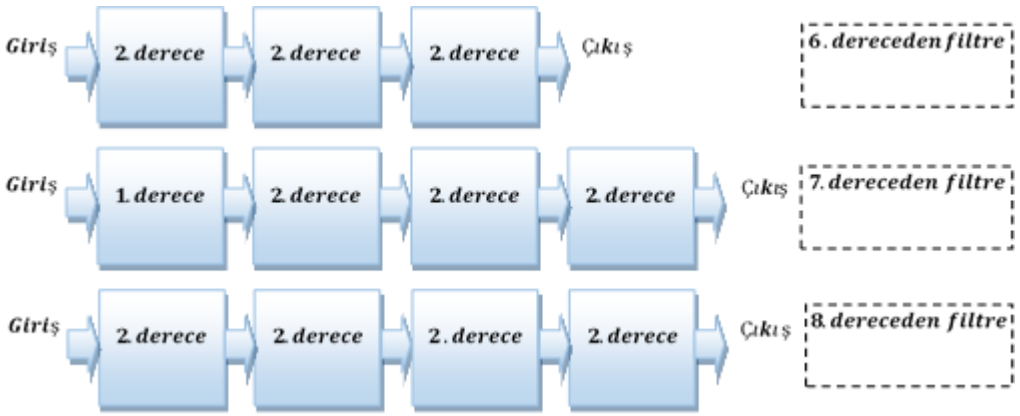
4.2. Gerçek Dünya Problemleri

Bu alt bölümde, önerilen gerçek dünya problemlerinin açıklamaları yapılacaktır. Bölüm 4'te geliştirilen ve kıyaslanan algoritmalar, aynı parametre değerlerinde, 300 iterasyon ve 30 bağımsız çalıştırma değerleri ile gerçek dünya problemlerine uygulanarak test

edilecektir. Sadece analog filtre tasarım problemlerinde 5000 iterasyon kullanılmıştır. Gerçek dünya problemleri R1-R22 aralığındaki numaralarla açıklanmıştır.

R1 Analog filtre tasarımı

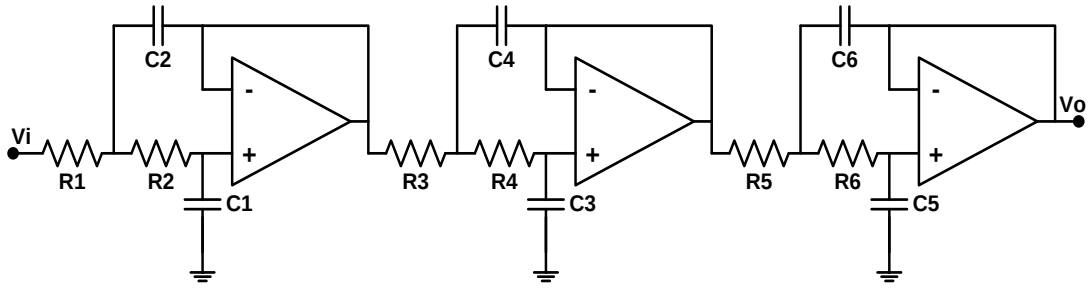
Analog filtreler, "Electronic Industries Association (EIA)" tarafından tanımlanan bir dizi pasif bileşenle tasarlanabilir. Bu üretilen bileşenler serisi, E12, E24, E48 ve E96 gibi tolerans değerlerine göre sınıflandırılır. Efektif bir tasarım yapabilmek için direnç ve kondansatör değerleri bu serilerden seçilebilmektedir. Filtre tipine bağlı olarak, bir dizi farklı pasif bileşen vardır ve seçilen seri altında mümkün olan bir dizi farklı bileşen kombinasyonu seçimi oluşmaktadır. Burada çözülmesi gereken problem, seçilen seri içerisinde tanımlanan pasif bileşen değerlerinin optimum kombinasyonunun seçilmesidir. Devredeki bileşen kombinasyonlarının sayısı, filtre derecesine göre arttığında, bu sorunun çözülmesini çok daha zor hale gelmektedir. Örnek olarak, sekiz pasif devre bileşeninden oluşan Gerilim Kontrollü Gerilim Kaynağı (VCVS) topolojisi kullanılarak, metasezgisel algoritmalarla dördüncü dereceden bir Butterworth (BW) filtresi tasarlandığında, problemin boyutu 16 olacaktır. Öte yandan, aynı topolojiye sahip sekizinci dereceden bir BW filtresi tasarlandığında, problem tanımında dikkate alınması gereken 32 boyut olacaktır. Bu nedenle, en uygun pasif bileşenleri belirlemek için gün geçtikçe daha karmaşık algoritmalar geliştirilmektedir. Diğer yandan, düşük dereceden filtrelerin kaskat bağlanmasıyla yüksek dereceden analog filtreler elde edilebilmektedir (Mancini 2002). Şekil 4.1'de aktif analog filtrelerin yüksek dereceden filtreleri elde etmede kaskat bağlanmasına ilişkin blok diyagramı verilmiştir.



Şekil 4.1. Yüksek dereceden analog filtrelerin kaskat yapı blokları

R1.1 Altıncı dereceden Butterworth filtre tasarımı

Altıncı dereceden VCVS BW alçak geçiren filtre topolojisi Şekil 4.2'de verilmektedir. Şekil 4.2'de gösterildiği gibi, filtre, üç adet ikinci dereceden BW alçak geçiren filtre bloğuna kaskat bağlanarak elde edilir. Filtrenin transfer fonksiyonu Denklem (4.1)'de verilmektedir. Bu denklemde, ω_{c_1}, ω_{c_2} ve ω_{c_3} ikinci derece blokların her birinin kesim frekansları ve Q_1, Q_2 ve Q_3 ikinci derece blokların her birinin kalite faktörleridir.



Şekil 4.2. Altıncı dereceden alçak geçiren Butterworth filtre

$$H(s) = \frac{\omega_{c_1}^2}{s^2+\frac{\omega_{c_1}}{Q_1}s+\omega_{c_1}^2} \times \frac{\omega_{c_2}^2}{s^2+\frac{\omega_{c_2}}{Q_2}s+\omega_{c_2}^2} \times \frac{\omega_{c_3}^2}{s^2+\frac{\omega_{c_3}}{Q_3}s+\omega_{c_3}^2} \quad (4.1)$$

İkinci derece blokların her birinin kesim frekansları, ilgili filtre bileşenlerine göre Denklem (4.2) ve Denklem (4.3)'te ifade edilmektedir.

$$\omega_{c_1} = \frac{1}{\sqrt{R_1R_2C_1C_2}} \quad , \quad \omega_{c_2} = \frac{1}{\sqrt{R_3R_4C_3C_4}} \quad , \quad \omega_{c_3} = \frac{1}{\sqrt{R_5R_6C_5C_6}} \quad (4.2)$$

$$Q_1 = \frac{\sqrt{R_1R_2C_1C_2}}{R_1C_1+R_2C_1} \quad , \quad Q_2 = \frac{\sqrt{R_3R_4C_3C_4}}{R_3C_3+R_4C_3} \quad , \quad Q_3 = \frac{\sqrt{R_5R_6C_5C_6}}{R_5C_5+R_6C_5} \quad (4.3)$$

Daha önceki çalışmalara benzer olarak (Dib ve El-Asir, 2018; De, Kar, Mandal ve Ghoshal, 2015a; 2015b; 2015c), toplam tasarım hatasını hesaplamak için kullanılan altıncı dereceden alçak geçiren Butterworth aktif filtresinin maliyet fonksiyonu (*CF*), Denklem (4.4)'te tanımlanmaktadır. Bu denklem, kesim frekans sapması ($\Delta\omega$) ve kalite faktörü sapmasının (ΔQ) ağırlıklı çarpımlarından oluşmaktadır.

$$CF = 0.5(\Delta\omega) + 0.5(\Delta Q) \quad (4.4)$$

$\Delta\omega$ ve ΔQ, her blok için kesim frekanslarına ve kalite faktörlerine göre, Denklem (4.5) ve Denklem (4.6)'daki gibi yeniden yazılabilmektedir.

$$\Delta\omega = \frac{|\omega_{c_1}-\omega_c|+|\omega_{c_2}-\omega_c|+|\omega_{c_3}-\omega_c|}{\omega_c} \tag{4.5}$$

$$\Delta Q = \left|Q_1 - \frac{1}{0.5176}\right| + \left|Q_2 - \frac{1}{1.4142}\right| + \left|Q_3 - \frac{1}{1.9319}\right| \tag{4.6}$$

Burada, ω_c 10 $krad/s$ büyüklüğünde hedef kesim frekansı, $1/0.5176$, $1/1.4142$ and $1/1.9319$ hedef kalite faktörleri olmaktadır (Mancini 2002). Şekil 4.3'te verilen çözüm vektörünün komponent elemanları denklemleri, Denklem (4.7)-(4.10)'da verilmektedir.

$$R_1 = A_1 \times 100 \times 10^{B_1}\ (\Omega)\ , \qquad R_2 = A_2 \times 100 \times 10^{B_2}\ (\Omega)\ ,$$
$$R_3 = A_3 \times 100 \times 10^{B_3}\ (\Omega) \tag{4.7}$$

$$R_4 = A_4 \times 100 \times 10^{B_4}\ (\Omega)\ , \qquad R_5 = A_5 \times 100 \times 10^{B_5}\ (\Omega)\ ,$$
$$R_6 = A_6 \times 100 \times 10^{B_6}\ (\Omega) \tag{4.8}$$

$$C_1 = D_1 \times 100 \times 10^{E_1}\ (pF)\ , \qquad C_2 = D_2 \times 100 \times 10^{E_2}\ (pF)\ ,$$
$$C_3 = D_3 \times 100 \times 10^{E_3}\ (pF) \tag{4.9}$$

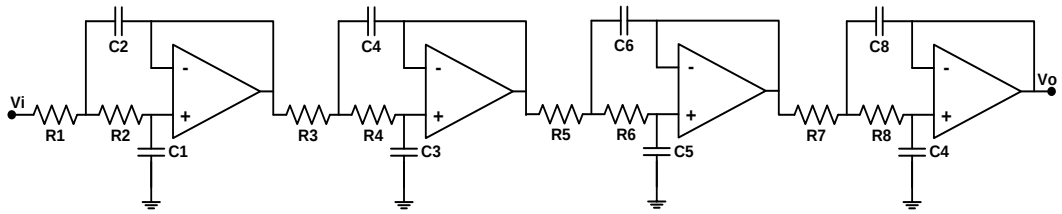
$$C_4 = D_4 \times 100 \times 10^{E_4}\ (pF)\ , \qquad C_5 = D_5 \times 100 \times 10^{E_5}\ (pF)\ ,$$
$$C_6 = D_6 \times 100 \times 10^{E_6}\ (pF) \tag{4.10}$$

A_1	B_1	A_2	B_2	A_3	B_3	A_4	B_4	A_5	B_5	A_6	B_6	D_1	E_1	D_2	E_2	D_3	E_3	D_4	E_4	D_5	E_5	D_6	E_6

Şekil 4.3. Altıncı dereceden alçak geçiren Butterworth filtre komponent vektör gösterimi

R1.2 Sekizinci dereceden Butterworth filtre tasarımı

Sekizinci dereceden bir VCVS alçak geçiren BW filtresi, Şekil 4.2'de gösterilen altıncı dereceden filtre tipinin sonuna bir ikinci dereceden blok eklenerek, Şekil 4.4'teki gibi tasarlanabilmektedir. Transfer fonksiyonu, Denklem (4.11)'de verilmektedir.



Şekil 4.4. Sekizinci dereceden alçak geçiren Butterworth filtre

$$H(s) = \frac{\omega_{c_1}^2}{s^2+\frac{\omega_{c_1}}{Q_1}s+\omega_{c_1}^2} \times \frac{\omega_{c_2}^2}{s^2+\frac{\omega_{c_2}}{Q_2}s+\omega_{c_2}^2} \times \frac{\omega_{c_3}^2}{s^2+\frac{\omega_{c_3}}{Q_3}s+\omega_{c_3}^2} \times \frac{\omega_{c_4}^2}{s^2+\frac{\omega_{c_4}}{Q_4}s+\omega_{c_4}^2} \tag{4.11}$$

Denklem (4.4) ile verilen, *CF* hata fonksiyonunda kullanılacak kesim frekansı ve kalite faktörü sapmaları Denklem (4.12) ve Denklem (4.13)'teki gibi olmaktadır.

$$\Delta\omega = \frac{|\omega_{c_1}-\omega_c|+|\omega_{c_2}-\omega_c|+|\omega_{c_3}-\omega_c|+|\omega_{c_4}-\omega_c|}{\omega_c} \tag{4.12}$$

$$\Delta Q = \left|Q_1 - \frac{1}{0.3902}\right| + \left|Q_2 - \frac{1}{1.1111}\right| + \left|Q_3 - \frac{1}{1.6629}\right| + \left|Q_4 - \frac{1}{1.9616}\right| \tag{4.13}$$

Burada ω_{c_1}, ω_{c_2}, ω_{c_3} ve ω_{c_4}ikinci dereceden blokların kesim frekansıdır. ω_c 10 $krad/s$ değerli hedef kesim frekansıdır, Q_1, Q_2, Q_3 ve Q_4 $1/0.3902$, $1/1.1111$, $1/1.6629$ ve $1/1.9616$ değerli hedef kesim frekanslarıdır (Mancini 2003). Komponent elemanları, Denklem (4.14) ve Denklem (4.15)'teki gibi ifade edilmektedir.

$$\left.\begin{array}{l} R_1 = A_1 \times 100 \times 10^{B_1}\ (\Omega)\ ,\ R_2 = A_2 \times 100 \times 10^{B_2}\ (\Omega) \\ R_3 = A_3 \times 100 \times 10^{B_3}\ (\Omega)\ ,\ R_4 = A_4 \times 100 \times 10^{B_4}\ (\Omega) \\ R_5 = A_5 \times 100 \times 10^{B_5}\ (\Omega)\ ,\ R_6 = A_6 \times 100 \times 10^{B_6}\ (\Omega) \\ R_7 = A_7 \times 100 \times 10^{B_7}\ (\Omega)\ ,\ R_8 = A_8 \times 100 \times 10^{B_8}\ (\Omega) \end{array}\right\} \tag{4.14}$$

$$\left.\begin{array}{l} C_1 = D_1 \times 100 \times 10^{E_1}\ (pF)\ ,\ C_2 = D_2 \times 100 \times 10^{E_2}\ (pF) \\ C_3 = D_3 \times 100 \times 10^{E_3}\ (pF)\ ,\ C_4 = D_4 \times 100 \times 10^{E_4}\ (pF) \\ C_5 = D_5 \times 100 \times 10^{E_5}\ (pF)\ ,\ C_6 = D_6 \times 100 \times 10^{E_6}\ (pF) \\ C_7 = D_7 \times 100 \times 10^{E_7}\ (pF)\ ,\ C_8 = D_8 \times 100 \times 10^{E_8}\ (pF) \end{array}\right\} \tag{4.15}$$

R2 Harmonik eliminasyon problemi

DC gerilimi AC gerilime çeviren dönüştürücü, evirici olarak adlandırılmaktadır. Bu çalışmada, harmonik eliminasyonu gerçekleştirmek amacıyla anahtarlama açılarını kontrol etmek için tek fazlı yarım köprü ve tek fazlı tam köprü eviriciler kullanılmıştır. Çıkış gerilimi, Fourier serisi kullanılarak, Denklem (4.16)'daki gibi ifade edilebilmektedir (Rajaram, Palanisamy, Ramasamy, ve Ramanathan, 2015; Babu, Priya, Maheswaran, Kumar, ve Rajasekar, 2015).

$$V_0 = a_o + \sum_{k=1}^{n} A_n \, cos(\omega nt) + \sum_{k=1}^{n} B_n \, sin(\omega nt) \tag{4.16}$$

Burada n bir tamsayıdır ve tek fazlı yarım köprü eviricisi için B_n değeri, Denklem (4.17)'deki gibi hesaplanır (Vatansever ve Kuyu, 2019).

$$B_n\left(\alpha_{1,}\alpha_{2,\ldots.}\alpha_{k-1},\alpha_k\right) = \frac{4V_{dc}}{n\pi}\sum_{k=1}^{k}(-1)^{k+1}\ [cos(n)\alpha_k] \tag{4.17}$$

B_n değeri tek fazlı tam köprü eviricisi için Denklem (4.18)'deki gibi hesaplanmaktadır (Babu, Priya, Maheswaran, Kumar, ve Rajasekar, 2015).

$$B_n\left(\alpha_{1,}\alpha_{2,\ldots.}\alpha_{k-1},\alpha_k\right) = \frac{4V_{dc}}{n\pi}(1 + 2\sum_{k=1}^{k}(-1)^{k+1}\ [cos(n)\alpha_k]) \tag{4.18}$$

$B_3, B_5, \ldots B_{2k-1}$ harmonik komponentlerin değerlerini bulmada, çıkış geriliminin sıfır veya mümkün olduğunca düşük tutulmasının istendiği, yarım çevrim başına *k* darbeleri için evirici çıkış gerilimi dalga formunun anahtarlama açıları $\alpha_{1,}\, \alpha_{2.} \ldots \alpha_{k-1}, \alpha_k$ olarak kullanılmaktadır. Hedef fonksiyon F ve sınır koşulu, Denklem (4.19) ve (4.20)'de belirtilmektedir (Patel, ve Hoft, 1973).

$$F(\alpha) = |B_3| + |\,B_5| + \cdots + |B_{2k-1}| \tag{4.19}$$

$$\alpha_1 \leq \alpha_{2,} \leq \cdots \leq \frac{\pi}{2} \tag{4.20}$$

Her iki tipteki eviricide, 5. ve 7. harmonikler minimize edilmek istenmektedir.

R3 IIR filtre tasarımı

Teknolojideki hızlı gelişmelerle sayısal işaret işlemede, sayısal filtreler sıklıkla kullanılan yapılar arasında yer almaktadır. Sayısal IIR filtreler, Denklem (4.21)'deki gibi fark formülasyonuyla hesaplanabilmektedir (Ifeachor ve Jervis, 2002).

$$y(k) + \sum_{j=1}^{N} b_j\, y(k-j) = \sum_{j=0}^{M} a_j x(k-j) \tag{4.21}$$

Burada, a ve b, filtre katsayılarını; $N\ (\geq M)$, filtrenin derecesini belirtmektedir. Filtrenin genel transfer fonksiyonu Denklem (4.22)'deki gibi yazılabilmektedir.

$$H(z) = \frac{A(z)}{1+B(z)} = \frac{\sum_{j=0}^{L} a_j z^{-i}}{1+\sum_{j=1}^{M} b_j z^{-i}} \tag{4.22}$$

Paydaki, $\{a_j\}_{j=0}^{L}$ ve paydadaki $\{b_j\}_{j=1}^{M}$ katsayılarından oluşan katsayılar vektörü, Denklem (4.23)'teki gibi tanımlanabilmektedir.

$$K = [a_0, \dots, a_{L,} b_1, \dots, b_M] \tag{4.23}$$

Hedef fonksiyon $J(K)$, ortalama karesel hata denklemi aracılığıyla, Denklem (4.24)'te verilmektedir.

$$J(K) = \frac{1}{L}\sum_{k=1}^{L} e(k)^2 \tag{4.24}$$

$e(k) = d(k) - y(k)$ ile ifade edilmekte; $d(k)$, filtrenin istenen cevabını, L örnek sayısını, J minimize edeceğimiz hata fonksiyonunu temsil etmektedir.

Chebyshev Tip I alçak geçiren ve Chebyshev Tip II yüksek geçiren filtrelerin uygulama örnekleri, kıyaslanan algoritmalarla karşılaştırılarak verilmektedir. Filtrelere ilişkin başlangıç şartları, Çizelge 4.10'da gösterilmektedir.

Çizelge 4.10. Örnek tasarımlara ilişkin özellikler/parametreler

Özellik/Parametre	Örnek tasarım 1	Örnek tasarım 2
Fitre türü	Chebyshev Tip I	Chebyshev Tip II
Filtre tipi	Alçak geçiren filtre	Yüksek geçiren filtre
Örnekleme frekansı	10 kHz	10 kHz
Geçirme bandı kesim frekansı	2 kHz	2.2 kHz
Durdurma bandı kesim frekansı	2.2 kHz	2 kHz
Geçirme bandı dalgalanması	0.1 dB	0.1 dB
Durdurma bandı zayıflaması	30 dB	30 dB

R4 Dairesel anten dizisi tasarım problemi

Dairesel şekilli anten dizilerinin, sonar, radar, mobil ve ticari uydu haberleşme sistemlerinde çeşitli uygulama alanları bulunmaktadır (Dessouky, Sharshar ve Albagory, 2006; Gurel ve Ergul, 2008). x-y düzleminde *r* yarıçaplı bir daire üzerinde aralıklı *N* anten elemanı olduğu düşünülsün. Anten elemanları, dairesel bir anten dizisi oluşturmaktadır ve bu dairesel dizi için dizi faktörü, Denklem (4.25)'teki gibi yazılabilmektedir.

$$AF(\emptyset) = \sum_{n=1}^{N} I_n \exp[jkr(\cos(\emptyset - \emptyset_{ang}^{n}) - \cos(\emptyset_0 - \emptyset_{ang}^{n})) + B_n] \qquad (4.25)$$

Burada;

- $Q_{ang}^{n} = 2\pi(n-1)/N$ x-y yüzeyindeki *n.* elementin açısal pozisyonu,
- $Kr = Nd$, *k* dalga numarası, *d* ise *r* yarıçaplı çember ile elemanların açısal uzaklığı,
- Q_0 maksimum radyasyonun yönü,
- $\emptyset$ düzlem dalgasının insidans açısı,
- I_n akım uyarımı,
- B_n *n.* elemanın faz uyarımı olmaktadır.

Hedef fonksiyon *OF,* Denklem (4.26)'da ifade edilmektedir.

$$OF = \frac{|AR(\varphi_{sll},I,B,\varphi_0)|}{|AR(\varphi_{max},I,B,\varphi_0|} + \frac{1}{DIR(\varphi_0,I,B)} + \sum_{k=1}^{num} |AR(\varphi_k, I, B, \varphi_0)| \qquad (4.26)$$

Burada ilk bileşen, yan lobları bastırmaya çalışmaktadır. φ_{sll} maksimum yan lob seviyesine ulaşıldığı açı olmaktadır. İkinci bileşen, dizi modelinin yönelimini en üst düzeye çıkarmaya çalışmaktadır. Üçüncü bileşen, dizi modelinin maksimumunu istenen maksimum φ_{des}'e yaklaştırmaya çalışmaktadır. Denklemde *num*, null kontrol yönleri sayısıdır ve φ_k ise *k.* "null" kontrol yönüdür (Das ve Suganthan, 2010).

R5 Frekans modülasyonlu (FM) ses dalgaları için parametre tahmini

Frekans Modülasyonlu (FM) ses dalgasının sentezi, birçok modern müzik sisteminde çok önemli bir rol oynamaktadır. Optimize edilecek değişkenlerin sırası, Denklem (4.27)'deki gibi altı boyutlu bir vektör olarak tanımlanabilmektedir.

$$X = \{a_1, w_1, a_2, w_2, a_3, w_3\} \tag{4.27}$$

Tahmini ses ve hedef ses dalgalarının tanımları, Denklem (4.28) ve Denklem (4.29)'da ifade edilmektedir. (Das ve Suganthan, 2010).

$$y(t) = a_1 sin\{\omega_1 t\theta + a_2 sin(\omega_2 t\theta + a_3\, sin(\omega_3 t\theta))\} \tag{4.28}$$

$$y_0(t) = sin\{5t\theta - 1.5 sin(4.8t\theta + 2\, sin(4.9t\theta))\} \tag{4.29}$$

Burada $\theta = 2\pi/100$ ve parametreler $[-6.4\ 6.35]$ aralığında olmaktadır. Hedef fonksiyon Denklem (4.30) 'da ifade edilmektedir.

$$F(\vec{X}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \tag{4.30}$$

R6 Yayılmış spektrumlu radar çok fazlı kod tasarımı

Darbe sıkıştırma kullanan bir radar sistemi tasarlarken, uygun dalga biçiminin seçimine çok dikkat edilmelidir. Darbe sıkıştırmasını mümkün kılan birçok radar darbe modülasyonu yöntemi bilinmektedir. Çok fazlı kodlar, sıkıştırılmış işaretteki diğer alt yan loblar ve sayısal işleme tekniklerinin daha kolay kullanımı sebebiyle, dikkat çekici olmaktadır. Söz konusu problem, sürekli değişkenlerde ve çok sayıda yerel optimale sahip bir min-maks doğrusal ve dışbükey olmayan optimizasyon problemi olarak

modellenebilmekte ve Denklem (4.31) ve (4.32)'deki gibi ifade edilebilmektedir (Dukic ve Dobrosavljevic, 1990).

$$global\ min_{x\epsilon X} f(x) = \max\{\emptyset_1(x), \dots, \emptyset_{2m}(2m)\} \quad (4.31)$$

$$X = \{(x_1, \dots, x_n)\epsilon R^n \mid 0 \le x_j \le 2\pi, j = 1, \dots, n\} \quad (4.32)$$

Burada $m = 2n - 1$ ve

$$\phi_{2i-1}(x) = \sum_{j=1}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} X_k\right), \ \ i = 1, \dots, n \quad (4.33)$$

$$\phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j|+1}^{j} X_k\right), \ \ i = 1, \dots, n-1 \quad (4.34)$$

$$\phi_{m+i}(x) = \phi_i(x), \ \ i = 1, \dots, m \quad (4.35)$$

olmaktadır. Burada amaç, değişkenler simetrik faz farklılıklarını temsil ederken, optimal alıcı çıkışında, sıkıştırılmış radar darbesinin karmaşık zarfıyla ilgili olan oto korelasyon fonksiyonunun örnekleri arasında en büyüğünün modülünü minimize etmektir (Das ve Suganthan, 2010).

R7 Doğrusal olmayan karıştırmalı tank reaktörünün optimum kontrolü

Sürekli karıştırılan bir tank reaktöründe (CSTR) gerçekleştirilen birinci dereceden tersinmez bir kimyasal reaksiyon, çok modlu bir optimal kontrol problemidir. Bu kimyasal işlem, iki doğrusal olmayan diferansiyel denklem ile Denklem (4.36) ve Denklem (4.37)'deki gibi modellenmektedir.

$$\dot{x}_1 = -(2 + u)(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) \quad (4.36)$$

$$\dot{x}_2 = -0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right) \quad (4.37)$$

Denklem (4.36) ve Denklem (4.37)'de;

- $u(t)$ = Soğutma sıvısının akış hızı,
- x_1 = Boyutsuz kararlı durum sıcaklığı,
- x_2 = Boyutsuz kararlı durum konsantrasyonundan sapma olmaktadır.

Optimizasyonun amacı, performans endeksinin uygun *u* değerini belirlemektir.

$$J = \int_0^{t_f=0.72} (x_1^2 + x_2^2 + 0.1u^2)dt \quad (4.38)$$

Denklem (4.38)'de, minimize edilmek istenmektedir. Başlangıç koşulu, $x(0) = [0.009\ 0.09]$ verilmektedir (Das ve Suganthan, 2010).

R8 İletim ağı genişletme planlaması problemi

Planlama boyunca, güvenlik kısıtlamaları olmayan temel iletim ağı genişletme planlaması (TNEP), inşa edilecek yeni hat setini belirlemektedir. Genişletme planı maliyetinin minimum olması istenmektedir ve aşırı yükler üretilmemektedir. TNEP problemi, Denklem (4.39)-(4.42) ile tanımlanabilmektedir.

$$min(y) = \sum_{l\in\Omega} c_l n_l \quad (4.39)$$

$$Sf + g = d \quad (4.40)$$

$$f_l - \gamma_l\left(n_l^0 + n_l\right)(\Delta\emptyset_l) = 0, \quad for\ l\epsilon 1{,}2\ \ldots. \quad (4.41)$$

$$|f_l| \le \left(n_l^0 + n_l\right)\overline{f_l}\ , \ \ for\ l \in 1{,}2, \ldots, nl \quad (4.41)$$

$$0 \le n_l \le \overline{n_l} \quad (4.42)$$

Denklem (4.39)-(4.42)'de:

- $n_l \ge 0$ ve $l \in 1{,}2 \ldots nl$ icin tam sayıdır,
- $c_l, l.$ yola eklenen maliyeti,
- S, güç sisteminin dal-düğüm insidans transpoze matrisini,
- f, f_i elementinin bir vektörünü,
- $\gamma_l,\ l.$ yola eklenen devrenin duyarlılığını,
- $n_{l,}\ \ l.$ yola eklenen devre sayısını,
- $n_l^0,$ en iyi durumdaki devre sayısını,
- $\Delta\emptyset_l\ l.$ yola eklenen faz açısı farkını,
- $f_l,\ \ l.$ yol ekli devredeki toplam güç akışını,

- $\overline{f_l}$ l. yol ekli devrede kabul edilen maksimim reel güç akışını,
- $\overline{n_l}$, l. yola eklenebilen maksimum devre sayısını,
- Ω, tüm eklenebilecek yol setlerini,
- nl, devredeki toplam yol sayısını ifade etmektedir.

Amaç, inşa edilen yeni iletim hatlarının toplam yatırım maliyetini en aza indirmektir. TNEP sorununun hedef fonksiyonu, *ol*'nin aşırı yüklenmiş çizgiler kümesi olduğu Denklem (4.43)'te gösterilmektedir (Das ve Suganthan, 2010).

$$F = \sum_{I\in\Omega} c_l n_l + W_l \sum_{ol}(abs(f_l) - \overline{f_l}) + W_2(n_l - \overline{n_l}) \quad (4.43)$$

R9 Kaynaklı kiriş tasarımı problemi

Kaynaklı kiriş tasarımında amaç, kesme gerilmesi (r), eğilme gerilmesi (σ), burkulma yükü (Pc) ve uç sapma (δ) sınırlarına göre, yapının toplam imalat maliyetini en aza indirmek için en uygun tasarım değişkenleri kümesini bulmaktır. Şekil (4.5)'te, $X_1 = h$, $X_2 = l$, $X_3 = t$, ve $X_4 = b$, hedef fonksiyon F, sınırmalarıyla birlikte Denklem (4.44)-(4-50)'de verilmektedir (Rao, 1996).

$$F(x) = 1.10471X_1^2X_2 + 0.04811X_3X_4(X_2 + 14) \quad (4.44)$$

Sınırlamalar:

$$g(1) = r(x) - r_{max} \leq 0 \quad (4.45)$$

$$g(2) = \sigma(x) - \sigma_{max} \leq 0 \quad (4.46)$$

$$g(3) = X_1 - X_4 \leq 0 \quad (4.47)$$

$$g(4) = 1.10471X_1^2X_2 + 0.04811X_3X_4(X_2 + 14) - 5 \leq 0 \quad (4.48)$$

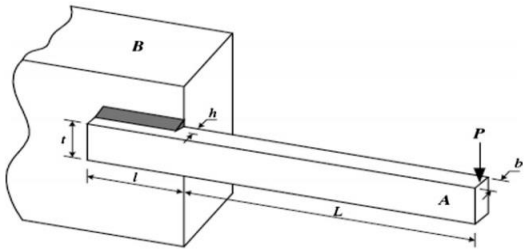
$$g(5) = 0.125 - X_1 \leq 0 \quad (4.48)$$

$$g(6) = \delta(x) - \delta_{max} \leq 0 \quad (4.49)$$

$$g(7) = P - Pc(x) \leq 0 \quad (4.50)$$

Tasarım değişkenlerinin sınırları Denklem (4.51)'de aşağıda verilmektedir.

$$0.1 \leq X_1 \leq 2,\ 0.1 \leq X_2 \leq 10,\ 0.1 \leq X_3 \leq 10,\ 0.1 \leq X_4 \leq 2 \tag{4.51}$$



Şekil 4.5. Kaynaklı kiriş yapısı (Kaveh ve Mahdavi, 2014).

R10 Germe/sıkıştırma yay tasarım problemi

Bu problem, minimum sapma, kesme gerilimi, darbe frekansı, dış çap limitleri ve tasarım değişkenleri üzerindeki kısıtlamalara tabi olan bir çekme-sıkıştırma yayının ağırlığının en aza indirilmesini amaçlamaktadır. Tasarım değişkenleri ortalama bobin çapı D, tel çapı d ve aktif bobin sayısı N olmaktadır. Denklem (4.52) tasarım hedef fonksiyonu F'i göstermekte ve problemin sınırları Denklem (4.53)- (4.56) vasıtasıyla sağlanmaktadır (Coello, 2000).

$$F(x) = (N+2)Dd^2 \tag{4.52}$$

Sınırlamalar:

$$g(1) = 1 - \frac{D^3 N}{71.785d^4} \leq 0 \tag{4.53}$$

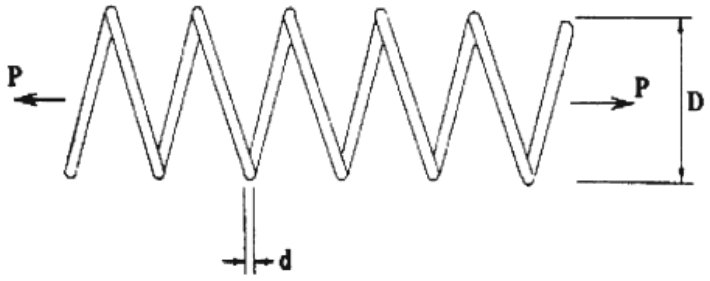
$$g(2) = \frac{4D^2 - dD}{12.566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \leq 0 \tag{4.54}$$

$$g(3) = 1 - \frac{140.45d}{D^2 N} \leq 0 \tag{4.55}$$

$$g(4) = \frac{D+d}{1.5} - 1 \leq 0 \tag{4.56}$$

Tasarım değişkenlerinin limitleri Denklem (4.57) ile verilmektedir. Temsili şekli Şekil (4.6)'da verilmektedir.

$$0.05 \leq d \leq 2,\ 0.25 \leq D \leq 1.3,\ 2 \leq N \leq 15 \tag{4.57}$$



Şekil 4.6. Germe yay yapısı (Coello, 2000).

R11-R12 Dinamik ekonomik yük dağıtımı (DED) problemi

Dinamik ekonomik yük dağıtımı (DED) problemi, saatlik dağıtım probleminin özelliklerini takip eder; ancak burada güç talebi her saate göre değişir ve 24 saatlik elektrik üretim programı belirlenmelidir. Üretim maliyetine karşılık gelen objektif fonksiyon, üretim birimlerinden gelen aktif güç çıkışlarının ikinci dereceden bir fonksiyonu olarak gösterilebilmekte ve Denklem (4.58) ile temsil edilmektedir.

$$Min(F_c) = \sum_{k=1}^{T} \sum_{i=1}^{N_G} F_{ih}(P_{ih}) \tag{4.58}$$

Burada,

$$F_{it}(P_{it}) = a_i P_{it}^2 + b_i P_{it} + c_i, \quad i = 1,2,\dots,N_G \tag{4.59}$$

olmaktadır. $a_i, b_i \; ve \; c_i$ maliyet katsayıları, P_{it} $i.$ jeneratörün $t.$ anında reel güç çıkışı (MW), N_G dağıtıma katılacak aktif ünite sayısı, T yük dağıtımının toplam süresini ifade etmektedir. Valf noktası yükleme etkisine sahip ünitenin maliyet fonksiyonu Denklem (4.60)'daki gibi hesaplanmaktadır.

$$F_{it}(P_{it}) = a_i P_{it}^2 + b_i P_{it} + c_i + |e_i \sin(f_{it}(P_{it}{}^{min} - P_{it}))| \tag{4.60}$$

$e_i \; ve \; f_i$ valf etkisiyle ilişkili maliyet katsayıları olmaktadır. DED problemi yapısı itibariyle birçok kısıtlamalara sahiptir. Enerji dengesi, rampa oranı sınırları ve yasaklı çalışma bölgeleri bunlar arasında gösterilebilmektedir. Enerji dengesi kısıtlaması, toplam sistem üretimi ile toplam sistem yükleri (PD) ve kayıplar (PL) arasındaki denge prensibine dayanmaktadır. Denklem (4.61)'deki gibi gösterilebilmektedir.

$$\sum_{i=1}^{N_G} P_{it} = P_{Dt} + P_{Lt} \quad (4.61)$$

Denklem (4.61)'de, P_{Lt}, B katsayıları aracılığıyla Denklem (4.62)'deki gibi elde edilmektedir.

$$P_{Lt} = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} P_{it} B_{ij} P_{jt} \quad (4.62)$$

Her bir üretim ünitesinin çıkış gücü, bu sınırlar arasında kalacak şekilde, bir alt ve üst sınıra sahiptir. Bu kısıtlama, Denklem (4.63)'teki gibi bir çift eşitsizlik kısıtlamasıyla temsil edilmektedir.

$$P_i^{min} \leq P_{it} \leq P_i^{max} \quad (4.63)$$

Burada P_i^{min} ve P_i^{max}, i. jeneratör ünitesinin alt ve üst sınırları olmaktadır. Rampa oranı limiti, jeneratörün çalışmasını iki çalışma periyodu arasında ayarlamak için tüm çevrimiçi ünitelerin çalışma aralığını sınırlamaktadır. Üretim, karşılık gelen üst ve aşağı rampa oranı sınırlarıyla artabilir veya azalabilmektedir. Bu nedenle üniteler, Denklem (4.64)'teki gibi bu rampa oranı nedeniyle sınırlandırılmaktadır.

$$\max(P_i^{min}, UR_i - P_i) \leq P_i \leq \min(P_i^{max}, P_i^{t-1} - DR_i) \quad (4.64)$$

Burada P_i^{t-1}, i. ünitenin önceki zamandaki güç üretimi, UR_i ve DR_i ise alt ve üst sınırları olmaktadır. Ünite ve sistem kısıtlarını karşılarken, yakıt maliyetlerini en aza indirmek amacıyla popülasyondaki her bireyin uygunluğunu değerlendirmek için, bu çalışmada Denklem (4.65), uygunluk fonksiyonu modeli *OF* benimsenmektedir.

$$OF = \sum_{t=1}^{n} \sum_{i=1}^{N} F_i(P_{it}) + c_1\{\sum_{t=1}^{n} \sum_{i=1}^{N} P_{it} - P_{Dt}\}^2 + c_2\{\sum_{t=1}^{n} \sum_{i=1}^{N} P_{it} - P_{rlim}\}^2 \quad (4.65)$$

Denklem (4.65)'te, c_1 ve c_2 ceza parametreleri, *n* saat sayısı ve *N* ünite sayısı olmaktadır (Das ve Suganthan, 2010).

R13-R17 Statik ekonomik yük dağıtımı (DED) problemi

Statik ELD problemi, işletim üniteleri arasında optimal üretim dağıtımını gerçekleştirmek ve karşılığında sistem yük talebi, yasak çalışma bölgeleri ve rampa oranı gibi kısıtlamaları karşılamak için, belirli bir işletme süresinde üretim birimlerinin yakıt maliyetini en aza

indirmeyi içermektedir. Bu nedenle, ELD için iki alternatif model, Denklem (4.66) ve Denklem (4.67)'de verilmektedir.

$$Min(F) = \sum_{i=1}^{N\_G} f_i(P_i) \tag{4.66}$$

$$f_i(P_i) = a_i P_i^2 + b_i P_i + c_i, \quad i = 1,2 \dots, N_G \tag{4.67}$$

Burada a_i, b_i ve c_i maliyet katsayıları; P_i, i jeneratörün t anında reel güç çıkışı (MW); N_G aktif jeneratör ünite sayısını ifade etmektedir. Valf noktası yükleme etkisi uygunluk fonksiyonu Denklem (4.68)'deki gibi ifade edilmektedir.

$$f_i(P_i) = a_i P_i^2 + b_i P_i + c_i + |e_i \sin(f_i(P_i^{min} - P_i))| \tag{4.68}$$

Burada e_i ve f_i, valf noktası yükleme etkisine ilişkin maliyet katsayılarıdır. ELD problemi çeşitli kısıtlamalar içermektedir. Bunlar, enerji dengesini hesaba katacak güç dengesi kısıtı, rampa oranı kısıtı ve yasaklanmış çalışma bölgelerini içermektedir. Güç dengesi kısıtı, toplam sistem üretimi, toplam sistem yükü (P_D) ve kayıplar (P_L) göz önüne alınarak, Denklem (4.69)'daki gibi yazılabilmektedir.

$$\sum_{i=1}^{N_G} P_i = P_D + P_L \tag{4.69}$$

Burada P_{Lt}, B katsayıları aracılığıyla Denklem (4.70)'teki gibi elde edilmektedir.

$$P_L = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} P_i B_{ij} P_j + \sum_{i=1}^{N_G} B_{oi} P_i + B_{00} \tag{4.70}$$

Her bir üretim ünitesinin çıkış gücü, bir alt ve üst sınıra sahiptir. Bu kısıtlama, Denklem (4.71)'deki gibi bir çift eşitsizlik kısıtlamasıyla temsil edilmektedir.

$$P_i^{min} \leq P_i \leq P_i^{max} \tag{4.71}$$

Burada P_i^{min} ve P_i^{max}, i. jeneratör ünitesinin alt ve üst sınırları olmaktadır. Rampa oranı sınırlaması DED problemiyle aynı şekilde kullanılmaktadır. Üretim üniteleri, makine parçalarının fiziksel sınırlamaları nedeniyle, çalışmalarının kısıtlandığı belirli bölgelere sahip olabilmektedirler. Bu bölgelerde üretimden tasarruf etmek için operasyondan kaçınmak gerekmektedir. Bu yasak çalışma bölgeleri, bir üretim ünitesi i için, Denklem (4.72)'de gösterilmektedir.

$$P_i \leq \widetilde{P^{pz}} \; ve \; P_i \geq \widehat{P^{pz}} \tag{4.72}$$

Burada, $\widetilde{P^{pz}}$ ve $\widehat{P^{pz}}$, ünite i için verilen yasaklı bölgenin alt ve üst sınırları olmaktadır (Das ve Suganthan, 2010).

R18-R20 Hidrotermal çizelgeleme problemi

Hidrotermal çizelgeleme, hem kısa vadeli hem de uzun vadeli problem tipinde olabilmektedir. Kısa vadeli, tipik olarak 24 saatlik süreyi ifade ederken, uzun vadeli haftalar veya aylar süresini ifade edebilmektedir. Bir hidrotermal güç sisteminin kısa vadeli çizelgelemesinin birincil amacı, bir günlük veya birkaç günlük çizelgeleme süresi içinde, hidrolik sistem ve güç sistem ağındaki sınırlamalara uyacak şekilde, sistemdeki termal ve hidro ünitelerin güç üretimlerini planlamaktır. Ele alınan sistem, dört hidro santralden ve eşdeğer bir termik santralden oluşan çok zincirli kademeli bir ağ yapısındadır. Birden fazla ısıl ünite varsa, bunlar birlikte alınabilir ve eşdeğer bir ısıl ağ olarak kabul edilebilmektedir. Termal sistemin, çizelgeleme periyodundaki yük taleplerini karşılayacak şekilde çalışması için toplam yakıt maliyeti, F ile verilmekte ve Denklem (4.73)'de ifade edilmektedir.

$$Min(F) = \sum_{i=1}^{M} f_i(P_{Ti}) \tag{4.73}$$

Burada, f_i, i. aralıktaki P_{Ti} termal ünitesinin güç ünitesi yani hedef fonksiyonu olmaktadır. M kısa dönemli program için dikkate alınan toplam aralık sayısıdır. Hedef fonksiyon f_i Denklem (4.74)'teki gibi yazılabilmektedir.

$$f_i(P_{Ti}) = a_i P_{Ti}^2 + b_i P_{Ti} + c_i + |e_i \sin(f_i(P_{Ti}^{min} - P_{Ti}))| \tag{4.74}$$

Yukarıdaki problem tanımı birçok sınırlamaya sahip olmaktadır. Talep kısıtlaması, enerjinin korunumu ilkesinden kaynaklanmaktadır. Termal ünite ve hidro ünitelerin bir araya getirdiği toplam güç, hem güç talebini hem de meydana gelen güç kaybını karşılamalıdır. Bu durum Denklem (4.75)'te ifade edilmektedir.

$$P_{T(i)} + \sum_{k=1}^{N} P_{H(k,i)} = P_{D(i)} + P_{Loss(i)} \tag{4.75}$$

Burada $P_{H(k,i)}$, i. aralıktaki k. hidro ünitesinden üretilen gücü; $P_{D(i)}$ ve $P_{Losss(i)}$ i. aralıktaki güç talebi ve güç kaybını; N ise toplam hidro ünite sayısını ifade etmektedir. Eşdeğer termal jeneratör, herhangi bir i aralığında yalnızca belirli bir alt ve üst limitler arasında güç üretebilir. Bu durum, Denklem (4.76)'da gösterilmektedir.

$$P_T{}^{min} \leq P_{T(i)} \leq P_T{}^{max} \tag{4.76}$$

Hidroelektrik santralinin elektrik üretiminin her biri, işletme üst ve alt sınırlarına ait olmakta ve Denklem (4.77) ile ifade edilmektedir.

$$P_{H(k)}{}^{min} \leq P_{H(k,i)} \leq P_{H(k)}{}^{max} \tag{4.77}$$

Her bir deponun herhangi bir i aralığındaki hacmi, deponun en düşük ve en yüksek kapasite limitleri arasında yer almalıdır. Ayrıca, rezervuarların sahip olabilecekleri ilk ve son depolama hacmi üzerinde kısıtlamalar olmaktadır. Bu kısıtlar, Denklem (4.78) ve (4.79) ile ifade edilmektedir.

$$V_{(k)}{}^{min} \leq V_{(k,i)} \leq V_{(k)}{}^{max} \tag{4.78}$$

$$V_{(k,0)} = V_{(k)}{}^{baslangıc} \qquad V_{(k,M)} = V_{(k)}{}^{son} \tag{4.79}$$

Her deponun su tahliye hızı, tüm aralıklarda minimum ve maksimum işletme sınırlarına sahip olmalıdır. Bu durum, Denklem (4.80)'de verilmektedir.

$$Q_{(k)}{}^{min} \leq Q_{(k,i)} \leq Q_{(k)}{}^{max} \tag{4.80}$$

(i+1) aralığı için *k.* depoda depolanan hacim, Denklem (4.81)'de elde edilmektedir.

$$V_{k,i+1} = V_{k,i} + \sum_{j=\Omega(k)} (Q_{(j,i-r)} + S_{j,i-r}) - Q_{k,i} - S_{k,i} + R_{k,i} \tag{4.81}$$

Burada $\Omega(k)$ *k.* depoya katkıda bulunan havzaların indeks setini, *r* zaman gecikmesini, *S* ve *R*, sırasıyla dökülme ve içeri akış oranını temsil etmektedir (Das ve Suganthan, 2010).

R21 Messenger: Uzay aracı yörünge optimizasyon problemi

Uzay Görev Tasarımı ile ilgili problemlerde küresel optimizasyon algoritmalarını test etmek için iyi bir kıyaslama, Çoklu Yerçekimi Yardımı (MGA) problemidir. Bu problem,

matematiksel olarak, lineer olmayan kısıtlamaları olan bir sonlu boyutlu küresel optimizasyon problemidir. Kimyasal bir itici motorla donatılmış gezegenler arası bir sondanın Dünya'dan başka bir gezegene veya asteroide gitmek amacıyla alabileceği en olası rotayı bulmak için kullanılabilmektedir (Addis, Cassioli, Locatelli ve Schoen, 2008). MGA-1DSM, her bir yörünge ayağı sırasında motorunu herhangi bir zamanda itebilen, kimyasal tahrik ile donatılmış bir uzay aracının gezegenler arası yörüngesini temsil etmektedir. Bu nedenle, bu sorunun çözümleri, gerçek uzay görevleri için ön hesaplama yapmak için uygun olmaktadır. "Messenger" yörünge optimizasyon problemi, MGA-1DSM problemi olarak modellenen Merkür için bir buluşma misyonunu temsil etmektedir. Seçilen uçuş sırası ve diğer parametreler, şu anda havada olan Messenger göreviyle uyumlu olmaktadır. Problem, 26 boyutlu bir küresel optimizasyon problemi olarak modellenmektedir (Das ve Suganthan, 2010).

R22 Cassini 2: Uzay aracı yörünge optimizasyon problemi

Bu problemde amaç fonksiyonu (kısıtlanmamış), derin uzay manevraları ile bir Dünya – Venüs, Venüs – Dünya, Jüpiter – Satürn uçuş dizisini kullanarak, Satürn'e ulaşmak için gereken rotayı değerlendirmektedir. Bu problem, gezegenlerin arasında derin uzay manevralarına izin vererek karmaşıklığı arttırmaktadır. Bu durum, yüksek maliyet fonksiyonu değerlerine sebep olabilmektedir. Problem 12 boyutlu olarak modellenmektedir (Das ve Suganthan, 2010).

4.3. Gerçek Dünya Problemleri Sonuçları

R1 nolu problemde, altıncı dereceden alçak geçiren filtreler için direnç ve kapasitor değerleri E24, sekizinci derece filtreler için direnç ve kapasitor değerleri E24 ve E96 serilerinden seçilmektedir. Analog filtre problemlerinde maksimum iterasyon sayısı 5000 olarak alınmaktadır. Çizelge 4.11-4.12 ve Şekil 4.7'de altıncı dereceden E24 ile tasarlanmış, Çizelge 4.13-4.14 ve Şekil 4.8'de sekizinci dereceden E24 ile tasarlanmış ve Çizelge 4.15-4.16 ve Şekil 4.9'da sekizinci dereceden E96 ile tasarlanmış filtrenin sonuçları verilmektedir. R2 numaralı problem için, Çizelge 4.17-4.18 ve Şekil 4.10'da yarı köprü ve tam köprü eviriciler için bulunan sonuçlar verilmektedir. R3 numaralı problem için Çizelge 4.19-4.21 alçak geçiren IIR filtre deneylerinin sonuçlarını, Çizelge

4.22-4.24 yüksek geçiren IIR filtre deneylerinin sonuçlarını vermektedir. Çizelge 4.25-4.27 diğer problemlerin sonuçlarını vermektedir. Bu çizelgelerin son satırında bulunan "No. Ort.", her bir algoritmanın bulduğu en iyi çözüm sayısını ortalama değere bağlı olarak ifade etmektedir.

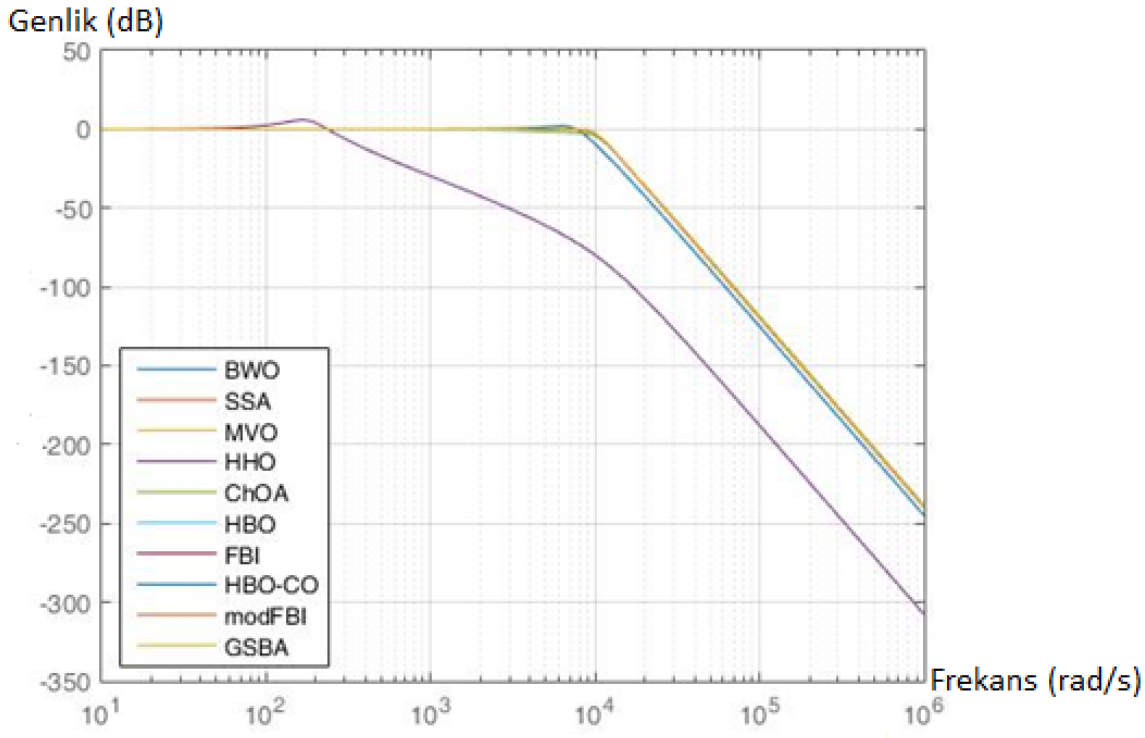
Çizelge 4.11-4.12 altıncı dereceden filtrenin sonuçlarında (R1.a), geliştirilen modFBI ve HBO-CO algoritmaları en iyi sonuçlara ulaşmaktadır. HBO-CO en düşük hataya, modFBI ise en düşük ortalama hataya ulaşmaktadır. Çizelge 4.13-4.14 sekizinci dereceden E24 filtrenin tasarım sonuçlarında (R1.b), HBO-CO daha önceki örnekte olduğu gibi, en iyi çözümü bulmada lider olmaktadır. Diğer yandan, SSA algoritması da bu problem üzerinde ortalama hata açısından en iyi performansı göstermektedir. modFBI algoritması ise ortalama hatada, SSA'dan sonra en iyi ikinci algoritma olmaktadır. Şekil 4.7 ve Şekil 4.8'de, güncel algoritmalarla tasarlanan alçak geçiren filtrenin genlik cevaplarında, HHO algoritması başarılı bir performans gösterememektedir.

Çizelge 4.11. Altıncı dereceden filtre tasarımı için klasik algoritmaların sayısal sonuçları (E24)

	GA	PSO	DE	CS	HS	GSBA
$R_1\ (k\Omega)$	10	3.9	8.2	8.2	9.1	3.6
$R_2\ (k\Omega)$	6.8	3.9	2.0	15	2.7	1.5
$R_3\ (k\Omega)$	7.5	120	9.1	39	10	4.7
$R_4\ (k\Omega)$	8.2	8.2	56	3.9	9.1	4.7
$R_5\ (k\Omega)$	33	11	9.1	30	18	10
$R_6\ (k\Omega)$	7.5	10	8.2	5.1	8.2	7.5
$C_1\ (nF)$	3	6.8	5.1	2.2	4.3	10
$C_2\ (nF)$	47	100	120	36	91	180
$C_3\ (nF)$	9.1	1.1	2.2	3.3	7.5	15
$C_4\ (nF)$	18	9.1	9.1	20	15	30
$C_5\ (nF)$	4.7	9.1	11	5.6	7.5	11
$C_6\ (nF)$	8.2	10	12	12	9.1	12
Hata (En iyi)	0.034	0.024	0.015	0.016	0.021	0.012
Ort	0.210	0.028	0.022	0.019	0.082	0.028
Std	0.278	0.004	0.004	0.002	0.096	0.009

Çizelge 4.12. Altıncı dereceden filtre tasarımı için gelişmiş algoritmaların sayısal sonuçları (E24)

	BWO	SSA	MVO	HHO	ChOA	HBO	FBI	HBO-CO	modFBI
$R_1\ (k\Omega)$	7.5	3.3	7.5	33	15	3.6	5.1	3.9	13
$R_2\ (k\Omega)$	6.2	3.6	27	150	1.6	1.2	5.1	2.4	16
$R_3\ (k\Omega)$	4.7	2.2	7.5	2.4	100	1.3	13	7.5	22
$R_4\ (k\Omega)$	0.62	68	11	9.1	1.1	8.2	9.1	8.2	5.6
$R_5\ (k\Omega)$	1.8	6.2	20	16	91	6.2	7.5	9.1	15
$R_6\ (k\Omega)$	20	20	5.1	3	150	11	10	12	30
$C_1\ (nF)$	5.1	7.5	1.5	16	3	11	5.1	8.2	1.8
$C_2\ (nF)$	75	110	33	390	130	220	75	130	27
$C_3\ (nF)$	27	2	7.5	11	1.3	15	6.2	9.1	5.1
$C_4\ (nF)$	130	33	16	33	47	62	13	18	16
$C_5\ (nF)$	9.1	7.5	7.5	13	1	11	11	9.1	4.3
$C_6\ (nF)$	30	11	13	16	1	13	12	10	5.1
Hata (En iyi)	0.157	0.025	0.016	0.628	0.262	0.028	0.026	**0.008**	0.015
Ort	0.281	0.155	0.027	0.680	0.310	0.029	0.029	0.007	**0.002**
Std	0.126	0.218	0.011	0.046	0.031	0.001	0.003	0.001	0.009



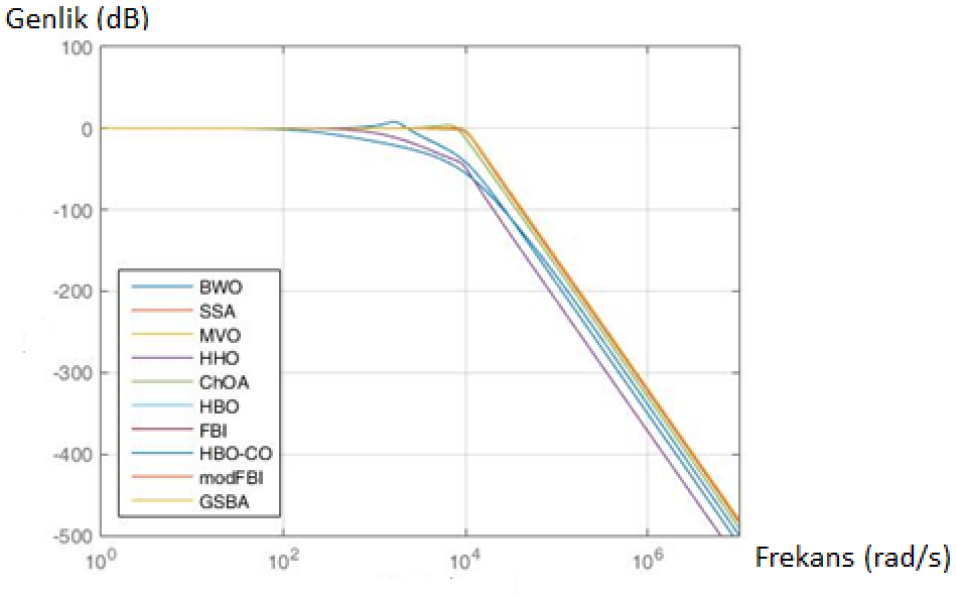
Şekil 4.7. Altıncı dereceden filtre tasarımında güncel algoritmaların frekans cevabı karşılaştırılması (E24)

Çizelge 4.13. Sekizinci dereceden filtre tasarımı için güncel algoritmaların sayısal sonuçları (E24)

	GA	PSO	DE	CS	HS	GSBA
$R_1\ (k\Omega)$	2.00	1.0	7.5	7.5	7.5	8.2
$R_2\ (k\Omega)$	2.70	1.2	12	4.7	3.3	8.2
$R_3\ (k\Omega)$	1.3	36	1	5.6	3	4.3
$R_4\ (k\Omega)$	4.3	16	1	8.2	13	5.1
$R_5\ (k\Omega)$	30	7.5	110	20	18	5.1
$R_6\ (k\Omega)$	15	100	27	13	9.1	56
$R_7\ (k\Omega)$	3.3	150	8.2	6.8	180	16
$R_8\ (k\Omega)$	1.8	6.2	39	36	10	11
$C_1\ (nF)$	8.2	18	2	3.6	3.6	2.4
$C_2\ (nF)$	220	470	56	100	100	62
$C_3\ (nF)$	20	2.2	56	8.2	6.2	12
$C_4\ (nF)$	91	8.2	180	27	39	39
$C_5\ (nF)$	3.9	1.5	1.2	5.1	6.2	2.7
$C_6\ (nF)$	5.6	9.1	2.7	7.5	9.1	13
$C_7\ (nF)$	39	1.3	4.3	4.7	1.2	7.5
$C_8\ (nF)$	43	8.2	7.5	9.1	4.3	7.5
Hata (En iyi)	0.039	0.057	0.035	0.083	0.045	0.033
Ort	0.126	0.059	0.044	0.119	0.081	0.039
Std	0.077	0.001	0.012	0.033	0.018	0.008

Çizelge 4.14. Sekizinci dereceden filtre tasarımı için güncel algoritmaların sayısal sonuçları (E24)

	BWO	SSA	MVO	HHO	ChOA	HBO	FBI	HBO-CO	modFBI
$R_1\ (k\Omega)$	20	3.3	7.5	3	1	3.6	6.2	3	2
$R_2\ (k\Omega)$	33	11	4.7	8.2	33	22	2.2	3.9	2.7
$R_3\ (k\Omega)$	3.3	8.2	16	9.1	100	30	4.7	39	8.2
$R_4\ (k\Omega)$	3.6	8.2	10	62	1	43	68	2.2	8.2
$R_5\ (k\Omega)$	3.9	5.6	11	2.7	1.5	3.6	15	11	8.2
$R_6\ (k\Omega)$	3.6	8.2	27	13	2	15	7.5	11	6.8
$R_7\ (k\Omega)$	5.1	30	8.2	91	1	3.3	100	5.1	6.8
$R_8\ (k\Omega)$	2.2	20	18	62	1	2.4	7.5	10	11
$C_1\ (nF)$	4.3	2.7	3.3	3.6	1.6	1.5	4.7	5.6	8.2
$C_2\ (nF)$	120	100	91	130	330	82	160	150	220
$C_3\ (nF)$	16	6.8	4.3	12	1.2	1.6	1.5	2.7	6.8
$C_4\ (nF)$	51	22	15	13	120	5.1	20	43	22
$C_5\ (nF)$	22	12	4.3	9.1	47	9.1	7.5	7.5	11
$C_6\ (nF)$	33	18	7.5	36	100	20	12	11	16
$C_7\ (nF)$	27	3.9	7.5	9.1	100	33	1.8	13	11
$C_8\ (nF)$	33	4.3	9.1	6.8	91	39	7.5	15	12
Hata (En iyi)	0.438	0.013	0.044	0.925	0.497	0.064	0.033	**0.012**	0.018
Ort	0.547	**0.019**	0.091	0.927	0.639	0.095	0.046	0.025	0.024
Std	0.136	0.005	0.075	0.009	0.128	0.031	0.013	0.010	0.005



Şekil 4.8. Sekizinci dereceden filtre tasarımında algoritmaların frekans cevabı karşılaştırılması (E24)

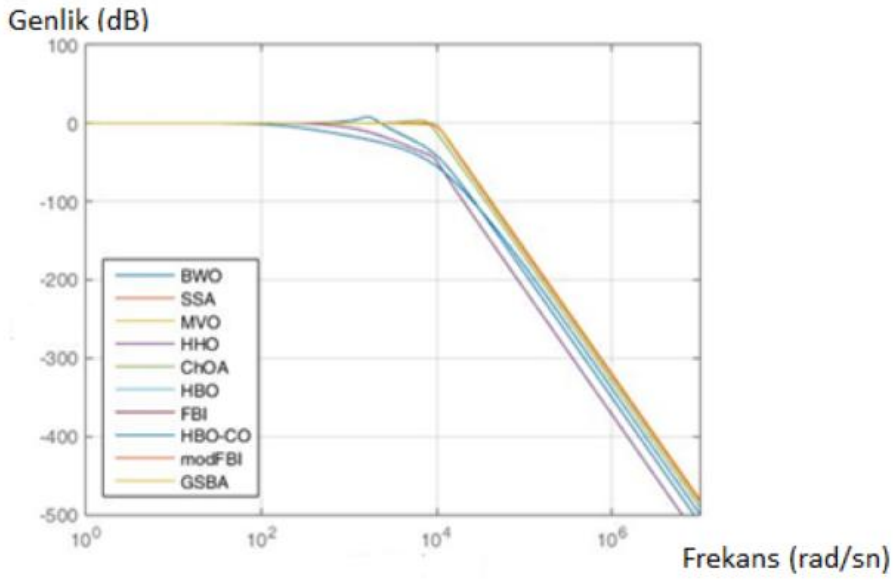
Çizelge 4.15-4.16 sekizinci dereceden E96 filtrenin tasarım sonuçlarında (R1.b), modFBI algoritması, hem ortalama hem en iyi hatada, en başarılı performansı göstermektedir. GSBA algoritması onun ardından, en düşük ortalama ve en iyi hata değerlerine ulaşan algoritmadır. MVO ise ortalama hata değerine göre en iyi üçüncü algoritma olmaktadır.

Çizelge 4.15. Sekizinci dereceden filtre tasarımı için klasik algoritmaların sayısal sonuçları (E96)

	GA	PSO	DE	CS	HS	GSBA
$R_1\ (k\Omega)$	32.4	3.74	12.4	3	8.25	5.62
$R_2\ (k\Omega)$	6.81	1	10	2.2	7.5	4.32
$R_3\ (k\Omega)$	6.19	25.5	1.4	3.6	10.5	6.34
$R_4\ (k\Omega)$	86.6	7.87	27.4	3	3.32	6.81
$R_5\ (k\Omega)$	1.69	11.8	13.3	4.3	158	3.16
$R_6\ (k\Omega)$	165	13.7	6.34	12	28.7	19.1
$R_7\ (k\Omega)$	71.5	1	33.2	2.7	11	14.7
$R_8\ (k\Omega)$	6.81	1	78.7	1	23.7	8.45
$C_1\ (nF)$	1	8.25	1.74	7.5	2.15	3.92
$C_2\ (nF)$	45.3	324	46.4	200	60.4	105
$C_3\ (nF)$	1.21	3.32	3.74	16	9.09	8.45
$C_4\ (nF)$	15.4	15	66.5	56	34.8	27.4
$C_5\ (nF)$	1	6.49	8.45	10	1.1	7.5
$C_6\ (nF)$	36.5	9.53	14	20	2.49	22.1
$C_7\ (nF)$	11	100	1.74	51	6.19	8.45
$C_8\ (nF)$	1.96	100	2.21	68	5.76	9.53
Hata (En iyi)	0.225	0.010	0.022	0.014	0.285	0.003
Ort	0.455	0.029	0.037	0.142	0.316	0.006
Std	0.212	0.017	0.013	0.163	0.045	0.004

Çizelge 4.16. Sekizinci dereceden filtre tasarımı için güncel algoritmaların sayısal sonuçları (E96)

	BWO	SSA	MVO	HHO	ChOA	HBO	FBI	HBO-CO	modFBI
$R_1\ (k\Omega)$	18.7	1.69	2.26	7.32	30.1	2.1	4.02	3.4	4.12
$R_2\ (k\Omega)$	14	3.09	2.26	3.01	23.7	1	4.22	1.74	5.36
$R_3\ (k\Omega)$	6.65	3.40	14.7	3.4	1.07	1.78	22.6	8.25	16.5
$R_4\ (k\Omega)$	1.5	5.76	10	8.06	102	9.53	5.9	8.06	17.8
$R_5\ (k\Omega)$	21	5.9	5.62	14.3	102	8.45	9.76	11	14
$R_6\ (k\Omega)$	2.8	57.6	30.1	40.2	102	3.57	29.4	8.25	21
$R_7\ (k\Omega)$	3.83	3.24	8.45	1.91	90.9	2.32	39.2	15.4	11
$R_8\ (k\Omega)$	2.55	23.7	2.49	12.7	13.3	12.1	12.4	8.25	5.62
$C_1\ (nF)$	1.18	8.25	8.66	4.12	1.43	14	4.75	7.5	4.12
$C_2\ (nF)$	32.4	237	226	93.1	127	422	124	221	110
$C_3\ (nF)$	13.7	12.1	4.53	12.4	1.1	9.76	3.92	6.81	3.24
$C_4\ (nF)$	73.2	42.2	15	34	86.6	60.4	19.1	22.1	10.5
$C_5\ (nF)$	6.98	2.61	4.64	205	1.13	14	4.22	8.66	4.75
$C_6\ (nF)$	24.3	11.3	12.7	90.9	1.1	23.7	8.25	12.7	7.15
$C_7\ (nF)$	30.9	7.32	17.8	42.2	2.49	13.7	3.83	8.25	11.8
$C_8\ (nF)$	33.2	17.8	26.7	10.7	4.02	25.5	5.36	9.53	13.7
Hata (En iyi)	0.019	0.009	0.011	0.983	0.329	0.068	0.012	0.011	**0.002**
Ort	0.326	0.332	0.012	0.997	0.574	0.090	0.046	0.026	**0.004**
Std	0.265	0.279	0.002	0.020	0.214	0.022	0.046	0.025	0.002



Şekil 4.9. Sekizinci dereceden filtre tasarımında güncel algoritmaların frekans cevabı karşılaştırılması (E96)

Çizelge 4.17-4.18 yarım ve tam köprü eviricisinin sonuçlarında (R2), GSBA ve HBO-CO algoritması yarım köprü evirici problemi üzerinde bir birine benzer sonuçlar vermekle beraber, GSBA biraz daha iyi performans göstermekte; modFBI bu iki algoritmayı takip etmektedir. Tam fazlı köprü evirici probleminde ise, modFBI algoritması GSBA algoritmasına yakın performans vermekle beraber; GSBA algoritması, bu problemde, tüm

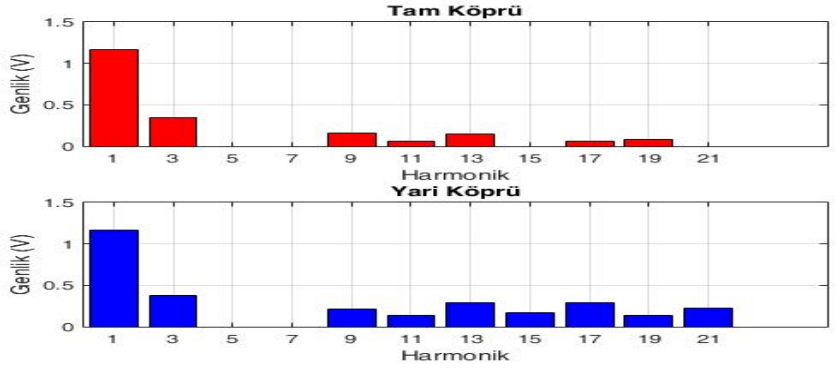
kıyaslanan algoritmalar arasında en iyi ortalama değere ulaşmaktadır. Şekil 4.10'da, geliştirilen GSBA algoritmasının sonuçlarının grafiksel gösterimi de verilmektedir.

Çizelge 4.17. Algoritmaların tek fazlı yarım köprü eviricisi üzerinde sonuçları

	α_1	α_2	En iyi	Ort	Std
BWO	10.46467	88.37026	2.680e-04	0.009	0.011
SSA	10.19792	88.50981	2.074e-08	0.004	0.006
MVO	10.25297	88.49358	9.760e-06	3.442e-04	3.909e-04
HHO	10.20329	88.70067	1.381e-04	0.002	0.002
ChOA	10.18536	88.44779	1.689e-05	2.325e-04	2.054e-04
GA	10.19743	88.51067	8.832e-09	2.345e-07	2.860e-07
PSO	10.34630	88.52983	6.888e-05	1.112e-04	1.531e-05
DE	10.00000	87.09147	1.034e-04	0.003	0.004
CS	10.41750	87.80152	0.001	0.017	0.022
HS	10.18583	88.53361	2.032e-06	1.833e-04	3.097e-04
HBO	10.19772	88.51231	4.065e-10	3.479e-09	4.119e-09
FBI	10.19534	88.51820	1.488e-07	3.214e-06	3.981e-06
HBO-CO	10.19768	88.51214	2.783e-12	2.795e-10	4.140e-10
modFBI	10.19757	88.51230	1.456e-10	1.963e-09	3.425e-09
GSBA	10.19771	88.51214	**4.086e-16**	**1.054e-10**	4.964e-10

Çizelge 4.18. Algoritmaların tek fazlı tam köprü eviricisi üzerinde sonuçları

	α_1	α_2	En iyi	Ort	Std
BWO	15.42512	87.42819	1.093e-08	4.485e-05	1.108e-04
SSA	15.40335	87.42329	6.022e-07	3.345e-04	5.244e-04
MVO	15.38214	87.28504	2.046e-05	1.479e-04	1.529e-04
HHO	15.48645	87.52902	1.206e-05	3.243e-04	2.919e-04
ChOA	15.74909	87.25908	1.230e-04	5.804e-04	6.626e-04
GA	15.42741	87.42826	1.302e-09	2.208e-07	2.514e-07
PSO	15.40607	87.36609	3.968e-06	1.543e-04	1.507e-04
DE	15.39579	87.09147	1.034e-04	8.427e-04	5.720e-04
CS	14.79265	87.19389	4.175e-04	0.004	0.003
HS	15.36275	87.57650	2.451e-05	3.849e-04	4.565e-04
HBO	15.42953	87.42889	9.254e-10	3.560e-08	3.469e-08
FBI	15.42863	87.42848	1.151e-11	3.221e-11	2.592e-11
HBO-CO	15.42851	87.42857	2.602e-12	1.443e-08	1.992e-08
modFBI	15.42853	87.4285	1.185e-12	5.865e-12	5.866e-12
GSBA	15.42857	87.42857	**9.291e-17**	**3.982e-14**	1.702e-13



Şekil 4.10. GSBA algoritması tarafından bulunan harmonikler

Çizelge 4.19-4.21'de alçak geçiren Chebyshev Tip I IIR filtre bulunan pay ve payda katsayılarına göre sonuçlarından (R3), en düşük hataya modFBI, en düşük ortalama hataya GSBA algoritmasının ulaştığı gözlemlenmektedir. Çizelge 4.22-4.24, Chebyshev Tip II IIR yüksek geçiren filtre bulunan pay ve payda katsayılarına göre sonuçlarından (R3), GSBA, modFBI, HBO-CO algoritmalarının benzer sonuçlar verdiği fakat GSBA algoritmasının, ortalama hatada, bulduğu 0.012 hata değeriyle biraz daha iyi olduğu gözlemlenmektedir.

Çizelge 4.19. Chebyshev Tip I IIR Filtre tasarımı klasik algoritmalar tarafından bulunan sonuçlar

	Pay Katsayıları					*Payda Katsayıları*				
No.	**GA**	**PSO**	**DE**	**CS**	**HS**	**GA**	**PSO**	**DE**	**CS**	**HS**
1	4.455e-05	5.388e-05	5.770e-05	4.663e-05	2.724e-05	1.000	1.000	1.000	1.000	1
2	8.819e-05	0.0004	0.0002	0.0005	0.0005	-5.831	-5.662	-5.806	-5.720	-5.720
3	0.002	0.001	0.001	0.0004	0.0013	18.145	18.254	18.005	18.053	18.090
4	0.006	0.006	0.001	0.007	0.0015	-38.506	-38.889	-38.149	-38.205	-38.616
5	0.008	0.025	0.025	0.008	0.013	60.931	60.805	60.529	60.719	60.361
6	0.016	0.047	0.046	0.049	0.025	-74.516	-74.449	-74.269	-74.605	-74.070
7	0.005	0.044	0.049	0.048	0.046	71.487	71.7109	71.222	71.796	71.438
8	0.043	0.046	0.045	0.016	0.011	-54.316	-54.193	-54.278	-54.556	-54.325
9	0.003	0.018	0.026	0.007	0.023	32.910	32.675	32.710	32.625	32.695
10	0.008	0.002	0.001	0.006	0.005	-15.632	-15.529	-15.559	-15.669	-15.053
11	0.002	0.0009	0.001	0.001	0.0004	5.518	5.386	5.411	5.943	5.473
12	0.0002	0.0004	0.0004	0.0002	0.0005	-1.178	-1.296	-1.250	-1.833	-1.787
13	1.564e-05	8.643e-06	2.245e-05	4.324e-05	3.342e-05	0.110	0.503	0.101	0.236	0.225

	GA	**PSO**	**DE**	**CS**	**HS**
En iyi hata	0.102	0.184	0.097	0.131	0.231
Ort	0.022	0.222	0.117	0.156	0.256
Std	0.056	0.024	0.014	0.021	0.019

Çizelge 4.20. Chebyshev Tip I IIR Filtre tasarımı güncel algoritmalar tarafından bulunan sonuçlar (1/2)

	Pay Katsayıları					*Payda Katsayıları*				
No.	**BWO**	**SSA**	**MVO**	**HHO**	**ChOA**	**BWO**	**SSA**	**MVO**	**HHO**	**ChOA**
1	2.468	6.000e-06	3.659e-05	5.164e-05	0.528	1.000	1.000	1.000	1.000	1.000
2	-0.458	0.0007	9.030e-05	9.461e-05	0.0001	-4.619	-6.000	-5.751	-5.995	-5.867
3	-0.335	0.0007	0.002	0.001	0.001	11.743	18.000	18.000	18.289	18.000
4	-0.008	0.010	0.006	0.004	0.003	-38.586	-38.000	-38.437	-38.438	-38.000
5	0.288	0.030	0.003	0.019	0.016	60.193	60.000	61.000	60.725	60.000
6	4.340	0.050	0.050	0.024	0.050	-74.266	-74.000	-74.214	-74.299	-74.000
7	0.374	0.050	0.050	0.047	0.046	71.230	72.000	-74.214	71.751	72.000
8	-4.757	0.050	0.049	0.033	0.035	-32.290	-55.000	-54.000	-54.350	-55.000
9	-4.694	0.003	0.007	0.017	0.005	32.467	33.000	33.000	32.392	33.000
10	-13.739	0.001	0.001	0.017	0.002	-14.957	-15.000	-16.000	-15.216	-15.000
11	-0.181	0.0004	0.003	0.001	0.0008	4.118	5.000	5.770	-15.216	5.000
12	-4.265	7.000e-05	0.0002	0.0004	0.0005	0.392	-1.000	-1.298	-1.886	-1.003
13	4.393	6.000e-05	4.765e-05	3.730e-05	0.0005	0.369	0.100	0.106	0.528	0.137

	BWO	**SSA**	**MVO**	**HHO**	**ChOA**
En iyi hata	0.008	0.114	0.028	0.122	0.130
Ort	0.019	0.155	0.060	0.143	0.173
Std	0.008	0.030	0.025	0.014	0.035

Çizelge 4.21. Chebyshev Tip I IIR Filtre tasarımı güncel algoritmalar tarafından bulunan sonuçlar (2/2)

	Pay Katsayıları					*Payda Katsayıları*				
No.	**HBO**	**FBI**	**HBO-CO**	**modFBI**	**GSBA**	**HBO**	**FBI**	**HBO-CO**	**modFBI**	**GSBA**
1	3.250e-05	0.0004	1.813e-05	3.445e-05	9.160e-06	1.000	1.000	1.000	1.000	1.000
2	0.0007	-0.010	0.0001	0.0002	0.0001	-5.776	-10.368	-5.849	-5.866	-5.935
3	0.001	-0.189	0.003	0.002	0.001	18.234	13.305	18.124	18.114	18.398
4	0.003	0.490	0.001	0.001	0.006	-38.603	-38.425	-38.483	-38.456	-38.802
5	0.016	-1.380	0.029	0.011	0.02	60.895	60.867	60.689	60.980	60.842
6	0.049	-2.457	0.049	0.036	0.048	-74.317	-76.907	-74.169	-74.816	-74.052
7	0.047	-2.380	0.031	0.047	0.046	71.788	71.090	71.398	71.988	71.143
8	0.048	-2.300	0.050	0.047	0.046	-54.742	-54.426	-54.288	-54.712	-54.137
9	0.012	-1.380	0.022	0.004	0.006	32.714	32.608	32.636	32.722	32.710
10	0.009	0.525	0.001	0.003	0.005	-15.102	-17.018	-15.336	-15.597	-15.550
11	0.0005	-0.206	0.002	0.001	0.003	5.458	5.606	5.723	5.964	5.925
12	0.0002	-0.010	0.0003	0.0001	0.0002	-1.724	-1.426	-1.674	-1.864	-1.760
13	1.162e-05	0.0002	1.820e-05	7.204e-06	1.865e-05	0.417	0.448	0.408	0.400	0.378

	HBO	**FBI**	**HBO-CO**	**modFBI**	**GSBA**
En iyi hata	0.028	0.028	0.027	**0.004**	0.006
Ort	0.031	0.087	0.028	0.016	**0.012**
Std	0.002	0.037	0.001	0.007	0.004

Çizelge 4.22. Chebyshev Tip II IIR Filtre tasarımı klasik algoritmalar tarafından bulunan sonuçlar

	Pay Katsayıları					*Payda Katsayıları*				
No.	**GA**	**PSO**	**DE**	**CS**	**HS**	**GA**	**PSO**	**DE**	**CS**	**HS**
1	0.149	0.149	0.158	0.138	0.185	1.000	1.000	1.000	1.000	1.000
2	-1.353	-1.493	-1.491	-1.466	-1.207	-4.698	-4.831	-4.914	-4.910	-4.517
3	5.152	5.399	5.445	5.285	5.451	13.168	13.141	13.670	13.759	13.805
4	-13.198	-13.384	-13.016	-13.196	-13.470	-25.832	-25.870	-25.005	-25.878	-25.696
5	24.563	24.671	24.000	24.956	24.712	35.545	35.631	35.957	35.833	35.506
6	-35.305	-35.315	-35.011	-35.407	-35.331	-39.898	-39.315	-39.621	-39.072	-39.974
7	39.805	39.642	39.764	39.219	39.622	34.728	34.661	34.960	34.939	34.720
8	-35.465	-35.113	-35.013	-35.034	-35.273	-23.652	-23.951	-23.532	-23.508	-23.585
9	24.542	24.313	24.000	24.831	24.622	13.069	13.235	13.326	13.200	13.484
10	-13.377	-13.227	-13.062	-13.399	-13.490	-5.930	-5.663	-5.976	-5.919	-5.670
11	5.334	5.216	5.375	5.055	5.452	1.874	1.524	1.521	1.901	1.879
12	-1.309	-1.337	-1.462	-1.197	-1.142	-0.327	-0.264	-0.365	-0.357	-0.182
13	0.197	0.173	0.162	0.166	0.153	0.083	0.081	0.097	0.042	0.081

	GA	**PSO**	**DE**	**CS**	**HS**
En iyi hata	0.041	0.043	0.032	0.058	0.057
Ort	0.092	0.063	0.045	0.087	0.077
Std	0.035	0.018	0.008	0.021	0.018

Çizelge 4.23. Chebyshev Tip II IIR Filtre tasarımı güncel algoritmalar tarafından bulunan sonuçlar (1/2)

	Pay Katsayıları					*Payda Katsayıları*				
No.	**BWO**	**SSA**	**MVO**	**HHO**	**ChOA**	**BWO**	**SSA**	**MVO**	**HHO**	**ChOA**
1	0.100	0.154	0.159	0.166	0.108	1.000	1.000	1.000	1.000	1.000
2	-1.189	-1.297	-1.500	-1.323	-1.437	-4.974	-4.847	-5.000	-4.797	-5.000
3	5.000	5.234	5.450	5.416	5.500	13.496	13.624	13.000	13.991	13.000
4	-13.017	-13.143	-13.288	-13.290	-13.500	-25.478	-25.645	-26.000	-25.097	-26.000
5	24.000	24.543	24.646	49.000	25.000	35.972	35.854	35.500	35.955	35.500

6	-35.000	-35.320	-35.460	-35.415	-35.500	-39.005	-39.564	-39.732	-39.302	-40.000
7	39.659	39.599	40.000	39.227	40.000	34.573	34.864	34.500	34.998	34.500
8	-35.000	-35.040	-35.500	-35.144	-35.500	-23.555	-23.598	-23.500	-23.629	-24.000
9	24.000	24.611	24.762	49.000	25.000	13.278	13.430	13.500	13.357	13.000
10	-13.000	-13.284	-13.500	-13.389	-13.500	-5.852	-5.759	-5.884	-5.977	-6.000
11	5.003	5.130	5.500	5.127	5.500	1.502	1.776	1.500	1.586	1.500
12	-1.481	-1.228	-1.488	-1.308	-1.500	-0.206	-0.357	-0.295	-0.150	-0.400
13	0.152	0.180	0.150	0.157	0.200	0.044	0.045	0.010	0.085	0.014

	BWO	SSA	MVO	HHO	ChOA
En iyi hata	0.038	0.024	0.005	0.326	0.022
Ort	0.056	0.031	0.008	0.339	0.026
Std	0.026	0.006	0.003	0.014	0.004

Çizelge 4.24. Chebyshev Tip II IIR Filtre tasarımı güncel algoritmalar tarafından bulunan sonuçlar (2/2)

	Pay Katsayıları					***Payda Katsayıları***				
No.	**HBO**	**FBI**	**HBO-CO**	**modFBI**	**GSBA**	**HBO**	**FBI**	**HBO-CO**	**modFBI**	**GSBA**
1	0.192	0.172	0.125	0.193	0.102	1.000	1.000	1.000	1.000	1.000
2	-1.226	-1.407	-1.165	-1.454	-1.393	-4.660	-4.800	-4.953	-4.898	-4.955
3	5.165	5.390	5.133	5.222	5.422	13.000	13.193	13.000	13.522	13.455
4	-13.418	-13.437	-13.228	-13.115	-13.413	-26.000	-25.101	-26.000	-24.081	-25.918
5	25.000	24.763	24.529	24.523	24.692	35.894	35.773	35.500	35.679	35.996
6	-35.319	-35.351	-35.113	-35.300	-35.360	-39.000	-39.005	-40.000	-39.461	-39.508
7	39.565	39.753	39.640	39.704	39.823	34.996	34.917	34.500	34.844	34.871
8	-35.000	-35.205	-35.456	-35.123	-35.374	-23.508	-23.505	-23.500	-25.053	-23.639
9	24.000	24.537	24.817	24.568	24.692	13.500	13.490	13.076	13.369	13.903
10	-13.000	-13.130	-13.300	-13.367	-13.364	-5.658	-5.582	-5.988	-5.663	-5.873
11	5.255	5.235	5.282	5.390	5.384	1.936	1.535	1.500	1.813	1.736
12	-1.293	-1.420	-1.364	-1.468	-1.424	-0.182	-0.251	-0.127	-0.283	-0.235
13	0.200	0.151	0.186	0.184	0.149	0.100	0.095	0.010	0.071	0.138

	HBO	FBI	HBO-CO	modFBI	GSBA
En iyi hata	0.032	0.007	0.005	0.006	**0.004**
Ort	0.042	0.008	0.008	0.007	**0.006**
Std	0.007	0.001	0.002	0.001	0.001

Diğer gerçek dünya problemlerinin sonuçları (R4-R22), Çizelge 4.25-4.27'de verilmektedir. Bu çizelgelere göre, GSBA, R4, R6, R7, R8, R15 ve R21 problemlerinde en iyi sonucu vermektedir. Buna en yakın olan modFBI algoritmasıdır ve 4 problemde en iyi sonuca ulaşmaktadır. HBO-CO ve MVO algoritmaları ise 3 tane en iyi çözüme ulaşarak üçüncü en iyi algoritma olmaktadır. Bu sonuçlardan yola çıkarak, GSBA algoritmasının gerçek dünya problemlerine iyi bir çözüm getirdiği gözlemlenmekte; en iyi çözümlere ulaşamadığı durumlarda, geliştirilen modFBI ve HBO-CO algoritmasının da başarılı çözümler verdiği gözlemlenmektedir. Diğer yandan, gerçek dünya problemlerinde modFBI ve HBO-CO, kendi orijinal versiyonları olan FBI ve HBO algoritmasından, 19 fonksiyonun (R4-R22) 16'sında daha iyi performans göstermektedirler.

Çizelge 4.25. Klasik algoritmaların sonuçları

		GA	PSO	DE	CS	HS	GSBA
R4	Ort	-12.014	-10.741	-15.148	-0.423	-14.776	**-21.458**
	Std	2.326	0.684	0.001	2.322	1.054	0.239
R5	Ort	20.862	21.659	19.620	16.374	**6.354**	14.574
	Std	4.113	2.445	1.433	1.950	7.501	6.803
R6	Ort	1.677	1.606	2.096	1.391	1.874	**0.957**
	Std	0.264	0.118	0.134	0.059	0.187	0.177
R7	Ort	15.658	14.164	18.595	13.975	14.941	**13.908**
	Std	2.317	0.232	3.084	0.234	2.045	0.237
R8	Ort	2.210e+02	2.417e+02	2.286e+02	2.233e+02	2.447e+02	**2.200e+02**
	Std	5.659	11.796	10.307	4.845	20.030	0.000
R9	Ort	2.955	1.907	2.211	1.902	2.44e+02	1.756
	Std	0.485	0.093	0.220	0.132	0.203	0.027
R10	Ort	0.015	0.013	0.013	**0.012**	0.013	0.013
	Std	0.001	8.403e-05	4.741-04	6.876e-06	4.400e-04	4.372e-04
R11	Ort	4.259e+05	1.393e+06	8.987e+07	2.254e+06	1.417e+07	7.205e+05
	Std	1.266e+05	1.884e+05	4.021e+07	1.639e+05	2.943e+06	8.147e+04
R12	Ort	2.588e+06	3.642e+06	1.076e+07	5.242e+06	4.073e+06	4.568e+06
	Std	2.529e+05	3.387e+05	8.499e+05	3.159e+05	1.797e+05	6.872e+04
R13	Ort	1.547e+04	1.551e+04	1.545e+04	1.545e+04	1.547e+04	1.545e+04
	Std	23.271	30.449	6.149	3.118	17.136	1.046
R14	Ort	1.955e+04	**1.904e+04**	1.961e+04	1.907e+04	2.049e+04	1.917e+04
	Std	8.170e+02	1.704e+02	5.287e+02	1.125e+02	1.065e+03	2.559e+02
R15	Ort	3.314e+04	3.319e+04	3.300e+04	3.296e+04	3.306e+04	**3.294e+04**
	Std	74.144	36.614	72.154	30.869	56.617	91.955
R16	Ort	1.458e+05	1.387e+05	1.472e+05	**1.351e+05**	1.444e+05	1.445e+05
	Std	3.313e+03	5.034e+03	7.317e+03	2.353e+03	5.472e+03	2.966e+03
R17	Ort	1.225e+07	2.061e+06	1.284e+09	7.683e+09	1.784e+07	1.571e+09
	Std	1.417e+07	3.118e+05	3.483e+08	3.339e+09	1.903e+07	3.794e+08
R18	Ort	3.536e+06	1.136e+06	1.523e+07	7.341e+06	5.461e+06	2.665e+06
	Std	2.791e+06	2.632e+05	5.481e+06	2.194e+06	1.205e+06	8.003e+05
R19	Ort	5.007e+06	**1.402e+06**	1.557e+07	8.810e+06	6.120e+06	2.819e+06
	Std	3.030e+06	3.523e+05	4.221e+06	2.837e+06	1.301e+06	8.736e+05
R20	Ort	3.893e+06	1.106e+06	1.605e+07	8.641e+06	5.444e+06	2.314e+06
	Std	3.406e+06	1.398e+05	4.335e+06	2.177e+06	1.133e+06	6.499e+05
R21	Ort	23.691	55.330	45.726	30.931	23.204	**21.388**
	Std	5.299	8.186	4.237	3.148	1.944	4.837
R22	Ort	28.362	39.890	35.684	30.754	28.170	26.709
	Std	2.672	4.446	3.776	1.788	3.819	4.021
No Ort		0	2	0	2	1	6

Çizelge 4.26. Güncel algoritmaların sonuçları (1/2)

		BWO	SSA	MVO	HHO	GSBA
R4	Ort	-10.224	-13.163	-16.464	-7.467	**-21.458**
	Std	0.564	3.089	4.785	0.912	0.239
R5	Ort	24.981	19.073	15.001	24.749	14.574
	Std	1.800	4.360	5.662	1.234	6.803
R6	Ort	1.616	1.265	1.575	1.969	**0.957**
	Std	0.197	0.239	0.159	0.141	0.177
R7	Ort	20.968	14.254	14.188	14.954	**13.908**
	Std	1.649	0.192	0.214	0.654	0.237
R8	Ort	2.287e+02	2.639e+02	2.215e+02	2.459e+02	**2.200e+02**
	Std	33.122	34.343	4.833	16.773	0.000
R9	Ort	3.354	1.871	1.739	2.188	1.756
	Std	0.665	0.133	0.010	0.184	0.027
R10	Ort	5.760e+02	0.014	0.017	0.016	0.013
	Std	1.056e+03	0.004	0.001	0.001	4.372e-04
R11	Ort	3.825e+05	1.277e+06	1.761e+06	2.348e+08	7.205e+05
	Std	1.118e+05	2.261e+05	7.851e+05	1.278e+07	8.147e+04
R12	Ort	2.798e+06	4.019e+06	1.352e+06	1.294e+07	4.568e+06
	Std	1.791e+05	4.522e+05	6.979e+04	4.339e+05	6.872e+04

R13	Ort	3.131e+04	1.553e+04	1.549e+04	1.661e+04	1.545e+04
	Std	2.200e+04	44.842	24.478	1.239e+03	1.046
R14	Ort	1.904e+04	1.907e+04	1.931e+04	2.105e+04	1.917e+04
	Std	2.317e+02	1.621e+02	1.735e+02	1.703e+03	2.559e+02
R15	Ort	7.346e+05	3.683e+04	3.310e+04	2.396e+05	**3.294e+04**
	Std	1.487e+06	1.370e+04	1.040e+02	6.510e+04	91.955
R16	Ort	1.396e+05	1.462e+05	1.434e+05	1.560e+07	1.445e+05
	Std	3.276e+03	5.909e+03	4.966e+03	8.531e+06	2.966e+03
R17	Ort	8.135e+08	4.541e+07	**2.881e+06**	8.213e+09	1.571e+09
	Std	5.101e+08	2.102e+08	8.284e+05	1.124e+09	3.794e+08
R18	Ort	3.317e+07	1.321e+06	**1.002e+06**	1.000e+08	2.665e+06
	Std	5.432e+06	1.003e+06	5.797e+04	8.738e+06	8.003e+05
R19	Ort	3.290e+07	2.206e+06	1.568e+06	1.008e+08	2.819e+06
	Std	5.362e+06	1.798e+06	2.044e+05	8.377e+06	8.736e+05
R20	Ort	3.116e+07	1.738e+06	**1.063e+06**	1.003e+08	2.314e+06
	Std	4.125e+06	1.701e+06	1.604e+05	1.012e+07	6.499e+05
R21	Ort	24.314	37.418	29.038	45.062	**21.388**
	Std	3.950	10.287	6.772	5.828	4.837
R22	Ort	29.062	30.862	27.847	51.859	26.709
	Std	1.829	4.419	4.278	1.318	4.021
No. Ort		0	0	3	0	6

Çizelge 4.27. Güncel algoritmaların sonuçları (2/2)

		ChOA	**HBO**	**FBI**	**HBO-CO**	**modFBI**
R4	Ort	-10.297	-9.947	-12.467	-10.287	-15.441
	Std	0.040	0.268	0.605	0.000	3.879
R5	Ort	22.323	18.376	10.694	14.978	9.159
	Std	2.828	2.038	1.307	5.873	4.572
R6	Ort	2.102	1.881	1.431	1.835	1.335
	Std	0.205	0.055	0.053	0.151	0.062
R7	Ort	14.424	19.427	17.612	17.486	16.615
	Std	0.297	1.624	0.259	3.493	2.873
R8	Ort	4.205e+02	**2.200e+02**	**2.200e+02**	**2.200e+02**	**2.200e+02**
	Std	2.675e+02	0.000	0.000	0.000	0.000
R9	Ort	1.828	2.096	1.731	1.989	**1.724**
	Std	0.023	0.234	0.013	0.228	3.844e-05
R10	Ort	0.013	0.012	0.012	0.012	0.012
	Std	3.845e-04	9.433e-05	9.018e-06	1.452e-04	1.461e-05
R11	Ort	6.466e+06	2.628e+06	3.645e+06	**2.092e+05**	1.356e+06
	Std	1.010e+06	6.847e+05	2.873e+05	7.932e+04	7.563e+04
R12	Ort	1.221e+07	1.407e+06	4.573e+06	**1.345e+06**	4.327e+06
	Std	5.780e+05	9.236e+04	2.634e+05	1.110e+05	2.113e+05
R13	Ort	1.560e+04	1.547e+04	1.545e+04	1.546e+04	**1.544e+04**
	Std	1.219e+02	5.797	6.557	10.249	2.101
R14	Ort	2.184e+04	1.990e+04	1.913e+04	1.947e+04	1.912e+04
	Std	2.514e+03	5.390e+02	1.309e+02	2.812e+02	1.576e+02
R15	Ort	4.312e+04	3.305e+04	3.304e+04	3.306e+04	3.306e+04
	Std	2.304e+04	38.622	43.618	53.801	55.145
R16	Ort	1.605e+05	1.444e+05	1.475e+05	1.425e+05	1.414e+05
	Std	1.681e+04	3.498e+03	1.033e+04	4.311e+03	2.237e+03
R17	Ort	8.986e+09	7.939e+06	9.394e+07	7.554e+06	8.117e+07
	Std	1.492e+09	3.710e+06	2.637e+07	4.011e+06	8.606e+07
R18	Ort	1.044e+07	4.620e+06	2.502e+07	3.597e+06	2.356e+07
	Std	2.278e+06	4.890e+05	5.308e+06	4.533e+05	3.415e+06
R19	Ort	1.062e+07	4.389e+06	2.807e+07	4.386e+06	1.794e+07
	Std	2.971e+06	6.807e+05	3.326e+06	7.314e+05	6.218e+06
R20	Ort	1.160e+07	3.991e+06	2.791e+07	3.090e+06	2.082e+07
	Std	3.011e+06	6.776e+05	2.635e+06	5.344e+05	4.528e+06
R21	Ort	47.556	34.201	26.727	25.180	22.653
	Std	6.740	2.784	2.332	4.210	0.724
R22	Ort	39.062	28.700	23.686	26.501	**19.760**
	Std	3.407	5.645	4.437	3.935	4.695
No. Ort		0	1	1	3	4

5. SONUÇLAR VE TARTIŞMA

Metasezgisel algoritmalar, geleneksel yöntemlerin başarısız olduğu veya yetersiz kaldığı, çözüme giden alternatif sonuçların geliştirilmesine ihtiyaç duyulduğu noktalarda sıklıkla başvurulan optimizasyon yöntemlerindendir. Bu algoritmalar, optimizasyon problemlerinin çözümünde, problemin uygunluk fonksiyonu adı verilen matematiksel modellenmesinden çıkan sonuçları kullanarak en iyiye yakınsamayı amaçlamaktadır. Bu amaçlar doğrultusunda birçok metasezgisel algoritma geliştirilmiş ve geliştirilmeye de devam edilmektedir.

Bu tezde, yeni bir metasezgisel algoritma olan GSBA algoritması geliştirilmiştir. Ayrıca literatürde bulunan HBO ve FBI metasezgisel algoritmalarında değişiklikler/iyileştirmeler yapılarak HBO-CO ve modFBI vesiyonları sunulmuştur.

Önerilen algoritmaların performansını göstermek için; GSBA, HBO-CO ve modFBI algoritmaları, 5 klasik algoritma (GA, DE, PSO, CS, HS) ve 7 güncel algoritma (BWO, SSA, MVO, HHO, ChOA, FBI, HBO) ile tek modlu, çok modlu ve yüksek boyutlu nümerik fonksiyonlar üzerinde kıyaslanarak, iki farklı nümerik fonksiyon seti üzerinde test edilmiştir. Wilcoxon ve Friedman testleri aracılığıyla elde edilen sonuçların istatistiki analizleri gerçekleştirilmiş ve grafiksel dağılımları gösterilmiştir. Önerilen modifiye algoritmalar olan HBO-CO ve modFBI'ın, kendi orjinal versiyonları olan HBO ve FBI'dan her iki numerik fonksiyon seti üzerinde birçok fonksiyonda istatistiksel olarak daha başarılı performans verdiği gözlemlenmiştir. Diğer yandan farklı çalıştırılmalarında, bulunan sonuçların grafiksel dağılımlarından yola çıkılarak, geliştirilen algoritmaların bir çok fonksiyonda daha dar dağılıma sahip olarak, daha stabil davranış sergilediği görülmektedir. Bu durum da, yapılan analizlere bütüncül bir şekilde bakıldığında, yapılan iyileştirmelerin, belirtilen algoritmaların performansını arttırdığı anlamına gelmektedir. Buna ek olarak, geliştirilen GSBA algoritması, her iki problem setinde gerek kıyaslanan klasik ve güncel algoritmalar üzerinde, gerekse de modifiye edilen algoritmalar üzerinde bir baskınlık sağlamış, istatistiksel olarak, 3.217 ve 3.950 değerleriyle, daha başarılı sonuçlara ulaşmıştır.

Ayrıca bu önerilen algoritmalar ile, klasik ve güncel algoritmalar, 22 adet gerçek dünya problemine uygulanmış ve elde edilen sonuçlar analiz edilmiştir. Bu problemler için, daha önceki nümerik problem setlerinde olduğu gibi, tüm problemlerde istatistiksel olarak en iyi olan bir algoritma bulunmamaktadır. Elektrik-Elektronik mühendisliği alanındaki analog filtre problemlerinde HBO-CO ve modFBI algoritmaları öne çıkarken, harmonik eliminasyon problemlerinde GSBA, IIR filtre tasarım problemlerinde GSBA ve modFBI algoritmaları, diğer algoritmalarla kıyaslandığında, ortalama ve en iyi hatalara göre daha az hatayla çözüme ulaşabilmektedirler.

Bu çalışmanın diğer bir gözlemi olarak, ele alınan tüm gerçek dünya problemleri değerlendirildiğinde, genel anlamda geliştirilen algoritmaların performanslarının değişkenlik gösterdiği gözlemlenmiştir. Benzer şekilde, yapılan bu tez çalışmasında, bir problem seti üzerinde başarılı performans gösteren algoritma, başka bir problemde aynı başarıyı sağlayamayıp, kıyaslanan algoritmalar arasında en iyi sonucu gösteremediğini ortaya koymuştur. Bu tez, aynı zamanda, algoritmaların genel performansına ışık tutup, hangi algoritmanın hangi problemler üzerinde daha başarılı sonuçlar verebileceği hakkında fikirler sunmaktadır. Gelecek çalışmalarda, metasezgisel algoritmaların çok-amaçlı veya kombinasyonel optimizasyon problemlerine adapte edilip, performanslarının bu iki tipteki problem setlerinde analiz edilmesi planlanmaktadır.

EKLER

EK 1 Kullanılan 23 adet nümerik fonksiyonun özellikleri

Fonksiyonlar	Boyut	Aralık	Global Min.
$f_1(x) = \sum_{i=1}^{n} x_i^2$	50	$[-100, +100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	50	$[-10, +10]$	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	50	$[-100, +100]$	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	50	$[-100, +100]$	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	50	$[-30, +30]$	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	50	$[-100, +100]$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	50	$[-1.28, +1.28]$	0
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	50	$[-500, +500]$	$-418.9829 Xdim$
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	50	$[-5.12, +5.12]$	0
$f_{10}(x) = -20\, exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	50	$[-32, +32]$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	50	$[-600, +600]$	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + y_n - 1^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 - a & < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	50	$[-50, +50]$	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	50	$[-50, +50]$	0
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65.536, +65.53$	1
$f_{15}(x) = \sum_{i=1}^{11} \left\lvert a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right\rvert^2$	4	$[-5, +5]$	0.0003
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, +5]$	-1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, +0]$ $[10,15]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, +2]$	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i\, exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	$[0, 1]$	-3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i\, exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 10]$	-3.32
$f_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.1532
$f_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.4028
$f_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

KAYNAKLAR

Abedinpourshotorban, H., Shamsuddin, S. M., Beheshti, Z., ve Jawawi, D. N. (2016). Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*, 26, 8-22. https://doi.org/10.1016/j.swevo.2015.07.002

Addis, B., Cassioli, A., Locatelli, M. ve Schoen, F. (2008). Global optimization for the design of space trajectories. *Computational Optimization and Applications*, 48(3), 635-652. https://doi.org/10.1007/s10589-009-9261-6

Aelterman, J., Goossens, R., Declercq, F., ve Rogier, H. (2009). Ant colony optimisation-based radiation pattern manipulation algorithm for electronically steerable array radiator antennas. *IET Science, Measurement & Technology*, *3*(4), 302-311. https://doi.org/10.1049/iet-smt.2008.0127

Alatas, B. (2010). Chaotic harmony search algorithms. *Applied mathematics and computation*, *216*(9), 2687-2699. https://doi.org/10.1016/j.amc.2010.03.114

Allaoua, B., Laoufı, A., Gasbaoui, B. ve Abderrahmani, A. (2009). Setting up PID DC motor speed control alteration parameters using particle swarm optimization strategy. *Leonardo Electronic Journal of Practices and Technologies*, 14, 19-32.

Alkhafaji, B. J, Salih, M. A, Nabat, Z. M. ve Shnain SA (2020). Segmenting video frame images using genetic algorithms. *Periodicals of Engineering and Natural Sciences*, 8(2), 1106–1114. http://dx.doi.org/10.21533/pen.v8i2.1351

Askari, Q. Saeed, M., ve Younas, I. (2020). Heap-based optimizer inspired by corporate rank hierarchy for global optimization. *Expert Systems with Applications*, *161*, 113702. https://doi.org/10.1016/j.eswa.2020.113702

Atashpaz-Gargari, E. ve Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In *2007 IEEE Congress on Evolutionary Computation*, 4661-4667. https://doi.org/10.1109/cec.2007.4425083

Arora, S. ve Singh, S. (2019). Butterfly optimization algorithm: a novel approach for global optimization. *Soft Computing*, *23*(3),715-734. https://doi.org/10.1007/s00500-018-3102-4

Babu, B.V. and Sastry, K. K.N. (1999). Estimation of heat transfer parameters in a trickle bed reactor using differential evolution and orthogonal collocation. *Computers & Chemical Engineering*, 23,327-339. https://doi.org/10.1016/S0098-1354(98)00277-4

Babu, B.V. and Chaturvedi, G. (2000). Evolutionary computation strategy for optimization of an alkylation reaction. *Proceedings of International Symposium & Annual Session of IIChE.*

Babu, B.V. and Munawar, S.A. (2000). Differential evolution for the optimal design of heat exchangers. *Proceedings of All India Seminar on Chemical Engineering Progress on Resource Development*. 233-242.

Babu, T. S., Priya, K., Maheswaran, D., Kumar, K. S. ve Rajasekar, N. (2015). Selective voltage harmonic elimination in PWM inverter using bacterial foraging algorithm. *Swarm and Evolutionary Computation*, 20, 74–81. https://doi.org/10.1016/j.swevo.2014.11.002.

Birbil, Ş. İ., ve Fang, S. C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, 25(3), 263-282. https://doi.org/10.3724/sp.j.1087.2010.02914

Boussaïd, I., Lepagnot, J. ve Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82-117. https://doi.org/10.1016/j.ins.2013.02.041

Bingol, H. ve Alatas, B. (2016). Chaotic league championship algorithms. *Arabian Journal for Science and Engineering*, 41(12), 5123-5147. https://doi.org/10.1007/s13369-016-2200-9

Blum, C. ve Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, *35*(3), 268-308. https://doi.org/10.1145/937503.937505

Boschetti, M. A., Maniezzo, V., Roffilli, M., ve Röhler, A. B. (2009). Matheuristics: Optimization, simulation and control. In *International Workshop on Hybrid Metaheuristics*, 171-177.

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

Chaib, L., Choucha, A., ve Arif, S. (2017). Optimal design and tuning of novel fractional order PID power system stabilizer using a new metaheuristic Bat algorithm. *Ain Shams Engineering Journal*, 8(2), 113-125. https://doi.org/10.1016/j.asej.2015.08.003

Castro, L. N., De Castro, L. N., & Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media.

Cheng, M. Y. ve Prayogo, D. (2014). Symbiotic organisms search: a new metaheuristic optimization algorithm. *Computers & Structures*, 139, 98-112. https://doi.org/10.1016/j.compstruc.2014.03.007

Cheung, N. J., Ding, X. M. ve Shen, H. B. (2014). Adaptive firefly algorithm: parameter analysis and its application. *PloS One*, 9(11), e112634. https://doi.org/10.1371/journal.pone.0112634

Chen, R, Liang, C-Y, Hong, W-C. ve Gu D-X (2015). Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm. *Applied Soft Computing*, 26, 434–443. https://doi.org/10.1016/j.asoc.2014.10.022

Chen, T., Wang, Y., ve Li, J. (2012). Artificial Tribe Algorithm and Its Performance Analysis. *JSW*, 7(3), 651-656. https://doi.org/10.4304/JSW.7.3.651-656

Chou, J. S.ve Nguyen, N. M. (2020). FBI inspired meta-optimization. *Applied Soft Computing*, 93, 106339. https://doi.org/10.1016/j.asoc.2020.106339.

Chunxia, F. ve Youhong, W. (2008). An adaptive simple particle swarm optimization algorithm. In *2008 Chinese Control and Decision Conference*, 3067-3072. https://doi.org/10.1109/ccdc.2008.4597890

Chouhan, S. S., Kaul, A. ve Singh, U. P (2018). Soft computing approaches for image segmentation: a survey. *Multimedia Tools and Applications*, 77(21), 28483–28537. https://doi.org/10.1007/s11042-018-6005-6

Civicioglu P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219, 8121-8144. https://doi.org/10.1016/j.amc.2013.02.017

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2),113-127. https://doi.org/10.1016/S0166-3615(99)00046-9

Dantzig, G. B., ve Thapa, M. N. (2003). Linear programming: Theory and extensions USA: Springer.

Das, S. ve Suganthan, P.N. (2010). Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur University and Nanyang Technological University, Technical rep

Datta, D, Amaral, A. R. S. ve Figueira, J. R (2011) Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, 213(2), 388–394. https://doi.org/10.1016/j.ejor.2011.03.034

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. UK: Wiley.

De-Castro, L. N. ve Von Zuben, F. J. (2000). The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, 36-39.

De-Castro, L. N. ve Timmis, J. (2002). An artificial immune network for multimodal function optimization. *Proceedings of the 2002 Congress on Evolutionary Computation*, 699-704. https://doi.org/10.1109/cec.2002.1007011

De-Castro, L. N. (2002). Immune, swarm, and evolutionary algorithms. Part II: philosophical comparisons. *Proceedings of the 9th International Conference on Neural Information Processing*, 3, 1469-1473. https://doi.org/10.1109/iconip.2002.1203070

De B. P, Kar, R., Mandal, D. ve Ghoshal, S. P. (2015a). Optimal selection of components value for analog active filter design using simplex particle swarm optimization. *International Journal of Machine Learning and Cybernetics*, 6, 621–636. https://doi.org/10.1007/s13042-014-0299-0

De B. P., Kar, R, Mandal, D. ve Ghoshal, S.P. (2015b). Optimal analog active filter design using craziness-based particle swarm optimization algorithm. *International Journal of Numerical Modelling*, 28, 593-609. https://doi.org/10.1002/jnm.2040

De B. P., Kar, R, Mandal, D. ve Ghoshal, S.P. (2015c). Particle swarm optimization with aging leader and challengers for optimal design of analog active filters. *Circuits, Systems, and Signal Processing*,. 34, 707-737. https://doi.org/10.1007/s00034-014-9872-8

Dessouky, M. I. H., Sharshar, A. ve Albagory, Y. (2006). Efficient sidelobe reduction technique for small-sized concentric circular arrays. *Progress In Electromagnetics Research*, 65: 187-200. https://doi.org/10.2528/PIER06092503

Dib, N ve El-Asir, B. (2018). Optimal design of analog active filters using symbiotic organisms search. *International Journal of Numerical Modelling*, 31, e2323. https://doi.org/10.1002/jnm.2323

Dizqah, A.M., Maheri, A.ve Busawon K. (2014). An accurate method for the PV model identification based on a genetic algorithm and the interior-point method. *Renewable Energy*, 72, 212-222. https://doi.org/10.1016/j.renene.2014.07.014

Doğan, B. ve Ölmez, T. (2015). A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Information Sciences*, 293, 125-145. https://doi.org/10.1016/j.ins.2014.08.053

Dorigo, M. (1992). *Optimization, learning and natural algorithms* (Doktora Tezi, Politecnico di Milano). Erişim adresi: https://ci.nii.ac.jp/naid/10027800670/

Dukic, M. L. ve Dobrosavljevic, Z. S. (1990). A method of a spreadspectrum radar polyphase code design. *IEEE Journal on Selected Areas in Communications*, 8(5): 743–749. https://doi.org/10.1109/49.56381

Dwivedi, Y. ve V. T. Kumar (2017). Dynamic stability improvement of alkali fuel cell integrated system using PSO optimized PID control design. Recent Developments in Control, Automation & Power Engineering (RDCAPE), 499-504. https://doi.org/10.1109/RDCAPE.2017.8358322

Eberhart, R.C. ve Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. *IEEE Proceedings of Evolutionary Computation*, 1, 94-100. 10.1109/CEC.2001.934376

Eberhart, R. ve Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43. https://doi.org/10.1109/mhs.1995.494215

Ekbatanifard, G. H., Monsefi, R., Akbarzadeh-T, M-R. ve Yaghmaee, M. (2010). A multi-objective genetic algorithm based approach for energy efficient qos-routing in two-tiered wireless sensor net-works. *IEEE 5th International Symposium on Wireless Pervasive Computing*, 80-85. https://doi.org/10.1109/ISWPC.2010.5483775

Erol, O. K. ve Eksin, I. (2006). A new optimization method: big bang–big crunch. *Advances in Engineering Software*, 37(2), 106-111. https://doi.org/10.1016/j.advengsoft.2005.04.005

Farahani, S. M., Abshouri, A. A., Nasiri, B. ve Meybodi, M. R. (2011). A Gaussian firefly algorithm. *International Journal of Machine Learning and Computing*, 1(5), 448. https://doi.org/10.7763/ijmlc.2011.v1.67

Fister Jr, I., Yang, X. S., Fister, I., Brest, J. ve Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*.

Fogel, L. J., Owens, A. J. ve Walsh, M. J. (1966). Artificial intelligence through simulated evolution, Wiley.

Goud, H. Ve Swarnkar, P. (2019). Investigations on metaheuristic algorithm for designing adaptive PID controller for continuous stirred tank reactor. Mapan, 34(1), 113-119. https://doi.org/10.1007/s12647-018-00300-w

Gandomi, A. H. (2014). Interior search algorithm (ISA): a novel approach for global optimization. *ISA Transactions*, 53(4), 1168-1183. https://doi.org/10.1016/j.isatra.2014.03.018

Gandomi, A. H. ve Alavi, A. H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12), 4831-4845. https://doi.org/10.1016/j.cnsns.2012.05.010

Geem, Z. W., Kim, J. H. ve Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60-68. https://doi.org/10.1177/003754970107600201

Gendreau, M. ve Potvin, J. Y. (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1), 189-213. https://doi.org/10.1007/s10479-005-3971-7

Glover, F. ve McMillan, C. (1986). The general employee scheduling problem. An integration of MS and AI. *Computers & Operations Research*, 13(5), 563-573. https://doi.org/10.1016/0305-0548(86)90050-x

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533-549. https://doi.org/10.1016/0305-0548(86)90048-1

Ghorbani, N., & Babaei, E. (2014). Exchange market algorithm. *Applied Soft Computing*, 19, 177-187. https://doi.org/10.1016/j.asoc.2014.02.006

Goldberg, DE (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley.

Goswami, B. ve Mandal, D. (2013). A genetic algorithm for the level control of nulls and side lobes in linear antenna arrays. *Journal of King Saud University-Computer and Information Sciences*, 25(2), 117–126. https://doi.org/10.1016/j.jksuci.2012.06.001

Gurel L. ve Ergul O. (2008). Design and simulation of circular arrays of trapezoidal-tooth logperiodic antennas via genetic optimization. *Progress in Electromagnetics Research*, 85: 243-260. https://doi.org/10.2528/pier08081809

Haupt R. L. (1994.) Thinned arrays using genetic algorithms. *IEEE Trans Antennas Propag* 42, 993–999. https://doi.org/10.1109/8.299602-5

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, *222*, 175-184. https://doi.org/10.1016/j.ins.2012.08.023

Hayyolalam, V. ve Kazem, A. A. P. (2020). Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence*, 87, 103249. https://doi.org/10.1016/j.engappai.2019.103249

Hefny, H. A. ve Azab, S. S. (2010). Chaotic particle swarm optimization. In *2010 The 7th International Conference on Informatics and Systems (INFOS)*, 1-8.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. ve Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, 97, 849-872. https://doi.org/10.1016/j.future.2019.02.028

Holland, J. (1975). Adaptation in natural and artificial systems. USA: MIT Press.

Holmes, T. R. (1991). Caesar's conquest of Gaul. UK: Clarendon Press.

Huang, G. (2016). Artificial infectious disease optimization: A SEIQR epidemic dynamic model-based function optimization algorithm. *Swarm and Evolutionary Computation*, 27, 31-67. https://doi.org/10.1016/j.swevo.2015.09.007

Ibrahim, Z., Aziz, N. H. A., Aziz, N. A. A., Razali, S., ve Mohamad, M. S. (2016). Simulated Kalman filter: a novel estimation-based metaheuristic optimization algorithm. *Advanced Science Letters*, 22(10), 2941-2946.

https://doi.org/10.1166/asl.2016.7083

Ifeachor, E. C. ve Jervis B. W. (2002). Digital signal processing, a practical approach. UK: Pearson Education.

Ilonen, J., Kamarainen, J. K. ve Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1), 93-105. https://doi.org/10.1023/A:1022995128597

Ishaque, K. ve Salam, Z. (2011). An improved modeling method to determine the model parameters of photovoltaic (PV) modules using differential evolution (DE). *Solar Energy*, 85(9), 2349-2359. https://doi.org/10.1016/j.solener.2011.06.025

James, J. Q. ve Li, V. O. (2015). A social spider algorithm for global optimization. *Applied Soft Computing*, *30*, 614-627. https://doi.org/10.1016/j.asoc.2015.02.014

Ji, Z. ve Dasgupta, D. (2007). Revisiting negative selection algorithms. *Evolutionary Computation*, 15(2), 223-251. https://doi.org/10.1162/evco.2007.15.2.223

Junru, W. ve Lan, H. (2014). Evolving gomoku Solver by Genetic Algorithm. *IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, 1064–1067. https://doi.org/10.1109/WARTIA.2014.6976460

Joshi, R. ve Sanderson, A. C. (1999). Minimal representation multi-sensor fusion using differential evolution. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 29: 63-76. https://doi.org/10.1109/3468.736361

Kale, I. R., & Kulkarni, A. J. (2018). Cohort intelligence algorithm for discrete and mixed variable engineering problems. *International Journal of Parallel, Emergent and Distributed Systems*, 33(6), 627-662. https://doi.org/10.1080/17445760.2017.1331439

Kandavanam, G., Botvich, D., Balasubramaniam, S. ve Jennings, B. (2010). A hybrid genetic algorithm/variable neighborhood search approach to maximizing residual bandwidth of links for route planning. *Artificial Evolution*. 49–60. https://doi.org/10.1007/978-3-642-14156-0_5

Kahn, A. D. (2000) The education of Julius Caesar: a biography, a reconstruction, USA: iUniverse.

Karaboga, D., ve Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459-471. https://doi.org/10.1007/s10898-007-9149-x

Kashan, A. H. (2015). A new metaheuristic for optimization: optics inspired optimization (OIO). *Computers & Operations Research*, 55, 99-125. https://doi.org/10.1016/j.cor.2014.10.011

Kashan, A. H. (2014). League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships. *Applied Soft Computing*, 16, 171-200. https://doi.org/10.1016/j.asoc.2013.12.005

Kaveh, A.ve Bakhshpoori, T. (2016). A new metaheuristic for continuous structural optimization: water evaporation optimization. *Structural and Multidisciplinary Optimization*, *54*(1), 23-43. https://doi.org/10.1007/s00158-015-1396-8

Kaveh, A. ve Mahdavi, V.R. (2014). Colliding bodies optimization: a novel meta-heuristic method, *Computers & Structures*, 139,18-27. https://doi.org/10.1016/j.compstruc.2014.04.005.

Kavitha, A. R. ve Chellamuthu, C. (2016). Brain tumour segmentation from MRI image using genetic algorithm with fuzzy initialisation and seeded modified region growing (GFSMRG) method. *The Imaging Science Journal*, 64(5), 285–297. https://doi.org/10.1080/13682199.2016.1178412

Kelsey, J. ve Timmis, J. (2003). Immune inspired somatic contiguous hypermutation for function optimisation. In *Genetic and Evolutionary Computation Conference*, 207-218. https://doi.org/10.1007/3-540-45105-6_26

Kennedy, J. (2003). Bare bones particle swarms. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03*, 80-87. https://doi.org/10.1109/sis.2003.1202251

Kennedy, J. ve Eberhart, R. (1995). Particle swarm optimization. *IEEE Proceedings of ICNN'95-International Conference on Neural Networks*, 4. 1942-1948. 10.1109/ICNN.1995.488968

Khishe, M. ve Mosavi, M. R. (2020). Chimp optimization algorithm. *Expert Systems with Applications*, 149, 113338. 10.1016/j.eswa.2020.113338

Kirkpatrick, S., Gelatt, C. D. ve Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680. https://doi.org/10.1126/science.220.4598.671

Kumar, V., Kumar, D. ve Chhabra, J. K. (2016). Improved Grey Wolf Algorithm for Optimization Problems. In *International Symposium on Fusion of Science & Technology*, 428-433. https://doi.org/10.4028/www.scientific.net/amm.851.553

Kuyu, Y. C., ve Vatansever, F. (2016). A new intelligent decision making system combining classical methods, evolutionary algorithms and statistical techniques for optimal digital FIR filter design and their performance evaluation. *AEU-International Journal of Electronics and Communications*, 70(12), 1651-1666. https://doi.org/0.1016/j.aeue.2016.10.004

Kuyu, Y. Ç. (2016). Evrimsel algoritmalarla filtre tasarımları (Yüksek lisans tezi). Erişim adresi: https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

Kuyu, Y. Ç., ve Vatansever F. (2017). Design of IIR Digital Filters Using The Current Evolutionary Algorithm. *5th International Symposium on Innovative Technologies in Engineering and Science*, 359-367.

Kuyu, Y. Ç., ve Vatansever, F. (2018). Real loss minimization in power systems via recent optimization techniques. In *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 1-4. https://doi.org/10.1109/ISMSIT.2018.8567060

Kuyu, Y. Ç., ve Vatansever, F. (2021). Modified forensic-based investigation algorithm for global optimization. *Engineering with Computers*, 1-22. https://doi.org/10.1007/s00366-021-01322-w

Kuyu, Y. Ç., ve Vatansever, F. (2022a). GOZDE: A novel metaheuristic algorithm for global optimization. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2022.05.022

Kuyu, Y. Ç., ve Vatansever, F. (2022b). Heap-based optimizer embedded with search strategies applied to high order analog filter designs: A comparative study with up-to-date metaheuristics. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-022-07835-9

Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems, Stanford, CA: Stanford University.

Lewis, H. R. (1983). Computers and intractability. A guide to the theory of NP-completeness, Freeman.

Li, M. D., Zhao, H., Weng, X. W. ve Han, T. (2016). A novel nature-inspired algorithm for optimization: Virus colony search. *Advances in Engineering Software*, 92, 65-88. https://doi.org/10.1016/j.advengsoft.2015.11.004

Lee, C. K. H. (2018). A review of applications of genetic algorithms in operations management. *Engineering Applications of Artificial Intelligence*, 76:1–12. https://doi.org/10.1016/j.engappai.2018.08.011

Lee, M. H., Han, C. ve Chang, K. S. (1999). Dynamic optimization of a continuous polymer reactor using a modified differential evolution. *Industrial & Engineering Chemistry Research*, 38 (12), 4825-4831. https://doi.org/10.1021/ie980373x

Lee K.S. ve Geem Z.W. (2004). A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194, 3902–3933. https://doi.org/10.1016/j.cma.2004.09.007

Lorenzo, B. ve Glisic, S. (2013). Optimal routing and traffic scheduling for multihop cellular networks using genetic algorithm. *IEEE Transactions on Mobile Computing*, 12(11), 2274–2288. https://doi.org/10.1109/TMC.2012.204

Mahdavi, S., Rahnamayan, S. ve Deb, K. (2018). Opposition based learning: A literature review. *Swarm and Evolutionary Computation*, *39*, 1-23. https://doi.org/10.1016/j.swevo.2017.09.010

Mancini, R. 2002. Op Amps for Everyone, Texas Instruments Incorporated, USA.

Meza, J. C. (2011). Newton's method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(1), 75-78. https://doi.org/10.1002/wics.129

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H. ve Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, 114, 163-191. https://doi.org/10.1016/j.advengsoft.2017.07.002

Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-based Systems*, 96, 120-133. https://doi.org/10.1109/icspis.2017.8311581

Mirjalili, S. ve Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili, S., Mirjalili, S. M. ve Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*, *27*(2), 495-513. https://doi.org/10.1007/s00521-015-1870-7

Mirjalili, S. (2015a). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-based Systems*, 89, 228-249. https://doi.org/10.1016/j.knosys.2015.07.006

Mirjalili, S. (2015b). The ant lion optimizer, *Advances in Engineering Software*. 83, 80-98. https://doi.org/10.1016/j.advengsoft.2015.01.010

Mirjalili, S., Mirjalili, S. M. ve Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Muthiah-Nakarajan, V. ve Noel, M. M. (2016). Galactic Swarm Optimization: A new global optimization metaheuristic inspired by galactic motion. *Applied Soft Computing*, 38, 771-787. https://doi.org/10.1016/j.asoc.2015.10.034

Mora-Gutiérrez, R. A., Ramírez-Rodríguez, J. ve Rincón-García, E. A. (2014). An optimization algorithm inspired by musical composition. *Artificial Intelligence Review*, 41(3), 301-315. https://doi.org/10.1007/s10462-011-9309-8

Mousavirad, S. J. ve Ebrahimpour-Komleh, H. (2017). Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence*, 47(3), 850-887. https://doi.org/10.1007/s10489-017-0903-6

Sharma, M. ve Kaur, P (2020). A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. *Arch Comput Methods Eng.* https://doi.org/10.1007/s11831-020-09412-6

Meng, Z., Li, G., Wang, X., Sait, S. M., ve Yıldız, A. R. (2020). A comparative study of metaheuristic algorithms for reliability-based design optimization problems. *Arch Comput. Methods*. https://doi.org/10.1007/s11831-020-09443-z

Oliveira J. A. H., Rembold P., M., Petraglia, A., Ingber, L. ve Augusta S. M. M. (2012). Stochastic global optimization and its applications with fuzzy adaptive simulated annealing., Springer Publishing Company.

Opara, K. ve Arabas, J. (2018). Comparison of mutation strategies in differential evolution–a probabilistic perspective. *Swarm and Evolutionary Computation*, 39, 53-69. https://doi.org/10.1016/j.swevo.2017.12.007

Osman, I. H. (2003). Focused issue on applied meta-heuristics. *Computers & industrial engineering*, 44(2). https://doi.org/10.1016/s0360-8352(02)00175-4

Osman, I. H. ve Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63, 513–562. https://doi.org/10.1007/bf02125421

Patel, H. S. ve Hoft, R. G. (1973). Generalized techniques of harmonic elimination and voltage control in thyristor inverters: part I-harmonic elimination. *IEEE Transactions on Industry Applications*, 9(3), 310-317. https://doi.org/10.1109/tia.1973.349908.

Paiva, F. A. P., Silva, C. R. M., Leite, I. V. O., Marcone, M. H. F. ve Costa J. A. F. (2017). Modified bat algorithm with Cauchy mutation and elite opposition-based learning. *IEEE Latin American Conference on Computational Intelligence*, 1–6. https://doi.org/10.1109/la-cci.2017.8285715

Patel, V. K., ve Savsani, V. J. (2015). Heat transfer search (HTS): a novel optimization algorithm. *Information Sciences*, 324, 217-246. https://doi.org/10.1016/j.ins.2015.06.044

Punnathanam, V. ve Kotecha, P. (2016). Yin-Yang-pair Optimization: A novel lightweight optimization algorithm. *Engineering Applications of Artificial Intelligence*, 54, 62-79. https://doi.org/10.1016/j.engappai.2016.04.004

Rajaram, R., Palanisamy, K., Ramasamy, S. ve Ramanathan, P. (2015). Selective harmonic elimination in PWM inverter using fire fly and fire works algorithm. *International Journal of Innovative Research in Advanced Engineering*, 1(8), 55–62.

Rao, A. P., ve Sarma, N. V. S. N. (2017). Synthesis of reconfigurable antenna array using differential evolution algorithm. *IETE Journal of Research*, 63(3), 428-434. https://doi.org/10.1080/03772063.2017.1284614

Rao, S. S. (1996). Engineering Optimization: Theory and Practice. Wiley, USA.

Rao, R. V., Savsani, V. J. ve Vakharia, D. P. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3), 303-315. https://doi.org//10.1016/j.cad.2010.12.015

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences*, 179(13), 2232-2248. https://doi.org/10.1016/j.ins.2009.03.004

Rechenberg, I. (1973). Evolution strategy: Optimization of technical systems by means of biological evolution. *Fromman-Holzboog, Stuttgart*, 104, 15-16.

Reynolds, R. G. (1994). An introduction to cultural algorithms. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, 131-139. https://doi.org/10.1142/9789814534116

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Ruiz-Vanoye, J. A., Díaz-Parra, O., Cocón, F., Soto, A., Arias, M. D. L. Á. B., Verduzco-Reyes, G. ve Alberto-Lira, R. (2012). Meta-heuristics algorithms based on the grouping of animals by social behavior for the traveling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 3(3), 104-123.

Salem, S. A. (2012). BOA: A novel optimization algorithm. *International Conference on Engineering and Technology (ICET)*, 1-5. https://doi.org/10.1109/icengtechnol.2012.6396156

Saremi, S., Mirjalili, S. ve Lewis, A. (2017). Grasshopper optimisation algorithm: theory and application. *Advances in Engineering Software*, 105, 30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004

Sastry, K.K.N., Behera, L. ve Nagrath, I. J. (1998). Differential evolution based fuzzy logic controller for nonlinear process control. *Fundamenta Informaticae: Special Issue on Soft Computation*, 37:121-136. https://doi.org/10.3233/FI-1999-371207

Shareef, H., Ibrahim, A. A. ve Mutlag, A. H. (2015). Lightning search algorithm. *Applied Soft Computing*, 36, 315-333. https://doi.org/10.1016/j.asoc.2015.07.028

Shi, Y. (2011). Brain storm optimization algorithm. In *International Conference in Swarm Intelligence*, 303-309.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary computation*, 12(6), 702-713. https://doi.org/10.1109/tevc.2008.919004

Simon, D. (2013). Evolutionary optimization algorithms. USA: Wiley.

Sivanandam, S. N. ve Deepa, S. N. (2008). Genetic algorithm optimization problems. In Introduction to Genetic Algorithms, Germany :Springer.

Snaselova, P. ve Zboril, F. (2015). Genetic algorithm using theory of chaos. *Procedia Computer Science*, 51, 316-325. https://doi.org/10.1016/j.procs.2015.05.248

Spears, W. M. (1993). Crossover or mutation?. *Foundations of genetic algorithms*, 2, 221-237. https://doi.org/10.1016/B978-0-08-094832-4.50020-9

Sharma, A, ve Mathur, S. (2018). A novel adaptive beamforming with reduced side lobe level using GSA (2018). *COMPEL-The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*. 37(6), 2263–2278. https://doi.org/10.1108/COMPEL-07-2017-0311

Stützle, T. ve Hoos, H. H. (2000). MAX–MIN ant system. *Future Generation Computer Systems*, 16(8), 889-914. https://doi.org/10.1016/S0167-739X(00)00043-1

Storn, R. ve Price, K. (1995). Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *International Computer Science Institute, Technical report, TR-95.012.*

Storn, R. ve Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341-359. https://doi.org/10.1023/A:1008202821328

Tamer, S. ve Karakuzu, C. (2006). Parçacık sürü optimizasyon algoritması ve benzetim örnekleri. *Elektrik-Elektronik ve Bilgisayar Sempozyumu-ELECO*, Bursa.

Tanyildizi, E. ve Demir, G. (2017). Golden sine algorithm: A novel math-inspired algorithm. *Advances in Electrical and Computer Engineering*, 17(2), 71-78. https://doi.org/10.4316/aece.2017.02010

Tizhoosh, H. R. (2005). Opposition-based learning: a new scheme for machine intelligence. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 1, 695-701.
https://doi.org/10.1016/10.1109/CIMCA.2005.1631345

Trivedi, I. N., Pradeep, J., Narottam, J., Arvind, K. ve Dilip, L. (2016). Novel adaptive whale optimization algorithm for global optimization. *Indian Journal of Science and Technology*, 9(38), 319-326. https://doi.org/10.17485/ijst/2016/v9i38/101939

Vatansever, F., ve Kuyu, Y. C. (2019). The harmonic elimination in inverters with metaheuristic approaches. *Uludağ University Journal of The Faculty of Engineering*, 24(3), 383-396. https://doi.org/10.17482/uumfd.595233

Walton, S., Hassan, O., Morgan, K. ve Brown, M. R. (2011). Modified cuckoo search: a new gradient free optimisation algorithm. *Chaos, Solitons & Fractals*, 44(9), 710-718. https://doi.org/10.1016/j.chaos.2011.06.004

Wang, H., Rahnamayan, S., Sun, H. ve Omran, M. G. (2013). Gaussian bare-bones differential evolution. *IEEE Transactions on Cybernetics*, 43(2), 634-647. https://doi.org/0.1109/tsmcb.2012.2213808

Wang, R.J. Zhan, Y.J. ve Zhou, H.F. (2015). Application of artificial bee colony in model parameter identification of solar cells. *Energies*, 8, 7563-7581. https://doi.org/10.3390/en8087563

Wang, F.S., Jing, C.H. ve Tsao, G.T. (1998). Fuzzy-decision-making problems of fuel ethanol production using genetically engineered yeast, *Industrial & Engineering Chemistry Research*, 37:3434-3443. https://doi.org/10.1021/ie970736d

Wang, F. S. ve Cheng, W. M. (1999). Simultaneous optimization of feeding rate and operation parameters for fed-batch fermentation processes. *Biotechnology Progress*, 15 (5), 949-952. https://doi.org/10.1021/bp990088o

Wolpert, D. H. ve Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. https://doi.org/10.1109/4235.585893

Xin, Y., Yong, L., ve Guangming, L. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2), 82–102. https://doi.org/10.1109/4235.771163

Yang, X., Huang, Z. (2011). Artificial bee colony with dynamic Cauchy mutation for numerical optimization. *Journal of Information and Computing Science*, 8 (15), 3371–3376.

Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, 65-74. https://doi.org/10.1007/978-3-642-12538-6_6

Yang, X. S. (2012). Flower pollination algorithm for global optimization. In *International Conference on Unconventional Computing and Natural Computation*, 240-249. https://doi.org/10.1007/978-3-642-32894-7_27

Yang, X. S. (2010). Engineering optimization: an introduction with metaheuristic applications. USA: Wiley.

Yang X. S. ve Deb, S. (2009). Cuckoo search via lévy flights. *World Congress on Nature&Biologically Inspired Computing*, 210-214. https://doi.org/10.1109/NABIC.2009.5393690

Yang, X. S. ve Deb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications*, 24, 169-174. https://doi.org/10.1007/s00521-013-1367-1

Yazdani, M. ve Jolai, F. (2016). Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, *3*(1), 24-36. https://doi.org/10.1016/j.jcde.2015.06.003

Ye, M.Y., Wang, X.D. ve Xu, Y.S. (2009). Parameter extraction of solar cells using particle swarm optimization. *Journal of Applied Physics*, 105 (9). https://doi.org/10.1063/1.3122082

Yun, S., Lee, J., Chung, W., Kim, E. ve Kim, S. (2009). A soft computing approach to localization in wireless sensor networks. *Expert System Applications*, 36(4), 7552–7561. https://doi.org/10.1016/j.eswa.2008.09.064

Zagrouba, M., Sellami, A., Bouaicha, M. ve Ksouri, M. (2010). Identification of PV solar cells and modules parameters using the genetic algorithms: Application to maximum power extraction. *Solar Energy*, 84(5), 860-866. https://doi.org/10.1016/j.solener.2010.02.012

Zou, F., Wang, L., Hei, X., Chen, D., Jiang, Q. ve Li, H. (2014). Bare-bones teaching-learning-based optimization. *The Scientific World Journal*, 1-17. https://doi.org/10.1155/2014/136920

ÖZGEÇMİŞ

Adı Soyadı : Yiğit Çağatay KUYU
Doğum Yeri ve Tarihi : Samsun, 08/02/1989
Yabancı Dil : İngilizce
Eğitim Durumu
Lise : Samsun Anadolu Lisesi
Lisans : Bursa Uludağ Üniversitesi
Yüksek Lisans : Bursa Uludağ Üniversitesi

Çalıştığı Kurum(lar) : KARSAN Otomotiv 2021-
DeustoTech 2020-2021
Bursa Uludağ Üniversitesi 2015-2020

İletişim (e-posta) : yigitkuyu@gmail.com

Tezden Çıkan Akademik Çalışmalar :

Kuyu, Y. Ç., ve Vatansever, F. (2021). Modified forensic-based investigation algorithm for global optimization. Engineering with Computers, 1-22. https://doi.org/10.1007/s00366-021-01322-w

Kuyu, Y. Ç., ve Vatansever, F. (2022). GOZDE: A novel metaheuristic algorithm for global optimization. Future Generation Computer Systems. https://doi.org/10.1016/j.future.2022.05.022

Kuyu, Y. Ç., ve Vatansever, F. (2022). Heap-based optimizer embedded with search strategies applied to high order analog filter designs: A comparative study with up-to-date metaheuristics. Neural Computing and Applications. https://doi.org/10.1007/s00521-022-07835-9