Journal of Quantum Technologies and Informatics Research
JQTAIR

KUANTUM TEKNOLOJİLERİ VE ENFORMATİK ARAŞTIRMALARI
Volume / Cilt: 2, Issue / Sayı: 2, 2024, pp. 87-103
E - ISSN: 3023-4735
URL: https://journals.gen.tr/index.php/jqtair/
DOI: https://doi.org/10.70447/ktve.2459

Received / Geliş: 23.06.2024
Acccepted / Kabul: 17.07.2024

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

PSO ve BA Algoritmalarının Farklı Test Fonksiyonları Üzerinde Karşılaştırmalı Analizi

Comparative Analysis of Particle Swarm Optimization and Bat Algorithm on Various Test Functions

İbrahim Işıklı[1] **Bayram Köse**[2]

[1] İzmir Bakırçay Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü, İzmir, Türkiye, e-mail: isikli.ibrahim@outlook.com

[2] İzmir Bakırçay Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü, İzmir, Türkiye, e-mail: bayram.kose@bakircay.edu.tr

Öz

Bu çalışmada, Parçacık Sürü Optimizasyonu (PSO) ve Yarasa Algoritması (BA) gibi doğadan ilham alan optimizasyon algoritmalarının performanslarını çeşitli test fonksiyonları üzerinde karşılaştırılmıştır. Analizler, her iki algoritmanın farklı test fonksiyonları üzerindeki performansını değerlendirmektedir. Bulgular, her iki algoritmanın da belirli test fonksiyonları için benzersiz güçlü ve zayıf yönlerine işaret etmektedir. Çalışmada, doğadan ilham alan optimizasyon algoritmalarının optimizasyon problemlerine yönelik etkili çözümler sağlama potansiyelini vurgulamakta ve gelecekteki araştırmalara yol göstermektedir.

Anahtar kelimeler: Parçacık Sürü Optimizasyonu, Yarasa Algoritması, Karşılaştırmalı Analiz, Test Fonksiyonları, Optimizasyon Problemleri.

Abstract

This study compares the performance of nature-inspired optimization algorithms such as Particle Swarm Optimization (PSO) and Bat Algorithm (BA) on various test functions. The analyses evaluate the performance of both algorithms on different test functions. The findings point out the unique strengths and weaknesses of both algorithms for specific test functions. The study highlights the potential of nature-inspired optimization algorithms to provide effective solutions to optimization problems and provides directions for future research.

Keywords: Particle Swarm Optimization, Bat Algorithm, Comparative Analysis, Test Functions, Optimization Problems.

Citation/Atıf: IŞIKLI, İ. & KÖSE, B. (2024). PSO ve BA Algoritmalarının Farklı Test Fonksiyonları Üzerinde Karşılaştırmalı Analizi. *Kuantum Teknolojileri ve Enformatik Araştırmaları*. 2(2): 87-103, DOI: 10.70447/ktve.2459

Corresponding Author/ Sorumlu Yazar:
İbrahim Işıklı
E-mail: isikli.ibrahim@outlook.com

1. GİRİŞ

Optimizasyon, belirli kısıtlamalar olsun veya olmasın, belirlenen amaç veya amaçlar doğrultusunda en uygun çözümü elde etme sürecidir. Bu süreç, bilim insanlarının yeni fikirler ortaya koyması ve bu fikirlerin geliştirilmesi açısından büyük önem taşır. Bir fikrin etkileyen parametrelerinin veya bilgilerinin elektronik formata dönüştürülebildiği sürece, bilgisayarlar mükemmel bir optimizasyon aracı olarak hizmet verebilir. Optimizasyon terminolojisinde her zaman en en uygun sonuca ulaşma hedefi vardır. Ancak bu en iyi tanımı, problemin doğasına, kullanılan çözüm metoduna ve izin verilen toleranslara bağlı olarak değişiklik gösterebilir. Tarih boyunca karşılaşılan problemleri çözmek amacıyla birçok optimizasyon tekniği geliştirilmiş ve bu teknikler farklı alanlara uygulanmıştır [1].

Günümüzde bilimsel çalışmalar ve teknolojik araçlar her an ivmeli artış ile gelişime devam etmektedir. Bu gelişmelerin başında olan optimizasyon (en iyileme), en büyükleme ve en küçükleme süreçlerinin tümel bir adı olup, ekonomiden tasarım mühendisliğine, otomasyondan enerji mühendisliğine hemen her bilim dalını ilgilendirir. Örneğin rüzgar çiftliklerinde, rüzgar türbinlerinin optimum yerleşimini hesaplanması [1]; akıllı bina ortamlarındaki enerji tüketiminin en aza indirgenmesi hedefleri optimizasyon işlemleridir [2].

Mühendislikte optimizasyon problemleri çözmede başvurulan klasik yöntemler, gerçek hayat problemlerini çözümünde kifayetsiz kaldığı durumlar ortaya çıkabilmektedir. Çözüm süresinin ve çözüm karmaşıklığının problem boyutuna bağlı olarak çok artması, sezgisel arama yöntemlerinin gelişimi sağlamıştır. Sezgisel arama yöntemleri ise bilgisayar teknolojilerinin gelişimi ile paralel olarak son kırk yıldır gelişmektedir. Tüm canlı türlerinin optimum tasarımda ideal bir şekilde ekosistemde bulunması, davranışlarında ve yiyecek bulma arayışlarında optimum çözümü bulma girişimleri, bilim insanlarının ilgisini çekmiş ve doğadan ilham alan algoritmaların çoğalmasıyla sonuçlanmıştır.

Optimizasyon problemleri çözümünde klasik yöntemler olarak isimlendirilen analitik yöntemler çok yaygın olarak kullanılmakla birlikte bu yöntemlerin esnek olmaması ve matematiksel fonksiyonlarla tanımlı olma ihtiyacı gibi dezavantajları, bilim insanlarını daha genel amaçlı ve yüksek performanslı yöntemler geliştirmeye yönlendirmiştir. Son zamanlarda, doğa olaylarından esinlenerek geliştirilen optimizasyon algoritmaları, sezgisel yöntemler olarak adlandırılmaktadır [3].

Bu sezgisel yöntemler arasında J. Holland tarafından 1975'te geliştirilen (GA) [4]; R. Storn tarafından 1997'de geliştirilen Diferansiyel Evrim Algoritması (DEA) [5]; X. Yang tarafından geliştirilen Yarasa Algoritması (BA) [6] ve J. Kennedy tarafından geliştirilen Parçacık Sürü Optimizasyonu (PSO) algoritmaları [7], optimizasyon problemlerinde popüler olarak kullanılmaktadır.

Sezgisel arama yöntemleri, 1970'lerden beri gelişmekte olan ve birçoğu doğadan esinlenilerek modellenen yöntemlerdir [8]. Bu yöntemlerin bazıları doğal fiziksel yasaları veya süreçleri temel alırken, büyük bir kısmı canlı varlıkların sosyal etkileşimlerini örnek almıştır. Örneğin, Genetik Algoritmalar (GA), canlılardaki genetik kalıtım süreçlerini örnek alarak geliştirilen popülasyon tabanlı bir evrim algoritmasıdır ve her bir jenerasyonda en iyi sonuca ulaşmayı hedefler. GA, doğal seçilim ve kalıtım prensiplerine dayanarak, popülasyon içerisindeki en iyi çözümleri seçer ve bu çözümleri çaprazlama ve mutasyon gibi genetik işlemlerle birleştirerek daha iyi çözümler üretmeye çalışır [9].

(DEA) ise, genetik çaprazlama metodu ve bireysel farklılıklardan esinlenerek geliştirilmiştir. Bu genetik çaprazlama işlemi, popülasyonun farklı bireyleri arasındaki varyasyonu kullanarak, global optimuma yakın sonuçlar elde etmeyi amaçlar [10]. DEA'nın temel avantajı, arama alanında geniş bir keşif yapabilme yeteneğidir, bu da onu karmaşık ve çok boyutlu problemlerde oldukça etkili kılar. (DEA), çözümleri birbirine yakınlaştırarak lokal minimumlara takılmadan global en iyi çözüme ulaşmayı hedefler. Ancak, bu hedef sadece DEA'ya özgü olmayıp, diğer sezgisel algoritmalar da benzer şekilde global en iyi çözüme ulaşmayı amaçlar [10].

Diğer bir sezgisel yöntemlerden biri de topluluktaki bireylerin birbiri ile etkileşimini baz alan (PSO) algoritmasıdır. PSO, sosyal davranışları

ve grup dinamiklerini taklit eder; sürüdeki her bir birey (parçacık), kendi ve komşularının deneyimlerinden öğrenerek, en iyi çözüme doğru hareket eder [11].

(BA) ise, yarasaların eko-lokasyon davranışlarını taklit eden bir diğer sezgisel optimizasyon yöntemidir. Yarasa Algoritması, yarasaların avlarını bulmak için kullandıkları yankı (eko) sinyallerini model alır ve bu sinyalleri kullanarak arama alanında çözümleri keşfeder [12]. Bu algoritma, frekans ve yankı gücünü ayarlayarak, yerel ve global arama yeteneklerini dengeler, bu da onu çeşitli optimizasyon problemlerinde etkili kılar [13].

Sezgisel araştırma yöntemleri, doğadaki canlıların ve fiziksel süreçlerin davranışlarını taklit ederek geliştirilmiş ve bu yöntemler, klasik matematiksel yöntemlerin yetersiz kaldığı durumlarda güçlü alternatifler sunmuştur. (DEA), (GA), (BA) ve (PSO) gibi yöntemler, geniş bir uygulama yelpazesinde başarılı sonuçlar elde etmiş ve optimizasyon problemlerinin çözümünde önemli bir rol oynamıştır. Bu yöntemlerin her biri, belirli bir problemi çözmek için uygun olan kendine özgü avantajlara sahiptir ve doğru seçildiklerinde, karmaşık problemlerin çözümünde büyük bir fark yaratabilirler.

Sezgisel optimizasyon tekniklerinin performansını ölçmek için kullanılan en önemli kriterlerden biri yakınsama hızıdır [14]. Bu çalışmada, test fonksiyonları kullanılarak (PSO) algoritması ile (BA) karşılaştırılmıştır. Makalenin organizasyonu şu şekildedir: Bölüm 2 ve alt başlıklarında PSO ve BA algoritmaları ile ilgili genel bilgiler verilmektedir. Bölüm 3'te ise test fonksiyonları ele alınmakta ve benzetim sonuçları sunulmaktadır. Son olarak, Bölüm 4'te genel sonuçlar ve değerlendirmeler yapılmaktadır.

Çalışma (PSO) ve Yarasa Algoritması (BA) algoritmalarının performansını karşılaştırmıştır. Analizler, PSO'nun Beale, Ackley, Levi ve Goldstein Price test fonksiyonlarında genellikle daha iyi sonuçlar verdiğini, ancak Cross in Tray ve Bukin fonksiyonlarında BA'nın üstün performans sergilediğini göstermiştir. Bu bulgular, her iki algoritmanın da farklı problem türlerinde avantajlar sunduğunu ve algoritma seçiminde problem yapısının dikkate alınması gerektiğini ortaya koymaktadır.

2. MATERYAL VE METOD

Araştırmacılar, doğanın akıllı sistemlerin geliştirilmesi için büyük bir ilham kaynağı olduğunu ve çeşitli karmaşık problemlere yenilikçi çözümler sunduğunu fark etmişlerdir. Bu gözlem, doğadan esinlenerek geliştirilen birçok algoritmanın ortaya çıkmasına neden olmuştur. Bu algoritmaların temelinde, doğadaki süreçlerin ve organizmaların karşılaştıkları zorlukları çözme biçimlerinin taklit edilmesi yatmaktadır. Bu yaklaşım, doğanın sunduğu muazzam bilgi birikimini ve adaptasyon yeteneklerini kullanarak, teknik ve mühendislik problemlerine etkili çözümler üretmeyi amaçlar. Bu çalışmada doğa esinli algoritmalar arasında özellikle dikkat çeken temel bir algoritma olan (PSO) ile (BA) analiz edilmiştir.

2.1. Parçacık Sürü Optimizasyonu

(PSO), Elektrik mühendisi olan Russel Eberhat ve psikolog olan James Kennedy tarafından geliştirilmiş bir optimizasyon algoritmasıdır [15]. PSO, toplu zekâ kavramına dayanarak, sosyal etkileşimlerin ve sürü davranışlarının matematiksel bir modelini oluşturur. Algoritmanın temel fikri, biyolojik toplumlardaki toplu zekâ olgusunun araştırılması sırasında, kuş sürülerinin ve balık gruplarının hareketlerinin modellenmesiyle ortaya çıkmıştır. Bu bağlamda PSO, yapay yaşam ve evrimsel programlama gibi iki ana disiplinin bir sentezidir [16].

Yapay yaşam teorisi, sürülerin, kuşların ve balıkların kolektif hareketlerini incelerken, evrimsel programlama ise doğal seleksiyon ve genetik algoritmalar gibi biyolojik evrim süreçlerini taklit eden bir optimizasyon tekniğidir. PSO, bu iki yaklaşımı birleştirerek, toplulukların bilgi paylaşımı ve bireysel deneyimlerinden yararlanarak optimizasyon problemlerini çözmeyi hedefler [15].

PSO'nun en dikkat çekici özelliklerinden biri, topluluk temelli bir yapı üzerine kurulmuş olması ve bu topluluğun dinamik etkileşimleriyle işleyen bir mekanizmaya sahip olmasıdır. Diğer evrimsel hesaplama yöntemlerinden farklı olarak, PSO sadece basit matematiksel operatörler kullanarak çalışır ve bu sayede daha az hesaplama gerektiren, daha verimli bir algoritma sunar. Kennedy ve Eberhart, (PSO) çeşitli global opti-

mizasyon problemlerinde aktif bir çözüm yöntemi olduğunu ve geri kalan evrimsel algoritmaların yüzleştiği bazı güçlüklerin, özellikle yerel minimumlara takılma ve yüksek hesaplama maliyetleri gibi, üstesinden kolaylıkla gelebildiğini göstermişlerdir [15].

PSO, rastgele evrimsel hesaplama yöntemleri arasında yer alır ve bu yöntem, sürülerin zekâsına ve hareketlerine dayalı bir benzetme ile açıklanabilir. Örneğin, bir tarladaki arı topluluğunu ele alalım. Bu arıların hedefi, tarladaki en yoğun çiçek bölgesini bulmaktır. Arıların daha önce tarla ile ilgili bir bilgiye sahip olmadığını kabul ettiğimizde; arılar, rastgele bölgelerde rassal hızlarla uçarak çiçek aramaya başlar. Her bir arı, en çok çiçeği bulduğu bölgeyi belleğinde tutar. Ayrıyeten, diğer arıların bulduğu çiçek miktarının en çok olan bölgeyi de bir şekilde bilmektedir [16].

Arıların hareketleri, kendi deneyimleri ile topluluk bilgisinin bir kombinasyonuna dayanır. Kendi bulduğu çiçek yoğunluğu en fazla olan bölge ile sürünün genel olarak belirlediği en yoğun bölge arasında bir denge kurmaya çalışır. Yolculuğu sırasında, daha önce bulduğundan daha fazla çiçeğe sahip bir bölge keşfederse, bu yeni bilgiye göre hareket etmeye başlar. Eğer bir arı, sürüdeki diğer arıların bulduğundan daha yoğun bir çiçek bölgesi bulursa, tüm sürü bu yeni bölgeye yönelir. Bu süreç, arıların tüm tarlayı taramasını, en yoğun çiçek bölgelerini bulmasını ve sonunda o bölgelerde toplanmasını sağlar [17].

Bu benzetme, PSO'nun nasıl çalıştığını anlamak için kullanışlıdır: Her bir "parçacık" (arı), arama uzayında potansiyel çözümler arar ve kendi deneyimlerinden ve diğer parçacıkların deneyimlerinden yararlanarak en iyi çözüme ulaşmaya çalışır. Parçacıklar, kendi en iyi bilinen konumları ile sürünün en iyi bilinen konumu arasında bir denge kurarak hareket ederler. Bu süreç, PSO'nun global optimizasyon problemlerini etkili bir şekilde çözmesini sağlar ve algoritmanın adaptif doğası, çeşitli zorluklara karşı esneklik sunar.

Parçacık Sürü Optimizasyonu, toplu zekâ ve sosyal etkileşimlere dayalı yenilikçi bir optimizasyon algoritması olarak, biyolojik sistemlerin davranışlarını modelleyerek geniş bir uygulama yelpazesinde başarılı sonuçlar elde etmektedir. PSO'nun basitliği ve etkinliği, onu evrimsel hesaplama yöntemleri arasında öne çıkaran önemli bir faktördür.

2.1.1. PSO Algoritmasının Temel Birimleri

Bu başlıkta, (PSO) algoritmasının temel kavramları ve terminolojisi detaylı bir şekilde tanıtılacaktır. Bu terimler, PSO'nun diğer evrimsel hesaplama yöntemlerinden nasıl farklılaştığını anlamamıza yardımcı olacaktır.

Parçacık: PSO algoritmasında, bireyler "parçacık" şeklinde isimlendirilir. Her bir parçacık, arama uzayında potansiyel bir çözüm olarak değerlendirilir ve belirli bir stratejiye göre hareket eder. Parçacıklar, her an kendi konumlarını günceller ve bu güncelleme sırasında iki temel prensibe dayanırlar: kendi kişisel en iyi konumlarına (pbest) ve sürünün genel en iyi konumuna (gbest) doğru yönelirler. Bu şekilde, parçacıklar hem bireysel deneyimlerinden hem de topluluk bilgeliğinden faydalanarak en iyi çözüme ulaşmaya çalışırlar.

Konum: PSO'da "konum", optimizasyon problemi için bir çözüm kümesini temsil eder. Arama uzayı, problemin boyutuna bağlı olarak n-boyutlu bir uzay olarak tanımlanır. Her bir konum, bu uzayda bir koordinat olarak değerlendirilir. Örneğin, üç boyutlu bir optimizasyon probleminde, çözüm uzayı x-y-z koordinat düzleminde tanımlanır ve herhangi bir (x›, y›, z›) konumu potansiyel bir çözüm olarak kabul edilir. PSO'nun amacı, bu çözüm uzayında en uygun sonucu sağlayan konumu bulmaktır.

Sürü: Bütün parçacıkların oluşturduğu topluluğa "sürü" denir. Sürü, PSO'nun temel yapı taşlarından biridir ve parçacıkların toplu hareketi ile optimize edilir. Her bir parçacık, sürünün genel en iyi konumuna ulaşmak için diğer parçacıklarla etkileşimde bulunur ve bu sayede sürü, kolektif bir şekilde en uygun çözümü bulmaya çalışır.

Hız: PSO'da hız, bir parçacığın en iyi konuma ulaşmak için kullandığı yön vektörüdür. Bu hız vektörü, parçacığın kendi kişisel en iyi konumu ile sürünün genel en iyi konumu arasında bir denge kurarak güncellenir ve parçacığın bir sonraki konumuna nasıl hareket edeceğini belirler.

Uygunluk: Evrimsel hesaplama tekniklerinde olduğu gibi, PSO'da da "uygunluk fonksiyonu" olası konumların uygunluğunu değerlendirmek için kullanılır. Uygunluk fonksiyonu, belirli bir konumun performansını değerlendirir ve o konumdan tek bir sayı değeri üretir. Uygunluk fonksiyonu, optimizasyon probleminin doğasına bağlı olarak farklılık gösterebilir ve nihai olarak hedeflenen uygunluk değerine ulaşmayı amaçlar.

Kişisel En İyi (*pbest*): Her bir parçacığın vardığı konumlar arasında en iyi uygunluk değerine sahip olan konum "kişisel en iyi" (*pbest*) olarak adlandırılır. Parçacık, daha sonraki hızını ve konumunu belirlerken bu kişisel en iyi konumunu referans alır. Parçacık, mevcut konumunu sürekli olarak kendi kişisel en iyi konumuyla karşılaştırır ve daha iyi bir uygunluk değeri elde ederse, bu yeni konumu kişisel en iyi konumu olarak günceller.

Küresel En İyi (gbest): "Küresel en iyi" (gbest), sürünün ulaştığı en iyi konumu temsil eder. Sürüdeki tüm parçacıklar, bu küresel en iyi konumu bilir ve kendi hareketlerini bu bilgiye göre düzenlerler. Parçacıklar, mevcut konumunu sürünün küresel en iyi konumuyla karşılaştırır ve daha iyi bir uygunluk değeri elde ederse, bu yeni konumu sürünün küresel en iyi konumu olarak günceller.

İterasyon: PSO algoritmasında, sürüdeki parçacıkların zaman adımlarını temsil eden bir terimdir. İterasyon, parçacıkların konumlarını ve hızlarını güncelledikleri her bir zaman dilimini ifade eder. Optimizasyon süreci, belirli bir iterasyon sayısı boyunca veya belirlenen bir kriteri sağlayıncaya kadar devam eder. Her iterasyon, parçacıkların arama uzayında daha iyi çözümler bulmasına olanak tanır.

2.1.2. PSO Algoritmasının İşleyişi

(PSO) algoritması, hayvanların sosyal etkileşimlerinden ilham alan bir optimizasyon yöntemidir. Bu algoritma, "parçacık" olarak adlandırılan bireyler arasındaki etkileşimleri kullanarak, arama uzayında en uygun çözüme uyum sağlayarak ilerlemelerini sağlar.

Parçacık Sürüsü Optimizasyonu (PSO), başlangıçta rastgele oluşturulan bir çözüm kümesi ile başlar ve bu çözümler güncellenerek en uygun sonuca ulaşılmaya çalışılır. Her iterasyonda, parçacıkların konumları en iyi performans gösteren iki parçacığa göre ayarlanır.İlk olarak, bugüne kadar kullanılan aynı numaralı parçacıklar arasında en üst düzeyde uygunluk sergileyen parçacık seçilir ve bu parça yerel en iyi (p_{ye}) olarak isimlendirilir; bu veri bellekte saklanmalıdır. Öteki ise parçacıkların tümü arasında elde edilen en üst düzeyde uygunluk değerine sahip olan global en iyi parçacığıdır. Global en iyi, " g_{best} " olarak tanımlanır. Örneğin, D boyutlu ve N tane parçacık bulunduğunu varsayalım. Bu varsayımda, popülasyon parçacık matrisi, Denklem 1'de belirtilen yapıya sahiptir [18].

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & \cdots & p_{1D} \\ p_{21} & p_{22} & \cdots & \cdots & p_{2D} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{N1} & p_{N2} & \cdots & \cdots & p_{ND} \end{bmatrix}_{N\times D} \quad (1)$$

Denklem 1'deki matriste bulunan i' inci parçacık $p_i = [p_{i1}, p_{i2}, \ldots . p_{iD}]$ifadesiyle belirtilir. Önceki iterasyonlarda, en üst düzeyde uygunluk değerine ulaşan ' inci parçacığın konumu $p_{ye_i} = [p_{ye_{il}}, p_{ye_{i2}}, \ldots \ldots p_{ye_{iD}}]$ olup, bu parçacık yerel en iyisi olarak anılır [17].

Global en iyi parçacık (g_{best}), her iterasyonda tüm parçacıklar için tek bir değerdir ve $g_{\text{best}} = [p_{gb_l}, p_{gb_2}, \ldots \ldots p_{gb_D}]$şeklinde ifade edilir. i' inci parçacığın hızı, her boyuttaki konum değişim miktarı olarak $v_i = [v_{i1}, v_{i2}, \ldots . v_{iD}]$ şeklinde tanımlanır. İki en iyi değerin belirlenmesinin ardından, parçacık hızları ve konumları, aşağıdaki 2 ve 3 numaralı denklemlere [19] göre güncellenir.

$$v_i^{k+1} = v_i^k + c_1 \times x \times r_1^k x\left(p_{ye}^k - p_i^k\right) + c_2 \times r_2^k \times \left(g_{\text{best}}^k - p_i^k\right) \quad (2)$$

$$p_i^{k+1} = p_i^k + v_i^{k+1} \quad (3)$$

Denklem 2'deki c_1 ve c_2, öğrenme faktörlerini ifade ederken k iterasyon sayısını ifade eder. Bu sabitler, her parçacığı kendi yerel en iyisine (p_{ye}) ve sürünün en iyisine (g_{best}) doğru çeken hızlanma terimlerini temsil eder. c_1 ve c_2 sırasıyla lokal ve global öğrenme katsayılarıdır. c_1'in yüksek değerlerde seçilmesi, pbest (lokal en iyi çözüm) çözümünden daha fazla öğrenme sağlamakta ve bu durum çözüm vektörünün lokal en iyi çözüm etrafında yakınsamasını teşvik etmektedir. c_2

'nin yüksek değerlerde seçilmesi ise, gbest (global en iyi çözüm) çözümünden daha fazla öğrenme sağlayarak elde edilen çözüm vektörünün global en iyi çözüm pozisyonuna doğru yakınsamasını desteklemektedir.

Denklem içindeki r_1 ve r_2, (0,1) arasında uniform olarak dağıtılmış rastgele sayılardır. İterasyon sayısını temsil eden k ise belirlenir. PSO algoritmasının genel kod yapısı Şekil.1'de özetlenmiştir [19].

2.2. Yarasa Algoritması

Doğa esinli algoritmalar arasında özellikle dikkat çekenlerden biri de (BA). Bu algoritma, yarasaların ekolokasyon yeteneklerinden ilham alarak geliştirilmiş olup, optimizasyon problemlerinde üstün performans göstermektedir. Yarasaların avlarını bulmak ve engellerden kaçınmak için kullandıkları bu ileri düzey biyolojik mekanizma, algoritmanın temelini oluşturur ve bu sayede Yarasa Algoritması, hem doğanın işleyişini anlamamıza katkıda bulunur hem de mühendislik alanında pratik uygulamalara olanak tanır [20].

Yarasalar, ekolokasyon yetenekleri sayesinde yönlerini bulma konusunda uzmanlaşmış, doğanın en etkileyici canlılarından biridir. Dünya genelinde yaklaşık bine yakın türü bulunduğu tahmin edilen yarasalar, büyüklük ve ağırlık bakımından geniş bir yelpazeye sahiptir. En küçük türlerinden biri olan yaban arısı yarasası sadece 1.5-2 gram ağırlığındadır. Buna karşılık, kanat açıklığı yaklaşık 2 metreye ulaşan ve ağırlığı yaklaşık 1 kilogram olan dev yarasalar da bulunmaktadır [20].

Yarasalar genel olarak mikro yarasalar ve mega yarasalar olarak iki gruba ayrılır. Mikro yarasalar, ekolokasyon becerilerini yoğun bir şekilde kullanarak çevrelerini algılar ve avlarını bulurlar. Mega yarasalar ise genellikle bu yeteneği kullanmazlar ve daha çok görme duyularına güvenirler. Ekolokasyon, yarasaların yüksek frekanslı ses dalgaları yayarak çevrelerindeki nesnelerin yerini tespit etmelerine olanak tanıyan bir tür biyolojik sonar sistemidir [21].

Mikro yarasalar, yüksek frekansta ve kısa sürede ses dalgaları gönderirler; bu dalgalar bir nesneye çarptığında geri dönen yankıyı algılarlar. Bu sayede, nesnenin uzaklığını ve konumunu hassas bir şekilde belirleyebilirler. Yayılan ses dalgalarının süresi yaklaşık 5-20 milisaniye arasında değişir ve genellikle 25-150 kHz frekans aralığında sabitlenir. Mikro yarasalar normalde saniyede yaklaşık 10-20 ses sinyali gönderirler, ancak avlarına yaklaştıklarında bu oran saniyede yaklaşık 200 sinyale kadar artar [21].

Ayrıca, mikro yarasalar av arama sürecinde yüksek sesli sinyaller (yaklaşık 110 desibel) üretirler; avlarına yaklaştıklarında ise bu sinyallerin şiddetini azaltarak daha sessiz hale gelirler. Mikro yarasaların bu ekolokasyon yeteneği, optimize

Şekil 1. Parçacık sürü optimizasyonunun sözde kodu

Başlangıç:

- Popülasyon boyutu, hareket sayısı, boyut, ekin hızı, kişisel ve sosyal hızlanma, doğruluk, minimum hareketler.

Fonksiyon Seçimi:

- Amaç fonksiyonunu seç ve x, y aralıklarını belirle.

Sürü Popülasyonunu Başlat:

- Rastgele pozisyonlar oluştur ve fitness hesapla.

En İyi Fitness Değerini Belirle:

- En iyi fitness değerini ve indeksini belirle.

Dinamikleri Başlat:

- Hız matrisini sıfırla.

Simülasyon:

- Hareket sayısı kadar döngü yap:
- Hızı güncelle (inertia, kişisel ve sosyal bileşenlerle). Pozisyonu güncelle.
- Fitness değerini hesapla.
- En iyi pozisyon ve fitness değerini güncelle.

edilecek bir amaç fonksiyonuyla ilişkilendirilebilir ve bu doğal davranışı taklit eden bir optimizasyon algoritması geliştirilebilir. Bu şekilde, mikro yarasaların ekolokasyon yeteneğinden ilham alınarak matematiksel ve mühendislik problemlerine çözüm üretmek mümkündür. Chawla ve Duhan (2015) bu doğal yeteneği kullanarak bir optimizasyon algoritması formüle etmişlerdir. Bu algoritma, doğanın sunduğu bu mükemmel mekanizmayı kullanarak optimum çözümler bulmayı hedefler [26].

2.2.1. Yarasa Algoritmasının Temel Birimleri

Bu amaç doğrultusunda harekete geçen Yang, yarasaların ekolokasyon yeteneklerini ve av bulma konusundaki karakteristik davranışlarını modelleyerek, 2010 yılında Yarasa Algoritmasını geliştirmiştir [22]. Yarasa algoritması, yarasaların yüksek frekanslı ses dalgalarını yayarak nesnelerin yerini belirleme ve avlarını bulma yeteneklerini matematiksel bir model haline getirmiştir [22]. Bu model, optimizasyon problemlerinin çözümünde kullanılmak üzere doğanın bu etkileyici mekanizmasını taklit eder. Yarasa Algoritması, özellikle karmaşık ve çok boyutlu problemlerde etkin çözümler sunarak, doğa esinli algoritmalar arasında önemli bir yer edinmiştir. Yang'ın çalışması, biyolojik süreçlerin bilgisayar bilimlerine uygulanması konusunda önemli bir adım olarak kabul edilir ve çeşitli mühendislik uygulamalarında başarılı bir şekilde kullanılmıştır.

(BA) Temel Birimleri

Yarasa Algoritması'nın yapısını iyileştirmek ve yarasaların ekolokasyon yeteneklerini analiz etmek amacıyla belirli ilkeler uygulanmaktadır:

Sinyal Yayımı ve Yankı Algılama: Yarasalar, yüksek frekanslı ses dalgaları yayarak çevrelerinde yankı oluşturur. Bu yankılar, avın yerini tespit etmek için yarasaların kulaklarına geri döner. (BA) bu ilke, çözüm arama sürecinde farklı noktaların değerlendirilmesi için kullanılır [23].

Frekans ve Genlik Değişimi: Yarasalar, avlarına yaklaştıkça yayılan sinyallerin frekansını ve genliğini değiştirebilir. Algoritmada bu değişim, çözüm arayışında daha hassas aramalar yapmak için farklı parametrelerin ayarlanması şeklinde uygulanır [23].

Sinyal Gücü ve Mesafe İlişkisi: Yarasalar, yayılan sinyalin gücünü kullanarak mesafeyi tahmin ederler. Bu prensip, algoritmada çözüme olan uzaklığın değerlendirilmesi ve uygun adımların atılması için kullanılır [23].

Dinamik Arama Alanı: Yarasalar, avlarını bulmak için dinamik bir arama alanı kullanır. Algoritmada bu ilke, arama alanının adaptif bir şekilde güncellenmesi ve optimize edilmesi için uygulanır [23].

Sinyal Yayım Hızı: Yarasalar, avlarına yaklaştıklarında sinyal yayım hızını artırır. Bu davranış, algoritmada çözümün doğruluğunu artırmak ve sonuca daha hızlı ulaşmak için kullanılır [23].

Bu ilkeler, Yarasa Algoritması'nın doğadaki yarasaların ekolokasyon yeteneklerini taklit ederek daha etkili ve verimli optimizasyon çözümleri sunmasını sağlar. Yarasaların bu biyolojik yeteneklerinin modellemesi, karmaşık mühendislik problemlerinin çözümünde yararlı bir araç olarak kullanılmaktadır.

2.2.2. (BA) İşleyişi

Yarasalar, avlarını bulmak için belirli bir hızla v_i, başlangıç pozisyonlarında x_i ve minimum frekansta f_{min} uçarken, değişken bir dalga boyu (r) ve başlangıç ses şiddeti L_0 ile rastgele hareket ederler. Hedeflerine olan mesafeye bağlı olarak yayılan dalgalarının frekanslarını ayarlama yeteneğine sahiptirler ve sinyal yayılım oranlarını 0 ile 1 arasında değiştirirler. Her yarasanın kendine özgü frekansı, ses şiddeti ve sinyal yayılım oranı olabilir. Bu ses şiddeti, L_m^{iter}, L_0 gibi daha yüksek bir değerden L_{min} gibi en düşük yerleşik değere ulaşana kadar farklı değerlerde olabilir. Optimizasyon sürecinde, her yarasanın pozisyonu x_i ve hızı v_i, belirlenebilir ve güncellenebilir. Zaman içerisinde, belirli bir t zaman aralığında, yarasanın yeni pozisyon x_i^t ve hız v_i^t değerleri aşağıdaki denklemlerle hesaplanır [24].

$$f_i = f_{min} + (f_{max} - f_{min})a \quad (4)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^{\varphi})f_i \quad (5)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (6)$$

Yarasaların avlanma stratejileri, karmaşık bir dizi parametrenin sürekli olarak ayarlanmasını içerir. Avlarını tespit etmek için ses dalgalarının frekanslarını hedefle olan uzaklığa göre dinamik olarak değiştirirler. Bu frekans değişiklikleri, ya-

rasanın etrafındaki ortamın akustik özelliklerine uyum sağlamasını ve böylece avını daha etkili bir şekilde bulmasını sağlar. Her bir yarasa, belirli bir ses şiddeti ile başlar ve bu şiddet zamanla, optimizasyon algoritması ilerledikçe farklı seviyelere değişir. Ses şiddeti, sinyalin yayılma oranı ve frekans parametreleri, her yarasanın benzersiz algılama ve avlanma stratejisini oluşturur.

Bu parametrelerin güncellenmesi, belirli denklemler aracılığıyla gerçekleştirilir. Her t zaman diliminde, yarasanın pozisyonu ve hızı, daha önceki değerlerine ve belirli optimizasyon kurallarına göre yeniden hesaplanır. Bu dinamik güncellemeler, yarasanın hedefe doğru ve hızlı bir şekilde yakınsaması içindir. Özetle, yarasaların avlanma süreci, frekans, ses şiddeti ve sinyal yayılım oranlarının sürekli ve karmaşık bir etkileşimi ile tanımlanır ve optimize edilir.

Denklem 4'teki a, 0 ile 1 arasında rastgele seçilen bir vektörü temsil ederken, f_i her bir yarasanın frekans değerini ifade eder. Bu frekans değerleri, f_{min} ve f_{max} ile belirlenmiş olan en küçük ve en büyük frekans sınırları arasında yer alır. x^*, sürü içindeki en iyi çözüm değerini gösterir. Hesaplamalar sonucunda elde edilen bu en iyi çözüm değeri, sistemin genel performansını optimize etmek için kullanılır [24].

Bu süreçte, en iyi çözüm değeri belirlendikten sonra, rastgele gerçekleştirilen yerel arama işlemleriyle daha iyi bir çözüm bulunmaya çalışılır. Bu işlemler, çözüm alanında küçük değişiklikler yaparak mevcut en iyi değerin etrafında yeni ve potansiyel olarak daha iyi bir çözüm arayışını kapsar. Böylece, algoritmanın yerel maksimum veya minimumda takılmadan, global optimuma daha yakın bir çözüm bulma olasılığı artırılmış olur [24] [25].

Denklem 4, algoritmanın hem küresel hem de yerel arama yeteneklerini dengeler. Rastgele vektör a, algoritmanın keşif yeteneğini artırırken, f_i frekans değerlerinin dinamik ayarı, yarasaların çevresel değişimlere uyum sağlama yeteneğini yansıtır. Sonuç olarak, bu parametrelerin uyumlu bir şekilde ayarlanması, optimize edilmiş ve etkin bir çözüm bulma sürecini destekler. Yerel rastgele işlemler ile elde edilen yeni çözüm değerleri, algoritmanın adaptasyon kabiliyetini ve arama alanındaki hassasiyetini geliştirir. Bu yöntem, optimize edilen çözüm değerlerinin doğruluğunu ve etkinliğini artırmada önemli bir rol oynar [26] [27].

$$x_{\text{new}} = x_{\text{old}} + \varepsilon L^t \qquad (7)$$

Eşitlik 7'de ε, -1 ile 1 arasında rastgele dağılım gösteren bir değeri temsil ederken, L^t ise t zamanı boyunca bütün yarasaların ortalama ses şiddetini belirtmektedir. Bu değer, zamanla yarasaların ortalama ses seviyelerinin nasıl değiştiğini gösterir. İterasyon süreci ilerledikçe ve algoritma istenen hedefe yakınsadıkça, yarasaların ekolokasyon aracılığı ile oluşturdukları sesin, sinyal yayılım oranı ve şiddetinin düzenli olarak güncellenmesi gereklidir.

Yarasanın hedefi tespit etme sürecinde, başlangıçta yüksek olan ses şiddeti, avın konumunu daha hassas bir şekilde belirledikçe genellikle azalır. Bu süreçte, yarasalar ekolokasyon sinyallerini kullanarak çevresel geri bildirim alır ve bu geri bildirimlere göre ses şiddetini ve sinyal yayılım oranını ayarlarlar. Ses şiddeti (L) düştüğünde, bu genellikle yarasanın avına daha yakın olduğunu ve daha hassas ayarlamalara ihtiyaç duyduğunu gösterir. Aynı zamanda, sinyal yayılım oranı (r) artar, bu da yarasanın daha sık sinyal göndererek çevresel bilgileri daha ayrıntılı bir şekilde toplamasını sağlar [24] [25].

Bu düzenleme mekanizması, yarasanın hedefe doğru daha kesin bir yol izlemesine yardımcı olur. Başlangıçta yüksek ses şiddeti ile geniş bir alan taranırken, hedefe yaklaşıldıkça daha düşük ses şiddeti ve daha yüksek sinyal yayılım oranı ile daha detaylı bir arama gerçekleştirilir. İterasyon ilerledikçe bu parametrelerin dinamik olarak ayarlanması, algoritmanın performansını optimize eder ve avın tespit edilme olasılığını artırır.

Dolayısıyla, eşitlik 7'de belirtilen ε ve L^t değerleri, yarasaların ekolokasyon mekanizmasını ve bunun optimizasyon sürecindeki rolünü anlamak için kritik öneme sahiptir. Ses şiddeti ve sinyal yayılım oranındaki bu değişiklikler, yarasaların çevresel uyum yeteneklerini ve arama etkinliklerini artırarak, hedefin daha hızlı ve doğru bir şekilde bulunmasını sağlar [24].

$$A_i^{t+1} = \beta A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \qquad (8)$$

Denklem 8'de β, 0 ile 1 arasında bir yerleşik de-

ğeri ve γ ise pozitif bir yerleşik değeri temsil etmektedir. Bu iki parametre, algoritmanın dinamik özelliklerini ve optimizasyon sürecindeki davranışlarını tanımlamak için kullanılır. Zaman t sonsuza yaklaştıkça, yarasaların çıkardıkları sesin şiddeti L_i^t sıfıra yaklaşır ve sinyal yayılım oranı r_i^t ise başlangıçtaki belirli bir sabit değere r_i^0 yaklaşır [27] [28].

Bu bağlamda, β ve γ, yarasaların ses şiddeti ve sinyal yayılım oranının zamanla nasıl değişeceğini belirleyen önemli parametrelerdir. İterasyonlar ilerledikçe, yarasalar çevresel geri bildirimlere dayalı olarak ses şiddetini azaltır. Bu azalma, yarasanın avına yaklaştığını ve daha hassas bir arama gerçekleştirdiğini gösterir. Aynı zamanda, sinyal yayılım oranı başlangıçtaki sabit değere r_i^0 yaklaşır, bu da yarasaların daha sık ve düzenli sinyal göndererek çevresel bilgileri daha doğru bir şekilde toplamasını sağlar[29][30].

Yani, t sonsuza yaklaşırken L_i^t'nin sıfıra yaklaşması, yarasaların avlarını başarılı bir şekilde tespit etmeye yaklaştıklarını ve bu nedenle daha az ses çıkararak daha az enerji harcadıklarını gösterir. r_i^t 'nin r_i^0 'a yaklaşması ise sinyal yayılımının optimize edilerek çevresel algılama sürecinin en verimli hale getirildiğini ifade eder.

Özetle, Denklem 8'de tanımlanan , β ve γ ve parametreleri, yarasaların ses şiddeti ve sinyal yayılım oranının zamanla nasıl optimize edildiğini gösterir. Bu parametreler, algoritmanın verimli çalışmasını ve yarasaların çevresel algılama yeteneklerini en üst düzeye çıkarmasını sağlar. Zamanla, bu optimizasyon süreci yarasaların enerji verimliliğini artırır ve avlarını daha etkili bir şekilde tespit etmelerine yardımcı olur. Yarasa algoritmasının akışı Şekil.3'te özetlenmiştir [19].

Her iki algoritmada kullanılan başlangıç paremetreleri ve sabitleri Tablo 1'de gösterilmiştir.

Şekil 2. (BA) kodu

Başlangıç:

- Popülasyon boyutu, hareket sayısı, boyut, frekans aralığı, gürültü, puls oranı, gamma, doğruluk.

Fonksiyon Seçimi:

- Amaç fonksiyonunu seç ve x, y aralıklarını belirle.

Bat Popülasyonunu Başlat:

- Rastgele pozisyonlar oluştur ve fitness hesapla.

En İyi Fitness Değerini Belirle:

- En iyi fitness değerini ve indeksini belirle.

Dinamikleri Başlat:

- Gürültü, puls oranı, frekans ve hız matrislerini başlat.

Simülasyon:

- Hareket sayısı kadar döngü yap:
    - Frekansı ve hızı güncelle.
    - Pozisyonu güncelle.
    - Rastgele yürüyüş gerçekleştir.
    - Fitness değerini hesapla.
    - En iyi pozisyon ve fitness değerini güncelle.
    - Hedef doğruluk sağlanmışsa döngüyü sonlandır.

Sonuçları Yazdır:

- Simülasyon süresi ve en iyi çözümü yazdır.

3. TEST FONKSİYONLARI VE PERFORMANS ANALİZİ

Parçacık sürü optimizasyonu ve Yarasa algoritmasının performanslarının değerlendirilmesi için altı farklı test fonksiyonu seçilmiştir. Bu fonksiyonlar, farklı özelliklere ve karmaşıklıklara sahiptir ve algoritmaların genel uygulanabilirlik ve etkinlik seviyelerini test etmek için ideal bir zemin sunar. Şekil 3' te görülen Beale, Ackey, CrossTray, Levi, Bukin ve GoldsteinPrice gibi bu fonksiyonlar, genellikle optimizasyon algoritmalarının performansını değerlendirmek için standart olarak kabul edilir. Bu çalışmada kullanılan test fonksiyonlarının arama uzayı alt ve üst sınırları, küreesel minimum ve formülleri Şekil 3'te gösterilmiştir.

PSO (Parçacık Sürü Optimizasyonu) ve BA (Yarasa Algoritması) gibi farklı teknikler, bu test fonksiyonları üzerinde karşılaştırmalı olarak değerlendirilmiştir. Her bir test fonksiyonu, belirli bir optimizasyon problemine odaklanır ve algoritmaların bu problemleri ne kadar etkili bir şekilde çözebildiğini ölçmek için kullanılır. Bu karşılaştırmalar, her algoritmanın güçlü ve zayıf yönlerini belirlemeye ve hangi tür problemlerde daha iyi performans gösterdiklerini anlamaya yardımcı olur. Bu bağlamda, test fonksiyonları, araştırmacıların algoritmalar arasında kapsamlı bir performans karşılaştırması yapmasını sağlar ve optimizasyon tekniklerinin genel etkinliğini değerlendirmede önemli bir araç olarak kullanılır.

Beale, Ackey, CrossTray, Levi, Bukin ve GoldsteinPrice gibi test fonksiyonları, MATLAB ortamında uygulanabilirliklerini değerlendirmek amacıyla PSO ve BA algoritmalarıyla ayrı ayrı test edilmiştir. Bu test fonksiyonları, çeşitli matematiksel özelliklere sahip olup, optimizasyon algoritmalarının performansını farklı zorluk seviyelerinde ölçmek için kullanılmıştır. Her bir test fonksiyonu, farklı bir optimizasyon problemi sunar ve algoritmaların bu problemleri ne kadar etkili bir şekilde çözebildiğini belirlemek için kullanılır.

MATLAB üzerinde yapılan bu uygulamalar, her bir test fonksiyonunun özelliklerini ve algoritmaların nasıl tepki verdiğini ayrıntılı bir şekilde inceleme olanağı sunar. Bu incelemeler, her bir algoritmanın test fonksiyonlarının farklı özelliklerine ve karmaşıklıklarına nasıl adapte olduğunu ve hangi durumlarda daha iyi performans gösterdiğini belirlemeye yardımcı olur.

Sonuçlar, her bir algoritmanın güçlü ve zayıf yönlerini anlamamıza ve gelecekteki optimizasyon problemlerine daha iyi çözümler geliştirmemize yardımcı olacak değerli bilgiler sunar. Bu çalışmalar, algoritmaların genel uygulanabilirliklerini ve performanslarını anlamak için önemli bir adımdır ve çeşitli optimizasyon tekniklerinin karşılaştırmalı analizini sağlar.

3.1. Beale Test Fonksiyonu

Beale test fonksiyonu, matematiksel optimizasyon problemlerinin çözümünde sıklıkla kullanılan bir test problemidir. Bu fonksiyon, iki değişken içeren ve global minimum noktasını bulmak için kullanılan bir test aracı olarak tasarlanmıştır. Beale test fonksiyonu, Denklem 9'daki gibi ifade edilir [31][32].

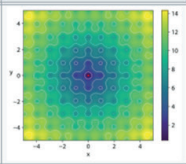
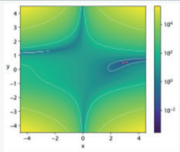
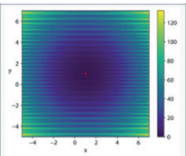
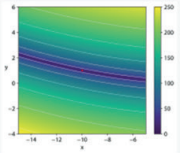
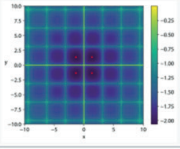
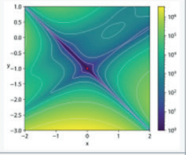
$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \quad (9)$$

Tablo 1. Algoritmaların parametreleri ve sabitleri

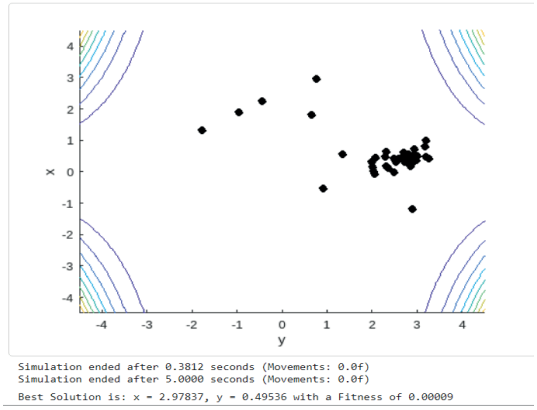
Parametre	BAT Algoritması	PSO Algoritması	Parametre	BAT Algoritması	PSO Algoritması
Population_Size	50	50	Gamma	0.5	-
Num_Movements	50	50	Accuracy	3	3
Dimension(s)	2	2	Inertia	-	0.40
Freq_Min	0	-	Accel_Personal	-	0.20
Freq_Max	1	-	Accel_Social	-	0.40
Loudness	0.5	-	Min_Movements	-	20
PulseRate	0.5	-	Gamma	0.5	-

Bu fonksiyon, birden fazla lokal minimuma sahip karmaşık bir yapıya sahiptir, bu nedenle optimizasyon algoritmalarının doğruluğunu ve performansını test etmek için sıklıkla kullanılır. Beale test fonksiyonu, hem teorik hem de pratik açıdan matematiksel optimizasyon problemlerini çözmek için geliştirilen algoritmaların etkinliğini değerlendirmek için bir referans olarak kullanılmaktadır. Bu test fonksiyonunun özellikleri ve çözümü, optimizasyon alanında araştırma ve geliştirme çalışmalarının temelini oluşturur. Şekil 4'te BA modelinin Beale testi sonucu, Şekil 5' te ise PSO modelinin Beale testi sonucu gösterilmiştir.

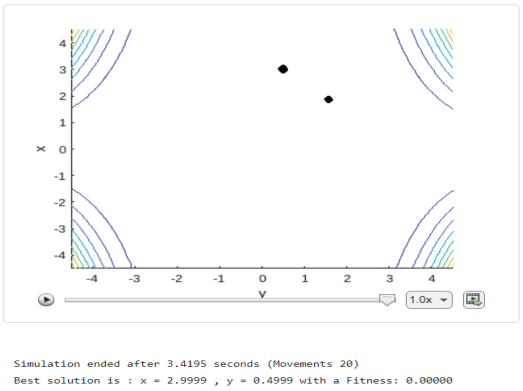
Şekil 3. Test fonksiyonları

İsim	Komplo	Formül	Küresel minimum	arama uzayı
Ackley function		$f(x,y) = -20\exp\left[-0.2\sqrt{0.5\left(x^2+y^2\right)}\right] - \exp[0.5\left(\cos 2\pi x + \cos 2\pi y\right)] + e + 20$	$f(0,0) = 0$	$-5 \le x, y \le 5$
Beale function		$f(x,y) = (1.5 - x + xy)^2 + \left(2.25 - x + xy^2\right)^2 + \left(2.625 - x + xy^3\right)^2$	$f(3,0.5) = 0$	$-4.5 \le x, y \le 4.5$
Lévi function N.13		$f(x,y) = \sin^2 3\pi x + (x-1)^2\left(1 + \sin^2 3\pi y\right) + (y-1)^2\left(1 + \sin^2 2\pi y\right)$	$f(1,1) = 0$	$-10 \le x, y \le 10$
Bukin function N.6		$f(x,y) = 100\sqrt{\left\lvert y - 0.01x^2\right\rvert} + 0.01\left\lvert x + 10\right\rvert.$	$f(-10,1) = 0$	$-15 \le x \le -5,$ $-3 \le y \le 3$
Cross-in-tray function		$f(x,y) = -0.0001\left[\left\lvert \sin x \sin y \exp\left(\left\lvert 100 - \frac{\sqrt{x^2+y^2}}{\pi}\right\rvert\right)\right\rvert + 1\right]^{0.1}$	$\text{Min} = \begin{cases} f(1.34941, -1.34941) & = -2.06261 \\ f(1.34941, 1.34941) & = -2.06261 \\ f(-1.34941, 1.34941) & = -2.06261 \\ f(-1.34941, -1.34941) & = -2.06261 \end{cases}$	$-10 \le x, y \le 10$
Goldstein–Price function		$f(x,y) = \left[1 + (x+y+1)^2\left(19 - 14x + 3x^2 - 14y + 6xy + 3y^2\right)\right]\left[30 + (2x-3y)^2\left(18 - 32x + 12x^2 + 48y - 36xy + 27y^2\right)\right]$	$f(0,-1) = 3$	$-2 \le x, y \le 2$

Şekil 4. BA modelinin Beale testi sonucu



Şekil 5. PSO modelinin Beale testi sonucu



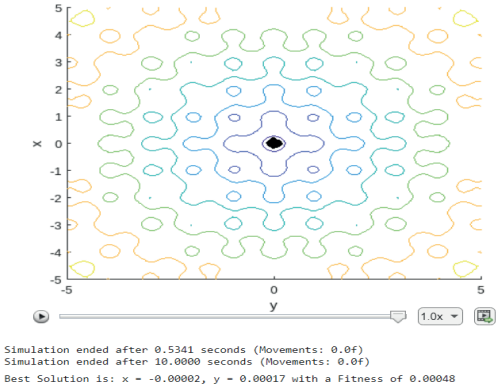
3.2. Ackley Test Fonksiyonu

Ackley test fonksiyonu, matematiksel optimizasyon problemlerinin çözümünde yaygın olarak kullanılan bir test problemidir. Bu fonksiyon, hem çok boyutlu hem de düzlemsel alanlarda optimize edilebilirlik analizlerinin yapılmasında yaygın olarak kullanılmaktadır. Ackley test fonksiyonu denklem 10'da yer almaktadır [33].

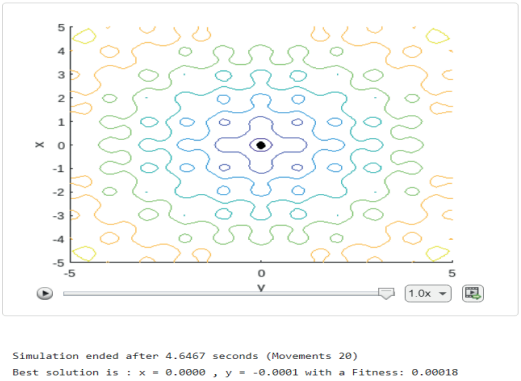
$$f(x,y) = -20\exp\left[-0.2\sqrt{0.5(x^2+y^2)}\right] - \exp[0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20 \quad (10)$$

Bu fonksiyon, birçok lokal minimuma sahip olmasıyla bilinir ve bu nedenle optimizasyon algoritmalarının performansını test etmek için ideal bir seçimdir. Ackley test fonksiyonu, farklı boyutlardaki ve şekillerdeki fonksiyonların optimize edilmesi durumunda algoritmaların karşılaşabileceği zorlukları belirlemek için kullanılır. Bu nedenle, Ackley test fonksiyonu, optimizasyon algoritmalarının doğruluğunu, hızını ve genel performansını değerlendirmek için güvenilir bir referans olarak kabul edilir. Şekil 6'da BA modelinin Ackley testi sonucu, Şekil 7' de ise PSO modelinin Ackley testi sonucu gösterilmiştir.

Şekil 6. BA modelinin Ackley testi sonucu



Şekil 7. PSO modelinin Ackley testi sonucu



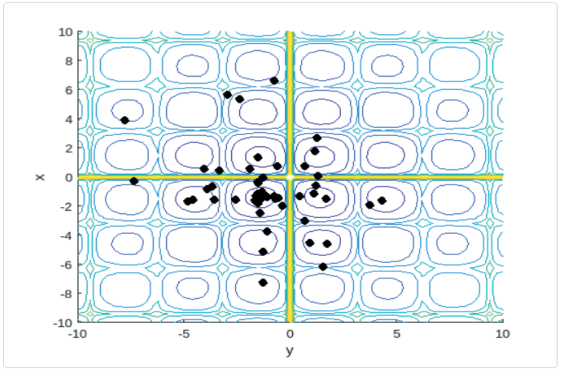
3.3. Cross in Tray Test Fonksiyonu

Cross in Tray test fonksiyonu, matematiksel optimizasyon problemlerinin değerlendirilmesi ve karşılaştırılması için kullanılan bir test problemidir. Bu fonksiyon, iki boyutlu uzayda optimize edilecek bir fonksiyon olarak tanımlanır. Cross in Tray test fonksiyonu, denklem 11'de yer almaktadır [34].

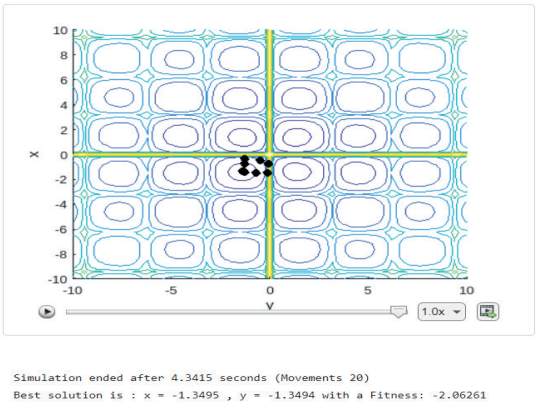
$$f(x,y) = -0.0001 \cdot \left(\left|\sin(x)\cdot\sin(y)\cdot\exp\left(\left|100-\frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right|+1\right)^{0.1} \quad (11)$$

Şekil 8'de BA modelinin Cross in Tray testi sonucu, Şekil 9' da ise PSO modelinin Cross in Tray testi sonucu gösterilmiştir.

Şekil 8. BA modelinin Cross in Tray testi sonucu



Şekil 9. PSO modelinin Cross in Tray testi sonucu

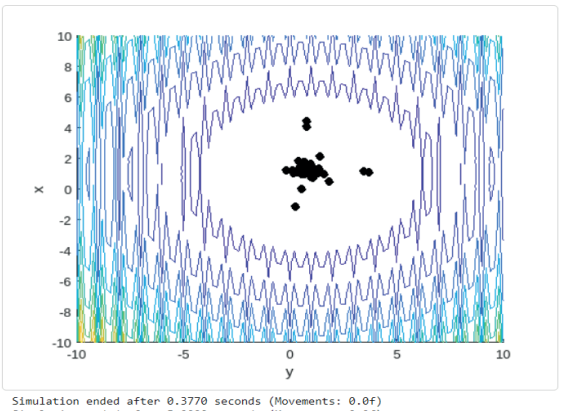


3.4. Levi Test Fonksiyonu

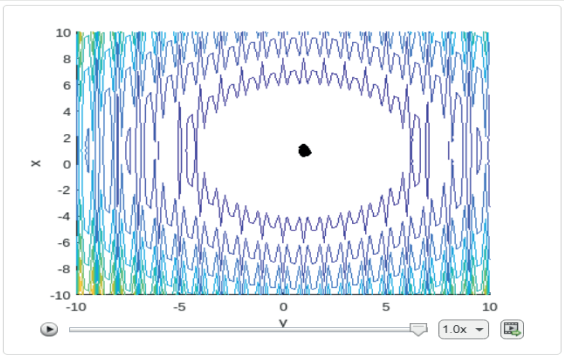
Levi test fonksiyonu, matematiksel bir ifade olarak Denklem 12'de yer almaktadır [35].

Şekil 10'da BA modelinin Levi testi sonucu, Şekil 11' de ise PSO modelinin Levi testi sonucu gösterilmiştir.

Şekil 10. BA modelinin Levi testi sonucu



Şekil 11. PSO modelinin Levi testi sonucu



3.5. Bukin Test Fonksiyonu

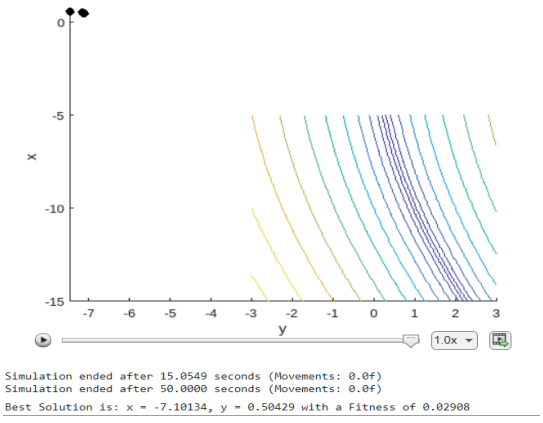
$$f(x,y) = \sin^2(3\pi x) + (x-1)^2 \cdot (1+\sin^2(3\pi y)) + (y-1)^2 \cdot (1+\sin^2(2\pi y)) \quad (12)$$

Bu fonksiyon, iki değişkenli bir fonksiyon olarak tanımlanır ve çeşitli optimizasyon algoritmalarının performansını test etmek için kullanılır. Matematiksel bir ifade olarak, Bukin test fonksiyonu Denklem 13'te gösterilmiştir [36].

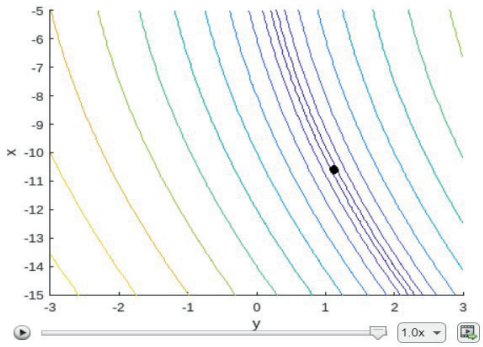
$$(x,y) = 100 \cdot \sqrt{|y - 0.01x^2|} + 0.01 \cdot |x+10| \quad (13)$$

Şekil 12'de BA modelinin Bukin testi sonucu, Şekil 13' te ise PSO modelinin Bukin testi sonucu gösterilmiştir.

Şekil 12. BA modelinin Bukin testi sonucu



Şekil 13. PSO modelinin Bukin testi sonucu

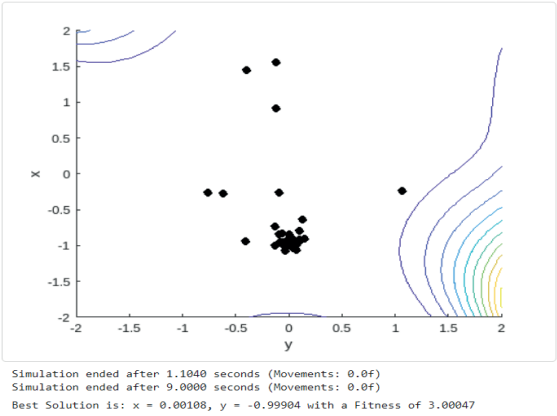


3.6. Goldstein Price Test Fonksiyonu

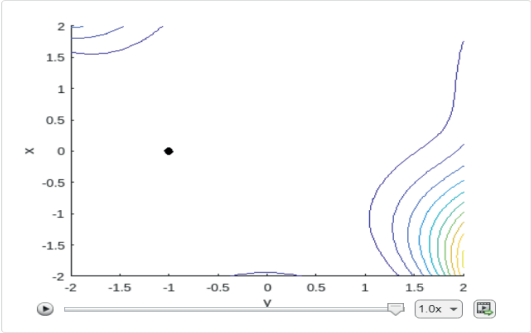
Denklem 14'teki Goldstein-Price test fonksiyonu, matematiksel optimizasyon algoritmalarının performansını değerlendirmek ve karşılaştırmak için kullanılan bir test problemidir. Bu fonksiyon, iki değişkenli bir fonksiyon olarak tanımlanır ve genellikle global minimumunu bulmak için optimizasyon algoritmalarının etkinliğini test etmek için tercih edilir [37].

Şekil 14'te BA modelinin Goldstein Price testi sonucu, Şekil 15' te ise PSO modelinin Goldstein Price testi sonucu gösterilmiştir.

Şekil 14. BA modelinin Goldstein Price testi sonucu



Şekil 15. PSO modelinin Goldstein Price testi sonucu



Tablo 2, Yarasa Algoritması (YA) ve Parçacık Sürüsü Optimizasyonu (PSO) algoritmalarının altı farklı test fonksiyonu üzerindeki performanslarını karşılaştırmaktadır. Tabloda kullanılan "Uygunluk Değeri", bir fonksiyonun minimum değerine ne kadar yakın olduğunu gösteren bir ölçüttür. "Optimum Çözüm" ise fonksiyonun minimum değerini veren x ve y değerlerini temsil etmektedir. Her bir fonksiyon için elde edilen sonuçlar ve algoritmaların performans değerlendirildiğinde Bukin fonksiyonu hariç YA ve PSO benzer performans göstermiştir denilebilse de çözüm hassasiyeti bakımından PSO daha iyidir. Bukin test fonksiyonu da ise YA, PSO'ya göre daha iyi performans göstermiştir.

Tablo 2. Test fonksiyonu sonuçları

Test Fonksiyonu	Algoritma	Fitness Değeri	Optimum Çözüm	
			x	y
Beale	BA	0.00009	2.9783	0.4933
	PSO	0.00008	2.9999	0.4999
Ackley	BA	0.00048	-0.00002	0.00017
	PSO	0.00018	0.0000	-0.0001
Cross in Tray	BA	-2.06259	-1.3407	-1.3394
	PSO	-2.06261	-1.3495	1.3494
Levi	BA	0.00013	1.0009	1.0069
	PSO	0.00000	1.0001	0.9985
Bukin	BA	0.02908	-7.1013	0.5042
	PSO	0.00986	-10.5980	1.1232
Goldstein Price	BA	3.00047	0.0010	-0.9999
	PSO	3.00000	0.0000	-1.0000

4. SONUÇ VE DEĞERLENDİRME

Bu çalışma, Parçacık Sürü Optimizasyonu (PSO) ile Yarasa Algoritması (BA) gibi doğa esinli optimizasyon algoritmalarının karşılaştırmalı performans analizini gerçekleştirmiştir. Analizlerde,. Beale, Ackley, Cross in Tray, Levi, Bukin ve Goldstein Price test fonksiyonları, algoritmaların performansını değerlendirmek için seçilmiştir.

Tablo 1'deki sonuçlar göz önüne alındığında, PSO'nun Beale, Ackley, Levi ve Goldstein Price gibi test fonksiyonları üzerinde genellikle daha düşük uygunluk değerleri elde ettiği gözlemlenmiştir. Özellikle Beale ve Levi testlerinde, PSO'nun BA'ya kıyasla belirgin şekilde daha iyi bir optimizasyon performansı sergilediği tespit edilmiştir. Bu sonuçlar, PSO'nun bu tür test fonksiyonları için daha etkili bir çözüm sağladığını ve genellikle daha hızlı yakınsama oranlarına ulaştığını öne sürmektedir.

Ancak, Cross in Tray ve Bukin gibi bazı test fonksiyonları için BA algoritmasının daha iyi bir performans sergilediği gözlemlenmiştir. Özellikle, Bukin test fonksiyonunda, BA'nın elde ettiği uygunluk değerinin PSO'dan önemli ölçüde daha düşük olduğu belirlenmiştir. Bu durum, BA'nın bazı özel problem alanları için daha uygun bir optimizasyon yaklaşımı sunduğunu ve PSO'nun bu tür test fonksiyonları için her zaman en iyi seçenek olmadığını ortaya koymaktadır.

Bu sonuçlar, test fonksiyonlarının karmaşıklığına ve özelliklerine bağlı olarak analizlerin değişiklik gösterebileceğini vurgulamaktadır. Bir algoritmanın performansı, değerlendirildiği problem türüne özgü olarak farklılık gösterebileceği unutulmamalıdır. Aynı zamanda bir algoritmanın tüm problem türlerinde en iyi sonucu vereceği garanti edilemez ve problem karmaşıklığı göz önünde bulundurularak farklı algoritmaların performansları dikkatle analiz edilmelidir.

Bu karşılaştırmalı analizlerin ışığında, belirli bir optimizasyon probleminde hangi algoritmanın kullanılacağı kararında problem yapısı ve optimizasyon hedefleri dikkate alınmalıdır. PSO ve BA gibi doğa esinli algoritmalar, farklı problem tipleri ve optimizasyon gereksinimleri için farklı avantajlar sunabilir. Örneğin, PSO daha hızlı yakınsama sağlayabilirken, BA bazı özel problem tiplerinde daha istikrarlı sonuçlar verebilir.

Bu çalışma, sadece belirli test fonksiyonları üzerinde gerçekleştirilmiş olup, gerçek dünya uygulamalarında algoritmaların performansını değerlendirmek, kapsamlı testler ve karşılaştırmalar gerekmektedir. Gelecekteki araştırmalarda, farklı problem ve optimizasyon gereksinimleri için daha kapsamlı bir analiz yapılması önerilmektedir. Ayrıca, algoritmaların parametrelerinin ve yapılandırmalarının performans üzerindeki etkilerinin daha ayrıntılı bir şekilde incelenmesi, algoritmaların daha iyi anlaşılmasını ve daha etkili bir şekilde kullanılmasını sağlayabilir.

KAYNAKÇA

[1] Cong, P., Xu, G., Wei, T., & Li, K. (2020). A survey of profit optimization techniques for cloud providers. ACM Computing Surveys (CSUR), 53(2), 1-35.

[2] Shah, A. S., Nasir, H., Fayaz, M., Lajis, A., & Shah, A. (2019). A review on energy consumption optimization techniques in IoT based smart building environments. Information, 10(3), 108.

[3] Çunkaş M., Elektrik Makinalarının Genetik Algoritmayla Optimizasyonu, Doktora Tezi, Selçuk Ünv. Fen Bilimleri Enst. 2004, Konya

[4] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

[5] Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341-359.

[6] Yang, X. S. (2010). A New Metaheuristic Bat-Inspired Algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, vol 284. Springer, Berlin, Heidelberg.

[7] Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942-1948.

[8] Karaboğa D., Yapay zeka optimizasyon algoritmaları, Atlas Yayınları, 2004.

[9] Rahimpour E., Rashtchi V., Pesaran M., "Parame-

ter identification of deep-bar induction motors using genetic algorithm", Electrical Engineering, 89(7): 547-552, 2007

[10] Özçelik, F. (2018). Basit düz ve U-tipi montaj hattı dengeleme problemleri için diferansiyel evrim algoritması. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 24(1), 130-140.

[11] Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft computing, 22(2), 387-408.

[12] Yang, X. S., & He, X. (2013). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149.

[13] Erdoğmuş, P. (2016). Doğadan esinlenen optimizasyon algoritmaları ve optimizasyon algoritmalarının optimizasyonu. Düzce üniversitesi bilim ve teknoloji dergisi, 4(1), 293-304.

[14] Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. simulation, 76(2), 60-68.

[15] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). ieee.

[16] Shi, Y., & Eberhart, R. C. (1999, July). Empirical study of particle swarm optimization. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 3, pp. 1945-1950). IEEE.

[17] Özsağlam, M. Y., & Çunkaş, M. (2008). Optimizasyon problemlerinin çözümü için parçaçık sürü optimizasyonu algoritması. Politeknik Dergisi, 11(4), 299-305.

[18] Robinson, J. and Rahmat-Samii, Y., 2004. Particle swarm optimization in electromagnetics, IEEE Transactions on Antennas and Propagation, 52(2), 397-407.

[19] Akbulut, İ. (2009). Parçacık sürü optimizasyonu ile anten tasarımı., Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Bilişim Enstitüsü, İstanbul.

[20] Yang, X. S., & He, X. (2013). Bat algorithm: literature review and applications. International Journal of Bio-inspired computation, 5(3), 141-149.

[21] Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., & Istanda, V. (2012). Bat algorithm inspired algorithm for solving numerical optimization problems. Applied mechanics and materials, 148, 134-137.

[22] Yang, X. S., & Hossein Gandomi, A. (2012). Bat algorithm: a novel approach for global engineering optimization. Engineering computations, 29(5), 464-483.

[23] Mirjalili, S., Mirjalili, S. M., & Yang, X. S. (2014). Binary bat algorithm. Neural Computing and Applications, 25, 663-681.

[24] Gandomi, A. H., Yang, X. S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. Neural Computing and Applications, 22, 1239-1255.

[25] Yıldızdan, G. (2021). Büyük ölçekli sürekli optimizasyon problemleri için yarasa algoritması tabanlı hibrit yöntemlerin geliştirilmesi.

[26] Chawla M. ve Duhan M., 2015, Bat algorithm: a survey of the state-of-the-art, Applied Artificial Intelligence, 29 (6), 617-34.

[27] Yang X.-S., 2010a, A new metaheuristic bat-inspired algorithm, Nature inspired cooperative strategies for optimization (NICSO 2010),284, Springer, Berlin, Heidelberg, 65-74.

[28] Ekinci S. (2015). Power system stabilizer design for multi-machine power system using bat search algorithm, Sigma Mühendislik ve Fen Bilimleri Dergisi, 33(4): 627-637, 2015.

[29] Doğru, A. S., Temel, B., & Eren, T. (2019). Kablosuz Sensör Ağlarında Konum Belirlemede Parçacık Sürü Optimizasyonu ve Yarasa Algoritması Yöntemlerinin Karşılaştırılması. International Journal of Engineering Research and Development, 11(3), 793-801.

[30] Doğru, A. S., & Tolga, E. R. E. N. (2020). Kablosuz Sensör Ağlarında Konum Belirlemede Parçacık Sürü Optimizasyonu, Yarasa Algoritması, Diferansiyel Gelişim Algoritması ve Ateşböceği Algoritması Yöntemlerinin Karşılaştırılması. International Journal of Engineering Research and Development, 12(3), 52-64.

[31] Beale, Evelyn ML. "On minimizing a convex function subject to linear inequalities." Journal of the Royal Statistical Society Series B: Statistical Methodology 17.2 (1955): 173-184.

[32] Beale, D., & Feinstein, A. (1976). Structure and function of the constant regions of immunoglobulins. Quarterly reviews of biophysics, 9(2), 135-180.

[33] Porter, B., & Xue, F. (2001, May). Niche evolution

strategy for global optimization. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546) (Vol. 2, pp. 1086-1092). IEEE.

[34] Vishwakarma, V., Haq, S. A., Schleicher, E., Schubert, M., & Hampel, U. (2021). Experimental analysis of the hydrodynamic performance of an industrial-scale cross-flow sieve tray. Chemical Engineering Research and Design, 174, 294-306.

[35] Andrews, S. C., Harrison, P. M., Yewdall, S. J., Arosio, P., Levi, S., Bottke, W., ... & Lobreaux, S. (1992). Structure, function, and evolution of ferritins. Journal of inorganic biochemistry, 47(1), 161-174.

[36] Mishra, S. K. (2006). Some new test functions for global optimization and performance of repulsive particle swarm method. Available at SSRN 926132.

[37] Fenwick, M., & McCrimmon, A. W. (2015). Test review: comprehensive executive function inventory by JA Naglieri and S. Goldstein.