Journal of Engineering Faculty, 3(2): 109-116, 2025

Journal of Engineering Faculty

| cumfad.cumhuriyet.edu.tr | Founded: 2023 Available online, ISSN:3023-7203 Publisher: Sivas Cumhuriyet Üniversitesi

Fintess-Distance Balance based Giant Armadillo Optimization Algorithm

Hasan Uzel[1,a,*], Yunus Hınıslıoğlu[2,b], Adem Dalcalı[3,c], Uğur Güvenç[4,d]

[1]*Akdağmadeni Meslek Yüksekokulu, Elektrik ve Enerji Bölümü, Yozgat Bozok Üniversitesi, Yozgat 66900, Türkiye*
[2]*Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Düzce Üniversitesi, Düzce 81620, Türkiye*
[3]*Mühendislik ve Doğa Bilimleri Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Bandırma Onyedi Eylül Üniversitesi, Balıkesir 10200, Türkiye*
[4] *Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Düzce Üniversitesi, Düzce 81620, Türkiye*
**Corresponding author*

Research Article

History

Received: 25/12/2024
Accepted: 02/01/2025

ABSTRACT

In this study, the Giant Armadillo Optimization (GAO) algorithm has been developed using the Fitness-Distance Balance (FDB) method, which is an effective method to eliminate the imbalance between exploration and exploitation in the algorithms. The proposed Fitness-Distance Balance based Giant Armadillo Optimization (FDBGAO) algorithm aims to provide a more effective balance between exploration and exploitation compared to the main GAO algorithm. In the simulation study, in order to examine the effectiveness of the main GAO and FDBGAO algorithms, analyzes were performed for 23 standard benchmark functions and the obtained results were compared in terms of mean and standard deviation values. The obtained results show that the FDBGAO algorithm provides more effective and successful results compared to the basic GAO algorithm in solving single-mode, multi-mode and multi-mode low-dimensional benchmark functions.

Keywords: Fitness-Distance Balance; Giant Armadillo Optimization Algorithm; Meta-heuristic Algorithms.

Uygunluk-Mesafe Dengesi Tabanlı Dev Armadillo Optimizasyon Algoritması

Süreç

Geliş: 25/12/2024
Kabul: 02/01/2025

ÖZ

Bu çalışmada, algoritmalardaki keşif ve sömürü arasındaki dengesizliği ortadan kaldırmak adına etkili bir yöntem olan Uygunluk-Mesafe Dengesi (FDB) yöntemi ile Dev Armadillo Optimizasyon (GAO) algoritması geliştirilmesi sağlanmıştır. Önerilen Uygunluk-Mesafe Dengesi tabanlı Dev Armadillo Optimizasyon (FDBGAO) algoritması ile temel GAO algoritmasına kıyasla keşif ve sömürü arasında daha etkili bir denge sağlanması amaçlanmıştır. Benzetim çalışmasında, temel GAO ve FDBGAO algoritmalarının etkinliğini incelemek amacıyla 23 standart kıyaslama fonksiyonu için analizler gerçekleştirilmiş ve elde edilen sonuçlar ortalama ve standart sapma değerleri açısından karşılaştırılmıştır. Elde edilen sonuçlar, FDBGAO algoritmasının tek-modlu, çok-modlu ve çok-modlu düşük boyutlu kıyaslama fonksiyonlarının çözümünde temel GAO algoritmasına kıyasla daha etkili ve başarılı sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Uygunluk-Mesafe Dengesi; Dev Armadillo Optimizasyon Algoritması; Meta-sezgisel Algoritmalar.

[a] *hasanuzel@bozok.edu.tr* *0000-0002-8238-2588*
[b] *yunushinislioglu@duzce.edu.tr* *0000-0002-8382-3795*
[c] *adalcali@bandirma.edu.tr* *0000-0002-9940-0471*
[d] *ugurguvenc@duzce.edu.tr* *0000-0002-5193-7990*

How to Cite: Uzel H, Hınıslıoğlu Y, Dalcalı A, Güvenç U (2025) Fintess-Distance Balance based Giant Armadillo Optimization Algorithm, Journal of Engineering Faculty, ??(?): ???-???

Giriş

Optimizasyon, bilim dünyasındaki matematiksel problemlerde ve gerçek dünya uygulamalarında yer alan çözüm uzayı içerisindeki birden fazla aday arasından en uygun çözüm adayının belirlenmesi işlemidir [1]. Optimizasyon işleminin gerçekleştirilebilmesi için, değişkenlerin tanımlanması, problem için kısıtlamaların belirlenmesi ve en önemlisi de amaç fonksiyonunun matematiksel olarak ifade edilmesi gerekmektedir. Genel bir ifadeyle, çeşitli kısıtlamalar çerçevesinde belirlenen amaç fonksiyonunun minimize veya maksimize edilmesi optimizasyon işlemi olarak tanımlanabilir. Teknolojik gelişmelerle birlikte, karmaşıklığı artan optimizasyon problemlerinin çözümünde kullanılan matematiksel yöntemler yetersiz kalmış ve günümüzde bu yöntemler yerini sezgisel ve meta-sezgisel yöntemlere bırakmıştır [2], [3].

Meta-sezgisel algoritmalar, geçmişten günümüze kadar olan süreçte araştırmacılar tarafından karmaşık optimizasyon problemlerinin çözümü için yaygın olarak kullanılmıştır [4], [5]. Geçmiş yıllarda, meta-sezgisel algoritmalar olarak Parçacık Sürü Optimizasyonu (PSO) [6], Genetik Algoritma (GA) [7] ve Diferansiyel Evrim (DE) [8] algoritmaları en yaygın kullanılan algoritmalar arasında yer almıştır. Sonrasında, doğadan ilham alınan farklı algoritmalar, araştırmacılar tarafından geliştirilmiş ve popülerlik kazanmıştır. Dorigo ve arkadaşları tarafından 2006 yılında literatüre kazandırılan Karınca Kolonisi Optimizasyon (ACO) algoritması, doğada yer alan bazı karınca türlerinin yiyecek arama davranışlarından ilham alınarak oluşturulmuştur [9]. 2007 yılında Karaboğa ve Baştürk tarafından literatüre sunulan çalışmada, bal arısı sürüsünden ilham alınarak geliştirilen Yapay Arı Kolonisi (ABC) algoritması önerilmiştir [10]. Mirjalili ve arkadaşları tarafından 2014 yılında literatüre kazandırılan Gri-Kurt Optimizasyon (GWO) Algoritması, doğadaki gri kurtların avlanma mekanizması ve liderlik hiyerarşisi taklit edilerek tasarlanmıştır [11]. 2014 yılında Cheng ve Prayogo tarafından yapılan başka bir çalışmada, organizmaların hayatta kalmak ve çoğalmak için kurmuş oldukları simbiyotik etkileşimlerinden ilham alınarak modellenen Simbiyotik Organizmalar Arama (SOS) algoritması önerilmiştir [12]. Son yıllarda da araştırmacılar tarafından Hipopotamus Optimizasyon (HO) algoritması [13], Arşimet Optimizasyon Algoritması (AOA) [14], Pelikan Optimizasyon Algoritması (POA) [15] ve Koati Optimizasyon Algoritması (COA) [16] gibi yeni meta-sezgisel algoritmalar önerilmeye devam etmektedir.

Her geçen gün yeni meta-sezgisel algoritmalar literatüre kazandırılırken, bunun yanı sıra karmaşık ve yüksek boyutlu optimizasyon problemlerinin çözümünde yetersiz kalan mevcut meta-sezgisel algoritmaların geliştirilmesi de önemli bir araştırma alanıdır. Meta-sezgisel algoritmaların etkili bir sonuç vermesi açısından keşif ve sömürü yetenekleri arasında denge sağlanmalıdır. Ancak, bazı algoritmalar keşif yeteneği açısından sömürü yeteneğine kıyasla daha iyi performans gösterirken bazılarında ise keşif yeteneği, sömürü yeteneğine kıyasla yetersiz kalmaktadır. Bu sebeple de keşif ve sömürü arasında bir dengesizlik oluşmaktadır. Literatürde yer alan Uygunluk-Mesafe Dengesi (FDB) yöntemi, bahsedilen dezavantajın üstesinden gelebilmek amacıyla geliştirilmiş bir yöntemdir [17]. FDB, geleneksel optimizasyon yöntemlerinde yer alan çözüm adaylarının uygunluk değerinin yanı sıra adaylar arasında oluşan mesafe değerinin de dikkate alındığı etkili bir seçim ve sıralama yöntemidir. Sosyal Ağ Arama (SNS) algoritmasının FDB yöntemi ile geliştirilmesinden elde edilen FDBSNS algoritması [18], Güve Sürü Algoritmasının (MSA) FDB yöntemi kullanılarak geliştirilmesinden elde edilen FDB-MSA algoritması [19], Stokastik Fraktal Arama (SFS) algoritmasının FDB yöntemi ile geliştirilmesinden elde edilen FDBSFS algoritması [20], Yapay Ekosistem Optimizasyon (AEO) algoritmasının FDB yöntemi ile geliştirilmesiyle elde edilen FDBAEO algoritması [21], Kepler Optimizasyon Algoritmasının (KOA) FDB yöntemi kullanılarak geliştirilmesi ile ortaya çıkan FDBKOA algoritması [22] ve FDB yönteminin seçim arayüzü olarak kullanılması ile elde edilen hiper-sezgisel yapıdaki HH-FDB-SHADE algoritması [23] gibi literatürde yer alan meta-sezgisel ve hiper-sezgisel algoritmaların performanslarını geliştirmek adına FDB yöntemi kullanılmıştır.

Son yıllarda Alsayyed ve arkadaşları tarafından literatüre kazandırılmış olan Dev Armadillo Optimizasyon (GAO) algoritması, doğada yer alan dev armadillo davranışlarından ilham alınarak geliştirilmiştir. GAO algoritması, keşif yeteneği olarak armadilloların termit yuvalarına doğru olan hareketlerini baz alırken, sömürü yeteneği olarak da termit yuvalarını açmak ve avlanmak için yapmış oldukları kazma işlemini dikkate almaktadır [24]. Bu çalışmada, GAO algoritması ile elde edilen sonuçlardan yola çıkarak bazı problemler için yerel noktalara takılma durumunun olduğu gözlemlenmiş ve bu dezavantajı ortadan kaldırmak adına GAO algoritmasının keşif aşamasına FDB yöntemi entegre edilerek algoritmanın geliştirilmesi amaçlanmıştır. FDB yöntemi entegre edilerek geliştirilen FDBGAO algoritmasının GAO algoritmasına kıyasla çok daha iyi sonuçlar verdiği gözlemlenmiştir.

Buna göre, bu çalışmanın literatüre katkıları aşağıda verildiği gibidir:

- FDB tabanlı GAO (FDBGAO) adı verilen yeni bir meta-sezgisel arama algoritması geliştirildi. En sağlam ve yeni seçim yöntemlerinden biri olan FDB, GAO'nın keşif yeteneği geliştirmek için kullanıldı. Böylece keşif ve sömürü arasında başarılı bir denge sağlandı.
- FDBGAO, farklı problem tipleri ve boyutları için CEC2017 kullanılarak yapılan analizlerde rakibi GAO'ya kıyasla daha üstün bir performans gösterdi.

Dev Armadillo Optimizasyon Algoritması

Algoritma Başlangıcı

Dev Armadillo Optimizasyon (GAO) algoritması, dev armadillonun termit yuvalarına saldırıp onları kazma stratejisinden esinlenerek geliştirilmiş bir popülasyon tabanlı meta-sezgisel optimizasyon yöntemidir. Algoritma, iteratif bir süreçle uygun çözümler üretir ve her bir dev armadillo, bir popülasyon üyesi olarak problem çözme uzayındaki bir çözümü temsil eder. Vahşi doğada her armadillonun pozisyonu, algoritmadaki üyelerin çözüm uzayındaki

konumuna karşılık gelir. GAO'nun tasarımında, armadilloların termit yuvalarını keşfetme ve kazma stratejilerinin matematiksel modellemesi kullanılarak, optimizasyon problemleri için etkili çözümler elde edilir. Belirli sayıdaki dev armadillolar birlikte algoritmanın popülasyonunu oluşturur ve bu popülasyon matematiksel olarak bir matris kullanılarak Denklem (1)'deki gibi modellenebilir [24].

$$P = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{Nxm} = \begin{bmatrix} x_{11} & \cdots & x_{1d} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i1} & \cdots & x_{id} & \cdots & x_{im} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nd} & \cdots & x_{Nm} \end{bmatrix}_{Nxm} \quad (1)$$

Dev armadilloların problem çözme uzayındaki birincil konumu, algoritma yürütmesinin başlangıcında Denklem (2) kullanılarak rastgele başlatılır.

$$x_{id} = lb_d + r.(ub_d - lb_d) \quad (2)$$

Burada, P GAO popülasyon matrisi, X_i i'inci GAO üyesi (aday çözüm), d her bir karar değişkeninin sırasıdır (1'den m'ye kadar), x_{id} arama uzayındaki d'inci boyutu (karar değişkeni), N dev armadillo sayısı (popülasyon büyüklüğü), m karar değişkeni sayısı, r [0, 1] aralığında rastgele bir sayı, lb_d ve ub_d sırasıyla d'inci karar değişkeninin alt sınırı ve üst sınırıdır [24].

Her bir dev armadillonun çözüm uzayındaki konumu, ilgili problem için bir aday çözümü temsil etmektedir. Bu nedenle, her bir dev armadillo için amaç fonksiyonu doğrultusunda bir değer belirlenebilir. Buna göre, uygunluk değerler kümesi Denklem (3) ile ifade edilebilir.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{Nx1} = \begin{bmatrix} F_{(X_1)} \\ \vdots \\ F_{(X_i)} \\ \vdots \\ F_{(X_N)} \end{bmatrix}_{Nx1} \quad (3)$$

Burada, F, amaç fonksiyonu değerlerinin vektörüdür ve F_i, i'inci GAO üyesine (dev armadillo) ait uygunluk değeridir.

Popülasyon üyelerinin pozisyonları, keşif (termit yuvalarına doğru hareket) ve sömürü (termit yuvalarında kazı yaparak avlanma) olarak iki aşamada güncellenir. Amaç fonksiyonu değerleri, çözümlerin kalitesini belirler ve her iterasyonda en iyi üye bu değerlere göre güncellenir. Algoritmanın sonunda, en iyi üyenin pozisyonu problemin çözümü olarak sunulur [24].

Termit Yuvalarına Saldırı (Keşif)

GAO algoritmasındaki keşif aşamasında, popülasyon üyelerinin konumları, dev armadillonun termit yuvalarına doğru saldırısını simüle edilerek güncellenir. Bu saldırı, dev armadillonun konumundaki büyük değişikliklere yol açarak algoritmanın küresel keşif gücünü artırır. Her popülasyon üyesi, daha iyi uygunluk değerine sahip diğer üyeleri birer termit yuvası olarak kabul eder ve bu yuvalara yönelerek konumlarını günceller. Her bir popülasyon üyesi için aday termit yuvalarının kümesi, Denklem (4) kullanılarak belirlenir [24].

$$TM_i = \{X_k : F_k > F_i \text{ ve } k \neq 1\}, \quad i = 1,2,\dots,N \text{ ve } k \in \{1,2,\dots,N\} \quad (4)$$

Burada, TM_i i'inci dev armadillo için aday termit yuvalarının konumları kümesidir. X_k i'inci dev armadillodan daha iyi bir amaç fonksiyon değerine sahip popülasyon üyesidir ve F_k onun uygunluk değeridir.

Dev armadillo, aday termit yuvalarından birini rastgele seçer ve ona saldırır. Dev armadilloların termit yuvalarına doğru hareketlerini modelleyerek, her popülasyon üyesi için yeni bir konum Denklem (5) kullanılarak hesaplanır. Ardından, bu yeni konum, eğer Denklem (6)'ya göre uygunluk değerini iyileştiriyorsa, ilgili üyenin önceki konumunun yerine geçer [24].

$$x_{ij}^{P1} = x_{ij} + r_{ij} \cdot \left(SMT_{ij} - I_{ij} \cdot x_{ij}\right) \quad (5)$$

$$X_i = \begin{cases} X_i^{P1}, F_i^{P1} \leq F_i \\ X_i, \quad else \end{cases} \quad (6)$$

Burada, SMT_i i'inci dev armadillo için seçilen termit yuvasıdır, SMT_{ij} bu yuvanın j'inci boyutudur, x_i^{P1} i'inci dev armadillo için önerilen GAO'nun saldırı fazına dayalı olarak hesaplanan yeni konumdur. x_{ij}^{P1}onun j'inci boyutudur, F_i^{P1} onun amaç fonksiyon değeridir, r_{ij} [0, 1] aralığından rastgele sayılardır ve I_{ij} 1 veya 2 olarak rastgele seçilen sayılardır [24].

Termit Yuvalarında Kazma (Sömürü)

GAO algoritmasının sömürü aşamasında, dev armadillonun termit yuvalarında kazı yaparak termitleri avlama süreci simüle edilerek, popülasyon üyelerinin konumları güncellenir. Bu süreç, konumda küçük değişiklikler yaparak algoritmanın yerel arama gücünü artırır. Her popülasyon üyesi için yeni bir konum Denklem (7) ile hesaplanır. Eğer bu yeni konum, uygunluk değerini iyileştirirse, Denklem (8)'e göre önceki konumunun yerine geçer [24].

$$x_{ij}^{P2} = x_{ij} + \left(1 - 2 \cdot r_{ij}\right) \cdot \frac{ub_j - lb_j}{t} \quad (7)$$

$$X_i = \begin{cases} X_i^{P2}, F_i^{P2} \leq F_i \\ X_i, \quad else \end{cases} \quad (8)$$

Burada, x_{ij}^{P2} i'inci dev armadillo için, önerilen GAO'nun kazma fazına dayalı olarak hesaplanan yeni konumdur, x_{ij}^{P2} bunun j'inci boyutudur, F_i^{P2} onun amaç fonksiyon değeridir, r_{ij} [0, 1] aralığından rastgele sayılardır ve t iterasyon sayacıdır.

Tekrarlama Süreci

GAO'nun ilk iterasyonu, dev armadilloların konumlarının saldırı ve kazı fazlarına dayalı olarak güncellenmesiyle tamamlanır. Sonraki iterasyonlarda, popülasyon üyelerinin konumları Denklem (4)–(8) ile güncellenir. Her iterasyonda, en iyi çözüm bulunur ve kaydedilir. Algoritma tamamlandığında, en iyi çözüm, problemin çözümü olarak sunulur [24].

FDBGAO

Meta-sezgisel arama algoritmalarındaki arama sürecinde en etkili çözüm adaylarını belirleme işlemi Uygunluk-Mesafe Dengesi yönteminin asıl amacıdır. FDB yönteminin diğer geliştirme yöntemlerinden farkı, uygunluk değeri ve değişken değerlerin kullanılmasıyla elde edilen bir puanlama sistemine göre seçim yapmasıdır. Puanlama sistemi, her bir bireyin en iyi adaya olan uzaklığı ve en iyi adayın uygunluk değeri ile farkı göz önüne alınarak hesaplanmaktadır. Sonuç olarak FDB puanları aşağıdaki gibi elde edilir: [17].

- İlk adımda, aday çözümlere ait uygunluk değerleri vektörü (*F*) ve Denklem (1)'e göre oluşturulan popülasyon vektörü (*P*) elde edilir.
- İkinci adımda, *i*'inci çözüm adayı (x_i) ile en iyi çözüm (x_{best}) arasındaki Öklid mesafesi Denklem (9)'a göre elde edilir.

$$_{i=1}^{N}\forall P_i D_{Pi} = \sqrt{\left(X_{i,1} - X_{best}\right)^2 + \left(X_{i,2} - X_{best}\right)^2 + \cdots + \left(X_{i,m} - X_{best}\right)^2} \quad (9)$$

- Üçüncü adımda, mesafe vektörü (D_P) Denklem (10)'da verilmiştir.

$$D_P \equiv \begin{bmatrix} d_1 \\ \vdots \\ d_N \end{bmatrix}_{Nx1} \quad (10)$$

- Sonraki adımda, *F* ve D_P vektörleri kullanılarak aday çözüme ait puan hesaplaması yapılır. Bu parametreler normalize edilmelidir. FDB puan değeri (S_{Pi}) Denklem (11)'e göre hesaplanmaktadır.

$$_{i=1}^{N}\forall P_i, \quad S_{Pi} = \omega * normF_i + (1-\omega) * normD_{Pi} \quad (11)$$

- Son olarak puan vektörü (S_P) Denklem (12)'de verilmiştir [17].

$$S_P \equiv \begin{bmatrix} S_1 \\ \vdots \\ S_N \end{bmatrix}_{Nx1} \quad (12)$$

Temel GAO algoritmasında, keşif aşamasında yer alan Denklem (5)'teki seçilen termit yuvası (birey, STM) rastgele bir yöntemle seçilmekte ve bu özellik bakımından algoritma keşif yeteneği açısından bir dezavantaj göstermektedir. Keşif aşamasında rastgele seçilen STM bireyi yerine önerilen FDB yöntemi ile elde edilen puan vektörü sonucunda belirlenmiş en iyi birey kullanılarak sistemdeki rastgelelik ortadan kaldırılmakta ve keşif aşaması daha etkili bir şekilde yapılmaktadır. Böylece, FDBGAO algoritması sunulmuş ve Denklem (5)'teki formül güncellenerek Denklem (13)'te verilmiştir.

$$x_{ij}^{P1} = x_{ij} + r_{ij} \cdot \left(FDB_{ij} - I_{ij} \cdot x_{ij}\right) \quad (13)$$

Denklemde, FDB ifadesi STM yerine seçilen bireyi temsil etmektedir.

Tablo 1. FDBGAO algoritmasının sözde kodu.
Table 1. Pseudocode of the FDBGAO Algorithm [22].

Başla
Girdi problem bilgisi: değişkenler, amaç fonksiyonu ve kısıtları belirt.
FDBGAO algoritmasının popülasyon boyutunu (*N*) ve iterasyon sayısını (*T*) ayarla.
Başlangıç popülasyon matrisi Denklem (2) kullanılarak rastgele oluştur
Amaç fonksiyonunu hesaplayın
for $t = 1 : T$
for *i* = 1 : *N*
Aşama 1: Termit tepeciklerine saldırı (keşif aşaması)
FDBGAO'nun i'inci üyesi için termit yuvaları kümesini Denklem (4)'ü kullanarak güncelle.
FDBGAO'nun i'inci üyesi için termit yuvalarını **FDB yöntemi ile** seçin.
FDBGAO'nun i'inci üyesinin yeni konumunu Denklem (13)'ü kullanarak hesapla.
FDBGAO'nun i'inci üyesini Denklem (6)'ü kullanarak güncelle.
Aşama 2: Termit Yuvalarında Kazma (Sömürme Fazı)
FDBGAO'nun i'inci üyesinin yeni konumunu Denklem (7)'yi kullanarak hesapla.
FDBGAO'nun i'inci üyesini Denklem (8)'ü kullanarak güncelle.
end
Şu ana kadar bulunan en iyi aday çözümü kaydedin.
end
FDBGAO ile elde edilen en iyi yarı-optimal çözümü çıktı olarak verin.
Bitiş FDBGAO

Benzetim Çalışmaları

Bu çalışmada, literatürde yer alan GAO algoritmasının FDB yöntemi kullanılarak geliştirilmesi gerçekleştirilmiş ve FDBGAO algoritması önerilmiştir. Önerilen bu algoritmanın verimliliğini ve geliştirilmesinin etkinliğini kanıtlamak adına literatürde yer alan 23 farklı kıyaslama fonksiyonu için çözümlemeler sağlanmıştır [25]. 23 kıyaslama fonksiyonuna ait detaylı bilgiler Tablo 2'de yer almaktadır. Karşılaştırma işleminin adil olabilmesi adına GAO ve FDBGAO algoritmaları, 30 popülasyon boyutu, 1000 iterasyon sayısı ve 20 bağımsız çalıştırma değerleri ile çalıştırılmıştır. GAO ve FDBGAO algoritmalarının her bir fonksiyon için bağımsız 20 çalıştırma ile elde ettikleri

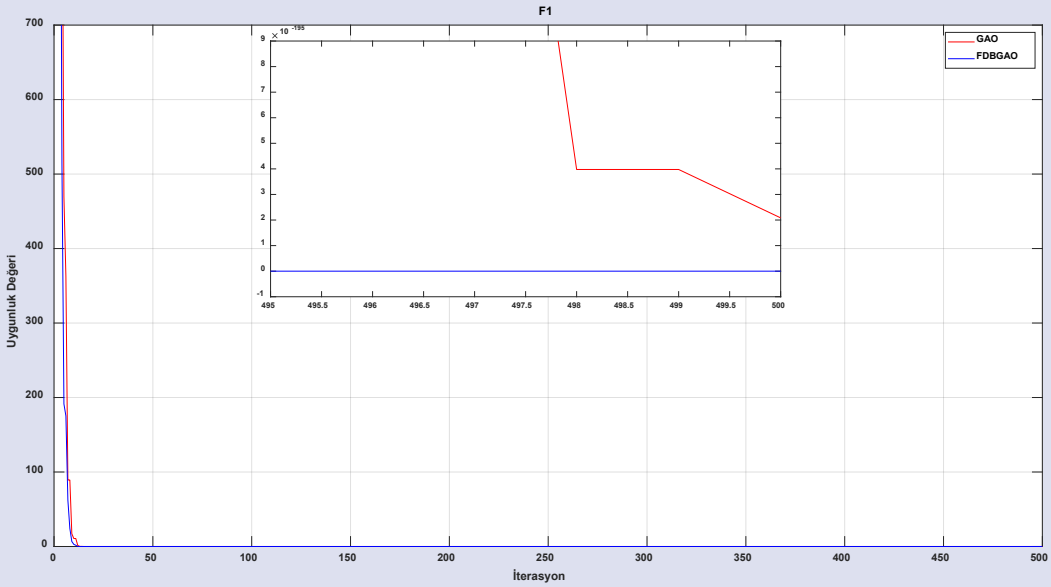
sonuçların ortalama ve standart sapma değerleri Tablo 3'te yer almaktadır.

Tablo 3'te yer alan sonuçlarda da görüldüğü üzere, önerilen FDBGAO algoritması ortalama değer olarak F1, F2, F3, F7, F8, F14 ve F23 arasındaki toplam 15 fonksiyonda temel GAO algoritmasına kıyasla daha iyi sonuçlar vermiştir. Aynı şekilde, F5, F6, F9 ve F13 arasındaki toplam 7 fonksiyonda ise temel GAO algoritması ile aynı sonuçları göstermiştir. Farklı bir şekilde ifade edilmek istenirse, FDBGAO algoritması GAO algoritmasını toplamda 15 kez yenmiş, 7 kez berabere kalmış ve 1 kez yenilmiştir. Bunun sonucunda da önerilen algoritmanın temel algoritmaya kıyasla çok daha iyi sonuçlar vermiştir. Analizler standart sapma değerine göre yapılırsa, yine önerilen FDBGAO algoritması temel GAO algoritmasını toplamda 9 kez yenmiş, 9 kez berabere kalmış ve 5 kez yenilmiştir.

Tablo 2. 23 standart kıyaslama fonksiyonlarının sınıflandırılması ve boyutları.
Table 2. Classification and dimensions of 23 standard benchmarking functions.

Çeşit	Test Fonksiyonları	Boyut	Aralık
Tek-Modlu	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	[-100,100]
	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}x_i$	30	[-10,10]
	$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	[-100,100]
	$f_4(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	30	[-100,100]
	$f_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	30	[-30,30]
	$f_6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	[-100,100]
	$f_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	[-1.28,1.28]
Çok-modlu	$f_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[-500, 100]
	$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[-5.12, 5.12]
	$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[-32, 32]
	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600, 600]
	$f_{12}(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,1$ $y_i=1+\frac{x_i+1}{4},u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	30	[-50, 50]
	$f_{13}(x)=0.1\left\{\sin^2(3\pi x_i)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[-50, 50]
Çok-modlu Düşük Boyutlu	$f_{14}(x)=\left(\frac{1}{500}\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[-65.536, 65.536]
	$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5, 5]
	$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5, 5]
	$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[-5, 10] [0, 15]
	$f_{18}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]\times$ $\left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	[-2, 2]
	$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	[0, 1]
	$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	[0, 1]

$f_{21}(x) = -\sum_{i=1}^{5}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	[0, 10]
$f_{22}(x) = -\sum_{i=1}^{7}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	[0, 10]
$f_{23}(x) = -\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T + c_i\right]^{-1}$	4	[0, 10]



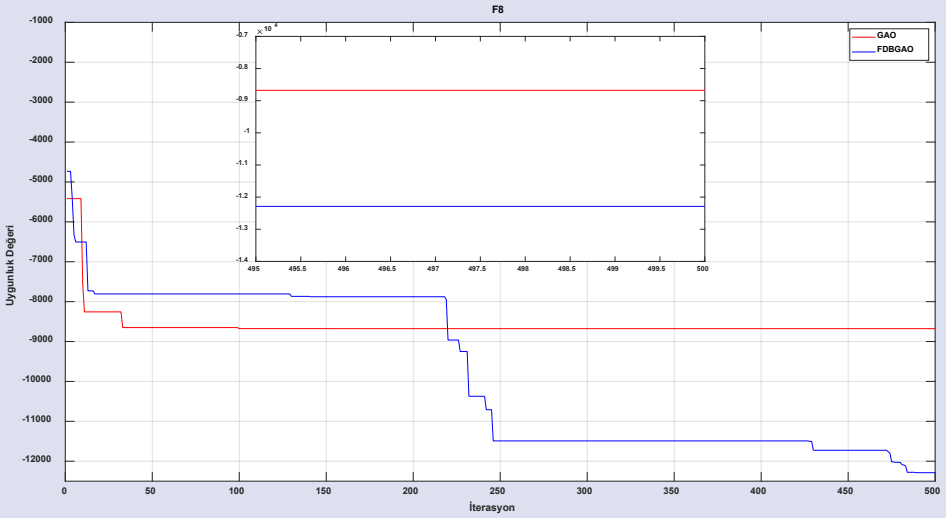
Resim. 1. F1 (Tek-Modlu) Fonksiyonuna Ait Yakınsama Eğrileri.
Figure 1. Convergence Curves for the F1 (Single-Mode) Function.

Tablo 3. 23 Standart Kıyaslama Fonksiyonlarının GAO Ve FDBGAO'ya Ait Ortalama, Standart Sapma Değerleri
Table 3. 23 Average and Standard Deviation Values of 23 Standard Comparison Functions Belonging to GAO and FDBGAO.

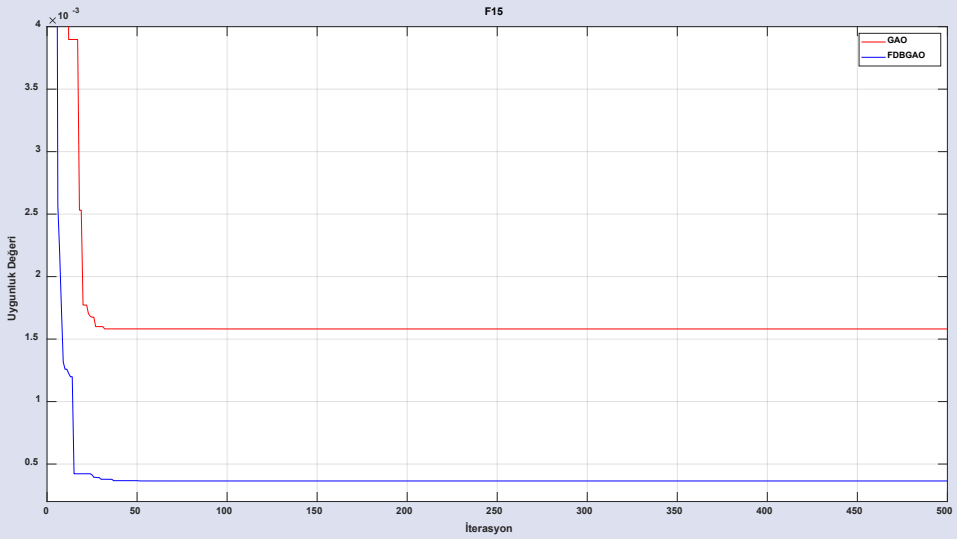
	GAO	FDBGAO	(+/=/-)	GAO	FDBGAO	(+/=/-)
	Ortalama			Standart Sapma		
F1	1.24E-192	2.23E-200	'+'	0.00E+00	0.00E+00	'='
F2	3.32E-98	5.05E-99	'+'	1.49E-97	2.50E-98	'+'
F3	5.38E-192	4.76E-195	'+'	0.00E+00	0.00E+00	'='
F4	4.67E-98	2.80E-81	'-'	2.45E-97	1.53E-80	'-'
F5	0.00E+00	0.00E+00	'='	0.00E+00	0.00E+00	'='
F6	0.00E+00	0.00E+00	'='	0.00E+00	0.00E+00	'='
F7	5.29E-05	4.34E-05	'+'	4.90E-05	3.90E-05	'+'
F8	-9.62E+03	-1.07E+04	'+'	1.55E+03	1.89E+03	'-'
F9	0.00E+00	0.00E+00	'='	0.00E+00	0.00E+00	'='
F10	8.88E-16	8.88E-16	'='	0.00E+00	0.00E+00	'='
F11	0.00E+00	0.00E+00	'='	0.00E+00	0.00E+00	'='
F12	1.57E-32	1.57E-32	'='	5.57E-48	5.57E-48	'='
F13	1.35E-32	1.35E-32	'='	5.57E-48	5.57E-48	'='
F14	9.98E-01	9.98E-01	'+'	1.12E-03	5.66E-07	'+'
F15	9.60E-04	5.02E-04	'+'	4.92E-04	2.52E-04	'+'
F16	-1.03E+00	-1.03E+00	'+'	3.85E-03	6.39E-06	'+'
F17	4.46E-01	3.98E-01	'+'	1.26E-01	5.42E-05	'+'
F18	7.31E+00	3.66E+00	'+'	7.55E+00	2.85E+00	'+'
F19	-3.69E+00	-3.79E+00	'+'	1.67E-01	4.78E-02	'+'
F20	-2.42E+00	-2.85E+00	'+'	3.13E-01	3.16E-01	'-'
F21	-7.85E+00	-8.19E+00	'+'	2.06E+00	2.31E+00	'-'
F22	-8.45E+00	-8.68E+00	'+'	2.10E+00	2.29E+00	'-'

F23	-7.89E+00	-8.84E+00	'+'	2.30E+00	2.04E+00	'+'
(+/=/-)			15/7/1			9/9/5

Tek-modlu, çok-modlu ve çok-modlu düşük boyutlu fonksiyonlar arasından sırasıyla F1, F8 ve F15 fonksiyonları seçilmiş ve algoritmaların yakınsama eğrileri için karşılaştırmalar yapılmıştır. F1 fonksiyonuna ait GAO ve FDBGAO ile elde edilen yakınsama eğrileri Resim 1'de yer almaktadır. Aynı şekilde, F8 ve F15 fonksiyonlarına ait yakınsama eğrileri de sırasıyla Resim 2 ve Resim 3'te verilmiştir.



Resim 2. F8 (çok-modlu) fonksiyonuna ait yakınsama eğrileri.
Figure 2. Convergence curves for the F8 (multi-mode) function.



Resim 3. F15 (çok-modlu düşük boyutlu) fonksiyonuna ait yakınsama eğrileri.
Figure 3. Convergence curves for the F15 (multi-mode low-dimensional) function.

Sonuç

Bu çalışmada, Uygunluk-Mesafe Dengesi (FDB) yöntemi ile son yıllarda literatüre kazandırılan Dev Armadillo Optimizasyon (GAO) algoritması geliştirilmiş ve Uygunluk-Mesafe Dengesi tabanlı Dev Armadillo Optimizasyon (FDBGAO) algoritması elde edilmiştir. Önerilen bu yeni algoritmanın, temel GAO algoritmasındaki keşif aşamasında yer alan ve yerel noktalara takılmasına sebep olan dezavantajı ortadan kaldırması amaçlanmıştır. Elde edilen geliştirilmiş algoritmanın etkinliğinin test edilmesi amacı ile temel GAO ve FDBGAO algoritmaları ile literatürdeki 23 standart kıyaslama fonksiyonları çözümlenmiş ve elde edilen sonuçlar ortalama ve standart sapma değerlerine göre karşılaştırılmıştır. Elde edilen bulgulardan yola çıkarak önerilen FDBGAO algoritmasının temel GAO algoritmasına kıyasla etkin ve başarılı sonuçlar verdiği gözlemlenmiştir.

Kaynaklar

[1] S. Zhao, T. Zhang, S. Ma, ve M. Chen, “Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications”, Eng. Appl. Artif. Intell., c. 114, s. 105075, Eyl. 2022, doi: 10.1016/J.ENGAPPAI.2022.105075.

[2] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness, ve A. H. Gandomi, “Aquila Optimizer: A novel meta-heuristic optimization algorithm”, Comput. Ind. Eng., c. 157, sayı October 2020, s. 107250, 2021, doi: 10.1016/j.cie.2021.107250.

[3] B. Akay ve D. Karaboga, “A survey on the applications of artificial bee colony in signal, image, and video processing”, Signal, Image Video Process., c. 9, sayı 4, ss. 967–990, 2015, doi: https://doi.org/10.1007/s11760-015-0758-4.

[4] M. Şeker, "Long term electricity load forecasting based on regional load model using optimization techniques: A case study," Energy Sources, Part A Recovery, Util. Environ. Eff., vol. 44, no. 1, pp. 21–43, 2021, doi: 10.1080/15567036.2021.1945170.

[5] E. Aslan and Y. Özüpak, "Detection of substation pollution in district heating and cooling systems: A comprehensive comparative analysis of machine learning and artificial neural network models," ITEGAM-J. Eng. Technol. Ind. Appl., vol. 10, no. 50, pp. 17–27, 2024, doi: 10.5935/jetia.v10i50.1289.

[6] J. Kennedy ve R. Eberhart, “Particle swarm optimization”, içinde Proceedings of ICNN’95 - International Conference on Neural Networks, c. 4, ss. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[7] J. H. Holland, “Genetic Algorithms and Adaptation”, Adapt. Control Ill-Defined Syst., ss. 317–333, 1984, doi: 10.1007/978-1-4684-8941-5_21.

[8] R. Storn ve K. Price, “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”, J. Glob. Optim., c. 11, sayı 4, ss. 341–359, 1997, doi: 10.1023/A:1008202821328.

[9] M. Dorigo, M. Birattari, ve T. Stutzle, “Ant colony optimization”, IEEE Comput. Intell. Mag., c. 1, sayı 4, ss. 28–39, Kas. 2006, doi: 10.1109/MCI.2006.329691.

[10] D. Karaboga ve B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm”, J. Glob. Optim., c. 39, sayı 3, ss. 459–471, Kas. 2007, doi: 10.1007/S10898-007-9149-X/METRICS.

[11] S. Mirjalili, S. M. Mirjalili, ve A. Lewis, “Grey Wolf Optimizer”, Adv. Eng. Softw., c. 69, ss. 46–61, Mar. 2014, doi: 10.1016/J.ADVENGSOFT.2013.12.007.

[12] M. Y. Cheng ve D. Prayogo, “Symbiotic Organisms Search: A new metaheuristic optimization algorithm”, Comput. Struct., c. 139, ss. 98–112, 2014, doi: 10.1016/j.compstruc.2014.03.007.

[13] M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, ve N. Khodadadi, “Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm”, Sci. Reports 2024 141, c. 14, sayı 1, ss. 1–50, Şub. 2024, doi: 10.1038/s41598-024-54910-3.

[14] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, ve W. Al-Atabany, “Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems”, Appl. Intell., c. 51, sayı 3, ss. 1531–1551, Mar. 2021, doi: 10.1007/S10489-020-01893-Z/TABLES/14.

[15] M. Leszczuk, S. Szott, P. Trojovský, ve M. Dehghani, “Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications”, Sensors 2022, Vol. 22, Page 855, c. 22, sayı 3, s. 855, Oca. 2022, doi: 10.3390/S22030855.

[16] M. Dehghani, Z. Montazeri, E. Trojovská, ve P. Trojovský, “Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems”, Knowledge-Based Syst., c. 259, s. 110011, Oca. 2023, doi: 10.1016/J.KNOSYS.2022.110011.

[17] H. T. Kahraman, S. Aras, ve E. Gedikli, “Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms”, Knowledge-Based Syst., c. 190, s. 105169, Şub. 2020, doi: 10.1016/j.knosys.2019.105169.

[18] E. Kaymaz, U. Güvenç, ve M. K. Döşoğlu, “Optimal PSS design using FDB-based social network search algorithm in multi-machine power systems”, Neural Comput. Appl., c. 35, sayı 17, ss. 12627–12653, Haz. 2023, doi: 10.1007/S00521-023-08356-9/TABLES/10.

[19] M. R. Sharifi, S. Akbarifard, K. Qaderi, ve M. R. Madadi, “Developing MSA Algorithm by New Fitness-Distance-Balance Selection Method to Optimize Cascade Hydropower Reservoirs Operation”, Water Resour. Manag., c. 35, sayı 1, ss. 385–406, Oca. 2021, doi: 10.1007/S11269-020-02745-8/FIGURES/6.

[20] S. Aras, E. Gedikli, ve H. T. Kahraman, “A novel stochastic fractal search algorithm with fitness-Distance balance for global numerical optimization”, Swarm Evol. Comput., c. 61, s. 100821, Mar. 2021, doi: 10.1016/J.SWEVO.2020.100821.

[21] Y. Sonmez, S. Duman, H. T. Kahraman, M. Kati, S. Aras, ve U. Guvenc, “Fitness-distance balance based artificial ecosystem optimisation to solve transient stability constrained optimal power flow problem”, J. Exp. Theor. Artif. Intell., c. 36, sayı 5, ss. 745–784, Tem. 2024, doi: 10.1080/0952813X.2022.2104388.

[22] Y. Hınıslıoğlu, E. Kaymaz, ve U. Güvenç, “Fitness Distance Balance Based Kepler Optimization Algorithm”, ss. 113–131, 2024, doi: 10.1007/978-3-031-56322-5_10.

[23] Y. Hinislioglu ve U. Guvenc, “A novel hyper-heuristic algorithm: an application to automatic voltage regulator”, Neural Comput. Appl., c. 36, sayı 34, ss. 21321–21364, Ara. 2024, doi: 10.1007/S00521-024-10313-Z/FIGURES/16.

[24] O. Alsayyed vd., “Giant Armadillo Optimization: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems”, Biomimetics 2023, Vol. 8, Page 619, c. 8, sayı 8, s. 619, Ara. 2023, doi: 10.3390/BIOMIMETICS8080619.

[25] X. Yao, Y. Liu, ve G. Lin, “Evolutionary programming made faster”, IEEE Trans. Evol. Comput., c. 3, sayı 2, ss. 82–102, 1999, doi: 10.1109/4235.771163.