**Kan Emici Sülük Optimizasyonu**

**Özet**

Bu makalede, pirinç tarlalarında kan emici sülüklerin yiyecek arama davranışından esinlenerek geliştirilen yeni bir meta-sezgisel optimizasyon algoritması olan Kan Emici Sülük Optimizatörü (BSLO) sunulmaktadır. BSLO, yönlü sülüklerin keşfi, yönlü sülüklerin sömürüsü, yönlü sülüklerin geçiş mekanizması, yönsüz sülüklerin arama stratejisi ve yeniden izleme stratejisi olmak üzere beş avlanma stratejisiyle modellenmiştir. BSLO ve on karşılaştırmalı meta-sezgisel optimizasyon algoritması, yirmi üç klasik benchmark fonksiyonu, CEC 2017 ve CEC 2019'u optimize etmek için kullanılmıştır. BSLO'nun güçlü dayanıklılığı ve optimizasyon verimliliği, dört nitel analiz, iki istatistiksel test ve yakınsama eğrileri ile doğrulanmıştır. Ayrıca, BSLO'nun kısıtlar altındaki gerçek dünya problemlerindeki üstünlüğü, beş klasik mühendislik problemi kullanılarak gösterilmiştir. Son olarak, BSLO tabanlı Yapay Sinir Ağı (ANN) ile eritme elektrospinning yazma lifinin çap tahmini için bir model önerilmiş ve bu, BSLO'nun gerçek dünya problemleri için uygulanabilirliğini daha da doğrulamıştır. Bu nedenle, BSLO çeşitli problemleri optimize etmek için potansiyel bir optimizatördür. BSLO'nun kaynak kodları, kamuya açık olarak erişilebilirdir.

1. **Giriş**

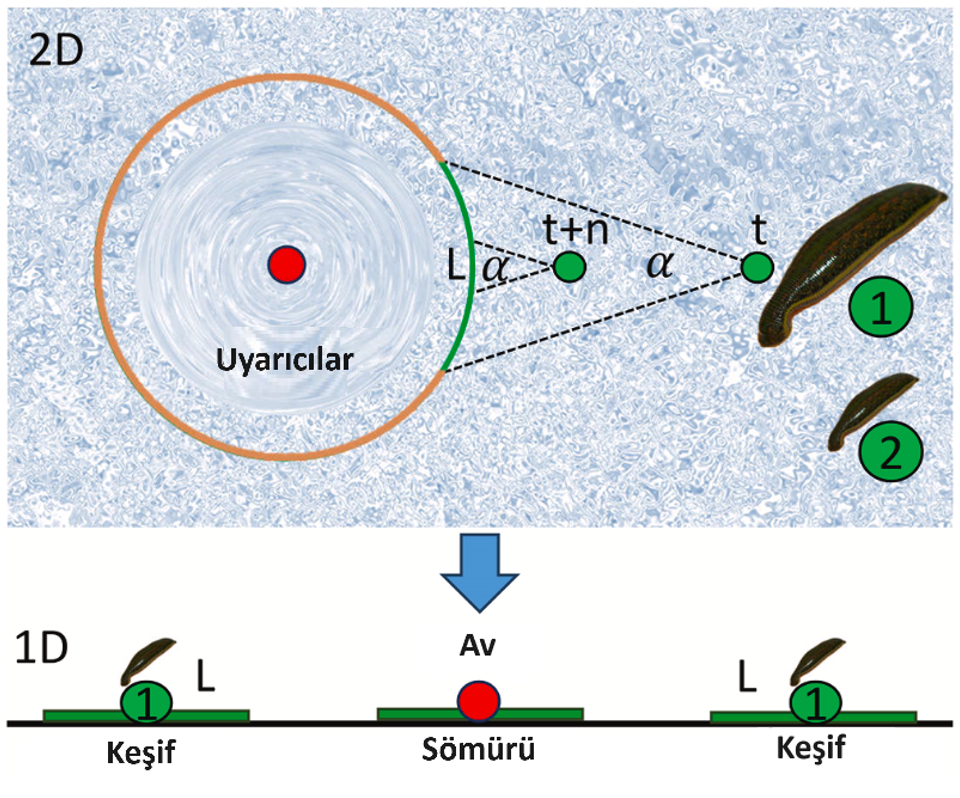
Optimizasyon, maksimum veya minimum değer arayışını hedefler ve bu süreçte kullanılan algoritmalar genellikle deterministik veya meta-sezgisel yöntemler olarak adlandırılır [1]. Deterministik yöntemler küçük ölçekli ve basit problemler için kesin çözümler üretebilir ancak genellikle yerel optimumlarda sıkışabilir, türev bilgisine ihtiyaç duyar ve hesaplama süreçleri zaman alıcı olabilir. Bilim ve toplumun ilerlemesiyle birlikte, daha büyük ölçekli ve karmaşık problemlerin çözümü için meta-sezgisel optimizasyon algoritmalarına olan ihtiyaç artmaktadır [2]. Meta-sezgisel algoritmalar, genellikle deterministik yöntemlere göre daha etkili olarak kabul edilir çünkü türev bilgisine ihtiyaç duymazlar ve çeşitli alanlardaki optimizasyon problemleri için yaygın olarak kullanılırlar [3,4].

Meta-sezgisel optimizasyon algoritmaları, "deneme-yanılma" yöntemiyle optimal çözüm arayışında dört kategoriye ayrılır: Evrimsel Algoritmalar (EA), Fizik Tabanlı Algoritmalar (PhA), İnsan Tabanlı Algoritmalar (HBA) ve Sürü Zekası (SI) Algoritmaları [5]. Evrimsel Algoritmalar, genetik algoritma gibi evrime dayalı yaklaşımlarla popülerdir ve çeşitli optimizasyon problemlerinde kullanılır [6-8]. Fizik Tabanlı Algoritmalar arasında ise Simüle Edilmiş Tavlama gibi yöntemler, termodinamik ilkelerden esinlenmiştir ve yerel optimumlardan kaçınma yetenekleri ile bilinir [14]. İnsan Tabanlı Algoritmalar, insan toplumundaki davranışlardan ilham alarak geliştirilmiştir, örneğin Öğretme-Öğrenme Tabanlı Optimizasyon (TLBO) bu kategoriye örnektir [23]. Sürü Zekası Algoritmaları ise hayvanların avlanma ve sosyal davranışlarından ilham alır ve geniş bir uygulama yelpazesine sahiptir [32-36].

SI algoritmaları, keşif ve sömürü aşamalarını kullanarak optimal çözüm ararlar. Keşif, aday çözümlerin geniş bir çözüm uzayını araştırmasıyla karakterizedir, sömürü ise keşif aşamasında elde edilen potansiyel çözümleri daha hassas bir şekilde işlemekle ilgilidir [74]. Bu süreçte, dengelemek zorlu bir görevdir; aşırı keşif düşük doğruluğa, aşırı sömürü ise yerel optimumlarda sıkışmaya neden olabilir [75]. Bu nedenle, yeni denge stratejileri ve iyileştirmeler sürekli olarak araştırma konusudur, örneğin WOA ve GWO gibi algoritmaların performanslarını artırmak için çeşitli yaklaşımlar geliştirilmiştir [76-83].

Sonuç olarak, meta-sezgisel optimizasyon algoritmaları, çeşitli problem tipleri için güçlü bir araç olup, keşif ve sömürü arasında etkili bir denge sağlama potansiyeline sahiptirler.

Ayrıca, Optimizasyon için No Free Lunch (Bedava Yemek Yok) Teoremi'ne göre hiçbir algoritma tüm problemleri optimize edemez [86]. Başka bir deyişle, bir algoritma bazı problemler için büyük verimlilik sağlayabilirken, diğer problemler için tatmin edici sonuçlar elde edemeyebilir; bu durum, çeşitli problemler için yeni algoritmaların ortaya çıkmasını veya mevcut algoritmaların modifiye edilmesini teşvik eder. Bahsedilen bu iki zorluk göz önüne alındığında, bu çalışmada, pirinç tarlalarındaki kan emici sülüklerin avlanma davranışından ilham alarak, keşif ve sömürü arasında yeni bir denge yöntemi sağlamak üzere Blood-Sucking Leech Optimizer (BSLO) adında yeni bir SI algoritması önerilmektedir. Kan emici sülüklerin uyarılması durumunda, çoğu sülük insanlara küçük bir açıyla yüzerken, bazı sülükler insanların yönünü kaybedebilir. Ayrıca, sülükler insanların ısırıklarını hissettiklerinde tekrar pirinç tarlalarına rastgele atılır ve insanları ararlar. Bu fenomenlere dayanarak, BSLO'nun matematiksel modeli ilk kez geliştirilmiştir. BSLO'nun üstün performansı, yirmi üç klasik benchmark fonksiyonu, CEC 2017, CEC 2019 ve beş klasik mühendislik tasarım problemi kullanılarak gösterilmiştir. Son olarak, BSLO, eritme elektrospinning yazma tekniği olan mikro-nano fiber üretim teknolojisinde lif çapının tahmin edilmesi için Yapay Sinir Ağı (ANN) tahmin modeli için kullanılmıştır [87,88].



Şekil 1 - Kan emici sülüklerin avlanma süreci

Tablo 1 - Kısmi SI Algoritmaları

|  |  |  |
| --- | --- | --- |
| **İsim** | **İlham** | **Yıl** |
| Karınca Aslanı Optimizasyonu (ALO) | Karınca aslanları | 2015 |
| Gizemli Alev Optimizasyonu (MFO) | Güveler | 2015 |
| Salp Sürüsü Algoritması (SSA) | Salp sürüleri | 2017 |
| Çekirge Optimizasyon Algoritması (GOA) | Çekirge sürüleri | 2017 |
| Ayçiçeği Optimizasyon Algoritması (SFO) | Ayçiçekleri | 2018 |
| Manta Ray Yemleme Optimizasyonu (MRFO) | Manta yılanbalıkları | 2020 |
| Dengeli Bileşik Hareket Optimizasyonu | Çözüm uzayı | 2020 |
| Yeni Kaledonya Karga Öğrenme Algoritması (NCCLA) | Yeni Kaledonya kargaları | 2020 |
| Orca Yırtıcılığı Algoritması (OPA) | Orkalar | 2021 |
| Yapay Süzme Kuş Algoritması (AHA) | Süzme kuşlar | 2021 |
| Bal Örümcek Algoritması (HBA) | Bal örümcekleri | 2021 |
| Kızıl Tilki Optimizasyon Algoritması (RFO) | Tilki | 2021 |
| Yapay Tavşanlar Optimizasyonu (ARO) | Tavşanlar | 2022 |
| Martı Optimizasyon Algoritması (GOA) | Martılar | 2022 |
| Koati Optimizasyon Algoritması (COA) | Koatiler | 2022 |
| Sığırcık Sürüsü Optimizasyonu (SMO) | Sığırcıklar | 2022 |
| Termit Hayat Döngüsü Optimizasyonu (TLCO) | Termit kolonisi | 2022 |
| Yangın Şahin Optimizasyonu (FHO) | Şahinler | 2023 |
| Surikat Optimizasyon Algoritması (MOA) | Surikatlar | 2023 |
| Cengiz Han Köpekbalığı Optimizasyonu (GKSO) | Cengiz Han köpekbalıkları | 2023 |
| Mors Optimizasyon Algoritması (WO) | Morslar | 2023 |

1. **Kan Emici Sülük Optimizasyonu**

**2.1 Fikrin Ortaya Çıkışı**

Sülükler, binlerce yıldır tıbbi tedavi için yaygın olarak araştırılan bir hayvandır ve günümüzde kardiyovasküler hastalıklar, rekonstrüktif ve mikrocerrahi, kanser ve metastaz, diabetes mellitus ve komplikasyonları ile enfeksiyon hastalıkları gibi modern tıbbi tedavilerde hala kullanılmaktadır [89]. Dünya genelinde binlerce sülük türü bulunmakta olup, kan emici sülükler tatlısu ortamlarında, örneğin pirinç tarlalarında yaşamayı tercih etmektedir. Kan emici sülükler özellikle Çin pirinç tarlalarında yaygındır ve insanlar dahil omurgalıların kanını emerler. Kan emici sülükler mekanoreseptörler ve kemoreseptörler gibi birçok reseptöre sahiptir [90–92]. Kan emici sülük, su dalgasıyla, sıcaklık gradyanıyla ve su taşınan yiyecek kokularıyla temas ettiğinde, sürekli olarak avı takip edebilir. Eğer av hareket ederse, kan emici sülük avın yeni konumuna yönelir. Ayrıca, çoğu sülük, bu uyaranları hissettiklerinde avına hafif bir açıyla yüzebilir ve yalnızca az bir kısmı yönünü kaybeder [93]. Dolayısıyla, kan emici sülük, avını etkili bir şekilde yönlendirebilir ve avını yakalayabilir. Pirinç tarlalarında insanların ayaklarını ısırıp bir süre kanlarını emdikten sonra insanlar tarafından çıkarılıp rastgele tekrar pirinç tarlalarına atılan bu sülükler, yeniden insanları arayacaklardır. BSLO, kan emici sülüklerin avlanma davranışlarından ilham almakta olup, BSLO'nun matematiksel modeli aşağıdaki bölümlerde sunulacaktır.

**2.2 Başlangıç**

BSLO, diğer SI algoritmaları gibi, ilk olarak çözüm uzayında stokastik olarak dağıtılmış aday çözümler kümesini başlatır. Her aday çözüm bir kan emici sülüğü temsil eder, aşağıdaki denklemde gösterildiği gibi:



burada D, problem boyutunu temsil eder. En küçültme problemleri için, tüm kan emici sülükler aşağıdaki denklemde gösterilir (2), arasında, kan emici sülükler en küçük uygunluk değeri elde eden mevcut yineleme olarak optimal çözüm olarak kabul edilir. Geçmiş en iyi çözüm, mevcut yinelemede daha küçük bir uygunluk değeri ile değiştirilir.

A black and white text

Description automatically generated

Burada Xall, (3) denklemi ile başlatılan kan emici sülükleri temsil eder. xi,j ise ith kan emici sülüğün jth pozisyonunu ve N kan emici sülük sayısını gösterir:



Burada, rand [0, 1] aralığında rastgele bir değer alırken, ub ve lb sırasıyla optimizasyon problemlerinin üst ve alt sınırlarını gösterir.

**2.3 BSLO'nun Matematiksel Modeli**

Kan emici sülüklerin avlanma davranışı Şekil 1'de açıklanmaktadır. Sülükler, yönlendirilmiş ve yönsüz olmak üzere iki türe ayrılır, sırasıyla Denklem (4) ve Denklem (5)'te gösterildiği gibi. Yönlendirilmiş sülükler, insanlar tarafından üretilen dairesel dalga uyarıcılarıyla temas ettiklerinde α○C'de avın yönüne doğru yüzebilirler. Diğer yönsüz sülükler ise rastgele bir şekilde arama alanında yüzerler. Bir süre insanları ısırıktan sonra, sülükler insanlar tarafından rastgele olarak pirinç tarlalarına geri atılır ve tekrar insan ararlar.

A black and white math equation

Description automatically generated

Burada, N1 ve N2 sırasıyla yönlü sülüklerin ve yönsüz sülüklerin sayılarıdır. floor, MATLAB'da aşağı yuvarlama işlevidir. T ve t maksimum ve mevcut yinelemeleri belirtir. m, yönlü sülüklerin ve yönsüz sülüklerin sayısını belirleyen oran parametresini temsil eder ve bu makalede 0.8 olarak ayarlanmıştır. Bu nedenle, sülüklerin çoğu avlanma davranışlarından dolayı başlangıçta yönlü olarak insanları arayabilir. Ayrıca, yinelemeler arttıkça daha fazla sülük insanları bulabilir.

**2.3.1 Yönlü Sülüklerin Keşif Stratejisi**

Kan emici sülükler, su dalgası gibi uyarıcıları hissettiğinde, N1 sülük hafif bir açıyla insanlara doğru yüzebilirler. Şekil 1'de gösterildiği gibi, "t" ve "t+n" farklı yinelemelerdeki sülüğün farklı pozisyonları gösterir, yeşil yay uzunluğu, "t" ve "t+n" pozisyonundaki aynı sülüğün su dalgalarına doğru yüzmesini temsil eder. Bu nedenle, yay uzunluğu L, sülükler insanlara doğru giderek azalır. İterasyonlar arttıkça, sülükler küresel optimuma doğru yaklaşır ve yay uzunluğu L 0'a yaklaşır. Yay uzunluğu L bir boyuta yansıtılır, yeşil L sülüklerin olası bir sonraki pozisyonunu temsil eder. Sülükler, insanlardan uzak L bölgesinde arama yapabilirler, bu da sülüklerin keşif aşaması olarak kabul edilir. Sülüklerin keşfi, aşağıdaki denklemle gösterilmiştir (6).

A group of black and white text

Description automatically generated

İterasyon t mevcut yinelemeyi belirtirken, Xt(i,j) ve Xt+1 (i,j) sırasıyla mevcut ve bir sonraki yinelemedeki i'nin j. pozisyonunu gösterir. k [1, D] aralığında rastgele bir tam sayıdır. Xt(i,k) i'nin k. rastgele pozisyonunu temsil eder. Xt+1 (i,j) 'nin son iki formülü, i'nin diğer boyutlarının mevcut boyutla olan bağlantısını sağlar, bu da i'nin diğer boyutsal pozisyonunun i'nin bir sonraki mevcut boyutsal pozisyonunu etkileyebileceği anlamına gelir. Bu yöntem, keşif aşamasında çeşitliliği artırabilir, ancak i'nin bir sonraki j. pozisyonu üzerindeki ana etki hala mevcut boyutsal pozisyondur. Bu nedenle, a bu çalışmada 0.97 olarak ayarlanmıştır. Burada, rand [0, 1] aralığında rasgele bir değerdir. Eğer rand a'dan küçükse, Xt+1 (i,j) ilk iki formülle hesaplanır. Aksi takdirde, Xt+1 (i,j) son iki formülle hesaplanır. Prey(j), şimdiye kadar elde edilen en iyi çözümün j. pozisyonunu temsil eder. W1, leeches için küçük bir rahatsızlık katsayısıdır, keşif aşamasında çeşitliliği artırır, Denklem (7) olarak verilmiştir. L1 ve L2, Denklem (8) ve Denklem (9) ile hesaplanan L'nin iki boyutudur. İ'nin bir sonraki j. pozisyonu mevcut boyutsal pozisyon tarafından etkileniyorsa, L1 seçilir. Aksi takdirde, L2 seçilir.



Burada, W1 bozulma katsayısının çok küçük olmasını sağlamak için küçük bir değer olan b olarak ayarlanmıştır, 0.001'e. LV(i,j), rastgele bozulmayı sağlayabilen (10) numaralı denkleme göre hesaplanan rastgele bir vektördür.

A close-up of math equations

Description automatically generated

Burada r1, [-1, 1] aralığında rasgele bir değeri, k2 ise [1, floor(N × (1 + t/T))] aralığından rasgele bir tamsayıyı ifade eder. PD, insanlardan sülükler tarafından algılanan mesafeyi taklit etmek için kullanılan, aşağıdaki denklemle hesaplanan algılanan mesafe olarak adlandırılır (12).

A math equations with numbers

Description automatically generated with medium confidence

Lévy(y), levy uçuş dağılım fonksiyonudur; μ ve v, [0, 1] aralığında rastgele değerler alır; σ, Denklem (11)'in ikinci ifadesine dayanarak hesaplanır ve β ise Denklem (14) ile hesaplanır.

A white background with black text

Description automatically generated

β, artan iterasyonlarla birlikte 1.5'ten 2'ye yükselir, bu da Lévy(y) değerlerini azaltabilir (Denklem 2'de gösterildiği gibi). Bu nedenle, sülükler başlangıçta büyük adımlarla çözüm uzayını arayabilir ve sonraki iterasyonlarda küçük adımlarla potansiyel bölgeleri sömürebilir, bu da sülüklerin insanları aramasına yardımcı olur.

**2.3.2 Yönlendirilmiş Sülüklerin Sömürü Stratejisi**

SI algoritmaları, keşif ve sömürü aşamalarına ayrılır ve önerilen BSLO da bu kurala uyar. Arama alanını birçok iterasyon boyunca keşfettikten sonra, N1 sülükleri yavaş yavaş insanlara yaklaşır ve daha yoğun uyarıcılara maruz kalarak insanların potansiyel bölgesini belirler. Bu aşamadan sonra, sülükler insanların yakınındaki L bölgesinde sömürü aşamasına girer, bu da Denklem (15) ile hesaplanır.

A white background with black text

Description automatically generated

A green and red line graph

Description automatically generated

Şekil 2 - İki çalışmada 1000 iterasyon boyunca algılanan mesafe değerleri

Wi, av için çok küçük bir rahatsızlık katsayısıdır ve Denklem (7) ile hesaplanır. W değerinin ve keşif aşamasının farkı, b'nin yinelemelere dayanmasıdır. t < 0.1 x T olduğunda, b değeri keşif aşamasındaki ile aynıdır. Birçok yinelemeden sonra (t > 0.1 x T), sülükler potansiyel bölgeyi bulurlar. En iyi çözümü daha doğru bir şekilde aramak için daha küçük bir rahatsızlık katsayısı gereklidir. Bu nedenle, b 0.00001 olarak ayarlanır. L ve L4, L'nin iki boyutudur ve Denklem (16) ve Denklem (17) ile hesaplanır. Keşif aşamasında belirtildiği gibi, mevcut boyutsal konum, istismar aşamasında i'nci sülüğün bir sonraki j'nci konumunu etkileyecek ana faktördür. Bu nedenle, rand < a olduğunda L3, aksi halde Ly seçilir:

A math equations with numbers

Description automatically generated with medium confidence

Burada r3 [−1,1] aralığında rastgele bir değerdir ve PD, Denklem (12) ile hesaplanır.

**BSLO Pseudo-kodu**

BSLO parametrelerini başlat: t, t1, t2, m, a, b, vb.

Popülasyon Xall'i Denklem (3) kullanarak başlat.

While (t ≤ T) do

Xall'in uygunluk (Fitness) değerlerini değerlendir.

Şu ana kadarki en iyi çözümü bul.

if t > t1 then

/\* Yeniden izleme stratejisi \*/

if F(Xt\_i) = F(Preyt−t2) then

Yeniden izleme çözümlerini güncelle, Denklem (20) kullanarak.

end if

end if

N1 ve N2'yi Denklem (4) ve (5) kullanarak güncelle.

s ve LV(i,j)'yi Denklem (13) ve (10) kullanarak güncelle.

Xall'in k ve k2 değerlerini güncelle.

/\* Yönlü sülüklerin arama stratejileri \*/

for (i = 1 to N1) do

for (j = 1 to D) do

PD'yi Denklem (12) kullanarak hesapla.

if PD ≥ 1 then

/\* Keşif aşaması \*/

Yönlü sülüklerin konumlarını Denklem (6) kullanarak güncelle.

else

/\* Sömürü aşaması \*/

Yönlü sülüklerin konumlarını Denklem (15) kullanarak güncelle.

end if

end for

end for

/\* Yönsüz sülüklerin arama stratejileri \*/

for (i = N2 to N) do

for (j = 1 to D) do

Yönsüz sülüklerin konumlarını Denklem (18) kullanarak güncelle.

end for

end for

t = t + 1.

end while

En iyi çözümü döndür.

**2.3.3 Yönlü Sülüklerin Yönlendirme Mekanizması**

Keşif ve sömürü için bir geçiş mekanizması gereklidir. Algılanan mesafe (PD), sülüklerin insanlardan algıladıkları mesafeyi taklit etmek üzere tasarlanmıştır. Çoğu yönlü sülük başlangıçta insanları keşfeder. Dolayısıyla, başlangıçta çoğu sülük insanlardan uzak olduklarını hisseder, bu yüzden PD değerlerinin çoğu büyüktür. Başlangıçta birkaç PD değeri küçüktür, çünkü bazı sülükler başlangıçta insanlara daha yakındır. Bu durumu tamamen taklit etmek için PD, Denklem (12) ile hesaplanır; Denklem (12), Denklem (13)'de gösterildiği gibi iki senaryoya dayanmaktadır. İterasyonların artmasıyla, daha fazla sülük nihayetinde optimal çözümü bulabilir veya ona yaklaşabilir. Bu nedenle, PD giderek sıfıra yaklaşır. PD > 1 olduğunda, sülükler kendilerini insanlardan uzakta hisseder ve BSLO keşif aşamasına girer. Aksi takdirde, sülükler kendilerini insanlara daha yakın hisseder ve BSLO sömürü aşamasına girer. İterasyonlardaki PD değerleri Şekil 3'te gösterilmiştir. Yönlü sülüklerin başlangıçta genellikle keşif yaptığı ve sonraki iterasyonlarda sömürü yaptığı görülebilir. Ayrıca, sömürü tüm iterasyonlar boyunca mevcuttur.

**2.3.4 Yönsüz Sülüklerin Arama Stratejisi**

Uyarılara tepki veren bazı sülükler (N2), bilgiyi yanlış değerlendirir ve yanlış yöne yüzerler. İterasyonlar arttıkça, daha fazla sülük insanları bulur veya onlara yaklaşırken, N2 sülüklerinin sayısı giderek azalır. Yönsüz sülüklerin matematiksel modelini tasarlamak için Lévy uçuşu adı verilen bir dağılım fonksiyonu eklenir. Bu sayede, yönsüz sülükler pirinç tarlalarında rastgele olarak insanları ararlar, bu da BSLO'nun çeşitliliğini ve yerel optimumlardan kaçınma yeteneğini artırır. Ayrıca, yönsüz sülükler kendileri etrafında veya insanların etrafında rastgele dolaşabilirler. Bu sülüklerin sonraki konumları Denklem (18) ile elde edilir.

A group of math equations

Description automatically generated

BSLO.m

function [Leeches\_best\_score, Leeches\_best\_pos, Convergence\_curve] = BSLO(Arama\_Ajanı\_Sayısı, Max\_iterasyon, lb, ub, dim, fobj)

%% En iyi sülükleri başlat

Leeches\_best\_pos = zeros(1, dim);

Leeches\_best\_score = inf;

%% Arama ajanlarının pozisyonlarını başlatma

Leeches\_Positions = initialization(Arama\_Ajanı\_Sayısı, dim, ub, lb);

Convergence\_curve = zeros(1, Max\_iterasyon);

Temp\_best\_fitness = zeros(1, Max\_iterasyon);

fitness = zeros(1, Arama\_Ajanı\_Sayısı);

%% Parametreleri başlatma

t = 0; m = 0.8; a = 0.97; b = 0.001; t1 = 20; t2 = 20;

% Ana döngü

while t < Max\_iterasyon

N1 = floor((m + (1 - m) \* (t / Max\_iterasyon)^2) \* Arama\_Ajanı\_Sayısı);

% Fitness değerlerini hesaplama

for i = 1:size(Leeches\_Positions, 1)

% Sınır kontrolü

Flag4ub = Leeches\_Positions(i, :) > ub;

Flag4lb = Leeches\_Positions(i, :) < lb;

Leeches\_Positions(i, :) = (Leeches\_Positions(i, :) .\* (~(Flag4ub + Flag4lb))) + ub .\* Flag4ub + lb .\* Flag4lb;

% Her arama ajanı için amaç fonksiyonunu hesaplama

fitness(i) = fobj(Leeches\_Positions(i, :));

% En iyi sülükleri güncelleme

if fitness(i) <= Leeches\_best\_score

Leeches\_best\_score = fitness(i);

Leeches\_best\_pos = Leeches\_Positions(i, :);

end

end

Prey\_Position = Leeches\_best\_pos;

% Yeniden izleme stratejisi

Temp\_best\_fitness(t + 1) = Leeches\_best\_score;

if t > t1

if Temp\_best\_fitness(t + 1) == Temp\_best\_fitness(t + 1 - t2)

for i = 1:size(Leeches\_Positions, 1)

if fitness(i) == Leeches\_best\_score

Leeches\_Positions(i, :) = rand(1, dim) .\* (ub - lb) + lb;

end

end

end

end

if rand() < 0.5

s = 8 - 1 \* (-(t / Max\_iterasyon)^2 + 1);

else

s = 8 - 7 \* (-(t / Max\_iterasyon)^2 + 1);

end

beta = -0.5 \* (t / Max\_iterasyon)^6 + (t / Max\_iterasyon)^4 + 1.5;

LV = 0.5 \* levy(Arama\_Ajanı\_Sayısı, dim, beta);

%% Rastgele tam sayılar oluşturma

minValue = 1; % Minimum tam sayı değeri

maxValue = floor(Arama\_Ajanı\_Sayısı \* (1 + t / Max\_iterasyon)); % Maksimum tam sayı değeri

k2 = randi([minValue, maxValue], Arama\_Ajanı\_Sayısı, dim);

k = randi([minValue, dim], Arama\_Ajanı\_Sayısı, dim);

for i = 1:N1

for j = 1:size(Leeches\_Positions, 2)

r1 = 2 \* rand() - 1; % r1 [0,1] arasında rastgele sayıdır

r2 = 2 \* rand() - 1;

r3 = 2 \* rand() - 1;

PD = s \* (1 - (t / Max\_iterasyon)) \* r1;

if abs(PD) >= 1

% Yönlü sülüklerin keşfi

b = 0.001;

W1 = (1 - t / Max\_iterasyon) \* b \* LV(i, j);

L1 = r2 \* abs(Prey\_Position(j) - Leeches\_Positions(i, j)) \* PD \* (1 - k2(i, j) / Arama\_Ajanı\_Sayısı);

L2 = abs(Prey\_Position(j) - Leeches\_Positions(i, k(i, j))) \* PD \* (1 - (r2^2) \* (k2(i, j) / Arama\_Ajanı\_Sayısı));

if rand() < a

if abs(Prey\_Position(j)) > abs(Leeches\_Positions(i, j))

Leeches\_Positions(i, j) = Leeches\_Positions(i, j) + W1 \* Leeches\_Positions(i, j) - L1;

else

Leeches\_Positions(i, j) = Leeches\_Positions(i, j) + W1 \* Leeches\_Positions(i, j) + L1;

end

else

if abs(Prey\_Position(j)) > abs(Leeches\_Positions(i, j))

Leeches\_Positions(i, j) = Leeches\_Positions(i, j) + W1 \* Leeches\_Positions(i, k(i, j)) - L2;

else

Leeches\_Positions(i, j) = Leeches\_Positions(i, j) + W1 \* Leeches\_Positions(i, k(i, j)) + L2;

end

end

else

% Yönlü sülüklerin kullanımı

if t >= 0.1 \* Max\_iterasyon

b = 0.00001;

end

W1 = (1 - t / Max\_iterasyon) \* b \* LV(i, j);

L3 = abs(Prey\_Position(j) - Leeches\_Positions(i, j)) \* PD \* (1 - r3 \* k2(i, j) / Arama\_Ajanı\_Sayısı);

L4 = abs(Prey\_Position(j) - Leeches\_Positions(i, k(i, j))) \* PD \* (1 - r3 \* k2(i, j) / Arama\_Ajanı\_Sayısı);

if rand() < a

if abs(Prey\_Position(j)) > abs(Leeches\_Positions(i, j))

Leeches\_Positions(i, j) = Prey\_Position(j) + W1 \* Prey\_Position(j) - L3;

else

Leeches\_Positions(i, j) = Prey\_Position(j) + W1 \* Prey\_Position(j) + L3;

end

else

if abs(Prey\_Position(j)) > abs(Leeches\_Positions(i, j))

Leeches\_Positions(i, j) = Prey\_Position(j) + W1 \* Prey\_Position(j) - L4;

else

Leeches\_Positions(i, j) = Prey\_Position(j) + W1 \* Prey\_Position(j) + L4;

end

end

end

end

end

% Yönsüz sülüklerin arama stratejisi

for i = (N1 + 1):size(Leeches\_Positions, 1)

for j = 1:size(Leeches\_Positions, 2)

if min(lb) >= 0

LV(i, j) = abs(LV(i, j));

end

if rand() > 0.5

Leeches\_Positions(i, j) = (t / Max\_iterasyon) \* LV(i, j) \* Leeches\_Positions(i, j) \* abs(Prey\_Position(j) - Leeches\_Positions(i, j));

else

Leeches\_Positions(i, j) = (t / Max\_iterasyon) \* LV(i, j) \* Prey\_Position(j) \* abs(Prey\_Position(j) - Leeches\_Positions(i, j));

end

end

end

t = t + 1;

Convergence\_curve(t) = Leeches\_best\_score;

end

initialization.m

function X = initialization(Arama\_Ajanı\_Sayısı, dim, ub, lb)

Sınır\_no = size(ub, 2); % Değişken sınırlarının sayısı

% Eğer tüm değişkenlerin sınırları eşitse

if Sınır\_no == 1

X = rand(Arama\_Ajanı\_Sayısı, dim) .\* (ub - lb) + lb;

end

% Eğer her bir değişkenin farklı lb ve ub değerleri varsa

if Sınır\_no > 1

for i = 1:dim

ub\_i = ub(i);

lb\_i = lb(i);

X(:, i) = rand(Arama\_Ajanı\_Sayısı, 1) .\* (ub\_i - lb\_i) + lb\_i;

end

end

main.m

clear all

clc

% benchmarksType = 1 için 23 Klasik benchmark fonksiyonları

% benchmarksType = 2 için CEC 2017

% benchmarksType = 3 için CEC 2019

benchmarksType = 1;

if benchmarksType == 1

maxFunc = 23;

elseif benchmarksType == 2

maxFunc = 30;

elseif benchmarksType == 3

maxFunc = 10;

else

exit;

end

Arama\_Ajanı\_Sayısı = 30;

Max\_iterasyon = 1000;

runs = 30;

for fn = 1:maxFunc

Fonksiyon\_adı = strcat('F', num2str(fn));

if benchmarksType == 1

[lb, ub, dim, fobj] = Get\_Functions\_details(Fonksiyon\_adı);

elseif benchmarksType == 2

if fn == 2

continue; % CEC-BC-*2017'nin* istikrarsız davranışı nedeniyle fonksiyon-*2'yi* atla

end

[lb, ub, dim, fobj] = CEC2017(Fonksiyon\_adı);

elseif benchmarksType == 3

[lb, ub, dim, fobj] = CEC2019(Fonksiyon\_adı);

end

En\_iyi\_skorlar\_T = zeros(runs, 1);

OrtalamaKonvEğrisi = zeros(Max\_iterasyon, 1);

Konverjans\_eğrisi = zeros(runs, Max\_iterasyon);

for run = 1:runs

[En\_iyi\_skor, En\_iyi\_pozisyon, cg\_eğrisi] = BSLO(Arama\_Ajanı\_Sayısı, Max\_iterasyon, lb, ub, dim, fobj);

En\_iyi\_skorlar\_T(run) = En\_iyi\_skor;

end

En\_iyi\_skor\_En\_iyi = min(En\_iyi\_skorlar\_T);

En\_iyi\_skor\_Ortalama = mean(En\_iyi\_skorlar\_T);

En\_iyi\_skor\_StandartSapma = std(En\_iyi\_skorlar\_T);

% Çıktı

format long

display([Fonksiyon\_adı, ' En İyi: ', num2str(En\_iyi\_skor\_En\_iyi), ' ', 'Ortalama: ', num2str(En\_iyi\_skor\_Ortalama), ' ', 'Standart Sapma: ', num2str(En\_iyi\_skor\_StandartSapma)]);

end

Kaynaklar

[1] Zaher H, Al-Wahsh H, Eid MH, Gad RSA, Abdel-Rahim N, Abdelqawee IM.

A novel harbor seal whiskers optimization algorithm. Alexandria Eng J 2023;80:

88–109.

[2] Rizk-Allah RM, Saleh O, Hagag EA, Mousa AAA. Enhanced tunicate swarm

algorithm for solving large-scale nonlinear optimization problems. Int J Comput

Intell Syst 2021;14.

[3] Bai J, Li Y, Zheng M, Khatir S, Benaisa B, Abualigah L, et al. A sinh cosh

optimizer. Knowledge-based systems. 2023, 111081.

[4] YiFei L, MaoSen C, Ngoc HT, Khatir S, Abdel Wahab M. Multi-parameter

identification of concrete dam using polynomial chaos expansion and slime

mould algorithm. Comput Struct 2023;281.

[5] Minh H-L, Sang-To T, Abdel Wahab M, Cuong-Le T. A new metaheuristic

optimization based on K-means clustering algorithm and its application to

structural damage identification. Knowl Based Syst 2022;251.

[6] Holland JH. Adaptation in natural and artificial systems: an introductory analysis

with applications to biology, control, and artificial intelligence. MIT press; 1992.

[7] YiFei L, Minh H-L, Khatir S, Sang-To T, Cuong-Le T, MaoSen C, et al. Structure

damage identification in dams using sparse polynomial chaos expansion

combined with hybrid K-means clustering optimizer and genetic algorithm. Eng

Struct 2023;283.

[8] Es-Haghi MS, Shishegaran A, Rabczuk T. Evaluation of a novel Asymmetric

Genetic Algorithm to optimize the structural design of 3D regular and irregular

steel frames. Front Struct Civil Eng 2020;14:1110–30.

[9] Beyer H-G, Schwefel H-P. Evolution strategies–a comprehensive introduction. Nat

Comput 2002;1:3–52.

[10] Juste K, Kita H, Tanaka E, Hasegawa J. An evolutionary programming solution to

the unit commitment problem. IEEE Trans Power Syst 1999;14:1452–9.

[11] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global

optimization over continuous spaces. J Global Optimiz 1997;11:341–59.

[12] Civicioglu P. Backtracking search optimization algorithm for numerical

optimization problems. Appl Math Comput 2013;219:8121–44.

[13] Ghaemi M, Feizi-Derakhshi M-R. Forest optimization algorithm. Expert Syst Appl

2014;41:6676–87.

[14] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing.

Science (1979) 1983;220:671–80.

[15] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search

algorithm. Inf Sci 2009;179:2232–48.

[16] Tahani M, Babayan N. Flow Regime Algorithm (FRA): a physics-based meta-

heuristics algorithm. Knowl Inf Syst 2018;60:1001–38.

[17] Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W. Archimedes

optimization algorithm: a new metaheuristic algorithm for solving optimization

problems. Applied Intell 2020;51:1531–51.

[18] Pereira JLJ, Francisco MB, Diniz CA, Antˆ onio Oliver G, Cunha SS, Gomes GF.

Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global

optimization. Expert Syst Appl 2021;170.

[19] Karami H, Anaraki MV, Farzin S, Mirjalili S. Flow direction algorithm (FDA): a

novel optimization approach for solving optimization problems. Comput Ind Eng

2021;156.

[20] Hashim FA, Mostafa RR, Hussien AG, Mirjalili S, Sallam KM. Fick’s Law

Algorithm: A physical law-based algorithm for numerical optimization. Knowl

Based Syst 2023;260.

[21] Deng L, Liu S. Snow ablation optimizer: A novel metaheuristic technique for

numerical optimization and engineering design. Expert Syst Appl 2023;225.

[22] Abdel-Basset M, Mohamed R, Azeem SAA, Jameel M, Abouhawwash M. Kepler

optimization algorithm: A new metaheuristic algorithm inspired by Kepler’s laws

of planetary motion. Knowl Based Syst 2023;268.

[23] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel

method for constrained mechanical design optimization problems. Comput-Aided

Design 2011;43:303–15.

[24] Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R. The social

engineering optimizer (SEO). Eng Appl Artific Intell 2018;72:267–93.

[25] Samareh Moosavi SH, Bardsiri VK. Poor and rich optimization algorithm: a new

human-based and multi populations algorithm. Eng Appl Artif Intell 2019;86:

165–81.

[26] Askari Q, Saeed M, Younas I. Heap-based optimizer inspired by corporate rank

hierarchy for global optimization. Expert Syst Appl 2020;161.

[27] Zhang Y, Jin Z. Group teaching optimization algorithm: a novel metaheuristic

method for solving global optimization problems. Expert Syst Appl 2020:148.

[28] Naik A, Satapathy SC. Past present future: a new human-based algorithm for

stochastic optimization. Soft comput 2021;25:12915–76.

[29] Dehghani M, Trojovska E, Trojovsky P. A new human-based metaheuristic

algorithm for solving optimization problems on the base of simulation of driving

training process. Sci Rep 2022;12:9924.

[30] Tanhaeean M, Tavakkoli-Moghaddam R, Akbari AH. Boxing Match Algorithm: a

new meta-heuristic algorithm. Soft comput 2022;26:13277–99.

[31] Zhang W, Pan K, Li S, Wang Y. Special Forces Algorithm: A novel meta-heuristic

method for global optimization. Math Comput Simul 2023;213:394–417.

[32] Hussain K, Mohd Salleh MN, Cheng S, Shi Y. Metaheuristic research: a

comprehensive survey. Artif Intell Rev 2018;52:2191–233.

[33] Camacho-Villal´

on CL, Stützle T, Dorigo M. Designing new metaheuristics: manual

versus automatic approaches. Intell Comput 2023;2:0048.

[34] Camacho-Villal´

on CL, Dorigo M, Stützle T. Exposing the grey wolf, moth-flame,

whale, firefly, bat, and antlion algorithms: six misleading optimization techniques

inspired by bestial metaphors. Int Trans Operat Res 2023;30:2945–71.

[35] Camacho-Villal´

on CL, Dorigo M, Stützle T. An analysis of why cuckoo search does

not bring any novel ideas to optimization. Comput Oper Res 2022;142:105747.

[36] Li C, Amanov A, Li Y, Wang C, Wang D, Wahab MA. Prediction of residual stress

distribution induced by ultrasonic nanocrystalline surface modification using

machine learning. Adv Eng Soft 2024;188:103570.

[37] Poli R, Kennedy J, Blackwell T. Particle swarm optimization. Swarm Intell 2007;

1:33–57.

[38] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell

Mag 2006;1:28–39.

[39] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function

optimization: artificial bee colony (ABC) algorithm. J Global Optimiz 2007;39:

459–71.

[40] Gad AG. Particle swarm optimization algorithm and its applications: a systematic

review. Arch Comput Methods Eng 2022;29:2531–61.

[41] Kaya E, Gorkemli B, Akay B, Karaboga D. A review on the studies employing

artificial bee colony algorithm to solve combinatorial optimization problems. Eng

Appl Artif Intell 2022;115.

[42]¨

Oztürk S

¸, Ahmad R, Akhtar N. Variants of Artificial Bee Colony algorithm and its

applications in medical image processing. Appl Soft Comput 2020;97.

[43] Faris H, Aljarah I, Al-Betar MA, Mirjalili S. Grey wolf optimizer: a review of recent

variants and applications. Neural Comput Appl 2017;30:413–35.

[44] Jain M, Singh V, Rani A. A novel nature-inspired algorithm for optimization:

squirrel search algorithm. Swarm Evol Comput 2019;44:148–75.

[45] Khishe M, Mosavi MR. Chimp optimization algorithm. Expert Syst Appl 2020;149.

[46] Li S, Chen H, Wang M, Heidari AA, Mirjalili S. Slime mould algorithm: a new

method for stochastic optimization. Future Generat Comput Syst 2020;111:

300–23.

[47] Hayyolalam V, Pourhaji Kazem AA. Black widow optimization algorithm: a novel

meta-heuristic approach for solving engineering optimization problems. Eng Appl

Artif Intell 2020;87.

[48] Mohammadi-Balani A, Dehghan Nayeri M, Azar A, Taghizadeh-Yazdi M. Golden

eagle optimizer: a nature-inspired metaheuristic algorithm. Comput Ind Eng

2021;152.

[49] Abdollahzadeh B, Gharehchopogh FS, Mirjalili S. African vultures optimization

algorithm: a new nature-inspired metaheuristic algorithm for global optimization

problems. Comput Ind Eng 2021;158.

[50] Braik M, Hammouri A, Atwan J, Al-Betar MA, Awadallah MA. White shark

optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization

problems. Knowl Based Syst 2022;243.

[51] Hashim FA, Hussien AG. Snake optimizer: a novel meta-heuristic optimization

algorithm. Knowl Based Syst 2022;242.

[52] El-kenawy E-SM, Khodadadi N, Mirjalili S, Abdelhamid AA, Eid MM, Ibrahim A.

Greylag goose optimization: nature-inspired optimization algorithm. Expert Syst

Appl 2024;238.

[53] Mirjalili S. The ant lion optimizer. Adv Eng Soft 2015;83:80–98.

[54] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic

paradigm. Knowl Based Syst 2015;89:228–49.

[55] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm

algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng

Soft 2017;114:163–91.

[56] Saremi S, Mirjalili S, Lewis A. Grasshopper optimisation algorithm: theory and

application. Adv Eng Soft 2017;105:30–47.

[57] Gomes GF, da Cunha SS, Ancelotti AC. A sunflower optimization (SFO) algorithm

applied to damage identification on laminated composite plates. Eng Comput

2018;35:619–26.

[58] Zhao W, Zhang Z, Wang L. Manta ray foraging optimization: an effective bio-

inspired optimizer for engineering applications. Eng Appl Artif Intell 2020;87.

J. Bai et al.Advances in Engineering Software 195 (2024) 103696

33

[59] Le-Duc T, Nguyen Q-H, Nguyen-Xuan H. Balancing composite motion

optimization. Inf. Sci. 2020;520:250–70.

[60] Al-Sorori W, Mohsen AM. New Caledonian crow learning algorithm: a new

metaheuristic algorithm for solving continuous optimization problems. Appl Soft

Comput 2020;92.

[61] Jiang Y, Wu Q, Zhu S, Zhang L. Orca predation algorithm: a novel bio-inspired

algorithm for global optimization problems. Expert Syst Appl 2022:188.

[62] Zhao W, Wang L, Mirjalili S. Artificial hummingbird algorithm: a new bio-

inspired optimizer with its engineering applications. Comput Methods Appl Mech

Eng 2022:388.

[63] Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W. Honey badger

algorithm: new metaheuristic algorithm for solving optimization problems. Math

Comput Simul 2022;192:84–110.

[64] Połap D, Wo´ zniak M. Red fox optimization algorithm. Expert Syst Appl 2021:166.

[65] Wang L, Cao Q, Zhang Z, Mirjalili S, Zhao W. Artificial rabbits optimization: a

new bio-inspired meta-heuristic algorithm for solving engineering optimization

problems. Eng Appl Artif Intell 2022;114.

[66] Pan J-S, Zhang L-G, Wang R-B, Sn´

aˇ sel V, Chu S-C. Gannet optimization algorithm:

a new metaheuristic algorithm for solving engineering optimization problems.

Math Comput Simul 2022;202:343–73.

[67] Dehghani M, Montazeri Z, Trojovsk´

a E, Trojovský P. Coati optimization

algorithm: a new bio-inspired metaheuristic algorithm for solving optimization

problems. Knowl Based Syst 2023;259:110011.

[68] Zamani H, Nadimi-Shahraki MH, Gandomi AH. Starling murmuration optimizer:

a novel bio-inspired algorithm for global and engineering optimization. Comput

Methods Appl Mech Eng 2022;392.

[69] Minh H-L, Sang-To T, Theraulaz G, Abdel Wahab M, Cuong-Le T. Termite life

cycle optimizer. Expert Syst Appl 2023;213.

[70] Azizi M, Talatahari S, Gandomi AH. Fire Hawk optimizer: a novel metaheuristic

algorithm. Artif Intell Rev 2022;56:287–363.

[71] Xian S, Feng X. Meerkat optimization algorithm: a new meta-heuristic

optimization algorithm for solving constrained engineering problems. Expert Syst

Appl 2023;231.

[72] Hu G, Guo Y, Wei G, Abualigah L. Genghis Khan shark optimizer: a novel nature-

inspired algorithm for engineering optimization. Adv Eng Inf 2023;58.

[73] Han M, Du Z, Yuen KF, Zhu H, Li Y, Yuan Q. Walrus optimizer: a novel nature-

inspired metaheuristic algorithm. Expert Syst Appl 2024;239.

[74] Chen H, Xu Y, Wang M, Zhao X. A balanced whale optimization algorithm for

constrained engineering design problems. Appl Math Modell 2019;71:45–59.

[75] Zhao S, Zhang T, Ma S, Wang M. Sea-horse optimizer: a novel nature-inspired

meta-heuristic for global optimization problems. Appl Intell 2022;53:11833–60.

[76] Chen H, Li W, Yang X. A whale optimization algorithm with chaos mechanism

based on quasi-opposition for global optimization problems. Expert Syst Appl

2020;158.

[77] Chakraborty S, Saha AK, Chakraborty R, Saha M. An enhanced whale

optimization algorithm for large scale optimization problems. Knowl Based Syst

2021;233.

[78] Nadimi-Shahraki MH, Taghian S, Mirjalili S. An improved grey wolf optimizer for

solving engineering problems. Expert Syst Appl 2021;166.

[79] Yu X, Xu W, Li C. Opposition-based learning grey wolf optimizer for global

optimization. Knowl Based Syst 2021;226.

[80] Jiang J, Zhao Z, Liu Y, Li W, Wang H. DSGWO: an improved grey wolf optimizer

with diversity enhanced strategy based on group-stage competition and balance

mechanisms. Knowl Based Syst 2022;250.

[81] Dong H, Xu Y, Li X, Yang Z, Zou C. An improved antlion optimizer with dynamic

random walk and dynamic opposite learning. Knowl Based Syst 2021;216.

[82] Si T, Miranda PBC, Bhattacharya D. Novel enhanced Salp Swarm Algorithms

using opposition-based learning schemes for global optimization problems.

Expert Syst Appl 2022;207.

[83] Mostafa RR, Gaheen MA, Abd ElAziz M, Al-Betar MA, Ewees AA. An improved

gorilla troops optimizer for global optimization problems and feature selection.

Knowl Based Syst 2023;269.

[84] Yao L, Yuan P, Tsai C-Y, Zhang T, Lu Y, Ding S. ESO: an enhanced snake optimizer

for real-world engineering problems. Expert Syst Appl 2023;230.

[85] Lou T-s, Yue Z-p, Jiao Y-z, He Z-d. A hybrid strategy-based GJO algorithm for

robot path planning. Expert Syst Appl 2024;238.

[86] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans

Evol Comput 1997;1:67–82.

[87] Bai J, Wang H, Gao W, Liang F, Wang Z, Zhou Y, et al. Melt electrohydrodynamic

3D printed poly (epsilon-caprolactone)/polyethylene glycol/roxithromycin

scaffold as a potential anti-infective implant in bone repair. Int J Pharm 2020;

576:118941.

[88] Xiong J, Wang H, Lan X, Wang Y, Wang Z, Bai J, et al. Fabrication of bioinspired

grid-crimp micropatterns by melt electrospinning writing for bone–ligament

interface study. Biofabrication 2022;14:025008.

[89] Abdualkader A, Ghawi A, Alaama M, Awang M, Merzouk A. Leech therapeutic

applications. Indian J Pharm Sci 2013;75:127.

[90] Harley CM, Cienfuegos J, Wagenaar DA. Developmentally regulated multisensory

integration for prey localization in the medicinal leech. J Experiment Biol 2011;

214:3801–7.

[91] Harley CM, Rossi M, Cienfuegos J, Wagenaar D. Discontinuous locomotion and

prey sensing in the leech. J Experiment Biol 2013;216:1890–7.

[92] Miao Y, Guo Q, Shi H, Wang J, Guo L. Study on foraging mechanism of leeches

with different feeding habits based on chemoreception and foraging behavior.

Invertebrate Biol 2023;142.

[93] Dickinson MH, Lent CM. Feeding behavior of the medicinal leech. Hirudo

Medicinalis L J Compar Physiol A 1984;154:449–55.

[94] Shaaban AM, Anitescu C, Atroshchenko E, Rabczuk T. An isogeometric Burton-

Miller method for the transmission loss optimization with application to mufflers

with internal extended tubes. Applied Acoustics 2022;185.

[95] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Soft 2014;69:

46–61.

[96] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Soft 2016;95:

51–67.

[97] Dhiman G, Kumar V. Spotted hyena optimizer: a novel bio-inspired based

metaheuristic technique for engineering applications. Adv Eng Soft 2017;114:

48–70.

[98] Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH. The arithmetic

optimization algorithm. Comput Methods Appl Mech Eng 2021;376.

[99] Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-qaness MAA, Gandomi AH.

Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput Ind Eng

2021;157.

[100] Chopra N, Mohsin Ansari M. Golden jackal optimization: a novel nature-inspired

optimizer for engineering applications. Expert Syst Appl 2022;198.

[101] Seyyedabbasi A, Kiani F. Sand Cat swarm optimization: a nature-inspired

algorithm to solve global optimization problems. Eng Comput 2022;39:2627–51.

[102] Mohammed H, Rashid T. FOX: a FOX-inspired optimization algorithm. Appl Intell

2022;53:1030–50.

[103] Zhao S, Zhang T, Ma S, Chen M. Dandelion optimizer: a nature-inspired

metaheuristic algorithm for engineering applications. Eng Appl Artif Intell 2022;

114.

[104] Gao Q, Xie C, Wang P, Xie M, Li H, Sun A, et al. 3D printed multi-scale scaffolds

with ultrafine fibers for providing excellent biocompatibility. Mater Sci Eng C

Mater Biol Appl 2020;107:110269.

[105] Lan X, Wang H, Bai J, Miao X, Lin Q, Zheng J, et al. Multidrug-loaded electrospun

micro/nanofibrous membranes: fabrication strategies, release behaviors and

applications in regenerative medicine. J Control Release 2021;330:1264–87.

[106] Xie Y, Chen J, Zhao H, Huang F. Prediction of the fiber diameter of melt

electrospinning writing by kriging model. J Appl Polym Sci 2022;139.

[107] Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network

methods for the solution of second order boundary value problems. Comput

Mater Contin 2019;59:345–59.

[108] Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh VM, Guo H, Hamdia K, et al.

An energy approach to the solution of partial differential equations in

computational mechanics via machine learning: concepts, implementation and

applications. Comput Methods Appl Mech Eng 2020;362:112790.

[109] Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis

of Kirchhoff plate. arXiv preprint arXiv; 2021. 210202617.

[110] Bai J, Khatir S, Abualigah L, Abdel Wahab M. Ameliorated Golden jackal

optimization (AGJO) with enhanced movement and multi-angle position

updating strategy for solving engineering problems. Adv Eng Soft 2024;194.

[111] Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced

physics informed neural network for phase-field modeling of fracture. Theoretic

Appl Fracture Mech 2020;106:102447.