Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 44, S. 27-34, Aralık 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 44, pp. 27-34, December 2022

Research Article

Karşıt Tabanlı Öğrenme İle Geliştirilmiş Yapay Denizanası Arama Algoritması

Gülnur Yıldızdan[1*]

[1*] Selçuk Üniversitesi, Kulu Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, Konya, Türkiye (ORCID: 0000-0001-6252-9012), gavsar@selcuk.edu.tr

(6th International Symposium on Innovative Approaches in Smart Technologies (ISAS) 2022 – 8-10 December 2022)

(**DOI:** 10.31590/ejosat.1219071)

ATIF/REFERENCE: Yıldızdan, G. (2022). Karşıt Tabanlı Öğrenme İle Geliştirilmiş Yapay Denizanası Arama Algoritması. *Avrupa Bilim ve Teknoloji Dergisi*, (44), 27-34.

Öz

Bu çalışmada denizanalarının okyanustaki yiyecek arama davranışının modellenmesi ile oluşturulan yapay denizanası arama algoritmasının (JS) performansını geliştirmek amacıyla yeni gelişmiş bir algoritma önerilmiştir. Bunun için JS'ye karşıt tabanlı öğrenme yaklaşımı dahil edilerek popülasyondaki bireylerin arama uzayına daha doğru şekilde dağıtılması sağlanmıştır. Geliştirilmiş algoritma(KJS), standart kıyaslama fonksiyonları üzerinde 10,30,50,100,500 ve 1000 boyut için test edilmiştir. Elde edilen sonuçlar JS ve literatürdeki algoritmalarla karşılaştırılmış, istatistik testler ile yorumlanmıştır. Sonuçlar değerlendirildiğinde önerilen KJS algoritmasının başarılı ve kabul edilebilir sonuçlar ürettiği tespit edilmiştir.

Anahtar Kelimeler: Karşıt tabanlı öğrenme, Sürekli optimizasyon, Yapay denizanası arama algoritması.

Artificial Jellyfish Search Algorithm Developed With Opposition-Based Learning

Abstract

In this study, a newly developed algorithm was proposed to improve the performance of the artificial jellyfish search algorithm (JS), which is created by modeling the foraging behavior of jellyfish in the ocean. For this, an oppositional-based learning approach was included in JS to provide a more accurate distribution of individuals in the population to the search space. The developed algorithm (KJS) was tested on standard benchmark functions for 10,30,50,100,500 and 1000 dimensions. The obtained results were compared with JS and algorithms in the literature and interpreted with statistical tests. When the results were evaluated, it was determined that the proposed KJS algorithm produced successful and acceptable results.

Keywords: Opposition-based learning, Continuous optimization, Artificial jellyfish search algorithm.

[*] Sorumlu Yazar: Selçuk Üniversitesi, Kulu Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, Konya, Türkiye (ORCID: 0000-0001-6252-9012), gavsar@selcuk.edu.tr

1. Giriş

Günlük hayatta karşılaştığımız problemleri içgüdüsel olarak en ideal şekilde çözümlemeye yöneliriz. Bu davranış, kısıtlı kaynakları en iyi şekilde kullanma çabası olarak ifade edilebilir. Farklı canlı türlerinde de rastlanan bu davranış, bir amacı gerçekleştirmek için belli sınır ve kısıtlar içinde en ideal çözümün bulunmasıdır ve optimizasyon olarak isimlendirilir. Mühendislik, ekonomi, ticaret, endüstri vb. bir çok alanda karşılaştığımız optimizasyon problemlerinde hedef, amaç fonksiyonunu minimum ya da maksimum yapacak karar değişkenlerinin tespitidir. Eşitlik ve eşitsizlik kısıtlaması içeren bir minimizasyon problemi Denklem 1'de verilmiştir. Denklemde $\boldsymbol{f}$ amaç fonksiyonu, $\boldsymbol{p}$ eşitsizlik kısıtlayıcı sayısı, $\boldsymbol{q}$ eşitlik kısıtlayıcı sayısı, $\boldsymbol{n}$ karar değişkeni sayısı, $\boldsymbol{Lb}$ ve $\boldsymbol{Ub}$ sırasıyla değişkenin alt ve üst sınır değerlerini ifade eder.

$$\text{Minimize: } f(x)$$

$$\text{Kısıtlar: } g_i(x_1, x_2, \dots, x_n) \leq 0 \quad i = 1,2, \dots, p$$

$$h_i(x_1, x_2, \dots, x_n) = 0 \qquad i = 1,2, \dots, q$$

$$Lb_i \leq i \leq Ub_i \qquad i = 1,2, \dots, n \tag{1}$$

Bu problemlerin çözümüne ve bilgisayar ortamında modellenmesine duyulan ihtiyaç birçok optimizasyon yönteminin önerilmesine sebep olmuştur. Bunlardan biri de son dönemde sayısı hızla artan metasezgisel algoritmalardır. Metasezgisel algoritmalar, doğal fenomenlerden esinlenen, kesin çözümü garanti edemeyen ancak kesin çözüm yakınındaki bir çözüme hızlı ve kolay şekilde ulaşmayı sağlayan algoritmalardır. Yapay Denizanası Arama Algoritması (JS) da son dönemde önerilmiş yeni metasezgisel algoritmalardan biridir (Chou & Truong, 2021). Algoritma, denizanalarının okyanustaki arama davranışlarının modellenmesi ile oluşturulmuştur. Denizanalarının okyanus akıntılarına yada sürüye bağlı olarak yaptığı iki hareket söz konusudur. Bu hareketlerin modellenmesi ile sırasıyla algoritmanın küresel ve yerel arama bölümleri oluşturulmuştur. Bu hareketler arasındaki geçiş ise zaman kontrol mekanizması sayesinde gerçekleştirilir. Algoritma literatürde, polimer değişim membranlı yakıt hücrelerinin optimizasyonu (Gouda, Kotb, & El-Fergany, 2021), güneş termoelektrik iklimlendirme sistemlerinin tahmini (Almodfer et al., 2022), birleşik ısı ve güç ekonomik dağıtımının çözümü (Ginidi, Elsayed, Shaheen, Elattar, & El-Sehiemy, 2021), güç sistemlerinde enerji tasarrufu ve tüketimin azaltılmasını sağlama (Jiang, Dao, Vu, & Ngo, 2021), frekans kısıtlı büyük ölçekli optimizasyon problemlerini çözme (Kaveh, Biabani Hamedani, Kamalinejad, & Joudaki, 2021), tıbbi görüntü bölütleme (Abdel-Basset et al., 2021), video damgalama (Dhevanandhini & Yamuna), küresel optimizasyon problemlerini çözme(Manita & Zermani, 2021; Rajpurohit & Sharma, 2022; Yıldızdan & Baykan, 2021) gibi daha birçok alanda uygulanmış ve başarılı sonuçlar elde etmiştir.

Bu çalışmada, Yapay Denizanası Arama algoritmasının (JS) yakınsama hızını artırmak için karşıt tabanlı öğrenme kavramı(Tizhoosh, 2005) ile geliştirilmiş yeni bir algoritma (KJS) önerilmiştir. Önerilen algoritma, farklı özellikte klasik kıyaslama fonksiyonları üzerinde 10,30,50,100,500 ve 1000 boyut için test edilmiştir. Sonuçlar standart algoritma ve literatürdeki diğer algoritmalarla karşılaştırılmış ve sonuçlar değerlendirilmiştir.

2. Materyal ve Metot

2.1. Yapay Denizanası Arama Algoritması (JS)

Denizanaları, hareketlerini kontrol edebilen canlılardır. Alt bölümleri bir şemsiye gibi kapanıp açılır ve böylece suyu dışarı iterek vücutlarını ileriye doğru hareket ettirebilirler. Bu özelliklerine rağmen çoğunlukla akıntılara ve gelgitlere bağlı olarak suda sürüklenirler. Denizanaları sürü oluşturabilirler ve büyük sürülere denizanası çiçeği ismi verilir. Zayıf yüzen bu canlıların akıntılara göre hareketleri denizanası çiçeklerinin korunması ve karaya oturmalarının engellenmesi için önemlidir. Denizanasının iki hareket durumu söz konusudur: sürü içinde kendi hareketi ve denizanası çiçeklenmesi için okyanus akıntısını takip eden hareketi. Bu hareketlilik, denizanalarına okyanusun her yerinde görünebilme kabiliyeti kazandırır. İşte denizanalarının bu arama davranışları ve hareketleri bu algoritmanın önerilmesi için motivasyon kaynağı olmuştur(Chou & Truong, 2021). JS algoritması, denizanalarının okyanus akıntılarını ya da sürüyü takip ettiği ve bu hareketler arası geçişin 'zaman kontrol mekanizmasını' ile gerçekleştirildiği bir algoritma olarak özetlenebilir.

2.1.1. Algoritma Adımları

- İlk olarak algoritmada popülasyon oluşturulur. JS algoritmasında çözümleri problemin arama uzayı içinde doğru bir şekilde dağıtacak daha iyi bir başlatma yolu bulmak için, tipik rastgele yöntemler yerine kaotik harita kullanılmıştır. Bu amaçla yapılan test işlemleri sonucu, Denklem 2'de verilen lojistik harita (Gandomi, Yang, Talatahari, & Alavi, 2013) formülü kullanarak başlangıç popülasyonunu oluşturmanın, erken yakınsamayı engelleyip, performansı artırdığı tespit edilmiştir (Chou & Truong, 2021). Denklemde X_0 denizanalarının başlangıç popülasyonunu oluşturmak için kullanılır. X_I, i.denizanasının konumunun lojistik kaotik değeridir. X_0 , (0, 1) aralığında rastgele bir sayıdır. $X_0 \notin \{0.0, 0.25, 0.5, 0.75\}$ ve η değeri 4'tür.

$$X_{i+1} = \eta X_i(1 - X_i), \quad 0 \leq X_0 \leq 1 \tag{2}$$

- Popülasyon oluşturulduktan sonra, popülasyondaki en iyi uygunluk değerine sahip denizanası (X^*) belirlenir. Sonra, her denizanasının konumu okyanus akıntısı ya da sürü hareketine göre güncellenir. Okyanus akıntısına göre yeni konumun belirlenmesi (küresel arama) Denklem 3'teki gibi formülize edilir. Denklemde, r ve $r1$ çarpanı (0, 1) aralığında rastgele bir sayıyı, $\beta > 0$ dağılım katsayısını, μ popülasyondaki denizanalarının konumlarının ortalamasını ifade eder. $\beta = 3$ olarak kullanılmaktadır (Chou & Truong, 2021).

$$X_i(t+1) = X_i(t) + r * (X^* - \beta * r1 * \mu) \tag{3}$$

- Denizanalarının sürü içindeki hareketi (yerel arama), pasif hareket ve aktif hareket olmak üzere iki türlüdür. Pasif hareket sırasında denizanaları o anki konumları etrafında Denklem 4'e göre arama yapar, konumlarını güncellerler. Denklem (4)'de $r2$ çarpanı (0,1) aralığında rastgele bir sayı, $\gamma > 0$ denizanalarının hareket miktarı ile ilgili hareket

katsayısı, U_b ve L_b ise sırasıyla arama uzayının üst ve alt sınır değerleridir. γ=0.1 olarak kullanılmaktadır.

$$X_i(t+1) = X_i(t) + r2 * \gamma * (U_b - L_b) \quad (4)$$

.

Aktif hareket ise Denklem 5'teki gibi formülize edilir. Denklemde *r3* çarpanı (0,1) aralığında rastgele bir sayı, D denizanasının hareket yönüdür. Bahsedilen hareket, daima en iyi uygunluk değerli denizanası yönündedir ve Denklem 6'da verilmiştir. Bu denklemde, *j* rastgele bir denizanasının indeksi, *f* uygunluk fonksiyonudur.

$$X_i(t+1) = X_i(t) + r3 * D \quad (5)$$

$$D = \begin{cases} X_i(t) - X_j(t), & Eğer\ f(X_i) < f(X_j) \\ X_j(t) - X_i(t), & aksi\ halde \end{cases} \quad (6)$$

- Okyanus akıntısı ve sürü içindeki hareketler arası geçiş algoritmada 'zaman kontrol mekanizması' sayesinde yapılır. Zaman kontrol fonksiyonu Denklem 7'deki gibi formülize edilir (Chou & Truong, 2021). Denklemde, t iterasyon sayısını, t_{max} maksimum iterasyon sayısını, r4 ise (0,1) aralığında rastgele bir sayıyı gösterir. C_0 ise sabit bir değerdir. $c(t) \geq c_0$ şartı sağlandığında denizanası okyanus akıntısına göre, aksi halde pasif ya da aktif hareket yaparak sürünün içinde hareket eder.

$$c(t) = \left| (1 - \frac{t}{t_{max}}) * (2 * r4 - 1) \right| \quad (7)$$

JS algoritmasının kaba kodu Şekil 1'de verilmiştir.

```
1.  Begin
2.    For i=1:Popülasyon boyutu(N) Do
3.      Denklem 7'ye göre zaman kontrol değerini c(t) hesapla
4.      If c(t)≥c0 then
5.        Denklem 3'e göre denizanası okyanus akıntısını takip eder
6.      Else
7.        If rand(0,1) > (1 − c(t)) then
8.          Denklem 4'e göre denizanası pasif hareket yapar
9.        Else
10.         Denklem 5'e göre denizanası aktif hareket yapar
11.       End if
12.     End if
13.   End For
14. End
```

Şekil 1. JS algoritmasının kaba kodu

2.2. Karşıt Tabanlı Öğrenme(OBL)

Metasezgisel algoritmalarda ilk olarak popülasyon oluşturulur. Popülasyon oluşturulurken genelde popülasyon bireyleri rastgele konumlara dağıtılır. Bir optimizasyon probleminin çözümüne iyi bir popülasyonla başlamak algoritma performansını artıracaktır. Örneğin JS algoritmasında bu amaçla popülasyon rastgele değil, kaotik harita yardımıyla oluşturulmuştur. Literatürde bu amaçla en sık kullanılan yöntem Tizhoosh tarafından 2005'de önerilmiş olan karşıt tabanlı öğrenmedir(OBL)(Tizhoosh, 2005). OBL, rastgele bir sayının karşıt konum değerinin, çözüme yeniden ve rastgele oluşturulacak olan bir sayıdan daha yakın olduğu mantığına dayanmaktadır. Bir sayının karşıt değeriyle birlikte oluşturulan bir popülasyonun daha küçük bir arama ile çözüme yakınsayacağı düşünülür (TEMURTAŞ, Yaşar, & ÖZYÖN, 2017).

- *Karşıt sayı:* $x \in [a, b]$ olmak üzere bir gerçel sayı olsun. x 'in karşıtı Denklem 8'de verildiği gibi tanımlanır.

$$x_k = a + b - x \quad (8)$$

- *Karşıt nokta:* d-boyutlu bir uzayda $x = (x_1, x_2, \dots, x_d)$ noktasının, $x_k = (x_{k1}, x_{k2}, \dots, x_{kd})$ karşıt noktası Denklem 9'da verildiği gibi tanımlanır.

$$x_{kj} = a_j + b_j - x_j \quad (9)$$

- *Karşıt tabanlı optimizasyon:* d-boyutlu bir uzayda $x = (x_1, x_2, \dots, x_d)$ noktası bir aday çözüm, $f(x)$ bu çözümün uygunluk değeri olsun. x'in karşıt noktasının $(x_k = (x_{k1}, x_{k2}, \dots, x_{kd}))$ uygunluk değeri $f(x_k)$, $f(x)$'den daha iyi ise x noktası x_k olarak güncellenir. Aksi halde, x noktası korunur.

2.2. Önerilen Algoritma(KJS)

Metasezgisel algoritmalar popülasyonun oluşturulması ile başlar ve genelde popülasyon rastgele oluşturulur. Yakınsamayı hızlandırmak, yerel optimumlara takılmayı önlemek ve çözümleri arama uzayı içerisinde doğru şekilde dağıtmayı sağlamak algoritma performansına önemli ölçüde katkı sağlayacaktır. Bu çalışmada, çok sayıda metasezgisel algoritmaya daha önce uygulanmış ve algoritmaların performanslarına önemli ölçüde katkı sağladığı ispatlanmış olan OBL mantığı (Mahdavi, Rahnamayan, & Deb, 2018; Xu, Wang, Wang, Hei, & Zhao, 2014) JS algoritmasına dâhil edilmiş ve algoritma performansına olan katkısı incelenmiştir.

Önerilen algoritmada (KJS), karşıt çözümlü popülasyonunun (KÇP) oluşturulmasına ait kaba kod Şekil 2'de verilmiştir.

1. Mevcut $P(N)$ popülasyonunun karşıt popülasyonu $KP(N)$ yi oluştur.
   $KP_{i,j} = a_j + b_j - P_{i,j}$
   $i = 1,2,\dots,N \quad j = 1,2,\dots,d$
2. Popülasyon için N adet en iyi bireyi $\{P \cup KP\}$ den seç.

Şekil 2. KÇP oluşturma kaba kodu

KÇP oluşturma yaklaşımı, önerilen algoritmada atlama oranı (A_o) denilen bir değere bağlı olarak evrimsel süreç içerisinde kullanılır. Böylece, algoritma mevcut popülasyona KÇP yaklaşımı uygulayarak, mevcut olan popülasyondan daha uygun olan yeni çözüm adaylarına atlamaya zorlanır. Bir atlama oranına (yani atlama olasılığına) dayanarak, JS'nin arama denklemleri ile yeni popülasyon üretildikten sonra, karşıt çözümlü popülasyon hesaplanır. Sonrasında, mevcut popülasyon ile karşıt çözümlü popülasyonun birleşiminden en iyi uygunluk değerli bireyler seçilir. Burada A_o için uygun değerin belirlenmesi önemlidir. Büyük bir değer olması JS'nin arama denklemlerinin rolünü

azaltır ve algoritmanın arama sürecinin yanlış yönlendirilmesine neden olur. Bu yüzden, bu çalışmada literatürde daha önce kullanılmış olan 0.1 değeri seçilmiştir (Gupta & Deep, 2019). Bu değer, yerel optimumlardan uzaklaşmaya ve yeni optimum yön sağlamaya yardımcı olur. Bu motivasyonla oluşturulmuş olan KJS algoritmasının kaba kodu Şekil 3'te verilmiştir.

```
Begin
If rand < A_o then
    Mevcut P(N) popülasyonunun karşıt popülasyonu KP(N) yi oluştur.
    Popülasyon için N adet en iyi bireyi {P ∪ KP} den seç.
Else
  For i=1:Popülasyon boyutu(N) Do
    Denklem 7'ye göre zaman kontrol değerini c(t) hesapla
    If c(t)≥c_0 then
      Denklem 3'e göre denizanası okyanus akıntısını takip eder
    Else
      If rand(0,1) > (1 − c(t)) then
        Denklem 4'e göre denizanası pasif hareket yapar
      Else
        Denklem 5'e göre denizanası aktif hareket yapar
      End if
    End if
  End For
End
End
```

Şekil 3. KJS kaba kodu

3. Araştırma Sonuçları ve Tartışma

Önerilen KJS algoritması, özellikleri Tablo 1'de verilen on iki adet standart kıyaslama fonksiyonu üzerinde 10,30,50,100,500 ve 1000 boyut için test edilmiştir. Bu fonksiyonların yedi tanesi tek modlu(U), beş tanesi çok modlu(M) yapıdadır. Algoritmalar her fonksiyon için bağımsız şekilde otuz defa çalıştırılmış ve elde edilen sonuçlara ait istatistiksel hesaplamalar (en iyi, ortanca, en kötü, ortalama ve standart sapma değerleri gibi) yapılmıştır.

Tablo 1. Klasik kıyaslama fonksiyonları

Fonksiyon	Aralık	f_{min}	Özellik
$f1(x)=\sum_{i=1}^{d} x_i^2$	$[-100,100]$	0	U
$f2(x)=\sum_{i=1}^{d} x_i^2+\prod_{i=1}^{d}\lvert x_i\rvert$	$[-10,10]$	0	U
$f3(x)=\sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]$	0	U
$f4(x)=max_i\{\lvert x_i\rvert, 1\le i\le d\}$	$[-100,100]$	0	U
$f5(x)=\sum_{i=1}^{d-1}[100\,(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]$	0	U
$f6(x)=\sum_{i=1}^{d}([x_i+0.5])^2$	$[-100,100]$	0	U
$f7(x)=\sum_{i=1}^{d} i*x_i^4+rand[0,1)$	$[-1.28,1.28]$	0	U
$f8=\sum_{i=1}^{d}-x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]$	$-418.9829\times d$	M
$f9=\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0	M
$f10(x)=\sum_{i=1}^{d}-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)+20+e$	$[-32,32]$	0	M
$f11(x)=\frac{1}{4}\times 10^{-3}\sum_{i=1}^{d}x_i^2-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}}+1)$	$[-600,600]$	0	M
$f12(x)=\frac{\pi}{d}\left\{10\sin(\pi y_1)+\sum_{i=1}^{d-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{d}u(x_i,10,100,4)$ $y_i=\frac{x_i+5}{4}\quad u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & Eğer\ x_i>a\\ k(-x_i-a)^m & Eğer\ x_i<-a\\ 0 & diğer\end{cases}$	$[-50,50]$	0	M

Bu test işlemleri sırasında algoritmalarda kullanılan parametre değerleri ise Tablo 2'de gösterilmiştir.

Tablo 2. Parametre değerleri

Parametre	**Değer**
Popülasyon boyutu (N)	50
Maksimum iterasyon sayısı	500
Yürütme(run) sayısı	30
C_0	0.5
A_0	0.1
Boyut(d)	10,30,50,100, 500,1000

Tablo 3'de standart JS ve önerilen KJS algoritmalarının d=10 için elde edilen sonuçları karşılaştırmalı olarak verilmiştir. Tablo 3 ve devamındaki diğer karşılaştırma tablolarında daha iyi olan sonuçlar kalın yazı tipi ile gösterilmiştir.

Tablo 3. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=10)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	5,29E-20	1,21E-17	3,19E-15	2,01E-16	6,08E-16
	KJS	3,28E-88	9,39E-82	8,08E-73	**2,69E-74**	1,48E-73
F2	JS	1,56E-14	1,72E-13	9,51E-13	2,95E-13	2,74E-13
	KJS	1,39E-46	6,51E-41	1,04E-38	**7,72E-40**	2,15E-39
F3	JS	2,14E-08	8,02E-06	6,34E-03	2,58E-04	1,16E-03
	KJS	6,98E-55	2,38E-45	1,97E-38	**6,98E-40**	3,59E-39
F4	JS	3,98E-04	4,36E-03	7,52E-02	8,78E-03	1,42E-02
	KJS	8,62E-42	2,94E-39	5,30E-37	**7,17E-38**	1,44E-37
F5	JS	6,77E-01	4,70E+00	5,93E+00	4,41E+00	1,27E+00
	KJS	5,47E-05	2,11E+00	5,75E+00	**2,28E+00**	1,86E+00
F6	JS	2,10E-19	8,77E-17	4,56E-14	1,80E-15	8,30E-15
	KJS	4,41E-19	7,38E-18	2,18E-16	**3,18E-17**	4,96E-17
F7	JS	1,95E-04	4,74E-04	1,83E-03	6,71E-04	4,77E-04
	KJS	2,14E-05	2,44E-04	7,28E-04	**2,75E-04**	1,86E-04
F8	JS	-4,19E+03	-3,92E+03	-3,48E+03	-3,89E+03	2,20E+02
	KJS	-4,19E+03	-4,19E+03	-3,43E+03	**-4,08E+03**	1,96E+02
F9	JS	0,00E+00	6,91E-12	2,98E+00	3,62E-01	7,12E-01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	1,25E-10	1,19E-09	8,75E-09	2,17E-09	2,14E-09
	KJS	8,88E-16	8,88E-16	4,44E-15	**1,72E-15**	1,53E-15
F11	JS	0,00E+00	9,99E-16	2,23E-02	3,29E-03	6,23E-03
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	8,15E-20	2,41E-18	1,90E-15	7,40E-17	3,46E-16
	KJS	2,17E-20	9,61E-19	4,51E-17	**4,42E-18**	1,05E-17

Tablo 3'de verilen sonuçlar incelendiğinde tüm fonksiyonlarda önerilen KJS algoritmasının daha başarılı ortalama değerler elde ettiği tespit edilmiştir.

Benzer şekilde, Tablo 4-8 'de verilen tablolarda, JS ve KJS algoritmalarının sırasıyla 30,50,100,500 ve 1000 boyut için karşılaştırmalı sonuçları sunulmuştur. Tablolar incelendiğinde, Tablo 3'de olduğu gibi tüm karşılaştırmalarda KJS algoritmasının daha başarılı olduğu görülmüştür.

Tablo 4. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=30)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	7,75E-07	3,80E-05	5,10E-03	2,93E-04	9,34E-04
	KJS	2,23E-79	3,27E-73	1,35E-66	**6,99E-68**	2,59E-67
F2	JS	2,35E-05	3,52E-04	1,02E-02	1,32E-03	2,54E-03
	KJS	5,18E-41	1,26E-35	8,94E-33	**6,47E-34**	1,99E-33
F3	JS	4,09E+00	8,32E+01	4,35E+02	1,05E+02	1,06E+02
	KJS	5,89E-36	2,89E-27	1,51E-21	**6,99E-23**	2,89E-22
F4	JS	3,47E-02	1,13E-01	7,18E-01	1,60E-01	1,46E-01
	KJS	3,33E-38	3,05E-35	4,44E-33	**3,36E-34**	9,05E-34
F5	JS	5,56E-01	4,22E+00	2,74E+01	8,66E+00	8,87E+00
	KJS	9,35E-03	3,54E+00	2,65E+01	**5,53E+00**	6,62E+00
F6	JS	6,54E-06	5,14E-05	2,32E-03	1,94E-04	4,69E-04
	KJS	1,69E-10	2,12E-08	1,43E-06	**7,99E-08**	2,58E-07
F7	JS	3,96E-04	1,29E-03	3,35E-03	1,41E-03	6,66E-04
	KJS	2,06E-05	1,89E-04	7,61E-04	**2,54E-04**	1,87E-04
F8	JS	-1,08E+04	-8,27E+03	-5,64E+03	-8,15E+03	1,43E+03
	KJS	-1,21E+04	-9,07E+03	-5,95E+03	**-9,17E+03**	1,52E+03
F9	JS	5,59E-05	7,58E-03	3,00E+00	2,18E-01	6,65E-01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	1,73E-04	1,84E-03	1,33E-02	2,61E-03	2,96E-03
	KJS	8,88E-16	4,44E-15	4,44E-15	**3,73E-15**	1,45E-15
F11	JS	2,47E-06	4,16E-05	1,53E-02	1,50E-03	3,74E-03
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	3,64E-08	5,87E-07	1,75E-05	1,81E-06	3,43E-06
	KJS	2,97E-11	4,15E-10	5,75E-09	**8,81E-10**	1,20E-09

Tablo 5. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=50)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	1,38E-04	1,74E-03	1,77E-01	9,91E-03	3,18E-02
	KJS	5,37E-81	4,91E-69	9,69E-63	**3,75E-64**	1,77E-63
F2	JS	2,18E-03	2,27E-02	1,45E-01	3,94E-02	4,18E-02
	KJS	1,90E-37	8,92E-34	7,12E-30	**4,49E-31**	1,47E-30
F3	JS	6,77E+01	6,39E+02	2,15E+03	8,51E+02	6,39E+02
	KJS	5,06E-32	1,65E-23	4,56E-18	**1,61E-19**	8,31E-19
F4	JS	3,20E-02	1,34E-01	7,13E-01	1,94E-01	1,80E-01
	KJS	1,27E-35	9,59E-34	5,63E-32	**8,66E-33**	1,70E-32
F5	JS	2,73E+01	1,24E+02	6,50E+02	1,58E+02	1,32E+02
	KJS	4,59E-01	7,27E+00	4,66E+01	**1,23E+01**	1,30E+01
F6	JS	2,23E-04	5,84E-03	5,32E-02	9,27E-03	1,08E-02
	KJS	7,89E-08	7,68E-07	9,14E-06	**1,71E-06**	2,53E-06
F7	JS	4,25E-04	1,50E-03	4,56E-03	1,67E-03	9,14E-04

	KJS	8,87E-05	2,80E-04	9,29E-04	**3,15E-04**	2,01E-04
F8	JS	-1,62E+04	-1,30E+04	-6,83E+03	-1,21E+04	2,72E+03
	KJS	-1,93E+04	-1,36E+04	-1,08E+04	**-1,40E+04**	2,20E+03
F9	JS	6,36E-03	1,10E-01	8,67E+00	7,54E-01	1,96E+00
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	1,94E-03	9,17E-03	4,65E-02	1,27E-02	1,05E-02
	KJS	8,88E-16	4,44E-15	4,44E-15	**3,14E-15**	1,74E-15
F11	JS	5,39E-05	4,37E-03	2,08E-01	1,77E-02	4,36E-02
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	4,01E-06	2,05E-05	6,78E-05	2,68E-05	1,75E-05
	KJS	1,23E-10	8,39E-09	1,61E-07	**2,05E-08**	3,64E-08

Tablo 6. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=100)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	1,76E-03	7,08E-02	2,15E+00	3,15E-01	5,70E-01
	KJS	1,11E-72	1,30E-66	1,01E-57	**3,44E-59**	1,85E-58
F2	JS	3,22E-02	1,29E-01	3,39E+00	2,90E-01	6,02E-01
	KJS	1,33E-35	3,19E-32	2,67E-25	**8,93E-27**	4,88E-26
F3	JS	8,72E+02	6,47E+03	1,17E+04	6,34E+03	3,09E+03
	KJS	2,88E-28	2,91E-18	7,79E-13	**3,23E-14**	1,43E-13
F4	JS	7,77E-02	2,84E-01	1,34E+00	3,64E-01	2,89E-01
	KJS	4,54E-36	2,11E-33	6,09E-30	**2,62E-31**	1,12E-30
F5	JS	9,66E-01	1,73E+01	9,77E+01	2,11E+01	2,13E+01
	KJS	7,73E-01	5,78E+00	3,79E+01	**8,65E+00**	9,20E+00
F6	JS	4,36E-03	9,79E-02	1,59E+00	2,26E-01	3,66E-01
	KJS	5,62E-07	9,83E-06	6,02E-04	**6,79E-05**	1,44E-04
F7	JS	7,33E-04	1,74E-03	1,63E-02	2,68E-03	3,46E-03
	KJS	6,89E-05	2,40E-04	1,18E-03	**3,28E-04**	2,45E-04
F8	JS	-2,83E+04	-1,93E+04	-1,03E+04	-1,91E+04	4,81E+03
	KJS	-3,74E+04	-2,35E+04	-1,58E+04	**-2,34E+04**	4,27E+03
F9	JS	1,12E-02	2,37E+00	4,10E+01	7,24E+00	1,15E+01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	7,90E-03	3,52E-02	5,83E-01	7,59E-02	1,17E-01
	KJS	8,88E-16	4,44E-15	4,44E-15	**4,09E-15**	1,08E-15
F11	JS	1,76E-03	2,84E-02	1,26E+00	1,33E-01	2,81E-01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	3,75E-05	1,81E-04	8,29E-04	2,16E-04	1,64E-04
	KJS	4,29E-11	4,72E-08	1,24E-06	**1,09E-07**	2,29E-07

Tablo 7. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=500)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	1,19E-01	9,24E-01	2,45E+02	1,33E+01	4,78E+01
	KJS	5,24E-73	1,17E-64	6,11E-52	**2,11E-53**	1,11E-52
F2	JS	5,12E-01	2,17E+00	2,40E+01	3,63E+00	4,48E+00
	KJS	2,43E-35	8,28E-30	4,54E-25	**1,78E-26**	8,26E-26
F3	JS	3,44E+04	2,24E+05	3,16E+05	2,13E+05	7,84E+04
	KJS	1,70E-26	4,19E-10	1,58E-03	**7,12E-05**	2,93E-04
F4	JS	1,71E-01	3,94E-01	1,50E+00	5,34E-01	3,77E-01
	KJS	9,78E-34	1,68E-31	6,98E-28	**3,14E-29**	1,29E-28
F5	JS	5,08E+00	2,20E+02	6,83E+02	2,77E+02	2,31E+02
	KJS	4,92E+00	1,61E+01	1,68E+02	**3,96E+01**	4,36E+01
F6	JS	1,65E-01	2,41E+00	6,66E+01	5,21E+00	1,20E+01
	KJS	7,45E-06	1,54E-04	1,64E-02	**1,12E-03**	3,15E-03
F7	JS	8,38E-04	2,58E-03	7,08E-03	2,81E-03	1,60E-03
	KJS	5,97E-05	3,30E-04	9,99E-04	**3,20E-04**	1,91E-04
F8	JS	-1,06E+05	-6,41E+04	-4,43E+04	-6,83E+04	1,29E+04
	KJS	-1,71E+05	-7,51E+04	-5,31E+04	**-8,28E+04**	2,71E+04
F9	JS	1,87E+00	3,25E+01	1,18E+03	1,20E+02	2,35E+02
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	2,18E-02	9,64E-02	2,27E+00	2,10E-01	4,06E-01
	KJS	8,88E-16	4,44E-15	4,44E-15	**4,32E-15**	6,49E-16
F11	JS	3,03E-02	2,30E-01	4,91E+00	6,50E-01	9,94E-01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	2,02E-04	9,35E-04	9,62E-03	1,26E-03	1,70E-03
	KJS	2,77E-09	1,04E-07	1,14E-06	**2,28E-07**	3,03E-07

Tablo 8. JS ve KJS algoritmalarının karşılaştırma sonuçları (d=1000)

		En iyi	Ortanca	En kötü	Ortalama	S. Sapma
F1	JS	1,67E-01	1,21E+00	2,69E+02	2,08E+01	6,59E+01
	KJS	9,98E-72	8,02E-59	1,16E-52	**4,14E-54**	2,12E-53
F2	JS	1,66E+01	2,45E+02	9,19E+02	2,93E+02	2,13E+02
	KJS	1,58E-33	1,69E-28	1,90E-21	**6,45E-23**	3,46E-22
F3	JS	3,46E+05	8,93E+05	1,63E+06	8,85E+05	2,74E+05
	KJS	1,67E-22	8,79E-09	6,25E+00	**4,44E-01**	1,52E+00
F4	JS	1,84E-01	6,22E-01	4,14E+00	7,75E-01	7,92E-01
	KJS	6,52E-33	9,83E-31	5,50E-27	**3,29E-28**	1,10E-27
F5	JS	2,10E+01	4,40E+02	2,72E+03	6,93E+02	7,08E+02
	KJS	2,47E+00	4,92E+01	6,86E+02	**9,35E+01**	1,36E+02
F6	JS	5,46E-01	7,00E+00	1,47E+02	2,02E+01	3,51E+01
	KJS	3,15E-05	2,66E-04	1,54E-01	**9,94E-03**	3,10E-02
F7	JS	8,57E-04	2,98E-03	1,16E-02	3,45E-03	2,17E-03
	KJS	7,17E-05	3,16E-04	9,45E-04	**3,53E-04**	2,37E-04
F8	JS	-1,76E+05	-1,16E+05	-9,11E+04	-1,21E+05	2,11E+04
	KJS	-3,41E+05	-1,36E+05	-8,15E+04	**-1,52E+05**	5,95E+04
F9	JS	2,30E+00	5,24E+01	2,65E+03	2,02E+02	4,83E+02
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F10	JS	1,68E-02	8,25E-02	8,77E-01	1,20E-01	1,56E-01
	KJS	8,88E-16	4,44E-15	4,44E-15	**4,32E-15**	6,49E-16
F11	JS	2,51E-03	2,61E-01	2,50E+00	4,88E-01	5,71E-01
	KJS	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	0,00E+00
F12	JS	1,00E-04	8,16E-04	1,19E+00	4,30E-02	2,16E-01
	KJS	4,99E-09	9,65E-08	3,02E-06	**3,19E-07**	5,94E-07

Tablo 9. JS ve KJS algoritmalarının literatürdeki algoritmalarla karşılaştırma sonuçları (d=30)

	JS	**KJS**	LXWOA	DDSCA	HGWO	m-SCA
F1	2,93E-04	6,99E-68	6,54E-77	**1,57E−125**	1,21E-32	5,70E-03
F2	1,32E-03	6,47E-34	6,09E-53	**1,25E−64**	9,33E-20	9,11E-04
F3	1,05E+02	**6,99E-23**	1,71E+04	2,43E+03	3,18E-08	8,48E+02
F4	1,60E-01	**3,36E-34**	1,90E+01	2,70E−03	4,16E-08	7,07E-01
F5	8,66E+00	5,53E+00	2,75E+01	**3,63E−01**	2,64E+01	2,96E+01
F6	1,94E-04	**7,99E-08**	3,01E-01	1,82E−07	3,78E-01	1,24E+00
F7	1,41E-03	**2,54E-04**	2,29E-03	4,30E−02	1,49E-03	1,95E-02
F8	-8,15E+03	-9,17E+03	**-1,05E+04**	−9,85E+03	-6,40E+03	-4,27E+03
F9	2,18E-01	**0,00E+00**	**0,00E+00**	**0,00E+00**	2,27E-01	7,81E+01
F10	2,61E-03	**3,73E-15**	4,44E-15	3,29E−14	4,26E-14	3,36E-03
F11	1,50E-03	**0,00E+00**	**0,00E+00**	**0,00E+00**	1,37E-03	3,84E-02
F12	1,81E-06	**8,81E-10**	1,70E-02	7,04E−09	3,43E-01	1,45E-01

Tablo 10. Wilcoxon işaretli sıra ve Friedman testi sonuçları

Wilcoxon işaretli sıra testi						
Algoritmalar	İyi	Kötü	Eşit	p	Sembol	
KJS ***vs*** JS	12	0	0	0,002	+	
KJS ***vs*** LXWOA	7	3	2	0,114	≈	
KJS ***vs*** DDSCA	6	4	2	0,445	≈	
KJS ***vs*** HGWO	12	0	0	0,002	+	
KJS ***vs*** m-SCA	12	0	0	0,002	+	
Friedman test						
	JS	KJS	LXWOA	DDSCA	HGWO	m-SCA
Ortalama Sıra	3,92	**1,75**	3,33	2,50	4,00	5,50

JS ve KJS algoritmaları literatürdeki LXWOA (Singh, 2019), DDSCA (Li, Zhao, & Liu, 2021), HGWO (Zhu, Xu, Li, Wu, & Liu, 2015) ve m-SCA (Gupta & Deep, 2019) algoritmalarıyla da karşılaştırılmıştır. Karşılaştırma literatürde sıklıkla kullanılan 30 boyut için yapılmıştır. Karşılaştırma sonuçları Tablo 9'da verilmiştir. Tablo incelendiğinde, KJS algoritmasının sekiz fonksiyonda en iyi ortalama değerleri elde edildiği tespit edilmiştir. Karşılaştırma sonuçları istatistiksel testler yardımıyla da yorumlanmıştır. Bunun için Wilcoxon işaretli sıra testi ve Friedman testleri kullanılmıştır (García, Molina, Lozano, & Herrera, 2009). Wilcoxon işaretli sıra testi, ilişkili iki ölçüm sonucu arasındaki farkın anlamlı olup olmadığını test etmek için kullanılır. Anlamlılık düzeyini belirleyen p değeri, $\alpha<0.05$ olacak şekilde kullanılmıştır. Test sonuçları Tablo 10'da verilmiştir. Tablodaki İyi, Kötü ve Eşit sütunları sırasıyla KJS algoritmasının diğer algoritmadan kaç fonksiyonda daha iyi, daha kötü ve eşit ortalama değer bulduğunun bilgisini verir. Sembol ise p değerine bağlı olarak karşılaştırılan algoritmalar arasında anlamlı bir fark olup olmadığını gösterir. + ve - sembolleri algoritmalar arasında anlamlı bir fark olduğu anlamına gelir. ≈ ise algoritmalar arasında anlamsal bir farklılık olmadığını gösterir. Tablo 10'a göre, KJS algoritması JS, HGWO ve m-SCA algoritmalarından daha iyi performans göstermiştir ve aralarında anlamlı bir fark vardır. LXWOA ve DDSCA algoritmaları ile ise aralarında anlamlı bir fark bulunmamaktadır ve benzer performans göstermişlerdir.

Friedman testi ise algoritmaları sıralamak ve aralarındaki farkı tespit etmek amacıyla bu çalışmada kullanılmıştır. Tablo 10'da verilen ortalama sıra değerlerine göre önerilen KJS algoritması, 1,75 ortalama sıra değeri ile ilk sırada yer almıştır. JS algoritması ise 3,92 ortalama sıra değeri ile dördüncü sıradadadır. Dolayısıyla yapılan düzenleme ile standart algoritmanın performansının arttığı ve algoritmalar içinde ilk sıraya yükseldiği anlaşılmıştır.

4. Sonuç

Bu çalışmada son dönemde önerilmiş JS algoritmasının performansını geliştirmek amacıyla karşıt tabanlı öğrenme yaklaşımı algoritmanın evrimsel süreci içerisine dahil edilmiştir. Elde edilen gelişmiş KJS algoritmasının performansı, on iki standart kıyaslama fonksiyonu üzerinde altı farklı boyut için test edilmiştir. Elde edilen sonuçlar incelendiğinde, KJS algoritmasının tüm boyutlarda JS'den daha iyi performans gösterdiği anlaşılmıştır. Artan problem boyutu, tüm metasezgisel algoritmalarda olduğu gibi KJS algoritmasının performansında da bir miktar azalmaya neden olmuştur. Buna rağmen, tüm boyutlarda standart algoritmadan daha iyi performans göstermeyi sürdürmüştür. Literatürdeki algoritmalarla da karşılaştırılan KJS algoritması, Friedman testine göre yapılan sıralamada algoritmalar arasında ilk sırada yer almıştır. Wilcoxon işaretli sıra testi sonuçlarına göre ise karşılaştırıldığı algoritmalarla benzer ya da daha iyi performans göstermiştir. Tüm bu sonuçlar ışığında, önerilen KJS algoritmasının tercih edilebilir, yarışmacı ve başarılı bir algoritma olduğu söylenebilir. Gelecek çalışma olarak, önerilen KJS algoritmasının gerçek dünya problemleri ve CEC fonksiyonları üzerindeki performansı incelenecektir.

Kaynakça

Abdel-Basset, M., Mohamed, R., Abouhawwash, M., Chakrabortty, R. K., Ryan, M. J., & Nam, Y. (2021). An Improved Jellyfish Algorithm for Multilevel Thresholding of Magnetic Resonance Brain Image Segmentations.

Almodfer, R., Zayed, M. E., Abd Elaziz, M., Aboelmaaref, M. M., Mohammed, M., & Elsheikh, A. H. (2022). Modeling of a solar-powered thermoelectric air-conditioning system using a random vector functional link network integrated with jellyfish search algorithm. *Case Studies in Thermal Engineering*, 101797.

Chou, J.-S., & Truong, D.-N. (2021). A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation, 389*, 125535.

Dhevanandhini, G., & Yamuna, G. An Efficient Lossless Video Watermarking With Multiple Watermarks Using Artificial Jellyfish Algorithm.

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation, 18*(1), 89-98.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics, 15*(6), 617-644.

Ginidi, A., Elsayed, A., Shaheen, A., Elattar, E., & El-Sehiemy, R. (2021). An Innovative Hybrid Heap-Based and Jellyfish Search Algorithm for Combined Heat and Power Economic Dispatch in Electrical Grids. *Mathematics, 9*(17), 2053.

Gouda, E. A., Kotb, M. F., & El-Fergany, A. A. (2021). Jellyfish search algorithm for extracting unknown parameters of PEM fuel cell models: Steady-state performance and analysis. *Energy, 221*, 119836.

Gupta, S., & Deep, K. (2019). A hybrid self-adaptive sine cosine algorithm with opposition based learning. *Expert Systems with Applications, 119*, 210-230.

Jiang, S.-J., Dao, T.-K., Vu, V.-D., & Ngo, T.-G. (2021). A Power System Economic Load Dispatch Using Jellyfish Search Algorithm. In *Soft Computing for Problem Solving* (pp. 321-331): Springer.

Kaveh, A., Biabani Hamedani, K., Kamalinejad, M., & Joudaki, A. (2021). Quantum-based jellyfish search optimizer for structural optimization, *11*(2), 329-356.

Li, Y., Zhao, Y., & Liu, J. (2021). Dimension by dimension dynamic sine cosine algorithm for global optimization problems. *Applied Soft Computing, 98*, 106933.

Mahdavi, S., Rahnamayan, S., & Deb, K. (2018). Opposition based learning: A literature review. *Swarm and evolutionary computation, 39*, 1-23.

Manita, G., & Zermani, A. (2021). A Modified Jellyfish Search Optimizer With Orthogonal Learning Strategy. *Procedia Computer Science, 192*, 697-708.

Rajpurohit, J., & Sharma, T. K. (2022). Chaotic active swarm motion in jellyfish search optimizer. *International Journal of System Assurance Engineering and Management*, 1-17.

Singh, A. (2019). Laplacian whale optimization algorithm. *International Journal of System Assurance Engineering and Management, 10*(4), 713-730.

TEMURTAŞ, H., Yaşar, C., & ÖZYÖN, S. (2017). Nümerik fonksiyonların optimizasyonu için karşıt tabanlı yeni bir meta-sezgisel algoritma. *Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, 17*(3), 922-937.

Tizhoosh, H. R. (2005). *Opposition-based learning: a new scheme for machine intelligence.* Paper presented at the International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06).

Xu, Q., Wang, L., Wang, N., Hei, X., & Zhao, L. (2014). A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence, 29*, 1-12.

Yıldızdan, G., & Baykan, Ö. K. (2021). *A Novel Artificial Jellyfish Search Algorithm Improved with Detailed Local Search Strategy.* Paper presented at the 2021 6th International Conference on Computer Science and Engineering (UBMK).

Zhu, A., Xu, C., Li, Z., Wu, J., & Liu, Z. (2015). Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. *Journal of Systems Engineering and Electronics, 26*(2), 317-328.