Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd
Anatolian Science

ISSN, e-ISSN: 2548-1304
Volume:9, Issue: 2, pp:151-168,2024
https://doi.org/10.53070/bbd.1455107
Research Paper

Adaptif Lineer Toplayıcı Ağırlıklarının Optimizasyonu: Benekli Sırtlan ve Kum Kedisi Sürü Algoritmalarının Karşılaştırması

Optimization of Adaptive Linear Combiner Weights: A Comparison of Spotted Hyena and Sand Cat Swarm Algorithms

Ebubekir Seyyarer*[1], Taha Yasir Yeşil[2], Abdulmelik Öztunç[3]
[1] Van Yüzüncü Yıl Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Van, Türkiye
[2] Van Yüzüncü Yıl Üniversitesi, Yapay Zeka ve Robotik Ana Bilim Dalı, Van, Türkiye
[3] Van Yüzüncü Yıl Üniversitesi, Yapay Zeka ve Robotik Ana Bilim Dalı, Van, Türkiye
(eseyyarer@yyu.edu.tr, tahayesil4040@gmail.com, abdulmelikoztunc@gmail.com)

Received:Mar.18, 2024 Accepted:Jul.28,2024 Published:Dec.25,2024

Özetçe— Meta sezgisel optimizasyon algoritmaları, optimize edilmesi gereken karmaşık problemlerin çözümünde kullanılan doğal fenomenlerden ve hayvan davranışlarından esinlenen sezgisel tekniklerdir. 2017 yılında benekli sırtlanların avlanma stratejilerine dayalı olarak geliştirilen Benekli Sırtlan Optimizasyon (Spotted Hyena Optimization, SHO) ile 2022 yılında kum kedilerinin sürü davranışlarından ilham alınarak oluşturulan Kum Kedisi Sürü Optimizasyon (Sand Cat Swarm Optimization, SCSO), sürü tabanlı meta sezgisel optimizasyon algoritmalarıdır. Bu çalışmada, SHO ve SCSO algoritmalarını kullanarak belirli bir yöntemle yapılan Uyarlanabilir Sonlu Darbe Yanıtlı (Finite Impulse Response, FIR) Süzgeç ağırlıklarının optimizasyonu yapılmaktadır. Belirli bir gürültüye sahip olan istenilen sinyali elde etmek amacıyla Adaptif Lineer Toplayıcıların ağırlıklarını optimize etmek için hata fonksiyonları kullanılmaktadır. Bu hata fonksiyonları Mean Square Error (MSE) fonksiyonu, Mean Absolute Error (MAE) fonksiyonu ve Least Mean Squared Error (LMS) fonksiyonudur. Bu çalışmada, tüm fonksiyonları kullanarak Adaptif Lineer Toplayıcı işaretleri SHO ve SCSO algoritmaları ile optimize edilmiş ve grafikler aracılığıyla kendi aralarında karşılaştırılmıştır. SHO ve SCSO algoritmaları kullanılarak hata fonksiyonlarından sırası ile SHO için 0.5083 (MSE), 0.7153 (LMS) ve 0.4168 (MAE); SCSO için ise 0.0695 (MSE), 0.2924 (LMS) ve 0.2151 (MAE) sonuçlarına ulaşılmıştır. Grafikler incelendiğinde, en optimal çözümün her iki algoritma için de MSE ile sağlandığı sonucuna varılmaktadır. Çalışma sonuçlarına göre, SCSO algoritmasının SHO algoritmasına göre Adaptif Lineer Toplayıcı tasarımında daha yüksek bir başarı oranına sahip olduğu sonucuna varılmıştır.

Anahtar Kelimeler: *Sezgisel Optimizasyon, Benekli Sırtlan Optimizasyon Algoritması, Kum Kedisi Sürü Optimizasyon Algoritması, Adaptif Lineer Toplayıcı, Hata Fonksiyonu*

Abstract— Meta-heuristic optimization algorithms are intuitive techniques inspired by natural phenomena and animal behaviors, utilized in solving complex problems that require optimization. Spotted Hyena Optimization (SHO), developed based on the hunting strategies of spotted hyenas in 2017, and Sand Cat Swarm Optimization (SCSO), created in 2022 by drawing inspiration from the herd behaviors of sand cats, are examples of herd-based meta-heuristic optimization algorithms. In this study, the optimization of Adaptive Finite Impulse Response (FIR) filter weights is performed using SHO and SCSO algorithms with a specific method. Error functions, including Mean Square Error (MSE), Mean Absolute Error (MAE), and Least Mean Squared Error (LMS), are employed to optimize the weights of Adaptive Linear Collectors aiming to obtain the desired signal with specific noise. Using all these functions, Adaptive Linear Collectors' signals are optimized with SHO and SCSO algorithms and compared through graphs. Results show that using SHO and SCSO algorithms, the respective error values are as follows: for SHO - 0.5083 (MSE), 0.7153 (LMS), and 0.4168 (MAE); for SCSO - 0.0695 (MSE), 0.2924 (LMS), and 0.2151 (MAE). Upon examining the graphs, it is concluded that the most optimal solution for both algorithms is achieved through MSE. According to the study results, SCSO algorithm demonstrates a higher success rate in the design of Adaptive Linear Collectors compared to the SHO algorithm.

Keywords: *Intuitive Optimization, Spotted Hyena Optimization Algorithm, Sand Cat Swarm Optimization Algorithm, Adaptive Linear Combiner, Error Function*

1. GİRİŞ

Adaptif Lineer Toplayıcı, bir işaret (input) ve bir referans (target) değeri arasındaki ilişkiyi öğrenmek için kullanılmaktadır. İşaret, öğrenmek istediğimiz bir özellikken referans ise gerçek çıktı değeridir. Adaptif Lineer Toplayıcı modeli, işaretler üzerindeki ağırlıkların lineer kombinasyonunu kullanarak bir tahmin değeri üretmektedir. Bu tahmin değeri ile gerçek çıktı arasındaki fark (hata) hesaplanarak hata fonksiyonları ile ağırlıklar güncellenmektedir. Örneğin, Adaptif Lineer Toplayıcı sınıflandırma problemlerinde kullanılıyorsa işaretler sınıfları temsil etmektedir. Her bir sınıfın doğru çıktı değeri olmaktadır. Adaptif Lineer Toplayıcı, işaretlerin üzerindeki ağırlıkları güncelleyerek doğru sınıfı tahmin etmek için kullanılmaktadır. Adaptif Lineer Toplayıcı ayrıca regresyon problemleri için de kullanılabilmektedir. Bu durumda işaretler üzerindeki ağırlıkların lineer kombinasyonu bir tahmin değeri üretmektedir. Referans ise gerçek çıktı değeri olmaktadır. Adaptif Lineer Toplayıcı, hata hesaplamak için bu gerçek çıktı değerini kullanarak ağırlıkları güncellemektedir (Karaboğa ve Koyuncu, 2005).

Adaptif Lineer Toplayıcı modeli, eğitim sürecinde işaretler ve referanslar arasındaki ilişkiyi öğrenmek için iteratif bir yaklaşım ile Stokastik Gradyan İniş gibi optimizasyon algoritmalarını kullanmaktadır. Bu nedenle Adaptif Lineer Toplayıcı modeli, eğitim verilerinin büyüklüğüne ve modelin karmaşıklığına bağlı olarak eğitim süresi ve bellek gereksinimleri açısından bazı kısıtlamaları olan bir yöntem olarak karşımıza çıkmaktadır (Brown ve ark, 1993).

Günümüzde bazı sistemlerin matematiksel olarak modellenmesi son derece zorlayıcıdır. Bu karmaşık sistemlerin çözümlenmesi için çeşitli yaklaşımlar önerilmektedir. Geliştirilen bu yaklaşımlar arasında en popüler olanlar genetik algoritma ve yapay sinir ağları gibi benzeri yöntemlerdir. Bu yöntemlerin avantajları olduğu kadar bazı dezavantajları da bulunmaktadır. En büyük avantajları, karmaşık matematiksel modellere ihtiyaç duymadan geniş bir problem yelpazesine uygulanabilir olmalarıdır ancak özellikle bazı özel problemlerde zaman açısından düşük performans gösterebilirler ki bu da dezavantajları arasında yer almaktadır. Bu tür özel problemlere yönelik olarak genel olmayan ve doğadan esinlenerek geliştirilen yöntemler önerilmektedir. Bu yöntemler genellikle optimizasyon odaklıdır ve bu alanda yüksek başarı oranlarına sahip olabilmektedir. Optimizasyon, bir problemde en iyi çözümü bulma sürecidir ve bu süreç, mühendislik gibi çeşitli alanlarda yaygın olarak kullanılmaktadır. Bu nedenle en iyi çözüme ulaşmak için farklı optimizasyon yöntemleri geliştirilmiştir (Aslan ve ark, 2023).

Optimizasyon yöntemleri; tıp, mimari, finans, fizik, elektrik, sosyal bilimler, mühendislik gibi farklı alanlardaki birçok gerçek hayat problemini çözmek için genellikle doğadan ilham alan meta sezgisel optimizasyon yöntemlerine evrilmişlerdir (Kumar ve ark, 2014). 21. yüzyılda teknolojik gelişmeler ve hayatın karmaşıklığındaki artışla birlikte problemlerin çözümünde daha iyi meta sezgisel tekniklere olan ihtiyaç artmaktadır. Bu teknikler en yüksek performansı elde etmek için kullanılmaktadır. Diğer mevcut klasik tekniklerle karşılaştırıldığında verimlilikleri ve karmaşıklıkları nedeniyle daha popüler olmaktadırlar (Blum ve Roli, 2003). Meta sezgisel yöntemler genel olarak doğadan ilham alınarak tasarlanmış veya doğadan esinlenmeyen yöntemler olarak sınıflandırılabilmektedir. Ayrıca yöntemlerin dinamik veya statik amaç fonksiyonuna sahip olması, komşuluk yapısı veya değişken komşuluk yapısına sahip olması, hafıza kullanıp kullanmaması ve tek çözüme veya toplum tabanlı olmasına bağlı olarak da farklı alt kategorilere ayrılabilmektedir (Dhiman ve Kumar, 2017).

Diğer meta sezgisel sınıflandırmalar:

- Evrimsel Algoritmalar
- Fizik Tabanlı Algoritmalar
- Sürü Tabanlı Algoritmalar
- Biyo Esinli Algoritmalar
- Doğadan Esinlenen Algoritmalar (Karaboğa ve Koyuncu, 2005)

Adaptif Lineer Toplayıcı optimizasyonu için farklı meta sezgisel algoritmalar kullanıldığı için bu algoritmaları kullanan pek çok araştırmacı ve çalışma mevcuttur. Burada bazı örnekler sıralanmaktadır:

Karaboga ve Koyuncu (2005) Diferansiyel Gelişim (Differential Evolution, DE) algoritması ile Adaptif Lineer Toplayıcı tasarımı yaparak DE'nin Adaptif Lineer Toplayıcı ile kullanılabileceği sonucuna ulaşıldığı görülmüştür (Karaboğa ve Koyuncu, 2005).

Seifossadat ve ark (2007) tarafından yapılan çalışmada bir iletim hattında voltaj ve akım kullanılarak arıza durumunun dalga biçimi Genetik algoritmalar (Genetic Algorithm, GA) ile test edilmiş ve sonuçlar başarılı bulunmuştur (Seifossadat ve ark, 2007).

Najjarzadeh ve Ayatollahi (2008) tarafından yapılan çalışmada Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) kullanarak FIR filtre tasarımı yapılmıştır. Ek olarak LMS ve Minimax gibi çeşitli hata normlarını ve bunların yakınsama davranışlarının optimal sonuç frekans tepkisi üzerindeki etkileri incelenmiştir (Najjarzadeh ve Ayatollahi, 2008).

Ababneh ve Bataineh (2008) tarafından yapılan çalışmada lineer fazlı bir FIR filtresi, parçacık sürü optimizasyonu ve genetik algoritma kullanılarak tasarlanmıştır. Çalışmada PSO ve GA, filtre katsayılarının sonlu kelime uzunluğu kullanılarak temsil edilen optimum FIR filtrelerini tasarlamak için kullanılmıştır. Tüm durumlarda, PSO'nun GA'ya kıyasla daha iyi performans gösterdiği sonucuna varılmıştır (Ababneh ve Bataineh, 2008).

Majhi ve ark (2009) tarafından yapılan çalışmada Parçacık Sürü Optimizasyonu kullanılarak Adaptif Lineer Toplayıcı tasarlanmıştır. Daha sonra tasarlanan bu Adaptif Lineer Toplayıcı ile hisse senedi endekslerinin tahmini yapılmıştır (Majhi ve ark, 2009).

Saha ve ark (2013) tarafından yapılan çalışmada FIR düşük geçiş, yüksek geçiş, bant geçişli ve bant duraklı filtrelerin en iyi optimal impulz tepki katsayılarını belirlemek için Kedi Sürüsü Optimizasyonu (Cat Swarm Optimization, CSO) algoritması kullanılmıştır. Elde edilen sonuçlar, Gerçek Kodlu Genetik Algoritma (Real Coded Genetic Algorithm, RGA), Standart Parçacık Sürü Optimizasyonu ve Diferansiyel Evrim gibi diğer iyi bilinen optimizasyon yöntemleriyle karşılaştırılmıştır. CSO ile tasarlanan FIR filtrelerinin performansının, RGA, PSO ve DE ile elde edilenlere kıyasla üstün olduğu sonucuna varılmıştır (Saha ve ark, 2013).

Boudjelaba ve ark (2014) tarafından yapılan çalışmada FIR filtre tasarımı için iki meta sezgisel yöntem olan Parçacık Sürü Optimizasyonu ve Genetik Algoritmaların performansı incelenmiştir. PSO'nun GA'yı bazı performans kriterlerinde aştığını ancak Adaptif Genetik Algoritma (AGA)'nın, CPU çalışma süresi hariç olmak üzere tüm kriterlerde daha iyi olduğu sonucuna varılmıştır (Boudjelaba ve ark, 2014).

Reddy ve Sahoo (2015) tarafından yapılan çalışmada Diferansiyel Evrim algoritması kullanılarak FIR süzgeç ağırlıkları optimize edilmiştir (Reddy ve Sahoo, 2015).

Shao ve ark (2017) tarafından yapılan çalışmada refrPSO adlı geliştirilmiş bir parçacık sürü optimizasyon algoritmasını doğrusal fazlı FIR düşük geçiş ve yüksek geçişli dijital filtrelerin tasarımı ve optimizasyonu için kullanılmıştır. RefrPSO algoritmasının FIR dijital filtrelerin tasarımı ve optimizasyonunda üstün sonuçlar elde ettiği sonucuna varılmıştır (Shao ve ark, 2017).

Dwivedi ve ark (2018) tarafından yapılan çalışmada 2018 yılına kadar literatürdeki optimizasyon algoritmaları ile yapılmış olan FIR süzgeç tasarımları bir araya getirerek karşılaştırılmıştır (Dwivedi ve ark, 2018).

Srivastava ve ark (2020) tarafından yapılan çalışmada çok amaçlı biyocoğrafya tabanlı optimizasyon kullanılarak seyrek FIR filtre tasarımı ele alınmıştır. Önerilen yöntemin mevcut algoritmalardan daha iyi performans sergilediği ve pratik uygulamalarda kullanılabileceği sonucuna varılmıştır (Srivastava ve ark, 2020).

Yadav ve ark (2021) tarafından yapılan çalışmada Çekirge Optimizasyon Algoritması (Grasshopper Optimization Algorithm, GOA) ile hata fonksiyonları kullanılarak FIR süzgeç tasarımı yapılmıştır (Yadav ve ark, 2021).

Singh ve ark (2022) tarafından yapılan çalışmada Çekirge Optimizasyon Algoritması (GOA) kullanarak sonlu impuls tepki filtresi (FIR), düşük geçiş filtresi (Low-Pass Filter, LPF) ve yüksek geçiş filtresi (High-Pass Filter, HPF) tasarlanmıştır. Ortalama kare hatası (MSE), hata amaç fonksiyonu olarak alınmıştır. Elde edilen sonuçlar, GOA'nın diğer iki algoritma olan Parçacık Sürü Optimizasyon algoritması ve Gri Kurt Optimizasyonu (Grey Wolf Optimization, GWO) algoritması ile karşılaştırılmıştır. Simülasyon sonuçları, GOA'nın FIR filtre tasarım problemi için en uygun algoritma olduğunu sonucu ortaya koyulmuştur (Singh ve ark, 2022).

Kaur ve ark (2023) Bat ve Seeker optimizasyonlarını içeren hibrit bir teknik kullanılarak yeni bir optimizasyon algoritması ortaya atılmıştır. Tüm tekniklerin FIR filtrelerinin değişken özellikleri ve farklı dereceleri açısından karşılaştırmalı analizi sunulmuştur. Hibrit yaklaşım, tüm tekniklerin tasarım sonuçlarını karşılaştırırken üstünlüğünü göstererek en iyi sonuçları ortaya koyduğu sonucuna varılmıştır (Kaur ve ark, 2023).

SHO algoritması ile SCSO algoritması, Adaptif Lineer Toplayıcı modelinin eğitiminde kullanılabilecek bir optimizasyon algoritmasıdır. Adaptif Lineer Toplayıcı, bir işaret ve referans arasındaki ilişkiyi öğrenmek için kullanılırken SHO ve SCSO algoritmaları bu öğrenme sürecinde ağırlıkların güncellenmesinde kullanılmaktadır. SHO algoritması, 2017 yılında benekli sırtlan davranışlarından ilham alan bir arama stratejisi olarak kullanılmaktadır (Dhiman ve Kumar, 2017). SCSO algoritması ise kum kedilerinin avlanma stratejilerinden ilham alınarak 2022 yılında ortaya atılmış bir algoritmadır (Seyyedabbasi ve Kiani, 2023). Adaptif Lineer Toplayıcı modeli, bir işaret üzerindeki ağırlıkları kullanarak bir tahmin değeri üretmektedir. Bu tahmin değeri ile gerçek çıktı arasındaki hata da hesaplanmaktadır. SHO ve SCSO algoritmaları, bu hatayı minimize etmek için işaretler üzerindeki ağırlıkları güncellemektedirler. Adaptif Lineer Toplayıcı modeli, işaretler üzerindeki ağırlıkları kullanarak bir tahmin değeri üretirken, SHO ve SCSO algoritmaları bu tahmin değeri ile gerçek çıktı arasındaki hatayı minimize etmek için işaretler üzerindeki ağırlıkları güncellemeyi hedeflemektedirler.

Bu makalede son yıllarda ortaya atılmış olan sürü tabanlı optimizasyon algoritmalarından olan SHO ve SCSO Algoritmaları ile Adaptif Lineer Toplayıcı tasarımının yapılması amaçlanmaktadır. Problemin daha önce iki algoritma ile de uygulanmamış olması çalışmanın özgün değerini vurgulamaktadır. 5 yıl aralık ile ortaya atılmış bu iki meta sezgisel algoritma birbiri ile karşılaştırılarak SCSO algoritmasının SHO algoritmasına göre Adaptif Lineer Toplayıcı tasarımında daha yüksek bir başarı oranına sahip olduğu sonucuna varılmaktadır.

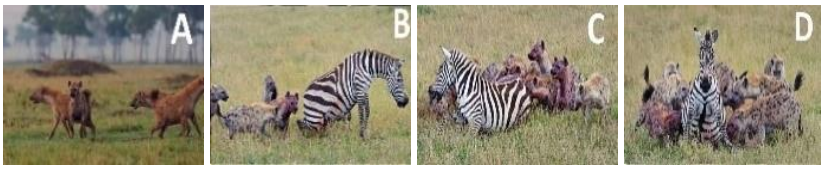
2. YÖNTEM

2.1 Benekli Sırtlan Optimizasyon (SHO) Algoritması

Benekli Sırtlan Optimizasyon Algoritması (SHO), Benekli sırtlanların grup halinde avlanma davranışlarından ilham almaktadır. Dhiman ve Kumar tarafından 2017 yılında yapılan bir çalışmada ortaya atıldığı bilinmektedir. SHO algoritması özellikle bilgisayar bilimlerindeki problemleri çözmek için bir grup benekli sırtlanın avlanma stratejilerinden ilham alan popülasyon tabanlı meta sezgisel bir yaklaşımdır (Ghafori ve Gharehchopogh, 2022).

2.1.1 SHO Algoritmasının Çalışma Mantığı

(Dhiman ve Kumar, 2017)'a göre SHO algoritması benekli sırtlanların avlanma davranışlarından ilham alınarak 4 temel adımda karşımıza çıkmaktadır.



Şekil 1. SHO algoritması adımları (Dhiman ve Kumar, 2017)

2.1.1.1 Avı Çevreleme

Bu aşamada avın hareketlerine göre kendi hareketlerini sürekli olarak güncelleyerek optimal bir şekilde avı çevrelemektedirler.

$$\vec{D}_m = \left| \vec{B}\, x\, \vec{P}_p(y) - \vec{P}(y) \right| \tag{1}$$

$$P(y+1) = \vec{P}_p(y) - \vec{E}\, x\, \vec{D}_m \tag{2}$$

Burada $\vec{D}_m$ av ile benekli sırtlan arasındaki mesafe vektörünü, **y** ise geçerli yinelemeyi vermektedir. $\vec{P}_p$ avın konum vektörünü temsil ederken $\vec{P}$, benekli sırtlanın konum vektörünü temsil etmektedir. $\vec{B}$ ve $\vec{E}$ ise katsayı vektörleridir.

$\vec{B}$ ve $\vec{E}$ vektörleri aşağıda verildiği gibi hesaplanmaktadır:

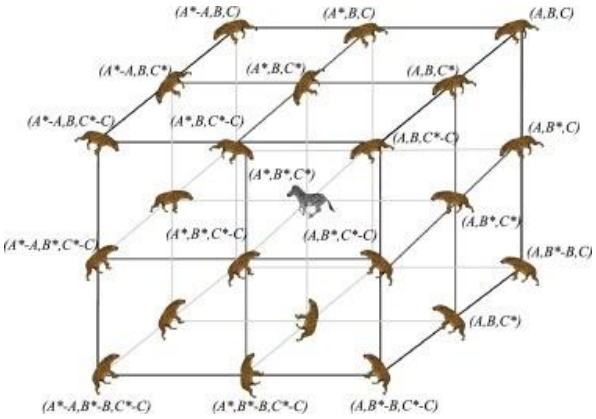
$$\vec{B} = 2\, x\, r\vec{d}_1 \tag{3}$$

$$\vec{E} = 2\ x\ \vec{m}\ x\ r\vec{d}_2 - \vec{m} \tag{4}$$

$$\vec{m} = 5 - \left(Iterasyon\ x\ \left(\frac{5}{Maksimum_{Iterasyon}} \right) \right) \tag{5.1}$$

$$Iterasyon = 0,1,2,3, \dots, Maksimum_{Iterasyon} \tag{5.2}$$

Şekil 2, benekli sırtlanların güncellenmiş konumunun 3 boyutlu ortamdaki gösterimidir.



Şekil 2. 3 boyutlu konum vektörleri ve benekli sırtlanların olası konumları (Dhiman ve Kumar, 2017)

2.1.1.2 Avlanma

Bu algoritmanın ikinci adımında avlanma için en iyi konumda olan benekli sırtlanın konumuna göre diğer sırtlanlar da konumlarını optimal bir şekilde güncellemektedirler.

$$\vec{D}_m = \left| \overrightarrow{B}\ x\ \vec{P}_p(y) - \vec{P}_k \right| \tag{6}$$

$$\vec{P}_k = \vec{P}_m - \vec{E}\ x\ \vec{D}_m \tag{7}$$

$$\vec{C}_m = \vec{P}_k + \vec{P}_{k+1} + \cdots + \vec{P}_{k+N} \tag{8}$$

Bu formüllerde $\vec{P}_m$, en iyi benekli sırtlanın konumunu temsil ederken $\vec{P}_k$ ise diğer benekli sırtlanların konumunu temsil etmektedir. N ise benekli sırtlan sayısını belirtmektedir. N aşağıdaki gibi hesaplanmaktadır.

$$N = Count_{ns}\left(\vec{P}_m, \vec{P}_{m+1}, \vec{P}_{m+2}, \dots, \left(\vec{P}_m + \vec{M} \right) \right) \tag{9}$$

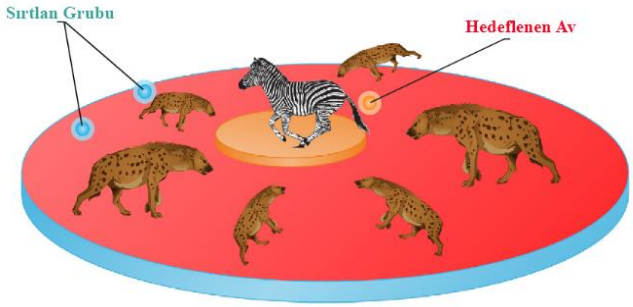
Burada, M 0,5 ve 1 arasında rastgele bir değişken olmaktadır. ns çözüm sayısını belirtirken $\vec{C}_m$'de N sayıdaki optimal çözümden oluşan grubu temsil etmektedir.

2.1.1.3 Ava Saldırma (İstismar)

Ava saldırma adımının gerçekleşmesi için $\vec{m}$ vektörünün değerinin düşürülmesi gerekmektedir. İterasyon sırasında 5'ten 0'a düşebilen $\vec{m}$ değerinin değişmesi sebebinden dolayı $\vec{E}$ vektöründeki varyasyon da azalmaktadır. Bu adımın matematiksel formülü aşağıdaki gibidir:

$$P(y+1) = \frac{\vec{C}_m}{N} \tag{10}$$

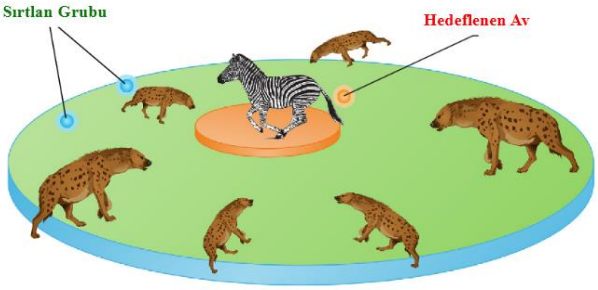
Burada, $\boldsymbol{P(y+1)}$ en iyi çözümü kaydederek diğer benekli sırtlanların konumlarını güncellemektedir.



Şekil 3. Ava Saldırma (İstismar) (Ranganathan ve Natarajan, 2022)

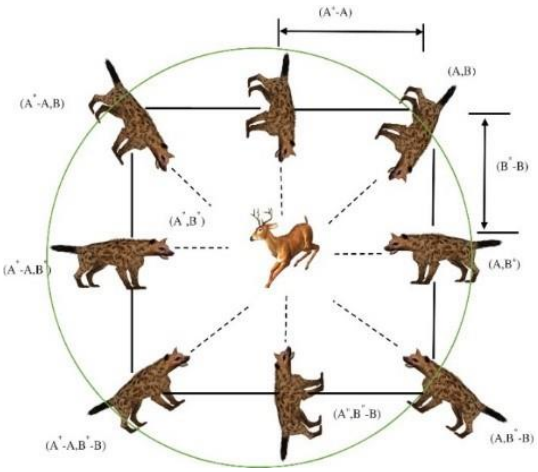
2.1.1.4 Av Arama (Keşif)

Arama mekanizması, bir algoritmanın keşif gücünü temsil etmektedir. SHO algoritması, uygun çözümü aramak için Denklem (4)' deki $\vec{E}$ vektörünü 1'den büyük veya -1' den küçük rastgele değerler ile kullanma kabiliyeti sağlamaktadır. Ayrıca $\overrightarrow{B}$ vektörü de SHO algoritmasının daha rastgele davranış göstermesinden ve ek olarak da yerel optimumdan kaçınmasından sorumlu olmaktadır.



Şekil 4. Av Arama (Keşif) (Ranganathan ve Natarajan, 2022)

SHO algoritmasını modellemek için her benekli sırtlan konumunun optimizasyon problemine bir çözüm olduğu ve en iyi benekli sırtlan konumunun avın konumunu tahmin ettiği varsayılmaktadır. Şekil 5'e göre benekli sırtlanlar yani problemin çözümleri, avı yani optimal çözümü daire içine alabilmektedir.



Şekil 5. Benekli sırtlan vektörlerinin 2B konumu (Dhiman ve Kumar, 2017)

Algoritma 1. SHO algoritmasının sözde kodu (Luo ve ark, 2019)

Input: Benekli Sırtlan Popülasyonu $\boldsymbol{P_i(i = 1, 2, 3, \ldots, n)}$

Output: Elde edilen iyi arama bireyi

SHO yöntemi başlat

B, E, m ve N parametrelerini başlat

Arama yapan her bir sırtlanın uygunluk değerini hesapla

$\vec{\boldsymbol{P}}_{\boldsymbol{m}}$ = ilk en iyi arama sırtlanı

$\vec{\boldsymbol{C}}_{\boldsymbol{m}}$ = elde edilen tüm optimal çözümlerin kümesi

while ($\boldsymbol{y < Maksimum_{Iterasyon}}$) **do**

for her bir arama sırtlanı için **do**

Denklem (10)' u kullanarak mevcut arama sırtlanının konumunu güncelle

end for

B, E, m ve N parametrelerini güncelle

Arama alanının dışarısına çıkan arama sırtlanlarını kontrol et ve değiştir

Her bir arama sırtlanının uygunluk değerini hesapla

Daha önce elde edilen optimal çözümden daha iyi bir çözüm varsa $\vec{\boldsymbol{P}}_{\boldsymbol{m}}$' yi güncelle

$\vec{\boldsymbol{C}}_{\boldsymbol{m}}$ kümesini $\vec{\boldsymbol{P}}_{\boldsymbol{m}}$' a göre güncelle

y = y + 1

end while

return $\vec{\boldsymbol{P}}_{\boldsymbol{m}}$

SHO yöntemini sonlandır

2.2 Kum Kedisi Sürü Optimizasyon Algoritması (SCSO)

Kum kedisi (Felis margarita), memeliler familyasından Felis hayvanlarının bir türüdür. Kum kedisi; Orta Asya Sahra, Afrika Sahra ve Arap Yarımadası gibi kumlu ve taşlı çöllerin zorlu ortamında yaşamaktadır. Küçük, çevik ve alçakgönüllü bir kum kedisi, avlanma ve yaşama konusunda farklı yaşam davranışlarına sahiptir. Kum kedisi boyut ve görünüş açısından her ne kadar ev kedisine benzese de yaşam davranışı olarak iki tür arasında büyük farklılıklar bulunmaktadır. Kum kedileri de pek çok kedigiller gibi grup halinde yaşamamaktadırlar. Kum kedisinin avuç içi ve ayak tabanları daha yoğun açık gri kürkle kaplıdır. Ayak tabanlarının kürk kaplaması, ayak pedlerini çölün sert sıcak ve soğuk havasına karşı yalıtmaktadır. Ayrıca kum kedisinin kürk özellikleri tespit ve takibini zorlaştırmaktadır. Kum kedilerindeki avlanma mekanizması oldukça ilginç ve farklıdır. Harika işitme duyularını kullanarak düşük frekanslı sesler almaktadırlar. Bu sayede kum kedileri yeraltında hareket eden avları (böcekler ve kemirgenler) ve bu avların varış zamanı farklarını da tespit edebilmektedirler. Kum kedisinin, avı yeraltındaysa hızla kazma gibi ilginç bir yeteneği de bulunmaktadır (Seyyedabbasi ve Kiani, 2023).

SCSO algoritması, doğadaki kum kedisi davranışlarından ilham almaktadır. Kum kedisinin iki ana eylemi avını aramak ve ava saldırmaktır. Bu algoritma kum kedisinin düşük frekanslı gürültüleri tespit edebilme yeteneğinden esinlenmektedir. Kum kedileri olağanüstü özelliklerinden yararlanarak avlarını yerde veya yer altında bulabilmektedirler (Seyyedabbasi ve Kiani, 2023).

Şekil 6. SCSO algoritması adımları (Seyyedabbasi ve Kiani, 2023)

2.2.1 Başlangıç Popülasyonu Oluşturma

D-boyutlu bir optimizasyon probleminde, kum kedisi popülasyonu büyüklüğü N olmaktadır. Her bir kum kedisinin çözümü $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ olarak temsil edilmektedir. İlk popülasyon, denklem 11 ile hesaplanmaktadır:

$$X_{i,j}(t) = lb_j + U(0,1) \times \left(ub_j - lb_j\right) \tag{11}$$

Burada, $X_{i,j}(t)$, X_i'nin güncel pozisyonunun j. boyutunu göstermektedir, t mevcut iterasyonu temsil etmektedir, U(0, 1), [0, 1] aralığında rastgele bir sayıdır. lb_j boyutun alt sınırını, ve ub_j boyutun üst sınırını göstermektedir.

2.2.2 Av Arama (Keşif)

Kum kedisinin duyarlılık aralığının 2 kHz ile 0 arasından başladığı varsayılmaktadır. Her kum kedisi, pozisyonunu en iyi aday pozisyonu (Pos_{bc}), mevcut pozisyonu (Pos_c) ve duyarlılık aralığı (r) temel alarak güncellemektedir. Denklem 12, 13 ve 14 av arama davranışını açıklar:

$$Pos\,(t+1) = r \cdot \left(Pos_{bc}(t) - rand\,(0,1) \cdot Pos_c(t)\right) \tag{12}$$

$$r = r_G \cdot rand\,(0,1) \tag{13}$$

$$r_G = S_M - \left(\frac{S_M \cdot t}{t_{max}}\right) \tag{14}$$

Burada, r_G geniş duyarlılık aralığını gösterir ve bu, 2'den 0'a doğru lineer olarak azaltılmaktadır. S_M değeri, kum kedilerinin işitme özelliklerinden esinlenilerek değeri varsayılan olan 2 olarak kabul edilmektedir. $rand(0,1)$, [0, 1] aralığında rastgele bir sayıdır ve t_{max} maksimum iterasyonları temsil etmektedir.

2.2.3 Ava Saldırma

Kum kedisinin duyarlılık aralığı bir daire olarak kabul edilmektedir. Bu şekilde hareket yönü daire üzerinde rastgele bir açı (θ) tarafından belirlenmektedir. SCSO, her kum kedisi için rastgele bir açı seçmek için rulet tekerleği seçim algoritmasından faydalanmaktadır. Denklem 15, 16 ve 17 avı ele geçirme davranışını açıklamaktadır:

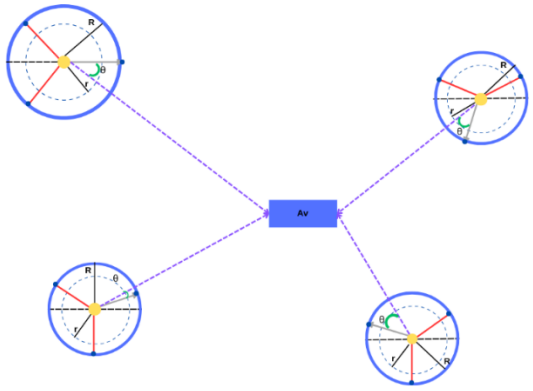
$$X\,(t+1) = \begin{cases} Pos_b(t) - Pos_{rnd} \cdot \cos(\theta) \cdot r & |R| \le 1 \\ r \cdot Pos_{bc}(t) - rand\,(0,1) \cdot Pos_c(t) & |R| > 1 \end{cases} \tag{15}$$

$$Pos_{rnd} = \left|rand(0,1) \cdot Pos_b(t) - Pos_c(t)\right| \tag{16}$$

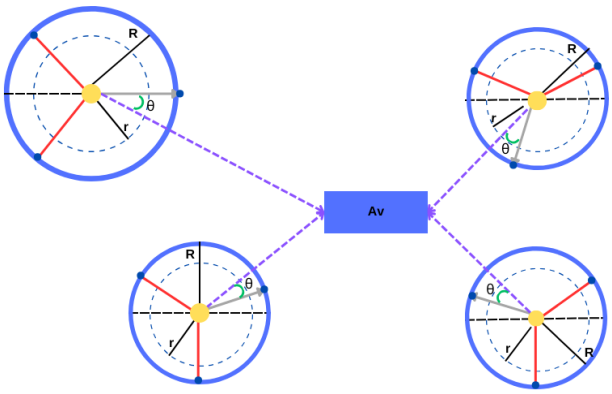
$$R = 2 \cdot r_G \cdot rand(0,1) - r_G \tag{17}$$

$$pos(t+1) = Pos_b(t) - r \times Pos_{rnd} \times \cos(\alpha) \tag{18}$$

Burada, Pos_b en iyi çözüm pozisyonunu, Pos_{rnd} rastgele pozisyonu göstermektedir. θ, 0 ile 360 arasında bir açı olup değeri -1 ile 1 arasındadır [26] (Chen & Zheng, 2023).



Şekil 7. konum güncelleme $İterasyon_i$ (Seyyedabbasi ve Kiani, 2023)



Şekil 8. konum güncelleme $İterasyon_{i+1}$ (Seyyedabbasi ve Kiani, 2023)

Algoritma 2. SCSO Algoritmasının sözde kodu (Wu ve ark, 2022)

Popülasyonu başlat

Amaç fonksiyonuna göre uygunluk fonksiyonunu hesapla

r, r_G, R başlat

While (t <= maksimum iterasyon)

  for her arama ajanı için

    Rulet çarkına göre rastgele bir açı elde et ($0° \leq \theta \leq 360°$)

    if (abs(R) > 1)

      Arama ajanı konumunu denklem 12'ye göre güncelle

    else Arama ajanı konumunu denklem 18'e göre güncelle

    end

  end

 t=t++

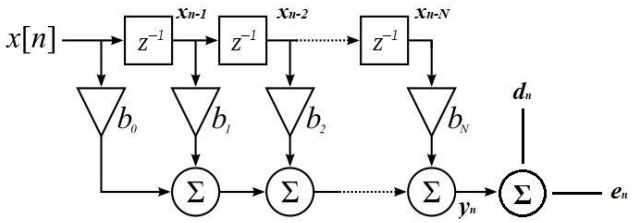
end

2.3 Adaptif Lineer Toplayıcı

Adaptif lineer toplayıcılar, otomatik olarak uyarlanabilen dijital filtrelerdir. Bu filtreler giriş sinyaline göre değişebilmektedir. Adaptif lineer toplayıcılar, farklı sinyal koşullarına yanıt olarak filtre özelliklerinin değişmesini gerektiren uygulamalarda kullanılmaktadır (Oppenheim ve ark, 1997).

Adaptif lineer toplayıcılarda işaret ve referans olmak üzere iki adet giriş gerekmektedir. Adaptif bir süzgeç, kendi katsayılarını ve ağırlıklarını güncelleme yeteneğine sahiptir. Bu yeni katsayılar, katsayı üreteci tarafından süzgece gönderilmektedir. Katsayı üreteci, adaptif bir algoritma kullanarak yeni katsayılar oluşturmaktadır. Bu katsayılar bir süzgeçten geçirilerek işlenmektedir. Süzgeç, referans giriş sinyaline uyum sağlamak için katsayıları sürekli olarak ayarlayan adaptif bir süzgeçtir. Katsayı üretecinin temel amacı, referans girişine uygun süzgeç katsayılarını üretmektir. Bu sebeple adaptif süzgeç, işaretten referans girişini çıkarabilmektedir. Referans giriş işareti değiştikçe katsayılar da güncellenmektedir. Bu sayede adaptif süzgeç, bu değişikliklere uyum sağlamaktadır. Bu çalışma prensibi nedeniyle adaptif süzgeç olarak adlandırılmaktadır (Karaboğa ve Koyuncu, 2005). Adaptif süzgeçler, sinyallerin ve görüntülerin

hem gürültüsünü azaltmak hem de istenilen frekansları geliştirmek veya bastırmak için kullanılabilmektedir (Karakaş ve Latifoğlu, 2021).



Şekil 9. Tek girişli adaptif lineer toplayıcı (Kumar N. , 2013)

Şekil 9'da Sonlu Darbe Yanıtlı (FIR) tek girişli bir adaptif lineer toplayıcının tasarımı gösterilmektedir. FIR süzgeç N+1 uzunluğunda olup katsayıları da b ile ifade edilmektedir. b katsayısı zamanla değişebilir ve ayrıca çevreye bağımlı olabilmektedir. Şekil 9'daki $x[n]$ adaptif FIR süzgecin giriş işaretini, y_n çıkış işaretini, d_n arzu edilen işareti ve e_n hata işaretini vermektedir.

2.4 Uygunluk Fonksiyonu

Adaptif lineer toplayıcı tasarımında uygunluk fonksiyonu, hata fonksiyonlarından tercih edilmektedir. Bunun sebebi tasarlanan filtrenin frekans cevabının önceden belirlenmiş ideal bir frekans cevabına mümkün olduğunca yakın olmasının amaçlanmasıdır. Bu ideal frekans cevabı, tasarlanacak filtrenin belirli bir işlevi yerine getirmesi için gerekli olan frekans cevabı olmalıdır. Bundan dolayı uygunluk fonksiyonu, tasarlanan filtrenin frekans cevabı ile ideal frekans cevabı arasındaki farkın ölçüsünü hesaplamaktadır. Bu farkın mümkün olduğunca küçük olması hedeflenmektedir. Bu nedenle uygunluk fonksiyonu hata fonksiyonu olarak belirlenmektedir (Saha ve ark, 2013). Adaptif Lineer Toplayıcı tasarımında sıkça kullanılan hata fonksiyonlarının tanımları aşağıda verilmektedir:

2.4.1 Ortalama Karesel Hata (MSE)

MSE fonksiyonu, arzu edilen işaretten tasarlanan süzgecin çıkış işaretinin çıkarılması daha sonra da karelerinin alınması ve ortalamasının kullanılmasını amaçlamaktadır.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}[\,\mathrm{d(n)} - \mathrm{y(n)}]^2 \tag{18}$$

2.4.2 En Küçük Karesel Hata (LMS)

LMS fonksiyonu, süzgeç çıkışının arzu edilen işaretten çıkarılması ile elde edilen hataların kareleri alınarak toplam değerinin bulunması ve bulunan bu toplam değerinin karekökü alınarak minimize edilmesini amaçlamaktadır.

$$LMS = \frac{1}{N}\sum_{n=1}^{N}[\mathrm{d(n)} - \mathrm{y(n)}^2]^{\frac{1}{2}} \tag{19}$$

2.4.3 Ortalama Mutlak Hata (MAE)

MAE fonksiyonu, süzgeç çıkışının arzu edilen işaretten çıkarılması ile mutlak değerler toplamının minimum yapılmasını amaçlamaktadır.

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\mathrm{d(n)} - \mathrm{y(n)}| \tag{20}$$

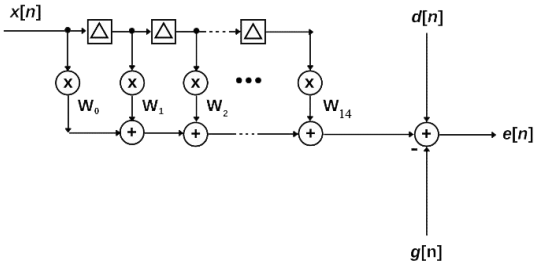
3. ARAŞTIRMA BULGULARI

Bu çalışmada SHO ve SCSO algoritmaları ile FIR süzgecin uygunluk fonksiyonunu iyileştirmek için 3 farklı hata fonksiyonu kullanılmaktadır. Bu fonksiyonlar MSE, LMS ve MAE fonksiyonlarıdır. Şekil 10'da tek girişli 15 ağırlıklı bir Adaptif Lineer Toplayıcı gösterilmektedir. Arzu edilen işaret, giriş işaretiyle birlikte, her biri birim zamanda olmak üzere 5 örnek ile örneklendirilmektedir. Bu çalışmada faydalanılan işaretler, denklem 21, 22 ve 23'de gösterilmektedir.

$$\mathrm{d(n)} = 2\cos\left(\frac{2\pi n}{5}\right) \quad n = 1,2,3,4,5,\dots 50 \tag{21}$$

$$\mathrm{x(n)} = \sin\left(\frac{2\pi n}{5}\right) \tag{22}$$

$$d1 = d(n) + g(n) \tag{23}$$



Şekil 10. 15 ağırlıklı Adaptif Lineer Toplayıcı (Karaboğa ve Koyuncu, 2005)

Şekil 10' da verilen 15 ağırlıklı Adaptif Lineer Toplayıcının x[n] giriş işaretini, y[n] izlenecek gerçek işareti, d[n] arzu edilen çıkış işaretini, g[n] normal dağılımlı olan rastgele bir gürültü işaretini temsil etmektedir. d1[n] ise 10 dB SNR değerine sahip bir işareti temsil etmektedir. Bu çalışmada MSE, LMS ve MAE fonksiyonları kullanılarak d1[n] arzu edilen çıkış işareti olarak kullanılıp y[n] ve d1[n] işaretleri sıfıra sürülmektedir. SHO ve SCSO algoritmalarının parametreleri Tablo 1 ve Tablo 2' de verilmektedir.

Tablo 1. SHO Algoritmasının Parametreleri

Popülasyon Büyüklüğü	120
Hareket Etme Yönü ve Büyüklüğü	5
Aday Konum Sayısı	10
İterasyon Sayısı	2000

Tablo 2. SCSO Algoritmasının Parametreleri

Popülasyon Büyüklüğü	50
İterasyon Sayısı	2000

Tablo 1 ve 2'de FIR süzgeç tasarımı için kullanılan SHO ve SCSO algoritmalarının çıkış ile arzu edilen işaret arasındaki hatayı minimuma düşüren parametreleri belirtilmektedir. Adaptif Lineer Toplayıcı tasarımı için MSE, LMS ve MAE fonksiyonları için ulaşılan ağırlıklar Tablo 3'de verilmektedir.

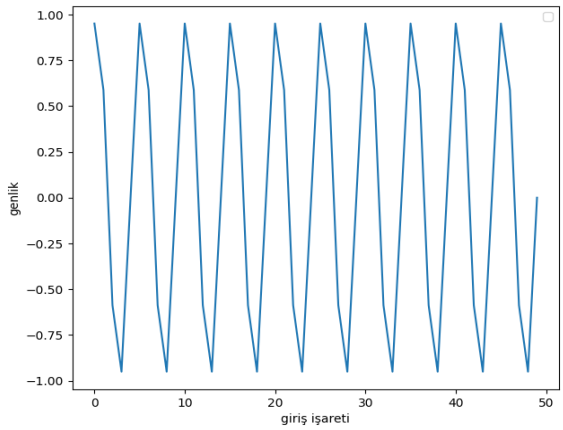
Tablo 3. SHO ve SCSO algoritmaları kullanılarak MSE, LMS ve MAE fonksiyonları ile Adaptif Lineer Toplayıcı için ulaşılan ağırlık değerleri

Ağırlıklar	**Elde Edilen Ağırlıklar**					
	SHO	***SCSO***	*SHO*	***SCSO***	*SHO*	***SCSO***
	MSE	**MSE**	LMS	**LMS**	MAE	**MAE**
w_0	-1.5083	**0.5720**	1.0616	**-0.0003**	0.5494	**0.5922**
w_1	-0.0920	**-1.9501**	-0.9344	**-1.7472**	-0.7014	**-1.9999**
w_2	-1.4659	**-0.0075**	1.6926	**-0.4057**	0.0764	**0.1161**
w_3	-1.5083	**0.4335**	1.8584	**0.3264**	-0.6479	**-0.0118**
w_4	0.2888	**-0.0378**	0.4704	**-0.2665**	0.1345	**0.1041**
w_5	-0.4941	**0.2805**	1.1644	**0.6503**	-0.7247	**-0.0037**
w_6	-0.6374	**0.6584**	-1.4742	**-0.0029**	-0.5472	**0.2393**
w_7	1.4312	**-0.3499**	1.6877	**0.0141**	-0.7247	**0.0137**
w_8	1.7278	**0.4627**	-1.2835	**-0.0027**	0.2370	**0.0395**
w_9	0.3595	**0.0798**	-0.8850	**-0.0002**	0.3040	**-0.0070**
w_{10}	1.2119	**-0.2286**	-0.4487	**0.0289**	-0.7247	**0.0029**
w_{11}	0.5472	**0.0171**	-0.0253	**-0.0728**	0.7247	**-0.3724**
w_{12}	-1.3190	**-0.0762**	1.0282	**0.0003**	0.1294	**-0.0087**
w_{13}	-0.4816	**-0.2845**	-0.7313	**0.0007**	-0.7247	**-0.2796**
w_{14}	-0.6942	**-0.0001**	0.0649	**-0.0289**	-0.4257	**0.0062**

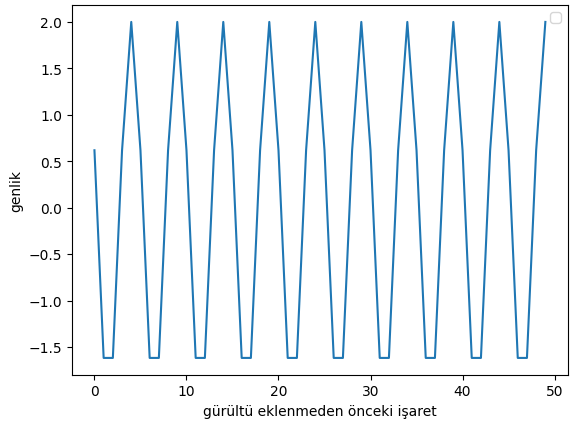
Tablo 4. Kullanılan Hata fonksiyonlarının SHO ve SCSO performans değerleri

	SHO	**SCSO**
Ortalama MSE	0.4248	**0.0711**
Ortalama LMS	0.7366	**0.2757**
Ortalama MAE	0.4254	**0.2139**
MSE Standart Sapması	0.0898	**0.0012**
LMS Standart Sapması	0.0858	**0.0102**
MAE Standart Sapması	0.0564	**0.0017**

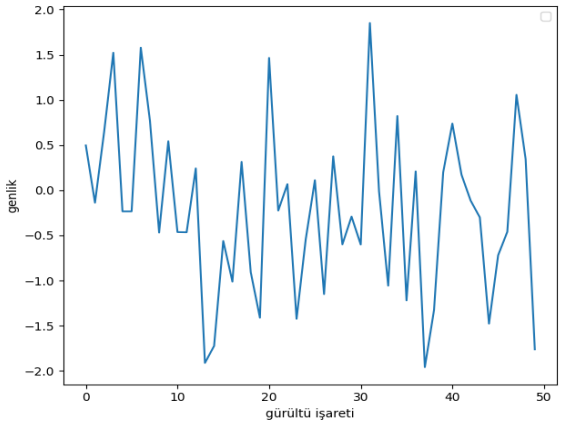
Her iki algoritma da beşer kez çalıştırılarak optimizasyon işlemi yapılmıştır. SHO algoritmasının ortalama MSE değeri 0.4248, ortalama LMS değeri 0.7366, ortalama MAE değeri ise 0.4254 olarak hesaplanmıştır. Ayrıca bu fonksiyonların standart sapmaları ise MSE için 0.0898, LMS için 0.0858 ve MAE için 0.0564'tür. Aynı hesaplamalar SCSO algoritması için de ortalama MSE 0.0711, ortalama LMS 0.2757 ve ortalama MAE 0.2139 olarak hesaplanmıştır. Standart sapma değerleri ise MSE için 0.0012, LMS için 0.0102 ve MAE için 0.0017'dir. Bu değerler Tablo 4'te açık bir şekilde belirtilmektedir.



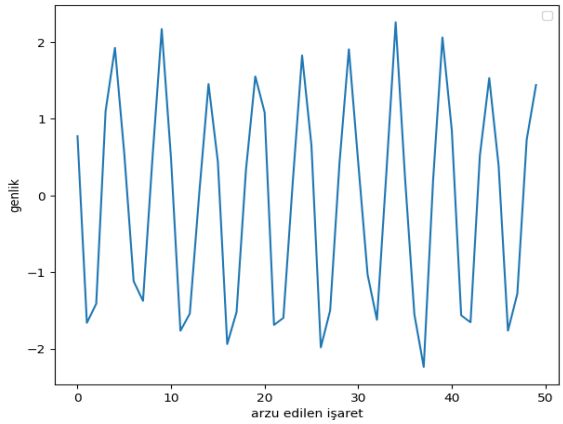
Şekil 11. x(n) giriş işareti



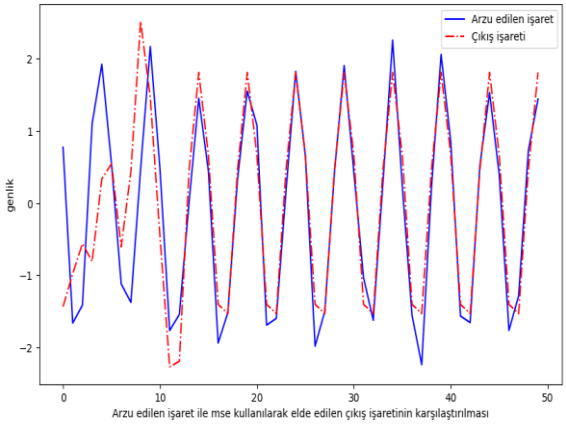
Şekil 12. d(n) gürültü eklenmeden önceki işaret



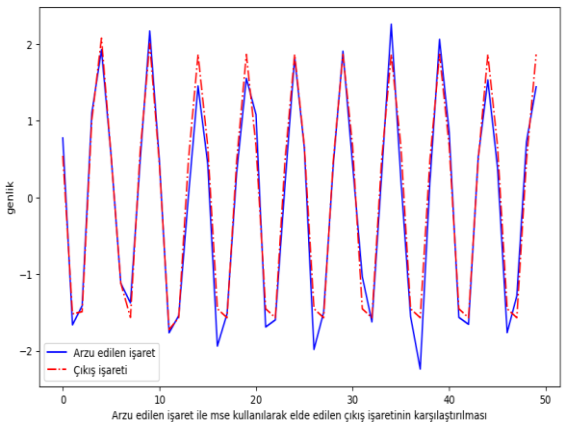
Şekil 13. g(n) Gürültü işareti



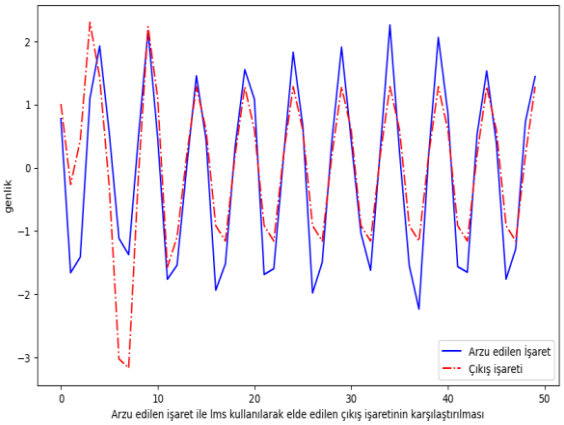
Şekil 14. d1(n) arzu edilen işaret



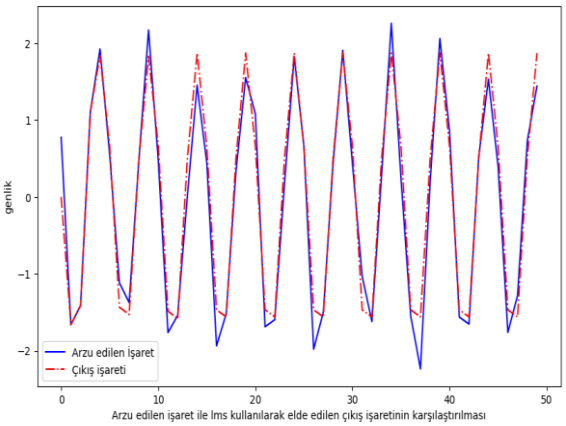
Şekil 15. SHO kullanılarak d1(n) arzu edilen işaret ile MSE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



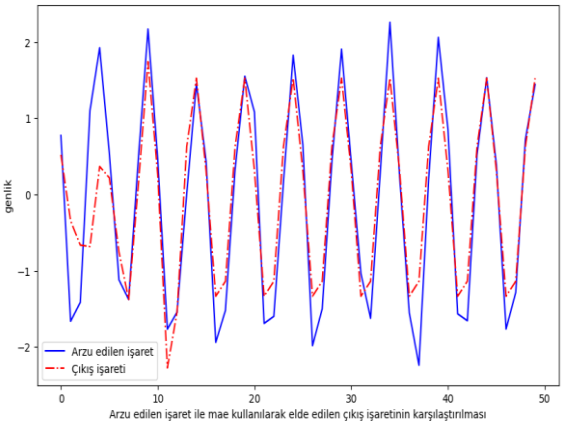
Şekil 16. SCSO kullanılarak d1(n) arzu edilen işaret ile MSE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



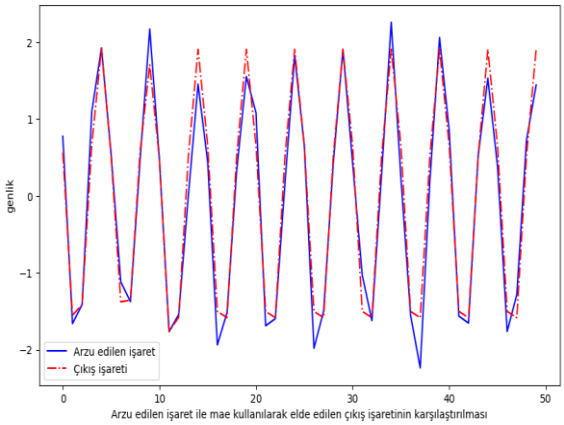
Şekil 17. SHO kullanılarak d1(n) arzu edilen işaret ile LMS kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



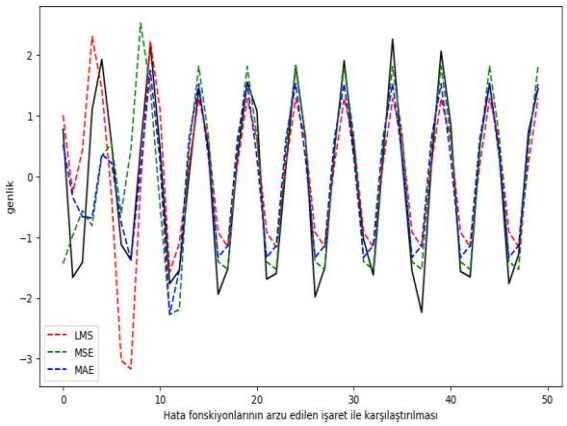
Şekil 18. SCSO kullanılarak d1(n) arzu edilen işaret ile LMS kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



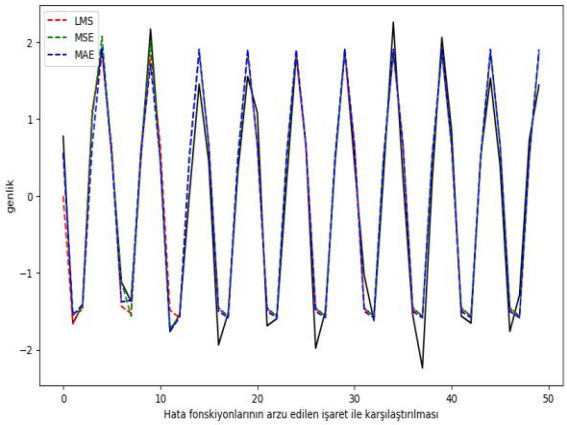
Şekil 19. SHO kullanılarak d1(n) arzu edilen işaret ile MAE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



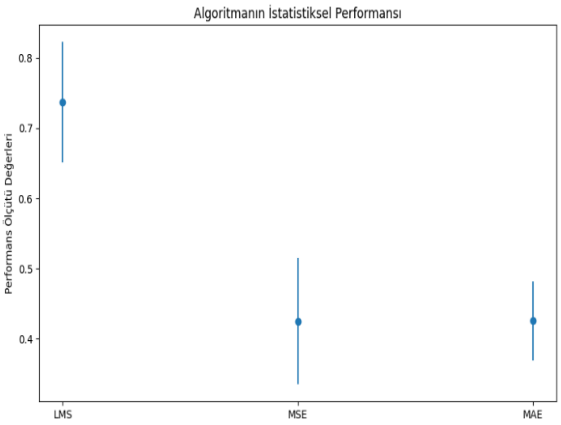
Şekil 20. SCSO kullanılarak d1(n) arzu edilen işaret ile MAE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



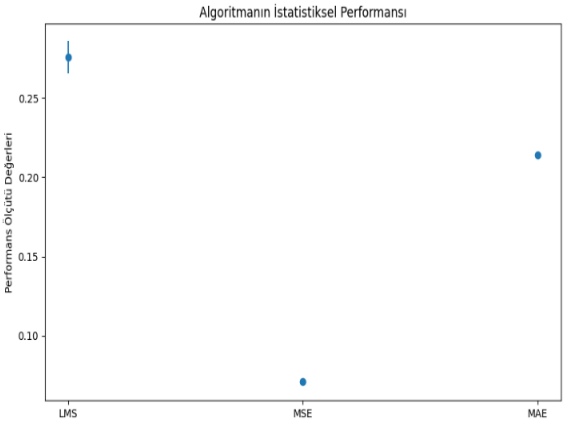
Şekil 21. SHO kullanılarak d1(n) arzu edilen işaret ile MSE, LMS ve MAE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



Şekil 22. SCSO kullanılarak d1(n) arzu edilen işaret ile MSE, LMS ve MAE kullanılarak elde edilen çıkış işaretlerinin karşılaştırılması



Şekil 23. SHO Algoritmasının istatiksel performansı



Şekil 24. SCSO Algoritmasının istatistiksel performansı

Şekil 11-14'te görüldüğü üzere giriş işareti bir sinüs işareti olup arzu edilen işaret de normal dağılımlı rastgele bir gürültü işareti ile bir kosinüs işaretinin birleşimi olmaktadır. Şekil 15-20'de iki algoritmanın MSE, LMS ve MAE fonksiyonları ile Adaptif Lineer Toplayıcı çıkışında elde edilen işaretler, d1(n) arzu edilen işaret ile karşılaştırılmaktadır. Bu fonksiyonların zaman içinde birbirlerine yaklaştığı gözlemlenmektedir. Bu sonuçla birlikte bu üç hata fonksiyonunun benzer bir optimize performans sergiledikleri belirlenmektedir. Şekil 21 ve 22'de ise MSE, LMS ve MAE fonksiyonlarından elde edilen çıkış işaretlerinin hepsi d1(n) ile karşılaştırılmaktadır. Son olarak Şekil 23 ve 24'te de iki algoritmanın istatistiksel performanslarının ölçülmesi için 5 farklı çalıştırmadan elde edilen yakınsama grafikleri gösterilmektedir.

4. SONUÇ

Bu çalışmada Adaptif Lineer Toplayıcı tasarımında SHO ve SCSO algoritmalarının performansları farklı hata fonksiyonları (MSE, LMS, MAE) kullanılarak incelenmiştir. Yapılan simülasyon çalışmaları ve elde edilen sonuçlar ile birlikte her iki algoritmanın farklı hata fonksiyonları altında nasıl performans gösterdiği değerlendirilmiştir.

SHO algoritması kullanılarak yapılan simülasyonlarda, MSE, LMS ve MAE hata fonksiyonları ile tasarlanan Adaptif Lineer Toplayıcıların performansları incelenmiştir. Sonuçlar, SHO algoritmasının bu hata fonksiyonları ile yeterince iyi performans gösteremediğini ortaya koymuştur:

- MSE Değeri: 0.5083
- LMS Değeri: 0.7153
- MAE Değeri: 0.4168

SHO algoritmasının özellikle gürültü içeren ortamlarda adaptif olarak işaret yakalama konusunda zorlandığı gözlemlenmiştir. MSE, LMS ve MAE hata fonksiyonları ayrı olarak kullanıldığında arzu edilen işareti etkin bir şekilde yakalayamayan tasarımlar ortaya çıkmıştır.

SCSO algoritması ise, SHO algoritmasına kıyasla çok daha optimal sonuçlar vermiştir. SCSO algoritması kullanılarak yapılan simülasyonlarda elde edilen hata değerleri şu şekildedir:

- MSE Değeri: 0.0695
- LMS Değeri: 0.2924
- MAE Değeri: 0.2151

Bu sonuçlar, SCSO algoritmasının adaptif lineer toplayıcı tasarımında SHO algoritmasına kıyasla daha başarılı olduğunu göstermektedir. Özellikle MSE hata fonksiyonu kullanıldığında, SCSO algoritmasının en iyi performansı sergilediği görülmüştür.

SHO ve SCSO algoritmalarının karşılaştırılması sonucunda SCSO algoritmasının daha optimize sonuçlar verdiği ve adaptif lineer toplayıcı tasarımı için daha uygun olduğu belirlenmiştir. 5 yıl aralıklarla ortaya atılmış bu sürü tabanlı optimizasyon algoritmalarının performanslarının kıyaslanması, SCSO algoritmasının daha modern ve gelişmiş bir yapıya sahip olduğunu ve bu nedenle SHO algoritmasına göre daha iyi performans gösterdiğini ortaya koymaktadır. Bu çalışmanın bulguları doğrultusunda, adaptif lineer toplayıcı tasarımında:

SCSO Algoritması: Bu çalışmaya göre adaptif lineer toplayıcı tasarımı için önerilmektedir. Daha düşük hata oranları ve optimal performans ile güvenilir ve etkin bir tasarım sunmaktadır.

SHO Algoritması: Adaptif lineer toplayıcı tasarımı için önerilmemektedir. Performansının düşük, hata oranlarının ise yüksek olması sebebi ile istenilen işareti yakalamakta zorlanmaktadır.

Sonuç olarak bu çalışma, sürü tabanlı optimizasyon algoritmalarının adaptif lineer toplayıcı tasarımındaki etkinliklerini karşılaştırarak SCSO algoritmasının üstünlüğünü ve güvenilirliğini ortaya koymuştur. Gelecekteki çalışmalarda farklı hata fonksiyonlarının kombinasyonlarının incelenmesi ve bu algoritmaların daha kompleks sistemlerde uygulanması optimizasyon performanslarını daha da geliştirebilir.

5. KAYNAKÇA

Ababneh, J., & Bataineh, M. (2008). Linear phase FIR filter design using particle swarm optimization and genetic algorithms. *Digital Signal Processing*, 657-668.

Aslan, C., Seyyarer, E., & Uçkan, T. (2023). Honey Badger Optimizasyon Algoritması ile Üç Elemanlı Kafes Sisteminin Ağırlık ve Maliyet Minimizasyonu. *Çukurova Üniversitesi Mühendislik Fakültesi Dergisi*, 441-449.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 268-308.

Boudjelaba, K., Ros, F., & Chikouche, D. (2014). Potential of particle swarm optimization and genetic algorithms for FIR filter design. *Circuits, Systems, and Signal Processing*, 3195-3222.

Brown M., A. P., J., H. C., & H., W. (1993). How Biased is Your Multi-Layered Perceptron?. *World Congress on Neural Networks*, (s. 507-511). San Diego.

Chen, S., & Zheng, J. (2023). Sand cat arithmetic optimization algorithm for global optimization engineering design problems. *Journal of Computational Design and Engineering*, 2122-2146.

Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 48-70.

Dwivedi, A., Ghosh, S., & Londhe, N. (2018). Review and analysis of evolutionary optimization-based techniques for FIR filter design. *Circuits, Systems, and Signal Processing*, 4409-4430.

Ghafori, S., & Gharehchopogh, F. (2022). Advances in spotted hyena optimizer: a comprehensive survey. *Archives of computational methods in engineering*, 1569-1590.

Karaboğa, N., & C. A. Koyuncu. (2005). Diferansiyel Gelişim Algoritması Kullanılarak Adaptif Lineer Toplayıcı Tasarımı. *III. Otamasyon Sempozyumu*, (s. 216-220). Denizli.

Karaboğa, N., & Koyuncu, C. A. (2005). Diferansiyel Gelişim Algoritması Kullanarak Sinyal Kestirimine Yönelik Adaptif SDY Süzgeç Tasarımı. *II. İletişim Teknolojileri Ulusal Sempozyumu-İTUSEM*, (s. 91-95). Adana.

Karakaş, M., & Latifoğlu, F. (2021). Optimizasyon Tabanlı FIR Süzgeç Tasarımlarında Performans Analizi. *Avrupa Bilim ve Teknoloji Dergisi*, 8-22.

Kaur, H., Saini, S., & Raut, Y. (2023). The Application of Hybrid Optimization For FIR Filter Design. *In 2023 1st International Conference on Innovations in High Speed Communication and Signal Processing (IHCSP)* (s. 386-391). IEEE.

Kumar, N. (2013). Optimal design of fir and iir filters using some evolutionary algorithms. *Master of Technology Thesis in Electrical Engineering, National Institute of Technology.* Durgapur.

Kumar, V., Chhabra, J., & Kumar, D. (2014). Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. *Journal of Computational Science*, 144-155.

Luo, Q., Li, J., & Zhou, Y. (2019). Spotted hyena optimizer with lateral inhibition for image matching. *Multimedia Tools and Applications*, 34277-34296.

Majhi, R., Panda, G., & Majhi, B. (2009). Robust prediction of stock indices using PSO based adaptive linear combiner. *In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)* (s. 312-317). Coimbatore: IEEE.

Najjarzadeh, M., & Ayatollahi, A. (2008). FIR digital filters design: particle swarm optimization utilizing LMS and minimax strategies. *In 2008 IEEE International Symposium on Signal Processing and Information Technology* (s. 129-132). IEEE.

Oppenheim, A., Buck, J., Daniel, M., Willsky, A., Nawab, S., & Singer, A. (1997). *Signals & systems.* Upper Saddle River, New Jersey, USA.

Ranganathan, E., & Natarajan, R. (2022). Spotted hyena optimization method for harvesting maximum PV power under uniform and partial-shade conditions. *Energies*, 2850.

Reddy, K., & Sahoo, S. (2015). An approach for FIR filter coefficient optimization using differential evolution algorithm. *AEU-International Journal of Electronics and Communications*, 101-108.

Saha, S., Ghoshal, S., Kar, R., & Mandal, D. (2013). Cat swarm optimization algorithm for optimal linear phase FIR filter design. *ISA transactions*, 781-794.

Seifossadat, S., Razzaz, M., Moghaddasian, M., & Monadi, M. (2007). Harmonic estimation in power systems using adaptive perceptrons based on a genetic algorithm. *WSEAS Transactions On Power Systems*, 239-244.

Seyyedabbasi, A., & Kiani, F. (2023). Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. *Engineering with Computers*, 2627-2651.

Shao, P., Wu, Z., Zhou, X., & Tran, D. (2017). FIR digital filter design using improved particle swarm optimization based on refraction principle. *Soft Computing*, 2631-2642.

Singh, S., Singh, G., Bose, S., & Shiva. (2022). Fir filter design using grasshopper optimization algorithm. *In Recent Advances in Metrology: Select Proceedings of AdMet 2021* (s. 249-257). Singapore: Springer Nature Singapore.

Srivastava, S., Dwivedi, A., & Nagaria, D. (2020). Low complexity FIR filter design using biogeography optimization algorithm and its improved version. *In 2020 IEEE Students Conference on Engineering & Systems (SCES)* (s. 1-5). IEEE.

Wu, D., Rao, H., Wen, C., Jia, H., Liu, Q., & Abualigah, L. (2022). Modified sand cat swarm optimization algorithm for solving constrained engineering optimization problems. *Mathematics*, 4350.

Yadav, S., Yadav, R., Kumar, A., & Kumar, M. (2021). A novel approach for optimal design of digital FIR filter using grasshopper optimization algorithm. *ISA transactions*, 196-206.