GU J Sci, Part C, 6(4): 909-918 (2018)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

http://dergipark.gov.tr/gujsc

Çadır Haritası Tabanlı Kaotik Optimizasyon Yöntemi

Türker TUNCER[1,*]

[1]*Fırat Üniversitesi, Teknoloji Fakültesi, Adli Bilişim Mühendisliği Bölümü, 23119, Merkez/ELAZIĞ*

Makale Bilgisi

Başvuru: 06/03/2018
Kabul: 14/10/2018

Anahtar Kelimeler

Kaotik optimizasyon
Çadır harita
Sayısal fonksiyonlar
Sürü optimizasyonu

Öz

Gerçek hayatta bazı problemler matematiksel yöntemler kullanılarak çözülememektedir Bu problemleri çözmek için genellikle meta sezgisel optimizasyon yöntemleri kullanılmaktadır. Meta-sezgisel optimizasyon yöntemlerinin başarımlarını arttırmak için kullanılan çözümlerden biriside kaotik haritaların kullanılmasıdır. Kaos doğrusal olmayan yöntemlerin fenomeni olarak literatürdeki yerini almaktadır. Bu makalede sıklıkla kullanılan kaotik haritalardan biri olan çadır (tent) harita kullanılarak yeni bir kaotik optimizasyon yöntemi önerilmiştir. Önerilen yöntem parçacık tabanlı bir yöntemdir ve bu yöntem rastgele parçacık oluşturma, en iyi değeri hesaplama, parçacık güncelleme ve en iyi değeri güncelleme aşamalarından oluşmaktadır. Çadır harita parçacık güncelleme aşamasında kullanılmaktadır. Önerilen yöntemim performansını test etmek için literatürde sıklıkla kullanılan 12 adet sayısal karşılaştırma fonksiyonu kullanılmıştır ve elde edilen sonuçlar literatürde daha önce önerilmiş ve yaygın olarak kullanılan optimizasyon yöntemleriyle karşılaştırılmıştır. Deneysel sonuçlar ve karşılaştırmalar, önerilen yöntemin başarılı bir optimizasyon yöntemi olduğunu göstermektedir.

Keywords

Chaotic optimization
Tent map
Numerical functions
Swarm optimization

Tent Map based Optimization Method

Abstract

In the real life, some problems cannot be solved using mathematical methods. Meta-heuristic optimization methods are usually used to solve these problems. One of the solutions used to increase the performance of meta-heuristic optimization methods is the use of chaotic maps. Chaos is a phenomenon of nonlinear methods. In this article, a new chaotic optimization method is proposed by using a tent map which is one of the frequently used chaotic maps. The proposed method is a particle-based method, which consists of random particle creation, best-fit calculation, particle update, and best-value update. The tent map is used during the particle update phase. In order to test the performance of the proposed method, 12 numerical comparison functions, which are frequently used in the literature, were used and the results obtained were compared with previously proposed and widely used optimization methods in the literature. Experimental results and comparisons show that the proposed method is a successful optimization method.

1. GİRİŞ (INTRODUCTION)

Yaşadığımız çağın adı bilgi çağıdır. Bu çağda gerçek hayat problemlerini çözmek için bilgi teknolojileri sıklıkla kullanılmaktadır. Gerçek hayat problemlerinin bazılarının matematiksel çözümü bulunmamaktadır. Bu tür problemleri çözmek için genellikle meta-sezgisel optimizasyon yöntemleri kullanılmaktadır ve meta-sezgisel optimizasyon son 20 yılda sıcak başlıklı bir araştırma konusu haline gelmiştir. Meta-sezgisel optimizasyon tekniklerinin temel amacı; sonlu bir arama uzayında en iyi sonuçları kullanarak arama uzayını daraltmak ve bu daraltılmış arama uzayında global en iyi çözümü elde etmektir. Bu yöntemler arama uzayını daraltmak için ise amaç fonksiyonunu her döngüde çağırmaktadırlar. Meta-sezgisel yöntemler genellikle evrimsel, doğa tabanlı ve matematiksel yöntemler kullanmaktadır [1,2]. Bunlardan birkaçı şu şekilde özetlenebilir. Genetik algoritma literatürde yaygın olarak kullanılan optimizasyon yöntemlerinden birisidir. Bu yöntemin temel amacı Darwin'in evrim kuramını modelleyerek optimum çözümü elde etmektir. Bu yöntem evrimsel optimizasyon yöntemlerinin en önemlilerinden birisidir ve başlangıç popülasyonu oluşturma, çaprazlama, mutasyon ve elitist seçim aşamalarından oluşmaktadır [3]. Doğa tabanlı optimizasyon yöntemleri Allah'ın yaratma sanatını

*İletişim yazarı, e-mail: turkertuncer@firat.edu.tr

DOI: 10.29109/gujsc.402410

modellemeye çalışan yöntemlerdir. Bu yöntemlerde böceklerin, kuşların, balıkların, ağaçların vb. canlıların yiyeceğe veya suya ulaşma davranışları modellenmektedir. Bu yöntemlerden en çok bilineni parçacık sürü optimizasyonu (PSO)' dur [4]. PSO'da balık ve kuş sürülerinin hareketleri modellenmiştir. PSO en sık kullanılan ve en çok varyasyonu olan parçacık tabanlı optimizasyon yöntemlerinden birisidir. PSO'nun önerilmesinin ardından literatürde birçok parçacık tabanlı optimizasyon yöntemi önerilmiştir. Bu yöntemler genel olarak, başlangıç popülasyonunu rastgele oluşturma, en iyi parçacığı seçme, hız hesaplama (Yöntemlerin farklılıkları burada oluşmaktadır [5]. Her bir yöntemin kendisine ait hız hesaplama denklemi bulunmaktadır.), parçacıkları güncelleme, en iyi çözümü güncelleme gibi aşamalardan oluşmaktadır. Matematik temelli optimizasyon yöntemleri ise bilinen bir matematiksel fonksiyonu kullanarak parçacık güncelleme işlemini yapan yöntemlerdir. Sinüs-kosinüs optimizasyon yöntemi bunlardan birisidir ve parçacık güncellemek için sinüs ve kosinüs fonksiyonlarının yanı sıra 4 adet rastgele değişken kullanmaktadır. Her bir iterasyonda parçacıklar ve bu rastgele değişkenler güncellenmektedir [6,7].

Hemen hemen tüm optimizasyon yöntemlerinde rastgele sayı üreteçleri kullanılmaktadır. Rastgele sayı üreteçlerinin kullanılmasının temel nedeni parametrelerin ne olduğunun bilinmemesidir. Bu sebeple başlangıç parametreleri rastgele atanmakta ve her bir döngüde bu parametreler güncellenerek problemi çözüme ulaştıracak parametreler elde edilmeye çalışılmaktadır. Optimizasyon yöntemlerinin sıklıkla kullandığı rastgele sayı üreteçlerinden birisi de kaotik haritalardır. Kaotik haritalar istatistiki özellikleri çok iyi olan rastgele sayı üreteçleridir. Kaotik haritaların bu tür özelliklerini kullanmak için kaos tabanlı birçok optimizasyon yöntemi önerilmiştir [8-11]. Bu yöntemlerde kaotik haritalar rastgele sayı üretimi veya dinamik ağırlık üretimi için kullanılmıştır ve kaotik haritaların iyi sonuç üretmede etkili olduğu gösterilmiştir.

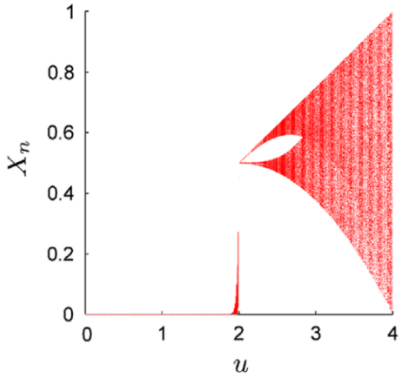
Bu makalede yaygın olarak kullanılan kaotik haritalardan çadır harita kullanılarak yeni bir kaotik sürü optimizasyon yöntemi önerilmiştir. Önerilen yöntem kaotik haritaları kullandığı için matematiksel temelli bir optimizasyon yöntemidir. Bu yöntemin literatüre karakteristiği aşağıdaki gibidir.

- Bu yöntemde çadır harita rastgele sayı üretme veya ağırlık üretmek için değil, parçacık güncellemek için kullanılmıştır.
- Önerilen yöntem basit matematiksel temellere dayanan, kolay uygulanabilir ve programlanabilir bir yöntemdir.
- Bu çalışmada çadır harita kullanan meta-sezgisel parçacık tabanlı bir arama algoritması önerilmiştir ve önerilen yöntem sayısal fonksiyon optimizasyonunda (en iyi değeri arama) başarılı elde etmiştir.

2. ÇADIR HARİTA (TENT MAP)

Çadır harita en sık kullanılan kaotik haritalardan birisidir. Bu harita literatürde rastgele sayı üretimi, şifreleme anahtarı üretimi, yer değiştirme kutusu üretimi ve optimizasyon yöntemleri için rastgele parçacık üretimi için kullanılmıştır. Çadır haritanın en önemli özelliği istatistiki olarak iyi sonuçlar üreten bir fonksiyon olmasıdır. Çadır haritanın matematiksel tanımı Eşitlik 1'de, çatallanma diyagramı ise Şekil 1' de verilmiştir [12,13].

$$x_{n+1} = \begin{cases} \dfrac{x_n}{0.7}, & if\ x_n < 0.7 \\ \dfrac{10}{3} x_n(1 - x_n), & if\ x_n \geq 0.7 \end{cases} \tag{2.1}$$



Şekil 1. *Çadır haritasının çatallanma diyagramı [13].*

Bu makalede çadır haritanın iyi istatistiksel özellikleri kullanılarak çadır harita tabanlı yeni bir sezgisel ve parçacık tabanlı bir optimizasyon yöntemi önerilmiştir.

3. ÇADIR HARİTA TABANLI KAOTİK OPTİMİZASYON YÖNTEMİ (TENT MAP BASED CHAOTIC OPTIMIZATION METHOD)

Bu makalede kaotik haritalardan olan çadır harita kullanılarak bir sürü optimizasyonu yöntemi önerilmiştir. Bu yöntemin temel amacı global optimum noktasını aramaktır. Kaotik arama algoritmaları literatürde ilk kez Hamaizia ve ark. [14] tarafından önerilmiştir ve bu çalışmada 2 boyutlu kaotik haritalar kullanılmıştır. Önerilen yöntem Hamaizia ve ark.'nın [14] yöntemine benzemektedir ancak bu yöntemde adım büyüklüğü yerine çadır haritasının ürettiği değerler kullanılmaktadır ve bir boyutlu bir kaotik harita kullanılarak arama işlemi yapılmaktadır. Çadır harita tabanlı kaotik optimizasyon yöntemi parçacık üretme, en iyi değeri seçme, tohum değeri oluşturma, çadır haritayla rastgele sayı üretme, parçacık güncelleme ve en iyi değeri güncelleme aşamalarından oluşmaktadır. Önerilen yöntemin adımları aşağıdaki gibidir.

Adım 1: Başlangıç popülasyonunu rastgele oluştur.

$$p_i = rand[0,1] \text{ x } (UB - LB) + LB, i = \{1,2, \dots, n\} \quad (3.1)$$

Adım 2: En iyi değeri seç.

$$p_{best} = \min\big(f(p_i)\big) \quad (3.2)$$

Eşitlik 3.2' de $f(.)$ Amaç fonksiyonunu, p_i parçacıkları ve p_{best} ise en iyi değeri ifade etmektedir.

Adım 3: Eşitlik 4'ü kullanarak tohum değerini oluştur.

$$x_1 = \frac{k}{iter} \quad (3.3)$$

k döngü değişkenin o anki değeri, iter maksimum döngü sayısı ve x_1 tohum değeri olarak ifade edilmektedir.

Adım 4: Her bir parçacık için Eşitlik 2.1'i kullanarak x sayı dizisini oluştur.

Adım 5: Eşitlik 3.4'ü kullanarak parçacıkları güncelle.

$$p_i = p_i + x_i(p_{best} - p_i) \quad (3.4)$$

Parçacık güncelleme de diferansiyel arama stratejisi ve kaos bir arada kullanılmıştır. Denklem 3.4'teki p_i o anki parçacığı x_i çadır haritayla üretilen rastgele sayı dizisini ve p_{best} lokal en iyi parçacık değerini ifade etmektedir.

Adım 6: Eğer parçacık değeri sınır değerlerini aşmışsa parçacık aşağıdaki gibi güncellenir.

$$p_i = \begin{cases} p_{best}, p_i < LB \\ p_{best}, p_i > UB \end{cases} \quad (3.5)$$

Eşitlik 3.5'te LB ve UB arama uzayının alt ve üst sınırlarını ifade etmektedir.

Adım 7: En iyi parçacığı güncelle.

Adım 8: Global optimum noktasına veya maksimum iterasyon sayısına ulaşılıncaya dek Adım 3-7' yi tekrarla.

1: Parametreleri ayarla (LB alt sınır, UB üst sınır, k maksimum iterasyon sayısı, pn parçacık sayısı)
2: Başlangıç popülasyonunu rastgele oluştur.
3: Her parçacığı değerlendir ve pbest (en iyi parçacık) değerini hesapla.
4: **for** k=1:MI **do**
5: $x_1 = \frac{k}{iter}$
6: **for** i=1:pn **do**
7: **if** x_i<0.7 **then**
8: $x_i = \frac{x_i}{0.7}$
9: **else**
10: $x_i = \frac{10}{3} x_i(1 - x_i)$
11: **End if**
12: $p_i = p_i + x_i(p_{best} - p_i)$
13: **if** p_i < LB or p_i>UB **then**
14: $p_i = p_{best}$
15: **End if**
16: Uygunluk (amaç) fonksiyonunu kullanarak tüm parçacıkları değerlendir.
17: **if** $p_i < p_{best}$ **then**
18: $p_i = p_{best}$
19: **End if**
20: **if** $p_{best} = amaçlanan\ değer$ **then**
21: break;
22: **End if**
23: **End for i**
24: **End for k**

Şekil 2. *Önerilen yöntemin sözde kodu.*

4. DENEYSEL SONUÇLAR (EXPERIMENTAL RESULTS)

Bu kısımda önerilen yöntemin performansı 12 adet sayısal kıyaslama fonksiyonu kullanılarak test edilmiştir. Bu fonksiyonlar tek ve çift modlu fonksiyonlardır ve literatürde yaygın olarak kullanılmaktadırlar. Ayrıca bu fonksiyonlardan elde edilen değerler literatürde yaygın olarak kullanılan optimizasyon yöntemleriyle de karşılaştırılmıştır. Test seti olarak kullanılan fonksiyonlar Tablo 1' de verilmiştir.

Tablo 1. *Test seti olarak kullanılan sayısal fonksiyonları [15].*

Grup	Adı	Test Fonksiyonu	Aralık	Global optimum
Tek modlu	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100, 100]^n	[0]^n
	Schwefel's 2.22	$f_2(x) \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]^n	[0]^n
	Schwefel's 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[-100, 100]^n	[0]^n
	Schwefel's 2.21	$f_4(x) = \max(\lvert x_i \rvert)$	[-100, 100]^n	[0]^n
	Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30, 30]^n	[0]^n
	Step	$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	[-100, 100]^n	[0.5]^n
	Noise	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	[-1.28, 1.28]^n	[0]^n
Çok modlu	Rastrigin	$f_8(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12, 5.12]^n	[0]^n
	Ackley	$f_9(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + \mathrm{e}$	[-32, 32]^n	[0]^n
	Griewank	$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	[-600, 600]^n	[0]^n
	Generalized Penalized I	$f_{11}(x) = \frac{\pi}{n} \Big\{ 10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)[1 + 10\sin^2(\pi y_{i+1}) + (y_n + 1)^2] \Big\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	[-50, 50]^n	[0]^n

	Generalized Penalized II	$f_{12}(x) = \frac{1}{10}\Big\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i)] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\Big\} + \sum_{i=1}^{n} u(x_i,5,100,4)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i-a)^m, x_i < -a \end{cases}$	$[-50, 50]^n$	$[0]^n$

Elde edilen sonuçların anlamlandırılması ve diğer optimizasyon yöntemleriyle karşılaştırmaların yapılabilmesi için Tablo 1' de listelenen veri seti kullanılmıştır. Bu veri seti literatürde sıklıkla kullanılan sayısal karşılaştırma fonksiyonlarından oluşmaktadır. Kaotik karides sürü (KKS) [16], kaotik biyocoğrafya optimizasyonu (KBO) [17] ve kaotik yerçekimi algoritması (KYA) [18] yöntemleri kullanılarak karşılaştırma sonuçları elde edilmiştir. Bu kaotik yöntemlerin yanı sıra literatürde sıklıkla kullanılan sinüs kosinüs algoritması (SKA) [6] ve yapay arı kolonisi algoritması (YAK) [19] da test sonuçlarını elde etmek için kullanılmıştır. SKA literatürde birçok varyasyonu olan matematiksel bir optimizasyon yöntemidir ve parçacık güncellemek için sinüs kosinüs fonksiyonlarını kullanmaktadır. Bu fonksiyonların yanı sıra 4 adet rastgele değişken kullanarak optimum noktayı arayan bir algoritmadır. YAK ise arıların hareketlerini ve polen arama süreçlerini modelleyen bir doğa tabanlı optimizasyon yöntemidir. YAK yöntemi ilk kez 2005 yılında Derviş Karaboğa tarafından önerilmiştir [20] ve sonraki yıllarda birçok versiyonu literatürde sunulmuştur. Karşılaştırma için kullanılan optimizasyon yöntemlerine ait parametreler ise Tablo 2'de listelenmiştir.

Tablo 2. *Optimizasyon yöntemlerinin kullandığı parametreler*

Yöntem	Popülasyon	Maksimum iterasyon	Boyut	Diğerleri
KKS	40	500	30	N_{max}=0.01, V_f=0.02,D_{max}=0.005
KBO	40	500	30	G_0=100, α=20
KYA	40	500	30	Mu=0.005, μ=0.8
SKA	40	500	30	r_1, r_2, r_3 ve r_4 rastgele üretilmiştir. Bu değerleri üretmek için MATLAB'ın rand fonksiyonu kullanılmıştır.
YAK	40	500	30	Limit=200
Önerilen yöntem	40	500	30	x_1=k/iter

Tablo 2'deki gibi parametreleri ayarlanan kaotik optimizasyon yöntemleri 30 kez çalıştırılarak en iyi değerler kaydedilmiş ve Tablo 3' te elde edilen ortalamalar ve standart sapmalar gösterilmiştir. Ayrıca elde edilen en iyi değerler Tablo 3'te kalın yazı fontuyla gösterilmiştir.

Tablo 3. *Performans karşılaştırma sonuçları*

f	Kriter	KKS	KBO	KYA	SKA	YAK	Önerilen yöntem
f_1	Ortalama	7.89×10^{-1}	1.77×10^{0}	1.18×10^{-16}	1.11×10^{1}	3.15×10^{-5}	$\mathbf{7.09 \times 10^{-77}}$
	Standart sapma	2.15×10^{-1}	8.92×10^{3}	2.14×10^{-17}	1.83×10^{1}	3.47×10^{-5}	$\mathbf{3.88 \times 10^{-76}}$
f_2	Ortalama	2.31×10^{2}	1.20×10^{-1}	2.70×10^{-1}	1.62×10^{-2}	4.42×10^{-3}	$\mathbf{3.38 \times 10^{-37}}$
	Standart sapma	3.03×10^{1}	6.60×10^{-2}	5.69×10^{-1}	2.12×10^{-2}	1.40×10^{-3}	$\mathbf{1.85 \times 10^{-36}}$
f_3	Ortalama	4.31×10^{2}	7.55×10^{3}	6.55×10^{2}	2.10×10^{4}	1.78×10^{4}	$\mathbf{1.10 \times 10^{-53}}$
	Standart sapma	1.31×10^{2}	3.23×10^{3}	2.69×10^{2}	1.12×10^{4}	2.92×10^{3}	$\mathbf{6.03 \times 10^{-53}}$
f_4	Ortalama	9.71×10^{0}	1.71×10^{1}	4.86×10^{0}	7.37×10^{1}	4.35×10^{1}	$\mathbf{2.17 \times 10^{-21}}$
	Standart sapma	1.44×10^{0}	3.00×10^{0}	1.23×10^{0}	2.11×10^{1}	5.05×10^{0}	$\mathbf{1.19 \times 10^{-20}}$
f_5	Ortalama	1.91×10^{2}	4.18×10^{2}	5.41×10^{1}	1.70×10^{6}	4.36×10^{1}	**0**
	Standart sapma	1.24×10^{2}	2.96×10^{2}	5.16×10^{1}	4.86×10^{6}	4.41×10^{1}	**0**
f_6	Ortalama	5.15×10^{-1}	1.45×10^{0}	7.77×10^{-17}	3.69×10^{1}	5.82×10^{-5}	**0**
	Standart sapma	1.90×10^{-1}	6.30×10^{-1}	2.40×10^{-17}	8.84×10^{1}	9.74×10^{-5}	**0**
f_7	Ortalama	9.44×10^{-2}	9.01×10^{-2}	1.64×10^{-1}	6.65×10^{-1}	3.03×10^{-1}	$\mathbf{1.6 \times 10^{-3}}$
	Standart sapma	3.07×10^{-2}	5.33×10^{-2}	2.76×10^{-1}	1.81×10^{0}	8×10^{-2}	$\mathbf{1.7 \times 10^{-3}}$
f_8	Ortalama	6.43×10^{1}	3.14×10^{2}	2.33×10^{1}	5.91×10^{1}	5.09×10^{0}	**0**
	Standart sapma	9.44×10^{1}	2.62×10^{0}	5.75×10^{0}	3.65×10^{1}	2.32×10^{0}	**0**
f_9	Ortalama	8.07×10^{0}	6.45×10^{-1}	6.35×10^{-9}	1.15×10^{1}	1.54×10^{-1}	$\mathbf{8.88 \times 10^{-16}}$
	Standart sapma	7.34×10^{-1}	4.88×10^{-1}	7.30×10^{-10}	9.95×10^{0}	1.43×10^{-1}	**0**
f_{10}	Ortalama	1.65×10^{-1}	8.25×10^{-1}	2.09×10^{1}	1.04×10^{0}	2.60×10^{-2}	$\mathbf{3.60 \times 10^{-3}}$
	Standart sapma	8.30×10^{-2}	1.57×10^{-1}	5.32×10^{0}	2.94×10^{-1}	2.86×10^{-2}	$\mathbf{8.00 \times 10^{-3}}$

f_{11}	Ortalama	4.37×10^{0}	2.10×10^{-2}	1.20×10^{0}	4.17×10^{6}	1.93×10^{-5}	**4.02×10^{-17}**
	Standart sapma	1.09×10^{0}	3.77×10^{-2}	8.20×10^{-1}	1.38×10^{7}	4.33×10^{-5}	**2.20×10^{-17}**
f_{12}	Ortalama	1.11×10^{1}	1.08×10^{-1}	1.36×10^{-1}	1.51×10^{7}	1.01×10^{-5}	**2.17×10^{-31}**
	S.D.	3.23×10^{1}	5.85×10^{-2}	3.50×10^{-1}	2.87×10^{7}	1.04×10^{-5}	**8.71×10^{-31}**

Tablo 3' te görüldüğü gibi, önerilen yöntem literatürde daha önceden önerilmiş ve yaygın olarak kullanılan 5 adet optimizasyon yöntemiyle 12 adet sayısal uygunluk fonksiyonu kullanılarak karşılaştırılmıştır. Karşılaştırma sonuçları önerilen yöntemin tüm fonksiyonlarında en iyi değeri elde ettiğini ve hatta f_5,f_6 ve f_8 fonksiyonlarında global optimum değerine ulaştığını açıkça göstermektedir.

5. TARTIŞMALAR (DISCUSSIONS)

Bu makalede yeni bir çadır harita tabanlı sürü optimizasyon tekniği önerilmiştir. Önerilen yöntem kaotik bir yöntemdir. Bu yöntem rastgele parçacık üretme, amaç fonksiyonu kullanarak parçacıkları değerlendirme, bireysel en iyi değerleri tohum değeri olarak hesaplama, çadır haritası kullanarak parçacıkları güncelleme ve en iyi değeri seçme aşamalarından oluşmaktadır. Bu yöntemde kaos kullanılmasının en önemli amacı, kaotik haritaların iyi istatistiksel özellikler göstermesidir. Bu sebepten dolayı literatürdeki birçok yöntemde kaotik haritalar kullanılmaktadır ve çadır harita optimizasyon yöntemlerinde sıklıkla kullanılan kaotik haritalardan birisidir. Önerilen yöntemi diğer yöntemlerden farklı kılan özellik parçacık güncellemesinin doğrudan çadır haritası kullanılarak yapılmasıdır. Diğer yöntemlerde [8,16-18] parçacık güncellemede ki rastgele değerler kaotik haritalar kullanılarak üretilmektedir. KKS [16] yönteminde literatürde sıklıkla kullanılan karides sürüsü optimizasyon algoritması 12 kaotik yöntem kullanılmıştır. Bunlardan en iyi sonucu verenlerden birisi de çadır haritadır. Karides sürüsü algoritmasındaki ağılıkları güncellemek için kaotik haritalardan istifade edilerek KKS elde edilmiştir. KBO [17] yönteminde de habitat seçme (göç belirleme) ve mutasyon olasılığı 10 adet kaotik harita kullanılarak seçilmiştir. Biyocoğrafya tabanlı optimizasyon yönteminde habitat seçme ve mutasyon işlemleri rastgele fonksiyonu kullanılarak seçilmektedir. KBO bu rastgele seçimi kaotik haritaları rastgele sayı üreteci kullanarak üretmiştir. KYO [18]' da ise yerçekimi optimizasyonu yönteminde yerel arama yöntemi lojistik harita kullanılarak gerçekleştirilmiştir. Arama stratejisi Denklem 5.1' de göterildiği gibidir.

$$X_i = X_{min} + cX_i(X_{max} - X_{min}) \tag{5.1}$$

Denklem 5.1'de $\boldsymbol{X_{min}}$ ve $\boldsymbol{X_{max}}$ minimum ve maksimum lokasyonları, $\boldsymbol{X_i}$ o anki lokasyonu belirtmektedir. $\boldsymbol{c}$ ise kaotik harita kullanılarak üretilen rastgele değerleri ifade etmektedir. Yerçekimi optimizasyon yönteminde rastgele üretilen değerler KYO yönteminde kaos kullanılarak üretilmiştir çünkü kaos iyi istatistiki özelliklere sahiptir. Referans [18]' deki çalışmada kullanılan kaotik harita lojistik haritadır.

Karşılaştırmalarda kullanılan ve yukarıda bahsedilen makaleler yaygın bir etkiye sahiptir ve Chen ve arkadaşlarının makalesinde [8] karşılaştırma amacıyla kullanılan kaotik optimizasyon yöntemleridir. Bu yöntemler ve önerilen çadır haritası tabanlı yöntem MATLAB 2017a programı kullanılarak simüle edilmiştir. Referanslarda gösterilen ve karşılaştırma amacıyla kaotik yöntemler ele alındığında [8,16-18] bu yöntemler var olan bir optimizasyon yönteminin başarımını arttırmak için rastgele değişkenler yerine kaotik haritaları kullanmaktadır. Bu makalede ise diferansiyel arama yaklaşımı ve çadır harita

kullanılarak yeni bir kaotik optimizasyon yöntemi önerilmiştir ve kaosta var olan istatiski başarımın parçacık güncellemeye doğrudan etki etmesi sağlanmıştır.

Kaotik yöntemlerin yanı sıra literatürde sıklıkla kullanılan ve yaygın bir etkiye sahip olan SKA ve YAK yöntemleri de karşılaştırmalar için kullanılmıştır. YAK yönteminin literatürde birçok versiyonu bulunmaktadır. Karşılaştırmalar için YAK yönteminin 2014 versiyonu kullanılmıştır. Tablo 2 önerilen yöntemin bu yöntemlerden de daha başarılı olduğunu göstermektedir.

Deneysel sonuçlar kısmında literatürde yaygın olarak kullanılan 12 adet kıyaslama fonksiyonu kullanılarak yöntemler test edilmiştir ve tüm kıyaslama fonksiyonlarında önerilen yöntemin en iyi başarımı elde ettiği gözlemlenmiştir. Bunun yanı sıra $\boldsymbol{f_5}, \boldsymbol{f_6}$ ve $\boldsymbol{f_8}$ fonksiyonlarında önerilen yöntem kullanılarak arzu edilen değer elde edilmiştir.

Ayrıca önerilen yöntem Özkaynak'ın [21] makalesinden ilham almıştır ve bu makalede kaosun evrimsel programlama üzerindeki etkisi gösterilmiştir ve kaotik haritaların rastgele sayı üretme performansları grafiksel olarak gösterilmiştir. Gelecekteki çalışmalarda daha iyi istatistiki özellik gösteren çok boyutlu kaotik haritalar kullanılarak daha başarılı yöntemlerin geliştirilebileceği de gösterilmiştir.

Kısacası bu makale diferansiyel yaklaşım ve çadır haritası kullanılarak parçacık güncelleme denklemi elde edilmiş ve bu denklem kullanılarak yeni bir sürü tabanlı optimizasyon algoritması önerilmiştir. Bu yöntem sayısal fonksiyon optimizasyonu için kullanılmış ve Tablo 2' de elde edilen değerler listelenmiştir. Deneysel sonuçlar kısmı önerilen parçacık güncelleme stratejisinin pozitif etkilerini açıkça göstermiştir ve karmaşık olmayan bir matematiksel altyapıyla başarılı sonuçlar elde edilmiştir. Kaosun pozitif etkisi doğrudan arama stratejisine yansıtılmıştır ve sayısal kıyaslama fonksiyonları için yerel minimumda takılma problemi kaotik çadır haritası kullanılarak çözülmeye çalışılarak daha iyi sonuçlar elde edilmiştir.

6. SONUÇLAR VE ÖNERİLER (CONCLUSIONS AND RECOMMANDATIONS)

Bir sürü tabanlı optimizasyon yöntemi oluştururken yapılması gereken en önemli husus parçacıkların nasıl güncelleneceğine karar vermektir. Bu çalışmada çadır haritası tabanlı yeni bir kaotik sürü optimizasyonu yöntemi önerilmiştir. Çadır haritasının en önemli özelliği rastgele sayı üretiminde başarılı sonuçlar elde etmesidir ve bu sebepten dolayı önerilen yöntemde parçacıklar çadır haritası kullanılarak güncellenmektedir. Önerilen yöntem basit ancak efektif bir yöntemdir ve bu çalışmada önerilen yöntemin adımları ve sözde kodu açık bir şekilde verilmiştir. Önerilen yöntemi test etmek için literatürde yaygın olarak kullanılan amaç fonksiyonları ve kaotik optimizasyon yöntemleri kullanılmıştır. Deneysel sonuçlar önerilen yöntemin tüm test setinde başarılı sonuçlar ürettiğini açık bir şekilde göstermektedir. Bu başarının temel sebebi, kaotik haritaların sadece rastgele sayı üreteci olarak kullanılmaması, parçacık güncellemede kullanılmasıdır.

Bu çalışma, gelecekteki çalışmalarda gerçek dünya problemlerinin de önerilen çadır harita tabanlı yöntemle çözülebileceğini ve tüm kaotik haritaların sunulan yapı içerisinde kullanılabileceğini göstermektedir.

KAYNAKLAR (REFERENCES)

[1] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Adv. Eng. Softw. 95 (2016) 51-67.

[2] E. Tanyıldızı, T. Cigal, Whale Optimization Algorithms With Chaotic Mapping, Science and Eng. J of Fırat Univ. (2017) 29(1), 309-319.

[3] W. Zang, L. Ren, A cloud model based DNA genetic algorithm for numerical optimization problems, Future Gener. Comp. Sy. 81 (2018) 465–477.

[4] R. Ebenhart, Kennedy, Particle swarm optimization, in: Proceeding IEEE Inter Conference on Neural Networks, 4, Perth, Australia, Piscat-away, 1995, pp. 1942–1948.

[5] Z. Beheshti, S.M. Shamsuddin, Non-parametric particle swarm optimization for global optimization, Appl. Soft Comput. 28 (2015) 345–359.

[6] S. Mirjalili, SCA: A Sine Cosine Algorithm for solving optimization problems, Lect. Notes Artif. Int. 96 (2016) 120–133.

[7] G. Demir, E. Tanyıldızı, The Use of Sine Cosine Algorithm (SCA) in Solution of Optimization Problems, Science and Eng. J of Fırat Univ. 29(1), 227-238.

[8] K. Chen, F. Zhou, A. Liu, Chaotic dynamic weight particle swarm optimization for numerical function optimization, Lect. Notes Artif. Int. (2018) 139, 23-40.

[9] Y. Zhou, J. Wu, L. Ji, Z. Yu, K. Lin, L. Hao, Transient stability preventive control of power systems using chaotic particle swarm optimization combined with two-stage support vector machine, Electr. Pow. Syst. Res. 155 (2018) 111–120.

[10] M. Mitic, N. Vukovic, M. Petrovic, Z. Miljkovic, Chaotic fruit fly optimization algorithm, Lect. Notes Artif. Int. 89 (2015) 446–458.

[11] A. Naanaa, Fast chaotic optimization algorithm based on spatiotemporal maps for global optimization, Appl. Math. Comput. 269 (2015) 402–411.

[12] J.A. Martinez-Nonthe, A. Castaneda-Solis, A. Diaz-Mendez, M. Cruz-Irrisson, R. Vazquez-Medina, Chaotic block cryptosystem using high precision approaches to tent map, Microelectron. Eng. 90 (2012) 168–172.

[13] Y. Zhou, L. Bao, C.L.P. Chen, A new 1D chaotic system for image encryption, Signal Process. 97 (2014) 172–182.

[14] T. Hamaizia, R. Lozi, N. Hamri, Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor, Appl. Math. Comput. 219 (2012) 188–196.

[15] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.

[16] G.G. Wang, L. Guo, A.H. Gandomi, et al., Chaotic krill herd algorithm, Inf. Sci. 274 (2014) 17–34.

[17] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimization with chaos, Neural Comput. Appl. 25 (5) (2014) 1077–1097.

[18] F.Y. Ju, W.C. Hong, Application of seasonal SVR with chaotic gravitational search algorithm in electricity forecasting, Appl. Math. Model. 37 (23) (2013) 9643–9651.

[19] G. Li, P. Niu, Y. Ma, et al., Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency, Lect. Notes Artif. Int. 67 (2014) 278–289.

[20] D. Karaboğa, B. Gorkemli, A quick artificial bee colony (qABC) algorithm and its performance on optimization problems, Appl. Soft Comput. 23 (2014) 227-238.

[21] F. Özkaynak, A novel method to improve the performance of chaos based evolutionary algorithms, Optik 126 (2015) 5434–5438.