Kaotik Tabanlı Diferansiyel (Farksal) Gelişim Algoritması

[1]Mehmet Eser *[1]Uğur Yüzgeç
[1]Bilecik Şeyh Edebali Üniversitesi, Bilgisayar Mühendisliği Bölümü, 11210, Gülümbe, Bilecik

Abstract

Differential evolution (DE) is one of the evolutionary optimization methods used for solving the problems which include nonlinear and multi-objective functions. The main idea of DE algorithm is that the difference value between two individuals adds to a third individual in population. Evolutionary Algorithms is based on the random number generators. One of the biggest problems of the DE algorithm is the local convergence in the optimization process. In this study, to avoid the local optimal point and to improve the performance of the algorithm, the chaotic random number generator is proposed instead of the random number generators. To evaluate the performance of the proposed algorithm, optimization test problems from the literature were used. Furthermore, classical DE and chaotic based DE algorithm were compared with each other.

Key words: Differential Evolution Algorithm, Optimization, Lorenz, Chaotic

Özet

Farksal Gelişim (Differential Algorithm, DE) çok amaçlı, doğrusal olmayan problemlerin çözümü için kullanılan evrimsel optimizasyon algoritmalarından birisidir. Temelde DE Algoritmasının fikri popülasyondaki iki bireyin arasındaki farkın bir üçüncü bireye ilave edilmesidir. Evrimsel algoritmaların genel olarak tüm işlemleri rastgele sayı üretimine dayanan işlemlerden oluşmaktadır. Rastgele sayı üretim sürecine dayanan DE algoritmasının en büyük sorunlarından birisi optimizasyon problemleri sonuçlarındaki lokal yakınsamadır. Bu çalışmada lokal optimal noktalardan kaçınmak ve algoritmanın performansını artırmak için rastgele sayı üretimi süreci yerine kaotik tabanlı bir sistem önerilmiştir. Önerilen algoritmanın performansını değerlendirmek için literatürden alınan optimizasyon problemleri kullanılmıştır. Ayrıca klasik DE algoritması ile kaotik tabanlı DE algoritmasının kıyaslaması yapılmıştır.

Anahtar Kelimeler: Farksal Gelişim Algoritması, Optimizasyon, Lorenz, Kaotik

1. Giriş

Optimizasyon problemleriyle karşılaşıldığında genellikle ilk yapılan, probleme özel sezgisel bir yaklaşım tekniği belirlemektir. Gelişime dayalı algoritmalar, genel metotlarla karşılaştırıldığında oldukça üstün performans göstermektedirler. Farksal Gelişim Algoritması basit, fakat güçlü popülasyon tabanlı bir algoritma olup, özellikle tamamen düzenlenmiş uzayda tanımlı ve gerçek değere sahip tasarım parametrelerini içeren fonksiyonların optimizasyonunda kullanılan bir direkt araştırma algoritmasıdır [1,2].

DE Algoritmasının temel fikri popülasyondaki iki bireyin arasındaki farkın bir üçüncü bireye ilave edilmesidir. Genetik Algoritmalardan farklı olarak, DE algoritması tüm optimizasyon süreci boyunca yalnızca çaprazlama oranı, ölçekleme faktörü ve popülasyon boyu gibi birkaç kontrol parametresine sahiptir ve kendi yapısı içerisinde kayan noktalı sayıların gerçek kodlamasını kullanır. Kontrol parametrelerinin değerlerinin seçim işlemi, arama algoritmasının verimi ve elde

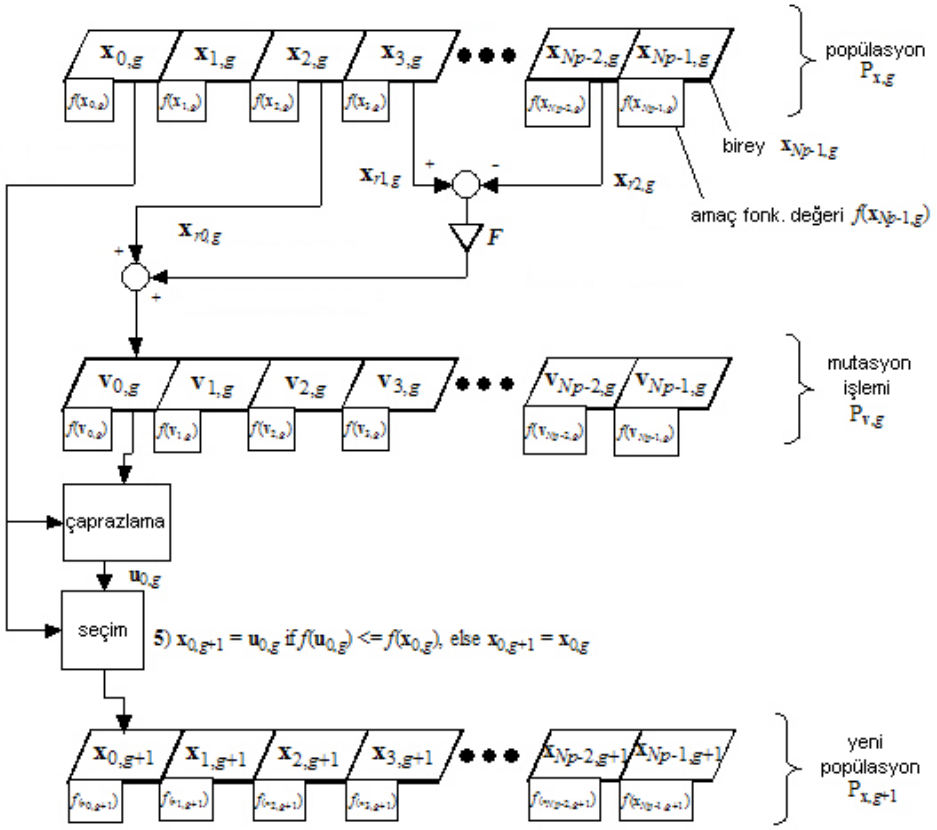
*Adres: Bilecik Şeyh Edebali Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 11210, Gülümbe, Bilecik. E-mail address: ugur.yuzgec@bilecik.edu.tr, Tel: +902282141555 Faks: +902282141017

edilen çözümün kalitesini artırmak için dikkatli bir şekilde yapılmalıdır. DE algoritmasının verimliliği ve gürbüzlüğü tam olarak bu kontrol parametrelerinin ayarlanmasına bağlıdır [3-5].

Literatürde Farksal Gelişim Algoritmasının pek çok değişik türü bulunmaktadır. Ölçekleme ve çaprazlama faktörünün rastgele değiştirildiği Adaptif Farksal Gelişim Algoritması (ADE) [6], her bir nesilde daha iyi bir çözüm bulmak için eş zamanlı bir tahmin ve onun karşıtı uygun bir tahmin göz önüne alınması prensibine dayanan Karşıtlık Temelli Farksal Gelişim Algoritması (ODE) [7-9] ve her iki yaklaşımın birleşimi olarak Adaptif Karşıtlık Temelli Farksal Gelişim Algoritması (AODE) [1,2] bunlardan sayılabilir. Bu çalışmada DE algoritmasının temelinde yer alan rastgele sayı üretim işlemi yerine kaotik tabanlı bir sistem önerilmiştir. Bu çalışmanın amacı optimizasyon sırasında farksal gelişim algoritmasının işlemsel yükünü azaltmak ve çözüm uzayındaki lokal optimal noktalara takılmayarak, arama performansını artırmaktır.

2. Farksal Gelişim Algoritması

Farksal Gelişim Algoritması Price ve Storn tarafından 1995 yılında geliştirilmiş, özellikle sürekli verilerin söz konusu olduğu problemlerde etkin sonuçlar verebilen, isleyiş ve operatörleri itibariyle genetik algoritmaya dayanan popülasyon temelli sezgisel optimizasyon tekniğidir [3-5]. Şekil 1'de DE algoritmasının temel işleyişi verilmiştir.



Şekil 1. Farksal Gelişim Algoritmasının işleyiş şeması [3-5]

Temel olarak, DE algoritması popülasyon içerisinden rastgele seçilen iki bireyin ağırlık farkının üçüncü bir bireye eklenmesi mantığına dayanmaktadır. Literatürde DE algoritmasının bu

işleminin farklı çeşitleri bulunmaktadır:

- en iyi birey / 1 fark vektörü / üstel çaprazlama
- rastgele bireyler / 1 fark vektörü / üstel çaprazlama
- rastgele ve en iyi bireyler / 1 fark vektörü / üstel çaprazlama
- en iyi birey / 2 fark vektörü / üstel çaprazlama
- rastgele bireyler / 2 fark vektörü / üstel çaprazlama
- en iyi birey / 1 fark vektörü / binom çaprazlama
- rastgele bireyler / 1 fark vektörü / binom çaprazlama
- rastgele ve en iyi bireyler / 1 fark vektörü / binom çaprazlama
- en iyi birey / 2 fark vektörü / binom çaprazlama
- rastgele bireyler / 2 fark vektörü / binom çaprazlama

Bu genel gösterim tarzında, ilk kısım popülasyon içinden seçilen bireyleri, ikinci kısım seçilen bireylerin oluşturacağı fark vektörünün sayısını, son kısım ise çaprazlama tipini belirtmektedir. Bu çalışmada pratikte [10] en sık kullanılan iki stratejisi rastgele ve en iyi bireyler / 1 fark vektörü / binom çaprazlama ile rastgele bireyler / 1 fark vektörü / binom çaprazlama olarak seçilmiştir. DE algoritması 3 önemli kontrol parametresine sahiptir: skala faktörü (SF), çaprazlama olasılık katsayısı (CR) ve popülasyon boyu (PS). DE algoritmasının başında PS optimizasyon parametrelerine göre belirlenir ve optimizasyon sürecinde değiştirilmez. DE algoritması 3 temel operatöre sahiptir: mutasyon, çaprazlama ve seçim [3]. Mutasyon ve çaprazlama operatörleri yeni bireyler üretir ve seçim operatörü ile uygun olanlar belirlenir, bu şekilde popülasyonda sürekli en iyi bireylerin bulunması sağlanır.

2.1. Mutasyon Operatörü

Her iki DE algoritma stratejisi için mutasyona uğramış bireyler $v_{i,g}$ aşağıdaki şekilde bulunur:

$$v_{i,g+1} = x_{i,g} + SF\left(x_{b,g} - x_{i,g}\right) + SF\left(x_{r1,g} - x_{r2,g}\right) \quad (1)$$

$$v_{i,g+1} = x_{r3,g} + SF\left(x_{r1,g} - x_{r2,g}\right) \quad (2)$$

Bu eşitliklerde $r_1, r_2, r_3 \in [1,PS]$ rastgele katsayılardır. $x_{b,g}$ popülasyon içerisindeki en iyi maliyet fonksiyonuna sahip bireyi göstermektedir. Rastgele katsayılar birbirinden farklı olmak zorundadır. Skala faktörü (SF) sabit bir değerdir ve $0 \le SF \le 2$ olarak değişir. Skala faktörü rastgele seçilen bireylerin fark vektörlerinin yükseltilmesinin kontrolünde kullanılır.

2.2. Çaprazlama Operatörü

Yeni bireyler çaprazlama operatörü ile aşağıdaki şekilde bulunabilir:

$$u_{i,g+1} = \begin{cases} v_{i,g+1}, eğer\ r \le CR \\ x_{i,g}, \quad eğer\ r > CR \end{cases} \quad (3)$$

Bu denklemde r ∈ [0, 1] rastgele sayı üretecini ve $u_{i,g+1}$ deneme bireylerini göstermektedir. Çaprazlama sabiti (*CR)* ∈ [0, 1] genellikle kullanıcı tarafından belirlenir. Eğer *CR* = 1 ise, oluşacak her yeni birey mutant olarak belirlenir, yani rastgele seçilen fark vektörüne ($v_{i,g+1}$) göre seçilir. Diğer taraftan *CR* = 0 durumunda, tüm bireyler önceki jenerasyondan ($x_{i,g}$) gelir. Diğer bir deyişle, populasyonda değişim olmaz, yeni bireyler oluşmaz. Rastgele sayı üreteci, mutant bireylerden ($v_{i,g+1}$) en az birinden bir sonraki jenerasyona ($u_{i,g+1}$) birey üretilmesini garanti eder.

2.3. Seçim Operatörü

Bir sonraki jenerasyondaki bireylerin $x_{i,g+1}$ tespitinde seçim operatörünün işletilmesi sırasında, deneme bireyinin $u_{i,g+1}$ maliyet fonksiyonu değeri hedef bireyin $x_{i,g}$ maliyet fonksiyonu ile karşılaştırılır ve sonraki jenerasyona optimizasyon problemine göre yeni birey seçilir. Aşağıdaki eşitlikte minimum problemi için seçim operatörünün denklemi verilmiştir:

$$x_{i,g+1} = \begin{cases} u_{i,g+1}, eğer\ f(u_{i,g+1}) < f(x_{i,g}) \\ x_{i,g}, \quad diğer\ durumlar \end{cases} \tag{4}$$

Burada f maliyet fonksiyonunu göstermektedir. Eğer deneme bireyi $u_{i,g+1}$ hedef bireyden $x_{i,g}$ daha iyi bir maliyet fonksiyonuna sahipse, bir sonraki jenerasyonda kullanılacak birey $x_{i,g+1}$ deneme bireyi $u_{i,g+1}$ olarak seçilir. Aksi durumda popülasyondaki hedef bireyde değişiklik olmaz.

3. Kaotik Sistemler

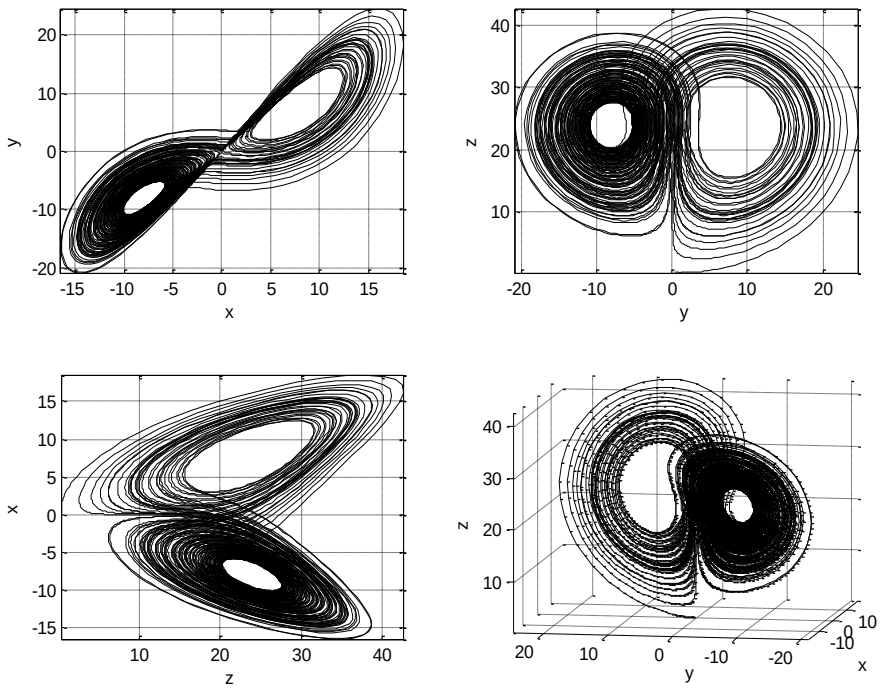
Kaos, başlangıç şartlarına aşırı duyarlı olan ve gürültü benzeri geniş güç spektrumuna sahip olan, düzensizliğin düzeni seklinde tanımlanabilir [11]. Literatürde kaotik sistemler ile ilgili Lorenz [12], ve Rössler [13,14] sistemleri gibi çok geniş bir şekilde çalışılmış pek çok örnek vardır. Bu çalışmada DE algoritması içerisinde rastgele sayı üreteci olarak Lorenz kaotik sistemi kullanılmıştır.

3.1. Lorenz Çekicisi

Kaotik sürekli-zamanlı dinamik sistemlerden birisi de Lorenz çekicisidir. Lorenz kaotik sistemindeki üç değişkenden ikisi sıcaklık, diğeri ise hız alanı katsayısıdır. Lorenz sisteminin denklemleri aşağıda verilmiştir:

$$\dot{x} = s(y - x) \tag{5}$$
$$\dot{y} = rx - y - xz \tag{6}$$
$$\dot{z} = xy - bz \tag{7}$$

Burada; s, r ve b durum değişkenleridir ve değerleri $s = 11,\ r = 25$ ve $b = 8/3$ olarak kullanılmıştır. Şekil 2'de bir lorenz çekicisinin $x_0 = 0.3, y_0 = 0.3, z_0 = 0.3$ başlangıç değerleri için runge-kutta yöntemi ile çözülmüş çıkışları gösterilmiştir.



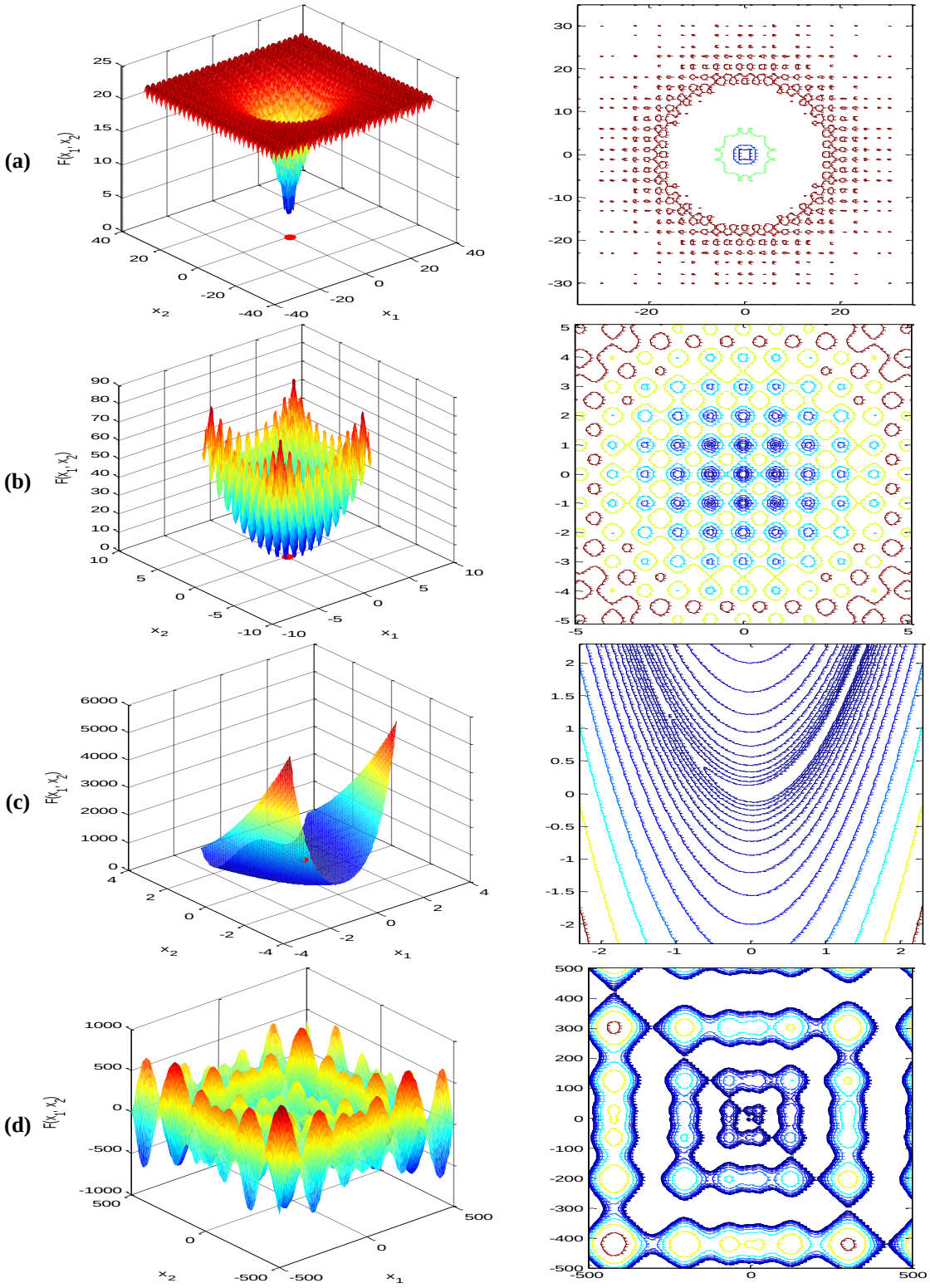
Şekil 2. Lorenz Çekicisine ait çıkışlar ($x_0 = 0.3, y_0 = 0.3, z_0 = 0.3$)

4. Kaotik Tabanlı DE Algoritması

Kaotik tabanlı farksal gelişim algoritmasının kaba kodu Şekil 3'de verilmiştir. Önerilen algoritmanın performansının değerlendirilmesi amacıyla literatürden alınan optimizasyon test problemlerinden dört tanesi ile algoritma 10 defa koşturulmuştur. Kullanılan test fonksiyonlarının iki değişken için çizimleri ile izdüşümleri Şekil 4'de gösterilmiştir.

Input: Popülasyon boyu, Problem boyutu, Sınır değerler,
Çaprazlama sabiti (*CR*), Skala faktörü (*SF*)
Output: En iyi maliyet fonksiyonu, En iyi bireyler
Popülasyon ilk değer atama_kaotik(Popülasyon boyu, Problem boyutu, Sınır değerler)
Maliyet hesapla(Popülasyon)
En iyi birey←Popülasyon içindeki en iyi maliyete sahip birey
En iyi maliyet değeri← Maliyet hesapla(En iyi birey)
While(Durdurma kriteri)
Yeni popülasyon sıfırla
Mutasyon bireyleri hesapla_kaotik (Skala faktörü, DE stratejisi)
Çaprazlama işlemi_kaotik (Çaprazlama sabiti)
Maliyet hesapla(Yeni popülasyon)
If (Maliyet(Yeni birey)≤ Maliyet(Eski birey))
Yeni Popülasyon←Yeni birey
Else
Yeni Popülasyon← Eski birey
End
Popülasyon← Yeni Popülasyon
En iyi birey←Popülasyon içindeki en iyi maliyete sahip birey
En iyi maliyet değeri← Maliyet hesapla(En iyi birey)
End
Return(En iyi maliyet fonksiyonu, En iyi bireyler)

Şekil 3. Kaotik Tabanlı DE Algoritması Kaba Kodu

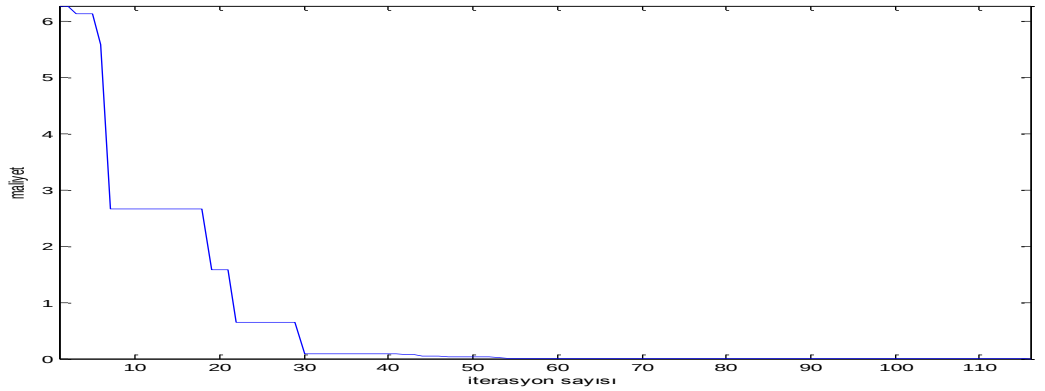


Şekil 4. Optimizasyon Test Problemleri.
(a) Ackley (b) Rastrigin (c) Rosenbrock (d) Schweffel

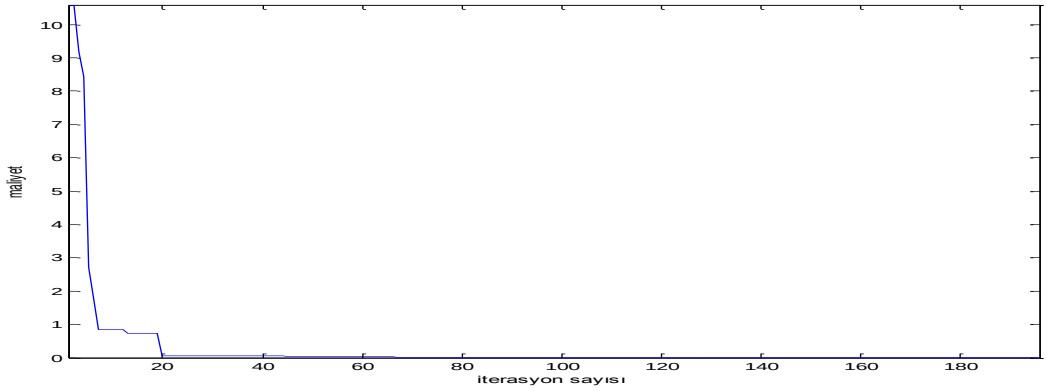
Optimizasyon test problemlerinin formülleri ve global minimum değerleri Tablo1'de verilmiştir. Önerilen algoritmanın performansının test edilmesi için Intel Core i5-3230M 2.60GHz'lik işlemcili ve 8GB RAM sahip bir bilgisayar kullanılmıştır. Yapılan benzetim çalışmalarında klasik DE algoritması ile kaotik tabanlı DE algoritmasının başlangıç parametreleri için popülasyon boyu (*PS*) 20, çaprazlama sabiti (*CR*) 0.7, skala faktörü (*SF*) 1.2, maksimum iterasyon sayısı (*itermax*) 200, en düşük maliyet değeri 1e-6 olarak seçilmiştir. Yine en sık tercih edilen DE stratejilerinden iki tanesi rastgele ve en iyi bireyler / 1 fark vektörü / binom çaprazlama ile rastgele bireyler / 1 fark vektörü / binom çaprazlama benzetim çalışmalarında kullanılmıştır. Şekil 5'de Ackley ve Rastrigin test fonksiyonları için önerilen kaotik tabanlı DE algoritmasının bir kere koşturulması sonucu elde edilen en iyi maliyet değerlerinin iterasyonlar boyunca değişimi gösterilmiştir. Ackley fonksiyonu için 118 iterasyonda, Rastrigin fonksiyonu için ise 188 iterasyonda algoritma çözüme ulaşmıştır.

Tablo 1. Test Fonksiyonları

Fonksiyon İsmi	Formülü	min f
Ackley	$f(x) = 20\left[1 - e^{-0.2\sqrt{0.5({x_1}^2+{x_2}^2)}}\right] - e^{0.5(\cos 2\pi x_1+\cos 2\pi x_2)} + e^1$	0
Rastrigin	$f(x) = {x_1}^2 + {x_2}^2 - 10\cos 2\pi\, x_1 - 10\cos 2\pi\, x_2 + 20$	0
Rosenbrock	$f(x) = 100({x_1}^2 - x_2)^2 + (1 - x_1)^2$	0
Schweffel	$f(x) = -x_1 \sin\sqrt{\lvert x_1\rvert} - x_2 \sin\sqrt{\lvert x_2\rvert}$	-837.965



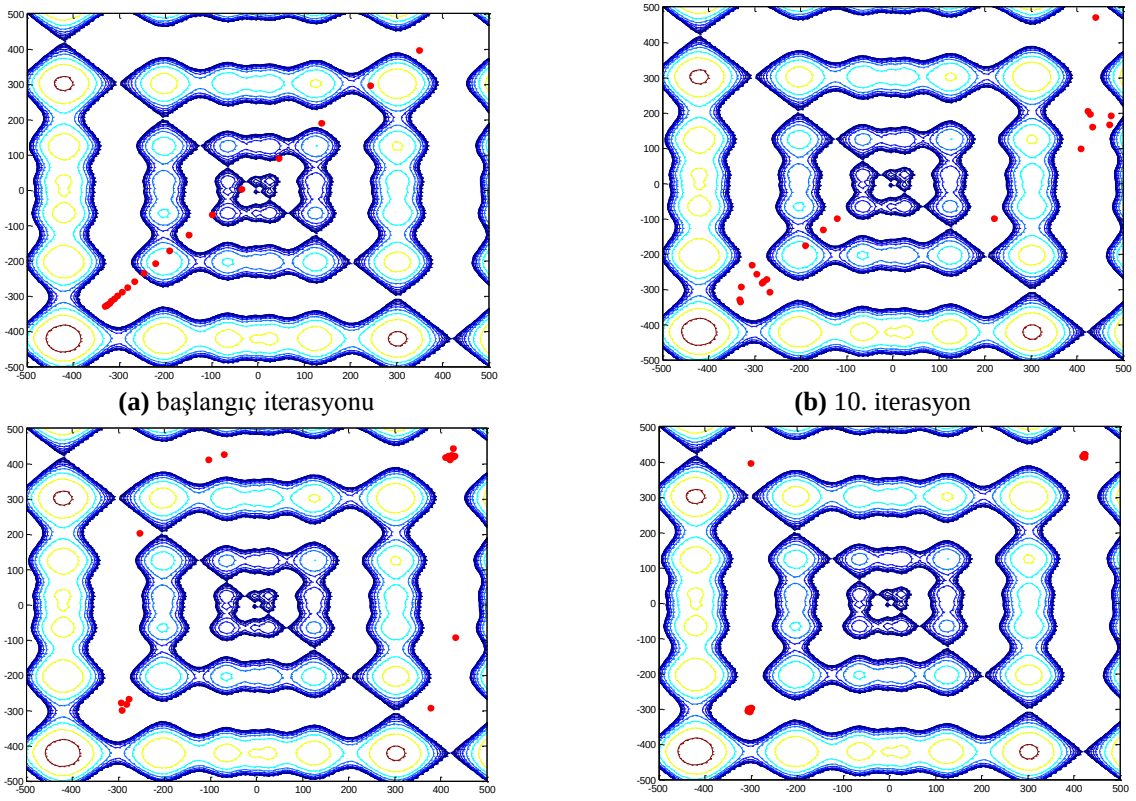
(a)



(b)

Şekil 5. Kaotik tabanlı DE algoritması ile yapılan optimizasyon sonuçları. (a) Ackley (b) Rastrigin [PS=20, SF=1.2, CR=0.7, itermax=200, Δ=1e-6, strateji:DE/rand-to-best/1]

Şekil 6'da Schweffel test fonksiyonu için önerilen kaotik tabanlı DE algoritması koşturulmuş ve farklı iterasyon adımlarında problem izdüşümü üzerinde anlık popülasyon değerleri gösterilmiştir. Şekilden de görüleceği üzere iterasyon adımları arttıkça çözüm uzayında popülasyondaki aday bireylerin çözüm olabilecek noktalara doğru yönelimi dikkat çekmektedir.



(a) başlangıç iterasyonu **(b)** 10. iterasyon

(c) 25. iterasyon **(d)** 40. iterasyon

Şekil 6. Schweffel optimizasyon test problemi için kaotik tabanlı DE algoritmasının farklı iterasyonlardaki popülasyonlarının değişimi [PS=20, SF=1.2, CR=0.7, itermax=200, Δ=1e-6, strateji:DE/rand-to-best/1]

Tablo 2'de aynı parametrelere sahip DE algoritması ve Kaotik DE algoritması (KDE) dört test fonksiyonu için iki farklı DE stratejisi ile on defa koşturulmuş ve bulunan en iyi değerlerin ortalamaları karşılaştırılmıştır. Tablodan anlaşılacağı gibi, her iki algoritmada fonksiyonların global minimum değerlerini bulmuşlardır. DE algoritması iki test fonksiyonunda önerilen KDE algoritmasından biraz daha başarılıdır.

Tablo 2. DE ve KDE Algoritmalarının Başarımlarının Karşılaştırılması

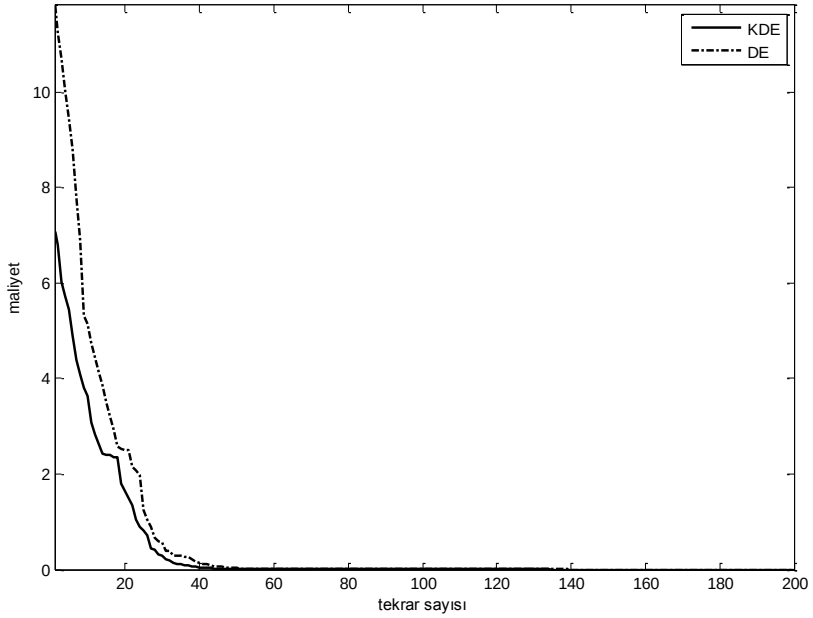
Fonksiyon İsmi	DE/rand-to-best/1		DE/rand/1		$f(x_1, x_2)$
	DE	KDE	DE	KDE	
Ackley	1e-6	1e-6	1e-6	1e-6	0
Rastrigin	**103e-6**	589e-6	**1e-6**	995e-4	0
Rosenbrock	1e-6	1e-6	1e-6	1e-6	0
Schweffel	**-837.965**	-832.229	**-837.965**	-743.215	-837.965

Tablo3'de ise her iki algoritmanın toplam iterasyon sayılarına göre karşılaştırılması yapılmış ve KDE algoritmasının genel olarak DE algoritmasından daha hızlı olduğu görülmüştür.

Tablo 3. DE ve KDE Algoritmalarının İterasyon Sayılarının Karşılaştırılması

Fonksiyon İsmi	DE/rand-to-best/1		DE/rand/1		$f(x_1, x_2)$
	DE	KDE	DE	KDE	
Ackley	126.1	**115**	146.7	**135.3**	0
Rastrigin	198.2	**182.2**	**112.6**	134.8	0
Rosenbrock	137.6	**68.4**	163	**79.9**	0
Schweffel	200	200	200	200	-837.965

Şekil 7'de Ackley fonksiyonu için her iki algoritma onar defa koşturulmuş ve ortalama değerleri gösterilmiştir. Sürekli çizgi ile gösterilen KDE algoritmasının DE algoritmasından daha önce global minimum noktasına ulaştığı şekilden açıkça anlaşılmaktadır.



Şekil 7. Ackley optimizasyon test problemi için KDE ve DE algoritmalarının onar defa koşturulmasının karşılaştırılması [PS=20, SF=1.2, CR=0.7, itermax=200, Δ=1e-6, strateji:DE/rand-to-best/1]

5. Sonuçlar

Bu çalışmada farksal gelişim algoritmasının temelindeki rastgele sayı üreteci yerine kaotik sistemlerden Lorenz Çekicisi temelli bir sayı üreteci geliştirilmiştir. Bu sayı üreteci ile DE algoritması içindeki iterasyonlarda rastgele sayı üretimi yapılmıştır. Önerilen algoritmanın performansının değerlendirilmesi için dört optimizasyon test fonksiyonu kullanılmış, ayrıca önerilen KDE algoritmasının klasik DE algoritması ile karşılaştırılması sunulmuştur. Yapılan benzetim çalışmalarından, önerilen KDE algoritmasının DE algoritmasına göre hızlı olduğu, başarım olarak birbirlerine yakın oldukları görülmüştür.

Kaynaklar

[1] Yüzgeç U., Performance comparison of differential evolution techniques on optimization of feeding profile for an industrial scale fed-batch baker's yeast fermentation process, ISA Transactions, vol. 49, no. 1, pp. 167-176, January 2010.

[2] Yüzgeç U., Kendiliğinden Uyarlanabilir Karşıtlık Tabanlı Farksal Gelişim Algoritması, Bilecik Şeyh Edebali Üniversitesi BAP Sonuç Raporu, 2011.

[3] Price, K., & Storn, R. (1997). Differential evolution: A simple evolution strategy for fast optimization. Dr. Dobb's J. Software Tools, 22 (4), 18-24.

[4] Storn, R., & Price, K. (1997). Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, 11, 341-359.

[5] Price, K. (1999). An Introduction to Differential Evolution, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79–108, ISBN: 007-709506-5, New Ideas in Optimization.

[6] Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput., 10(6), 646-657.

[7] Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. in Proc. Int. Conf. Comput. Intell. Modeling Control and Autom., Vienna, Austria, 695-701.

[8] Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2006). Opposition-based differential evolution algorithms. In Proc. IEEE Congr. Evol. Comput., Vancouver, BC, Canada, 2010-2017.

[9] Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2008). Opposition-based differential evolution. IEEE Trans. Evol. Comput., 12(1), 64-79.

[10] Liu, J., & Lampinen, J. (2002). On setting the control parameter of the differential evolution method. In Proc. 8th Int. conf. Soft Computing (MENDEL 2002), 11-18.

[11] Pamuk, N., Dinamik Sistemlerde Kaotik Zaman Dizilerinin Tespiti, BAÜ Fen Bil. Enst. Dergisi Cilt 15(1) 77-91 (2013).

[12] E.N. Lorenz, "Deterministic nonperiodic flow", J. Atmo.Sci., vol. 20, pp. 130-141, 1963.

[13] O.E. Rossler, "An equation for continuous chaos", Phys. Lett. A, vol 57, pp. 397-398, 1976.

[14] O.E. Rossler , "An equation for hyperchaos", Phys. Lett. A, vol 71, pp. 155-157, 1979.