BMBAD BJESR Mühendislik Bilimleri ve Araştırmaları Dergisi
Journal of Engineering Sciences and Researches

CEC 2019 Problemleri İçin Kendinden Uyarlamalı Bir Diferansiyel Gelişim Algoritması

A Self-Adaptive Differential Evolution Algorithm for CEC 2019 Problems

[1]Hatem DUMLU, [2]Gürcan YAVUZ

[1]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Kütahya, Türkiye
[2]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Kütahya, Türkiye

[1]hatem.dumlu@dpu.edu.tr, [2]gurcan.yavuz@dpu.edu.tr

Araştırma Makalesi/Research Article

ARTICLE INFO

Article history

Received : 8 June 2023
Accepted : 6 September 2023

Keywords:

Self-Adaptive Differential Evolution Algorithm, CEC 2019, Differential Evolution Algorithm, Optimization

ABSTRACT

In this study, a variant of the Differential Evolution algorithm enhanced for global optimization has been proposed. It differentiates from the original Differential Algorithm in terms of the mutation step. This variant, named KU-DGA, operates with a large number of randomly selected mutation equations. The performance of the KU-DGA algorithm has been tested using the CEC 2019 benchmark suite. The results obtained by the algorithm have been compared with WOAmM (Wider Optimization with Artificial Whales), WOA (Whale Optimization Algorithm), MFO (Moth Flame Optimization Algorithm), BOA (Butterfly Optimization Algorithm), SCA (Sine Cosine Algorithm), and JAYA algorithms in the literature. The obtained results have been analyzed using the Friedman test. As a result, in terms of the "average" value results, the proposed KU-DGA algorithm has outperformed its competitors in seven out of ten functions included in CEC 2019. Furthermore, in the "best" results category, the proposed variant has achieved the most successful outcomes in seven out of ten functions, surpassing rival algorithms.

MAKALE BİLGİSİ

Makale Tarihleri

Gönderim : 8 Haziran 2023
Kabul : 6 Eylül 2023

Anahtar Kelimeler:

Kendinden Uyarlamalı Diferansiyel Gelişim Algoritması, CEC 2019, Diferansiyel Gelişim Algoritması, Optimizasyon

ÖZET

Bu çalışmada, Global optimizasyon için geliştirilmiş bir Diferansiyel Gelişim algoritması varyantı önerilmiştir. Orijinal Diferansiyel Algoritmasından mutasyon adımı ile farklılaşmaktadır. KU-DGA adını verdiğimiz bu varyant rastgele çok sayıda belirlenmiş mutasyon denklemleri ile çalışmaktadır. KU-DGA algoritmasının başarımı CEC 2019 ölçüt seti ile test edilmiştir. Algoritmanın elde ettiği sonuçlar literatürdeki WOAmM (Genişletilmiş WOA), WOA (Balina Optimizasyon Algoritması), MFO (Güve Alevi Optimizasyon Algoritması), BOA (Kelebek Optimizasyon Algoritması), SCA (Sinüs Kosinüs Algoritması) ve JAYA algoritmaları ile karşılaştırılmıştır. Elde edilen sonuçlar Friedman testi ile analiz edilmiştir. Sonuç olarak önerilen KU-DGA algoritmasının karşılaştırılan algoritmalara kıyasla "ortalama" değer sonuçları baz alındığında CEC 2019' da yer alan on fonksiyonun yedisinde rakiplerini geçmiştir. Ayrıca önerilen varyant "en iyi" sonuçlarda da on fonksiyonun yedisinde en başarılı sonuçları alarak rakip algoritmaları geride bırakmıştır.

ORCID ID: [1]0000-0002-9056-4437
[2]0000-0002-2540-1930

1. GİRİŞ

Son yıllarda optimizasyon algoritmaları büyük gelişme göstermiştir. Bu optimizasyon algoritmalarının bir kısmını da Evrimsel Algoritmalar temsil eder. Doğadan ilham alan mekanizmaları kullanan ve canlı organizmaların davranışlarını taklit eden süreçleri kapsayarak sorunları çözen Evrimsel Algoritmalar, optimizasyon algoritmaları arasında büyük öneme sahiptir. Evrimsel algoritmalardan birisi de Diferansiyel Gelişim (DE) algoritmasıdır.

DE algoritması güçlü bir meta sezgisel algoritma olmasına rağmen algoritma yerel optimuma takılma gibi¸ çeşitli problemleri de mevcuttur. Bu problemlerin üstesinden gelebilmek için araştırmacılar, DE algoritmasının adımlarına¸ çeşitli iyileştirmeler önermişlerdir. İyileştirme yapılan adımlardan birisi de DE algoritmasının mutasyon adımıdır. Araştırmacılar bu adımda yeni mutasyon operatörü önererek algoritmada iyileştirmeye gitmişler ve daha iyi sonuçlar elde etmişlerdir.

Meng vd. iki Aşamalı Diferansiyel Gelişim algoritması (Two-Stage Differential Evolution (TDE)) adını verdikleri varyantlarında DE evrimsel sürecini iki safhaya ayırmışlardır [1]. Bu iki safha birbirlerinden farklı mutasyon stratejilerine sahiptir. Bunlara ek olarak algoritmalarında Kesirsel Ölçek F (Scaling Factor F- Kesirsel Ölçek) ve Çaprazlama Katsayısı (Crossover Rate CR- Çaprazlama Katsayısı) parametrelerini belirlemek için kontrol stratejileri önermişlerdir. Algoritmalarının performansını CEC 2013, CEC 2014 ölçüt setlerinde test etmişlerdir. Zhang vd. ise" DE/current-to-pbest" mutasyon stratejisini kullanan ve F ve CR parametrelerini Cauchy ve Gaussian dağılım ile belirleyen JADE algoritmasını geliştirmişler ve birçok algoritmanın temelini oluşturmuşlardır [2]. Tanabe ve Fukunaga JADE algoritmasını temel alan ve tarihsel bellek depolama mekanizmasını parametre belirleme yöntemi olarak kullanan DE varyantı olan SHADE algoritmasını sunmuşlardır [3]. Daha sonra, SHADE algoritmasının popülasyon boyutunu zamana göre azaltılmasını sağlayan stratejiyi ekleyerek L-SHADE algoritmasını önermişlerdir [4]. Houssein vd. popülasyonu çeşitlendirilmesi için üç tane mutasyon mekanizması ile uyarlamalı parametre yöntemini DE algoritmasına entegre ederek Uyarlanabilir Güdümlü Diferansiyel Gelişim (Adaptive Guided Differential Evolution) Algoritmasını mühendislik problemlerine uygulamıştır [5]. Deng vd. Komşuluk Mutasyonu (Neighborhood Mutation - Komşuluk Mutasyonu) mekanizmasını DE algoritmasına eklemişlerdir [6]. Tan vd. Fitness Landscape yaklaşımını temel alan uyarlamalı mutasyon stratejisini DE algoritmasına dahil etmişlerdir [7].

Araştırmacılar bir diğer iyileştirme kategorisi olarak ise mutasyon denklem havuzu yaklaşımını önermişlerdir. Örneğin Qin vd., Self-Adaptive DE (SaDE) adını verdikleri "DE/rand/1", "DE/rand-to-best/2", "DE/rand/2" ve "DE/current-to-rand/1" mutasyon stratejilerini önceki başarılarına göre bir havuzdan seçen bir DE varyantı önermişlerdir [8]. Wang vd., large scale feature selection problemi için multi-popülasyon stratejisini, F ve CR parametrelerini dinamik olarak belirlemek için kendinden uyarlamalı (Self-Adaptive - Kendinden Uyarlamalı) yöntemini, mutasyon denklemi için sekiz adet mutasyon denklemi ile doldurulmuş olan havuz stratejisini ve son olarak özellik belirlemek için ağırlıklandırılmış model (Weighted Model – Ağırlıklandırılmış Model) yöntemini DE'ye uyarlayarak bir varyant önermişlerdir [9]. Ayrıca Wang vd., üç parametre ayarı ve üç mutasyon stratejisini içeren aday havuzuna sahip bir DE varyantı (Composite DE (CoDE)) önermişlerdir [10]. Bu algoritmaya göre, üç parametre ayarı ve mutasyon stratejileri rastgele seçilerek belirlenmektedir.

Literatürde çok sayıda çok sayıda metasezgisel algoritma yer almaktadır. Algoritmayı öneren araştırmacılar algoritmalarını test etmek için ölçüt setlerine başvurmaktadırlar. Bu ölçüt setlerinden bir tanesi de IEEE evrimsel algoritmalar 2019 oturumuna ait CEC 2019 ölçüt setidir. Bu ölçüt seti de literatürde yoğun olarak kullanılmaktadır. Örneğin; Sağ Vortex arama algoritması, Sulaiman vd. evrimsel çiftleşme algoritması, Ekinci vd. otomatik voltaj regülatörü için bir aquila optimizasyon algoritması önermişler ve algoritmalarının başarımını CEC 2019 ölçüt seti ile test etmişlerdir [11] [12] [13]. Duan ve Yu mühendislik problemleri için önerdikleri hibrit Gri kurt-Sinüs Cosinüs algoritmasının mühendislik problemlerine uygulamadan önce CEC 2019 ölçüt setinde algoritmanın başarımını test etmişlerdir [14]. Yine Shen vd. gerçek dünya problemlerini çözmeden önce ilk olarak algoritmalarını CEC 2019 ile sınamışlardır [15].

Bu çalışmada, Diferansiyel Gelişim Algoritmasının arama gücünü oluşturan mutasyon adımı için farklı bir yaklaşım önerilmektedir. Orijinal DE algoritmasında mutasyon adımı algoritma tasarlanırken belirlenmektedir ve evrimsel arama sürece devam ettiği müddetçe bu adım değiştirilmez aynı kalır. Algoritmanın çözüm arama yeteneklerini önemli ölçüde etkileyen bu adımın sürekli aynı denklemi kullanarak değişmemesi algoritmanın farklı problem türleri ile karşılaştığında yetersiz kalmasına yol açmaktadır. Önerilen iyileştirme ile DE'nin sahip olduğu bu yetersizliğin üstesinden gelinmeye çalışılmaktadır. Bu iyileştirme, tek bir sabit mutasyon denklemi yerine içerisinden çok sayıda rastgele üretilmiş bir denklem havuzundan seçilmesine dayanan bir yaklaşıma sahiptir. Böylelikle algoritma karşılaşabileceği farklı türde problemleri havuzdan elde edeceği mutasyon denklemleri ile aşabilecektir. Önerilen algoritma CEC 2019 ölçüt seti ile başarımı test edilmiş olup literatürde yer alan çeşitli metasezgisel algoritmalar sonuçları ile karşılaştırılmıştır.

Çalışmanın geri kalanı şu şekilde düzenlenmiştir: Bölüm 2'de Orijinal Diferansiyel Gelişim ve önerilen algoritmanın detayları ile deneylerde kullanılan CEC 2019 ölçüt seti özelliklerini içeren Materyal ve Metot kısmı verilmiştir. Bölüm 3'de ise önerilen algoritma ve literatürde yer alan algoritmaların CEC 2019 sonuçları listelenmiştir. Bölüm 4 ile çalışma özetlenmiştir.

2. MATERYAL VE METOT

Bu bölümde, ilk olarak Orijinal Diferansiyel Gelişim Algoritmasına değinilecek ve daha sonra da önerilen algoritma detaylandırılacaktır. Bölümün sonunda da deneylerde kullanılan CEC 2019 ölçüt setinin özelliklerinden bahsedilecektir.

2.1. Diferansiyel Gelişim Algoritması

Diferansiyel Gelişim Algoritması (DE), R. Storn ve K. Price tarafından optimizasyon problemlerinin çözülmesi için önerilmiş olan popülasyon tabanlı bir algoritmadır [16], [17]. Orijinal DE algoritması, ilklendirme, mutasyon, çaprazlama ve seçme adımlarından oluşmaktadır [18] ve sözde kodu Algoritma 1'de verilmiştir.

Algoritma 1: Diferansiyel Gelişim Algoritması sözde kodu

```
1: İlklendirme adımı
2:  iter ← 0
3: while Sonlandırma Kriteri Karşılanmaz İse do
4:       Mutasyon adımı
5:       Çaprazlama adımı
6:       Seçme adımı
7:       iter ← iter + 1
```

2.1.1. İlklendirme

Mühendislik ve DE algoritması optimizasyon işlemine, D boyutlu çözümleri rastgele üreterek Np boyutlu popülasyon ile başlar. Bu popülasyona ilklendirme popülasyonu adı verilir ve aşağıda yer alan denklem ile üretilmektedir:

$$x_{iJ} = x_L + rand(x_U - x_L) \quad (1)$$

Burada x_U ve x_L problemin en büyük ve en küçük sınır değerlerini göstermektedir. rand, tekdüze dağılım kullanılarak [0,1] aralığından üretilmiş olan rastgele bir sayıdır.

2.1.2. Mutasyon

Her iterasyonda t'de, her birey X_i'ye -hedef birey olarak tanımlanır- mutasyon operatörü adı verilen bir işlem uygulanır. Bu işlemin sonucunda hedef bireye ait olan bir mutant birey adı verilen bir birey V_i üretilir. Bunun için literatürde araştırmacıların önerdiği çok sayıda mutasyon operatörü mevcuttur. Bunlardan bazıları aşağıdaki gibidir [19]:

DE\rand\1:

$$V_i = X_{r1} + F(X_{r2} - X_{r3}) \quad (2)$$

DE\rand\2:

$$V_i = X_{r1} + F(X_{r2} - X_{r3} + X_{r4} - X_{r5}) \quad (3)$$

2.1.3. Çaprazlama

Çaprazlama katsayısı (CR-Crossover Rate), geçici çözümü uygulamak için kullanılır.

$$u_{ij} = \begin{cases} V_{ij}, & if\ rand_j < CR\ or\ j == j_{rand} \\ X_{ij}, & diğer\ durumda \end{cases} \quad (4)$$

u_{ij}, i. geçici¸ çözümü temsil etmektedir. Bu çözüm, X_i ve V_i'nin çaprazlanması sonucunda oluşmaktadır. CR çaprazlama oranı olarak bilinmektedir ve algoritmanın¸ çalışmasının başında kullanıcı tanımlı olarak belirlenen bir parametredir. Geçici çözümün oluşumunda hangi elemanın geleceğinin belirlenmesinde kullanılmaktadır. j_{rand} , D boyutu göstermek üzere $[1, D]$ aralığından rastgele seçilen bir sayıdır.

2.1.4. Seçme

Diferansiyel Gelişim algoritmasındaki son adımdır. Bu adımda üretilmiş olan geçici çözüm ile mevcut çözüm arasında karar verilmektedir. Hangi çözüm daha iyi ise bir sonraki iterasyondaki popülasyona dahil olur.

$$X_i^{t+1} = \begin{cases} X_i^t, & if\ \ f(X_i^t) < f(u_i^t) \\ X_i^t, & diğer\ durumda \end{cases} \quad (5)$$

2.2. Kendinden Uyarlamalı Diferansiyel Gelişim Algoritması (Ku-Dga)

Kendiden denklem uyarlamalı bir diferansiyel Gelişim Algoritması önerilmiştir ve önerilen algoritmanın sözde kodu Algoritma 2'de verilmiştir.

DE'nin performansı sahip olduğu mutasyon, çaprazlama ve seçme adımlarına bağlıdır. Bu adımlar birbirini takip eden adımlar olduğu için özellikle mutasyon adımının başarısı diğerlerini doğrudan etkilemektedir. Bu nedenle, burada seçilen mutasyon operatörü algoritmanın en önemli safhasını oluşturmaktadır. Bu çalışmada, literatürde yer alan yaklaşımlar tek mutasyon operatöründen veya birkaç mutasyon operatöründen oluşan yaklaşımın aksine terim bazlı oluşturulan kendinden uyarlamalı bir yöntem tercih edilmiştir.

Bu çalışmada, terim bazlı olarak operatörler oluşturularak mutasyon havuzuna MP) doldurulmaktadır. Bu operatörler, Algoritma 3 [20], [21], şablonu kullanılarak algoritma başlangıcında rastgele üretilmektedir. Bu şablon kullanılarak üretilen denklemler bir ile dört terimli olabilmektedir. Her bir terim Tablo 1'den [20] ilgili terim sütunundaki seçeneklerden bir tanesi rastgele olarak seçilmektedir. Bu denklem üretilme işlemi, algoritma başlangıcında belirtilen denklem sayısı kadar yapılır. Üretilen her bir denkleme karşılık, denklemin kalitesini gösteren bir başarı sayısı atanır. Mutasyon stratejisi sözde kodu Algoritma 4' de gösterilmiştir.

Algoritma 2: KU-DGA sözde kodu

```
Algoritma parametrelerini belirle (NP, Ps, maxFes, MPS)
MP denklem havuzunu rastgele denklemler ile doldur.
OldBestArchive ← 1
equationCounter ← 1
while Sonlandırma Kriteri Karşılanmaz İse do
    for i = 1 to NP do
        KendindenUyarlamaliMutasyonStratejisiniUygula(CR, i,EquCount)
    EquCount ← EquCount + 1
    if EquCount == MPS then
    Başarı oranlarınına göre denklemleri sırala
        MP havuzunun boyutunu küçült.
        EquCount ← 1
     if OldBestArchive > NP then
        OldBestArchive arşivindeki çözümleri amaç fonksiyonuna göre sırala
        İlk Np kadarını sakla. Geri kalan bireyleri havuzdan çıkar.
     iter ← iter + 1
```

Algoritma 3: KU-DGA Mutasyon Operatörü Üretme Şablonu

$X_{i,j} = terim_1 + terim_2 + terim_3 + terim_4$

Tablo 1. Algoritmanın genelleştirilmiş arama denklemindeki her bir bileşen için alternatif seçenekler

$terim_1$	$terim_2$	$terim_3$	$terim_4$
$X_{i,j}$	$rand(-1, 1)(X_{best,j} - X_{i,j})$	$rand(-1, 1)(X_{best,j} - X_{i,j})$	$rand(-1, 1)(X_{best,j} - X_{i,j})$
	$rand(-1, 1)(X_{best,j} - X_{r1,j})$	$rand(-1, 1)(X_{best,j} - X_{r1,j})$	$rand(-1, 1)(X_{best,j} - X_{r1,j})$
	$rand(-1, 1)(X_{best,j} - X_{wo,j})$	$rand(-1, 1)(X_{best,j} - X_{wo,j})$	$rand(-1, 1)(X_{best,j} - X_{wo,j})$
$X_{G,j}$	$rand(-1, 1)(X_{r1,j} - X_{r2,j})$	$rand(-1, 1)(X_{r1,j} - X_{r2,j})$	$rand(-1, 1)(X_{r1,j} - X_{r2,j})$
$X_{r1,j}$	$rand(-1, 1)(X_{r1,j} - X_{i,j})$	$rand(-1, 1)(X_{r1,j} - X_{i,j})$	$rand(-1, 1)(X_{r1,j} - X_{i,j})$ $rand(-1, 1)(X_{pbestTop,j} - X_{i,j})$
	$rand(-1, 1)(X_{pbestTop,j} - X_{i,j})$	$rand(-1, 1)(X_{pbestTop,j} - X_{i,j})$	
	0	0	0

Evrimsel süreç boyunca her iterasyonda denklem havuzundan sırası ile rastgele üretilmiş olan bir denklem alınır ve mutasyon operatörü olarak kullanılır. Eğer ki bu denklem mevcut bireyi iyileştirmede başarılı olursa denklemin başarı değeri artırılır. Denklem havuzundaki bütün denklemler kullanıldıktan sonra, havuz içindeki denklemler başarı değerine göre sıralanır ve başarısız olan denklemler havuzdan çıkarılır. Böylelikle denklem havuzunun boyutu MPS küçültülür. Çalıştırma bütçesinin sonuna doğru denklem havuzunda çok az sayıda başarılı denklem kalır. Mutasyon denkleminin başarısı, başarılı olan güncellemesi ile hesaplanmaktadır.

$$MPS = \frac{MPS^2}{itr_{MAX}} \tag{6}$$

itr_{MAX} maksimum iterasyon sayısını göstermektedir. *MAXFES* maksimum fonksiyon çağrım sayısı $(itr_{MAX} \times NP)$

2.3. CEC 2019

CEC (Constrained Evolutionary Competition) ölçüt setleri, genetik algoritmalar ve evrimsel hesaplama alanında kullanılan bir test kümeleridir. CEC ölçüt setleri, optimizasyon problemlerinin çözülmesinde algoritmaların performansını değerlendirmek için standartlaştırılmış bir ortam sağlar. CEC 2019, 2019 yılında düzenlenen bir CEC yarışmasının bir parçası olarak oluşturulan, 100 basamak problemi olarak da bilinen bir ölçüt setidir [22]. Bu

ölçüt seti, karmaşık optimizasyon problemlerini temsil eden 10 adet test fonksiyonundan oluşur. Tablo 2'de CEC 2019 fonksiyonları listelenmiştir [23].

Algoritma 4: KU-DGA mutasyon stratejisi sözde kodu

1: **procedure** KendindenUyarlamaliMutasyonStratejisiniUygula(CR,i,EquCount)
2: **for** d=1 to Dim **do**

$$u_{i,j}^{t} = \begin{cases} MP(EquCount) & if(rand \leq CR \parallel d = rand[1,D]) \\ 0 & diğer\ durumda \end{cases}$$

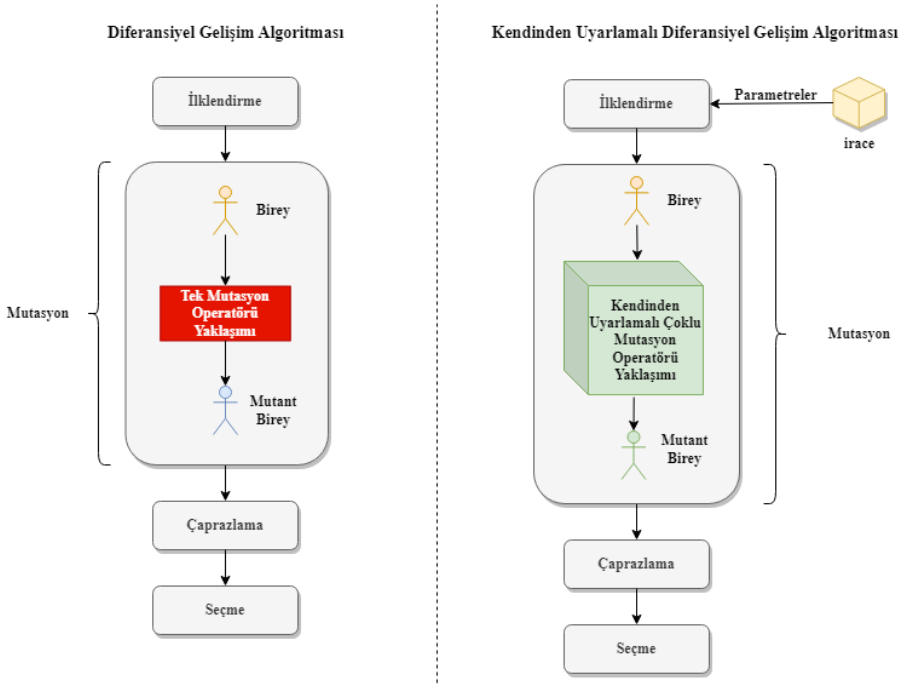
3: **if** $f(u^t) \leq f(x^t)$ **then**
4: $X_i^{t+1} = u_i^t$
5: $X_i^t \rightarrow OldBestArchive$
6: MP (EquCount) başarı artırma
7: **else**
8: $X_i^{t+1} = X_i^t$

Tablo 2. CEC 2019 ölçüt setindeki fonksiyonlar.

Fonksiyon No	Fonksiyonlar	Boyut	Aralık
1	Storn's Chebyshev Polynomial Fitting Problem	9	[-8192, 8192]
2	Inverse Hilbert Matrix Problem	16	[-16384, 16384]
3	Lennard-Jones Minimum Energy Cluster	18	[-4, 4]
4	Rastrigin's Function	10	[-100, 100]
5	Griewangk's Function	10	[-100, 100]
6	Weierstrass Function	10	[-100, 100]
7	Modified Schwefel's Function	10	[-100, 100]
8	Expanded Schaffer's F6 Function	10	[-100, 100]
9	Happy Cat Function	10	[-100, 100]
10	Ackley Function	10	[-100, 100]

3. DENEYSEL ÇALIŞMALAR

Bu bölümde, yapılan deneylerin sonuçları verilecektir. KU-DGA bir DE varyantı algoritmadır. KU-DGA, IEEE'nin 2019 Evrimsel Hesaplama konferansının ölçüt seti CEC 2019 [22], [23] ile algoritmanın performansı test edilmiş ve elde edilen sonuçlar literatürdeki diğer çalışmalar ile karşılaştırılmıştır. Ayrıca önerilen algoritmanın akış diyaramı Şekil 1'de verilmiştir.



Şekil 1. Önerilen algoritmanın akış diyagramı ve orijinal diferansiyel gelişim algoritması arasındaki farklar.

Şekil 1'de görüldüğü üzere orijinal diferansiyel gelişim algoritmasının mutasyon evresine yapılan değişiklik ile, tek mutasyon operatörü yaklaşımı önerilen algoritmada kendinden denklem uyarlamalı çoklu mutasyon operatörü ile değiştirilmiştir. Ayrıca ilklendirme aşamasında en iyi parametrelerle algoritmayı başlatmak için irace paramatre belirleme aracı kullanılmıştır. Yapılan değişiklik ve önerilen yaklaşım bölüm 2.2'de detaylandırılmıştır. Bu yaklaşımın mutasyon evresinde tek bir mutasyon operatörü yerine denklem havuzundan seçilen birden fazla mutasyon operatörü ile mutant bireyler elde edilmekte, en iyi mutant bireyleri elde eden denklemlerin skoru

artırılarak mutasyonda kullanılan mutasyon operatörlerinin en iyi operatörlerden seçilmesi sağlanmakta ve kötü denklemlerde denklem havuzundan atılarak en iyi denklemlerin kullanılmasının önünü açmaktadır. Böylece daha güçlü bir diferansiyel gelişim algoritması elde edilmiştir.

3.1. Irace

Önerilen KU-DGA'da seçilen parametreler algoritmanın performansını doğrudan etkilemektedir. Dolayısıyla bu parametre değerlerinin doğru seçilmesi uygulamanın performansını gerçekçi bir şekilde yansıtması için elzemdir. Bu sebeple deneyde kontrol parametrelerini belirlemek için bir otomatik yapılandırma aracı olan irace (iterative racing) kullanılmıştır [24]. Irace, bir R paketidir. Irace, parametre ayarlaması için yarışma (racing) tabanlı bir yaklaşım kullanır. Bu yaklaşım, bir dizi aday parametre konfigürasyonunu birbiriyle karşılaştırarak en iyi konfigürasyonu belirlemeyi amaçlar. Irace algoritması, iteratif bir şekilde aday parametre konfigürasyonlarını oluşturur, bu konfigürasyonları değerlendirir ve en iyi sonuçları veren konfigürasyonları bir sonraki iterasyona taşır. Bu işlem, birçok iterasyon sonunda en iyi parametre ayarlarını elde etmek için tekrarlanır.

3.2. Deneysel Ayarlar

Önerilen algoritma 50 popülasyon boyutu ve iterasyonu 1000 olacak şekilde CEC 2019 problemlerinde çalıştırılmıştır. Adil bir karşılaştırma yapılması için Chakraborty vd. çalıştırma koşullarına uygun olarak deneyler gerçekleştirilmiştir [25]. Popülasyon boyutu 50 ve iterasyonu 1000 olarak belirlenmiştir ve Chakraborty vd. yapmış olduğu çalışmanın sonuçları da deneylere dahil edilmiştir [25]. Ölçüt setinde yer alan her fonksiyon önerilen algoritma ile 30 defa çalıştırılarak her fonksiyon için 30 sonuç elde edilmiştir. Ortalama, standart sapma ve en iyi değerleri hesaplanmıştır. Önerilen KU-DGA algoritması için Çaprazlama Oranı 0.4 ve denklem havuzu boyutu 200 seçilmiştir. CEC 2019 fonksiyon deneyleri Ubuntu Linux 18.04 işletim sistemine sahip Intel Xeon E5 2670 işlemcili ve 64 GB hafızaya sahip bir bilgisayarda gerçekleştirilmiştir.

3.3. CEC 2019 Sonuçları ve Diğer Metasezgizeller ile Karşılaştırılması

Chakraborty vd. önerdikleri WOAmM algoritmasını JAYA, SCA, BOA MFO, WOA algoritmaları ile kıyaslamışlar ve önerdikleri WOAmM algoritmasının rakiplerine kıyasla üstün geldikleri sonuçları çalışmalarında göstermişlerdir [25]. Parametre değerleri olarak ilgili makalelerin [26]–[30] parametre değerlerini kullanmışlardır. Bizim önerdiğimiz KU-DGA algoritmasının sonuçları ve karşılaştırma metrikleri, Chakraborty vd. önerdiği WOAmM algoritması ve karşılaştırılan diğer algoritmaların sonuçları ile birleştirilip Tablo 3'de verilmiştir [25]. Buna ek olarak KU-DGA algoritması ve diğer algoritmaları sıralamak için istatistiksel test olarak Friedman testi kullanılmıştır. Elde edilen sonuçlar Tablo 4'te verilmiştir.

Tablo 3. Önerilen algoritma ile diğer algoritmaların CEC 2019 karşılaştırma sonuçları.

No			KU-DGA	WOAmM	WOA	MFO	BOA	SCA	JAYA
	Ortalama		$1.77x10^{6}$	1	$3.35x10^{6}$	1	1	1	$8.39x10^{6}$
F1	Std.	Sapma	$1.93x10^{6}$	≈0	$3.94x10^{6}$	0	$4.13x10^{-8}$	0	$3.12x10^{6}$
	En İyi		**$1.50x10^{-1}$**	1	$4.54x10^{2}$	1	1	1	$3.87x10^{6}$
	Ortalama		$4.17x10^{3}$	4.31	$7.71x10^{3}$	5	5.01	5	$5.35x10^{3}$
F2	Std.	Sapma	$9.59X10^{2}$	$9.04x10^{-2}$	$2.73x10^{3}$	0	$3.47x10^{-3}$	0	$9.02x10^{2}$
	En İyi		$2.12x10^{3}$	4.23	$3.56x10^{3}$	5	5	5	$3.71x10^{3}$
	Ortalama		4.01	2.88	4.11	7.43	4.77	6.12	9.24
F3	Std.	Sapma	1.6	1.11	1.82	$5.02x10^{-2}$	$9.38x10^{-1}$	1.45	$9.91x10^{-1}$
	En İyi		1.49	1.41	1.43	7.33	3.28	4.41	6.40
	Ortalama		**$2.57x10^{1}$**	$3.42x10^{1}$	$5.04x10^{1}$	$7.80x10^{1}$	$9.88x10^{1}$	$7.52x10^{1}$	$3.71x10^{1}$
F4	Std.	Sapma	**4.05**	8.42	$2.09x10^{1}$	7.84	$1.17x\ 10^{1}$	$1.52x10^{1}$	5.05
	En İyi		$1.77x10^{1}$	$1.89x10^{1}$	$1.30x10^{1}$	$7.11x10^{1}$	$7.85x10^{1}$	$4.69x10^{1}$	$2.61x10^{1}$
	Ortalama		**$7.16x10^{-1}$**	1.60	2.05	$5.48x10^{1}$	$1.18x10^{2}$	$1.49x10^{1}$	2.70
	Std	Sapma	**$2.06x10^{-1}$**	$2.48x10^{-1}$	$4.55x10^{-1}$	$1.09x10^{1}$	$2.19x10^{1}$	9.61	$2.38x10^{-1}$
F5	En İyi		**$3.26x10^{-2}$**	1.25	1.45	$3.79x10^{1}$	$7.42x10^{1}$	4.33	2.22
	Ortalama		**3.44**	5.55	8.13	$1.05x10^{1}$	$1.12x10^{1}$	8.62	7.67
F6	Std.	Sapma	$8.64x10^{-1}$	1.56	1.83	$3.87x10^{-1}$	1.22	1.54	1.06
	En İyi		**$6.50x10^{-1}$**	3.33	3.85	$1.04x10^{1}$	6.66	6.54	5.81
	Ortalama		**$7.78x10^{2}$**	$1.14x10^{3}$	$1.30x10^{3}$	$1.69x10^{3}$	$2.81x10^{3}$	$2.6x10^{3}$	$1.22x10^{3}$
F7	Std.	Sapma	**$1.62x10^{2}$**	$2.62x10^{2}$	$3.5x10^{2}$	$1.81x10^{2}$	$2.07x10^{2}$	$3.88x10^{2}$	$1.82x10^{2}$
	En İyi		**$3.69x10^{2}$**	$4.9x10^{2}$	$5.42x10^{2}$	$1.59x10^{3}$	$1.65x10^{3}$	$1.44x10^{3}$	$9.03x10^{2}$
	Ortalama		**2.99**	4.19	4.53	4.49	4.96	5.16	4.45
F8	Std.	Sapma	$2.33x10^{-1}$	$3.67x10^{-1}$	$3.29x10^{-1}$	$4.66x10^{-4}$	$2.08x10^{-1}$	$1.57x10^{-1}$	$1.58x10^{-1}$
	En İyi		**2.49**	3.23	3.93	4.49	4.42	4.74	4.08
	Ortalama		**$3.26x10^{-1}$**	1.34	1.36	2.32	4.52	1.93	1.6
F9	Std.	Sapma	**$6.29x10^{-2}$**	$1.2x10^{-1}$	$1.78x10^{-1}$	$9.3x10^{-1}$	$4.62x10^{-1}$	$2.15x10^{-1}$	$1.17x10^{-1}$
	En İyi		**$1.31x10^{-1}$**	1.06	1.11	1.52	3.23	1.48	1.36
	Ortalama		**$2.01x10^{1}$**	$2.11x10^{1}$	$2.12x10^{1}$	$2.13x10^{1}$	$2.15x10^{1}$	$2.20x10^{1}$	$2.14x10^{1}$
F10	Std.	Sapma	**$4.24x10^{-2}$**	$7.36x10^{2}$	$1.05x10^{-1}$	$8.55x10^{-2}$	$8.73x10^{-2}$	$1.78x10^{-1}$	$8.64x10^{-2}$
	En İyi		**$2.00x10^{1}$**	$2.1x10^{1}$	$2.1x10^{1}$	$2.13x10^{1}$	$2.13x10^{1}$	$2.16x10^{1}$	$2.12x10^{1}$

Önerilen KU-DGA'nın rakiplerinden başarılı olan değerleri koyu olarak yazılmıştır.

Tablo 4. Friedman testi ile genel sıralama.

No	KU-DGA	WOAmM	WOA	MFO	BOA	SCA	JAYA
F1	5	2.5	6	2.5	2.5	2.5	7
F2	5	1	7	2.5	4	2.5	6
F3	2	1	3	6	4	5	7
F4	1	2	4	6	7	5	3
F5	1	2	3	6	7	5	4
F6	1	2	4	6	7	5	3
F7	1	2	4	5	7	6	3
F8	1	2	5	4	6	7	3
F9	1	2	3	6	7	5	4
F10	1	2	3	4	6	7	5
Ortalama sıralama	1.9	1.9	4.2	4.8	5.8	5.0	4.5

Önerilen KU-DGA algoritması Tablo 3'de görüldüğü üzere CEC 2019 fonksiyonlarının "ortalama" sonuçları baz alındığında F1, F2, F3 fonksiyonları hariç diğer tüm fonksiyonlarda rakiplerini geride bırakarak birinci sırada gelmiştir. "En iyi" sonuçlarda F2, F3, F4 fonksiyonları hariç tüm fonksiyonlarda önerilen algoritma KU-DGA, rakiplerini geride bırakarak birinci olmuştur. "En iyi" ve "ortalama" sonuçlarda toplamda CEC 2019 on test fonksiyonunun yedisinde, önerilen algoritma birinci gelmiştir. Tablo 4 incelendiğinde KU-DGA algoritmasının sonuçları karşılaştırıldığı algoritmalara göre oldukça rekabetçidir. F4-F10 aralığında bütün algoritmaların önünde yer aldığı görülmektedir. Önerilen algoritmanın elde ettiği ortalama sıralama değerini ilk iki test fonksiyonunda aldığı kötü sıralamaların etkilediği görülmektedir.

4. SONUÇLAR VE TARTIŞMA

Bu çalışmada yeni bir algoritma önerilmiştir. Önerilen algoritma bir DE varyantı olup, farklı bir mutasyon stratejisi uygulanmış, CEC 2019 fonksiyonlarında test edilmiştir. Ayrıca literatürde yer alan diğer metasezgizeller ile sonuçlar karşılaştırılmıştır. Önerilen algoritma KU-DGA diğerlerine göre daha zor olan toplamda yedi fonksiyonda (F4-F10) "ortalama" sonuçlarda rakiplerini geride bırakmıştır. Ayrıca "en iyi" sonuçlarda CEC 2019'un on fonksiyonun yedisinde en başarılı sonuca sahiptir. KU-DGA algoritması CEC 2019 fonksiyonları kendi sonuçları içerisinde karşılaştırıldığında en başarılı "ortalama" değerini F9 test fonksiyonunda, en başarılı "en iyi" değerini F5 fonksiyonunda almıştır.
Sonraki çalışmalarda KU-DGA algoritmasına yeni stratejiler eklenerek performansı artırılabilir. Farklı DE varyantları ve diğer meta sezgisel algoritmalarla da karşılaştırılabilir.

Yazar Katkıları

Yazarlar makaleye eşit derecede katkı sağlamıştır.

Çıkar Çatışması

Makale yazarları aralarında herhangi bir çıkar çatışması olmadığını beyan ederler

KAYNAKÇA

[1] Z. Meng and C. Yang "Two-stage differential evolution with novel parameter control," Inf Sci (NY), vol. 596, pp. 321–342, 2022.

[2] J. Zhang and A. C. Sanderson "JADE: adaptive differential evolution with optional external archive," IEEE Transactions on evolutionary computation, vol. 13, no. 5, pp. 945–958, 2009.

[3] R. Tanabe and A. Fukunaga "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in 2013 IEEE Congress on evolutionary computation, IEEE, 2013, pp. 1952–1959.

[4] R. Tanabe and A. S. Fukunaga "Improving the search performance of SHADE using linear population size reduction," in 2014 IEEE congress on evolutionary computation (CEC), IEEE, 2014, pp. 1658–1665.

[5] E.H. Houssein, H. Rezk, A. Fathy, M. A. Mahdy, and A. M. Nassef "A modified adaptive guided differential evolution algorithm applied to engineering applications," Eng Appl Artif Intell, vol. 113, p. 104920, 2022.

[6] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu "An improved differential evolution algorithm and its application in optimization problem," Soft comput, vol. 25, pp. 5277–5298, 2021.

[7] Z. Tan, K. Li, and Y. Wang, "Differential evolution with adaptive mutation strategy based on fitness landscape analysis," Inf Sci (NY), vol. 549, pp. 142–163, 2021.

[8] A.K. Qin, V.L. Huang, and P.N. Suganthan "Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE transactions on Evolutionary Computation, vol. 13, no. 2, pp. 398–417, 2008.

[9] X. Wang, Y. Wang, K.-C. Wong, and X. Li "A self-adaptive weighted differential evolution approach for large-scale feature selection," Knowl Based Syst, vol. 235, p. 107633, 2022.

[10] Y. Wang, Z. Cai, and Q. Zhang "Differential evolution with composite trial vector generation strategies and control parameters," IEEE transactions on evolutionary computation, vol. 15, no. 1, pp. 55–66, 2011.

[11] T. Sağ "PVS: a new population-based vortex search algorithm with boosted exploration capability using polynomial mutation", Neural Comput and Applic, vol. 34, pp. 18211-18287, 2022.

[12] S. Ekinci, D. Izci, E. Eker, and L. Abualigah "An effective control design approach based on novel enhanced aquila optimizer for automatic voltage regulator", Springer Netherlands, vol. 56, no. 2, 2023.

[13] M.H. Sulaiman, Z. Mustaffa, M.M. Saari, H. Daniyal, and S. Mirjalili "Evolutionary mating algorithm," Neural Comput Appl, vol. 35, no. 1, pp. 487–516, 2023.

[14] Y. Duan and X. Yu "A collaboration-based hybrid GWO-SCA optimizer for engineering optimization problems," Expert Syst Appl, vol. 213, no. PB, p. 119017, 2023.

[15] B. Shen, M. Khishe, and S. Mirjalili "Evolving Marine Predators Algorithm by dynamic foraging strategy for real-world engineering optimization problems," Eng Appl Artif Intell, vol. 123, p. 106207, 2023.

[16] R. Storn, "Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report," International Computer Science Institute, vol. 11, 1995.

[17] R. Storn and K. Price "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, no. 4, p. 341, 1997.

[18] S. Das and P. N. Suganthan "Differential evolution: A survey of the state-of-the-art," IEEE transactions on evolutionary computation, vol. 15, no. 1, pp. 4–31, 2010.

[19] C.-W. Chiang, W.-P. Lee, and J.-S. Heh "A 2-Opt based differential evolution for global optimization," Appl Soft Comput, vol. 10, no. 4, pp. 1200–1207, 2010.

[20] D. Aydın, G. Yavuz, and T. Stützle "ABC-X: a generalized, automatically configurable artificial bee colony framework," Swarm Intelligence, vol. 11, pp. 1–38, 2017.

[21] G. Yavuz and D. Aydın "Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy," Swarm Evol Comput, vol. 51, p. 100582, 2019.

[22] K.V. Price, N.H. Awad, M.Z. Ali, and P.N. Suganthan "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," in Technical Report, Nanyang Technological University Singapore, 2018.

[23] G. Yavuz "100 Basamak Probleminin JADE Algoritması ile Çözülmesi," Avrupa Bilim ve Teknoloji Dergisi, no. 21, pp. 493–500, 2021.

[24] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle "The irace package: Iterated racing for automatic algorithm configuration," Operations Research Perspectives, vol. 3, pp. 43–58, 2016.

[25] S. Chakraborty, A. K. Saha, S. Sharma, S. Mirjalili, and R. Chakraborty "A novel enhanced whale optimization algorithm for global optimization," Comput Ind Eng, vol. 153, p. 107086, 2021.

[26] R. Rao "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," International Journal of Industrial Engineering Computations, vol. 7, no. 1, pp. 19–34, 2016.

[27] S. Mirjalili "SCA: a sine cosine algorithm for solving optimization problems," Knowl Based Syst, vol. 96, pp. 120–133, 2016.

[28] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," Soft comput, vol. 23, pp. 715–734, 2019.

[29] S. Mirjalili "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," Knowl Based Syst, vol. 89, pp. 228–249, 2015.

[30] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Advances in engineering software, vol. 95, pp. 51–67, 2016.