INTERNATIONAL SCIENTIFIC AND VOCATIONAL JOURNAL (ISVOS JOURNAL)

Vol.: 6 Issue: 2 Date: 31.12.2022 Received: 14.12.2022 Accepted: 30.12.2022 Final Version: 31.12.2022

ISVOS Journal, 2022, 6(2): 144-155 – DOI : 10.47897/bilmes.1219033

Runge Kutta Optimization for Fixed Size Multimodal Test Functions

Fatih Cantaş [a], Serdar Özyön [b,1], Celal Yaşar [c]

[a] Kütahya Dumlupınar University, Institute of Graduate Education, Kütahya, Turkey
ORCID ID: 0000-0001-5573-021X
[b] Kütahya Dumlupınar University, Faculty of Engineering, Kütahya, Turkey
ORCID ID: 0000-0002-4469-3908
[c] Kütahya Dumlupınar University, Faculty of Engineering, Kütahya, Turkey
ORCID ID: 0000-0002-1251-3562

Abstract

In this study, it is aimed to increase the success of the Runge Kutta (RUN) algorithm, which is used in the solution of many optimization problems in the literature, on fixed-size test functions by changing the parameter values. Optimization can be defined as making a system most efficient at the least possible cost under certain constraints. For this process, many optimization algorithms have been designed in the literature and used to obtain the best solutions for certain problems. The most important parts in solving these problems are modeling the problem correctly, determining the parameters and constraints of the problem, and finally choosing a suitable meta-heuristic algorithm for the solution of the objective function. Not every algorithm is suitable for every problem structure. Therefore, in this study, the suitability of the RUN algorithm for the solution of fixed-size functions will be evaluated. Theoretically, Runge-Kutta methods used in numerical analysis are an important type of the family of closed and open iterative methods for solution approximations of ordinary differential equations. The RUN algorithm is also designed with inspiration from these methods. In order to evaluate the performance of the RUN algorithm on fixed-size functions in the study, 10 fixed-size multimodal test functions (Shekel's Foxholes, Kowalik, Six-Hump Camel-Back, Branin, Goldstein-Price, Hartman3, Hartman6, Shekel5, Shekel7, Shekel10) have been found in the literature before was selected. Solutions for each of the selected functions are obtained by changing the parameter values of the RUN algorithm. The obtained solution values were evaluated by comparing the solutions obtained with Slime Mold Algorithm (SMA) and Hunger Games Search (HGS) algorithms.

Keywords: "Runge kutta algorithm (RUN), slime mould algorithm (SMA), hunger games search (HGS), fixed size multimodal test functions."

1. Giriş

Günümüz koşullarında problemlere hızlı ve kolay çözüm üreten optimizasyon yöntemleri büyük önem arz etmektedir. Optimizasyon, kısaca en iyi çözümü bulmaktır. Bir problemin optimizasyonu, verilen kısıtlar altında o problem için mevcut olan çözümler arasında en iyi olana ulaşmak demektir. Metasezgisel optimizasyon algoritmaları ise klasik yöntemlerle çözümü zor ya da uzun zaman alan problemlere, kabul edilebilir sürede optimuma yakın çözümler bulmaya çalışan algoritmalardır [1].

Bu kapsamda değerlendirilen Runge Kutta algoritması (RUN) sayısal analizde kullanılan, adi diferansiyel denklemlerin çözüm yaklaşımları için kapalı ve açık yinelemeli yöntemler ailesinin önemli bir tipidir. Mühendislik ve uygulamalı bilimlerin farklı alanlarındaki problemlerin çözümlerinde kolayca uyarlanabilir olması, ayrıca dördüncü dereceye kadar olan diferansiyel denklemlerin çözümlerinde kabul edilebilir bir zaman dilimi içerisinde başarılı sonuçlar vermesi nedeniyle araştırmacılar tarafından yaygın olarak kullanılan bir yöntem olmuştur [2].

Literatürde yer alan birçok çalışmada üretilen algoritmalar daha iyi ve kararlı hale getirilmek için algoritma içinde yer alan parametreler ya da yöntemler ilk çıkış halinden farklı formlara dönüştürülmüştür. Yapılan literatür taramasında son yıllarda oldukça geniş bir uygulama alanı bulan bazı güçlü algoritmalar, açlık oyunları arama algoritması (Hunger Games Search- HGS) [3,4], balçık kalıp algoritması (slime mould algorithm-SMA) [5,6], sinüs-kosinüs algoritması (Sine Cosine Algorithm-SCA) [7], çoklu evren optimizasyonu (Multi-Verse Optimizer-MVO) [8] ve merkezi kuvvet optimizasyonu (Central Force Optimization-CFO) [9] şeklinde sayılabilir.

[1] Corresponding Author
E-mail Address: serdar.ozyon@dpu.edu.tr

Literatürdeki birçok çalışmada, üretilen benzer algoritmaları daha iyi ve kararlı hale getirmek için algoritma içinde yer alan parametreler ya da yöntemler ilk çıkış halinden farklı formlara da dönüştürülmüştür [4,6,10]. Bu çalışmada, RUN algoritmasında kullanılan parametrelerden biri olan a için, değişik aralıktaki değerler kullanılarak, RUN algoritmasının sabit boyutlu test fonksiyonları üzerindeki başarısı gözlemlenmiştir. Elde edilen eniyi değerler, aynı platform üzerinde, aynı şartlarla geliştirilmiş farklı algoritmalarla karşılaştırılarak değerlendirilmiştir.

2. Runge Kutta Algoritması (RUN)

RUN algoritması, stokastik bileşenlere sahip yeni bir sürü tabanlı model olarak geliştirilmiştir. Bu algoritma, metaforsuz ve temel matematiksel yapılara sıkı bağla karakterize edilmiştir. Metaforların popülasyona dayalı bir modelde kullanılması uygun olmaz, çünkü böyle bir yöntemin tek faydası optimizasyon algoritmalarında kullanılan denklemlerin gerçek doğasını gözden kaçıracaktır. Bu nedenle RUN algoritması, Runge-Kutta (RK) tekniğinin ana mantığını ve bir ajan sürüsünün popülasyona dayalı evrimini açıklar. Aslında RK, eğimi hesaplamak ve adi diferansiyel denklemleri çözmek için özel bir formülasyon (yani, RK4 yöntemi) kullanır. RUN'un ana fikri, RK yönteminde önerilen hesaplanmış eğim kavramına dayanmaktadır. RUN algoritması, hesaplanan eğimi, arama uzayındaki gelecek vaat eden alanı keşfetmek ve sürü tabanlı optimizasyon algoritmasının mantığına göre bir popülasyon kümesinin evrimi için bir dizi kural oluşturmak için bir arama mantığı olarak kullanır [2].

RK yöntemi, Euler yöntemindeki ortalama eğim hesaplama işlemlerini bir adım daha ileri götürmektedir. Lineer interpolasyonda kullanılan eğim, aralığın sol ve sağ sınırındaki eğimler ile aralığın içinde bulunan birkaç nokta alınarak ağırlıklı ortalama olarak hesaplanır. Eğimler için kullanılan bölmeleme ve bağıl ağırlıklar için birçok alternatif bulunmaktadır. RK yöntemi birinci mertebeden adi diferensiyel denklemlerin çözüm yöntemi olarak bilinir. Bütün bu yöntemler, fonksiyonun Taylor serisi açılımında h^k (k yöntemin mertebesi olmak üzere) mertebe terimlerine kadar uyumludur. Bundan dolayı ikinci, üçüncü ve dördüncü mertebeden RK yöntemleri vardır. İkinci mertebeden yöntemler h^2'li terime kadar bütün terimlere sahiptir. Dördüncü mertebeden RK yöntemi ise h^4 'lü terime kadar bütün terimlere sahiptir. En yaygın kullanılan RK yöntemi olup klasik RK diye de isimlendirilir [2].

2.1. Dördüncü Mertebeden RK Yöntemi

Bir adi diferansiyel denklemi çözerken en çok kullanılan yöntem dördüncü mertebeden RK yöntemidir. Dördüncü mertebe yöntem için y_{i+1} denklem (1)'den elde edilir.

$$y_{i+1} = y_i + \frac{1}{6}(\mathrm{k}_1 + 2\,\mathrm{k}_2 + 2\,\mathrm{k}_3 + \mathrm{k}_4)\,\mathrm{h} \tag{1}$$

Bu denklemde k_i ağırlık değerleri ile elde edilen y değeri, tahmini artışı ifade etmekte olup k_1, k_2, k_3 ve k_4 eşitlikleri denklem (2)'de verilmiştir.

$$\left.\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f\left(x_i + h/2, y_i + k_1.h/2\right) \\ k_3 &= f\left(x_i + h/2, y_i + k_2.h/2\right) \\ k_4 &= f\left(x_i + h, y_i + k_3.h\right) \end{aligned}\right\} \tag{2}$$

Burada k_1 sol sınırın eğim tahminini, k_4 sağ sınırın eğim tahminini, k_2 ve k_3 ise aradaki iki noktanın eğim tahminlerini göstermektedir. k_i değerlerinin ağırlıklı ortalaması y'deki artışı belirlemek için kullanılır ve y_{i+1} değerinin bulunması için y_i değerine eklenir. RUN algoritmasının çalışma prensipleri başlatma adımı, arama mekanizmasının kökü, çözümlerin güncellenmesi ve gelişmiş çözüm kalitesi gibi başlıklar halinde aşağıda özetlenmiştir [2].

2.2. Başlatma Adımı

Bu adımda mantık, izin verilen yineleme sayısı içinde geliştirilecek bir başlangıç sürüsü ayarlamaktır. RUN algoritmasında, N büyüklüğündeki bir popülasyon için N pozisyon rastgele oluşturulur. Popülasyonun her bir üyesi x_n $(n = 1,2,3 \ldots, N)$, bir optimizasyon problemi için D boyutuna sahip bir çözümdür. Genel olarak, başlangıç konumları aşağıdaki denklem (3) tarafından rastgele oluşturulur [2].

$$x_{n,l} = L_l + rand(\mathrm{U}_l - L_l) \tag{3}$$

Denklemde L_l ve U_l değerleri, problemin *l*. değişkeninin alt ve üst sınırlarını ($l=1,2,\ldots,D$), ve *rand* ise **[0, 1**] aralığında rastgele bir sayıyı göstermektedir. Bu kural sadece sınırlar içinde bazı çözümler üretir.

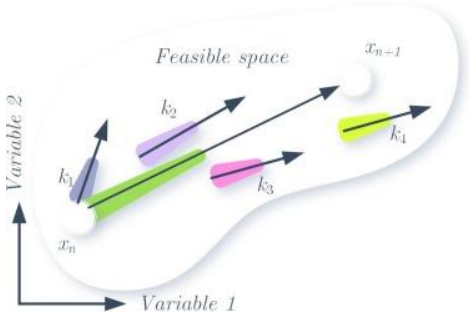
2.3. Arama Mekanizmasının Kökü

Herhangi bir optimize edicinin gücü, keşif ve kullanım modellerini oluşturmak için yinelemeli çekirdeklerine bağlıdır. Keşif çekirdeğinde, bir optimizasyon algoritması, uygulanabilir uzayın gelecek vaat eden alanlarını keşfetmek için yüksek rastgelelik oranına sahip bir dizi rastgele çözüm kullanır. Çekirdek çözümlerin kullanımındaki küçük ve kademeli varyasyonlar ve rastgele davranışlar, keşif mekanizmasındakilerden oldukça düşüktür. Bu çalışmada, RUN algoritmasının önde gelen arama mekanizması, bir dizi rasgele çözüm kullanarak karar uzayını aramak ve uygun bir küresel ve yerel arama uygulamak için RK yöntemine dayanmaktadır [2].

2.4. Çözümlerin Güncellemesi

RUN algoritması, optimizasyon sürecini bir dizi rastgele bireyle (çözümler) başlatır. Her yinelemede çözümler, RK yöntemini kullanarak konumlarını günceller. Bunu yapmak için RUN algoritması, bir çözüm ve RK yöntemiyle elde edilen arama mekanizmasını kullanır. Şekil 1, bir konumun RK yöntemini kullanarak konumunu nasıl güncellediğini göstermektedir. Bu çalışmada, global (keşif) ve yerel (işletme) aramayı sağlamak için, bir sonraki yinelemede konum oluşturmak için Denklem (4)'teki akış uygulanmıştır [2].

$$\left.\begin{array}{l} if \quad rand < 0,5 \\ x_{n+1} = (x_C) + SF \times SM + \mu \times x_S \quad \text{(keşif aşaması)} \\ else \\ x_{n+1} = (x_m) + SF \times SM + \mu \times x_{S'} \quad \text{(işletme aşaması)} \\ end \end{array}\right\} \tag{4}$$



Şekil 1. Bir sonraki konumu elde etmek için RK tarafından kullanılan eğimler (x_{k+1}) [2]

Denklem (4)'deki μ rasgele bir sayı olup eşitliği denklem (5)'te, x_S ve $x_{S'}$ değerleri ise denklem (6)'da ve X_C ve X_m değerleri ise denklem (7)'de verilmiştir. *randn* ise normal dağılıma sahip rasgele bir sayıdır.

$$\mu = 0,5 + 0,1 randn \tag{5}$$

$$\left.\begin{array}{l} x_S = randn(x_m - x_C) \\ x_{S'} = randn(x_{r1} - x_{r2}) \end{array}\right\} \tag{6}$$

$$\left.\begin{array}{l} x_C = \varphi \times x_n + (1-\varphi) \times x_{r1} \\ x_m = \varphi \times x_{best} + (1-\varphi) \times x_{lbest} \end{array}\right\} \tag{7}$$

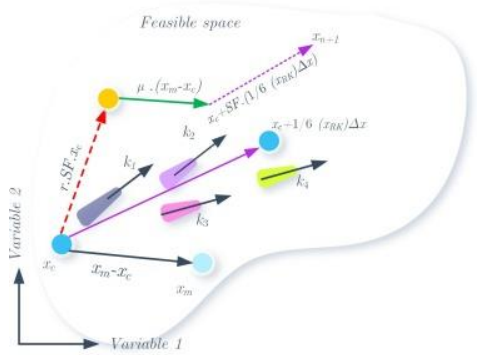
Denklemlerdeki φ (0,1) aralığında rastgele bir sayıyı, x_{best} o ana kadarki en iyi çözümü ve x_{lbest} ise her yinelemede elde edilen en iyi konumu göstermektedir. SF adaptif bir faktördür, denklem (8)'de verildiği gibidir.

$$\left.\begin{array}{l} SF = 2(0,5 - rand) \times f \\ f = a \times \exp(-b \times rand \times \dfrac{i}{Maxi}) \end{array}\right\} \tag{8}$$

Denklemde a ve b iki sabit sayıyı, i yineleme sayısını ve $Maxi$ ise maksimum yineleme sayısını göstermektedir. Bu çalışmada, keşif ve işletme arasında uygun bir denge sağlamak için SF kullanılmıştır. Çeşitliliği artırmak ve keşif aramasını geliştirmek için başlangıçtaki yinelemelerde büyük bir SF değeri belirlenir; sonra, yineleme sayısı arttığında işletme arama yeteneğini desteklemek için değeri azaltılır. RUN algoritmasın ana kontrol parametreleri, (SF)'de kullanılan a ve b olan iki parametreyi içermektedir. Denklem (4)'deki kural, önerilen RUN algoritmasının, $rand < 0,5$ koşuluna dayalı olarak keşif işletme aşamalarını seçtiğini göstermektedir. RUN algoritmasının optimizasyon için kullanılan bu yeni prosedüründe $rand < 0,5$ ise, çözüm uzayında global bir aramanın uygulanmasını ve aynı anda x_c çözümü etrafında yerel bir aramanın yapılmasını sağlar. Yeni bir küresel arama (keşif) uygulayarak, algoritma arama alanının üstün gelecek vaat eden bölgelerini keşfedebilir. Diğer taraftan, $rand \geq 0,5$ ise RUN algoritması, X_m çözümü etrafında yerel bir arama gerçekleştirir. Bu yerel arama aşamasını uygulayarak, önerilen algoritma yakınsama hızını etkili bir şekilde artırabilir ve yüksek kaliteli çözümlere odaklanabilir. X_c ve X_m çözümleri etrafında yerel arama yapmak ve arama uzayında gelecek vaat eden bölgeleri keşfetmek için denklem (4) aşağıdaki gibi yeniden düzenlenir [2].

$$\left.\begin{array}{l} if \quad rand < 0,5 \\ x_{n+1} = (\mathrm{x}_C + r \times SF \times g \times x_C) + \mathrm{SF} \times \mathrm{SM} + \mu \times x_S \quad \text{(keşif aşaması)} \\ else \\ x_{n+1} = (\mathrm{x}_m + r \times SF \times g \times x_m) + \mathrm{SF} \times \mathrm{SM} + \mu \times x_{S'} \quad \text{(işletme aşaması)} \\ end \end{array}\right\} \tag{9}$$

Denklemde g, [0, 2] aralığında rastgele bir sayıyı göstermekte, r ise, 1 veya -1 olan bir tam sayı olup arama yönünü değiştirerek çeşitliliği artırmaktadır. Denklem (9)'a göre, yineleme (iterasyon) sayısı arttıkça X_c etrafındaki yerel arama azalır. Bir sonraki yinelemede X_{n+1} konumunun nasıl oluşturulacağını gösteren RUN algoritmasının arama mekanizması Şekil 2'de verilmiştir [2].



Şekil 2. RUN algoritmasının arama mekanizması [2]

2.5. Gelişmiş Çözüm Kalitesi (ESQ)

RUN algoritmasında, çözümlerin kalitesini artırmak ve her yinelemede yerel optimumdan kaçınmak için geliştirilmiş çözüm kalitesi (enhanced solution quality-ESQ) kullanılır. RUN algoritması, ESQ uygulayarak her çözümün daha iyi bir konuma doğru hareket etmesini sağlar. Önerilen ESQ'da, üç rastgele çözümün (x_{ort}) ortalaması hesaplanır ve yeni bir çözüm (x_{yeni1}) oluşturmak için en iyi konumla (x_b) birleştirilir. ESQ kullanılarak sonraki çözümü (x_{yeni2}) oluşturmak için denklem (10)'daki işlem yürütülür [2]:

$$\left.\begin{array}{l} if \quad rand < 0,5 \\ \quad if \quad w < 1 \\ x_{yeni2} = x_{yeni1} + \mathrm{r.w} \left| x_{yeni1} - x_{ort} \right| + randn \\ else \\ x_{yeni2} = \left(x_{yeni1} - x_{ort} \right) + \mathrm{r.w} \left| \left(u.x_{yeni1} - x_{ort} + randn \right) \right| \\ end \\ end \end{array}\right\} \tag{10}$$

Denklemdeki w, x_{ort} ve x_{yeni1} değerleri sırasıyla denklem (11), (12) ve (13)'te verilmiştir.

$$w = rand(0,2)\exp\left(-c\left(\frac{i}{Maxi}\right)\right) \tag{11}$$

$$x_{ort} = \frac{x_{r1} + x_{r2} + x_{r3}}{3} \tag{12}$$

$$x_{yeni1} = \beta \times x_{ort} + (1-\beta) \times x_{best} \tag{13}$$

Denklemlerde β , [0, 1] aralığında rastgele bir sayıyı, c ise seçilebilecek rastgele bir sayıyı, w artan yineleme sayısıyla azalan rastgele bir sayıyı ve r ise 1, 0 veya -1 olan bir tam sayıyı göstermektedir. Denklem (10)'nun ikinci koşulunda çeşitliliği artırmak için u parametresinin tanımlanmıştır. Bu bölümde hesaplanan çözüm (x_{yeni2}) mevcut çözümden daha iyi uygunluk göstermeyebilir (yani, ($f(x_{yeni2}) > f(x_n)$). İyi bir çözüm oluşturmak için başka bir şansa sahip olmak için başka bir yeni çözüm (x_{yeni3}) oluşturulur ve aşağıdaki gibi tanımlanır.

$$\left.\begin{array}{l} if \quad w < 1 \\ x_{yeni3} = \left(x_{yeni2} - rand.x_{yeni2}\right) + SF\left(rand.x_{RK} + \left(v.x_b - x_{yeni2}\right)\right) \end{array}\right\} \tag{14}$$
$$end$$

Denklemde v, 2x$rand$ değerine sahip rastgele bir sayıdır. Aslında, yeni çözüm (x_{yeni3}), $rand$ <w koşulu karşılandığında uygulanır. Denklem (14)'ün temel amacı, x_{yeni2} çözümünü daha iyi bir konuma taşımaktır. Bu denklemin ilk kuralında x_{yeni2} civarında yerel bir arama oluşturulur ve ikinci kuralda RUN en iyi çözüme doğru hareketle gelecek vaat eden bölgeleri keşfetmeye çalışır. Bu nedenle, en iyi çözümün önemini vurgulamak için v katsayısı kullanılır. X_{rk}'yı hesaplamak için x_b ve x_w çözümlerinin sırasıyla x_k ve x_{yeni2} olduğu unutulmamalıdır, çünkü x_n'nin uygunluk değeri x_{yeni2}'kinden küçüktür (yani, ($f(x_{yeni2}) > f(x_n)$). RUN algoritmasının sözde kodu, Algoritma 1'de verilmiştir [2].

Algoritma 1. RUN Algoritmasının sözde kodu [2]

Aşama 1. Başlatma
a, b'yi gir.
x_n $(n = 1,2,3 \ldots, N)$ RUN algoritmasının popülasyonunu oluştur
Popülasyonun her üyesinin amaç fonksiyonunu hesaplayın
x_w, x_b ve x_{best} çözümlerini belirleyin
Aşama 2. RUN operatörleri
for *i=1:Maxi*
for *n=1:N*
for *l=1:D*
Denklem (9)'u kullanarak $x_{n+1,l}$ konumunu hesaplayın
end for
Çözüm kalitesini artırın
if *rand* <0,5
Denklem (10)'u kullanarak x_{yeni2} konumunu hesaplayın
if $f(x_n) < f(x_{yeni2})$
if rand < *w*
Denklem (11)'i kullanarak x_{yeni3} konumunu hesaplayın
end
end
end
x_w ve x_b konumlarını güncelle
end for
x_{best} konumunu güncelle
i=i+1
end
Aşama 3. x_{best}*'e* dön.

3. Araştırma ve Bulgular

3.1. Test Fonksiyonları

RUN algoritmasının, performansının değerlendirilebilmesi için literatürde farklı araştırmacılar tarafından daha önce farklı algoritmalar ile çözümü yapılmış olan 10 adet test fonksiyonu seçilmiştir. Bu fonksiyonlar Tablo 1'de verilmiştir. Çizelgede, *D* değeri fonksiyonun boyutunu, *S* arama uzayını, f_{min} ise fonksiyonun minimum değerini göstermektedir. Bu fonksiyonlar düşük ve sabit boyutlu olup az sayıda lokal minimum noktaları olan multimodal fonksiyonlardır [11].

Tablo 1. Test fonksiyonları

	Formül	***Fonksiyon Adı***	***D***	***Aralık (S)***	***f*min**
1	$f_1(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	Shekel's Foxholes	2	[–65.53,65.53]2	0,998
2	$f_2(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]$	Kowalik	4	[–5,5]4	0,0003
3	$f_3(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-Hump Camel Back	2	[–5,5]2	-1,0316
4	$f_4(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	Branin	2	[–5,10]x[0,15]	0,397887
5	$f_5(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]x \ldots$ $[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Goldstein-Price	2	[–2,2]2	3
6	$f_6(x) = -\sum_{J=1}^{3} ci\ exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	Hartman3	3	[0,1]3	-3,86278
7	$f_7(x) = -\sum_{i=1}^{4} ci\ exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	Hartman6	6	[0,1]6	-3,32237
8	$f_8(x) = -\sum_{i=1}^{5}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i)\right]^{-1}$	Shekel5	4	[0,10]4	-10,1532
9	$f_9(x) = -\sum_{i=1}^{7}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i)\right]^{-1}$	Shekel7	4	[0,10]4	-10,4029
10	$f_{10}(x) = -\sum_{i=1}^{10}\sum_{j=1}^{4}\left[(x_j - a_{ij})(x_j - a_{ij})^T + c_i)\right]^{-1}$	Shekel10	4	[0,10]4	10,5364

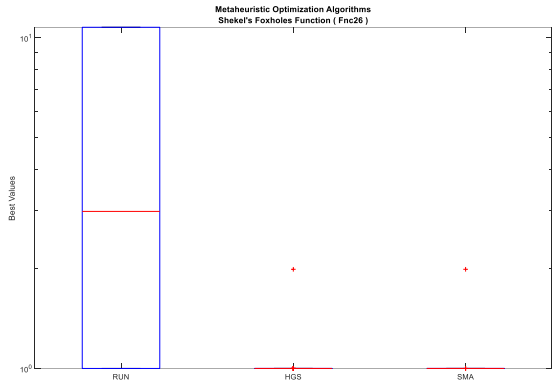
3.2. Sayısal Sonuçlar

Bu çalışmada yer alan test fonksiyonlarının çözümünde kullanılan a parametresi değiştirilirken, b parametresi, literatürde RUN algoritmasıyla aynı değer olarak alınmıştır (*b=12*). Kullanılan diğer değişkenler popülasyon boyutu (N) 50, iterasyon sayısı (iteN) 1000 olarak alınmıştır. Karşılaştırma yapılan bütün algoritmalar için bu değerler eşit alınmıştır. Bunun nedeni algoritmalar arasındaki karşılaştırmanın doğru ve sağlıklı yapılabilmesi içindir. Böylelikle geliştirilen metotlar performans, yakınsama, kararlılık ve hız açısından daha doğru değerlendirilmiş olacaktır. Çalışmada sonuçları verilen bütün algoritmalar ve test fonksiyonlarının çözümü için MATLAB R2022b'de geliştirilen programlar AMD Ryzen 7 3800X 8-Core Processor 3.90 GHz işlemcili ve 16 GB RAM bellekli iş istasyonunda Çizelge 1'deki fonksiyonlar için 1000 iterasyon (50000 fonksiyon çağırımı, FCall) çalıştırılmıştır. Tablo 1'de RUN algoritmasına ait farklı *a* değerleri için elde edilen sonuçlar verilmiştir.

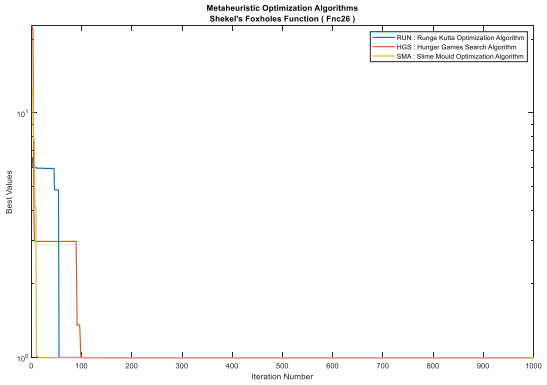
Tablo 2. Farklı *a* değerleri için elde edilen veriler (30 bağımsız çalışma- 1000 iterasyon)

		10	*15*	*20*	*25*	*30*
f_1	EnKötü	1.076318067e+01	1.076318067e+01	1.076318067e+01	1.076318067e+01	1.076318067e+01
	Ortalama	2.409653679e+00	2.051013549e+00	4.197281388e+00	2.834298759e+00	3.452439833e+00
	Enİyi	9.980038378e-01	9.980038378e-01	9.980038378e-01	9.980038378e-01	9.980038378e-01
	StdSapma	2.416788689e+00	1.874362411e+00	4.046554275e+00	2.805311741e+00	3.378033051e+00
	Süre (s)	0.857953	0.863482	0.863241	0.868943	0.866918
f_2	EnKötü	1.889337489e-01	1.889337487e-01	1.889337486e-01	1.889337491e-01	1.889337495e-01
	Ortalama	6.724412604e-02	5.065008662e-02	2.852470093e-02	5.065008698e-02	5.065008710e-02
	Enİyi	2.299335416e-02	2.299335416e-02	2.299335416e-02	2.299335416e-02	2.299335416e-02
	StdSapma	7.338160347e-02	6.184233369e-02	2.978721236e-02	6.184233359e-02	6.184233356e-02
	Süre (s)	0.424347	0.424921	0.424211	0.423041	0.427185
f_3	EnKötü	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00
	Ortalama	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00
	Enİyi	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00	-1.031628453e+00
	StdSapma	5.978281758e-14	1.327760395e-13	1.321611528e-13	1.569940362e-13	4.045270849e-13
	Süre (s)	0.444946	0.443917	0.44607	0.446965	0.448546
f_4	EnKötü	3.978873577e-01	3.978873577e-01	3.978873577e-01	3.978873578e-01	3.978873577e-01
	Ortalama	3.978873577e-01	3.978873577e-01	3.978873577e-01	3.978873577e-01	3.978873577e-01
	Enİyi	3.978873577e-01	3.978873577e-01	3.978873577e-01	3.978873577e-01	3.978873577e-01
	StdSapma	8.537860065e-13	9.296393350e-13	2.905012321e-12	8.092891212e-12	2.917507966e-12
	Süre (s)	0.419307	0.420943	0.419064	0.421273	0.421646
f_5	EnKötü	3.000000213e+00	3.000000241e+00	3.000000063e+00	3.000000655e+00	3.000000421e+00
	Ortalama	3.000000014e+00	3.000000034e+00	3.000000009e+00	3.000000046e+00	3.000000045e+00
	Enİyi	3.000000000e+00	3.000000000e+00	3.000000000e+00	3.000000000e+00	3.000000000e+00
	StdSapma	3.998206129e-08	6.092053966e-08	1.650698400e-08	1.276370719e-07	8.684382822e-08
	Süre (s)	0.418445	0.416773	0.417586	0.417793	0.419598
f_6	EnKötü	-3.862782145e+00	-3.862782147e+00	-3.862782142e+00	-3.862782141e+00	-3.862782145e+00
	Ortalama	-3.862782148e+00	-3.862782148e+00	-3.862782147e+00	-3.862782147e+00	-3.862782148e+00
	Enİyi	-3.862782148e+00	-3.862782148e+00	-3.862782148e+00	-3.862782148e+00	-3.862782148e+00
	StdSapma	4.260309090e-10	1.574606466e-10	1.271676321e-09	1.357638186e-09	5.817528482e-10
	Süre (s)	0.476469	0.475015	0.473804	0.474769	0.473759
f_7	EnKötü	-3.203102039e+00	-3.203102036e+00	-3.203102024e+00	-3.203102042e+00	-3.203101890e+00
	Ortalama	-3.254622583e+00	-3.274438173e+00	-3.286327525e+00	-3.286327527e+00	-3.254622574e+00
	Enİyi	-3.321995589e+00	-3.321995589e+00	-3.321995589e+00	-3.321995589e+00	-3.321995589e+00
	StdSapma	5.891598494e-02	5.824570156e-02	5.448386777e-02	5.448386526e-02	5.891599260e-02
	Süre (s)	0.483162	0.486115	0.484211	0.488591	0.486961
f_8	EnKötü	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640981e+01
	Ortalama	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01
	StdSapma	3.238524258e-11	9.329384990e-11	1.565975890e-10	3.552990315e-10	5.964816456e-10
	Süre (s)	0.50228	0.506674	0.503974	0.503205	0.502636
f_9	EnKötü	-1.053640982e+01	-1.053640982e+01	-5.128480787e+00	-1.053640981e+01	-1.053640981e+01
	Ortalama	-1.053640982e+01	-1.053640982e+01	-1.035614552e+01	-1.053640982e+01	-1.053640982e+01
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01
	StdSapma	9.295259264e-11	8.139412708e-11	9.707529697e-01	8.082734482e-10	4.078465329e-10
	Süre (s)	0.506504	0.509365	0.504749	0.504194	0.505743
f_{10}	EnKötü	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-5.128480787e+00	-5.128480787e+00
	Ortalama	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.035614552e+01	-1.035614552e+01
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01	-1.053640982e+01
	StdSapma	7.688594525e-11	1.763667712e-10	2.022781615e-10	9.707529697e-01	9.707529696e-01
	Süre (s)	0.507534	0.509114	0.505672	0.50273	0.505777

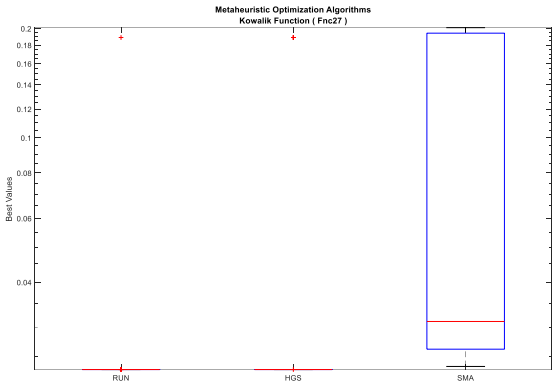
Tabloda elde edilen sonuçlar incelendiğinde farklı *a* değerlerinin ortalama sonuçları elde etmede benzer etkiler gösterildiği görülmektedir. *a=15, a=20* değerleri için daha fazla test fonksiyonunda iyi sonuç elde ettiği görülmektedir. Aynı fonksiyonların aynı platforma ve aynı fonksiyon çağırım sayılarıyla RUN, SMA ve HGS algoritmalarıyla elde edilen kutu grafikleri ve yakınsama eğrileri Şekil 3.1-13.2 arasında verilmiştir.



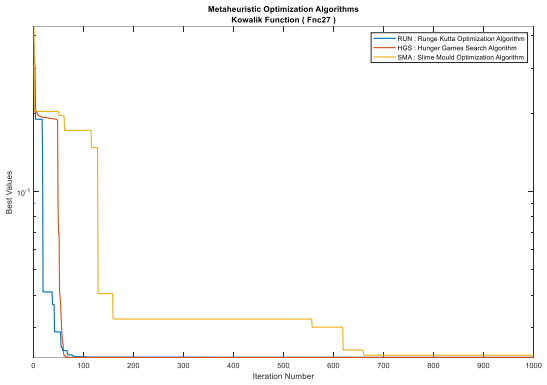
Şekil 3.1 f_1 için kutu grafiği



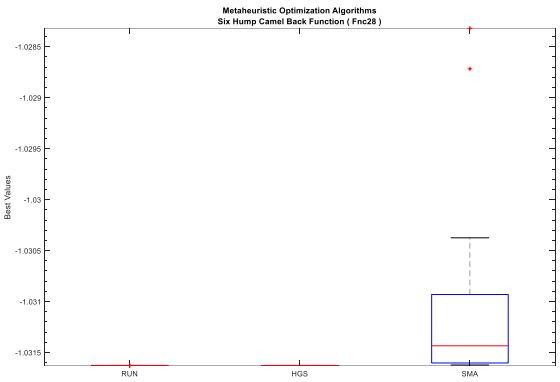
Şekil 3.2 f_1 için yakınsama eğrileri



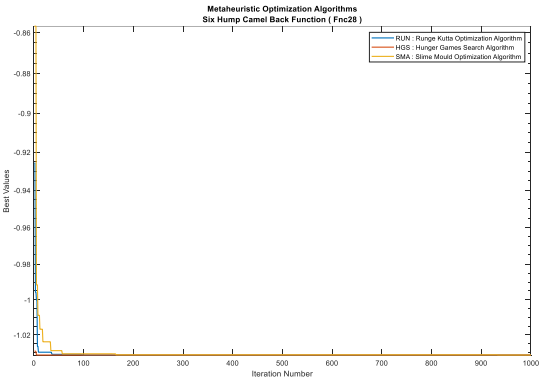
Şekil 4.1 f_2 için kutu grafiği



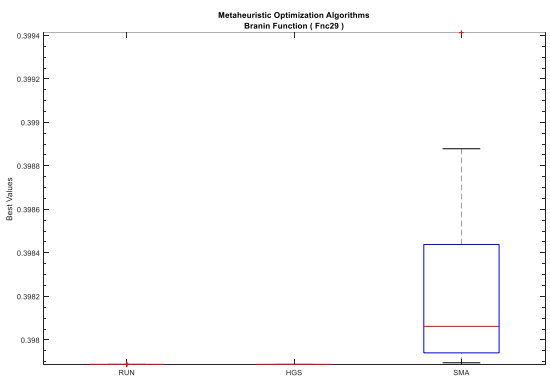
Şekil 4.2. f_2 için yakınsama eğrileri



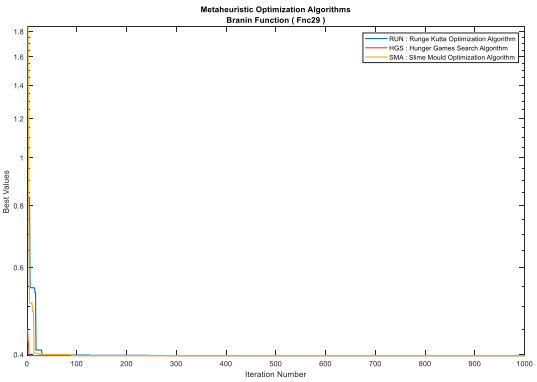
Şekil 5.1 f_3 için kutu grafiği



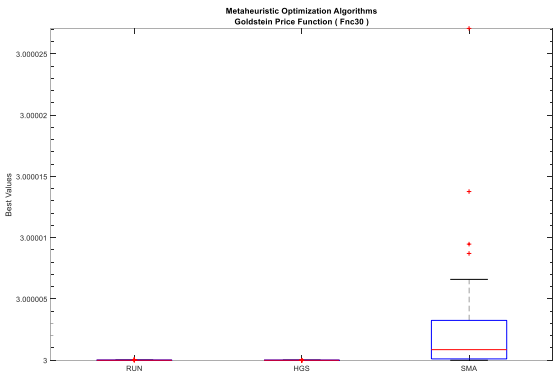
Şekil 5.2 f_3 için yakınsama eğrileri



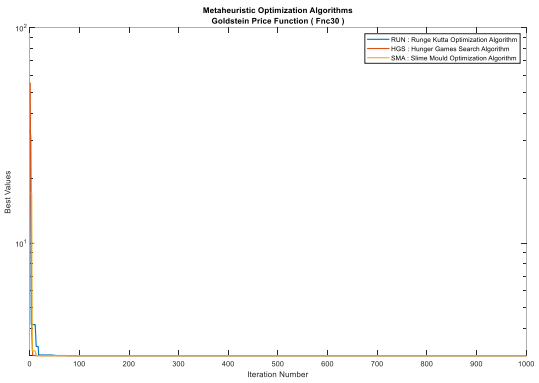
Şekil 6.1 f_4 için kutu grafiği



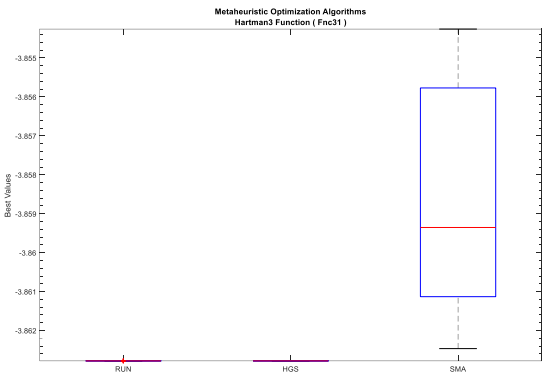
Şekil 6.2 f_4 için yakınsama eğrileri



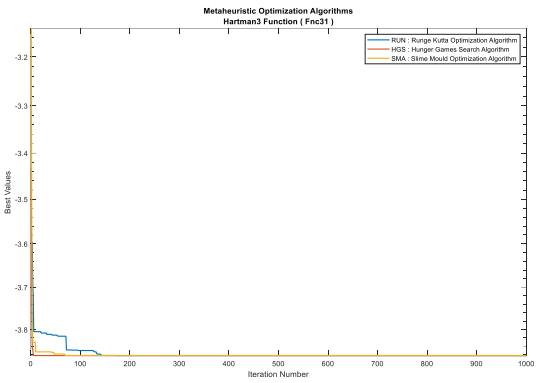
Şekil 7.1 f_5 için kutu grafiği



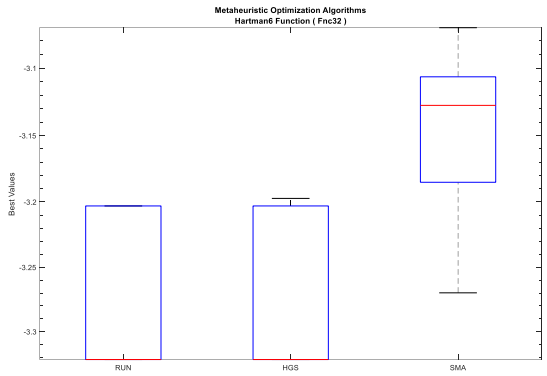
Şekil 7.2 f_5 için yakınsama eğrileri



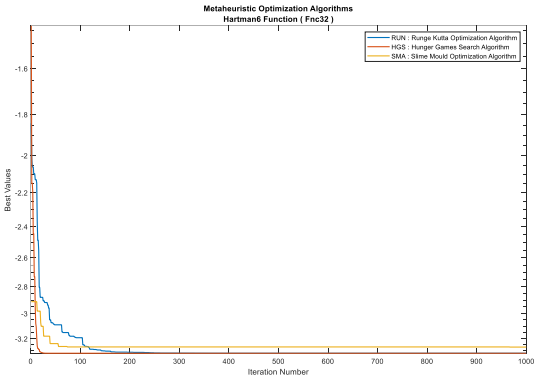
Şekil 8.1 f_6 için kutu grafiği



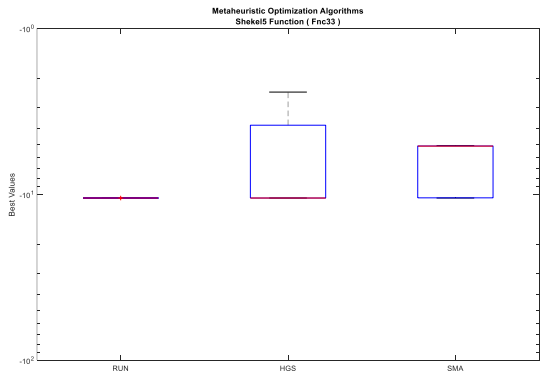
Şekil 8.2 f_6 için yakınsama eğrileri



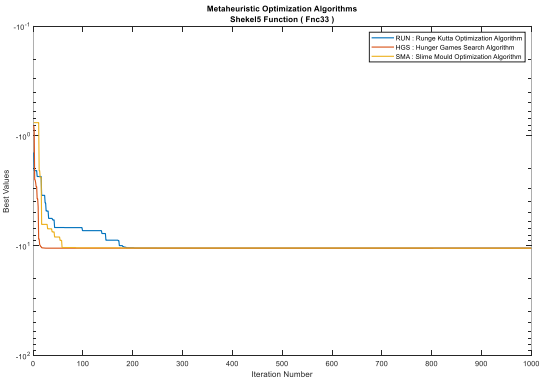
Şekil 9.1 f_7 için kutu grafiği



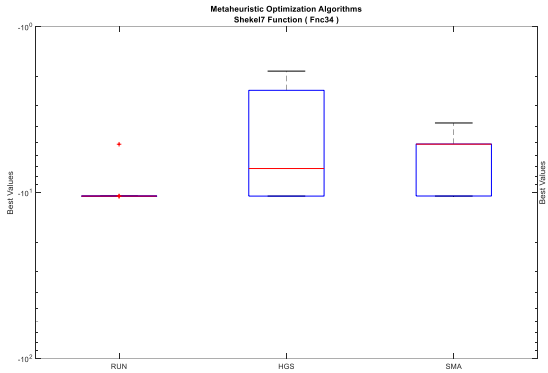
Şekil 9.2 f_7 için yakınsama eğrileri



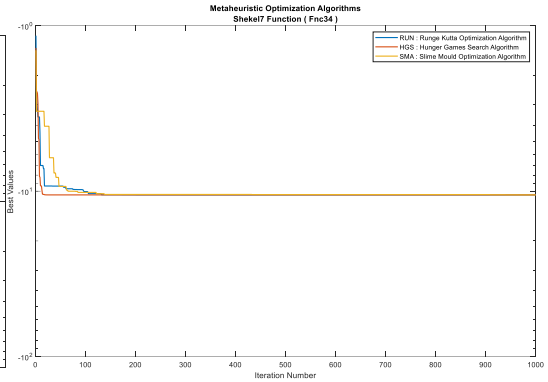
Şekil 10.1 f_8 için kutu grafiği



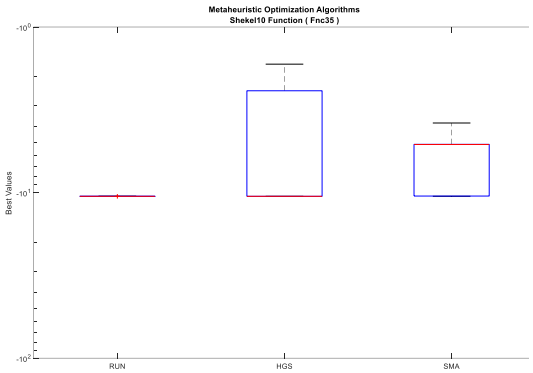
Şekil 10.2 f_8 için yakınsama eğrileri



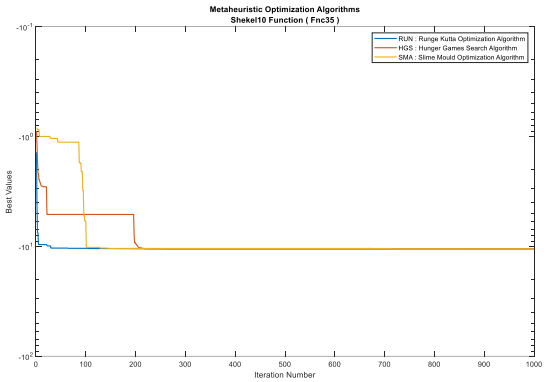
Şekil 11.1 f_9 için kutu grafiği



Şekil 11.2 f_9 için yakınsama eğrileri



Şekil 12.1 f_{10} için kutu grafiği



Şekil 12.2 f_{10} için yakınsama eğrileri

Şekil 3.1-13.2 arasında verilen, RUN, SMA ve HGS algoritmalarıyla elde edilen yakınsama eğrileri incelendiğinde HGS algoritmasının diğer iki algoritmaya göre oldukça iyi bir performans gösterdiği görülmektedir. RUN algoritması sadece f_{10} için, SMA algoritması ise sadece f_1 için daha iyi bir yakınsama göstermiştir. Algoritmalarla elde edilen en iyi sonuçların karşılaştırmalar ise Tablo 3'te verilmiştir. Karşılaştırma yapılırken bütün sonuçlar aynı bilgisayardan alınmıştır.

Tablo incelendiğinde RUN algoritmasının en iyi çözüm ve ortalama sonuçlar bakımından f_2, f_4, ve f_{10} da, diğer fonksiyonlarda ise HGS ve SMA algoritmalarının en iyi değerleri yakaladıkları görülmektedir. Ortalama süre açısından da RUN algoritmasının en iyi sürede sonuca ulaştığı fonksiyon görülmemektedir. HGS ve SMA algoritmasının en iyi değeri yakaladığı durumlar için ortalama değerler ve standart sapmalar açısından karşılaştırıldığında ise, RUN algoritmasından daha iyi olduğu görülmektedir.

Tablo 3. Literatür karşılaştırması

		RUN	***HGS [4]***	***SMA [6]***
f_1	EnKötü	1.076318067e+01	1.992030900e+00	1.992030900e+00
	Ortalama	4.197281388e+00	1.031138073e+00	1.031138833e+00
	EnİYi	9.980038378e-01	9.980038378e-01	9.980038378e-01
	StdSapma	4.046554275e+00	1.784333185e-01	1.784331773e-01
	Süre (s)	0.863241	0.308757	0.551339
f_2	EnKötü	1.889337486e-01	1.889337486e-01	2.010852839e-01
	Ortalama	2.852470093e-02	4.511874008e-02	9.469342197e-02
	EnİYi	2.299335416e-02	2.299335416e-02	2.344709703e-02
	StdSapma	2.978721236e-02	5.640888729e-02	7.810053422e-02
	Süre (s)	0.424211	0.0521708	0.29629
f_3	EnKötü	-1.031628453e+00	-1.031628453e+00	-1.028320341e+00
	Ortalama	-1.031628453e+00	-1.031628453e+00	-1.031130208e+00
	EnİYi	-1.031628453e+00	-1.031628453e+00	-1.031624076e+00
	StdSapma	1.321611528e-13	5.438959822e-16	7.722245746e-04
	Süre (s)	0.44607	0.0601073	0.297698

Tablo 3. Literatür karşılaştırması (devamı)

		RUN	*HGS [4]*	*SMA [6]*
f_4	EnKötü	3.978873577e-01	3.978873577e-01	3.994133960e-01
	Ortalama	3.978873577e-01	3.978873577e-01	3.982144398e-01
	Enİyi	3.978873577e-01	3.978873577e-01	3.978950035e-01
	StdSapma	2.905012321e-12	0.000000000e+00	3.650342261e-04
	Süre (s)	0.419064	0.0448801	0.285088
f_5	EnKötü	3.000000063e+00	3.000000000e+00	3.000027083e+00
	Ortalama	3.000000009e+00	3.000000000e+00	3.000003146e+00
	Enİyi	3.000000000e+00	3.000000000e+00	3.000000001e+00
	StdSapma	1.650698400e-08	1.329798179e-15	5.559972441e-06
	Süre (s)	0.417586	0.0447413	0.280431
f_6	EnKötü	-3.862782142e+00	-3.862782148e+00	-3.854258088e+00
	Ortalama	-3.862782147e+00	-3.862782148e+00	-3.858693374e+00
	Enİyi	-3.862782148e+00	-3.862782148e+00	-3.862466113e+00
	StdSapma	1.271676321e-09	2.467260961e-15	2.666409285e-03
	Süre (s)	0.473804	0.0732032	0.317267
f_7	EnKötü	-3.203102024e+00	-3.197382698e+00	-3.070466802e+00
	Ortalama	-3.286327525e+00	-3.270284411e+00	-3.145451199e+00
	Enİyi	-3.321995589e+00	-3.321995589e+00	-3.269828897e+00
	StdSapma	5.448386777e-02	5.914250450e-02	5.227637375e-02
	Süre (s)	0.484211	0.0824719	0.356651
f_8	EnKötü	-1.053640982e+01	-2.421734027e+00	-5.089399323e+00
	Ortalama	-1.053640982e+01	-8.479465018e+00	-7.276342830e+00
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053396209e+01
	StdSapma	1.565975890e-10	3.423413380e+00	2.641601190e+00
	Süre (s)	0.503974	0.092968	0.374028
f_9	EnKötü	-5.128480787e+00	-1.859480301e+00	-3.818636754e+00
	Ortalama	-1.035614552e+01	-6.688432405e+00	-7.192217548e+00
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053516915e+01
	StdSapma	9.707529697e-01	3.874415479e+00	2.736496390e+00
	Süre (s)	0.504749	0.0941297	0.370598
f_{10}	EnKötü	-1.053640982e+01	-1.676553250e+00	-3.800228293e+00
	Ortalama	-1.053640982e+01	-7.097308651e+00	-6.605600513e+00
	Enİyi	-1.053640982e+01	-1.053640982e+01	-1.053325327e+01
	StdSapma	2.022781615e-10	3.952848584e+00	2.587773966e+00
	Süre (s)	0.505672	0.0945474	0.370248

4. Sonuçlar

Bu çalışmada literatürde son yıllarda ortaya atılmış RUN, sabit boyutlu multimodal test fonksiyonlarına uygulanmıştır. Algoritmanın farklı parametre değerleri için de fonksiyonların minimum noktasını bulmak için çözüm aranmıştır. RUN algoritmasıyla elde edilen sonuçlar, literatürde farklı çalışmalarda uygulama alanı bulan SMA ve HGS sonuçlarıyla karşılaştırılmıştır. Bu çalışma RUN algoritmasının test fonksiyonları üzerindeki performansının araştırılması ve algoritmaya ait farklı parametre değişimlerindeki davranışlarının gözlemlenmesi için bir ön çalışma niteliği taşımaktadır. Bundan sonra yapılacak çalışmalarda algoritma elektrik elektronik mühendisliği alanında önemli bir yere sahip olan farklı yapılardaki ekonomik güç dağıtımı problemlerine uygulanacaktır.

Bilgi

Bu çalışma; 4. Uluslararası Palandöken Bilimsel Araştırmalar Kongresinde özet bildiri olarak sunulmuştur.

Referanslar

[1] Özyön, S., Yaşar, C., Temurtaş, H., Test fonksiyonları için kaos tabanlı yerçekimsel arama algoritmaları (CbGSA-X), Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 8(3),1771-1793, 2020.

[2] Ahmadianfar, I., Heidari, AA., Gandomi, AH., Chu, X., Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method, Expert Systems with Applications, 181, 2021, 115079.

[3] Yang, Y., Chen, H., Heidari, AA., Gandomi, AH., Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, Expert Systems with Applications, 177, 2021, 114864.
[4] Çam, H., Yaşar, C., Özyön, S. Açlık oyunları arama algoritmasının değişken boyutlu test fonksiyonlarına uygulanması, 4. Uluslararası Palandöken Bilimsel Çalışmalar Kongresi, Erzurum, 880-891, 28-29.04.2022.
[5] Li, S., Chen, H., Wang, M., Heidari, AA., Mirjalili, S., Slime mould algorithm: a new method for stochastic optimization, Future generation computer systems, 111, 300-323, 2020.
[6] Kaya, MF., Yaşar, C., Özyön, S. Meta-Sezgisel Balçık Kalıp Algoritmasının Performans Analizi - Performance Analysis of Meta Heuristic Slime Mould Algorithm, 4. Uluslararası Palandöken Bilimsel Çalışmalar Kongresi, Erzurum, 910-921, 28.04.2022.
[7] Mirjalili, S., SCA: A sine cosine algorithm for solving optimization problems, Knowledge-Based Systems, 96, 120-133, 2016.
[8] Mirjalili, S., Mirjalili, SM., Hatamlou, A. Multi-verse optimizer: a natüre-inspired algorithm for global optimization, Neural Computing and Applications, 27, 495-513, 2016.
[9] Formato, R., Central force optimization: A new metaheuristic with applications in applied electromagnetics, Progress in Electromagnetics Research, 77(1), 425-491, 2007.
[10] Özyön, S., Durmuş, B., Kuvat, G. Özcan, G. Yüksek boyutlu problemlerin optimizasyonunda parametre seçiminin genetik algoritma performansına etkileri, International Multidisciplinary Congree of Eurasia (IMCOFE'15), Üsküp, 01.09.2015.
[11] Özyön, S., Yaşar, C. Temurtaş, H. Incremental gravitational search algorithm for high-dimensional benchmark functions, Neural Computing and Applications, 31(8), 3779-3803, 2019.