Pamukkale Univ Muh Bilim Derg, 23(5), 628-636, 2017

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Turnuva seçim operatörü kullanan bir havai fişek algoritması

A fireworks algorithm using tournament selection operator

Bilal BABAYİĞİT[1], Sema HASPAYLAN[2]*

[1]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Erciyes Üniversitesi, Kayseri, Türkiye.
bilalb@erciyes.edu.tr
[2]Fen Bilimleri Enstitüsü, Erciyes Üniversitesi, Kayseri, Türkiye.
semahaspaylan@gmail.com

Geliş Tarihi/Received: 06.05.2016, Kabul Tarihi/Accepted: 23.11.2016 doi: 10.5505/pajes.2016.46793
* Yazışılan yazar/Corresponding author Araştırma Makalesi/Research Article

Öz

Son on yılda doğa olaylarından esinlenerek çeşitli sürü zekasına dayalı optimizasyon teknikleri geliştirilmiştir. Kabul edilebilir bir sürede optimuma yakın çözümler üretebilen bu teknikler, fen ve sosyal bilimlerdeki birçok problemin çözümünde başarıyla uygulanmıştır. Havai Fişek Algoritması (HFA), havai fişeklerin patlamalarından esinlenilmiş yeni bir sürü zekası algoritmasıdır. Oldukça yeni sayılabilecek bu teknik, çok çeşitli problemlerde başarılı bir şekilde kullanılmış ve özellikle parçacık sürü optimizasyonu, karınca koloni ve genetik algoritma gibi tekniklere göre daha iyi sonuçlar elde edilmiştir. Elde edilen başarılı sonuçlara rağmen, HFA optimum çözüme ulaşmak için uzun zamana ihtiyaç duymaktadır. Bu hesaplama zamanı yetersizliğini giderebilmek amacıyla bu çalışmada turnuva seçimi kullanan bir HFA önerilmiştir. Turnuva seçme operatörüne sahip HFA'nın başarımı 15 adet nümerik optimizasyon probleminde test edilmiştir. Deneysel sonuçlar önerilen HFA'nın klasik HFA'ya göre hesaplama zamanı ve çözüm kalitesinde önemli performans iyileşmeleri sağladığını göstermiştir.

Anahtar kelimeler: Sürü zekası, Optimizasyon teknikleri, Nümerik optimizasyon

Abstract

In recent decade, several nature-inspired swarm intelligence-based optimization techniques have been improved. These techniques, which give solutions close to optimum in an acceptable time, have been applied successfully to solve the problems in science and social sciences. Fireworks Algorithm (FA), inspired by observing fireworks explosion, is a new swarm intelligence algorithm. This relatively new technique has been utilized to tackle diverse problems and obtained better performance than other popular techniques such as particle swarm optimization, ant colony, and genetic algorithm. Despite the good results obtained, FA requires long computation time to achieve the optimum solution. To eliminate long computation time drawback of FA, in this study, a FA using tournament selection is proposed. The performance of the proposed FA, which involves tournament selection operator, is tested on well-known 15 numerical optimization problems. Experimental results reveal that proposed FA has a significant performance improvement in term of computation time and solution quality in comparison with original FA.

Keywords: Swarm intelligence, Optimization techniques, Numerical optimization

1 Giriş

Havada kuş, denizde balık, karada hayvan sürüleri ve hatta arılar, termitler, karıncalar gibi sosyal böcek kolonileri doğada görülen sürü örnekleridir. Tüm bu sürü örneklerinde, her bir bireyin belirli basit bir görevi olmasına rağmen, sürü veya koloni tek bir vücut gibi çalışabilmekte ve kolektif bir davranış sergilemektedir. Karıncaların yuva kurması ve görev dağılımları, kuş sürülerinin uçuş hareketleri, hayvan sürülerinin göç hareketleri, balık sürülerinin eşgüdümlü hareketleri ve hatta balık sürülerinin tehlike anında göstermiş oldukları ani hareketleri, bu büyük sosyal organizmalarda görülen kolektif davranışlardandır. Sürü zekası doğada gelişen bu kolektif davranışları incelemektedir [1].

Literatürde, sürü zekası davranışlarına dayanan çok çeşitli optimizasyon tekniği önerilmiştir [2]. Çok sayıda sürü zekasına dayanan optimizasyon tekniğinin önerilmesinin nedeni, bu tekniklerin popülasyon tabanlı yapılarından dolayı yerel minimuma takılmaması ve dolayısıyla da küresel optimal çözümleri bulma şanslarının yüksek olmasıdır. Biyolojik olaylardan esinlenilen sürü zekası algoritmaları olduğu gibi biyolojik olmayan olaylardan esinlenerek oluşturulmuş çeşitli sürü zekası algoritmaları da bulunmaktadır. Biyolojik olaylardan esinlenilmiş sürü zekası algoritmalardan en bilinenleri; genetik algoritma (GA), karınca koloni algoritması (KKA), parçacık sürü optimizasyonu (PSO), yapay bağışıklık algoritmaları ve yapay arı koloni algoritmasıdır. Bu teknikler farklı optimizasyon problemine başarılı bir şekilde uygulanmıştır [2]-[14]. Biyolojik olmayan olaylardan esinlenerek oluşturulmuş sürü zekası algoritmalarına; su damlaları algoritması [15], beyin fırtınası optimizasyonu [16], manyetik optimizasyon algoritması [17] ve havai fişek algoritması [18] örnek olarak verilebilir.

Havai Fişek Algoritması (HFA), 2010 yılında sunulmuş [18] ve gece gökyüzünde havai fişeklerin patlamasından esinlenerek oluşturulmuş bir sürü zekası algoritmasıdır. Algoritma temel olarak bir havai fişek patladığı zaman havai fişeğin etrafında oluşan kıvılcımlarla birlikte uzayda en iyi konumu aramak üzerine kuruludur.

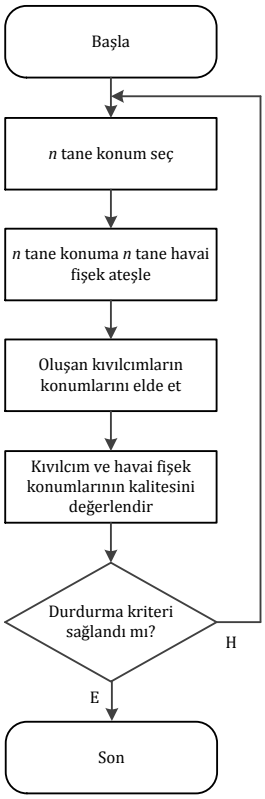
Deneysel sonuçlar, HFA'nın performansının GA, PSO ve KKA'ya göre daha iyi olduğunu ortaya koymuştur [18],[19]. Bu sebeple son zamanlarda araştırmacıların dikkatini çeken HFA, farklı alanlara uygulanmış ve hatta her geçen gün algoritmanın performansını artırmak amacıyla çeşitli geliştirmeler ve karma yapılar önerilmektedir [19].

Başarılı sonuçlara rağmen, HFA'nın en büyük eksiği hesaplama zamanının (çalışma zamanı) fazla olmasıdır [19]. Bu eksikliği giderebilmek amacıyla bu çalışmada klasik HFA'da var olan mesafeye dayalı seçme stratejisi yerine turnuva seçim yöntemine dayanan bir HFA sunulmuştur. Önerilen turnuva seçimine sahip HFA'nın performansı on beş adet nümerik test problemi üzerinde değerlendirmiş ve elde edilen sonuçlar HFA ile elde sonuçlarla karşılaştırılmıştır.

Makale şu şekilde düzenlenmiştir. 2. bölümde havai fişek algoritması anlatılmıştır. 3. bölümde önerilen yeni operatör, 4. bölümde ise deneysel sonuçlar sunulmuştur. Sonuç ve gelecek çalışmalar 5. bölümde verilmiştir.

2 Havai fişek algoritması (HFA)

Gerçek bir havai fişeğin gece patlayıp gökyüzünü aydınlatmasına benzer olarak, HFA algoritmasında havai fişekler potansiyel arama uzayında patlar. Her bir havai fişek patladığında, etrafında kıvılcımlar oluşur. Havai fişekler ve oluşan kıvılcımlar HFA'da arama uzayındaki potansiyel çözümleri temsil etmektedir. HFA, potansiyel uzayı yerel alanda patlama işlemi ile sürekli arayarak en iyi çözümü bulmaya çalışmaktadır [18]. HFA'nın akış diyagramı Şekil 1'de verilmiştir.



Şekil 1: HFA akış diyagramı.

Şekil 1'de görülebileceği gibi, ilk olarak *N* tane başlangıç konumu rastgele seçilir. Sonra, her bir havai fişeğe patlama ve mutasyon operatörleri uygulanır. Patlama operatörü ile havai fişeklerin kıvılcım sayıları ve genlik değerleri belirlenir. Belirlenen bu değerlere göre havai fişekler etrafında kıvılcımlar üretilir. Çeşitliliği sağlamak amacıyla Gauss mutasyon yöntemiyle Gauss kıvılcımları üretilir. Gerekli olduğunda kıvılcım konumları için haritalama kuralı uygulanır. Haritalama ile olurlu bölgede (feasible region) olmayan kıvılcımlar, olurlu bölgeye haritalanır. Daha sonra kıvılcım ve havai fişek konumlarının kalitesi değerlendirilir. Durdurma kriteri sağlanınca algoritma durur. Aksi takdirde yeni nesil patlama için mevcut kıvılcım ve havai fişeklerden en iyi konum korunarak *N* tane konum seçilir.

HFA temel olarak dört operatörden oluşmaktadır [18],[19]. Bunlar; patlama operatörü, mutasyon operatörü, haritalama kuralı ve seçme stratejileridir. Bu operatörler sırayla açıklanmıştır.

2.1 Patlama operatörü

Havai fişek gösterileri gözlemlendiğinde, havai fişeklerin iki tür patlama gerçekleştirdiği görülmektedir. Havai fişekler iyi üretildiğinde, patlama sonrası çok sayıda kıvılcım patlayan havai fişeği merkezileştirecek şekilde dağılır. Bu durumda görkemli bir havai fişek gösteri izlenir. Kötü havai fişek patlamalarında ise oldukça az kıvılcım üretilmekte ve kıvılcımlar uzayda dağılmaktadır. İyi bir havai fişek, aramanın umut vaat eden bölgelerde (optimal konumda veya yakınında) olduğunu gösterir. Bu nedenle havai fişek etrafındaki yerel alan aramalarında daha fazla havai fişek kullanımı daha uygundur. Kötü bir havai fişek, optimal konumdan uzak bir havai fişektir. Kötü havai fişek patlamalarında arama çapı büyük olur. HFA'da iyi bir havai fişek kötü havai fişeğe göre daha fazla kıvılcım üretir ve patlama genliği de küçüktür.

2.1.1 Kıvılcımların sayısı

İyi bir havai fişeğin kıvılcım sayısının fazla olması istenir. HFA'da her bir havai fişek x_i tarafından oluşturulan kıvılcım sayısı

$$S_i = m.\frac{Y_{maks} - f(x_i) + \varepsilon}{\sum_{i=1}^{N}(Y_{maks} - f(x_i)) + \varepsilon} \quad (1)$$

ile belirlenir. Burada S_i her bir havai fişek için kıvılcım sayısı, m N adet havai fişek tarafından üretilen toplam kıvılcım sayısı, Y_{maks} N adet amaç fonksiyon değerinden en kötü amaç fonksiyon değeri, $f(x_i)$ x_i'nin amaç fonksiyon değeri, ε ise pay ve paydadaki ifadelerin sıfır olmasını engellemek amacıyla kullanılan küçük sabit bir değerdir.

Kıvılcım sayısı sınırı

$$\hat{S}_i = \begin{cases} round(a.m), & if\ S_i < a.m \\ round(b.m), & if\ S_i > b.m, a < b < 1 \\ round(a.m), & diğer \end{cases} \quad (2)$$

ile hesaplanır. Burada, $\hat{S}_i$ kıvılcım sayısı sınır değeri, a ve b sabit değerlerdir, $round()$ ise ifade içindeki değeri en yakın tamsayıya yuvarlamak için kullanılır.

2.1.2 Patlama genliği

İyi bir havai fişek patlamasında patlama genliğinin küçük olması istenir. Her bir havai fişek için patlama genliği

$$A_i = \hat{A}.\frac{f(x_i) - Y_{min} + \varepsilon}{\sum_{i=1}^{N}(f(x_i) - Y_{min}) + \varepsilon} \quad (3)$$

dir. Burada, $\hat{A}$ en yüksek patlama genliği, Y_{min} N adet amaç fonksiyon değerinden en iyi amaç fonksiyon değeridir.

2.1.3 Kıvılcımların üretimi

Patlamada kıvılcımları rastgele z adet yönden (boyut) gelen patlama etkilerine maruz kalabilirler. Havai fişek algoritmasında etkilenen bu patlama yönlerinin sayısı,

$$z = round(d.rand(0,1)) \quad (4)$$

ile bulunur Burada, d x konumunun boyutu, $rand(0,1)$ ise [0,1] aralığında rastgele bir sayıdır. x_i havai fişeğinin bir kıvılcımının konumu Algoritma 1 ile elde edilir. Patlama işleminden esinlenerek ilk olarak kıvılcım konumu $\hat{x}_j$ üretilir. Eğer elde edilen kıvılcım konumu potansiyel uzayın dışındaysa algoritmaya göre potansiyel uzaya eşleştirecek şekilde haritalama işlemi yapılır.

Algoritma 1: Kıvılcım konumu elde etme [18].

Kıvılcımın başlangıç konumu: $\hat{x}_j = x_i$
$z = round(d.rand(0,1))$
$\hat{x}_j$'nin z sayıdaki boyutunu rastgele seç
$h = A_i.rand(-1,1)$ ile mesafeyi hesapla
döngü $\hat{x}_k^j$'nin daha önceden seçilmiş z boyutuna kadar
  $\hat{x}_k^j = \hat{x}_k^j + h$
  eğer $\hat{x}_k^j < \hat{x}_k^{min}$ veya $\hat{x}_k^j > \hat{x}_k^{maks}$ ise
    $\hat{x}_k^j = \hat{x}_k^{min} + |\hat{x}_k^j|\%(\hat{x}_k^{maks} - \hat{x}_k^{min})$ ile haritala
  eğersonu
döngüsonu

2.2 Mutasyon operatörü

HFA'da iki tür patlama bulunmaktadır [18],[19]. Birinci patlama için Algoritma 1 kullanılarak patlama kıvılcımlarının konumları elde edilir. İkinci patlama çeşidi ise Gauss patlamasıdır. Gauss patlaması, kıvılcım çeşitliliğini sağlamak amacıyla kullanılmaktadır ve Algoritma 2'de verilmiştir.

Algoritma 2: Gauss kıvılcımının konumunu elde etme [18].

Kıvılcımın başlangıç konumu: $\hat{x}_j = x_i$
$z = round(d.rand(0,1))$
$\hat{x}_j$'nin z sayıdaki boyutunu rastgele seç
$g = Gauss(1,1)$ ile Gauss patlama katsayısını hesapla
döngü $\hat{x}_k^j$'nin daha önceden seçilmiş z boyutuna kadar
  $\hat{x}_k^j = \hat{x}_k^j.\mathrm{g}$
  eğer $\hat{x}_k^j < \hat{x}_k^{min}$ veya $\hat{x}_k^j > \hat{x}_k^{maks}$ ise
    $\hat{x}_k^j = \hat{x}_k^{min} + |\hat{x}_k^j|\%(\hat{x}_k^{maks} - \hat{x}_k^{min})$ ile haritala
  eğersonu
döngüsonu

Algoritma 2'de g ortalaması 1 ve standart sapması 1 olan Gauss dağılımlı rastgele bir sayıdır. Algoritma 2 ile her bir iterasyonda $\hat{m}$ adet Gauss kıvılcımı üretilir.

2.3 Haritalama kuralı

Haritalama kuralı üretilen kıvılcımların olurlu bölgede olmasını sağlamak amacıyla kullanılır. Üretilen kıvılcımların konumu arama uzayı sınırları dışında ise,

$$\hat{x}_k^j = \hat{x}_k^{min} + |\hat{x}_k^j|\%(\hat{x}_k^{maks} - \hat{x}_k^{min}) \quad (5)$$

ile haritalanır. Burada $\hat{x}_k^{maks}$ ve $\hat{x}_k^{min}$ arama uzayının alt ve üst sınır değerleridir.

2.4 Seçme stratejisi

Her bir iterasyonun (patlama neslinin) başlangıcında, havai fişek patlaması için N tane konum seçilmelidir. Algoritmada, en iyi konum olan x^* gelecek patlama nesli için daima tutulur. Daha sonra kıvılcım çeşitliliğini devam ettirecek şekilde birbirlerine olan mesafelerine dayalı olarak mevcut havai fişek ve kıvılcımların içinden *N-1* tane konum seçilir. x_i konumu ve diğer konumlar arasındaki mesafe,

$$R(x_i) = \sum_{j=1}^{K} d(x_i, x_j) = \sum_{j=1}^{K} \|x_i - x_j\| \quad (6)$$

ile tanımlanır. K havai fişek ve kıvılcımların konumlarının tamamının kümesidir. Her bir x_i konumunun seçim olasılığı ise,

$$p(x_i) = \frac{R(x_i)}{\sum_{j \in K} R(x_i)} \quad (7)$$

ile hesaplanır. HFA algoritmasının sözde kodları Algoritma 3'te verilmiştir.

Algoritma 3: HFA sözde kodu [18].

Havai fişekler için rastgele *N* tane konum seç
döngü Durdurma kriteri sağlanmadığı sürece
  N tane konuma *N* tane havai fişeği patlat
  döngü her bir x_i havai fişeği için
    Kıvılcımların sayısını hesapla
    Algoritma 1'i kullanarak havai fişeğin kıvılcımlarının konumunu elde et
  döngüsonu
  döngü *k*=1'den Gauss kıvılcım sayısına kadar
    Rastgele bir x_j havai fişeğini seç
    Algoritma 2'yi kullanarak havai fişek için özel bir kıvılcım üret
  döngüsonu
  Gelecek patlama nesli için en iyi konumu seç ve tut
  Olasılık hesabını göre tüm havai fişek ve üretilen kıvılcımlar arasından *N-1* konum seç
döngüsonu

3 Turnuva yöntemiyle seçim

Gelecek (bir sonraki) nesil için iyi bireylerin seçilmesi istenir. Turnuva yöntemiyle seçimde, S tane rekabetçi arasında turnuva düzenlenmekte ve en iyi birey seçilmektedir. Bu seçim işlemi bir sonraki nesle aktarılacak popülasyon sayısı elde edilinceye kadar uygulanmaktadır [2].

HFA, Denklem (6) ve (7)'de verilen mesafeye dayalı seçim stratejisi kullanmaktadır. Bu seçim stratejisi ile arama uzayının daha az kalabalık bölgelerinden havai fişek ve kıvılcımlar seçilmektedir. Daha az kalabalık bölgelerden yüksek olasılıklı olarak konumların seçilmesine rağmen, bu seçme stratejisi mesafe dayalı hesaplamalardan dolayı her bir iterasyonda yüksek hesaplama maliyeti getirmektedir. Bu ise, HFA'nın optimum çözümü bulması için uzun zamana ihtiyaç duymasına sebep olmaktadır. Bu sebeple, bu çalışmada Denklem (6) ve (7)'deki mesafeye dayalı seçim stratejisinin yerine turnuva yöntemiyle seçim önerilmiştir.

Turnuva yöntemiyle seçimde, havai fişekler, patlama kıvılcımları ve gauss kıvılcımlarından oluşan popülasyondan gelecek nesil için *N* tane birey seçilir. *N* tane bireyin seçilmesi şu şekildedir: Öncelikle popülasyon içindeki en iyi konum aynen korunur. Sonra kalan *N-1* konumun her biri için ayrı ayrı turnuva düzenlenir. Bu işlemde mevcut havai fişek, patlama ve gauss kıvılcımları arasından iki birey rastgele seçilir ve kalite değerinin iyi olanı bir sonraki nesil için alınır.

Klasik turnuva yöntemiyle seçimde *N* adet bireyin turnuva ile seçilmesi yapılmakta iken bu çalışmada en iyi birey korunmakta ve geri kalan *N-1* birey turnuva seçim yöntemiyle seçilmektedir. Böylelikle daha düşük bir hesaplama maliyeti ve dolayısıyla da çalışma zamanında bir iyileşme olmaktadır.

4 Deneyler ve nümerik sonuçlar

Bu çalışmada, klasik HFA ve turnuva seçme operatörüne sahip HFA'nın (M-HFA) performansı literatürde yaygın olarak kullanılan 15 nümerik test fonksiyonu üzerinde incelenmiştir. Tablo 1'de deney çalışması için kullanılan 15 test fonksiyonu verilmiştir.

Tablo 1: Deneylerde kullanılan numerik test fonksiyonları.

	Fonksiyon	Formül	Tip	B	Min	Aralık
f_1	Sphere	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	U	30	0	[-100, 100]^D
f_2	Schwefel's 2.22	$f_2(\vec{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	U	30	0	[-10, 10]^D
f_3	Schwefel's 1.2	$f_3(\vec{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	U	30	0	[-100, 100]^D
f_4	Rosenbrock	$f4(\vec{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} + x_i^2)^2 + (x_i - 1)^2]$	U	30	0	[-30, 30]^D
f_5	Step	$f_5(\vec{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	U	30	0	[-100, 100]^D
f_6	Quartic	$f_6(\vec{x}) = \sum_{i=1}^{D} \mathrm{i}\, x_i^4 + rand[0, 1)$	U	30	0	[-1.28, 1.28]^D
f_7	Schwefel	$f_7(\vec{x}) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	M	30	-12569.5	[-500, 500]^D
f_8	Rastrigin	$f_8(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	M	30	0	[-5.12, 5.12]^D
f_9	Penalized	$f_9(\vec{x}) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$; $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & if\, x_i > a \\ x, & if\, -a \le x_i \le a \\ k(-x_i - a)^m, & if\, x_i < a \end{cases}$; $y_i = 1 + \frac{1}{4}(x_i + 1)$	M	30	0	[-50, 50]^D
f_{10}	Ackley	$f_{10}(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$	M	30	0	[-32, 32]^D
f_{11}	Griewank	$f_{11}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	M	30	0	[-600, 600]^D
f_{12}	Penalized2	$f_{12}(\vec{x}) = \frac{1}{10}\{10\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi y_{1+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	M	30	0	[-50, 50]^D
f_{13}	FletcherPowell2	$f_{13}(\vec{x}) = \sum_{i=1}^{D} (A_i - B_i)^2$; $A_i = \sum_{i=1}^{D} (a_{ij}\, sin\, \alpha_j + b_{ij}\, cos\, \alpha_j)$; $B_i = \sum_{i=1}^{D} (a_{ij}\, sin\, x_j + b_{ij}\, cos\, x_j)$	M	2	0	$[-\pi, \pi]^D$
f_{14}	FletcherPowell5	$f_{14}(\vec{x}) = \sum_{i=1}^{D} (A_i - B_i)^2$; $A_i = \sum_{i=1}^{D} (a_{ij}\, sin\, \alpha_j + b_{ij}\, cos\, \alpha_j)$; $B_i = \sum_{i=1}^{D} (a_{ij}\, sin\, x_j + b_{ij}\, cos\, x_j)$	M	5	0	$[-\pi, \pi]^D$
f_{15}	FletcherPowell10	$f_{15}(\vec{x}) = \sum_{i=1}^{D} (A_i - B_i)^2$; $A_i = \sum_{i=1}^{D} (a_{ij}\, sin\, \alpha_j + b_{ij}\, cos\, \alpha_j)$; $B_i = \sum_{i=1}^{D} (a_{ij}\, sin\, x_j + b_{ij}\, cos\, x_j)$	M	10	0	$[-\pi, \pi]^D$

Tablo 1'de verilen fonksiyonlardan f_1-f_6 tek modlu (Unimodal, U), f_7-f_{15} ise çok modludur (Multimodal, M). U, global optimum dışında yerel optimum değerlere sahip olmayan, M ise global optimum dışında yerel optimumlara sahip fonksiyonlardır. Test fonksiyonlarından f_{13}, f_{14} ve f_{15}'in boyutları sırasıyla 2, 5 ve 10 olarak, diğer fonksiyonlar için ise 30 olarak belirlenmiştir. Eşit şartlarda karşılaştırma yapabilmek için HFA parametreleri N, m, a, b, $\hat{A}$ ve $\hat{m}$ sırasıyla 5, 50, 0.04, 0.8, 40 ve 5 olarak seçilmiştir. İki algoritmanın bir problem üzerinde başarımının gerçekçi bir şekilde karşılaştırılması eşit sayıda amaç fonksiyonun değerlendirilmesi ile mümkündür ve bu şekilde doğru bir çalışma zamanı değeri de elde edilebilir. Yapılan deneylerde, algoritmalar 30 kez ve toplamda 300000 amaç fonksiyon değerlendirme sayısı kadar koşulmuştur. Her bir deneyin en iyi değeri kaydedilmiştir. 30 kez çalıştırma sonucundaki elde edilen ortalama, standart sapma ve en iyi değer Tablo 2'de listelenmiştir.

Tablo 2: *f1-f15* için HFA ve M-HFA ile elde edilen sonuçlar.

Fonksiyon		HFA	M-HFA
f_1	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_2	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_3	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_4	Ortalama değer	21.34	8.87
	Standart sapma	9.47	11.95
	En iyi değer	0.26	0.07
f_5	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_6	Ortalama değer	0.00	0.01
	Standart sapma	0.00	0.01
	En iyi değer	1.57E-05	0.00
f_7	Ortalama değer	-11611.71	-11918.17
	Standart sapma	605.62	725.97
	En iyi değer	-12567.76	-12569.48
f_8	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_9	Ortalama değer	1.90E-32	1.57E-32
	Standart sapma	1.82E-32	2.78E-48
	En iyi değer	1.57E-32	1.57E-32
f_{10}	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_{11}	Ortalama değer	0	0
	Standart sapma	0	0
	En iyi değer	0	0
f_{12}	Ortalama değer	0.02	0,00
	Standart sapma	0.04	0,00
	En iyi değer	0.00	0,00
f_{13}	Ortalama değer	0.00	4.37E-05
	Standart sapma	0.00	6.29E-05
	En iyi değer	5.47E-06	6.07E-07
f_{14}	Ortalama değer	6.31E+00	4.04E+00
	Standart sapma	10.69	9.47
	En iyi değer	0.07	0.01
f_{15}	Ortalama değer	6.13E+02	2.17E+02
	Standart sapma	578.07	151.20
	En iyi değer	22.55	5.57

Ayrıca, 30 deney sonucunda algoritmaların milisaniye cinsinden koşma sürelerinin ortalaması, standart sapması ve standart hatası ile en iyi değeri Tablo 3'te verilmiştir. Deneysel sonuçlar Intel Core i7 3.40 GHz işlemciye ve 4 Gbayt RAM'e sahip olan bir bilgisayarda elde edilmiştir.

f_1-f_{15} için HFA ve M-HFA ile elde edilen en iyi ve çalışma zamanı değerlerinin t istatistik değeri ve anlamlılık karşılaştırması Tablo 4'te verilmiştir. Tablo 4'teki istatiksel analizin amacı M-HFA'nın problem çözme başarımını daha iyi test etmek (karşılaştırmak) ve M-HFA ile elde edilen değerlerin HFA'ya göre önemli olup olmadığını tespit edebilmektir.

Tablo 3: *f1-f15* için HFA ve M-HFA ile elde edilen çalışma zamanı sonuçları.

Fonksiyon		HFA	M-HFA
f_1	Ortalama değer	9055.15	509.68
	Standart sapma	114.92	11.80
	Standart hata	20.98	2.16
	En iyi değer	8823.69	497.12
f_2	Ortalama değer	8088.29	526.56
	Standart sapma	137.78	6.19
	Standart hata	25.15	1.13
	En iyi değer	7830.94	518.50
f_3	Ortalama değer	9022.59	876.37
	Standart sapma	142.03	7.61
	Standart hata	25.93	1.39
	En iyi değer	8689.07	863.02
f_4	Ortalama değer	8753.75	1404.27
	Standart sapma	118.70	25.07
	Standart hata	21.67	4.58
	En iyi değer	8545.87	1371.28
f_5	Ortalama değer	8910.58	759.74
	Standart sapma	111.72	9.23
	Standart hata	20.40	1.69
	En iyi değer	8676.34	746.76
f_6	Ortalama değer	7568.83	1225.86
	Standart sapma	23.87	9.05
	Standart hata	4.36	1.65
	En iyi değer	7530.91	1209.15
f_7	Ortalama değer	8697.35	868.57
	Standart sapma	123.20	5.00
	Standart hata	22.49	0.91
	En iyi değer	8506.97	860.02
f_8	Ortalama değer	9211.11	1316.87
	Standart sapma	96.53	10.08
	Standart hata	17.62	1.84
	En iyi değer	9047.19	1301.43
f_9	Ortalama değer	9697.41	1399.33
	Standart sapma	111.38	14.79
	Standart hata	20.34	2.70
	En iyi değer	9473.54	1379.44
f_{10}	Ortalama değer	9526.75	1214.46
	Standart sapma	137.92	4.14
	Standart hata	25.18	0.76
	En iyi değer	9265.075	1207.33
f_{11}	Ortalama değer	9532.06	1299.55
	Standart sapma	105.97	6.57
	Standart hata	19.35	1.20
	En iyi değer	9322.51	1282.38
f_{12}	Ortalama değer	9276.57	1873.63
	Standart sapma	104.81	6.32
	Standart hata	19.14	1.15
	En iyi değer	9075.80	1859.17
f_{13}	Ortalama değer	1456.17	769.60
	Standart sapma	14.23	11.96
	Standart hata	2.60	2.18
	En iyi değer	1421.94	756.78
f_{14}	Ortalama değer	3109.33	1713.96
	Standart sapma	28.07	21.11
	Standart hata	5.12	3.85
	En iyi değer	3053.14	1690.61
f_{15}	Ortalama değer	7427.82	4946.67
	Standart sapma	42.88	34.59
	Standart hata	7.83	6.32
	En iyi değer	7348.94	4900.76

Tablo 4'teki değerleri elde etmek için çift yönlü t-testinde 0.05 anlamlılık düzeyi (%95 güven aralığı) alınmıştır, '+' M-HFA ile HFA arasındaki farkın istatistiksel olarak anlamlı olduğu, '~' e

ise her ikisi ile aynı doğrulukta sonuçların elde edildiğini göstermektedir.

Tablo 4: *f1-f15* için HFA ile M-HFA'nın *t*-test karşılaştırması.

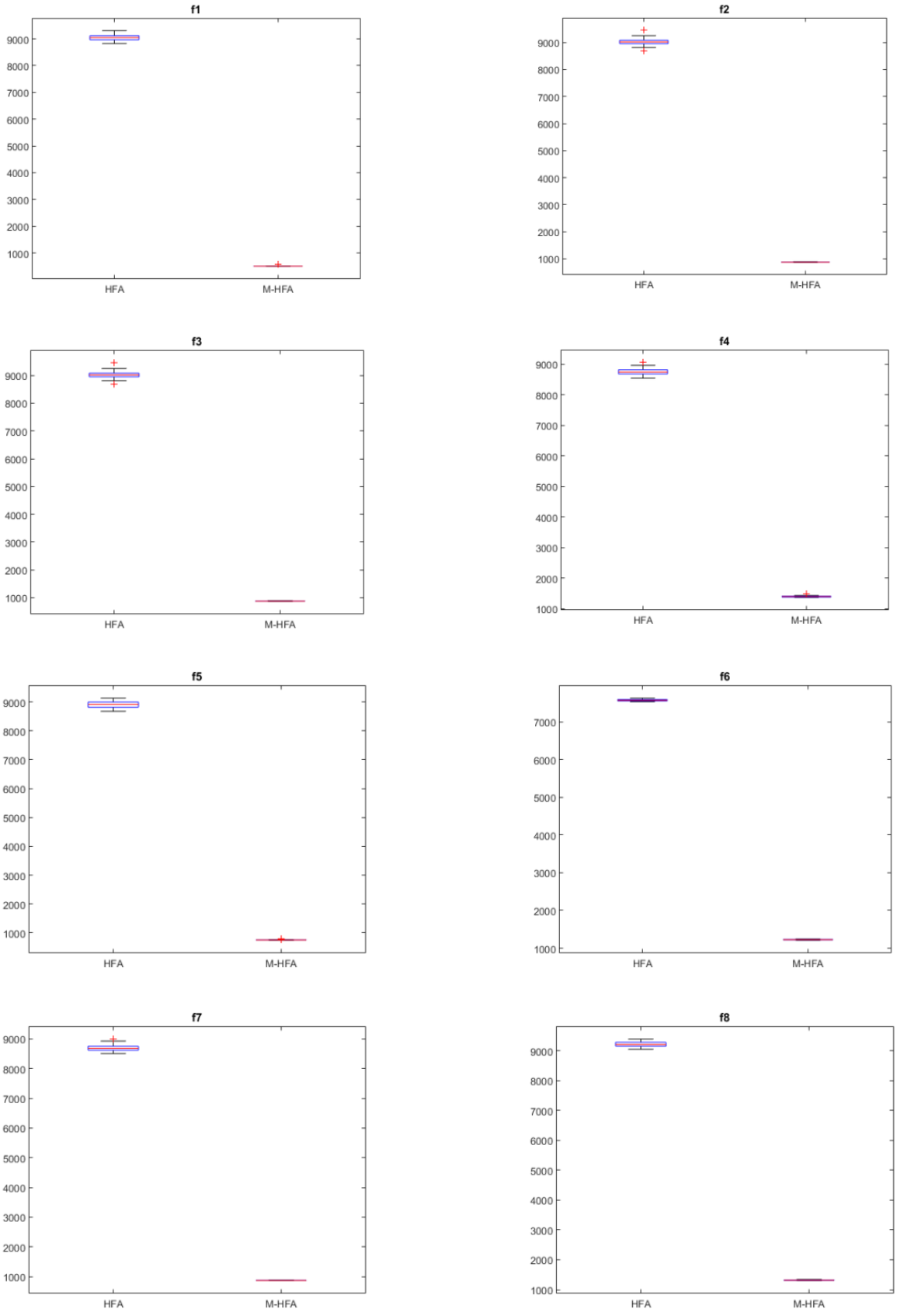
Fonksiyon	En iyi değerler		Çalışma zamanı	
	t değeri	Anlamlılık	*t* değeri	Anlamlılık
f_1	NA	NA	-405.15	+
f_2	NA	NA	-300.31	+
f_3	NA	NA	-313.70	+
f_4	-4.48	+	-331.82	+
f_5	NA	NA	-398.23	+
f_6	NA	NA	-1361.09	+
f_7	-1.78	~	-347.75	+
f_8	NA	NA	-445.51	+
f_9	-1	-	-404.52	+
f_{10}	NA	NA	-329.96	+
f_{11}	NA	NA	-424.70	+
f_{12}	-2.62	+	-386.16	+
f_{13}	-3.86	+	-202.25	+
f_{14}	-0.87	~	-217.63	+
f_{15}	-3.63	+	-246.65	+

Tablo 2'deki elde edilen sonuçlardan ortalama değerler incelendiğinde 8 tanesinde benzer sonuç geri kalan 7 fonksiyonun ikisi hariç 5'inde M-HFA ile daha iyi değerler elde edilmiştir. En iyi değerler incelendiğinde ise 9 fonksiyonda benzer sonuçlar elde edilmiştir 6 fonksiyonda ise M-HFA ile daha iyi sonuçlar bulunmuştur. Her ne kadar ortalama değerlerde M-HFA ile iki fonksiyonda kötü değerler bulunmuş olsa da, en iyi değerler incelendiğinde ikisinde de daha iyi sonuçlar elde ettiği görülebilir. Ancak Tablo 4'teki en iyi değerler incelendiğinde, 15 test fonksiyonun 10 tanesinde istatistiksel olarak benzer sonuçlar elde edilmiş, geri kalan 5 fonksiyonun biri hariç geri kalan dördünde (f_4, f_{12}, f_{13}, ve f_{15}) M-HFA ile daha iyi değerler elde edildiği görülmektedir.

Tablo 3'te listelenen sonuçlar ile Tablo 4'teki *t*-test sonuçları incelendiğinde, öncelikli olarak bütün fonksiyonlar için M-HFA ile çalışma zamanında çok önemli iyileştirmeler elde edildiği görülmektedir. Örneğin, Tablo 3'te f_1 fonksiyonu için M-HFA'nın hesaplama zamanı HFA'dan hem ortalama hem de en iyi değer olarak 17 kat daha azdır. Kutu grafiği bir veya daha fazla veri kümesinin dağılımını görsel şekilde özetlemek için kullanılmaktadır [20]. Bunun için veri kümesinin medyan, en küçük, en büyük, alt çeyrek ve üst çeyrek değeri kutu grafiğinde belirtilmektedir. Tablo 3 ve 4'teki değerler incelendiğinde her ne kadar M-HFA'nın HFA'ya göre hesaplama zamanı bakımından oldukça önemli iyileştirmeye sahip olduğu görülmesine rağmen Tablo 3 ve 4'teki karşılaştırma sonuçlarını daha iyi analiz edebilmek, HFA ve M-HFA ile elde edilen hesaplama değerlerinin düzenli dağılıp dağılmadığını görmek ve ayrıca hangi bölgede değerlerin yığıldığını anlamak amacıyla f_1-f_{15} için kutu grafikleri Şekil 2'de verilmiştir. Şekil 2'de verilen kutu grafiklerdeki her bir fonksiyonun medyan, en küçük, en büyük, alt çeyrek ve üst çeyrek değerleri de Tablo 5'te görülmektedir. Şekil 2 ve Tablo 5'teki verilerden bütün fonksiyonlar için M-HFA ile elde edilen medyan değerlerinin oldukça düştüğü, düşük bir bölgede yığılım yaptığı görülmektedir. Yine Tablo 5'teki değerler incelendiğinde M-HFA ile elde edilen değerlerin büyük bir açıklık göstermediği birbirine yakın olduğu ve düzenli dağıldığı görülmektedir. Şekil 2'deki kutu grafikler incelendiğinde HFA ile elde edilen grafiklerde f_2, f_4, f_7 ve f_{15} için bir, f_3 için iki ayrık değerler bulunmakta iken M-HFA ile elde edilen grafiklerde f_1, f_4, f_5, f_9, f_{11}, f_{12}, f_{14} için bir ayrık değer bulunmaktadır. Fakat M-HFA sonuçlarındaki ayrık değer HFA'daki en iyi değerden oldukça düşüktür. Sonuç olarak, M-HFA ile elde edilen değerler düzgün bir şekilde dağılım içerisindedir ve medyan değerleri de minimum değere yakındır.

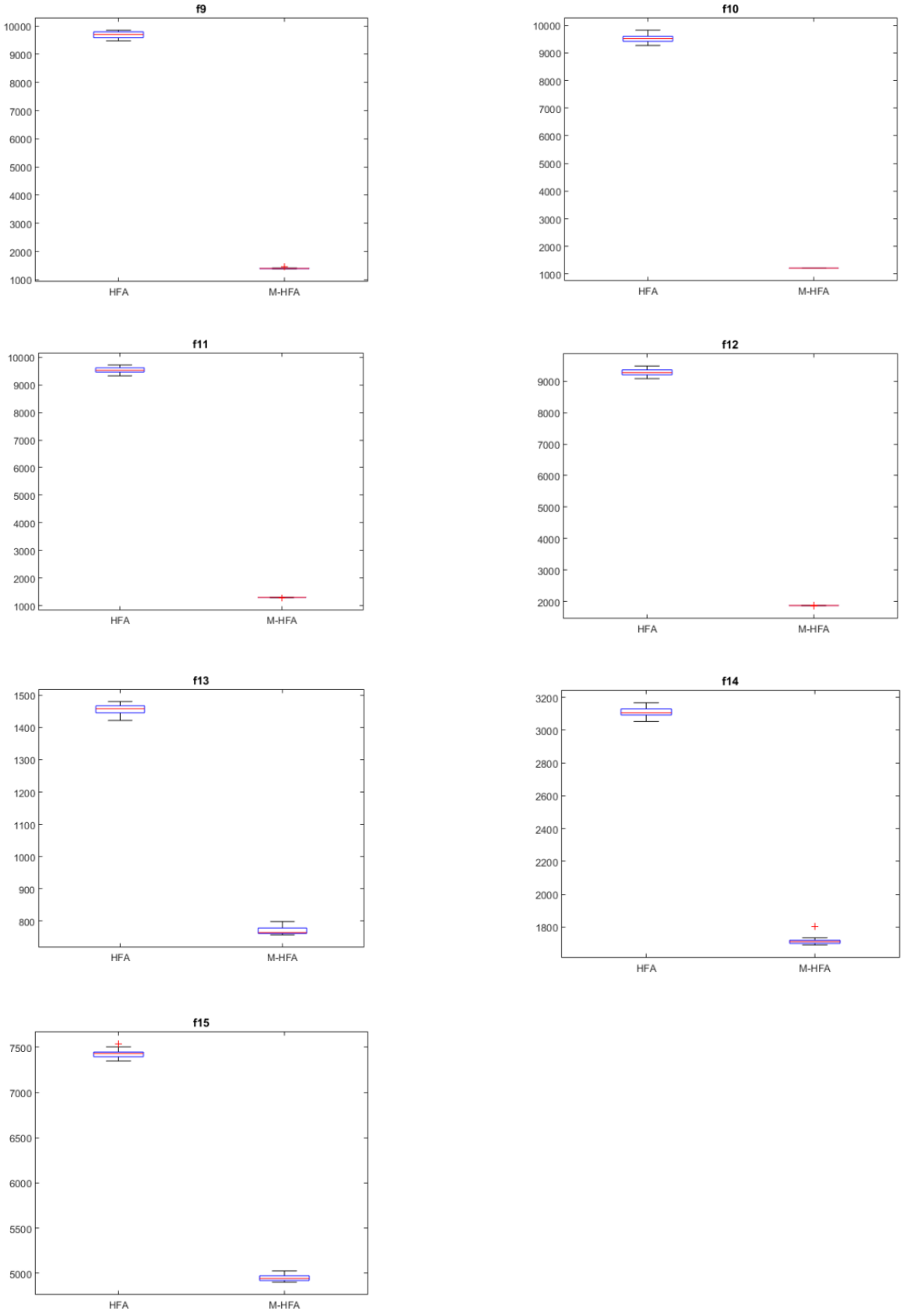
Tablo 5: Kutu grafikleri için elde edilen çalışma zamanı sonuçları.

Fonksiyon	Algoritma	Medyan	En küçük	En büyük	Alt çeyrek	Üst Çeyrek
f_1	HFA	9047.65	8823.69	9309.23	8962.27	9138.80
	M-HFA	507.93	497.12	566.63	503.85	511.16
f_2	HFA	8129.99	7830.94	8306.55	7980.16	8195.38
	M-HFA	525.74	518.50	551.31	522.68	529.10
f_3	HFA	9018.53	8689.07	9456.88	8950.92	9091.06
	M-HFA	875.54	863.02	890.43	870.81	882.04
f_4	HFA	8747.54	8545.87	9068.79	8670.56	8827.81
	M-HFA	1404.11	1371.28	1494.72	1385.15	1419.45
f_5	HFA	8917.45	8676.34	9135.58	8815.88	9003.46
	M-HFA	756.80	746.76	786.71	753.47	762.88
f_6	HFA	7566.40	7530.91	7627.03	7547.91	7586.98
	M-HFA	1226.44	1209.15	1243.20	1218.50	1233.48
f_7	HFA	8686.90	8506.97	8996.34	8603.24	8765.96
	M-HFA	868.72	860.02	879.34	864.27	873.19
f_8	HFA	9203.40	9047.19	9388.92	9138.08	9284.76
	M-HFA	1313.77	1301.43	1341.10	1308.64	1324.32
f_9	HFA	9700.18	9473.54	9852.37	9585.71	9797.80
	M-HFA	1398.34	1379.44	1461.13	1388.36	1405.00
f_{10}	HFA	9516.44	9265.08	9822.15	9415.37	9604.04
	M-HFA	1214.65	1207.33	1221.56	1211.08	1218.35
f_{11}	HFA	9522.48	9322.51	9715.80	9454.55	9631.90
	M-HFA	1300.56	1282.38	1309.14	1296.78	1305.16
f_{12}	HFA	9267.95	9075.80	9478.30	9198.00	9363.48
	M-HFA	1873.30	1859.17	1893.24	1870.14	1877.35
f_{13}	HFA	1458.27	1421.94	1480.73	1443.75	1467.51
	M-HFA	764.32	756.78	798.75	761.61	779.76
f_{14}	HFA	3106.10	3053.14	3166.95	3086.47	3131.73
	M-HFA	1711.40	1690.62	1804.92	1700.46	1721.60
f_{15}	HFA	7431.77	7348.94	7536.98	7394.04	7448.60
	M-HFA	4943.27	4900.76	5025.44	4918.50	4972.31



Şekil 2: f_1-f_{15} için hesaplama zamanı değerlerinin kutu grafikleri.

Şekil 2'nin devamı.



Şekil 2: f_1-f_{15} için hesaplama zamanı değerlerinin kutu grafikleri.

5 Sonuçlar

Bu çalışmada, hesaplama zamanı ve çözüm kalitesinde iyileştirme sağlamak amacıyla turnuva seçim operatörüne sahip bir havai fişek algoritması (M-HFA) önerilmiştir. Önerilen M-HFA'nın performansı tek modlu ve çok modlu 15 karşılaştırma fonksiyonu üzerinde test edilmiştir ve elde edilen sonuçlar HFA ile karşılaştırılmıştır. İstatistiki sonuçlar çalışma zamanında çok önemli iyileştirmeler sağlandığını göstermiştir. Çalışma zamanının yanında elde edilen çözümlerin kalitesinde de iyileştirmeler elde edilmiştir.

Gelecek çalışma olarak, çözüm kalitesini daha da artırmak amacıyla HFA'nın mutasyon ve haritalama operatörü üzerinde çalışılacaktır.

6 Teşekkür

Bu çalışmayı FYL-2016-6471 proje kodu ile destekleyen, ERÜ-BAP koordinasyon birimine teşekkür ederiz.

7 Kaynaklar

[1] Merkle D, Middendorf M. "Swarm intelligence and signal processing". *IEEE Signal Processing Magazine*, 25(6), 152-158, 2008.

[2] Karaboğa D. "*Yapay Zeka Optimizasyon Algoritmaları*". 3. Baskı. Ankara, Türkiye, Nobel Akademik Yayıncılık, 2014.

[3] Akdagli A, Guney K, Karaboga D, Babayigit B. "Finding failed element positions in linear antenna arrays using genetic algorithm". *3rd International Conference on Electrical and Electronics Engineering*, Bursa, Turkey, 3-7 December, 2003.

[4] Chen Y, An A. "Application of ant colony algorithm to geochemical anomaly detection". *Journal of Geochemical Exploration*, 164, 75-85, 2016.

[5] Gao S, Wang Y, Cheng J, Inazumi Y, Tang Z. "Ant colony optimization with clustering for solving the dynamic location routing problem". *Applied Mathematics and Computation*, 285, 149-173, 2016.

[6] Kerdphol T, Fuji K, Mitani Y, Watanabe M, Qudaih Y. "Optimization of a battery energy storage system using particle swarm optimization for stand-alone microgrids". *International Journal of Electrical Power & Energy Systems*, 81, 32-39, 2016.

[7] Chuang LY, Moi SH, Lin Y-D, Yang CH. "A comparative analysis of chaotic particle swarm optimizations for detecting single nucleotide polymorphism barcodes". *Artificial Intelligence in Medicine*, 73, 23-33, 2016.

[8] Gong M, Yan J, Shen B, Ma L, Cai Q. "Influence maximization in social networks based on discrete particle swarm optimization". *Information Sciences*, 367-368, 600-614, 2016.

[9] Babayigit B, Akdagli A, Guney K. "A clonal selection algorithm for null synthesizing of linear antenna array by amplitude control". *Journal of Electromagnetic Waves and Applications*, 20(8), 1007-1020, 2006.

[10] Akdagli A, Guney K, Babayigit B. "Clonal selection algorithm for design of reconfigurable antenna array with discrete phase shifters". *Journal of Electromagnetic Waves and Applications*, 21(2), 215-227, 2007.

[11] Souza SSF, Romero R, Pereira J, Saraiva JT. "Artificial immune algorithm applied to distribution system reconfiguration with variable demand". *International Journal of Electrical Power & Energy Systems*, 82, 561-568, 2016.

[12] Tavana M, Kazemi MR, Vafadarnikjoo A, Mobin M. "An artificial immune algorithm for ergonomic product classification using anthropometric measurements". *Measurement*, 94, 621-629, 2016.

[13] Hong PN, Ahn CW. "Linkage artificial bee colony for solving linkage problems". *Expert Systems with Applications*, 61, 378-385, 2016.

[14] Li B, Zhou C, Liu H, Li Y, Cao H. "Image retrieval via balance-evolution artificial bee colony algorithm and lateral inhibition". *Optik-International Journal for Light and Electron Optics*, 127(24), 11775-11785, 2016.

[15] Shah-Hosseini H. "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm". *International Journal of Bio-Inspired Computation*, 1(1), 71-79, 2009.

[16] Shi Y. "Brain storm optimization algorithm". *Advances in Swarm Intelligence*, 6728, 303-309, 2011.

[17] Tayarani NMH, Akbarzadeh TMR. "Magnetic optimization algorithms a new synthesis". *IEEE World Congress on Computational Intelligence Evolutionary Computation (CEC)*, Hong Kong, China, 1-6 June 2008.

[18] Tan Y, Zhu A. "Fireworks algorithm for optimization". *Advances in Swarm Intelligence*, 6145, 355-364, 2010.

[19] Tan Y. *Fireworks Algorithm A Novel Swarm Intelligence Optimization Method*. 1st ed. New York, USA, Springer, 2015.

[20] Tukey JW. *Exploratory Data Analysis*. Boston, MA, USA, Addison-Wesley, 1977.