T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BÜYÜK ÖLÇEKLİ SÜREKLİ OPTİMİZASYON PROBLEMLERİ İÇİN YARASA ALGORİTMASI TABANLI HİBRİT YÖNTEMLERİN GELİŞTİRİLMESİ

Gülnur YILDIZDAN

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Şubat-2021
KONYA

TEZ KABUL VE ONAYI

Gülnur YILDIZDAN tarafından hazırlanan "BÜYÜK ÖLÇEKLİ SÜREKLİ OPTİMİZASYON PROBLEMLERİ İÇİN YARASA ALGORİTMASI TABANLI HİBRİT YÖNTEMLERİN GELİŞTİRİLMESİ" adlı tez çalışması 01/02/2021 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Halife KODAZ	……..
Danışman Dr. Öğr. Üyesi Ömer Kaan BAYKAN	……..
Üye Prof. Dr. Harun UĞUZ	……..
Üye Doç. Dr. Mehmet HACIBEYOĞLU	……..
Üye Doç. Dr. Hüseyin HAKLI	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Gülnur YILDIZDAN

Tarih: 01.02.2021

ÖZET

DOKTORA TEZİ

BÜYÜK ÖLÇEKLİ SÜREKLİ OPTİMİZASYON PROBLEMLERİ İÇİN YARASA ALGORİTMASI TABANLI HİBRİT YÖNTEMLERİN GELİŞTİRİLMESİ

Gülnur YILDIZDAN

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Ömer Kaan BAYKAN

2021, 108 Sayfa

Jüri
Dr. Öğr. Üyesi Ömer Kaan BAYKAN
Prof. Dr. Halife KODAZ
Prof. Dr. Harun UĞUZ
Doç. Dr. Mehmet HACIBEYOĞLU
Doç. Dr. Hüseyin HAKLI

Yarasa Algoritması (BA), 2010 yılında önerilmiş doğa esinli bir metasezgisel algoritmadır. Yarasa algoritması, diğer metasezgisel algoritmalarda olduğu gibi boyut artışına bağlı performans düşüşleri yaşayan bir algoritmadır. Bu tez çalışmasında, BA'nın yapısal problemlerini azaltmak ve büyük ölçekli problemler üzerindeki performansını artırmak amacıyla iki hibrit algoritma önerilmiştir. İlk hibrit algoritma (MBADE), BA'nın lokal arama becerisine katkı sağlamak amacıyla, BA ve Diferansiyel Evrim (DE) algoritmalarının birlikte kullanılmasıyla oluşturulmuştur. Bu algoritmada, performansa dayalı bir olasılık değerine göre, her iterasyonda bireye uygulanacak olan algoritmaya karar verilir. İkinci hibrit algoritma (BA_ABC), BA'nın global arama becerisini artırmak amacıyla, BA ve Yapay Arı Kolonisi (ABC) algoritmalarının birlikte kullanılmasıyla geliştirilmiştir. Bu hibrit algoritmada, popülasyon iki alt popülasyona ayrılır ve BA ve ABC algoritmaları farklı alt popülasyonlar üzerinde çalışır. Belirli koşullar sağlandığında, alt popülasyonlar arasında bilgi değişimi yapılır. Önerilen hibrit algoritmalar, CEC2005 küçük ölçekli kıyaslama fonksiyonları, CEC2010 büyük ölçekli kıyaslama fonksiyonları ve CEC2011 gerçek dünya problemlerinden seçilen büyük ölçekli problemler üzerinde test edilmiştir. Elde edilen sonuçlar, literatürden seçilen hem BA versiyonlarının hem de diğer metasezgisel algoritmaların sonuçları ile karşılaştırılmıştır. Ayrıca, sonuçlar istatistiksel testler yardımıyla yorumlanmış ve algoritmalar arasında anlamlı bir fark olup olmadığı incelenmiştir. Önerilen hibrit algoritmalar, test edilen kıyaslama fonksiyonlarının çoğunda standart BA algoritmasından daha iyi sonuçlar üretmiştir. MBADE algoritması küçük ölçekli kıyaslama fonksiyonlarında daha başarılı iken, BA_ABC algoritması büyük ölçekli kıyaslama fonksiyonlarda daha başarılı olmuştur. Literatürden seçilen BA versiyonları ve diğer algoritmalarla yapılan karşılaştırmalar, önerilen hibrit algoritmaların başarılı, rekabetçi ve kabul edilebilir sonuçlar ürettiğini göstermiştir.

Anahtar kelimeler: büyük ölçekli optimizasyon, diferansiyel evrim algoritması, sürekli optimizasyon, yapay arı kolonisi algoritması, yarasa algoritması.

ABSTRACT

PhD THESIS

DEVELOPMENT OF HYBRID METHODS BASED ON BAT ALGORITHM FOR LARGE-SCALE CONTINUOUS OPTIMIZATION PROBLEMS

Gülnur YILDIZDAN

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Asst. Prof. Dr. Ömer Kaan BAYKAN

2021, 108 Pages

Jury
Asst. Prof. Dr. Ömer Kaan BAYKAN
Prof. Dr. Halife KODAZ
Prof. Dr. Harun UĞUZ
Assoc. Prof. Dr. Mehmet HACIBEYOĞLU
Assoc. Prof. Dr. Hüseyin HAKLI

Bat Algorithm (BA) is a nature-inspired metaheuristic algorithm proposed in 2010. The Bat algorithm is an algorithm that experiences performance decreases due to dimension increase, as in other metaheuristic algorithms. In this thesis, two hybrid algorithms have been proposed to reduce the structural problems of BA and increase its performance on large-scale problems. The first hybrid algorithm (MBADE) has been created by using BA and Differential Evolution (DE) algorithms together to contribute to the local search capability of BA. In this algorithm, the algorithm to be applied to the individual in each iteration is decided according to a probability value based on performance. The second hybrid algorithm (BA_ABC) has been developed by using BA and Artificial Bee Colony (ABC) algorithms together to increase the global search capability of BA. In this hybrid algorithm, the population is divided into two subpopulations, and BA and ABC algorithms run on different subpopulations. When certain conditions are provided, information exchange is made between subpopulations. The proposed hybrid algorithms have been tested on CEC2005 small-scale benchmark functions, CEC2010 large-scale benchmark functions, and large-scale problems selected from CEC2011 real-world problems. The obtained results have been compared with the results of both BA versions and other metaheuristic algorithms selected from the literature. Besides, the results have been interpreted with the help of statistical tests, and it has been examined whether there is a significant difference between the algorithms. The proposed hybrid algorithms have produced better results than the standard BA algorithm for most of the tested benchmark functions. While the MBADE algorithm is more successful in small-scale benchmark functions, the BA_ABC algorithm is more successful in large-scale benchmark functions. Comparisons with the BA versions and other algorithms selected from the literature show that the proposed hybrid algorithms have produced successful, competitive, and acceptable results.

Keywords: artificial bee colony algorithm, bat algorithm, continuous optimization, differential evolution algorithm, large-scale optimization.

ÖNSÖZ

Tez çalışmam süresince yardımlarını, desteğini ve tecrübesini benden esirgemeyen Danışmanım Dr. Öğr. Üyesi Ömer Kaan BAYKAN'a; fikirleriyle beni yönlendiren tez izleme komitesi üyeleri Prof. Dr. Halife KODAZ ve Doç. Dr. Mehmet HACIBEYOĞLU'na teşekkür ederim.

Bu süreçte gösterdikleri sabır, anlayış ve desteklerinden ötürü başta annem Hatice AVŞAR'a, ikiz kızlarım Defne ve Gökçe YILDIZDAN'a ve diğer tüm aile fertlerime sonsuz teşekkür ederim.

Gülnur YILDIZDAN
KONYA-2021

İÇİNDEKİLER

KISALTMALAR

ABC	:	Yapay arı kolonisi algoritması (Artificial bee colony algorithm)
ACO	:	Karınca kolonisi optimizasyon algoritması (Ant colony optimization algorithm)
aEUS	:	Uyarlanabilir gelişmiş tek boyutlu arama (Adaptive enhanced unidimensional search)
AHDE-csw:	:	Dairesel sürgülü pencereli uyarlanabilir hibrit DE (Adaptive hybrid differential evolution with circular sliding window)
APS	:	Adaptif popülasyona dayalı simpleks algoritması (Adaptive population-based simplex algorithm)
APS9	:	Adaptif popülasyona dayalı simpleks algoritmasının varyantı (Variant of adaptive population-based simplex algorithm)
BA	:	Yarasa algoritması (Bat algorithm)
BA_ABC	:	BA ve ABC algoritmalarından oluşan hibrit algoritma
BAIS	:	Gelişmiş aramalı yarasa algoritması (Bat algorithm with improved search)
BCL	:	Taraflı merkezi öğrenme (Biased center learning)
CC	:	İşbirliğine dayalı birlikte evrim (Cooperative coevolution)
CEC2005	:	Evrimsel hesaplama kongresi 2005 (Congress of evolutionary computation 2005)
CEC2010	:	Evrimsel hesaplama kongresi 2010 (Congress of evolutionary computation 2010)
CEC2011	:	Evrimsel hesaplama kongresi 2011 (Congress of evolutionary computation 2011)
CEC2013	:	Evrimsel hesaplama kongresi 2013 (Congress of evolutionary computation 2013)
CEC2020	:	Evrimsel hesaplama kongresi 2020 (Congress of evolutionary computation 2020)
DE	:	Diferansiyel evrim algoritması (Differantial evolution algorithm)
DLS	:	Yönlü lokal arama (Directional local search)
EBA	:	Üstel sıralı yarasadan esinlenen algoritma (Exponential rank bat-inspired algorithm)
FA	:	Ateşböceği algoritması (Firefly algorithm)
FEs	:	Fonksiyon değerlendirme sayısı (Function evolution number)
GA	:	Genetik algoritma (Genetic algorithm)
GA_MPC	:	Çok ebeveynli çaprazlamalı GA (GA with multi-parent crossover)
GBA	:	Global-en iyi yarasadan esinlenen algoritma (Global-best bat-inspired algorithm)
HBA	:	Hibrit yarasa algoritması (Hybrid bat algorithm)
HBA1	:	Önerilmiş ilk hibrit algoritma (MBADE)
HBA2	:	Önerilmiş ikinci hibrit algoritma (BA_ABC)
$IACO_R$	:	Geliştirilmiş ACO (Improved ant colony optimization)
ILS	:	Yinelemeli yerel arama (Iterative local search)
ILSSIWBA	:	Yinelemeli yerel arama ve stokastik atalet ağırlığına dayalı BA (BA based on iterative local search and stochastic inertia weight)
ISCA	:	Gelişmiş sinüs cosinüs algoritması (Improved sine cosine algorithm)
ITGO	:	Yayılan tümör büyümesi algoritması (Invasive tumor growth optimization algorithm)
IWO	:	İstilacı ot algoritması (Invasive weed optimization)
iBA	:	Ada modelli yarasa algoritması (Island bat algorithm)

LBA	:	Doğrusal sıralı yarasadan esinlenen algoritma (Linear rank bat-inspired algorithm)
LGWO	:	Lévy uçuşlu gri kurt optimizasyon algoritması (Grey wolf optimizer with Lévy flight)
L_{covn}SHADE	:	Kovaryans matrisi öğrenmeli L-SHADE (Covariance matrix learning with L-SHADE)
L-SHADE	:	Doğrusal popülasyon büyüklüğü azaltmalı SHADE (SHADE with linear population size reduction)
LSGO	:	Büyük ölçekli (boyutlu) global optimizasyon (Large-scale global optimization)
MBA	:	Düzenlenmiş BA algoritması
MBADE	:	MBA ve DE algoritmalarından oluşan hibrit algoritma
MDS	:	Çok yönlü arama algoritması (Multi-directional search algorithm)
MIMO	:	Geliştirilmiş kitle hafızalı optimize edici (Modified intellects-masses optimizer)
NBA	:	Yeni yarasa algoritması (Novel bat algorithm)
OBL	:	Muhalefet tabanlı öğrenme (Opposition based learning)
PBA	:	Orantılı yarasadan esinlenen algoritma (Proportional bat-inspired algorithm)
PSO	:	Parçacık sürü optimizasyon algoritması (Particle swarm optimization algorithm)
PSO-CSC	:	Yakınsama hızı denetleyicili PSO (PSO with convergence speed controller)
RBA	:	Rastgele yarasadan esinlenen algoritma (random bat-inspired algorithm)
RBLSO	:	Sıralama tabanlı taraflı öğrenme sürüsü optimize edici (Ranking-based biased learning swarm optimizer)
RPL	:	Sıralı eşli öğrenme (Ranking paired learning)
RSQPSO	:	Rastgele seçimli kuantum davranışlı PSO (Quantum-behaved PSO with random selection)
SADE	:	Kendi kendine uyarlanabilir DE (Self adaptive DE)
SAMODE	:	Kendi kendine uyarlanabilir çoklu operatörlü DE (Self-adaptive multi-operator DE)
SBA	:	Sosyal tabanlı algoritma (Social-based Algorithm)
SCA	:	Sinüs kosinüs algoritması (Sine cosine algorithm)
SCE	:	Karıştırılmış karmaşık evrim algoritması (Shuffled complex evolution algorithm)
SHADE	:	Başarı geçmişine dayalı uyarlanabilir DE (Success-history based adaptive DE)
TBA	:	Turnuva yarasalarından esinlenen algoritma (Tournament bat-inspired algorithm)
VNS	:	Değişken komşuluk arama (Variable neighborhood search)
µDSDE	:	Yerel yönlü aramalı mikro DE (Micro differential evolution with local directional search)

1. GİRİŞ

Günlük hayatta karşımıza çıkan bazı problemlere farkında olarak ya da olmayarak en ideal çözümü bulmaya yöneliriz. Bu yönelimin amacı, kısıtlı kaynakları en ideal şekilde kullanma çabası olarak nitelendirilebilir. Sadece insanlarda değil, bir çok canlı türünde rastlanan bu davranış, belirli bir amacı gerçekleştirmek için belirli kısıtlar ve sınırlar dahilinde en uygun çözümün bulunması işlemi olarak tanımlanır ve optimizasyon olarak isimlendirilir. Bir yere giderken en kısa yolu bulma çabamız, markette en az sayıda poşet kullanacak şekilde ürünleri yerleştirmemiz, elimizdeki sınırlı parayla mümkün olabilecek en kaliteli ve en uygun fiyatlı ürünü almaya çalışmamız günlük hayatta karşılaştığımız optimizasyon probleminden sadece birkaçıdır. Optimizasyon problemleri, sadece günlük hayatla sınırlı kalmayıp matematik, finans, bilgisayar bilimi ve mühendislik gibi daha birçok alanda sıklıkla karşımıza çıkmaktadır.

Optimizasyon kelime anlamı olarak 'en iyileme' anlamına gelmektedir. Optimizasyon, bir problem için verilen koşullar altında olası tüm çözümler içerisinden en iyisini belirleme işi olarak da ifade edilebilir (Alataş, 2007). Bir optimizasyon probleminde hedef, belirlenmiş amaç fonksiyonunu maksimum ya da minimum yapacak karar değişkeni değerlerinin tespitidir. Belirlenen amaca göre, problem maksimizasyon ya da minimizasyon problemi olarak sınıflandırılır. Belli sayıda eşitlik ve eşitsizlik kısıtı içeren bir minimizasyon problemi Denklem 1.1'de gösterilmiştir.

$$
\begin{aligned}
&Minimize: f(x)\\
&Kısıtlar: \ g_i(x_1, x_2, \dots, x_n) \leq 0 \quad i = 1,2, \dots, p\\
&h_i(x_1, x_2, \dots, x_n) = 0 \quad i = 1,2, \dots, q\\
&lb_i \leq x_i \leq ub_i \quad i = 1,2, \dots, n
\end{aligned} \tag{1.1}
$$

$f(x)$ problemin amaç fonksiyonunu, n karar değişkeni sayısını, p eşitsizlik kısıtlayıcısı sayısını, q eşitlik kısıtlayıcısı sayısını, lb_i ve ub_i ise sırasıyla $i.$ değişkenin alt ve üst sınır değerini ifade eder.

Birçok alanda karşılaştığımız optimizasyon problemlerinin çözülmesine ve bilgisayar ortamında modellenmesine olan ihtiyaç, farklı optimizasyon tekniklerinin ortaya çıkmasına neden olmuştur. Optimizasyon problemlerinin çözümünde, kesin çözümü bulmayı garanti eden klasik teknikler ve yaklaşık optimal bir çözüm bulmayı vadeden metasezgisel algoritmalar kullanılır. Klasik teknikte, en iyi değerinin bulunmaya çalışıldığı amaç fonksiyonunun, tam ve doğru bir şekilde matematiksel olarak ifade edilebilmesi gerekir. Ancak, gerçek dünya problemlerinin çoğunda böyle bir matematiksel model mevcut değildir. Ayrıca uygulanabilirliği problemin türüne bağlıdır ve oluşturulan matematiksel model genellikle probleme özgüdür (Akay, 2009). Klasik yöntemler, çoğunlukla tüm problem uzayını tarayarak çözüm bulma eğiliminde oldukları için, diğer yöntemlere göre maliyeti ve süresi daha fazladır. Ayrıca gerçek dünya problemlerinde, çözüm uzayı olası tüm çözümlerin değerlendirilemeyeceği kadar büyüktür ve bunu gerçekleştirebilmek çoğu zaman imkânsızdır. Bu yüzden, tüm olası çözümler yerine, makul bir süre içerisinde çözüm uzayında belli sayıda çözümü değerlendirmek gerekir (Yang, 2010b).

Metasezgisel algoritmalar, bir problemi çözmek için olası çözümlerden daha etkili olanına karar verme amacı ile tanımlanan ve doğal fenomenlerden esinlenen algoritmalardır. Bu algoritmalar, yakınsama özelliğine sahip, kesin çözümü garanti edemeyen ancak makul bir sürede kesin çözüm yakınındaki bir çözümü bulmayı vadeden algoritmalardır. Genelde, en iyiye yakın olan bir çözüme hızlı ve kolay şekilde ulaştıkları için tercih edilirler. Ayrıca metasezgisel algoritmalar probleme özgü değil, genel amaçlıdırlar ve birçok probleme uyarlanabilirler. Metasezgisel algoritmaların en bilinenleri şöyle sıralanabilir: Genetik Algoritma (Genetic algorithm (GA)), Diferansiyel Evrim Algoritması (Differantial Evolution Algorithm (DE)), Yapay Arı Kolonisi Algoritması (Artificial Bee Colony Algorithm (ABC)), Parçacık Sürü Optimizasyon Algoritması (Particle Swarm Optimization Algorithm (PSO)), Karınca Koloni Algoritması (Ant Colony Algorithm (ACO)), Yarasa Algoritması (Bat Algorithm (BA)), Ateşböceği Algoritması (Firefly Algorithm (FA)).

1.1. Tezin Amacı

Günümüzde karşılaşılan gerçek dünya problemlerinin bir kısmı, çok sayıda karar değişkenine sahip büyük ölçekli (boyutlu) global optimizasyon problemleridir (Large-Scale Global Optimization (LSGO)). Bu problemler sürekli, ayrık ya da melez yapıda olabilir. Sürekli problemler, verilen aralıkta sonsuz sayıda giriş değeri alabilir ve bu girişlere karşılık sonsuz sayıda çıkış değeri üretir. Sürekli ve LSGO yapıdaki problemlerde, çözüm uzayı tüm çözümlerin değerlendirilemeyeceği kadar büyüktür ve matematiksel modelinin oluşturulması oldukça zordur. Dolayısıyla, LSGO problemlerinde klasik yöntemleri kullanmak zaman alan maliyetli bir iştir. Bu yüzden, LSGO problemlerinde, tüm olası çözümleri değerlendiren klasik yöntemlerin yerine, makul bir süre içerisinde çözüm uzayında belli sayıda çözümü değerlendiren metasezgisel algoritmaları tercih etmek daha uygun bir yaklaşım olacaktır.

Metasezgisel algoritmalar, küçük ölçekli (boyutlu) problemlerde genellikle iyi performans sergilerler. Ancak, boyutun artışı genelde metasezgisel algoritmalarda performansın düşmesine neden olur. Özellikle sürekli problemlerde problem boyutunun artışı, problem karmaşıklığında üstel bir artışa neden olur. Ayrıca, arama yapılacak bölgenin büyümesi de bölgesel optimuma takılma olasılığını artırır. Bu sebepler, küçük ölçekli optimizasyon problemlerinde başarılı olan bir metasezgisel algoritmanın, büyük ölçekli problemlerde performansının düşmesine neden olabilir. (Tang ve ark., 2009). Bu durum, metasezgisel algoritmaların bu alanlarda performansının artırılması gerekliliğini ortaya koymaktadır. Bu amaçla, büyük ölçekli optimizasyon problemlerinin çözümü için ayrışma esaslı (Cooperative Coevolution (CC)) ya da ayrışma esaslı olmayan (non-decomposition) çeşitli yaklaşımlar önerilmiştir. Ayrışma esaslı olan yaklaşımlar, büyük boyutlu problemi tek ya da çoklu daha küçük boyutlu alt bileşenlere bölen böl-yönet mantığına dayanır. Ayrışma esaslı olmayan yaklaşımlar ise problemi bir bütün olarak ele alır. Algoritma, etkili operatörler kullanılarak, popülasyon boyutunu artırmaya ya da azaltmaya yönelik stratejiler uygulayarak, lokal arama ya da global aramayı destekleyici düzenlemeler yapılarak, başka bir metasezgisel algoritma ile hibrit yapıda kullanılarak geliştirilir ve büyük boyutlu problemlerdeki performansı artırılmaya çalışılır (Mahdavi ve ark., 2015).

Bu tez çalışması kapsamında kullanılmak üzere, metasezgisel algoritma olarak, 2010 yılında Xin She Yang tarafından önerilen Yarasa algoritması (Bat algorithm (BA) (Yang, 2010a)) seçilmiştir. BA algoritması, kodlanması ve anlaşılması kolay, ayarlanması gereken parametre sayısı az, hızlı yakınsama karakteristiği gösteren bir algoritmadır. Bunun yanında, BA'nın büyük ölçekli problemler üzerindeki performansı hakkında yapılmış literatürde kısıtlı sayıda çalışma mevcuttur. Dolayısıyla, bu tez çalışması kapsamında, BA'nın zayıf yönlerini güçlendirmek, LSGO performansını artırmak ve algoritmayı diğer metasezgisel algoritmalarla yarışmacı hale getirmek amacıyla ayrışma esaslı olmayan iki hibrit algoritma önerilmiştir. Bu hibrit algoritmalar sayesinde, literatüre BA'nın LSGO performansı konusunda önemli katkılar sağlanmıştır.

1.2. Tezin Literatüre Katkısı

BA kolay anlaşılması ve kodlanabilmesi gibi avantajlarına rağmen, yapısal problemleri olan bir algoritmadır. Algoritma, o anki mevcut iyi çözüme hızlı bir yakınsama eğilimindedir. Bu karakteristik, yeni konum bulma denkleminin yapısından kaynaklanmaktadır. Özellikle çok modlu (multimodal) fonksiyonlarda bu durum lokal minimuma takılma sorununu ortaya çıkarmaktadır. Yine, algoritmanın lokal ve global arama arasındaki denge iyi değildir. Algoritmadaki ses yayılım oranı (pulse rate (r)) ve ses şiddeti (loudness (A)) parametreleri bu dengeyi belirlemede önemlidir. Yarasa algoritmasında, yarasaların biyolojik davranışına bağlı olarak iterasyon ilerledikçe r'nin değeri artar, A'nın değeri azalır. BA'nın lokal arama bölümü, r parametresinin rastgele üretilen [0-1] aralığındaki bir değerden küçük olmasına bağlı olarak yürütülür. Dolayısıyla, iterasyon ilerledikçe artan r değeri sebebiyle, algoritma başlangıçta yüksek olasılıkla lokal arama, iterasyonun ilerleyen aşamalarında global arama yapma eğilimindedir. Başlangıçta lokal arama yapma ihtimalinin yüksek oluşu, algoritmanın kolaylıkla lokal minimuma takılmasına ve arama uzayında ayrıntılı bir arama yapılamamasına neden olur. A parametresi ise yeni ve daha iyi uygunluk değerli bir çözümün popülasyona dâhil edilip edilmeyeceğine karar vermede etkilidir. A parametresinin rastgele üretilen [0,1] aralığındaki bir değerden daha büyük olması ve uygunluk değeri daha iyi bir çözüm bulunması şartlarına bağlı olarak yeni bulunan çözüm popülasyona dahil edilir. Dolayısıyla iterasyon ilerledikçe A değerinin azalması, yeni başarılı bir çözümün popülasyona dâhil edilmesi ihtimalini azaltır. Bu durum, lokal

minimumdan çıkmayı zorlaştırır ve bulunan daha iyi çözümlerin kaybedilmesine neden olur. Bahsedilen bu yapısal problemlerin olumsuz etkilerini azaltmak ve algoritma performansını büyük ölçekli problemlerde artırmak için bu tez çalışması kapsamında iki hibrit algoritma önerilmiştir.

İlk çalışmada, BA'nın hem global hem de lokal arama yeteneğini geliştirecek bazı düzenlemeler yapılmış, ardından elde edilen gelişmiş algoritmayı lokal arama yeteneği BA'ya göre daha güçlü olan Diferansiyel Evrim Algoritması (DE) ile hibrit kullanan bir mekanizma oluşturulmuştur. Bu hibrit sistemde, ortak bir popülasyon kullanılır ve popülasyon bireyleri için hangi algoritmanın yürütüleceğine algoritmaların performansına bağlı olarak hesaplanan bir olasılık değerine göre karar verilir.

İkinci çalışmada ise, BA'nın performansını artırma amaçlı yapılan bir düzenleme sonrası elde edilen gelişmiş algoritma, global arama yeteneği BA'ya göre daha iyi olan Yapay Arı Kolonisi Algoritması (ABC) ile birlikte çalışan hibrit bir algoritmaya dönüştürülmüştür (Karaboga ve Basturk, 2008 ; Gandomi ve Yang, 2014 ; Yılmaz ve ark., 2014). Bu hibrit algoritmada, popülasyon ikiye bölünür. Popülasyonun ilk yarısında BA algoritması, diğer yarısında ABC algoritması yürütülür. Her belli sayıda iterasyon tamamlandığında, algoritmaların bu geçen süreçteki performansı değerlendirilir. Başarılı olan algoritmanın popülasyonunda bulunan belli sayıdaki başarılı bireyler, diğer algoritmanın popülasyonunda bulunan aynı sayıdaki en başarısız bireylerinin yerine yazılır. Böylece, popülasyonlar arası bilgi değişimi sağlanır.

Önerilen hibrit algoritmaların literatüre yaptığı katkılar şöyle sıralanabilir:

- Yapılan düzenlemeler sayesinde, BA'nın yapısal problemlerinin performans üzerinde neden olduğu olumsuzluklar azaltılmıştır.
- BA'nın zayıf olan global arama yeteneği geliştirilmiştir. Arama uzayını daha etkili araştırması sağlanmıştır.
- BA'nın ve önerilen hibrit algoritmaların farklı kıyaslama fonksiyonu kümelerinde farklı boyutlar için performansı incelenmiştir.
- BA'nın ve önerilen hibrit algoritmaların büyük boyutlu kıyaslama fonksiyonları ve gerçek dünya problemlerindeki performansı değerlendirilmiştir.

- Rekabetçi ve başarılı sonuçlar üreten iki ayrı hibrit algoritma geliştirilmiş ve literatüre kazandırılmıştır.

1.3. Tezin Organizasyonu

Bu tez çalışması altı bölümden oluşmaktadır. İlk bölümde tezin amacı, literatüre katkısı ve organizasyonu hakkında bilgi verilmiştir.

İkinci bölümde, literatürdeki diğer çalışmalar hakkında bilgi verilmiştir.

Üçüncü bölümde, yarasa algoritması, diferansiyel evrim algoritması, yapay arı kolonisi algoritması ve kullanılan istatistik testler hakkında bilgi verilmiştir.

Dördüncü bölümde, bu tez kapsamında önerilen hibrit algoritmaların çalışma ve oluşturulma prensipleri ayrıntılı olarak anlatılmıştır.

Beşinci bölümde, ilk olarak önerilen hibrit algoritmalardaki bazı önemli parametre değerlerinin belirlenmesi için yapılan test işlemlerinden bahsedilmiştir. Ardından, önerilen algoritmalar sırasıyla CEC2005 küçük ölçekli kıyaslama fonksiyonları, CEC2010 büyük ölçekli kıyaslama fonksiyonları ve CEC2011 gerçek dünya problemlerinden seçilen büyük ölçekli problemler üzerinde test edilmiştir. Sonuçlar, birbirleriyle ve literatürdeki farklı algoritmaların sonuçlarıyla karşılaştırılmış ve istatistik testler yardımıyla yorumlanmıştır. Son olarak, standart algoritma ve önerilen hibrit algoritmaların karmaşıklıkları hesaplanarak, birbirleriyle karşılaştırılmıştır.

Altıncı bölümde ise sonuçlar değerlendirilip, önerilerde bulunulmuştur.

2. KAYNAK ARAŞTIRMASI

Literatürde, metasezgisel algoritmaların performansını geliştirmeyi amaçlayan birçok çalışma bulunmaktadır. Bu çalışmalar sürekli, ayrık ve melez yapıdaki farklı problemlere yönelik olarak önerilmiştir. Bu tez çalışması kapsamında, BA'nın büyük ölçekli sürekli optimizasyon problemleri için performansını geliştirmeye yönelik hibrit algoritmalar geliştirilmiştir. Literatürde, BA'nın büyük boyut performansı üzerine yapılmış sınırlı sayıda çalışma bulunmaktadır. Dolayısıyla bu bölümde, öncelikle BA algoritmasının sürekli optimizasyon problemleri üzerindeki performansını iyileştirmeye yönelik yapılmış literatür çalışmaları incelenmiştir. Ardından, büyük ölçekli optimizasyon problemleri üzerinde, BA ve farklı metasezgisel algoritmalar ile yapılan son dönem literatür çalışmalarından bahsedilmiştir.

Meng ve arkadaşları (2015), BA'nın performansını geliştirmeye yönelik yeni bir algoritma önermişlerdir. Önerilen algoritma ile BA'ya Doppler etkisi, Habitat seçim ve Gaussian dağılımlı lokal arama özelliklerini kazandırmışlardır. Doppler etkisi sayesinde, yarasanın avına doğru hareketi süresince değişen ses frekansı matematiksel olarak modellenmiş ve algoritmaya eklenmiştir. Habitat seçim ile yarasanın arama davranışı çeşitlendirilmiştir. Böylece, bir yarasa sadece kendi habitatında değil, farklı habitatlarda da arama yapabilir hale gelmiştir. Bu durum, yarasalara kuantum davranışı kazandırılması ile sağlanmıştır. Çünkü kuantum davranışına sahip bir yarasa, belirli bir olasılıkla tüm arama uzayının herhangi bir konumunda bulunabilir. Son olarak, algoritmaya lokal aramayı geliştirecek Gaussian dağılımlı bir lokal arama denklemi eklenmiştir. Algoritma, yirmi klasik kıyaslama fonksiyonu ve dört tane gerçek dünya mühendislik problemi üzerinde test edilmiştir. Sonuçlar, standart BA ve literatürde iyi bilinen diğer algoritmalarla karşılaştırılmıştır. Önerilen algoritmanın, diğer algoritmalara kıyasla daha başarılı sonuçlar ürettiği görülmüştür (Meng ve ark., 2015).

Yılmaz ve Küçüksille (2015), BA'nın lokal ve global arama yeteneğini geliştirmek için algoritmada üç tane düzenleme yapmıştır. İlk iki düzenleme, BA'nın hız denkleminde yapılmıştır. Denkleme eklenen atalet ağırlığı sayesinde, lokal ve global arama arasındaki denge sağlanmıştır, yani başlangıçta global arama, sona doğru lokal arama daha baskın olmuştur. İkinci olarak, denklemin lokal aramayı temsil eden ikinci bölümünde bir düzenleme daha yapılmıştır. Sadece global en iyi çözüme doğru değil,

popülasyondan rastgele seçilen bir çözüme de yakınsama sağlanmıştır. Bu düzenleme sayesinde, özellikle arama sürecinin sonuna doğru oluşabilecek lokal minimuma takılma probleminin üstesinden gelmeyi amaçlamışlardır. Son olarak, algoritmanın lokal aramasını güçlendirmek için algoritma, istilacı ot algoritmasıyla (invasive weed optimization (IWO)) hibrit hale getirilmiştir. Klasik kıyaslama fonksiyonları ve mühendislik problemleri üzerinde yapılan test işlemleri sonucunda, algoritmanın standart algoritmaya göre çok daha başarılı olduğu tespit edilmiştir. Mühendislik problemlerinden elde edilen sonuçlar ise literatürdeki algoritmalara göre, önerilen algoritmanın daha iyi çözümler sunduğunu ortaya koymuştur (Yılmaz ve Küçüksille, 2015) .

Wang ve arkadaşları (2016), BA'nın global arama yeteneğini güçlendirmek için değişken komşuluk arama (variable neighborhood search (VNS)) kavramını algoritmaya eklemişlerdir. Geliştirdikleri bu algoritma, ilk olarak BA ile arama uzayını araştırır ve arama alanını önemli ölçüde daraltır. Yarasalar, standart BA ile çözümü geliştiremedikleri zaman, değişken komşuluk arama mantığını kullanarak aramaya devam ederler. Değişken komşuluk arama mantığı kullanıldığında, yarasaların yakınında yeni komşu yarasalar oluşturulur ve bu yeni yarasalar sayesinde daha dar ve ümit vadeden alanlar taranmış olur. Önerilen bu algoritma, on altı klasik kıyaslama fonksiyonu üzerinde test edilmiştir ve algoritmanın standart BA'ya göre performansının daha iyi olduğu sonucuna varılmıştır (Wang ve ark., 2016).

Shan ve arkadaşları (2016), Levy uçuşu rastgele adım mantığını ve BA'da yeni bir aday çözüm üretmek için global en iyi değerin yanında, popülasyondan rastgele seçilen dört farklı çözümden de yararlanan gelişmiş bir arama denklemini algoritmaya eklemişlerdir. Bunlara ek olarak, muhalefet tabanlı öğrenme (opposition based learning (OBL)) kavramı da çeşitliliği ve yakınsama kabiliyetini artırmak için algoritmaya dâhil edilmiştir. Önerilen algoritma, on altı klasik kıyaslama fonksiyonu üzerinde test edilmiştir. Elde edilen sonuçlar, bu düzenlemeler sayesinde, BA'nın global arama performansının ciddi şekilde arttığını göstermiştir (Shan ve ark., 2016).

Zhu ve arkadaşları (2016), BA'nın lokal minimuma takılma probleminin üstesinden gelmek için, kuantum davranışlı ve yarasaların ortalama en iyi çözüme doğru yönlendirildiği bir BA algoritması önermişlerdir. Kuantum davranışının algoritmaya

dahil edilmesi ile popülasyon çeşitliliği sağlanmış ve erken yakınsamanın önüne geçilmiştir. Ayrıca, algoritma aramanın erken aşamasında global en iyi çözüme, aramanın sonraki aşamalarında ise ortalama en iyi çözüme doğru yönlendirilmiştir. Bu düzenleme ile BA'nın yakınsama hızı ve verimliliği artırılmıştır. Önerilen algoritma, yirmi dört klasik kıyaslama fonksiyonu üzerinde test edilmiştir ve sonuçlar diğer BA versiyonlarıyla karşılaştırılmıştır. Elde edilen sonuçlar, algoritmanın daha başarılı sonuçlar ürettiğini göstermiştir (Zhu ve ark., 2016).

Tawhid ve Ali (2017), yarasa algoritmasını, çok yönlü arama algoritması (multi-directional search (MDS) algorithm) ile hibrit hale getirerek, çok yönlü yarasa algoritması adını verdikleri yeni bir algoritma önermişlerdir. MDS algoritması, optimum çözüm bölgesine doğru yakınsamayı hızlandırma konusunda başarılı bir algoritmadır. Önerilen algoritmada, başlangıçta standart yarasa algoritmasıyla aramaya başlanır, daha sonra MDS algoritması yarasa algoritmasının yaptığı araştırma bilgilerini kullanarak devam eder. Bu yaklaşım, MDS algoritmasının aramayı rastgele ilk çözüm yerine, iyi bir çözümden başlatmasına yardımcı olur. Hibrit algoritma sayesinde, BA'nın arama yeteneği geliştirilmiş ve hızlandırılmıştır. Hibrit algoritma, on altı kısıtsız kıyaslama fonksiyonu üzerinde test edilmiş ve sekiz ayrı algoritma ile karşılaştırılmıştır. Sonuçlar, algoritmanın çoğu durumda daha iyi çözüm ürettiğini göstermiştir (Tawhid ve Ali, 2017).

Chakri ve arkadaşları (2017), BA'nın performansını artırmak için algoritmada bazı düzenlemeler yapmışlardır. İlk olarak, BA'nın düşük global arama kabiliyeti nedeniyle oluşabilecek erken yakınsama probleminin üstesinden gelmek için, algoritmaya yönlü ekolokasyon mantığını dahil etmişlerdir. İkinci olarak, BA'nın lokal arama denklemine bir azaltım parametresi eklenmiştir. Bu parametre, büyük bir değerle başlar ve iterasyon ilerledikçe azaltılır. Böylece, arama sürecinin başlarında büyük adımlar atılarak daha geniş bir alan değerlendirilir, iterasyon ilerledikçe adım miktarı azalır ve çözüm etrafında daha ayrıntılı arama yapılmış olur. Üçüncü olarak, lokal ve global arama arasındaki dengeyi sağlamak için kullanılan A ve r parametrelerinin hesaplama denklemleri üzerinde bir düzenleme yapılmıştır. Bu düzenleme ile başlangıçta global arama teşvik edilmiş, sona doğru lokal arama ön plana çıkarılmıştır. Ayrıca, iterasyonun sonuna doğru elde edilen iyi çözümlerin kaybedilmesi olasılığı azaltılarak, lokal minimuma düşme sorunu aşılmaya çalışılmıştır. Son olarak ise, yeni

çözümün kabul edilmesi süreciyle ilgili bir düzenleme yapılmıştır. Standart BA'daki global en iyi çözümden daha iyi uygunluk değerine sahip olan çözümün kabul edilmesi mantığı, çözüm kendi uygunluk değerini geliştiriyorsa çözümün kabul edilmesi şeklinde değiştirilmiştir. Bu değişiklik, özellikle aramanın sürecinin sonuna doğru oluşabilecek lokal minimuma takılma sorununu aşmaya yardımcı olur. Önerilen algoritma, CEC2005 ve yirmi adet klasik kıyaslama fonksiyonu için test edilmiştir. Sonuçlar on farklı algoritma ve BA versiyonları ile karşılaştırmıştır. Ayrıca, sonuçlar yapılan parametrik olmayan istatistik testler yardımıyla yorumlanmıştır. İstatistik test sonuçlarına göre, algoritmanın daha iyi performans gösterdiği görülmüştür (Chakri ve ark., 2017).

Cai ve arkadaşları (2018), BA'da global arama kabiliyetini belirleyen hız güncelleme denklemi için üçgen çevirme stratejisini (triangle-flipping strategy) uyarlayarak, gelişmiş bir BA algoritması önermişlerdir. BA'da bir çözümün konumunun güncellenmesi iki şarta bağlı olarak gerçekleşir. Birincisi, [0,1] aralığında rastgele üretilen bir sayıdan ses şiddetinin daha büyük olması, ikincisi ise yeni bulunan çözümün uygunluğunun, mevcut çözümden daha iyi olmasıdır. Bu durum, bazı yarasaların bazı iterasyonlarda güncellenmediği anlamına gelir. Güncellenmeyen yarasalar için hız değeri sıfır olmalıdır. Bu mantıkla, hız ve konum güncelleme denklemlerinde düzenleme yapılmıştır. Ardından, bu düzenlemenin arama aralığını çok küçülttüğü görülmüştür ve bu durumun üstesinden gelmek için üçgen çevirme stratejisi kullanılmıştır. Üçgen çevirme stratejisi üzerinden konum güncellemesi yapan bir denklem önerilmiştir. Bu önerilen denklemde kullanılan en iyi, en kötü ve ortalama konumun denklemde bulunduğu yere göre üç farklı üçgen çevirme stratejisi elde edilmiştir (direk, rastgele, hibrit). Bu düzenleme ile elde edilen BA versiyonları CEC2013 kıyaslama fonksiyonları üzerinde test edilmiştir. Karşılaştırma sonuçlarına göre, en başarılı sonuçların hibrit stratejili BA versiyonundan elde edildiği bulunmuştur. Hibrit stratejili BA, literatürdeki dört ayrı BA versiyonuyla kıyaslanmıştır. Algoritmalar arası sıralamada, önerilen algoritma ilk sırada yer almış ve daha başarılı sonuçlar üretmiştir (Cai ve ark., 2018).

Al-Betar ve Awadallah (2018), BA'nın çeşitliliğini güçlendirmek ve arama süreci boyunca çeşitliliği kontrol etmek amacıyla ada modeli stratejisini algoritmaya eklemişlerdir. Ada modelinde, popülasyon her biri popülasyonun bir bölümüne sahip olan çok sayıda alt popülasyona (veya adalara) bölünür. Algoritma, eş zamanlı veya eş

zamansız olarak her adada çalışır. Önceden tanımlanmış bir sürenin ardından, her adadaki belirli bireyler, belirli göç politikalarına bağlı olarak, verilen göç topolojisi kullanılarak başka bir adaya taşınır. Eş zamanlı göç sürecinde, bireyler aynı anda adalar arasında değiştirir. Eş zamansız göç sürecinde ise, farklı bir zaman diliminde ada bireyleri değiştirilir. Eş zamanlı göç süreci durumunda, adalar arasındaki üyelerin değiş tokuşu için statik veya dinamik olan belirli bir göç topolojisi tanımlanmalıdır. Statik göç topolojisi, adalar arasındaki yolu önceden çizer ve evrim süreci boyunca bu yolu kullanır. Dinamik göç topolojisinde ise, bireyleri adalar arasında değiş tokuş etmek için dinamik bir yol belirler. Göç politikası ise, hangi bireylerin başka bir adaya göç edeceğini belirlemekten sorumludur. Popüler politika, bir adadaki en kötü bireyleri başka bir adadaki en iyi bireylerle değiştirmektir. Bu çalışmada, eşit sayıda birey içeren adalar oluşturulmuş, göç topolojisi olarak dinamik yapıdaki rastgele halka topolojisi kullanılmış ve göç politikası olarak belli sayıdaki en iyi ile en kötü bireylerin değişimini esas alan politika benimsenmiştir. Elde edilen gelişmiş BA algoritması, CEC2005 kıyaslama fonksiyonları üzerinde test edilmiştir. Elde edilen sonuçlar, on yedi farklı algoritma ile karşılaştırılmış ve algoritmanın diğer algoritmalara göre daha başarılı sonuçlar ürettiği görülmüştür (Al-Betar ve Awadallah, 2018).

Liu ve arkadaşları (2018), BA'nın erken yakınsama probleminin üstesinden gelebilmek için hibrit bir algoritma önermişlerdir. BA'nın yerel arama kabiliyetini geliştirmek ve yerel optimumlardan kaçabilmesini sağlamak amacıyla standart algoritmada üç tane düzenleme yapmışlardır. İlk düzenleme, standart BA'nın başlangıç popülasyon çeşitliliğinin artırılması ve algoritmanın yerel optimuma takılma problemini aşmak için, kaotik başlatma stratejisinin algoritmaya eklenmesiyle yapılmıştır. Buna göre, yarasa popülasyonu Sinüzoidal harita kullanılarak kaotik yapıda oluşturulmuştur. İkinci düzenleme, yeni konum hesaplama denkleminde yapılmıştır. Standart yarasa algoritmasında yeni konum, bir önceki iterasyondaki konumla, hızın toplamından elde edilir. Bu denklemde, zaman faktörü sabit yani 1 olarak kabul edilir. Algoritmada, iterasyon ilerledikçe ve yarasalar global en iyi çözüme yaklaştıkça, yarasaların uçuş süresi kısaltılmalıdır. Böylece çözüm etrafında daha küçük adımlar atılarak ayrıntılı bir arama yapılmış olur. Bu nedenle, zaman faktörü doğrusal olmayan ve iterasyona bağlı olarak azalan bir fonksiyon kullanılarak güncellenmiştir. Üçüncü olarak, BA'nın zayıf olan lokal arama becerisini geliştirmek ve lokal minimum noktalara takılmasını önlemek için lokal arama becerisi oldukça güçlü olan extremal optimizasyon

algoritmasıyla hibrit olarak kullanılmıştır. Bu düzenlemeler sonrası elde edilen hibrit algoritma, CEC2014 kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, BA versiyonları ve diğer güçlü algoritmalarla karşılaştırılmıştır. Önerilen algoritmanın, oldukça başarılı sonuçlar ürettiği, istatistiksel testler yardımıyla kanıtlanmıştır (Liu ve ark., 2018).

Gan ve arkadaşları (2018), BA'nın lokal minimuma takılma ve düşük global arama yeteneği sebebiyle istikrarsız sonuç üretme gibi zayıflıklarını gidermek için, yinelemeli yerel arama ve stokastik atalet ağırlığına dayalı gelişmiş bir yarasa algoritması önermişlerdir. Lokal minimuma takılma problemini aşmak için yinelemeli yerel arama (iterative local search (ILS)) adı verilen bir tür yerel arama algoritması, önerilen algoritmaya dâhil edilmiştir. İkinci düzenleme, BA'nın global arama yeteneğini geliştirmek ve istikrarsız sonuç üretmesini gidermek amacıyla, hız denkleminde yapılmıştır. Bu denkleme, önceki hız değerinin etkisini ayarlamak amaçlı stokastik atalet ağırlığı eklenmiştir. Stokastik atalet ağırlığı, rastgele değişkenlerin özelliklerini kullanarak, eylemsizlik ağırlığını ayarlamak için kullanılır. Böylece, algoritma hızlı bir şekilde yerel optimumdan atlayabilir ve optimizasyon sonuçlarının kararlılığını artırılabilir. Son düzenlemede ise, global ve lokal arama arasındaki dengeyi sağlayan ses yayılım oranı (r) ve yeni üretilen bir çözümün kabul edilip edilmeyeceğinin karar verilmesinde etkili olan ses şiddeti (A) denklemlerinin, Chakri ve arkadaşlarının çalışmasındaki (Chakri ve ark., 2017) gibi düzenlenmesi ile yapılmıştır. Geliştirilen algoritma, klasik kıyaslama fonksiyonları, CEC2005 kıyaslama fonksiyonları ve iki gerçek dünya problemi üzerinde test edilmiştir. Sonuçlar, önerilen algoritmanın, optimizasyon doğruluğu, çözme hızı ve yakınsama kararlılığında dikkate değer avantajları olduğunu göstermiştir (Gan ve ark., 2018).

Al-Betar ve arkadaşları (2018), çalışmalarında BA için alternatif seçim mekanizmalarını incelemişlerdir. BA'da yarasalar, avlarını iyileştirmek için arama sırasında bulunan en iyi yarasa konumlarının etrafında rastgele uçarlar. Bunun için, en iyi yarasa konumlarından bir tanesi seçilir. Bu seçim mekanizması, Genetik Algoritma gibi diğer gelişmiş algoritmalarda benimsenen, doğal seçim mekanizmaları kullanılarak geliştirilebilir. Bu nedenle, en iyi yarasa konumunu seçmek için altı seçim mekanizması incelenmiştir: global-en iyi, turnuva, orantılı, doğrusal sıra, üstel sıra ve rastgele. Buna göre, global-en iyi yarasadan esinlenen algoritma (GBA), turnuva yarasalarından

esinlenen algoritma (TBA), orantılı yarasadan esinlenen algoritma (PBA), doğrusal sıralı yarasadan esinlenen algoritma (LBA), üstel sıralı yarasadan esinlenen algoritma (EBA) ve rastgele yarasadan esinlenen algoritma (RBA) olmak üzere, altı BA versiyonu önerilmiştir. Altı versiyonun kendi arasındaki karşılaştırma işlemi, klasik kıyaslama fonksiyonları üzerinde, farklı boyutlar ve parametre değerleri için yapılmıştır. Klasik yaklaşım olan GBA yöntemine kıyasla, sonuçlarda ciddi iyileşme olduğu gözlenmiştir. CEC2005 kıyaslama fonksiyonları için yapılan test işlemleri, literatürdeki on sekiz farklı algoritma ile karşılaştırılmıştır. Sonuç olarak, önerilen BA versiyonlarının rekabetçi sonuçlar ürettiği tespit edilmiştir (Al-Betar ve ark., 2018).

Wang ve arkadaşları (2019), BA'nın karmaşık problemleri çözmedeki zayıflığını gidermek için, çok stratejili yeni bir yarasa algoritması önermişlerdir. Buna göre, algoritmada hız ve yeni konum belirleme denklemlerinde, farklı stratejilere yer verilmiştir. Bu belirlenen stratejiler, orijinal algoritma yapısını, en kötü konumu, en iyi konumu, iki farklı levy uçuşu yapısını, genetik algoritma mantığını, parçacık sürü optimizasyon algoritması mantığını ve atalet ağırlığı katsayısını kullanma şeklinde toplam sekiz tanedir. Algoritmada, her bir yarasa için uygulanacak strateji her iterasyonda farklı olabilir ve olasılık tabanlı bir seçim mekanizmasına göre seçilir. Her stratejiye ait olasılık, ürettiği uygunluk değerinin başarı durumuna göre güncellenir ve zamanla başarılı olan stratejinin seçilme olasılığı artar. Önerilen algoritma, CEC2013 kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, farklı BA versiyonları ve algoritmalarla da karşılaştırılmıştır. Ayrıca, parametrik olmayan istatistiksel testler yardımıyla yorumlanmıştır. Sonuç olarak, önerilen algoritmanın performansının diğer algoritmalara göre daha iyi olduğu tespit edilmiştir (Wang ve ark., 2019).

Chaudhary ve Banati (2019), BA'nın yapısal problemlerini aşmak için Gelişmiş aramalı yarasa algoritması (Bat algorithm with improved search (BAIS)) ve Karıştırılmış karmaşık evrim algoritması (Shuffled complex evolution algorithm (SCE)) ile hibrit yapıda, yeni bir algoritma önermişlerdir. Önerilen bu algoritmada, en iyi çözüm geliştirilemezse, BAIS, rastgele hareket aralığını azaltarak yarasanın kendi etrafında yaptığı yerel aramayı iyileştirir. SCE ise, popülasyonu uygunluk değerlerine göre alt popülasyonlara ayırır. Bu alt popülasyonların her biri bağımsız bir evrim geçirir, ardından popülasyon karıştırılır ve yeni alt popülasyonlara bölünür. Popülasyonun bağımsız alt popülasyonlara bölünmesi, algoritmanın çözüm uzayının

farklı alanlarında arama yapmasına, alt popülasyonları karıştırmak ise, popülasyon içinde bilgi paylaşımına yardımcı olur. SBAIS aynı zamanda popülasyonun genel çeşitliliğini de kontrol eder. Çeşitlilik, belirli bir eşik değerin altına düşerse, popülasyona yeni rastgele çözümler eklenir. Önerilen algoritma, CEC2005 ve CEC2014 kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, diğer BA versiyonlarıyla karşılaştırılmış ve istatistik testler yardımıyla yorumlanmıştır. Yapılan testler, önerilen algoritmanın, karşılaştırılan diğer algoritmalara göre performansının daha yüksek olduğunu göstermiştir (Chaudhary ve Banati, 2019).

Ghanem ve Jantan (2019), BA'nın çeşitliliğini artıran özel bir mutasyon operatörü kullanarak, lokal minimuma takılma probleminin üstesinden gelmeyi amaçlayan, yeni bir algoritma önermişlerdir. Bu algoritmada, BA'nın lokal arama bölümünün devamına, mutasyon işlemi dahil edilmiştir. Limit1 ve Limit2 adı verilen iki parametre değeri ile bu işlem gerçekleştirilir. Eğer Limit1 değeri [0,1] aralığında rastgele üretilen bir sayıdan büyükse yeni konum olarak popülasyondan rastgele seçilen bir bireyin konumu atanır. Ardından Limit2 değerine bakılır. Limit2 değeri [0,1] aralığında rastgele üretilen bir sayıdan büyükse, yeni konum mevcut en iyi ve en kötü konuma göre hesaplanır. Bu yaklaşım, global en iyiden uzak bir konumda lokal minimuma takılma durumunda, kolaylıkla bu konumdan çıkmayı sağlar. Kalan durumda ise (yani rand>Limit1) arama uzayının rastgele bir konumu, yeni konum olarak belirlenir. Önerilen algoritma, yirmi dört kıyaslama fonksiyonu üzerinde test edilmiştir. Farklı boyutlar için elde edilen sonuçlar, standart yarasa algoritmasıyla, literatürde sıklıkla kullanılan algoritmalarla ve BA versiyonlarıyla karşılaştırılmıştır. Sonuçlar incelendiğinde, algoritmanın ümit verici olduğu, standart BA ve birkaç diğer algoritmaya göre daha iyi sonuç ürettiği görülmüştür (Ghanem ve Jantan, 2019).

Lyu ve arkadaşları (2019), BA'nın yerel optimuma takılma eğiliminde oluşunun ve düşük çözüm doğruluğunun önüne geçmek için uyarlanabilir adım kontrolü ve mutasyon mekanizmaları ile geliştirilmiş ve kendi kendini uyarlayabilen bir yarasa algoritması önermişlerdir. Önerilen yöntemde, kullanılan uyarlanabilir adım kontrolü, hem yerel hem de global arama sürecinde, her iterasyondaki adım büyüklüğünü kontrol eder. Orijinal BA, yarasalar için optimum çözümlerdeki değişikliklerin hızlar üzerindeki etkisini temsil etmek için bir frekans değeri kullanır. Önerilen algoritmada ise, sırasıyla bireysel ve tüm yarasalar için optimum çözümlerdeki değişikliklerin hızlar üzerindeki

etkilerini temsil eden iki frekans değeri kullanılır. Yine bu algoritmada, hız denkleminde azalan bir atalet ağırlığı katsayısı kullanılarak başlangıçta global arama, ilerleyen adımlarda yerel aramanın önem kazanması sağlanmıştır. Ayrıca, algoritmanın yerel optimumdan kaçma yeteneğini daha da iyileştirmek için, algoritmaya ses şiddeti A'yı içeren bir mutasyon mekanizması eklenmiştir. Geliştirilen algoritma, klasik kıyaslama fonksiyonları üzerinde test edilmiştir. Elde edilen sonuçlar, iyi bilinen temel algoritmalarla karşılaştırılmıştır. Sonuç olarak, algoritmanın temel algoritmalara göre yüksek doğrulukta, başarılı sonuçlar ürettiği görülmüştür (Lyu ve ark., 2019).

Li ve arkadaşları (2019), BA'nın performansını artırmak amacıyla Levy uçuşu ve ayarlama faktörü temelli gelişmiş bir yarasa algoritması önermişlerdir. Önerilen algoritmada, global ve lokal aramayı etkin bir şekilde dengeleyen ve dinamik olarak azalan bir atalet ağırlığı hız denkleminde kullanılmıştır. Ayrıca, Levy uçuşu arama stratejisi konum güncelleme denklemine eklenmiş, böylece algoritma iyi bir popülasyon çeşitliliği ve global arama yeteneği kazanmıştır. Son olarak, algoritmanın hızını ve doğruluğunu etkili bir şekilde artıran, hız ayarlama faktörü de algoritmaya dâhil edilmiştir. Algoritma, klasik kıyaslama fonksiyonları ve iki adet mühendislik problemi üzerinde test edilmiştir. Sonuçlar, önerilen algoritmanın standart yarasa algoritması ve seçilen diğer metasezgisel algoritmalara göre daha güçlü performansa ve daha yüksek verimliliğe sahip olduğunu göstermiştir (Li ve ark., 2019).

Yu ve arkadaşları (2020), BA'nın global aramada karşılaştığı problemlerin üstesinden gelebilmek için kaos teorisiyle geliştirilmiş bir yarasa algoritması önermişlerdir. Bu algoritmada, kaos haritalama adımları, bir eşik değeri ile kontrol edilir ve atalet ağırlığı kullanılarak yarasaların hızları senkronize edilir. Böylece, yarasa algoritmasının yakınsama hızı ve kararlılığının artırılması sağlanır. Algoritma, farklı test problemleri kullanılarak değerlendirilmiştir ve algoritmanın son dönem metasezgisel algoritmalara göre daha üstün sonuçlar ürettiği görülmüştür (Yu ve ark., 2020).

Chaudhary ve Banati (2020), popülasyonu çok sayıda alt popülasyona bölen ve her bir bağımsız alt popülasyonda farklı parametrelerin kullanıldığı gelişmiş bir yarasa algoritması önermişlerdir. Benzer mantıkta çalışan algoritmalardan farklı olarak, bu algoritmada muhalefet temelli öğrenme kullanılır. Ayrıca, alt popülasyonlar arasındaki bilgi alışverişi, popülasyonun en iyi bireyine en yakın konumdaki bireyin, bir sonraki alt

popülasyona taşınması ile gerçekleştirilir. Hangi alt popülasyondan birey taşınacağının belirlenmesi işlemi ise dört farklı tekniğe göre yapılır. Bu teknikler, başlangıçta aynı olasılıkla seçilir ancak iterasyonlar ilerledikçe başarı durumuna bağlı olarak değişen bir olasılık değerine göre seçim işlemi yapılır. Algoritma, yirmi kıyaslama fonksiyonu üzerinde test edilmiş ve sekiz farklı algoritma ile karşılaştırılmıştır. Sonuçlar, önerilen algoritmanın daha başarılı olduğunu göstermiştir (Chaudhary ve Banati, 2020).

Rauf ve arkadaşları (2020), BA'nın erken yakınsama sorununu aşmak için Sugeno fonksiyonlu bulanık arama ile uyarlanabilir atalet ağırlıklı yarasa algoritmasını önermişlerdir. Bu algoritmada ilk olarak, BA'ya yarasaların hızını etkili bir şekilde artırmak için uyarlanabilir bir atalet ağırlığı eklenmiştir. İkinci olarak, standart BA'nın rastgele arama yöntemi, her yarasanın uygunluğunun kendi deneyimlerine ve komşu yarasaların deneyimlerine göre dinamik olarak ayarlanacağı Sugeno-fonksiyonu düşüş eğrilerini kullanan bir bulanık arama ile değiştirilmiştir. Önerilen algoritma, farklı kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, BA versiyonları ve diğer birçok başarılı algoritmanın son dönem versiyonları ile kıyaslanmıştır. Önerilen algoritmanın, karşılaştırılan algoritmalara göre oldukça başarılı sonuçlar ürettiği gözlenmiştir (Rauf ve ark., 2020).

Literatürde, BA'nın büyük ölçekli optimizasyon problemleri üzerindeki performansını inceleyen sınırlı sayıda çalışma bulunmaktadır. Bu yüzden, devam eden bölümde, son dönemde büyük ölçekli optimizasyon problemleri üzerinde hem BA hemde farklı metasezgisel algoritmalarla yapılmış olan çalışmalara yer verilmiştir.

Sun ve arkadaşları (2018), Balina optimizasyon algoritmasının LSGO problemlerindeki performansını iyileştirmek amacıyla, geliştirilmiş bir algoritma önermişlerdir. Lokal minimumdan kaçmak için Levy uçuş stratejisini, lokal ve global aramayı dengelemek için kontrol parametresi güncellemede kosinüs fonksiyonuna dayalı doğrusal olmayan dinamik bir stratejiyi ve yerel arama kabiliyetini artırmak için popülasyonun en iyi bireyine ikinci dereceden bir enterpolasyon yöntemi uygulayan bu yeni algoritma, boyutları 100 ile 1000 arasında değişen yirmi beş kıyaslama fonksiyonu üzerinde test edilmiştir. Deneysel sonuçlar, önerilen algoritmanın LSGO'da çözüm doğruluğu, yakınsama hızı ve kararlılık açısından diğer son dönem optimizasyon

algoritmalarına kıyasla, üstün performans gösterdiğini ortaya koymuştur (Sun ve ark., 2018).

Omran ve Clerc (2018), APS algoritmasının APS9 adındaki yeni bir versiyonunu mühendislik problemlerini çözmek amacıyla önermişlerdir. Bu yeni algoritmada, standart algoritmanın yansıtma, daraltma ve lokal arama stratejileri revize edilmiştir. Durgunluk algılama mekanizması ve tekrarları silme adımı algoritmaya eklenmiştir. Algoritma, hem küçük hemde büyük boyutlu gerçek dünya problemleri içeren CEC2011 kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, APS9'un karşılaştırıldığı algoritmalardan daha üstün olduğunu göstermiştir (Omran ve Clerc, 2018).

Cai ve arkadaşları (2019), büyük ölçekli optimizasyon problemlerini çözmek için BA'ya altı farklı stratejiye göre düzenleme yapmış ve bir BA versiyon grubu oluşturmuşlardır. Bu stratejilerin birincisinde, hız denklemi standart BA'da olduğu gibidir. Değişiklik yapılmamıştır. İkinci strateji, standart BA'daki hız denkleminde en iyi konuma göre yapılan hesaplamanın, en kötü konuma doğru olacak şekilde değiştirilmesiyle elde edilmiştir. Üçüncü ve dördüncü stratejiler, üçgen çevirme stratejisinin iki ayrı versiyonu alınarak oluşturulmuştur. Beşinci stratejide, yeni konum GA'da daha önce kullanılmış ikili çaprazlama stratejisinden esinlenerek belirlenmiştir. Altıncı stratejide ise, yerel arama denkleminde kavis azaltma stratejisi kullanılmıştır. Bu stratejilerin seçimi, her stratejiye ait olasılık değerine göre yapılır. Burada iki farklı yaklaşım söz konusudur: Sabit olasılık ve Dinamik olasılık. Sabit olasılıkta önceden belirlenmiş bir olasılık değerine göre stratejiye karar verilir. Dinamik olasılıkta ise her stratejinin başarı durumuna göre değişen ve başarılı olanın seçilme ihtimalinin arttığı dinamik bir yapı söz konusudur. Bu çalışmada, bahsedilen stratejiler ve olasılık tercihine göre toplam yedi farklı BA versiyonu oluşturulmuştur. Bunlar, yedi klasik kıyaslama fonksiyonu üzerinde farklı boyutlar (100, 500 ve 1000) için test edilmiştir. En başarılı versiyon belirlenerek, literatürden seçilen algoritmaların sonuçlarıyla karşılaştırılmıştır. Önerilen algoritmanın, büyük boyutlu problemler üzerinde daha başarılı sonuçlar ürettiği görülmüştür (Cai ve ark., 2019).

Deng ve arkadaşları (2019), LSGO problemlerinin çözümü için sıralama tabanlı taraflı öğrenme stratejili gelişmiş bir PSO algoritması (RBLSO) önermişlerdir. Önerilen algoritma iki öğrenme stratejisi içerir: sıralı eşli öğrenme (ranking paired learning

(RPL)) ve taraflı merkezi öğrenme (biased center learning (BCL)). RPL'de, daha kötü parçacıklar, sıralamalarına göre daha iyi parçacıklardan öğrenirler, böylece yakınsama artar. BCL'de ise her parçacık, tüm sürünün uygunluk değeri ağırlık merkezi olarak tanımlanan taraflı merkezden öğrenir. Bu operatör, algoritmanın global arama kabiliyetini güçlendirmek için kullanılır. Geliştirilen algoritma, CEC2010 ve CEC2013 kıyaslama fonksiyonları üzerinde test edilmiş ve son dönem önerilen algoritmalarla karşılaştırılmıştır. Sonuçlar, önerilen algoritmanın büyük ölçekli optimizasyon problemlerinin çözümünde etkili olduğunu göstermiştir (Deng ve ark., 2019).

Long ve arkadaşları (2019), LSGO problemlerini çözmek için Sinüs Kosinüs algoritmasının (SCA) gelişmiş bir versiyonunu önermişlerdir. Önerilen algoritmada, yakınsamayı hızlandırmak amacıyla, konum güncelleme denklemine bir atalet ağırlığı eklenmiştir. Ek olarak, SCA'nın global ve lokal arama dengesini sağlamak için, Gauss işlevine dayalı yeni doğrusal olmayan dönüşüm parametresi azaltma stratejisi kullanılmıştır. Algoritma, yirmi dört kıyaslama problemi, CEC2010 büyük boyutlu test fonksiyonları ve birkaç gerçek dünya mühendislik problemi üzerinde test edilmiştir. Sonuçlar, önerilen algoritmanın standart SCA ve diğer popülasyon temelli algoritmalara göre daha başarılı olduğunu göstermiştir (Long ve ark., 2019).

Mohamed ve arkadaşları (2019), DE algoritmasının performansını artırmak için algoritmaya iki yeni mutasyon stratejisi kazandırmıştır. Ardından, bu mutasyon stratejilerini kullanan iki DE versiyonu oluşturulmuştur. Önerilen mutasyonlar, zorlu ve karmaşık optimizasyon problemlerinde algoritmaların arama yeteneklerini geliştirmek için DE ailesi algoritmalarıyla (SHADE ve LSHADE) birleştirilmiştir. Algoritmaların performansını doğrulamak ve analiz etmek için, CEC2013 ve CEC2017 fonksiyonları kullanılarak sayısal deneyler yapılmıştır. Algoritmaların performansı, LSGO için tasarlanmış CEC2010 fonksiyonları üzerinde de değerlendirilmiştir. Deneysel sonuçlar, elde edilen çözümün sağlamlığı, kararlılığı ve kalitesi açısından her iki mutasyon stratejisinin de oldukça rekabetçi olduğunu göstermiştir (Mohamed ve ark., 2019).

Wu ve arkadaşları (2019), LSGO problemlerini çözmek için yeni bir hibrit algoritma önermişlerdir. Önerilen algoritmada, ilk olarak büyük ölçekli problemi, birkaç küçük ölçekli probleme bölmek için, bir gruplama algoritması kullanılmıştır. İkinci olarak, değiştirilmiş ve kendi kendine uyarlanabilir ayrık tarama yöntemi, tüm arama

alanını kabaca taramak ve ardından aramayı gelecek vadeden bölgelere odaklamak için tasarlanmıştır. Üçüncü olarak, sırasıyla ayrılabilir, kısmen ayrılabilir problemler veya ayrılamaz problemlerin alt problemlerini çözmek için, tek boyutlu arama şemasını veya kovaryans matrisi adaptasyon evrimsel stratejisini uyarlamalı olarak seçen bir hibrit arama stratejisi önerilmiştir. Önerilen algoritmanın performansını değerlendirmek için, CEC2013 kıyaslama fonksiyonları ve on beş LSGO problemi üzerinde test işlemleri yapılmıştır. Sonuçlar, önerilen algoritmanın çözüm doğruluğu açısından, son dönem önerilen algoritmalardan daha etkili olduğunu göstermiştir (Wu ve ark., 2019).

Yıldız ve Topal (2019), LSGO problemlerini çözmek için küçük popülasyon boyutu kullanan, yönlü yerel aramalı mikro DE algoritmasını önermişlerdir. Bu algoritmada, en iyi birey konumunu korur, ikinci en iyi birey DE'nin mutasyon ve çaprazlama süreçlerine maruz kalır ve geri kalan bireyler ise arama alanında rasgele konumlara dağıtılır. Algoritmada, global arama bireylerin rasgele konumlara dağıtılmasıyla yapılırken, lokal arama DE operatörleri ve yönlü yerel arama (Directional Local Search(DLS)) aracılığıyla gerçekleştirilir. 5000 boyuta kadar iki ayrı test kümesi üzerinde elde edilen sonuçlar incelendiğinde, algoritmanın yakınsama oranı ve çözüm kalitesi açısından, karşılaştırıldığı diğer algoritmalardan daha iyi performans gösterdiği tespit edilmiştir (Yildiz ve Topal, 2019).

Zhang ve Zhang (2019), yaptıkları çalışmada gelişmiş bir ABC algoritması ve uyarlanabilir komşuluk arama tekniğini içeren, LSGO problemlerini daha iyi çözmeyi amaçlayan yeni bir algoritma önermişlerdir. Gelişmiş ABC algoritmasında, yerel bir alandan daha iyi yararlanmak için en iyi kılavuzlu (gbest-guided) strateji kullanılırken, global arama kabiliyetini artırmak için eş zamanlı olarak zıt tabanlı (opposite-based) bir öğrenme tekniği kullanılmıştır. Ek olarak, lokal ve global arama sürecini dengelemek için Gaussian ve Cauchy operatörleri ile kendi kendine uyarlanabilir bir komşu arama stratejisi benimsenmiştir. CEC2010 test fonksiyonları üzerinde yapılan performans değerlendirmesi, sadece bu yöntemin etkililiğini ve verimliliğini doğrulamakla kalmamış, aynı zamanda gelişmiş ABC algoritmasının, LSGO'da diğer bazı algoritmalarla karşılaştırılabilir veya onlardan daha iyi bir çözüm kalitesi elde edebilir performansta olduğunu göstermiştir (Zhang ve Zhang, 2019).

Huang ve arkadaşları (2019), LSGO'da parçacık sürü optimizasyonu algoritmasının karşılaştığı yüksek yakınsama hızına bağlı, lokal minimuma takılma problemini aşmak için yakınsama hızını adaptif olarak ayarlayan yeni bir PSO algoritması önermişlerdir. Bu algoritmada, ek bir PSO operatörü olarak algoritmaya eklenen denetleyici, periyodik ve bağımsız olarak uygulanmaktadır. Yakınsama hız denetleyicili PSO'nun performansı, CEC2010 kıyaslama fonksiyonları üzerinde değerlendirilmiştir. Sonuçlar, önerilen denetleyicinin, optimizasyon işlemi sırasında yakınsama hızı ve popülasyon çeşitliliği arasında bir denge sağlanmasına katkı sağladığını göstermiştir. Ayrıca, LSGO'da diğer PSO sürümlerinden ve işbirliğine dayalı birlikte evrim (CC) yöntemlerinden daha iyi performans gösterdiğini de ortaya koymuştur (Huang ve ark., 2019).

Gu ve arkadaşları (2019), Gri Kurt Algoritmasının LSGO problemlerindeki performasını iyileştirmek amacıyla standart algoritmaya üç genetik operatör yerleştirerek, Hibrit Genetik Gri Kurt Algoritmasını önerilmişlerdir. İlk olarak, algoritma muhalefet temelli öğrenme stratejisini kullanan bir popülasyon ile başlatılır. İkinci olarak, algoritmada iyi bireylerin sonraki nesile aktarılması için yapılan seçim işlemine, elit rezervasyon stratejisi birleştirilir. Üçüncü olarak, popülasyonun çeşitliliğini artırmak için, tüm popülasyon, boyut azaltma ve popülasyon bölünmesine dayalı, çapraz operasyon için birkaç alt popülasyona bölünür. Son olarak, popülasyondaki elit bireyler, algoritmanın yerel optimuma düşmesini önlemek için mutasyona uğratılır. Algoritma performansı, klasik fonksiyonlar ve CEC2008 LSGO problemleri üzerinde değerlendirilmiş ve son dönemde önerilen bazı algoritmalarla karşılaştırılmıştır. Sonuçlar, önerilen algoritmanın yakınsama doğruluğunu artırdığını ve standart algoritmanın LSGO problemlerini çözmedeki etkinliğini geliştirdiğini göstermiştir (Gu ve ark., 2019).

Omran ve Al-Sharhan (2019), daha önce sürekli optimizasyon problemleri için geliştirilmiş olan ACO_R algoritmasının iki yeni versiyonunu önermişlerdir. İlk algoritmada, global ve lokal aramayı dengelemek amacıyla Brownian hareketi ve Lévy uçuşu arasında başarıya dayalı bir rastgele yürüyüş seçimi yapan bir mekanizma kullanılmıştır. Önerilen ikinci algoritmada ise, kolonideki çözümleri geliştirmek için yerel arama kullanılır ve ilk algoritmanın memetik bir versiyonudur. Algoritmalar, hem küçük boyutlu hem de büyük boyutlu gerçek dünya problemlerinin yer aldığı CEC2011

kıyaslama fonksiyonları üzerinde test edilmiştir. Sonuçlar, önerilen algoritmaların ACO_R'den daha başarılı performans gösterdiğini ortaya koymuştur (Omran ve Al-Sharhan, 2019).

Ma ve Bai (2020), geliştirdikleri en iyi rastgele mutasyon stratejisini ve çoklu popülasyon kullanımını temel alan yeni bir DE algoritması önermişlerdir. Önerdikleri mutasyon stratejisi, temel vektörü oluşturmak için en iyi bireyi ve rastgele seçilen bir bireyi kullanır. Popülasyon, uygunluk değerlerine göre üç alt popülasyona ayrılır. Her bir alt popülasyon, farklı mutasyon stratejileri ve kontrol parametreleri kullanır. Bireyler ise alt popülasyonlar arasında geçiş yaparak farklı mutasyon stratejileri ve kontrol parametrelerini paylaşır. Önerilen algoritma, CEC2013 LSGO kıyaslama fonksiyonları üzerinde değerlendirilmiş ve beş tane son dönem optimizasyon algoritması ile karşılaştırılmıştır. Karşılaştırma sonuçları, diğer algoritmalara göre önerilen algoritmanın LSGO'da rekabetçi bir performansa ve daha iyi bir verime sahip olduğunu göstermiştir (Ma ve Bai, 2020).

Pathak ve Srivastava (2020), BA'nın büyük ölçekli ve kısıtlı mühendislik tasarım problemlerinin çözümünde uygulanabilirliğini geliştirmek amacıyla, guguk kuşu arama algoritmasını ve Sugeno atalet ağırlığını kullanarak yeni gelişmiş bir yarasa algoritması önermişlerdir. İlk olarak, arama alanında optimum çözümlerden yararlanma becerisine sahip yarasa algoritması, Levy uçuşunu kullanarak global en iyi çözümü keşfetme yeteneğine sahip guguk kuşu arama algoritması ile birleştirilmiştir. İkinci olarak, yarasanın en iyi aday çözümü araştırdığı, yeni bir hız ve konum araştırma denklemi algoritmaya eklenmiştir. Son olarak, Sugeno bulanık atalet ağırlığı, hız güncelleme denklemine dâhil edilerek, yarasa popülasyonunun esnekliği ve çeşitliliği artırılmış ve sonuçların kararlılığı sağlanmıştır. Algoritma, farklı boyut değerleri için on altı klasik kıyaslama fonksiyonu, yedi kısıtlı mühendislik tasarım problemi ve on iki CEC2015 kıyaslama fonksiyonu üzerinde değerlendirilmiştir. Test sonuçları, önerilen algoritmanın optimizasyon doğruluğunun yüksek olduğunu, standart BA'dan daha iyi performans gösterdiğini ve yakınsama açısından rekabetçi sonuçlar elde ettiğini ortaya koymuştur (Pathak ve Srivastava, 2020).

Literatürde, BA'nın performansını geliştirmeye yönelik birçok çalışma olmasına rağmen, LSGO problemleri üzerine yapılan çalışmalar oldukça kısıtlıdır. Ayrıca metasezgisel algoritmalarda yaşanan boyut artışına bağlı performans düşüşleri, BA algoritmasında da devam eden bir sorundur. Bu tez kapsamında, BA'nın farklı boyutlarda farklı test kümeleri üzerinde performansını iyileştirecek iki hibrit algoritma önerilmiştir. Özellikle, LSGO problemlerinde yapılan çalışmalarla literatüre önemli katkı sağlanmıştır. Ayrıca, bu tez çalışması sayesinde, aynı test kümeleri ya da metasezgisel algoritmalarla çalışacak araştırmacılar için yol gösterici olacak ve test sonuçlarını kullanabilecekleri bir kaynak oluşturulmuştur.

3. MATERYAL VE YÖNTEM

3.1. Yarasa Algoritması

Araştırmacılar, doğanın akıllı sistemlerin geliştirilmesi için harika bir fikir kaynağı olduğunu ve birçok karmaşık soruna çözüm sağladığını farkederek, bu alandan esinlenen birçok algoritma önermişlerdir. Doğa esinli algoritmalardan biri de Yarasa Algoritmasıdır.

Yarasalar bine yakın türü olduğu tahmin edilen ve ekolokasyon sayesinde yönlerini bulan etkileyici hayvanlardır. Boyutları, 1.5-2 gr ağırlığında olan minik yaban arısı yarasasından, kanat açıklığı yaklaşık 2 m ve ağırlığı yaklaşık 1 kg olan dev yarasalara kadar oldukça değişkendir. Mikro yarasalar, ekolokasyonu yoğun bir şekilde kullanırken, mega yarasalar genelde kullanmamaktadır. Ekolokasyon bir tür sonar olarak çalışır. Mikro yarasalar, yüksek ve kısa bir ses sinyali yayarlar, bir nesneye çarpmasını beklerler ve bir süre sonra yankı kulaklarına geri döner. Böylece yarasalar bir nesneden ne kadar uzakta olduklarını hesaplayabilirler. Yaydıkları ses sinyali, yaklaşık 5-20 ms kadar sürer ve genellikle 25-150 kHz aralığında sabit bir frekansa sahiptir. Mikro yarasalar, her saniye yaklaşık 10–20 arası ses sinyali yayarlar ancak avlarına yakın uçtuklarında bu ses sinyalinin yayılma oranı artar (saniyede yaklaşık 200 ses sinyaline kadar). Ayrıca, bu yarasalar av ararken yüksek ses sinyali üretir (110 dB aralığında), avlarına yaklaştıklarında ise daha sessiz hale gelirler. İşte mikro yarasaların bu ekolokasyon yeteneği, optimize edilecek amaç fonksiyonu ile ilişkilendirilebilir ve optimum çözümü bulmak için bu davranışı taklit eden bir optimizasyon algoritması formüle edilebilir (Chawla ve Duhan, 2015).

Bu amaçla yola çıkan Yang, yarasaların ekolokasyon yeteneğini ve av bulma konusundaki tipik davranışlarını modelleyerek, 2010 yılında Yarasa Algoritmasını önermiştir (Yang, 2010a).

3.1.1. Yarasa algoritmasının temel adımları

Yang, yarasaların ekolokasyon özelliklerinden bazılarını idealize ederek yarasa algoritmasını oluşturmuştur. Buna göre, yarasa algoritması aşağıdaki kurallara göre gerçekleşir (Yang, 2010a).

- Her bir yarasa, avın ne kadar uzak olduğunu ölçmek için ekolokasyon kullanır.
- Yarasalar avını tespit etmek için, v_i hızıyla, x_i konumuna, sabit bir frekans aralığında $[f_{min}, f_{max}]$, çeşitli dalga boyu (λ) ve ses şiddetinde (loudness (A)) sinyal yayarak uçarlar.
- Yarasalar avlarına olan mesafeyi hesaplarken, gönderdikleri sinyalin dalga boyu ile birlikte, ses yayılım oranını da (pulse emission rate (r)) ayarlayabilirler.
- Ses şiddetindeki çeşitliliğe rağmen, A değerinin büyük bir değere sahip A_0 değerinden sabit minimum bir değere (A_{min}) doğru azaldığı kabul edilir.

3.1.1.1. Popülasyonun oluşturulması

Popülasyon, probleme ait arama uzayında olası çözümleri tutmak için kullanılır. Algoritma için arama uzayı, yarasalar için olası yiyecek kaynaklarının bulunduğu bölge olarak ifade edilebilir. Yiyecek kaynağının yeri bilinmediği için, algoritmada yarasalar arama uzayına rastgele dağıtılır (N sayıda, d boyutta yarasalar) (Yılmaz, 2014). Yarasaların uygunluk değerleri değerlendirilerek, en kaliteli yiyecek kaynağına doğru yönlendirilmesi amaçlanır. Birçok algoritmada olduğu gibi, yarasa algoritmasında da popülasyonun başlatılma süreci, hesaplama karmaşıklığını azaltmak için Denklem 3.1'de görüldüğü gibi basit bir yapıdadır.

$$x_{i,j} = lb_j + \varphi(ub_j - lb_j) \tag{3.1}$$

Denklem 3.1'de; $i = 1,2,\dots,N$ ve $j = 1,2,\dots,d$ olmak üzere $x_{i,j}$ ifadesi i. yarasanın j. boyutunu ifade eder. lb ve ub ise sırasıyla ilgili boyuta ait minimum ve maksimum değerleri ifade eder. φ ise [0,1] aralığında rastgele bir sayıdır.

3.1.1.2. Global arama

Yarasa popülasyonu oluşturulduktan sonra, her bir yarasanın hareket yönü ve adım miktarı belirlenerek yeni konum bilgisi oluşturulur. Frekans, algoritmada adım büyüklüğünü belirler.f_{min} ve f_{max} değerleri sırasıyla frekansın alabileceği minimum ve maksimum değerleri ifade eder. Yeni hız değeri (v_i^t), yarasaya ait bir frekans değeri (f_i), yarasanın konumu (x_i^t), global en iyi bireyin konumu (x^*) ve bir önceki hız değerine (v_i^{t-1}) bağlı olarak hesaplanır. Yeni konum (x_i^{t+1}) ise o anki konuma (x_i^t), yeni hız değerinin (v_i^t) eklenmesi ile elde edilir. Bu adımlar, algoritmanın global arama bölümünü ifade etmektedir. Algoritmada, i. yarasa için frekans, hız ve konum değerleri Denklem 3.2, 3.3 ve 3.4'e göre hesaplanır. Denklem 3.2'deki, β çarpanı [0,1] aralığında rastgele bir değeri ifade eder.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (3.2)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*)f_i \quad (3.3)$$

$$x_i^{t+1} = x_i^t + v_i^t \quad (3.4)$$

3.1.1.3. Lokal arama

Algoritmada, her birey için yeni bir konum belirlendikten sonra r parametresine göre tanımlanmış bir koşula bağlı olarak, o bireyin popülasyondaki mevcut bir yiyecek kaynağı (x_m) etrafında daha ayrıntılı bir arama yapmasına izin verilir. Bu kısım, algoritmanın lokal arama bölümü olarak nitelendirilir. Etrafında arama yapılacak yiyecek kaynağı (x_m), popülasyondan uygunluk değerinin kalitesine göre seçilir. Bu seçim işlemi rulet tekerleği, elitizm vb. mekanizmalar kullanılarak yapılır (Yılmaz, 2014). Bahsedilen lokal arama işlemi, algoritmada Denklem 3.5'e göre yapılmaktadır.

$$x_{yeni} = x_m + \varepsilon\overline{A^t} \quad (3.5)$$

Bu denklemde, $\varepsilon \in [-1, 1]$ aralığında olmak üzere, adımın büyüklüğünü ve yönünü ifade eden rastgele bir sayıdır. $\overline{A^t}$ ise tüm yarasaların o anki ses şiddeti değerlerinin ortalamasını ifade eder.

3.1.1.4. A ve r parametreleri

Yarasa algoritmasında, her yarasa avına yaklaştıkça ve iterasyon ilerledikçe ekolokasyon ile ürettikleri sesin şiddeti ve ses yayılım oranını günceller(Yılmaz, 2014). Yarasalar avına yaklaştığında ses şiddeti azalır, ses yayılım oranı artar. Yarasaların bu karakteristik özelliği Denklem 3.6 ve 3.7'ye göre tanımlanmıştır.

$$A_i^{t+1} = \alpha A_i^t \tag{3.6}$$

$$r_i^{t+1} = r_i^0[1 - e^{(-\gamma t)}] \tag{3.7}$$

α ve γ değerleri birer sabittir. Basitlik için $\alpha = \gamma$ alınabilir (Yang ve Gandomi, 2012). Algoritmanın temel adımlarına ait sözde kodu Şekil 3.1'de verilmiştir.

1. Hedef fonksiyonu $f(x)$ belirle. $x = (x_1, x_2, \ldots, x_n)^T$
2. Yarasa popülasyonunu (x_i) ve başlangıç hızını (v_i) oluştur. $i = (1,2,3, \ldots, N)$
3. x_i için f_i frekansını tanımla.
4. Ses yayılım oranı r_i ve ses şiddeti A_i değerlerini başlangıç durumuna getir.
5. While (t < maksimum iterasyon sayısı)
6. Frekans, hız ve konumu Denklem 3.2, 3.3 ve 3.4'e göre güncelle.
7. if ($rand > r_i$)
8. En iyi sonuçlar arasından bir sonuç seç.
9. Denklem 3.5 ile seçilen en iyi sonucun yakınında lokal bir sonuç üret.
10. Endif
11. If($rand < A_i$) ve ($f(x_i) < f(x^*)$)
12. Yeni sonucu kabul et.
13. Denklem 3.6 ve 3.7'ye göre $r_i$arttır , A_i azalt.
14. Endif
15. Yarasaları sırala ve o anki global en iyi değeri (x^*) bul.
16. Endwhile
17. Sonuçları görüntüle.

Şekil 3.1. BA'nın sözde kodu

3.2. Yapay Arı Kolonisi Algoritması

Yapay Arı Kolonisi Algoritması (ABC), 2005 yılında Karaboğa tarafından önerilmiş sürü zekası temelli bir optimizasyon algoritmasıdır (Karaboga, 2005). Arıların yiyecek arama davranışının modellenmesi ile oluşturulan algoritmada, görevli(işçi), kâşif ve gözcü olmak üzere üç tip arı bulunmaktadır. Algoritmada aşağıdaki kabuller yapılmıştır.

- Her bir besin kaynağı, sadece bir görevli arı tarafından kullanılır.
- Algoritmada besin kaynağı sayısı kadar görevli arı bulunur.
- Görevli arıların sayısı, gözcü arıların sayısına eşittir.
- Tükenmiş besin kaynağının görevli arısı, kâşif arıya dönüşür.
- Besin kaynağının yeri, optimizasyon problemi için olası çözümü, kaynağın nektar miktarı da çözümün kalitesini (uygunluk) belirler.

ABC algoritmasının temel işleyişi şöyledir: Yiyecek arama sürecinin başlangıcında kâşif arılar çevrede rastgele yiyecek kaynağı ararlar. Yiyecek kaynaklarının bulunmasının ardından, kâşif arılar bulduğu kaynağın görevli arısı olur. Görevli arılar, kaynaklarından kovana nektar taşımaya başlarlar. Kovana nektarı boşalttıktan sonra, görevli arılar ya kaynağa geri dönerler ya da kaynakla ilgili bilgiyi gözcü arılara iletmek için dans alanında dans ederler. Bu dans aracılığı ile kaynak hakkında bilgi alan gözcü arılar, belirli bir olasılık ile bir yiyecek kaynağı seçerler ve bu kaynağa yönelerek nektar depolamaya başlarlar. Bir kaynak tükendiğinde, o kaynağın görevli arısı, kâşif arıya dönüşür. Kâşif arı, rastgele bir yiyecek kaynağı bularak nektar depolamaya devam eder. Bu işleyiş sonlandırma kriteri sağlanıncaya kadar tekrarlanır.

3.2.1. ABC algoritmasının temel adımları

3.2.1.1. Başlangıç

Bu adımda, kâşif arılar yiyecek kaynaklarını rastgele aramaya başlarlar. Başlangıçtaki yiyecek kaynakları, her bir parametrenin alt (lb_j) ve üst sınır (ub_j) değerleri arasında Denklem 3.8'e göre üretilir. Denklem 3.8'de, D parametre sayısı ve

FN yiyecek kaynağı sayısı olmak üzere, $i = 1,2,\dots,FN,\ j = 1,2,\dots,D$'dir. Bir yiyecek kaynağının, nektar miktarının tükenip tükenmediğini ifade etmek için kullanılan bir sayaç tanımlanır. Bu kontrol parametresi "limit" olarak adlandırılmaktadır. Tanımlanan limit sayısı, algoritmada önemli bir kontrol parametresidir (Karaboga ve Akay, 2009).

$$x_{i,j} = lb_j + rand(0,1) * (ub_j - lb_j) \tag{3.8}$$

3.2.1.2. Görevli (işçi) arı fazı

Görevli arı, mevcut yiyecek kaynağının (x_i) etrafında, yeni bir yiyecek kaynağı arar. Bulunan yiyecek kaynağını değerlendirir. Yeni yiyecek kaynağındaki nektar miktarı daha iyi ise bu yeni kaynağı kaydeder, eski yiyecek kaynağını siler. Yeni yiyecek kaynağı (v_i) Denklem 3.9'a göre hesaplanır. Denklemde j, $[1, D]$ aralığında rastgele üretilen bir sayıyı, $x_{k,j}$, $k \in \{1,2,\dots,FN\}$ olmak üzere rastgele seçilmiş bir yiyecek kaynağını, $\emptyset$ ise $[-1, 1]$ aralığında rastgele bir sayıyı ifade eder.

$$v_{i,j} = x_{i,j} + \emptyset_{i,j} * (x_{i,j} - x_{k,j}) \tag{3.9}$$

Bulunan kaynağın kalitesi (amaç fonksiyon değeri) hesaplandıktan sonra, kaynağın uygunluk değeri atanır. Denklem 3.10'da f_i amaç fonksiyonunun değerini yani kaynak çözümün kalitesini ifade eder. Mevcut yiyecek kaynağı ile yeni yiyecek kaynağı arasında seçim işlemi yaparken uygunluk değerine göre açgözlü seçim yöntemi uygulanır. Eğer v_i yiyecek kaynağı, mevcut x_i kaynağından daha kaliteli ise v_i yeni kaynak olarak kullanılır ve "limit" sayacı sıfırlanır. Aksi halde, x_i kaynağı ile devam edilir ve "limit" sayacı bir artırılır.

$$uygunluk_i = \begin{cases} \frac{1}{1+f_i} & f_i \geq 0 \\ 1 + |f_i| & f_i < 0 \end{cases} \tag{3.10}$$

3.2.1.3. Gözcü arı fazı

Gözcü arılar görevli arılardan elde ettikleri kaynak bilgilerini toplarlar. Gözcü arı uygunluk değerleri ile orantılı olasılıkla bir kaynak seçer. Her bir kaynak için

Denklem 3.11'e göre bir olasılık değeri hesaplanır. Denklem 3.11'de, p_i, i. kaynağın olasılığını, $uygunluk_i$ ise i. kaynağın uygunluk değerini ifade eder. Her kaynak için hesaplanan olasılık değeri, $[0, 1]$ arasında üretilen rastgele bir değerden büyük ise gözcü arılar Denklem 3.9'a göre yeni bir kaynak üretirler. Mevcut kaynak ile yeni üretilen kaynak açgözlü seçim yöntemine göre değerlendirilir (Karaboga ve Akay, 2009).

$$p_i = \frac{uygunluk_i}{\sum_{k=1}^{FS} uygunluk_k} \tag{3.11}$$

3.2.1.4. Kâşif arı fazı

Eğer bir yiyecek kaynağı tükenmişse veya bu kaynak için belirlenen bir deneme sayısı (limit) boyunca çözüm geliştirilememişse, kaynağın görevli arısı kâşif arı olur ve x_i kaynağı yerine Denklem 3.8'e göre rastgele yeni bir yiyecek kaynağı üretilir.
ABC algoritmasının sözde kodu Şekil 3.2'de verilmiştir.

1. Başlangıç popülasyonunu Denklem 3.8'e göre oluştur.
2. Görevli arıları yiyecek kaynaklarına gönder ve komşu kaynakları araştır (Denklem 3.9 ve 3.10)
3. Gözcü arıları kaynak nektar miktarına göre belirlenen bir olasılıkla (Denklem 3.11) seçilen kaynaklara gönder ve komşu kaynakları araştır (Denklem 3.9, 3.10)
4. Nektarı biten ve terk edilen kaynaklar yerine rastgele yeni bir kaynak oluştur (Denklem 3.8)
5. Popülasyondaki en iyi kaynağı kaydet
6. Durdurma kriteri sağlanmadıysa 2, sağlandıysa 7. adıma git
7. En iyi sonucu belirle ve bitir

Şekil 3.2. ABC'nin sözde kodu

3.3. Diferansiyel Evrim Algoritması

Diferansiyel Evrim Algoritması (DE), 1995 yılında Price ve Storn tarafından önerilmiş popülasyon tabanlı bir algoritmadır (Storn ve Price, 1997). Algoritma, basit ama güçlüdür ve özellikle sürekli optimizasyon problemlerinin çözümünde oldukça başarılı bir algoritmadır. Temeli Genetik Algoritmaya (GA) dayanır. Ancak klasik GA'dan farklı olarak değişkenler gerçek değerleri ile temsil edilir. Zaman zaman GA'da da gerçek değerli değişkenler kullanılabilmesine rağmen, asıl fark genetik operatörlerin kullanımında yapılan bir takım değişikliklerdir. Çaprazlama, mutasyon ve seçim operatörleri DE'de de kullanılır. Ancak GA'dan farklı olarak, her bir operatör, popülasyondaki her bir bireye sırayla uygulanmaz. DE'de bireyler tek tek ele alınır ve rastgele seçilen bireyler yardımıyla yeni birey elde edilir (mutasyon ve çaprazlama işlemi). Ardından mevcut birey, yeni bireyle karşılaştırılır ve uygunluğu daha iyi olan birey popülasyona dâhil edilir (seçim işlemi) (Keskintürk, 2006).

DE'de kullanılan mutasyon, algoritma performansını geliştiren ve onu daha gürbüz hale getiren önemli bir işlemdir. Yine algoritmanın hızlı, basit, kolay kullanılabilir, değiştirilebilir ve kodlanabilir olması, global arama kabiliyetinin yüksek olması, hesaplama maliyetinin düşük olması, tamsayı, ayrık ve karma parametre optimizasyonuna izin vermesi v.b. özellikleri nedeniyle sıklıkla tercih edilir (Karaboga, 2004).

3.3.1. Diferansiyel evrim algoritmasının temel adımları

DE algoritması çaprazlama, mutasyon ve seçim gibi operatörleri kullanan popülasyon tabanlı bir algoritmadır. Algoritmanın önemli parametreleri:

- N (Popülasyondaki birey sayısı),
- C_r (Çaprazlama oranı),
- F (Ölçek faktörü) olarak sayılabilir.

DE algoritması, N adet rastgele oluşturulmuş D boyutlu gerçek değerli parametre vektörleri ile başlatılır. Bu parametre vektörleri, optimizasyon problemi için

aday çözümleri ifade eder. Popülasyon bireyleri oluşturulduktan sonra çaprazlama, mutasyon ve seçim işlemleri, optimal sonucu elde etmek için bitirme koşulu sağlanıncaya kadar bireylere uygulanır.

$G = 0,1,2,3, \ldots\ldots, G_{max}$ sonraki nesilleri ifade etmek üzere, şu anki jenerasyonun i. bireyi Denklem 3.12'de verildiği gibi ifade edilir.

$$X_{i,G} = \left(X_{i,G}^{1}, X_{i,G}^{2}, X_{i,G}^{3}, \ldots, X_{i,G}^{D}\right) \quad j = 1,2,3, \ldots, D \tag{3.12}$$

3.3.1.1. Popülasyonun oluşturulması

DE algoritmasında, popülasyonu oluşturmak için N adet birey Denklem 3.13' de verildiği gibi oluşturulur. Denklemde l ve u değerleri sırasıyla o parametre için alt ve üst sınır değerlerini ifade eder.

$$\forall_i \leq N \ \Lambda \ \forall_j \leq D: \quad x_{i,G}^{j} = x_i^l + rand[0,1] * (x_i^u - x_i^l) \tag{3.13}$$

3.3.1.2. Mutasyon işlemi

DE algoritmasında, her jenerasyonda, her bir $X_{i,G}$ bireyi için mutasyon oparatörü kullanarak mutant vektör $V_{i,G}$ oluşturulur. $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, $X_{r4,G}$, $X_{r5,G}$ popülasyondan seçilen rastgele bireyler ve F arama işleminin ne kadar uzakta yapılacağını belirleyen bir ölçek faktörüdür. $X_{best,G}$, popülasyonun o anki en iyi bireyidir. Buna göre, mutasyon sonucu üretilen mutant vektörü oluşturmak için en çok kullanılan yaklaşımlar Denklem 3.14-3.20'de verilmiştir (Qin ve Suganthan, 2005 ; Epitropakis ve ark., 2011).

1. DE/rand/1 $V_{i,G} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G})$ (3.14)
2. DE/best/1 $V_{i,G} = X_{best,G} + F * (X_{r1,G} - X_{r2,G})$ (3.15)
3. DE/rand/2 $V_{i,G} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G}) + F * (X_{r4,G} - X_{r5,G})$ (3.16)
4. DE/best/2 $V_{i,G} = X_{best,G} + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$ (3.17)
5. DE/current-to-best/1 $V_{i,G} = X_{i,G} + F * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$ (3.18)
6. DE/current-to-rand/1 $V_{i,G} = X_{i,G} + F * (X_{r1,G} - X_{i,G}) + F * (X_{r2,G} - X_{r3,G})$ (3.19)

7. DE/rand-to-best/2 $V_{i,G} = X_{i,G} + F * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$ (3.20)

3.3.1.3. Çaprazlama işlemi

Mutasyon sonrasında oluşan $V_{i,G}$ mutant vektörü ve onun ilgili olduğu $X_{i,G}$ hedef vektörünün her bir çiftine çaprazlama uygulanır. DE'de kullanılan bu çaprazlama işlemi, GA'da kullanılan düzenli çaprazlamaya (uniform crossover) benzer mantıktadır. Ancak, düzenli çaprazlamada her bir gen ayrı değerlendirilir ve eşit olasılıkla iki ebeveyn kromozomun birinden seçilir. DE'de ise eşit olasılık yerine C_r olasılığı kullanılır. 0 ile 1 arasında üretilen rastgele sayı C_r'den küçükse gen $V_{i,G}$'den, aksi halde $X_{i,G}$'den seçilir (Keskintürk, 2006). Buna göre, $U_{i,G}$ deneme vektörü Denklem 3.21'e göre elde edilebilir.

$$U_{i,G}^{j} = \begin{cases} V_{i,G}^{j}, & Eğer\ rand_{i,j}[0,1] \leq C_r\ \ ya\ da\ \ j = j_{rand} \quad j = 1,2 \ldots\ldots, D \\ X_{i,G}^{j}, & diğer \end{cases} \tag{3.21}$$

Denklem 3.21'de C_r, [0,1] aralığında değer alan çaprazlama kontrol parametresidir. Mutant vektörden bir deneme vektörü için parametre oluşturma olasılığını ifade eder. $j_{rand} \in [0, D]$ rastgele seçilmiş bir tamsayıdır ve deneme vektörünün ($U_{i,G}^{j}$) en az bir boyutunun hedef vektörden ($X_{i,G}^{j}$) farklı olmasını sağlar.

3.3.1.4. Seçim işlemi

Seçme işlemi, deneme ($U_{i,G}^{j}$) ve hedef vektör ($X_{i,G}^{j}$) arasında seçim yapmayı sağlar. Şayet deneme vektörü, hedef vektörden daha iyi bir uygunluk değerine sahipse bu vektör bir sonraki nesile aktarılır (Denklem 3.22).

$$X_{i,G+1} = \begin{cases} U_{i,G}, & Eğer\ f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & diğer \end{cases} \tag{3.22}$$

DE algoritmasının sözde kodu Şekil 3.3'de verilmiştir.

1. Başlangıç popülasyonunu oluştur ($X_{i,G}$, $i = 1,2, \dots, N$)
2. While G ≤ G_{max}
3. For i=1 : N
4. Denklem 3.14-3.20 arasından seçilmiş olan bir yaklaşıma göre mutant vektörü oluştur ($V_{i,G}$).
5. Deneme vektörünü ($U_{i,G}$) Denklem 3.21'e göre oluştur.
6. Denklem 3.22'ye göre seçim işlemini yap
7. End for
8. G=G+1;
9. End While
10. Sonucu görüntüle ve bitir.

Şekil 3.3. DE'nin sözde kodu

3.4. İstatistik Testler

Son yıllarda, metasezgisel algoritmaların deneysel analizine yönelik ilgi ve ihtiyaç artmıştır. Bu deneysel karşılaştırmalar, farklı metodolojileri ve istatistik teknikleri kullanan çok sayıda makalenin bulunması nedeniyle dikkat çekicidir. Metasezgisel algoritmalar için parametrik testlerden daha az kısıtlayıcı olmaları ve küçük boyutlu sonuç örnekleri üzerinde kullanılabilmeleri sebebiyle, parametrik olmayan istatistiksel testlerin kullanılması önerilmektedir (García ve ark., 2009).

Bu tez çalışmasında, algoritmalardan elde edilen sonuçların karşılaştırılması ve yorumlanmasında Wilcoxon işaretli sıra testi ve Friedman testi kullanılmıştır (García ve ark., 2009). İkili karşılaştırmalar için kullanılan Wilcoxon işaretli sıra testi, eşleştirilmiş iki örneklemin arasındaki farkı kullanarak algoritmaları değerlendirir. Şöyleki, önce örneklemler arasındaki fark bulunur, ardından bu fark değerleri sıralanır. Fark değerlerinin işaretlerine bakılmaksızın sıra puanı atanır. Ardından, pozitif ve negatif sıraların toplamı hesaplanır. Buna bağlı olarak hesaplanan test istatistiği değerine göre karar verilir. Çoklu karşılaştırmalar için kullanılan Friedman testi ise iki veya daha fazla algoritma arasındaki farkları belirlemek ve bunları sıralamak için kullanılır (Derrac ve

ark., 2011). Bahsedilen testlerde, p-değeri istatistiksel bir hipotez testinin anlamlı olup olmadığı hakkında bilgi sağlar ve bu değer ne kadar küçükse, hipoteze karşı kanıt o kadar güçlü olur.

4. ÖNERİLEN HİBRİT ALGORİTMALAR

Bazı araştırmacılar, metasezgisel algoritmaların problem çözme başarısını artırmak için algoritmaların eksik yönlerini bir başka algoritmanın yeteneklerini kullanarak takviye etme yolunu seçmişlerdir. Literatürde bu mantıkla oluşturulmuş birçok hibrit algoritma bulunmaktadır. Bu yaklaşımdaki amaç, algoritmaların global arama, lokal arama, çeşitlilik, hızlı yakınsama gibi becerilerini birleştirerek daha iyi sonuçlar elde etmektir. Benzer ya da daha iyi sonuçların daha hızlı elde edilmesi de hibrit algoritma geliştirmek için yönlendirici olabilmektedir. Dolayısıyla, bir hibrit algoritmada en ideal olan, daha iyi sonuçların daha kısa sürede elde edilebilmesi olarak ifade edilebilir (Kıran, 2014).

Yarasa algoritmasında, yerel ve global arama dengesinin kurulmasında r ve A parametreleri belirleyicidir. Algoritmanın yapısı gereği iterasyon ilerledikçe r artar, A azalır. Algoritmada lokal aramayı gerçekleştiren ifade (Denklem 3.5), r parametresinin [0,1] aralığında rastgele üretilen bir sayıdan küçük olma şartına bağlı olarak yürütülür. Dolayısıyla r değerinin artışı, iterasyonun ilk adımlarında lokal aramanın daha yüksek olasılıkla yapılmasına neden olur. Bu durum, özellikle LSGO problemlerinde çözüm uzayının ayrıntılı şekilde aranmasını engeller, çeşitliliği azaltır ve lokal minimuma düşme ihtimalini artırır. Algoritmada yeni ve daha iyi çözümlerin popülasyona dahil edilmesi ise A parametresinin [0,1] aralığında rastgele üretilen bir sayıdan büyük olma şartına bağlı olarak gerçekleştirilir. İterasyonlar ilerledikçe A değerinin azalması, iterasyonun sonlarına doğru bulunan yeni ve daha iyi çözümlerin popülasyona dâhil edilmesi ihtimalini oldukça düşürür. Bu durum, bulunan daha iyi çözümlerin kaybedilmesine ve şayet lokal minimuma takılma durumu söz konusu ise algoritmanın lokal minimumdan çıkamamasına neden olur (Yılmaz ve ark., 2014).

Bu tez çalışmasında, yarasa algoritmasının bu yapısal sorunları sebebiyle yaşadığı performans kayıplarını azaltmak amacıyla iki yeni hibrit algoritma önerilmiştir. İlk hibrit algoritmada DE, ikinci hibrit algoritmada ise ABC algoritması BA ile hibrit olarak kullanılmıştır. DE ve ABC algoritmalarının literatürde sıklıkla kullanılması, ortaya koydukları başarılı sonuçlar, DE'nin mutasyon ve çaprazlama özelliği sayesinde sahip olduğu detaylı arama ve popülasyon çeşitliliğini koruma becerisi ve ABC'nin güçlü global arama yeteneği, bu algoritmaların tercih edilmesinde belirleyici olmuştur.

4.1. BA ve DE ile Oluşturulan Hibrit Algoritma (MBADE)

Yarasa algoritmasının yapısı incelendiğinde ve yarasa algoritması kullanılarak gerçekleştirilen çözümler analiz edildiğinde, yarasa algoritmasının hem lokal hem de global arama yeteneğinin geliştirilmesine ihtiyaç duyulduğu görülmektedir. BA algoritmasının yerel arama bölümünü ifade eden Denklem 3.5 incelendiğinde, popülasyondan seçilen bir çözüm etrafında arama yapıldığı görülür. Literatürde, bu arama işlemi için, çoğunlukla o anki çözüm ya da popülasyondaki global en iyi çözüm kullanılır. Yerel arama işleminde, yeni çözüm üretilirken yapılan hesaplama, bireyi oluşturan tüm boyutlarda gerçekleştirilmektedir. Bu durum, BA'da bireylerin birbirine hızla benzemesine ve çeşitliliğin kaybolmasına neden olur. DE algoritmasında yeni çözüm oluşturulurken tek bir çözüm etrafında arama yapmak yerine, popülasyondan seçilen üç veya daha fazla birey arasında belli stratejiler doğrultusunda mutasyon ve çaprazlama gibi işlemler uygulanarak yeni çözüm oluşturulur. Böylece, popülasyon çeşitliliği daha uzun süre korunur ve daha detaylı bir arama gerçekleştirilir. Bu tez çalışmasında önerilen, BA ve DE'nin birlikte kullanıldığı hibrit algoritmayı oluşturma fikri de bu sebepten dolayı ortaya çıkmıştır.

Önerilen hibrit algoritmanın geliştirilmesi sürecinde, öncelikle BA'nın performansını artırmaya yönelik bazı düzenlemeler yapılarak gelişmiş bir BA algoritması (MBA) elde edilmiştir. Ardından, algoritmanın performansını daha fazla artırmak ve DE'nin güçlü yönlerini BA'ya kazandırmak amacıyla DE ile hibrit çalışan bir algoritma (MBADE) oluşturulmuştur.

Standart BA algoritmasının, global arama bölümündeki konum güncelleme denklemi incelendiğinde, popülasyonda bulunan çözümlerin, hızlı bir biçimde sistemdeki en iyi çözüme yakınsama eğiliminde olduğu görülmektedir. Çünkü yeni hız değeri, popülasyondaki global en iyi çözüme bağlı olarak oluşturulur ve yeni konumu oluşturmak için yeni hız değeri ile yapılan toplama işlemi mevcut çözümün tüm boyutlarını etkiler. Bu durum, yarasa algoritmasının global arama yeteneğini zayıflatmaktadır. Bu nedenle, ilk olarak algoritma yeni bir konum belirleme stratejisi kullanılarak geliştirilmiştir. Bu yeni stratejiye ait denklemler,

k_1, k_2 : popülasyondan seçilmiş rastgele çözümleri,

x^* : global en iyi çözümü,

$j = 1,2, \dots, D$ olmak üzere rastgele seçilmiş bir boyutu,

α: ağırlık katsayısını ifade etmek üzere Denklem 4.1 ve 4.2'de verilmiştir.

$$\begin{cases} x_i^{t+1} = x_i^t + \alpha * (x^* - x_i^t) * f_1 + (x_i^t - x_{k1}^t) * f_2 & Eğer\ f(x_{k1}^t) < f(x_i^t) \\ x_{i,j}^{t+1} = x_{i,j}^t + (rand - 0.5) * 2 * \left(x_{i,j}^t - x_{k2,j}^t\right) & diğer \end{cases} \quad (4.1)$$

$$\begin{cases} f_1 = f_{min} + (f_{max} - f_{min}) * rand \\ f_2 = f_{min} + (f_{max} - f_{min}) * rand \end{cases} \quad (4.2)$$

Denklem 4.1'de yeni konum belirlemek için iki ayrı yaklaşım kullanılmıştır. İlk yaklaşım olarak, Chakri ve arkadaşları tarafından daha önce önerilmiş bir yöntem (Chakri ve ark., 2017) geliştirilerek kullanılmıştır. Bu yaklaşımda, yeni konum belirleme işleminde, standart BA'daki gibi sadece global en iyi çözümün değil, popülasyondan rastgele seçilen bir çözümünde etkisi bulunmaktadır. Yine BA'dan farklı olarak iki frekans değeri (f_1 ve f_2) bulunmaktadır (Denklem 4.2). Bu frekans değerleri, yeni konum belirleme işleminde, adım yönünün ve miktarının belirlenmesini sağlar. Önerilen hibrit algoritmada, bu yaklaşım kullanılırken denkleme bir ağırlık katsayısı (α) eklenerek geliştirilmiştir. Ağırlık katsayısı sayesinde, global en iyi çözümün yeni konum belirlemedeki etkisi bir miktar azaltılmıştır. Bu şekilde, global en iyi çözüme doğru yakınsama hızı yavaşlatılmış ve arama uzayının daha farklı bölgelerinin araştırılması sağlanmıştır. Denklem 4.1'deki ikinci yaklaşımda ise, ABC algoritmasının arama stratejisinden yararlanılmıştır. Mevcut çözüm ile rastgele seçilen bir komşu çözümün, yine rastgele seçilen bir boyutu arasındaki farktan yararlanılarak yeni çözüm belirlenmiştir. Bu yaklaşım her kullanıldığında, o anki çözümün sadece bir boyutunda değişiklik olmaktadır. Böylece ilgili çözüm etrafında daha ayrıntılı bir arama yapılmış olur. Denklemdeki bu yaklaşımlardan hangisinin kullanılacağına karar verilmesi ise yine Chakri ve arkadaşlarının önerdiği yönteme göre yapılmaktadır (Chakri ve ark., 2017).Yani rastgele seçilen çözümün uygunluğu, mevcut çözümden daha iyiyse ilk strateji, diğer durumlarda ikinci strateji uygulanır.

Denklem 4.2'de f_{min} ve f_{max} sırasıyla frekans değerinin alabileceği minimum ve maksimum değerlerini ifade eder. Önerilen algoritmada, $f_{min} = 0, \mathrm{f_{max}} = 2$ ve $\alpha = 0.7$ olarak alınmıştır. α değeri, Bölüm 5.1'de yapılan test işlemi sonucunda belirlenmiştir.

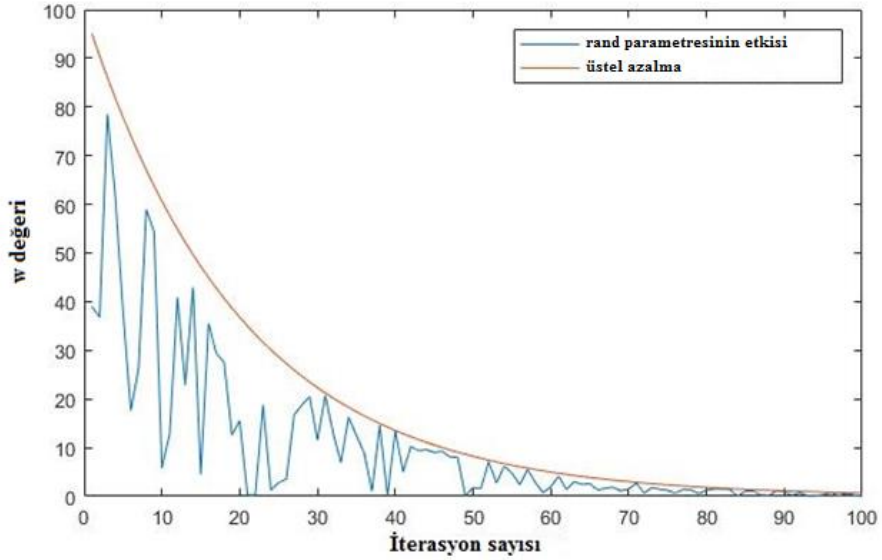
İkinci olarak, lokal arama stratejisinde düzenleme yapılmıştır. Yapılan düzenlemeye ait denklemler Denklem 4.3-4.5'de verilmiştir. Algoritmada kullanılan yeni konum güncelleme denkleminde (Denklem 4.3) yer alan, w_i^t parametresi iterasyon sayısına bağlı olarak Denklem 4.5'e göre azaltılarak kullanılmıştır. Denklem 4.4'te geçen ub ve lb sırasıyla fonksiyonun üst ve alt sınır değerini, t o anki iterasyon sayısı, t_{max} maksimum iterasyon sayısını ifade etmektedir.

$$x_i^{t+1} = x_i^t + \varepsilon * \overline{A^t} * w_i^t \tag{4.3}$$

$$w_0 = \frac{ub-lb}{4} \tag{4.4}$$

$$w_i^t = w_0 * rand * 2 * \exp(-5 * \frac{t}{t_{max}}) \tag{4.5}$$

w_i^t parametresinin değişim grafiği Şekil 4.1'de verildiği gibidir. Bu grafikte, üst ve alt sınır değeri sırasıyla 100 ve -100 olarak belirlenmiştir ve Denklem 4.5'e göre w_i^t değerleri hesaplanmıştır. Grafik incelendiğinde, rand parametresinin kullanımının adım büyüklüğüne çeşitlilik kazandırdığı görülmektedir.



Şekil 4.1. w_i^t parametresinin değişim grafiği

Önerilen hibrit algoritmada, Denklem 4.3'e göre yeni konumun belirlenmesi, başlangıçta büyük adımlarla, ilerleyen iterasyonlarda ise azalarak daha küçük adımlarla arama işlemi gerçekleştirilmesini sağlar. Bu yaklaşım, algoritmaya lokal arama sürecinin başlarında çözüm etrafında daha geniş bir alanda, süreç ilerledikçe çözüm etrafında daha dar bir alanda, daha ayrıntılı bir arama yapma becerisi kazandırır. Ayrıca, adım büyüklüğündeki çeşitlilik sayesinde lokal minimuma takılma sorunu azaltılır.

Son olarak, algoritmadaki r ve A parametreleri Chakri ve arkadaşlarının önerdiği yöntemdeki (Chakri ve ark., 2017) gibi düzenlenip, Denklem 4.6 ve 4.7'de verildiği gibi kullanılmıştır. Standart algoritmada, r parametresi Denklem 3.7'ye göre artırılarak, A parametresi ise Denklem 3.6'ya göre azaltılarak kullanılır. Standart algoritmada, iteratif süreç içerisinde bu parametreler hızlı şekilde son değerlerine ulaşır. Sonrasında, yüksek r değeri sebebiyle global aramadan lokal aramaya geçiş ve yine düşük A değeri sebebiyle de yeni çözümlerin popülasyona dahil edilmesi ihtimali düşer. Bu sebeple, Denklem 4.6 ve 4.7 kullanılarak, bu parametrelerin artışı ve azalışı daha monoton hale getirilmiştir. Bu çalışmada, $r_{ilk} = 0.1,\ r_{son} = 0.7,\ A_{ilk} = 0.9,\ A_{son} = 0.6$ olarak belirlenmiştir.

$$r_i^t = \frac{r_{ilk} - r_{son}}{1 - t_{max}} * (t - t_{max}) + r_{son} \tag{4.6}$$

$$A_i^t = \frac{A_{ilk} - A_{son}}{1 - t_{max}} * (t - t_{max}) + A_{son} \tag{4.7}$$

BA algoritmasının lokal arama bölümü, $rand > r_i^t$ koşuluna bağlı olarak çalışır. r için belirlenen bu değer aralığı sayesinde, algoritma iterasyonun başlangıcında arama uzayını daha ayrıntılı araştırabilmek için lokal rastgele adımlarla global aramayı destekleme eğilimindedir. Bu mekanizma, r'nin başlangıç değerinin küçük verilmesi ile elde edilmiştir. Benzer şekilde, A parametresi için belirlenen değer aralığı sayesinde, arama süreci boyunca yeni çözümlerin popülasyona dâhil edilme ihtimali korunmuştur. Algoritmada, yeni ve daha iyi bir çözümün popülasyona dâhil edilmesi, $rand < A_i^t$ şartının sağlanmasına bağlıdır. A parametresinin alabileceği en küçük değerin 0.6 olarak ayarlanması, bu ihtimalin arama sürecinin sonlarında da korunmasını sağlayacaktır. Böylece, algoritmanın lokal minimum tuzaklara takılmasının ve erken yakınsamasının önüne geçilecektir.

Yukarıdaki bahsedilen düzenlemeler sonrası elde edilen gelişmiş BA algoritmasının performansı incelendiğinde, standart algoritmaya göre performansının ciddi şekilde arttığı görülse de, hala geliştirilmeye ihtiyaç duyduğu tespit edilmiştir. Gelişmiş BA algoritmasında, kullanılan lokal arama formülü sebebiyle algoritmanın ürettiği en iyi çözüme hızlı yakınsama davranışı devam etmektedir. Popülasyondaki bir çözüme hızlı yakınsama, çalışılan problemin türüne ve boyutuna göre kimi zaman avantaj, kimi zaman dezavantaj olabilir. Bu açıdan hızlı yakınsama karakteristiğini de kaybetmeden algoritmanın performansını geliştirme ihtiyacı ortaya çıkmıştır. BA algoritmasının lokal arama yeteneğini daha fazla desteklemek ve çeşitlendirmek amacıyla, DE algoritmasıyla birlikte kullanıldığı hibrit MBADE algoritması önerilmiştir. DE algoritması, rastgele seçilen bireyler arasında bilgi değişimini farklı stratejilere göre yapan lokal arama yeteneği güçlü bir algoritma olması sebebi ile tercih edilmiştir. DE algoritmasında yeni çözüm üretilirken, BA'dan farklı olarak tek bir çözüm kullanmak yerine popülasyondan seçilen en az üç çözüm kullanılır. Bu yapısı, belli bir çözüme doğru hızlı yakınsamanın azaltılmasına ve popülasyon çeşitliliğinin artmasına neden olur. Önerilen hibrit algoritmada, popülasyondaki bir bireye hangi algoritmanın uygulanacağı performans odaklı bir olasılık değerine göre karar verilir. Algoritmalara ait olasılık, ürettiği uygunluk değerinin başarı durumuna göre güncellenir ve zamanla başarılı olan algoritmanın seçilme olasılığı artar. Aynı iterasyonda, popülasyondaki bireylere farklı algoritmalar uygulanabilir, bu da farklı lokal arama stratejilerinin bir arada kullanılabilmesine imkan verir. Böylece, popülasyon çeşitliliği daha uzun süre korunur ve yakınsama hızı probleme ve boyuta göre adapte edilir.

Bu tez çalışması kapsamında geliştirilen hibrit algoritmada kullanılan DE algoritmasında mutasyon seçme stratejisi olarak Self Adaptive DE (SaDE) (Qin ve Suganthan, 2005) versiyonundaki mantık benimsenmiştir. Buna göre iki mutasyon stratejisi (DE/rand/1 ve DE/current-to-best/1) seçilir ve 50 iterasyon boyunca devam eden bir öğrenme periyodundan sonra, bireye hangi mutasyon stratejisinin uygulanacağına, stratejilerin önceki performansına göre hesaplanan bir olasılık değerine (P_m) göre karar verilir. Öğrenme periyodunun amacı, mutasyon stratejilerinin performansa katkısını belirlemek ve her stratejinin çözüme katkısı ile orantılı olacak şekilde bir olasılık değerine sahip olmasını sağlamaktır.

Öğrenme periyodu sonrası, bireye uygulanacak stratejiye bir olasılık değeri (P_m) ile karar verilir. Yani, bir stratejinin olasılık değeri arttıkça seçilme ihtimali de artar. Bu olasılık değeri Denklem 4.8'e göre hesaplanmıştır. Bu denklemde, ms1 ve ms2, bir sonraki nesile başarıyla aktarılan aday çözümlerin sayısı iken; mf1 ve mf2, bir sonraki nesile aktarılmayan aday çözümlerin sayısıdır. ms1 / mf1 ve ms2 / mf2 çiftleri sırasıyla DE/rand/1 ve DE/current-to-best/1 stratejileri tarafından üretilen çözüm sayılarıdır.

Önerilen algoritmada, DE'nin önemli parametreleri olan F ve C_r parametrelerinin değeri sırasıyla 0.8 ve 0.2 olarak seçilmiştir.

$$P_m = \frac{ms1*(ms2+mf2)}{ms1*(ms2+mf2)+ms2*(ms1+mf1)} \tag{4.8}$$

Geliştirilen hibrit algoritmada, arama stratejileri (yani MBA ve DE) tek bir popülasyonu ortak kullanır. Bu hibrit yapıda, her bir birey için uygulanacak arama stratejisinin seçimi, stratejinin önceki iterasyonlardaki performansı değerlendirilerek elde edilen bir olasılık değerine göre yapılır. Yani, bir arama stratejisi önceki aramada daha iyi performans gösterirse, gelecek nesil için aday çözümleri üretmede daha fazla şansa sahip olacaktır. Her aday çözüm üretiminde MBA'nın seçilme olasılığı (P_{MBA}), Denklem 4.9'a göre hesaplanır. Bu durumda, DE'nin seçilme olasılığı $(1 - P_{MBA})$ olur. Seçilen yöntem, bireye uygulanarak aday birey oluşturulur. Denklem 4.9'da, Ns_{MBA} ve Ns_{DE} sırasıyla MBA ve DE tarafından yeni nesile başarıyla aktarılan aday çözüm sayısını gösterir. Nf_{MBA} ve Nf_{DE} ise sırasıyla MBA ve DE tarafından üretilen ancak yeni nesile aktarılamayıp atılan aday çözüm sayısıdır.

$$P_{MBA} = \frac{Ns_{MBA}*(Ns_{DE}+Nf_{DE})}{Ns_{MBA}*(Ns_{DE}+Nf_{DE})+Ns_{DE}*(Ns_{MBA}+Nf_{MBA})} \tag{4.9}$$

Ayrıca, hibrit algoritmada önceki deneyimlerin etkisini sıfırlamak için belli iterasyon sayılarına ulaşıldığında, olasılık hesabında kullanılan parametrelerin değerleri sıfırlanmıştır. Çünkü, lokal minimuma takılma durumunda bazen arama stratejisini değiştirmek bu çözümden daha farklı bir bölgede arama yapmayı sağlayabilir ve lokal minimumdan çıkma olasılığını artırabilir. Aynı sıfırlama işlemi, uygulanan mutasyon stratejisi parametreleri içinde yapılmıştır.

MBADE algoritmasının sözde kodu Şekil 4.2'de verilmiştir. Sözde kodda geçen *Ogrenme* parametresi, [0,1] aralığında rastgele değerler içeren bir dizidir. *m_strateji* ise hangi mutasyon stratejisinin (DE/rand/1 yada DE/current-to-best/1) uygulandığının bilgisini tutan bir değişkendir. o*gr_sayac* ise öğrenme periyodunda kullanılan bir sayaçtır.

1. Başlangıç popülasyonunu oluştur. ($x_i, i = 1,2, \dots, N$)
2. DE ve MBA için parametreleri tanımla ve başlangıç değerlerini ata.
3. Popülasyondaki en iyi çözümü (x^*) bul.
4. While (G < maksimum çevrim sayısı)
5. Arama stratejisinin seçiminde kullanılan sayaçlara başlangıç değeri ata ($Ns_{MBA} = 1, Ns_{DE} = 1, Nf_{MBA} = 1, Nf_{DE} = 1$).
6. Mutasyon stratejisinin seçiminde kullanılan sayaçlara başlangıç değeri ata (ms1=1,ms2=1,mf1=1,mf2=1).
7. Öğrenme sürecinde kullanılacak bir sayaç tanımla (ogr_sayac =1)
8. Öğrenme sürecinde kullanılmak üzere bir dizi tanımla.(Ogrenme [1,50])
9. r_{ilk} , r_{son}, A_{ilk} ve A_{son} için başlangıç değerlerini ata.
10. For t=1: t_{max}
11. Denklem 4.9'a göre MBA'nın seçilme olasılığını (P_{MBA}) hesapla.
12. For i=1: N
13. *If* ($rand < P_{MBA}$) *// MBA algoritması seçilirse//*
14. Rastgele iki çözüm seç (k_1, k_2)
15. f_1 ve f_2'yi Denklem 4.2'ye göre hesapla.
16. Denklem 4.1'e göre bir aday çözüm (x_i^{t+1}) üret.
17. If (t==1)
18. r_i^t ve A_i^t değerlerini sırasıyla Denklem 4.6 ve 4.7'ye göre hesapla.
19. End If
20. If ($rand > r_i^t$)
21. Denklem 4.5'e göre w_i^t değerini hesapla.
22. Denklem 4.3'e göre bir aday çözüm (x_i^{t+1}) üret.
23. End If
24. If ($rand < A_i^t$) ve ($f(x_i^{t+1}) < f(x_i^t)$)
25. Aday çözümü (x_i^{t+1}) kabul et.
26. Denklem 4.6 ve 4.7'ye göre r_i^t artır, A_i^t azalt.
27. $Ns_{MBA} = Ns_{MBA} + 1$
28. If ($f(x_i^{t+1}) < f(x^*)$)
29. x^* güncelle.
30. End If
31. Else
32. $Nf_{MBA} = Nf_{MBA} + 1$
33. End If
34. *Else* *// DE algoritması seçilmişse//*
35. Popülasyondan rastgele üç birey seç (r_1, r_2, r_3)
36. If (ogr_sayac <= 50) *//Öğrenme periyodu //*
37. If (Ogrenme(1,ogr_sayac) < = 0.5)
38. Denklem 3.14'e göre bir mutant vektör ($V_{i,t}$) oluştur.
39. m_strateji=1;
40. Else
41. Denklem 3.18'e göre bir mutant vektör ($V_{i,t}$) oluştur.
42. m_strateji =2;
43. End If

Şekil 4.2.a. MBADE sözde kodu

```
                    ogr_sayac=ogr_sayac+1;
                Else
                  Mutasyon stratejisi seçimi için Denklem 4.8'e göre olasılık (P_m) hesapla.
                  If (rand < P_m)
                    Denklem 3.14'e göre bir mutant vektör (V_i,t) oluştur.
                    m_strateji=1;
                  Else
                    Denklem 3.18'e göre bir mutant vektör (V_i,t) oluştur.
                    m_strateji=2
                  End If
                End If
                Denklem 3.21'e göre bir aday vektör (U_i,t) oluştur.
                If (f(U_i,t) < f(x_i^t))
                 Yeni çözümü U_i,t kabul et.
                    If (f(U_i,t) < f(x*))
                      x* güncelle
                    End If
                    If ( m_strateji==1)
                       ms1=ms1+1;
                    Else
                       ms2=ms2+1;
                    End If
                    Ns_DE = Ns_DE + 1
                  Else
                    If ( m_strateji==1)
                       mf1=mf1+1;
                    Else
                       mf2=mf2+1;
                    End If
                   Nf_DE = Nf_DE + 1
                  End If
              End If
          End For
          If (mod(t ,t_reset) == 0)
            Ns_MBA = 1, Ns_DE = 1, Nf_MBA = 1, Nf_DE = 1
            Ogr_sayac=1;
            ms1=1,ms2=1, mf1=1,mf2=1;
          End If
        End For
      G=G+1;
    End While
  Sonuçları görüntüle.
```

Şekil 4.2.b. MBADE sözde kodu

4.2. BA ve ABC ile Oluşturulan Hibrit Algoritma (BA_ABC)

BA'nın yapısal olarak arama sürecinin başlangıcında lokal arama yapma eğiliminde olması, aramada kullanılan denklem nedeniyle popülasyondaki birey çeşitliliğinin azalması, arama uzayının ayrıntılı aranamamasına ve lokal minimum tuzaklarına hızlıca düşmesine neden olmaktadır. Özellikle, arama uzayının geniş olduğu büyük boyutlu problemlerde, bu sorunlar daha belirgin hale gelmekte ve ciddi performans düşüşlerine neden olmaktadır. BA'nın global arama yeteneğinin

geliştirilmesi konusundaki araştırmalar, global arama konusunda daha başarılı olan bir algoritmadan faydalanma fikrini ortaya çıkarmıştır. Bu doğrultuda, tez çalışması kapsamında, literatürde global arama becerisinin yüksek olmasıyla bilinen ABC algoritmasıyla (Karaboga ve Basturk, 2008 ; Abu-Mouti ve El-Hawary, 2012 ; Babayigit ve Ozdemir, 2012) BA'nın beraber kullanıldığı bir hibrit algoritma önerilmiştir.

BA'da global arama işlemi sırasında (Denklem 3.2, 3.3 ve 3.4) yeni çözüm oluşturulurken mevcut çözümün tüm boyutlarında değişiklik olur. Global aramada kullanılan bu denklemler nedeniyle, iterasyon ilerledikçe popülasyondaki bireyler (çözümler) hızla en iyi bireye ve birbirlerine benzemektedir. Dolayısıyla, popülasyonun çeşitliliği de hızla azalmaktadır. ABC'de ise yeni bir çözümün oluşturulması aşamasında, sadece rastgele seçilen bir boyutta değişiklik olmaktadır (Denklem 3.9). Bu durum, en iyi çözüme doğru hızlı yakınsamayı engeller. Ayrıca, ABC algoritmasında belirli bir limit değeri boyunca çözüm geliştirilemezse, o çözüm unutulur ve yerine rastgele bir çözüm atanır. Bu yaklaşım, gerektiğinde arama uzayının rastgele noktalarından yeniden aramaya başlamak anlamına gelir. Böylece daha iyi bir global arama ve popülasyon çeşitliliği sağlanmış olur. Bu olumlu yönlerinin yanı sıra, ABC'nin bu arama karakteristiği arama uzayının genişliği arttıkça bir dezavantaj haline gelebilir. Çünkü büyük boyutlu bir arama uzayında, sadece bir boyutta değişiklik yaparak yeni çözüm üretmek, yavaş bir arama yapmaya ve buna bağlı olarak arama uzayının bir kısmının hiç araştırılamamasına neden olabilir. Bu yüzden, ABC'nin global arama becerisinden faydalanırken, BA'nın hızlı yakınsama karakteristiğini de kaybetmeyen bir hibrit algoritmaya ihtiyaç vardır.

Yukarıda bahsedilen amaçlar doğrultusunda geliştirilen hibrit BA_ABC algoritmasında, öncelikle BA'nın global arama yeteneğini geliştirecek bir düzenleme yapılmıştır. Buna göre BA'nın hız formülüne bir atalet ağırlığı (w) eklenmiştir. Literatürde daha önce de BA'da kullanılmış (Yilmaz ve Kucuksille, 2013 ; Yılmaz ve Küçüksille, 2015 ; Arora ve Lodhi, 2016) olan atalet ağırlığı katsayısı sayesinde, algoritmanın arama yeteneği geliştirilmiştir. Buna göre, hız formülüne Denklem 4.10'da görüldüğü gibi atalet ağırlığı katsayısı eklenmiş ve [0.9, 0.4] aralığında kaotik (Feng ve ark., 2007) olarak azaltılarak kullanılmıştır. Denklem 4.11, kaotik azaltma denklemini göstermektedir. Bu denklemde z, [0,1] aralığında rastgele bir sayıyı, w_{max} atalet ağırlığı

için belirlenen üst sınır değerini, w_{min} atalet ağırlığı için belirlenen alt sınır değerini, t o anki iterasyon sayısını, t_{max} ise algoritmanın yürütüleceği maksimum iterasyon sayısını ifade eder.

$$v_i^t = w * v_i^{t-1} + (x_i^t - x^*) * f_i \tag{4.10}$$

$$z = rand(), \quad z = 4 * z * (1 - z)$$
$$w = (w_{max} - w_{min}) * \frac{(t_{max} - t)}{t_{max}} + w_{min} * z \tag{4.11}$$

Yapılan bu düzenleme sayesinde önceki hız değerinin, yeni hıza etkisi başlangıçta daha fazla olmuştur ve iterasyon ilerledikçe önceki hızın etkisi azalmıştır. Başka bir deyişle, başlangıçta algoritmanın ilerleme hızı (adım miktarı) daha fazladır. İterasyon ilerledikçe azalır. Bu durum başlangıçta global aramanın, daha sonra lokal aramanın önem kazanacağı anlamına gelir.

BA'da yapılan bu düzenleme sonrası, algoritmanın global arama yeteneğini ve çeşitliliğini artırmak amacıyla, bu gelişmiş BA algoritmasıyla standart ABC algoritmalarının birlikte kullanıldığı bir hibrit algoritma oluşturulmuştur. Bu hibrit algoritma sayesinde, BA'nın hala devam eden global en iyi çözüme hızlı yakınsama ve zamanla popülasyon çeşitliliğini kaybetme problemlerinin giderilmesi amaçlanmıştır. Bu doğrultuda, BA ve ABC'nin paralel yürütüldüğü hibrit bir algoritma (BA_ABC) geliştirilmiştir. Hibrit algoritmada popülasyon önce ikiye bölünür. Popülasyonun ilk yarısındaki bireylere BA algoritması, ikinci yarısındaki bireylere ise ABC algoritması uygulanarak yeni çözümler üretilir. En iyi çözüm değeri (x^*), algoritmalar tarafından daha iyi çözümler bulundukça güncellenir. Her belli sayıda iterasyon (bdk kadar) tamamlandığında, algoritmaların performansları incelenir. Yani BA ve ABC için tanımlanmış ve global en iyi çözümden, daha iyi çözüm buldukları zaman artırılan sayaçlara (ba_yb ya da abc_yb) bakılır. Geçen süreçte, hangi algoritma daha fazla yeni çözüm üretmişse (yani hangi sayaç daha büyükse) o algoritmaya ait popülasyondaki uygunluk değeri en iyi olan bireylerin bir kısmı (dm sayısı kadar) diğer algoritmanın aynı sayıda ve uygunluk değeri en kötü bireylerinin yerine yazılır. Böylece popülasyonlar arası bilgi değişimi sağlanır. Her bilgi değişimi sonrasında, daha fazla sayıda yeni çözüm üretmiş olan başarılı algoritmaya ait sayaç (ba_bs ya da abc_bs)

artırılır. Bu sayaçlardan herhangi biri maksimum değişim sayısına (mds) ulaştığında ise sayaca ait algoritmanın o problem için daha başarılı olduğu kabul edilir ve kalan iterasyonlara tüm popülasyon üzerinde bu algoritma yürütülerek devam edilir.

Geliştirilen algoritmaya ait sözde kod Şekil 4.3'de verilmiştir. BA_ABC'nin sözde kodunda kullanılan bazı parametreler aşağıdaki gibi açıklanabilir.

x^*: Popülasyondaki uygunluk değeri en iyi olan bireydir.
BA_yeni_birey (ba_yb) ve *ABC_yeni_birey (abc_yb):* Sırasıyla BA ve ABC'nin kaç adet yeni çözüm ürettiğini tutan sayaçlardır.
Başarı durumu kontrol (bdk): Önceden belirlenmiş bir sayısıdır. Her *bdk* değeri kadar iterasyon tamamlandığında algoritmaların performansı incelenir. Yani BA ve ABC'nin ürettiği yeni çözüm sayısına bakılır. Buna göre bilgi değişiminin yönü belirlenir.
BA_başarı_sayısı (ba_bs) ve ABC_başarı_sayısı (abc_bs): Sırasıyla BA ve ABC'nin bilgi değişiminde kaç defa başarı olan algoritma olduğunun sayısını tutan sayaçtır.
Değişim miktarı (dm): bdk değeri kadar iterasyon tamamlandığında, başarısı yüksek olan algoritmanın popülasyonundaki kaç adet iyi uygunluk değerli bireyin, diğer algoritmanın popülasyonundaki uygunluk değeri kötü olan bireylerinin yerine yazılacağını belirleyen sayıdır. Algoritmada bu sayı, popülasyondaki birey sayısının %10'u olarak belirlenmiştir. Bu sayının büyük olması popülasyon bireylerinin hızlı biçimde birbirine benzemesine ve çeşitliliğin azalmasına neden olabileceği için küçük bir değer seçilmiştir.
Maksimum değişim sayısı (mds*)*: Popülasyonlar arasında belli bir yönde yapılabilecek bilgi değişiminin maksimum sayısıdır. Bu sayı çalışmada Denklem 4.12'ye göre hesaplanmıştır. Buna göre toplam değişim sayısının %60'ında bir algoritma başarılı olmuşsa kalan iterasyonlara tüm popülasyon üzerinde bu algoritma ile devam edilir. (Denklemdeki 0.6 çarpanı, Bölüm 5.1'de sonuçları verilen test işlemi sonrasında belirlenmiştir.)

$$mds = \frac{t_{max}}{bdk} * 0.6 \tag{4.12}$$

1. Hedef fonksiyonu $f(x)$ belirle.
2. N adet D boyutlu birey içeren başlangıç popülasyonunu oluştur. $x = (x_1, x_2, \dots, x_N)^D$
3. BA ve ABC algoritmalarına ait parametreleri tanımla ve başlangıç durumuna getir.
4. Popülasyonun en iyi uygunluk değerini (x^*) bul.
5. bdk değerini belirle.
6. Denklem 4.12'ye göre mds değerini belirle.
7. While (G < genel çevrim sayısı)
8. $ba\_yb = 0\ ,\ abc\_yb = 0, ba\_bs = 0, abc\_bs = 0$
9. For t=1: maksimum iterasyon sayısı
10. If ($ba\_bs\ <\ mds$ ve $abc\_bs\ <\ mds$)
11. For i=1:N/2
12. BA'ya göre yeni çözüm (x_{yeni}) üret.
13. If $(f(x_{yeni}) < f(x^*))$
14. Yeni çözümü kabul et.
15. x^*güncelle.
16. $ba\_yb = ba\_yb + 1;$
17. End if
18. End For
19. For i=(N/2+1):N
20. ABC'ye göre yeni çözüm (x_{yeni}) üret.
21. If $(f(x_{yeni}) < f(x^*))$
22. Yeni çözümü kabul et.
23. x^* güncelle.
24. $abc\_yb = abc\_yb + 1;$
25. End if
26. End For
27. If (mod (t, bdk) = = 0)
28. If $(ba\_yb >= abc\_yb)$
29. BA ve ABC populasyonlarındaki bireyleri uygunluk değerine göre sırala
30. BA'nın uygunluk değeri en iyi olan dm sayısı kadar bireyini, ABC popülasyonundaki aynı sayıda uygunluğu en kötü olan bireylerinin yerine yaz.
31. $ba\_bs = ba\_bs + 1;$
32. Else
33. BA ve ABC populasyonlarındaki bireyleri uygunluk değerine göre sırala
34. ABC'nin uygunluk değeri en iyi olan dm sayısı kadar bireyini, BA popülasyonundaki aynı sayıda uygunluğu en kötü olan bireylerinin yerine yaz.
35. $abc\_bs = abc\_bs + 1;$
36. End if
37. $abc\_yb = 0;$
38. $ba\_yb = 0;$
39. End if
40. Else If ($ba\_bs\ >= mds$)
41. Kalan iterasyon sayısını hesapla
42. Kalan iterasyonlara BA ile devam et.
43. Döngüden çık
44. Else
45. Kalan iterasyon sayısını hesapla
46. Kalan iterasyonlara ABC ile devam et.
47. Döngüden çık
48. End if
49. End For
50. End While

Şekil 4.3. BA_ABC sözde kodu

5. DENEYSEL SONUÇLAR VE TARTIŞMA

5.1. Hibrit Algoritmalarda Kullanılan Önemli Parametrelerin Değerlerinin Belirlenmesi

Metasezgisel algoritmalarda parametre değerlerinin belirlenmesi, algoritma performansını önemli ölçüde etkiler. Problemin yapısal özelliklerine ve boyutuna göre en uygun parametre değeri değişebilir. Bazen bir problem için iyi çözümler üreten parametre değerleri, problemin boyutunda ya da yapısındaki değişiklikler sonrası aynı performansı göstermeyebilir. Dolayısıyla, en uygun parametre değerini kesin olarak belirlemek pek mümkün değildir ve belirsizlikler içerir. Bu sebeple, araştırmacılar genellikle benzer problemler için kullanılan parametre değerlerini seçerler ya da testler yaparak en uygun parametre değerini bulmaya çalışırlar.

Bu tez çalışması kapsamında, önerilen algoritmalar için parametre değerleri, çoğunlukla literatürdeki benzer problemlerde sıklıkla kullanılan parametre değerlerine bakılarak seçilmiştir. Yapılan test işlemlerinde, popülasyondaki birey sayısı 10 ile 100 aralığında seçilmiştir. Fonksiyon değerlendirme sayısı ve genel çevrim sayısı ise çalışılan fonksiyon kümesinin kurallarına göre seçilmiştir ve ilgili bölümlerde belirtilmiştir.

Standart BA algoritması ile yapılan test işlemlerinde ise $f_{min} = 0$, $f_{max} = 1$, $r = 0.5$, $A_{min} = 0$, $A_0 = 0.9$ ve $\alpha = \gamma = 0.9$ olarak alınmıştır. Bunlara ek olarak, performansı önemli ölçüde etkileyeceği düşünülen bazı parametre değerleri ise, bu bölümde yapılan test işlemleri sonucunda tespit edilmiştir. Bu bölümdeki test işlemlerinde, Çizelge 5.1'de verilen on adet klasik kıyaslama fonksiyonu kullanılmıştır. Bu on adet fonksiyonun üç tanesi (F6,F7,F10) tek modlu(unimodal), kalan yedi fonksiyon ise çok modlu (multimodal) özelliktedir. Çizelgede arama aralığı ve fonksiyon için minimum değerlere de yer verilmiştir. * sembolü ise ilgili fonksiyonun minimum değerinin boyuta bağlı olarak değiştiğini göstermektedir. Yapılan test işleminde genel çevrim sayısı 10, D boyut olmak üzere FEs değeri 10000*D olarak belirlenmiştir. D, küçük boyut için 10, büyük boyut için 1000 olarak seçilmiştir.

Çizelge 5.1. Klasik kıyaslama fonksiyonları

Fonksiyon		Açıklama	Aralık /Özellik	Minimum değer
F1	Griewangk	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600] / Ç	0
F2	Rastrigin	$f(x) = 10 * n + \sum_{i=1}^{n} (x_i^2 - 10 * \cos(2\pi x_i))$	[-15,15] / Ç	0
F3	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} (100\,(\,x_{i+1} - x_i^2\,)^2 + (\,x_i - 1\,)^2)$	[-15,15] / Ç	0
F4	Ackley	$f(x) = \sum_{i=1}^{n-1} (20 + e^{-20} e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))})$	[−32.768, 32.768] / Ç	0
F5	Schwefel	$f(x) = 418.9829\,n - \sum_{i=1}^{n} \left[-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	[-500,500] / Ç	0
F6	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[-600,600] / T	0
F7	Easom	$f(x) = -(-1^n)\left(\prod_{i=1}^{n} cos^2(x_i)\right) exp\left[-\sum_{i=1}^{n} (x_i - \pi)^2\right]$	$[-2\pi, 2\pi]$ / T	-1
F8	Michalewicz	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left[sin\left(\frac{ix_i^2}{\pi}\right)\right]^{20}$	$[0, \pi]$ / Ç	*
F9	Xin-She Yang	$f(x) = \left(\sum_{i=1}^{n} \lvert x_i \rvert\right) exp\left[-\sum_{i=1}^{n} \sin(x_i^2)\right]$	$[-2\pi, 2\pi]$ / Ç	0
F10	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} ix_i\right)^4$	[-5,10] / T	0

Bu bölüm ve sonrasında verilen tüm çizelge ve grafiklerde, tez çalışması kapsamında önerilen ilk algoritma (MBADE) Hibrit Algoritma 1 ifadesinin kısaltılmasıyla HBA1, ikinci algoritma (BA_ABC) ise Hibrit Algoritma 2 ifadesinin kısaltılmasıyla HBA2 ismiyle gösterilmiştir. Ayrıca tez boyunca ölçek ve boyut ifadeleri eş anlamlı olarak kullanılmıştır.

5.1.1. HBA1 algoritmasında kullanılan ağırlık katsayısı (α) değerinin belirlenmesi

HBA1 algoritmasında konum güncelleme formülünde iki ayrı strateji bulunmaktadır (Denklem 4.1). Bunların ilkinde kullanılan α katsayısı, en iyi çözümün yeni konum belirlemedeki etkisi azaltmak ve en iyi çözüme doğru olan yakınsamayı yavaşlatmak amacıyla kullanılmıştır. α katsayısının değeri, Çizelge 5.1'de verilen fonksiyonlar üzerinde test işlemi yapılarak belirlenmiştir. Test işlemi sonucunda elde edilen ortalama değerler Çizelge 5.2'de gösterilmiştir. Çizelge 5.2'de satırlar fonksiyonları, sütunlar ise farklı α değerlerini göstermektedir. Anlaşıldığı üzere, her fonksiyon [0.1,1] aralığında 0.1 artışla elde edilen toplam 10 farklı α değeri için yürütülmüş ve sonuçlar çizelgede listelenmiştir. Ayrıca çizelgede, her fonksiyon için elde edilen en küçük ortalama değer koyu fontta gösterilmiştir.

Sonuçlar incelendiğinde, $\alpha = 0.7$ değeri için toplam üç fonksiyonda en iyi (minimizasyon problemi olduğu için en küçük) ortalamanın elde edildiği görülmektedir. Sonuçlar Friedman testine göre sıralandığında ise 2.75 olan en küçük ortalama sıra değeri ile 0.7 değeri birinci sırada yer almıştır. Bu sonuç doğrultusunda, HBA1 algoritmasındaki α değeri 0.7 olarak belirlenmiştir.

Çizelge 5.2. α değerinin belirlenmesi için yapılan testin sonuçları

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
F1	8.1748E-02	8.5735E-02	7.5228E-02	7.8368E-02	6.4123E-02	5.8732E-02	**5.8528E-02**	5.9266E-02	7.1664E-02	8.5751E-02
F2	8.2095E-01	6.0404E-01	7.4386E-01	7.1639E-01	6.7183E-01	**1.9979E-02**	1.4732E-01	2.9805E-01	3.7307E-01	5.4332E-01
F3	3.0994E+00	1.4818E+00	8.1489E-01	1.4612E-01	1.9210E-02	4.0549E-01	**1.5528E-05**	3.5015E-05	4.8554E-05	8.8783E-03
F4	1.3997E+02	1.3999E+02	1.4004E+02	1.4000E+02	**1.3998E+02**	1.3999E+02	1.3999E+02	1.4002E+02	1.4003E+02	1.4004E+02
F5	2.2860E+02	1.9522E+02	9.2710E+01	5.2537E+01	1.4174E+01	2.0375E+01	7.2435E+00	**2.8889E+00**	3.8390E+00	3.7629E+00
F6	9.5012E-04	3.6618E-05	1.1694E-06	1.0189E-08	9.2173E-11	**4.0788E-12**	4.7543E-12	1.0243E-11	9.2959E-11	1.0914E-09
F7	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**
F8	-9.6387E+00	-9.6380E+00	-9.6362E+00	-9.6336E+00	-9.6398E+00	**-9.6496E+00**	-9.6445E+00	-9.6409E+00	-9.6442E+00	-9.6470E+00
F9	6.5594E-04	6.2999E-04	6.2934E-04	6.3708E-04	**6.1504E-04**	6.2678E-04	6.1975E-04	6.1813E-04	6.4158E-04	6.2659E-04
F10	4.6042E-02	2.7176E-04	7.0428E-07	2.3228E-09	2.3133E-12	1.0193E-14	4.7887E-15	**3.6838E-15**	3.2988E-14	2.8456E-12
Friedman	8.15	7.55	7.70	7.15	4.45	3.55	**2.75**	3.25	5.05	5.40
Sıra	10	8	9	7	4	3	**1**	2	5	6

5.1.2. HBA1 algoritmasında kullanılan ogr_sayac değerinin belirlenmesi

HBA1 algoritması içerisinde kullanılan DE algoritmasında, mutasyon seçme stratejisi olarak Self Adaptive DE (SaDE) versiyonundaki mantık benimsenmiştir. Buna göre, seçilen mutasyon stratejilerinden hangisinin bireye uygulanacağına bir öğrenme periyodu sonrasında karar verilir. Öğrenme periyodu, orijinal çalışmada 50 olarak belirlenmiştir. HBA1 algoritmasında öğrenme periyodunun kaç iterasyon süreceğini belirleyen sayacın (ogr_sayac) değerine, yapılan test işlemi sonrasında karar verilmiştir. Buna göre, Çizelge 5.1' de verilen kıyaslama fonksiyonları üzerinde [50, 100, 150, 200, 250] gibi beş farklı sayaç değeri belirlenerek algoritma yürütülmüş ve elde edilen ortalama değerler karşılaştırılmıştır. Bu test işlemine ait sonuçlar Çizelge 5.3'te verilmiştir. Sonuçlar incelendiğinde, ogr_sayac değeri 50 alındığında toplam üç fonksiyonda en iyi ortalama değer bulunmuştur ve Friedman testine göre yapılan sıralamada bu değer 2.50 ortalama sıra değeri ile birinci sırada yer almıştır. Bu sonuç doğrultusunda, HBA1 algoritmasında ogr_sayac değeri 50 olarak kullanılmıştır.

Çizelge 5.3. ogr_sayac değerinin belirlenmesi için yapılan testin sonuçları

	50	**100**	**150**	**200**	**250**
F1	**2.9900E-02**	3.1900E-02	5.0400E-02	6.6600E-02	9.5800E-02
F2	9.8800E-02	2.3070E-01	9.1500E-02	2.8450E-01	**3.4600E-02**
F3	2.2698E-05	**5.9532E-06**	6.5388E-06	5.4801E-04	1.4297E-05
F4	1.4013E+02	**1.3988E+02**	1.3992E+02	1.4001E+02	1.3989E+02
F5	8.0477E+00	1.3002E+01	1.1449E+01	9.9974E+00	**6.2357E+00**
F6	1.1856E-08	4.9680E-08	**1.0622E-08**	8.5095E-08	2.6812E-08
F7	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**
F8	-9.6511E+00	-9.6405E+00	-9.6493E+00	**-9.6518E+00**	-9.6490E+00
F9	6.1006E-04	6.1971E-04	6.1462E-04	6.1112E-04	**6.0140E-04**
F10	**1.8810E-15**	2.8682E-15	3.0013E-15	5.6226E-15	9.6345E-15
Friedman	**2.50**	3.20	2.80	3.70	2.80
Sıra	**1**	3	2	4	2

5.1.3. HBA2 algoritmasında kullanılan maksimum değişim sayısı (mds) değerinin belirlenmesi

Maksimum değişim sayısı, popülasyonlar arasındaki bilgi değişiminin belli bir yönde en fazla kaç defa yapılacağını belirleyen sayıdır. Bu değer, HBA2 algoritmasında Denklem 4.12'ye göre hesaplanmıştır. HBA2 algoritmasında, toplam değişim sayısının belli bir oranında başarılı olan algoritma ile kalan iterasyonlara, tüm popülasyon üzerinde devam edilir. İşte bu oran, Çizelge 5.4'te sonuçları verilen test işlemine göre belirlenmiştir. Buna göre, Çizelge 5.1'de verilen kıyaslama fonksiyonları üzerinde, [0.1, 1] aralığında 0.1 artışla belirlenen on farklı değer için fonksiyonlardan elde edilen ortalama değerler karşılaştırılmıştır. Çizelge 5.4'te verilen bu karşılaştırma sonuçları incelendiğinde, 0.6 değeri için toplam sekiz fonksiyonda en iyi ortalama değerin bulunduğu ve bu değerin Friedman testine göre yapılan sıralamada 4.35 ortalama sıra değeri ile birinci sırada yer aldığı görülmüştür. Bu sonuç doğrultusunda, HBA2 algoritmasında kullanılan mds değeri için 0.6 çarpanı seçilmiştir. Bu da toplam değişim sayısının %60'ında başarılı olan algoritmanın, kalan iterasyonlara tüm popülasyon üzerinde tek başına devam edeceği anlamına gelir.

Çizelge 5.4. mds parametresinin hesaplanmasında kullanılan çarpan değerinin belirlenmesi için yapılan testin sonuçları

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F1	3.9383E-03	9.6231E-03	1.0838E-02	3.2026E-03	5.0441E-03	**2.7100E-03**	3.6992E-03	5.1758E-03	9.6012E-03	5.4209E-03
F2	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F3	5.1017E-01	5.3635E-01	8.4681E-02	4.6986E-03	2.2237E-02	**1.5923E-03**	5.3061E-03	2.0684E-03	2.8696E-02	2.0130E-03
F4	1.3967E+02	1.3967E+02	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**
F5	2.3688E+01	2.3688E+01	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**
F6	7.3508E-17	8.1603E-18	6.0557E-18	3.9466E-18	**2.8216E-18**	8.1774E-18	8.9556E-18	5.9109E-18	2.2645E-17	2.3747E-17
F7	-9.9990E-01	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**
F8	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**
F9	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**
F10	1.4509E-04	1.9397E-04	2.0690E-04	1.9377E-04	1.1601E-04	1.7700E-05	5.7181E-06	6.9820E-07	1.7333E-07	**7.2323E-08**
Friedman	7.55	7.35	6.25	4.65	4.85	**4.35**	4.95	4.55	5.55	4.95
Sıra	9	8	7	3	4	**1**	5	2	6	5

5.1.4. HBA2 algoritmasında kullanılan limit değerinin belirlenmesi

HBA2 algoritması, BA ve ABC algoritmalarından oluşur. ABC algoritmasının önemli parametrelerinden birisi limit değeridir. Limit değeri, yiyecek kaynağının nektar miktarının tükenip tükenmediğini ifade etmek için kullanılan bir sayaçtır. ABC algoritmasında, bir yiyecek kaynağı limit değeri kadar araştırmada geliştirilemezse, terk edilir ve yerine rastgele yeni bir yiyecek kaynağı atanarak, arama işlemine devam edilir. Limit değerinin seçimi, algoritma performansını etkileyen bir faktördür. Küçük bir değer seçilmesi, kaynağın kısa sürede terk edilmesine ve kaynak etrafında yeterince araştırma yapılmamasına, gerekenden büyük seçilmesi ise bir yiyecek kaynağının uzun süre geliştirilememesine rağmen, çevresinde yapılan aramanın devam etmesine neden olur. Literatürde, limit değerinin belirlenmesine yönelik bazı genel yaklaşımlar ($limit = (n * d)$, $limit = (n * d * 0.5)$ gibi) mevcuttur (Akay, 2009). Ancak bu yaklaşımlar, özellikle büyük boyutlu problemlerde, artan boyuta ve birey sayısına bağlı olarak oldukça büyük limit değerlerinin elde edilmesine neden olabilir. Gerekenden büyük limit değeri nedeniyle, geliştirilemeyen bir çözümün etrafında uzun süre arama yapılabilir ve bu sebeple algoritmanın keşif yeteneğinin azalabilir.

Büyük boyutlu optimizasyon problemleri üzerinde de performansı incelenecek olan HBA2 algoritmasında, genel bir limit belirleme denklemi kullanmak yerine, farklı özellikte fonksiyonlardan oluşan bir fonksiyon kümesi üzerinde test işlemi yapılarak limit değerine karar verilmiştir. HBA2 algoritmasında kullanılacak limit değeri, küçük ve büyük boyutlu problemlerde kullanılmak üzere iki ayrı değer olarak belirlenmiştir. Bu değerler, Çizelge 5.1'de verilen kıyaslama fonksiyonları üzerinde yapılan test işlemleri sonucunda elde edilmiştir. Çizelge 5.5'te, D=10 için, limit değerinin [10, 25, 50, 100, 250, 500, 1000] alınarak, fonksiyonların yürütülmesi sonucu elde edilen ortalama değerler verilmiştir. Sonuçlar incelendiğinde, 100 değeri için toplam altı fonksiyonda en iyi ortalama değer bulunmuştur ve bu değer Friedman testine göre yapılan sıralamada 2.65 ortalama sıra değeri ile birinci sırada yer almıştır. Bu sonuç doğrultusunda, küçük boyutlu problemler için HBA2 algoritmasında limit değeri 100 olarak kullanılmıştır.

Çizelge 5.6'da ise, D=1000 için, limit değeri [10, 100, 500, 1000, 2000, 4000] alınarak fonksiyonların yürütülmesi sonucu elde edilen ortalama değerler verilmiştir.

Sonuçlar incelendiğinde, 2000 değeri için toplam dört fonksiyonda en iyi ortalama değer elde edilmiştir ve bu değer Friedman testine göre yapılan sıralamada 2.65 ortalama sıra değeri ile birinci sırada yer almıştır. Bu sonuç doğrultusunda, büyük boyutlu problemler için, HBA2 algoritmasında limit değeri 2000 olarak belirlenmiştir.

Çizelge 5.5. Küçük boyutlu problemlerde limit değerinin belirlenmesi için yapılan testin sonuçları

	10	25	50	100	250	500	1000
F1	2.0695E-01	1.3593E-03	7.4016E-04	**3.5503E-06**	3.7939E-03	5.9142E-03	6.4064E-03
F2	7.8552E+00	2.2595E-13	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F3	2.0510E+01	2.6279E-01	3.8399E-02	**2.1460E-02**	3.6746E-02	1.0900E-01	1.5022E-01
F4	1.4018E+02	1.3975E+02	1.3969E+02	1.3967E+02	**1.3966E+02**	**1.3966E+02**	**1.3966E+02**
F5	3.5736E+02	1.2749E-04	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**	**1.2728E-04**
F6	3.3990E-03	2.3965E-16	7.7605E-17	7.8704E-17	7.9151E-17	**7.1628E-17**	7.4759E-17
F7	-2.9492E-04	-1.9939E-03	**-9.2554E-03**	-4.3521E-03	-4.1168E-03	-7.8593E-04	-3.7033E-03
F8	-9.0555E+00	-9.6601E+00	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**	**-9.6602E+00**
F9	7.1657E-04	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**	**5.6607E-04**
F10	2.2344E+02	2.1426E+02	7.8465E+01	3.0014E+01	3.9078E+01	**2.4161E+01**	3.4050E+01
Friedman	7.0	5.35	3.15	**2.65**	3.25	3.15	3.45
Sıra	6	5	2	**1**	3	2	4

Çizelge 5.6. Büyük boyutlu problemlerde limit değerinin belirlenmesi için yapılan testin sonuçları

	10	100	500	1000	2000	4000
F1	8.4495E+03	1.1701E-11	3.4306E-14	1.7208E-14	**1.2546E-14**	1.2800E-14
F2	3.1559E+04	2.5396E+01	2.3649E+01	**2.3478E+01**	2.4792E+01	2.3612E+01
F3	2.0012E+08	9.6377E+01	2.2293E+02	1.7632E+02	1.4373E+02	**1.1451E+02**
F4	1.7318E+04	1.5574E+04	**1.5508E+04**	**1.5508E+04**	**1.5508E+04**	**1.5508E+04**
F5	2.5752E+05	1.1844E+04	**1.1556E+04**	1.1925E+04	1.1690E+04	1.2467E+04
F6	3.3084E+07	2.5909E-12	4.1584E-14	3.5735E-14	3.6406E-14	**3.4800E-14**
F7	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F8	-2.8492E+02	-8.6315E+02	-9.6845E+02	-9.6895E+02	**-9.6918E+02**	**-9.6918E+02**
F9	**1.7341E-84**	4.9161E-82	8.5810E-77	1.3475E-75	3.0617E-77	2.3900E-70
F10	1.6354E+05	1.6069E+05	1.5819E+05	**1.5669E+05**	1.5843E+05	1.6261E+05
Friedman	5.25	3.85	3.30	2.90	**2.65**	3.05
Sıra	6	5	4	2	**1**	3

5.2. CEC2005 Küçük Ölçekli Kıyaslama Fonksiyonları Üzerinde Yapılan Test İşlemi Sonuçları

Bu tez çalışması kapsamında önerilen HBA1 ve HBA2 algoritmalarının performans karşılaştırması, ilk olarak literatürde BA versiyonlarında sıklıkla kullanılmış olan CEC2005 (Congress of Evolutionary Computation 2005) küçük ölçekli kıyaslama fonksiyonları (Suganthan ve ark., 2005) üzerinde yapılmıştır. Bu şekilde, algoritmaların literatüre göre performansının incelenmesi amaçlanmıştır. CEC2005 kıyaslama fonksiyonları ve fonksiyonlara ait özellikler Çizelge 5.7'de verilmiştir. CEC2005 kıyaslama fonksiyonları tek modlu, çok modlu, genişletilmiş ve hibrit birleşim olmak üzere dört grupta ele alınmıştır ve toplam yirmi beş fonksiyondan oluşmaktadır. CEC2005 kriterlerine uygun olarak, FEs değeri $10000 * D$ olarak verilmiştir. Genel çevrim sayısı ise 25 olarak alınmıştır. Yani test edilen algoritma bağımsız olarak 25 defa yürütülmüştür. Her bir çevrim sonrası, fonksiyon hatası $(f(x) - f(x^*))$ kaydedilmiş ve küçükten büyüğe doğru sıralanmıştır. Fonksiyon hatası algoritmadan elde edilen sonuçla, global optimum değer arasındaki farkı ifade etmektedir. Sıralanmış hata değerlerinin 1. (en iyi), 7. , 13. (ortanca), 19. ve 25. (en kötü) değerleri ile ortalama ve standart sapma değerleri kaydedilmiştir. CEC2005 kıyaslama fonksiyonlarında, D=10 için standart BA, HBA1 ve HBA2 için elde edilen sonuçlar Çizelge 5.8'de verilmiştir. Bu çizelgede, her fonksiyon için minimum ortalama değer koyu fontla belirtilmiştir. Ayrıca, 10^{-8} den küçük değerler 0,00E+00 olarak gösterilmiştir.

Çizelge 5.7.a. CEC2005 kıyaslama fonksiyonları

Fonksiyon	Ad	Sınır	Minimum
Tek Modlu Fonksiyonlar			
F1	Kaydırılmış Sphere Fonksiyonu	[−100, 100]	−450
F2	Kaydırılmış Schwefel 1.2 Fonksiyonu	[−100, 100]	−450
F3	Kaydırılmış Döndürülmüş Elliptic Fonksiyonu	[−100, 100]	−450
F4	Kaydırılmış Schwefel 1.2 Fonksiyonu (gürültülenmiş uygunluk değeri)	[−100, 100]	−450
F5	Schwefel 2.6 Fonksiyonu (arama uzayı sınıra kaydırılmış global optimum noktası)	[−100, 100]	−310
Çok Modlu Fonksiyonlar			
F6	Kaydırılmış Rosenbrock Fonksiyonu	[−100, 100]	390
F7	Kaydırılmış Döndürülmüş Griewank Fonksiyonu (sınırlandırılmamış arama uzayı)	[0, 600]	−180
F8	Kaydırılmış Döndürülmüş Ackley Fonksiyonu (arama uzayı sınıra kaydırılmış global optimum noktası)	[−32, 32]	−140
F9	Kaydırılmış Rastrigin Fonksiyonu	[−5, 5]	−330
F10	Kaydırılmış Döndürülmüş Rastrigin Fonksiyonu	[−5, 5]	−330
F11	Kaydırılmış Döndürülmüş Weierstrass Fonksiyonu	[−0 .5, 0.5]	90
F12	Schwefel 2.13 Fonksiyonu	$[-\pi,\pi]$	−460
Genişletilmiş Fonksiyonlar			
F13	Genişletilmiş Geliştirilmiş Griewank ve Rosenbrock Fonksiyonu (F8F2)	[−3, 1]	−130
F14	Geliştirilmiş Döndürülmüş Genişletilmiş Scaffer Fonksiyonu F6	[−100, 100]	−300

Çizelge 5.7.b. CEC2005 kıyaslama fonksiyonları

Fonksiyon	Ad	Sınır	Minimum
Hibrit Bileşim Fonksiyonu			
F15	Hibrit Bileşim Fonksiyonu(Rastrigin, Weierstrass, Griewank, Ackley, Sphere fonksiyonları)	[−5, 5]	120
F16	Döndürülmüş Hibrit Bileşim Fonksiyonu (Döndürülmüş F15 fonksiyonu)	[−5, 5]	120
F17	Döndürülmüş Hibrit Bileşim Fonksiyonu (Uygunluk değerine gürültü eklenmiş F16 fonksiyonu)	[−5, 5]	120
F18	Döndürülmüş Hibrit Bileşim Fonksiyonu 2 (Ackley, Rastrigin, Sphere, Weierstrass, Griewank fonksiyonları)	[−5, 5]	10
F19	Döndürülmüş Hibrit Bileşim Fonksiyonu 2 (global optimum havzası daraltılmış F18 fonksiyonu)	[−5, 5]	10
F20	Döndürülmüş Hibrit Bileşim Fonksiyonu 2 (global optimum noktası arama uzay sınırına kaydırılmış F18 fonksiyonu)	[−5, 5]	10
F21	Döndürülmüş Hibrit Bileşim Fonksiyonu 3 (Döndürülmüş Genişletilmiş Scaffer F6, Rastrigin, F8F2, Weierstrass, Griewank fonksiyonları)	[−5, 5]	360
F22	Döndürülmüş Hibrit Bileşim Fonksiyonu 3 (döndürme matrisi koşulları artırılmış)	[−5, 5]	360
F23	Sürekli olmayan Döndürülmüş Hibrit Bileşim Fonksiyonu 3	[−5, 5]	360
F24	Döndürülmüş Hibrit Bileşim Fonksiyonu 4 (Weierstrass, Döndürülmüş Genişletilmiş Scaffer F6, F8F2, Ackley, Rastrigin, Griewank fonksiyonları)	[−5, 5]	260
F25	Döndürülmüş Hibrit Bileşim Fonksiyonu 4 (Sınırlandırılmamış arama uzayı)	[2, 5]	260

Çizelge 5.8'de verilen sonuçlar incelendiğinde, BA'nın bir fonksiyonda (F14), HBA2'nin altı fonksiyonda (F8, F12, F13, F15, F21, F23) en iyi ortalama değeri ürettiği görülmüştür. HBA1 ve HBA2, dört fonksiyonda (F1, F9, F22, F24) aynı ortalama değeri, kalan on dört fonksiyonda ise HBA1 algoritması en iyi ortalama değerleri üretmiştir. Çizelgede verilen Friedman test sonuçlarına göre ise HBA1 algoritması 1.38 ortalama ile ilk sırada, HBA2 algoritması 1.80 ortalama ile ikinci sırada, BA ise 2.82 ortalama ile üçüncü sırada yer almıştır. Buna göre, HBA1 algoritmasının performansının, CEC2005 kıyaslama fonksiyonları üzerinde BA ve HBA2'ye göre daha iyi olduğu söylenebilir.

Çizelge 5.9-5.11'de algoritmaların ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Bu test işleminde, algoritmaların her bir çevrim sonrasında elde ettiği ortalama değerler ikili olarak karşılaştırılmıştır. Belirtilen çizelgelerde, Daha iyi sütunu, yapılan ikili karşılaştırmaların kaçında ilk algoritmanın daha iyi sonuç ürettiğini yani minimizasyon problemi için daha küçük bir değer ürettiğinin bilgisini vermektedir. Daha kötü sütunu, ikili karşılaştırmaların kaçında ilk algoritmanın daha kötü sonuç ürettiğini yani minimizasyon problemi için daha büyük bir değer ürettiğini göstermektedir. Eşit sütunu ise iki algoritmanın, yapılan ikili karşılaştırmaların kaç tanesinde eşit değerler ürettiğinin bilgisini vermektedir. Wilcoxon işaretli sıra testi $\alpha < 0.05$ anlamlılık düzeyinde kullanılmıştır. Çizelgelerdeki p değeri anlamlılık düzeyini verir ve $p < 0.05$ olması algoritmalar arasında anlamlı bir fark olduğunu ifade

eder. Çizelgelerdeki Karar sütununda yer alan + ve – sembolleri algoritmalar arasında anlamlı bir fark olduğunu, ≈ sembolü ise algoritmalar arasında anlamlı bir fark olmadığını göstermektedir. + sembolü, algoritmanın karşılaştırma yapılan diğer algoritmaya göre anlamlı bir gelişme gösterdiğini, - sembolü ise algoritmanın karşılaştırma yapılan diğer algoritmadan anlamlı bir şekilde geride kaldığını gösterir. Karşılaştırılan algoritmalar arasında anlamlı bir fark varsa bunun + veya – olduğuna, algoritmaların ürettiği sonuçlar incelenerek karar verilir.

Çizelge 5.8.a. CEC2005 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları (D=10)

No		1.(En iyi)	7.	13.(Ortanca)	19.	25.(En kötü)	Ortalama	Std.Sapma
	BA	1.59E-03	2.89E-03	4.01E-03	5.47E-03	6.74E-03	4.17E-03	1.53E-03
F1	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	BA	2.44E-04	9.29E-04	1.16E-03	1.71E-03	2.71E-03	1.34E-03	6.83E-04
F2	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	6.02E-07	9.92E-06	2.80E-05	7.01E-05	3.35E-04	7.77E-05	1.10E-04
	BA	9.63E+02	9.16E+03	2.48E+04	7.85E+04	1.80E+05	4.28E+04	4.45E+04
F3	HBA1	2.31E-01	1.30E+01	8.08E+01	1.85E+03	8.32E+03	**1.33E+03**	2.18E+03
	HBA2	1.73E+02	8.71E+02	2.11E+03	4.14E+03	9.90E+03	2.95E+03	2.62E+03
	BA	2.81E+03	7.86E+03	1.05E+04	1.59E+04	2.25E+04	1.14E+04	5.16E+03
F4	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	4.89E-04	7.95E-04	2.03E-03	4.30E-03	4.15E-02	4.72E-03	8.66E-03
	BA	6.62E+01	5.41E+02	9.78E+02	1.71E+03	5.35E+03	1.20E+03	1.10E+03
F5	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	8.59E-08	1.81E-05	3.11E-04	1.16E-02	1.99E+00	1.47E-01	4.29E-01
	BA	5.49E-01	4.25E+00	5.73E+00	9.65E+00	3.01E+02	4.77E+01	8.98E+01
F6	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	7.31E-08	2.76E-02	7.72E-01	2.92E+00	1.68E+01	2.29E+00	3.98E+00
	BA	8.92E+02	1.11E+03	1.26E+03	1.38E+03	1.78E+03	1.26E+03	2.21E+02
F7	HBA1	2.32E+02	4.25E+02	5.07E+02	5.68E+02	6.75E+02	**4.85E+02**	1.18E+02
	HBA2	1.27E+03	1.27E+03	1.27E+03	1.27E+03	1.27E+03	1.27E+03	4.57E-13
	BA	2.01E+01	2.03E+01	2.03E+01	2.04E+01	2.04E+01	2.03E+01	9.19E-02
F8	HBA1	2.01E+01	2.02E+01	2.03E+01	2.03E+01	2.04E+01	2.03E+01	7.49E-02
	HBA2	2.00E+01	2.01E+01	2.02E+01	2.03E+01	2.04E+01	**2.02E+01**	1.18E-01
	BA	1.25E+01	2.45E+01	3.34E+01	4.16E+01	5.53E+01	3.32E+01	1.08E+01
F9	HBA1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	HBA2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	BA	2.04E+01	2.87E+01	3.41E+01	4.35E+01	5.57E+01	3.58E+01	9.62E+00
F10	HBA1	6.96E+00	8.95E+00	9.94E+00	1.19E+01	1.79E+01	**1.05E+01**	2.99E+00
	HBA2	7.96E+00	1.39E+01	1.69E+01	2.38E+01	4.58E+01	1.99E+01	9.27E+00
	BA	7.99E+00	8.62E+00	9.01E+00	9.47E+00	1.00E+01	9.10E+00	5.46E-01
F11	HBA1	3.70E-01	1.80E+00	2.64E+00	3.64E+00	4.78E+00	**2.61E+00**	1.21E+00
	HBA2	2.33E+00	3.50E+00	4.74E+00	5.37E+00	6.06E+00	4.50E+00	1.10E+00
	BA	3.87E+02	5.50E+02	8.12E+02	1.09E+03	2.49E+03	9.60E+02	5.47E+02
F12	HBA1	1.09E+02	1.82E+02	2.73E+02	4.21E+02	7.93E+02	3.15E+02	1.74E+02
	HBA2	1.52E+01	8.42E+01	1.66E+02	3.16E+02	6.07E+02	**2.17E+02**	1.64E+02

Çizelge 5.8.b. CEC2005 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları (D=10)

No		1.(En iyi)	7.	13.(Ortanca)	19.	25.(En kötü)	Ortalama	Std.Sapma
	BA	1.64E+00	2.82E+01	3.09E+01	3.54E+01	3.89E+01	3.12E+01	5.65E-01
F13	HBA1	3.95E-02	2.90E-01	3.58E-01	4.03E-01	5.53E-01	3.42E-01	9.84E-02
	HBA2	2.84E-14	1.42E-01	1.93E-01	2.39E-01	4.15E-01	**2.03E-01**	1.08E-01
	BA	3.54E+00	4.00E+00	4.25E+00	4.40E+00	4.54E+00	**4.20E+00**	2.66E-01
F14	HBA1	4.49E+00	4.61E+00	4.62E+00	4.64E+00	4.67E+00	4.62E+00	3.74E-02
	HBA2	4.77E+00	4.80E+00	4.81E+00	4.81E+00	4.84E+00	4.80E+00	1.71E-02
	BA	1.77E+02	3.01E+02	4.59E+02	5.53E+02	6.74E+02	4.43E+02	1.39E+02
F15	HBA1	0.00E+00	5.20E+01	4.00E+02	4.00E+02	4.06E+02	2.36E+02	1.78E+02
	HBA2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	BA	1.43E+02	1.55E+02	1.70E+02	1.98E+02	6.60E+02	2.17E+02	1.28E+02
F16	HBA1	9.38E+01	1.10E+02	1.16E+02	1.19E+02	1.33E+02	**1.14E+02**	8.45E+00
	HBA2	1.00E+02	1.17E+02	1.27E+02	1.35E+02	1.75E+02	1.29E+02	1.76E+01
	BA	1.44E+02	1.73E+02	2.02E+02	2.16E+02	5.11E+02	2.20E+02	9.09E+01
F17	HBA1	9.45E+01	1.08E+02	1.13E+02	1.23E+02	1.34E+02	**1.14E+02**	1.04E+01
	HBA2	1.22E+02	1.43E+02	1.58E+02	1.65E+02	1.73E+02	1.53E+02	1.54E+01
	BA	3.41E+02	8.24E+02	9.31E+02	1.01E+03	1.05E+03	9.11E+02	1.48E+02
F18	HBA1	3.00E+02	3.00E+02	3.00E+02	3.00E+02	1.02E+03	**3.88E+02**	2.11E+02
	HBA2	3.00E+02	3.57E+02	4.44E+02	5.00E+02	9.82E+02	5.00E+02	1.97E+02
	BA	3.24E+02	8.22E+02	9.59E+02	9.86E+02	1.03E+03	8.88E+02	1.51E+02
F19	HBA1	3.00E+02	3.00E+02	3.00E+02	3.00E+02	1.02E+03	**3.97E+02**	2.31E+02
	HBA2	3.56E+02	4.53E+02	5.00E+02	8.00E+02	9.32E+02	6.15E+02	1.96E+02
	BA	3.03E+02	8.01E+02	9.67E+02	9.97E+02	1.09E+03	8.36E+02	2.59E+02
F20	HBA1	3.00E+02	3.00E+02	3.00E+02	8.00E+02	1.02E+03	**5.14E+02**	2.97E+02
	HBA2	3.00E+02	3.56E+02	5.00E+02	8.00E+02	9.19E+02	5.28E+02	1.99E+02
	BA	3.25E+02	9.22E+02	1.08E+03	1.16E+03	1.25E+03	9.50E+02	3.11E+02
F21	HBA1	5.00E+02	5.00E+02	5.00E+02	5.00E+02	5.00E+02	5.00E+02	4.87E-13
	HBA2	2.00E+02	4.10E+02	4.11E+02	5.00E+02	9.00E+02	**4.45E+02**	1.57E+02
	BA	7.67E+02	7.77E+02	8.74E+02	9.08E+02	9.37E+02	8.55E+02	6.24E+01
F22	HBA1	3.00E+02	7.38E+02	7.46E+02	7.52E+02	8.00E+02	**7.29E+02**	9.06E+01
	HBA2	3.00E+02	7.58E+02	7.65E+02	7.71E+02	8.00E+02	**7.29E+02**	1.29E+02
	BA	5.59E+02	7.21E+02	1.04E+03	1.22E+03	1.25E+03	9.67E+02	2.77E+02
F23	HBA1	5.59E+02	5.59E+02	5.59E+02	5.59E+02	5.59E+02	5.59E+02	6.13E-13
	HBA2	4.25E+02	5.48E+02	5.48E+02	5.48E+02	5.59E+02	**5.24E+02**	5.04E+01
	BA	2.05E+02	3.96E+02	4.09E+02	7.22E+02	1.24E+03	5.79E+02	3.10E+02
F24	HBA1	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	**2.00E+02**	0.00E+00
	HBA2	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	**2.00E+02**	1.53E-12
	BA	2.06E+02	3.95E+02	4.15E+02	5.63E+02	1.15E+03	5.34E+02	2.34E+02
F25	HBA1	3.46E+02	4.34E+02	4.69E+02	5.01E+02	5.53E+02	**4.65E+02**	5.55E+01
	HBA2	6.01E+02	6.23E+02	8.20E+02	8.24E+02	8.31E+02	7.41E+02	1.03E+02
Friedman		BA	HBA1	HBA2				
		2.82	**1.38**	1.80				

Çizelge 5.9'da, CEC2005 kıyaslama fonksiyonlarında, HBA1 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, iki fonksiyonda (F8, F25), algoritmalar arasında anlamlı bir fark bulunamamıştır. BA'nın bir fonksiyonda (F14), HBA1'in ise kalan yirmi iki fonksiyonda anlamsal olarak daha iyi olduğu görülmüştür.

Çizelge 5.10'da, CEC2005 kıyaslama fonksiyonlarında, HBA2 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, bir fonksiyonda (F7), algoritmalar arasında anlamlı bir fark bulunamamıştır. BA'nın iki fonksiyonda (F14, F25), HBA2'nin ise kalan yirmi iki fonksiyonda anlamsal olarak daha iyi olduğu tespit edilmiştir.

Çizelge 5.11'de ise, CEC2005 kıyaslama fonksiyonlarında, HBA1 ve HBA2 algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, üç fonksiyonda (F1, F12, F20), algoritmalar arasında anlamlı bir fark bulunamamıştır. HBA2'nin altı fonksiyonda (F8, F13, F15, F21, F23, F24), HBA1'in ise kalan on altı fonksiyonda anlamsal olarak daha iyi olduğu bulunmuştur.

Çizelge 5.9. CEC2005 kıyaslama fonksiyonları için HBA1 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	25	0	0	1.20E-05	+	**F14**	0	25	0	1.20E-05	-
F2	25	0	0	1.20E-05	+	**F15**	20	5	0	1.90E-04	+
F3	24	1	0	1.40E-05	+	**F16**	25	0	0	1.20E-05	+
F4	25	0	0	1.20E-05	+	**F17**	25	0	0	1.20E-05	+
F5	25	0	0	1.20E-05	+	**F18**	25	0	0	1.20E-05	+
F6	25	0	0	1.20E-05	+	**F19**	24	1	0	1.60E-05	+
F7	25	0	0	1.20E-05	+	**F20**	22	3	0	4.00E-04	+
F8	19	6	0	6.15E-02	≈	**F21**	20	5	0	7.20E-05	+
F9	25	0	0	1.20E-05	+	**F22**	25	0	0	1.20E-05	+
F10	25	0	0	1.20E-05	+	**F23**	25	0	0	1.20E-05	+
F11	25	0	0	1.20E-05	+	**F24**	21	4	0	4.93E-03	+
F12	22	3	0	4.10E-05	+	**F25**	11	14	0	7.16E-01	≈
F13	25	0	0	1.20E-05	+						

Çizelge 5.10. CEC2005 kıyaslama fonksiyonları için HBA2 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	25	0	0	1.20E-05	+	**F14**	0	25	0	1.20E-05	-
F2	24	1	0	1.40E-05	+	**F15**	25	0	0	1.20E-05	+
F3	23	2	0	1.80E-05	+	**F16**	24	1	0	2.00E-05	+
F4	25	0	0	1.20E-05	+	**F17**	24	1	0	2.00E-05	+
F5	25	0	0	1.20E-05	+	**F18**	24	1	0	2.30E-05	+
F6	22	3	0	2.66E-04	+	**F19**	22	3	0	6.50E-05	+
F7	13	12	0	8.61E-01	≈	**F20**	20	5	0	1.00E-03	+
F8	21	4	0	1.26E-04	+	**F21**	23	2	0	5.80E-05	+
F9	25	0	0	1.20E-05	+	**F22**	25	0	0	1.20E-05	+
F10	21	4	0	8.10E-05	+	**F23**	25	0	0	1.20E-05	+
F11	25	0	0	1.20E-05	+	**F24**	25	0	0	1.20E-05	+
F12	23	2	0	1.80E-05	+	**F25**	3	22	0	1.00E-03	-
F13	25	0	0	1.20E-05	+						

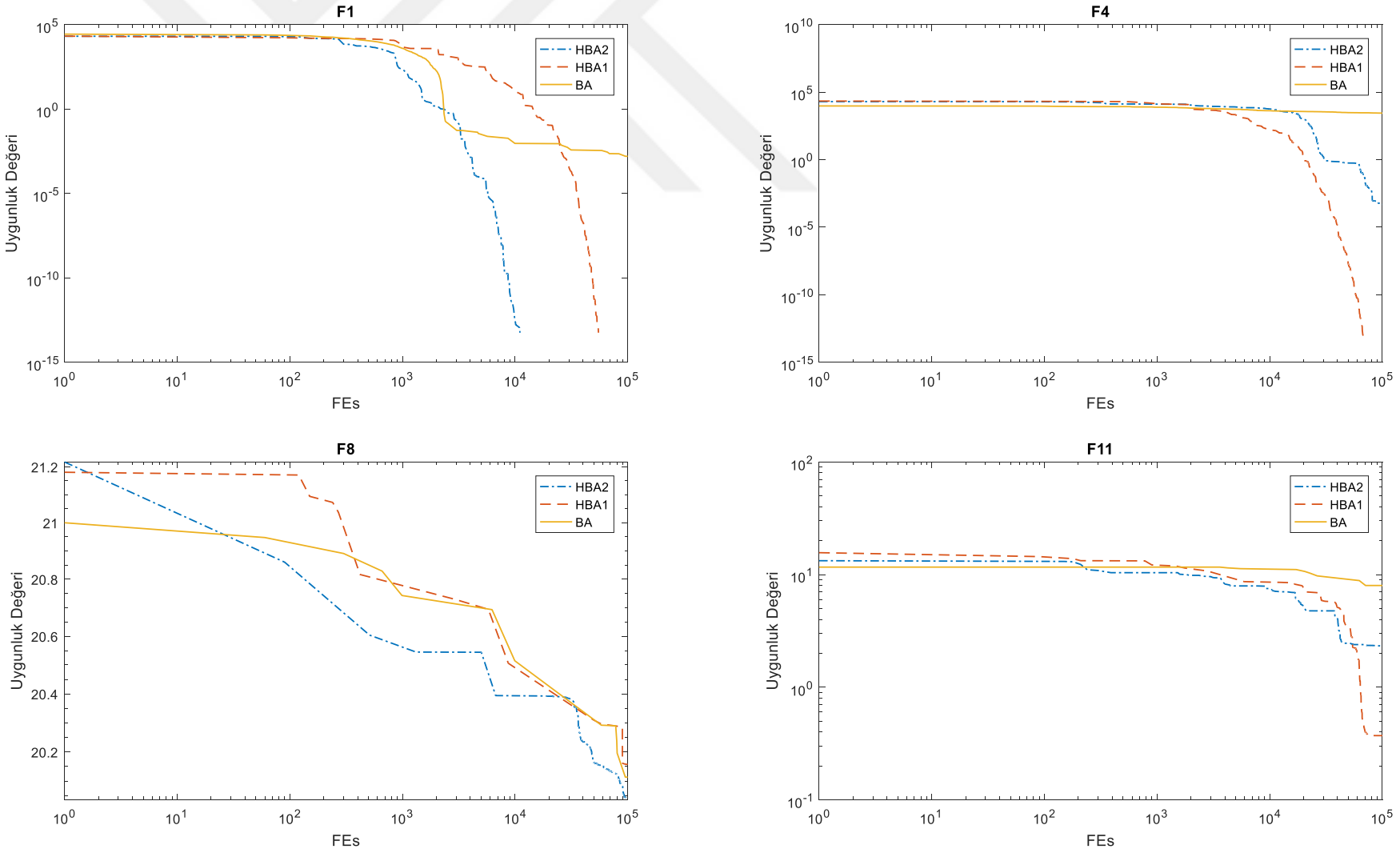
Çizelge 5.11. CEC2005 kıyaslama fonksiyonları için HBA1 ve HBA2 algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	0	0	25	1.00E+00	≈	**F14**	25	0	0	1.20E-05	+
F2	25	0	0	1.20E-05	+	**F15**	0	22	3	3.10E-05	-
F3	21	4	0	1.19E-02	+	**F16**	18	7	0	1.72E-03	+
F4	25	0	0	1.20E-05	+	**F17**	24	1	0	1.80E-05	+
F5	25	0	0	1.20E-05	+	**F18**	18	4	3	4.95E-02	+
F6	25	0	0	1.20E-05	+	**F19**	22	3	0	2.03E-03	+
F7	25	0	0	1.20E-05	+	**F20**	14	9	2	6.92E-01	≈
F8	6	19	0	1.19E-03	-	**F21**	2	23	0	2.08E-03	-
F9	25	0	0	1.20E-05	+	**F22**	21	4	0	9.42E-03	+
F10	23	2	0	2.30E-05	+	**F23**	0	25	0	1.20E-05	-
F11	24	1	0	2.30E-05	+	**F24**	0	6	19	2.77E-02	-
F12	8	17	0	5.78E-02	≈	**F25**	25	0	0	1.20E-05	+
F13	4	21	0	1.43E-03	-						

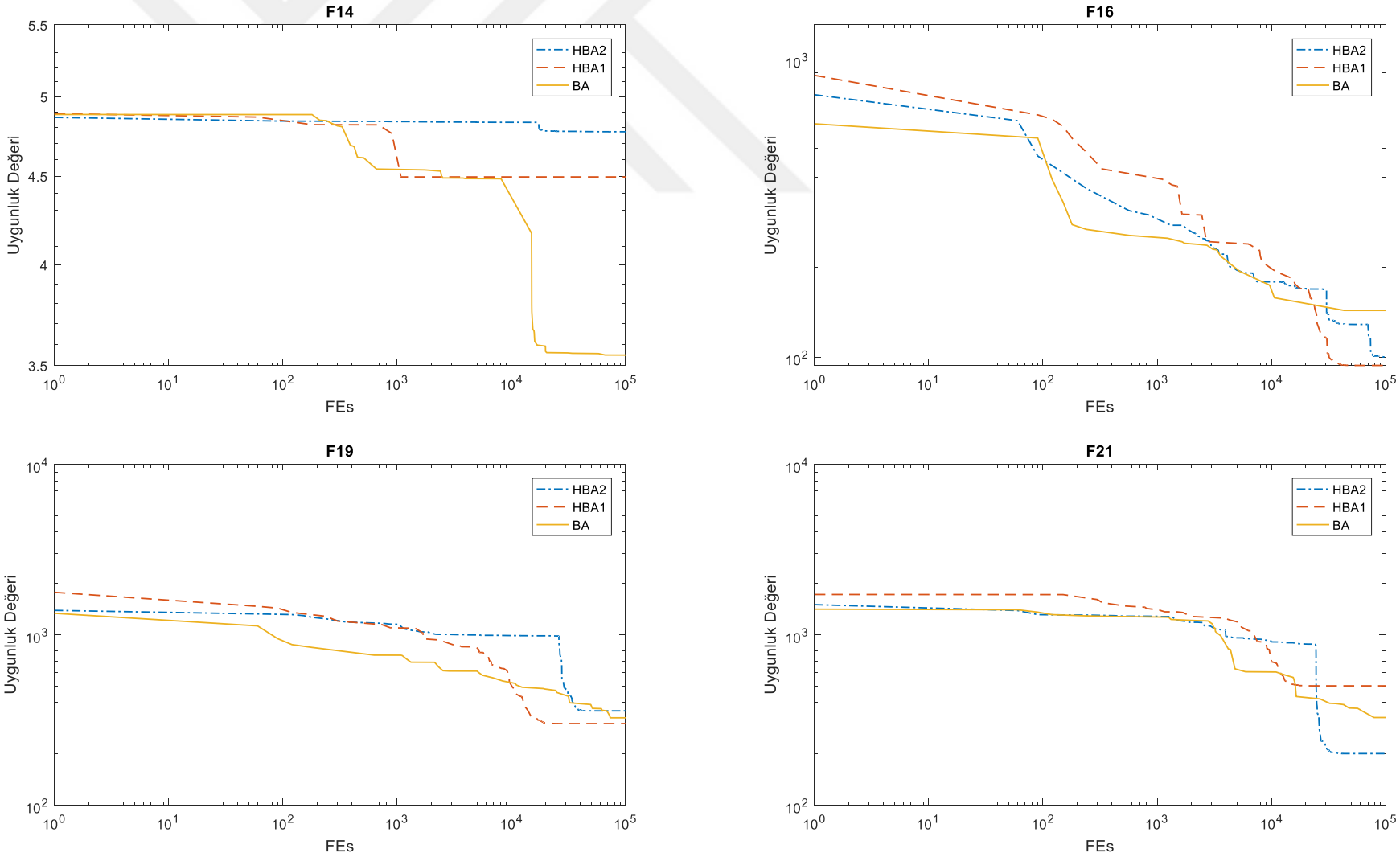
Şekil 5.1'de, CEC2005 kıyaslama fonksiyonlarının farklı gruplarından seçilen on adet fonksiyon için BA, HBA1 ve HBA2 algoritmalarından elde edilen en iyi değerlere ait yakınsama grafikleri verilmiştir. Grafikler incelendiğinde şu sonuçlar elde edilebilir. F1 fonksiyonunda, HBA1 ve HBA2 aynı uygunluk değerini üretmiştir ancak HBA2 daha hızlı yakınsamıştır. BA'nın performansı ise bu fonksiyonda diğer

algoritmaların gerisinde kalmıştır. F4 fonksiyonunda, en iyi uygunluk değeri HBA1 algoritmasında elde edilmiştir. BA algoritması çözümünü fazla geliştirememiştir. HBA2 ise çözümü geliştirmiş ancak performans olarak HBA1'in gerisinde kalmıştır. F8 fonksiyonunda, BA ve HBA1 benzer bir uygunluk değerini benzer bir yakınsama hızıyla bulmuştur. En iyi uygunluk değeri ise HBA2 algoritması tarafından elde edilmiştir ve daha hızlı yakınsamıştır. F11 fonksiyonunda, en iyi uygunluk değeri HBA1 algoritmasından elde edilmiştir. BA algoritması çözümünü fazla geliştirememiştir. HBA2 ise çözümü geliştirmiş ancak performans olarak HBA1'in gerisinde kalmıştır. HBA1 ve HBA2 algoritmaları 10000 FEs değerine kadar çözümü çok fazla geliştirememişlerdir. Sonrasında ise iki algoritmanında yakınsama hızı artmıştır. Ancak, HBA1'in yakınsama ve yeni çözüm uygunluk değeri açısından başarısı daha yüksek olmuştur. F14 fonksiyonunda, en iyi uygunluk değeri ve yakınsama hızı BA algoritması ile elde edilmiştir. HBA2 algoritması çözümü fazla geliştirememiştir. HBA1 ise 1000 FEs sonrası çözümü geliştirememiştir. F16 fonksiyonunda, en iyi uygunluk değeri ve yakınsama hızı bakımından HBA1'in ilk sırada, HBA2'nin ikinci sırada, BA'ın üçüncü sırada olduğu görülmektedir. F19 fonksiyonunda da, en iyi uygunluk değeri ve yakınsama hızı bakımından HBA1 ilk sırada, BA ikinci sırada ve HBA2 ise üçüncü sırada yer almıştır. F21 fonksiyonunda, yaklaşık olarak 5000 FEs değerine kadar algoritmaların yakınsama hızı ve yeni çözüm üretme performansı düşüktür. Devam eden süreçte ise en iyi uygunluk değeri HBA2 algoritmasından elde edilmiştir. Bu fonksiyon için en düşük performansı HBA1 algoritması göstermiştir. F22 ve F24 fonksiyonlarında, HBA1 ve HBA2 algoritmaları aynı uygunluk değerini üretmişlerdir. Ancak HBA2 algoritması daha hızlı yakınsamıştır. BA algoritmasının performansı ise bu fonksiyonlarda diğer algoritmaların gerisinde kalmıştır.

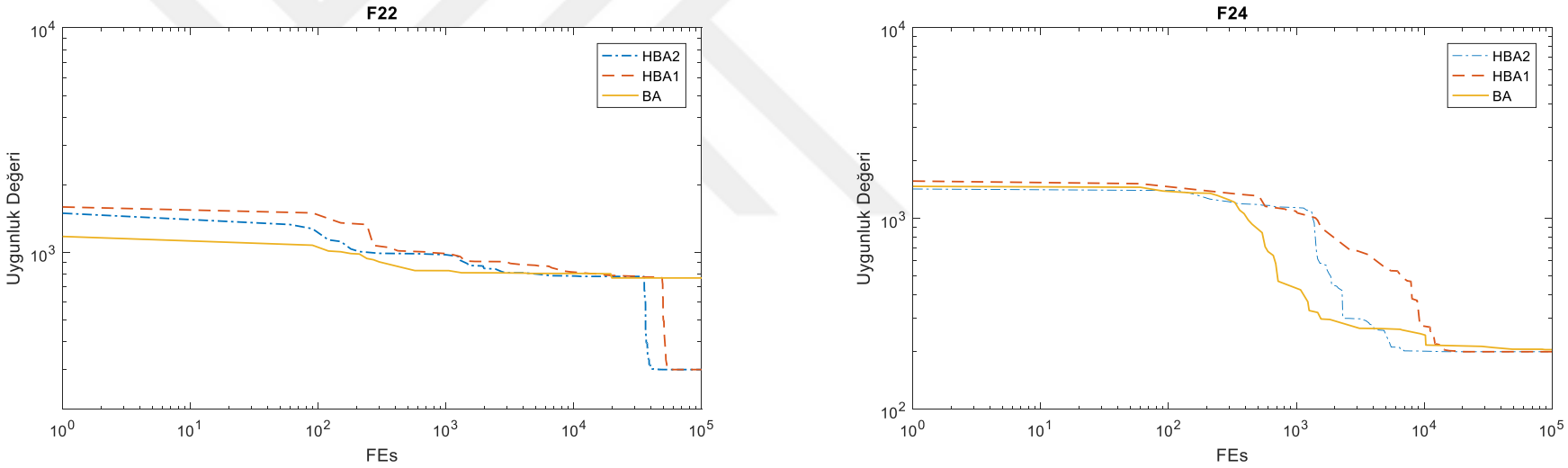
Genel bir değerlendirme yapıldığında, önerilen hibrit algoritmaların, standart BA algoritmasına göre fonksiyonların çoğunda daha hızlı yakınsadığı ve daha iyi uygunluk değerli çözümler ürettiği görülmektedir.



Şekil 5.1.a. CEC2005 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri



Şekil 5.1.b. CEC2005 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri



Şekil 5.1.c. CEC2005 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri

Önerilen hibrit algoritmalar, literatürdeki BA versiyonlarıyla da karşılaştırılmıştır. Karşılaştırmada kullanılan BA algoritmalarına ait bilgiler Çizelge 5.12'de verilmiştir. Karşılaştırılan tüm algoritmalarda, CEC2005 kıyaslama fonksiyonları D=10 için yürütülmüştür. Fonksiyon kümesinin kuralları gereğince, FEs değeri $10000 \times D$ ve genel çevrim sayısı 25 olarak alınmıştır. Elde edilen ortalama değerlere ait karşılaştırma sonuçları Çizelge 5.13'de verilmiştir.

Çizelge 5.12. Karşılaştırma yapılan BA algoritmalarına ait bilgiler

Algoritma	Makale Adı	Referans
HBA	A New Hybrid Bat Algorithm and its Application to the ROP Optimization in Drilling Processes	(Gan ve ark., 2020)
NBA	A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization	(Meng ve ark., 2015 ; Al-Betar ve ark., 2018)
iBA	Island bat algorithm for optimization	(Al-Betar ve Awadallah, 2018)
ILSSIWBA	A new bat algorithm based on iterative local search and stochastic inertia weight	(Gan ve ark., 2018)
GBA, TBA PBA, LBA EBA, RBA	Bat-inspired algorithms with natural selection mechanisms for global optimization	(Al-Betar ve ark., 2018)
dBA	New directional bat algorithm for continues optimization problems	(Chakri ve ark., 2017)

Çizelge 5.13 incelendiğinde, HBA1 algoritmasının on bir fonksiyonda, HBA2'nin ise altı fonksiyonda en iyi ortalama değeri ürettiği görülmektedir. Karşılaştırılan algoritmalardan HBA yedi fonksiyonda, NBA bir fonksiyonda, iBA dört fonksiyonda, ILSSIWBA beş fonksiyonda, GBA üç fonksiyonda, TBA, PBA, LBA ve EBA iki fonksiyonda ve dBA sekiz fonksiyonda en iyi ortalama değeri üretmiştir. RBA ise hiçbir fonksiyonda en iyi ortalama değeri üretememiştir. Algoritmalar için Friedman testine göre yapılan sıralamada 4.08 ortalama sıra değeri ile dBA algoritması birinci olmuştur. HBA1, 4.34 ortalama sıra değeri ile ikinci sırada, HBA2 ise 5.72 ortalama sıra değeri ile beşinci sırada yer almıştır.

Çizelge 5.14 ve 5.15'te ise sırasıyla HBA1 ve HBA2 algoritmalarının karşılaştırıldıkları diğer BA algoritmalarıyla olan ikili Wilcoxon işaretli sıra testi sonuçları yer almaktadır.

Çizelge 5.13. CEC2005 kıyaslama fonksiyonları için HBA1 ve HBA2 algoritmalarının BA versiyonları ile karşılaştırma sonuçları (D=10)

Fonksiyon	**Algoritmalar**												
	HBA1	**HBA2**	HBA	NBA	iBA	ILSSIWBA	GBA	TBA	PBA	LBA	EBA	RBA	dBA
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**00.00E+00**	**0.00E+00**	8.09E+03	**0.00E+00**
F2	**0.00E+00**	7.77E-05	**0.00E+00**	4.38E-07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.88E+03	**0.00E+00**
F3	1.33E+03	2.95E+03	**6.43E-01**	3.46E+03	2.48E+01	1.67E+03	2.41E+02	1.43E+02	2.74E+03	5.11E+02	2.21E+02	1.66E+07	2.38E+05
F4	**0.00E+00**	4.72E-03	**0.00E+00**	2.54E+03	2.99E+03	1.92E-04	3.81E+02	1.54E+02	9.79E+03	5.88E+03	1.77E+03	1.16E+04	1.13E-04
F5	**0.00E+00**	1.47E-01	2.92E-02	8.81E-02	1.47E-01	**0.00E+00**	2.92E+01	6.26E+00	1.22E+02	2.56E+01	5.74E+00	3.60E+03	**0.00E+00**
F6	**0.00E+00**	2.29E+00	3.35E+01	7.97E-01	1.69E−03	5.64E+00	4.96E−02	2.91E−02	4.36E+02	2.20E−02	1.44E+00	4.48E+08	6.64E+01
F7	4.85E+02	1.27E+03	**2.40E-01**	9.32E+00	4.59E+01	1.26E+03	7.04E+01	5.36E+01	8.74E+01	8.98E+01	5.42E+01	4.26E+02	3.87E-01
F8	2.03E+01	2.02E+01	2.03E+01	2.01E+01	**2.00E+01**	2.01E+01	2.01E+01	2.01E+01	2.02E+01	2.02E+01	2.02E+01	2.02E+01	2.03E+01
F9	**0.00E+00**	**0.00E+00**	1.75E+01	3.82E+01	2.48E+01	2.09E+01	3.94E+01	3.59E+01	7.35E+01	4.79E+01	5.96E+01	7.87E+01	7.88E+00
F10	**1.05E+01**	1.99E+01	1.97E+01	6.64E+01	4.08E+01	1.97E+01	5.53E+01	4.58E+01	9.17E+01	5.22E+01	7.85E+01	9.88E+01	1.08E+01
F11	2.61E+00	4.50E+00	4.61E+00	7.07E+00	3.12E+00	5.54E+00	**1.17E+00**	1.30E+00	7.94E+00	1.44E+00	1.46E+00	8.66E+00	3.22E+00
F12	3.15E+02	2.17E+02	8.50E+02	3.69E+03	**2.20E-02**	7.29E+01	7.54E+01	4.25E+01	2.82E+01	4.18E+01	7.40E+01	1.02E+05	1.54E+02
F13	3.42E-01	**2.03E-01**	1.03E+00	2.61E+00	2.56E+00	1.03E+00	1.68E+00	1.34E+00	1.43E+00	1.03E+00	1.99E+00	4.65E+00	8.93E-01
F14	4.62E+00	4.80E+00	3.02E+00	3.34E+00	3.92E+00	3.88E+00	3.97E+00	3.94E+00	4.03E+00	3.99E+00	4.24E+00	4.13E+00	**2.95E+00**
F15	2.36E+02	**0.00E+00**	1.41E+02	4.99E+02	2.83E+02	1.25E+02	6.52E+02	6.23E+02	6.62E+02	6.66E+02	7.31E+02	7.02E+02	2.07E+02
F16	**1.14E+02**	1.29E+02	1.39E+02	2.43E+02	1.93E+02	1.39E+02	4.58E+02	4.99E+02	4.80E+02	4.61E+02	4.38E+02	4.67E+02	1.15E+02
F17	**1.14E+02**	1.53E+02	1.42E+02	2.89E+02	1.81E+02	1.39E+02	2.03E+02	2.31E+02	4.73E+02	3.39E+02	3.76E+02	5.61E+02	1.21E+02
F18	**3.88E+02**	5.00E+02	4.32E+02	9.81E+02	9.53E+02	7.48E+02	1.15E+03	1.04E+03	1.08E+03	1.11E+03	1.07E+03	1.16E+03	4.85E+02
F19	3.97E+02	6.15E+02	**3.32E+02**	1.02E+03	8.97E+02	7.89E+02	9.77E+02	8.94E+02	1.00E+03	1.03E+03	1.09E+03	1.09E+03	4.20E+02
F20	5.14E+02	5.28E+02	4.51E+02	1.07E+03	8.05E+02	6.97E+02	8.80E+02	9.95E+02	1.01E+03	1.03E+03	1.08E+03	1.14E+03	**3.96E+02**
F21	5.00E+02	4.45E+02	4.91E+02	1.09E+03	1.23E+03	**3.20E+02**	1.29E+03	1.23E+03	1.29E+03	1.27E+03	1.29E+03	1.41E+03	4.08E+02
F22	7.29E+02	7.29E+02	7.32E+02	9.25E+02	8.54E+02	7.84E+02	8.26E+02	8.55E+02	9.10E+02	8.24E+02	9.82E+02	9.66E+02	**6.30E+02**
F23	5.60E+02	**5.24E+02**	5.94E+02	1.20E+03	1.23E+03	6.93E+02	1.42E+03	1.40E+03	1.39E+03	1.40E+03	1.42E+03	1.47E+03	5.46E+02
F24	**2.00E+02**	**2.00E+02**	**2.00E+02**	1.04E+03	9.49E+02	**2.00E+02**	1.34E+03	7.43E+02	9.76E+02	9.95E+02	7.60E+02	1.43E+03	**2.00E+02**
F25	4.65E+02	7.41E+02	5.02E+02	1.07E+03	9.46E+02	1.64E+03	8.14E+02	5.72E+02	9.82E+02	9.36E+02	5.43E+02	1.12E+03	**3.60E+02**
Friedman	4.34	5.72	4.42	8.28	6.00	5.38	7.64	6.50	9.52	8.06	8.72	12.34	**4.08**
Sıra	2	5	3	10	6	4	8	7	12	9	11	13	**1**

Çizelge 5.14 incelendiğinde, HBA1 algoritmasının HBA2, HBA ve dBA algoritmalarıyla benzer performans gösterdiği ve aralarında anlamlı bir fark bulunmadığı görülmüştür. HBA1'in performansı, kalan dokuz algoritmaya göre daha iyidir ve aralarında anlamsal bir fark mevcuttur. Çizelge 5.15 sonuçları incelendiğinde ise HBA2 algoritmasının HBA1, HBA, ILSSIWBA, TBA, EBA ve dBA algoritmasıyla benzer performans gösterdiği ve aralarında anlamlı bir fark olmadığı bulunmuştur. HBA2 algoritması, kalan altı algoritmadan daha iyi performans göstermiştir ve aralarında anlamlı bir fark mevcuttur. Bu sonuçlara göre, önerilen hibrit algoritmaların literatüre kıyasla yarışmacı, kabul edilebilir ve başarılı sonuçlar ürettiği söylenebilir.

Çizelge 5.14. CEC2005'te HBA1 ve BA versiyonları için ikili Wilcoxon işaretli sıra testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA1-HBA2	15	6	4	8.53E-02	≈
HBA1-HBA	13	7	5	7.09E-01	≈
HBA1-NBA	21	3	1	2.55E-04	+
HBA1- iBA	17	5	3	1.85E-02	+
HBA1- ILSSIWBA	16	5	4	1.89E-02	+
HBA1-GBA	16	6	3	1.70E-02	+
HBA1-TBA	16	6	3	2.20E-02	+
HBA1-PBA	19	4	2	5.26E-04	+
HBA1-LBA	17	6	2	8.90E-03	+
HBA1-EBA	17	6	2	8.90E-03	+
HBA1-RBA	22	3	0	3.20E-05	+
HBA1-dBA	11	9	5	4.78E-01	≈

Çizelge 5.15. CEC2005'te HBA2 ve BA versiyonları için ikili Wilcoxon işaretli sıra testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA2-HBA1	6	15	4	8.53E-02	≈
HBA2-HBA	11	12	2	7.38E-01	≈
HBA2-NBA	18	6	1	1.24E-03	+
HBA2- iBA	15	8	2	4.15E-02	+
HBA2- ILSSIWBA	12	11	2	2.48E-01	≈
HBA2-GBA	16	8	1	3.70E-02	+
HBA2-TBA	15	9	1	9.18E-02	≈
HBA2-PBA	18	5	2	4.25E-03	+
HBA2-LBA	17	7	1	1.91E-02	+
HBA2-EBA	15	8	2	5.15E-02	≈
HBA2-RBA	22	2	1	1.15E-04	+
HBA2-dBA	7	16	2	1.21E-01	≈

5.3. CEC2010 Büyük Ölçekli Kıyaslama Fonksiyonları Üzerinde Yapılan Test İşlemi Sonuçları

Tez kapsamında önerilen HBA1 ve HBA2 algoritmalarının performansı, ikinci olarak büyük ölçekli kıyaslama fonksiyonlarından oluşan CEC2010(Congress of Evolutionary Computation 2010) kıyaslama fonksiyonları üzerinde incelenmiştir. CEC2010 kıyaslama fonksiyonları (Tang ve ark., 2009) Çizelge 5.16'da verilmiştir. Çizelgede görüldüğü gibi CEC2010 fonksiyonları 20 tanedir ve toplam beş gruptan oluşur. CEC2010 kuralları gereği FEs 3.00E+06, genel çevrim sayısı 25 olarak alınır. Ayrıca 1.20E+05, 6.00E+05 ve 3.00E+06 FEs'de elde edilen sonuçlar kaydedilerek, üretilen çözümlerin gelişim durumu incelenir. Bu fonksiyonlarda sırasıyla D=1000 ve m=50 alınmaktadır.

Çizelge 5.16. CEC2010 kıyaslama fonksiyonları

No	***İsim***	***Özellik***	***Aralık***
Bölünebilir fonksiyonlar			
F1	Kaydırılmış Elliptic Fonksiyonu	Tek modlu	[−100, 100]
F2	Kaydırılmış Rastrigin Fonksiyonu	Çok modlu	[−5, 5]
F3	Kaydırılmış Ackley Fonksiyonu	Çok modlu	[−32, 32]
Tek grup m-Bölünemeyen fonksiyonlar			
F4	Tek grup kaydırılmış ve m-döndürülmüş Elliptic Fonksiyonu	Tek modlu	[−100, 100]
F5	Tek grup kaydırılmış ve m-döndürülmüş Rastrigin Fonksiyonu	Çok modlu	[−5, 5]
F6	Tek grup kaydırılmış ve m-döndürülmüş Ackley Fonksiyonu	Çok modlu	[−32, 32]
F7	Tek grup kaydırılmış m-boyutlu Schwefel 1.2 Fonksiyonu	Tek modlu	[−100, 100]
F8	Tek grup kaydırılmış m-boyutlu Rosenbrock Fonksiyonu	Çok modlu	[−100, 100]
D/2m grup m-Bölünemeyen fonksiyonlar			
F9	D/2m -grup kaydırılmış ve m-döndürülmüş Elliptic Fonksiyonu	Tek modlu	[−100, 100]
F10	D/2m -grup kaydırılmış ve m-döndürülmüş Rastrigin Fonksiyonu	Çok modlu	[−5, 5]
F11	D/2m -grup kaydırılmış ve m-döndürülmüş Ackley Fonksiyonu	Çok modlu	[−32, 32]
F12	D/2m -grup kaydırılmış m-boyutlu Schwefel 1.2 Fonksiyonu	Tek modlu	[−100, 100]
F13	D/2m -grup kaydırılmış m-boyutlu Rosenbrock Fonksiyonu	Çok modlu	[−100, 100]
D/m grup m-Bölünemeyen fonksiyonlar			
F14	D/m -grup kaydırılmış ve m-döndürülmüş Elliptic Fonksiyonu	Tek modlu	[−100, 100]
F15	D/m -grup kaydırılmış ve m-döndürülmüş Rastrigin Fonksiyonu	Çok modlu	[−5, 5]
F16	D/m -grup kaydırılmış ve m-döndürülmüş Ackley Fonksiyonu	Çok modlu	[−32, 32]
F17	D/m -grup kaydırılmış m-boyutlu Schwefel 1.2 Fonksiyonu	Tek modlu	[−100, 100]
F18	D/m -grup kaydırılmış m-boyutlu Rosenbrock Fonksiyonu	Çok modlu	[−100, 100]
Bölünemeyen fonksiyonlar			
F19	Kaydırılmış Schwefel 1.2 Fonksiyonu	Tek modlu	[−100, 100]
F20	Kaydırılmış Rosenbrock Fonksiyonu	Çok modlu	[−100, 100]

Çizelge 5.17, CEC2010 fonksiyonlarında standart BA, HBA1 ve HBA2 algoritmalarının, 1.20E+05 FEs için elde ettikleri sonuçları karşılaştırmalı olarak göstermektedir. Her fonksiyon için en iyi ortalama uygunluk değeri koyu fontla belirtilmiştir. Bu üç algoritmanın, ilk 1.20E+05 FEs için performansları değerlendirildiğinde, BA'nın bir fonksiyonda(F8), HBA1'in beş fonksiyonda (F5, F6, F11, F15, F16), HBA2'nin ise kalan on dört fonksiyonda en iyi ortalama uygunluk değerini ürettiği görülmektedir. Ayrıca bölünebilir ve bölünemeyen gruplardaki tüm fonksiyonlarda (F1,F2,F3,F19,F20), HBA2 algoritması en iyi ortalama uygunluk değerlerini üretmiştir. Dolayısıyla, bu fonksiyon gruplarında HBA2'nin daha başarılı olduğu sonucuna varılabilir. Çizelgenin sonundaki Friedman testi sıralama sonuçları incelendiğinde ise, 1.35 ortalama sıralama değeri ile HBA2 ilk sırada, 1.90 ortalama sıralama değeri ile HBA1 ikinci sırada, 2.75 ortalama sıralama değeri ile BA üçüncü sırada yer almıştır.

Çizelge 5.18, CEC2010 fonksiyonlarında standart BA, HBA1 ve HBA2 algoritmalarının, 6.00E+05 FEs için elde ettikleri sonuçları karşılaştırmalı olarak göstermektedir. Çizelgedeki sonuçlar incelendiğinde, BA'nın bir fonksiyonda (F4), HBA1'in yedi fonksiyonda (F5, F6, F7, F8, F11, F15, F16), HBA2'nin ise kalan on iki fonksiyonda en iyi ortalama uygunluk değerini elde ettiği görülmektedir. Yine, bölünebilen ve bölünemeyen gruplardaki tüm fonksiyonlarda HBA2 algoritması diğer algoritmalardan daha iyi performans göstermiştir. BA algoritması, 1.20E+05 FEs değerine kadar F8 fonksiyonunda performansı diğer algoritmalara göre daha iyi iken, 6.00E+05 FEs değerine ulaştığında bu fonksiyondaki performansı düşmüştür. F4 fonksiyonunda ise diğer algoritmalara göre daha iyi performans göstermiştir. HBA1 algoritması, 6.00E+05 FEs'e ulaşıldığında, 1.20E+05 FEs değerine kadarki değerlendirmede performansının yüksek olduğu beş fonksiyondaki başarısını sürdürmüştür ve ek olarak F7 ve F8 fonksiyonlarındaki performansını da artırarak en iyi ortalama uygunluk değerlerini elde etmiştir. 6.00E+05 FEs'e ulaşıldığında HBA2 algoritması ise F4 ve F7 fonksiyonlarındaki performansında düşüş yaşamıştır. F4 ve F7 için en iyi ortalama uygunluk değeri sırasıyla BA ve HBA1 algoritmalarında elde edilmiştir. Çizelgenin sonundaki Friedman testi sıralama sonuçları incelendiğinde ise, 1.55 ortalama sıralama değeri ile HBA2 ilk sırada, 1.65 ortalama sıralama değeri ile HBA1 ikinci sırada, 2.80 ortalama sıralama değeri ile BA üçüncü sırada yer almıştır.

Çizelge 5.19, CEC2010 fonksiyonlarında standart BA, HBA1 ve HBA2 algoritmalarının, 3.00E+06 FEs için elde ettikleri sonuçları karşılaştırmalı olarak göstermektedir. Çizelgedeki sonuçlar incelendiğinde, HBA1'in sekiz fonksiyonda (F5, F6, F7, F8, F11, F15, F16, F18), HBA2'nin ise kalan on iki fonksiyonda en iyi ortalama uygunluk değerini elde ettiği görülmektedir. BA algoritması, hiçbir fonksiyonda en iyi ortalama uygunluk değerini elde edememiştir. Dolayısıyla, 6.00E+05 FEs için yapılan incelemede, en iyi ortalama uygunluk değerini ürettiği F4 fonksiyonundaki performansını kaybetmiştir. HBA1 algoritması, 6.00E+05 FEs için yapılan değerlendirmede iyi performans gösterdiği fonksiyonlardaki başarısını sürdürmüştür ve ek olarak 3.00E+06 FEs 'e ulaşıldığında F18 fonksiyonunda da en iyi ortalama uygunluk değerini elde etmiştir. HBA2 algoritması ise, 3.00E+06 FEs değerine ulaşıldığında, 6.00E+05 FEs değerlendirmesinde en iyi ortalama uygunluk değeri ürettiği F18 fonksiyonundaki başarısını kaybetmiştir, ancak bunun yanında F4 fonksiyonundaki performansı artmış ve bu fonksiyon için en iyi ortalama uygun değerini üretmiştir. Yine, bölünebilir ve bölünemeyen gruplardaki tüm fonksiyonlarda HBA2 algoritması diğer algoritmalardan daha iyi performans göstermiştir. Friedman testi sıralama sonuçları incelendiğinde ise, 1.45 ortalama sıralama değeri ile HBA2 ilk sırada, 1.60 ortalama sıralama değeri ile HBA1 ikinci sırada, 2.95 ortalama sıralama değeri ile BA üçüncü sırada yer almıştır.

Genel olarak, önerilen hibrit algoritmalar BA algoritmasına göre daha başarılı sonuçlar üretmiştir. Friedman testine göre yapılan sıralamalar doğrultusunda ise CEC2010 fonksiyonlarında başarı sıralamasının en başarılıdan başarısıza doğru HBA2, HBA1 ve BA şeklinde olduğu belirlenmiştir.

Çizelge 5.17. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları (FEs=1.20E+05)

	F1			F2			F3			F4		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.89E+10	9.58E+09	1.02E+09	1.87E+04	1.37E+04	7.50E+03	2.11E+01	2.00E+01	2.00E+01	1.72E+12	9.05E+12	3.67E+12
Ortanca	7.67E+10	1.26E+10	1.29E+09	2.39E+04	1.43E+04	8.06E+03	2.14E+01	2.02E+01	2.00E+01	4.72E+12	1.71E+13	5.00E+12
En kötü	8.96E+10	1.39E+10	1.45E+09	2.47E+04	1.48E+04	8.43E+03	2.15E+01	2.03E+01	2.01E+01	2.11E+13	2.53E+13	9.26E+12
Ortalama	7.55E+10	1.23E+10	**1.27E+09**	2.38E+04	1.43E+04	**8.04E+03**	2.14E+01	2.02E+01	**2.00E+01**	6.22E+12	1.74E+13	**5.36E+12**
Std.Sapma	1.22E+10	1.03E+09	1.08E+08	1.14E+03	2.55E+02	2.29E+02	6.88E-02	6.53E-02	4.33E-02	4.16E+12	3.88E+12	1.37E+12
	F5			**F6**			**F7**			**F8**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.86E+08	2.74E+08	4.20E+08	2.06E+07	1.34E+06	1.98E+07	3.66E+08	3.18E+09	4.93E+07	4.90E+07	1.15E+09	7.48E+08
Ortanca	7.61E+08	3.52E+08	5.55E+08	2.09E+07	1.84E+06	2.03E+07	1.65E+10	5.20E+09	7.93E+07	5.26E+07	2.07E+09	1.69E+09
En kötü	3.96E+08	3.80E+08	6.85E+08	2.11E+07	2.39E+06	2.07E+07	5.50E+10	8.81E+09	1.08E+08	5.52E+09	7.70E+09	7.35E+09
Ortalama	4.45E+08	**3.52E+08**	5.50E+08	2.09E+07	**1.79E+06**	2.03E+07	2.09E+10	5.19E+09	**7.86E+07**	**4.14E+08**	2.68E+09	2.24E+09
Std.Sapma	1.37E+08	2.63E+07	7.19E+07	1.37E+05	2.91E+05	2.44E+05	1.42E+10	1.46E+09	1.55E+07	1.13E+09	1.43E+09	1.87E+09
	F9			**F10**			**F11**			**F12**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	3.50E+10	1.67E+10	1.48E+09	1.88E+04	1.37E+04	1.13E+04	2.32E+02	2.00E+02	2.29E+02	3.36E+06	2.64E+06	4.44E+05
Ortanca	1.09E+11	1.95E+10	1.74E+09	2.40E+04	1.48E+04	1.25E+04	2.35E+02	2.04E+02	2.29E+02	5.48E+06	2.97E+06	5.27E+05
En kötü	8.66E+10	2.10E+10	1.92E+09	2.46E+04	1.59E+04	1.37E+04	2.35E+02	2.11E+02	2.30E+02	6.29E+06	3.13E+06	5.97E+05
Ortalama	8.62E+10	1.95E+10	**1.74E+09**	2.38E+04	1.48E+04	**1.24E+04**	2.35E+02	**2.05E+02**	2.29E+02	5.39E+06	2.97E+06	**5.33E+05**
Std.Sapma	1.42E+10	1.12E+09	1.13E+08	1.11E+03	6.46E+02	5.47E+02	6.64E-01	3.25E+00	5.91E-01	5.58E+05	1.16E+05	3.33E+04
	F13			**F14**			**F15**			**F16**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	8.48E+10	6.15E+10	4.07E+06	3.67E+10	1.98E+10	1.97E+09	1.84E+04	1.36E+04	1.45E+04	4.22E+02	4.05E+02	4.16E+02
Ortanca	5.40E+11	7.29E+10	5.55E+06	9.34E+10	2.23E+10	2.16E+09	2.44E+04	1.43E+04	1.52E+04	4.28E+02	4.10E+02	4.18E+02
En kötü	6.34E+11	9.48E+10	1.01E+07	1.12E+11	2.41E+10	2.37E+09	2.50E+04	1.58E+04	1.62E+04	4.29E+02	4.18E+02	4.21E+02
Ortalama	5.31E+11	7.40E+10	**5.71E+06**	9.26E+10	2.24E+10	**2.16E+09**	2.42E+04	**1.44E+04**	1.53E+04	4.28E+02	**4.11E+02**	4.18E+02
Std.Sapma	1.06E+11	8.06E+09	1.25E+06	1.41E+10	1.04E+09	1.20E+08	1.24E+03	5.92E+02	4.87E+02	1.30E+00	3.08E+00	1.11E+00
	F17			**F18**			**F19**			**F20**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	5.71E+06	3.76E+06	1.14E+06	3.64E+11	4.50E+11	3.11E+07	1.08E+07	9.64E+06	3.57E+06	5.52E+11	5.31E+11	2.82E+07
Ortanca	8.36E+06	3.99E+06	1.21E+06	2.30E+12	5.04E+11	4.14E+07	1.41E+07	9.80E+07	3.87E+06	2.56E+12	6.12E+11	3.85E+07
En kötü	1.15E+07	4.58E+06	1.32E+06	2.58E+12	5.79E+11	4.90E+07	1.91E+07	1.15E+08	4.19E+06	2.86E+12	6.89E+11	4.45E+07
Ortalama	8.42E+06	4.03E+06	**1.22E+06**	2.22E+12	5.05E+11	**4.08E+07**	1.43E+07	9.30E+07	**3.88E+06**	2.49E+12	6.07E+11	**3.82E+07**
Std.Sapma	1.31E+06	2.16E+05	4.94E+04	4.08E+11	2.80E+10	4.82E+06	2.20E+06	2.59E+07	1.59E+05	4.35E+11	3.62E+10	3.72E+06
		BA			HBA1			HBA2				
Friedman		2.75			1.90			**1.35**				

Çizelge 5.18. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları (FEs=6.00E+05)

	F1			F2			F3			F4		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	1.51E+10	6.80E+07	1.30E+05	1.35E+04	4.37E+03	7.03E+02	2.11E+01	8.02E+00	4.94E+00	5.28E+11	9.9E+11	1.10E+12
Ortanca	4.56E+10	8.54E+07	2.63E+05	2.37E+04	4.77E+03	7.80E+02	2.14E+01	8.49E+00	5.48E+00	1.24E+12	1.41E+12	3.25E+12
En kötü	5.35E+10	1.12E+08	6.22E+05	2.46E+04	5.20E+03	8.17E+02	2.14E+01	8.88E+00	6.16E+00	4.18E+12	2.75E+12	5.52E+12
Ortalama	4.47E+10	8.74E+07	**2.89E+05**	2.14E+04	4.77E+03	**7.76E+02**	2.14E+01	8.48E+00	**5.49E+00**	**1.38E+12**	1.55E+12	3.34E+12
Std.Sapma	7.96E+09	1.18E+07	1.25E+05	4.28E+04	2.02E+02	3.04E+01	6.72E-02	2.44E-01	2.82E-01	7.68E+11	4.35E+11	1.32E+12
	F5			**F6**			**F7**			**F8**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.83E+08	5.97E+07	2.95E+08	2.05E+07	2.02E+01	1.90E+07	5.86E+06	4.61E+06	2.15E+07	3.41E+07	4.13E+07	5.65E+07
Ortanca	3.76E+08	8.35E+07	3.75E+08	2.08E+07	2.06E+01	1.96E+07	8.71E+08	5.68E+06	3.63E+07	3.51E+07	4.54E+07	2.36E+08
En kötü	4.90E+08	1.15E+08	5.47E+08	2.10E+07	2.08E+01	2.01E+07	1.50E+10	6.61E+06	4.64E+07	5.28E+08	1.18E+08	1.30E+09
Ortalama	3.74E+08	**8.24E+07**	3.89E+08	2.08E+07	**2.06E+01**	1.96E+07	2.73E+09	**5.63E+06**	3.44E+07	9.58E+07	**6.22E+07**	3.51E+08
Std.Sapma	6.04E+07	1.78E+07	7.04E+07	1.56E+05	1.42E-01	2.77E+05	4.01E+09	5.34E+05	7.33E+06	1.24E+08	2.91E+07	3.26E+08
	F9			**F10**			**F11**			**F12**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	1.76E+10	1.03E+09	4.55E+08	1.36E+04	6.99E+03	5.06E+03	2.32E+02	8.73E+01	2.08E+02	2.63E+06	6.08E+05	1.34E+05
Ortanca	5.02E+10	1.16E+09	6.67E+08	2.37E+04	7.35E+03	5.65E+03	2.35E+02	1.04E+02	2.09E+02	4.39E+06	7.25E+05	1.55E+05
En kötü	6.78E+10	1.32E+09	7.85E+08	2.46E+04	7.74E+03	8.29E+03	2.35E+02	1.25E+02	2.13E+02	4.95E+06	8.28E+05	1.81E+05
Ortalama	5.07E+10	1.17E+09	**6.55E+08**	2.19E+04	7.36E+03	**5.86E+03**	2.35E+02	**1.03E+02**	2.10E+02	4.28E+06	7.28E+05	**1.55E+05**
Std.Sapma	9.59E+09	7.52E+07	7.64E+07	3.80E+03	2.15E+02	7.26E+02	6.55E-01	8.95E+00	1.42E+00	4.36E+05	6.32E+05	1.10E+04
	F13			**F14**			**F15**			**F16**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.57E+10	7.19E+06	6.35E+03	1.83E+10	2.36E+09	1.05E+09	1.44E+04	7.474E+03	9.36E+03	4.22E+02	2.83E+02	4.02E+02
Ortanca	2.54E+11	1.01E+07	9.74E+03	5.34E+10	2.63E+09	1.15E+09	2.42E+04	8.00E+03	9.97E+03	4.28E+02	2.96E+02	4.04E+02
En kötü	3.05E+11	1.27E+07	2.16E+04	6.75E+10	2.86E+09	1.27E+09	2.50E+04	8.29E+03	1.11E+04	4.29E+02	3.67E+02	4.12E+02
Ortalama	2.48E+11	1.03E+07	**1.17E+04**	5.38E+10	2.63E+09	**1.15E+09**	2.29E+04	**7.93E+03**	1.01E+04	4.28E+02	**3.03E+02**	4.05E+02
Std.Sapma	5.50E+10	1.37E+06	4.75E+03	9.46E+09	1.20E+08	5.87E+07	3.46E+03	2.38E+02	5.21E+02	1.28E+00	1.80E+01	2.36E+00
	F17			**F18**			**F19**			**F20**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	3.82E+06	1.13E+06	5.63E+05	1.83E+11	2.92E+09	3.37E+04	6.37E+06	3.47E+05	2.19E+06	3.00E+11	4.41E+09	1.52E+04
Ortanca	5.60E+06	1.21E+06	6.19E+05	1.44E+12	3.93E+09	5.92E+04	8.83E+06	3.41E+06	2.48E+06	1.65E+12	5.32E+09	2.30E+04
En kötü	7.68E+06	1.45E+06	6.63E+05	1.65E+12	4.88E+09	9.38E+04	1.16E+07	3.82E+06	2.68E+06	1.89E+12	7.06E+09	3.08E+04
Ortalama	5.73E+06	1.21E+06	**6.17E+05**	1.41E+12	3.87E+09	**6.01E+04**	8.80E+06	3.30E+06	**2.48E+06**	1.62E+12	5.28E+09	**2.30E+04**
Std.Sapma	8.26E+05	6.99E+04	2.66E+04	2.74E+11	4.67E+08	1.53E+04	1.21E+06	6.58E+05	1.28E+05	2.98E+11	5.37E+08	3.84E+03
	BA			HBA1			HBA2					
Friedman	2.80			1.65			**1.55**					

Çizelge 5.19. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları (FEs=3.00E+06)

	F1			F2			F3			F4		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	7.89E+09	1.87E+02	1.64E-11	1.35E+04	3.67E+03	3.89E+01	1.98E+01	5.32E+00	1.42E-06	2.91E+11	9.85E+10	7.80E+09
Ortanca	2.74E+10	1.23E+03	1.72E-10	1.44E+04	4.24E+03	4.85E+01	2.04E+01	5.91E+00	3.34E-06	5.14E+11	1.87E+11	1.82E+10
En kötü	3.23E+10	4.64E+04	6.05E-08	2.46E+04	4.66E+03	5.30E+01	2.14E+01	6.62E+00	6.43E-06	8.87E+11	3.63E+11	4.07E+10
Ortalama	2.63E+10	3.91E+03	**4.81E-09**	1.73E+04	4.21E+03	**4.74E+01**	2.06E+01	5.93E+00	**3.54E-06**	5.18E+11	1.97E+11	**2.15E+10**
Std.Sapma	4.93E+09	9.11E+03	1.46E-08	4.47E+03	2.69E+02	3.58E+00	7.83E-01	3.02E-01	1.21E-06	1.07E+11	6.86E+10	9.19E+09
	F5			**F6**			**F7**			**F8**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.83E+08	5.97E+07	2.78E+08	1.94E+07	1.04E+01	1.88E+07	5.73E+06	1.45E+04	6.09E+04	5.79E+06	1.15E+05	5.93E+04
Ortanca	3.76E+08	8.35E+07	3.71E+08	1.97E+07	1.23E+01	1.94E+07	6.05E+06	6.85E+04	1.64E+05	6.30E+06	2.97E+05	3.06E+05
En kötü	4.90E+08	1.15E+08	5.43E+08	2.08E+07	1.39E+01	1.98E+07	5.41E+08	1.82E+05	2.97E+05	2.21E+08	1.94E+07	7.24E+07
Ortalama	3.74E+08	**8.23E+07**	3.81E+08	2.01E+07	**1.24E+01**	1.93E+07	4.11E+07	**7.30E+04**	1.74E+05	3.73E+07	**1.57E+06**	1.06E+07
Std.Sapma	6.04E+07	1.77E+07	7.12E+07	5.56E+05	7.55E-01	2.53E+05	1.25E+08	3.81E+04	5.32E+04	7.21E+07	4.02E+06	1.93E+07
	F9			**F10**			**F11**			**F12**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	8.82E+09	8.74E+07	2.63E+06	1.35E+04	6.48E+03	4.21E+03	2.15E+02	6.25E+01	1.95E+02	2.04E+06	2.06E+04	2.19E-01
Ortanca	2.99E+10	1.05E+08	3.52E+06	1.47E+04	6.82E+03	4.78E+03	2.34E+02	9.28E+01	1.96E+02	3.53E+06	2.55E+04	2.65E-01
En kötü	4.13E+10	1.21E+08	7.11E+06	2.45E+04	7.27E+03	7.48E+03	2.35E+02	1.14E+02	2.00E+02	4.01E+06	2.97E+04	3.16E-01
Ortalama	2.96E+10	1.03E+08	**3.78E+06**	1.76E+04	6.81E+03	**4.95E+03**	2.27E+02	**9.13E+01**	1.96E+02	3.46E+06	2.56E+04	**2.74E-01**
Std.Sapma	6.21E+09	8.78E+06	1.05E+06	4.61E+03	2.25E+02	7.45E+02	8.74E+00	1.12E+01	1.80E+00	3.68E+05	2.20E+03	2.37E-02
	F13			**F14**			**F15**			**F16**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	4.8E+09	2.00E+03	4.27E+02	9.23E+09	3.43E+08	7.48E+06	9.42E+03	6.84E+03	8.74E+03	3.90E+02	2.42E+02	3.90E+02
Ortanca	1.02E+11	3.28E+03	8.36E+02	3.07E+10	3.74E+08	9.64E+06	1.49E+04	7.67E+03	9.38E+03	3.96E+02	2.68E+02	3.91E+02
En kötü	1.33E+11	5.20E+03	2.97E+03	3.94E+10	4.37E+08	1.18E+07	2.49E+04	7.30E+03	1.06E+04	4.29E+02	3.35E+02	3.99E+02
Ortalama	1.00E+11	3.26E+03	**1.11E+03**	3.10E+10	3.77E+08	**9.59E+06**	1.90E+04	**7.33E+03**	9.45E+03	4.09E+02	**2.73E+02**	3.92E+02
Std.Sapma	2.53E+10	7.43E+02	6.47E+02	5.93E+09	2.13E+07	1.07E+06	5.35E+03	2.31E+02	4.76E+02	1.61E+01	1.97E+01	2.20E+00
	F17			**F18**			**F19**			**F20**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.69E+06	1.30E+05	1.15E+00	8.15E+10	6.56E+03	8.36E+02	4.06E+06	9.13E+04	3.51E+04	1.50E+11	4.94E+03	2.46E+02
Ortanca	4.00E+06	1.46E+05	1.34E+00	8.79E+11	8.37E+03	1.05E+04	5.70E+06	9.57E+05	4.71E+04	1.04E+12	5.63E+03	6.60E+02
En kötü	5.44E+06	1.63E+05	1.63E+00	1.01E+12	2.02E+04	3.41E+04	7.51E+06	1.04E+06	6.60E+04	1.22E+12	6.05E+03	1.18E+03
Ortalama	4.08E+06	1.48E+05	**1.33E+00**	8.58E+11	**8.87E+03**	1.38E+04	5.86E+06	8.95E+05	**4.72E+04**	1.01E+12	5.62E+03	**6.77E+02**
Std.Sapma	5.57E+05	8.12E+03	1.27E-01	1.77E+11	2.95E+03	1.07E+04	7.57E+05	2.44E+05	8.31E+03	1.98E+11	3.03E+02	2.26E+02
		BA			HBA1			HBA2				
Friedman		2.95			1.60			**1.45**				

Çizelge 5.20-5.22'de algoritmaların ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Bu test işleminde, 3.00E+06 FEs için algoritmaların her bir çevrim sonrasında elde ettiği ortalama değerler ikili olarak karşılaştırılmıştır.

Çizelge 5.20'de, CEC2010 kıyaslama fonksiyonlarında, HBA1 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, tüm fonksiyonlarda HBA1'in BA'dan anlamsal olarak daha iyi olduğu görülmektedir.

Çizelge 5.21'de, CEC2010 kıyaslama fonksiyonlarında, HBA2 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, bir fonksiyonda (F5), algoritmalar arasında anlamlı bir fark bulunamamıştır. Kalan on dokuz fonksiyonda ise HBA2'nin BA'dan anlamsal olarak daha iyi olduğu tespit edilmiştir.

Çizelge 5.22'de ise, CEC2010 kıyaslama fonksiyonlarında, HBA2 ve HBA1 algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, iki fonksiyonda (F8, F18), algoritmalar arasında anlamlı bir fark bulunamamıştır. Altı fonksiyonda (F5, F6, F7, F11, F15, F16), HBA1 algoritması HBA2'den daha iyi performans göstermiştir ve aralarında anlamlı bir fark mevcuttur. Kalan on iki fonksiyonda ise HBA2'nin HBA1'den anlamsal olarak daha iyi olduğu bulunmuştur.

Çizelge 5.20. CEC2010 kıyaslama fonksiyonları için HBA1 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	25	0	0	1.20E-05	+	**F11**	25	0	0	1.20E-05	+
F2	25	0	0	1.20E-05	+	**F12**	25	0	0	1.20E-05	+
F3	25	0	0	1.20E-05	+	**F13**	25	0	0	1.20E-05	+
F4	25	0	0	1.20E-05	+	**F14**	25	0	0	1.20E-05	+
F5	25	0	0	1.20E-05	+	**F15**	25	0	0	1.20E-05	+
F6	25	0	0	1.20E-05	+	**F16**	25	0	0	1.20E-05	+
F7	25	0	0	1.20E-05	+	**F17**	25	0	0	1.20E-05	+
F8	24	1	0	1.40E-04	+	**F18**	25	0	0	1.20E-05	+
F9	25	0	0	1.20E-05	+	**F19**	25	0	0	1.20E-05	+
F10	25	0	0	1.20E-05	+	**F20**	25	0	0	1.20E-05	+

Çizelge 5.21. CEC2010 kıyaslama fonksiyonları için HBA2 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	25	0	0	1.20E-05	+	**F11**	25	0	0	1.20E-05	+
F2	25	0	0	1.20E-05	+	**F12**	25	0	0	1.20E-05	+
F3	25	0	0	1.20E-05	+	**F13**	25	0	0	1.20E-05	+
F4	25	0	0	1.20E-05	+	**F14**	25	0	0	1.20E-05	+
F5	14	11	0	9.68E-01	≈	**F15**	24	1	0	1.40E-05	+
F6	24	1	0	2.90E-05	+	**F16**	23	2	0	6.50E-05	+
F7	25	0	0	1.20E-05	+	**F17**	25	0	0	1.20E-05	+
F8	16	9	0	3.24E-02	+	**F18**	25	0	0	1.20E-05	+
F9	25	0	0	1.20E-05	+	**F19**	25	0	0	1.20E-05	+
F10	25	0	0	1.20E-05	+	**F20**	25	0	0	1.20E-05	+

Çizelge 5.22. CEC2010 kıyaslama fonksiyonları için HBA2 ve HBA1 algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

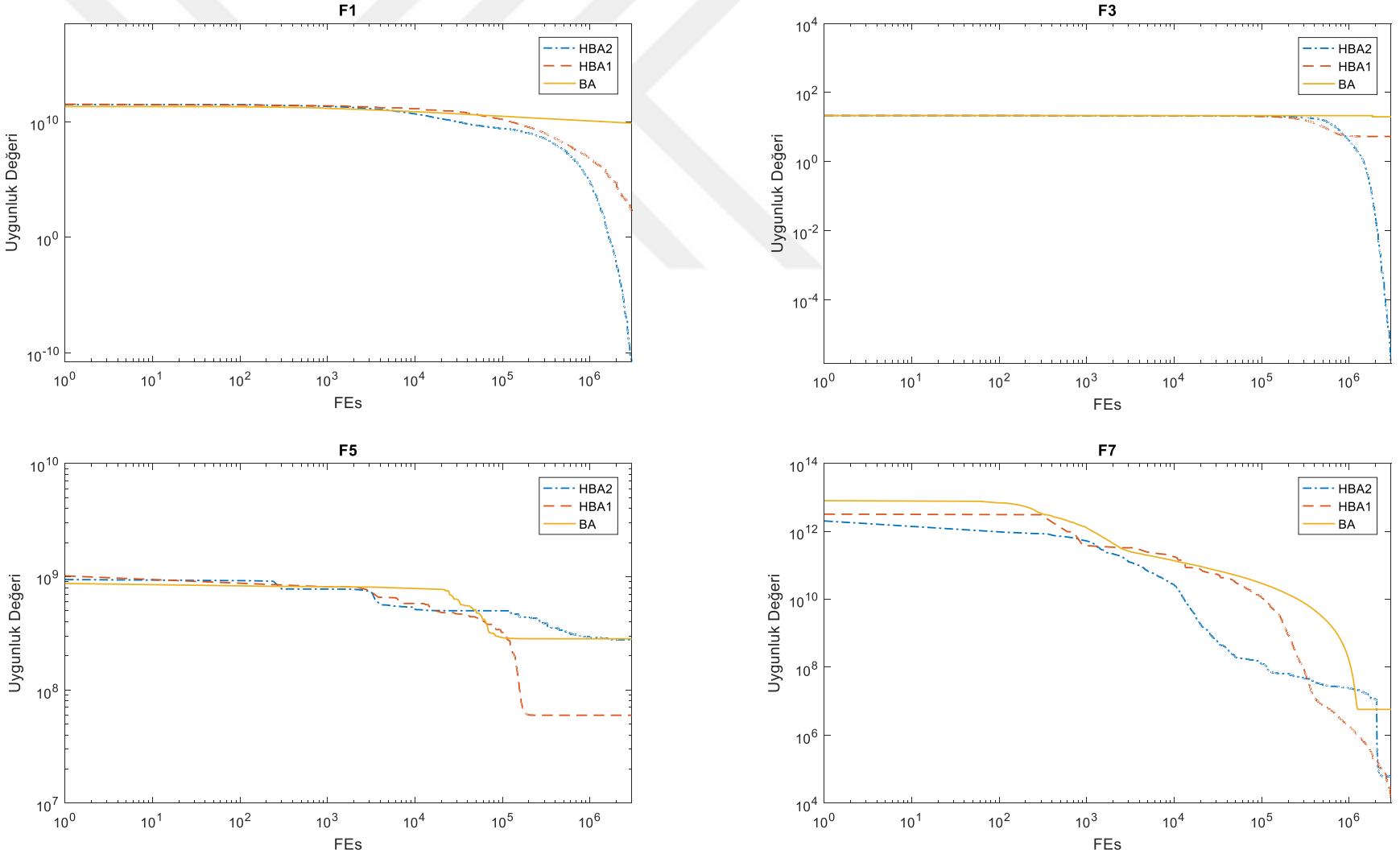
No	Daha iyi	Daha kötü	Eşit	p değeri	Karar	No	Daha iyi	Daha kötü	Eşit	p değeri	Karar
F1	25	0	0	1.20E-05	+	**F11**	0	25	0	1.20E-05	-
F2	25	0	0	1.20E-05	+	**F12**	25	0	0	1.20E-05	+
F3	25	0	0	1.20E-05	+	**F13**	24	1	0	1.80E-05	+
F4	25	0	0	1.20E-05	+	**F14**	25	0	0	1.20E-05	+
F5	0	25	0	1.20E-05	-	**F15**	0	25	0	1.20E-05	-
F6	0	25	0	1.20E-05	-	**F16**	0	25	0	1.20E-05	-
F7	1	24	0	2.50E-05	-	**F17**	25	0	0	1.20E-05	+
F8	13	12	0	4.12E-01	≈	**F18**	12	13	0	1.04E-01	≈
F9	25	0	0	1.20E-05	+	**F19**	25	0	0	1.20E-05	+
F10	23	2	0	1.80E-05	+	**F20**	25	0	0	1.20E-05	+

Şekil 5.2'de, CEC2010 kıyaslama fonksiyonlarının farklı gruplarından seçilen on adet fonksiyon için BA, HBA1 ve HBA2 algoritmalarından elde edilen en iyi değerlere ait yakınsama grafikleri verilmiştir. Grafikler incelendiğinde şu sonuçlar elde edilebilir. F1 fonksiyonunda, en iyi uygunluk değeri HBA2 algoritmasında elde edilmiştir. Algoritmalar yaklaşık 10000 FEs değerine kadar benzer değerler üretmiş ve yakınsama hızları oldukça durağan kalmıştır. İlerleyen süreçte, HBA1 ve HBA2 algoritmaların yakınsama hızı artmıştır. En hızlı yakınsamayı ve en iyi uygunluk değerli çözümü HBA2 algoritması elde etmiştir. F3 fonksiyonunda, yaklaşık 6.00E+05 FEs değerine kadar algoritmalar çözümlerini geliştirememişlerdir. Daha sonrasında, hibrit

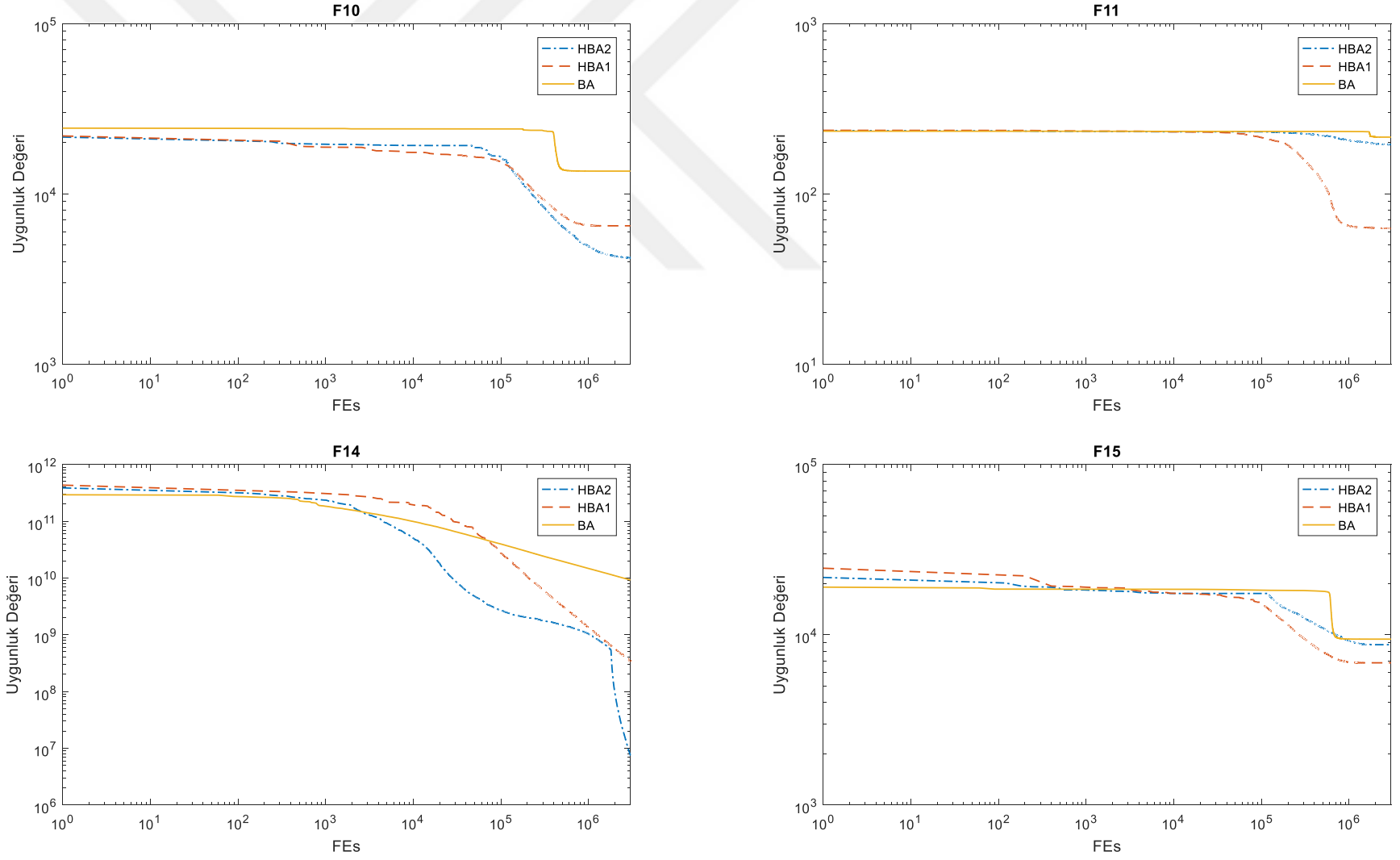
algoritmaların yakınsama hızları bir miktar artmıştır. HBA2, HBA1 algoritmasına göre daha iyi uygunluk değerine sahip bir çözüm üretmiştir ve daha hızlı yakınsamıştır. F5 fonksiyonunda, en iyi uygunluk değeri HBA1 algoritmasından elde edilmiştir. Zaman zaman durağanlaşan HBA1 yakınsama grafiği yaklaşık 5.00E+05 FEs sonrasında çözümü geliştirememiştir. HBA2 ve BA algoritmaları ise bu fonksiyonda performans olarak HBA1'in gerisinde kalmışlardır. F7 fonksiyonunda, HBA1 algoritması uygunluk değeri en iyi olan çözümü üretmiştir. BA ve HBA2 algoritmaları iyi bir yakınsama karakteristiği göstemiş olsada son iterasyonlarda çözümlerini geliştirememiş ve HBA1'in gerisinde kalmışlardır. F10 fonksiyonunda, BA algoritması, arama sürecinin sonuna doğru, bir miktar çözümünü geliştirmiş ve yakınsamış olsa da, genelde durağan bir yakınsama karakteristiği göstermiştir. HBA1 ve HBA2 algoritmaları ise benzer bir yakınsama karakteristiği göstermişlerdir. Ancak son iterasyonlara doğru HBA2'nin yakınsaması, HBA1'e göre daha hızlı olmuştur ve daha iyi uygunluk değerli bir çözüm üretmiştir. F11 fonksiyonunda, üç algoritmada 1.00E+05 FEs değerine kadar durağan bir yakınsama karakteristiği göstermiştir. Sonrasında, HBA1 algoritması diğer algoritmalara göre daha hızlı yakınsamış ve daha iyi uygunluk değerli bir çözüm üretmiştir. F14 fonksiyonunda, yaklaşık 1000 FEs değerine kadar algoritmalar çözümlerini pek fazla geliştirememiş ve yakınsama grafikleri durağan bir karakteristik göstermiştir. Sonrasında, algoritmalar daha iyi çözümlere doğru yakınsamışlardır. En hızlı yakınsama ve en iyi uygunluk değerli çözüm HBA2 algoritmasından, en yavaş yakınsama ve diğerlerine kıyasla en kötü uygunluk değerli çözüm ise BA algoritmasından elde edilmiştir. F15 fonksiyonunda, BA algoritması 1.00E+06 FEs değerine yakın bir FEs değerine kadar çözümünü geliştirememiş, ardından bir miktar çözümünü geliştirmiş ve tekrar yakınsama grafiği durağanlaşmıştır. HBA1 ve HBA2 algoritmaları ise, benzer bir yakınsama karakteristiği sergilemişlerdir. Ancak, sonuçta en iyi uygunluk değerli çözüm HBA1'den elde edilmiştir. F17 fonksiyonunda, algoritmaların zaman zaman durağanlaşan yakınsama karakteristiklerine rağmen, en iyi çözüm HBA2 algoritmasından elde edilmiştir. HBA2'ye benzer şekilde çoğunlukla durağan ve yavaş bir yakınsama karakteristiği gösteren diğer algoritmalardan, HBA1 daha kötü bir başlangıç çözümüyle başlamasına rağmen, araştırma sürecinin bitiminde, BA'dan daha iyi bir çözüm üretmiştir. En kötü uygunluk değerli çözüm ise BA'dan elde edilmiştir. Son olarak, F20 fonksiyonunda, BA algoritması çözümünü pek fazla geliştirememiştir ve yakınsama hızı oldukça yavaş kalmıştır. Kalan algoritmalar incelendiğinde ise HBA2 algoritmasının daha hızlı yakınsadığı ve daha iyi uygunluk değerli bir çözüm ürettiği görülmüştür.

Şekil 5.2'de verilen yakınsama grafikleri ve Çizelge 5.17-19'da verilen sonuçlar doğrultusunda, algoritmaların gelişim durumlarının artan FEs sayısından nasıl etkilendiği değerlendirilebilir. Buna göre, BA algoritmasında birkaç fonksiyon (F4,F7,F8,F18,F19) hariç genel olarak artan FEs sayısı sonuçlarda dikkate değer iyileştirmelere neden olmamıştır. HBA1 algoritmasında fonksiyonların tümünde artan FEs sayısına bağlı olarak sonuçlar iyileşmiştir. Sadece F16 fonksiyonunda çözümün iyileşme miktarı diğerlerine kıyasla daha az olmuştur. HBA2 algoritmasında ise tüm fonksiyonlarda, artan FEs değerine bağlı olarak çözümler iyileşmiştir. Ancak F5, F6, F11 ve F16 fonksiyonlarında iyileşme miktarı, diğer fonksiyonlara kıyasla daha az olmuştur. HBA2 sonuçlarına göre, fonksiyonların yarısında 6.00E+05 FEs sonrasında çözümlerdeki iyileşme miktarının daha fazla olduğu tespit edilmiştir.

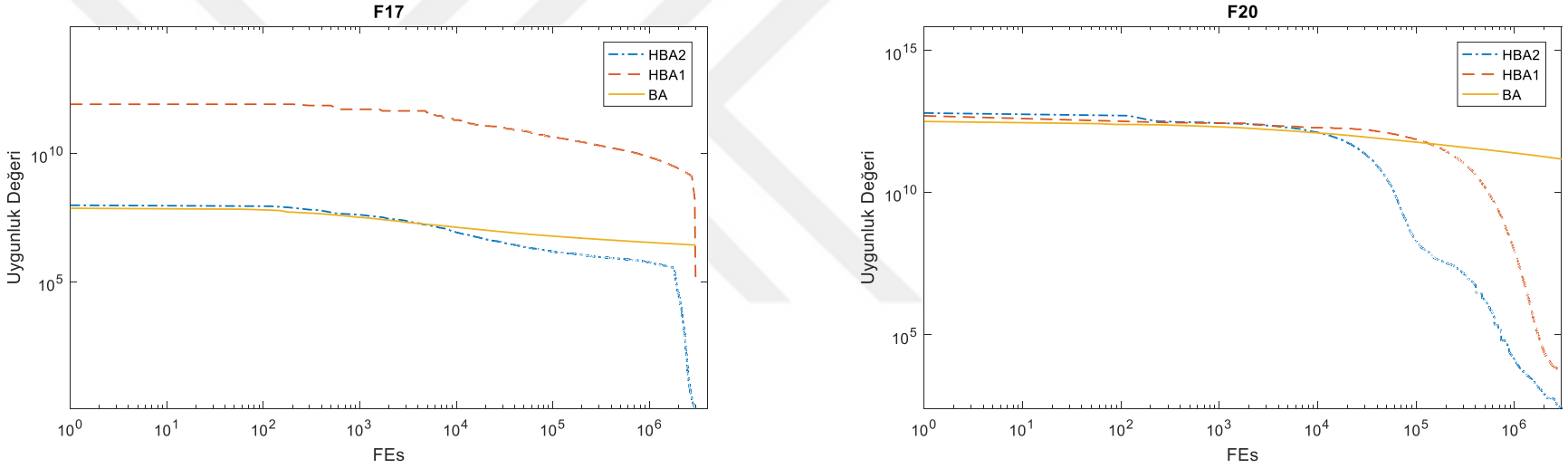
Bu sonuçlara göre, BA algoritmasında artan FEs sayısının çözümün gelişmesine olan etkisinin daha az olduğu, HBA1 ve HBA2 algoritmalarında ise artan FEs sayısının çözümün gelişmesine katkı sağladığı söylenebilir.



Şekil 5.2.a. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri



Şekil 5.2.b. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri



Şekil 5.2.c. CEC2010 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 için yakınsama grafikleri

Hibrit algoritmalar, literatürden seçilen algoritmalarla da karşılaştırılmıştır. Karşılaştırmada kullanılan algoritmalara ait bilgiler Çizelge 5.23'de verilmiştir. Literatürdeki algoritmalara göre, önerilen hibrit algoritmaların performansını belirlemek için yapılan karşılaştırmada, tüm algoritmalar için FEs değeri 3.00E+06 ve genel çevrim sayısı 25 olarak alınmıştır. Elde edilen ortalama değerlere göre yapılan karşılaştırmanın sonuçları Çizelge 5.24'de verilmiştir.

Çizelge 5.23. Karşılaştırma yapılan LGSO algoritmalarına ait bilgiler

Algoritma	Makale Adı	Referans
AHDE-csw	Adaptive hybrid differential evolution with circular sliding window for large scale optimization	(Ge ve ark., 2016)
RSQPSO	A novel quantum-behaved particle swarm optimization with random selection for large scale optimization	(Fang ve ark., 2017)
ISCA	Solving high-dimensional global optimization problems using an improved sine cosine algorithm	(Long ve ark., 2019)
µDSDE	Large scale continuous global optimization based on micro differential evolution with local directional search	(Yildiz ve Topal, 2019)
aEUS	Adaptive pattern search for large-scale optimization	(Gardeux ve ark., 2017)
PSO-CSC	Particle swarm optimization with convergence speed controller for large-scale numerical optimization	(Huang ve ark., 2019)
RBLSO	Ranking-based biased learning swarm optimizer for large-scale optimization	(Deng ve ark., 2019)
SBA	Social-Based Algorithm	(Ramezani ve Lotfi, 2013)
ITGO	ITGO: Invasive tumor growth optimization algorithm	(Tang ve ark., 2015)

Çizelge 5.24'te verilen karşılaştırma sonuçları incelendiğinde, HBA2 ve RBLSO algoritmalarının beş tane fonksiyonda, PSO-CSC algoritması dört tane fonksiyonda, aEUS algoritması üç fonksiyonda, RSQPSO algoritması iki tane fonksiyonda, ISCA algoritmasının ise bir tane fonksiyonda en iyi ortalama uygunluk değerini ürettiği görülmektedir. Bunun yanında, HBA1, AHDE-csw, µDSDE, SBA ve ITGO algoritmaları ise hiçbir fonksiyonda en iyi ortalama uygunluk değeri üretememiştir. Algoritmalar için Friedman testine göre yapılan sıralamada 3.23 ortalama sıra değeri ile PSO-CSC algoritması birinci olmuştur. Önerilen algoritmalardan HBA1, 5.65 ortalama sıralama değeri ile altıncı, HBA2 ise 4.45 ortalama sıralama değeri ile üçüncü sırada yer almıştır.

Çizelge 5.24. CEC2010 kıyaslama fonksiyonlarında HBA2 ve HBA1 algoritmalarının literatürdeki LSGO algoritmaları ile karşılaştırma sonuçları

Fonksiyon	Algoritmalar										
	HBA1	**HBA2**	AHDE-csw	RSQPSO	ISCA	µDSDE	aEUS	PSO-CSC	RBLSO	SBA	ITGO
F1	3.91E+03	4.81E-09	1.81E-20	**8.66E-32**	1.55E+11	2.05E+04	8.32E-24	3.16E−04	1.55E −20	3.49E+04	4.11E+06
F2	4.21E+03	4.74E+01	3.57E-06	1.18E+01	1.64E+00	1.47E+02	**0.00E+00**	5.53E+00	9.82E + 02	2.62E+00	5.02E+03
F3	5.93E+00	3.54E-06	3.73E-09	**1.27E-14**	3.10E-01	2.45E+00	1.90E-12	4.53E−06	9.91E −14	1.62E-02	1.92E+01
F4	1.97E+11	**2.15E+10**	2.33E+12	7.09E+11	1.37E+12	3.16E+11	2.84E+11	9.79E+10	7.40E + 11	3.10E+12	2.22E+12
F5	8.23E+07	3.81E+08	3.88E+08	5.39E+08	3.27E+08	1.11E+08	7.18E+07	**5.52E+07**	2.32E + 08	3.44E+08	1.85E+08
F6	1.24E+01	1.93E+07	1.97E+07	1.94E+07	1.29E+05	1.50E+07	1.99E+07	4.45E−02	**8.43E −09**	5.79E+06	2.73E+06
F7	7.30E+04	1.74E+05	2.01E+06	7.50E+00	3.71E+10	5.53E+05	2.73E+03	**1.87E−02**	1.37E + 03	2.13E+09	1.70E+06
F8	1.57E+06	1.06E+07	9.09E+05	8.78E+07	1.59E+13	4.08E+07	1.29E+08	**9.24E+00**	2.67E + 07	5.25E+07	1.33E+08
F9	1.03E+08	**3.78E+06**	7.06E+07	1.75E+07	1.66E+11	7.16E+08	7.57E+06	3.74E+07	3.61E + 07	3.00E+08	4.02E+08
F10	6.81E+03	4.95E+03	5.84E+03	5.33E+03	2.96E+03	4.63E+03	7.15E+03	4.38E+03	**1.00E + 03**	7.28E+03	6.33E+03
F11	9.13E+01	1.96E+02	2.01E+02	1.98E+02	1.89E+02	1.98E+02	1.99E+02	1.43E+02	**5.84E −13**	2.01E+02	2.75E+05
F12	2.56E+04	**2.74E-01**	3.99E+04	1.19E+02	1.33E+06	2.59E+05	3.28E-01	1.73E+02	3.72E + 04	2.46E+05	2.75E+05
F13	3.26E+03	1.11E+03	1.64E+03	1.06E+03	6.30E+10	1.10E+03	1.09E+03	**9.94E+02**	1.05E + 03	1.67E+04	4.07E+04
F14	3.77E+08	**9.59E+06**	1.71E+08	5.58E+07	2.29E+08	1.85E+09	1.71E+07	1.19E+08	1.38E + 08	1.09E+09	1.09E+09
F15	7.33E+03	9.45E+03	1.09E+04	1.34E+04	**1.64E+03**	8.79E+03	1.42E+04	5.35E+03	1.03E + 04	1.47E+04	7.64E+03
F16	2.73E+02	3.92E+02	3.98E+02	9.37E+02	3.18E+02	3.90E+02	3.98E+02	3.70E+02	**8.94E −13**	4.01E+02	3.87E+02
F17	1.48E+05	**1.33E+00**	2.00E+05	2.31E+03	3.27E+06	1.03E+06	3.98E+00	6.39E+03	1.77E + 05	5.13E+05	7.51E+05
F18	8.87E+03	1.38E+04	4.77E+03	2.28E+03	1.41E+04	3.01E+04	2.97E+03	2.15E+03	**1.93E + 03**	4.78E+04	9.87E+04
F19	8.95E+05	4.72E+04	5.84E+05	3.08E+06	5.90E+06	4.56E+06	**9.44E+03**	5.89E+05	7.44E + 06	2.99E+06	2.32E+06
F20	5.62E+03	6.77E+02	1.21E+03	1.13E+03	1.59E+11	5.29E+03	**4.51E+02**	1.21E+03	1.34E + 03	3.73E+03	5.19E+04
Friedman	5.65	4.45	6.48	5.28	7.65	7.33	4.53	**3.23**	4.35	8.50	8.58
Sıra	6	3	7	5	9	8	4	**1**	2	10	11

Çizelge 5.25 ve 5.26'da ise sırasıyla HBA1 ve HBA2 algoritmalarının karşılaştırıldıkları diğer algoritmalarla olan ikili Wilcoxon işaretli sıra testi sonuçları yer almaktadır. Çizelge 5.25'e göre, HBA1 algoritmasının ISCA ,µDSDE, SBA ve ITGO algoritmalarından performansı daha iyidir ve aralarında anlamlı bir fark mevcuttur. PSO-CSC algoritmasına göre performansı kötüdür ve aralarında anlamlı bir fark bulunmaktadır. Kalan beş algoritma ile arasında anlamlı bir fark bulunamamıştır ve benzer performans göstermişlerdir. Çizelge 5.26 incelendiğinde ise, HBA2 algoritmasının, AHDE-csw, RSQPSO, ISCA,µDSDE, SBA ve ITGO algoritmalarından daha iyi performans gösterdiği ve aralarında anlamlı bir fark olduğu tespit edilmiştir. HBA2, kalan dört algoritmayla benzer performans göstermiştir ve aralarında anlamlı bir fark bulunamamıştır. Yapılan bu test işlemleri sonucunda, önerilen hibrit algoritmaların literaratüre kıyasla yarışmacı, kabul edilebilir ve başarılı sonuçlar ürettiği görülmüştür.

Çizelge 5.25. CEC2010 kıyaslama fonksiyonlarında HBA1 ve LSGO algoritmalarının ikili Wilcoxon işaret testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA1-HBA2	8	12	0	3.50E-01	≈
HBA1-AHDE-csw	9	11	0	9.70E-01	≈
HBA1- RSQPSO	8	12	0	8.23E-01	≈
HBA1- ISCA	15	5	0	4.55E-03	+
HBA1- µDSDE	15	5	0	3.18E-03	+
HBA1- aEUS	7	13	0	2.32E-01	≈
HBA1- PSO-CSC	2	18	0	2.54E-04	-
HBA1- RBLSO	7	13	0	8.52E-01	≈
HBA1- SBA	17	3	0	5.17E-04	+
HBA1- ITGO	19	1	0	1.63E-04	+

Çizelge 5.26. CEC2010 kıyaslama fonksiyonlarında HBA2 ve LSGO algoritmalarının ikili Wilcoxon işaret testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA2-HBA1	12	8	0	3.51E-01	≈
HBA2-AHDE-csw	15	5	0	8.96E-03	+
HBA2- RSQPSO	14	6	0	1.23E-02	+
HBA2- ISCA	13	7	0	2.50E-02	+
HBA2- µDSDE	14	6	0	3.82E-02	+
HBA2- aEUS	11	9	0	4.55E-01	≈
HBA2- PSO-CSC	9	11	0	7.94E-01	≈
HBA2- RBLSO	10	10	0	3.70E-01	≈
HBA2- SBA	17	3	0	7.18E-03	+
HBA2- ITGO	16	4	0	1.11E-02	+

5.4. CEC2011 Gerçek Dünya Problemleri Üzerinde Yapılan Test İşlemi Sonuçları

Bu tez kapsamında önerilen HBA1 ve HBA2 algoritmalarının performansı, üçüncü olarak CEC2011 (Congress of Evolutionary Computation 2011) gerçek dünya problemleri (Das ve Suganthan, 2010) üzerinde incelenmiştir. Toplam 22 adet problemden oluşan bu problem kümesinden büyük ölçekli olan yedi tanesi seçilmiştir. Seçilen bu problemler ve özellikleri Çizelge 5.27'de verilmiştir. CEC2011 kuralları gereği, bu problemlerde genel çevrim sayısı 25, FEs 1.50E+05 olarak alınmaktadır.

Çizelge 5.27. CEC2011 gerçek dünya problemleri

Problem	Boyut	Kısıtlamalar	Açıklama
P9	126	Lineer eşitlik kısıtlamaları	Büyük ölçekli iletim fiyatlandırması
P11.1	120	Eşitsizlik kısıtlamaları	Dinamik Ekonomik Gönderim- Örnek 1
P11.2	216	Eşitsizlik kısıtlamaları	Dinamik Ekonomik Gönderim - Örnek 2
P11.7	140	Eşitsizlik kısıtlamaları	Statik Ekonomik Yük Gönderimi- Örnek5
P11.8	96	Eşitsizlik kısıtlamaları	Hidrotermal Planlama - Örnek 1
P11.9	96	Eşitsizlik kısıtlamaları	Hidrotermal Planlama - Örnek 2
P11.10	96	Eşitsizlik kısıtlamaları	Hidrotermal Planlama - Örnek 3

Çizelge 5.28, seçilen büyük boyutlu CEC2011 gerçek dünya problemlerinde standart BA, HBA1 ve HBA2 algoritmalarının, 1.50E+05 FEs için elde ettikleri sonuçları karşılaştırmalı olarak göstermektedir. Her problem için en iyi ortalama uygunluk değeri koyu fontla belirtilmiştir. Çizelgedeki sonuçlar incelendiğinde, HBA1'in bir problemde (P9), HBA2'nin ise kalan altı problemde en iyi ortalama uygunluk değerini ürettiği görülmektedir. Friedman testi sıralama sonuçları incelendiğinde ise, 1.21 ortalama sıralama değeri ile HBA2 ilk sırada, 1.79 ortalama sıralama değeri ile HBA1 ikinci sırada, 3.00 ortalama sıralama değeri ile BA üçüncü sırada yer almıştır.

Çizelge 5.28. CEC2011 kıyaslama fonksiyonlarında BA, HBA1 ve HBA2 algoritmalarının karşılaştırma sonuçları

	P9			P11.1		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.9321E+06	6.1974E+02	4.8582E+03	1.8610E+05	5.0923E+04	5.1410E+04
Ortanca	3.3873E+06	1.4325E+03	9.2071E+03	5.6688E+05	5.2241E+04	5.1990E+04
En kötü	3.6421E+06	2.8904E+03	1.7956E+04	1.1173E+06	5.3422E+04	5.2429E+04
Ortalama	3.4033E+06	**1.2947E+03**	9.9839E+03	5.6579E+05	5.2286E+04	**5.1948E+04**
Std.Sapma	1.8603E+05	5.2099E+02	3.5162E+03	2.3767E+05	6.8407E+02	3.0189E+02
	P11.2			**P11.7**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	4.4843E+06	1.0692E+06	1.0664E+06	1.9416E+06	1.9259E+06	1.8570E+06
Ortanca	5.1551E+06	1.0739E+06	1.0714E+06	5.0373E+08	1.9454E+06	1.8820E+06
En kötü	5.7044E+06	1.0783E+06	1.0764E+06	1.8157E+09	1.9613E+06	1.9773E+06
Ortalama	5.1587E+06	1.0739E+06	**1.0717E+06**	4.7139E+08	1.9451E+06	**1.8915E+06**
Std.Sapma	3.3675E+05	2.0114E+03	2.5048E+03	4.9551E+08	8.9314E+03	3.2037E+04
	P11.8			**P11.9**		
	BA	**HBA1**	**HBA2**	**BA**	**HBA1**	**HBA2**
En iyi	2.5601E+06	9.4475E+05	9.3798E+05	3.6776E+06	1.0990E+06	9.4191E+05
Ortanca	3.7587E+06	1.0221E+06	9.4238E+05	4.3840E+06	1.5608E+06	9.6805E+05
En kötü	6.4453E+06	1.6123E+06	9.5230E+05	5.1635E+06	1.9464E+06	1.0410E+06
Ortalama	3.6933E+06	9.4901E+05	**9.4267E+05**	4.3552E+06	1.6164E+06	**9.7480E+05**
Std.Sapma	7.2853E+05	1.7275E+05	3.0934E+03	3.8420E+05	2.2575E+05	2.6781E+04
	P11.10					
	BA	**HBA1**	**HBA2**			
En iyi	3.0295E+06	9.4032E+05	9.4083E+05			
Ortanca	3.9112E+06	1.0016E+06	9.4431E+05			
En kötü	5.2406E+06	1.3405E+06	9.5010E+05			
Ortalama	3.7591E+06	9.4775E+05	**9.4526E+05**			
Std.Sapma	6.8998E+05	1.1114E+05	2.6583E+03			
Friedman	BA		HBA1		HBA2	
	3.00		1.79		**1.21**	

Çizelge 5.29-5.31'de algoritmaların ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Bu test işleminde, algoritmaların her bir çevrim sonrasında elde ettiği ortalama değerler, ikili olarak karşılaştırılmıştır.

Çizelge 5.29. CEC2011 gerçek dünya problemleri için HBA1 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

Problem	**Daha iyi**	**Daha kötü**	**Eşit**	**p değeri**	**Karar**
P9	25	0	0	1.20E-05	+
P11.1	25	0	0	1.20E-05	+
P11.2	25	0	0	1.20E-05	+
P11.7	24	1	0	1.40E-05	+
P11.8	25	0	0	1.20E-05	+
P11.9	25	0	0	1.20E-05	+
P11.10	25	0	0	1.20E-05	+

Çizelge 5.30. CEC2011 gerçek dünya problemleri için HBA2 ve BA algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

Problem	Daha iyi	Daha kötü	Eşit	p değeri	Karar
P9	25	0	0	1.20E-05	+
P11.1	25	0	0	1.20E-05	+
P11.2	25	0	0	1.20E-05	+
P11.7	25	0	0	1.20E-05	+
P11.8	25	0	0	1.20E-05	+
P11.9	25	0	0	1.20E-05	+
P11.10	25	0	0	1.20E-05	+

Çizelge 5.31. CEC2011 gerçek dünya problemleri için HBA1 ve HBA2 algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları

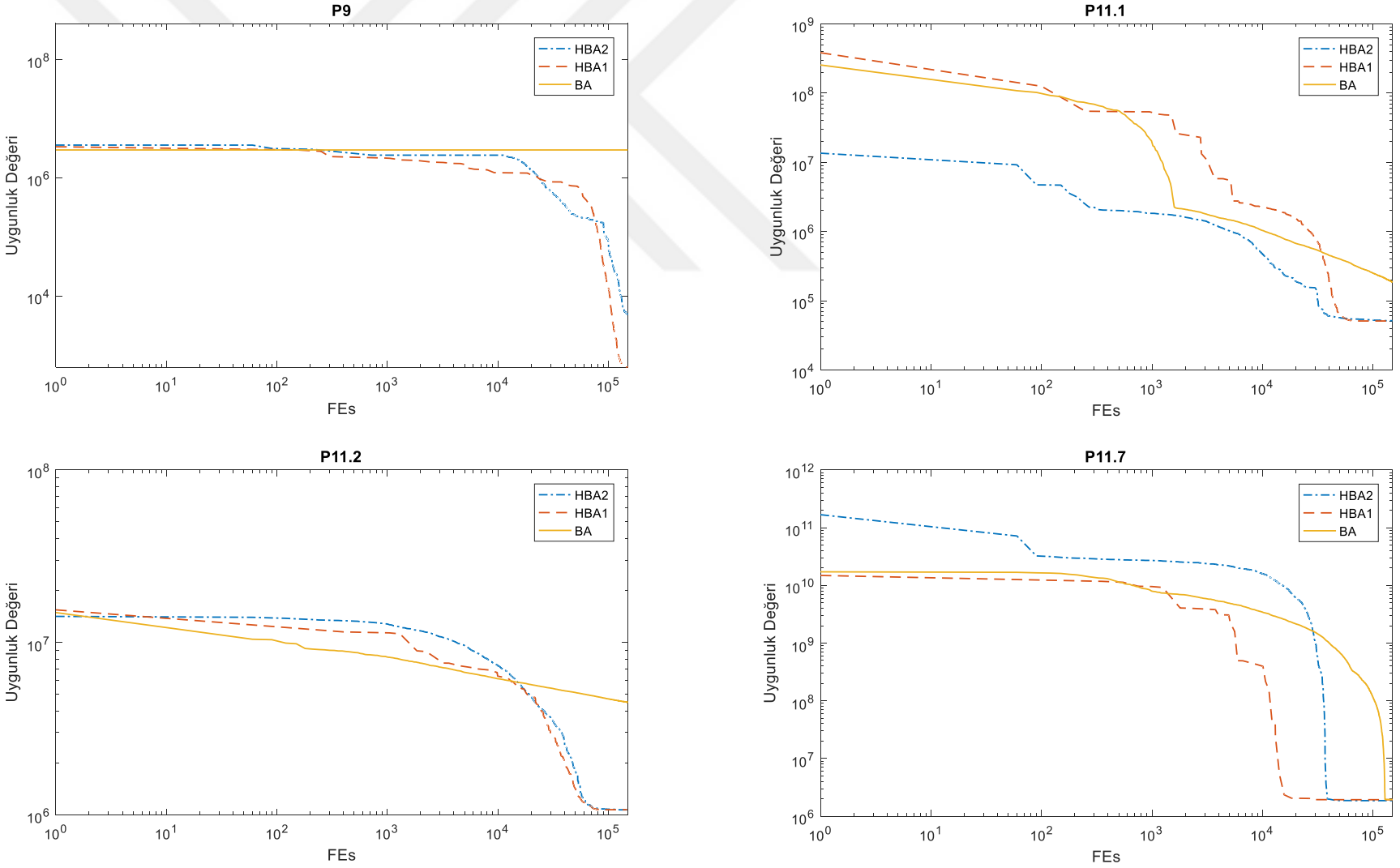
Problem	Daha iyi	Daha kötü	Eşit	p değeri	Karar
P9	25	0	0	1.20E-05	+
P11.1	11	14	0	1.28E-01	≈
P11.2	9	16	0	6.31E-03	-
P11.7	2	23	0	2.90E-05	-
P11.8	1	24	0	2.50E-05	-
P11.9	0	25	0	1.20E-05	-
P11.10	8	17	0	6.31E-03	-

Çizelge 5.29'da, CEC2011 büyük boyutlu gerçek dünya problemlerinde, HBA1 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, HBA1'in BA'ya göre tüm problemlerde anlamsal olarak daha iyi olduğu ve daha yüksek performans gösterdiği görülmektedir.

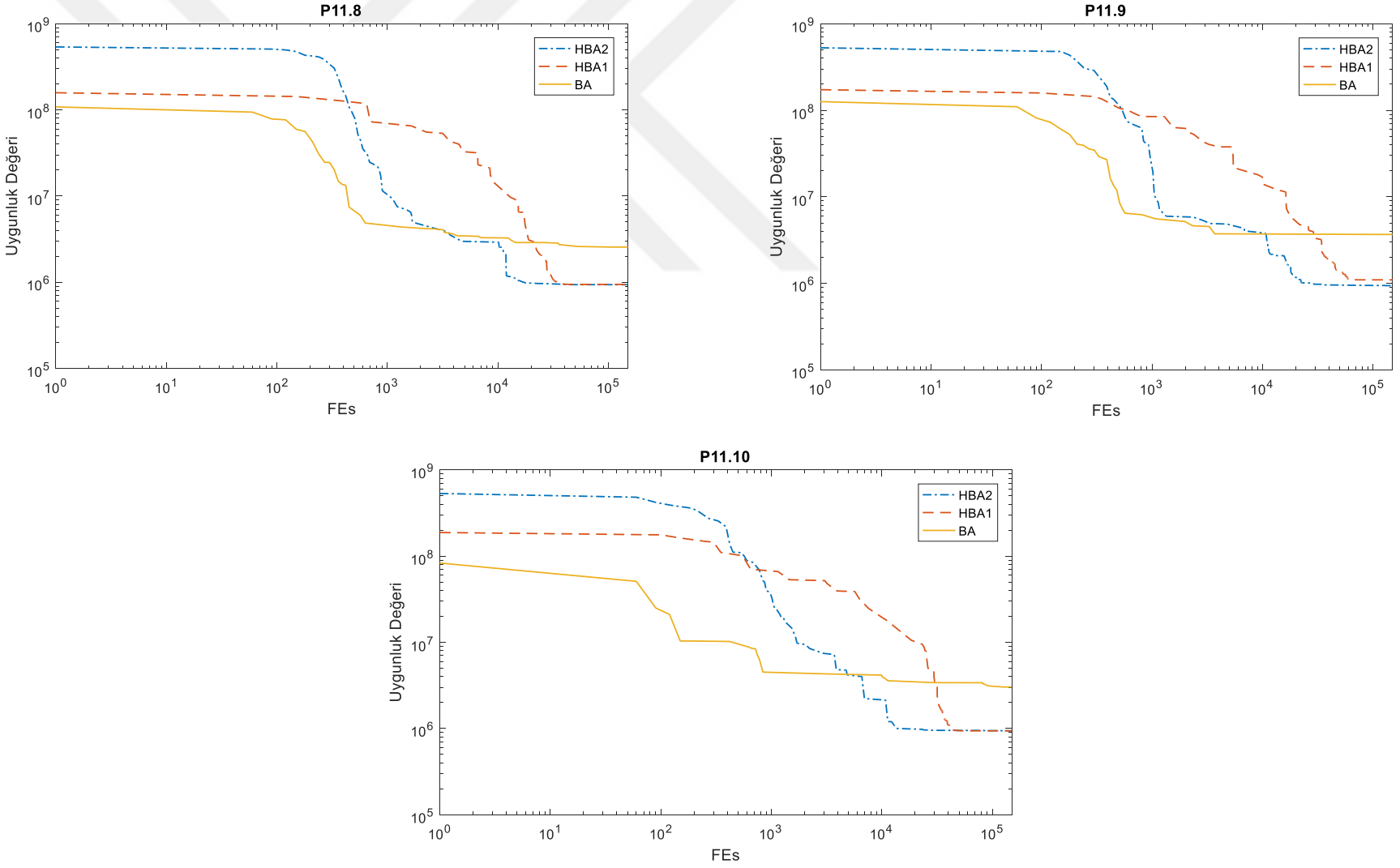
Çizelge 5.30'da, CEC2011 büyük boyutlu gerçek dünya problemlerinde, HBA2 ve standart BA algoritmasının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, HBA2'nin BA'ya göre tüm problemlerde daha iyi performans gösterdiği ve aralarında anlamlı bir fark olduğu bulunmuştur.

Çizelge 5.31'de ise, CEC2011 büyük boyutlu gerçek dünya problemlerinde, HBA1 ve HBA2 algoritmalarının ikili Wilcoxon işaretli sıra testi sonuçları verilmiştir. Sonuçlar incelendiğinde, HBA1'in bir problemde (P9) anlamsal olarak HBA2'den daha iyi, bir problemde (P11.1) ise HBA2 ile benzer performans gösterdiği tespit edilmiştir. Kalan beş problemde ise HBA1, HBA2'den daha kötü performans göstermiştir ve aralarında anlamlı bir fark mevcuttur.

Şekil 5.3'de, CEC2011 büyük boyutlu gerçek dünya problemleri için BA, HBA1 ve HBA2 algoritmalarından elde edilen en iyi değerlere göre yakınsama grafikleri verilmiştir. Grafikler incelendiğinde şu sonuçlar elde edilebilir. P9 probleminde, BA algoritması çözümü geliştirememiş ve performansı diğer algoritmaların gerisinde kalmıştır. HBA1 ve HBA2 algoritmalarının ise 10000 FEs sonrası yakınsama hızları artmıştır ve HBA1 algoritması HBA2'ye göre daha iyi bir çözüm üretmiştir. P11.1 probleminde, karşılaştırılan üç algoritmada araştırma süreci boyunca yeni çözüm üretme ve yakınsama eğiliminde olmuştur. Araştırma tamamlandığında, en iyi çözüm HBA1 algoritmasından elde edilmiştir. HBA2 algoritması bulunan en iyi çözüm çevresine daha hızlı yakınsamış olmasına rağmen arama sürecinin sonuna doğru yakınsama hızı düşmüş ve pek fazla yeni çözüm üretememiştir. P11.2 probleminde, BA algoritması daha yavaş yakınsamış ve performans olarak diğer algoritmaların gerisinde kalmıştır. HBA1 ve HBA2 algoritmaları ise birbirine benzer bir yakınsama karakteristiği sergilemiş olmalarına rağmen arama süreci sonunda HBA2 daha iyi bir çözüm üretmiştir. P11.7 probleminde, algoritmalar araştırma uzayında birbirine yakın çözümlere farklı yakınsama karakteristikleriyle yakınsamışlardır. HBA1 algoritması çözüme daha hızlı yakınsamış ancak en iyi çözüm HBA2 algoritması tarafından bulunmuştur. P11.8 probleminde, BA algoritması performans olarak diğer algoritmaların gerisinde kalmıştır. HBA1 ve HBA2 algoritması birbirine yakın çözümler bulmuşlardır. Ancak en iyi çözüm, HBA2 algoritmasından elde edilmiştir ve algoritma daha hızlı yakınsamıştır. P11.9 probleminde, üç algoritmada iterasyonun başında ve sonunda belli bir süre yakınsama hızı açısından stabil kalmış ve çözümlerini geliştirememişlerdir. BA algoritması, başlangıçta daha iyi bir çözümle aramaya başlamasına rağmen, arama sürecinin bitiminde performans olarak diğer algoritmaların gerisinde kalmıştır. Kalan algoritmalara ait grafikler incelendiğinde ise HBA2 algoritmasının daha iyi bir çözüm ürettiği görülmektedir. P11.10 probleminde, en hızlı yakınsama HBA2 algoritması tarafından gerçekleşmiştir ancak en iyi çözüm HBA1 algoritmasından elde edilmiştir. BA algoritması ise performans olarak diğer algoritmaların gerisinde kalmıştır.



Şekil 5.3.a. CEC2011 gerçek dünya problemlerinde BA, HBA1 ve HBA2 için yakınsama grafikleri



Şekil 5.3.b. CEC2011 gerçek dünya problemlerinde BA, HBA1 ve HBA2 için yakınsama grafikleri

HBA1 ve HBA2 algoritmaları, literatürden seçilen algoritmalarla da karşılaştırılmıştır. Karşılaştırmada kullanılan algoritmalara ait bilgiler Çizelge 5.32'de verilmiştir. Literatürdeki algoritmalara göre, önerilen hibrit algoritmaların performansını belirlemek için yapılan karşılaştırmada, tüm algoritmalar için FEs değeri 1.50E+05 ve genel çevrim sayısı 25 olarak alınmıştır. Elde edilen ortalama değerlere göre yapılan karşılaştırmanın sonuçları Çizelge 5.33'de verilmiştir.

Çizelge 5.32. Karşılaştırma yapılan algoritmalara ait bilgiler

Algoritma	Makale Adı	Referans
LGWO	An efficient modified grey wolf optimizer with Lévy flight for optimization tasks	(Heidari ve Pahlavani, 2017)
$IACO_R$	Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems	(Omran ve Al-Sharhan, 2019)
MIMO	A modified Intellects-Masses Optimizer for solving real-world optimization problems	(Omran ve ark., 2018)
L-covnSHADE	A novel differential crossover strategy based on covariance matrix learning with Euclidean neighborhood for solving real-world problems	(Awad ve ark., 2017)
APS9	APS 9: an improved adaptive population-based simplex method for real-world engineering optimization problems	(Omran ve Clerc, 2018)
dynNP-DE	On tenfold execution time in real world optimization problems with differential evolution in perspective of algorithm design	(Zamuda ve Brest, 2018)
GA_MPC	GA with a new multi-parent crossover for solving IEEE CEC2011 competition problems	(Elsayed ve ark., 2011b)
SAMODE	Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems	(Elsayed ve ark., 2011a)

Çizelge 5.33'te verilen karşılaştırma sonuçları incelendiğinde, seçilen büyük boyutlu gerçek dünya problemlerinin bir tanesinde LGWO (P9), üç tanesinde L-covnSHADE (P11.1, P11.2, P11.7) ve kalan üç tanesinde de APS9 algoritması en iyi ortalama uygunluk değerini üretmiştir. Algoritmalar için Friedman testine göre yapılan sıralamada 2.57 ortalama sıra değeri ile L-covnSHADE algoritması birinci olmuştur. HBA1 algoritması 6.14 ortalama sıralama değeri ile altıncı sırada, HBA2 algoritması ise 3.71 ortalama sıra değeri ile ikinci sırada yer almıştır. Bu karşılaştırma sonuçlarına göre, önerilen hibrit algoritmaların literatüre göre kabul edilebilir ve rekabetçi sonuçlar ürettiği söylenebilir. Özellikle HBA2 algoritması, ikinci sırada yer alarak, literatürdeki birçok algoritmadan daha başarılı olduğunu ortaya koymuştur.

Çizelge 5.33. CEC2011 gerçek dünya problemlerinde HBA1 ve HBA2 algoritmalarının literatürden seçilen algoritmalar ile karşılaştırma sonuçları

Problem	Algoritmalar									
	HBA1	**HBA2**	LGWO	$IACO_R$	MIMO	L-covnSHADE	APS9	dynNP-DE	GA_MPC	SAMODE
P9	1.2947E+03	9.9839E+03	**8.8895E-01**	3.2044E+04	2.5169E+03	2.7681E+04	2.8977E+03	2.1449E+03	1.2205E+03	1.2205E+03
P11.1	5.2286E+04	5.1948E+04	5.8128E+04	5.2421E+04	5.2156E+04	**5.1867E+04**	2.2386E+05	5.3178E+04	5.2054E+04	5.2054E+04
P11.2	1.0739E+06	1.0717E+06	1.0734E+06	2.1654E+07	2.4995E+07	**1.0711E+06**	1.7654E+07	1.0747E+06	1.0733E+06	1.0733E+06
P11.7	1.9451E+06	1.8915E+06	1.9240E+06	2.2015E+06	1.9181E+06	**1.8457E+06**	2.0665E+06	1.9905E+06	1.9533E+06	1.9533E+06
P11.8	9.4901E+05	9.4267E+05	9.4633E+05	1.1791E+06	9.4026E+05	9.3319E+05	**9.2776E+05**	9.4429E+05	9.7128E+05	9.7128E+05
P11.9	1.6164E+06	9.7480E+05	1.0235E+06	1.8758E+06	1.0880E+06	9.3973E+05	**9.3493E+05**	1.1810E+06	1.0562E+06	1.0562E+06
P11.10	9.4775E+05	9.4526E+05	9.4927E+05	1.1791E+06	9.4026E+05	9.3119E+05	**9.2776E+05**	9.4429E+05	9.7510E+05	9.7510E+05
Friedman	6.14	3.71	5.14	9.43	5.29	2.57	5.29	6.43	5.50	5.50
Sıra	6	2	3	8	4	**1**	4	7	5	5

Çizelge 5.34 ve 5.35'de ise sırasıyla HBA1 ve HBA2 algoritmalarının karşılaştırıldıkları diğer algoritmalarla olan ikili Wilcoxon işaretli sıra testi sonuçları yer almaktadır. Çizelge 5.34'e göre, HBA1 algoritmasının IACOR algoritmasından performansı daha iyidir ve aralarında anlamlı bir fark mevcuttur. Kalan sekiz algoritma ile arasında anlamlı bir fark bulunamamıştır ve benzer bir performans göstermişlerdir. Çizelge 5.35 incelendiğinde ise, HBA1 algoritmasına benzer şekilde, HBA2 algoritması da IACOR algoritmasından anlamsal olarak daha iyidir. Kalan sekiz algoritma ile de benzer performans göstermiştir ve aralarında anlamlı bir fark bulunamamıştır. Yapılan bu test işlemleri sonucunda, önerilen hibrit algoritmaların literaratüre kıyasla yarışmacı, kabul edilebilir ve başarılı sonuçlar ürettiği görülmüştür.

Çizelge 5.34. CEC2011 gerçek dünya problemlerinde HBA1 ve literatürden seçilen algoritmaların ikili Wilcoxon işaret testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA1-HBA2	1	6	0	1.28E-01	≈
HBA1-LGWO	2	5	0	3.10E-01	≈
HBA1-IACO$_R$	7	0	0	1.79E-02	+
HBA1-MIMO	2	5	0	3.98E-01	≈
HBA1-L_covnSHADE	1	6	0	1.28E-01	≈
HBA1-APS9	4	3	0	6.12E-01	≈
HBA1- dynNP-DE	4	3	0	7.35E-01	≈
HBA1-GA_MPC	3	4	0	8.66E-01	≈
HBA1-SAMODE	3	4	0	8.66E-01	≈

Çizelge 5.35. CEC2011 gerçek dünya problemlerinde HBA2 ve literatürden seçilen algoritmaların ikili Wilcoxon işaret testi sonuçları

Algoritmalar	Daha iyi	Daha kötü	Eşit	p değeri	Karar
HBA2-HBA1	6	1	0	1.28E-01	≈
HBA2-LGWO	6	1	0	1.28E-01	≈
HBA2-IACO$_R$	7	0	0	1.80E-02	+
HBA2-MIMO	4	3	0	3.98E-01	≈
HBA2-L_covnSHADE	1	6	0	1.28E-01	≈
HBA2-APS9	3	4	0	4.98E-01	≈
HBA2- dynNP_DE	5	2	0	1.76E-01	≈
HBA2-GA_MPC	6	1	0	6.30E-02	≈
HBA2-SAMODE	6	1	0	6.30E-02	≈

5.5. Algoritma Karmaşıklığı

Bu tez kapsamında yapılan deneysel çalışmalarda da görüldüğü gibi, bir problemin çözümü için birden fazla algoritma tercih edilebilir. Hangi algoritmanın probleme daha uygun olduğunun karar verilmesinde algoritmanın karmaşıklığı önemli bir rol oynar. Karmaşıklık, bir algoritmanın çok sayıda parametre karşısında maliyetindeki değişimi gösteren kavramsal bir ifadedir (Çölkesen, 2004). İdeal bir algoritmanın, daha iyi bir çözümü daha az karmaşıklıkla elde etmesi beklenir. Bilgisayar biliminde, her geçen gün hızla sayısı artan algoritmalar nedeniyle, bunlar arasından seçim yapma ya da yeni önerilen bir algoritmanın etkinliğini ispatlamada algoritmanın karmaşıklığı önemli bir kriter haline gelmiştir.

Algoritma karmaşıklığının, daha etkin ve standardize edilmiş bir şekilde değerlendirilebilmesi için, bazı CEC kıyaslama fonksiyonlarında bu amaçla bir karmaşıklık hesaplama kriteri oluşturulmuştur. Bu çalışmada önerilen algoritmaların karmaşıklık analizinde, CEC2020 kıyaslama fonksiyonları (Yue ve ark., 2019) için verilen kriterler kullanılmıştır. CEC2020 için verilen karmaşıklık hesaplama adımları aşağıdaki gibi açıklanabilir.

T0, Denklem 5.1'de verilen problem için geçen hesaplama süresini gösterir.

```
x = 0.55;
for i = 1: 1000000
   x = x + x;  x = x / 2;   x = x* x;  x = sqrt(x); x = log(x);
   x = exp(x);  x = x / (x+ 2);
end
```
(5.1)

T1, CEC2020 kıyaslama fonksiyonlarından Fonksiyon 1'in, belirlenmiş bir boyut(D) ve 2.00E+05 FEs için yürütülmesi sırasında geçen hesaplama süresini ifade eder. Hesaplamada kullanılan 1 nolu fonksiyon, kaydırılmış ve döndürülmüş özellikteki Bent Cigar fonksiyonudur (Yue ve ark., 2019). T2 ise algoritmanın, aynı fonksiyon üzerinde, aynı boyut değerleri için FEs değeri 2.00E+05 olacak şekilde yürütülmesi sırasında geçen toplam hesaplama süresidir. T2 hesaplama işlemi beş defa yürütülür ve elde edilen beş T2 değerlerinin ortalaması hesaplanır ve kaydedilir. Son olarak, algoritma karmaşıklığı (T2 - T1) / T0 olacak şekilde gösterilir. Algoritma karmaşıklığına, boyutun etkisini tespit etmek için 5, 10 ve 15 boyut için hesaplama yapılır.

Bu tez çalışması kapsamında önerilen algoritmalara ait karmaşıklık değerleri, Intel CPU (1.50GHz) işlemci ve 4GB Ram özellikli bir dizüstü bilgisayarda kodların yürütülmesi sonucu elde edilmiştir. BA, HBA1 ve HBA2 için elde edilen karmaşıklık değerleri Çizelge 5.36'da verilmiştir. Çizelgedeki değerler saniye cinsinden gösterilmiştir. Bu çizelgede görüldüğü üzere, algoritmaların üç farklı boyut için karmaşıklıkları hesaplanmıştır. Boyut artışı, üç algoritma içinde karmaşıklığın bir miktar artmasına sebep olmuştur. Algoritmalar kıyaslandığında ise standart BA'nın karmaşıklığı, tüm boyutlarda diğer algoritmalardan daha düşüktür. Hibrit algoritmalar kendi arasında değerlendirildiğinde ise, tüm boyutlar için HBA1'in karmaşıklığının HBA2'den daha yüksek olduğu bulunmuştur. BA ve HBA2 için hesaplanan karmaşıklık değerleri nispeten birbirine daha yakın değerlerdedir ve HBA2 algoritması standart BA'nın karmaşıklığını kabul edilebilir ölçüde artırmıştır. HBA1 algoritmasının karmaşıklığı ise tüm boyutlarda diğer algoritmaların karmaşıklığının iki katından daha fazladır.

Çizelge 5.36. BA, HBA1 ve HBA2'nin CEC2020 kriterlerine göre algoritma karmaşıklıkları

		F1			
		T0	*T1*	*T2*	*(T2-T1)/T0*
BA	**D=5**	0.2901	1.5922	13.0710	39.5684
	D=10		1.6110	13.1026	39.6125
	D=15		1.9029	13.6737	40.5749
HBA1	**D=5**	0.2901	1.5922	28.3885	92.3691
	D=10		1.6110	28.4096	92.3771
	D=15		1.9029	29.4730	95.0365
HBA2	**D=5**	0.2901	1.5922	13.0957	39.6535
	D=10		1.6110	13.3799	40.5684
	D=15		1.9029	14.2354	42.5112

CEC2020 fonksiyonları, sadece 5,10 ve 15 boyut için çalışacak şekilde tanımlanmıştır. Algoritmaların büyük boyut performansını iyileştirme amacında olan bu tez çalışması kapsamında, algoritmaların büyük boyutlu problemler üzerindeki karmaşıklığının da incelenmesi gerekmektedir. CEC2010 büyük boyutlu optimizasyon problemleri için tanımlanmış bir karmaşıklık hesaplama kriteri yoktur. Bu yüzden, CEC2010 büyük boyutlu optimizasyon problemlerinden farklı özelliklerde beş fonksiyon seçilmiştir. D=1000 olmak üzere, CEC2020'de verilen aynı kriterlere göre algoritmaların karmaşıklık değerleri hesaplanmıştır. Sonuçlar, Çizelge 5.37'de verilmiştir. Çizelge incelendiğinde, tüm fonksiyonlarda en düşük karmaşıklık değerleri standart BA algoritmasında, en yüksek karmaşıklık değeri ise HBA1 algoritmasında elde edilmiştir.

Çizelge 5.37. BA, HBA1 ve HBA2'nin seçilen CEC2010 fonksiyonları için algoritma karmaşıklıkları

		F1		
	T0	*T1*	*T2*	*(T2-T1)/T0*
BA			185.7949	292.3906
HBA1	0.2901	100.9724	705.3226	2083.248
HBA2			450.6813	1205.477
		F6		
BA			78.7660	125.1627
HBA1	0.2901	42.4563	405.4619	1251.312
HBA2			212.3968	585.7997
		F11		
BA			122.7833	100.7828
HBA1	0.2901	93.5462	729.4275	2191.938
HBA2			471.3327	1302.263
		F14		
BA			295.1337	425.6215
HBA1	0.2901	171.6609	1.1417E+03	3343.809
HBA2			627.1924	1570.257
		F20		
BA			107.6552	279.5684
HBA1	0.2901	26.5524	290.0021	908.1341
HBA2			220.7531	669.4267

Tüm sonuçlar değerlendirildiğinde, karmaşıklık sıralaması küçükten büyüğe doğru standart BA, HBA2 ve HBA1 şeklinde olmuştur. Fonksiyonların değişmesi ya da boyutun artışı algoritmalar arası karmaşıklık sıralamasında bir değişiklik yapmamıştır. BA ve HBA2 algoritması, küçük boyutlarda birbirine daha yakın karmaşıklık değerleri elde etmelerine rağmen, boyutun artışı elde ettikleri karmaşıklık değerleri arasındaki farkı artırmıştır. HBA1 algoritması ise tüm boyutlar ve tüm fonksiyonlarda en yüksek karmaşıklık değerine sahiptir.

5.6. Tartışma

Bu tez çalışması kapsamında, BA algoritmasının LSGO problemlerindeki performansını artırma amaçlı iki hibrit algoritma önerilmiştir. Bu hibrit algoritmalar, farklı CEC kıyaslama fonksiyonları üzerinde test edilmiştir ve sonuçlar istatistik testler yardımıyla yorumlanmıştır.

Önerilen algoritmalar, ilk olarak CEC2005 küçük ölçekli kıyaslama fonksiyonlarında 10 boyut için test edilmiştir. CEC2005 kıyaslama fonksiyonları, literatürde birçok BA versiyonunda daha önce kullanılmış olan bir fonksiyon kümesidir ve literatürde genellikle D=10 boyut için sonuçlar yer almaktadır. Bu tez çalışmasında önerilen HBA1 ve HBA2'nin, diğer BA

versiyonlarına göre performansını değerlendirmek amacıyla bu fonksiyon kümesi tercih edilmiştir. Daha fazla BA versiyonuyla kıyaslayabilmek adına da D=10 seçilmiştir. Yapılan test işlemleri sonucunda, önerilen HBA1 ve HBA2 algoritmalarının, standart BA algoritmasının performansını ciddi şekilde iyileştirdiği tespit edilmiştir. Hibrit algoritmalar arasında ise HBA1'in daha fazla sayıda fonksiyon için en iyi ortalama değeri bulduğu ve algoritmalar arası sıralamada ilk sırada yer aldığı görülmüştür. Yakınsama grafiklerinde de çoğunlukla önerilen hibrit algoritmaların daha hızlı bir yakınsama eğiliminde olduğu söylenebilir. HBA1 ve HBA2 algoritmaları literatürdeki BA versiyonlarıyla karşılaştırıldığında ise her ikisinin de kabul edilebilir, rekabetçi sonuçlar elde ettiği görülmüştür. Toplam on iki algoritma arasında yapılan sıralamada, HBA1 algoritması ikinci sırada, HBA2 algoritması dördüncü sırada yer almıştır. Algoritmaların diğer BA versiyonlarıyla olan ikili Wilcoxon test sonuçları ise algoritmaların diğer algoritmalarla çoğunlukla benzer ya da onlardan daha iyi performans gösterdiğini ortaya koymuştur.

Önerilen algoritmalar, ikinci olarak CEC2010 büyük boyutlu kıyaslama fonksiyonlarında 1000 boyut için test edilmiştir. Elde edilen test sonuçları birbirleriyle ve literatürden seçilen farklı algoritmalarla karşılaştırılmıştır. İlk olarak BA, HBA1 ve HBA2 algoritmalarında 1.20E+05, 6.00E+05 ve 3.00E+06 FEs değerleri için elde edilen sonuçlar karşılaştırmalı olarak verilmiştir. Böylece algoritmaların arama sürecindeki gelişim durumları incelenmiştir. Bu verilen FEs değerleri için algoritmalardan elde edilen sonuçlar birlikte değerlendirildiğinde, HBA2 algoritması BA ve HBA1'e kıyasla daha fazla fonksiyonda en iyi ortalama değeri elde etmiştir. Farklı FEs değerleri için yapılan performans sıralamalarının hepsinde HBA2 ilk sırada, HBA1 ikinci sırada ve BA üçüncü sırada yer almıştır. Bu sonuçlara göre, hibrit algoritmaların, BA algoritmasının büyük boyutlu problemlerdeki performansını önemli ölçüde geliştirdiği tespit edilmiştir. Yine yakınsama grafikleri incelendiğinde hibrit algoritmaların genel olarak daha hızlı yakınsadığı görülmektedir. Hibrit algoritmaların, literatürden seçilen algoritmalarla olan karşılaştırma sonuçları incelendiğinde ise algoritmaların kabul edilebilir, başarılı sonuçlar ürettiği görülmüştür. Özellikle, karşılaştırılan on bir algoritma arasında HBA2 algoritması üçüncü sırada yer alarak rakabetçi bir performans göstermiştir. Diğer algoritmalarla olan ikili Wilcoxon test sonuçları ise HBA1 ve HBA2'nin diğer algoritmalarla çoğunlukla benzer ya da onlardan daha iyi performans gösterdiğini ortaya koymuştur.

Önerilen algoritmalar, son olarak CEC2011 gerçek dünya problemleri üzerinde test edilmiştir. Seçilen büyük boyutlu yedi problem üzerinde BA, HBA1 ve HBA2 algoritmalarının sonuçları değerlendirildiğinde, problemlerin altısında HBA2 algoritmasının daha iyi çözümler sunduğu görülmüştür. Kalan bir fonksiyonda ise en iyi çözümü HBA1 algoritması üretmiştir. Algoritmaların performans sıralamasında ise HBA2 ilk sırada, HBA1 ikinci sırada ve BA üçüncü sırada yer almıştır. Dolayısıyla, hibrit algoritmaların BA algoritmasının performansını önemli ölçüde geliştirdiği söylenebilir. Yakınsama grafikleri incelendiğinde ise hibrit algoritmaların genel olarak daha hızlı yakınsadığı görülmektedir. Hibrit algoritmalar, literatürden seçilen algoritmalarla da karşılaştırılmıştır. Elde edilen sonuçlar, HBA1 ve HBA2'nin kabul edilebilir başarılı sonuçlar ürettiğini göstermiştir. Özellikle karşılaştırılan on algoritma arasında, HBA2 algoritması ikinci sırada yer alarak rekabetçi bir performans göstermiştir. Diğer algoritmalarla olan ikili Wilcoxon test sonuçları ise, HBA1 ve HBA2'nin diğer algoritmalarla çoğunlukla benzer ya da onlardan daha iyi performans gösterdiğini ortaya koymuştur.

Seçilen bu üç farklı fonksiyon kümesinden elde edilen sonuçlar genel olarak değerlendirildiğinde, HBA1 ve HBA2'nin standart BA'ya göre, problemlerin çoğunda daha başarılı sonuçlar ürettiği ve standart algoritmanın performansını artırdığı tespit edilmiştir. Hibrit algoritmalar kendi arasında değerlendirildiğinde ise, küçük ölçekli problemlerde HBA1 algoritmasının, büyük ölçekli problemlerde ise HBA2 algoritmasının daha iyi performans sergilediği görülmüştür. HBA1 algoritmasında, DE algoritmasının kullanılmasıyla lokal arama becerisi önemli ölçüde desteklenmiştir. HBA2 algoritmasında ise ABC algoritmasıyla ağırlıklı olarak global arama becerisi artırılmıştır. Dolayısıyla, araştırma uzayının arttığı LSGO problemlerinde HBA2'nin performansının daha iyi olması beklenen bir sonuç olmuştur. Benzer mantıkla, LSGO problemlerine nazaran, arama uzayı boyutunun azaldığı küçük boyutlu problemlerde, lokal arama becerisi daha fazla önem kazanmaktadır. Bu yüzden, DE algoritmasıyla lokal arama becerisi desteklenmiş olan HBA1 algoritması, bu problemlerde daha başarılı olmuştur.

Ayrıca, önerilen hibrit algoritmaların ortalama ve standart sapma değerlerine göre gürbüzlüğü incelendiğinde, standart algoritmaya göre tüm test kümelerinde HBA1 ve HBA2 algoritmalarının daha gürbüz olduğu söylenebilir. Hibrit algoritmalar karşılaştırıldığında ise küçük boyutlu problemlerde HBA1, büyük boyutlu problemlerde ise HBA2 algoritması daha gürbüzdür.

Deneysel sonuçların değerlendirilmesinin ardından, algoritmaların karmaşıklıkları da incelenmiştir. Hem küçük hem de büyük boyutlu fonksiyonlar üzerinde karmaşıklık hesabı yapılmıştır. CEC2020 kıyaslama fonksiyonlarından Fonksiyon 1 üzerinde yapılan küçük boyutlardaki karmaşıklık hesabında, tüm boyutlarda BA ve HBA2'nin karmaşıklık değerlerinin çok yakın değerler olduğu tespit edilmiştir. Ancak, HBA1 algoritmasının karmaşıklığı, tüm boyutlarda her iki algoritmadan daha yüksek olmuştur. Dolayısıyla, küçük boyut için karmaşıklık değerleri sıralaması küçükten büyüğe doğru BA, HBA2 ve HBA1 şeklinde olmuştur. CEC2010 büyük boyutlu fonksiyonlar üzerinde yapılan değerlendirmede ise karmaşıklık değerlerinin sıralamasında, tüm fonksiyonlarda herhangi bir değişiklik olmamıştır. Algoritma karmaşıklıkları küçükten büyüğe doğru yine BA, HBA2 ve HBA1 şeklinde olmuştur. Her iki algoritmada, büyük boyut için standart algoritmanın karmaşıklığını artırmıştır. Hibrit bir algoritmanın karmaşıklığının, standart bir algoritmanın karmaşıklığından fazla olması beklenen bir durumdur. Ancak karmaşıklıktaki artışın ve performansın çözülen problemler için kabul edilebilirliği önemlidir. Bu anlamda değerlendirildiğinde, HBA2'nin karmaşıklığında standart algoritmaya göre artışın daha az olması ve ürettiği başarılı sonuçlar sebebiyle başarılı ve rekabetçi bir algoritma olduğu söylenebilir. HBA1 algoritması ise özellikle BA versiyonlarıyla olan karşılaştırmalarda gösterdiği performans nedeniyle karmaşıklıkta neden olduğu artışa rağmen tercih edilebilir bir algoritma olarak görünmektedir.

6. SONUÇLAR VE ÖNERİLER

6.1. Sonuçlar

Metasezgisel algoritmalar, bir problemin çözüm uzayındaki tüm olası çözümlerini değerlendirmek yerine, makul bir süre içerisinde ümit vadeden belli sayıda çözümü değerlendirme eğiliminde olan algoritmalardır. Çözüm uzayı geniş olan LSGO problemlerinde, olası tüm çözümleri değerlendirmek maliyetli, zaman alan ve çoğu zaman gerçekleştirmesi zor bir iştir. Bu yüzden, LSGO problemlerinde metasezgisel algoritmalar sıklıkla tercih edilmektedir. Ancak, metasezgisel algoritmalar boyut artışına bağlı performans düşüşleri yaşayan algoritmalardır. Metasezgisel algoritmaları LSGO problemlerine uygularken ya problem daha küçük boyutlu alt problemlere ayrıştırılır ya da problem bütün olarak ele alınıp algoritmanın performansını artıracak bir takım modifikasyonlar (hibrit kullanım, etkili operatörler kullanma, lokal ve global arama yeteneğini geliştirmeye yönelik düzenlemeler vs.) yapılır.

Yarasa algoritması, anlaşılması ve kodlanması kolay, ayarlanması gereken parametre sayısı az olan bir algoritmadır. Yapısal problemleri sebebiyle, algoritmanın lokal ve global arama arasındaki dengenin desteklenmesi ve en iyi çözüme hızlı yakınsama eğiliminin yavaşlatılması gerekir. LSGO problemleri için algoritmanın hızlı yakınsama özelliği tercih sebebi olabilir. Ancak tek başına yeterli değildir ve güçlü bir global ve lokal arama becerisi ve dengesine ihtiyaç vardır. BA, bu yapısal problemleri azaltıldığı taktirde, LSGO problemlerinde başarılı sonuçlar üretebilecek kapasitede bir algoritmadır. Bu amaçla, bu tez çalışması kapsamında, büyük ölçekli sürekli optimizasyon problemlerinde yarasa algoritmasının performansını geliştirmeye yönelik iki hibrit algoritma önerilmiştir. MBADE (HBA1) ve BA_ABC (HBA2) isimleriyle literatüre kazandırılan bu iki hibrit algoritmanın, farklı test fonksiyonu kümeleri üzerinde farklı boyutlar için performansları incelenmiş ve istatistik testler yardımıyla yorumlanmıştır.

Tez çalışmasında önerilen HBA1 algoritması, BA'nın lokal arama yeteneğini artırmak ve popülasyon çeşitliliğini daha uzun süre korumak amacıyla BA'nın DE algoritmasıyla birleştirilmesiyle elde edilmiştir. Öncelikle, BA'nın yapısal problemlerini aşmaya yönelik bazı modifikasyonlar yapılarak algoritma geliştirilmiş, ardından popülasyondaki bireylere hangi algoritmanın uygulanacağına performansa dayalı bir olasılık değerine göre karar veren bir mekanizma oluşturulmuştur. Bu sayede, hibrit algoritmada aynı iterasyonda, popülasyondaki bireylere farklı algoritmalar uygulanabilir, bu durum da farklı lokal arama stratejilerinin bir arada

kullanılabilmesine imkan verir. Önerilen HBA1 algoritması, CEC2005 küçük ölçekli kıyaslama fonksiyonları, CEC2010 büyük ölçekli kıyaslama fonksiyonları ve CEC2011 gerçek dünya problemlerinden seçilen büyük ölçekli problemler üzerinde test edilmiştir. Sonuçlar, HBA1 algoritmasının, test edilen fonksiyonların/problemlerin çoğunda standart BA algoritmasına göre performansının daha iyi olduğunu ortaya koymuştur. Lokal arama yeteneği DE sayesinde geliştirilmiş olan bu algoritmanın, özellikle CEC2005 küçük ölçekli test fonksiyonlarında BA ve HBA2'den daha iyi performans gösterdiği tespit edilmiştir. Literatürdeki BA versiyonlarıyla da karşılaştırılan algoritma, algoritmalar arası sıralamada ikinci olmuştur. Bu sonuçlar, HBA1'in rekabetçi, başarılı ve ümit vadeden bir algoritma olduğunu ortaya koymuştur.

Tez çalışmasında önerilen HBA2 algoritması ise BA'nın global arama yeteneğini artırmak ve özellikle büyük ölçekli problemlerdeki performansını geliştirmek amacıyla BA ve ABC algoritmalarının birleştirilmesiyle elde edilmiştir. Hem BA'nın hızlı yakınsama becerisini, hem de ABC'nin global arama yeteneğini bir arada kullanabilmek adına popülasyon ikiye bölünmüş ve BA ve ABC algoritmaları farklı alt popülasyonlarında yürütülmüştür. Belli sayıda iterasyon tamamlandığında, algoritmaların başarı durumu incelenmiştir. Başarılı algoritmanın popülasyonundaki iyi bireylerin bir kısmı, diğer popülasyondaki aynı sayıdaki kötü bireylerin yerine yazılmıştır. Böylece, alt popülasyonlar arasında bilgi değişimi sağlanmıştır. Toplam değişim sayısının %60'ında başarılı olan bir algoritma tespit edilirse, kalan iterasyonlara tüm popülasyon üzerinde bu başarılı algoritma ile devam edilmiştir. HBA2 algoritması da, CEC2005 küçük ölçekli kıyaslama fonksiyonları, CEC2010 büyük ölçekli kıyaslama fonksiyonları ve CEC2011 gerçek dünya problemlerinden seçilen büyük ölçekli problemler üzerinde test edilmiştir. Sonuçlar, HBA2'nin test edilen fonksiyonların/problemlerin çoğunda, standart BA algoritmasına göre performansının daha iyi olduğunu ortaya koymuştur. ABC algoritması ile global arama yeteneği güçlendirilmiş olan bu algoritmanın, özellikle büyük ölçekli kıyaslama fonksiyonları ve gerçek dünya problemlerinde, standart BA ve HBA1'e göre daha iyi performans gösterdiği tespit edilmiştir. Literatürdeki algoritmalarla da karşılaştırılan HBA2 algoritmasının rekabetçi, başarılı ve kabul edilebilir sonuçlar ürettiği görülmüştür.

Yapılan algoritma karmaşıklığı testleri sonuçlarına göre hibrit algoritmaların karmaşıklık değerlerinin, standart algoritmaya göre daha fazla olduğu tespit edilmiştir. Boyutun büyümesi, standart algoritmayla hibrit algoritmaların karmaşıklık değerleri arasındaki farkın artmasına neden olmuştur. Ancak, hibrit algoritmaların performansa olan katkıları düşünüldüğünde, bu artışın kabul edilebilir seviyede olduğu söylenebilir.

Sonuç olarak, bu tez kapsamında geliştirilen hibrit algoritmalar sayesinde, BA'nın farklı özellik ve boyutlardaki kıyaslama fonksiyonları ve gerçek dünya problemleri üzerindeki performansı ciddi şekilde artırılmıştır. Diğer BA versiyonları ve metasezgisel algoritmalarla karşılaştırıldığında kabul edilebilir, rekabetçi ve başarılı sonuçlar üreten iki hibrit algoritma literatüre kazandırılmıştır. Standart BA'ya göre daha gürbüz iki yeni hibrit algoritma geliştirilmiştir. Ayrıca, bu çalışma sayesinde, LSGO ve büyük ölçekli gerçek dünya problemleri üzerinde BA'nın performansı hakkında literatüre önemli bir kaynak oluşturulmuştur. Önerilen hibrit algoritmalar farklı problem türlerine adapte edilebilir özelliktedir. Bu sayede, gerçek dünyada karşımıza çıkan farklı problemlerin çözümünde kullanılabilecek iki hibrit algoritmanın geliştirildiği ve literatüre kazandırıldığı söylenebilir.

6.2. Öneriler

Sürekli optimizasyon problemleri için önerilmiş HBA1 ve HBA2 algoritmalarının ikili, ayrık, kısıtlı veya çok amaçlı gibi farklı özelliklerdeki problemler üzerindeki performansı incelenebilir. Yine farklı CEC kıyaslama fonksiyonları ve gerçek dünya problemleri üzerindeki performansı değerlendirilebilir. Hibrit algoritmalarda kullanılan BA, DE ve ABC algoritmalarının yerine, farklı metasezgisel algoritmalar entegre edilerek yeni hibrit algoritmalar elde edilebilir. LSGO problemlerini daha küçük boyutlu alt problemler şeklinde ele alan ve temeli böl yönet mantığına dayanan ayrıştırma esaslı yaklaşımlar, önerilen algoritmalara dâhil edilerek, CC temelli hibrit algoritmalar oluşturulabilir.

KAYNAKLAR

Abu-Mouti F.S. ve El-Hawary M.E., 2012, Overview of Artificial Bee Colony (ABC) algorithm and its applications, *2012 IEEE International Systems Conference SysCon 2012*, 1-6.

Akay B., 2009, Nümerik optimizasyon problemlerinde yapay arı kolonisi (artificial bee colony) algoritmasının performans analizi, Doktora Tezi, *Fen Bilimleri Enstitüsü*, Erciyes Üniversitesi.

Al-Betar M.A. ve Awadallah M.A., 2018, Island bat algorithm for optimization, *Expert Systems with Applications*, 107, 126-45.

Al-Betar M.A., Awadallah M.A., Faris H., Yang X.-S., Khader A.T. ve Alomari O.A., 2018, Bat-inspired algorithms with natural selection mechanisms for global optimization, *Neurocomputing*, 273, 448-65.

Alataş B., 2007, Kaotik haritalı parçacık sürü optimizasyonu algoritmaları geliştirme, Doktora Tezi, *Fen Bilimleri Enstitüsü*, Fırat Üniversitesi.

Arora U. ve Lodhi M.E.A., 2016, PID Parameter Tuning Using Modified BAT Algorithm, *Journal of Automation and Control Engineering Vol*, 4 (5), 347-52.

Awad N.H., Ali M.Z., Suganthan P.N., Reynolds R.G. ve Shatnawi A.M., 2017, A novel differential crossover strategy based on covariance matrix learning with Euclidean neighborhood for solving real-world problems, *2017 IEEE Congress on Evolutionary Computation (CEC)*, 380-6.

Babayigit B. ve Ozdemir R., 2012, A modified artificial bee colony algorithm for numerical function optimization, *2012 IEEE Symposium on Computers and Communications (ISCC)*, 000245-9.

Cai X., Wang H., Cui Z., Cai J., Xue Y. ve Wang L., 2018, Bat algorithm with triangle-flipping strategy for numerical optimization, *International Journal of Machine Learning and Cybernetics*, 9 (2), 199-215.

Cai X., Zhang J., Liang H., Wang L. ve Wu Q., 2019, An ensemble bat algorithm for large-scale optimization, *International Journal of Machine Learning and Cybernetics*, 10 (11), 3099-113.

Chakri A., Khelif R., Benouaret M. ve Yang X.-S., 2017, New directional bat algorithm for continuous optimization problems, *Expert Systems with Applications*, 69, 159-75.

Chaudhary R. ve Banati H., 2019, Swarm bat algorithm with improved search (SBAIS), *Soft Computing*, 23 (22), 11461-91.

Chaudhary R. ve Banati H., 2020, Adaptive multi-swarm bat algorithm (AMBA), Soft Computing for Problem Solving, *Springer*, 805-21.

Chawla M. ve Duhan M., 2015, Bat algorithm: a survey of the state-of-the-art, *Applied Artificial Intelligence*, 29 (6), 617-34.

Çölkesen R., 2004, Bilgisayar programlama ve yazılım mühendisliğinde veri yapıları ve algoritmalar, *Papatya Yayıncılık*, İstanbul.

Das S. ve Suganthan P.N., 2010, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, *Jadavpur University, Nanyang Technological University, Kolkata*, 341-59.

Deng H., Peng L., Zhang H., Yang B. ve Chen Z., 2019, Ranking-based biased learning swarm optimizer for large-scale optimization, *Information Sciences*, 493, 120-37.

Derrac J., García S., Molina D. ve Herrera F., 2011, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary Computation*, 1 (1), 3-18.

Elsayed S.M., Sarker R.A. ve Essam D.L., 2011a, Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems, *2011 IEEE Congress of Evolutionary Computation (CEC)*, 1041-8.

Elsayed S.M., Sarker R.A. ve Essam D.L., 2011b, GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems, *2011 IEEE congress of evolutionary computation (CEC)*, 1034-40.

Epitropakis M.G., Tasoulis D.K., Pavlidis N.G., Plagianakos V.P. ve Vrahatis M.N., 2011, Enhancing differential evolution utilizing proximity-based mutation operators, *IEEE Transactions on Evolutionary Computation*, 15 (1), 99-119.

Fang W., Zhang L., Zhou J., Wu X. ve Sun J., 2017, A novel quantum-behaved particle swarm optimization with random selection for large scale optimization, *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2746-51.

Feng Y., Teng G.-F., Wang A.-X. ve Yao Y.-M., 2007, Chaotic inertia weight in particle swarm optimization, *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, 475-.

Gan C., Cao W., Wu M. ve Chen X., 2018, A new bat algorithm based on iterative local search and stochastic inertia weight, *Expert Systems with Applications*, 104, 202-12.

Gan C., Cao W.H., Liu K.Z., Wu M., Wang F.W. ve Zhang S.B., 2020, A New Hybrid Bat Algorithm and its Application to the ROP Optimization in Drilling Processes, *IEEE Transactions on Industrial Informatics*, 16 (12), 7338-48.

Gandomi A.H. ve Yang X.-S., 2014, Chaotic bat algorithm, *Journal of Computational Science*, 5 (2), 224-32.

García S., Molina D., Lozano M. ve Herrera F., 2009, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, *Journal of Heuristics*, 15 (6), 617.

Gardeux V., Omran M.G., Chelouah R., Siarry P. ve Glover F., 2017, Adaptive pattern search for large-scale optimization, *Applied Intelligence*, 47 (2), 319-30.

Ge H., Sun L. ve Yang X., 2016, Adaptive hybrid differential evolution with circular sliding window for large scale optimization, *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 87-94.

Ghanem W.A. ve Jantan A., 2019, An enhanced Bat algorithm with mutation operator for numerical optimization problems, *Neural Computing and Applications*, 31 (1), 617-51.

Gu Q., Li X. ve Jiang S., 2019, Hybrid genetic grey wolf algorithm for large-scale global optimization, *Complexity*, 2019.

Heidari A.A. ve Pahlavani P., 2017, An efficient modified grey wolf optimizer with Lévy flight for optimization tasks, *Appl Soft Comput*, 60, 115-34.

Huang H., Lv L., Ye S. ve Hao Z., 2019, Particle swarm optimization with convergence speed controller for large-scale numerical optimization, *Soft Computing*, 23 (12), 4421-37.

Karaboga D., 2004, Yapay zekâ optimizasyonu algoritmaları, *Atlas Yayın Dağıtım*, İstanbul.

Karaboga D., 2005, An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri.

Karaboga D. ve Basturk B., 2008, On the performance of artificial bee colony (ABC) algorithm, *Appl Soft Comput*, 8 (1), 687-97.

Karaboga D. ve Akay B., 2009, A comparative study of artificial bee colony algorithm, *Appl Math Comput*, 214 (1), 108-32.

Keskintürk T., 2006, Diferansiyel gelişim algoritması, *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, 5 (9), 85-99.

Kıran M.S., 2014, Optimizasyon problemlerinin çözümü için yapay arı kolonisi algoritması tabanlı yeni yaklaşımlar, Doktora Tezi, *Fen Bilimleri Enstitüsü*, Selçuk Üniversitesi.

Li Y., Li X., Liu J. ve Ruan X., 2019, An improved bat algorithm based on lévy flights and adjustment factors, *Symmetry*, 11 (7), 925.

Liu Q., Wu L., Xiao W., Wang F. ve Zhang L., 2018, A novel hybrid bat algorithm for solving continuous optimization problems, *Appl Soft Comput*, 73, 67-82.

Long W., Wu T., Liang X. ve Xu S., 2019, Solving high-dimensional global optimization problems using an improved sine cosine algorithm, *Expert Systems with Applications*, 123, 108-26.

Lyu S., Li Z., Huang Y., Wang J. ve Hu J., 2019, Improved self-adaptive bat algorithm with step-control and mutation mechanisms, *Journal of computational science*, 30, 65-78.

Ma Y. ve Bai Y., 2020, A multi-population differential evolution with best-random mutation strategy for large-scale global optimization, *Applied Intelligence*, 1-17.

Mahdavi S., Shiri M.E. ve Rahnamayan S., 2015, Metaheuristics in large-scale global continues optimization: A survey, *Information Sciences*, 295, 407-28.

Meng X.-B., Gao X.Z., Liu Y. ve Zhang H., 2015, A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization, *Expert Systems with Applications*, 42 (17-18), 6350-64.

Mohamed A.W., Hadi A.A. ve Jambi K.M., 2019, Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization, *Swarm and Evolutionary Computation*, 50, 100455.

Omran M.G., Al-Sharhan S. ve Clerc M., 2018, A modified Intellects-Masses Optimizer for solving real-world optimization problems, *Swarm and evolutionary computation*, 41, 159-66.

Omran M.G. ve Clerc M., 2018, APS 9: an improved adaptive population-based simplex method for real-world engineering optimization problems, *Applied Intelligence*, 48 (6), 1596-608.

Omran M.G. ve Al-Sharhan S., 2019, Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems, *Engineering Applications of Artificial Intelligence*, 85, 818-29.

Pathak V.K. ve Srivastava A.K., 2020, A novel upgraded bat algorithm based on cuckoo search and Sugeno inertia weight for large scale and constrained engineering design optimization problems, *Engineering with Computers*, 1-28.

Qin A.K. ve Suganthan P.N., 2005, Self-adaptive differential evolution algorithm for numerical optimization, *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 1785-91.

Ramezani F. ve Lotfi S., 2013, Social-based algorithm (SBA), *Appl Soft Comput*, 13 (5), 2837-56.

Rauf H.T., Malik S., Shoaib U., Irfan M.N. ve Lali M.I., 2020, Adaptive inertia weight Bat algorithm with Sugeno-Function fuzzy search, *Appl Soft Comput*, 90, 106159.

Shan X., Liu K. ve Sun P.-L., 2016, Modified bat algorithm based on lévy flight and opposition based learning, *Scientific Programming*, 2016.

Storn R. ve Price K., 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J Global Optim*, 11 (4), 341-59.

Suganthan P.N., Hansen N., Liang J.J., Deb K., Chen Y.-P., Auger A. ve Tiwari S., 2005, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, *KanGAL report*, 2005005 (2005), 2005.

Sun Y., Wang X., Chen Y. ve Liu Z., 2018, A modified whale optimization algorithm for large-scale global optimization problems, *Expert Systems with Applications*, 114, 563-77.

Tang D., Dong S., Jiang Y., Li H. ve Huang Y., 2015, ITGO: Invasive tumor growth optimization algorithm, *Appl Soft Comput*, 36, 670-98.

Tang K., Li X., Suganthan P., Yang Z. ve Weise T., 2009, Benchmark Functions for the CEC 2010 Special Session and Competition on Large-Scale Global Optimization (2010), *Nature Inspired Computation and Applications Laboratory, USTC, China*.

Tawhid M.A. ve Ali A.F., 2017, Multi-directional bat algorithm for solving unconstrained optimization problems, *Opsearch*, 54 (4), 684-705.

Wang G.-G., Lu M. ve Zhao X.-J., 2016, An improved bat algorithm with variable neighborhood search for global optimization, *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1773-8.

Wang Y., Wang P., Zhang J., Cui Z., Cai X., Zhang W. ve Chen J., 2019, A novel bat algorithm with multiple strategies coupling for numerical optimization, *Mathematics*, 7 (2), 135.

Wu X., Wang Y., Liu J. ve Fan N., 2019, A new hybrid algorithm for solving large scale global optimization problems, *IEEE Access*, 7, 103354-64.

Yang X.-S., 2010a, A new metaheuristic bat-inspired algorithm, Nature inspired cooperative strategies for optimization (NICSO 2010),284, *Springer*, Berlin, Heidelberg, 65-74.

Yang X.-S., 2010b, Nature-inspired metaheuristic algorithms, *Luniver press*.

Yang X.S. ve Gandomi A.H., 2012, Bat algorithm: a novel approach for global engineering optimization, *Eng Computation*.

Yildiz Y.E. ve Topal A.O., 2019, Large scale continuous global optimization based on micro differential evolution with local directional search, *Information Sciences*, 477, 533-44.

Yilmaz S. ve Kucuksille E.U., 2013, Improved bat algorithm (IBA) on continuous optimization problems, *Lecture Notes on Software Engineering*, 1 (3), 279.

Yılmaz S., 2014, Yarasa algoritmasının unimodal, multimodal ve kaydırılmış sayısal optimizasyon problemleri (cec05) üzerinde geliştirilmesi, Yüksek Lisans Tezi, *Fen Bilimleri Enstitüsü*, Süleyman Demirel Üniversitesi.

Yılmaz S., Kucuksille E.U. ve Cengiz Y., 2014, Modified bat algorithm, *Elektron Elektrotech*, 20 (2), 71-8.

Yılmaz S. ve Küçüksille E.U., 2015, A new modification approach on bat algorithm for solving optimization problems, *Appl Soft Comput*, 28, 259-75.

Yu H., Zhao N., Wang P., Chen H. ve Li C., 2020, Chaos-enhanced synchronized bat optimizer, *Applied Mathematical Modelling*, 77, 1201-15.

Yue C., Price K., Suganthan P., Liang J., Ali M., Qu B., Awad N. ve Biswas P., 2019, Problem definitions and evaluation criteria for the CEC 2020 special session and competition on single objective bound constrained numerical optimization, *Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Tech. Rep*, 201911.

Zamuda A. ve Brest J., 2018, On tenfold execution time in real world optimization problems with differential evolution in perspective of algorithm design, *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 1-5.

Zhang F. ve Zhang F., 2019, Large-scale Global Optimization based on ABC Algorithm with Gbest-guided Strategy and Opposite-based Learning Technique, *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 101-7.

Zhu B., Zhu W., Liu Z., Duan Q. ve Cao L., 2016, A novel quantum-behaved bat algorithm with mean best position directed for numerical optimization, *Computational intelligence and neuroscience*, 2016.