Journal of the Faculty of Engineering and Architecture of Gazi University 38:1 (2023) 15-27

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

# A new lattice based artificial bee colony algorithm for EEG noise minimization

Sibel Arslan[1]*, Selçuk Aslan[2]
[1]Department of Software Engineering, Technology Faculty, Sivas Cumhuriyet University, 58140, Sivas, Türkiye
[2]Department of Aeronautical Engineering, Aeronautics and Astronautics Faculty, Erciyes University, 38039, Kayseri, Türkiye

**Highlights:**

- The ABC algorithm was remodeled by considering the big data concept.
- A new big data optimizer called lattice based ABC (LBABC) was introduced.
- LBABC were compared to other meta-heuristics by using a big data noise minimization problem.

**Keywords:**

- ABC Algorithm
- Lattice based Search
- Big Data Optimization

**Article Info:**

Research Article
Received: 24.08.2021
Accepted: 12.01.2022

**DOI:**

10.17341/gazimmfd.986747

**Correspondence:**

Author: Sibel Arslan
e-mail: sibelarslan@cumhuriyet.edu.tr
phone: +90 346 219 1241-2318

**Graphical/Tabular Abstract**

The big data concept and its unique features have modified the descriptions of the real world optimization problems and investigating the performances of the previously introduced solving techniques and developing new methods by considering the properties of the big data concept have become critical. By considering this requirement, the ABC algorithm was remodeled for solving a recent big data optimization problem that requires noise minimization on the brain signals and a new big data optimizer called Lattice based ABC (LBABC) was proposed. The capabilities of the proposed big data optimizer were evaluated with six different problem instances and obtained results were compared to the different meta-heuristics.

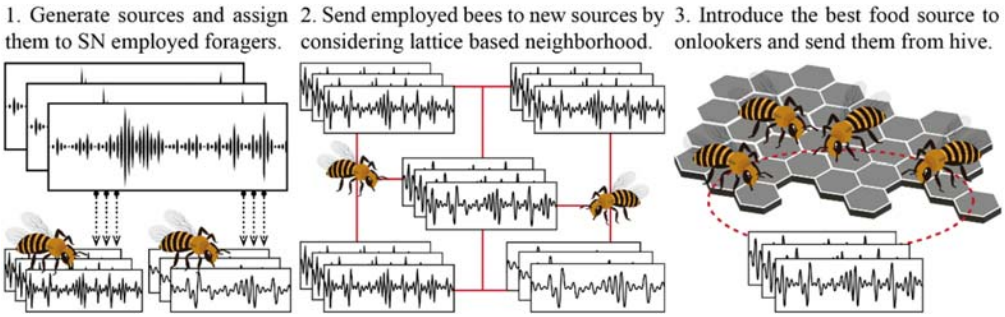


**Figure A.** Lattice based ABC algorithm

**Purpose:**

The main purpose of this study is modifying the bee phases of the ABC algorithm appropriately by considering the different aspects of the big data concept and developing an ABC based big data optimization technique.

**Theory and Methods:**

The existence of big data increases the difficulty of the problem being optimized and stages of a meta-heuristic should be subtly adjusted. In the proposed technique, the initialization schema was specialized by guiding the source signal data. Moreover, the employed bee phase was modified for increasing the exploration capabilities with a new schema called lattice based neighborhood. Finally, in order to balance the increased exploration capability with an appropriate exploitation mechanism, the search routines of the onlookers were changed in a manner that each of them flies to examine the vicinity of the best food source found until current evaluation.

**Results:**

The performance of the LBABC was investigated in detail by assigning different values to the specific control parameters such as colony size and limit and solving six problem instances. For analyzing the short, medium and long term responses of the LBABC, different termination conditions were used. The solutions of the LBABC were compared with the solutions of the different optimizers under the same conditions. The comparative studies showed that LBABC produces at least 120.74 times better solutions for the instances with twelve-time series and 156.40 times better solutions for the instances with nineteen-time series when compared to its nearest ABC based competitor.

**Conclusion:**

Promising results obtained by the LBABC proved that when a meta-heuristic is decided to be used as a solver for big data problems, stages of the algorithm should be designed by considering the availability of data and its effect on the dimensionality and constraints. Generating initial solutions by guiding data and increasing search capability properly as in the LBABC significantly contribute to the qualities of the final solutions.

Journal of the Faculty of Engineering and Architecture of Gazi University 38:1 (2023) 15-27

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

# EEG gürültü minimizasyonu için kafes temelli yeni bir yapay arı koloni algoritması

Sibel Arslan[1]*, Selçuk Aslan[2]
[1]Sivas Cumhuriyet Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, 58140, Sivas, Türkiye
[2]Erciyes Üniversitesi, Havacılık ve Uzay Bilimleri Fakültesi, Uçak Mühendisliği Bölümü, 38039, Kayseri, Türkiye

## ÖNEÇIKANLAR

- Yapay Arı Koloni (ABC) algoritması, büyük veri kavramı dikkate alınarak yeniden şekillendirildi
- Kafes temelli ABC (LBABC) adlı yeni bir büyük veri en iyileme yöntemi tanıtıldı
- LBABC gürültü minimizasyonu problemi üzerinden diğer meta-sezgisel yöntemler ile karşılaştırıldı

Makale Bilgileri
Araştırma Makalesi
Geliş: 24.08.2021
Kabul: 12.01.2022

DOI:
10.17341/gazimmfd.986747

Anahtar Kelimeler:
ABC algoritması,
kafes temelli arama,
büyük veri en iyileme
problemi

ÖZ

Geçtiğimiz yıllar büyük veri olarak adlandırılan yeni bir kavramla başlayan değişimlere tanıklık etmiştir. Bu yeni kavram ve özellikleri gerçek hayat en iyileme problemlerinin tanımlarını değiştirmiş ve daha önce önerilen çözüm yöntemlerinin performanslarının incelenmesi ve büyük veri kavramının özelliklerini dikkate alarak yeni yöntemlerin geliştirilmesi kritik hale gelmiştir. Arıların yiyecek arama davranışlarından ilham alan Yapay Arı Koloni (ABC) algoritması sürü zekâsı temelli yöntemlerinin en başarıları arasındadır. Bu çalışmada, ABC algoritmasının işçi ve gözcü arı fazları elektroensefalografi sinyallerinde gürültü minimizasyonunu gerektiren büyük veri en iyileme probleminin çözümü için düzenlenmiş ve kafes temelli ABC algoritması (LBABC) tanıtılmıştır. Önerilen yöntemin çözüm kapasitesinin analizi için farklı problem örneklerini içeren bir dizi uygulama gerçekleştirilmiştir. Elde edilen sonuçlar yaygın kullanılan yöntemler tarafından elde edilen sonuçlar ile de kıyaslanmıştır. Karşılaştırma sonuçlarından, yeni yönteminin test problemlerinin tamamına yakınında diğer yöntemlerden daha iyi ya da oldukça yakın çözümlere ulaşabildiği anlaşılmıştır.

# A new lattice based artificial bee colony algorithm for EEG noise minimization

## HIGHLIGHTS

- The ABC algorithm was remodeled by considering the big data concept
- A new big data optimizer called Lattice Based ABC (LBABC) was introduced
- LBABC were compared to other meta-heuristics by using a big data noise minimization problem

Article Info
Research Article
Received: 24.08.2021
Accepted: 12.01.2022

DOI:
10.17341/gazimmfd.986747

Keywords:
ABC algorithm,
lattice based search,
big data optimization

ABSTRACT

The last decades have witnessed the changes stemming from the existence of a new term called big data. This new concept and its features have modified the descriptions of real world optimization problems and investigating the performances of the previously introduced solving techniques and developing new methods by considering the properties of big data concept have become critical. Artificial Bee Colony (ABC) algorithm inspired by foraging behaviors of the real honey bees is one of the most successful swarm intelligences based techniques. In this study, employed and onlooker bee phases of the ABC algorithm were remodeled for solving a recent big data optimization problem that requires noise minimization on the electroencephalography signals and lattice based ABC (LBABC) was proposed. For analyzing the solving capabilities of the proposed technique, a set of experimentals has been carried out by using different problem instances. The results obtained from the experimental studies were also compared with the results of well-known techniques. From the comparative studies, it was understood that the newly introduced big data optimization technique by referencing the ABC algorithm is capable of producing better or relatively similar results compared to the other techniques for the vast majority of the problem instances.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *sibelarslan@cumhuriyet.edu.tr, selcukaslan@erciyes.edu.tr /
Tel: +90 346 219 1241-2318

## 1. Giriş (Introduction)

Hızla gelişen ağ ve haberleşme teknolojileri, aralarında akıllı telefonların yer aldığı sayısal sistemlerin yayıncılık, bankacılık, satış alanlarında kullanımını neredeyse zorunlu hale getirmiş ve üretici firma ya da servis sağlayıcıların ihtiyaç duyulan temel ve biyometrik verileri almasını kolaylaştırmıştır [1, 2]. Verilerin artan hacmi ve çeşitliliği sebebi ile sadece depolanması bir mühendislik problemi olarak değerlendirilirken, işlenmesi daha da önemli bir problem olarak karşımıza çıkar [3-5]. Bu bağlamda, hacim ve çeşitliliği sebebi ile geleneksel yöntemler kullanılarak işlenmesi ve kıymetlendirilmesi sonlu ya da kabul edilebilir zamanlarda mümkün olmayan kayıt kümeleri için büyük veri tanımı yapılmıştır [6]. Hacim ve çeşitlilik kavramları sadece depolanacak verinin yeniden tanımlanmasına sebep olmamış, bu verilere bağlı olarak bir ya da daha fazla amaç fonksiyonunun minimum (maksimum) değerlerinin bulunmasını gerektiren en iyileme problemlerinin de büyük veri en iyileme (Big Data Optimization, BDO) problemleri olarak değerlendirilerek yakın bir zaman önce literatüre girmesini sağlamıştır [7, 8]. BDOP, çok sayıda karar değişkeninden, çeşitli kısıtlama türlerinden, amaç fonksiyonlarının çeşitli matematiksel özelliklerinden ve büyük hesaplama zaman gereksinimlerinden oluşur [9]. Bu nedenle, gerçek dünyadaki uygulamaları ve zorlukları nedeniyle, BDO sağlık, ulaşım, lojistik ve diğerleri gibi çeşitli alanlar için önemli bir araştırma konusudur.

Büyük veri ve en iyileme kavramlarının birlikte kullanıldığı araştırma konularından biri 2015 yılında Goh vd. tarafından araştırmacılara duyurulmuştur [10]. Goh vd. tarafından tanıtılan bu problem, elektroensefalografi (EEG) sinyallerinde gürültünün temel istatistik hesaplamalara bağlı olarak minimizasyonunu hedeflemektedir. Belirtilen BDO probleminin çözümü için ilk çalışmanın Elsayed ve Sarker tarafından yapıldığı söylenebilir. Elsayed ve Sarker, Diferansiyel Gelişim (Differential Evolution, DE) algoritmasının BDO problemlerinin çözümünde daha iyi sonuçlar üretebilmesi amacıyla bir dizi değişiklik yapmışlardır [11]. Yapılan ilk düzenleme başlangıç popülasyon sayısı üzerine olmuştur. Tek bir başlangıç popülasyonu yerine birden fazla başlangıç popülasyonu ile problemlerin çözümüne başlanmış, her popülasyon için aday üretme mekanizmaları ve kontrol parametreleri farklı olan DE algoritmalarından faydalanılmıştır. Daha önceden belirlenen iterasyon sayısına ulaşıldığında farklı popülasyonların en kaliteli çözümleri birleştirilerek tek bir popülasyon oluşturulmuş, oluşturulan popülasyonun kalitesi Interior Point adı verilen yerel arama algoritması ile bir miktar daha iyileştirilmiştir. Elde edilen popülasyon, başlangıç popülasyonları için kullanılan DE varyantlarından en başarılısı kullanılarak kalan iterasyonlarda da iyileştirilmeye çalışılmıştır. Majdouli vd. Havai Fişek (Fireworks algorithm, FW) algoritmasından faydalanarak geliştirdikleri yeni modelleri ile EEG sinyallerinde gürültüyü filtrelemeye çalışmışlardır [12]. Önerdikleri bu yaklaşımda başlangıç çözümlerini, parametrelerin alacağı değerlerin alt ve üst sınırları arasında rastgele üretmek yerine problem için verilen gürültülü sinyalini bir miktar değiştirerek elde etmişlerdir. Sabar vd. devam eden iterasyonlarda kontrol parametrelerinin değerlerini, aday çözüm üretme stratejilerini ve yerel arama yöntemlerini dinamik olarak belirleyebildikleri heterojen-memetik DE algoritması önermişler ve önerdikleri DE temelli yaklaşımın performansını Goh vd. tarafından önerilen probleminin çözümü üzerinden incelemişlerdir [13]. Önerilen memetik yaklaşımda dört farklı mutasyon, iki farklı çaprazlama stratejisi, çaprazlama oranı için üç ve ölçekleme faktörü içinde üç olmak üzere toplamda altı farklı parametre değeri ve son olarak iki farklı yerel arama yöntemi arasından ilgili iterasyonda hangilerinin kullanılacağının kararı Page-Hinkley istatistik testi ile verilmektedir. Bahsedilen önemli değişikliklere ek olarak, Sabar vd. memetik yöntemin performansını bir miktar daha artırabilmek için problemin daha kolay optimize edilebilir alt problemlere ayrılıp çözülmesi prensibine dayanan cooperative co-evolution yaklaşımından da faydalanmışlardır. Wang vd. önerdikleri hibrit Ateş Böceği (Firefly Algorithm, FA) algoritmasını kullanarak EEG sinyal minimizasyonu temelli BDO problemini çözmeye çalışmışlardır [14]. Önerilen yaklaşımda, varsayılan aday çözüm üretme işlemi DE algoritmasının aday çözüm üretme stratejisi ile değiştirilmiştir. Yi vd. Genetik Algoritmanın (Genetic Algorithm, GA) en yaygın kullanılan çok amaçlı varyantı olan NSGA (Non-dominated Sorting Genetic Algorithm) ile adaptif mutasyon yaklaşımını birleştirerek önerdikleri hibrit yaklaşımı üç farklı çaprazlama operatörü ile altı farklı sinyal grubu üzerinden EEG gürültü minimizasyonu için test etmişlerdir [15]. Elaziz vd. çok amaçlı BDO problemlerini çözmek için Salp Sürü Algoritması (Salp Swarm Algorithm, SSA) ve DE algoritmalarını kullanarak 3 fazlı hibrit bir yaklaşım önerdiler [16]. Çok amaçlı Salp Sürü Algoritması ve Diferansiyel Gelişim (Multiobjective SSA and Differential Evolution, MSSADE) olarak adlandırılan yaklaşım, DE'yi yerel arama operatörleri olarak kullanarak SSA'nın yerel optimum noktalarına takılmasını engellemeye çalışmıştır. MSSADE BDO problemlerinde yüksek hesaplama zamanı nedeniyle dezavantaja sahip olsa da kıyaslanan NSGA algoritmasına karşı başarılı sonuçlar elde etmiştir. Abdi vd. farklı arama stratejilerini bir algoritmada birleştiren Arama Yönetimi (Seach Menager, SM) algoritmasının başarısını BDO problemlerinde test etmişlerdir [17]. Test sonuçlarına göre SM algoritması oldukça iyi yakınsama hızına sahiptir. Meselhi vd. paralel hesaplama yapabilen BDO problemleri için DE temelli yeni bir algoritma önerdiler [18]. Algoritmanın yakınsama hızını arttırmak için gradyen temelli lokal aramayı da algoritmaya dahil edilmiş ve EEG sinyalleri üzerinde başarısı gözlemlenmiştir. Celil vd. BDO problemlerini Harmoni Arama (Harmony Search, HS) temelli slinkHSA olarak adlandırdıkları yeni bir algoritma ile çözdüler [19]. Algoritmanın bilinen meta-sezgisel algoritmalarla kıyaslanmasından elde edilen sonuçlar slinkHSA algoritmasının BDO problemlerine daha uygun olduğunu göstermiştir.

Aslan, Yapay Arı Koloni (Artificial Bee Colony, ABC) algoritması ve farklı varyantlarını EEG sinyallerinde gürültüyü filtrelemek amacıyla kullanmıştır [20]. Elde edilen sonuçlar gözcü arı fazında yapılan değişikliklerin ABC algoritmasının çözüm kapasitesini artırdığını ve bu fazda algoritmanın standart uygulamasına kıyasla değişiklik öneren varyantların oldukça başarılı sonuçlar üretebildiğini göstermiştir. Bu değişikliklerin ABC algoritmasının kapasitesi üzerindeki olumlu katkısı, EEG sinyal minimizasyonu temelli BDO probleminin çözümü için ABC algoritmasının kullanıldığı yeni yöntemlerin geliştirilebileceği hususunda bilgi vermektedir. Bu çalışmada, başlangıç çözümlerinin üretilme mekanizması, işçi ve gözcü arı fazları BDO probleminin özellikleri dikkate alınarak değiştirilmiş yeni bir ABC algoritması olan Kafes Temelli ABC algoritması (Lattice Based ABC, LBABC) tanıtılmıştır. LBABC algoritmasının başarımı önce ABC algoritmasının standart uygulaması ve gbest-guided ABC (GABC) [21], ABC/best/1 [22], ABC/best/2 [22], crossover ABC (CABC) [23], converge-onlookers ABC (COABC) [24] ve quick ABC (qABC) [25] gibi varyantları ile yapılan karşılaştırmalar üzerinden incelenmiştir. LBABC algoritmasının başarımı ayrıca DE, GA, FA, FW, Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) ve Faz Optimizasyon (Phase Based Optimization, PBO) büyük veri temelli en iyileme yöntemleri ile yapılan karşılaştırmalar üzerinden incelenmiştir. Karşılaştırmalar LBABC algoritmasının diğer yöntemlere kıyasla neredeyse tüm test senaryolarında daha başarılı sonuçlara ulaşabildiğini göstermiştir. Çalışmanın ikinci bölümünde EEG sinyal minimizasyonu temelli BDO problemi tanıtılmış, üçüncü bölümünde ABC algoritmasının temel işlem adımlarından ve LBABC

algoritmasının detaylarından bahsedilmiştir. Dördüncü bölümde uygulama sonuçlarına yer verilmiştir. Son bölümde ise genel değerlendirme yapılmış ve muhtemel çalışmalar ile ilgili hususlara değinilmiştir.

## 2. Deneysel Method (Experimental Method)

### *2.1. Büyük Veri Gürültü Minimizasyon Problemi (Big Data Noise Minimization Problem)*

IEEE CEC 2015 konferansı ile araştırmacılara Goh vd. tarafından duyurulan BDO problemi, saniyede 20 KB (kilobyte) nümerik, 512 KB metin temelli veri aktarabilen bir cihazdan alınmış EEG sinyallerinden faydalanır [10]. Tanımlanan BDO probleminin temel amacı, cihazdan her saniye yapılacak ölçüm için gürültüyü minimize ederek doğru EEG sinyallerine ulaşabilmektir [10]. Goh vd. kullandıkları cihaz ile 4, 12 ve 19 kanal üzerinden 256 örnek alarak bir saniyelik ölçümler yapmış ve ölçüm sonuçlarını kaydetmiştir. Kaydedilen bir saniyelik ölçümlerin varsayılan gürültülü halleri için kanal sayılarından faydalanarak D4, D12 ve D19, kaydedilen ölçümlerin varsayılan gürültüye ek olarak belirli bir oranda gürültü eklenmiş halleri için ise D4N, D12N ve D19N isimleri kullanılmıştır [10]. $N$ zaman serilerinin sayısını ya da ölçüm yapılan kanal sayısını, $M$ ise her bir serinin uzunluğunu ya da alınan örnek sayısını gösteriyor olmak üzere, yapılan ölçümün sonucunun $NxM$ boyutlu $X$ matrisinde saklandığı kabul edilsin. $N$ satır ve $N$ sütundan oluşan $A$ lineer transformasyon matrisi, $N$ satır ve $M$ sütundan oluşan $S$ sinyal matrisi olmak üzere $X$ matrisi Eş. 1'de verildiği üzere tanımlanabilir [10].

$$X = A\ x\ S \tag{1}$$

$S$ matrisi EEG sinyalini temsil eden matris olarak tanımlandığı için bu matrisin toplamları yine $S$ matrisine eşit ve $S$ matrisi ile aynı boyutlarda olan $S_1$ ve $S_2$ matrislerine ayrılması gerekir [10]. $S_1$ matrisi $S$ matrisinin gürültüden ayrılmış bölümünü, $S_2$ matrisi ise $S$ matrisinin gürültüye karşılık gelen bölümünü temsil etmektedir. Ayrıca $S_1$ ve $S_2$ matrisleri, $A$ matrisi ile ayrı ayrı çarpıldıktan sonra toplanır ise sonuç yine $X$ matrisine eşit bulunmalıdır. $S$, $S_1$, $S_2$, $A$ ve $X$ matrisleri arasındaki ilişki Eş. 2 ve Eş. 3 üzerinden kolaylıkla anlaşılabilir [10].

$$S\ =\ S_1\ +\ S_2 \tag{2}$$

$$X = A\ x\ S_1 +\ A\ x\ S_2 \tag{3}$$

$S$, $S_1$ ve $S_2$ matrisleri arasındaki ilişki basit eşitlikler ile gösterilebiliyor olmasına karşın, $S$ matrisinin en uygun şekilde $S_1$ ve $S_2$ matrislerine ayrılması için belirli bir yöntem bulunmamaktadır. Ancak Goh vd. tahmin edilecek $S_1$ için $X$, $A$ ve yine $S_1$ matrisinden faydalanarak Eş. 4'te verildiği gibi hesaplanabilecek ve $C$ ile gösterilen Pearson koreleasyon katsayılarının kullanılabileceğini tespit etmiştir [10]. Eş. 4'te $covar(X, A\ x\ S_1)$, $X$ matrisi ile $A$ ve $S_1$ matrislerinin çarpım sonucu kullanılarak hesaplanan kovaryans matrisine, $var(X)$, $X$ matrisinin varyansına, $var(\ A\ x\ S_1)$, ise $A$ ve $S_1$ matrislerinin çarpım sonucunun varyansına karşılık gelmektedir [10].

$$C = \frac{covar(X, A\ x\ S_1)}{var(X)\ x\ var(\ A\ x\ S_1\ )} \tag{4}$$

Tahmin edilen $S_1$ matrisinin, $S$ matrisinin gürültüden filtrelenmiş karşılığına uygunluğunu $C$ korelasyon katsayılarından faydalanılarak belirlenebilir [10]. $C$ korelasyon katsayı matrisinin köşegen elemanlarının değerleri tanımlı oldukları uzayda maksimize edilirken diğer elemanlarının minimizasyonu gerçekleştirilmelidir. Aynı zamanda $S$ matrisinin gürültüden filtrelenmiş karşılığı yani $S_1$ matrisi arasındaki uzaklık minimize edilmeye çalışılmaktadır. Diğer bir deyişle $S_1$ matrisi mümkün olduğunca $S$ matrisine benzemelidir. $C$ korelasyon katsayı matrisinin elemanları ile ilgili durumlar dikkate alınarak Eş. 5 ile gösterilen $f_1$ amaç fonksiyonu tanımlanabilir. $S_1$ matrisinin uygunluğuna dair bir diğer kontrol doğrudan $S$ matrisinden faydalanılarak yapılabilir. Daha önce belirtildiği üzere $S$ matrisi $S_1$ ve $S_2$ matrislerinin toplamlarına eşit olarak tanımlanmaktadır. $S_1$ matrisi ihtiyaç duyulan gürültüsüz EEG sinyalini temsil ettiği için orijinal sinyal ölçümüne karşılık gelen $S$ matrisine mümkün olduğu kadar yakın seçilmelidir. Matrisler arasındaki bu ilişki bağlamında Eş. 6 ile gösterilen $f_2$ amaç fonksiyonu tanımlanabilir. Eğer $S_1$ matrisi belirlenmeye çalışılırken $C$ korelasyon katsayı matrisinin köşegen elemanlarının tanımlı oldukları uzayda maksimizasyonu, diğer elemanlarının minimizasyonu hedefleniyor ayrıca $S_1$ matrisi ile $S$ matrisleri arasındaki fark mümkün olduğu kadar azaltılmaya çalışılıyor ise Eş. 5 ve Eş. 6 ile tanımlı fonksiyonların birlikte minimizasyonunun gerektiği kolaylıkla anlaşılır ve farklı meta-sezgisel algoritmalar ile $S_1$ matrisi araştırılabilir.

$$f_1 = \left(\frac{1}{N^2 - N}\right)\sum_i\sum_{j\neq i} C_{ij}^2 + \left(\frac{1}{N}\right)\sum_i (1 - C_{ii})^2 \tag{5}$$

$$f_2 = \left(\frac{1}{NxM}\right)\sum_i\sum_j (S_{ij} - S_{1_{ij}})^2 \tag{6}$$

### *2.2. Yapay Arı Koloni Algoritması (Artificial Bee Colony Algorithm)*

ABC algoritması, bal arılarının yiyecek kaynağı arama davranışlarından esinlenerek geliştirilen ve çok boyutlu nümerik problemlerin en uygun çözümünü arayan, sürü zekâsına dayalı bir en iyileme algoritmasıdır [26]. Literatürde, ABC algoritmasının çok sayıda farklı en iyileme problemlerinin çözümleri için kullanıldığı ve iyi bilinen meta-sezgisel algoritmalarla karşılaştırıldığında başarılı sonuçlar elde edildiği görülmüştür [27-30]. ABC algoritmasında bir yiyecek kaynağının pozisyonu, problem için mümkün olan bir çözümü; yiyecek kaynağının nektar miktarı, çözümün kalitesini ifade etmektedir.

ABC algoritmasında işçi arılar, gözcü arılar ve kâşif arılar olmak üzere üç farklı göreve sahip arı bulunmaktadır. İşçi arılar yuvadan çıktıklarında hafızalarında belirli bir yiyecek kaynağı vardır ve yuvaya döndüklerinde faydalandıkları yiyecek kaynakları ile ilgili gözcü arılar ile bilgi paylaşımında bulunurlar. Gözcü arılar, işçi arılar tarafından paylaşılan bilgiyi değerlendirerek yiyecek kaynağının nektar miktarına göre gidecekleri yiyecek kaynağına karar verirler. Kâşif arılar ise diğer arı türlerinden farklı olarak, bilgi paylaşımında bulunmaksızın yeni yiyecek kaynağı ararlar. Yiyecek kaynaklarını bulduktan sonra çalışmalarına işçi arı olarak devam ederler.

ABC algoritmasının başlangıç aşamasında, kolonideki tüm arılar kâşif arı durumundadır. Algoritma en iyileme işlemine problemin olası çözümlerine karşılık gelen her biri $D$ parametreli popülasyondaki toplam $SN$ adet farklı besin kaynağının üretilmesi ile başlar. $x_j^{min}$ ve $x_j^{max}$, algoritmanın $i$'nci çözümünün $j$'nci parametresine karşılık gelen $x_{ij}$ için atanabilecek değerlerin alt ve üst sınırlarını gösteriyor olmak üzere, $x_{ij}$ Eş. 7'de verildiği üzere hesaplanabilir. Eş. 7'de *rand(0,1)* 0 ile 1 arasında normal dağılıma bağlı rastgele üretilen bir reel sayıya karşılık gelmektedir.

$$x_{ij} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right)$$
$$j = 1, 2, \dots, D\ ve\ i = 1, 2, \dots, SN \tag{7}$$

Yiyecek kaynaklarının nektar miktarı, başka bir deyişle kalitesi uygunluk fonksiyonuyla değerlendirilir. İşçi arılar hafızalarındaki yiyecek kaynağının daha iyisini başka bir yiyecek kaynağından da yararlanarak, Eş. 8 ile komşuluk araştırması yaparak bulmaya çalışırlar.

$$v_{ij} = x_{ij} + \emptyset(x_{ij} - x_{kj}) \quad (8)$$

Eş. 8'de, $v_{ij}$, j'nci parametresi dışında diğer tüm parametrelerinin değerleri $x_i$ çözümü ile aynı olan $v_i$ çözümünün j'nci parametresini göstermektedir. $x_{ij}$ ve $x_{kj}$, $i$ ve $k$ birbirinden farklı olmak üzere, sırasıyla $x_i$ ve $x_k$ kaynaklarının $j$'nci parametrelerinin değerleri göstermektedir. Son olarak, $\emptyset$ ise [-1,1] aralığında rastgele üretilen bir reel sayıya karşılık gelmektedir. Önerilen aday çözüm üretme yaklaşımından da kolaylıkla anlaşılacağı üzere, ABC algoritması mevcut kaynak komşuluğunda yeni çözümü tek parametrenin değiştirilmesine bağlı üretecek şekilde bir yaklaşım kullanmaktadır [30].

Uygunluk değeri en yüksek olan bireyi belirlemek için aday çözüm $v_i$ ve mevcut çözüm $x_i$ arasından açgözlü seleksiyon uygulanır. Uygunluk değeri $fit(x_i)$'nin belirlenmesinde Eş. 9'dan yararlanılır.

$$fit(x_i) = \begin{cases} \dfrac{1}{1+f(x_i)}, f(x_i) \geq 0 \\ 1 + |f(x_i)|, f(x_i) < 0 \end{cases} \quad (9)$$

Eşitlikteki $f(x_i)$, $x_i$'inci çözümün uygunluk fonksiyonu değerini ifade eder. İşçi arılar, yiyecek kaynaklarının kalitesi hakkındaki bilgileri gözcü arılarla paylaşırlar. Gözcü arılar yiyecek kaynaklarının kalitesine bağlı olarak, Eş. 10 ile elde edilen olasılıklarla, yiyecek kaynağı seçer ve işçi arılar aşamasında olduğu gibi Eş. 8'i kullanarak seçilen kaynağı geliştirir.

$$p(x_i) = \frac{fit(x_i)}{\sum_{i=1}^{SN} fit(x_i)} \quad (10)$$

İşçi ve gözcü arıların aday çözüm üretmek için kullandığı matematiksel modelde mevcut kaynaktan faydalanma karakteristiğinin yeni çözüm üretme karakteristiğine kıyasla daha baskın olduğu görülür [30]. Ancak, bir meta-sezgisel algoritmaların başarısı algoritmanın yeni çözümler üretme ve mevcut çözümlerden faydalanma operasyonlarını dengeli yürütebilmesine bağlıdır [31]. ABC algoritmasında bahsedilen temel operasyonlar arasındaki hassas denge $limit$ adı verilen bir kontrol parametresinin kullanımı ile sağlanır. Bu parametre ile kâşif arı aşamasında tüm yiyecek kaynaklarının tükenip tükenmediği kontrol edilir. Her bir kaynak için iyileştirme denemelerinin sayısı tutulur ve her bir iterasyonda denemelerin sayısının $limit$ parametre değerini aşıp aşmadığı kontrol edilir. Aştıysa yiyecek kaynağı tükenmiş kabul edilerek kaynak terk edilir ve o kaynağın işçi arısı kâşif arıya dönüşerek rasgele üretilen yeni bir kaynakla çalışmasına devam eder. $limit$ parametresinin değeri için ABC algoritması katı kurallar belirlememiş olmasına karşın, $a$ bir pozitif reel sayı olmak üzere popülasyon büyüklüğü ve parametre sayısından faydalanarak $[a\ x\ SN\ x\ D]$ şeklinde tanımlanmış işlem sonucundan faydalanılabilir.

### *2.3. Kafes Temelli ABC Algoritması (Lattice Based ABC Algorithm)*

Başlangıç çözümlerinin üretilmesi, işçi, gözcü, kâşif arı fazlarında aday çözümlerin belirlenmesi ve son olarak uygunluk ve olasılık hesaplamalarında kullanılan matematiksel modelleri incelendiğinde, ABC algoritmasının BDO problemlerinin çözümünde mevcut yapısı ile kullanılabileceği görülmektedir. Bir çözüm veya besin kaynağı tahmin edilecek $S_1$ matrisini temsil edebilir ve parametre sayısı, $S_1$ matrisinin eleman sayısı ile eşleştirilebilir. Ancak Goh vd. başlattığı çalışmalar, sürü zekası temelli algoritmalar veya evrimsel algoritmaların standart uygulamalarının, EEG sinyaline gürültü minimizasyonu gerektiren BDO problemleri için başarılı sonuçlar üretemeyeceğini gösterdi [10]. Bu nedenle, ABC algoritması bahsedilen BDO probleminin çözümü için kullanılmadan önce, probleme ve özelliklerine göre değiştirilmeli ya da yeni mekanizmalar ile güçlendirilmelidir.

EEG sinyallerinin problem ile birlikte dikkate alınması gerektiğinden yola çıkılarak ABC algoritmasında yapılacak ilk değişikliğin başlangıç çözümlerinin üretim mekanizmasında olabileceği düşünülebilir. ABC algoritmasının standart uygulamasında ve neredeyse tüm varyantlarında, çözüme karşılık gelen besin kaynaklarının elemanların ilk değerleri verilen alt ve üst sınırlar içinde rastgele belirlenir. Başlangıç çözümlerinin kalitelerini artırmak amacıyla D4 ve D4N problemlerinde 1024 (4×256) parametre, D12 ve D12N problemlerinde 3072 (12 × 256) parametre, D19 ve D19N problemlerinde ise 4864 (19 × 256) parametre olduğu dikkate alındığında başlangıç çözümlerinin ilk elemanlarına atanacak değerlerin dikkatle seçilmesi gerektiği anlaşılır. En iyileme problemlerinin büyük çoğunluğu için, başlangıç parametre değerlerini belirlemenin kolay bir yolu yoktur. Ancak dikkate alınan BDO probleminde farklı meta-sezgisel yöntemlerle bulunmaya çalışılan uygun $S_1$ matrisi özgün $S$ matrisinin bir parçasıdır ve $S$ matrisinden yola çıkılarak başlangıç çözümleri üretilebilir. Majdouli vd. $S$ ve $S_1$ matrisleri arasında bahsedilen ilişkiyi göz önünde bulundurarak basit ve etkili bir başlatma şeması sunmuşlardır [12]. Önerilen üretim modelinde, BDO problemleri için başlangıç çözümlerinin üretilmesi Algoritma 1'de görüldüğü üzere ilk çözümün tüm parametrelerine doğrudan $S$ matrisinin karşılık gelen elemanları atanır ve daha sonra parametreler ($-10^{-5}$) ile ($10^{-5}$) arasında üretilen rastgele sayılar eklenerek değiştirilir [12].

*Algoritma 1*

1. $SN \leftarrow$ çözümlerin sayısı
2. $N \leftarrow S$ matrisindeki zaman serilerinin sayısı
3. $M \leftarrow$ zaman serisindeki örnek sayısı
4. **for** $i \leftarrow 1...SN$ **do**
5. **for** $j \leftarrow 1...N$ **do**
6. **for** $k \leftarrow 1...M$ **do**
7. $rnd \leftarrow (-10^{-5})$ ve $(10^{-5})$ arasında rastgele bir sayı
8. $x_{i,j,k} \leftarrow S_{j,k} + rnd$
9. **end for**
10. **end for**
11. **end for**

ABC algoritmasında işçi arılarının çalışma modeli incelendiğinde, komşuluk araştırmasının sadece bir parametrenin değiştirilmesiyle elde edilen aday çözümün üzerinden yapıldığı anlaşılır. Bu tür aday üretim şeması, ele alınan yiyecek kaynağının çevresinde daha hassas bir arama sağlasa da, sınırlı değişim etkisi özellikle BDO problemlerindeki gibi çok sayıda parametreye sahip problemler için yakınsama performansını düşürebilir. Standart ABC algoritmasının aday üretme modelinden kaynaklanan bu sorunu çözmek için işçi arı fazı, Zhong vd. tarafından önerilen komşuluk rekabeti ve çaprazlama operatörleri ile destekli kafes temelli organizasyon yaklaşımı ile güçlendirilmiştir [32]. Kafes temelli yaklaşımın nasıl çalıştığının anlaşılabilmesi için $L$ ile $L_w\ x\ L_h$ boyutlu bir kafes ortamı olduğu ve $SN$ besin kaynağının bulunduğunu varsayalım. Ayrıca, $SN$ adet yiyecek kaynaklarının tamamının $L$ kafesindeki ilgili pozisyonlara yerleştirilmesini sağlamak için sırasıyla $L_w$ ve $L_h$ ile gösterilen genişlik ve yükseklik uygun şekilde belirlendiği kabul edilsin. Eğer işçi arı fazında $x_i$ yiyecek kaynağı, $L$ kafesinin $L_{r,c}$ ile gösterilen $r$'nci satırının $c$'nci sütununa atanmış ise bu kaynağın kafes temelli komşuluk kümesi $N_{r,c} = \{L_{next(r),c}, L_{r,next(c)}, L_{prev(r),c}, L_{r,prev(c)}\}$ ile gösterilir ve $next(r)$, $prev(r)$, $next(c)$ ve $prev(c)$ Eş. 11'de tanımlandığı gibi belirlenir [32].

$$\left.\begin{array}{l} next(r) = \begin{cases} r+1; & eğer\ r \neq L_h \\ 1; & değilse \end{cases} \\ prev(r) = \begin{cases} r-1; & eğer\ r \neq 1 \\ L_h; & değilse \end{cases} \\ next(c) = \begin{cases} c+1; & eğer\ r \neq L_w \\ 1; & değilse \end{cases} \\ prev(r) = \begin{cases} c-1; & eğer\ c \neq 1 \\ L_w; & değilse \end{cases} \end{array}\right\} \quad (11)$$

Komşuluk kümesini belirledikten sonra $x_i$ yiyecek kaynağı ile ilişkili işçi arı komşuluk rekabet operatörünü kullanır ve $x_i$ yiyecek kaynağının kalitesini $L_{next(r),c}$, $L_{r,next(c)}$, $L_{prev(r),c}$ ve $L_{r,prev(c)}$ pozisyonlarında depolanan yiyecek kaynaklarının kaliteleri ile karşılaştırmaya başlar [32]. Eğer $x_i$ yiyecek kaynağının kalitesi, $N_{r,c}$ komşuluk kümesindeki yiyecek kaynaklarının kalitesinden daha iyiyse, $x_i$ yiyecek kaynağı kafeste kalabilir ve işçi arı Eş. 12 ile tanımlanan komşuluk çaprazlama operatörünü uygular ve $x_i$ çözümünün tüm parametrelerini değiştirerek aday çözüm üretir. Eş. 12'de $x_{max}$ çözümü $N_{r,c}$ komşuluk kümesindeki en kaliteli yiyecek kaynağını, $v_{i,j}$ ise $v_i$ aday çözümünün j'nci parametresini temsil etmektedir.

$$v_{i,j} = x_{i,j} + \emptyset\left(x_{i,j} - x_{max,j}\right), \quad j = \{1,2,\ldots,D\} \quad (12)$$

$x_i$ yiyecek kaynağının uygunluk değeri $N_{r,c}$ komşuluk kümesindeki en iyi çözüm olan $x_{max}$ çözümünün uygunluk değerinden yüksek değilse $x_i$ çözümü komşuluk rekabetini kaybeder ve kafes konumuna $x_{max}$ geçer. Başka bir deyişle, $x_i$ çözümü $x_{max}$ ile değiştirilir ve ilgili işçi arı $x_{max}$ çözümünü hafızaya alır ve ardından Eş. 13 kullanarak $x_{max}$ komşuluğunda bir aday çözüm üretilir.

$$v_{max,j} = x_{max,j} + \emptyset\left(x_{max,j} - x_{i,j}\right), \quad j = \{1,2,\ldots,D\} \quad (13)$$

Daha önce de bahsedildiği üzere gözcü arılar işçi arılar tarafından sunulan yiyecek kaynaklarını seçerler ve aday çözüm üretmeye başlarlar. Gözcü arıların kaliteli yiyecek kaynaklarını seçmek olasılıkları yüksek olup ABC algoritması bu davranışı göz önünde bulundurularak tasarlansa da, bir gözcü arının kalite değeri zayıf kaynağı seçme olasılığı ihmal edilmemeli ve seçim verimliliğini artırmak için daha deterministik bir şema yürütülmelidir. Bu kapsamda gözcü arıları seçtikleri besin kaynaklarına yönlendirmek yerine, Algoritma 2'de açıklandığı gibi mevcut değerlendirmeye kadar gözcü arıları bulunan en iyi besin kaynağına gönderme yaklaşımı tercih edilebilir [25]. ABC algoritmasının gözcü arı fazında bu tür bir seçim şeması kullanılarak algoritmanın fazlarına ek bir yerel arama yöntemi eklenmeden en iyi besin kaynağının komşuluğu daha verimli bir şekilde aranmış olur.

Algoritma 2'te açıklandığı gibi gözcü arı fazında yapılacak değişiklikler bulunan en iyi yiyecek kaynağının çevresini aramaya yardımcı olur. Gözcü arıların tümünün bahsedilen bu kaynağın yalnızca bir parametresini değiştirerek ürettikleri aday çözümler ABC algoritmasının gözcü arı fazının mevcut kaynakları kullanma yeteneğini artırmaktadır. ABC algoritmasının Algoritma 2'te verildiği gibi düzenlenen gözcü arı fazının verimliliği, aday çözüm üretmek için kullanılan Eş. 8'de yapılacak bir değişiklikle daha da arttırılabilir. Eş. 8 incelendiğinde aday için değiştirilen parametrenin, seçilen çözümlerin karşılık gelen parametrelerinin farkı ile doğrudan ilişkili olduğu kolaylıkla görülmektedir. Buna ek olarak, parametrelerin farkı değeri [-1,1] aralığında değişen rastgele bir sayı ile ağırlıklandırılır. ABC algoritmasının fark temelli aday üretme modeli, problemlerin büyük çoğunluğu için uygunluğunu kanıtlasa da, yeni kullanılacak başlangıç çözüm üretme şeması fark temelli aday üretme modeli mekanizmanın etkisini sınırlandırabilir.

*Algoritma 2*

1. *current* ← 1
2. *numOfOnlookerBees* ← 0
3. $x_b$ ← ilgili iterasyona kadar bulunan en iyi çözüm
4. Eş. 10 kullanarak olasılıkları hesapla
5. **while** *numOfOnlookerBees* ≠ $SN$ **do**
6. **if** $p(x_{current}) > rand(0,1)$ **then**
7. *numOfOnlookerBees* ← *numOfOnlookerBees*+1
8. $v_b \leftarrow x_b$
9. $-PM$ ve $PM$ arasında rastgele ∅ sayısı üret
10. $v_{b,j} \leftarrow x_{b,j} + \emptyset(x_{b,j} - x_{current,j})$
11. **if** $fit(v_b) > fit(x_b)$ **then**
12. $x_b$ çözümü ile $v_b$ çözümünü değiştir
13. **end if**
14. **end if**
15. *current* ← ( *current* + 1) mod $SN$
16. **end while**

Algoritmanın başlangıç aşamasında, $S$ matrisinin elemanlarına ($-10^{-5}$) ile ($10^{-5}$) arasında rastgele üretilmiş sayılar eklenerek ilk çözümlerin üretileceğinden bahsedilmişti. Bu başlangıç mekanizmasının bir sonucu olarak, iki çözümün aynı parametreleri arasındaki fark oldukça küçük bulunur. Bu küçük fark, [-1,1] arasında değişen rastgele bir sayı ile çarpıldığında, Eş. 8'deki aday üretme modeli etkisini kaybeder ve algoritmanın yakınsama hızını azaltabilir. $x_{ij}$ ve $x_{kj}$ parametreleri arasındaki fark, büyüklüğü varsayılan konfigürasyona göre daha geniş bir aralıkta adaptif olarak belirlenen bir katsayı ile çarpılırsa değişim miktarı daha hassas belirlenebilir. Bu amaç için, ∅ katsayısının alt ve üst sınırlarını $-1$ ve 1 olarak belirlemek yerine, değeri adaptif olarak ayarlanan $PM$ değerinden faydalanarak $[-PM, PM]$ olarak belirlenmesinin daha uygun olacağı düşünülmektedir [33]. $PM$ parametresini ilk değerini atadıktan sonra yeni değerlerinin adaptif olarak ayarlanması için Rechenberg 1/5 mutasyon kuralı kullanılabilir. Rechenberg 1/5 mutasyon kuralı için, dikkate alınan parametrelerin yeni değeri, başarılı değişimlerin tüm değişimlere oranına göre belirlenir. Eş. 14, Rechenberg 1/5 mutasyon kuralının her $m$ iterasyonu sonunda $PM$ değerini nasıl değiştirdiğini açıklar [33]. $\varphi(m)$ ile gösterilen oran 1/5 değerine eşitse, $PM$ değişmeden bırakılır. Aksi takdirde, $PM$ değeri sonraki gözcü arı fazı için artırılır veya azaltılır.

$$PM(m+1) = \begin{cases} PM(m) * 0.85, & \varphi(m) < 1/5 \\ PM(m) \,/\, 0.85, & \varphi(m) > 1/5 \\ PM(m), & \varphi(m) = 1/5 \end{cases} \quad (14)$$

BDO problemlerinin çözümünde kullanılacak yeni ABC algoritmasında ilk yiyecek kaynakları Algoritma 1'de açıklanan yöntemden yararlanılarak üretilir. Buna ek olarak, önerilen ABC algoritmasının işçi arı fazı kafes temelli komşuluk yaklaşımı, komşuluk rekabeti ve çaprazlama operatörleri ile desteklenmiştir. Son olarak, önerilen ABC algoritmasının gözcü arıları, Algoritma 2'te özetlenen işlemler kullanılarak bulunan en iyi yiyecek kaynağına gönderilir ve değişimin miktarı Rechenberg 1/5 mutasyon kuralından faydalanılarak kontrol edilir. Bu çalışmanın ilerleyen bölümlerinde ABC algoritması temelli yeni BDO yöntemi Kafes temelli ABC (LBABC) olarak adlandırılacaktır.

## 3. Sonuçlar ve Tartışmalar (Results and Discussions)

Deneysel çalışmaların ilk bölümünde $limit$, $SN$ ve $MCN$ gibi parametrelere farklı değerler atanarak LBABC algoritmasının performansı detaylı olarak incelenmiştir. Deneysel çalışmaların ikinci bölümü, LBABC algoritması ile ABC, gbest-ABC, ABC/best/1, ABC/best/2, CABC, COABC ve qABC algoritmaları arasındaki

karşılaştırmalara ayrılmıştır. Deneysel çalışmaların üçüncü bölümünde, LBABC algoritması aralarında DE, GA, FA, FW, PBO ve PSO temelli BDO yöntemlerinin bulunduğu bir dizi meta-sezgisel yöntem ile eşit koşullarda elde edilen sonuçlar kullanılarak karşılaştırılmıştır.

### *3.1. LBABC Algoritmasının Çözme Yeteneklerini Analizi (Analyzing Solving Capabilities of LBABC Algorithm)*

LBABC algoritması ile elde edilen çözümlerin niteliklerinin $limit$ parametresi ile nasıl değiştiğine karar vermek için $(0{,}10 \times N \times M)$, $(0{,}25 \times N \times M)$, $(0{,}50 \times N \times M)$ ve $(0{,}75 \times N \times M)$ olmak üzere dört farklı değer kullanılmıştır. LBABC algoritması 2 ile 15 arasında değişen farklı $PM$ değerleri ile test edildikten sonra $PM$ için uygun başlangıç değeri 10 olarak bulunmuş ve tüm denemelerde bu değer kullanılmıştır. Gerektiğinde $PM$ değerini güncellemek için her iterasyonun sonunda Rechenberg 1/5 mutasyon kuralı uygulanmıştır. LBABC algoritması için koloni büyüklüğü 100 olarak belirlenmiştir. Başka bir deyişle, $SN$ parametresine 50 değeri atanmıştır. Kafes genişliği ve yüksekliği sırasıyla 5 ve 10 olarak belirlenmiştir. Algoritmanın kısa, orta ve uzun dönem yanıtlarını araştırmak için $MCN$ (Maximum Cycle Number) sırasıyla 300, 400, 500 ve 1000 olarak kabul edilmiştir. D4, D4N, D12, D12N, D19 ve D19N problemi, $limit$ ve $MCN$ değerleri ile ilgili her bir kombinasyon için 30 bağımsız koşma gerçekleştirilmiştir. 30 koşmanın en iyi, ortalama en iyi amaç fonksiyonu değerleri ve standart sapmaları kaydedilmiş ve Tablo 1- Tablo 4'te gösterilmiştir. Tablo 1-Tablo 4'te verilen sonuçlar incelendiğinde, LBABC algoritmasının $limit$ parametresinin farklı değerleri için oldukça yakın çözümler üretebildiği açıkça görülmektedir.

LBABC ile bulunan çözümlerin değerleri farklı *limit* konfigürasyonları için birbirine yakın olsa da, problem ve $MCN$ değerine bağlı seçilmesi gerektiği yine ilgili tablolar incelendiğinde anlaşılabilir. D4 problemi için $MCN$ 300, 400 veya 1000 alındığında $limit$ değerinin $(0{,}10 \times N \times M)$ olarak belirlenmesi uygun iken, D19N

**Tablo 1.** 300 iterasyon için LBABC algoritmasının sonuçları (Results of the LBABC algorithm for 300 iterations)

| Pr. | | limit | | | |
|---|---|---|---|---|---|
| | | $0{,}10 \times N \times M$ | $0{,}25 \times N \times M$ | $0{,}50 \times N \times M$ | $0{,}75 \times N \times M$ |
| D4 | Ort. | $6{,}44068\times10^{-2}$ | $6{,}44804\times10^{-2}$ | $6{,}46049\times10^{-2}$ | $6{,}46981\times10^{-2}$ |
| | En | $6{,}35853\times10^{-2}$ | $6{,}37185\times10^{-2}$ | $6{,}39273\times10^{-2}$ | $6{,}35992\times10^{-2}$ |
| | Std. | $3{,}73992\times10^{-4}$ | $3{,}99991\times10^{-4}$ | $3{,}84827\times10^{-4}$ | $1{,}87426\times10^{-3}$ |
| D4N | Ort. | $6{,}26832\times10^{-2}$ | $6{,}23878\times10^{-2}$ | $6{,}22629\times10^{-2}$ | $6{,}22891\times10^{-2}$ |
| | En | $6{,}15838\times10^{-2}$ | $6{,}13046\times10^{-2}$ | $6{,}13149\times10^{-2}$ | $6{,}15765\times10^{-2}$ |
| | Std. | $1{,}96439\times10^{-3}$ | $4{,}86765\times10^{-4}$ | $5{,}05347\times10^{-4}$ | $4{,}25690\times10^{-4}$ |
| D12 | Ort. | $2{,}08872\times10^{-3}$ | $2{,}08885\times10^{-3}$ | $2{,}08210\times10^{-3}$ | $2{,}08695\times10^{-3}$ |
| | En | $2{,}06378\times10^{-3}$ | $2{,}05070\times10^{-3}$ | $2{,}05899\times10^{-3}$ | $2{,}05510\times10^{-3}$ |
| | Std. | $1{,}94821\times10^{-5}$ | $1{,}98324\times10^{-5}$ | $1{,}30234\times10^{-5}$ | $2{,}07823\times10^{-5}$ |
| D12N | Ort. | $1{,}96915\times10^{-3}$ | $1{,}96903\times10^{-3}$ | $1{,}96370\times10^{-3}$ | $1{,}96678\times10^{-3}$ |
| | En | $1{,}94133\times10^{-3}$ | $1{,}93870\times10^{-3}$ | $1{,}94355\times10^{-3}$ | $1{,}94743\times10^{-3}$ |
| | Std. | $1{,}86665\times10^{-5}$ | $1{,}87249\times10^{-5}$ | $1{,}64928\times10^{-5}$ | $1{,}44127\times10^{-5}$ |
| D19 | Ort. | $2{,}69211\times10^{-3}$ | $2{,}69116\times10^{-3}$ | $2{,}69813\times10^{-3}$ | $2{,}69207\times10^{-3}$ |
| | En | $2{,}65999\times10^{-3}$ | $2{,}65630\times10^{-3}$ | $2{,}66175\times10^{-3}$ | $2{,}65986\times10^{-3}$ |
| | Std. | $1{,}95603\times10^{-5}$ | $2{,}19890\times10^{-5}$ | $2{,}34996\times10^{-5}$ | $1{,}98167\times10^{-5}$ |
| D19N | Ort. | $2{,}75223\times10^{-3}$ | $2{,}75354\times10^{-3}$ | $2{,}75049\times10^{-3}$ | $2{,}74816\times10^{-3}$ |
| | En | $2{,}72024\times10^{-3}$ | $2{,}71969\times10^{-3}$ | $2{,}71863\times10^{-3}$ | $2{,}72220\times10^{-3}$ |
| | Std. | $1{,}89272\times10^{-5}$ | $1{,}91687\times10^{-5}$ | $1{,}75228\times10^{-5}$ | $1{,}70970\times10^{-5}$ |

**Tablo 2.** 400 iterasyon için LBABC algnın sonuçları (Results of the LBABC algorithm for 400 iterations)

| Pr. | | limit | | | |
|---|---|---|---|---|---|
| | | $0{,}10 \times N \times M$ | $0{,}25 \times N \times M$ | $0{,}50 \times N \times M$ | $0{,}75 \times N \times M$ |
| D4 | Ort. | $6{,}30080\times10^{-2}$ | $6{,}30091\times10^{-2}$ | $6{,}31224\times10^{-2}$ | $6{,}33371\times10^{-2}$ |
| | En | $6{,}25613\times10^{-2}$ | $6{,}24392\times10^{-2}$ | $6{,}25892\times10^{-2}$ | $6{,}24625\times10^{-2}$ |
| | Std. | $2{,}70125\times10^{-4}$ | $3{,}03946\times10^{-4}$ | $2{,}84130\times10^{-4}$ | $2{,}10573\times10^{-3}$ |
| D4N | Ort. | $6{,}13345\times10^{-2}$ | $6{,}09613\times10^{-2}$ | $6{,}08548\times10^{-2}$ | $6{,}08622\times10^{-2}$ |
| | En | $6{,}03218\times10^{-2}$ | $6{,}02043\times10^{-2}$ | $6{,}01235\times10^{-2}$ | $6{,}05124\times10^{-2}$ |
| | Std. | $2{,}18530\times10^{-3}$ | $3{,}44203\times10^{-4}$ | $3{,}55436\times10^{-4}$ | $2{,}66676\times10^{-4}$ |
| D12 | Ort. | $2{,}07183\times10^{-3}$ | $2{,}07043\times10^{-3}$ | $2{,}06314\times10^{-3}$ | $2{,}06923\times10^{-3}$ |
| | En | $2{,}04396\times10^{-3}$ | $2{,}03387\times10^{-3}$ | $2{,}03887\times10^{-3}$ | $2{,}03591\times10^{-3}$ |
| | Std. | $2{,}60498\times10^{-5}$ | $2{,}54583\times10^{-5}$ | $1{,}62160\times10^{-5}$ | $2{,}56200\times10^{-5}$ |
| D12N | Ort. | $1{,}95344\times10^{-3}$ | $1{,}95285\times10^{-3}$ | $1{,}94656\times10^{-3}$ | $1{,}94936\times10^{-3}$ |
| | En | $1{,}92204\times10^{-3}$ | $1{,}91698\times10^{-3}$ | $1{,}92613\times10^{-3}$ | $1{,}92308\times10^{-3}$ |
| | Std. | $2{,}31482\times10^{-5}$ | $2{,}35183\times10^{-5}$ | $2{,}06591\times10^{-5}$ | $1{,}80362\times10^{-5}$ |
| D19 | Ort. | $2{,}68117\times10^{-3}$ | $2{,}68145\times10^{-3}$ | $2{,}68948\times10^{-3}$ | $2{,}68222\times10^{-3}$ |
| | En | $2{,}64781\times10^{-3}$ | $2{,}64090\times10^{-3}$ | $2{,}65339\times10^{-3}$ | $2{,}65053\times10^{-3}$ |
| | Std. | $2{,}09608\times10^{-5}$ | $2{,}56509\times10^{-5}$ | $2{,}48645\times10^{-5}$ | $2{,}18398\times10^{-5}$ |
| D19N | Ort. | $2{,}74555\times10^{-3}$ | $2{,}74437\times10^{-3}$ | $2{,}74123\times10^{-3}$ | $2{,}73761\times10^{-3}$ |
| | En | $2{,}71037\times10^{-3}$ | $2{,}70672\times10^{-3}$ | $2{,}70798\times10^{-3}$ | $2{,}70358\times10^{-3}$ |
| | Std. | $2{,}20082\times10^{-5}$ | $2{,}33127\times10^{-5}$ | $1{,}97527\times10^{-5}$ | $1{,}93742\times10^{-5}$ |

**Tablo 3.** 500 iterasyon için LBABC algoritmasının sonuçları (Results of the LBABC algorithm for 500 iterations)

| Pr. | | limit | | | |
|---|---|---|---|---|---|
| | | $0{,}10 \times N \times M$ | $0{,}25 \times N \times M$ | $0{,}50 \times N \times M$ | $0{,}75 \times N \times M$ |
| D4 | Ort. | $6{,}24009\times10^{-2}$ | $6{,}23977\times10^{-2}$ | $6{,}24739\times10^{-2}$ | $6{,}27860\times10^{-2}$ |
| | En | $6{,}19782\times10^{-2}$ | $6{,}19802\times10^{-2}$ | $6{,}20932\times10^{-2}$ | $6{,}19925\times10^{-2}$ |
| | Std. | $2{,}18241\times10^{-4}$ | $2{,}30159\times10^{-4}$ | $1{,}98368\times10^{-4}$ | $2{,}20035\times10^{-3}$ |
| D4N | Ort. | $6{,}07352\times10^{-2}$ | $6{,}03311\times10^{-2}$ | $6{,}02404\times10^{-2}$ | $6{,}02996\times10^{-2}$ |
| | En | $5{,}98781\times10^{-2}$ | $5{,}98117\times10^{-2}$ | $5{,}97283\times10^{-2}$ | $5{,}99587\times10^{-2}$ |
| | Std. | $2{,}28734\times10^{-3}$ | $2{,}50838\times10^{-4}$ | $2{,}45574\times10^{-4}$ | $2{,}28843\times10^{-4}$ |
| D12 | Ort. | $2{,}05461\times10^{-3}$ | $2{,}05278\times10^{-3}$ | $2{,}04343\times10^{-3}$ | $2{,}04887\times10^{-3}$ |
| | En | $2{,}02382\times10^{-3}$ | $2{,}01856\times10^{-3}$ | $2{,}01981\times10^{-3}$ | $2{,}01009\times10^{-3}$ |
| | Std. | $3{,}27158\times10^{-5}$ | $3{,}02853\times10^{-5}$ | $1{,}94018\times10^{-5}$ | $3{,}07497\times10^{-5}$ |
| D12N | Ort. | $1{,}93701\times10^{-3}$ | $1{,}93717\times10^{-3}$ | $1{,}92946\times10^{-3}$ | $1{,}93272\times10^{-3}$ |
| | En | $1{,}90686\times10^{-3}$ | $1{,}90469\times10^{-3}$ | $1{,}90416\times10^{-3}$ | $1{,}90303\times10^{-3}$ |
| | Std. | $2{,}78406\times10^{-5}$ | $2{,}83795\times10^{-5}$ | $2{,}50337\times10^{-5}$ | $2{,}09205\times10^{-5}$ |
| D19 | Ort. | $2{,}67251\times10^{-3}$ | $2{,}67331\times10^{-3}$ | $2{,}68184\times10^{-3}$ | $2{,}67223\times10^{-3}$ |
| | En | $2{,}63664\times10^{-3}$ | $2{,}63377\times10^{-3}$ | $2{,}63947\times10^{-3}$ | $2{,}63262\times10^{-3}$ |
| | Std. | $2{,}31079\times10^{-5}$ | $2{,}83170\times10^{-5}$ | $2{,}82055\times10^{-5}$ | $2{,}49071\times10^{-5}$ |
| D19N | Ort. | $2{,}73591\times10^{-3}$ | $2{,}73519\times10^{-3}$ | $2{,}72988\times10^{-3}$ | $2{,}72782\times10^{-3}$ |
| | En | $2{,}68981\times10^{-3}$ | $2{,}69616\times10^{-3}$ | $2{,}69757\times10^{-3}$ | $2{,}69321\times10^{-3}$ |
| | Std. | $2{,}72738\times10^{-5}$ | $2{,}68585\times10^{-5}$ | $2{,}30544\times10^{-5}$ | $2{,}17820\times10^{-5}$ |

**Tablo 4.** 1000 iterasyon için LBABC algoritmasının sonuçları (Results of the LBABC algorithm for 1000 iterations)

| Pr. | | limit | | | |
|---|---|---|---|---|---|
| | | $0{,}10 \times N \times M$ | $0{,}25 \times N \times M$ | $0{,}50 \times N \times M$ | $0{,}75 \times N \times M$ |
| D4 | Ort. | $\mathit{6{,}20211\times10^{-2}}$ | $6{,}20350\times10^{-2}$ | $6{,}20561\times10^{-2}$ | $6{,}24153\times10^{-2}$ |
| | En | $6{,}17231\times10^{-2}$ | $6{,}17240\times10^{-2}$ | $6{,}16799\times10^{-2}$ | $6{,}17008\times10^{-2}$ |
| | Std. | $1{,}69625\times10^{-4}$ | $2{,}04357\times10^{-4}$ | $1{,}66989\times10^{-4}$ | $2{,}26648\times10^{-3}$ |
| D4N | Ort. | $6{,}03600\times10^{-2}$ | $5{,}99393\times10^{-2}$ | $\mathit{5{,}98569\times10^{-2}}$ | $5{,}99600\times10^{-2}$ |
| | En | $5{,}95554\times10^{-2}$ | $5{,}95561\times10^{-2}$ | $5{,}95446\times10^{-2}$ | $5{,}96281\times10^{-2}$ |
| | Std. | $2{,}35415\times10^{-3}$ | $2{,}08908\times10^{-4}$ | $1{,}62569\times10^{-4}$ | $1{,}93703\times10^{-4}$ |
| D12 | Ort. | $1{,}99999\times10^{-3}$ | $1{,}99639\times10^{-3}$ | $\mathit{1{,}98256\times10^{-3}}$ | $1{,}99124\times10^{-3}$ |
| | En | $1{,}96836\times10^{-3}$ | $1{,}96387\times10^{-3}$ | $1{,}96587\times10^{-3}$ | $1{,}96252\times10^{-3}$ |
| | Std. | $5{,}26564\times10^{-5}$ | $4{,}72272\times10^{-5}$ | $2{,}90518\times10^{-5}$ | $4{,}27079\times10^{-5}$ |
| D12N | Ort. | $1{,}88313\times10^{-3}$ | $1{,}88646\times10^{-3}$ | $1{,}87783\times10^{-3}$ | $\mathit{1{,}87712\times10^{-3}}$ |
| | En | $1{,}85413\times10^{-3}$ | $1{,}85692\times10^{-3}$ | $1{,}85891\times10^{-3}$ | $1{,}85687\times10^{-3}$ |
| | Std. | $4{,}14986\times10^{-5}$ | $4{,}39072\times10^{-5}$ | $3{,}72402\times10^{-5}$ | $2{,}89683\times10^{-5}$ |
| D19 | Ort. | $2{,}62212\times10^{-3}$ | $2{,}62797\times10^{-3}$ | $2{,}63456\times10^{-3}$ | $\mathit{2{,}62047\times10^{-3}}$ |
| | En | $2{,}58842\times10^{-3}$ | $2{,}58474\times10^{-3}$ | $2{,}57508\times10^{-3}$ | $2{,}58125\times10^{-3}$ |
| | Std. | $3{,}71829\times10^{-5}$ | $4{,}59113\times10^{-5}$ | $4{,}79888\times10^{-5}$ | $3{,}52186\times10^{-5}$ |
| D19N | Ort. | $2{,}69385\times10^{-3}$ | $2{,}68913\times10^{-3}$ | $2{,}67803\times10^{-3}$ | $\mathit{2{,}67623\times10^{-3}}$ |
| | En | $2{,}63363\times10^{-3}$ | $2{,}63621\times10^{-3}$ | $2{,}63785\times10^{-3}$ | $2{,}63500\times10^{-3}$ |
| | Std. | $4{,}80690\times10^{-5}$ | $5{,}10615\times10^{-5}$ | $4{,}28841\times10^{-5}$ | $3{,}68792\times10^{-5}$ |

problemi için $MCN$ 300, 400, 500 veya 1000 alındığında *limit* değerinin ($0{,}75 \times N \times M$) olarak belirlenmesinin daha uygun olacağı anlaşılmaktadır. $MCN$ değeri 300, 400 veya 500 alındığında LBABC algoritmasının D4N, D12 ve D12N problemleri için uygun $limit$ değerinin ($0{,}50 \times N \times M$) olduğu ortalama en iyi amaç fonksiyonu değerleri dikkate alındığında kolaylıkla anlaşılabilir. $MCN$ değeri 1000 alındığında ise D4N ve D12 problemleri için ($0{,}50 \times N \times M$) olarak belirlenecek $limit$ değerinin diğer üç değere kıyasla LBABC algoritmasının çözüm kapasitesini artırdığı görülür. Tüm bu özel durumlar dikkate alınarak, D4N, D12, D12N, D19 ve D19N problemleri için limit parametre değerinin ($0{,}50 \times N \times M$) *ile* ($0{,}75 \times N \times M$) arasında D4 örneği için ($0{,}10 \times N \times M$) ve ($0{,}25 \times N \times M$) seçilmesi gerektiği şeklinde genelleme yapılabilir.

Çözüm sayısı veya yiyecek kaynakları, diğer popülasyon temelli meta-sezgiseller gibi LBABC algoritması ile bulunacak çözümlerin kalitelerini değiştirebilir. Bu nedenle, $SN$ parametresine 50 değerine ek olarak 20, 30 ve 40 değerleri sırasıyla atanmış ve LBABC algoritması D4, D4N, D12, D12N, D19 ve D19N problemlerinin çözümü amacıyla tekrar test edilmiştir. Yeni koşmalarda, limit parametresi ($0{,}5 \times N \times M$) olarak ayarlanmış ve $MCN$ için 500 değeri kullanılmıştır. $SN$ değeri 20 iken kafes genişliği ve yüksekliği sırasıyla 4 ve 5, $SN$ değeri 30 iken kafes genişliği ve yüksekliği sırasıyla 5 ve 6, $SN$ değeri 40 iken kafes genişliği ve yüksekliği sırasıyla 4 ve 10 olarak belirlenmiştir.

30 bağımsız koşmanın sonucunda en iyi, ortalama en iyi amaç fonksiyon değerleri kaydedilmiş, standart sapma değerleri hesaplanmış ve ilgili sonuçlar Tablo 5 ve Tablo 6'da gösterilmiştir. Tablo 5 ve Tablo 6'da verilen sonuçlar incelendiğinde, LBABC algoritmasının toplam değerlendirme sayıları farklı olmasına rağmen $SN$ değerinin 30 ya da daha büyük seçilmesi durumunda çözüm kalitesini koruyabildiği görülmektedir. LBABC algoritmasında kullanılan başlangıç çözüm üretme yaklaşımı, büyük popülasyonlara ihtiyaç duymadan kaliteli çözümlere ulaşabilmektedir. Ayrıca işçi arı fazında kullanılan kafes temelli yaklaşımı, uygun çözümlerin

**Tablo 5.** 500 iterasyon için farklı $SN$ değerlerine sahip LBABC algoritmasının sonuçları
(Results of the LBABC algorithm with different $SN$ values for 500 iterations)

| Pr. | | $SN$ | | | |
|---|---|---|---|---|---|
| | | 20 | 30 | 40 | 50 |
| D4 | Ort. | $6{,}97462\times10^{-2}$ | $6{,}48237\times10^{-2}$ | $6{,}31646\times10^{-2}$ | *$6{,}24739\times10^{-2}$* |
| | En | $6{,}67339\times10^{-2}$ | $6{,}40130\times10^{-2}$ | $6{,}26750\times10^{-2}$ | $6{,}20932\times10^{-2}$ |
| | Std. | $2{,}96566\times10^{-3}$ | $5{,}04147\times10^{-4}$ | $3{,}06231\times10^{-4}$ | $1{,}98368\times10^{-4}$ |
| D4N | Ort. | $6{,}77035\times10^{-2}$ | $6{,}27660\times10^{-2}$ | $6{,}11300\times10^{-2}$ | *$6{,}02404\times10^{-2}$* |
| | En | $6{,}46498\times10^{-2}$ | $6{,}16332\times10^{-2}$ | $6{,}06142\times10^{-2}$ | $5{,}97283\times10^{-2}$ |
| | Std. | $2{,}94483\times10^{-3}$ | $5{,}18284\times10^{-4}$ | $3{,}39807\times10^{-4}$ | $2{,}45574\times10^{-4}$ |
| D12 | Ort. | $2{,}12551\times10^{-3}$ | $2{,}08923\times10^{-3}$ | $2{,}07188\times10^{-3}$ | *$2{,}04343\times10^{-3}$* |
| | En | $2{,}09178\times10^{-3}$ | $2{,}06335\times10^{-3}$ | $2{,}04195\times10^{-3}$ | $2{,}01981\times10^{-3}$ |
| | Std. | $2{,}02883\times10^{-5}$ | $1{,}30591\times10^{-5}$ | $2{,}33773\times10^{-5}$ | $1{,}94018\times10^{-5}$ |
| D12N | Ort. | $2{,}00455\times10^{-3}$ | $1{,}97666\times10^{-3}$ | $1{,}95098\times10^{-3}$ | *$1{,}92946\times10^{-3}$* |
| | En . | $1{,}97196\times10^{-3}$ | $1{,}95025\times10^{-3}$ | $1{,}92720\times10^{-3}$ | $1{,}90416\times10^{-3}$ |
| | Std. | $1{,}79381\times10^{-5}$ | $1{,}96583\times10^{-5}$ | $1{,}91390\times10^{-5}$ | $2{,}50337\times10^{-5}$ |
| D19 | Ort. | $2{,}72632\times10^{-3}$ | $2{,}70658\times10^{-3}$ | $2{,}70259\times10^{-3}$ | *$2{,}68184\times10^{-3}$* |
| | En | $2{,}69402\times10^{-3}$ | $2{,}66699\times10^{-3}$ | $2{,}65150\times10^{-3}$ | $2{,}63947\times10^{-3}$ |
| | Std. | $1{,}38118\times10^{-5}$ | $1{,}82964\times10^{-5}$ | $2{,}32734\times10^{-5}$ | $2{,}82055\times10^{-5}$ |
| D19N | Ort. | $2{,}77847\times10^{-3}$ | $2{,}76066\times10^{-3}$ | $2{,}74739\times10^{-3}$ | *$2{,}72988\times10^{-3}$* |
| | En | $2{,}74952\times10^{-3}$ | $2{,}72542\times10^{-3}$ | $2{,}71966\times10^{-3}$ | $2{,}69757\times10^{-3}$ |
| | Std. | $1{,}64436\times10^{-5}$ | $1{,}68739\times10^{-5}$ | $1{,}53897\times10^{-5}$ | $2{,}30544\times10^{-5}$ |

**Tablo 6.** 500 iterasyon için LBABC algoritması ve diğer ABC varyantları arasında karşılaştırma
(Comparison between LBABC algorithm and other ABC variants for 500 iterations)

| Pr. | | ABC | GABC | ABC/best/1 | ABC/best/2 | CABC | COABC | qABC | LBABC |
|---|---|---|---|---|---|---|---|---|---|
| D4 | Ort. | $1{,}663\times10^{1}$ | $1{,}526\times10^{1}$ | $1{,}514\times10^{1}$ | $1{,}534\times10^{1}$ | $1{,}232\times10^{1}$ | $8{,}146\times10^{-2}$ | $2{,}090\times10^{-1}$ | $6{,}247\times10^{-2}$ |
| | En | $1{,}614\times10^{1}$ | $1{,}495\times10^{1}$ | $1{,}489\times10^{1}$ | $1{,}500\times10^{1}$ | $1{,}161\times10^{1}$ | $7{,}734\times10^{-2}$ | $1{,}630\times10^{-1}$ | $6{,}209\times10^{-2}$ |
| | Std. | $2{,}676\times10^{-1}$ | $1{,}904\times10^{-1}$ | $1{,}453\times10^{-1}$ | $1{,}803\times10^{-1}$ | $2{,}575\times10^{-1}$ | $2{,}333\times10^{-3}$ | $3{,}416\times10^{-2}$ | $1{,}983\times10^{-4}$ |
| D4N | Ort. | $1{,}652\times10^{1}$ | $1{,}537\times10^{1}$ | $1{,}509\times10^{1}$ | $1{,}537\times10^{1}$ | $1{,}232\times10^{1}$ | $8{,}022\times10^{-2}$ | $2{,}030\times10^{-1}$ | $6{,}024\times10^{-2}$ |
| | En | $1{,}574\times10^{1}$ | $1{,}507\times10^{1}$ | $1{,}469\times10^{1}$ | $1{,}504\times10^{1}$ | $1{,}186\times10^{1}$ | $7{,}558\times10^{-2}$ | $1{,}566\times10^{-1}$ | $5{,}972\times10^{-2}$ |
| | Std. | $3{,}515\times10^{-1}$ | $1{,}530\times10^{-1}$ | $1{,}753\times10^{-1}$ | $2{,}057\times10^{-1}$ | $2{,}657\times10^{-1}$ | $2{,}585\times10^{-3}$ | $2{,}554\times10^{-2}$ | $2{,}455\times10^{-4}$ |
| D12 | Ort. | $2{,}059\times10^{1}$ | $2{,}013\times10^{1}$ | $2{,}012\times10^{1}$ | $2{,}021\times10^{1}$ | $1{,}883\times10^{1}$ | $2{,}467\times10^{-1}$ | $3{,}231\times10^{0}$ | $2{,}043\times10^{-3}$ |
| | En | $2{,}022\times10^{1}$ | $1{,}989\times10^{1}$ | $1{,}991\times10^{1}$ | $2{,}004\times10^{1}$ | $1{,}832\times10^{1}$ | $2{,}118\times10^{-1}$ | $3{,}013\times10^{0}$ | $2{,}019\times10^{-3}$ |
| | Std. | $1{,}967\times10^{-1}$ | $1{,}124\times10^{-1}$ | $9{,}894\times10^{-2}$ | $1{,}077\times10^{-1}$ | $2{,}002\times10^{-1}$ | $1{,}755\times10^{-2}$ | $1{,}201\times10^{-1}$ | $1{,}940\times10^{-5}$ |
| D12N | Ort. | $2{,}062\times10^{1}$ | $2{,}015\times10^{1}$ | $2{,}013\times10^{1}$ | $2{,}018\times10^{1}$ | $1{,}885\times10^{1}$ | $1{,}597\times10^{0}$ | $3{,}226\times10^{0}$ | $1{,}929\times10^{-3}$ |
| | En | $2{,}028\times10^{1}$ | $1{,}997\times10^{1}$ | $1{,}988\times10^{1}$ | $1{,}971\times10^{1}$ | $1{,}852\times10^{1}$ | $1{,}541\times10^{0}$ | $2{,}966\times10^{0}$ | $1{,}904\times10^{-3}$ |
| | Std. | $1{,}558\times10^{-1}$ | $1{,}058\times10^{-1}$ | $1{,}036\times10^{-1}$ | $1{,}540\times10^{-1}$ | $1{,}517\times10^{-1}$ | $3{,}893\times10^{-2}$ | $1{,}324\times10^{-1}$ | $2{,}503\times10^{-5}$ |
| D19 | Ort. | $2{,}148\times10^{1}$ | $2{,}116\times10^{1}$ | $2{,}118\times10^{1}$ | $2{,}122\times10^{1}$ | $2{,}035\times10^{1}$ | $4{,}352\times10^{-1}$ | $6{,}497\times10^{0}$ | $2{,}681\times10^{-3}$ |
| | En | $2{,}119\times10^{1}$ | $2{,}092\times10^{1}$ | $2{,}107\times10^{1}$ | $2{,}092\times10^{1}$ | $1{,}994\times10^{1}$ | $4{,}038\times10^{-1}$ | $6{,}231\times10^{0}$ | $2{,}639\times10^{-3}$ |
| | Std. | $1{,}256\times10^{-1}$ | $1{,}005\times10^{-1}$ | $7{,}313\times10^{-2}$ | $1{,}211\times10^{-1}$ | $1{,}749\times10^{-1}$ | $1{,}743\times10^{-2}$ | $1{,}372\times10^{-1}$ | $2{,}820\times10^{-5}$ |
| D19N | Ort. | $2{,}146\times10^{1}$ | $2{,}115\times10^{1}$ | $2{,}117\times10^{1}$ | $2{,}119\times10^{1}$ | $2{,}037\times10^{1}$ | $4{,}269\times10^{-1}$ | $6{,}506\times10^{0}$ | $2{,}729\times10^{-3}$ |
| | En | $2{,}127\times10^{1}$ | $2{,}089\times10^{1}$ | $2{,}102\times10^{1}$ | $2{,}089\times10^{1}$ | $2{,}012\times10^{1}$ | $3{,}811\times10^{-1}$ | $6{,}256\times10^{0}$ | $2{,}697\times10^{-3}$ |
| | Std. | $1{,}247\times10^{-1}$ | $1{,}137\times10^{-1}$ | $8{,}110\times10^{-2}$ | $1{,}087\times10^{-1}$ | $1{,}324\times10^{-1}$ | $1{,}994\times10^{-2}$ | $1{,}185\times10^{-1}$ | $2{,}305\times10^{-5}$ |

korunmasına yardımcı olur ve kaliteli çözümlerin sayısını artırır. Bahsedilen mekanizmaların beklenen sonucu olarak, LBABC algoritması çözüm sayısı 30 ile 50 arasında seçildiğinde benzer çözümler elde etmektedir.

### *3.2. LBABC Algoritmasının Standart ABC ve Varyantları ile Karşılaştırılması*
*(Comparing LBABC Algorithm with Standard ABC and Its Variants )*

LBABC algoritması ile elde edilen sonuçların ABC algoritması ve GABC [12], ABC/best/1 [13], ABC/best/2 [13], CABC [14], qABC [15] ve COABC [24] gibi başarılı ABC varyantlarına göre daha iyi olup olmadığına karar vermek için farklı bir dizi test gerçekleştirilmiştir. D4, D4N, D12, D12N, D19 ve D19N problemlerinin tamamı bahsedilen ABC algoritmaları ile $SN$ değeri 100, $limit$ değeri $(0{,}50\times N\times M)$ ve $MCN$ değeri 500 olarak belirlenip çözülmüştür. GABC için $C$ sabiti ve qABC için komşuluk yarıçapı sırasıyla 1,5 ve $\infty$ alınmıştır. D4, D4N, D12, D12N, D19 ve D19N problemleri $limit$ ve $MCN$ değerlerinin kombinasyonları için ABC algoritmaları 30 bağımsız sefer koşturulmuş, 30 koşmanın sonucunda en iyi, ortalama en iyi amaç fonksiyon değerleri ve standart sapmaları kaydedilip Tablo 6'da gösterilmiştir.

Tablo 6'da verilen sonuçlar, LBABC algoritmasının problemlerin tamamında ABC algoritması ve diğer varyantlarını en iyi, ortalama en iyi amaç fonksiyon değerleri ve standart sapmalar dikkate alındığına geride bırakabildiğini göstermiştir. Sırasıyla $limit$ ve $MCN$ parametrelerinin $(0{,}50\times N\times M)$ ve 500 olarak ayarlandığı senaryoda, LBABC algoritması D4 problemi için en yakın rakibi COABC algoritmasından ortalama en iyi amaç fonksiyonu değerleri incelendiğinde 1,30 kat ve D4N problemi için ortalama en iyi amaç fonksiyon değerleri incelendiğinde 1,33 kat daha iyi çözümler üretmektedir. LBABC ve COABC algoritmaları arasındaki farklar D12, D12N, D19 ve D19N problemleri incelendiğinde daha net görülmektedir. D12 probleminde LBABC algoritmasının COABC algoritmasından 120,74 kat, D12N probleminde 828,10 kat daha iyi çözümler üretebildiği yine ortalama en iyi amaç fonksiyon değerleri incelendiğinde anlaşılabilir. Majdouli vd. etkinliğini kanıtladığı başlangıç şeması, LBABC algoritmasının başlangıç çözümlerinin kalitesini önemli oranda iyileştirmiştir [12]. Buna ek olarak, LBABC

algoritmasının işçi ve gözcü arı fazları, ABC, GABC, ABC/best/1, ABC/best/2, CABC ve qABC algoritmalarına göre daha başarılı arama performansı göstererek farkın açılmasına katkıda bulunmuşlardır. COABC ve LBABC algoritmalarında olduğu gibi tüm gözcü arıların bulunan en iyi çözüme gönderilmesi ve komşuluğunda aday çözümlerin oluşturulması şeklinde özetlenecek yöntemlerinin, ele alınan BDO problemleri için test edilen diğer ABC algoritmalarının gözcü arı fazlarının aday üretme yaklaşımlarından daha uygun olduğu söylenebilir.

Tablo 6'da verilen sonuçlar, LBABC algoritmasının ABC algoritması ve varyantlarına kıyasla oldukça başarılı olduğunun anlaşılması için yeterli delil sağlar. Ancak, LBABC algoritmasının başarımı uygun bir istatistiksel test kullanılarak ayrıca kanıtlanmalıdır. Bu amaçla, yaygın kullanılan testlerden biri olan Wilcoxon Signed Rank testi seçilmiş ve $\rho$ ile gösterilen anlamlılık düzeyi 0,05 olarak ayarlanmıştır [24]. İki algoritma için test ile hesaplanan $\rho$ 0,05 değerinden küçükse, algoritmalar arasındaki farkın hangisinin daha iyi olduğuna karar vermek için yeterli olduğu söylenir. Aksi halde iki algoritma arasındaki farkın algoritmalardan birinin lehinde karar vermek için yeterli olmadığı anlaşılır. Wilcoxon Signed Rank testi için, $limit$ parametresinin $(0{,}50 \times N \times M)$ ve $MCN$ parametresinin 500 olarak alındığı 30 bağımsız koşmanın her birinde bulunan en iyi çözümler kullanılmış ve test sonuçları Tablo 7'de verilmiştir. Tablo 7'de verilen test sonuçları incelendiğinde LBABC ile diğer varyantlar arasındaki farkın tüm örnekler için LBABC algoritması lehine olduğu anlaşılmaktadır. 0,000002 olarak hesaplanan $\rho$ değeri LBABC algoritmasının ABC, GABC, ABC/best/1, ABC/best/2, CABC, COABC ve qABC algoritmalarına göre 30 farklı koşmanın tamamına yakınında daha iyi çözümler elde ettiği hususunda ayrıca bilgi vermektedir.

### *3.3. LBABC Algoritmasının Diğer Metasezgiseller ile Karşılaştırılması*
*(Comparing LBABC Algorithm with Other Metaheuristics)*

LBABC algoritması ile elde edilen çözümlerin diğer evrimsel veya sürü zekâsı temelli BDO yöntemlerinin çözümleriyle rekabet edip edemediğini analiz etmek için Tablo 8 ve Tablo 9'da verilen sonuçlar değerlendirilebilir. LBABC ile NSGA-II [1], FAF-SOFWA [12], CC-HDE [13], HDE [13], ACDE [13], SanSDE-CC [13], JADE [13], SHADE [13], HMOFA [14], MOFA [14], PSO [34] ve PBO [34] algoritmalarının en iyi, ortalama en iyi amaç fonksiyon değerleri ve standart sapmaları Tablo 8' ve Tablo 9'da özetlemiştir. Her iki tabloda verilen sonuçlar elde edilirken LBABC, NSGA-II, CC-HDE, HDE, ACDE, SanSDE-CC, JADE, SHADE, HMOFA ve MOFA için popülasyon veya koloni büyüklüğü 100 ve $MCN$ 1000 olarak alınmıştır. Koloni büyüklüğü ve maksimum iterasyon sayıları dikkate alınarak, toplam değerlendirme sayısı (100×1000) olarak hesaplanabilir. NSGA-II için ikinci amaç fonksiyonunun değeri sıfıra alınarak BDO problemi tek amaçlı hale getirilmiştir [1]. PSO ve PBO algoritmalarında, popülasyon büyüklüğü 30 olarak belirlenmiş ve maksimum değerlendirme sayısı $(10000 \times D)$ olarak alınmıştır [23]. Burada $D$, D4 ve D4N problemleri için 1024, D12 ve D12N problemleri için 3072 ve D19 ve D19N problemleri için 4864 değerine eşittir. FAF-SOFWA için iterasyon sayısı 10000 olarak belirlenmiştir [12]. LBABC algoritmasının $limit$ değeri, D12N, D19 ve D19N problemleri için $(0{,}75 \times N \times M)$, D4N ve D12 problemleri için

**Tablo 7.** 500 iterasyon için LBABC ve diğer varyantlar arasındaki istatistiksel karşılaştırma
(Statistical comparison between LBABC and other variants for 500 iterations)

| Pr. | ABC ile LBABC | | GABC ile LBABC | | ABC/best/1 ile LBABC | | ABC/best/2 ile LBABC | | CABC ile LBABC | | COABC ile LBABC | | qABC ile LBABC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\rho$ | Anlam. | $\rho$ | Anlam. | $\rho$ | Anlam. | $\rho$ | Anlam. | $\rho$ | Anlam. | $\rho$ | Anlam. | $\rho$ | Anlam. |
| D4 | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |
| D4N | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |
| D12 | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |
| D12N | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |
| D19 | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |
| D19N | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC | <0.05 | LBABC |

**Tablo 8.** 1000 iterasyon için LBABC algoritmasının DE temelli yöntemler ile karşılaştırılması
(Comparing LBABC algorithm with DE based techniques for 1000 iterations)

| Pr. | | LBABC | CC-HDE [8] | HDE [8] | ACDE [8] | SaNSDE-CC [8] | JADE [8] | SHADE [8] |
|---|---|---|---|---|---|---|---|---|
| D4 | Ort. | $6{,}20\times10^{-2}$ | $6{,}11\times10^{-2}$ | $6{,}54\times10^{-2}$ | $6{,}55\times10^{-2}$ | $5{,}11\times10^{0}$ | $2{,}70\times10^{0}$ | $3{,}22\times10^{0}$ |
| | En | $6{,}17\times10^{-2}$ | $6{,}10\times10^{-2}$ | $6{,}32\times10^{-2}$ | $6{,}13\times10^{-2}$ | $3{,}28\times10^{0}$ | $1{,}81\times10^{0}$ | $1{,}42\times10^{0}$ |
| | Std. | $1{,}69\times10^{-4}$ | $3{,}00\times10^{-6}$ | $1{,}21\times10^{-5}$ | $3{,}01\times10^{-6}$ | $1{,}57\times10^{-1}$ | $4{,}76\times10^{-1}$ | $7{,}23\times10^{-1}$ |
| D4N | Ort. | $5{,}98\times10^{-2}$ | $5{,}91\times10^{-2}$ | $7{,}87\times10^{-2}$ | $5{,}93\times10^{-2}$ | $1{,}11\times10^{1}$ | $9{,}45\times10^{0}$ | $4{,}19\times10^{0}$ |
| | En | $5{,}95\times10^{-2}$ | $5{,}90\times10^{-2}$ | $6{,}01\times10^{-2}$ | $5{,}93\times10^{-2}$ | $1{,}79\times10^{0}$ | $1{,}02\times10^{0}$ | $2{,}11\times10^{0}$ |
| | Std. | $1{,}62\times10^{-4}$ | $3{,}10\times10^{-6}$ | $1{,}20\times10^{-5}$ | $3{,}51\times10^{-6}$ | $2{,}11\times10^{0}$ | $9{,}62\times10^{-1}$ | $1{,}41\times10^{-1}$ |
| D12 | Ort. | $1{,}98\times10^{-3}$ | $3{,}85\times10^{-3}$ | $5{,}78\times10^{-2}$ | $5{,}12\times10^{-2}$ | $6{,}22\times10^{0}$ | $8{,}05\times10^{0}$ | $5{,}16\times10^{0}$ |
| | En | $1{,}96\times10^{-3}$ | $1{,}24\times10^{-3}$ | $1{,}98\times10^{-3}$ | $3{,}61\times10^{-2}$ | $4{,}22\times10^{0}$ | $4{,}12\times10^{0}$ | $2{,}64\times10^{0}$ |
| | Std. | $2{,}90\times10^{-5}$ | $7{,}69\times10^{-3}$ | $1{,}21\times10^{-2}$ | $9{,}60\times10^{-3}$ | $1{,}02\times10^{-1}$ | $2{,}73\times10^{0}$ | $1{,}84\times10^{-1}$ |
| D12N | Ort. | $1{,}87\times10^{-3}$ | $1{,}08\times10^{-2}$ | $1{,}21\times10^{-2}$ | $5{,}09\times10^{-2}$ | $1{,}27\times10^{1}$ | $1{,}38\times10^{1}$ | $7{,}73\times10^{0}$ |
| | En | $1{,}85\times10^{-3}$ | $1{,}38\times10^{-3}$ | $3{,}41\times10^{-3}$ | $3{,}63\times10^{-2}$ | $5{,}11\times10^{0}$ | $7{,}28\times10^{0}$ | $3{,}17\times10^{0}$ |
| | Std. | $2{,}89\times10^{-5}$ | $2{,}17\times10^{-3}$ | $1{,}03\times10^{-2}$ | $1{,}11\times10^{-2}$ | $1{,}13\times10^{0}$ | $1{,}09\times10^{0}$ | $1{,}27\times10^{-1}$ |
| D19 | Ort. | $2{,}62\times10^{-3}$ | $1{,}04\times10^{-1}$ | $1{,}06\times10^{-1}$ | $2{,}15\times10^{-1}$ | $1{,}97\times10^{2}$ | $1{,}08\times10^{1}$ | $9{,}10\times10^{0}$ |
| | En | $2{,}58\times10^{-3}$ | $2{,}13\times10^{-3}$ | $1{,}02\times10^{-2}$ | $6{,}28\times10^{-2}$ | $1{,}23\times10^{2}$ | $5{,}17\times10^{0}$ | $6{,}17\times10^{0}$ |
| | Std. | $3{,}52\times10^{-5}$ | $2{,}17\times10^{-3}$ | $8{,}67\times10^{-2}$ | $7{,}49\times10^{-2}$ | $2{,}32\times10^{1}$ | $2{,}30\times10^{0}$ | $1{,}76\times10^{-1}$ |
| D19N | Ort. | $2{,}67\times10^{-3}$ | $4{,}60\times10^{-3}$ | $5{,}76\times10^{-2}$ | $1{,}36\times10^{-1}$ | $1{,}86\times10^{2}$ | $1{,}67\times10^{1}$ | $8{,}26\times10^{0}$ |
| | En | $2{,}63\times10^{-3}$ | $2{,}13\times10^{-3}$ | $2{,}12\times10^{-3}$ | $7{,}02\times10^{-2}$ | $1{,}47\times10^{2}$ | $9{,}72\times10^{0}$ | $4{,}42\times10^{0}$ |
| | Std. | $3{,}68\times10^{-5}$ | $4{,}71\times10^{-2}$ | $7{,}54\times10^{-2}$ | $5{,}16\times10^{-2}$ | $1{,}26\times10^{1}$ | $1{,}28\times10^{0}$ | $1{,}28\times10^{-1}$ |

$(0{,}50\times N\times M)$ olarak belirlenmiştir. D4N örneği için $limit$ parametresi $(0{,}10\times N\times M)$ değerine eşit alınmıştır. Karşılaştırma tablosunda verilen sonuçlar incelendiğinde, LBABC algoritmasının altı problemin tamamında daha iyi veya rakiplerine oldukça yakın sonuçlar üretebildiği kolaylıkla görülür. Özellikle D12, D12N, D19 ve D19N problemleri için LBABC algoritması, ortalama en iyi amaç fonksiyonu değerlerini dikkate alındığında test edilmiş BDO yöntemlerini geride bırakmayı başarmıştır. D4 ve D4N problemleri için, HDE-CC ve FAF-SOFWA algoritmaları daha başarılı ortalama en iyi amaç fonksiyonu değerleri üretirken, LBABC algoritması bu yöntemlerin bir miktar gerisinde kalmıştır. Ancak, LBABC algoritmasının $6{,}20\times10^{-2}$ ortalama en iyi amaç fonksiyonu değeri ile D4 problemi için tüm algoritmalar arasında üçüncü ve $5{,}98\times10^{-2}$ ortalama en iyi amaç fonksiyonu değeri ile D4N problemi için tüm algoritmalar arasında dördüncü sırada olduğuna ayrıca vurgulanabilir.

LBABC algoritması ile elde edilen sonuçların diğer algoritmalar tarafından elde edilen sonuçlara göre durumu üzerine başka bir değerlendirme Tablo 10 ve Tablo 11 dikkate alınarak yapılabilir. Tablo 10 ve Tablo 11'de en iyi, ortalama en iyi amaç fonksiyon değerleri ve standart sapmaları özetlenen algoritmalar için popülasyon veya koloni büyüklüğü 100 olarak belirlenmiştir [3]. $MCN$, D4 ve D4N problemleri için 300, D12 ve D12N problemleri için 400, D19 ve D19N problemleri için 500 olarak alınmıştır [3]. Bunlara ek olarak, LBABC algoritmasının $limit$ değerleri D4 ve D4N örnekleri için $(0{,}10\times N\times M)$ ve $(0{,}50\times N\times M)$, D12 ve D12N örnekleri için $(0{,}50\times N\times M)$, D19 ve D19N örnekleri için $(0{,}75\times N\times M)$ olarak atanmıştır. Tablo 10 ve Tablo 11'de verilen sonuçlar incelendiğinde D12, D12N, D19 ve D19N problemlerinin tamamında LBABC algoritmasının ADEF, IADEF-NLS, ADEFNLS, DECC-DG, JADE, SHADE algoritmaları ve DE1, DE2, DE3, DE4 isimli temel DE varyantlarına kıyasla daha başarılı sonuçlara ulaştığı anlaşılmaktadır. Ayrıca D4 ve D4N problemleri için LBABC algoritması sırasıyla $6{,}44\times10^{-2}$ ve $6{,}22\times10^{-2}$ ortalama en iyi amaç fonksiyonu değerleri ile karşılaştırmalarda kullanılan yöntemler arasında ikinci sırada olmaya başarmıştır. LBABC algoritması, karşılaştırılan BDO yöntemlerinde olduğu gibi farklı harici yerel arama yöntemleri, problem bölme yaklaşımları veya karmaşık parametre ayarlama stratejileri

**Tablo 9.** 1000 iterasyon için LBABC algoritmasının diğer yöntemlerle karşılaştırılması
(Comparing LBABC algorithm with other techniques for 1000 iterations)

| Pr. | | LBABC | NSGA-II [1] | PSO [23] | PBO [23] | FAF-SOFWA [7] | HMOFA [9] | MOFA [9] |
|---|---|---|---|---|---|---|---|---|
| D4 | Ort. | $6{,}20\times10^{-2}$ | - | $7{,}29\times10^{0}$ | $5{,}79\times10^{-1}$ | $6{,}11\times10^{-2}$ | $6{,}14\times10^{-2}$ | $9{,}76\times10^{-1}$ |
| | En | $6{,}17\times10^{-2}$ | $1{,}87\times10^{0}$ | - | - | $6{,}11\times10^{-2}$ | - | - |
| | Std. | $1{,}69\times10^{-4}$ | - | $3{,}83\times10^{-1}$ | $5{,}66\times10^{-2}$ | $3{,}47\times10^{-6}$ | - | - |
| D4N | Ort. | $5{,}98\times10^{-2}$ | - | $7{,}21\times10^{0}$ | $5{,}98\times10^{-1}$ | $5{,}90\times10^{-2}$ | $5{,}94\times10^{-2}$ | $9{,}75\times10^{-1}$ |
| | En | $5{,}95\times10^{-2}$ | $1{,}74\times10^{0}$ | - | - | $5{,}90\times10^{-2}$ | - | - |
| | Std. | $1{,}62\times10^{-4}$ | - | $2{,}72\times10^{-1}$ | $5{,}98\times10^{-1}$ | $3{,}58\times10^{-6}$ | - | - |
| D12 | Ort. | $1{,}98\times10^{-3}$ | - | $7{,}63\times10^{0}$ | $2{,}97\times10^{0}$ | $2{,}03\times10^{-3}$ | $3{,}60\times10^{-3}$ | $1{,}01\times10^{0}$ |
| | En | $1{,}96\times10^{-3}$ | $2{,}93\times10^{0}$ | - | - | $2{,}02\times10^{-3}$ | - | - |
| | Std. | $2{,}90\times10^{-5}$ | - | $2{,}88\times10^{-1}$ | $5{,}94\times10^{-1}$ | $2{,}09\times10^{-6}$ | - | - |
| D12N | Ort. | $1{,}87\times10^{-3}$ | - | $7{,}62\times10^{0}$ | $3{,}29\times10^{0}$ | $1{,}92\times10^{-3}$ | $3{,}18\times10^{-3}$ | $9{,}67\times10^{-1}$ |
| | En | $1{,}85\times10^{-3}$ | $2{,}82\times10^{0}$ | - | - | $1{,}91\times10^{-3}$ | - | - |
| | Std. | $2{,}89\times10^{-5}$ | - | $3{,}37\times10^{-1}$ | $8{,}31\times10^{-1}$ | $2{,}31\times10^{-6}$ | - | - |
| D19 | Ort. | $2{,}62\times10^{-3}$ | - | $7{,}74\times10^{0}$ | $4{,}68\times10^{0}$ | $2{,}65\times10^{-3}$ | $1{,}42\times10^{-2}$ | $3{,}12\times10^{0}$ |
| | En | $2{,}58\times10^{-3}$ | $3{,}19\times10^{0}$ | - | - | $2{,}65\times10^{-3}$ | - | - |
| | Std. | $3{,}52\times10^{-5}$ | - | $3{,}50\times10^{-1}$ | $1{,}33\times10^{0}$ | $2{,}66\times10^{-3}$ | - | - |
| D19N | Ort. | $2{,}67\times10^{-3}$ | - | $7{,}74\times10^{0}$ | $4{,}75\times10^{0}$ | $2{,}70\times10^{-3}$ | $1{,}57\times10^{-2}$ | $3{,}08\times10^{0}$ |
| | En | $2{,}63\times10^{-3}$ | $3{,}17\times10^{0}$ | - | - | $2{,}70\times10^{-3}$ | - | - |
| | Std. | $3{,}68\times10^{-5}$ | - | $2{,}50\times10^{-1}$ | $1{,}61\times10^{2}$ | $2{,}37\times10^{-6}$ | - | - |

**Tablo 10.** LBABC ile ADEF, IADEF-NLS, ADEF-NLS, DECC-DG, JADE yöntemlerinin karşılaştırılması
(Comparing LBABC with ADEF, IADEF-NLS, ADEF-NLS, DECC-DG, JADE techniques)

| Pr. | | LBABC | ADEF [6] | IADEF-NLS [6] | ADEF-NLS [6] | DECC-DG [6] | JADE [6] |
|---|---|---|---|---|---|---|---|
| D4 | Ort. | $6{,}44\times10^{-2}$ | $6{,}13\times10^{-2}$ | $6{,}28\times10^{-1}$ | $7{,}70\times10^{-1}$ | $3{,}82\times10^{0}$ | $1{,}10\times10^{0}$ |
| | En | $6{,}35\times10^{-2}$ | $6{,}13\times10^{-2}$ | $5{,}48\times10^{-1}$ | $5{,}99\times10^{-1}$ | $3{,}47\times10^{0}$ | $9{,}58\times10^{-1}$ |
| | Std. | $3{,}73\times10^{-4}$ | $2{,}97\times10^{-6}$ | - | - | $2{,}26\times10^{-1}$ | $8{,}37\times10^{-2}$ |
| D4N | Ort. | $6{,}22\times10^{-2}$ | $5{,}93\times10^{-2}$ | $6{,}35\times10^{-1}$ | $7{,}75\times10^{-1}$ | $8{,}89\times10^{0}$ | $1{,}11\times10^{0}$ |
| | En | $6{,}13\times10^{-2}$ | $5{,}93\times10^{-2}$ | $5{,}89\times10^{-1}$ | $5{,}90\times10^{-1}$ | $8{,}24\times10^{0}$ | $9{,}53\times10^{-1}$ |
| | Std. | $5{,}05\times10^{-4}$ | $1{,}87\times10^{-6}$ | - | - | $3{,}18\times10^{-1}$ | $7{,}71\times10^{-2}$ |
| D12 | Ort. | $2{,}06\times10^{-3}$ | $2{,}39\times10^{-3}$ | $1{,}12\times10^{0}$ | $1{,}46\times10^{0}$ | $7{,}16\times10^{0}$ | $2{,}32\times10^{0}$ |
| | En | $2{,}03\times10^{-3}$ | $2{,}22\times10^{-3}$ | $1{,}03\times10^{0}$ | $1{,}15\times10^{0}$ | $6{,}96\times10^{0}$ | $2{,}04\times10^{0}$ |
| | Std. | $1{,}62\times10^{-5}$ | $8{,}73\times10^{-5}$ | - | - | $9{,}86\times10^{-2}$ | $1{,}10\times10^{-1}$ |
| D12N | Ort. | $1{,}94\times10^{-3}$ | $2{,}28\times10^{-3}$ | $1{,}10\times10^{0}$ | $1{,}46\times10^{0}$ | $7{,}39\times10^{0}$ | $2{,}32\times10^{0}$ |
| | En | $1{,}92\times10^{-3}$ | $2{,}14\times10^{-3}$ | $1{,}03\times10^{0}$ | $1{,}17\times10^{0}$ | $7{,}15\times10^{0}$ | $2{,}15\times10^{0}$ |
| | Std. | $2{,}06\times10^{-5}$ | $8{,}25\times10^{-5}$ | - | - | $8{,}10\times10^{-2}$ | $1{,}24\times10^{-1}$ |
| D19 | Ort. | $2{,}67\times10^{-3}$ | $7{,}61\times10^{-3}$ | $1{,}42\times10^{0}$ | $1{,}83\times10^{0}$ | $2{,}70\times10^{2}$ | $2{,}73\times10^{0}$ |
| | En | $2{,}63\times10^{-3}$ | $6{,}21\times10^{-3}$ | $1{,}33\times10^{0}$ | $1{,}72\times10^{0}$ | $2{,}66\times10^{2}$ | $2{,}49\times10^{0}$ |
| | Std. | $2{,}49\times10^{-5}$ | $7{,}48\times10^{-4}$ | - | - | $2{,}05\times10^{0}$ | $1{,}11\times10^{-1}$ |
| D19N | Ort. | $2{,}72\times10^{-3}$ | $7{,}88\times10^{-3}$ | $1{,}46\times10^{0}$ | $1{,}84\times10^{0}$ | $2{,}73\times10^{2}$ | $2{,}66\times10^{0}$ |
| | En | $2{,}69\times10^{-3}$ | $6{,}54\times10^{-3}$ | $1{,}31\times10^{0}$ | $1{,}78\times10^{0}$ | $2{,}69\times10^{2}$ | $2{,}46\times10^{0}$ |
| | Std. | $2{,}17\times10^{-5}$ | $7{,}45\times10^{-4}$ | - | - | $2{,}16\times10^{-2}$ | $1{,}02\times10^{-1}$ |

**Tablo 11.** LBABC ile SHADE, DE1, DE2,DE3 ve DE4 yöntemlerinin karşılaştırılması
(Comparing LBABC with DE1, DE2,DE3 and DE4 techniques)

| Pr. | | LBABC | SHADE [6] | DE1 [6] | DE2 [6] | DE3 [6] | DE4 [6] |
|---|---|---|---|---|---|---|---|
| D4 | Ort. | $6{,}44\times10^{-2}$ | $8{,}97\times10^{-1}$ | $1{,}07\times10^{0}$ | $9{,}29\times10^{-1}$ | $2{,}56\times10^{0}$ | $6{,}86\times10^{-1}$ |
| | En | $6{,}35\times10^{-2}$ | $7{,}71\times10^{-1}$ | $9{,}76\times10^{-1}$ | $8{,}43\times10^{-1}$ | $2{,}40\times10^{0}$ | $6{,}11\times10^{-1}$ |
| | Std. | $3{,}73\times10^{-4}$ | $6{,}51\times10^{-2}$ | - | - | - | - |
| D4N | Ort. | $6{,}22\times10^{-2}$ | $9{,}07\times10^{-1}$ | $1{,}08\times10^{0}$ | $9{,}26\times10^{-1}$ | $2{,}54\times10^{0}$ | $6{,}97\times10^{-1}$ |
| | En | $6{,}13\times10^{-2}$ | $7{,}98\times10^{-1}$ | $9{,}91\times10^{-1}$ | $8{,}34\times10^{-1}$ | $2{,}34\times10^{0}$ | $6{,}28\times10^{-1}$ |
| | Std. | $5{,}05\times10^{-4}$ | $5{,}04\times10^{-2}$ | - | - | - | - |
| D12 | Ort. | $\mathit{2{,}06\times10^{-3}}$ | $2{,}09\times10^{0}$ | $1{,}69\times10^{0}$ | $1{,}77\times10^{0}$ | $4{,}13\times10^{0}$ | $1{,}61\times10^{0}$ |
| | En | $2{,}03\times10^{-3}$ | $1{,}81\times10^{0}$ | $1{,}59\times10^{0}$ | $1{,}66\times10^{0}$ | $4{,}00\times10^{0}$ | $1{,}46\times10^{0}$ |
| | Std. | $1{,}62\times10^{-5}$ | $9{,}39\times10^{-2}$ | - | - | - | - |
| D12N | Ort. | $\mathit{1{,}94\times10^{-3}}$ | $2{,}08\times10^{0}$ | $1{,}67\times10^{0}$ | $1{,}78\times10^{0}$ | $4{,}14\times10^{0}$ | $1{,}60\times10^{0}$ |
| | En | $1{,}92\times10^{-3}$ | $1{,}85\times10^{0}$ | $1{,}59\times10^{0}$ | $1{,}64\times10^{0}$ | $3{,}85\times10^{0}$ | $1{,}46\times10^{0}$ |
| | Std. | $2{,}06\times10^{-5}$ | $8{,}81\times10^{-2}$ | - | - | - | - |
| D19 | Ort. | $\mathit{2{,}67\times10^{-3}}$ | $2{,}45\times10^{0}$ | $1{,}85\times10^{0}$ | $2{,}07\times10^{0}$ | $4{,}56\times10^{0}$ | $1{,}91\times10^{0}$ |
| | En | $2{,}63\times10^{-3}$ | $2{,}29\times10^{0}$ | $1{,}77\times10^{0}$ | $1{,}90\times10^{0}$ | $4{,}39\times10^{0}$ | $1{,}74\times10^{0}$ |
| | Std. | $2{,}49\times10^{-5}$ | $9{,}52\times10^{-2}$ | - | - | - | - |
| D19N | Ort. | $\mathit{2{,}72\times10^{-3}}$ | $2{,}45\times10^{0}$ | $1{,}84\times10^{0}$ | $2{,}03\times10^{0}$ | $4{,}55\times10^{0}$ | $1{,}93\times10^{0}$ |
| | En | $2{,}69\times10^{-3}$ | $2{,}29\times10^{0}$ | $1{,}74\times10^{0}$ | $1{,}85\times10^{0}$ | $4{,}34\times10^{0}$ | $1{,}80\times10^{0}$ |
| | Std. | $2{,}17\times10^{-5}$ | $9{,}33\times10^{-2}$ | - | - | - | - |

kullanmamaktadır. Buna rağmen işçi ve gözcü arı fazlarında ve başlangıç çözümlerini üretme mekanizmasında yapılan uygun değişiklikler ile ABC algoritmasının özellikle on iki veya on dokuz zaman serisi içeren problemler için çözüm kapasitesinin büyük ölçüde artırılabileceğini göstermiştir.

## 4. Sonuçlar (Conclusions)

Büyük veri kavramı hesaplamalarda verinin doğrudan kullanıldığı optimizasyon problemlerinin tanımlarını önemli ölçüde değiştirmiş, optimizasyon problemlerinin çözümü için yaygın olarak kullanılan evrimsel ve sürü zekası temelli algoritmaların problemlerin değişen tanım ve karmaşıklıkları sebebi ile performanslarının incelenmesini zorunluluk haline gelmiştir. Arılarının yiyecek kaynağı arama davranışlarındaki zekiliği modelleyen ABC algoritması, basit, anlaşılabilir yapısı, az sayıda kontrol parametresi ve çözüm arama-kullanma operasyonları arasında sağladığı hassas denge sebebi ile pek çok araştırmacının dikkatini çekmiş, numerik ve ayrık gerçek hayat ve mühendislik probleminin çözümünde başarı ile kullanılmıştır. Bu çalışmada, ABC algoritması BDO problemlerinin özellikleri dikkate alınarak güncellenmiş ve kafes temelli LBABC adlı yeni bir varyant tanıtılmıştır. LBABC algoritmasında başlangıç besin kaynakları ya da çözümler BDO'ya daha uygun bir şema yürütülerek problem verisi aracılığı ile oluşturulmuştur. İşçi arı fazı, BDO'nun ihtiyaç duyduğu mevcut çözümleri kullanma-tüketme süreçlerine de destek verecek kafes temelli en iyileme ile güçlendirilmiş, bu sayede işçi arıların sadece komşuluğundaki diğer arılar ile etkileşimi sağlanmıştır. Gözcü arı fazında, varsayılan olasılık temelli kaynak seçme stratejisi deterministik bir yaklaşım ile değiştirilmiştir. Gözcü arılar rastgele seçtikleri kaynakların komşuluğunda araştırma yapmak üzere değil, ilgili iterasyona kadar bulunmuş en iyi besin kaynağının komşuluğunda araştırma yapmak üzere kovandan gönderilmiştir. Ayrıca, gözcü arıların araştırma performansını iyileştirebilmek için Rechenberg 1/5 mutasyon kuralı ile adaptif adım büyüklüğü belirlenmiştir. LBABC algoritmasının başarımı EEG sinyallerindeki gürültünün minimizasyonunu gerektiren BDO problemi üzerinden incelenmiş $limit$, $SN$ ve $MCN$ gibi kontrol parametrelerinin farklı değerleri ile nihai çözümlerin kalitelerinin nasıl değiştiği araştırılmıştır. LBABC ile bulunan çözümler önce standart ABC algoritması ve oldukça başarılı varyantları olan GABC, ABC/best/1, ABC/best/2, CABC, COABC ve qABC algoritmaları ile bulunan çözümler kullanılarak kıyaslanmıştır. Bunlara ek olarak LBABC algoritması DE, GA, FA, FW, PBO ve PSO temelli BDO yöntemleri ile de karşılaştırılmıştır. Karşılaştırma sonuçları, ABC algoritmasının işçi ve gözcü arı fazlarında LBABC algoritmasındaki olduğu gibi büyük veri dikkate alınarak uygun değişikliklerin yapılması halinde, diğer meta-sezgisel yöntemlere kıyasla oldukça başarılı sonuçlar elde edilebileceğini göstermiştir. İşçi, gözcü ve kâşif arı fazları için veriyi kullanarak araştırmayı yönlendirecek farklı yaklaşımların geliştirilmesi ve ABC algoritması temelli yeni yöntemlerinin tek ve çok amaçlı BDO problemlerinin çözümünde kullanılması ilerleyen süreçlerde yapılacak çalışmaların temel konularını oluşturabilir.

## Kaynaklar (References)


1. Chen M., Mao I., Liu Y., Big Data: A Survey, Mobile Networks and Applications, 19 (2), 171-209, 2014.
2. Mohammed A.S., Amrahov Ş.E., Çelebi F.V., Interpolated binary search: An efficient hybrid search algorithm on ordered datasets, Eng. Sci. Technol. Int. J., 24 (5), 1072-1079, 2021.
3. Kambatla K., Kollias G., Kumar V., Grama A., Trends in big data analytics, J. Parallel Distrib. Comput., 74 (7), 2561-2573, 2014.
4. Amrahov Ş.E., Tuğrul B., A Community Detection Algorithm on Graph Data, 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya-Türkiye, 1-4, 28-30 Eylül, 2018.
5. Amrahov Ş.E., Mohammed A.S., Çelebi F.V., New and improved search algorithms and precise analysis of their average-case complexity, Future Gener. Comput. Syst., 95, 743-753, 2019.
6. Wu X., Zhu X., Wu G.Q., Ding W., Data mining with big data, IEEE Transaction on Knowledge and Data Engineering, 26 (1), 97-107, 26 Haziran, 2013.
7. Tsai C.W., Lai C.F., Chao H.C., Vasilakos A.V., Big data analytics: a survey, Journal of Big Data, 2 (21), 1-32, 2015.
8. Hassanien A.E., Azar A.T., Snasael V., Kacprzyk J., Abawajy J.H., Big Data in Complex Systems, 9, Springer International Publishing, 2015.
9. Zhou Z.-H., Chawla N.V., Jin Y., Williams G.J., Big Data Opportunities and Challenges: Discussions from Data Analytics Perspectives, IEEE Comput. Intell. Mag., 62-74, 1 Kasım 2014.
10. Goh S.K., Tan K.C., Al-Mamun A., Abbass H.A., Evolutionary Big Optimization (BigOpt) of Signals, 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai-Japan, 3332-3339, 25-28 Mayıs, 2015.
11. Elsayed S., Sarker R., Differential Evolution Framework for Big Data Optimization, Memetic Computing, 8, 17-33, 2016.
12. Majdouli M.A.E., Rbouh I., Bougrine S., Benani B. E., Imrani A. A. E., Fireworks algorithm framework for Big Data optimization, Memetic Computing, 8, 333-347, 2016.

**13.** Sabar N. R., Abawajy J., Yearwood J., Heterogeneous Cooperative Co-Evolution Memetic Differential Evolution Algorithm for Big Data Optimization Problems, IEEE Trans. Evol. Comput., 21 (2), 315-327, Nisan, 2017.

**14.** Wang H., Wang W., Cui L., Sun H., Zhao J., Wang Y., Xue Y., A Hybrid Multi-Objective Firefly Algorithm for Big Data Optimization, Appl. Soft Comput., 69, 806-815, 2018.

**15.** Yi J.H., Deb S., Dong J., Alavi A.H., Wang G.G., An Improved NSGA-III Algorithm with Adaptive Mutation Operator for Big Data Optimization Problems, Future Gener. Comput. Syst., 88, 571-585, 2018.

**16.** Elaziz M.A., Li L., Jayasena K.P.N., Xiong S. (2020). Multiobjective Big Data Optimization based on a Hybrid Salp Swarm Algorithm and Differential Evolution, Appl. Math. Modell., 80 (9), 929-943, 2019.

**17.** Abdi Y., Feizi-Derakhshi M.-R., Hybrid multi-objective evolutionary algorithm based on Search Manager framework for big data optimization problems, Appl. Soft Comput., 87, 105991, 2020.

**18.** Meselhi M.A., Elsayed S.M., Sarker R.A., Essam D. L., Parallel Evolutionary Algorithm for EEG Optimization Problems, 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków-Poland, 2577-2584, 28 Haziran-1 Temmuz, 2021.

**19.** Celil S., Aslan S., Demirci S., A Novel Harmony Search Based Method for Noise Minimization on EEG Signals, 2021 6th International Conference on Computer Science and Engineering (UBMK), Ankara-Türkiye, 747-750, 15-17 Eylül, 2021.

**20.** Aslan S., A Comparative Study Between Artificial Bee Colony (ABC) Algorithm and Its Variants on Big Data Optimization, Memetic Computing, 12 (2), 129-150, 2020.

**21.** Zhu G., Kwong S., Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput., 217 (7), 3166-3173, 2010.

**22.** Gao W., Liu S., Huang L., A global best artificial bee colony algorithm for global optimization, J. Comput. Appl. Math, 236 (11), 2741-2753, 2012.

**23.** Gao W., Liu S., Huang, L., A Novel Artificial Bee Colony Algorithm Based on Modified Search Equation and Orthogonal Learning, IEEE Trans. Cybern., 43 (3), 1011-1014, Haziran, 2013.

**24.** Karaboğa D., Görkemli B., A Quick Artificial Bee Colony (qABC) Algorithm and Its Performance on Optimization Problems, Appl. Soft Comput., 23, 227-238, 2014.

**25.** Luo J., Wang Q., Xiao X., A Modified Artificial Bee Colony Algorithm Based on Converge-onlookers Approach for Global Optimization, Appl.Math. Comput., 219 (20), 10253–10262, 2013.

**26.** Karaboga D., 2005. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

**27.** Öztürk C., Hançer E., Karaboğa D., Automatic Clustering with Global Best Artificial Bee Colony Algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 29 (4), 677-687, 2014.

**28.** Dağdeviren U., Kaymak B., Investigation of Parameters Affecting Optimum Cost Design of Reinforced Concrete Retaining Walls Using Artificial Bee Colony Algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 33 (1), 239-253, 2018.

**29.** Eke İ., Taplamacıoğlu M.C., Kocaarslan İ., Design of Robust Power System Stabilizer Based on Artificial Bee Colony Algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 26 (3), 683-690, 2011.

**30.** Durgut R., Aydin M., Adaptive Binary Artificial Bee Colony for Multi-Dimensional Knapsack Problem, Journal of the Faculty of Engineering and Architecture of Gazi University, 36 (4), 2333-2348, 2021.

**31.** Toktaş A., Akdağlı A., Computation of Resonant Frequency of E-Shaped Compact Microstrip Antennas, Journal of the Faculty of Engineering and Architecture of Gazi University, 27 (4), 847-854, 2012.

**32.** Zhong W., Liu J., Xue M., Jiao L., A Multiagent Genetic Algorithm for Global Numerical Optimization, IEEE Trans. Syst. Man Cybern. Part B Cybern., 34 (2), 1128–1141, 2004.

**33.** Akay B., Karaboğa D., A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization, Information Sciences, 192 (1), 120-142, 2012.

**34.** Cao Z., Wang L., Hei X., Jiang Q., Lu X., Wang X., A Phase Based Optimization Algorithm for Big Optimization Problems, 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC-Canada, 5209-5214, 24-29 Temmuz, 2016.