BÜYÜK ÖLÇEKLİ SÜREKLİ OPTİMİZASYON PROBLEMLERİ İÇİN ELİT BİREY TABANLI YAPAY ARI KOLONİSİ ALGORİTMASI

Doğan AYDIN[1*], Ümit GÜVEN[2]

[1]İzmir Demokrasi Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İzmir,
ORCID No : https://orcid.org/0000-0003-2478-4818
[2]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kütahya,
ORCID No : https://orcid.org/0000-0002-2787-8847

Anahtar Kelimeler

Yapay Arı Kolonisi
Sürü Zekâsı
Büyük Ölçekli Optimizasyon
Sürekli Optimizasyon
SOCO11

Öz

Optimizasyon problemlerinin boyutu büyüdükçe çözümleri de zorlaşmaktadır. Bu problemlerin üstesinden gelmek için sürü zekâsı algoritmalarından faydalanılabilir. Birçok sürü zekâsı algoritmalarından bir tanesi de Yapay Arı Kolonisi (Artificial Bee Colony, ABC) algoritmasıdır. Büyük ölçekli optimizasyon problemlerinde yapay arı kolonisi algoritmasından faydalanabilmek için orijinal ABC algoritmasında bazı iyileştirmeler yapmak gerekmektedir. Bu çalışmada, ABC algoritması için yapılan iyileştirmeler "Elit Birey Tabanlı Uyarlanabilir Yapay Arı Kolonisi Algoritması' adını verdiğimiz yeni bir ABC algoritması içerisinde tanımlanmıştır. Klasik ABC algoritmalarından farklı olarak işçi ve gözcü arı adımlarında farklı arama denklemleri kullanılmış ve bu arama denklemlerinde elit bireylerden yararlanılmıştır. Ayrıca bir yerel arama tekniği ile algoritma performansı güçlendirilmiştir. Algoritmalara ait parametre değerlerinin doğru olarak seçilmesi algoritmaların başarısında büyük etkiye sahiptirler. Bu çalışmada irace aracı kullanılarak algoritmaya ait parametreler en iyi şekilde ayarlanmaya çalışılmıştır. Geliştirdiğimiz algoritma büyük ölçekli sürekli optimizasyon fonksiyonlarını barındıran SOCO11 ölçüt fonksiyon kümesinde test edilmiştir. Elde ettiğimiz sonuçlar ABC algoritmalarıyla ve SOCO11 yarışmasına katılan algoritmalar ile karşılaştırılmış ve başarılı sonuçlar elde edilmiştir.

ELITE INDIVIDUAL BASED ARTIFICIAL BEE COLONY ALGORITHM FOR LARGE-SCALE CONTINUOUS OPTIMIZATION PROBLEMS

Keywords

Artificial Bee Colony
Swarm Intelligence
Large-Scale Optimization
Continuous Optimization
SOCO11

Abstract

As the size of the optimization problems grows, it becomes more difficult to solve. Swarm intelligence algorithms can be used to tackle these problems. One of the swarm intelligence algorithms is the Artificial Bee Colony (ABC) algorithm. In order to benefit from the artificial bee colony algorithm in large-scale optimization problems, some improvements are required in the original ABC algorithm. In this study, the improvements made for the ABC algorithm are defined in a new ABC algorithm that we call "Elite Individual Based Adaptive Artificial Bee Colony Algorithm". Unlike classical ABC algorithms, different search equations are used in employed and onlooker bees steps and elite individuals are used in these search equations. In addition, the algorithm performance is enhanced with a local search technique. Choosing the correct parameter values of algorithms has a great effect on the success of algorithms. In this study, using the irace tool, the parameters of the algorithm are tried to be adjusted in the best way. The algorithm we developed is tested on the SOCO11 benchmark function set, which includes large-scale continuous optimization functions. The results we obtained were compared with ABC algorithms and the algorithms that previously participated in SOCO11 competition, and successful results are obtained.

* Sorumlu yazar; e-posta : dogan.aydin@idu.edu.tr

Araştırma Makalesi		Research Article	
Başvuru Tarihi	: 12.02.2021	Submission Date	: 12.02.2021
Kabul Tarihi	: 24.07.2021	Accepted Date	: 24.07.2021

1. Giriş

Optimizasyon kavram olarak; mevcut sistemde bulunan kaynakları en uygun verim ile kullanarak belirlenen hedeflere (maliyetin düşürülmesi, karın artırılması, kullanım kapasitesinin yükseltilmesi, verimin artırılması vb.) ulaşımı amaçlayan bir teknolojidir (Gass, 2000). Optimizasyon teknolojisi karar niteliğini artırmak ve süreci hızlandırmak gibi süreçlerde uygulanarak günlük hayatta karşılaşılan problemlerin çözümünde etkin olarak kullanılmaktadır (Winston, 2003).

Optimizasyon problemlerinin belirli bir boyut büyüklüğüne kadar olanları tam sayılı programlama yöntemleri ile çözülebilirken, büyük boyutlu problemlerin sezgisel ve metasezgisel yöntemlerle çözülmesi gerekmektedir (Serkan ve Fığlalı, 2016). Sezgisel algoritmalar kesin sonuçları garanti etmemekle beraber, kısa sürede optimuma yakın sonuçlar verirler. Klasik sezgisel algoritmalar genellikle problemlere özgü bir yapıya sahiptirler. Sadece bir problem türü için geliştirilen sezgisel yöntemin farklı bir problem türüne uygulanması zordur. Metasezgisel yöntemler ise problem özelliklerinden bağımsız ve genel tanımlanmış çözücülerdir.

Yapay Arı Kolonisi (Artificial Bee Colony -ABC) algoritması da sürü zekâsı temelli bir metasezgisel yöntem olup birçok optimizasyon probleminin çözümünde başarılı bir şekilde kullanılmıştır. Orijinal ABC algoritması birçok problemde iyi sonuçlar verse de problem boyutu büyüdükçe performansı düşmektedir. Bunun temel nedeni kullanılan arama denkleminin büyük boyutlu problemler için uygun olmamasıdır.

Ayrıca metasezgisel algoritmaların başarısı uygun yerel arama algoritmaları kullanılarak iyileştirilebilmektedir (De Oca, Aydın ve Stützle, 2011; Liao, Aydin ve Stützle, 2013). Fakat Orijinal ABC algoritması herhangi bir yerel arama algoritmasını kullanmamaktadır. ABC algoritmalarında da uygun yerel arama algoritmaları kullanıldığında performans artışı gözlemlenebilir.

Orijinal ABC algoritmasında çeşitlendirme davranışı işçi arı adımında, odaklanma davranışı ise gözcü arı adımında gerçekleştirilir. Bu çalışmada önerilen algoritmada ise işçi arı adımındaki her bir arının referans çözümleri iyi çözümlere yaklaştırılmakta, gözcü arıların referans aldıkları iyi çözümler ise rasgele seçilen çözümlere yaklaştırılmaktadır.

Yukarıda sözü edilen dezavantajları ortadan kaldırmak için yeni bir ABC varyantı önerilmiştir. "Elit Birey Tabanlı Uyarlanabilir Yapay Arı Kolonisi Algoritması (Elite Individual Based Adaptive Artificial Bee Colony-EIAABC)" adını verdiğimiz algoritmada aşağıda verilen iyileştirmeler önerilmiştir:

- Elit bireyleri ve o ana kadar ki en iyi çözümü kullanan yeni arama denklemleri işçi arı adımı için önerilmiştir.
- Her defasında birden çok boyutu değiştirebilecek şekilde arama denklemi uyarlanmıştır.
- Orijinal ABC algoritmasının aksine, işçi arı adımında odaklanma, gözcü arı adımında çeşitlendirme davranışı sergileyen yeni bir yaklaşım ile uyarlanabilen bir arama denklemi stratejisi önerilmiştir. Odaklanma davranışında elit bireyler kullanılmıştır.
- Yüksek boyutlu problemlerde başarılı sonuçlar elde edilebilen bir yerel arama algoritması EIAABC algoritmasına uyarlanmıştır.

Bu çalışmada, EIAABC algoritması Soft Computing dergisi özel sayısında (special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, SOCO11) belirtilen ölçüt test problemi üzerinde problem çözme standartlarına bağlı kalınarak test edilmiştir. Elde edilen sonuçlar literatürde çokça bilinen Yapay Arı Kolonisi algoritmaları ve SOCO11'e katılan yarışmacı algoritmalar ile karşılaştırılmıştır.

Bu makale organizasyonu şu şekildedir: İkinci bölümde; Yapay arı kolonileri algoritmalarının temelini oluşturan orijinal ABC algoritması anlatılmıştır. Üçüncü bölümde; geliştirmiş olduğumuz elit birey tabanlı uyarlanabilir yapay arı kolonisi algoritması anlatılmıştır. Dördüncü bölümde; büyük ölçekli problem kümesi ve çalıştırma koşulları tanıtılmış, parametre ayarlarının nasıl gerçekleştirildiğine yer verilmiştir. Beşinci bölümde; algoritmamızın SOCO11 problemleri üzerindeki başarısı ve diğer algoritmalarla olan karşılaştırmalara yer verilmiştir. Son bölümle birlikte sonuçlar tartışılarak makale sonuçlandırılmıştır.

2. Bilimsel Yazın Taraması

2.1 Büyük ölçekli sürekli optimizasyon problemleri

Birçok gerçek dünya problemi büyük ölçekli sürekli optimizasyon problemi olarak formüle edilebilmektedir. Sürekli optimizasyon problemlerinin amacı verilen bir $f(\boldsymbol{X})$ fonksiyonu eniyileyecek değişkenlerin $\boldsymbol{X} = \{x_1, x_{2,} \dots, x_D\}$ bulunmasıdır. Burada D problemin boyutunu ifade eder (Aydın, Yavuz ve Stützle, 2017). Problem boyutu büyüdükçe

algoritmaların birçoğunun performansı düşer. Bunun temel olarak iki nedeni vardır. Birincisi, problemin karmaşıklığı genellikle problemin büyüklüğü ile artmaktadır. İkincisi, problemin çözüm uzayı problemin büyüklüğü ile katlanarak artar ve belirli bir zaman bütçesinde gelecek vaat eden tüm bölgeleri keşfetmek için daha verimli bir arama stratejisi gerekir. Bu nedenle, yüksek boyutlu problemler için ölçeklenebilirlik, modern sürekli optimizasyon algoritması yaklaşımları için temel bir gereklilik haline gelir (Lozano, Molina ve Herrera, 2011).

Büyük ölçekli optimizasyon problemlerinin zorluğu ile birlikte gerçek dünyada birçok problem örneğinin bulunması konunun önemli bir araştırma alanı haline gelmesini sağlamıştır. Sezgisel optimizasyon algoritmalarının büyük ölçekli problemlerde başarısız olmaları nedeniyle bu konuda metasezgisel algoritmalar ile yapılan çalışmalar öne çıkmaktadır. Literatürde konu ile ilgili yayınlanmış çokça makaleye rastlamak mümkündür. Son yıllardaki güncel çalışmalar incelendiğinde önerilen algoritmaların üç temel alana ayrıldığı gözlemlenebilir. Birinci grup algoritmalar bir metasezgisel yöntemin iyileştirilerek performans artışını amaçlar (Gungor, Emiroglu, Cinar ve Kiran, 2020; Li ve diğ., 2021; Long, Wu, Liang ve Xu, 2019; Yang, Tang ve Yao, 2019). İkinci grup algoritmalar metasezgisel algoritmalar yerel arama prosedürleri ile melezleştiren memetik yöntemlerdir (Gao ve diğ., 2019; Yıldız ve Topal, 2019). Üçüncü grupta yer alan algoritmalar ise birden çok metasezgisel algoritmayı tek bir çatı altında tutmaya çalışarak sinerji oluşturmayı amaçlamaktadır (Aydın ve diğ., 2017; Gökalp ve Uğur, 2020; Meselhi, Elsayed, Sarker ve Essam, 2020). Bu çalışmada da hem algoritmada iyileştirmeler yapılmakta hem de yerel arama prosedürleri ile melezleştirme önerilmektedir. Dolayısıyla birinci ve ikinci grup çalışmalara örnek olabilecek türden bir algoritma önerilmiştir.

2.2. Orijinal yapay arı kolonisi algoritması

ABC, (Karaboğa, 2005) 2005 yılında doğadan ilham alınarak Derviş KARABOĞA tarafından geliştirilen bir algoritmadır. Bal arılarının zekice ve organize davranışlarına dayanır. Algoritma çıktığı günden bu yana çok farklı problemlerin çözümünde başarıyla kullanılmıştır. Yapay arı kolonisinin yapısı dört bileşenden oluşmaktadır. Bunlar ilkleme adımı, işçi arı, gözcü arı ve kâşif arı adımlarıdır.

İlkleme: ilkleme adımında algoritma başlama parametrelerini ayarlar. Bu parametreler yiyecek kaynaklarının büyüklüğü, yiyecek ömrünü temsil eden *limit* değeri ve algoritmanın sonlandırma kriteridir.

Popülasyondaki her birey yiyecek kaynağı olarak isimlendirilir. Bu yiyecek kaynaklarının, her biri problemi çözmek için optimize edilecek değişkenlerin değerlerini çözüm ihtimallerini temsil eden *D*-boyutlu vektörlerdir. Algoritmanın bu aşamasında, popülasyondaki her bir çözüm (x_i) başlangıç arama uzayındaki homojen dağılımla aşağıdaki gibi üretilir:

$$x_{i,j} = l_j + rand(0,1)(u_j - l_j) \quad (1)$$

Buradaki $j(1 \geq j \geq D)$ problem değişkeninin indeks numarasıdır, u_j ve l_j sırasıyla *j.* değişkeninin üst ve alt sınırlarıdır. Sonrasında her bir çözüm x_i'nin amaç fonksiyon değerleri $f(x_i)$ hesaplanır. Algoritmanın aşağıdaki adımlarında, her bir çözüm minimize edilmeye çalışılmaktadır. Her bir çözüm x_i üzerindeki minimizasyon denemelerinin sayısı, başlatma adımında sıfıra ayarlanan $trial_i$'dir.

İşçi Arı Adımı: Bu aşamada, *i.* işçi arının sorumlu olduğu çözüm x_i'ye yakın yeni bir çözüm v_i oluşturur. Gerçekte bu, x_i'nin rastgele seçilen bir j değişkeni seçilerek aşağıdaki denklem kullanılarak bulunur:

$$v_{i,j} = x_{i,j} + \varphi_{i,j}(x_{i,j} - x_{r,j}) \quad (2)$$

Burada $\varphi_{i,j}$ *[-1,1]* aralığında rastgele oluşturulmuş bir sayı ve $x_{r,j}$ ($r \neq i$ ve $r \in \{1,2,\dots,SN\}$) rastgele seçilen bir çözümdür. Sonrasında, yeni aday çözümün amaç fonksiyon değerleri hesaplanır. Eğer aday çözüm, referans çözümden daha iyiyse referans çözümün yerini alır ve $trial_i$ değeri sıfırlanır. Aksi halde referans çözümün deneme değeri $trial_i$ bir artırılır.

Gözcü arı adımı: İşçi arı adımında tüm çözümlerin kalitesi değerlendirildikten sonra algoritma, gözcü arı adımında en iyi çözümleri kullanır. İşçi arıların aksine, her bir gözcü arı, x_i olarak tanımlanan her bir çözümün olasılık değerini (p_i) temel alan bir rasgele seçim kuralına göre ziyaret edeceği çözümü belirler. Her bir çözüm için olasılık değeri şu şekilde hesaplanır:

$$p_i = \frac{\frac{1}{1+f(x_i)}}{\sum_{n=1}^{SN}\frac{1}{1+f(x_n)}} \quad (3)$$

Olasılıklı seçim kuralı ile yüksek kaliteli çözümler daha yüksek bir olasılık ile seçilir. Bir gözcü arı çözümü ziyaret ettiğinde, işçi arı adımındaki Denklem (2) kullanılarak ziyaret edilen çözümün etrafında yeni bir çözüm arar. Eğer aday çözüm, referans çözümden daha iyiyse referans çözümün yerini alır ve $trial_i$ değeri sıfırlanır. Aksi halde referans çözümün deneme değeri $trial_i$ bir artırılır.

Kâşif Arı Adımı: $trial_i$ değeri, çözüm x_i için sınır değere (*limit*) eşit olduğunda bu, çözümün ziyaret sınırına ulaştığı anlamına gelir. Sınıra erişen çözümler popülasyondan atılır ve Denklem (1) kullanılarak bir kâşif arı tarafından keşfedilen rasgele yeni bir çözüm popülasyona eklenir. Bu şekilde, algoritmanın keşif kapasitelerinin arttırması amaçlanmaktadır.

2.3. Orijinal ABC algoritmasını iyileştirmek için yapılan çalışmalar

Algoritmayı arama denklemi aracılığıyla iyileştirmeye yönelik ilk çalışmalar hem işçi arılar hem de gözcü arılar adımları için tek ama etkili arama denklemlerinin önerilmesine dayanıyordu. Diwold, Aderhold, Scheidler ve Middendorf (2011), arıların aramasını, rastgele seçilen çözümler yerine şimdiye kadarki en iyi çözüme yönlendirildi. Gao ve Liu (2011), Diferansiyel Evrim (DE) algoritmasından esinlenen bir arama denklemi kullandı. Akay ve Karaboğa (2012), orijinal ABC'de aday çözümlerin üretiminde tek bir boyut değiştirilmesinin yetersiz olduğunu iddia etmişler ve değiştirme oranı (modification ratio, MR) isimli yeni bir parametre ile etkilenecek boyut değerini belirleyebiliyorlardı. Banharnsakun, Achalakul ve Sirinaovakul (2011), orijinal ABC'nin komşu çözüme dayalı yaklaşımına ek olarak şimdiye kadarki en iyi çözüme yaklaşımı önerdi. En iyi komşu kılavuzlu ABC algoritmasında ise en iyi komşu kılavuzlu arama denklemi önerilmişti.

Son zamanlarda çoklu arama denklemleri kullanan bazı ABC varyantları ortaya çıktı. Örnek olarak Kiran, Hakli, Gunduz, ve Uguz, (2015) rulet çarkı tekniği ile beş güncelleme kuralı stratejisinden uygun olanı belirlemeye çalıştı. Gao ve diğ. (2015) ise aynı çözüm için üç farklı arama denklemi kullanmayı tercih etmiş ve hangisi daha iyi ise onunla devam etmeyi uygun bulmuştur. Xue, Jiang, Zhao ve Ma (2018) da üç farklı arama stratejisi kullanmış ve rulet çarkı mekanizması ile arama denklemlerini seçmiştir. Yavuz ve Aydın (2019) çoklu arama denklemlerini uyarlanabilir bir yöntemle seçmişlerdir.

Performansı etkileyen başka bir algoritmik bileşen, yerel arama prosedürleridir. Literatürde, ABC algoritmaları ile hibritlenebilen birkaç yerel arama yöntemi vardır. Aydın, Liao, De Oca ve Stützle (2011), ABC algoritmasını artan sosyal öğrenme çerçevesi ve yerel bir arama yöntemi ile birleştirmiştir. Rosenbrock ABC'de (Kang, Li ve Ma, 2011), Rosenbrock'un dönüş yönü yöntemi, odaklanma davranışını geliştirmek için kullanılmıştır. Badem, Basturk, Caliskan ve Yuksel (2018), ABC ve sınırlı belleğe sahip Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algoritmalarını kullanan bir hibrit optimizasyon algoritması önermiştir.

3. Yöntem

3.1. Elit Birey Tabanlı Uyarlanabilir Yapay Arı Kolonisi Algoritması

Orijinal ABC algoritması birçok problemde iyi sonuçlar verse de problem boyutu büyüdükçe performansı düşmektedir. Bunun temel nedeni Denklem (2)'de belirtilen arama denkleminin büyük boyutlu problemler için uygun olmamasıdır. Çünkü bu arama denklemi aşağıdaki dezavantajlara sahiptir:

Arama denklemi, rastgele bir çözümü seçerek işçi veya gözlemci arıların referans çözümün etrafında yeni çözümler bulmasına dayanır. Seçilen rastgele çözüm uygun bir yiyecek kaynağı değilse algoritma bütçesinden gereksiz yere tüketim yapılır. Bu da algoritmanın yakınsama davranışına olumsuz etki eder.

Her defasında sadece referans alınan çözümün bir boyutu değiştirilir. Bu nedenle problem boyutu büyüdükçe algoritma daha yavaş yakınsamaktadır.

Orijinal ABC algoritmasında işçi arı ve gözcü arı adımlarında aynı arama denklemi kullanılmaktadır. Fakat problem türü ve boyutu değiştikçe bu arama denklemi yetersiz kalmaktadır. Her bir adım için ayrı ve uygun bir arama denkleminin seçilmesi gerekir.

İşçi arı adımı algoritmanın çeşitlendirme davranışına katkı sağlarken gözcü arı adımı odaklanma davranışına katkı sağlar. Fakat bu katkının ölçüsü net olmayıp problem türüne göre farklılık göstermelidir. Eşit oranda çeşitlendirme ve odaklanma problem boyutu büyüdükçe her zaman başarılı olmayabilir.

Ayrıca metasezgisel algoritmaların başarısı uygun yerel arama algoritmaları kullanılarak iyileştirilebilir (De Oca ve diğ., 2011). Fakat Orijinal ABC algoritması herhangi bir yerel arama algoritmasını kullanmamaktadır. Bu da algoritma performansına olumsuz etki yapar.

Yukarıda sözü edilen dezavantajları ortadan kaldırmak için bu çalışmada yeni bir ABC varyantı önerilmiştir. "Elit Ajan Tabanlı Uyarlanabilir Yapay Arı Kolonisi Algoritması (Elite Individual Based Adaptive Artificial Bee Colony-EIAABC)" adını verdiğimiz algoritmada aşağıda verilen iyileştirmeler önerilmiştir:

Elit ve o ana kadar ki en iyi çözümleri kullanan yeni arama denklemleri işçi arı adımı için önerilmiştir.

Her defasında birden çok boyutu değiştirebilecek şekilde arama denklemi farklılaştırılmıştır.

Orijinal ABC algoritmasının aksine olarak işçi arı adımında odaklanma, gözcü arı adımında çeşitlendirme davranışı sergileyen yeni bir arama denklemi stratejisi önerilmiştir.

Yüksek boyutlu problemlerde başarılı sonuçlar elde edilebilen Çoklu Yörünge Aramasındaki 1. Yerel arama (local search 1 in Multi Trajectory Search, MTSls1)

algoritması (Tseng ve Chen, 2008) EISEABC algoritmasına entegre edilmiştir. Aşağıda önerilen bu yaklaşımların ayrıntılarına değinilmiştir.

3.1.1. Uyarlanabilen bir arama denklemi stratejisi

Orijinal ABC algoritmasında çeşitlendirme davranışı işçi arı adımında, odaklanma davranışı ise gözcü arı adımında gerçekleştirilir. Bu çalışmada ise bu davranışı tersine çevirecek bir yaklaşım getirilmiştir. Bunun için işçi arı adımındaki her bir arının referans çözümler iyi çözümlere yaklaştırılmakta, gözcü arıların referans aldıkları iyi çözümler ise rasgele seçilen çözümlere yaklaştırılmaktadır. Problem türü değiştikçe uygulanması gerekli olan arama denklemi de değişebileceğinden problem türü için uygun arama denkleminin seçimi de çalışma esnasında uyarlanabilen bir strateji ile belirlenmiştir. Bu stratejide şablon bir arama denklemi önerilmiş olup üç bileşenden oluşmaktadır:

$$v_{i,j} = bileşen1_{i,j} + bileşen2_{i,j} + bileşen3_{i,j} \qquad (4)$$

Burada tüm bileşenler işçi arı ve gözcü arı adımlarına göre farklı değerler alabilmektedir. Bu değerler Tablo 1 ve 2'de belirtilmiştir.

Tablo 1
İşçi Arı Adımında Kullanılan Arama Denklemi Bileşenleri

$\boldsymbol{bileşen1_{i,j}}$	$\boldsymbol{bileşen2_{i,j}}$	$\boldsymbol{bileşen3_{i,j}}$
$\boldsymbol{x_{b,j}}$	$\varphi 1_{i,j}(x_{i,j} - x_{r1,j})$	$\varphi 2_{i,j}(x_{i,j} - x_{r1,j})$
$\boldsymbol{x_{e,j}}$	$\varphi 1_{i,j}(x_{r1,j} - x_{r2,j})$	$\varphi 2_{i,j}(x_{r1,j} - x_{r2,j})$

Tablo 2
Gözcü Arı Adımında Kullanılan Arama Denklemi Bileşenleri

$\boldsymbol{bileşen1_{i,j}}$	$\boldsymbol{bileşen2_{i,j}}$	$\boldsymbol{bileşen3_{i,j}}$
$\boldsymbol{x_{i,j}}$	$\varphi 1_{i,j}(x_{g,j} - x_{i,j})$	$\varphi 2_{i,j}(x_{g,j} - x_{i,j})$
$\boldsymbol{x_{r1,j}}$	$\varphi 1_{i,j}(x_{g,j} - x_{r1,j})$	$\varphi 2_{i,j}(x_{g,j} - x_{r1,j})$
	$\varphi 1_{i,j}(x_{i,j} - x_{e,j})$	$\varphi 2_{i,j}(x_{i,j} - x_{e,j})$
	0 (kullanma)	0 (kullanma)

Tablo 1 ve 2'de $x_{b,j}$ o ana kadarki en iyi çözümü, $\boldsymbol{x_{e,j}}$ seçilen rasgele bir elit çözümü, $x_{r1,j}$ ve $x_{r2,j}$ birbirinden farklı iki rastgele seçilen çözümü, $\varphi 1$ ve $\varphi 2$ [-a,+a] arasında rastgele seçilen rastgele bir reel sayıyı ifade etmektedir (a sayısı Orijinal ABC algoritmasında 1'e eşit iken bu yaklaşımda bu değer otomatik olarak belirlenebilmektedir). Elit çözümlerin nasıl seçildiği Bölüm 3.3'te açıklanmıştır. Çizelgelerdeki ilk sütunlarda görüldüğü gibi işçi arı adımında çözümler iyi çözümlere, gözcü adımındaki çözümler ise rasgele ve referans çözümlere yaklaştırılmaya çalışılır.

Önerdiğimiz şablon arama denklemi ile farklı kombinasyonlarda işçi ve gözcü arı adımları için birçok arama denklemi üretilebilmektedir. Bileşenlerden rastgele seçilerek oluşturulan arama denklemleri işçi arı ve gözcü arı adımları için iki ayrı arama denklemi havuzunda tutulmaktadır. Her bir iterasyon da havuzlardan sıradaki bir arama denklemi alınarak ilgili adımlarda uygulanır. Orijinal ABC algoritmasından farklı olarak arama denklemi problemin birden fazla boyut değişkeni için uygulanır. Aday çözümde yapılacak değişikliğin oranı MR (Modification Rate) parametresi ile belirlenir. MR değeri [0,1] aralığında olup $\lfloor MR * D \rfloor$ sonucu tek seferde kaç değişkenin arama denklemi için kullanılacağını belirler. Arama denkleminde kullanılacak boyut değişkenleri ise rasgele belirlenir. Eğer uygulanan seçilen boyut değişkenlerinde uygulanan arama denklemi ile referans çözüm iyileştirilmiş ise bu arama denklemi ödüllendirilir. Bunun için aşağıdaki formül kullanılır:

$$s_k = s_k + \omega s_k \qquad (5)$$

Burada s_k başarılı olmuş havuzun k. İndeksindeki arama denkleminin başarı değeri, ω başarı ağırlık değeridir. ω için uygun değer otomatik parametre ayar aracı ile belirlenmiştir. İşçi ve gözcü adımlarına ait arama denklemi havuzlarındaki tüm arama denklemleri kullanıldığında başarısız arama denklemleri problem için uygun olmadığı tespit edildiğinden bir sonraki iterasyonlar için elenir. Bunun için arama denklemleri başarılarına göre büyükten küçüğe sıralanarak havuz büyüklüğü, h, aşağıdaki formüle göre tekrar hesaplanarak azaltılır:

$$h = \frac{h^2}{itr_{max}} \qquad (6)$$

Burada itr_{max} maksimum iterasyon sayısını ifade eder. Bu yaklaşım ile algoritma çalışırken havuzun boyutu doğrusal olarak azaltılır. Böylece algoritma ilerledikçe sadece o problem türü için uygun arama denklemleri hayatta kalır. Hayatta kalan bu arama denklemleri bu strateji ile uyarlanmış bir şekilde tespit edilmiş olur.

3.1.2. Elit bireylerin seçilmesi

Büyük ölçekli sürekli optimizasyon problemlerinde birçok yerel optimum noktaları bulunabilir. Bu noktalardan kaçınmak için farklı uzaklıklardaki iyi çözümlerden yararlanılabilir. Bu çalışmada bu tür çözümler için Elit bireyler ismi kullanılmıştır. Bu bireylerin kaydedildiği listeye de Elit çözümler listesi denilmiştir. Kullandığımız bu elit çözümler listesi her iterasyon sonunda güncellenmektedir. Optimum nokta bilinemediğinden listeye en iyi çözümler eklenerek devam edilir. Ayrıca yeni elit çözüm adayları Elit listesindeki çözümlerden belli bir mesafe uzaklıkta olmalıdır. Dolayısıyla elit çözümün belirlenmesi aşamasında iki parametre önemlidir. Bu parametrelerden birincisi popülasyondaki aday bireylerin elit kabul edilebilmesi için en iyi çözüme ne kadar yaklaştıklarını belirten Omega (Ω) parametresi, diğeri ise listedeki bütün yiyecek kaynaklarından uzaklığının ölçü eşiğini belirten yarıçap (r) parametresidir.

Her bir iterasyon tamamlandığında çözümlerin amaç fonksiyonu olarak değerlendirilir. Amaç fonksiyon değerler en iyi Yiyecek kaynağının amaç fonksiyon değerine Omega' dan daha küçük bir değerse ($|f(x_b) - f(x_i)| < \Omega$) aday çözüm elit çözümler içerisine alınır fakat hemen listeye dahil edilmez. Sonrasında aday elit çözümler elit listesindeki diğer elit bireyler ile karşılaştırılır. Bu durum sonunda aday elit çözümler elit listesinde yer alan bütün elitlerden en az r kadar Öklit uzaklıkta ise bu çözüm elit çözümler listesine eklenir ve liste güncellenir.

3.1.3. Elit bireylerin güncellenmesi

Her bir iterasyon bitiminde elit çözümler listesine farklı ve yeni elit bireyler ilave edilebilir ya da algoritmanın o zamana dek bulduğu en iyi çözüm değiştirilebilir. Bu durumda, listedeki bazı elit çözümlerin amaç fonksiyonu değeri ile yeni keşfedilen en iyi çözümün amaç fonksiyon değeri arasındaki fark Omega'dan büyük olabilmektedir. Bu durumda artık bu koşulu sağlayan çözümler elit olmayacaktır. Elit çözümler listesi güncellenirken bu durum listedeki tüm çözümler için kontrol edilir. $|f(x_b) - f(x_i)| < \Omega$ koşulunu sağlamayan çözümler elit listesinden çıkarılarak elenir.

3.1.4. Yerel arama algoritmasının kullanılması

Yerel arama yöntemleri kullanılarak EIAABC algoritmasının melezleştirme yolu ile iyileştirilmesi, her iterasyon sonunda MTSls1 yerel arama algoritması çağırarak gerçekleştirilir. Şimdiye kadarki en iyi çözüm, genellikle yerel aramanın çağrıldığı ilk çözüm olarak kullanılır. Yerel arama sonucunda, ilk çözümde bir gelişme varsa, şimdiye kadarki en iyi çözümün yerini alır. Yerel arama algoritması, maksimum sayıda yinelemeden sonra, itr_{max} veya toleransın ($FTol$) altında bir iyileşme olduğunda sonlandırılır. Her yerel arama algoritması için uyarlanabilir bir adım boyutu kullanılır. Adım boyutu, rastgele seçilen bir çözümü şimdiye kadarki en iyi çözümden ayıran vektörün maksimum normuna ($\|.\|_\infty$) ayarlanır.

Algoritmanın durağanlığa girebilmesini önlemek için yerel arama algoritması, bir parametresi tarafından kontrol edilen maksimum sayıda tekrarlanan aramadan sonra sonuçları iyileştiremezse, rastgele seçilen bir konuma uygulanır. Yerel arama algoritmalarının orijinal versiyonları, arama uzayı sınır kısıtlarını kontrol etmez. Sınır kısıtlamalarına değerleri yönlendirebilmek için, yerel arama algoritmasında aşağıdaki ceza fonksiyonu kullanılır (De Oca ve diğ., 2011; Liao, De Oca, Aydin, Stützle ve Dorigo, 2011).

$$P(x_i) = fes \times \sum_{j=1}^{D} Bound(x_{i,j}) \quad (7)$$

Burada fes, şimdiye kadar kullanılmış olan fonksiyon değerlendirmelerinin sayısıdır. $Bound(x_{i,j})$ ise şu şekilde tanımlanır:

$$Bound(x_{i,j}) = \begin{cases} 0 & u_j > x_{i,j} > l_j \; ise \\ (l_j - x_{i,j})^2 & x_{i,j} < l_j \; ise \\ (u_j - x_{i,j})^2 & x_{i,j} > u_j \; ise \end{cases} \quad (8)$$

3.2. SOCO11 fonksiyon kümesi ve çalıştırma koşulları

Bu çalışmada 19 adet ölçeklenebilir ve büyük ölçekli fonksiyonundan oluşan SOCO11 ölçüt kümesi kullanılmıştır. Bu fonksiyonlar, F. Herrera ve arkadaşları tarafından Soft Computing Dergisinin 2011'deki özel sayısı için önerilmiştir (Herrera, Lozano ve Molina, 2010). Fonksiyonlara ait matematiksel tanımlar yine Herrera ve diğ. (2010) teknik raporunda ayrıntılı bir şekilde açıklanmaktadır.

Deneyler, 100 ve 500 boyutlu problem örnekleri üzerinde gerçekleştirilmiştir. Hem algoritma parametrelerinin ayarlanması sırasında hem de tüm deneylerde algoritma $5000 \times D$ fonksiyon değerlendirmesi sonrasında sonlandırılmıştır. Ayrıca, hata değeri 10^{-14} eşiğinin altına düştüğünde çalışmalar durdurulmuş ve optimum çözüm bulunduğu kabul edilmiştir. İstatistiksel olarak elde edilen sonuçların güvenli olması için algoritmalar her problemde 25 kez çalıştırılmıştır. Yapılan tüm çalışmada araştırma ve yayın etiğine uyulmuştur.

Geliştirdiğimiz EIAABC ve karşılaştırmalarda kullandığımız tüm ABC algoritmalarına ait ayar

parametre değerleri Yinelenen F-Race algoritması (irace) (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari ve Stützle, 2016; López-Ibáñez, Cáceres, Dubois-Lacoste, Stützle ve Birattari, 2016) kullanılarak otomatik olarak belirlenmiştir. F-Race (Birattari, Yuan, Balaprakash ve Stützle, 2010) parametre ayar algoritması yarışmaya dayalı bir yaklaşımdır. F-Race algoritması belirli bir aday parametre kümesi ile başlar. Daha sonra F-Race'e seçilen problem örnek seti yönlendirilir. Bu, aday yapılandırmaların her test edildiğinde farklı sorun örneklerini gördüğü anlamına gelir. Yarıştaki her aday yapılandırması değerlendirildikten sonra, o ana kadar diğerlerinden önemli ölçüde kötü performans gösteren konfigürasyonların olup olmadığını belirlemek için Friedman testi uygulanır. Yenilenebilir F-Race ile parametre ayarlama yapabilmek için ayarlanacak parametreleri, aralıkları, alanları ve ayarlamanın yapılacağı örnek kümeleri seçmeniz gerekir. Bizim durumumuzda, birçok problem örneği oluşturmak kolay değildir. Bu yüzden 19 problemin 10 boyutlu örnekleri problem örnek kümesi olarak seçilmiştir. Ayarlama sırasında, her adımda, aday yapılandırmalar her fonksiyonda bir kez çalıştırılmıştır.

Yalnızca tüm konfigürasyonlar, 19 fonksiyonun tümü çalıştırıldığında, istatistiksel test uygulanmıştır. Daha sonra kümülatif ortalama çözüm değeri hesaplanmış ve bu değer yinelenen F-Race'nin bir sonraki yinelemesine giden en iyi aday yapılandırmalarını belirlemek için kabaca bir kalite göstergesi olarak kullanılmıştır. İyi bir aday yapılandırması seçerek daha yüksek bir şansa sahip olabilmek için, yinelenen F-Race algoritmasını 10 kez çalıştırmış ve büyük boyutlu problemlere uygulamak için uygun objektif en düşük ortalama değere sahip olanı seçilmiştir. Her bir ABC algoritması için irace ile parametre değerleri Tablo 3'te listelenmiştir.

Tablo 3
Deneylerde Kullanılan Algoritmalara Parametreler ve Özellikleri

Parametre	ABC	BABC	MABC	ImpABC	CABC	OPIABC	EIAABC	Değer Aralıkları	Açıklama
SN	10	6	11	28	17	24	13	3 - 100 (Tam Sayı)	İlk popülayon büyüklük sayısı
lf	2,356	2,164	1,978	1,62	2,819	2,238	2,73	0 - 4 (Reel Sayı)	Sınır faktörü *lf = limit/(SNxD)*
h	–	–	–	–	–	–	100	10-100-500-1000	Arama denklem sayısının büyüklüğü
Yarıçap	–	–	–	–	–	–	0,1	0,1-0,2-0-0,4-0,5	Elit bireyler arası min. yarıçap uzunluğu
SN_{max}	–	–	–	–	–	–	27	20-10 (Tam Sayı)	Maksimum popülasyon büyüklüğü sayısı
ω	–	–	–	–	–	–	0,2	0,1-0,2-0-0,4-0,5	Arama denklemleri için başarı ağırlık değeri
w_{min}	–	0,333	–	–	–	–	–	0 - 0,5 (Reel Sayı)	BABC'nin izci arı adımında kullanılan parametre
w_{max}	–	0,725	–	–	–	–	-	0,5 – 1 (Reel Sayı)	BABC'nin izci arı adımında kullanılan parametre
MR	–	–	0,774	0,408	–	–	0.124	0 – 1 (Reel Sayı)	İşçi arı ve gözcü arı basamaklarında kullanılan modifikasyon oranı
SF	–	–	0,971	–	–	–	–	0 – 2 (Reel Sayı)	MABC'nin işçi arı ve gözcü arı basamaklarında kullanılan modifikasyon oranı
CMap	–	–	–	Tend	Gauss	–	-	7 Kaotik İçerisinden	Kaotik haritalama fonksiyonu
p	–	–	–	0,468	–	–	–	0 – 1 (Reel Sayı)	ImpABC'de bir arama denkleminin seçim olasılığı oranı
LSitr	–	–	–	–	–	–	98	1 – 100 (Tam Sayı)	Yerel arama algoritmasındaki iterasyon sayısı
Ftol	–	–	–	–	–	–	10^{-4}	$10^{(-1,-10)}$ (Tam Sayı)	Yerel arama algoritmasında tolerans değeri

4. Sonuçlar

EIAABC algoritması SOCO11 ölçüt test problemi üzerinde problem çözme standartlarına bağlı kalınarak test edilmiştir. Elde edilen sonuçlar literatürde çokça bilinen Yapay Arı Kolonisi algoritmaları ve SOCO11'e katılan yarışmacı algoritmalar ile karşılaştırılmıştır. Karşılaştırmalarda EIAABC algoritmasının istatistiksel olarak üstünlüğünü göstermek için ikili Wilcoxon testi uygulanmıştır. Buna göre ikili karşılaştırmalarda p-değeri 0.05'den küçük olduğu durumda istatistiksel

olarak iki sonuç arasında farklılık olacağından EIAABC'in daha iyi olduğu durumlarda istatistiksel olarak da anlamlı bir şekilde üstün olduğu gösterilmiştir. Çizelgelerde geçen "Kazanma", "Kaybetme" ve "Beraberlik" karşılaştırılan algoritmanın EIAABC algoritması karşısında sırasıyla kazandığı, kaybettiği ve berabere kaldığı anlamına gelmektedir. Aşağıdaki alt bölümlerde bu karşılaştırma sonuçlarına yer verilmiştir.

4.1. ABC algoritmaları ile karşılaştırma sonuçları

EIAABC algoritması altı tane ABC varyantı ile karşılaştırılmıştır. Karşılaştırma için seçilen algoritmalar şunlardır:

- Orijinal ABC (Karaboğa, 2005): Original Artificial Bee Colony Algorithm (Orijinal ABC algoritması)
- BABC (Banharnsakun ve diğ., 2011): Best-so-far Selection ABC (O ana kadar en iyi seçimli ABC algoritması)
- CABC (Alatas, 2010): Chaotic ABC (Kaotik ABC algoritması)
- ImpABC (Gao ve Liu, 2011): Improved ABC (Gelişmiş ABC algoritması)
- MABC (Akay ve Karaboğa, 2012): Modified ABC (Değiştirilmiş ABC)
- OPIABC (Zhang ve Yuen, 2013): One Point Inheritance ABC (Tek Noktada Kalıtımlı ABC algoritması)

100 boyutlu problemler üzerinden elde edilen sonuçlar Tablo 4'te gösterilmektedir.

Tablo 4
100 Boyutlu SOCO11 Fonksiyonlarında EIAABC ve ABC Varyantları Arasındaki Karşılaştırma Sonuçları

Fonksiyonlar	ABC	BABC	MABC	ImpABC	CABC	OPIABC	EIAABC
f1	**1.00E-14**	**1.00E-14**	**1.00E-14**	1.44E+00	**1.00E-14**	**1.00E-14**	**1.00E-14**
f2	3.71E+01	4.41E+01	6.29E+01	2.69E+01	3.68E+01	2.31E+01	**3.29E-07**
f3	4.33E+01	1.88E+02	1.81E+02	1.71E+02	8.89E+00	1.80E+01	**2.18E+00**
f4	2.45E+00	8.27E+01	1.88E+02	4.44E+02	**1.00E-14**	**1.00E-14**	**1.00E-14**
f5	**1.00E-14**	5.91E-02	**1.00E-14**	9.22E-01	**1.00E-14**	**1.00E-14**	**1.00E-14**
f6	**1.00E-14**	2.13E-11	8.29E-08	8.82E+00	**1.00E-14**	**1.00E-14**	**1.00E-14**
f7	**1.00E-14**	**1.00E-14**	1.47E-14	1.06E-14	**1.00E-14**	**1.00E-14**	**1.00E-14**
f8	3.21E+04	5.04E+03	7.03E+04	1.22E+03	3.87E+04	3.14E+04	**2.66E+02**
f9	**4.36E-07**	1.48E+01	1.58E+02	3.39E+02	8.49E-04	6.77E-04	1.35E-01
f10	**1.00E-14**	4.16E+00	2.08E+00	2.21E+01	**1.00E-14**	**1.00E-14**	**1.00E-14**
f11	**6.12E-07**	2.33E-01	1.39E+02	3.49E+02	9.41E-04	8.12E-05	1.24E-01
f12	2.14E-08	3.36E-05	1.41E+00	1.88E+02	9.62E-05	2.41E-06	**1.00E-14**
f13	2.11E+00	6.81E+01	2.10E+02	3.52E+02	8.30E-01	**4.08E-01**	4.96E+00
f14	2.01E+00	9.51E+01	1.39E+02	3.31E+02	4.19E-06	1.88E+00	**1.00E-14**
f15	**1.00E-14**	**1.00E-14**	2.71E-14	3.21E+00	**1.00E-14**	**1.00E-14**	**1.00E-14**
f16	**2.04E-08**	1.11E-02	2.37E+01	3.51E+02	2.30E-04	5.50E-03	4.15E-08
f17	1.49E+01	3.01E+01	3.22E+02	5.89E+02	2.39E+00	**1.61E+00**	8.38E+00
f18	9.89E-01	5.27E+01	9.71E+01	1.23E+02	**1.09E-05**	9.88E-01	4.35E-02
f19	**1.00E-14**	7.21E+00	2.11E+00	1.85E+01	**1.00E-14**	**1.00E-14**	**1.00E-14**
Optimum	**7**	**3**	**2**	**0**	**8**	**8**	**10**
Kazanma	**4**	**0**	**0**	**0**	**4**	**4**	
Beraberlik	**7**	**3**	**2**	**0**	**8**	**8**	
Kaybetme	**8**	**16**	**17**	**19**	**7**	**7**	
p-değeri	**0.0466**	**0 .00044**	**0.0003**	**0.00014**	**0.4777**	**0.18352**	

Buna göre, EIAABC algoritması 19 problemin 10 tanesinde optimum sonuç elde etmiştir. Buna karşılık OPIABC ve CABC 8, ABC 7, MABC 3 ve BABC 2 problemde optimum sonuç elde etmiştir. ImpABC ise hiç bir problemde optimum sonuç verememektedir. İkili olarak karşılaştırma yapılırsa;

- Orijinal ABC ile karşılaştırıldığında, 8 problemde EIAABC algoritmasının daha iyi, 4 problemde daha kötü, 7 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda EIAABC algoritmasının istatistiksel olarak daha iyi olduğu tespit edilmiştir (p= 0.0466).
- BABC ile karşılaştırıldığında, 16 problemde EIAABC algoritmasının daha iyi, 3 problemde ise aynı sonucu verdiği görülmüştür. Hiçbir problemde EIAABC algoritmasından daha iyi bir sonuç alamamıştır. Wilcoxon testinde de bu fark istatistiksel bir üstünlük olarak da gösterilmiştir (p= 0.00044).
- MABC ile karşılaştırıldığında, 17 problemde EIAABC algoritmasının daha iyi, 2 problemde ise aynı sonucu verdiği görülmüştür. Hiçbir

problemde EIAABC algoritmasından daha iyi bir sonuç alamamıştır. İkili istatistik testlerinde de EIAABC algoritmasının üstünlüğü net bir şekilde görülmüştür (p= 0.0003).

- ImpABC ile karşılaştırıldığında, 19 problemin tamamında kazanmış kaybettiği veya berabere kaldığı hiçbir problem bulunmamaktadır. Wilcoxon testinde yine EIAABC algoritmasının istatistiksel olarak daha iyi olduğu görülmektedir (p= 0.00014).
- CABC ile karşılaştırıldığında, 7 problemde EIAABC algoritmasının daha iyi, 4 problemde daha kötü, 8 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda iki algoritma arasında istatistiksel olarak bir üstünlük tespit edilememiştir (p= 0.4777)
- OPIABC ile karşılaştırıldığında, 7 problemde EIAABC algoritmasının daha iyi, 4 problemde daha kötü, 8 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda iki algoritma arasında istatistiksel olarak bir üstünlük tespit edilememiştir (p= 0.18352)

500 boyutlu problemler üzerinden elde edilen sonuçlar Tablo 5'te gösterilmektedir. Buna göre, EIAABC algoritması 19 problemin 8 tanesinde optimum sonuç elde etmiştir. Buna karşılık, CABC 7, ABC 6, OPIABC 4, BABC 1 problemde optimum sonuç elde etmiştir. ImpABC ve MABC ise hiçbir problemde optimum sonuç verememektedir. İkili olarak karşılaştırma yapılırsa;

- ABC ile karşılaştırıldığında, 3 problemde EIAABC algoritmasının daha iyi, 9 problemde daha kötü, 7 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda iki algoritma arasında istatistiksel olarak bir üstünlük tespit edilememiştir (p= 0.238)
- BABC ile karşılaştırıldığında, 15 problemde EIAABC algoritmasının daha iyi, 3 problemde daha kötü, 1 problemde ise aynı sonucu verdiği görülmüştür. Ayrıca, EIAABC'nin istatistiksel olarak da BABC algoritmasından iyi olduğu gözlemlenmiştir (p= 0.00988).
- MABC ile karşılaştırıldığında, 19 problemin tamamında EIAABC algoritmasının kazandığı kaybettiği veya berabere kaldığı hiçbir problem olmadığı görülmektedir. Algoritmanın üstünlüğü istatistiksel testler sonucunda da açıkça görülebilmektedir (p= 0.00014).
- ImpABC ile karşılaştırıldığında, 18 problemde EIAABC algoritmasının daha iyi, 1 problemde daha kötü görülmüş, hiçbir problemde aynı sonuca ulaşamadıkları gözlemlenmiştir. Wilcoxon testine tabii tutulduğunda EIAABC algoritmasının istatistiksel olarak üstünlük sağladığı tespit edilmiştir (p= 0.00168).
- CABC ile karşılaştırıldığında, 3 problemde EIAABC algoritmasının daha iyi, 9 problemde daha kötü, 7 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda iki algoritma arasında istatistiksel olarak bir üstünlük tespit edilememiştir (p= 0.238)

OPIABC ile karşılaştırıldığında, 7 problemde EIAABC algoritmasının daha iyi, 8 problemde daha kötü, 4 problemde ise aynı sonucu verdiği görülmüştür. Wilcoxon testine tabii tutulduğunda iki algoritma arasında istatistiksel olarak bir üstünlük tespit edilememiştir (p= 0.30772)

Şekil 1'de dört fonksiyon üzerinden algoritmaların yakınsama grafikleri verilmiştir. Seçilen dört fonksiyon farklı özellikte olup her birinde birden fazla ABC algoritması iyi sonuçlar vermiştir. Burada, f1 fonksiyonu tek tepeli (unimodal), f5 fonksiyonu çok tepeli (unimodal), f14 ve f19 fonksiyonları ise hibrit fonksiyonlardır. Şekil 1 incelendiğinde EIAABC algoritmasının diğerlerine göre daha hızlı bir şekilde yakınsadığı görülmektedir. BABC algoritmasının da f1, f5 ve f19 gibi problemlerde hızlı bir şekilde yakınsadığı görülse de diğer problemlerde (f14 örneğinde olduğu gibi) hızlı bir şekilde algoritmanın durağanlığa girdiği ve yerel minimum noktaları aşamadığı gözlemlenmektedir. Öte yandan, önerilen algoritma ile rekabetçi sonuçlar veren CABC algoritmasının yavaş bir şekilde yakınsadığı görülmektedir. Sonuç olarak Şekil 1, EIAABC algoritmasının karşılaştırılan ABC algoritmalarına göre daha hızlı bir şekilde yakınsadığını ve yerel minimum noktalardan daha iyi kaçabildiğini göstermektedir.

4.2. SOCO11 yarışmasına katılan algoritmalar ile karşılaştırma sonuçları

SOCO11 yarışmasına 13 tane algoritma katılmıştır. Ayrıca 2011 yılından önceki başka yarışmalarda birincilik elde eden üç algoritma ve sonuçları yarışmada referans olarak verilmiştir. Bu algoritmalar "A restart CMA evolution strategy with increasing population size (GCMAES)", "Diferansiyel Gelişim (DE)" ve "Real Coded Genetic Algorithm (CHC)"'dir.

100 boyutlu problemler üzerinden elde edilen sonuçların özeti Tablo 6'da gösterilmektedir. Buna göre, SEABC algoritması 19 problemin 10 tanesinde optimum sonuç elde etmiştir. Buna karşılık MOS 14, DEDM ve GaDE 11, GODE, EM323 ve VXQR 6 MASSW 10, RPSOvm 5, SOUPDE ve IPSOLS 9, EVOPROpt 3, jDElscop 13, SaDEMMTS 12 problemde optimum sonuç elde etmiştir. Optimum sonuç veremeyen hiçbir algoritma olmamıştır. İkili olarak karşılaştırma yapılırsa EISEABC algoritmasının VXQR, RPSOvm ve

EVOPROpt algoritmalarından istatistiksel olarak daha iyi sonuçlar verdiği görülmüştür.

500 boyutlu problemler üzerinden elde edilen sonuçların özeti Tablo 7'de gösterilmektedir. Buna göre, SEABC algoritması 19 problemin 8 tanesinde optimum sonuç elde etmiştir. Buna karşılık MOS 14, DEDM ve GODE 6, GaDE 9, EM323 ve MASSW 4, VXQR ve SOUPDE 5, RPSOvm 3, ve IPSOLS 8, EVOPROpt 2, jDElscop 12, SaDEMMTS 10 problemde optimum sonuç elde etmiştir. optimum sonuç veremeyen hiçbir algoritma olmamıştır. İkili olarak karşılaştırma yapıldığında EISEABC algoritmasının EM323, MASSW, VXQR ve RPSOvm algoritmalarından istatistiksel olarak daha iyi sonuçlar verdiği görülmüştür.

Tablo 5
500 Boyutlu SOCO11 Fonksiyonlarında EIAABC ve ABC Varyantları Arasındaki Karşılaştırma Sonuçları

Fonksiyonlar	ABC	BABC	MABC	ImpABC	CABC	OPIABC	EIAABC
f1	**1.00E-14**	**1.00E-14**	4.61E+00	2.04E+03	**1.00E-14**	**1.00E-14**	**1.00E-14**
f2	1.04E+02	4.89E+01	1.41E+02	4.69E+01	1.12E+02	9.17E+01	**3.31E+00**
f3	2.29E+01	4.37E+02	4.70E+04	2.42E+08	**2.33E+00**	1.41E+01	6.85E+01
f4	2.11E-06	9.35E+02	2.18E+03	3.61E+03	3.01E-08	1.99E+00	**1.00E-14**
f5	**1.00E-14**	3.52E-10	7.37E-01	2.19E+01	**1.00E-14**	**1.00E-14**	**1.00E-14**
f6	**1.00E-14**	1.10E+00	1.84E+01	1.88E+01	**1.00E-14**	2.43E-10	**1.00E-14**
f7	**1.00E-14**	**1.00E-14**	1.31E-02	1.43E+01	**1.00E-14**	1.99E-05	**1.00E-14**
f8	4.41E+05	9.17E+04	1.44E+06	**8.44E+04**	5.22E+05	5.56E+05	1.60E+05
f9	1.19E-01	3.11E+02	3.59E+03	3.04E+03	**6.55E-03**	1.42E-02	4.15E+00
f10	**1.00E-14**	4.18E+01	4.77E+01	1.26E+02	**1.00E-14**	**1.00E-14**	**1.00E-14**
f11	1.40E-01	1.47E+02	3.51E+03	2.89E+03	**6.29E-03**	4.42E-02	5.24E+00
f12	2.69E-03	6.31E-02	9.55E+02	2.71E+03	**7.45E-04**	1.55E-01	2.50E-01
f13	4.41E+01	6.64E+02	2.32E+03	7.13E+07	**7.86E+00**	1.27E+01	8.83E+01
f14	9.89E-01	6.39E+02	1.78E+03	2.80E+03	**8.86E-05**	7.74E+00	3.19E+00
f15	**1.00E-14**	1.11E+00	7.61E+00	6.09E+01	**1.00E-14**	4.38E-03	**1.00E-14**
f16	2.99E-02	1.79E-01	1.89E+03	3.01E+03	**3.01E-03**	9.20E-02	2.31E+00
f17	6.31E+00	1.20E+02	3.50E+03	4.40E+03	**5.99E-01**	2.08E+00	6.73E+01
f18	1.11E+00	5.53E+02	9.41E+02	1.17E+03	**1.98E-04**	3.05E+00	7.10E+00
f19	**1.00E-14**	2.07E+00	2.46E+01	1.21E+02	**1.00E-14**	**1.00E-14**	**1.00E-14**
Optimum	**6**	**2**	**0**	**0**	**7**	**4**	**8**
Kazanma	**3**	**3**	**0**	**1**	**9**	**8**	
Beraberlik	**7**	**1**	**0**	**0**	**7**	**4**	
Kaybetme	**9**	**15**	**19**	**18**	**3**	**7**	
p-değeri	**0.238**	**0.00988**	**0.00014**	**0.00168**	**0.238**	**0.30772**	

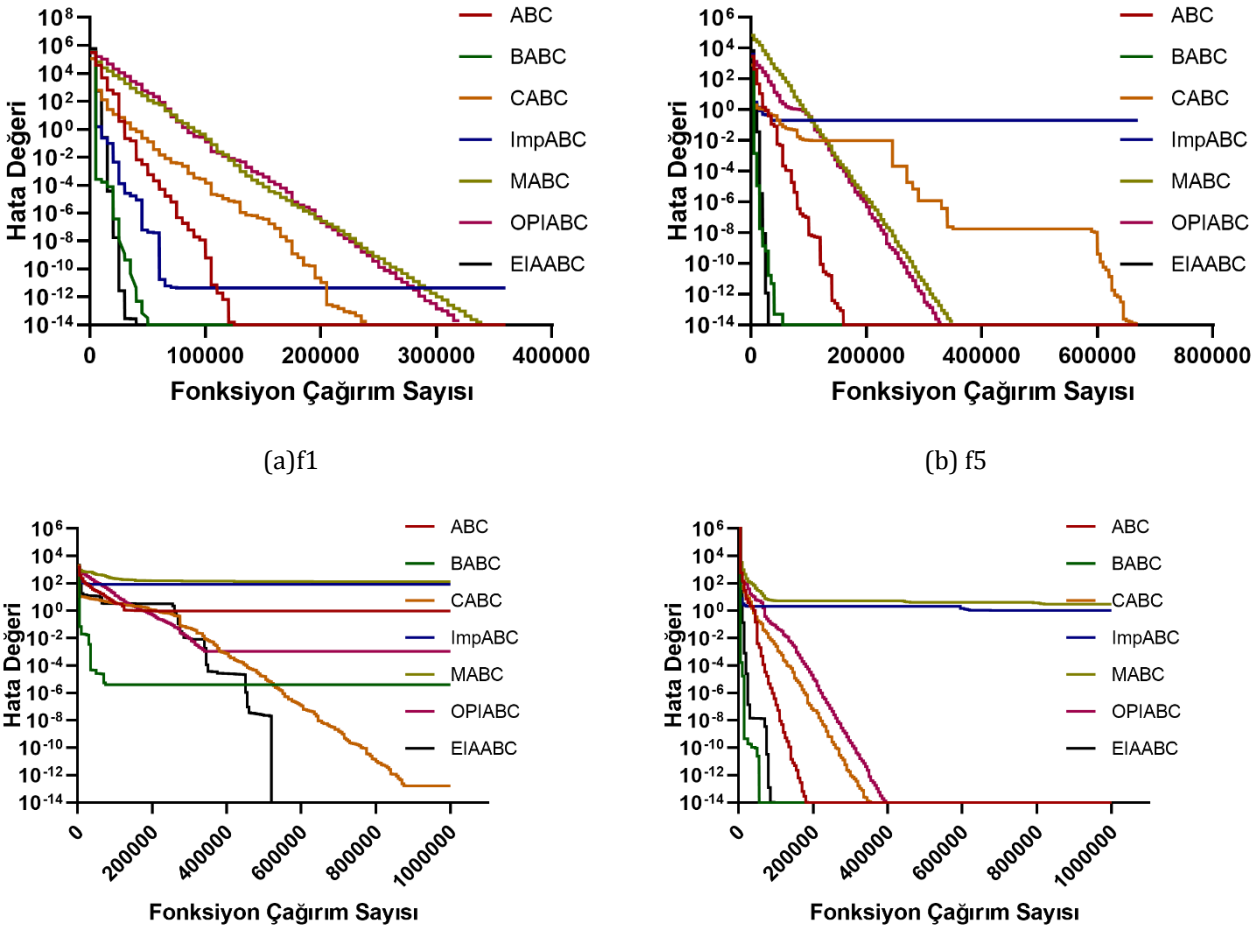
Tablo 6
100 Boyutlu SOCO11 Fonksiyonlarında EIAABC ve Yarışmacı Algoritmalar Arasındaki Karşılaştırma Sonuçları

	MOS	DEDM	GaDE	GODE	EM323	MASSW	VXQR	RPSOvm	SOUPDE	EVOPROpt	jDElscop	SaDEMMTS	IPSOLS	EIAABC
Optimum	**14**	**11**	**11**	**6**	**6**	**10**	**6**	**5**	**9**	**3**	**13**	**12**	**9**	**10**
Kazanma	**7**	**5**	**6**	**5**	**4**	**3**	**2**	**5**	**5**	**-**	**6**	**5**	**9**	
Beraberlik	**10**	**8**	**9**	**6**	**4**	**9**	**5**	**5**	**7**	**3**	**9**	**9**	**5**	
Kaybetme	**2**	**6**	**4**	**8**	**11**	**7**	**12**	**9**	**7**	**16**	**4**	**5**	**5**	
p-değeri	**1,007**	**0,131**	**0,509**	**0,134**	**0,078**	**0,139**	**0,022**	**0,011**	**0,116**	**0,0004**	**0,447**	**0,803**	**0,0009**	

5. Tartışma

Bu çalışmada büyük ölçekli optimizasyon problemlerinin çözümü için "Elit Birey Tabanlı Uyarlanabilir Yapay Arı Kolonisi Algoritması" (EIAABC) algoritması geliştirilmiştir. Bu Algoritma geliştirilirken, işçi arı ve gözcü arı adımlarında kullanılan arama denkleminde iyileştirmeler yapılmıştır ve bu denklemlerde elit bireylerden yararlanılmıştır. Ayrıca bir yerel arama tekniği ile algoritma performansı güçlendirilmiştir.

SOCO11 büyük ölçekli optimizasyon problem kümesi üzerinden algoritmaya ait deneysel sonuçlar elde edilmiştir. Çalışmalarda SOCO11 şartnamesinde vurgulanan çalıştırma koşulları dikkate alınmış ve algoritmamıza ait parametre ayarları irace aracı ile otomatik olarak gerçekleştirilmiştir. EISEABC algoritmasının performansı literatürdeki ABC varyantları ve birçok yarışmacı algoritma ile kıyaslanmıştır. Deneysel sonuçlarda, geliştirdiğimiz algoritmanın hem birçok ABC varyantından hem de SOCO11 büyük ölçekli optimizasyon problem kümesi yarışmasına ortamına katılan birçok algoritmadan daha yüksek performans sergilediği görülmüştür.



(a)f1 (b) f5

(c)f14 (d) f19

Şekil 1. Farklı Fonksiyonlar Üzerinde ABC Algoritmalarının Yakınsama Grafikleri

Tablo 7

500 Boyutlu SOCO11 Fonksiyonlarında EIAABC ve Yarışmacı Algoritmalar Arasındaki Karşılaştırma Sonuçları

	MOS	DEDM	GaDE	GODE	EM323	MASSW	VXQR	RPSOvm	SOUPDE	EVOPROpt	jDElscop	SaDEMMTS	IPSOLS	EIAABC
Optimum	**14**	**6**	**9**	**6**	**4**	**4**	**5**	**3**	**5**	**2**	**12**	**10**	**8**	**8**
Kazanma	**10**	**6**	**8**	**7**	**6**	**5**	**2**	**8**	**7**	**1**	**8**	**6**	**10**	
Beraberlik	**9**	**5**	**7**	**6**	**4**	**4**	**4**	**4**	**3**	**2**	**7**	**6**	**5**	
Kaybetme	**-**	**8**	**4**	**6**	**9**	**10**	**13**	**7**	**9**	**16**	**4**	**7**	**4**	
p-değeri	**0,003**	**0,096**	**0,307**	**0,347**	**0,026**	**0,030**	**0,012**	**0,126**	**0,062**	**0,005**	**0,347**	**0,638**	**0,001**	

Çıkar Çatışması

Yazarlar tarafından herhangi bir çıkar çatışması beyan edilmemiştir.

Araştırmacıların Katkısı

Bu araştırmada; Doğan AYDIN, algoritma tasarımı, deneysel ortamın ve makalenin hazırlanmasında ve deney sonuçlarının yorumlanmasında; Ümit GÜVEN algoritmanın gerçekleştirimi, deney ve analizlerin gerçekleştirimi ve makalenin genel kontrolünün yapılması konularında katkı sağlamışlardır.

Kaynaklar

Akay, B. ve Karaboga, D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, 192, 120–142. doi: https://doi.org/10.1016/j.ins.2010.07.015

Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization, *Expert Systems with Applications*, 37(8), 5682–5687. doi: https://doi.org/10.1016/j.eswa.2010.02.042

Aydın, D., Liao, T., De Oca, M. A. M. ve Stützle, T. (2011). Improving performance via population growth and local search: the case of the artificial bee colony algorithm. *Proceeding of International Conference on Artificial Evolution (Evolution Artificielle),* 85-96. doi: https://doi.org/10.1007/978-3-642-35533-2_8

Aydın, D., Yavuz, G. ve Stützle. T. (2017). ABC-X: a generalized, automatically configurable artificial bee colony framework. *Swarm Intelligence,* 11(1), 1-38. doi: https://doi.org/10.1007/s11721-017-0131-z

Badem, H., Basturk, A., Caliskan, A. ve Yuksel, M. E. (2018). A new hybrid optimization method combining artificial bee colony and limited-memory BFGS algorithms for efficient numerical optimization. *Applied Soft Computing*, 70, 826-844. doi: https://doi.org/10.1016/j.asoc.2018.06.010

Banharnsakun, A., Achalakul, T. ve Sirinaovakul, B. (2011). The best-so-far selection in Artificial Bee Colony algorithm, *Applied Soft Computing*, 11, 2888–2901. doi: https://doi.org/10.1016/j.asoc.2010.11.025

Birattari, M., Yuan, Z., Balaprakash, P. ve Stützle, T. (2010). F-race and iterated F-race: An overview, *Experimental Methods for the Analysis of Optimization Algorithms* 311–336. Springer Berlin Heidelberg. doi: https://doi.org/10.1007/978-3-642-02538-9_13

De Oca, M. A. M., Aydın, D. ve Stützle, T. (2011). An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re) design of optimization algorithms, *Soft Computing*, 15(11), 2233–2255. doi: https://doi.org/10.1007/s00500-010-0649-0

Diwold, K., Aderhold, A., Scheidler, A. ve Middendorf, M. (2011). Performance evaluation of artificial bee colony optimization and new selection schemes, *Memetic Computing*, 3(3), 149. doi: https://doi.org/10.1007/s12293-011-0065-8

Gao, W. ve Liu, S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters*, 111(17), 871–882.

Gao, W. F., Huang, L. L., Liu, S. Y., Chan, F. T., Dai, C. ve Shan, X. (2015). Artificial bee colony algorithm with multiple search strategies. *Applied Mathematics and Computation*, 271, 269-287. doi: https://doi.org/10.1016/j.amc.2015.09.019

Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J. ve Zhou, M. (2021). Chaotic local search-based differential

evolution algorithms for optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(6), 3954-3967. doi: https://doi.org/10.1109/TSMC.2019.2956121

Gass, S. I. (2000). Making decisions with precision, *Business Week*, 1(1), 45. Erişim adresi: https://www.bloomberg.com/news/articles/2000-10-29/making-decisions-with- precision

Gökalp, O. ve Uğur, A. (2020). A high-level and adaptive metaheuristic selection algorithm for solving high dimensional bound-constrained continuous optimization problems. *Turkish Journal of Electrical Engineering & Computer Sciences*, 28(3), 1549-1566. https://doi.org/10.3906/elk-1908-9

Gungor, I., Emiroglu, B. G., Cinar, A. C. ve Kiran, M. S. (2020). Integration search strategies in tree seed algorithm for high dimensional function optimization. *International Journal of Machine Learning and Cybernetics*, 11(2), 249-267. doi: https://doi.org/10.1007/s13042-019-00970-1

Herrera, F., Lozano, M. ve Molina, D. (2010). Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems Functions F1-F11, 1–12. Erişim adresi: https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/testfunctions-SOCO.pdf

Kang, F., Li, J., & Ma, Z. (2011). Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Information Sciences*, 181(16), 3508-3531. doi: https://doi.org/10.1016/j.ins.2011.04.024

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization, *Technical report-tr06*, Erciyes University, Engineering faculty, computer. Erişim adresi: https://abc.erciyes.edu.tr/pub/tr06_2005.pdf

Karaboga, D. ve Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39(3), 459-471. doi: https://doi.org/10.1007/s10898-007-9149-x

Kiran, M. S., Hakli, H., Gunduz, M. ve Uguz, H. (2015). Artificial bee colony algorithm with variable search strategy for continuous optimization. *Information Sciences*, 300, 140-157. doi: https://doi.org/10.1016/j.ins.2014.12.043

Li, D., Guo, W., Lerch, A., Li, Y., Wang, L. ve Wu, Q. (2021). An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. *Swarm and Evolutionary Computation*, 60, 100789. doi: https://doi.org/10.1016/j.swevo.2020.100789

Liao, T., Aydin, D. & Stützle, T. (2013). Artificial bee colonies for continuous optimization: Experimental analysis and improvements, *Swarm Intelligence*. 7(4), 327-356. doi: https://doi.org/10.1007/s11721-013-0088-5

Liao, T., De Oca, M. A. M., Aydin, D., Stützle, T. ve Dorigo, M. (2011). An incremental ant colony algorithm with local search for continuous optimization, *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 125–132. doi: https://doi.org/10.1145/2001576.2001594

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, 108-126. doi: https://doi.org/10.1016/j.eswa.2018.11.032

Lozano, M., Molina, D. ve Herrera, F. (2011). Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15(11), 2085-2087. doi: https://doi.org/10.1007/s00500-010-0639-2

López-Ibáñez, M., Cáceres, L. P., Dubois-Lacoste, J., Stützle, T. ve Birattari, M. (2016). The irace package: User guide, *Technical Report TR/IRIDIA/2016-004*. IRIDIA, Université Libre de Bruxelles, Belgium. Erişim adresi: https://cran.r-project.org/web/packages/irace/vignettes/irace-package.pdf

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M. ve Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration, *Operations Research Perspectives*, 3, 43–58. doi: https://doi.org/10.1016/j.orp.2016.09.002

Meselhi, M. A., Elsayed, S. M., Sarker, R. A. ve Essam, D. L. (2020). Contribution Based Co-Evolutionary Algorithm for Large-Scale Optimization Problems. *IEEE Access*, 8, 203369-203381. doi: https://doi.org/10.1109/ACCESS.2020.3036438

Serkan, K. ve Fığlalı, N. (2016). Çok Amaçlı Optimizasyon Problemlerinde Pareto Optimal Kullanımı, *Sosyal Bilimler Araştırma Dergis*i, 5(2), 9–18. Erişim adresi: https://dergipark.org.tr/tr/pub/ssrj/issue/29053/310744

Tseng, L., Chen, C. ve Definition, A. P. (2008). Multiple Trajectory Search for Large Scale Global Optimization, *Proceedings of 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence),* 3052–3059. doi: https://doi.org/10.1109/CEC.2008.4631210

Winston, W. L. (2003). *Operations research: applications and algorithms*, International Thomson Publishing, Belmont, California.

Xue, Y., Jiang, J., Zhao, B. ve Ma, T. (2018). A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Computing*, 22(9), 2935-2952. doi: https://doi.org/10.1007/s00500-017-2547-1

Yang, P., Tang, K. ve Yao, X. (2019). A parallel divide-and-conquer-based evolutionary algorithm for large-scale optimization. *IEEE Access*, 7, 163105-163118. doi: https://doi.org/10.1109/ACCESS.2019.2938765

Yavuz, G. ve Aydın, D. (2019). Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy. *Swarm and Evolutionary Computation*, 51, 100582. doi: https://doi.org/10.1016/j.swevo.2019.100582

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, 533-544. doi: https://doi.org/10.1016/j.ins.2018.10.046

Zhang, X. ve Yuen, S. Y. (2013). Improving artificial bee colony with one-position inheritance mechanism, *Memetic Computing*, 5(3), 187–211. doi: https://doi.org/10.1007/s12293-013-0117-3