Modifiye Yapay Arı Koloni Algoritması ile Nümerik Fonksiyon Optimizasyonu
Modified Artificial Bee Colony Algorithm for Numerical Function Optimization

Bilal Babayiğit[1], Resul Özdemir[2]

[1]Bilgisayar Mühendisliği Bölümü
Erciyes Üniversitesi
bilalb@erciyes.edu.tr

[2]ÜSET Meslek Yüksekokulu
Nevşehir Üniversitesi
resulozdemir@nevsehir.edu.tr

Özet

Yapay arı koloni (Artificial Bee Colony, ABC) algoritması son yıllarda oldukça popüler olmuş bir optimizasyon algoritmasıdır. Popülasyon-tabanlı mevcut algoritmalara göre daha iyi sonuçlar üreten ABC algoritması az sayıda kontrol parametresi içermesi ve kolay uygulanabilir olması sebebiyle çok çeşitli optimizasyon problemlerinin çözümünde kullanılmıştır. Son yıllarda yapılan çalışmalar ABC algoritmasının yeni çözümleri keşfetme mekanizmasının çok iyi çalıştığını fakat komşu çözümler arasında yerel araştırma yapma mekanizmasının geliştirilebileceğini ortaya koymuştur. Bu sebeple, bu çalışmada ABC algoritması modifiye edilerek, standart ABC algoritmasında gözcü arıların seleksiyonda kullandıkları ve çözümün kalitesiyle ters orantılı olan yeni bir olasılık hesaplama yöntemi ile yeni arama mekanizmaları sunulmuştur. Sunulan yeni yaklaşımın başarımı yedi adet nümerik optimizasyon probleminde test edilmiştir. Sonuçlar modifiye ABC algoritmasının standart ABC algoritmasına göre çözüm kalitesi ve yakınsama hızı bakımından daha iyi çözümler üretttiğini göstermiştir.

Abstract

Artificial bee colony (ABC) algorithm has become an increasingly popular optimization algorithm in recent years. Due to the advantages of employing fewer control parameters and ease of implementation, ABC algorithm which outperforms other population-based algorithms can be easily used for solving various kinds of optimization problems. Recent studies revealed that ABC is good at investigating the new solutions (exploration) but poor at local search between neighbouring solutions (exploitation). For this reason, in this paper a modified ABC is proposed by using a new probability calculation function which is inversely proportional to solution quality and new searching mechanisms. Performance of the modified ABC algorithm is tested on seven numerical optimization problems. Results show that the modified ABC algorithm outperforms the ABC algorithm on solution quality and faster convergence.

1. Giriş

Sürü zekâsı, birbirleriyle etkileşim içerisinde olan bireylerin topluca sergilemiş oldukları zeki davranışlardır. Sürü içerisindeki bireyler, kendi deneyimleriyle veya sürünün diğer elemanlarının davranışlarını gözlemleyerek karşılaştıkları problemlerin üstesinden kolaylıkla gelebilirler. Sürü içerisindeki bireylerin bu şekilde sergilemiş oldukları zeki davranışlar birçok problemin çözülmesinde esin kaynağı olmuştur [1].

Yapay arı koloni algoritması (ABC) 2005 yılında Karaboga tarafından ortaya atılan sürü zekasına dayalı bir optimizasyon tekniğidir [1]. Az sayıda parametre içermesi, kolay uygulanabilir ve sade bir çalışma prensibine sahip olan ABC algoritması araştırmacılar tarafından çok çeşitli optimizasyon problemlerinin çözümünde kullanılmıştır [2–8].

Son yıllarda yapılan çalışmalarda [7,8], ABC algoritmasının yeni çözümler keşfetme mekanizmasının iyi olduğu ancak geliştirilmesi gereken yönleri bulunduğu belirtilmiştir. Özellikle yerel çözümlerden yararlanma mekanizmasının geliştirilebilir olduğu ortaya konulmuştur.

Bu çalışmada, standart ABC algoritmasının yerel çözümlerden yararlanma mekanizmasını geliştirerek performansını artırmak amacıyla modifiye bir ABC algoritması önerilmiştir. Önerilen modifiye ABC algoritmasında, standart ABC algoritmasında gözcü arıların seleksiyonundaki olasılık hesabı yerine yeni bir olasılık hesaplama yöntemi ile yeni arama mekanizmaları kullanılmıştır.

2. Yapay Arı Koloni Algoritması

ABC algoritması, bal arıların zeki yiyecek arama davanışlarını modelleyen bir optimizasyon algoritmasıdır. ABC algoritmasına göre yapay arı kolonisi; görevli arılar, gözcü arılar ve kâşif arılar olmak üzere üç çeşit arıdan oluşmaktadır. ABC algoritmasının temel işleyişi şu şekildedir. İlk olarak başlangıç yiyecek kaynakları rastgele oluşturulur. Görevli arılar birer yiyecek kaynağı seçerler ve nektar depolayıp kovana dönerler. Gözcü arılar kovana gelen görevli arıların danslarını izleyerek belirli olasılık ile yiyecek kaynağını seçerler. Seçilen yiyecek kaynağına yönelen gözcü arılar, görevli arılar gibi nektar depolamaya başlar. Belirli bir deneme sayısı içerisinde (*limit*) yiyecek kaynaklarını tüketen görevli arılar yeni kaynaklar aramak için kâşif arı olurlar. Kâşif arılar da rastgele bir yiyecek kaynağı bularak nektar depolama işlemine devam eder. Bütün bu adımlar algoritmanın bir çevrimini oluşturur ve sonlanma kriteri sağlanana kadar bu adımlar devam eder.

ABC algoritmasında temelde kullanılan iki adet kontrol parametresi vardır: "*popsize*" ve "*limit*". "*popsize*" algoritmada kullanılacak popülâsyonun sayısıdır. *limit* ise görevli arıların yiyecek kaynağını terk etmesi için belirlenmiş deneme sayısı değeridir. Yani bir kaynağı ifade eden çözüm *limit* değeri kadar deneme ile geliştirilememişse bu kaynak terk edilir ve bu kaynağa gidip gelen görevli arı kaşif arı haline gelir. Algoritmada görevli arıların sayısı gözcü arıların sayısına eşittir. Aynı zamanda görevli arıların sayısı yiyecek kaynaklarının sayısına (*NS*) eşittir. Yiyecek kaynaklarının sayısı popülasyon sayısının yarısıdır (*NS*=*popsize*/2). Algoritmada yiyecek kaynağının konumu olası bir çözümü, sahip olduğu nektar miktarı ise çözümün kalitesini temsil etmektedir. ABC algoritmasının temel adımları Algoritma 1'de verilmiştir.

Algoritma 1: ABC Algoritması temel adımları

1: Başlangıç yiyecek kaynaklarının rastgele belirlenmesi
2: **do**
3: Görevli arılan yiyecek kaynaklarına gönderilmesi
4: Seleksiyonda kullanacak yiyecek kaynaklarının olasılık değerlerinin hesaplanması
5: Gözcü arıların olasılık değerlerine göre seçtikleri yiyecek kaynaklarına gönderilmeleri
6: Kaşif arıların yeni yiyecek kaynaklarını bulmaya çıkması
7: **while** (sonlandırma koşulu sağlandı mı)

ABC algoritmasının Algoritma 1'de verilen temel adımları incelendiğinde ilk adımda rastgele yiyecek kaynakları şu şekilde oluşturulur:

$$x_{i,j} = x_j^{min} + rand(0,1)\left(x_j^{max} - x_j^{min}\right). \quad (1)$$

Burada, i = 1...*NS*, j = 1...*D*. *NS* yiyecek kaynaklarının sayısını, *D* ise optimize edilecek olan problemin parametre sayısını ifade eder. x^{min} ve x^{max} sırasıyla j. sıradaki parametrenin alt ve üst limitleridir. Başlangıçta bütün yiyecek kaynakları için geliştirilememeyi ifade eden sayaç (*trial*) sıfırlanır.

Görevli arılar yeni yiyecek kaynağı arama işleminde komşu bir çözüm bulur ve mevcut çözümle yeni çözüm arasında kıyaslama yapar. Eğer yeni çözüm daha iyi bir çözümse bu çözüm hafızada tutulur ve diğeri unutulur. Eğer yeni çözüm iyi bir çözüm değilse mevcut çözümün geliştirilememe sayacı bir artırılır. Görevli arılar arama işlemini şu şekilde gerçekleştirir:

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

Eşitlik 2'de x_i görevli arının seçmiş olduğu o anki çözümü belirtir. v_i ise x_i nin komşuluğunda bulunan yeni bir çözümdür. k [1, *NS*] aralığında seçilmiş rastgele bir tam sayıdır. x_k yiyecek kaynağı popülasyonu içerisinden seçilmiş rastgele bir çözümdür ve x_i den farklı olmalıdır. j ise [1, *D*] aralığında seçilmiş rastgele bir tam sayıdır ve optimize edilecek problemin rastgele bir parametresini belirtir. $\varphi_{i,j}$ katsayısı [-1, 1] aralığında rastgele seçilmiş bir değerdir.

Görevli arılar araştırmalarını tamamladıktan sonra, gözcü arıların yiyecek kaynaklarını seçmeleri için olasılık değerleri

$$p_i = \frac{fit_i}{\sum_{i=1}^{NS} fit_i} \quad (3)$$

ile hesaplanır. Eşitlik 3'te fit_i normalize edilmiş maliyet fonksiyonu değeridir. Her bir çözüm için olasılık değerleri (p_i) hesaplandıktan sonra görevli arılar rastgele belirlenmiş bir değer ile o çözümün olasılık değerini karşılaştırır. Eğer çözümün seçilme olasılığı bu değerden büyükse, gözcü arı bu yiyecek kaynağına yönelir ve Eşitlik 2'yi kullanarak yeni çözümler arar. Eşitlik 3'te çözümlerin olasılık değerleri kaliteleriyle doğru orantılıdır.

Bütün görevli ve gözcü arılar arama işlemlerini tamamladıktan sonra her bir çözümün geliştirilememe sayacı kontrol edilir. Eğer bir çözümün sayaç değeri *limit* değerine ulaşmış ise o kaynağı kullanan görevli arı kâşif arı olur, Eşitlik 1'i kullanarak çözüm uzayında yeni bir noktaya yönelir ve araştırmasına buradan devam eder. Bütün işlemler maksimum çevrim sayısına veya maksimum maliyet fonksiyonu değerlendirme sayısına ulaşıncaya kadar devam eder. Bulunmuş olan en iyi çözüm kaydedilir.

3. Modifiye ABC Algoritması

ABC algoritmasında gözcü arılar optimum çözümün bulunmasında önemli bir rol oynamaktadır. Gözcü arıların yiyecek kaynakları hakkında bilgi edinmesi için hesaplanan olasılık değerleri çözümün kalitesiyle doğru orantılıdır. Matematiksel modelde eğer iyi bir çözüm bulunmuşsa o çözümün bulunduğu bölgeye daha çok şans verilmektedir ve iyi bir çözümün etrafında daha iyi çözümler bulunabilir mantığı benimsenmiştir. Bunun yanında kâşif arılar çözüm uzayında rastgele bir çözüm bulduğunda, keşfedilen yeni çözüm muhtemelen diğer çözümlere göre daha düşük kalitededir. Olasılık hesabında yüksek kaliteli çözümler ön planda olduğu için düşük kaliteli çözümlerin bulunduğu bölgelerde araştırma yapma şansı düşmektedir. Yeni kaynaklar bulunduktan sonra o kaynak etrafında daha etkili komşuluk araştırması yapmak ve düşük kalitede olan çözümlere de yüksek derecede olasılılık değeri atanması için

$$p_i = (1 + fit_i)\ \exp(-fit_i) \quad (4)$$

kullanılmıştır. Eşitlik 4'teki olasılık değerleri, çözümün kalitesiyle ters orantılıdır ve böylelikle kaliteli çözümlerin yanında düşük kaliteli çözümlere de şans verilmektedir.

ABC algoritmasında optimum çözüme ulaşmada rol oynayan diğer bir etken ise komşuluk araştırması yapmak için kullanılan denklemdir (Eşitlik 2). ABC algoritmasının performansını artırmak için

$$v_{i,j} = x_{k,j} + \varphi_1\left(x_{k,j} - x_{m,j}\right) \quad (5)$$

arama mekanizması önerilmiştir. Eşitlik 5'te; k ve m [1, *NS*] aralığında rastgele seçilmiş tam sayılardır. x_k ve x_m birbirinden ve görevli arının sahip olduğu o anki çözümünden (x_i) farklı çözümlerdir. v_i ise Eşitlik 5 sonucunda bulunmuş ve görevli arının sahip olduğu çözümün komşuluğunda yer alan yeni bir çözümdür. j [1, *D*] aralığında seçilmiş rastgele bir

tam sayıdır ve optimize edilecek problemin rastgele bir parametresini belirtir. Eşitlikte kullanılan φ_1 ise [-1, 1] aralığında rastgele seçilmiş bir katsayıdır.

Görevli arılar arama işlemlerini tamamladıktan sonra gözcü arılar seleksiyon işlemini yaparlar. Bir gözcü arı seçmiş olduğu yiyecek kaynağı etrafında komşuluk araştırması yaparken, görevli arıların kullanmış oldukları Eşitlik 5'i kullanarak sahip olduğu yiyecek kaynağının herhangi bir parametresini değiştirir. Bu işlemden sonra ikinci bir eşitlik kullanarak bu parametreden farklı bir parametre üzerinde değişiklik yapar. Gözcü arının ikinci parametre değişikliği için kullanacağı yeni arama mekanizması şu şekildedir:

$$v_{i,n} = x_{best,n} + \varphi_1(x_{k,n} - x_{m,n}). \quad (6)$$

Eşitlik 6'te x_k ve x_m Eşitlik 5'te kullanılan, gözcü arının sahip olduğu o anki çözümünden farklı rastgele seçilmiş çözümlerdir. v_i ise Eşitlik 6 sonucunda bulunmuş ve gözcü arının sahip olduğu çözümün komşuluğunda yer alan yeni bir çözümdür. n [1, D] aralığında seçilmiş rastgele bir tam sayıdır ve Eşitlik 5'te kullanılan j parametresinden farklı bir parametredir. Eşitlikte kullanılan φ_1 ise Eşitlik 5'te kullanılmış ve rastgele seçilmiş katsayıdır. Bir sonraki bölümde kolaylık sağlaması bakımından önerilen modifiye ABC algoritması ABCinv olarak isimlendirilmiştir.

4. Deneysel Sonuçlar

Bu çalışmada, standart ABC ve ABCinv algoritmalarının performansı literatürde yaygın olarak kullanılan nümerik test fonksiyonları üzerinde gerçekleştirilmiştir. Deney çalışması için kullanılan test fonksiyonları Çizelge 1'de listelenmiştir. Çizelge 1'de görülebileceği üzere $f_1 - f_3$ tek modlu (unimodal, UM) ve $f_4 - f_7$ çok modlu (multimodal, MM) test fonksiyonlarıdır. UM, global optimum dışında yerel optimum değerlere sahip olmayan, MM ise global optimum dışında yerel optimumlara sahip fonksiyonlardır. Deneysel çalışmalarda kullanılan algoritmalar ve fonksiyonlar JAVA programlama dilinde kodlanmıştır. Deneysel çalışmalar Intel Core i3 2.13 GHz işlemci ve 3.00 GB RAM'e sahip olan bir bilgisayarda yapılmıştır.

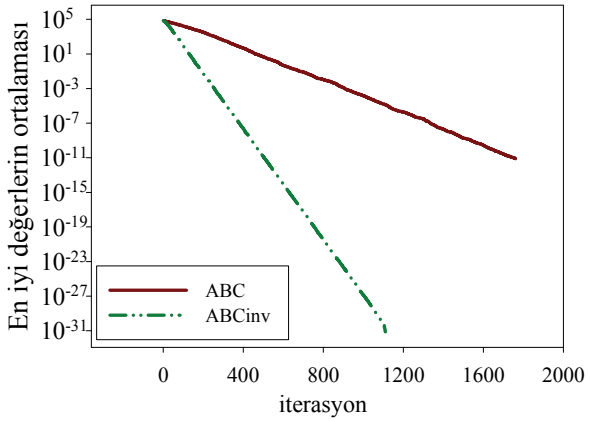
Test fonksiyonları için, parametre sayısı (D = 30) olarak belirlenmiştir. Eşit şartlarda kıyaslama yapmak için, *popsize* = 50 (NS = 25), *limit* = 150 ve maliyet fonksiyonu değerlendirme sayısı da 50.000 olacak şekilde seçilmiştir. Yapılan deneyde algoritmalar 30 farklı seed değeri için koşulmuştur. Deney sonucunda elde edilen en iyi, en kötü, ortalama ve standart sapma değerleri ile algoritmaların milisaniye (ms) cinsinden ortalama koşma süreleri Çizelge 2'de listelenmiştir. Daha iyi olan sonuçlar kalın yazılmıştır. Şekil 1–9'da ise algoritmaların yakınsama grafikleri verilmiştir.

Çizelge 1: Nümerik Test Fonksiyonları

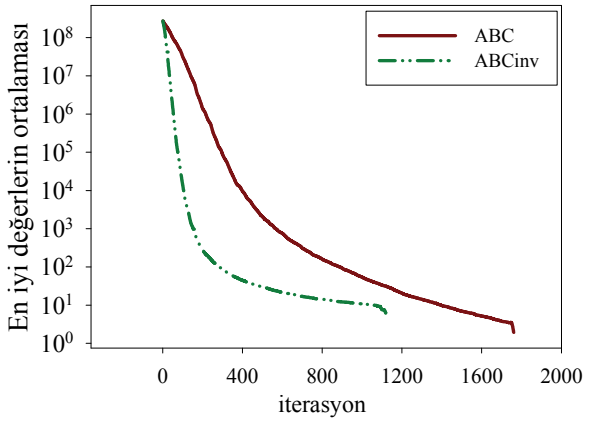
Fonk.No	Fonksiyon	Aralık	Tip	Formül
f_1	Sphere	$[-100, 100]^D$	UM	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$
f_2	Rosenbrock	$[-30, 30]^D$	UM	$f_2(\vec{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} + x_i^2)^2 + (x_i - 1)^2]$
f_3	Quartic	$[-1.28, 1.28]^D$	UM	$f_3(\vec{x}) = \sum_{i=1}^{D} \mathrm{i}\, x_i^4 + rand[0,1)$
f_4	Schwefel's 2.26	$[-500, 500]^D$	MM	$f_4(\vec{x}) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$
f_5	Rastrigin	$[-5.12, 5.12]^D$	MM	$f_5(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$
f_6	Ackley	$[-32, 32]^D$	MM	$f_6(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$
f_7	Griewank	$[-600, 600]^D$	MM	$f_7(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

Çizelge 2: ABC ve ABCinv algoritmalarının karşılaştırılması

Fonk.No	Algoritma	Ortalama	En İyi	En Kötü	Std. Sap.	Süre (ms)
f_1	ABC	8,04E-12	7,69E-13	4,28E-11	3,31E-11	1132,27
	ABCinv	**4,97E-31**	**3,66E-32**	**1,83E-30**	**1,81E-30**	**817,73**
f_2	ABC	**3,27E+00**	5,54E-01	**8,41E+00**	**7,96E+00**	1298,47
	ABCinv	9,59E+00	**1,65E-01**	7,62E+01	7,13E+01	**927,60**
f_3	ABC	1,91E-01	2,75E-01	1,01E-01	1,59E-01	1737,00
	ABCinv	**3,19E-02**	**6,48E-02**	**1,51E-02**	**4,54E-02**	**1385,43**
f_4	ABC	-12216,2	-12566,0	-11863,2	5,23E+02	1280,20
	ABCinv	**-12510,2**	**-12569,5**	**-12214,2**	**3,50E+02**	**940,07**
f_5	ABC	9,91E-01	3,64E-03	2,02E+00	2,18E+00	1283,33
	ABCinv	**2,15E-01**	**0,00E+00**	**9,95E-01**	**1,54E+00**	**890,10**
f_6	ABC	2,59E-05	4,53E-06	9,47E-05	6,12E-05	1292,20
	ABCinv	**2,83E-14**	**2,18E-14**	**3,24E-14**	**1,23E-14**	**909,37**
f_7	ABC	8,46E-04	2,39E-10	1,77E-02	1,32E-02	1301,57
	ABCinv	**5,78E-09**	**0,00E+00**	**1,73E-07**	**1,20E-07**	**942,70**



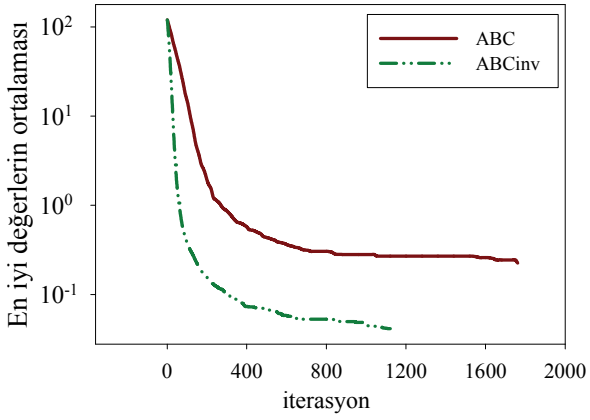
Şekil 1: f_1 yakınsama grafiği.



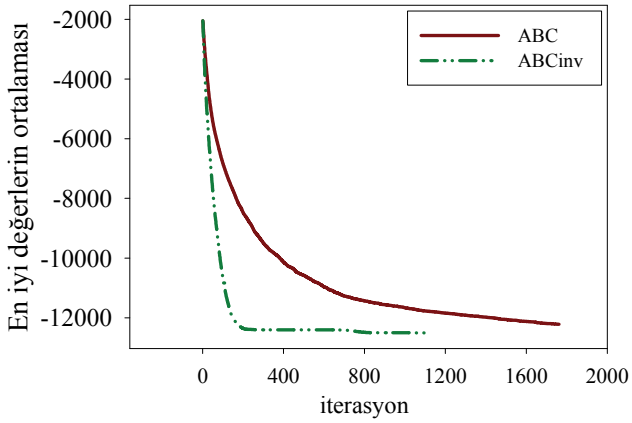
Şekil 2: f_2 yakınsama grafiği.

Çizelge 2 ve Şekil 1–9'de yer alan sonuçlara göre Standart ABC ve ABCinv algoritmalarının performansları değerlendirildiğinde; ABCinv algoritması, f_2 fonksiyonu dışında bütün fonksiyonlar için standart ABC algoritmasından daha iyi sonuçlar üretmiştir. MM fonksiyonlarda ABCinv algoritmasının bariz bir üstünlüğü olmasının yanında UM fonksiyonlarda da (f_2 fonksiyonu hariç) başarılı sonuçlar elde edilmiştir. Algoritmaların sonuç üretme hızlarına baktığımızda ABCinv algoritmasının standart ABC algoritmasına göre daha hızlı sonuç ürettiği de görülmektedir. Ayrıca Şekil 1–9 incelendiğinde ABCinv algoritmasının daha hızlı yakınsama yaparak sonuca ulaştığı çok açıktır.

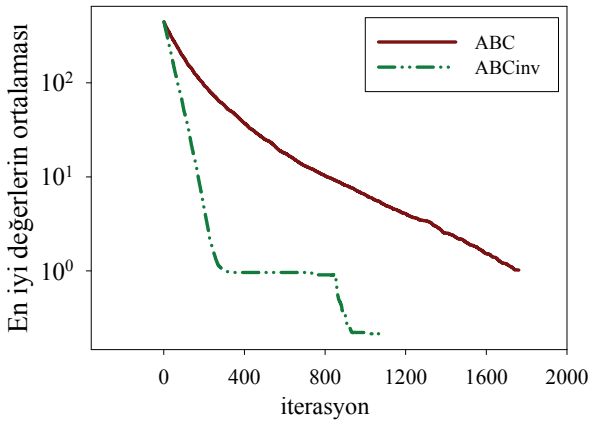
Standart ABC algoritması için kullanılan arama mekanizması ile olasılık hesaplama yönteminin ne kadar önemli olduğu Çizelge 2 ve Şekil 1–9'deki sonuçlar incelendiğinde görülebilir. Yeni arama mekanizmaları ve olasılık hesaplama yöntemi ile ABC algoritmanın daha iyi sonuç üretmesini sağlamış, sonuç üretme hızına da önemli bir etkisi olmuştur.



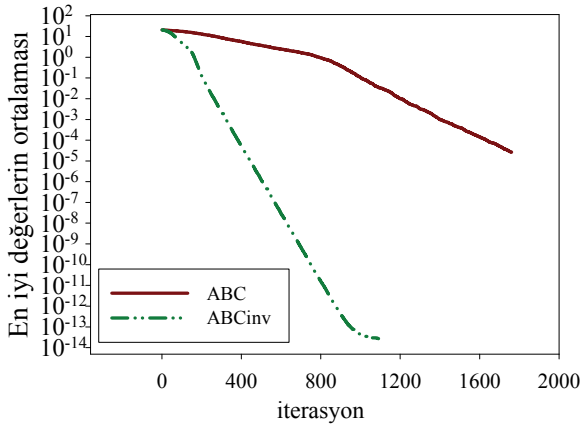
Şekil 3: f_3 yakınsama grafiği.



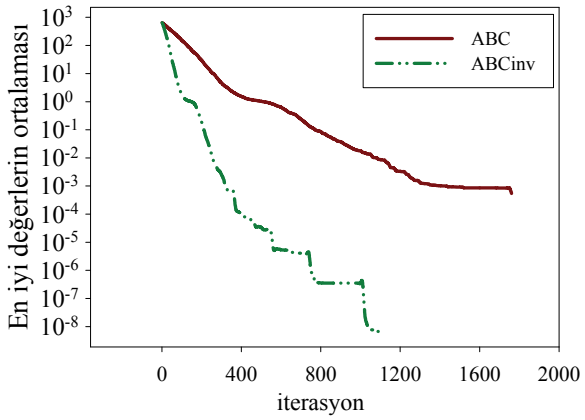
Şekil 4: f_4 yakınsama grafiği.



Şekil 5: f_5 yakınsama grafiği



Şekil 6: f_6 yakınsama grafiği.



Şekil 7: f_7 yakınsama grafiği.

ABCinv algoritmasında kullanılan ve çözümün kalitesiyle ters orantılı olan olasılık hesaplama yöntemi, düşük kaliteli çözümlere de yüksek kaliteli çözümler kadar şans vermektedir. Bununla birlikte önerilen modifiye ABC algoritmasında gözcü arılar standart ABC algoritmasına göre daha verimli kullanılmaktadır. Bir çevrim içerisinde daha fazla maliyet fonksiyonu değerlenndirme sayısının artması algoritmanın performansına katkıda bulunmuştur.

5. Sonuçlar

Bu çalışmada, yeni bir modifiye ABC algoritması sunulmuştur. Sunulan ABCinv algoritmasının performansı test fonksiyonları kullanılarak standart ABC algoritmasıyla karşılaştırılmıştır. ABCinv algoritması standart ABC algoritmasına göre çok daha iyi sonuçlar üretmiştir. Kullanılan yeni seçme ve arama mekanizmaları ABC algoritmasının performansını artırmıştır.

Gelecek çalışma olarak ABCinv algoritması dairesel anten dizisi tasarımı ve spektrum algılama gibi çeşitli gerçek hayat problemlerine uygulanacaktır.

6. Kaynaklar

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

[2] Karaboga, D. ve Basturk, B., "On the performance of artificial bee colony (ABC) algorithm", *Appl. Soft Comput.*, 8, 687–697, 2008.

[3] Karaboga, D. ve Akay, B., "A comperative study of artificial bee colony algorithm", *Appl. Math. Comput.*, 214, 108–132, 2009.

[4] Karaboga, N., "A new design method based on artificial bee colony algorithm for digital IIR filters", *J. Franklin I.*, 346 (4), 328–348, 2009.

[5] Ma, M., Liang, J., Guo, M., Fan, Y. ve Yin, Y. "SAR image segmentation based on artificial bee colony algorithm", *Appl. Soft Comput.*, 11, 5205–5214, 2011.

[6] Kashan, M.H., Navahandi, N. ve Kashan, A.H.. "Design and economic optimization of shell and tube heat exchangers using artificial bee colony (abc) algorithm", *Appl. Soft Comput.*, 12, 352–352, 2012.

[7] Zhu, G. ve Kwong, S. "Gbest-guided artificial bee colony algorithm for numerical function optimization", *Appl. Math. Comput.*, 217, 3166–3173, 2010.

[8] Li, G., Niu, P. ve Xiao, X., "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization", *Appl. Soft Comput.*, 12, 320–332, 2012.