Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı, S. 407-415, Ağustos 2020
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue, pp. 407-415, August 2020
Copyright © 2020 EJOSAT
Research Article

Dinamik Parametre Değerli Yapay Arı Koloni Algoritması (DPD-YAK)

Dursun Ekmekci[1*]

[1] Karabük Üniversitesi, TOBB Teknik Bilimler MYO, Bilgisayar Teknolojileri Bölümü, Karabük, Türkiye (ORCID: 0000-0002-9830-7793)

(Bu yayın 26-27 Haziran 2020 tarihinde HORA-2020 kongresinde sözlü olarak sunulmuştur.)

(**DOI:** 10.31590/ejosat.780659)

ATIF/REFERENCE: Ekmekci, D. (2020). Dinamik Parametre Değerli Yapay Arı Koloni Algoritması (DPD-YAK). *Avrupa Bilim ve Teknoloji Dergisi*, (Special Issue), 407-415.

Öz

Kriterleri sağlayan çok sayıda alternatif içinden, en yararlı olanı seçebilmek, hayatı meşgul eden problemlerden biridir. Çoğu birbiriyle çelişkili kriterler için en doğru tercih, çok fazla zaman alır. Bu bağlamda "optimizasyon" (en iyileme) kavramı, bireysel yaşamda farklı örnekleriyle karşılaştığımız ve farklı sektörlerde birçok firmanın, üzerinde titizlikle yoğunlaştığı konulardan biridir. Optimizasyon problemleri için genellikle, makul sürelerde geçerli çözümler sunabilen metasezgisel yöntemler tercih edilmektedir. Ancak optimizasyon problemlerine başarıyla uygulanabilen bu algoritmalar için en büyük problemlerden biri, algoritma parametrelerine uygun değerlerinin atanabilmesidir. Algoritmanın, arama alanına yeterince dağılabilmesi ve bulduğu çözümlerden daha iyi çözümler türetebilmesi için kontrol parametrelerine uygun değerler atanmalıdır. Dolayısıyla algoritma performansı, parametre değerleriyle doğrudan ilişkilidir. Araştırmacılar son dönemde, optimizasyon algoritmaları için parametre değerlerini en uygun değere ayarlayan, çevrimiçi ve çevrimdışı uygulanan birçok yöntem geliştirdiler. Yapay Arı Koloni (YAK) Algoritması da oluşturulduğu günden bugüne, yöneylem araştırmacılarının ilgisini çeken, geliştirilen farklı birçok versiyonu ile literatürde yer edinmiş, sürü zekâsı temelli bir metasezgisel yöntemdir. Algoritma, çözüm oluşturma ve yeni çözümler türetmede farklı prosedürler kullansa da tüm bunları iki kontrol parametresinde birleştirmektedir. Bu çalışmada, YAK algoritmasının keşif ve sömürü performansını geliştirmek için, parametre değerlerini, çözüm arama sürecinde değiştiren, Dinamik Parametre Değerli Yapay Arı Koloni (DPD-YAK) Algoritması önerilmektedir. Önerilen yöntem, sekiz farklı bilindik sayısal optimizasyon fonksiyonları üzerinde test edilerek, çözüm arama başarısı araştırılmıştır. Birbirinden bağımsız olarak 30'ar denemede elde edilen sonuçların aritmetik ortalaması ve standart sapma değeri hesaplanmıştır. Bu sonuçlar, literatürdeki farklı bir çalışmada, standart YAK ve diğer popüler metasezgisel yöntemlerle elde edilmiş sonuçlarla karşılaştırılmıştır. DPD-YAK, fonksiyonların birçoğu için, en iyi sonucu üretmiş ve YAK algoritması performansını önemli seviyede artırmıştır. Sonuçlar, DPD-YAK algoritmasının optimizasyon problemleri için başarıyla uygulanabileceğini ispatlamaktadır.

Anahtar Kelimeler: Optimizasyon, YAK, DPD-YAK, Parametre ayarlaması.

Artificial Bee Colony Algorithm with Dynamic Parameter Values (ABC-DPV)

Abstract

Choosing the most useful among the many alternatives that provide the criteria is one of the problems that occupy life. For many conflicting criteria, the right choice takes a lot of time. In this context, the concept of "optimization" is one of the subjects that we encounter with different examples in individual life, and many companies in different sectors are focused on meticulously. For optimization problems, generally, meta-heuristic methods, which can provide solutions that are valid at reasonable times, are preferred. However, one of the biggest problems for these algorithms, which can be applied successfully to optimization problems, is to assign appropriate values to algorithm parameters. In order for the algorithm to explore efficiently in the search area and to derive better solutions from the existing solutions it finds, appropriate values should be assigned to the control parameters. Therefore, algorithm performance is directly related to parameter values. Researchers have recently developed several methods that tune optimal parameter values for optimization algorithms, applied online, or offline. Artificial Bee Colony (ABC) Algorithm is also a swarm-intelligence based metaheuristic method with many different versions that have attracted the attention of operations researchers since the day it was created. Although different procedures are used in the algorithm, solution evaluation, and deriving new solutions, it combines all of these in two control parameters. In this study, Artificial Bee Colony with Dynamic Parameter Value (ABC-DPV)

[*] Sorumlu Yazar: Karabük Üniversitesi, TOBB Teknik Bilimler MYO, Bilgisayar Teknolojileri Bölümü, Karabük, Türkiye, ORCID: 0000-0002-9830-7793, dekmekci@karabuk.edu.tr

Algorithm, which changes parameter values in the searching process, is proposed to improve the exploration and exploitation performance of the ABC algorithm. The proposed method was tested on eight different numerical optimization functions to examine its searching strategy. The arithmetic mean and standard deviation value of the results, which were obtained in 30 trials independently, were calculated. These results were compared with results obtained in a different study in the literature with standard ABC and other popular metaheuristic methods. ABC-DPV has produced the best result for many of the functions and significantly improved the ABC algorithm performance. The results prove that the ABC-DPV algorithm can be successfully applied for optimization problems.

Keywords: Optimization, ABC, ABC-DPV, Parameter tuning.

1. Giriş

Günlük yaşamda farklı örnekleriyle karşılaştığımız optimizasyon problemleri için, farklı tür ve yapılarda çözüm algoritmaları geliştirilmiştir. Optimizasyon algoritmaları özelliklerine göre, popülasyona dayalı, deterministik, stokastik, yinelemeli ya da doğadan esinlenen algoritmalar olarak gruplandırılabilir (Akay & Karaboga, 2009). Bu bağlamda algoritma performansını etkileyen temel konulardan biri, algorimanın, simule ettiği yapıdır. Dolayısıyla optimizasyon problemi için, çözüm algoritması tercihinde, problem karakteristiğine göre, taklit edilen model göz önünde tutulmalıdır. Optimizasyon algoritmaları tasarlanırken, taklit edilen modelin davranışına etki eden bileşenler, algoritma operatörleri olarak belirlenir. Bu kapsamda çözüm performansını etkileyen diğer bir konu, algoritma operatörleri için belirlenen parametre değerleridir. Operatörlerdeki her bir prosedür, popülasyon üyelerinin bireysel faaliyetlerine kararlar vererek algoritmanın keşif yeteneğini ya da popülasyonun kolektif faaliyetinde modifikasyonlar yaparak algoritmanın sömürü yenteğini etkiler. Başarılı bir arama için, algoritmanın, bu prosedürleri yerine getiren kontrol parametreleri için uygun değerler atanmalıdır.

Yapay arı koloni (YAK) algoritması, Karaboğa (Karaboga, 2005) tarafından, bal arısı kolonisinin yiyecek arama davranışı modellenerek literatüre kazandırılan, sürü-zekâsı temelli bir metasezgisel yöntemdir. Algoritma kapsamında, yiyecek arama ve toplama sürecine katılan üç farklı bal arısının davranşı taklit edilmektedir. Yiyecek arama faaliyeti, *kâşif* arıların, kovan merkezli arama bölgesinde, rastgele besin kaynağı arayışıyla başlar. Besin kaynakları belirlendikten sonra, yiyecek toplama faaliyeti *işçi* arılarla devam eder. Tüm işçi arılar, görevli oldukları kaynağı, komşuluğunu da kontrol ederek ziyaret ederler. Daha uygun bir kaynak bulduklarında ise, eski kaynağı terkedip, sonraki aramalarda yeni kaynağı referans alırlar. İşçi arıların diğer bir faaliyeti ise, görevli oldukları besin kaynağına ilişkin bilgiyi, kendilerinden sonra toplamaya katılan *gözcü* arılarla paylaşmaktır. İşçilerin tecrübelerinden yararlanan gözcüler, daha elverişli besin kaynağına yönelme eğilimindedirler. Gözcü arılar da işçi arılar gibi, uğradığı kaynağın civarında daha elverişli bir kaynak bulduklarında, aç gözlü yaklaşımla, eski kaynağı terk ederler. Süreç içinde besini tükenen kaynaklar, terk edilir ve kaşif arılar yeniden rastgele kaynak arayışını başlatırlar (Akay & Karaboga, 2012). Biyolojik yapısı özetlenen bu süreci, daha etkin kılabilmek için, araştırmalar halen devam etmekte ve farklı teknikler geliştirilmektedir (Bansal, Sharma, & Jadon, 2013) (Karaboga, Gorkemli, & Ozturk, 2014), (Akay & Karaboga, 2015), (Sharma & Bhambu, 2016).

YAK algoritmasında besin kaynağı ile muhtemel her bir çözüm ifade edilmektedir. Bu bağlamda kâşif arılar stokastik arama prosedürleri, işçi ve gözcü arılar ise sömürüyü güçlendiren ajanlardır. Algoritmanın orijinalinde, yiyecek toplanan kaynak sayısı (başlangıçtaki çözüm sayısı) ile işçi ve gözcü arı sayıları birbirine eşittir. Bu bağlamda kolonide, besin kaynağının iki katı kadar bal arısı bulunur. Algoritmanın diğer bir parametresi ise *limit* değerdir. Bir besin kaynağı komşuluğunda, limit kez ziyaret sonucu daha uygun bir kaynak tespit edilememişse, algoritma yeni rastsal çözümler üretecektir (Bacanin & Tuba, 2012). Algoritmayı popüler kılan özelliklerden birisi, az sayıdaki kontrol parametresiyle arama yapması ve kolay uygulanabilir olmasıdır (Karaboga & Akay, 2009). Optimizasyon problem türüne göre, bu parametreler için farklı değerler tercih edilmiştir. YAK parametrelerinin, algoritma performansına etkisinin analiz edildiği bir çalışmada (Akay & Karaboga, 2009), algoritma, sayısal optimizasyon problemleri üzerinde farklı parametre değerleriyle test edilmiştir. Deneylerde, çözümdeki bileşen sayısı (*ÇBS*) ve arama bölgesi genişliği gibi probleme ait parametreler ile koloni boyutu ve limit gibi algoritmaya ait parametrelerde farklı değerler seçilmiştir. Farklı bir çalışmada, (Bensebti & Bouchibane, 2018) baz istasyonunda enerji verimliliğini en üst düzeye çıkarmak için, YAK algoritması kullanılarak, anten boyutu ve aktif kullanıcı sayısı için uygun bir kombinasyon hedeflenmiştir. Çözüm sürecinde, YAK kontrol parametreleri için optimal değerler araştırılmıştır. Diğer bir çalışmada, QoS servis hizmetinde optimal kombizasyon için uygun YAK parametre değerleri aranmıştır (Liu, Wang, & Xu, 2014). Probleminin beş özelliği, YAK parametreleri ve son çözümün iki ölçümü tanımlanmış, YAK parametre ayarı C4.5 algoritması kullanılarak gerçekleştirilmiştir. Bu kapsamda problem özellikleri ile YAK parametreleri arasındaki bağımlılık çoklu doğrusal regresyon yöntemleri kullanılarak belirlenmiştir. Görüntü işleme alanındaki bir çalışmada (Kockanat & Karaboga, 2013), YAK parametre değerleri, gri seviye dijital görüntülerde gürültü giderme problemindeki performansı için incelenmiştir. Farklı bir çalışmada, algoritmanın limit parametresi için, en verimli prametre değeri araştırılmış (Veček, Liu, Črepinšek, & Mernik, 2017).

Araştırmacılar ayrıca, YAK algoritmasının yakınsama performansını iyileştirmek için parametre değerlerini otomatik olarak düzenleyen ya da arama sürecinde dinamik olarak değiştiren teknikler geliştirmişlerdir. Bu teknikelerin birinde (Elkhateeb & Badr, 2017), her bir yeniden başlatma işleminde çözüm sayısını kademeli olarak azaltılan ve bal arılarını genel en iyi çözüme doğru yönlendiren yöntem önerilmiştir. Diğer bir çalışmada (Sakib & Mahzabeen, 2018), sömürü ve keşif oranı, işçi ve gözcü arıların bir statik - beş dinamik oranı korunarak değiştirilmiş ve hangi kombinasyonun belirgin performans gösterdiği analiz edilmiştir. Sürü-zekâsı tabanlı metasezgisel yöntemlerin birçoğunda olduğu gibi YAK algoritması tasarımında da yaygın olarak kullanılan yöntemlerden biri F-Yarışı ve yinelemeli F- Yarışı yöntemleridir (Bartz-Beielstein, Chiarandini, Paquete, & Preuss, 2010). Makine öğrenimindeki yarış algoritmalarından esinlenerek geliştirilen bu yöntemde temel fikir, belirli bir örnek konfigürasyon kümesini

yinelemeli olarak bir örnek akışı üzerinde değerlendirmektir. Yöntemde, öncelikle rastgele parametre değerlerinden oluşan aday konfigürasyonlar oluşturulur. Ardından her değerlendirme turundan sonra, içlerinden en az birinin, diğerlerinden önemli ölçüde farklı olduğunu kanıtlamak için, konfigürasyonlara parametrik olmayan Friedman testi uygulanır. Bazı aday konfigürasyonlara karşı yeterli istatistiksel kanıt toplandığında, kötü konfigürasyonlar ortadan kaldırılır ve yarış sadece hayatta kalanlarla devam eder. Ajan tabanlı birçok optimizasyon algoritması için, parametre değerlerini en uygun şekilde atamayı hedefleyen bir çalışmada (Korkmaz Tan & Bora, 2019), uyarlanabilir parametre ayarlama yöntemi, farklı yöntemlerle karşılaştırılıp kontrol edilerek hibridleştirilen yeni bir yaklaşım önerilmiştir.

YAK algoritması, belirlenen koloni boyutu için popülasyonda yeterli çeşitliliği sağlayarak keşif sürecini verimli olarak kullanabildiğinden, algoritma, büyük boyutlardaki optimizasyon problemlerini çözmek için büyük koloni boyutuna ihtiyaç duymaz (Akay & Karaboga, 2009). Birçok çalışmada, YAK algoritması performansı farklı koloni seviyeleri için test edilmiş ve işlem yükünü ağırlaştırmaya gerek kalmadan, algoritmanın, az sayıdaki koloni üyeleriyle bile başarılı çözümler üretebildiği kanıtlanmıştır. Bu çalışmada, önceki literatür çalışmalarından edinilen tecrübeler ışığında, YAK algoritmasının işçi arı sayısı, başlangıçta belirlenen çözüm sayısı ile sabitlenmiş, algoritmanın keşif ve sömürü yeteneğini daha da güçlendirmek amacıyla, her bir çevrimde gözcü arı sayısı için elitist yaklaşım uygulanmış ve çözümler için belirlenen limit değer, her bir çözümün başarı seviyesiyle oranlanmıştır. Önerilen metot, sayısal optimizasyon problemleri üzerinde test edilmiş, elde edilen sonuçlar algoritmanın farklı versiyonları ve diğer metasezgisel yöntemlerle karşılaştırılmıştır. Makalenin kalan bölümleri şu şekilde tasarlanmıştır: bu bölümde biyolojik özellikleri ve yiyecek arama senaryosu anlatılan YAK algoritması, Bölüm 2'de, kullandığı matematiksel fonksiyonlarla açıklanmış, Bölüm 3'te önerilen çözüm yöntemi detaylı olarak izah edilmiş ve geliştirilen uygulama, Bölüm 4'te sayısal optimizasyon problemleri üzerinde test edilerek sonuçlar karşılaştırmalı olarak yorumlanmıştır. Son olarak Bölüm 5'te tüm çalışma, genel hatlarıyla değerlendirilmiştir.

2. Standart Yapay Arı Koloni Algoritması

Önceki bölümde, optimizasyon problemleri için çözüm arama yaklaşımı, esinlendiği arı kolonisi faaliyetleriyle açıklanan YAK algoritması, bu bölümde, operatörlerinde kullandığı matematiksel fonksiyonlar ve algoritmik prosedürlerle açıklanmaktadır. Geleneksel YAK algoritmasının genel adımları, Şekil 1'deki gibi oluşturulmuştur.

Algoritma 1- YAK Algoritması adımları
Başlangıç çözümlerinin oluşturulması
Çözüm değerlerinin belirlenmesi
Kriterler sağlanıncaya kadar tekrarla
İşçi arı aşaması
Gözcü arı aşaması
Kâşif arı aşaması
Bulunan en iyi çözümün kaydedilmesi
Çevimi sonlandır
Çözüm: Kaydedilen çözüm

Şekil 1. YAK algoritması

Algoritma, kâşif arıların rastgele çözümler oluşturmasıyla başlamaktadır.

2.1. Kâşif Arı Safhası

Kâşif arı safhasında, (1) eşitliği kullanılarak, belirlenen çözüm sayısı kadar (*ÇS*) rastgele çözümler oluşturulur.

$$Ç_{m,i} = a_i + rastgele(0,1) * (a_i - ü_i) \quad (1)$$

(1)'de $Ç_m$, çözümler kümesindeki (*Ç*) *m*. çözüm, $Ç_{m,i}$ çözümün *i.* bileşenidir. a_i *i.* bileşen için seçilebilecek en küçük değer, u_i ise ilgili bileşen için atanabilecek en üst değerdir.

Başlangıç çözümleri oluşturulduktan sonra, her bir çözüm için, problemin amaç fonksiyonuna göre *f(Çm)* değeri ve bu değer referans alınarak, (2) ile çözümün uygunluk değeri *uyg(Çm)* hesaplanır.

$$uyg(Ç_m) = \begin{cases} 1/(1+f(Ç_m)) & eğer\ (f(Ç_m)) \geq 0 \\ 1+mutlak(f(Ç_m)) & eğer\ (f(Ç_m)) < 0 \end{cases} \quad (2)$$

Kâşif arı safhasında, başlangıç çözümleri oluşturulduktan sonra, herbir çözüm için başarısızlık sayacı (*bs*), 0 olarak atanır.

2.2. İşçi Arı Safhası

Kâşif arıların oluşturduğu çözümlerin herbiri için bir işçi arı görevlendirilir. Her bir işçi arı, (3) eşitliğini kullanarak görevli olduğu çözüm ile yeni bir çözüm (Y_m) türetir.

$$Y_{m,i} = Ç_{m,i} + \phi_{m,i}(Ç_{m,i} - Ç_{r,i}) \quad (3)$$

(3) eşitliğinde, $Ç_m$ sıradaki çözüm, $Ç_r$ yeni çözüm türetmede rastgele seçilen yardımcı çözüm, $\phi_{m,i}$ ise [-1, 1] aralığında rastgele seçilen bir değerdir.

Yeni çözüm türetildikten sonra, bu çözüm için de amaç fonksiyonuna göre $f(Y_m)$ ve (2) eşitliğiyle, uygunluk değeri $uyg(Y_m)$ hesaplanır. Yeni çözümün uygunluk değeri ile mevcut çözümün uygunluk değeri karşılaştırılır. Eğer yeni çözümün uygunluk değeri daha büyükse ($Y_m > Ç_m$), diğer bir ifadeyle türetilen çözüm, mevcut çözümden daha başarılıysa, çözüm matrisinde ($Ç$) mevcut çözüm silinerek yerine yeni çözüm kaydedilir ve yeni çözümün başarısızlık sayacı (bs_m) sıfırlanır. Aksi durumda ise yalnızca mevcut çözümün başarısızlık sayacı 1 artırılacaktır ($bs_m = bs_m + 1$).

2.3. Gözcü Arı Safhası

Herbir işçi arı ile yeni çözümler türetildikten sonra, sıradaki işlem, işçi arıların, tecrübesini gözcü arılarak aktarabilmesi işlemidir. Bu sayede gözcü arıların, daha başarılı çözümleri kullanarak yeni çözümler türetmesi hedeflenir. Standart YAK algoritmasında bu etkileşim için rulet tekerleği kullanılır. (4) ile, herbir çözümün seçilme olasılığı (S) hesaplanır ve gözcü arılar, (3) ile yeni çözüm türetirken, seçtikleri ($Ç_m$) ve yardımcı ($Ç_r$) çözümleri bu olasılığı dikkate alarak belirlerler. ((4) eşitliğinde m çözümünün seçilme olasılığı (s_m) hesaplanmaktadır.)

$$s_m = \frac{uyg(Ç_m)}{\sum_{n=1}^{ÇS} uyg(Ç_n)} \tag{4}$$

Gözcü arılar da tıpkı işçiler gibi, yeni bir besin çözüm türettiklerinde, bu çözümü mevcut çözümle karşılaştırır ve sonraki çevrimlerde daha iyi çözümü kullanmak üzere aç gözlü yaklaşımla tercih yaparlar.

Algoritma, iteratif olarak işçi ve gözcülerin ardışık çözüm üretme işlemleriyle devam eder. Herhangi bir çözümün başarısızlık sayacı *limit* seviyesine ulaştığında ise, bu çözüm yerine kâşif arı işleviyle (1) kullanılarak yeniden rastgele çözüm oluşturur. Limit, YAK algoritmasının yerel en iyi çözümden kurtulması için kullanılan parametredir.

3. Önerilen Metot

Çalışma kapsamında, YAK algoritmasının sömürü yeteneğini daha verimli hale getirebilmek, diğer taraftan keşif performansını da artırabilmek için, algoritmanın parametre değerlerini, arama sürecinin herbir çevriminde güncelleyen Dinamik Parametre Değerli Yapay Arı Koloni (DPD-YAK) algoritması önerilmektedir. Standart YAK modelinde yalnızca "ÇS" ve "limit" parametreleri kullanılmaktadır. Başlangıçta oluşturulan rastgele çözümler gözardı edilirse, ÇS, daha başarılı çözümler türetmeye çalışan işçi ve gözcü arı sayısını belirler. Algoritma kabulünde işçi ve gözcü arı sayısı ÇS'ye eşittir. Ancak DPD-YAK, gözcü arı safhasında daha az gözcü arı prosedürü kullanır. YAK algoritmasının limit parametresi ise, algoritmanın sömürü/keşif dengesini belirleyen parametredir ve tüm çözümler için eşit değerdedir. Detaylı analizler, limit parametresi için en verimli değerin, (5) ile hesaplanabileceğini göstermektedir (Karaboga & Akay, 2009).

$$limit = ÇS * ÇBS \tag{5}$$

DPD-YAK'ta ise limit, herbir çözüm için başarı seviyesine göre ayrı ayrı atanmaktadır.

3.1. DPD-YAK'ta Gözcü Arı Sayısı

DPD-YAK yönteminde gözcü arı sayısı için elitist bir strateji uygulanmaktadır. Herbir çevrimde işçi arı safhasındaki çözüm türetme işlemleri tamamlandıktan sonra, (2) denklemiyle, güncel çözümlerin herbiri için uygunluk değeri ve (6) ile çözüm kümesinin ortalama uygunluk değeri hesaplanır.

$$uyg(ort) = \frac{\sum_{n=1}^{ÇS} uyg(Ç_n)}{ÇS} \tag{6}$$

Gözcü arı safhasında, hesaplanan ortalama uygunluk değerinden daha yüksek uygunluk değerine sahip çözüm sayısı kadar gözcü arı kullanılır ve gözcü arılar, bu çözümlere yönlendirilir. Böylece, daha az işlem yüküyle, algoritmayı başarılı kılan çözümlerin istisnasız tamamı ziyaret edilmiş olur. Bu yaklaşımın amacı, algoritmaya daha aç gözlü arama stratejisi kazandırabilmektir.

3.2. DPD-YAK'ta Limit Seviyesi

YAK algoritmasının çözüm arama yaklaşımına göre, çözüm komşuluklarının detaylı olarak taranabilmesi için limit değer artırılmalı, yerel en iyi çözümden kaçabilmek için ise azaltılmalıdır. Kâşif arı safhasında kötü çözümler oluşturulsa bile, işçi ve gözcü arı safhalarında bu çözümler komşuluğunda daha iyi çözümlerin bulunması umulur ve yeni çözümlerin başarısızlık sayacı sıfırlanır. Bu sayede, daha iyi çözüm komşuluklarında aramalar yapılır. Aksi takdirde algoritma, kötü çözümlerin etrafındaki daha kötü çözümlerle oyalanacaktır. Dolayısıyla limit hesabını etkileyen ana unsurun, iyi çözümlerin komşuluğunun yeterince araştırılabilmesi olduğu düşünülebilir.

Bu düşünce doğrultusunda, DPD-YAK metodunda çözümlerin limit değerleri, başarı seviyelerine göre belirlenmektedir. Standart YAK algoritmasında, tüm çözümlere eşit seviyede limit değeri atanmaktadır. Dolayısyla YAK algoritmasının kullandığı toplam limit sayısı ($limit_{top}$), (5) denklemiyle hesaplanan değerin, $ÇS$ ile çarpılan sonucudur. DPD-YAK metodunda bu sayı, çözümlerin uygunluk

değerine göre, genel en iyi çözümün uygunluk değeriyle orantlılı olarak paylaştırılır. Herbir çevrimde, gözcü arı safhasından sonra, (7) kullanılarak güncel çözümler ve genel en iyi çözümden oluşan popülasyonun toplam uygunluk değeri (uyg_{top}) hesaplanır.

$$uyg_{top} = \sum_{n=1}^{ÇS} uyg(Ç_n) + uyg(Ç_{Enİyi}) \quad (7)$$

Daha sonra $limit_{top}$, çözüm kümesindeki güncel çözümlere paylaştırılır. Bu bağlamda m. çözüme atanacak limit değer (8) ile hesaplanır.

$$limit_m = limit_{top} * uyg(Ç_m)/uyg_{top} \quad (8)$$

4. Deneysel Çalışmalar

Önerilen metot, .net ortamında, C# programlama dili kullanılarak kodlanmış ve uygulama, i7-5600U CPU 2.60 GHz x64 tabanlı işlemcili, 8 GB RAM hafızaya sahip, Windows 8.1 64 bit işletim sistemi kurulu makinede çalıştırılmıştır. Uygulama, standart YAK algoritmasının farklı parametre değerleriyle annaliz edildiği test fonksiyonları üzerinde ve elde edilen sonuçların, diferansiyel evrim (DE) ve parçacık sürüsü optimizasyonu (PSO) algoritmalarının sonuçlarıyla da karşılaştırıldığı çalışmadaki (Akay & Karaboga, 2009) koşullarda çalıştırılmıştır. Buna göre, seçilen test fonksiyonları Tablo 1'de sunulmaktadır.

Tablo 1. Seçilen Test Fonksiyonları

Fonksiyon	**ÇBS**	**Aralık**	**Minimum**	**Formulasyon**
Ackley	30	[-32, 32]	F_{min}=0	$f(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i))$
Dixon-Price	30	[-10, 10]	F_{min}=0	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2$
Griewank	30	[-600, 600]	F_{min}=0	$f(x) = \frac{1}{4000}(\sum_{i=1}^{D} x_i^2) - (\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)) + 1$
Rastrigin	30	[-5.12, 5.12]	F_{min}=0	$f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
Rosenbrock	30	[-100, 100]	F_{min}=0	$f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
Schwefel	30	[-500, 500]	F_{min}=-12,569.5	$f(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{x_i})$
Step	2	[-100, 100]	F_{min}=0	$f(x) = \sum_{i=1}^{n}(\lvert x_i + 0.5\rvert)^2$
Sphere	30	[-100, 100]	F_{min}=0	$f(x) = \sum_{i=1}^{D} x_i^2$

DPD-YAK algoritması, seçilen test fonksiyonlarının hepsi için birbirinden bağımsız olarak 30 kez çalıştırılmış, herbir denemede maksimum çevrim sayısı 10000 olarak belirlenmiştir. Koloni boyutu 100 değerinde sabitlenerek, fonksiyonların farklı ÇBS seviyeleri için algoritmaların elde ettiği sonuçlar Tablo 2'de ve tüm fonksiyonların ÇBS parametreleri 100 seviyesinde sabitlenerek, algoritmaların farklı koloni sayılarıyla elde ettiği sonuçlar Tablo 3'te gösterilmektedir.

Tablo 2. Koloni boyutu=100 ve farklı ÇBS değerleri için elde edilen sonuçlar

		PSO			**DE**			**YAK**			**DPD-YAK**		
	PBS	10	100	1000	10	100	1000	10	100	1000	10	100	1000
Sphere	*Ort.*	4,13E-17	5,14E-16	9,72E+03	4,41E-17	8,84E-17	3,29E+05	4,88E-17	1,08E-15	5,83E-02	4,76E-17	1,01E-15	5,65E-02
	SS	7,71E-18	3,12E-16	3,92E+03	8,09E-18	4,29E-17	9,18E+05	5,21E-18	1,04E-16	2,11E-02	5,13E-18	1,03E-16	2,01E-02
Rosenbrock	*Ort.*	4,26E-01	1,13E+02	1,68E+06	4,22E-17	1,32E+02	1,44E+10	1,31E-02	5,49E-02	2,60E+03	1,42E-02	6,73E-01	4,77E+04
	SS	1,19E+00	4,90E+01	6,48E+05	1,09E-17	4,17E+01	3,61E+08	8,66E-03	4,56E-02	5,99E+02	9,86E-03	9,89E-02	5,38E+03
Rastrigin	*Ort.*	7,36E+00	1,48E+02	2,72E+03	9,95E-02	1,33E+02	1,67E+03	4,76E-17	1,08E-15	7,36E+02	3,99E-17	9,87E-16	6,75E+02
	SS	2,49E+00	1,78E+01	8,31E+01	2,98E-01	1,07E+02	9,69E+01	4,40E-18	8,99E-17	2,48E+01	4,21E-18	6,49E-17	2,31E+01
Griewank	*Ort.*	5,93E-02	4,86E-02	8,60E+01	8,13E-03	7,40E-04	2,66E+02	5,10E-19	4,92E-17	1,03E-01	2,01E-18	5,11E-17	1,27E-01
	SS	3,37E-02	6,32E-02	2,92E+01	9,48E-03	2,22E-03	3,35E+02	1,93E-19	4,25E-18	6,82E-02	1,03E-18	4,72E-18	6,95E-02
Schwefel	*Ort.*	-2,65E+03	-2,01E+04	-1,88E+05	-4,17E+03	-3,12E+04	-2,53E+05	-4,19E+03	-4,19E+04	-3,51E+05	-4,17E+03	-4,01E+04	-3,75E+05
	SS	2,47E+02	1,76E+03	1,11E+04	4,74E+01	2,08E+03	1,77E+09	9,09E-13	3,30E-10	2,28E+03	1,41E-10	3,89E-08	1,58E+03
Ackley	*Ort.*	4,67E-17	7,32E-01	8,74E+00	4,86E-17	2,14E-16	1,75E+01	1,71E-16	4,21E-15	3,20E+00	9,88E-17	4,01E-15	1,07E+00
	SS	8,06E-18	7,55E-01	7,85E-01	6,55E-18	4,53E-17	3,82E+00	3,57E-17	3,09E-16	1,34E-01	1,07E-17	3,64E-16	1,03E-01
Step	*Ort.*	0	1,70E+00	1,35E+04	0	0	4,77E+04	0	0	0	0	0	0
	SS	0	2,61E+00	3,11E+03	0	0	2,71E+04	0	0	0	0	0	0
Dixon Price	*Ort.*	6,67E-01	2,08E+00	6,16E+05	6,67E-01	6,67E-01	3,25E+09	4,07E-16	1,26E-06	2,70E+03	1,90E-16	7,63E-07	2,00E+03
	SS	1,67E-14	4,23E+00	6,43E+05	4,97E-17	3,51E-17	1,08E+09	1,22E-16	7,21E-07	3,60E+02	8,13E-17	1,02E-07	2,66E+02

Tablo 2'deki sonuçlar değerlendirildiğinde, diğer algoritmalarda olduğu gibi DPD-YAK algoritmasında da problem parametreleri artırıldığında çözümlerin kötüleştiği görülmektedir. YAK algoritması, *Rosenbrock* fonksiyonu haricindeki diğer test fonksiyonlarında çok başarılı çözümler üretmiştir. YAK'ın önerilen bir türevi olarak DPD-YAK ise; *Sphere*, *Rastrigin*, *Schwefel*, *Ackley* ve *Dixon-Price* fonksiyonlarında standart YAK'tan daha iyi sonuçlara ulaşmış, *Step* fonksiyonunda ise her iki algoritma da 1000 ÇBS'de bile optimum sonuca ulaşmışlardır.

Tablo 3. ÇBS=100 ve farklı koloni boyutları ile elde edilen sonuçlar

		PSO			**DE**			**YAK**			**DPD-YAK**		
	PBS	10	100	1000	10	100	1000	10	100	1000	10	100	1000
Sphere	Ort.	6,16E-16	5,14E-14	3,11E-14	1,54E-16	8,84E-15	1,11E-14	1,12E-15	1,08E-15	1,02E-15	7,42E-16	1,08E-15	1,02E-15
	SS	2,86E-16	3,12E-16	5,02E-15	1,08E-17	4,29E-15	4,87E-15	8,78E-17	1,04E-16	7,56E-17	3,19E-17	1,04E-16	7,56E-17
Rosenbrock	Ort.	1,53E+02	1,13E+02	1,30E+02	2,44E+02	1,32E+09	6,56E+09	3,34E-01	5,49E-02	3,38E-02	9,63E-01	5,49E-02	3,38E-02
	SS	5,52E+01	4,90E+01	7,20E+01	1,67E+02	4,17E+09	2,54E+09	5,79E-01	4,56E-02	5,83E-02	6,87E-01	4,56E-02	5,83E-02
Rastrigin	Ort.	1,62E+02	1,48E+02	1,59E+02	9,07E+01	1,33E+09	6,67E+09	1,14E-15	1,08E-15	1,04E-15	8,54E-16	1,08E-15	1,04E-15
	SS	3,00E+01	1,78E+01	2,50E+01	1,16E+01	1,07E+09	5,31E+09	6,26E-17	8,99E-17	1,13E-16	2,71E-17	8,99E-17	1,13E-16
Griewank	Ort.	6,61E-02	4,86E-02	3,20E-03	3,48E-02	7,40E+05	2,52E-15	8,33E-17	4,92E-17	4,93E-17	7,98E-17	4,92E-17	4,93E-17
	SS	7,37E-02	6,32E-02	5,06E-03	3,86E-02	2,22E+06	2,86E-16	9,96E-17	4,25E-18	2,90E-18	5,36E-17	4,25E-18	2,90E-18
Schwefel	Ort.	-1,89E+04	-2,01E+04	-2,12E+04	-3,08E+04	-3,12E+04	-1,67E+04	-4,19E+04	-4,19E+04	-4,19E+04	-4,19E+04	-4,19E+04	-4,19E+04
	SS	1,45E+03	1,76E+03	2,16E+03	9,30E+02	2,08E+03	1,66E+03	2,24E-01	3,30E-10	3,99E-12	1,81E-01	8,09E-11	4,03E-13
Ackley	Ort.	1,86E+00	7,32E-01	6,80E-16	2,67E+00	2,14E-14	1,74E-12	4,48E-13	4,21E-15	4,11E-15	3,80E-13	1,77E-15	2,01E-15
	SS	7,39E-01	7,55E-01	9,64E-17	6,07E-01	4,53E-15	5,61E-13	2,14E-14	3,09E-16	1,59E-16	7,41E-15	7,61E-17	8,91E-17
Step	Ort.	3,85E+01	1,70E+01	2,00E-01	0	0	0	0	0	0	0	0	0
	SS	5,69E+01	2,61E+00	4,00E-01	0	0	0	0	0	0	0	0	0
Dixon Price	Ort.	1,70E+00	2,08E+00	6,67E-01	1,23E+01	6,67E-01	6,67E-01	1,64E-06	1,26E-06	6,23E-07	4,64E-07	1,00E-07	5,53E-08
	SS	3,09E+00	4,23E+00	2,81E-16	8,51E+00	3,51E-17	3,51E-17	1,25E-06	7,21E-07	4,45E-07	1,13E-07	8,01E-08	2,65E-08

Tablo 3 verilerine bakarak YAK algoritmasının farklı koloni boyutlarında başarılı çözümler üretebildiği görülebilmektedir. DPD-YAK versiyonu ise, *Rosenbrock* fonksiyonu haricinde diğer algoritmalardan daha başarılı çözümler elde etmiş, Step fonksiyonunda da YAK ile benzer olarak tüm koloni boyutlarında optimal sonuca ulaşmıştır.

5. Sonuç

YAK algoritması, bal arısı kolonisindeki iş bölümünde, yiyecek aramadan sorumlu üyelerin bireysel ve kolektif faaliyetlerini taklit eden, sürü zekâsı temelli metasezgisel bir yöntemdir. Algoritma, tüm prosedürleri için gerekli parametre değerlerini, iki kategoride birleştirebilmiştir: koloni boyutu (ÇS*2) ve limit. Bu yöntemi popüler kılan ana unsurlardan biri de zaten az sayıda kontrol parametresine ihtiyaç duymasıdır. Çalışmalar, YAK'ın farklı koloni boyutlarında bile, keşif için yeterli alanı tarayabildiğini göstermiştir.

Çalışmada, YAK'ın parametre değerlerini, çözüm arama sürecinde dinamik olarak güncelleyen metot önerilmektedir. Bu bağlamda gözcü arı sayısı, çözüm sayısına eşitlenmek yerine, herbir çevrimdeki güncel çözüm sonuçlarının ortalamasından daha iyi sonuç değerine sahip çözüm sayısına eşitlenmektedir. Yapay gözcü arılar, sırasıyla bu iyi çözümleri kullanarak yeni çözümler türetmektedir. Bu elitist yaklaşımla, algoritmanın, daha başarılı çözümleri kontrol etmesi, rulet tekerleğindeki olasılığa bırakılmamakta, daha aç gözlü bir strateji uygulanmaktadır. Ayrıca daha az gözcü arı, algoritmanın işlem yükünü hafifletmektedir. Diğer bir öneri ise, algoritmanın keşif/sömürü dengesini sağlayan limit parametresi için uygulanmaktadır. Standart YAK metodunda, tüm çözümler için atanan toplam limit değeri, her bir çevrimde, genel en iyi çözüm dikkate alınarak, çözüm başarılarına göre paylaştırılmaktadır. Bu sayede, iyi çözümlerin komşuluğu daha çok araştırılarak algoritmanın sömürü yeteneği geliştirilmekte, kötü çözümler etrafında oyalanmadan kaçınılarak, keşif yeteneği güçlendirilmektedir.

Uygulama, YAK parametre değerlerini analiz eden literatür çalışmalarında kullanılan sayısal optimizasyon problemleri üzerinde test edilmiştir. Farklı ÇS ve ÇBS değerleri ile yapılan deneylerden elde edilen sonuçlar, standart YAK ve diğer güncel metasezgisellerin sonuçlarıyla karşılaştırılmıştır. Analizler, önerilen metodun YAK algoritmasını daha da güçlendirdiğini ve diğer metasezgisel algoritmalarla yarışabilir seviyede olduğunu göstermektedir.

Sonraki çalışmada, DPD-YAK performansı, farklı sayısal optimizasyon problemleri üzerinde test edilecek ve arama sürecinde kullanılan yapay arı sayıları ile değişen limit değerleri analiz edilecektir.

Kaynakça

Akay, B., & Karaboga, D. (2009). Parameter Tuning for the Artificial Bee Colony Algorithm (Vol. 5796, pp. 608–619). https://doi.org/10.1007/978-3-642-04441-0_53

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization, 1001–1014. https://doi.org/10.1007/s10845-010-0393-4

Akay, B., & Karaboga, D. (2015). A survey on the applications of artificial bee colony in signal, image, and video processing. *Signal, Image and Video Processing*, *9*(4), 967–990. https://doi.org/10.1007/s11760-015-0758-4

Bacanin, N., & Tuba, M. (2012). Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. *Studies in Informatics and Control*, *21*(2), 137–146. https://doi.org/10.24846/v21i2y201203

Bansal, J. C., Sharma, H., & Jadon, S. S. (2013). Artificial bee colony algorithm: A survey. *International Journal of Advanced Intelligence Paradigms*, *5*(1–2), 123–159. https://doi.org/10.1504/IJAIP.2013.054681

Bartz-Beielstein, T., Chiarandini, M., Paquete, L., & Preuss, M. (2010). *Experimental Methods for the Analysis of Optimization Algorithms*. (T. Bartz-Beielstein, M. Chiarandini, L. Paquete, & M. Preuss, Eds.), *Experimental Methods for the Analysis of Optimization Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02538-9

Bensebti, M., & Bouchibane, F. Z. (2018). Artificial bee colony algorithm for energy efficiency optimisation in massive MIMO system. *International Journal of Wireless and Mobile Computing*, *15*(2), 97. https://doi.org/10.1504/IJWMC.2018.10016726

Elkhateeb, N., & Badr, R. (2017). A Novel Variable Population Size Artificial Bee Colony Algorithm with Convergence Analysis for Optimal Parameter Tuning. *International Journal of Computational Intelligence and Applications*, *16*(03), 1750018. https://doi.org/10.1142/S1469026817500183

Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Kayseri, Turkey. Retrieved from https://www.researchgate.net/publication/255638348_An_Idea_Based_on_Honey_Bee_Swarm_for_Numerical_Optimization_Technical_Report_-_TR06

Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132. https://doi.org/10.1016/j.amc.2009.03.090

Karaboga, D., Gorkemli, B., & Ozturk, C. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57. https://doi.org/10.1007/s10462-012-9328-0

Kockanat, S., & Karaboga, N. (2013). Parameter tuning of artificial bee colony algorithm for Gaussian noise elimination on digital images. In *2013 IEEE INISTA* (Vol. 1, pp. 1–4). IEEE. https://doi.org/10.1109/INISTA.2013.6577621

Korkmaz Tan, R., & Bora, Ş. (2019). Adaptive parameter tuning for agent-based modeling and simulation. *SIMULATION*, *95*(9), 771–796. https://doi.org/10.1177/0037549719846366

Liu, R., Wang, Z., & Xu, X. (2014). Parameter Tuning for ABC-Based Service Composition with End-to-End QoS Constraints. In *2014 IEEE International Conference on Web Services* (pp. 590–597). IEEE. https://doi.org/10.1109/ICWS.2014.88

Sakib, S., & Mahzabeen, E. (2018). ABC-T : Modified Artificial Bee Colony Algorithm with Pa- rameter Tuning for Continuous Function Optimization. *International Journal of Applied Information Systems (IJAIS)*, *12*(17), 1–7. https://doi.org/10.5120/ijais2018451781

Sharma, S., & Bhambu, P. (2016). Artificial Bee Colony Algorithm: A Survey. *International Journal of Computer Applications*, *149*(4), 11–19. https://doi.org/10.5120/ijca2016911384

Veček, N., Liu, S.-H., Črepinšek, M., & Mernik, M. (2017). On the Importance of the Artificial Bee Colony Control Parameter 'Limit.' *Information Technology And Control*, *46*(4), 566–604. https://doi.org/10.5755/j01.itc.46.4.18215