



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN YAPAY ARI KOLONİSİ ALGORİTMASININ İYİLEŞTİRİLMESİ

Ahmet ÖZKIŞ

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Eylül-2013
KONYA
Her Hakkı Saklıdır

ÖZET

YÜKSEK LİSANS TEZİ

SAYISAL OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN YAPAY ARI KOLONİSİ ALGORİTMASININ İYİLEŞTİRİLMESİ

Ahmet ÖZKIŞ

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Ahmet BABALIK

2013, 107 Sayfa

Jüri
Prof. Dr. Ahmet ARSLAN
Doç. Dr. Mehmet ÇUNKAŞ
Yrd. Doç. Dr. Ahmet BABALIK

Yapay arı kolonisi (Artificial Bee Colony - ABC) algoritması Karaboğa (2005) tarafından geliştirilen bal arılarının kendilerine özgü zeki davranışlarını modelleyen bir optimizasyon algoritmasıdır. Son yıllarda ABC optimizasyon alanında oldukça önemli bir yere sahip olmuştur ve sayısal optimizasyon, medikal veri sınıflandırma, veri madenciliği gibi birçok alanda kullanılmaktadır.

Bu tez çalışmasında, ABC algoritması üzerinde çeşitli düzenlemeler yapılarak Adaptif Boyut Limitli ABC (ABL-ABC, *Adaptive Dimension Limit ABC, ADL-ABC)* ve Hızlandırılmış ABC (H-ABC, *Accelerated ABC, A-ABC)* olarak isimlendirilen iki farklı algoritma önerilmiştir. Önerilen algoritmaların performans analizi, literatürde sıklıkla kullanılan sayısal optimizasyon problemleri üzerinde orijinal ABC algoritmasının ve yaygın olarak kullanılan diğer optimizasyon algoritmalarının elde ettiği sonuçlarla kıyaslanarak yapılmıştır. Elde edilen sonuçlar çizelgelerde ayrıntılı olarak verilmiştir.

Sonuçlar incelendiğinde önerilen *ADL-ABC* ve *A-ABC* algoritmalarının çoğu problemin çözümünde standart ABC algoritmasından daha iyi sonuç elde ettiği görülmüştür.

Anahtar Kelimeler: Sayısal Optimizasyon Problemleri, Optimizasyon Algoritmaları, Sürü Zekâsı, ABC Algoritması, Yapay Zekâ.

ABSTRACT

MS THESIS

IMPROVING OF ARTICIAL BEE COLONY ALGORITHM FOR SOLVING NUMERICAL OPTIMIZATION PROBLEMS

Ahmet ÖZKIŞ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Asst. Prof. Dr. Ahmet BABALIK

2013, 107 Pages

Jury
Prof. Dr. Ahmet ARSLAN
Assoc. Prof. Dr. Mehmet CUNKAŞ
Asst. Prof. Dr. Ahmet BABALIK

Artificial Bee Colony (ABC) algorithm is an optimization algorithm developed by Karaboga (2005) modeling particular smart behavior of honey bees. In recent years, ABC has gained a very important place in optimization field and is used in many fields such as numerical optimization, medical data classification, data mining.

In this thesis, two different modifications named as *Adaptive Dimension Limit (ADL-ABC)* and *Accelerated ABC (A-ABC)* algorithms are proposed by introducing some changes into the original ABC algorithms. Performance analysis of the proposed algorithms is done by comparing them with the standard ABC algorithm and other extensively-used optimization algorithms on the frequently-used numerical optimization problems. Obtained results are provided in detail through tables.

The results of the study have shown that proposed *ADL-ABC* and *A-ABC* algorithms generally produce better results than the standard version of the ABC algorithm.

Keywords: Numerical Optimization Problems, Optimization Algorithms, Swarm Intelligence, Artificial Bee Colony Algorithm, Artificial Intelligence.

ÖNSÖZ

Bu çalışmaya bilgi ve tecrübesi ile katkıda bulunan ve benden desteğini hiçbir zaman esirgemeyen danışman hocam Yrd. Doç. Dr. Ahmet BABALIK'a ve Selçuk Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü Öğretim Elemanları'na teşekkürlerimi sunarım.

Doğduğum günden beri üzerimde emekleri olan, beni her konuda destekleyen ve yüreklendiren, haklarını hiçbir zaman ödeyemeyeceğim sevgili aileme ve desteğinden dolayı sevgili eşime en içten teşekkürlerimi sunarım.

Ahmet ÖZKIŞ
KONYA-2013

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

SİMGELER VE KISALTMALAR ..... **ix**

1. GİRİŞ **1**

2. KAYNAK ARAŞTIRMASI **4**

3. GENEL KAVRAMLAR **9**

3.1. Optimizasyon 9
3.1.1. Amaç sayısına göre optimizasyon problemleri 11
3.1.2. Kısıtlara göre optimizasyon problemleri 11
3.1.3. Fonksiyon yapısına göre optimizasyon problemleri 11
3.1.4. Amaç fonksiyonunun karakteristiğine göre optimizasyon problemleri 12
3.1.5. Tasarım Değişkenlerinin Tipine Göre Optimizasyon Problemleri 13
3.2. Optimizasyon Algoritmaları 15
3.2.1. Deterministik Algoritmalar 16
3.2.2. Olasılıksal Algoritmalar 18
3.3. Bal Arılarının Doğası 25
3.3.1. Arıların Görev Paylaşımı 26
3.3.2. Gerçek Arıların Besin Arama (Foraging) Davranışı 28
3.4. Yapay Arı Kolonisi Algoritması 29
Başlangıçta rassal olarak yiyecek kaynaklarının oluşturulması: 30
Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi 31
Gözcü arılar için nektar miktarına göre kaynakların uygunluk değerlerinin hesaplanması 32
Gözcü arıların uygunluk değerlerine göre kaynak seçimi yapması 33
Kaynağı bırakma kriteri: limit ve kâşif arı üretme 33
Limit değeri 34
Seleksiyon mekanizmaları 36
ABC'nin temel özellikleri 36

4. ABC ALGORİTMASI ÜZERİNDE ÖNERİLEN DÜZENLEMELER **37**

4.1. Adaptive Dimension Limit ABC (ADL-ABC) Algoritması 37
4.1.1 ADL-ABC algoritmasında komşuluk ve boyut seçimi 37
4.1.2. ADL-ABC algoritmasında geliştirilememe sayacı 40
4.1.3. ADL-ABC algoritmasında limit değeri 41
4.1.4. ADL-ABC algoritmasında kaynağı bırakma kriteri ve kâşif arı üretimi 42
4.2. Accelerated ABC Algoritması (A-ABC) 45
4.2.1 Modification Rate (MR) 45

4.2.2 Step Size (SS) Modifikasyonu 46

5. ARAŞTIRMA SONUÇLARI VE TARTIŞMA 53

5.1. *ADL-ABC* ve *A-ABC* Algoritmalarının Temel Sayısal Optimizasyon Test Problemlerinin Çözümünde Performans Analizi 53
5.1.1. Temel Test Problemlerinin Çözümünde Kullanılan Parametreler 54
5.1.2. Temel Test Problemlerinin Sonuçları 56
5.2. CEC05 Test Problemleri 66
5.2.1. CEC05 test problemlerinin çözümünde kullanılan parametreler 66
5.2.2. CEC05 test problemlerinin sonuçları 67

6. SONUÇLAR VE ÖNERİLER 73

6.1. Sonuçlar 73
6.2. Öneriler 75

KAYNAKLAR 76

EKLER 81

ÖZGEÇMİŞ 107

SİMGELER VE KISALTMALAR

Simgeler

B : Boyut
Diff : Difference
GB_{fail} : Global best failure counter
GB_{succ} : Global best success counter
p : Besin kaynaklarının seçilme olasılığı
v_i : Aday kaynak
x_i : Var olan kaynak
ε : Kabul edilebilir bir hata miktarı

Kısaltmalar

ABC : Artificial Bee Colony
A-ABC : Accelerated Artificial Bee Colony
ADL-ABC : Adaptive Dimension Limit Artificial Bee Colony
BA : Bee Algorithm
BL : Boyut Listesi
DEA : Diferansiyel Evrim Algoritması
DE : Diferansiyel Evrim
FA : Firefly Algorithm
GA : Genetik Algoritma
GABC : Gbest-guided ABC
Gbest : Global Best
HS : Harmony Search
ILS : Iterated Local Search
Iter : İterasyon
KS : Kaynak Sayısı
MinLim : Minimum Limit
MakLim : Maksimum Limit
MÇS : Maksimum Çevrim Sayısı
MR : Modification Rate
NP-Zor : Non-deterministic Polinominal Hard
KKA : Karınca Koloni Algoritması
PS-EA : Particle Swarm Inspired Evolutionary Algorithm
PSO : Particle Swarm Optimization
RC : Reset Counter
SA : Simulated Annealing
SS : Step Size
YAK : Yapay Arı Kolonisi
YBA : Yapay Bağışıklık Algoritması
al : Alt Limit
ul : Üst Limit

1. GİRİŞ

Optimizasyon "en iyileme" anlamına gelmektedir. Optimizasyon, belirli şartlar ve kısıtlamalar altında en iyi sonuca (optimum) ulaşma sürecidir. Optimum, problemin türüne göre minimum ya da maksimum sonucu üreten çözümü ifade eder. Örneğin bir otomobilin 1 km yol gitmesi için harcayacağı yakıt miktarını azaltmaya çalışmak *minimizasyon*; bir litre yakıt ile en fazla yol gitmesini sağlamaya çalışmak ise *maksimizasyon* problemi olacaktır.

Optimizasyonda amaç $X=\{x_1,x_2,\ldots,x_n\}$ *girdi değerlerine* göre şekillenen $F=\{f_1, f_{2,\ldots,}f_n\}$ *maliyet değerleri* için X kümesinden mümkün olan en iyi değeri bulmaktır. Bunun kriteri matematiksel bir fonksiyon olarak tanımlanır ve *amaç fonksiyonu* olarak isimlendirilir. Bir hedef değere ulaşmak amacıyla çözülmeye çalışılan probleme *optimizasyon problemi*, bu problemin optimum çözümünü bulmak amacıyla geliştirilen algoritmalara da *optimizasyon algoritmaları* adı verilir. Optimizasyon algoritmaları genel olarak iki başlık altında toplanabilir: *deterministik algoritmalar* ve *olasılıksal algoritmalar*.

Bilimin cevabını aradığı sorular gün geçtikçe daha kompleks hale gelmektedir. Bu da insanoğlunu daha karmaşık sorulara cevap aramaya yöneltmektedir. Bir olayın meydana gelişini etkileyen faktörler çoğaldıkça, bu faktörlerin en uygun değerlerini bulmak da o derece zorlaşmaktadır. Zorluk seviyesi yüksek (Non-deterministic Polynomial-time Hard - NP-Hard) bazı problemlerin optimum çözümüne ulaşmak için matematiksel yöntemler çok uzun zamana ihtiyaç duyabilmektedir. İşte burada devreye olasılıksal optimizasyon kavramı girmektedir. NP-Hard olarak tanımlanan bu problemlerin çözümünde, sezgisel optimizasyon algoritmalarından sıklıkla faydalanılmaktadır. Sezgisel optimizasyon algoritmaları olasılıksal çözümler üreten çok sayıda yöntemi ifade eden bir üst başlıktır. Sezgisel optimizasyon yöntemlerinin alt başlıklarından biri de sürü zekası (swarm intelligence) algoritmalarıdır. Sürü zekâsı algoritmaları, doğada sürü halinde yaşayan bazı canlıların besin arama davranışları incelenerek geliştirilen sezgisel algoritmalardır. Literatürde Genetik Algoritma (Genetic Algorithm - GA) (Holland, 1975), Karınca Koloni Algoritması (Ant Colony Algorithm ACA) (Dorigo, 1991), Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) (Kennedy ve Eberhart, 1995), Diferansiyel Evrim Algoritması (Differantial Evoluation Algorithm - DEA) (Storn, 1995) başta olmak üzere çok sayıda doğa esinli sezgisel optimizasyon algoritması mevcuttur. Bu algoritmalardan biri de bal

arılarının besin arama davranışı modellenerek geliştirilen Yapay Arı Kolonisi (Artificial Bee Colony - ABC[*]) (Karaboğa, 2005), algoritmasıdır. ABC algoritması, son yıllarda sayısal optimizasyon problemlerinin çözümünde kullanılan popüler bir algoritmadır. Her ne kadar başarılı bir algoritma olsa da, yapılan son çalışmalar ABC algoritmasında yapılacak bazı değişikliklerin algoritmanın performansını artırmada yardımcı olacağını ortaya çıkarmaktadır. Böylece algoritmanın NP-Hard yapıdaki gerçek dünya problemlerinin çözümünde daha başarılı sonuçlar üretmesi sağlanarak literatüre katkı sağlanması amaçlanmaktadır.

Bu çalışmada, ABC algoritması üzerinde 2 farklı düzenleme önerilmiştir. İlki işçi arıların komşu seçimi düzenlemesi ile oluşturulan Adaptif Boyut Limitli ABC (ABL-ABC, *Adaptive Dimension Limit ABC, ADL-ABC*[**]*) algoritması*, ikincisi ise aynı anda birden fazla boyut değişimine imkân sağlayan Hızlandırılmış ABC (H-ABC, *Accelerated ABC, A-ABC*[***]*)* algoritmasıdır. Geliştirilen yöntemlerin performans kıyaslaması için literatürde sıklıkla kullanılan test fonksiyonlarından faydalanılmıştır.

ABC algoritmasının performansının artırılması amaçlanan bu çalışmanın ikinci bölümünde *kaynak araştırması* sunulmuş ve ABC algoritmasının gerçek hayat problemlerinin ve sayısal optimizasyon problemlerinin çözümünde kullanımıyla ilgili çalışmalara yer verilmiştir. Ayrıca ABC algoritmasının performansını iyileştirmek amacıyla yapılan düzenleme çalışmaları ve ABC algoritmasının diğer yöntemlerle melez kullanımını içeren çalışmalar da bu bölümde yer almaktadır.

Genel kavramlar bölümünde tez çalışmasında sıklıkla kullanılan optimizasyon, optimizasyon problemleri, problemlerin çözümünde kullanılan matematiksel ve sezgisel algoritmalar başlıkları hakkında bilgiler verilmiştir. Ayrıca gerçek bal arılarının görev paylaşımı ve nektar arama süreçleri hakkında bilgi verilmiş ardından ABC algoritmasının çalışma aşamaları ifade edilmiştir.

ABC algoritması üzerinde önerilen düzenlemeler bölümünde ABC algoritmasının performansını iyileştirmek amacıyla geliştirilen ADL-ABC ve A-ABC düzenlemeleri önerilmektedir.

---

[*] Derviş Karaboğa tarafından 2005 tarihinde önerilen Yapay Arı Kolonisi algoritması literatürde Artificial Bee Colony olarak kabul gördüğünden dolayı tezin ilerleyen kısımlarında ABC kısaltması tercih edilmiştir.

[**] Önerilen *Adaptif Boyut Limitli ABC* algoritması için tezin ilerleyen kısımlarında ADL- ABC kısaltması tercih edilmiştir.

[***] Önerilen *Hızlandırılmış ABC* algoritması için tezin ilerleyen kısımlarında A- ABC kısaltması tercih edilmiştir.

Araştırma sonuçları ve tartışma bölümünde önerilen yöntemler literatürde yaygın olarak kullanılan sayısal optimizasyon problemleri ile test edilmiş ve test sonuçları değerlendirilerek önerilen yöntemlerin hangi tür problemlerde başarı elde ettikleri analiz edilmiştir.

Sonuçlar ve öneriler bölümünde tez çalışması hakkında genel bir değerlendirme yapılmış ve önerilen yöntemlerin başarısı hakkında bilgi verilmiştir. Ayrıca sayısal optimizasyon problemleri alanında yapılacak yeni çalışmalar için önerilerde bulunulmuştur.

2. KAYNAK ARAŞTIRMASI

Yaklaşık son on yıldan bu yana, doğa esinli arı algoritmaları umut verici ve güçlü bir yöntem olarak ortaya çıkmaya başlamıştır. Arı algoritmalarının ilk formüle edildiği tarihi tam olarak belirlemek zor olsa da, son yıllarda farklı araştırmacı gruplar tarafından bağımsız olarak geliştirilen arı algoritması yöntemleri bilinmektedir.

Literatür araştırmaları incelendiğinde, ilk olarak Craig A. Tovey ve Sunil Nakrani (2004) tarafından farklı istemciler ve web sunucuları arasında bigisayarları paylaştırmak için Honey Bee Algoritm (HBA) yönteminin geliştirildiği görülmektedir. Daha sonra Xin-She Yang (2005), iki boyutlu fonksiyonları ve ayrık (discrete) sayısal optimizasyon problemlerini çözmek için Virtual Bee Algorithm (VBA) yöntemini geliştirmiştir. İlerleyen dönemlerde Haddad ve ark. (2005) Honey Bee Mating Optimization (HBMO) yöntemini sunmuş ve sırasıyla depo modelleme (reservoir modelling) ve kümeleme (clustering) alanında kullanmışlardır. Aynı dönemde Karaboğa (2005) sayısal optimizasyon problemleri için Artificial Bee Colony (ABC) algoritmasını geliştirmiş ve algoritmanın performansını karşılaştırma çalışmaları yapmıştır. Yukarıda bahsedilen doğa esinli arı algoritmaları her geçen gün daha popüler hale gelmektedir. Arı algoritmalarının temelinde arılar arası iletişim veya bir arının komşu arılara bilgi aktarma yeteneği vardır. Böylece arılar optimizasyon görevini tamamlayabilmek için en iyi kaynağı, bölgeleri veya rotaları bilebilir ve takip edebilirler (Yang, 2010).

ABC algoritması, Karaboğa (2005) tarafından bal arılarının besin arama davranışları modellenerek geliştirilmiş bir algoritmadır. ABC algoritması üzerine yapılan ilk çalışmalar benchmark fonksiyonları olarak bilinen sayısal optimizasyon problemlerinin çözümünü içermektedir. Yapay sinir ağlarının eğitimi, medikal veri sınıflandırma, kümeleme, gezgin satıcı probleminin (GSP) çözümü, sinyal işleme gibi farklı konularda geniş bir kullanım alanına sahip olduğu görülmektedir.

İlk olarak Karaboğa (2005), ABC algoritmasını önermiş ve 3 farklı benchmark fonksiyonu ile algoritmanın performansını test etmiştir. Karaboğa ve Baştürk (2007, 2008) 5 farklı benchmark problemi üzerinde GA, PSO, PS-EA ve ABC algoritmalarının, daha sonra ise DE, PSO, EA ve ABC algoritmalarının performans karşılaştırmasını yapmıştır.

Karaboğa ve Akay (2009), ABC algoritmasının performansını farklı karakteristikteki 50 benchmark fonksiyonu üzerinde test etmiş ve sonuçları GA, PSO ve DE algoritmalarıyla karşılaştırarak ABC algoritmasının üstünlüğünü sergilemiştir.

Bahsi geçen çalışmalar ABC algoritmasının farklı karakteristikteki fonksiyonların optimizasyonunda, karşılaştırma yapılan diğer sezgisel algoritmalara göre başarılı olduğunu göstermiş ve bu alandaki yeni çalışmaların önünü açmıştır. Akay (2009), sayısal optimizasyon problemlerinin çözümünde ve bazı gerçek hayat problemlerinin optimize edilmesinde ABC algoritmasının performansını analiz eden tez çalışmasını yayınlamıştır.

ABC algoritması sayısal optimizasyon alanında başarılı bir performansa sahip olmasına rağmen, araştırmacılar tarafından algoritmanın zayıf noktaları olduğu ifade edilmekte ve özellikle algoritmanın yerel arama (exploitation) ve/veya global arama (exploration) yeteneklerinin geliştirilmesinde kullanılabilecek iyileştirme önerileri sunulmaktadır.

Tsai ve ark., (2009) işçi ve gözcü arıların birbirini etkilemesinde Newton'un yer çekimi kanununu ABC algoritmasındaki gözcü arı fazına uygulayarak yeni bir yöntem önermişlerdir. ABC algoritmasında gözcü arıların kaynak seçiminde rassallık belirleyici olduğu için yerel arama yeteneği zayıftır. Yer çekimi kanunundan esinlenerek geliştirilen yöntem seçilen işçi arı ve gözcü arı arasında uygulanarak yerel arama yeteneği geliştirilmeye çalışılmıştır.

Alataş (2010), ABC algoritmasının yakınsama karakteristiğini geliştirmek ve yerel minimum değerlere takılmasını engellemek amacıyla kaynakların ve kaşif arıların başlangıç pozisyonunu, 7 farklı kaotik (düzensiz) yöntem kullanarak belirlediği bir yöntem önermiştir ve literatürde yaygın olarak kullanılan 3 farklı benchmark fonksiyonuyla önerilen yöntemi test ederek sonuçları analiz edilmiştir.

Zhu and Kwong (2010), ABC algoritmasının yerel arama yeteneğini geliştirmek için PSO algoritmasından esinlenerek, o ana kadar bulunan en iyi kaynağın (global best, g-best) koordinat bilgilerini aday kaynağın konumunun belirlenmesinde etkin hale getiren bir yöntem önermişlerdir.

Banharnsakun ve ark. (2011) ABC algoritmasının hem yerel hem de global arama yeteneğini geliştirmek için 3 temel düzenleme önermişlerdir: Bunar gözcü arı fazında o ana kadarki en iyi kaynağı komşu almak, ayarlanabilir arama yarıçapı ve fonksiyonun uygunluk (fitness) değeri yerine amaç (objective) değeri tabanlı karşılaştırmadır. Önerilen yöntemin performansı sayısal benchmark problemleri ve imge çakıştırma (image registration) uygulamalarında denenmiştir.

Kang ve ark. (2011), yerel arama yeteneği yüksek rosenbrock yöntemi (RM) ile ABC algoritmasını birleştirerek global minimumu daha hızlı yakınsayan, daha sağlam

bir teknik (Rosenbrock ABC, RABC) geliştirmişlerdir. ABC ve RABC yaygın olarak kullanılan 41 farklı benchmark fonksiyonu üzerinde karşılaştırılmıştır.

Karaboğa ve Akay (2011) Deb's kuralını kullanarak ABC algoritmasını kısıtlı (constrained) optimizasyon problemlerine uyarlamışlar, literatürdeki 13 kısıtlı optimizasyon problemi üzerinde ABC algoritmasının performansını hesaplamışlar ve elde edilen sonuçları 9 farklı algoritma ile kıyaslamışlardır.

Literatürde yaygın olarak kullanılan tek modlu, çok modlu ve hibrid yapıdaki sürekli optimizasyon problemlerinin ABC algoritması ile çözümünde algoritmanın yakınsama performansını geliştirmek amacıyla Akay ve Karaboğa (2012) tarafından algoritmaya *modification rate* (MR) parametresi eklenmiş ve bu sayede yukarıda sözü edilen problemlerin birden çok boyutu tek modifikasyonda güncellenebilmiştir. Aynı çalışmada *scaling factor* (SF) parametresi eklenerek arama uzayında uygun bir adım genişliğiyle arama yapılması sağlanmıştır.

Gao ve Liu (2012) ABC algoritmasının yerel arama yeteneğini geliştirmek için DE algoritmasından ilham alarak bir önceki iterasyonun en iyi kaynağı etrafında arama yapan bir yöntem geliştirmişlerdir. Yerel ve global arama arasındaki dengeyi sağlamak için de *selective probability* (P) parametresini kullanmışlardır. Ayrıca global yakınsamayı geliştirmek için başlangıç popülasyonunu kaotik sistemler ve karşıtlık tabanlı yöntemler kullanılarak oluşturmuşlardır. Önerilen yöntem 28 farklı benchmark fonksiyonu üzerinde denenmiştir.

Kıran ve Gündüz (2013), ABC ve PSO algoritmalarını birleştirerek yeni bir yöntem önermişlerdir. ABC ve PSO tarafından elde edilen en iyi çözümler birleşim için kullanılmış ve elde edilen birleşim PSO algoritmasına global en iyi, ABC algoritmasına ise gözcü arı fazında komşu olarak verilmiştir. ABC ve PSO arasındaki bilgi akışının hem yerel hem de global arama yeteneğini arttırdığını belirtmişlerdir.

ABC algoritmasının sayısal optimizasyon alanında göstermiş olduğu başarı, araştırmacıları gerçek hayat problemlerini ABC algoritmasını kullanarak çözmeye yönlendirmiştir.

Sinyal işleme sistemlerinde, digital sonsuz darbe cevaplı filtre (Infinite Impulse Response - IIR) tasarımında ABC algoritması temelli yeni bir yöntem geliştirilmiş ve bu yöntemin performansı diğer yöntemlerle kıyaslanmıştır (N. Karaboğa, 2009).

Yaprak kısıtlı minimum kapsama ağacı problemi (Spanning Tree Problem - STP)'nin çözümünde ABC algoritması kullanılmıştır (Singh, 2009).

Zhang ve ark. (2010), ABC algoritmasını popüler bir veri madenciliği tekniği olan veri kümeleme (data clustering) işlemi için kullanmışlardır. Geliştirilen yöntem literatürde yaygın olarak kullanılan iris, tiroid bezi ve şarap veri setleri üzerinde denenmiş ve sonuçlar diğer algoritmalar ile kıyaslanmıştır.

Sabat ve ark. (2010), ABC algoritmasını yarı iletken alan etkili transistörlerin küçük sinyal modeli parametre çıkarımına uygulamışlardır.

Karaboğa ve Öztürk (2011), kanser, diyabet ve kalp hastalıklarının teşhisi gibi 13 farklı kümeleme problemini ABC algoritması ile çözmüş, sonuçları PSO algoritması ve literatürde yaygın olarak kullanılan 9 farklı kümeleme tekniği ile kıyaslamıştır.

Taşgetiren ve ark. (2011), ABC algoritmasının kesikli bir modelini geliştirmişler ve bu modelle iş akış (flow-shop) diziliminde toplam süreyi minimize etmeyi amaçlamışlardır. Geliştirilen yöntem akış tipi çizelgeleme (flow shop scheduling) problemlerinde kafile aktarma (lot-streaming) işlemi için kullanılmıştır.

Pan ve ark. (2011), hem rölanti hem de rölanti olmayan durumlarda, erken gelme ve geç kalma (rötar) durumlarında ağırlıklandırılmış ceza kriterleriyle kafile aktarma akış tipi çizelgeleme problemini çözmek için kesikli bir ABC algoritması modeli önermiştir.

Horng (2011), resim bölütleme için ABC algoritması üzerinde maksimum düzensizlik eşiği (entropy thresholding) tabanlı bir yöntem önermiştir.

Ma ve ark. (2011), sentetik aralıklı radar (Synthetic Aperture Radar - SAR) resim bölütleme için ABC algoritmasını kullanmışlardır.

Omkar ve ark. (2011), katmanlı bileşim parçalarının çok amaçlı tasarımının optimizasyonunda VEABC'yi önermişler ve VEABC'nin performansını diğer popülasyon tabanlı yöntemlerle karşılaştırmışlardır.

De Oliveira ve Schirru (2011), çekirdek yakıt yönetimi optimizasyonunda, Samanta ve Chakraborty (2011) geleneksel olmayan mekanik işleme sürecinin parametre optimizasyonunda, Yeh ve Hsieh (2011) güvenilirlik fazlalık tahsisi sorunlarının çözümlerinde, Sönmez (2011) kafes yapısını minimize etmek için, Gözde ve Taplamacıoğlu (2011) otomatik voltaj regülatörü sistemlerinin optimum kontrolünü sağlamak amacıyla, Szeto ve Wu (2011) araç rotalama problemi (vehicle routing problem) çözümünde, Manoj ve Elias (2012) düzensiz filtre set çoklayıcısı (Nonuniform filter bank transmultiplexer - NUFB TMUX) tasarımında donanım karmaşıklığını azaltmak ve enerji verimliliği sağlamak amacıyla, ABC algoritmasını kullanmışlardır.

Banharnsakun ve ark. (2012), iş çizelgeleme problemlerini çözmek için *Best-so-far Artificial Bee Colony* yöntemini kullanmışlardır. Bu yöntemde yeni oluşacak çözümler o ana kadar ulaşılan en iyi çözüme doğru yönlendirilmiştir. Önerilen yöntem 62 farklı benchmark fonksiyonu üzerinde test edilmiş ve başarılı sonuçlar elde ettiği görülmüştür.

3. GENEL KAVRAMLAR

Tez çalışmasında, sayısal optimizasyon problemlerinin çözümünde ABC algoritmasının performansını artırıcı değişiklikler önerilmiştir. ABC'nin temel algoritması üzerinde yapılan değişiklikler ile ortaya konulan iki farklı yaklaşım, test problemlerine uygulanmış ve elde edilen sonuçlar literatürde bulunan sonuçlar ile kıyaslanmıştır.

3.1. Optimizasyon

Optimizasyon mühendislik tasarımından ekonomi dünyasına, bilgisayar dünyasından endüstriyel uygulamalara, günlük aktivitelerimizden tatil planlamalarımıza kadar tüm alanlarda mevcuttur. Hayatta her zaman bir şey maksimize ya da minimize edilmeye çalışılmaktadır. Bir durumun maksimum fayda getirmesi, maliyetinin minimum olması yani maksimum performans elde edilmesi arzu edilir. Tatilimizi planlarken bile, en az harcama yaparak (ya da ideal olarak hiç harcama yapmaksızın) en fazla eğlenceyi elde etmek isteriz. Aslında karşılaştığımız tüm problemlerde, sürekli olarak optimal çözümü elde etmeye çalışırız. Bu açıdan, optimizasyon problemlerini çözmek insanlık tarihi ile yaşıt bir uğraştır demek abartılı bir ifade olmayacaktır (Yang, 2010).

En basit anlamı ile optimizasyon eldeki kısıtlı kaynakları en uygun biçimde kullanmak olarak tanımlanabilir (Bunday, 1984). Matematiksel olarak ifade etmek gerekirse optimizasyon kısaca bir fonksiyonun minimize veya maksimize edilmesi olarak tanımlanabilir (Kahaner ve ark., 1989). Diğer bir deyişle optimizasyon "en iyi amaç kriterinin en iyi değerini veren kısıtlardaki değişkenlerin değerini bulmaktır" (Himmelblau ve Edgar, 1989). Başka bir tanımlama ile "belirli amaçları gerçekleştirmek için en iyi kararları verme sanatı" veya "belirli koşullar altında herhangi bir şeyi en iyi yapma" (Kübat,C., 1983) olarak da tanımlanan optimizasyon kısaca "en iyi sonuçları içeren işlemler topluluğudur" (Rao,1978).

Optimizasyonda amaç maksimum kâr veya minimum maliyeti sağlayacak üretim miktarını kısıtlara bağlı olarak tespit etmektir. Günümüzün bilgisayar teknolojisi kadar güncel bir kavram olan optimizasyon kavramı çok çeşitli endüstri kesimlerinde uygulama olanağı bulmuştur.

Optimizasyon, bir sistemde varolan kaynakların (işgücü, zaman, kapital, süreçler, hammaddeler, kapasite, ekipman gibi) en verimli şekilde kullanılarak belirli amaçlara (maliyet en azaltılması, kâr en çoklanması, kapasite kullanımının en yükseltilmesi ve verimliliğin en çoklanması gibi) ulaşmayı sağlayan bir teknoloji olarak tanımlanmaktadır (Gass, 2000).

Optimizasyon yöntemleri, karar verme süreçlerini hızlandırmakta ve karar kalitesini arttırarak gerçek hayatta karşılaşılan problemlerin etkin, doğru ve gerçek zamanlı çözümüne katkı sağlamaktadır (Winston, 2003). Optimizasyon, ekonomik açılardan getirdiği kazançların yanı sıra müşteri, işveren ve çalışanların istek ve kısıtlarının karar sürecinde yer almasında ve sistemde yer alan kaynakların kalitesinin yükseltilmesinde de etkin bir şekilde faydalanılan bir yöntem olarak kullanılmaktadır.

Değişen teknolojilerin, sınırlı kaynakların, artan rekabetin, karmaşık hale gelen sistemlerin doğurduğu problemlerin klasik yöntemlerle (matematiksel veya matematiksel olmayan, analitik veya sayısal) çözümünün güçleşmesi optimizasyon kavramını güncelleştiren en önemli sebeptir. Bu yönüyle optimizasyonun kullanılmadığı bir bilim dalı hemen hemen yok gibidir (Kara, 1986).

Optimize edilecek bir tasarımın faktörleri aşağıda verilmiştir (Yang 2010; Çelik 2013):

$$\text{min imize et } x \in R \qquad f_i(x), \qquad (i = 1, 2, ..., M), \tag{3.1}$$

$$\text{amaç} \qquad \phi_j(x) = 0, \ (j = 1, 2, ..., J), \tag{3.2}$$

$$\psi_k(x) \le 0, \ (k = 1, 2, ..., K), \tag{3.3}$$

Burada $f_i(x)$, $\phi_j(x)$, $\psi_k(x)$ tasarım vektörünün fonksiyonlarıdır. $f_i(x)$ *amaç (maliyet) fonksiyonu,* $\phi_j(x)$ eşitliği ve $\psi_k(x)$ eşitsizliği ise *tasarım vektörünün kısıtları* olarak ifade edilmektedir. R *tasarım alanı* ya da *arama uzayı* olarak isimlendirilir. Optimizasyon problemlerinde arama uzayı belirlenen alt limit (*al*) ve üst limit (*ul*) değeriyle sınırlandırılır ve arama yalnız bu sınırlar aralığında sürdürülür.

$$f_i(x) = \{x_1, x_2, ..., x_n\} \tag{3.4}$$

$$1 \le m \le n \text{ olmak üzere, } al_m \le x_m \le ul_m \tag{3.5}$$

Eşitlik 3.4'te x çözümünün her bir x_m parametresi *tasarım değişkeni* olarak isimlendirilir. Bu değişkenler sürekli, ayrık ya da melez olabilirler.

Optimizasyon problemleri fonksiyonun amaç ya da kısıt sayısına, fonksiyon yapısına, amaç fonksiyonunun karakteristiğine, tasarım değişkenlerinin tipine,

değerlerin belirsizliğine, hesaplama güçlüğüne göre aşağıdaki gibi sınıflandırılmıştır (Yang, 2010):

3.1.1. Amaç sayısına göre optimizasyon problemleri

Optimizasyon problemlerini amaç sayısına göre sınıflandırırsak iki farklı kategori oluşur. Eşitlik 3.1'de verilen:

- *M=1* ise *tek amaçlı,*
- *M >1* ise *çok amaçlı* (çok kriterli, çok nitelikli),

optimizasyon problemi anlamına gelir.

Gerçek dünya tasarımlarının ya da karar alma problemlerinin birçoğu çok amaçlıdır. Bu problemlerde m (m>1) adet amaç fonksiyonunun her biri optimize edilecek bir kriteri belirtir. (Kalyanmoy, 2001; Coello 1999; Fonseca ve Fleming, 1998).

3.1.2. Kısıtlara göre optimizasyon problemleri

Optimizasyon problemleri kısıt açısından J ve K değerlerine bağlı olarak aşağıdaki gibi sınıflandırılabilir:

- *J=K=0* (problemde herhangi bir kısıt yoksa) ise kısıtsız optimizasyon problemi,
- $J \geq 1$ ve *K=0* ise eşitlik kısıtlı,
- *J=0* ve $K \geq 1$ ise eşitsizlik kısıtlı,

optimizasyon problemi olarak ifade edilir.

3.1.3. Fonksiyon yapısına göre optimizasyon problemleri

Optimizasyon problemleri fonksiyon yapısına göre de sınıflandırılabilir:

- Kısıt fonksiyonları $\phi_j(x)$ ve $\psi_k(x)$'nın her ikisi de doğrusal (lineer) ise doğrusal kısıtlı,
- Amaç fonksiyonu $f(x)$ ve kısıt fonksiyonları $\phi_j(x)$ ve $\psi_k(x)$'nın tamamı doğrusal ise doğrusal,

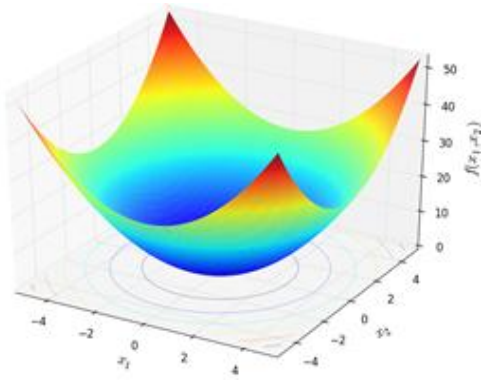
- Amaç fonksiyonu $f(x)$ ve kısıt fonksiyonları $\phi_j(x)$ ve $\psi_k(x)$'nın tamamı doğrusal değil (non-lineer) ise doğrusal olmayan,

optimizasyon problemi olarak ifade edilir.

3.1.4. Amaç fonksiyonunun karakteristiğine göre optimizasyon problemleri

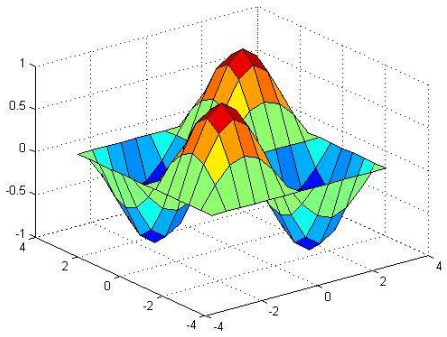
Optimizasyon problemleri amaç fonksiyonunun karakteristiğine göre de sınıflandırılabilir (Yang, 2010):

Tek modlu (Unimodal) Fonksiyonlar: Fonksiyon sadece bir adet vadiye (global minimum) ya da zirveye (global maksimum) sahip ise tek modlu olarak isimlendirilir. Bu durumda yerel optimum, vadideki tek optimum aynı zamanda global optimum değer olur. Örneğin $f(x,y) = x^2 + y^2$ yerel minimumu (0,0) noktasında yer alan tek modlu bir fonksiyondur. Bu nokta aynı zamanda global minimumu da göstermektedir. Tek modlu fonksiyonlar konveks (dış bükey) yapıda ve amaç fonksiyonunun iyileşme yönünün global minimum çözüme gittiğini garanti eden fonksiyonlardır. Tek modlu $f(x,y) = x^2 + y^2$ fonksiyonunun grafiği Şekil 3.1'de görülmektedir.



Şekil 3.1 Tek modlu $f(x,y) = x^2+y^2$ fonksiyonunun grafiği

Çok modlu (Multimodal) Fonksiyonlar: Fonksiyon birden fazla global minimum ya da global maksimum değere sahip ise çok modlu olarak isimlendirilir. Bu durumda yerel optimum noktalardan biri ya da birkaçı global optimum noktayı işaret etmektedir. Çok modlu $f(x,y) = \sin(x)\sin(y)$ fonksiyonunun grafiği Şekil 3.2'de görülmektedir.



Şekil 3.2 Çok modlu *f(x,y)=sin(x)sin(y)* fonksiyonunun grafiği

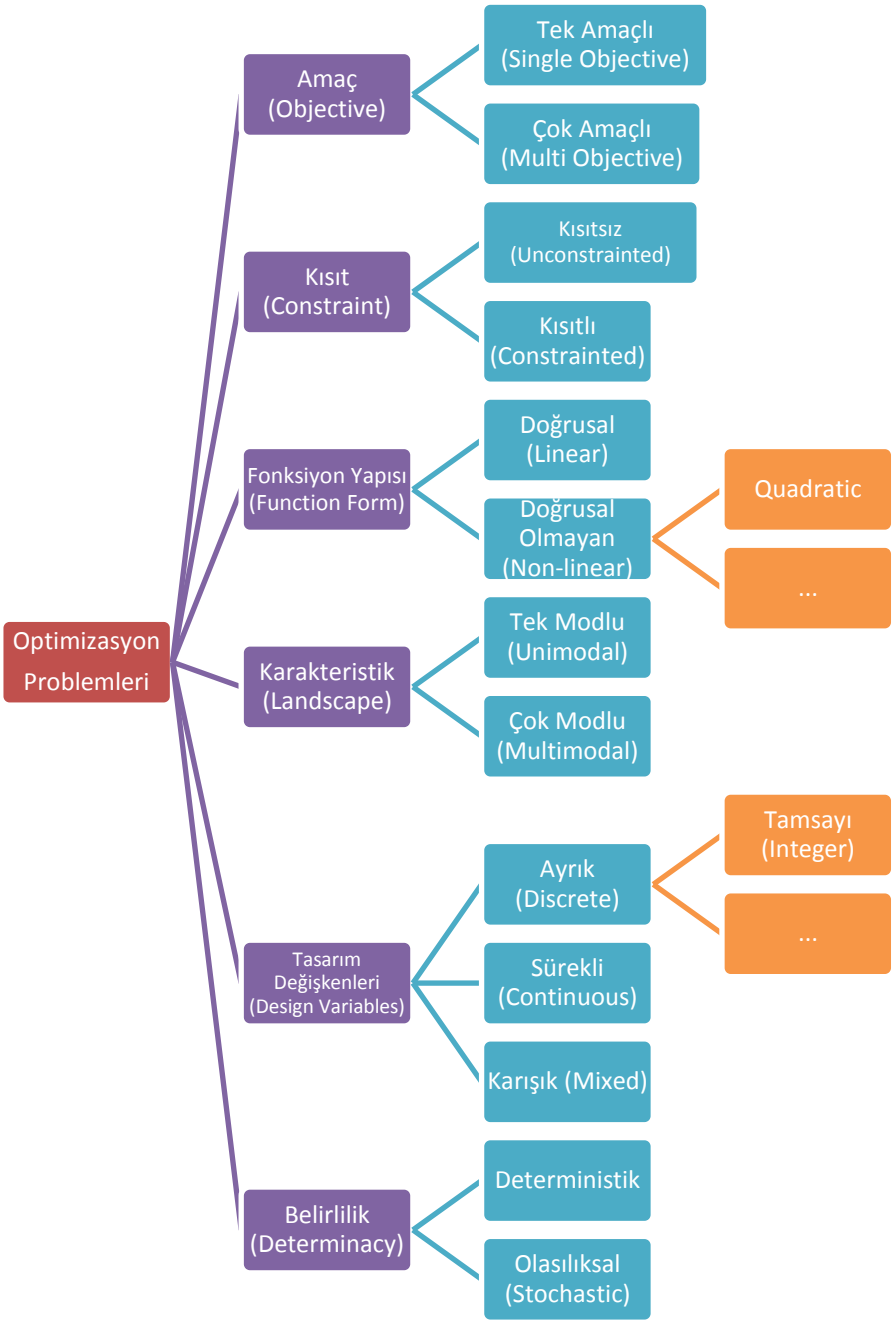
3.1.5. Tasarım Değişkenlerinin Tipine Göre Optimizasyon Problemleri

$f_i(x) = \{x_1, x_2, ..., x_n\}$ fonksiyonundaki tasarım değişkenlerinin tipi de optimizasyon problemlerinin sınıflandırılmasında kullanabilir.

- Tasarım değişkenleri ayrık ise *ayrık optimizasyon problemi*,
- Tasarım değişkenleri reel sayı (sürekli) ise *sürekli optimizasyon problemi*,
- Tasarım değişkenleri hem ayrık hem de sürekli olabiliyorsa *karma (melez) optimizasyon problemi*,

olarak ifade edilir.

Bazı kaynaklar ayrık optimizasyonun tamsayı programlama ve kombinasyonal optimizasyonu da kapsayan bir ifade olduğunu belirtse de, ayrık optimizasyon kombinasyonal optimizasyon anlamında da kullanılmaktadır. Kombinasyonal optimizasyon gezgin satıcı problemi, asgari tarama ağacı problemi (minimum spanning tree problem), araç rotalama problemi (vehicle routing problem), hava yolu planlama (airline scheduling) ve knapsack problemleri gibi graf teorisi, rotalama ile ilgili alanlarda yaygın olarak kullanılan bir yöntemdir. Ayrıca tüm tasarım değişkenleri belli bir aralıkta sürekli ya da reel değerler alıyorsa optimizasyon *sürekli optimizasyon problemi* olarak isimlendirilir. Tasarım değişkenlerinin hem sürekli hem de ayrık değerler alabildiği problemler karma tür olarak isimlendirilir. Şekil 3.3'te optimizasyon problemlerinin türleri gösterilmektedir.



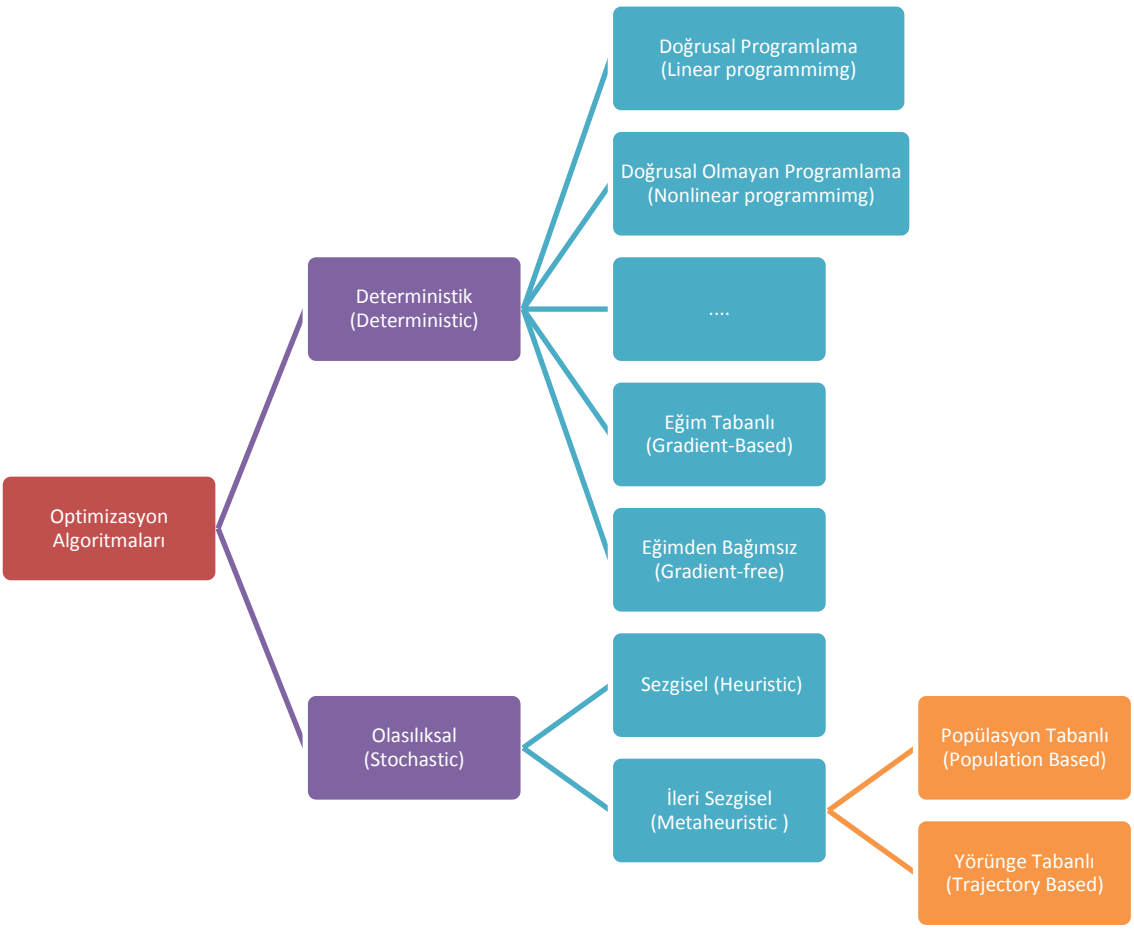
Şekil 3.3 Optimizasyon Problemlerinin Sınıflandırılması (Yang, 2010)

Deterministik optimizasyon problemlerinde hem amaç fonksiyonunun hem de kısıtların tasarım değişkenleri kesin olarak bellidir. Gerçek dünya problemlerinde ise bazı tasarım değişkenleri tam olarak bilinirken bazıları bilinemeyebilir. Tasarım değişkenlerinde, amaç fonksiyonunda ya da kısıtlarda herhangi bir belirsizlik söz konusu olursa optimizasyon *olasılıksal (stochastic) optimizasyon problemi* olur (Yang, 2010).

3.2. Optimizasyon Algoritmaları

Optimizasyon problemlerinin çözümünde kullanılan algoritmalar kısaca optimizasyon algoritmaları olarak da isimlendirilmektedir. Genel olarak optimizasyon algoritmaları iki kategoriye ayrılır: deterministik algoritmalar, sezgisel algoritmalar. Deterministik algoritmalar kesin bir işlem dizisini takip ederler. Böylece tasarım değişkenleri ve amaç fonksiyonu aynı değeri alarak aynı çözüm rotasını tekrarlayabilir.

Örneğin tepe tırmanma (hill-climbing) algoritması (Russell ve Norvig, 2002; Yang, 2010) deterministik bir algoritmadır ve aynı başlangıç noktası için her çalıştırmada aynı çözüm yolunu izleyecektir. Diğer yandan sezgisel algoritmalar her zaman bir rastgelelik içerirler. Örneğin genetik algoritmalar çözüm sürecinde rastgele oluşturulan bazı değerleri kulllandığı için elde edilen çözümler birbirlerine çok yakın olsalar dahi kesin olarak aynı yolu kullanmazlar. Ayrıca deterministik ve sezgisel yöntemlerin melez (hibrit) kullanımı ile farklı yöntemler de geliştirilmiştir. Şekil 3.4'te optimizasyon algoritmalarının sınıflandırılması gösterilmektedir.



Şekil 3.4 Optimizasyon Algoritmalarının Sınıflandırılması (Yang, 2010).

3.2.1. Deterministik Algoritmalar

3.2.1.1. Doğrusal programlama algoritmaları

Doğrusal programlamanın amacı, doğrusal amaç fonksiyonunu minimize ya da maksimize eden doğrusal sınır değerler arasındaki karar değişkenlerini belirlemektir. Bir diğer deyişle doğrusal programlamadaki temel hedef, amaç fonksiyonunu minimize eden ve kısıtları sağlayan noktayı bulmaktır. Kısıt değerleri sağlayan tüm noktalar *uygun nokta* olarak isimlendirilir. Bir doğrusal programlama probleminde amaç fonksiyonu doğrusaldır ve uygun noktalar bir grup eşitlik ya da eşitsizlikle tanımlanır. (Yang, 2010).

3.2.1.2. Doğrusal olmayan programlama algoritmaları

Gerçek dünya problemlerinin birçoğu doğrusal değildir. Bu sebeple doğrusal olmayan matematiksel programlama yöntemleri, matematiksel optimizasyon yöntemleri içinde çok önemli bir yere sahiptir. Doğrusal olmayan programlama problemlerinin önemli bir bölümü kısıtsız amaç fonksiyonlarının minimizasyonu ya da maksimizasyonunu, diğer bir bölümü ise ikinci dereceden, doğrusal olmayan kısıtlı amaç fonksiyonlarının minimizasyonunu hedeflemektedir. Farklı yapılarda daha birçok doğrusal olmayan programlama problemleri mevcuttur.

Doğrusal olmayan programlama problemleri tanımlanan fonksiyonun konveksliğine (dışbükey) göre sınıflandırılır. Konveks fonksiyonların önemli bir özelliği de eğimi sıfır olan çözüm noktasının amaç fonksiyonunu global minimum ya da maksimum noktaya götürmeyi garanti etmesidir (Yang, 2010).

3.2.1.3. Eğim tabanlı programlama algoritmaları

R^n uzayında reel değerli fonksiyonlar için arama yapan yöntemlerdir. Bu yöntemler arama yapmak için verilen fonksiyonun eğim bilgisinden faydalanırlar.

Newton yöntemi gibi türev tabanlı ya da sabit nokta yöntemleri, analitik bölgesel arama yapan yöntemlerdir. Türev tabanlı yöntemlerde türev ya da yaklaşık türev bilgisi hesaplanarak arama uzayında hangi yönde hareket edileceği belirlenir. Sabit nokta

yöntemleri ise bir fonksiyonun ürettiği sonucun sabit noktasının bulunması ile optimal değere iteratif bir şekilde yakınsarlar (Akay, 2009).

Eğim bilgisinden faydalanarak arama yapan algoritmaların bazıları aşağıda sıralanmıştır:

- Newton Algoritması
- Dik İniş (Steepest Descent) Algoritması
- Çizgi Arama (Line Search) Algoritması
- Birleşik Eğim (Conjugate Gradient) Algoritması

3.2.1.4. Eğimden bağımsız algoritmalar

Eğimden bağımsız (Non-gradient-based ya da gradient-free) algoritmalar isimlerinden de anlaşılacağı gibi türev bilgisinden faydalanmazlar. Bu algoritmalar sadece fonksiyonun maliyet değerinden yararlanırlar. Hooke-jeeves pattern search ve Nelder-Mead downhill simplex eğimden bağımsız algoritmalara örnek verilebilir (Yang, 2010).

Deterministik yöntemlerin avantaj ve dezavantajları aşağıdaki gibi sıralanmıştır (Akay, 2009; Dumitrescu ve Stützle, 2003):

Avantajları

- Algoritma başarılı olarak sonuca ulaştığında elde edilen çözüm optimum değerdir.
- Alt ve üst sınırlardaki optimum çözüme ulaştıran bilgiler elde edilebilir.
- Optimal çözümün olmadığı bölgeler saf dışı bırakılmaktadır.

Dezavantajları

- Çözülen problemlerin çoğunun boyutu küçüktür. Problem boyutunun artması tekniğin hesaplama maliyetini çok fazla yükseltmektedir.
- Hafıza tüketimi sorunundan dolayı program erken sonlanmaktadır.
- Bir problem için çok iyi sonuç veren bir yöntemin formüldeki bir değişime uyarlanması oldukça zordur.
- Bir problem için çok iyi sonuç veren bir yöntemin başka bir probleme uyarlanması zor olabilir.

Çok fazla sayıda optimum değerin olduğu durumlarda, yerel arama yöntemleri yerel bir optimumu bulabilirler. Yerel optimuma takılmayı önlemek için komşuluk boyutu arttırılabilir. Böyle bir durumda komşuluk boyutunun artması algoritmanın yakınsama hızını düşürecektir. Yakınsama hızının azalması ise zor problemlerde kabul edilemeyecek ve üstel olarak artan bir çalışma süresine neden olacaktır (Vesterstrom ve Riget, 2002; Akay, 2009). Deterministik yöntemlerin yukarda bahsedilen dezavantajlarından dolayı araştırmacılar doğada sürüler halinde yaşayan canlıların davranışlarını taklit ederek olasılıksal algoritmalar geliştirmişlerdir.

3.2.2. Olasılıksal Algoritmalar

Dağlık bir arazide gizlenmiş bir hazineyi sınırlı bir zaman dilimi içinde bulmaya çalıştığımızı düşenelim. Durumu zorlaştıran ilk kriter olarak gözlerimizin bağlı olduğunu ve herhangi bir rehber olmaksızın rastgele arama yaptığımızı farzedelim. Bir diğer zorlaştırıcı unsur olarak hazine dağın rastgele bir noktasına saklanmış olsun yani dağdaki zirve ya da vadiler hazine hakkında bir ipucu vermesin. Çok geniş bir alan üzerinde her bir noktayı aramaya çalışmak zahmetli bir düşüncedir. Yapılacak en mantıklı hareket rassal adımlar atarak bazı ipuçları aramak olacaktır. İlk olarak bazı bölgeler rassal bir şekilde aranır, daha sonra makul bölgeler üzerinde arama devam eder. Bu mantıktaki rassal aramalar modern arama algoritmalarının temelidir. Alternatif olarak, bir grup avcının altın aradığını ve birbirleriyle elde ettikleri bilgiyi paylaştıklarını düşünelim. Böyle bir senaryoda avcılar *sürü zekasını* kullanırlar. Bazı avcılar hazine arama sürecinde diğerlerinden daha başarılı olacaktır. Sadece daha başarılı olan avcıları elimizde tutup, başarısız olanlar yerine yeni avcılar alalım. Tüm modern algoritmalarda, avcı örneğinde olduğu gibi eldeki iyi çözümler kullanılarak rassal etkilerle çok iyi çözüm üretmeyen parçacıkların konumları değiştirilir. Böylece arazide en çok hazineyi içeren noktaya ulaşılmaya çalışılır (Yang, 2010). Olasılıksal algoritmalar sezgisel (heuristic) ve ileri sezgisel (metaheuristic) olmak üzere iki başlık altında incelenebilir.

3.2.2.1 Sezgisel (Heuristic) Algoritmalar

Heuristic buluşsal yol, kestirme yol, sezgisel yöntem anlamlarına gelen bir kelimedir. Optimizasyon alanında ise kompleks problemler üzerinde deneme yanılma yöntemi uygulayarak kabul edilebilir çözümleri makul bir sürede üretme stratejisi olarak isimlendirilebilir. Çözülmeye çalışılan problemin kompleksliği her bir aday çözümü ya da kombinasyonu değerlendirmeyi imkansız kılabilir. Böyle durumlarda sezgisel yöntemler kullanılarak iyi ve uygun çözümleri makul bir zaman diliminde bulmak amaçlanır. Bulunan makul çözümler içinden bazılarının optimum çözüme yakın olması hedeflenir. Ancak en iyi çözümün hatta makul bir çözümün bile bulunup bulunamayacağı garanti edilemez (Yang, 2010).

Sezgisel algoritmalar ilk defa Alan Turing tarafından II.Dünya Savaşı esnasında Alman Enigma şifrelerini kırmak amacıyla kullanılmıştır. 1940'ta Turing, İngiliz matematikçi Gordon Welchman ile birlikte şifre kırmada kendilerine yardımcı olması amacıyla *Bombe* adını verdikleri bir makine geliştirmiştir. Bombe, 10^{22} olası kombinasyondan doğru olduğu varsayılan mesajı aramak için Turing'in deyimiyle *sezgisel algoritma* yöntemini kullanmıştır. Sezgisel algoritma yöntemi çok büyük bir başarı elde etmiştir (Yang, 2010).

Sezgisel algoritmalar üzerine bir sonraki önemli gelişme 1960'larda gerçekleşmiştir. İlk olarak Michigan Üniversitesi'nde John Holland ve yardımcıları genetik algoritmayı geliştirmişlerdir. 1962'de Holland adaptif sistemler üzerine çalışmış ve sistemleri modellemek için çaprazlama (crossover) ve birleştirme (recombination) üzerine çalışan ilk kişi olmuştur. Holland GA'nın gelişimini özetleyen kitabını 1975 yılında yayınlamıştır. Aynı yıl Kenneth De Young, GA'nın çok modlu, sürekli ve gürültülü amaç fonksiyonlarının çözümü için yüksek bir potansiyele ve güce sahip olduğunu gösteren tez çalışmasını sunmuştur.

Son yıllarda çoğu mühendislik problemlerinin çözümünde sezgisel yöntemlerin rolü önemli bir oranda artmıştır. Bunun sebepleri arasında; eğim bilgisine gerek duyulmadığından matematiksel modelin çıkarılamadığı ve eğim bilgisinin türetilemediği ya da türetilmesinin çok maliyetli olduğu problemlerde iyi sonuçlar alınabilmesi, hesaplama gücünün iyi olması, uygulamasının basit olması ve dönüştürülebilir yani bir problem için yazılmış bir sezgisel programın kolaylıkla başka problemlere uygulanabilir olması sıralanabilir (Alataş, 2007).

3.2.2.2 Metasezgisel (Metaheuristic) Algoritmalar

Yaklaşık son 30 yıldan bu yana basit sezgisel algoritmaları daha yüksek bir seviyede birleştirerek arama uzayını daha verimli ve etkin bir şekilde keşfetmeyi amaçlayan yöntemler ortaya çıkmıştır. Bu yöntemler günümüzde metasezgisel algoritmalar olarak da bilinmektedir. Yunan kökenli iki kelimenin birleşiminden meydana gelen *metaheuristic* terimi ilk olarak Glover (1986) tarafından kullanılmıştır. Meta; *ileri, önünde, daha yüksek seviye* anlamlarına gelirken, heuristic; *arama, buluşsal yol* anlamlarına gelmektedir. Metaheuristic terimi geniş bir kullanıma sahip olmadan önce, *modern heuristic* terimi de aynı anlam için kullanılmaktaydı (Reeves, 1993).

Metasezgisel teriminin araştırmacılar tarafından üzerinde hemfikir olunan bir tanımı mevcut değildir. Ancak son yıllarda çeşitli araştırmacılar bu kavramı aşağıdaki şekilde tanımlamışlardır (Blum ve Roli, 2003):

Metasezgisel, optimal sonuca yakın çözümleri verimli bir şekilde bulabilmek için arama uzayında global ve yerel arama sürecini sezgisel yöntemlerin zeki davranışlarla ve öğrenme stratejileriyle yönlendirdiği yinelemeli (iteratif) üretim işlemi olarak tanımlanır (Osman ve Laporte, 1996).

Metasezgisel yöntemler, yüksek seviyeli stratejilerdir. Temel amaç, iteratif iyileştirmenin dezavantajlarından kaçınmak ve özellikle yerel optimumlardan kaçmak için çoklu yerel aramaya izin vermektir. Bu, adımların kötüleşmesine izin vererek veya yerel arama için rassal başlangıç çözümleri üretmek yerine daha zeki yöntemlerle çözümler üreterek başarılmıştır. Metasezgisel yöntemlar arama esnasında olasılıksal kararlar almaya yatkındır. Ancak basit rassal aramaya göre temel farklılık metasezgisel yöntemlerde körü körüne bir rassallık yerine zeki ve yönlendirilmiş bir yapı olmasıdır (Stützle, 1999).

Metasezgisel, farklı yöntemler kullanarak arama uzayını keşfetmek için geliştirilmiş yüksek seviyeli stratejilerdir. Bu stratejiler *çeşitlenme* (diversification) ve *yoğunlaşma* (intensification) arasında dinamik bir denge oluşturacak şekilde seçilmelidir. Çeşitlenme terimi arama uzayını keşfetme (exploration), yoğunlaşma ise o zamana kadarki arama tecrübelerinden faydalanılması (exploitation) anlamına gelmektedir (Blum ve Roli, 2003).

Birçok metasezgisel algoritma doğadan esinlenerek geliştirilmiştir. Doğa binlerce yıldır karşılaştığı sorunlara mükemmel çözümler bulmayı başarmıştır. Bu

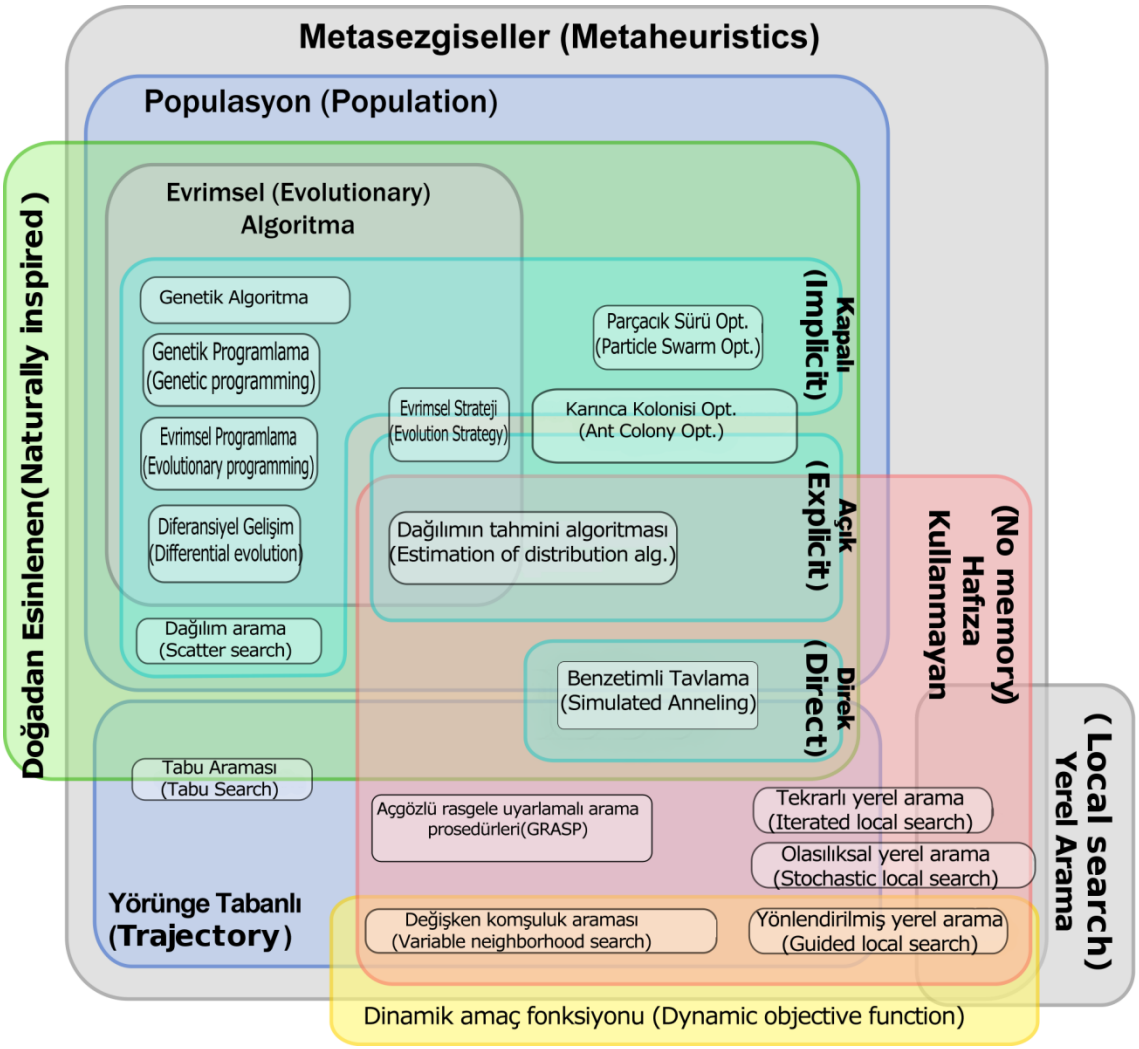
sebeple problem çözme yeteneğini doğayı taklit ederek öğrenebilir ve doğa esinli sezgisel ya da metasezgisel algoritmalar geliştirebiliriz.

Metasezgisel algoritmaların 2 temel bileşeni vardır: En iyi çözümün seçimi ve rassallık. Rassallık çözümlerin yerel bir iyiye takılmasını önler ve çözüm çeşitliliğini arttırırken, en iyinin seçimi çözümlerin optimale yakınsamasını garanti eder. Bu iki özelliğin iyi bir kombinasyonu global optimum değerin erişilmesini mümkün kılar. Metasezgisel algoritmalar birçok şekilde sınıflandırılabilir. Onları sınıflandırmanın bir yolu populasyon tabanlı (population-based) ve yörünge tabanlı (trajectory-based) olarak ayırmaktır. Örneğin populasyon tabanlı algoritmalardan GA kromozom adı verilen bir diziyi kullanırken PSO çok sayıda parçacık ya da ajanı kullanır. PSO aynı zamada ajan tabanlı algoritma olarak da bilinir. Diğer taraftan tavlama benzetimi arama uzayı boyunca tek bir ajan kullanır. Daha iyi bir yol ya da çözüm her zaman kabul edilir. Atılan adımlar arama uzayında bir yol oluşturur, oluşturulan bu yol sıfırın dışında bir olasılıkla global optimuma ulaşabilir.

Metasezgisel algoritmaların genel özellikleri aşağıdaki gibi sıralanabilir (Blum ve Roli, 2003):

- Arama süreçlerine rehberlik eden stratejilerdir.
- Arama uzayındaki en iyi ya da en iyiye yakın başka bir çözüme ulaşmayı amaçlarlar.
- Basit yerel arama algoritmalarından karmaşık öğrenme yöntemlerine kadar tüm arama süreçleri metasezgisel algoritmaların tekniklerine dahil edilebilir.
- Yerel optimum noktalara takılmamak için bazı önlemler mevcuttur.
- Probleme özgü çözüm sunmazlar. Farklı problem türleri için uyarlanabilirler.
- Farklı yöntemler kullanarak arama uzayını keşfeden yüksek seviyeli algoritmalardır.

Literatürde GA, KKA, Arı Algoritması (Bee Algorithm - BA), PSO, Tavlama Benzetimi (Simulated Annealing - SA), Yinelemeli Yerel Arama (Iterated Local Search - ILS), Harmony Search (HS), Ateşböceği Algoritması (Firefly algorithm - FA) gibi birçok metasezgisel algoritma mevcuttur (Yang, 2010). Şekil 3.5'te metasezgisel optimizasyon algoritmalarının türlerine göre sınıflandırılması verilmiştir.



Şekil 3.5 Optimizasyon algoritmalarının cinslerine göre sınıflandırılması (Weise, 2009).

3.2.2.3 Sürü zekâsı

Sürü, beraber ve işbirliği halinde yaşayan canlı gruplarını ifade eden bir kelimedir. Tabiatta birçok hayvan hayatta kalmalarını kolaylaştırmak amacıyla topluluklar halinde yaşarlar. Bu topluluk içinde işbirliği, bilgi paylaşımı ve beraberlik söz konusudur. Uzun yıllar önce araştırmacılar, doğadaki hayvan ve böcek sürülerinin zeki davranışlarını keşfettiler. Kuş sürülerinin havada süzülmesi ve farklı şekiller alması; karıncaların, kuşların, arıların birlikte yiyecek araması; balık sürülerinin beraberce yüzmesi ve kaçışması sürü davranışlarına örnek olarak verilebilir. Son yıllarda araştırmacılar sürü davranışına sahip canlıların birbirleriyle nasıl iletişim kurduğunu, hedefe nasıl ulaştıklarını anlamak için biyolojik sürüleri matematiksel olarak modellemişlerdir. Sürü davranışlarının "sürü zekâsı" adı altında optimizasyondan robot-bilime, askeri uygulamalardan iletişim ve taşımacılık

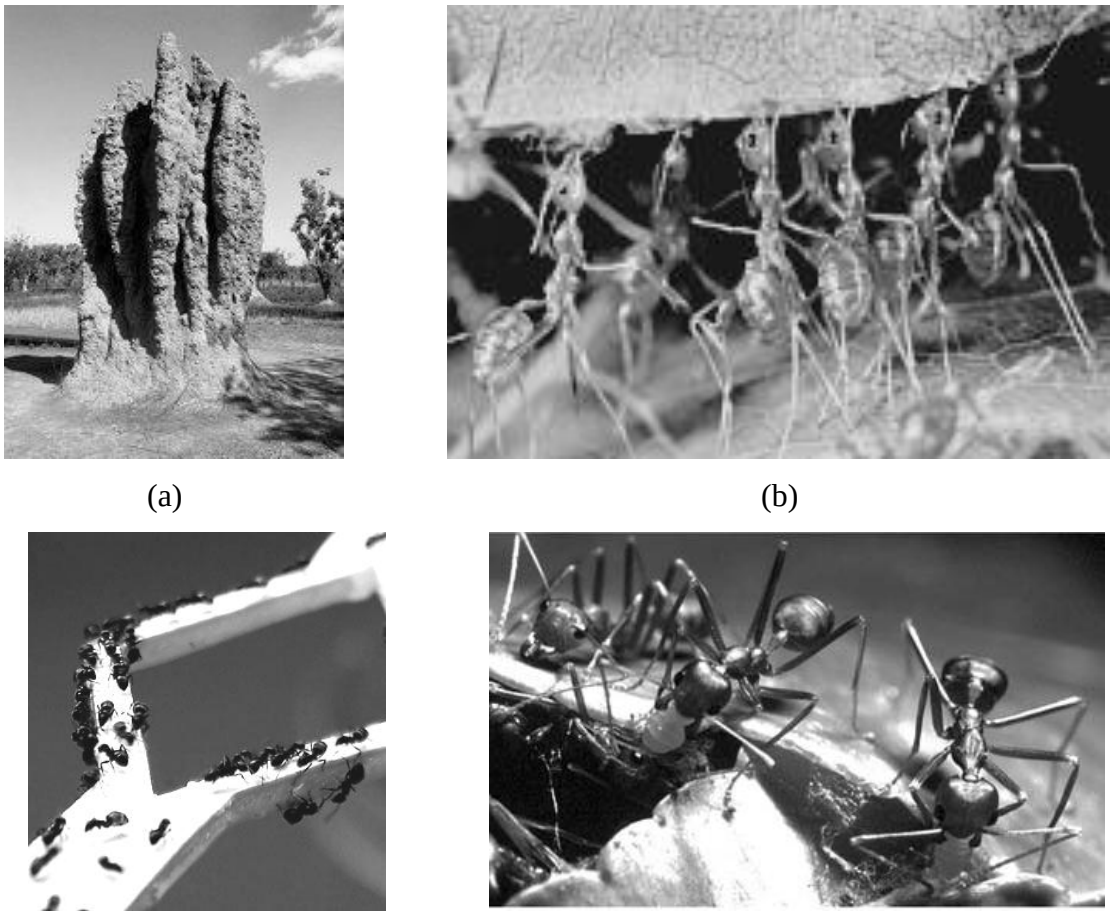
sistemlerine kadar farklı alanlarda uygulanabilir olması sürü zekâsı kavramının önemini arttıran bir faktördür. Sürü zekâsı davranışlarının dayandığı temel ilkeler aşağıdaki gibi açıklanmıştır (Aydın, 2011):

- **Stigmergy:** Grassé tarafından 1959'da ortaya atılan bir kavramdır. Termitlerin yuva inşa ederken sergiledikleri davranışları açıklayan bir terimdir. Yuva inşasında termitlerin sergiledikleri davranışlar karıncaların kendilerine bağlı değil yuvanın o anki şekline bağlıdır. Diğer bir ifadeyle bilgi yerel ortamdan gelmekte ve süregelen iş her bir karıncanın davranışında belirleyici rol oynamaktadır. Grassé'nin termitlerin aktiviteleri için kullandığı stigmergy terimi daha sonra diğer böcek türleri için de gözlemlenen ortamda değişiklik yaparak gerçekleştirilen dolaylı iletişim yöntemi anlamında kullanılmıştır (Theraulaz, E. Bonabeau, 1999).

- **Özörgütleme (self-organization):** Alt-seviyeli bileşenlerin etkileşimleriyle küresel seviyedeki yapıları oluşturan dinamik mekanizmalardır. Alt bileşenlerde gerçekleştirilen faaliyetler tamamen yerel bilgilerden faydalanılarak gerçekleştirilir küresel bilgi kullanılmaz. Bu sayede dışardan bir etki olmaksızın tüm grup aynı hedefi gerçekleştirmek için eşgüdümle hareket eder. Özörgütleme dört temel esasa dayanır:
    - **Pozitif geri-besleme**: Elverişli ve uygun davranışların oluşmasını destekleyen davranışsal kurallar. Karıncaların feromon salgılaması, arıların dans etmesi pozitif geri beslemeye örnek olarak verilebilir.
    - **Negatif geri-besleme:** Sürünün birliğini sağlamak için pozitif geri beslemeyi dengeler nitelikte çalışan kurallardır. Karıncalar tarafından salgılanan feromon maddesinin buharlaşması negatif geri beslemeye örnek olarak verilebilir.
    - **Kararsızlığın güçlendirilmesi:** Rastgelelik özörgütlemede oldukça önemli bir yere sahiptir. Rastgelelik koloninin yeni çözümler bulmasına olanak tanıdığından yaratıcılık ve yenilik açısından önem taşır.
    - **Çoklu etkileşim:** Bireyler arasında direk ya da stigmergy aracılığı ile dolaylı olarak çoklu bir etkileşim vardır. Bu etkileşimden ve bilgi

paylaşımından dolayı her birey büyük yapının oluşmasına katkı sağlar.

- **Ortaklaşa davranış çeşitleri:** Sürülerdeki ortaklaşa davranış çeşitleri eşgüdüm, dayanışma, tartışma (danışma) ve işbirliğidir**.**
    - **Eşgüdüm (coordination)**: Bir problemin çözümünde farklı birimlerin aynı amaç ve hedefi gerçekleştirmek doğrultusunda işbirliği yapması durumudur. Bu sürülerde, karşılıklı etkileşim ve bireylerin hedeflerine ulaşabilmesi için senkronize haraketlerini sağlar.
    - **Dayanışma (cooperation):** Dayanışma bireylerin tek başına yapamayacakları işleri birlikte yapma zorunluluğu olan durumlarda ortaya çıkar. Bireyler problemi çözmek için kendi güçlerini birleştirirler.
    - **Tartışma-danışma (deliberation):** Bu davranış, sürü birçok ihtimalle (fırsatla) karşılaştığında sürü için en iyi ihtimale karar vermesini sağlar. Bal arılarının nektar toplanacak kaynağı seçerken yaptıkları dans tartışma-danışmaya örnek olarak verilebilir.
    - **İşbirliği (deliberation):** Farklı aktivitelerin aynı koloni içerisinde farklı gruplar tarafından eşzamanlı olarak gerçekleştirilmesidir. Karınca kolonilerindeki aynı anda av taşıma veya yavrularla ilgilenme işbirliğine örnek olarak verilebilir.



(a) (b)

(c) (d)

Şekil 3.6 Karınca kolonilerinde (a) eşgüdüm (yuva inşası), (b) dayanışma (yem taşıma), (c) danışma (yön bulma) ve (d) işbirliği örnekleri (görev paylaşımı)

3.3. Bal Arılarının Doğası

Bal arıları fotoğrafik hafızaya, uzay çağı sensörlerine, navigasyon sistemine, sezgisel kavrama yeteneğine, yeni yuva yeri seçerken grup olarak karar verme özelliğine sahiptirler. Ayrıca yiyeceklerin saklanması, balın getirilmesi ve dağıtılması, iletişim ve yiyecek arama gibi işleri de en uygun şekilde yerine getirebilmektedirler. Bu karakteristikler, araştırmacıları arıların davranışlarını modellemeleri konusunda teşvik edici olmuştur (Akay, 2009).

Arılar koloniler halinde yaşayan sürü zekâsına sahip sosyal böceklerdir. Bir kolonide üç çeşit arı bulunmaktadır: kraliçe (queen) arı, erkek (drone) arı ve dişi olan işçi (worker) arılar.

Kraliçe Arı (Queen Bee): Arı kolonilerinin her birinde sadece bir kraliçe bulunur ve bu kraliçe arı diğer dişilere göre daha büyüktür. Kraliçe arılar kolonideki tüm arıların annesidir. Temel görevi ise yumurtlamaktır. Üreme sadece kraliçe arı vasıtasıyla olur, onun dışında diğer dişiler erkeklerle çiftleşemezler. Kraliçe arı çiftleşmede depoladığı spermlerle iki yıl boyunca yumurtlayabilir. Yumurtlamadan başka, kolonin bütünlüğünü ve kovandaki sistemin işleyişini sağlayan önemli maddeler de salgılarlar. Kraliçe arının salgıladığı bu madde sayesinde bunu diğer arılar tanır, bu koku onun etrafında toplanan tüm arı ailesine sirayet eder. Yabancı kraliçe arı ve onun ailesi bu şekilde fark edilir. Bu sayede kraliçe arı kolonin bütünlüğünü sağlamış olur (Akay, 2009; Tokmak, 2011).

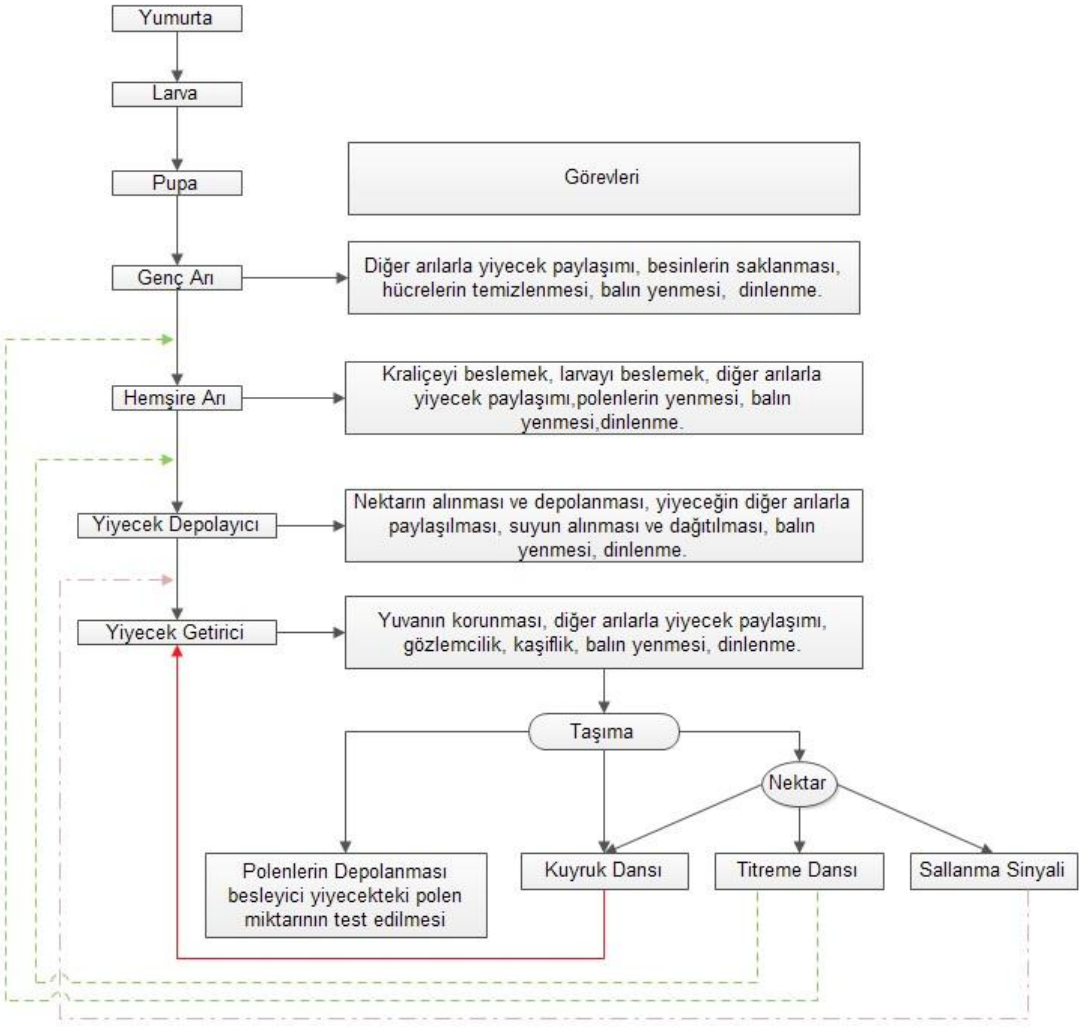
Erkek Arılar (Drones): Erkek arılar 6 aydan daha fazla yaşayamazlar. Görünüş itibariyle dişi arılardan iridirler ancak iğneleri ve besin toplayacak organları yoktur. Erkek arılar döllenmemiş yumurtalardan oluşurken kraliçe ve işçi arılar döllenmiş yumurtalardan meydana gelir. Erkek arıların kolonideki görevi kraliçe arının döllenmesidir. Kraliçe arı ile çiftleşen erkek arı ölür (Akay, 2007; Tokmak, 2011).

İşçi Arılar (Workers): Arı kolonisinin sayıca çoğunluğunu teşkil eden işçi arılar döllenmiş yumurtalardan çıkan üreme yeteneği olmayan dişilerdir. Koloninin devamını sağlayan her türlü içgüdüsel ve yapısal yeteneklere sahiptir. Kendi aralarında iş bölümü yapmak suretiyle çeşitli işleri düzen içinde yürütürler. Besin toplamak, bu besinleri saklamak, kovanın temizliği, ölü arılardan ve molozlardan kurtulmak, larvaların ve diğer arıların beslenmesi gibi birçok görevi yerine getirirler. İşçi arının yapacağı görev yaşına ve koloninin gereksinimine bağlıdır. Yaşamlarının ikinci yarısında yiyecek arayıcı (forager) olarak çalışırlar. Yazları 6 hafta, kışın ise 4-9 ay arasında yaşayabilmektedirler (Akay, 2009; Tokmak, 2011).

3.3.1. Arıların Görev Paylaşımı

Bal arısı kolonisinde çok sayıdaki farklı iş için uygun sayıda bireyin görevlendirilmesi gerekmektedir (Dornhaus, 1998). Arılar kovandaki işler konusunda uzmanlaşmıştır. Uzmanlaşmada hangi kriterlerin (yaş, hormonlar, bireysel yatkınlık vb.) etkili olduğu ise tartışılmaktadır. Kraliçe arının yumurta olarak bırakması ile bir arının gelişimi başlar. Bu yumurta zamanla larva sonra da pupaya dönüşür. Daha sonra ise

genç arı olarak yumurtadan çıkar. Yumurtadan çıkan tüm arıların ilk görevi beslenme olacaktır. Genç arılar ve hemşire arılar beslenme alanında besleme, depolama, bal ve polenlerin elde edilmesi ve dağıtılması, iletişim ve yiyecek arama gibi görevleri yerine getirirler. Herhangi bir anda bir arının gerçekleştirdiği görev o anki davranışsal rolüne, çevreden topladığı algılara ve ihtiyaca göre değişiklik göstermektedir. Arılar zamanla algıladıkları sinyallere (titreme dansı), içgüdülerine ve fizyolojik yapılarına göre farklı görevler alabilirler. Görev dağılımında ihtiyaç da önemli bir faktördür. Mesela yiyecek tükendiğinde hemşire arılar da besin arama durumuna geçebilirler. Şekil 3.7'de bal arılarının görev paylaşımı gösterilmektedir (Akay, 2009).



Şekil 3.7 Arılarda Görev Paylaşımı (Akay, 2009)

3.3.2. Gerçek Arıların Besin Arama (Foraging) Davranışı

Arıların yaşamını sürdürebilmesi için en önemli nokta besin (nektar) aramadır. Arının kovandan ayrılmasıyla başlayan arama süreci, başlangıçta rastgele yapılan yiyecek araştırmaları ile devam eder. Bulunan kaynakta yiyecek miktarının azalması neticesinde arılar yeni nektar aramaya ya da arılardan aldıkları bilgiye göre başka kaynaklara yönelmeye başlarlar. Bulunan kaynakların bilgilerinin arılarca birbirine iletilmesi ve bulunan polen, su vb. kaynakların kovana getirilmesi bu süreç içinde yapılan faaliyetlerdir (Tokmak, 2011).

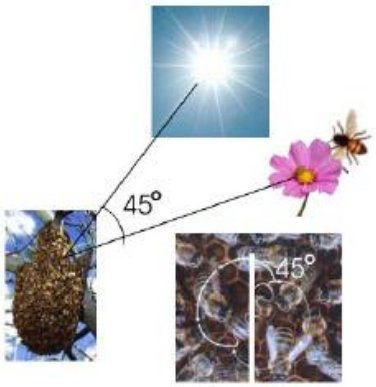
Tereshko bal arılarının besin aramasında üç temel öğe olduğunu öne sürmüştür: yiyecek kaynakları, görevi belirli işçi arılar, görevi belli olmayan işçi arılar. Bu model bir yiyecek kaynağına yönelme ve kaynağı bırakma mantığıyla çalışmaktadır (Tereshko ve Loengarov, 2005).

1) **Yiyecek Kaynakları:** Arıların besin aramak için gittikleri kaynaklardır. Bir kaynağın değeri besin çeşitliliği, yuvaya uzaklığı, nektar miktarı ve nektara ulaşma güçlüğü gibi faktörlere bağlı olmasına rağmen kolaylık olması açısından sadece nektar miktarı (kaynağın zenginliği) dikkate alınabilir (Akay, 2009).
2) **Görevi Belirli İşçi Arılar:** Daha önceden keşfedilmiş kaynaklara ait nektarın yuvaya getirilmesinden sorumlu arılardır. Aynı zamanda nektarını taşıdıkları kaynağın kalitesi ve konumu hakkındaki bilgiyi kovanda bekleyen diğer arılarla paylaşırlar. Bu paylaşım arılara özgü bir dans ile gerçekleşir. Bundan sonraki kısımlarda görevi belirli işçi arılar için *görevli arılar* ifadesi kullanılacaktır (Akay, 2009).
3) **Görevi Belirsiz İşçi Arılar:** Nektar toplayabilecekleri kaynak arayışında olan arılardır. Görevi belirsiz işçi arılar ikiye ayrılır: İçsel bir dürtüye ya da bir dış etmene bağlı olarak rastgele kaynak arayan kâşif arılar ve kovanda bekleyip görevli arıların dans yoluyla paylaştığı bilgilere göre bir kaynağa yönelen gözcü arılar. Kâşif arıların sayısı tüm koloninin %5-10'u kadardır (Seeley, 1995).

Kolektif bilginin oluşmasında en önemli faktör arılar arasındaki bilgi paylaşımıdır. Arıların yaşadığı kovan farklı bölümlere ayrılmıştır. Bu bölümler içinde bilgi paylaşımı için diğerlerine nazaran daha önemli olan bölge dans alanıdır (Tereshko ve Loengarov, 2005).

Görevli arılar taşıdıkları nektarı bırakmak için kovana geldiklerinde, gözcü arılara kendi kaynakları hakkında dans ederek bilgi verirler. Dansın şekli ve ritmi

kaynağın kovana yakınlığı, nektarın tatlılığı ve çıkarılma kolaylığı gibi faktörlere bağlı olarak değişiklik göstermektedir. Gözcü arılar çok sayıda dans izledikten sonra hangi kaynağa gideceklerine karar verirler. Daha kaliteli nektar içeren ve kaynağa yakın olan kaynakların seçilme olasılığı daha yüksek olacaktır. Gözcü arıların seçtiği kaynağa gidebilmesi için kaynağın konum bilgisine de sahip olması gerekir. Yön bilgisi Şekil 3.8'deki 8 rakamı şeklindeki dansın açı bilgisinden elde edilir. Görevli arı tarafından yapılan dans, güneş ışığıyla kaynak arasındaki açının 45^{o} olduğu bilgisini vermektedir (Akay, 2009).



Şekil 3.8 Arılarda Dans ile Konum Bilgisinin Aktarılması (Akay, 2009)

3.4. Yapay Arı Kolonisi Algoritması

Sürü zekâsına sahip arıların besin arama sürecindeki zeki davranışlarını modelleyen Karaboğa Yapay Arı Kolonisi (ABC) algoritmasını geliştirmiştir (Karaboğa, 2005). Karaboğa'nın geliştirdiği bu modelde dikkate alınması gereken ön kabuller Akay (2009) tarafından aşağıdaki gibi ifade edilmiştir:

- Her bir kaynağa sadece bir görevli arı gönderilmektedir. Bir diğer deyişle, görevli arı sayısı besin kaynağı sayısına eşittir.
- İşçi arıların sayısı aynı zamanda gözcü arıların sayısına eşittir.
- Nektarı tükenen kaynağın görevli arısı, kaşif arıya dönüşür ve yeni besin kaynağı arayışına girer.
- Nektar kaynaklarının konumları, optimizasyon problemine ait olası çözümlere, kaynaklardaki nektar miktarı ise o kaynakların çözüm kalitesine (uygunluk değerine) karşılık gelir.

- ABC algoritmasındaki amaç, en fazla nektara sahip olan kaynağın arama uzayındaki konumunu bularak problemin maksimum ya da minimum çözümünü veren parametre değerlerine erişmektir.

ABC algoritmasının adımları aşağıdaki gibidir (Akay, 2009):

1. İlk olarak kâşif arılar arama uzayında rastgele dağılarak besin arama sürecini başlatırlar.

2. Yiyecek kaynakları bulunduktan sonra kâşif arılar görevli arı haline gelirler ve kendi kaynaklarından kovana besin taşırlar. Görevli arılar kovana besinlerini boşalttıktan sonra çalıştıkları kaynakla ilgili bilgiyi kovanda bekleyen gözcü arılara iletmek için dans alanında dans ederler. Çalışılan kaynakta nektar tükenirse görevli arılar kâşif arıya dönüşür ve arama uzayında rastgele yeni kaynak arayışına girerler.

3. Gözcü arılar görevli arıların danslarını izlerler ve dansların frekansına bağlı olarak bir kaynağı seçerler. Yani dans frekansı ne kadar yüksekse bir kaynağın seçilme olasılığı da o oranda güçlüdür.

ABC algoritmasının sözde kodu Şekil 3.9'da görülmektedir.

ABC Algoritması

---

1) Başlangıçta rassal olarak yiyecek kaynağı bölgelerinin oluşturulması
2) **REPEAT**
- Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi
- Gözcü arılar için nektar miktarına göre kaynakların uygunluk değerlerinin hesaplanması
- Gözcü arıların uygunluk değerlerine göre kaynak seçimi yapması
- Kaynağı bırakma kriteri: limit ve kaşif arı üretimi

UNTIL Maksimum Çevrim Sayısı

Şekil 3.9 ABC algoritmasının sözde kodu

Başlangıçta rassal olarak yiyecek kaynaklarının oluşturulması:

ABC algoritmasında, besin kaynağının başlangıçtaki konumu optimizasyon problemi için olası bir çözümü ifade etmektedir. Kaynağın konumu parametre değerlerinin alt ve üst sınırları aralığında rassal olarak belirlenir. Yiyecek kaynaklarının başlangıç konumları Eşitlik 3.6'ya göre belirlenir (Akay, 2009).

$$x_{ij} = x_j{}^{min} + rand(0,1)(x_j{}^{max} - x_j{}^{min}) \quad (3.6)$$

Burada i = 1…KS, j = 1...B ve KS kaynak sayısını, B boyut sayısını, *rand(0,1)* ise [0,1] aralığında rassal bir değeri ifade etmektedir.

Algoritma başlatıldığında her kaynağın geliştirilememe sayısını ifade eden başarısızlık (failure) sayacı sıfırlanır. Bu aşamadan sonra görevli arılar, gözcü arılar ve kâşif arılar vasıtasıyla arama uzayındaki en zengin besin kaynağına sahip yer bulunmaya çalışılır. Bu süreç için farklı durdurma kriterleri düşünülmüştür (Karaboğa, 2008; Akay, 2009; Teodorovic ve Nikolic, 2013).

- Maksimum çevrim sayısı (MÇS),
- Uygunluk fonksiyonunun maksimum geliştirilememe sayısı,
- İzin verilen maksimum CPU süresi,
- Kabul edilebilir bir hata miktarı (ε).

Burada dikkat edilmesi gereken nokta, çözülmeye çalışılan problemin optimum noktası biliniyorsa durdurma kriteri olarak ε değeri kullanılabilir aksi durumlarda durdurma kriteri olarak MÇS, uygunluk fonksiyonunun maksimum geliştirilememe sayısı, izin verilen maksimum CPU süresi değerini kullanmak gerekir.

Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi

Her görevli arı kendi kaynağına gittikten sonra kendi kaynağının çevresinde daha iyi nektara sahip bir kaynak olup olmadığını araştırmak ister. Görevli arı kendi kaynağının komşuluğundaki yeni kaynak arayışını Eşitlik 3.7'ye göre yapar.

$$v_{ij} = x_{ij} + rand(-1,1)(x_{ij} - x_{kj}) \quad (3.7)$$

x_i ile gösterilen kaynağın rassal olarak belirlenen j. boyutu (parametresi) değiştirilerek x_i'nin kendi komşuluğundaki v_i kaynağı bulunur.

- *j*, [1, B] aralığında rastgele belirlenen bir tamsayıdır.
- *k*, [1, KS] aralığında rastgele seçilen komşu kaynağı ifade eden bir tamsayıdır (i ≠ j).
- x_i. kaynağın *j*. boyutu ile komşu olarak seçilen x_k. kaynağın *j*. boyutunun farkları, [-1,1] aralığında rastgele değer alan *rand(-1,1)* katsayısı ile çarpılır ve mevcut kaynağın (x_i) *j*. boyutuna eklenir (Akay, 2009).

İterasyonlar ilerledikçe kaynaklar birbirine yaklaşacak ve x_{ij} - x_{kj} farkı da adaptif olarak azalacaktır. Bu da bölgesel optimal noktaya yaklaşıldığı anlamına gelecektir. Eşitlik 3.7'nin uygulanması sonucunda elde edilen v_{ij}'nin j. parametresinin arama uzayında j. parametreye ait olan alt veya üst sınırı aşması durumunda Eşitlik 3.8 uygulanarak j. parametre yeniden arama uzayına çekilmiş olur.

$$v_{ij} = \begin{cases} {x_j}^{min}, v_{ij} < {x_j}^{min} \\ v_{ij}, {x_j}^{min} \leq v_{ij} \leq {x_j}^{max} \\ {x_j}^{max}, v_{ij} < {x_j}^{max} \end{cases} \quad (3.8)$$

Eşitlik 3.8 kullanılarak olası sınır ihlalleri ortadan kaldırılır ve sınırlar dahilinde v_i kaynağı elde edilir. Elde edilen v_i kaynağının kalitesine göre bir uygunluk değeri ($fitness_i$) Eşitlik 3.9'a göre hesaplanır (Akay, 2009).

$$fitness_i = \begin{cases} \frac{1}{1+f_i}, f_i \geq 0 \\ 1 + abs(fi), f_i < 0 \end{cases} \quad (3.9)$$

Burada f_i (amaç fonksiyonu), v_i kaynağının sahip olduğu nektar miktarını simgelemektedir. x_i ile v_i kaynakları arasında sahip oldukları nektar miktarlarına yani maliyet değerlerine göre bir açgözlü (greedy) arama işlemi uygulanır. Sonuçta v_i kaynağı daha çok nektara sahipse görevli arı x_i kaynağını hafızasından siler ve yeni kaynak olarak v_i'yi hafızasına alır. Kaynağın yeri değiştiği için geliştirilememe (failure) değeri de sıfırlanır. x_i kaynağının daha çok nektara sahip olduğu durumda ise görevli arı hafızasında x_i'nin konumunu tutmaya devam eder ve x_i kaynağına ait failure sayacı 1 arttırılır (Akay, 2009).

Gözcü arılar için nektar miktarına göre kaynakların uygunluk değerlerinin hesaplanması

Görevli arılar arama uzayında yeni bir çevrimi tamamladıktan sonra kovana dönerek buldukları kaynakların nektar miktarları ile ilgili bilgiyi gözcü arılarla paylaşırlar. Gözcü arılar dans yoluyla aktarılan nektar bilgisine göre olasılıksal olarak bir kaynağı seçerler. Nektar miktarıyla orantılı olan uygunluk değerine bağlı seçim işlemi, rulet tekerleği, sıralamaya dayalı, stokastik örnekleme, turnuva yöntemi gibi yöntemlerden biriyle yapılabilir. ABC algoritmasında bu seçim işlemi rulet tekerleği ile

yapılmaktadır. İhtimallerin tamamını bir bütün pasta olarak düşünürsek, daha yüksek uygunluk değerine sahip olan kaynak daha büyük bir dilime sahip olacak ve daha büyük dilime sahip olan kaynağın seçilme olasılığı da o oranda artacaktır. Yani bir kaynağın uygunluk değerinin tüm kaynakların uygunluk değerleri toplamına oranı, o kaynağın seçilme ihtimalini vermektedir (Eşitlik 3.10) (Akay, 2009).

$$p_i = \frac{fitness_i}{\sum_{i=1}^{KS} fitness_i} \tag{3.10}$$

Eşitlik 3.10'a göre bir kaynağın nektar miktarı arttıkça gözcü arılar tarafından seçilme ihtimali de o oranda artacaktır. Bu özellik ABC'nin pozitif geri besleme özelliğine karşılık gelir (Akay, 2009).

Gözcü arıların uygunluk değerlerine göre kaynak seçimi yapması

Gözcü arıların kaynak seçmesi için hesaplanan p_i değerleri [0,1] aralığında rastgele olarak üretilen sayılarla karşılaştırılır. Hesaplanan p_i olasılık değeri rassal olarak belirlenen sayıdan büyükse Eşitlik 3.7 kullanılarak bu kaynağın çevresinde yeni bir kaynak belirlenir. Yeni belirlenen kaynağın nektar miktarı yani uygunluk değeri hesaplanır. Ardından yeni kaynak ve mevcut kaynak arasında daha iyi uygunluk değerine sahip olanın seçildiği açgözlü seçim işlemi uygulanır. Yeni kaynak daha iyiyse eski kaynak silinir ve yeni kaynak hafızaya alınır. Bu durumda failure değeri de sıfırlanır. Eski kaynak daha iyiyse bu çözüm yerini korur ve failure değeri bir arttırılır. Bu yöntemle tüm gözcü arılar yiyecek kaynaklarına gönderilir (Akay, 2009).

Kaynağı bırakma kriteri: limit ve kâşif arı üretme

Tüm arılar için bir çevrim tamamlandıktan sonra tüm kaynakların *geliştirilememe sayacı* (failure) değerlerine bakılır. Failure değeri, bir kaynağın kendi bölgesindeki optimum değere ulaştığını yani o kaynağın yeterli miktarda aranıp tükendiğini anlamamızı sağlayan bir parametredir. Bir kaynağın geliştirilememe sayacı, *limit* adı verilen ve maksimum başarısız olma hakkını ifade eden parametreden daha büyükse o kaynak artık doyuma ulaşmış ve yeteri kadar aranmıştır. Bu yüzden o kaynaktaki görevli arı kâşif arı durumuna geçer ve arama uzayında rastgele bir nokta belirler (Eşitlik 3.6). Her bir çevrimde yalnız bir kâşif arının çıkmasına izin verilir.

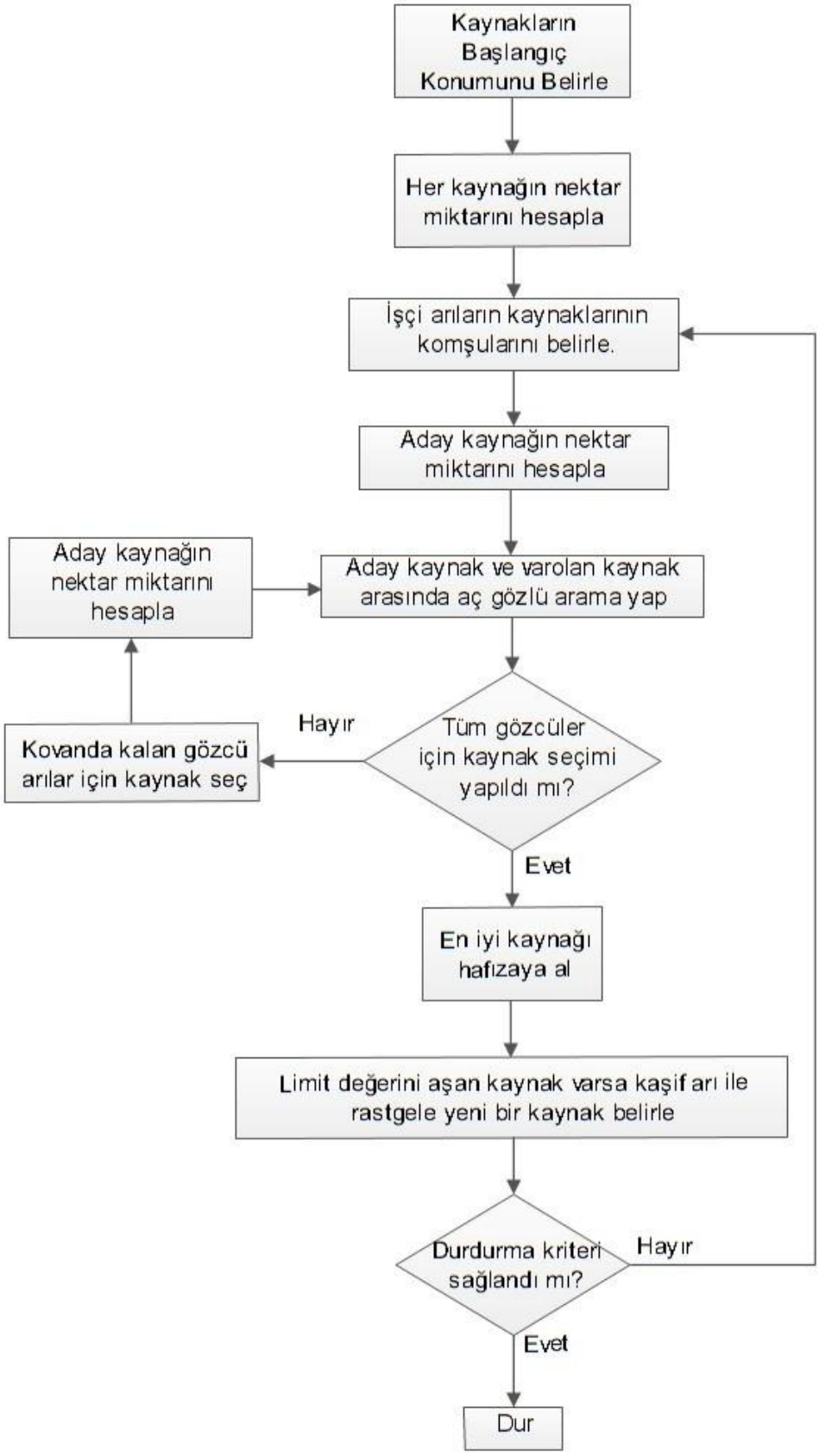
ABC algoritmasının tüm adımları arasındaki ilişki Şekil 3.10'daki akış diyagramı ile ifade edilmiştir (Akay, 2009).

Limit değeri

ABC algoritmasında her bir çevrimde yalnız bir adet boyut değiştiğinden tüm boyutların değişmesi için en az B kadar çevrim gerekir. KS tane arının bir kaynak üzerinde değişiklik yaptığı düşünüldüğünde genel olarak limit değeri Eşitlik 3.11'deki formüle göre belirlenmektedir.

$$Limit = KS \text{ x } B \tag{3.11}$$

Sphere, Ackley, Step gibi düz yüzeylere sahip fonksiyonlarda kaynakları iyileştirmek çok zor olmadığı için kaynaklar sürekli daha iyi noktalara taşınacak ve failure değerleri de sıklıkla sıfırlanacaktır. Bu sebeple yukardaki gibi fonksiyonlarda limit değerinin yüksek olması fonksiyonun optimizasyonu üzerinde olumlu etki yapmamaktadır. Algoritmada küçük limit değerleri kullanılması fazla kâşif arının üretilmesine sebep olur, yüksek limit değerleri ise az sayıda kâşif arının üretilmesine sebep olur ve keşif yeteneğini kısıtlar. Bu nedenle arama uzayının zorlaştığı fonksiyonlarda Eşitlik 3.11'de önerilen limit değerinin daha düşük olması gerekmektedir (Akay, 2009). Böylece küçük limit değerleri daha fazla kâşif arının üretilmesine neden olacağından keşif yeteneğini artıracaktır. Bunun da optimum çözüme ulaşma ihtimalini artıracağı düşünülmektedir.



Şekil 3.10 ABC Algoritmasının Akış Diyagramı (Akay, 2009).

Seleksiyon mekanizmaları

ABC algoritmasında 4 farklı seleksiyon işlemi vardır (Akay, 2009). Bunlar;

- ***Rastgele Seleksiyon Süreci:*** Kâşif arıların Eşitlik 3.6'ya göre arama uzayında rastgele kaynak belirlediği yöntemdir.
- ***Bölgesel Olasılık Tabanlı Seleksiyon Süreci :*** Eşitlik 3.7 ile görevli ve gözcü arıların mevcut kaynağın çevresinde yerel arama yapmasını sağlayan bir yöntemdir.
- ***Aç Gözlü (Greedy) Seleksiyon Süreci:*** İşçi ve gözcü arıların daha iyi olan kaynağı seçmek için kullandıkları yöntemdir.
- ***Global Olasılık Temelli Seleksiyon Süreci:*** Eşitlik 3.10 ile kaynakların nektar miktarlarına paralel olarak uygunluk değerlerinin (p_i) hesaplandığı yöntemdir.

Bu seleksiyon yöntemleri sayesinde ABC algoritması iyi bir global arama ve yerel arama yapabilmektedir.

ABC'nin temel özellikleri

Akay (2009), ABC algoritmasının temel özelliklerini aşağıdaki gibi ifade etmiştir:

- Basit ve esnektir.
- Gerçek arıların davranışlarını oldukça iyi taklit eder.
- Sürü zekâsına sahiptir.
- Sürekli problemler için geliştirilmiş olmasına rağmen ayrık problemeler için de kullanılabilir.
- Az sayıda kontrol parametresine sahiptir.
- Görevli ve gözcü arılar tarafından yapılan bölgesel arama ve kâşif arılar tarafından yapılan global arama uyumlu bir şekilde yürütülerek arama uzayı detaylı bir şekilde incelenebilmektedir.

4. ABC ALGORİTMASI ÜZERİNDE ÖNERİLEN DÜZENLEMELER

4.1. Adaptive Dimension Limit ABC (ADL-ABC) Algoritması

Bu bölümde ağırlıklı olarak ABC algoritmasının eksik görülen noktalarından bahsedilecek ve bu eksiklikleri iyileştirmek için önerilen ADL-ABC algoritması anlatılacaktır.

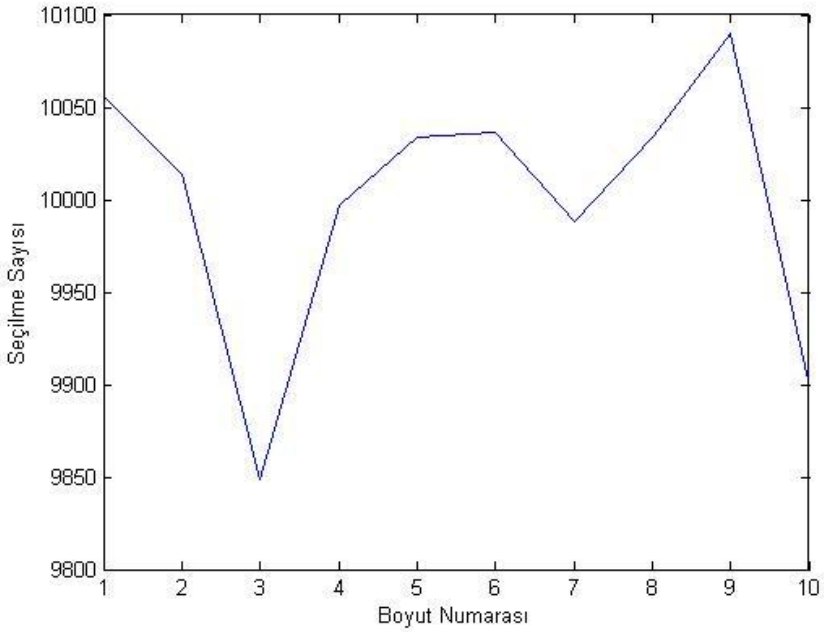
4.1.1 ADL-ABC algoritmasında komşuluk ve boyut seçimi

ABC algoritmasında görevli ve gözcü arılar yerel arama yapan unsurlardır. Kaynakların konumları çözüm için atanmış boyut değerlerinden meydana gelmektedir. Bu boyut değerleri Eşitlik 3.7'yi kullanarak aday kaynağı oluştururlar.

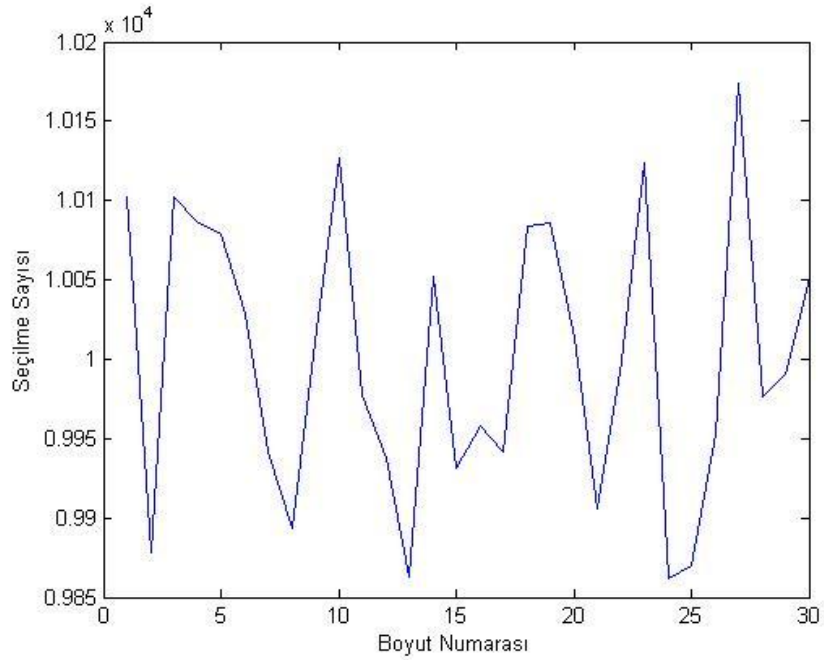
Gözcü arılara baktığımızda olasılık değerine (p_i) göre seçilen komşunun rassal olarak belirlenen bir boyutu üzerinde Eşitlik 3.7 uygulanmaktadır. Yani komşu seçiminde p_i değerinin kullanılmasıyla arıların önceki tecrübelerinden faydalanılmış, rassal boyut belirlenmesi ile de stokastik etki korunmuştur.

İşçi arılara baktığımızda ise rassal olarak belirlenen komşunun yine rassal olarak belirlenen bir boyutu üzerinde Eşitlik 3.7 uygulanmaktadır. Yani işçi arıların komşu seçiminde sadece rassal etki sözkonusudur.

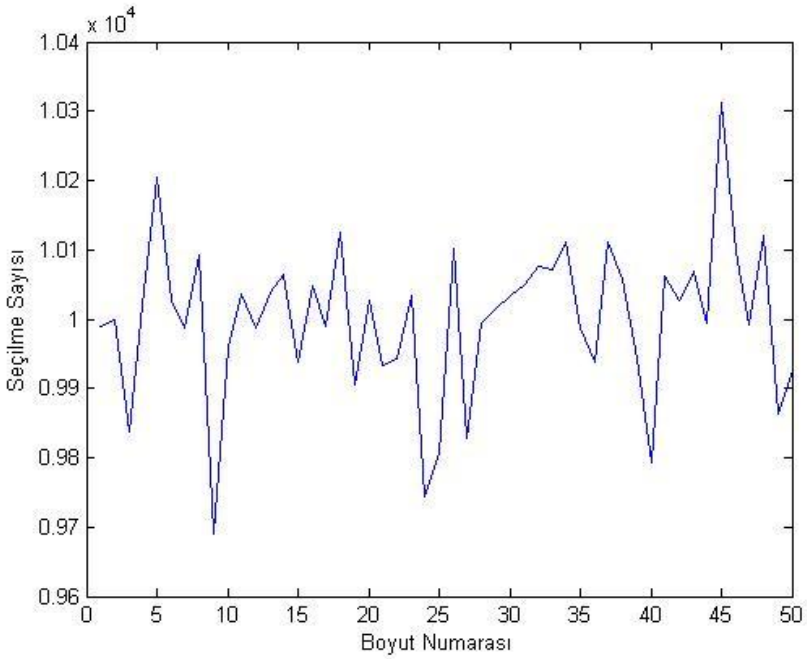
Şekil 4.1, Şekil 4.2, Şekil 4.3 sırasıyla 10, 30, 50 boyut için gidilecek boyutların rassal olarak seçilmesi durumunda her bir boyuta verilen seçilme sayısına ait bir örnek görülmektedir. Toplamda B*10.000 defa rassal sayı üreterek (Suganthan ve ark., 2005) elde edilen şekillerden de görüldüğü gibi mevcut ABC algoritmasındaki boyut seçimi tamamen rassal bir özellik göstermektedir.



Şekil 4.1 10 Boyut için Rassal Parametre Üretimi



Şekil 4.2 30 Boyut için Rassal Parametre Üretimi



Şekil 4.3 50 Boyut için Rassal Parametre Üretimi

Bu rassal yapıyı ortadan kaldırarak işçi arı mekanizmasında her bir boyutta eşit sayıda güncelleme yapılmasının optimizasyon sürecini olumlu etkileyeceği düşünülmektedir.

Bu sebeple ADL-ABC algoritmasında işçi arıların komşu seçimi yine rassal olarak yapılırken boyut seçiminde tüm boyutlara eşit sayıda hak tanıyacak *boyut listesi* adında bir yapı önerilmektedir. Bu yapı karınca kolonisi algoritmasındaki tabu liste benzetilebilir. Her bir çevrimde nektar kaynaklarının hangi boyutunda güncelleme yapılacağı boyut listesine göre belirlenir. Boyut listesi oluşumu Şekil 4.4'teki sözde kodda açıklanmaktadır.

Boyut Listesi Oluşumu

---

1) For i=1 to KS //toplam kaynak kayısı
2) BL(i,:) = Randdizi(1,B) // 1'den B'ye kadar olan sayıları rassal olarak yerleştirir.
3) End for

Şekil 4.4 Boyut Listesi Oluşumu

Burada i = 1…KS, j = 1…B ve KS kaynak sayısını, j ise boyut (B) sayısını ifade etmektedir. Boyut listesi KSxB boyutlarında 2 boyutlu bir matristir. Her satırda belli bir işçi arının kaynak güncellemesinde gideceği boyutların listesi bulunmaktadır. Her satırda rassal olarak yerleştirilmek üzere tüm boyutlar bir kez bulunmaktadır. Boyut

listesinin birinci sütununda bulunan boyutlar işçi arılar tarafından ilk çevrimde, ikinci sütununda bulunanlar ikinci çevrimde ve B. sütununda bulunanlar B. çevrimde güncellenecektir. B. çevrim sonunda listedeki tüm boyutlar ziyaret edildiğinden Şekil 4.4'e göre yeni bir liste oluşturulur. Bu döngü arama süreci sona erene kadar devam eder. 5 adet işçi arı ile 10 boyutlu bir problemi çözmek için oluşturulmuş örnek bir boyut listesi Şekil 4.5'te gösterilmiştir.

	Güncellenecek boyutların rassal olarak belirlenmiş sırası									
İşçi$_1$	7	4	8	3	1	6	10	2	9	5
İşçi$_2$	3	5	1	9	4	7	2	8	10	6
İşçi$_3$	8	6	2	4	5	9	7	10	1	3
İşçi$_4$	5	1	4	10	8	3	9	2	7	6
İşçi$_5$	9	3	7	5	10	8	2	6	4	1

Şekil 4.5 10 boyutlu bir problemde 5 adet işçi arı için oluşturulmuş bir örnek boyut listesi

4.1.2. ADL-ABC algoritmasında geliştirilememe sayacı

Standart ABC algoritmasında, Eşitlik 3.7 kullanılarak yeni bir aday kaynak oluşturulur. Oluşturulan aday kaynağın uygunluk değeri mevcut kaynağın uygunluk değerinden kötü (küçük) ise tüm kaynağın geliştirilememe sayısını ifade eden failure değeri 1 arttırılır, iyi ise failure değeri sıfırlanır. Mevcut kaynağın uygunluk değerinin değişmesine neden olan tek bir boyut güncellemesi olmasına rağmen, ödüllendirme (failure değerini sıfırlama) ya da ceza (failure değerini bir arttırma) tüm boyutların ortak kullandığı failure değerini etkilemektedir. Böylece bir boyutta yapılan değişiklik sonucu oluşacak iyileşme ya da kötüleşme durumunda oluşacak etkiden (ödül/ceza) tüm boyutlar eşit oranda etkilenecektir.

Bu durumdan kaynaklanan olumsuzluğu önlemek amacıyla, ADL-ABC'de her kaynağın her bir boyutu için ayrı bir geliştirilememe sayacı (dimension failure) önerilmiştir. Hangi boyut yeni kaynağın uygunluk değerine etki etmişse o boyutun geliştirilememe sayacı değeri etkilenir (Yani başarısızlık durumunda 1 artırılır, başarı durumunda ise sıfırlanır).

4.1.3. ADL-ABC algoritmasında limit değeri

Standart ABC algoritmasında limit değerinin Eşitlik 3.11'e göre belirlendiği daha önce belirtilmişti. Eşitlik 3.11'e göre elde edilen limit değeri, çözümü nispeten kolay olan bazı fonksiyonlar için gereğinden fazla kâşif arı üretilmesine sebep olurken, daha karmaşık fonksiyonlar için ise yeterli miktardan daha az sayıda kâşif arı üretimine neden olmaktadır. Böylece kâşif arı sayısı nispeten yetersiz kalmaktadır. ABC algoritmasındaki bu sorun ADL-ABC algoritmasında limit değerinin adaptif hale getirilmesiyle çözülmeye çalışılmıştır. Bu düzenleme ile her boyuta eşit sayıda güncellenme hakkı verilerek boyut seçiminde rassallık etkisinin azaltılması amaçlanmıştır. Bu amaca ulaşmak için, ADL-ABC'de *maksimum limit (MakLim)* ve *minimum limit (MinLim)* olmak üzere iki farklı limit değeri önerilmektedir. Limit değeri başlangıçta MakLim'e eşitken çevrimler ilerledikçe MinLim'e doğru bir azalma gerçekleşir ve MinLim seviyesinde durur.

Standart ABC algoritmasında bir kaynak için belirlenen limit değeri KSxB değerine eşittir (Eşitlik 3.11). Önerilen ADL-ABC algoritmasında Minlim değeri limit değerinin boyut sayısına bölünmesiyle hesaplanmıştır (Eşitlik 4.1). MakLim değeri ise bir arı için toplam çevrim sayısının yani iterasyon sayısının boyut sayısına bölünmesi ile elde edilir (Eşitlik 4.2).

$$\begin{aligned} MinLim &= \frac{Limit}{B} \\ MinLim &= (KS \times B)/B \\ MinLim &= KS \end{aligned} \tag{4.1}$$

$$MakLim = MÇS/B \tag{4.2}$$

ADL-ABC algoritmasında her çevrimde limit değeri o anki çevrim sayısına bağlı olarak adaptif bir şekilde değişmektedir. Anlık limit değeri Eşitlik 4.3 uygulanarak hesaplanmaktadır.

$$Limit = MakLim - int\left[\frac{MakLim - MinLim}{MÇS}\right] \times iter \tag{4.3}$$

Eşitlik 4.3'teki *int* ifadesi elde edilen değerin tam kısmının hesaplamaya dahil edileceğini göstermektedir. *iter*, o anki iterasyon sayısını, *MÇS*, maksimum iterasyon sayısını temsil etmektedir. ADL-ABC algoritmasında limit değerine ulaşan bir boyut standart ABC'de olduğu gibi arama uzayında rassal bir arama yapmaz. Bu boyuta, o ana

kadar ki çevrimler içinde en iyi değere sahip olan (Global Best) kaynağın aynı boyut için elde ettiği değeri atanır. Örneğin i. kaynağın 23. boyutu limit değerini aşarsa, global best durumdaki kaynağın 23. boyutu için elde edilen değeri i. kaynağın 23. boyutuna atanır.

4.1.4. ADL-ABC algoritmasında kaynağı bırakma kriteri ve kâşif arı üretimi

ADL-ABC algoritmasında bir boyutun geliştirilememe sayacının limit değerini aşması kâşif arı üretimi için yeterli sebep değildir. Bir kaynağın limit değerini aşan bir boyutuna, arama uzayındaki en iyi kaynağın ilgili boyutunun değeri atandıktan sonra bu kaynak ile en iyi kaynağın konumları karşılaştırılır. Bu iki kaynak bire bir aynı konumdaysa en iyi nektara sahip olan kaynak diğer kaynağı tam olarak kendi konumuna çekmiş demektir. En iyi kaynak, arama uzayındaki en iyi değeri (optimum) bulmuş olabileceği gibi yerel bir minimuma da takılmış olabilir. Bu sebeple arama uzayında en iyi kaynak ile aynı konumu paylaşan diğer kaynaklardaki işçi arılar kaşif arı halini alır ve rassal olarak yeni bir kaynak arayışına giderler. Yeni kaynakların geliştirilememe sayacı değerleri sıfırlanır. Bir çevrimde birden fazla kâşif arı üretilebilir.

Önerilen ADL-ABC algoritmasında, x_i ve v_i kaynakları arasında açgözlü aramaya göre seçim yaparken uygunluk değeri yerine, Banharnsakun ve ark.,(2011)'nın önerdiği şekilde amaç değerine göre karşılaştırma yöntemi kullanılmıştır. Standart ABC algoritmasında önerilen ve Banharnsakun ve ark.,(2011)'nın önerdiği aç gözlü arama yöntemleri Şekil 4.6'da gösterilmiştir.

Fitness değerine göre açgözlü arama	Amaç değerine göre açgözlü arama
If (fitness(f(v_i))>fitness(x_i)) *v_i'yi yeni kaynak yap.* *Else* *x_i'yi tutmaya devam et.*	*If (Minimizasyon Amaçlanıyorsa)* *If (f(x_i))>f(v_i))* *v_i'yi yeni kaynak yap.* *Else* *x_i'yi tutmaya devam et.* *Else If (Maksimizasyon Amaçlanıyorsa)* *If (f(v_i))>f(x_i))* *v_i'yi yeni kaynak yap.* *Else* *x_i'yi tutmaya devam et.*

Şekil 4.6 Fitness değerine göre ve amaç değerine göre açgözlü arama yöntemleri

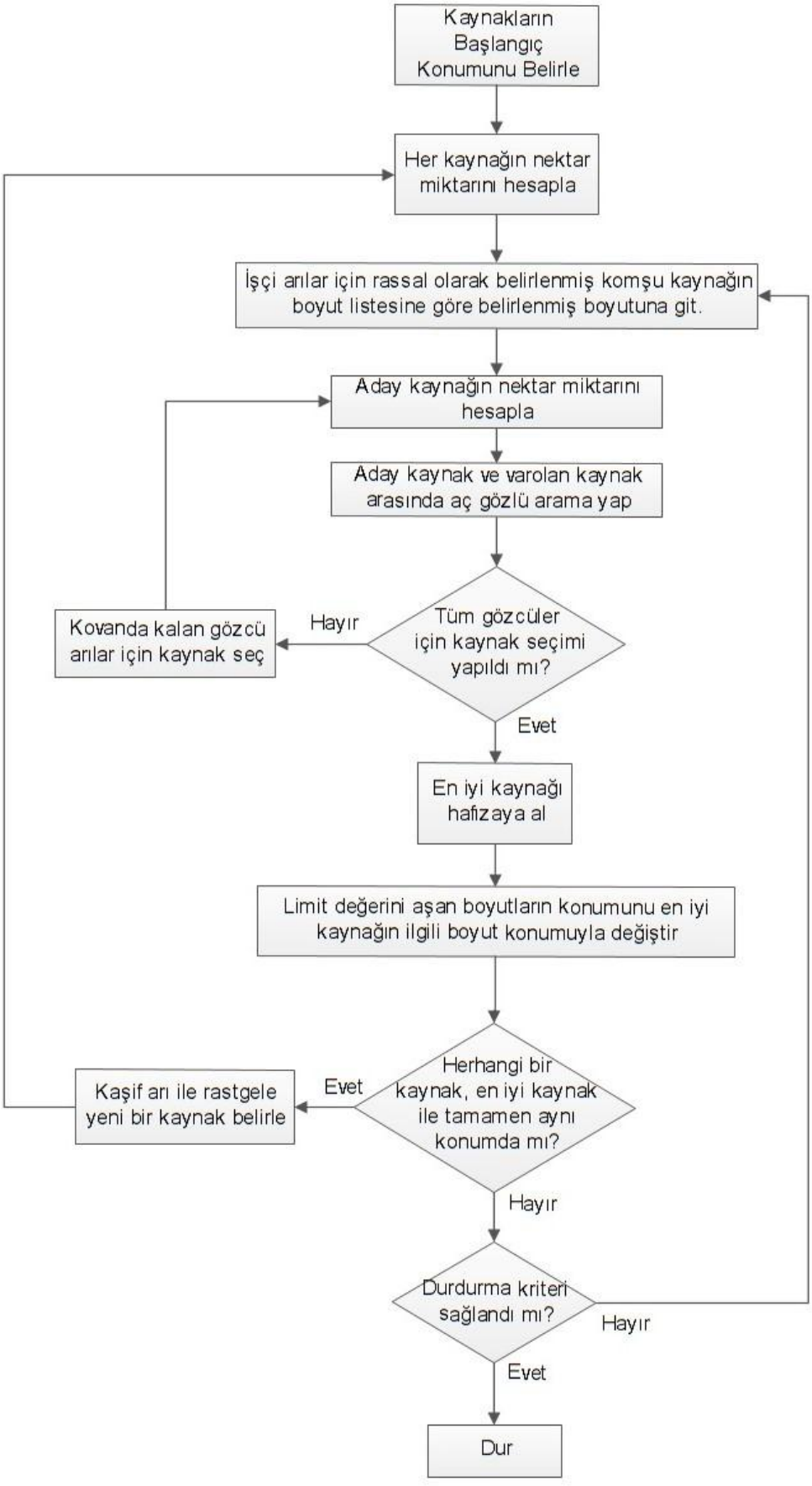
Yukarıda ADL-ABC algoritmasının temel adımları üzerinde durulmuş, hangi noktalarda ABC algoritmasından ayrıldığı belirtilmiştir. Bu esaslara bağlı olarak oluşturulmuş ADL-ABC sözde kodu Şekil 4.7'de ve akış diyagramı ise Şekil 4.8'de görülmektedir.

ADL-ABC Algoritması

1) Başlangıçta Eşitlik 3.6'e göre rassal olarak yiyecek kaynağı bölgelerinin oluşturulması
2) Şekil 4.4'e göre *Boyut Listesi* oluşturulması
3) **REPEAT**
- İşçi arıların boyut listesine göre yiyecek kaynağı bölgelerine gönderilmesi
- Nektar miktarı kullanılarak Eşitlik 3.9'a göre kaynakların uygunluk değerlerinin hesaplanması
- Gözcü arıların uygunluk değerlerine göre kaynak seçimi yapması
- Boyutu bırakma kriterine ulaşınca (boyut limitinin aşımı) boyut değerini Gbest değerine göre güncelle
- Kaynağı bırakma kriteri (Global en iyi kaynak): Eşitlik 3.6'e göre kaşif arı üretimi
- Boyut listesi tükenmişse Şekil 4.4'e göre yenile

UNTIL Maksimum Çevrim Sayısı

Şekil 4.7 ADL-ABC Algoritması sözde kodu



Şekil 4.8 ADL-ABC Algoritması Akış Diyagramı

4.2. Accelerated ABC Algoritması (A-ABC)

Bu bölümde ağırlıklı olarak ABC algoritmasının eksik görülen noktalarından bahsedilecek ve bu eksiklikleri iyileştirmek için önerilen A-ABC algoritması anlatılacaktır.

4.2.1 Modification Rate (MR)

A-ABC algoritmasında Akay ve Karaboğa (2012) tarafından önerilen *MR* (Modification Rate) parametresinin adaptif hesaplanmasına ait yeni bir yaklaşım önerilmektedir. *MR* parametresi bir modifikasyonda birden fazla boyutun güncellenmesine imkan sağlamaktadır. *MR* parametresi ile optimal çözüme standart ABC algoritmasından daha hızlı bir yakınsama amaçlanmıştır (Akay ve Karaboğa, 2012).

A-ABC algoritmasında *MR* parametresi için sabit bir değer kullanmak yerine kaynakların kalitesine göre *MR* değerinin adaptif olarak belirlendiği bir yaklaşım önerilmektedir.

ABC algoritmasında x_i kaynağının etrafında yerel arama yapmak için rastgele seçilen bir komşu kaynak ile x_i'nin sadece bir boyutu değiştirilir. Her modifikasyonda tek bir boyut üzerinde güncelleme yapılıyor olması arama uzayını daha detaylı bir şekilde inceleme olanağı sağlaması açısından olumlu bir etki gösterirken, optimum değere yakınsama (convergence) hızının yavaşlamasına neden olması açısından da olumsuz bir etki gösterir.

Yakınsama hızını arttırmak için literatürde çeşitli çalışmalar bulunmaktadır. Akay ve Karaboğa (2012), standart ABC algoritmasına modification rate (*MR*) adı verilen bir parametre eklemişlerdir. Başlangıçta, tüm besin kaynaklarının tüm boyutları için [0,1] aralığında rassal bir değer üretilir. Rassal olarak üretilen bu değerler *MR* değerinden küçük ise Eşitlik 4.4 kullanılarak ilgili boyutlar güncellenir. Aksi durumda boyutlar güncellenmez. Bu düzenleme Şekil 4.9'da gösterilmiştir. Akay ve Karaboğa (2012) tarafından önerilen bu düzenleme standart ABC algoritmasının performansını arttıran bir etki göstermiştir. Ancak yapılan literatür araştırmasında bu düzenlemeye ilişkin farklı formların denenmediği gözlenmiştir.

$$v_{ij} = x_{ij} + rand(-1,1)\left(x_{ij} - x_{kj}\right), if\ R_{ij} < MR \quad (4.4)$$

X_i	$Boyut_1$	$Boyut_2$	$Boyut_3$	$Boyut_4$	…….	$Boyut_m$
$R_{ij}[0,1]$	0.337	0.647	0.108	0.814	…….	…….
MR	0.4	0.4	0.4	0.4	0.4	0.4
Durum ($MR>R_{ij}$)	√	⊗	√	⊗		

Şekil 4.9 Akay ve Karaboğa'nın (2012) orijinal MR modifikasyonu

Bu çalışmada; MR değerinin belirlenmesinde, besin kaynakları kalite değerlerine göre gruplandırılarak her bir grup için farklı MR değerleri kullanılmıştır. Besin kaynaklarının kalite değerleri amaç fonksiyonundan gelen değerler kullanılarak belirlenmektedir. Besin kaynaklarından en iyi değere sahip 1/5'lik grup 0.5, ikinci 1/5'lik grup 0.4, üçüncü grup 0.3, dördüncü grup 0.2 ve en kötü fitness değerine sahip son 1/5'lik grup 0.1 MR değerini alır. Arama uzayında en iyi konumda bulunan 1/5'lik besin kaynağı grubu yüksek MR değeri sayesinde yerel aramaya daha fazla yönelerek optimal çözüme yakınsama hızını arttırması, konumları daha kötü noktalarda bulunan besin kaynakları için MR değeri düşürülerek global aramayı sürdürmesi amaçlanmıştır. Bu sebeple en kötü konumda bulunan 1/5'lik besin kaynağı grubunun MR değeri 0.1 olarak kullanılmıştır.

4.2.2 Step Size (SS) Modifikasyonu

Gao ve Liu (2012) ve Zhu ve Kwong'un (2010) yaptığı çalışmalar ABC algoritmasının global aramada elde ettiği başarıya rağmen yerel arama yeteneğinin zayıf olduğunu göstermektedir. Bu çalışmada yerel arama sorununun üstesinden gelmek için *step size* (SS) adı verilen bir düzenleme denenmiştir. Bu düzenleme kaynakların yerel arama yeteneğini global en iyi kaynağın (gbest) konumuna göre ölçeklendirir. Düzenlemenin çalışma mantığı aşağıdaki gibidir:

İlk olarak, birinci iterasyonda üretilen en iyi kaynak gbest olarak alınır. İkinci iterasyonun gbest değeri ilk iterasyonun gbest değeriyle karşılaştırılır ve iyileşme olup olmadığı kontrol edilir. Gelişme varsa gbest değerin iyileşme sayısını hafızada tutmayı

sağlayan GB_{succ} sayacı 1 arttırılır. Gelişme yoksa gbest değerin iyileşmeme sayısını hafızada tutmayı sağlayan GB_{fail} sayacı 1 arttırılır.

Bir sonraki adımda GB_{succ} sayacının değerinden GB_{fail} sayacının değeri çıkarılır. Çıkarma işlemi sonucunda *difference* (*diff*) değeri elde edilir. *Diff* değeri *SS* miktarını hesaplamak için Eşitlik 4.6'daki gibi kullanılır. *SS* değerinin çok büyük ya da çok küçük olması yerel arama yeteneğini olumsuz yönde etkileyecektir. Bu yüzden *SS* parametresinin uç değerler almasını engellemek amacıyla *reset counter (RC)* adı verilen bir sabit değer kullanılmaktadır. RC sabiti sayesinde her RC. iterasyonda GB_{succ} ve GB_{fail} değerleri sıfırlanmakta ve böylece s*tep size* parametresi kullanışlı bir aralıkta tutulmaktadır. RC parametresi yapılan ön çalışmada 3 olarak belirlenmiştir. Bu ön çalışmada kullanılan test problemleri ve sonuçları Çizelge 4.1 ve Çizelge 4.2'de gösterilmektedir.

Çizelge 4.1 Bazı Temel Test Fonsiyonları Krk.: Karakteristik

No	Fonksiyon	Krk.	Boyut	Aralık	Formül
1	Sphere	T/S	30	[-100,100]	$f(x)=\sum_{i=1}^{n} x_i^2$
2	Rosenbrock	T/N	30	[-30,30]	$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$
3	Ackley	Ç/N	30	[-32,32]	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$
4	Griewank	Ç/N	30	[-600,600]	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
5	Rastrigin	Ç/S	30	[-5.12,5.12]	$f(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$
6	Zakharov	T/N	30	[-5,10]	$f(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$
7	Bohachevsky3	Ç/N	2	[-100,100]	$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$

$$diff = \mathrm{GB}_{succ} - \mathrm{GB}_{fail} \tag{4.5}$$

$$SS = \left[\frac{RC+diff}{RC}\right]^{|diff|} \tag{4.6}$$

SS değeri, işçi ve gözcü arıların komşuluk mekanizmalarında Eşitlik 4.7'de görüldüğü gibi kullanılır.

$$v_{ij} = x_{ij} + rand(-1,1)\left(x_{ij} - x_{kj}\right) * SS \tag{4.7}$$

Burada v_{ij} aday çözümü, x_{ij} konumu güncellenen besin kaynağını, *SS* Eşitlik 4.6'dan elde edilen adım ölçeğini, rand(-1,1) ise konumu değişecek her bir parametre için [-1,1] aralığında rassal olarak belirlenmiş reel bir sayıdır.

Çizelge 4.2 Bazı temel test fonsiyonları üzerinde ABC ve A-ABC algoritmalarının performans karşılaştırması B: Boyut, F: Fonksiyon, I: İyi, K:Kötü, Ort: Ortalama, Sdt.S. :Standart Sapma

	B	**ABC**				**A -ABC**			
	10	I	K	O	Std. S.	I	K	O	Std. S.
f1		6,09E-17	2,15E-16	1,1E-16	4,49E-17	1,26E-38	2,64E-37	**9,19E-38**	7,68E-38
f2		0,023029	0,929684	0,372901	0,272132	0,000263	0,242697	**0,021115**	0,048446
f3		4,11E-11	2,58E-10	1,3E-10	6,01E-11	4,44E-15	7,99E-15	**6,15E-15**	1,81E-15
f4		8,3E-14	0,012247	**0,001082**	0,003097	0	0,009865	0,001408	0,003258
f5		0	4,83E-13	*4,09E-14*	1,06E-13	0	0	**0**	0
f6		2,489346	21,94919	11,45552	5,609705	0,002528	0,084956	**0,034154**	0,024854
	30	I	K	O	Std. S.	I	K	O	Std. S.
f1		4,77E-16	9,64E-16	7,49E-16	1,17E-16	1,86E-33	4,42E-32	**9,03E-33**	8,94E-33
f2		0,077905	1,352967	0,405661	0,326522	0,001067	1,010108	**0,219398**	0,360087
f3		2,6E-10	1,29E-09	7,01E-10	3,03E-10	2,22E-14	2,93E-14	**2,48E-14**	2,99E-15
f4		2,22E-16	3,44E-12	2,05E-13	7,2E-13	0	0	**0**	0
f5		5,68E-14	3,31E-09	2,64E-10	7,24E-10	0	1,14E-13	**5E-14**	2,5E-14
f6		212,0386	295,6898	257,8872	21,41915	6,226592	24,80571	**15,64847**	6,099647
	50	I	K	O	Std. S.	I	K	O	Std. S.
f1		1,18E-15	2,08E-15	1,73E-15	1,87E-16	2,77E-30	1,9E-29	**1,04E-29**	4,84E-30
f2		0,146785	1,711159	0,512477	0,433976	0,007955	1,780321	**0,350099**	0,440172
f3		4,59E-10	3,02E-09	1,3E-09	6,38E-10	4E-14	1,39E-13	**5,99E-14**	2,51E-14
f4		8,88E-16	2,8E-11	1,15E-12	5,6E-12	0	0	**0**	0
f5		7,96E-12	0,001047	4,76E-05	0,000209	5,68E-14	1,71E-13	**9,32E-14**	3,23E-14
f6		421,4083	578,6242	511,0507	46,14513	41,51674	129,2201	**94,52427**	22,02562
f7	2	4,08E-07	0,000463	0,000163	0,00015	1,64E-09	1,28E-05	**9,88E-07**	2,56E-06

Ön çalışmada elde edilen başarılı sonuçların ardından RC'nin daha uygun bir değeri olup olmadığını test etmek amacıyla RC için 1 ile 5 arasındaki değerler Karaboğa ve Akay (2009) çalışmalarında kullandıkları 50 temel test problemi üzerinde denenmiş ve sonuçlar Çizelge 4.3'te gösterilmiştir. 50 temel test problemi hakkında detaylı bilgi EK-1'de mevcuttur.

Çizelge 4.3 RC parametresinin farklı değerlerinin 50 Temel sayısal optimizasyon problemi üzerindeki etkisi, F: Fonksiyon, Ort: Ortalama

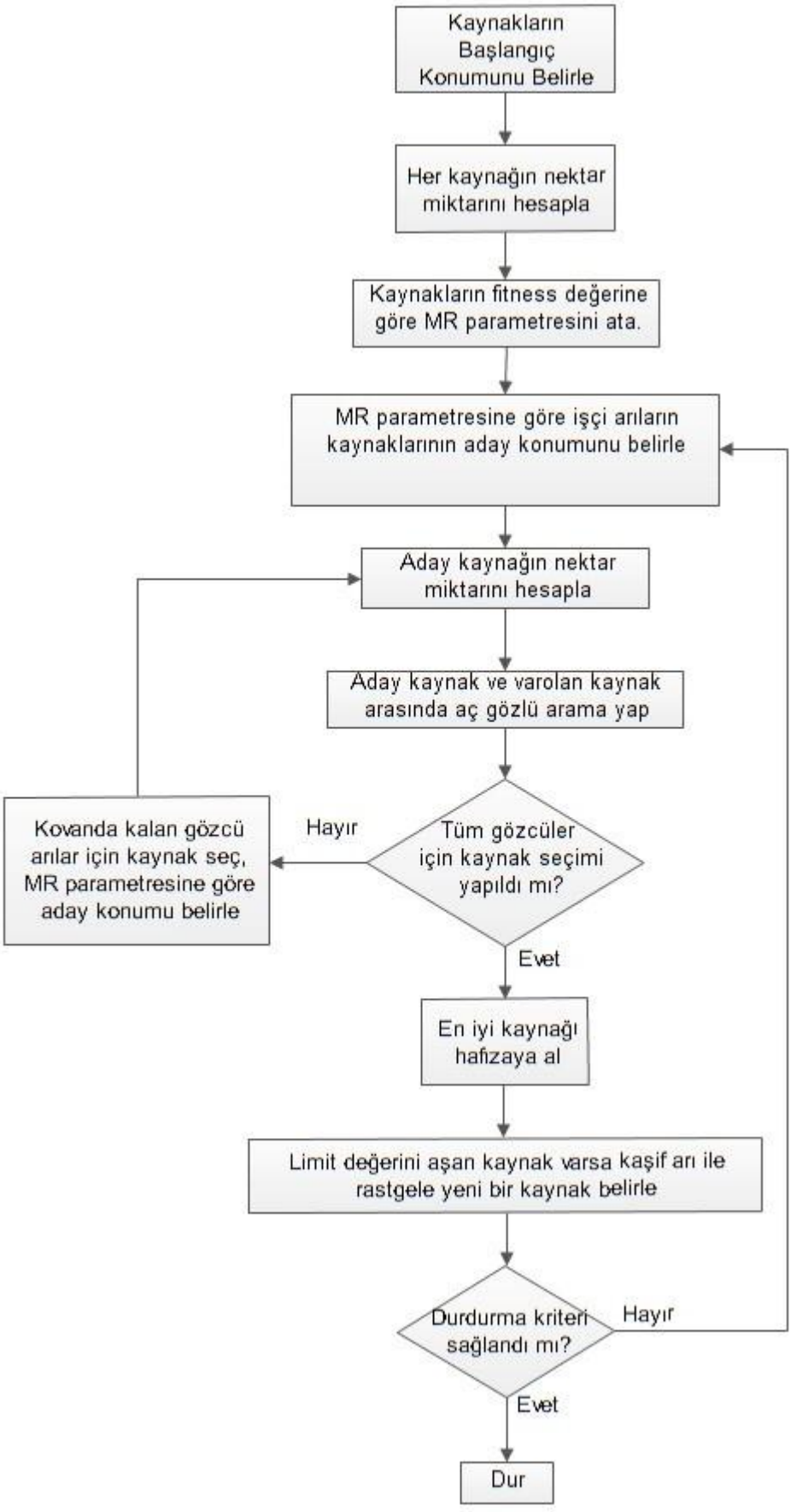
	Reset Counter				
	RC=1	RC=2	RC=3	RC=4	RC=5
F	**Ort.**	**Ort.**	**Ort.**	**Ort.**	**Ort.**
1	0	0	0	0	0
2	0	0	0	0	0
3	**4,2E-161**	1,5E-151	5E-148	3,3E-148	7,2E-147
4	8,29E-97	**4,7E-153**	3,2E-148	5,2E-149	2,3E-147
5	**0,008502**	0,011966	0,012934	0,016291	0,018241
6	4,98E-24	1,12E-26	**1,11E-32**	2,06E-32	3,76E-32
7	-1	-1	-1	-1	-1
8	2,23E-50	4,79E-46	2,39E-62	**4,89E-71**	3,15E-59
9	2,71E-06	**1,68E-06**	1,36E-05	9,53E-05	0,003452
10	-50	-50	-50	-50	-50
11	-210	-210	-210	-210	-210
12	1,01E-17	8,48E-23	**2,02E-28**	3,09E-27	1,14E-27
13	**0,000152**	0,000198	0,000191	0,000209	0,000195
14	**1,8E-103**	5,86E-95	6,76E-92	8,56E-93	1,89E-91
15	8,988612	**2,702158**	3,010722	6,483435	4,84725
16	8,964869	5,466549	5,108056	4,070548	**1,808846**
17	0,481163	0,376299	**0,2425**	0,348757	0,376545
18	0,998004	0,998004	0,998004	0,998004	0,998004
19		0,397887		0,397887	0,397887
20	0	0	0	0	0
21	0	0	0	0	0
22	9,09E-15	**4,55E-15**	1,14E-14	1,59E-14	1,82E-14
23	-12569,5	-12569,5	-12569,5	-12569,5	-12569,5
24	-1,8013	-1,8013	-1,8013	-1,8013	-1,8013
25	-4,68766	-4,68766	-4,68766	-4,68766	-4,68766
26	-9,66015	-9,66015	-9,66015	-9,66015	-9,66015
27	0	0	0	0	0
28	-1,03163	-1,03163	-1,03163	-1,03163	-1,03163
29	0	0	0	0	0
30	0	0	0	0	0
31	-186,731	-186,731	-186,731	-186,731	-186,731
32	3	3	3	3	3
33	0,000308	**0,000308**	0,000309	0,000316	0,000359
34	-10,1532	-10,1532	-10,1532	-10,1532	-10,1532
35	-10,4029	-10,4029	-10,4029	-10,4029	-10,4029
36	-10,5364	-10,5364	-10,5364	-10,5364	-10,5364
37	**0,001017**	0,001262	0,002351	0,002344	0,00401
38	**0,000259**	0,000333	**0,000304**	0,000353	0,000579
39	-3,86278	-3,86278	-3,86278	-3,86278	-3,86278

40	-3,32237	-3,32237	-3,32237	-3,32237	-3,32237
41	0	0	0	0	0
42	**1,54E-14**	1,65E-14	1,65E-14	1,78E-14	1,89E-14
43	1,57E-32	1,57E-32	1,57E-32	1,57E-32	1,57E-32
44	1,35E-32	1,35E-32	1,35E-32	1,35E-32	1,35E-32
45	-1,08094	-1,08094	-1,08094	-1,08094	-1,08094
46	-1,5	-1,5	-1,5	-1,5	-1,5
47	**-1,23435**	-1,10988	-1,02248	-1,02966	-0,91421
48	0	0	0	0	0
49	7,88E-05	0,000111	**6,86E-05**	6,09E-06	0,000164
50	**4,591389**	5,093061	4,905674	5,588798	12,19631

Çizelge 4.3'teki sonuçlar incelendiğinde *RC* parametresi 1 alındığında fonksiyonlar üzerinde genel olarak daha başarılı sonuçlar elde edildiği görülmektedir. Bu sebeple RC değeri 1 alınmış bir başka deyişle SS düzenlemesi kaldırılarak sadece MR düzenlemesi uygulanmıştır.

Ayrıca önerilen A-ABC algoritmasında, x_i ve v_i kaynakları arasında açgözlü aramaya göre seçim yaparken uygunluk değeri yerine, Banharnsakun ve ark.,(2011)'nın önerdiği şekilde amaç değerine göre karşılaştırma yöntemi kullanılmıştır. Standart ABC algoritmasında önerilen ve Banharnsakun ve ark.,(2011)'nın önerdiği aç gözlü arama yöntemleri Şekil 4.6'da gösterilmiştir.

İleriki test işlemlerinde uygulanacak A-ABC algoritmasının akış diyagramı Şekil 4.10'da gösterilmiştir.



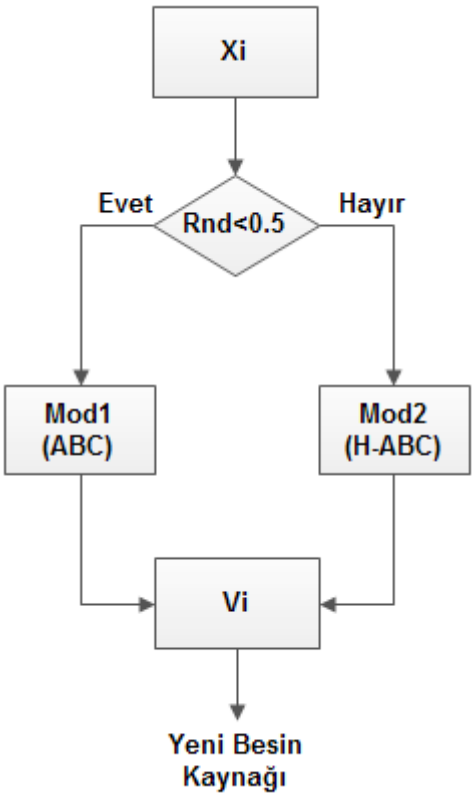
Şekil 4.10 A-ABC Algoritması Akış Diyagramı

Bu çalışmada önerilen *Değişken MR* değeri, 25 kaynağın olduğu ve kaynakların kalitesine göre sıralandığı varsayılan bir durum için Şekil 4.11'de gösterilmiştir.

MR=0.5	MR=0.4	MR=0.3	MR=0.2	MR=0.1
X_1	X_6	X_{11}	X_{16}	X_{21}
X_2	X_7	X_{12}	X_{17}	X_{22}
X_3	X_8	X_{13}	X_{18}	X_{23}
X_4	X_9	X_{14}	X_{19}	X_{24}
X_5	X_{10}	X_{15}	X_{20}	X_{25}

Şekil 4.11 Değişken MR modifikasyonu

Bu çalışmada standart ABC algoritmasının kaynak konumunun güncellenmesi Mod1; değişken MR düzenlemesinin kullanıldığı A-ABC algoritması ise Mod2 olarak isimlendirilmiştir. Önerilen yöntemde Mod1 ya da Mod2'den biri rassal olarak seçilmekte ve seçilen algoritma ile kaynakların yeni konumu belirlenmektedir. Her iki algoritmanın seçilme olasılığı birbirine eşittir. Şekil 4.12'de bu seçim işlemi gösterilmektedir.



Şekil 4.12 Mod1 veya Mod2'nin Seçim Şekli

5. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Optimizasyon problemlerinin çözümünde sıklıkla kullanılan doğa esinli algoritmaların belli tip problemlerin çözümünde sağladığı başarıyı arttırmak için algoritma üzerinde o problem tipine özgü bazı değişiklikler yapılması gerekebilir. Bazen de algoritmanın hali hazırda başarılı bir şekilde çözebildiği bazı problem setleri üzerinde daha başarılı sonuçlar elde etmesini sağlamak amacıyla algoritma üzerinde değişiklikler yapılabilmektedir. Yapılan bu değişikliklerin ne derece etkili olduğunu anlamak için literatürde yaygın olarak kullanılan test fonksiyonlarından faydalanılmaktadır.

Sayısal optimizasyon problemlerinde farklı algoritmaları geniş bir test seti üzerinde kıyaslamak oldukça yaygın bir uygulamadır. Algoritmanın performansını çeşitli yönlerden (sağlamlık, yerel minimumdan kurtulabilme vb.) değerlendirebilmek için farklı özelliklere sahip fonksiyonlar kullanılır. Ancak bir algoritmanın tüm fonksiyonlar üzerinde kusursuz bir performans göstermesi beklenemez. Bu sebeple geliştirilen algoritmanın hangi türden fonksiyonlarda daha başarılı olduğunu ortaya çıkarmak daha doğru olacaktır (Akay, 2009).

ABC algoritmasının performansını iyileştirmek için önerilen *ADL-ABC* ve *A-ABC* isimlerinde iki farklı düzenleme bölüm 4.1 ve 4.2'de detaylı bir şekilde anlatılmıştır. Bu düzenlemelerin ABC algoritmasının performansını nasıl etkilediğini anlayabilmek için bazı sayısal optimizasyon problemleri üzerinde performans analizleri yapılmıştır. Kullanılan problemler *temel test problemleri* ve *Congress of Evolutionary Computation 2005* (CEC05) fonksiyonları olmak üzere iki ana başlığa ayrılmıştır. Elde edilen sonuçlar ilerleyen bölümlerde sunulmuştur.

5.1. *ADL-ABC* ve *A-ABC* Algoritmalarının Temel Sayısal Optimizasyon Test Problemlerinin Çözümünde Performans Analizi

Algoritmalar üzerinde gerçekçi bir performans analizi yapmak için test işlemini gerçekleştirirken farklı karakterdeki test problemlerinin kullanılması çok önemlidir. Böylece önerilen yöntemin güçlü ve zayıf yanları detaylı bir şekilde analiz edilebilir. Bu amaçla ilk problem seti olarak Karaboğa ve Akay'ın (2009) ABC algoritmasının performans analizi için kullandıkları farklı karakteristik yapıdaki 50 temel sayısal

optimizasyon problemi kullanılmıştır. Bu problemler hakkında detaylı bilgi EK-1'de sunulmuştur.

Temel test problemleriyle ilgili ikinci veri seti olarak Zhu ve Kwong (2010) tarafından "*Gbest-guided artificial bee colony algorithm for numerical function optimization*" yayınında kullanılan 6 farklı sayısal optimizasyon problemi ele alınmıştır. Bahsedilen sayısal optimizasyon problemleri hakkında detaylı bilgi Çizelge 5.1'de sunulmuştur.

5.1.1. Temel Test Problemlerinin Çözümünde Kullanılan Parametreler

ADL-ABC ve A-ABC algoritmasının 50 temel test problemi üzerinde performans analizi yapılırken Karaboğa ve Akay'ın (2009) çalışmasındaki parametre değerleri esas alınmıştır. Böylece ADL-ABC ve A-ABC algoritmalarından elde edilen sonuçlar Karaboğa ve Akay'ın GA, PSO, DEA ve ABC algoritmaları için elde ettikleri sonuçlarla karşılaştırılmıştır. Bu çalışmada populasyon sayısı 50 (25 işçi arı, 25 gözcü arı), maksimum hesaplama sayısı (MHS – Maximum Number of Function Evaluations - MaxFES) 500.000 alınmıştır. Standart ABC ve A-ABC algoritmaları için limit değeri Eşitlik 5.1'e göre belirlenmiştir.

$$Limit = KS \text{ x } B \tag{5.1}$$

ADL-ABC algoritmasının doğası gereği limit değeri Eşitlik 4.3'e göre belirlenmiştir. Performans analizi 30 tekrarlı çalıştırma için sonuçların ortalaması ve standart sapma değerleri üzerinden yapılmıştır.

ADL-ABC ve A-ABC algoritmasının 6 temel test fonksiyonu üzerinde performans analizi yapılırken Zhu ve Kwong'un (2010) çalışmasındaki parametre değerleri kullanılmıştır. Böylece ADL-ABC ve A-ABC algoritmalarından elde edilen sonuçlar Zhu ve Kwong'un (2010) önerdikleri GABC algoritması ile karşılaştırılmıştır. Bu çalışmada populasyon sayısı 80 (40 işçi arı, 40 gözcü arı), maksimum hesaplama sayısı 400.000 alınmıştır. ADL-ABC ve A-ABC algoritmalarının test aşamasında, bu çalışmaya paralel olarak populasyon sayısı 80, maksimum hesaplama sayısı 400.000 alınmıştır. Limit değeri ise A-ABC algoritması için Eşitlik 5.1'e göre, ADL-ABC algoritması için ise Eşitlik 4.3'e göre belirlenmiştir. Performans analizi 30 tekrarlı çalışma için, sonuçların ortalaması ve standart sapma değerleri üzerinden yapılmıştır.

Çizelge 5.1. 6 Temel sayısal optimizasyon problemi T: Tek Modlu, Ç:Çok Modlu, S:Ayrıştırılabilir, N:Ayrıştırılamaz

No	F	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
1	Schaffer	2,3	Ç/N	[-100, 100]	0	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
2	Rosenbrock	2,3	T/N	[-50, 50]	0	$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$
3	Sphere	30,60	T/S	[-100, 100]	0	$f(x) = \sum_{i=1}^{n} x_i^2$
4	Griewank	30,60	Ç/N	[-600, 600]	0	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
5	Rastrigin	30,60	Ç/S	[-5.12, 5.12]	0	$f(x) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
6	Ackley	30,60	Ç/N	[-32,768 32,768]	0	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$

5.1.2. Temel Test Problemlerinin Sonuçları

ABC algoritmasının limit parametresi üzerinde yapılan düzenleme sonucu önerilen ADL-ABC ve A-ABC algoritmalarının performansı GA, PSO, DEA, ABC ve BCO algoritmalarının performansları ile karşılaştırılmıştır. GA, PSO, DEA ve ABC algoritmalarının fonksiyonların çözümünde elde ettiği değerler Karaboğa ve Akay (2009)'ın çalışmasından alınmıştır. Çalışmalarında yazarlar 10^{-12} ve daha küçük değerleri 0 olarak varsaydıklarını belirtmişlerdir. BCO algoritmasının fonksiyonların çözümünde elde ettiği değerler Nikolic ve Teodorovic (2013)'in çalışmasından alınmıştır. EK-2 ve EK-3'te ADL-ABC ve A-ABC algoritmaları kullanılarak elde edilen çözümlerden 10^{-20} ve daha küçük olanları 0 olarak varsayılmıştır. Çizelgede en iyi sonuçlar koyu renkte belirtilmiştir.

EK-2 incelendiğinde f5, f9, f13, f37,f40, f46, f47 ve f49 fonksiyonlarında ADL-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar ürettiği; f18, f26, f31, f39 fonksiyonlarının çözümünde ise ADL-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar üretmiş olmasına rağmen sonuçların birbirlerine oldukça yakın olduğu gözlenmektedir. f12, f15, f16, f33, f35, f38, f41, f50 fonksiyonlarında ABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar ürettiği; f17, f19, f23, f24, f25, f28, f34, f36 ve f45 fonksiyonlarının çözümünde ise ABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar üretmiş olmasına rağmen sonuçların birbirlerine oldukça yakın olduğu gözlenmektedir.

Çizelge 5.2'de EK-2'ye göre ADL-ABC algoritması ile diğer algoritmaların ortalama değerler kullanılarak bire bir karşılaştırmaları yapılmıştır. + işareti ADL-ABC algoritmasının daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiğini ifade etmektedir. Çizelge incelendiğinde ADL-ABC algoritmasının GA'dan 36 fonksiyonda daha başarılı, 5 fonksiyonda daha başarısız olduğu ve 9 fonksiyonda ise aynı sonucu ürettiği görülmüştür. ADL-ABC algoritmasının PSO'dan 26 fonksiyonda daha başarılı, 7 fonksiyonda daha başarısız olduğu ve 17 fonksiyonda ise aynı sonucu ürettiği görülmüştür. ADL-ABC algoritmasının DEA'dan 20 fonksiyonda daha başarılı, 12 fonksiyonda daha başarısız olduğu ve 18 fonksiyonda ise aynı sonucu ürettiği görülmüştür. ADL-ABC algoritmasının ABC'den 12 fonksiyonda daha başarılı, 18 fonksiyonda daha başarısız olduğu ve 20 fonksiyonda ise aynı sonucu ürettiği görülmüştür.

Çizelge 5.2. EK-2'ye göre oluşturulmuş 50 Temel sayısal optimizasyon problemi üzerinde GA, PSO, DEA, ABC, BCO ve ADL-ABC algoritmalarının bire bir performans karşılaştırması

F	GA / ADL-ABC	PSO / ADL-ABC	DEA / ADL-ABC	ABC / ADL-ABC	BCO / ADL-ABC
1	•	•	•	•	•
2	+	•	•	•	•
3	+	•	•	•	+
4	+	•	•	•	+
5	+	-	-	+	-
6	•	•	•	•	+
7	•	•	•	•	•
8	•	•	•	•	+
9	-	-	+	+	-
10	+	•	•	•	+
11	+	•	•	•	+
12	+	-	-	-	-
13	+	-	-	+	-
14	•	•	•	•	+
15	+	-	-	-	-
16	+	+	+	-	+
17	+	+	+	-	+
18	+	+	+	+	-
19	-	+	+	-	+
20	•	•	•	•	+
21	•	•	•	•	+
22	+	+	+	•	+
23	+	+	+	-	•
24	+	+	+	-	•
25	+	+	+	-	•
26	+	+	+	+	+
27	+	•	•	•	+
28	-	-	+	-	•
29	+	•	•	•	+
30	•	•	•	•	+
31	-	-	+	+	•
32	+	•	•	•	•
33	+	+	-	-	-
34	+	+	-	-	•
35	+	+	-	-	-
36	+	+	-	-	-
37	+	+	-	+	-
38	+	+	-	-	-
39	+	+	+	+	•
40	+	+	+	+	+
41	+	+	+	-	+
42	+	+	-	-	+
43	+	+	•	•	+
44	+	+	+	•	+
45	-	+	+	-	-
46	+	+	+	+	+
47	+	+	-	+	-
48	•	•	•	•	+
49	+	+	+	+	-
50	+	+	+	-	+

ADL-ABC algoritmasının ABC'den daha başarısız görünmesinin sebebi Karaboğa ve Akay (2009)'ın çalışmalarında 10^{-12} değerini 0 olarak kabul etmiş olduklarından kaynaklandığı düşünülmektedir. ADL-ABC algoritmasının BCO'dan 25 fonksiyonda daha başarılı, 11 fonksiyonda daha başarısız olduğu ve 14 fonksiyonda ise aynı sonucu ürettiği görülmüştür. Çizelge 5.3'te ADL-ABC ve diğer sezgisel algoritmaların karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.3. EK-2'ye göre ADL-ABC ve diğer sezgisel algoritmaların karşılaştırma sonuçları

		GA	PSO	DEA	ABC	BCO
ADL-ABC	İyi	36	26	20	12	25
	Eşit	9	17	18	20	14
	Kötü	5	7	12	18	11

EK-3 incelendiğinde f5, f9, f12, f13, f33, f35, f37, f38, f40, f46, f47, f49 ve f50 fonksiyonlarında A-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar ürettiği; f18, f19, f26, f31, f39, f45 fonksiyonlarının çözümünde ise A-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar üretmiş olmasına rağmen sonuçların birbirlerine oldukça yakın olduğu gözlenmektedir. f15, f16, f17, f22, f25, f28, f34, f36, f42 fonksiyonlarında ABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar ürettiği; f11, f23, f24 fonksiyonlarının çözümünde ise ABC algoritmasının A-ABC algoritmasından daha iyi sonuçlar üretmiş olmasına rağmen sonuçların birbirlerine oldukça yakın olduğu gözlenmektedir.

Sonuçların standart sapmaları açısından değerlendirildiğinde A-ABC algoritmasının karşılaştırılan diğer 5 algoritmadan genel olarak daha küçük standart sapma değerlerine sahip olduğu gözlenmektedir. Bu gözlem genel olarak A-ABC algoritmasının diğer 5 algoritmadan daha kararlı sonuçlar ürettiğini göstermektedir.

Çizelge 5.4. EK-3'e göre oluşturulmuş 50 Temel sayısal optimizasyon problemi üzerinde GA, PSO, DEA, ABC, BCO ve A-ABC algoritmalarının bire bir performans karşılaştırması

F	GA / A-ABC	PSO / A-ABC	DEA / A-ABC	ABC / A-ABC	BCO / A-ABC
1	•	•	•	•	•
2	+	•	•	•	•
3	+	•	•	•	+
4	+	•	•	•	+
5	+	-	-	+	-
6	•	•	•	•	+
7	•	•	•	•	•
8	•	•	•	•	+
9	+	-	+	+	-
10	+	•	•	•	+
11	+	-	-	-	+
12	+	-	-	+	+
13	+	-	-	+	-
14	•	•	•	•	+
15	+	-	-	-	-
16	+	+	+	-	+
17	+	+	+	-	-
18	+	+	+	+	-
19	•	+	+	+	+
20	•	•	•	•	+
21	•	•	•	•	+
22	+	+	+	-	+
23	+	+	+	-	+
24	+	+	+	-	•
25	+	+	+	-	-
26	+	+	+	+	+
27	+	•	•	•	+
28	-	-	+	-	+
29	+	•	•	•	+
30	•	•	•	•	+
31	-	-	+	+	-
32	+	•	•	•	•
33	+	+	+	+	-
34	+	+	-	-	-
35	+	+	+	+	-
36	+	+	-	-	+
37	+	+	+	+	-
38	+	+	-	+	-
39	+	+	+	-	-
40	+	+	+	-	+
41	+	+	+	•	+
42	+	+	-	-	+
43	+	+	•	•	+
44	+	+	+	•	+
45	-	+	+	-	-
46	+	+	+	+	+

47	+	+	+	+	+
48	•	•	•	•	+
49	+	+	+	+	-
50	+	+	+	+	+

Çizelge 5.4'te EK-3'e göre A-ABC algoritması ile diğer algoritmaların ortalama değerler kullanılarak bire bir karşılaştırmaları yapılmıştır. + işareti A-ABC algoritmasının daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiğini ifade etmektedir. Çizelge incelendiğinde A-ABC algoritmasının GA'dan 37 fonksiyonda daha başarılı, 3 fonksiyonda daha başarısız olduğu ve 10 fonksiyonda ise aynı sonucu ürettiği görülmüştür. A-ABC algoritmasının PSO'dan 26 fonksiyonda daha başarılı, 8 fonksiyonda daha başarısız olduğu ve 16 fonksiyonda ise aynı sonucu ürettiği görülmüştür. A-ABC algoritmasının DEA'dan 24 fonksiyonda daha başarılı, 9 fonksiyonda daha başarısız olduğu ve 17 fonksiyonda ise aynı sonucu ürettiği görülmüştür. A-ABC algoritmasının ABC'den 18 fonksiyonda daha başarılı, 13 fonksiyonda daha başarısız olduğu ve 19 fonksiyonda ise aynı sonucu ürettiği görülmüştür. A-ABC algoritmasının BCO'dan 29 fonksiyonda daha başarılı, 16 fonksiyonda daha başarısız olduğu ve 5 fonksiyonda ise aynı sonucu ürettiği görülmüştür. Çizelge 5.5'te A-ABC ve diğer sezgisel algoritmaların karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.5. EK-3'ye göre A-ABC ve diğer sezgisel algoritmaların karşılaştırma sonuçları

		GA	PSO	DEA	ABC	BCO
A-ABC	İyi	37	26	24	18	29
	Eşit	10	16	17	19	5
	Kötü	3	8	9	13	16

Karaboğa ve Akay (2009) çalışmalarında 10^{-12} ve daha küçük değerleri 0 olarak değerlendirmişlerdir. Önerilen algoritmaların sonuçlarını ABC algoritmasının ürettiği sonuçlarda hiçbir yuvarlama işlemi yapılmadan gerçek ortalama, standart sapma ve ortalama standart hata değerleri kıyaslayabilmek için 50 arı (25 işçi, 25 gözcü), 500.000 hesaplama sayısı ve limit değeri de ABC ve A-ABC algoritmaları için Eşitlik 5.1'e göre, ADL-ABC algoritması için ise Eşitlik 4.3'e göre belirlenerek 30 tekrarlı olarak

yeniden çalıştırılmıştır. ADL-ABC algoritmasının ürettiği sonuçlar EK-4'te ve A-ABC algoritmasının ürettiği sonuçlar EK-5'te sunulmuştur. EK-4 ve EK-5'te en iyi değerler kalın yazı tipi ile belirtilmiştir.

EK-4 incelendiğinde f3, f4, f5, f6, f8, f9, f14, f16, f21, f27, f30, f37, f42, f43, f44, f47, f48, f49 fonksiyonlarında ADL-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir. f12, f13, f15, f17, f33, f35, f38, f41, f50 fonksiyonlarında ABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir.

Sonuçlar standart sapmaları açısından değerlendirildiğinde ADL-ABC algoritmasının standart ABC algoritmasından 28 fonksiyonda daha küçük, 16 fonksiyonda ise daha büyük standart sapma değerlerine sahip olduğu gözlenmektedir. Bu da fonksiyonlarda genel olarak ADL-ABC algoritmasının ABC algoritmasından daha kararlı sonuçlar ürettiğini göstermektedir.

EK-5 incelendiğinde f3, f4, f5, f6, f8, f9, f12, f13, f14, f15, f21, f27, f30, f33, f37, f38, f41, f42, f43, f44, f47, f48, f49, f50 fonksiyonlarında A-ABC algoritmasının ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir. f16, f17, f22 fonksiyonlarında ABC algoritmasının A-ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir.

Sonuçlar standart sapmaları açısından değerlendirildiğinde A-ABC algoritmasının standart ABC algoritmasından 30 fonksiyonda daha küçük, 14 fonksiyonda ise daha büyük standart sapma değerlerine sahip olduğu gözlenmektedir. Bu da fonksiyonlarda genel olarak A-ABC algoritmasının ABC algoritmasından daha kararlı sonuçlar ürettiğini göstermektedir.

Çizelge 5.6. EK-4 ve EK-5'e göre oluşturulmuş 50 temel sayısal optimizasyon problemi üzerinde ABC algoritması ile ADL-ABC ve A-ABC algoritmalarının bire bir performans karşılaştırması

F	ABC / ADL-ABC	ABC / A-ABC
1	•	•
2	•	•
3	+	+
4	+	+
5	+	+
6	+	+
7	•	•
8	+	+
9	+	+
10	•	•
11	•	•

12	-	+
13	-	+
14	+	+
15	-	+
16	+	-
17	-	-
18	•	•
19	•	•
20	•	•
21	+	+
22	•	-
23	•	•
24	•	•
25	•	•
26	•	•
27	+	+
28	•	•
29	•	•
30	+	+
31	•	•
32	•	•
33	-	+
34	•	•
35	-	•
36	•	•
37	+	+
38	-	+
39	•	•
40	•	•
41	-	+
42	+	+
43	+	+
44	+	+
45	•	•
46	•	•
47	+	+
48	+	+
49	+	+
50	-	+

Çizelge 5.6'da EK-4 ve Ek-5'e göre standart ABC algoritması ile ADL-ABC ve A-ABC algoritmalarının 50 temel test fonksiyonu için ortalama değerler kullanılarak bire bir karşılaştırmaları yapılmıştır. + işareti önerilen algoritmaların daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiklerini ifade etmektedir. Çizelge incelendiğinde ADL-ABC algoritmasının ABC algoritmasından 18 fonksiyonda daha başarılı, 9 fonksiyonda daha başarısız olduğu ve

23 fonksiyonda ise aynı sonucu ürettiği görülmüştür. A-ABC algoritmasının ise ABC'den 24 fonksiyonda daha başarılı, 3 fonksiyonda daha başarısız olduğu ve 23 fonksiyonda ise aynı sonucu ürettiği görülmüştür. Çizelge 5.7'de ADL-ABC ve A-ABC algoritmalarının standart ABC algoritması ile karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.7. EK-4 ve EK-5'e göre ADL-ABC ve A-ABC algoritmalarının karşılaştırma sonuçları

		ABC			ABC
ADL-ABC	İyi	18	A-ABC	İyi	24
	Eşit	23		Eşit	23
	Kötü	9		Kötü	3

Zhu ve Kwong (2010)'un ABC ve GABC algoritmalarının 6 farklı fonksiyonun çözümünde elde ettiği değerler, aynı fonksiyonlar üzerinde Çizelge 5.8'de ADL-ABC algoritması ile Çizelge 5.10'da ise A-ABC algoritması ile karşılaştırılmıştır. Çizelgelerde en iyi sonuçlar koyu renkte belirtilmiştir.

Çizelge 5.8. 6 Temel sayısal optimizasyon problemi üzerinde ABC, GABC ve ADL-ABC algoritmalarının performans karşılaştırması F: Fonksiyon, B: Boyut, O: Ortalama, S:Standart Sapma

F	B	ABC		GABC		ADL-ABC	
		O	S	O	S	O	S
1	2	0	0	0	0	0	0
	3	5,77E-06	1,65E-05	**1,85E-18**	1,01E-17	0,000324	0,0017738
2	2	9,93E-03	8,14E-03	**1,684969E0-4**	1,45E-04	0,003921	0,0074397
	3	6,45E-02	4,85E-02	**2,66E-03**	2,22E-03	0,0288146	0,0622356
3	30	6,38E-16	1,20E-16	4,18E-16	7,37E-17	**2,06E-80**	5,41E-80
	60	2,28E-15	3,18E-16	1,43E-15	1,38E-16	**2,05E-36**	1,46E-36
4	30	1,27E-15	1,46E-15	2,96E-17	4,99E-17	**1,85E-17**	1,01E-16
	60	2,51E-13	7,51E-13	7,55E-16	4,13E-16	**0**	0
5	30	1,35E-13	7,97E-14	1,33E-14	2,45E-14	**0**	0
	60	2,06E-08	1,12E-07	3,52E-13	1,24E-13	**2,39E-13**	1,70E-13
6	30	4,70E-14	5,95E-15	3,22E-14	3,25E-15	**2,41E-14**	2,76E-15
	60	1,66E-13	2,22E-14	1,00E-13	6,09E-15	**8,51E-14**	8,96E-15

Çizelge 5.8 incelendiğinde 30 ve 60 boyutlu f3, f4, f5, f6 fonksiyonlarında ADL-ABC algoritmasının GABC algoritmasından daha iyi sonuçlar ürettiği; 3 boyutlu f1, 2

ve 3 boyutlu f2 fonksiyonlarında ise GABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir.

Çizelge 5.9. Çizelge 5.8'e göre oluşturulmuş 6 temel optimizasyon problemi üzerinde ADL-ABC algoritması ile ABC ve G-ABC algoritmalarının bire bir performans karşılaştırması

F	Boyut	ABC / ADL-ABC	GABC / ADL-ABC
1	2	•	•
	3	-	-
2	2	+	-
	3	+	-
3	30	+	+
	60	+	+
4	30	+	+
	60	+	+
5	30	+	+
	60	+	+
6	30	+	+
	60	+	+

Çizelge 5.9'da Çizelge 5.8'e göre ADL-ABC algoritması ile standart ABC ve G-ABC algoritmalarının 6 temel test fonksiyonu için ortalama değerler kullanılarak bire bir karşılaştırmaları yapılmıştır. + işareti önerilen algoritmaların daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiklerini ifade etmektedir. Çizelge incelendiğinde ADL-ABC algoritmasının ABC algoritmasından 10 karşılaştırmada daha başarılı, 1 karşılaştırmada daha başarısız olduğu ve 1 karşılaştırmada ise aynı sonucu ürettiği görülmüştür. Ayrıca ADL-ABC

Çizelge 5.10. 6 Temel sayısal optimizasyon problemi üzerinde ABC, GABC, A-ABC algoritmalarının performans karşılaştırması F: Fonksiyon, B: Boyut, O: Ortalama, S:Standart Sapma

F	**B**	**ABC**		**GABC**		**A-ABC**	
		O	**S**	**O**	**S**	**O**	**S**
1	2	0	0	0	0	0	0
	3	5,77E-06	1,65E-05	**1,85E-18**	1,01E-17	6,16E-11	2,23E-10
2	2	9,93E-03	8,14E-03	1,684969E0-4	1,45E-04	**8,31E-08**	1,87E-07
	3	6,45E-02	4,85E-02	2,66E-03	2,22E-03	**5,19E-07**	9,79E-07
3	30	6,38E-16	1,20E-16	4,18E-16	7,37E-17	**2,36E-77**	5,95E-77
	60	2,28E-15	3,18E-16	1,43E-15	1,38E-16	**2,58E-31**	2,37E-31
4	30	1,27E-15	1,46E-15	2,96E-17	4,99E-17	**0**	0
	60	2,51E-13	7,51E-13	7,55E-16	4,13E-16	**0**	0
5	30	1,35E-13	7,97E-14	1,33E-14	2,45E-14	**3,79E-15**	1,44E-14
	60	2,06E-08	1,12E-07	**3,52E-13**	1,24E-13	8,53E-13	1,54E-12
6	30	4,70E-14	5,95E-15	3,22E-14	3,25E-15	**1,58E-14**	3,01E-15
	60	1,66E-13	2,22E-14	1,00E-13	6,09E-15	**4,40E-14**	4,04E-15

algoritmasının GABC algoritmasından 8 karşılaştırmada daha başarılı, 3 karşılaştırmada daha başarısız olduğu, 1 karşılaştırmada ise aynı sonucu ürettiği görülmüştür.

Çizelge 5.10 incelendiğinde 30 ve 60 boyutlu f2, f3, f4, f6 ve 60 boyutlu f5, fonksiyonlarında A-ABC algoritmasının GABC algoritmasından daha iyi sonuçlar ürettiği; 3 boyutlu f1 ve 60 boyutlu f5 fonksiyonlarında ise GABC algoritmasının ADL-ABC algoritmasından daha iyi sonuçlar ürettiği gözlenmektedir.

Çizelge 5.11 Çizelge 5.10'a göre oluşturulmuş 6 temel optimizasyon problemi üzerinde A-ABC algoritması ile ABC ve G-ABC algoritmalarının bire bir performans karşılaştırması

F	Boyut	ABC / A-ABC	GABC / A-ABC
1	2	•	•
	3	+	-
2	2	+	+
	3	+	+
3	30	+	+
	60	+	+
4	30	+	+
	60	+	+
5	30	+	+
	60	+	-
6	30	+	+
	60	+	+

Çizelge incelendiğinde A-ABC algoritmasının ABC algoritmasından 11 karşılaştırmada daha başarılı olduğu ve 1 karşılaştırmada aynı sonucu ürettiği görülmüştür. Ayrıca A-ABC algoritmasının GABC algoritmasından 9 karşılaştırmada daha başarılı, 2 karşılaştırmada daha başarısız olduğu, 1 karşılaştırmada ise aynı sonucu ürettiği görülmüştür. Çizelge 5.12'de ADL-ABC ve A-ABC algoritmalarının standart ABC ve GABC algoritmaları ile karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.12. Çizelge 5.8 ve Çizelge 5.10'a göre ADL-ABC ve A-ABC algoritmalarının karşılaştırma sonuçları

		ABC	**GABC**			**ABC**	**GABC**
ADL-ABC	İyi	10	8	**A-ABC**	İyi	11	9
	Eşit	1	1		Eşit	1	1
	Kötü	1	3		Kötü	-	2

5.2. CEC05 Test Problemleri

Son yirmi yıldan beri farklı yapıda optimizasyon algoritmaları geliştirilmiş ve gerçek parametreli fonksiyonların optimizasyonunda kullanılmışlardır. Bu algoritmaları test etmek amacıyla Sphere, Schwefel's, Rosenbrock's, Rastrigin's gibi standart test problemleri kullanılmıştır. Bazı araştırma çalışmalarında karşılaştırmalar yapılmasına rağmen genellikle bu karşılaştırmalar çalışmada kullanılan test problemleriyle sınırlıdır. Bazı durumlarda seçilen algoritma ve kullanılacak test problemleri birbirini tamamlayıcı olurken aynı algoritma farklı problem tiplerinde başarılı olmayabilir. Bu algoritmaları hesaplamada sonlandırma kriteri, problem boyutu, popülasyonu başlatma şekli ve bağlantı/dönüş noktaları gibi kriterleri standartlaştırarak daha sistematik bir yapıya ihtiyaç vardır (Suganthan ve ark, 2005). Ayrıca iki farklı algoritma ya da bir algoritmanın standart hali ve düzenleme yapılmış hali aynı test problemi üzerinde optimum sonucu elde ediyorsa burada test problemi algoritmalar arasındaki performans farkını ortaya çıkarmakta yetersiz kalıyor demektir. Bu eksikliği giderebilmek için Suganthan ve ark. (2005) zorluk seviyesi yüksek kaydırılmış (shifted), döndürülmüş (rotated) ve melez (hybrid) 25 test probleminden oluşan CEC05 fonksiyonlarını önermişlerdir. Bu fonksiyonlardan f_1-f_5 aralığı tek modlu, f_6-f_{14} aralığı çok modlu, f_{15}-f_{25} aralığı hibrid fonksiyonlardan oluşmaktadır. CEC05 fonksiyonları algoritmalar arasındaki performans farkını daha iyi bir şekilde ortaya koyabildikleri için literatürde sayısal optimizasyon problemleri üzerinde yapılan çalışmalarda sıklıkla tercih edilen fonksiyonlar olmuşlardır. Çalışmanın bu aşamasında kullanılan CEC05 foksiyonları hakında detaylı bilgiye Suganthan ve ark. (2005) tarafından yapılan çalışmadan erişilebilir.

5.2.1. CEC05 test problemlerinin çözümünde kullanılan parametreler

ADL-ABC algoritmasının CEC05 test problemleri üzerinde performansı standart ABC algoritması ile karşılaştırılmıştır. Bu çalışmada populasyon sayısı fonksiyonların 10 boyutlu çözümlemeleri için 10, 30 boyutlu çözümlemeleri için 30, 50 boyutlu çözümlemeleri için 50 olarak alınmıştır. Maksimum hesaplama sayısı Suganthan ve ark. (2005)'a göre; 10 boyut için 100.000, 30 boyut için 300.000 ve 50 boyut için 500.000 alınmıştır. Tüm fonksiyonlar için 25 farklı çalıştırma sonucu elde edilen sonuçların en iyi, en kötü, ortalama ve standart sapma değerleri ele alınmıştır. ABC algoritmasının

çalıştırılmasında limit değeri Eşitlik 5.1'e göre, ADL-ABC algoritması için ise limit değeri Eşitlik 4.3'e göre belirlenmiştir.

5.2.2. CEC05 test problemlerinin sonuçları

Çizelge 5.13. CEC05 sayısal optimizasyon problemleri üzerinde ABC ve ADL-ABC algoritmalarının performans karşılaştırması Populasyon (P):10, Boyut (B):10, Limit KS*B, Çalıştırma Sayısı: 25, Maksimum Hesaplama Sayısı: 500.000, F: Fonksiyon, I:İyi, K:Kötü, O: Ortalama, S:Standart Sapma.

		ABC				ADL-ABC			
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
1	**10**	6,01E-17	2,18E-16	1,23E-16	4,82E-17	1,18E-17	7,20E-17	**3,61E-17**	1,70E-17
2		1,682896	46,00959	15,38789	12,18597	0,003087	0,907059	**0,179783**	0,232885
3		393640,4	3001099	1265622	575915	97175,51	787171,5	**314774,7**	170621,5
4		1476,325	4393,39	2644,261	700,2277	9,667236	944,7594	**179,1457**	220,2367
5		21,79231	315,8453	100,7702	62,75743	0,118612	84,94921	**8,798228**	19,83374
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
6	**10**	0,366136	40,25459	**6,608418**	9,341562	0,006125	96,20108	31,15534	29,30621
7		0,101499	0,493731	**0,248393**	0,082958	0,032401	0,736791	0,334164	0,228869
8		20,24188	20,46311	20,34004	0,057992	20,06222	20,35757	**20,22677**	0,068899
9		0	1,44E-13	1,04E-14	2,96E-14	0	0	**0**	0
10		14,76077	38,79397	26,31536	5,64291	6,111838	47,60011	**23,06685**	9,864668
11		3,270187	6,662484	5,623207	0,707604	3,886772	7,243593	**5,506954**	0,888875
12		43,23047	462,3648	188,4667	100,6597	2,525609	1347,791	**102,5439**	261,5491
13		0,01558	0,334729	0,110749	0,062194	1,11E-16	0,280061	**0,06161**	0,083539
14		3,306431	3,741352	3,531604	0,11649	2,207918	3,674827	**3,152723**	0,403765
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
15	**10**	0	58,78555	11,78193	21,83509	0	101,4965	**11,13981**	30,91825
16		118,1173	189,209	**149,664**	17,58075	100,691	284,275	155,464	54,379
17		151,3293	236,2039	193,2315	20,93016	120,8267	221,0179	**159,6784**	27,29083
18		462,6105	726,5231	**533,4061**	60,82473	500,0034	1005,059	796,0347	184,8014
19		446,9703	642,9839	**524,6495**	40,96059	370,0816	963,7852	818,9442	186,6552
20		500	766,0914	**542,907**	61,72235	393,4169	987,012	753,5064	196,7913
21		213,0552	485,677	**352,4318**	94,23453	300,136	1089,818	577,0833	238,2776
22		304,3241	824,768	**680,379**	209,4189	300	911,351	759,4379	146,3356
23		425,1725	559,6237	**515,2763**	58,57502	547,6006	1088,796	708,7287	210,035
24		200	200	**200**	0	200	800	295,1584	167,965
25		200	200	200	1,18E-08	200	200	200	0

Çizelge 5.14. Çizelge 5.13 'e göre oluşturulmuş 10 boyutlu CEC05 sayısal optimizasyon problemi üzerinde ABC ve ADL-ABC algoritmalarının bire bir performans karşılaştırması

	ABC/ADL-ABC				
F	Tek Modlu	F	Çok Modlu	F	Melez
1	+	6	-	15	+
2	+	7	-	16	-
3	+	8	+	17	+
4	+	9	+	18	-
5	+	10	+	19	-
		11	+	20	-
		12	+	21	-
		13	+	22	-
		14	+	23	-
				24	-
				25	•

Çizelge 5.14'te Çizelge 5.13'e göre standart ABC algoritması ile ADL-ABC algoritmasının 10 boyutlu CEC05 test fonksiyonları için ortalama değerler kullanılarak karşılaştırmaları yapılmıştır. + işareti önerilen algoritmaların daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiklerini ifade etmektedir. Çizelge incelendiğinde tek modlu CEC05 fonksiyonları üzerinde ADL-ABC algoritmasının ABC algoritmasından 5 fonksiyonda da daha başarılı olduğu, çok modlu fonksiyonlar üzerinde ADL-ABC algoritmasının ABC algoritmasından 7 fonksiyonda daha başarılı, 2 fonksiyonda ise başarısız olduğu, melez fonksiyonlar üzerinde ise ADL-ABC algoritmasının ABC algoritmasından 2 fonksiyonda daha başarılı, 8 fonksiyonda başarısız olduğu ve 1 fonksiyonda ise aynı sonucu ürettiği görülmüştür. Çizelge 5.15'te ADL-ABC algoritmasının standart ABC algoritması ile CEC05 fonksiyonları üzerinde 10 boyutlu karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.15. Çizelge 5.13'e göre ADL-ABC ve ABC algoritmalarının karşılaştırma sonuçları

		ABC
ADL-ABC	İyi	14
	Eşit	1
	Kötü	10

Çizelge 5.17'de Çizelge 5.16'ya göre standart ABC algoritması ile ADL-ABC algoritmasının 30 boyutlu CEC05 test fonksiyonları için ortalama değerler kullanılarak

karşılaştırmaları yapılmıştır. + işareti önerilen algoritmaların daha iyi sonuç ürettiğini, – işareti daha kötü sonuç ürettiğini ve • işareti ise aynı sonucu ürettiklerini ifade etmektedir. Çizelge incelendiğinde tek modlu CEC05 fonksiyonları üzerinde ADL-ABC algoritmasının ABC algoritmasından 3 fonksiyonda daha başarılı, 2 fonksiyonda başarısız olduğu, çok modlu fonksiyonlar üzerinde ADL-ABC algoritmasının ABC

Çizelge 5.16 CEC05 sayısal optimizasyon problemleri üzerinde ABC ve ADL-ABC algoritmalarının performans karşılaştırması Populasyon (P):30, Boyut (B):30, Limit KS*B, Çalıştırma Sayısı: 25, Maksimum Hesaplama Sayısı: 500.000, F: Fonksiyon, I:İyi, K:Kötü, O: Ortalama, S:Standart Sapma.

		ABC				**ADL-ABC**			
F	**P/B**	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
1	**30**	3,03E-16	5,49E-16	4,7E-16	7,03E-17	6,42E-17	2,01E-16	**1,32E-16**	4,1E-17
2		778,531	3613,185	**2154,713**	764,7955	1905,007	4355,85	2914,537	731,5187
3		1593184	7789067	**4894192**	1463075	2512199	10336741	6463918	1690740
4		23325,53	47380,46	37719,23	6674,428	6245,33	31514,23	**16439,88**	6149,295
5		7879,86	13052,65	10512,51	1554,13	3376,532	5709,472	**4355,487**	586,3029
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
6	**30**	0,010618	33,18455	9,059766	10,35868	0,001109	38,68275	**3,577231**	8,486917
7		0,002158	0,015638	**0,008922**	0,003523	0,003778	0,014307	0,009898	0,002489
8		20,66958	20,95393	20,86035	0,064408	20,67135	20,83806	**20,75384**	0,047835
9		0	0	0	0	0	0	0	0
10		236,4302	485,4024	361,3804	61,97547	139,1358	206,5857	**173,85**	20,18974
11		22,59386	30,93809	**27,30002**	2,046049	22,72501	30,31059	27,34407	1,867144
12		851,2397	15933,41	7052,337	3506,633	1844,8	10778,6	**6090,753**	2337,14
13		0,115087	0,381303	0,239759	0,066089	0,000631	0,17921	**0,059288**	0,051852
14		12,14691	13,2262	12,90353	0,282009	12,30171	13,27039	**12,83001**	0,290472
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
15	**30**	0	200	**14,35603**	44,14507	1,63E-07	222,957	37,2694	63,10786
16		244,8814	400,7878	347,2996	43,03022	138,0156	357,2895	**234,7516**	58,11489
17		316,1256	468,0118	378,9	47,32288	191,3508	419,7467	**280,5642**	50,78659
18		401,7068	929,3558	899,7572	103,8429	400,9111	916,2764	**887,0832**	103,7223
19		401,959	927,8657	**894,3856**	105,4085	910,237	913,3968	911,9733	1,18341
20		400,5906	928,752	**889,3914**	107,126	908,0192	915,1162	911,8408	2,032516
21		410,2072	500	**496,4083**	17,95856	410,808	500	496,4324	17,83841
22		1007,51	1127,134	1074,594	32,83686	890,7098	979,7176	**942,1279**	18,59732
23		514,412	534,1665	**531,2365**	6,865833	534,1641	534,1653	534,1647	0,000389
24		200	1244,69	501,6664	448,47	200	200	**200**	4,93E-11
25		0	1219,072	**232,5498**	301,8932	200	1176,551	268,4186	239,2676

algoritmasından 6 fonksiyonda daha başarılı, 2 fonksiyonda başarısız ve 1 fonksiyonda ise aynı sonucu ürettiği, melez fonksiyonlar üzerinde ise ADL-ABC algoritmasının ABC algoritmasından 5 fonksiyonda daha başarılı, 6 fonksiyonda ise başarısız olduğu görülmüştür.

Çizelge 5.17. Çizelge 5.16'ya göre oluşturulmuş 30 boyutlu CEC05 sayısal optimizasyon problemi üzerinde ABC ve ADL-ABC algoritmalarının bire bir performans karşılaştırması

	ABC/ADL-ABC				
F	Tek Modlu	F	Çok Modlu	F	Melez
1	+	6	+	15	-
2	-	7	-	16	+
3	-	8	+	17	+
4	+	9	•	18	+
5	+	10	+	19	-
		11	-	20	-
		12	+	21	-
		13	+	22	+
		14	+	23	-
				24	+
				25	-

Çizelge 5.18'de ADL-ABC algoritmasının standart ABC algoritması ile CEC05 fonksiyonları üzerinde 30 boyutlu karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.18. Çizelge 5.16'e göre ADL-ABC ve ABC algoritmalarının karşılaştırma sonuçları

		ABC
ADL-ABC	İyi	14
	Eşit	1
	Kötü	10

Çizelge 5.20'de Çizelge 5.19'a göre standart ABC algoritması ile ADL-ABC algoritmasının 50 boyutlu CEC05 test fonksiyonları için ortalama değerler kullanılarak karşılaştırmaları yapılmıştır. Çizelge incelendiğinde tek modlu CEC05 fonksiyonları üzerinde ADL-ABC algoritmasının ABC algoritmasından 3 fonksiyonda daha başarılı, 2 fonksiyonda başarısız olduğu, çok modlu fonksiyonlar üzerinde ADL-ABC algoritmasının ABC algoritmasından 6 fonksiyonda daha başarılı, 2 fonksiyonda başarısız ve 1 fonksiyonda ise aynı sonucu ürettiği, melez fonksiyonlar üzerinde ise ADL-ABC algoritmasının ABC algoritmasından 9 fonksiyonda daha başarılı, 1 fonksiyonda başarısız olduğu ve 1 fonksiyonda ise aynı sonucu ürettiği görülmüştür.

Çizelge 5.19 CEC05 sayısal optimizasyon problemleri üzerinde ABC ve ADL-ABC algoritmalarının performans karşılaştırması Populasyon (P):50, Boyut (B):50, Limit KS*B, Çalıştırma Sayısı: 25, Maksimum Hesaplama Sayısı: 500.000, F: Fonksiyon, I:İyi, K:Kötü, O: Ortalama, S:Standart Sapma.

		ABC				**ADL-ABC**			
F	**P/B**	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
1	**50**	7,48E-16	1,16E-15	9,47E-16	9,79E-17	1,81E-16	3,56E-16	**2,73E-16**	4,96E-17
2		7604,81	24454,42	**17532,25**	3653,713	18275,2	27016,95	21136,16	1969,142
3		8207439	19040634	**12499840**	2753035	11436865	22902643	17104403	2763475
4		69000	117243,2	99683,19	10773,63	57755,92	100369,7	**71135,99**	11740,26
5		21010,89	29201,6	24914,77	2259,859	9819,625	14797,77	**11819,08**	1075,47
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
6	**50**	0,004644	36,85482	**4,705335**	9,023756	0,001538	37,91254	6,289943	10,57354
7		0,000317	0,001551	**0,000684**	0,000352	0,00014	0,002301	0,000789	0,000682
8		20,82809	20,99011	20,92234	0,037777	20,7284	21,01992	**20,90427**	0,071375
9		0	0	0	0	0	0	0	0
10		739,6821	1264,086	1050,837	114,8396	360,9865	592,4597	**494,5021**	54,41609
11		48,20375	60,39377	56,28765	2,918492	51,46162	59,08297	**55,48483**	2,066071
12		9949,958	56736,85	36467,7	14170,92	16516,85	40875,26	**30550,32**	6660,752
13		0,316484	0,82544	0,546638	0,143205	0,039091	0,400425	**0,143938**	0,089468
14		21,88917	23,1078	22,71875	0,264147	22,01493	23,20554	**22,61598**	0,275581
	P/B	**I**	**K**	**O**	**S**	**I**	**K**	**O**	**S**
15	**50**	0	1,114778	**0,044658**	0,222942	0	10,14791	1,122045	2,72317
16		323,38	401,1114	394,5543	18,89701	222,4775	400,2817	**340,9821**	48,08293
17		448,0769	478,3672	465,7261	7,37664	326,0127	506,4535	**451,0361**	50,71283
18		939,6877	1007,587	966,5333	17,54448	931,3681	948,7648	**937,5721**	4,220101
19		933,5307	1029,097	971,8805	25,3667	930,5868	949,6158	**938,9326**	4,242589
20		947,3574	1051,574	986,3518	25,9585	931,1582	945,942	**938,6**	3,616777
21		500	500	500	1,09E-07	500	500	500	1,02E-12
22		1054,841	1233,041	1157,242	46,51804	975,4626	1031,592	**1003,653**	13,31723
23		539,1223	539,1244	539,1233	0,0006	539,1223	539,1238	**539,1227**	0,000326
24		1272,875	1370,886	1334,428	23,18234	1077,523	1216,339	**1177,723**	31,24762
25		1533,327	1647,563	1591,812	32,21084	1208,72	1270,894	**1237,751**	15,85168

Çizelge 5.20. Çizelge 5.19'a göre oluşturulmuş 50 boyutlu CEC05 sayısal optimizasyon problemi üzerinde ABC ve ADL-ABC algoritmalarının bire bir performans karşılaştırması

	ABC/ADL-ABC				
F	Tek Modlu	F	Çok Modlu	F	Melez
1	+	6	-	15	-
2	-	7	-	16	+
3	-	8	+	17	+
4	+	9	•	18	+
5	+	10	+	19	+
		11	+	20	+
		12	+	21	•
		13	+	22	+
		14	+	23	+
				24	+
				25	+

Çizelge 5.21'de ADL-ABC algoritmasının standart ABC algoritması ile CEC05 fonksiyonları üzerinde 50 boyutlu karşılaştırma sonuçları toplu olarak verilmiştir.

Çizelge 5.21. Çizelge 5.19'a göre ADL-ABC ve ABC algoritmalarının karşılaştırma sonuçları

		ABC
ADL-ABC	İyi	18
	Eşit	2
	Kötü	5

A-ABC algoritmasının CEC05 test problemi üzerinde performans analizi yapıldığında A-ABC algoritmasının standart ABC algoritmasının ürettiği sonuçlardan daha kötü sonuçler ürettiği gözlenmiştir. Bu nedenle A-ABC algoritmasının CEC05 test fonksiyonları üzerindeki sonuçlarına yer verilmemiştir.

6. SONUÇLAR VE ÖNERİLER

6.1. Sonuçlar

Optimizasyon günlük hayatımızda karşımıza çıkan en önemli kavramlar arasında yer almaktadır. Optimizasyon mühendislik tasarımından ekonomi dünyasına, bilgisayar dünyasından endüstriyel uygulamalara, günlük aktivitelerimizden tatil planlamalarımıza kadar kısıtlı kaynakların verimli kullanımı ve maksimum fayda elde etmek amacıyla günümüzde neredeyse tüm sektörlerde karşımıza çıkmaktadır. Kaynakları verimli kullanmak amacıyla geliştirilen çok sayıda optimizasyon algoritması mevcuttur. Bu algoritmalardan biri de 2005 yılında Karaboğa (2005) tarafından bal arılarının besin arama süreçleri modellenerek geliştirilen Yapay Arı Kolonisi (Artificial Bee Colony – ABC) algoritmasıdır. Bu tez çalışmasında ABC algoritması üzerinde algoritmanın temel yapısını bozmayan iki farklı düzenleme önerilmiştir: Bunlar işçi arıların komşu seçimi modifikasyonu ile oluşturulan *Adaptif Dimension Limit ABC (ADL-ABC) algoritması* ve aynı anda birden fazla boyut değişimine imkân sağlayan *Accelerated ABC (A-ABC)* algoritmalarıdır.

Performans testi amacıyla 3 farklı sayısal optimizasyon problemi seti kullanılmıştır: Bunlar Karaboğa ve Akay (2009) tarafından kullanılan 50 temel sayısal optimizasyon problemi, Zhu ve Kwong (2010) tarafından kullanılan 6 temel sayısal optimizasyon problemi ve Suganthan ve ark. (2005) tarafından önerilen 25 farklı CEC05 sayısal optimizasyon problemidir. Önerilen ADL-ABC ve A-ABC algoritmalarının performansını kıyaslamak amacıyla Karaboğa ve Akay (2009)'ın çalışmasından ABC ve bazı diğer sezgisel algoritmaların 50 temel sayısal optimizasyon probleminin çözümünde elde ettiği sonuçlar, Zhu ve Kwong (2010)'un çalışmasından 6 temel sayısal optimizasyon probleminin çözümünde ABC ve GABC'nin elde ettiği sonuçlar kullanılmıştır. CEC05 fonksiyonları ise bölüm 5.2.1'de belirlenen parametreler kullanılarak ABC, ADL-ABC ve A-ABC algoritmaları için çalıştırılmış ve sonuçlar değerlendirilmiştir.

Gerçek değerli performans karşılaştırması sonucunda 50 temel test problemi üzerinde ADL-ABC algoritması ABC algoritmasına göre 18 fonksiyonda daha başarılı, 23 fonksiyonda eşit, 9 fonksiyonda ise daha başarısız performans göstermiştir. A-ABC algoritması ABC algoritmasına göre 24 fonksiyonda daha başarılı, 23 fonksiyonda eşit, 3 fonksiyonda ise daha başarısız performans göstermiştir.

6 temel test problemi üzerinde ADL-ABC algoritması ABC algoritmasına göre 10 karşılaştırmada daha başarılı, 1 karşılaştırmada eşit,1 karşılaştırmada ise daha

başarısız performans göstermiştir. ADL-ABC algoritması GABC algoritmasına göre 8 karşılaştırmada daha başarılı, 1 karşılaştırmada eşit ve 3 karşılaştırmada ise daha başarısız performans göstermiştir. 6 temel test problemi üzerinde A-ABC algoritması ABC algoritmasına göre 11 karşılaştırmada daha başarılı, 1 karşılaştırmada eşit performans göstermiştir. A-ABC algoritması GABC algoritmasına göre 9 karşılaştırmada daha başarılı, 1 karşılaştırmada eşit ve 3 karşılaştırmada ise daha başarısız performans göstermiştir.

CEC05 test problemlerinin 10 boyutlu çalıştırmalarında unimodal 5 fonksiyonun tamamında ADL-ABC algoritması ABC'den daha iyi performans göstermiştir. Multimodal 7 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 2 fonksiyonda ise daha başarısız performans göstermiştir. Melez 2 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 1 fonksiyonda eşit ve 8 fonksiyonda daha başarısız performans göstermiştir.

CEC05 test problemlerinin 30 boyutlu çalıştırmalarında unimodal 3 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 2 fonksiyonda ise daha başarısız performans göstermiştir. Multimodal 6 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 1 fonksiyonda eşit ve 2 fonksiyonda ise daha başarısız performans göstermiştir. Melez 5 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı ve 6 fonksiyonda daha başarısız performans göstermiştir.

CEC05 test problemlerinin 50 boyutlu çalıştırmalarında unimodal 3 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 2 fonksiyonda ise daha başarısız performans göstermiştir. Multimodal 6 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 1 fonksiyonda eşit ve 2 fonksiyonda ise daha başarısız performans göstermiştir. Melez 9 fonksiyonda ADL-ABC algoritması ABC'den daha başarılı, 1 fonksiyonda eşit ve 1 fonksiyonda daha başarısız performans göstermiştir. Melez fonksiyonlarda boyut sayısı arttıkça ADL-ABC algoritmasının daha başarılı sonuçlar elde ettiği görülmektedir.

Sonuç olarak önerilen her iki algoritma da temel test problemleri üzerinde başarı elde ederken yerel arama özelliği daha kuvvetli olan A-ABC algoritmasının ADL-ABC algoritmasından temel test problemlerinde daha başarılı olduğu görülmektedir. Temel test problemlerinden daha karmaşık bir yapıya sahip olan CEC05 problemleri üzerinde ADL-ABC algoritması standart ABC algoritmasından genel olarak daha iyi sonuçlar elde etmiştir. A-ABC algoritması ise CEC05 fonksiyonları üzerinde başarısız olmuştur.

6.2. Öneriler

ADL-ABC ve A-ABC algoritmalarının gerçek dünya problemleri, kısıtlı optimizasyon problemleri ve bu çalışmada kullanılmamış diğer sayısal optimizasyon problemleri üzerinde performans testi yapılabilir. ADL-ABC algoritmasının çok yüksek boyutlu problemlerin çözümünde nasıl bir sonuç elde edeceği test edilebilir. CEC05 fonksiyonları gibi zorluk seviyesi yüksek fonksiyonların çözümünde A-ABC algoritmasının performansını arttırmak amacıyla global arama özelliğini geliştirecek düzenlemeler yapılabilir.

Ayrıca önerilen iyileştirmelerin veri madenciliği, sınıflandırma, kümeleme, tıbbi verilerin işlenmesi, görüntü işleme ve sınıflandırma gibi gerçek dünya problemleri üzerindeki başarısı test edilebilir. Şu ana kadar yapılan çalışmalar neticesinde bu tür problemlerin çözümünde başarılı sonuçlar elde edilebileceği düşünülmektedir. Önerilen iyileştirmeler sürekli fonksiyonlarda test edilmiştir. Önerilen yöntemlerin kesikli fonksiyonlarda kullanılabilecek formu geliştirilerek TSP, araç rotalama gibi problemlerin çözümünde başarısı irdelenebilir.

KAYNAKLAR

Akay, B., 2009, Nümerik Optimizasyon Problemlerinde Yapay Ari Kolonisi (Artificial Bee Colony, ABC) Algoritmasinin Performans Analizi, Doktora Tezi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü*, Kayseri.

Akay, B., Karaboğa, D., 2012, A modified artificial bee colony algorithm for real-parameter optimization, *Information Sciences*, 192, 120–142.

Alatas, B., 2010, Chaotic bee colony algorithms for global numerical optimization, *Expert Systems with Applications*, 37(8), 5682–5687.

Alataş, B., 2007, Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları Geliştirme, Doktora Tezi, *Fırat Üniversitesi Fen Bilimleri Enstitüsü*, Elazığ.

Aydın, D., 2011, Sürü Zekası Yaklaşımlarının Renkli Görüntü Kesimlemeye Uyarlanması Ve Tanıma Sistemleri Üzerinde Gerçekleştirimi, Doktora Tezi, *Ege Üniversitesi Fen Bilimleri Enstitüsü*, İzmir.

Banharnsakun, A., Achalakul, T., Sirinaovakul, B., 2011, The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing*, 11(2), 2888–2901.

Banharnsakun, A., Sirinaovakul, B., Achalakul, T., 2012, Job shop scheduling with the best-so-far ABC. *Engineering Applications of Artificial Intelligence*, 25(3), 583–593.

Blum, C., Roli, A., 2003, Metaheuristics in combinatorial optimization: Overview and conceptural comparison, *ACM Computing Surveys*, 35, 268-308.

Bunday, B.D., 1984, Basic Optimization Methods, *Edward Arnold Ltd*, London.

Carlos Artemio Ceollo Coello, 1999, An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends, *In 1999 Congress on Evolutionary Computation*, 3–13.

Carlos M. Fonseca and Peter J. Fleming, 1998, Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part i: A unified formulation, *IEEE Transactions on Systems*, Man, and Cybernetics, Part A: Systems and Humans, 28(1):26–37.

Chong, E., Zak, S., 2004, An introduction to optimization (2nd ed.), Graham, R., L., Lenstra, J., K., Spencer, J., H., *A Wiley-lnterscience Publication*.

Çelik,Y., 2013, Optimizasyon Problemlerinde Bal Arıları Evlilik Optimizasyonu Algoritmasının (Marriage of Honey Bee Optimiztion-MBO) Performansının Geliştirilmesi, Doktora Tezi, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*, Konya.

De Oliveira, I. M. S., Schirru, R., 2011, Swarm intelligence of artificial bees applied to In-Core Fuel Management Optimization. *Annals of Nuclear Energy*, *38*(5), 1039–1045.

Dorigo, M., Maniezzo, V., Colorni A., 1991, Positive feedback as a search strategy, Technical Report 91-016, Politecnico di Milano, Italy.

Dumitrescu, I., Stützle, T., 2003, Combinations of local search and exact algorithms, Applications of Evolutionary Computing, *LNCS*, 2611,57-68.

Gao, W., Liu, S. 2012, A modified artificial bee colony algorithm, *Computers & Operations Research*, 39(3), 687–697.

Gass, S.I.,2000, Making Decisions with Precision, *Business Week*.

Glover, F., 1986,.Paths for Integer Programming, *13*(5), 533–549.

Gozde, H., Taplamacioglu, M. C., 2011, Comparative performance analysis of artificial bee colony algorithm for automatic voltage regulator (AVR) system. *Journal of the Franklin Institute*, *348*(8), 1927–1946.

Himmelblau,D.M., Edgar,T.F., 1989, Optimization of Chemical Processes, *McGraw-Hill*, Inc.NY.

Holland, J.H., 1975, Adaptation in Natural and Artificial Systems, Ann Arbor: *The University of Michigan Press*.

Horng, M.,H., 2011, Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation, *Expert Systems with Applications*, 38(11), 13785–13791.

Kahaner , D.,Moler, C., Nash, S., 1989, Numerical Methods and Software, *Prentice Hall*, Inc.Englewood Cliffs, NJ.

Kalyanmoy Deb., 2001, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley Interscience Series in Systems and Optimization, *John Wiley & Sons*, Inc., New York, NY, USA.

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.

Kara,İ., 1986, Yöneylem Araştırması, Doğrusal Olmayan Modeller , *Anadolu Üniversitesi Basımevi*, Eskişehir.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, Techn. Rep. TR06, *Erciyes Univ. Press*, Erciyes.

Karaboğa, D., Akay, B., 2009, A comparative study of artificial bee colony algorithm, *Applied Mathematics and Computation*, 214(1), 108–132.

Karaboğa, D., Akay, B., 2011, A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, *Applied Soft Computing*, 11(3), 3021–3031.

Karaboğa, D., Baştürk, B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39(3), 459–471.

Karaboğa, D., Baştürk, B., 2008, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*, 8(1), 687–697.

Karaboğa, D., Ozturk, C., 2011, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, *Applied Soft Computing*, 11(1), 652–657.

Karaboğa, N., 2009, A new design method on ABC algorithm for digital IIR filters, *Journal of the Franklin Institute*, 346(4), 328–348.

Kennedy, J., Eberhart, R., Particle Swarm Optimization, IEEE International Conference on Neural Networks, Perth, Australia, *IEEE Servive Center*, Piscataway, NJ, 1942-1948, 1995.

Kıran, M. S., Gündüz, M., 2013, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, *Applied Soft Computing*, *13*(4), 2188–2203.

Kübat,C., 1983, Yöneylem Araştırması, 8. Ulusal Kongresi, Ankara,.

Ma, M., Liang, J., Guo, M., Fan, Y., Yin, Y.,2011, SAR image segmentation based on Artificial Bee Colony algorithm, Applied Soft Computing, 11(8), 5205–5214.

Manoj, V. J., Elias, E., 2012, Artificial bee colony algorithm for the design of multiplier-less nonuniform filter bank transmultiplexer, *Information Sciences*, *192*, 193–203.

Omkar, S. N., Senthilnath, J., Khandelwal, R., Naik, G. N., Gopalakrishnan, S.,2011, Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures, 11(1), 489–499.

Osman,I. H., Laporte, G., 1996, Metaheuristic: A bibliography, *Annals of Operations Research*, 63, 513-623.

Pan, Q.K., Fatih Tasgetiren, M., Suganthan, P. N., Chua, T. J, 2011, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, *Information Sciences*, 181(12), 2455–2468.

Rao,S.S., 1978, Optimization Theory and Applications, 2nd.Edition, Halsted, Inc.

Reeves, C. R., Ed., 1993, Modern Heuristic Techniques for Combinatorial Problems, *Blackwell Scientific Publishing*, Oxford, England.

Sabat, S. L., Udgata, S. K., Abraham, A., 2010, Artificial bee colony algorithm for small signal model parameter extraction of MESFET, *Engineering Applications of Artificial Intelligence*, 23(5), 689–694.

Samanta, S., Chakraborty, S., 2011), Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm, *Engineering Applications of Artificial Intelligence*, *24*(6), 946–957.

Seeley T., D., 1995, The wisdom of the hive, *Harvard University Press*, Cambridge, MA Selvakumar AAL, Nazer GM (2011) An implementation of expert system in garlic using (abc) algorithm, In: 2011 3rd international conference on electronics computer technology (ICECT), vol 1, pp 45–48.

Singh, A., 2009, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, *Applied Soft Computing*, 9(2), 625–631.

Sonmez, M., 2011, Artificial Bee Colony algorithm for optimization of truss structures, *Applied Soft Computing*, *11*(2), 2406–2418.

Storn, R., Price, K., 1995, Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, *International Computer Science Institute*, Berkley.

Stuart J. Russell, Norvig, P., 2002, Artificial Intelligence: A Modern Approach, Prentice Hall second edition.

Stützle, T. G., 1998, Local Search Algorithms for Combinatorial Problems, Doktora Tezi, Darmstadt Teknik Üniversitesi, Hessen.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y., Auger, A., Tiwari, S., 2005, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, 1–50.

Szeto, W. Y., Wu, Y., Ho, S. C., 2011, An artificial bee colony algorithm for the capacitated vehicle routing problem, *European Journal of Operational Research*, *215*(1), 126–135.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P. N., Chen, A. H. L., 2011, A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops, *Information Sciences*, 181(16), 3459–3475.

Teodorovic, D., Nikolic, M., 2013, Empirical study of the Bee Colony Optimization (BCO) algorithm, *Expert Systems with Applications*, *40*, 4609–4620.

Tereshko, V., Loengarov, A., 2005, Collective decision making in honey-bee foraging Dynamics, *Computing and Informaton Systems*, 9(3), 1-7.

Theraulaz, G., Bonebeau, E., 1999, A Brief History of Stigmergy, Artificial Life, 5, 97-116pp.

Tokmak, M., 2011, Yapay Arı Kolonisi Algoritması İle Ders Çizelgeleme Probleminin Çözümü, Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü, Isparta.

TSai, P., Pan, J., Liao, B., Chu, S., 2009, Enhanced artificial bee colony optimization, International Journal of Innovative Computing, *Information and Control*, 5, 5081–5092.

Weise, T., 2009, Global Optimization Algorithms, *Theory an Aplication*.

Winston, W.L., 2003, Operations Research: Applications and Algorithms, 4. baskı, *International Thomson Publishing*, Belmont, CA.

Yang, X. S., 2010, Engineering Optimization, Hoboken, NJ, USA: *John Wiley & Sons, Inc.*

Yeh, W.C., Hsieh, T.-J., 2011, Solving reliability redundancy allocation problems using an artificial bee colony algorithm, *Computers & Operations Research, 38*(11), 1465–1473.

Zhang, C., Ouyang, D., Ning, J., 2010, An artificial bee colony approach for clustering. *Expert Systems with Applications*, 37(7), 4761–4767.

Zhu, G., Kwong, S., 2010, Gbest-guided artificial bee colony algorithm for numerical function optimization, *Applied Mathematics and Computation*, 217(7), 3166–3173.

EKLER

EK-1 50 Temel sayısal optimizasyon problemleri T: Tek Modlu, Ç:Çok Modlu, S:Ayrıştırılabilir, N:Ayrıştırılamaz

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
1	Stepint	5	T/S	[-5.12, 5.12]	0	$f(x) = 25 + \sum_{i=1}^{n} \lfloor x_i \rfloor$
2	Step	30	T/S	[-100, 100]	0	$f(x) = \sum_{i=1}^{n} \left(\left[x_i + 0.5 \right] \right)^2$
3	Sphere	30	T/S	[-100, 100]	0	$f(x) = \sum_{i=1}^{n} x_i^2$
4	SumSquares	30	T/S	[-10, 10]	0	$f(x) = \sum_{i=1}^{n} i x_i^2$
5	Quartic	30	T/S	[-1.28, 1.28]	0	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$
6	Beale	2	T/N	[-4.5, 4.5]	0	$f(x) = \left(1.5 - x_1 + x_1 x_2\right)^2 + \left(2.25 - x_1 + x_1 x_2^2\right)^2 + \left(2.625 - x_1 + x_1 x_2^3\right)^2$
7	Easom	2	T/N	[-100, 100]	-1	$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$
8	Matyas	2	T/N	[-10, 10]	0	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$
9	Colville	4	T/N	[-10, 10]	0	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1\left(\left(x_2 - 1\right)^2 + \left(x_4 - 1\right)^2\right) + 19.8\left(x_2 - 1\right)\left(x_4 - 1\right)$
10	Trid6	6	T/N	[-D^2, D^2]	-50	$f(x) = \sum_{i=1}^{n} \left(x_i - 1\right)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
11	Trid10	10	T/N	[-D^2, D^2]	-210	$f(x) = \sum_{i=1}^{n} \left(x_i - 1\right)^2 - \sum_{i=2}^{n} x_i x_{i-1}$

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
12	Zakharov	10	T/N	[-5,10]	0	$f(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$
13	Powell	24	T/N	[-4,5]	0	$f(x)=\sum_{i=1}^{n/k}\left(x_{4i-3}+10x_{4i-2}\right)^2+5\left(x_{4i-1}+x_{4i}\right)^2$ $+\left(x_{4i-2}+x_{4i-1}\right)^4+10\left(x_{4i-3}+10x_{4i}\right)^4$
14	Schwefel2.22	30	T/N	[-10, 10]	0	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$
15	Schwefel1.2	30	T/N	[-100, 100]	0	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$
16	Rosenbrock	30	T/N	[-30, 30]	0	$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$
17	Dixon-Price	30	T/N	[-10, 10]	0	$f(x)=\left(x_1-1\right)^2+\sum_{i=2}^{n}i\left(2x_i^2-x_{i-1}\right)^2$
18	Foxholes	2	Ç/S	[-65.536, 65.536]	0.998	$f(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right]^{-1}$
19	Branin	2	Ç/S	[-5,10]; [0,15]	0.398	$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$
20	Bohachevsky1	2	Ç/S	[-100, 100]	0	$f(x)=x_1^2+2x_2^2-0.3\cos\left(3\pi x_1\right)-0.4\cos\left(4\pi x_2\right)+0.7$
21	Booth	2	Ç/S	[-10, 10]	0	$f(x)=\left(x_1+2x_2-7\right)^2+\left(2x_1+x_2-5\right)^2$
22	Rastrigin	30	Ç/S	[-5.12, 5.12]	0	$f(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$
23	Schwefel	30	ÇS	[-500, 500]	-12569.5	$f(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{x_i}\right)$

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
24	Michalewicz	2	Ç/S	[0,pi]	-1.8013	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
25	Michalewicz	5	Ç/S	[0,pi]	-4.6877	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
26	Michalewicz	10	Ç/S	[0,pi]	-9.6602	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2/\pi))^{2m}$ $m = 10$
27	Schaffer	2	Ç/N	[-100, 100]	0	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
28	SHCB	2	Ç/N	[-5,5]	-1.03163	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x1x2 - 4x_2^2 + 4x_2^4$
29	Bohachevsky2	2	Ç/N	[-100, 100]	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
30	Bohachevsky3	2	Ç/N	[-100, 100]	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
31	Shubert	2	Ç/N	[-10, 10]	-186.73	$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)\left(\sum_{i=1}^{5} i\cos\left(i+1\right)x_2 + i\right)$
32	Goldstein-price	2	Ç/N	[-2,2]	3	$f(x) = \begin{bmatrix} 1 + (x_1 + x_2 + 1)^2 \\ \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right) \end{bmatrix}$ $\begin{bmatrix} 30 + (2x_1 - 3x_2)^2 \\ \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right) \end{bmatrix}$

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
33	Kowalik	4	Ç/N	[-5,5]	3.0748e-4	$f(x)=\sum_{i=1}^{11}\left(a_i-\frac{x_1\left(b_i^2+b_i x_2\right)}{b_i^2+b_i x_3+x_4}\right)^2$
34	Shekel5	4	Ç/N	[0,10]	-10.1532	$f(x)=-\sum_{i=1}^{5}\left[\left(x-a_i\right)\left(x-a_i\right)^T+c_i\right]^{-1}$
35	Shekel7	4	Ç/N	[0,10]	-10.40294	$f(x)=-\sum_{i=1}^{7}\left[\left(x-a_i\right)\left(x-a_i\right)^T+c_i\right]^{-1}$
36	Shekel10	4	Ç/N	[0,10]	-10.53641	$f(x)=-\sum_{i=1}^{10}\left[\left(x-a_i\right)\left(x-a_i\right)^T+c_i\right]^{-1}$
37	Perm	4	Ç/N	[-D,D]	0	$f(x)=\sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k+\beta\right)\left(\left(x_i\ / i\right)^k-1\right)\right]^2$
38	PowerSum	4	Ç/N	[0,D]	0	$f(x)=\sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n}x_i^k\right)-b_k\right]^2$
39	Hartman3	3	Ç/N	[0,1]	-3.86	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right]$
40	Hartman6	6	Ç/N	[0,1]	-3.32	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right]$
41	Griewank	30	Ç/N	[-600, 600]	0	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
42	Ackley	30	Ç/N	[-32, 32]	0	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
43	Penalized1	30	Ç/N	[-50, 50]	0	$f(x) = \frac{\pi}{n}\left\{\begin{array}{l}10\sin^2(\pi y_1) \\ +\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] \\ +(y_n-1)^2\end{array}\right\} + \sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i = 1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \left\{\begin{array}{l}k(x_i-a)^m, x_i > a \\ 0, \\ k(-x_i-a)^m, x_i < -a\end{array}\right\}$
44	Penalized2	30	Ç/N	[-50, 50]	0	$f(x) = 0.1\left\{\begin{array}{l}\sin^2(\pi x_1) \\ +\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] \\ +(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\end{array}\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$
45	Langerman2	2	Ç/N	[0,10]	-1.08	$f(x) = -\sum_{i=1}^{m} c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
46	Langerman5	5	Ç/N	[0,10]	-1.5	$f(x) = -\sum_{i=1}^{m} c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$

No	Fonksiyon	Boyut	Karakteristik	Aralık	Opt. Sonuç.	Formül
47	Langerman10	10	Ç/N	[0,10]	NA	$f(x)=-\sum_{i=1}^{m} c_i \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2 \right) \cos\left(\pi \sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2 \right) \right)$
48	FletcherPowell2	2	Ç/N	[-pi,pi]	0	$f(x)=\sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i=\sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right),\ B_i=\sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
49	FletcherPowell5	5	Ç/N	[-pi,pi]	0	$f(x)=\sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i=\sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right),\ B_i=\sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
50	FletcherPowell10	5	Ç/N	[-pi,pi]	0	$f(x)=\sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i=\sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right),\ B_i=\sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$

EK-2 50 Temel sayısal optimizasyon problemi üzerinde GA, PSO, DEA, ABC, BCO ve ADL-ABC algoritmalarının performans karşılaştırması F: Fonksiyon, Min: Minimum, Ort: Ortalama, Sdt.S. :Standart Sapma, OSH: Ortalamanın standart hatası

F	Min.		GA*	PSO*	DEA*	ABC*	BCO**	ADL-ABC***
		Ort.	0	0	0	0	0	0
		Std.S.	0	0	0	0	0	0
1	0	OSH	0	0	0	0	0	0
		Ort.	1170	0	0	0	0	0
		Std.S.	76,56145	0	0	0	0	0
2	0	OSH	13,978144	0	0	0	0	0
		Ort.	1110	0	0	0	7,58145E-11	0
		Std.S.	74,214474	0	0	0	2,65698E-11	0
3	0	OSH	13,549647	0	0	0	4,85097E-12	0
		Ort.	148	0	0	0	2,92851E-10	0
		Std.S.	12,4092893	0	0	0	1,25473E-10	0
4	0	OSH	2,265616	0	0	0	2,29082E-11	0
		Ort.	0,1807	0,00115659	0,0013633	0,0300166	5,28384E-05	0,016557207
		Std.S.	0,027116	0,000276	0,000417	0,004866	2,03331E-05	0,003689904
5	0	OSH	0,004951	0,0000504	0,0000761	0,000888	3,71231E-06	0,000673684
		Ort.	0	0	0	0	1,33701E-12	0
		Std.S.	0	0	0	0	1,7955E-12	0
6	0	OSH	0	0	0	0	3,27812E-13	0
		Ort.	-1	-1	-1	-1	-1	-1
		Std.S.	0	0	0	0	1,72469E-11	0
7	-1	OSH	0	0	0	0	3,14885E-12	0
		Ort.	0	0	0	0	3,98899E-14	0
		Std.S.	0	0	0	0	4,06916E-14	0
8	0	OSH	0	0	0	0	7,42924E-15	0
		Ort.	0,014938	0	0,0409122	0,0929674	3,79251E-08	0,04074354
		Std.S.	0,007364	0	0,081979	0,066277	4,75467E-08	0,053926195
9	0	OSH	0,001344	0	0,014967	0,0121	8,6808E-09	0,009845577
		Ort.	-49,9999	-50	-50	-50	-49,9997398	-50
		Std.S.	0,0000225	0	0	0	0,000400312	4,86229E-14
10	-50	OSH	0,00000411	0	0	0	7,30867E-05	8,87733E-15
		Ort.	-209,476	-210	-210	-210	-209,959472	-210
		Std.S.	0,193417	0	0	0	0,011587643	8,00461E-08
11	-210	OSH	0,035313	0	0	0	0,002115605	1,46144E-08
		Ort.	0,013355	0	0	0,0002476	2,28926E-09	0,000254441
		Std.S.	0,004532	0	0	0,000183	2,1658E-09	0,000203847
12	0	OSH	0,000827	0	0	0,0000334	3,95419E-10	3,72174E-05

		Ort.	9,703771	0,00011004	2,17E-07	0,0031344	4,08028E-07	0,000518853
		Std.S.	1,547983	0,00016	1,36E-07	0,000503	3,84256E-07	9,16023E-05
13	0	OSH	0,282622	0,0000292	2,48E-08	0,0000918	7,01552E-08	1,67243E-05
		Ort.	0	0	0	0	0,000020557	0
		Std.S.	1,386856	0	0	0	3,01608E-06	0
14	0	OSH	0,253204	0	0	0	5,50658E-07	0
		Ort.	7400	0	0	0	2,87601E-08	1419,604507
		Std.S.	1140	0	0	0	2,19555E-08	472,2927793
15	0	OSH	208,1346	0	0	0	4,0085E-09	86,22887228
		Ort.	196000	15,088617	18,203938	0,0887707	10,57166275	0,107379715
		Std.S.	38500	24,170196	5,036187	0,07739	16,58267899	0,145033379
16	0	OSH	7029,10616	4,412854	0,033333	0,014129	3,027569116	0,026479475
		Ort.	1220	0,66666667	0,6666667	0	0,450003633	2,4033E-14
		Std.S.	266	0,00000001	1E-09	0	0,152565513	1,06584E-13
17	0	OSH	48564733	1,8257E-09	1,8257E-10	0	0,027854524	1,94596E-14
		Ort.	0,998004	0,99800393	0,9980039	0,9980039	0,998003838	0,998003869
		Std.S.	0	0	0	0	3,55295E-16	2,14251E-16
18	0,998	OSH	0	0	0	0	6,48677E-17	3,91169E-17
		Ort.	0,397887	0,39788736	0,3978874	0,3978874	0,397887387	0,397887358
		Std.S.	0	0	0	0	6,35409E-08	0
19	0,398	OSH	0	0	0	0	1,16009E-08	0
		Ort.	0	0	0	0	6,18629E-06	0
		Std.S.	0	0	0	0	9,13214E-06	0
20	0	OSH	0	0	0	0	1,66729E-06	0
		Ort.	0	0	0	0	6,44059E-14	0
		Std.S.	0	0	0	0	9,80376E-14	0
21	0	OSH	0	0	0	0	1,78991E-14	0
		Ort.	52,92259	43,9771369	11,716728	0	3,84426E-09	0
		Std.S.	4,56486	11,728676	2,538172	2,538172	1,14559E-09	0
22	0	OSH	0,833426	2,141353	0,463405	0,463405	2,09154E-10	0
		Ort.	-11593,4	-6909,1359	-10266	-12569,487	-12569,4866	-12569,4866
		Std.S.	93,254224	457,957783	521,849292	0	1,00007E-08	2,02667E-12
23	-12569,5	OSH	17,025816	83,611269	95,276209	0	1,82588E-09	3,70019E-13
		Ort.	-1,8013	-1,5728692	-1,801303	-1,8013034	-1,80130341	-1,80130341
		Std.S.	0	0,11986	0	0	3,93507E-11	9,03362E-16
24	-1,8013	OSH	0	0,021883	0	0	7,18443E-12	1,64931E-16
		Ort.	-4,64483	-2,4908728	-4,683482	-4,6876582	-4,68765818	-4,68765818
		Std.S.	0,09785	0,256952	0,01529	0	8,04178E-10	2,5013E-15
25	-4,6877	OSH	0,017865	0,046913	0,002287	0	1,46822E-10	4,56674E-16
26	-9,6602	Ort.	-9,49683	-4,007183	-9,591151	-9,6601517	-9,66015171	-9,66015172

		Std.S.	0,141116	0,502628	0,064205	0	4,23107E-09	1,04311E-15
		OSH	0,025764	0,091767	0,011722	0	7,72485E-10	1,90446E-16
		Ort.	0,004239	0	0	0	4,86463E-15	0
		Std.S.	0,004763	0	0	0	7,63391E-15	0
27	0	OSH	0,0087	0	0	0	1,39375E-15	0
		Ort.	-1,03163	-1,0316285	-1,031628	-1,0316285	-1,03162845	-1,03162845
		Std.S.	0	0	0	0	1,17607E-11	6,77522E-16
28	-1,03163	OSH	0	0	0	0	2,14719E-12	1,23699E-16
		Ort.	0,06829	0	0	0	1,63206E-13	0
		Std.S.	0,078216	0	0	0	2,27566E-13	0
29	0	OSH	0,01428	0	0	0	4,15477E-14	0
		Ort.	0	0	0	0	6,73076E-12	0
		Std.S.	0	0	0	0	7,43862E-12	0
30	0	OSH	0	0	0	0	1,3581E-12	0
		Ort.	-186,731	-186,73091	-186,7309	-186,73091	-186,730909	-186,730909
		Std.S.	0	0	0	0	2,00751E-10	2,74241E-14
31	-186,73	OSH	0	0	0	0	3,6652E-11	5,00696E-15
		Ort.	5,250611	3	3	3	3	3
		Std.S.	5,870093	0	0	0	3,49952E-10	1,44705E-14
32	3	OSH	1,071727	0	0	0	6,38922E-11	2,64195E-15
		Ort.	0,005615	0,00049062	0,0004266	0,0004266	0,000307486	0,000427762
		Std.S.	0,008171	0,000366	0,000273	0,0000604	4,34546E-10	9,35128E-05
33	3,07E-04	OSH	0,001492	0,0000668	0,0000498	0,000011	7,93369E-11	1,70731E-05
		Ort.	-5,66052	-2,0870079	-10,1532	-10,1532	-10,1531997	-10,1531997
		Std.S.	3,866737	1,17846	0	0	3,60383E-10	1,23849E-09
34	-10,1532	OSH	0,705966	0,215156	0	0	6,57967E-11	2,26117E-10
		Ort.	-5,34409	-1,9898713	-10,40294	-10,402941	-10,4029406	-10,4027242
		Std.S.	3,517134	1,420602	0	0	3,78657E-10	0,001011682
35	-10,40294	OSH	0,642138	0,259365	0	0	6,9133E-11	0,000184708
		Ort.	-3,82984	-1,8796753	-10,53641	-10,53641	-10,5364098	-10,5363924
		Std.S.	2,451956	0,432476	0	0	2,50189E-10	6,72545E-05
36	-10,53641	OSH	0,447664	0,078959	0	0	4,56781E-11	0,000012279
		Ort.	0,302671	0,03605158	0,0240069	0,0411052	0,000929463	0,026237722
		Std.S.	0,193254	0,048927	0,046032	0,023056	0,001642492	0,032005715
37	0	OSH	0,035283	0,008933	0,008404	0,004209	0,000299877	0,005843445
		Ort.	0,010405	11,3904479	0,0001425	0,0029468	9,73981E-07	0,004860959
		Std.S.	0,009077	7,3558	0,000145	0,0029468	1,09971E-06	0,003339842
38	0	OSH	0,001657	1,342979	0,0000265	0,002289	2,00778E-07	0,000609772
		Ort.	-3,86278	-3,6333523	-3,862782	-3,8627821	-3,86278215	-3,86278215
39	-3,86	Std.S.	0	0,116937	0	0	7,0696E-11	2,71009E-15

		OSH	0	0,02135	0	0	1,29073E-11	4,94794E-16
		Ort.	-3,29822	-1,8591298	-3,226881	-3,3219952	-3,27777	-3,32236801
		Std.S.	0,05013	0,439958	0,047557	0,047557	0,053700841	4,51681E-16
40	-3,32	OSH	0,009152	0,080325	0,008683	0,008683	0,009804387	8,24657E-17
		Ort.	10,63346	0,01739118	0,0014792	0	0,012713431	1,40462E-05
		Std.S.	1,161455	0,020808	0,002958	0	0,017814131	7,69341E-05
41	0	OSH	0,212052	0,003799	0,00054	0	0,0032524	1,40462E-05
		Ort.	14,67178	0,16462236	0	0	7,35252E-06	2,07834E-14
		Std.S.	0,178141	0,493867	0	0	1,55191E-06	2,39689E-15
42	0	OSH	0,032524	0,0207338	0	0	2,83339E-07	4,37612E-16
		Ort.	13,3772	0,0207338	0	0	1,77429E-11	0
		Std.S.	1,448726	0,041468	0	0	2,41315E-11	0
43	0	OSH	0,2645	0,007571	0	0	4,40579E-12	0
		Ort.	125,0613	0,00767535	0,0021975	0	2,45571E-10	0
		Std.S.	12,001204	0,016288	0,004395	0	1,80035E-10	0
44	0	OSH	2,19111	0,002974	0,000802	0	3,28697E-11	0
		Ort.	-1,08094	-0,67268	-1,080938	-1,0809384	-3,06774756	-1,08093844
		Std.S.	0	0,274621	0	0	5,27957E-11	4,60995E-16
45	-1,08	OSH	0	0,050139	0	0	9,63913E-12	8,41662E-17
		Ort.	-0,96842	-0,5048579	-1,499999	-0,93815	-1,49994381	-1,49999922
		Std.S.	0,287548	0,213626	0	0,000208	5,73078E-08	6,06692E-10
46	-1,5	OSH	0,052499	0,039003	0	0,000038	1,04629E-08	1,10767E-10
		Ort.	-0,63644	-0,0025656	-1,0528	-0,4460925	-1,17242327	-0,87239
		Std.S.	0,374682	0,003523	0,302257	0,133958	0,294476459	0,181824959
47	NA	OSH	0,068407	0,000643	0,055184	0,024457	0,0537638	0,033196699
		Ort.	0	0	0	0	9,44478E-12	0
		Std.S.	0	0	0	0	2,26617E-11	0
48	0	OSH	0	0	0	0	4,13745E-12	0
		Ort.	0,004303	1457,88344	5,988783	0,1735495	8,11416E-06	0,000198212
		Std.S.	0,009469	1269,36239	7,334731	0,068175	1,08904E-05	0,000936027
49	0	OSH	0,001729	231,752805	1,339133	0,012447	1,98831E-06	0,000170895
		Ort.	29,57348	1364,45555	781,55028	8,2334401	124,8629713	9,769376763
		Std.S.	16,02078	1325,37966	1048,81349	8,092742	286,3093074	15,07916564
50	0	OSH	2,925035	1325,37966	241,980111	1,477526	52,27268869	2,753079246

* Karaboğa ve Akay (2009) çalışmasından alınmıştır. Bu çalışmada yazarlar 10^{-12} ve daha küçük değerleri 0 olarak kabul ettiklerini belirtmişlerdir.
** Nikolic ve Teodorovic (2013) çalışmasından alınmıştır.
*** 10^{-20} ve daha küçük değerler 0 olarak değerlendirilmiştir.

EK-3 50 Temel sayısal optimizasyon problemi üzerinde GA, PSO, DEA, ABC, BCO ve A-ABC algoritmalarının performans karşılaştırması F: Fonksiyon, Min: Minimum, Ort: Ortalama, Sdt.S. :Standart Sapma, OSH: Ortalamanın standart hatası

F	Min.		GA*	PSO*	DEA*	ABC*	BCO**	A-ABC***
1	0	Ort.	0	0	0	0	0	0
		Std.S.	0	0	0	0	0	0
		OSH	0	0	0	0	0	0
2	0	Ort.	1170	0	0	0	0	0
		Std.S.	76,56145	0	0	0	0	0
		OSH	13,978144	0	0	0	0	0
3	0	Ort.	1110	0	0	0	7,58145E-11	0
		Std.S.	74,214474	0	0	0	2,66E-11	0
		OSH	13,549647	0	0	0	4,85E-12	0
4	0	Ort.	148	0	0	0	2,92851E-10	0
		Std.S.	12,4092893	0	0	0	1,25E-10	0
		OSH	2,265616	0	0	0	2,29E-11	0
5	0	Ort.	0,1807	0,00115659	0,0013633	0,0300166	5,28384E-05	0,008502231
		Std.S.	0,027116	0,000276	0,000417	0,004866	2,03E-05	0,002064508
		OSH	0,004951	0,0000504	0,0000761	0,000888	3,71E-06	0,000376928
6	0	Ort.	0	0	0	0	1,33701E-12	0
		Std.S.	0	0	0	0	1,80E-12	0
		OSH	0	0	0	0	3,28E-13	0
7	-1	Ort.	-1	-1	-1	-1	-1	-1
		Std.S.	0	0	0	0	1,72E-11	0
		OSH	0	0	0	0	3,15E-12	0
8	0	Ort.	0	0	0	0	3,98899E-14	0
		Std.S.	0	0	0	0	4,07E-14	0
		OSH	0	0	0	0	7,43E-15	0
9	0	Ort.	0,014938	0	0,0409122	0,0929674	3,79251E-08	2,71437E-06
		Std.S.	0,007364	0	0,081979	0,066277	4,75E-08	5,20E-06
		OSH	0,001344	0	0,014967	0,0121	8,68E-09	9,50E-07
10	-50	Ort.	-49,9999	-50	-50	-50	-49,9997398	-50
		Std.S.	0,0000225	0	0	0	0,000400312	3,41E-14
		OSH	0,00000411	0	0	0	7,31E-05	6,23E-15
11	-210	Ort.	-209,476	-210	-210	-210	-209,959472	-209,999995
		Std.S.	0,193417	0	0	0	0,011587643	1,21E-05
		OSH	0,035313	0	0	0	0,002115605	2,20E-06
12	0	Ort.	0,013355	0	0	0,0002476	2,28926E-09	1,01176E-17
		Std.S.	0,004532	0	0	0,000183	2,17E-09	4,84E-17
		OSH	0,000827	0	0	0,0000334	3,95E-10	8,83E-18
13	0	Ort.	9,703771	0,00011004	2,17E-07	0,0031344	4,08028E-07	0,000152125

		Std.S.	1,547983	0,00016	1,36E-07	0,000503	3,84E-07	4,00E-05
		OSH	0,282622	0,0000292	2,48E-08	0,0000918	7,02E-08	7,30E-06
14	0	Ort.	0	0	0	0	2,06E-05	0
		Std.S.	1,386856	0	0	0	3,02E-06	0
		OSH	0,253204	0	0	0	5,51E-07	0
15	0	Ort.	7400	0	0	0	2,87601E-08	8,988612425
		Std.S.	1140	0	0	0	2,20E-08	11,46585391
		OSH	208,1346	0	0	0	4,01E-09	2,093378718
16	0	Ort.	196000	15,088617	18,203938	0,0887707	10,57166275	8,964868767
		Std.S.	38500	24,170196	5,036187	0,07739	16,58267899	19,95737437
		OSH	7029,10616	4,412854	0,033333	0,014129	3,027569116	3,643718391
17	0	Ort.	1220	0,66666667	0,6666667	0	0,450003633	0,481163034
		Std.S.	266	0,00000001	1,00E-09	0	0,152565513	0,303618361
		OSH	48564733	1,83E-09	1,83E-10	0	0,027854524	0,055433134
18	0,998	Ort.	0,998004	0,99800393	0,9980039	0,9980039	0,998003838	0,998003869
		Std.S.	0	0	0	0	3,55E-16	2,27E-16
		OSH	0	0	0	0	6,49E-17	4,14E-17
19	0,398	Ort.	0,397887	0,39788736	0,3978874	0,3978874	0,397887387	0,397887
		Std.S.	0	0	0	0	6,35E-08	0
		OSH	0	0	0	0	1,16E-08	0
20	0	Ort.	0	0	0	0	6,18629E-06	0
		Std.S.	0	0	0	0	9,13E-06	0
		OSH	0	0	0	0	1,67E-06	0
21	0	Ort.	0	0	0	0	6,44059E-14	0
		Std.S.	0	0	0	0	9,80E-14	0
		OSH	0	0	0	0	1,79E-14	0
22	0	Ort.	52,92259	43,9771369	11,716728	0	3,84426E-09	9,09495E-15
		Std.S.	4,56486	11,728676	2,538172	2,538172	1,15E-09	2,13E-14
		OSH	0,833426	2,141353	0,463405	0,463405	2,09E-10	3,88E-15
23	-12569,5	Ort.	-11593,4	-6909,1359	-10266	-12569,487	-12569,4866	-12569,48662
		Std.S.	93,254224	457,957783	521,849292	0	1,00E-08	3,24E-12
		OSH	17,025816	83,611269	95,276209	0	1,83E-09	5,91E-13
24	-1,8013	Ort.	-1,8013	-1,5728692	-1,801303	-1,8013034	-1,80130341	-1,80130341
		Std.S.	0	0,11986	0	0	3,94E-11	6,80E-16
		OSH	0	0,021883	0	0	7,18E-12	1,24E-16
25	-4,6877	Ort.	-4,64483	-2,4908728	-4,683482	-4,6876582	-4,68765818	-4,687658179
		Std.S.	0,09785	0,256952	0,01529	0	8,04E-10	1,79E-15
		OSH	0,017865	0,046913	0,002287	0	1,47E-10	3,26E-16
26	-9,6602	Ort.	-9,49683	-4,007183	-9,591151	-9,6601517	-9,66015171	-9,660151716
		Std.S.	0,141116	0,502628	0,064205	0	4,23E-09	0
		OSH	0,025764	0,091767	0,011722	0	7,72E-10	0

27	0	Ort.	0,004239	0	0	0	4,86463E-15	0
		Std.S.	0,004763	0	0	0	7,63E-15	0
		OSH	0,0087	0	0	0	1,39E-15	0
28	-1,03163	Ort.	-1,03163	-1,0316285	-1,031628	-1,0316285	-1,03162845	-1,031628453
		Std.S.	0	0	0	0	1,18E-11	6,80E-16
		OSH	0	0	0	0	2,15E-12	1,24E-16
29	0	Ort.	0,06829	0	0	0	1,63206E-13	0
		Std.S.	0,078216	0	0	0	2,28E-13	0
		OSH	0,01428	0	0	0	4,15E-14	0
30	0	Ort.	0	0	0	0	6,73E-12	0
		Std.S.	0	0	0	0	7,44E-12	0
		OSH	0	0	0	0	1,36E-12	0
31	-186,73	Ort.	-186,731	-186,73091	-186,7309	-186,73091	-186,730909	-186,7309088
		Std.S.	0	0	0	0	2,01E-10	3,71E-14
		OSH	0	0	0	0	3,67E-11	6,78E-15
32	3	Ort.	5,250611	3	3	3	3	3
		Std.S.	5,870093	0	0	0	3,50E-10	9,29E-16
		OSH	1,071727	0	0	0	6,39E-11	1,70E-16
33	3,07E-04	Ort.	0,005615	0,00049062	0,0004266	0,0004266	0,000307486	0,000307636
		Std.S.	0,008171	0,000366	0,000273	0,0000604	4,35E-10	2,54E-07
		OSH	0,001492	0,0000668	0,0000498	0,000011	7,93E-11	4,63E-08
34	-10,1532	Ort.	-5,66052	-2,0870079	-10,1532	-10,1532	-10,1531997	-10,15319968
		Std.S.	3,866737	1,17846	0	0	3,60E-10	4,80E-15
		OSH	0,705966	0,215156	0	0	6,58E-11	8,76E-16
35	-10,40294	Ort.	-5,34409	-1,9898713	-10,40294	-10,402941	-10,4029406	-10,40294057
		Std.S.	3,517134	1,420602	0	0	3,79E-10	2,97E-15
		OSH	0,642138	0,259365	0	0	6,91E-11	5,42E-16
36	-10,53641	Ort.	-3,82984	-1,8796753	-10,53641	-10,53641	-10,5364098	-10,53640982
		Std.S.	2,451956	0,432476	0	0	2,50E-10	3,52E-15
		OSH	0,447664	0,078959	0	0	4,57E-11	6,42E-16
37	0	Ort.	0,302671	0,03605158	0,0240069	0,0411052	0,000929463	0,001017406
		Std.S.	0,193254	0,048927	0,046032	0,023056	0,001642492	0,00085296
		OSH	0,035283	0,008933	0,008404	0,004209	0,000299877	0,000155729
38	0	Ort.	0,010405	11,3904479	0,0001425	0,0029468	9,74E-07	0,000258582
		Std.S.	0,009077	7,3558	0,000145	0,0029468	1,10E-06	0,000160208
		OSH	0,001657	1,342979	0,0000265	0,002289	2,01E-07	2,93E-05
39	-3,86	Ort.	-3,86278	-3,6333523	-3,862782	-3,8627821	-3,86278215	-3,862782148
		Std.S.	0	0,116937	0	0	7,07E-11	2,27E-15
		OSH	0	0,02135	0	0	1,29E-11	4,14E-16
40	-3,32	Ort.	-3,29822	-1,8591298	-3,226881	-3,3219952	-3,27777	-3,322368011
		Std.S.	0,05013	0,439958	0,047557	0,047557	0,053700841	2,56E-16

		OSH	0,009152	0,080325	0,008683	0,008683	0,009804387	4,68E-17
41	0	Ort.	10,63346	0,01739118	0,0014792	0	0,012713431	0
		Std.S.	1,161455	0,020808	0,002958	0	0,017814131	0
		OSH	0,212052	0,003799	0,00054	0	0,0032524	0
42	0	Ort.	14,67178	0,16462236	0	0	7,35252E-06	1,53833E-14
		Std.S.	0,178141	0,493867	0	0	1,55E-06	4,46E-15
		OSH	0,032524	0,0207338	0	0	2,83E-07	8,14E-16
43	0	Ort.	13,3772	0,0207338	0	0	1,77429E-11	0
		Std.S.	1,448726	0,041468	0	0	2,41E-11	0
		OSH	0,2645	0,007571	0	0	4,41E-12	0
44	0	Ort.	125,0613	0,00767535	0,0021975	0	2,45571E-10	0
		Std.S.	12,001204	0,016288	0,004395	0	1,80E-10	0
		OSH	2,19111	0,002974	0,000802	0	3,29E-11	0
45	-1,08	Ort.	-1,08094	-0,67268	-1,080938	-1,0809384	-3,06774756	-1,080938442
		Std.S.	0	0,274621	0	0	5,28E-11	6,00E-16
		OSH	0	0,050139	0	0	9,64E-12	1,09E-16
46	-1,5	Ort.	-0,96842	-0,5048579	-1,499999	-0,93815	-1,49994381	-1,499999223
		Std.S.	0,287548	0,213626	0	0,000208	5,73E-08	6,49E-16
		OSH	0,052499	0,039003	0	0,000038	1,05E-08	1,18E-16
47	NA	Ort.	-0,63644	-0,0025656	-1,0528	-0,4460925	-1,17242327	-1,234354274
		Std.S.	0,374682	0,003523	0,302257	0,133958	0,294476459	0,334336367
		OSH	0,068407	0,000643	0,055184	0,024457	0,0537638	0,061041475
48	0	Ort.	0	0	0	0	9,44478E-12	0
		Std.S.	0	0	0	0	2,27E-11	0
		OSH	0	0	0	0	4,14E-12	0
49	0	Ort.	0,004303	1457,88344	5,988783	0,1735495	8,11E-06	7,88E-05
		Std.S.	0,009469	1269,36239	7,334731	0,068175	1,09E-05	0,000243259
		OSH	0,001729	231,752805	1,339133	0,012447	1,99E-06	4,44E-05
50	0	Ort.	29,57348	1364,45555	781,55028	8,2334401	124,8629713	4,591389032
		Std.S.	16,02078	1325,37966	1048,81349	8,092742	286,3093074	5,450341615
		OSH	2,925035	1325,37966	241,980111	1,477526	52,27268869	0,995096329

* Karaboğa ve Akay (2009) çalışmasından alınmıştır. Bu çalışmada yazarlar 10^{-12} ve daha küçük değerleri 0 olarak kabul ettiklerini belirtmişlerdir.
** Nikolic ve Teodorovic (2013) çalışmasından alınmıştır.
*** 10^{-20} ve daha küçük değerler 0 olarak değerlendirilmiştir.

EK-4 50 Temel sayısal optimizasyon problemi üzerinde ABC, ADL-ABC algoritmalarının gerçek değerli performans karşılaştırması F: Fonksiyon, Min: Minimum, Ort: Ortalama, Sdt.S. : Standart Sapma, OSH: Ortalamanın standart hatası

		ABC			ADL-ABC		
F	Min.	Ort.	Std.S.	OSH	Ort.	Std.S.	OSH
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	4,53E-16	7,01E-17	1,28E-17	**2,4E-156**	7,4E-156	1,4E-156
4	0	4,43E-16	6E-17	1,09E-17	**1,3E-160**	7,1E-160	1,3E-160
5	0	0,031075	0,006537	0,001193	**0,016557**	0,00369	0,000674
6	0	8,75E-15	2,01E-14	3,66E-15	**3,75E-31**	4,04E-31	7,38E-32
7	-1	-1	1,05E-06	1,92E-07	-1	0	0
8	0	9,51E-16	1,03E-15	1,88E-16	**2,1E-105**	1,2E-104	2,1E-105
9	0	0,107723	0,063184	0,011536	**0,040744**	0,053926	0,009846
10	-50	-50	1,52E-13	2,78E-14	-50	4,86E-14	8,88E-15
11	-210	-210	1,29E-07	2,35E-08	-210	8E-08	1,46E-08
12	0	**0,000219**	0,000214	3,91E-05	0,000254	0,000204	3,72E-05
13	0	**0,000359**	2,83E-05	5,17E-06	0,000519	9,16E-05	1,67E-05
14	0	1,18E-15	6,81E-17	1,24E-17	**6,48E-82**	1,17E-81	2,13E-82
15	0	**1330,481**	506,9713	92,56031	1419,605	472,2928	86,22887
16	0	0,136845	0,190254	0,034736	**0,10738**	0,145033	0,026479
17	0	**3,95E-15**	2,46E-15	4,5E-16	2,4E-14	1,07E-13	1,95E-14
18	0.998	0,998004	8,88E-15	1,62E-15	0,998004	2,14E-16	3,91E-17
19	0.398	0,397887	0	0	0,397887	0	0
20	0	0	0	0	0	0	0
21	0	1,6E-18	1,75E-18	3,19E-19	**0**	0	0
22	0	0	0	0	0	0	0
23	-12569.5	-12569,5	1,96E-12	3,59E-13	-12569,5	2,03E-12	3,7E-13
24	-1.8013	-1,8013	6,8E-16	1,24E-16	-1,8013	9,03E-16	1,65E-16
25	-4.6877	-4,68766	1,81E-15	3,31E-16	-4,68766	2,5E-15	4,57E-16
26	-9.6602	-9,66015	6,28E-16	1,15E-16	-9,66015	1,04E-15	1,9E-16
27	0	1,24E-09	4,81E-09	8,79E-10	**0**	0	0
28	-1.03163	-1,03163	6,72E-16	1,23E-16	-1,03163	6,78E-16	1,24E-16
29	0	0	0	0	0	0	0
30	0	2,19E-14	8,67E-14	1,58E-14	**0**	0	0
31	-186.73	-186,731	2,78E-14	5,08E-15	-186,731	2,74E-14	5,01E-15
32	3	3	2,45E-12	4,47E-13	3	1,45E-14	2,64E-15
33	3.0748e-4	**0,000411**	4,98E-05	9,1E-06	0,000428	9,35E-05	1,71E-05
34	-10.1532	-10,1532	3,72E-15	6,78E-16	-10,1532	1,24E-09	2,26E-10
35	-10.40294	**-10,4029**	3,75E-15	6,85E-16	-10,4027	0,001012	0,000185
36	-10.53641	-10,5364	1,92E-15	3,5E-16	-10,5364	6,73E-05	1,23E-05
37	0	0,036873	0,024744	0,004518	**0,026238**	0,032006	0,005843
38	0	**0,004175**	0,003082	0,000563	0,004861	0,00334	0,00061
39	-3.86	-3,86278	1,89E-15	3,46E-16	-3,86278	2,71E-15	4,95E-16

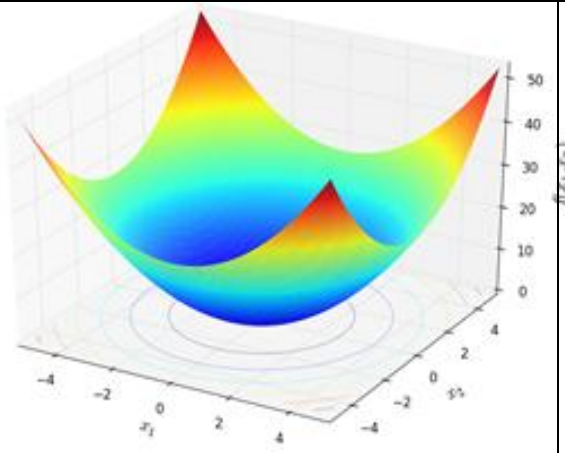
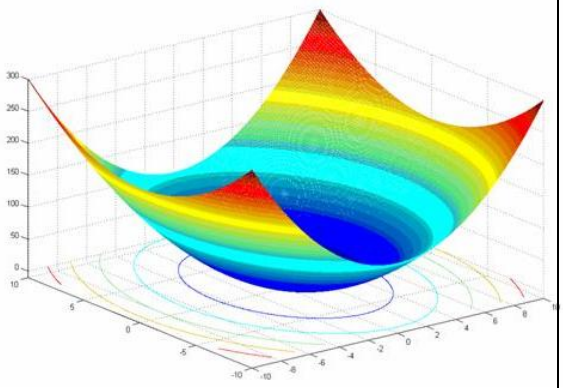
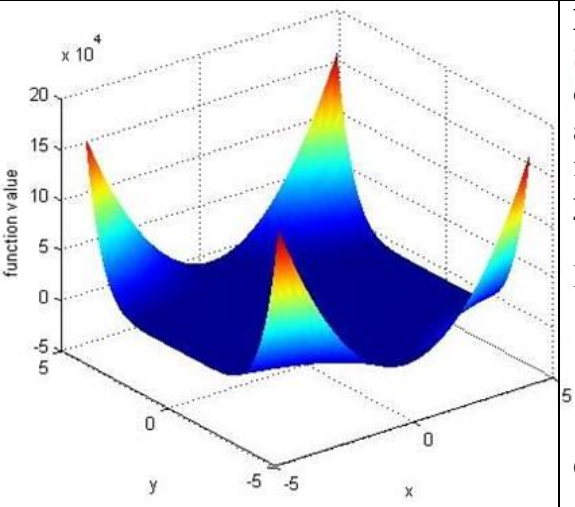
40	-3.32	-3,32237	7,25E-16	1,32E-16	-3,32237	4,52E-16	8,25E-17
41	0	**7,11E-17**	2E-16	3,65E-17	1,4E-05	7,69E-05	1,4E-05
42	0	3,4E-14	3,8E-15	6,94E-16	**2,08E-14**	2,4E-15	4,38E-16
43	0	3,99E-16	8,35E-17	1,52E-17	**1,57E-32**	5,57E-48	1,02E-48
44	0	4,1E-16	7,78E-17	1,42E-17	**1,35E-32**	5,57E-48	1,02E-48
45	-1.08	-1,08094	5,07E-16	9,25E-17	-1,08094	4,61E-16	8,42E-17
46	-1.5	-1,5	1,52E-06	2,77E-07	-1,5	6,07E-10	1,11E-10
47	NA	-0,85828	0,194355	0,035484	**-0,87239**	0,181825	0,033197
48	0	2,11E-19	2,16E-19	3,95E-20	**0**	0	0
49	0	0,061056	0,06227	0,011369	**0,000198**	0,000936	0,000171
50	0	**8,756695**	6,373139	1,163576	9,769377	15,07917	2,753079

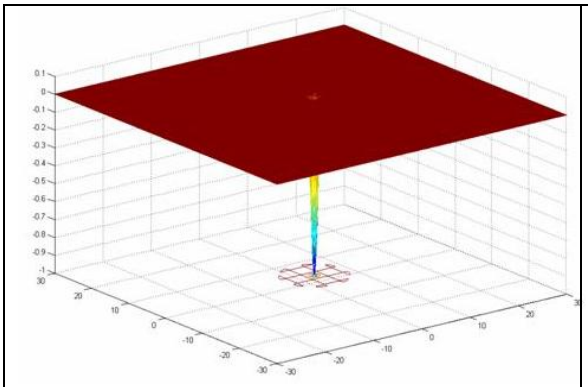
EK-5 50 Temel sayısal optimizasyon problemi üzerinde ABC, A-ABC algoritmalarının gerçek değerli performans karşılaştırması F: Fonksiyon, Min: Minimum, Ort: Ortalama, Sdt.S. : Standart Sapma, OSH: Ortalamanın standart hatası

		ABC			A-ABC		
F	Min.	Ort.	Std.S.	OSH	Ort.	Std.S.	OSH
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	4,53E-16	7,01E-17	1,28E-17	**4,2E-161**	9,1E-161	1,7E-161
4	0	4,43E-16	6E-17	1,09E-17	**8,29E-97**	4,15E-96	7,57E-97
5	0	0,031075	0,006537	0,001193	**0,008502**	0,002065	0,000377
6	0	8,75E-15	2,01E-14	3,66E-15	**4,98E-24**	2,49E-23	4,54E-24
7	-1	-1	1,05E-06	1,92E-07	-1	0	0
8	0	9,51E-16	1,03E-15	1,88E-16	**2,23E-50**	5,75E-50	1,05E-50
9	0	0,107723	0,063184	0,011536	**2,71E-06**	5,2E-06	9,5E-07
10	-50	-50	1,52E-13	2,78E-14	-50	3,41E-14	6,23E-15
11	-210	-210	1,29E-07	2,35E-08	-210	1,21E-05	2,2E-06
12	0	0,000219	0,000214	3,91E-05	**1,01E-17**	4,84E-17	8,83E-18
13	0	0,000359	2,83E-05	5,17E-06	**0,000152**	4E-05	7,3E-06
14	0	1,18E-15	6,81E-17	1,24E-17	**1,8E-103**	1,6E-103	3E-104
15	0	1330,481	506,9713	92,56031	**8,988612**	11,46585	2,093379
16	0	**0,136845**	0,190254	0,034736	8,964869	19,95737	3,643718
17	0	**3,95E-15**	2,46E-15	4,5E-16	0,481163	0,303618	0,055433
18	0.998	0,998004	8,88E-15	1,62E-15	0,998004	2,27E-16	4,14E-17
19	0.398	0,397887	0	0	0,397887	0	0
20	0	0	0	0	0	0	0
21	0	1,6E-18	1,75E-18	3,19E-19	**0**	0	0
22	0	**0**	0	0	9,09E-15	2,13E-14	3,88E-15
23	-12569.5	-12569,5	1,96E-12	3,59E-13	-12569,5	3,24E-12	5,91E-13
24	-1.8013	-1,8013	6,8E-16	1,24E-16	-1,8013	6,8E-16	1,24E-16
25	-4.6877	-4,68766	1,81E-15	3,31E-16	-4,68766	1,79E-15	3,26E-16
26	-9.6602	-9,66015	6,28E-16	1,15E-16	-9,66015	0	0
27	0	1,24E-09	4,81E-09	8,79E-10	**0**	0	0
28	-1.03163	-1,03163	6,72E-16	1,23E-16	-1,03163	6,8E-16	1,24E-16
29	0	0	0	0	0	0	0
30	0	2,19E-14	8,67E-14	1,58E-14	**0**	0	0
31	-186.73	-186,731	2,78E-14	5,08E-15	-186,731	3,71E-14	6,78E-15
32	3	3	2,45E-12	4,47E-13	3	9,29E-16	1,7E-16
33	3.0748e-4	0,000411	4,98E-05	9,1E-06	**0,000308**	2,54E-07	4,63E-08
34	-10.1532	-10,1532	3,72E-15	6,78E-16	-10,1532	4,8E-15	8,76E-16
35	-10.40294	-10,4029	3,75E-15	6,85E-16	-10,4029	2,97E-15	5,42E-16
36	-10.53641	-10,5364	1,92E-15	3,5E-16	-10,5364	3,52E-15	6,42E-16
37	0	0,036873	0,024744	0,004518	**0,001017**	0,000853	0,000156
38	0	0,004175	0,003082	0,000563	**0,000259**	0,00016	2,92E-05
39	-3.86	-3,86278	1,89E-15	3,46E-16	-3,86278	2,27E-15	4,14E-16

40	-3.32	-3,32237	7,25E-16	1,32E-16	-3,32237	2,56E-16	4,68E-17
41	0	7,11E-17	2E-16	3,65E-17	**0**	0	0
42	0	3,4E-14	3,8E-15	6,94E-16	**1,54E-14**	4,46E-15	8,14E-16
43	0	3,99E-16	8,35E-17	1,52E-17	**1,57E-32**	5,59E-48	1,02E-48
44	0	4,1E-16	7,78E-17	1,42E-17	**1,35E-32**	5,59E-48	1,02E-48
45	-1.08	-1,08094	5,07E-16	9,25E-17	-1,08094	6E-16	1,09E-16
46	-1.5	-1,5	1,52E-06	2,77E-07	-1,5	6,49E-16	1,18E-16
47	NA	-0,85828	0,194355	0,035484	**-1,23435**	0,334336	0,061041
48	0	2,11E-19	2,16E-19	3,95E-20	**0**	0	0
49	0	0,061056	0,06227	0,011369	**7,88E-05**	0,000243	4,44E-05
50	0	8,756695	6,373139	1,163576	**4,591389**	5,450342	0,995096

EK-6 Çalışmada Kullanılan Bazı Test Problemlerinin Grafikleri ve Karakteristikleri

Fonksiyonun Grafiği	**Fonksiyon Karakteristiği**
	**Sphere** Sınırlamasız test fonksiyonlarının en basitidir. Tek modlu, dış bükey ve sürekli bir fonksiyondur. Literatürde De Jong 1 fonksiyonu olarak da bilinir (de Jong,1975). Sphere fonksiyonunun 2 boyutlu uzayda grafiğiyandaki gibidir. Formül: $f(x)=\sum_{i=1}^{n} x_i^2$ Global En iyi Değer: *f(x)*=0.
	**SumSquares** Tek modlu ve ayrıştırılabilir bir fonksiyondur. Sumsquareas fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir. Formül: $f(x)=\sum_{i=1}^{n} i x_i^2$ Global En iyi Değer: *f(x)*=0.
	**Beale** Standart olarak 2 boyuttan oluşmaktadır. Tek modlu ve ayrıştırılamaz bir fonksiyondur. Beale fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir. Formül: $f(x)=\left(1.5-x_1+x_1x_2\right)^2+\left(2.25-x_1+x_1x_2^2\right)^2$ $+\left(2.625-x_1+x_1x_2^3\right)^2$ Global En iyi Değer: *f(x)*=0.



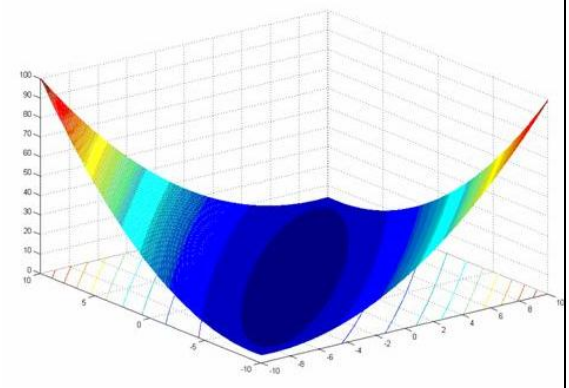
Easom

Standart olarak 2 boyuttan oluşmaktadır. Tek modlu ve ayrıştırılamaz bir fonksiyondur. Easom fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = -\cos(x_1)\cos(x_2)$$
$$\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$$

Global En iyi Değer: *f(x)*=-1.



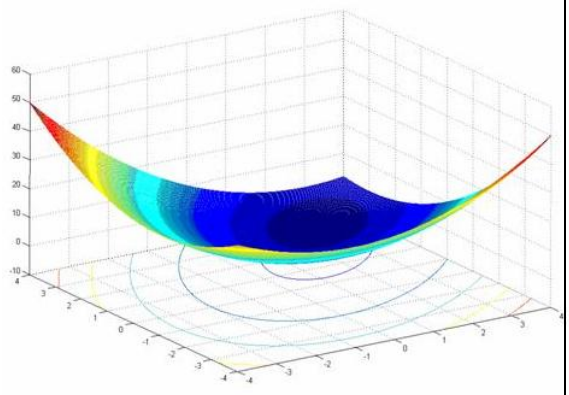
Matyas

Standart olarak 2 boyuttan oluşmaktadır. Tek modlu ve ayrıştırılamaz bir fonksiyondur. Matyas fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$$

Global En iyi Değer: *f(x)*=0



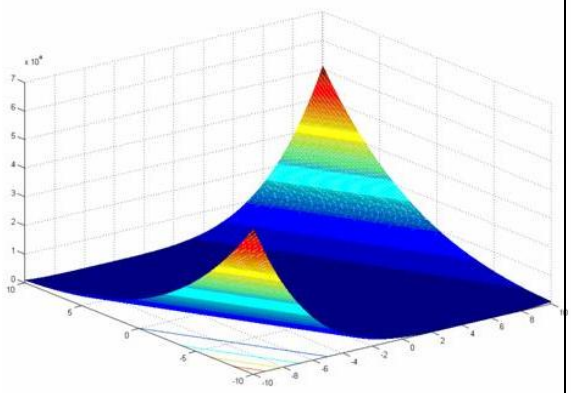
Trid6

Tek modlu ve ayrıştırılamaz bir fonksiyondur. Arama uzayı boyut sayısının karesine göre belirlenir. Trid fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$$

Global En iyi Değer:
n=6 ise *f(x)*=-50
n=10 ise *f(x)*=-210



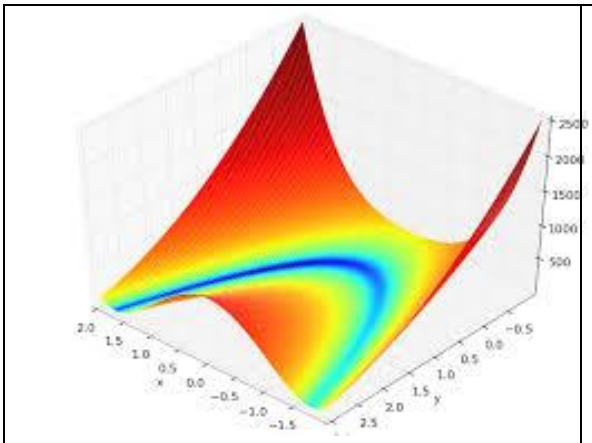
Zakharov

Tek modlu ve ayrıştırılabilir bir fonksiyondur. Zakharov fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$$

Global En iyi Değer: *f(x)*=0.



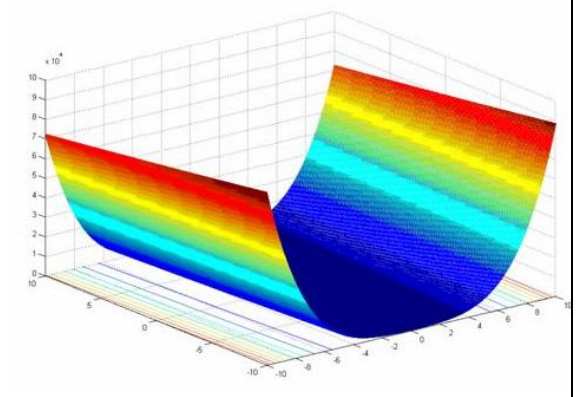
Rosenbrock

Tek modlu sürekli bir fonksiyondur. Şekli yassı iki vadiden oluştuğu için global en iyi değeri bulmak oldukça zordur. *Muz fonksiyonu* ya da *De Jong 2 fonksiyonu* olarak da bilinir. Rosenbrock fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$$

Global En iyi Değer: *f(x)*=0.



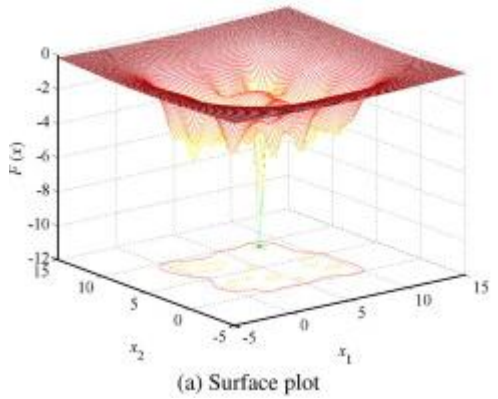
Dixon-Price

Tek modlu ve ayrıştırılamaz bir fonksiyondur. Dixon-price fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=\left(x_1-1\right)^2+\sum_{i=2}^{n} i\left(2x_i^2-x_{i-1}\right)^2$$

Global En iyi Değer: *f(x)*=0.



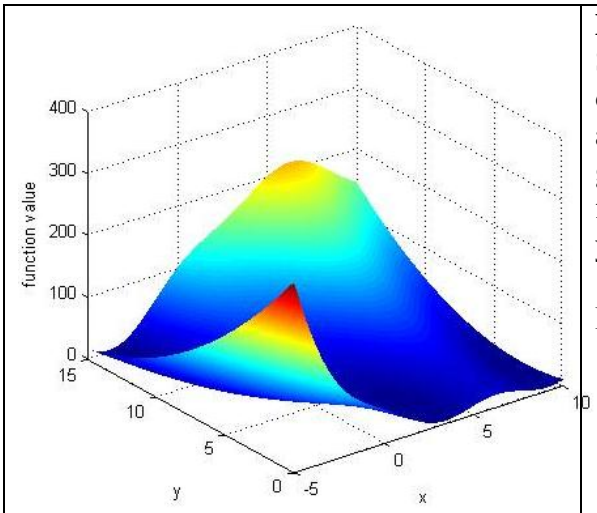
Foxholes

Çok modlu ve ayrıştırılabilir bir fonksiyondur. Foxholes fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right]^{-1}$$

Global En iyi Değer: *f(x)*=0.998



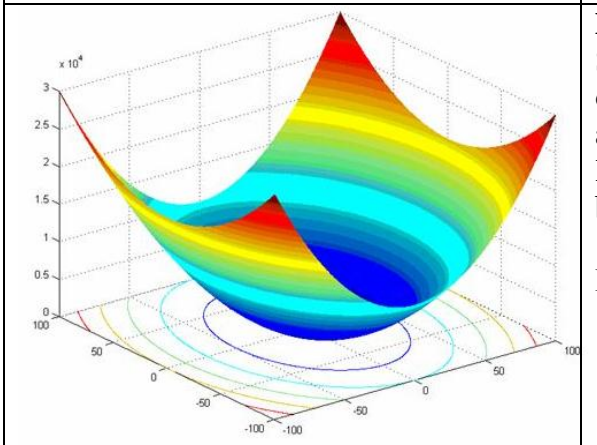
Branin

Standart olarak 2 boyuttan oluşmaktadır. Çok modlu ve ayrıştırılabilir bir fonksiyondur. 3 adet global minimuma sahiptir. Branin fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2$$
$$+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$

Global En iyi Değer: *f(x)*=0.398



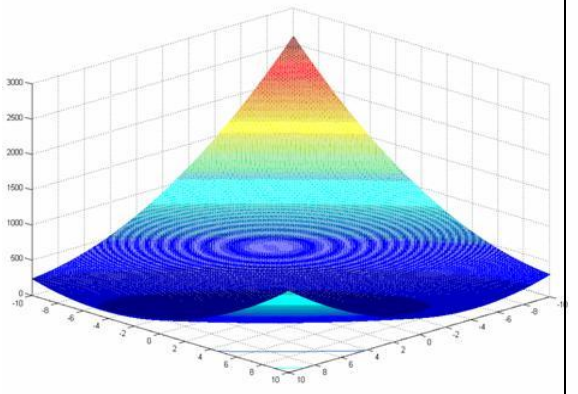
Bohachevsky1

Standart olarak 2 boyuttan oluşmaktadır. Çok modlu ve ayrıştırılabilir bir fonksiyondur. Bohachevsky1 fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)$$
$$-0.4\cos(4\pi x_2)+0.7$$

Global En iyi Değer: *f(x)*=0



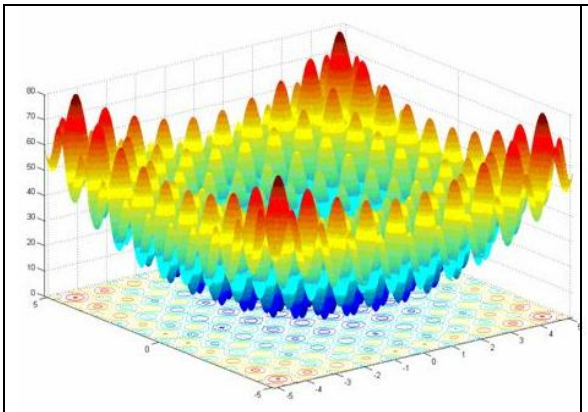
Booth

Standart olarak 2 boyuttan oluşmaktadır. Çok modlu ve ayrıştırılabilir bir fonksiyondur. Booth fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x)=\left(x_1+2x_2-7\right)^2+\left(2x_1+x_2-5\right)^2$$

Global En iyi Değer: *f(x)*=0.



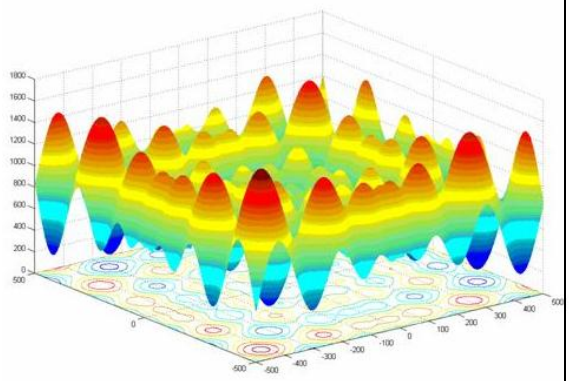
Griewank

Çok yüksek derecede çok modlu ve sürekli bir test fonksiyonudur. Griewank fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Global En iyi Değer: *f(x)*=0.



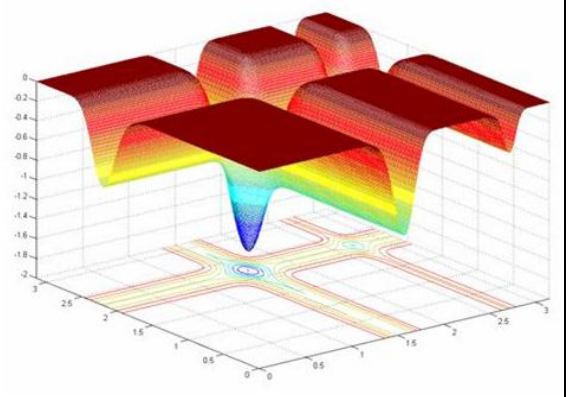
Schwefel

Çok modlu ve sürekli bir optimizasyon fonksiyonudur Global ve yerel optimum değerleri geometrik belirsizlikler üzerinde olup çözülmesi oldukça zor bir problemdir. Schwefel fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{x_i}\right)$$

Global En iyi Değer: *f(x)*= -418.9829*n



Michalewicz2

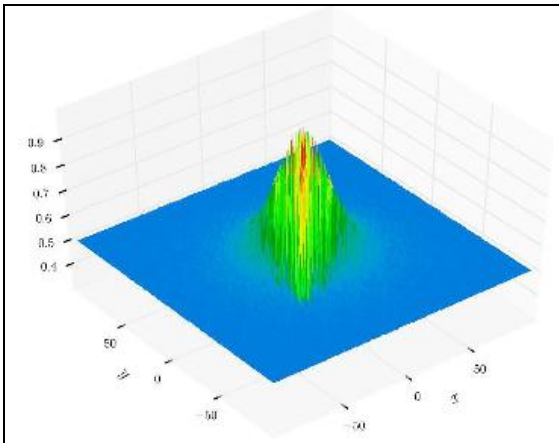
Literatürde standart olarak 2, 5 ve 10 boyutlu olarak bilinmektedir. Çok modlu ve ayrıştırılabilir bir fonksiyondur. Michalewicz2 fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(ix_i^2 / \pi))^{2m}$$

$m = 10$

n=2 için,
Global En iyi Değer: *f(x)*= -1.8013



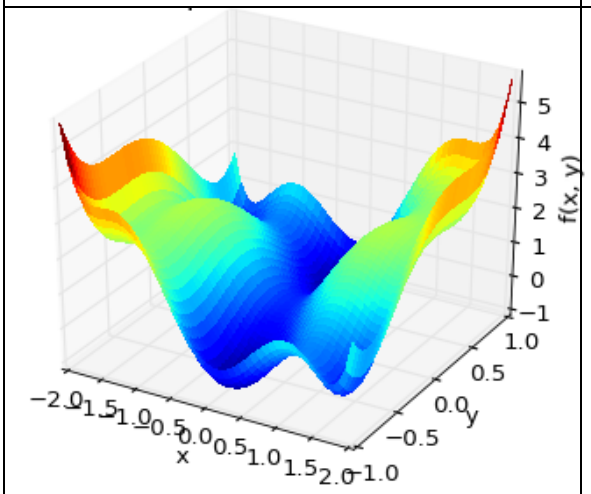
Schaffer

Standart olarak 2 boyuttan oluşmaktadır. Çok modlu ve ayrıştırılamaz bir fonksiyondur. Schaffer fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$$

Global En iyi Değer: *f(x)*= 0



Six Hump Camel Back

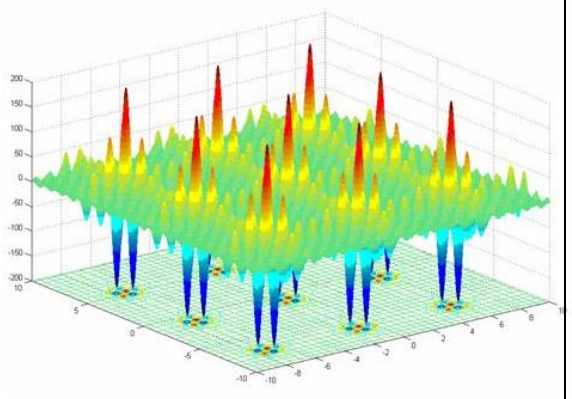
Standart olarak 2 boyuttan oluşan ayrıştırılamaz bir fonksiyondur. 2 farklı global optimuma sahiptir. Fonksiyonun gafiği deve sırtına benzediği için bu ismi almıştır. Six hump camel back fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x1x2$$

$$-4x_2^2 + 4x_2^4$$

Global En iyi Değer: *f(x)*= -1.03163



Shubert

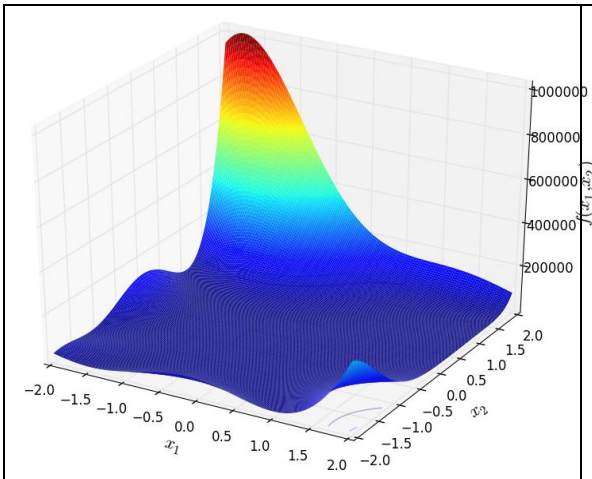
Standart olarak 2 boyuttan oluşmaktadır. Çok boyutlu ve ayrıştırılamaz bir fonksiyondur. 760 yerel minimuma, 18 global minimuma sahiptir. Shubert fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)$$

$$\left(\sum_{i=1}^{5} i\cos\left(i+1\right)x_2 + i\right)$$

Global En iyi Değer: *f(x)*= -186.73



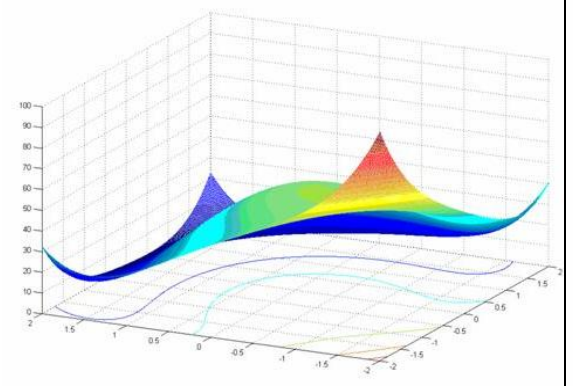
GoldStein-Price

Çok modlu ve ayrıştırılamaz bir optimizasyon fonksiyonudur. 4 yerel minimuma sahiptir. Goldstein-Price fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \left[1+(x_1+x_2+1)^2\right]\left[\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\left[30+(2x_1-3x_2)^2\right]\left[\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$$

Global En iyi Değer: *f(x)*= 3



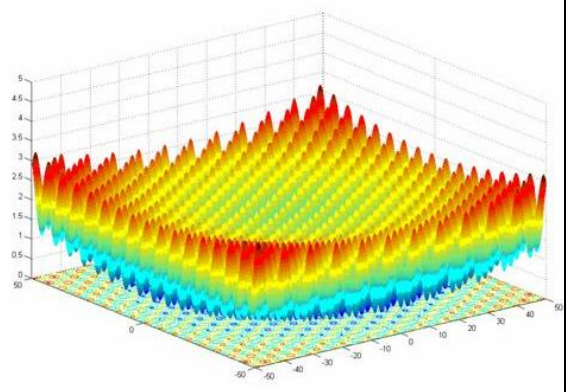
Perm

Çok modlu ve ayrıştırılamaz bir fonksiyondur. Arama uzayı boyut sayısına bağlı olarak belirlenir. Perm fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k+\beta\right)\left(\left(x_i\ / i\right)^k-1\right)\right]^2$$

Global En iyi Değer: *f(x)*=0.



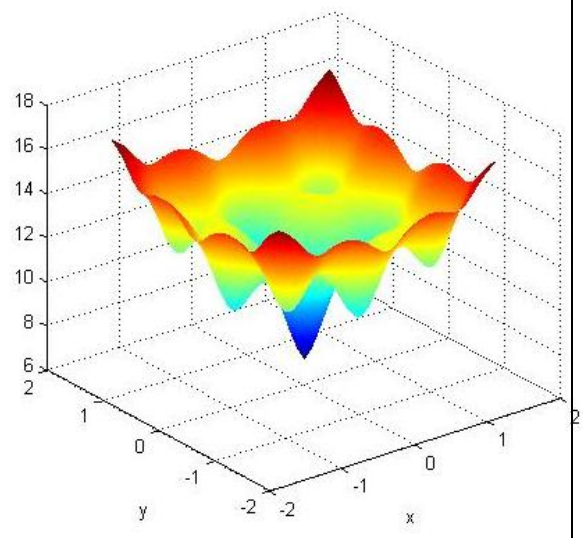
Rastrigin

Çok yüksek derecede çok modlu ve sürekli bir test fonksiyonudur. Rastrigin fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = \sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$$

Global En iyi Değer: *f(x)*=0.



Ackley

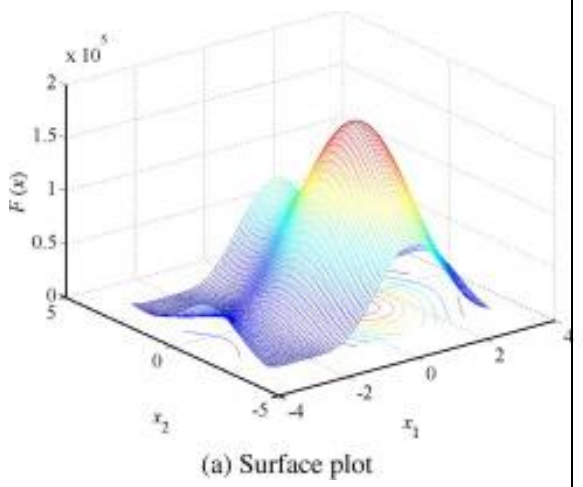
Çok modlu ve sürekli bir optimizasyon fonksiyonudur. Global ve yerel optimum değerleri geometrik belirsizlikler üzerinde olup, çözülmesi oldukça zor bir fonksiyondur. Ackley fonksiyonunun 2 boyutlu uzayda grafiği yandaki gibidir.

Formül:

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)$$

$$-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$$

Global En iyi Değer: *f(x)*=0.



(a) Surface plot

Fletcher-Powell2

Standart olarak 2 boyuttan oluşmaktadır. Çok modlu ve ayrıştırılamaz bir fonksiyondur. Yerel minimum sayısı bilinmemektedir.

Formül:

$$f(x) = \sum_{i=1}^{n}\left(A_i - B_i\right)^2$$

$$A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right),$$

$$B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$$

Global En iyi Değer: *f(x)*=0.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Ahmet ÖZKIŞ
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Konya/ 26.03.1986
Telefon	**:**	0506 619 33 68
e-mail	**:**	ahmetozkis@selcuk.edu.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	**:**	Dolapoğlu Anadolu Lisesi, Selçuklu, Konya	2004
Üniversite	**:**	Selçuk Ünv. TEF Bilgisayar Sis. Öğr., Selçuklu, Konya	2010
Yüksek Lisans:		Selçuk Ünv. Bilgisayar Mühendisliği A.B.D., Selçuklu, Konya	2011-2013

Yıl	**Kurum**	**Görevi**
2011	Selçuk Ünv. Bilgisayar Mühendisliği Bölümü	**Arş. Gör.**
2011	Balıkesir Ünv. Bilgisayar Mühendisliği Bölümü	**Arş. Gör.**
2010	Çubuk Endüstri Meslek Lisesi	**Bilişim Tek. Öğr.**

UZMANLIK ALANI: Optimizasyon, sürü zekâsı, yapay zekâ teknikleri.

YABANCI DİL: İngilizce

YAYINLAR

1 – *Özkış A. ve Babalık A., "Accelerated ABC (A-ABC) Algorithm for Continuous Optimization Problems", International Conference on Software Computere Applications ICSCA-2013,1-2 Haziran 2013, Paris, Fransa.

2 – Babalık A., Babaoğlu, İ., Özkış A., "A Pre-Processing Approach Based on Artificial Bee Colony for Classification by Support Vector Machine", 2012 International Conference on Sorfware and Computere Engineering ICSCE-2012, 6-7 Ekim2012, Singapur.

* Yüksek lisans tez çalışmasından üretilmiştir.