



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKÂ TABANLI OPTİMİZASYON ALGORİTMALARI GELİŞTİRİLMESİ

Utku KÖSE

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Temmuz-2017
KONYA
Her Hakkı Saklıdır

ÖZET

DOKTORA TEZİ

YAPAY ZEKÂ TABANLI OPTİMİZASYON ALGORİTMALARI GELİŞTİRİLMESİ

Utku KÖSE

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ahmet ARSLAN

2017, 186 Sayfa

Jüri
Prof. Dr. Şirzat KAHRAMANLI
Prof. Dr. Ahmet ARSLAN
Doç. Dr. Harun UĞUZ
Doç. Dr. Mustafa Servet KIRAN
Yrd. Doç. Dr. Ömer Kaan BAYKAN

Yapay Zekâ (YZ), hiç kuşkusuz ki Bilgisayar Bilimleri'nin en önemli ilgi alanlarından birisidir. İnsan düşünce - davranış şekilleri ve doğa dinamiklerinin benzetimi neticesinde sunduğu etkin ve tutarlı çözümler nedeniyle hızlı bir şekilde gelişmiş ve modern yaşamın neredeyse her alanında yerini alarak gücünü kanıtlamıştır. Çok-disiplinli uygulanabilir olma özelliği, aynı zamanda farklı problem çeşitlerine kolaylıkla adapte edilebilmesiyle ilgilidir. Bu durum, YZ'nin kendi içerisinde alt-araştırma alanlarına ayrılmasına da sebep olmuştur.

Optimizasyon, YZ'nin yoğun bir şekilde ilgili olduğu problem alanlarından birisidir. Genel olarak, eldeki kaynaklar bağlamında 'optimumu – en uygunu bulma' çabaları olarak tanımlayabileceğimiz optimizasyon, klasik tekniklerin yetersiz kalmaya başlaması sonrasında, çözümü YZ bünyesinde bulmuştur. Elde edilen başarılı sonuçlar, optimizasyon problemleri için kullanılan tekniklerin tasarlanmasına sebep olmuş ve sonuç olarak ilgili literatürde, YZ tabanlı optimizasyon tekniklerine yönelik bir alt-araştırma ortamı ortaya çıkmıştır.

YZ tabanlı optimizasyon teknikleri tipik olarak çeşitli mantıksal ve matematiksel çözüm yaklaşımları çerçevesinde şekillenen ve çoğunlukla doğal dinamiklerden esinlenerek tasarlanan algoritmalar olarak bilinmektedir. Bu tür algoritmalara yönelik literatür özellikle son yıllarda büyük bir ivme kazanmış ve bilim insanlarının ilgi alanları içerisine girmiştir.

Bu tez çalışmasının amacı, daha kolay kodlanabilen, alternatif YZ tabanlı optimizasyon algoritmaları geliştirmektir. Bu bağlamda, arkaplanı anlamak adına öncelikli olarak 'YZ ve optimizasyon' konusuna yönelik temel bilgiler sunulmuş ve literatürde ön plana çıkan bazı algoritmalardan bahsedilmiş, sonrasında ise, iki farklı sürekli optimizasyon algoritması geliştirilmiştir. Sırasıyla Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) adı verilen algoritmalar, Sürü Zekâsı adı altında incelenebilen çeşitli mekanizmaları ve basit matematiksel süreçleri içermektedir. Geliştirilen algoritmalar, çeşitli değerlendirme süreçlerinden geçirilmiş ve elde edilen bulgular, algoritmaların zeki optimizasyonda yeter düzeyde başarılı olduklarını göstermiştir.

Anahtar Kelimeler: Yapay Zekâ, Optimizasyon, Zeki Optimizasyon, Girdap Optimizasyon Algoritması, Bilişsel Gelişim Optimizasyon Algoritması, Sürekli Optimizasyon

ABSTRACT

Ph.D THESIS

DEVELOPMENT OF ARTIFICIAL INTELLIGENCE BASED OPTIMIZATION ALGORITHMS

Utku KÖSE

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

Advisor: Prof. Dr. Ahmet ARSLAN

2017, 186 Pages

Jury
Prof. Dr. Şirzat KAHRAMANLI
Prof. Dr. Ahmet ARSLAN
Assoc. Prof. Dr. Harun UĞUZ
Assoc. Prof. Dr. Mustafa Servet KIRAN
Assist. Prof. Dr. Ömer Kaan BAYKAN

It is clear that Artificial Intelligence (AI) is one of the most important interest areas of Computer Science. Because of its effective and efficient solutions provided as a result of efforts on simulating human thinking - behaviors and nature dynamics, it has improved rapidly and proved its power by taking place within almost all fields of the modern life. This feature of being applicable multidisciplinary is related to being also easily adaptable to different types of problems. This situation has also caused AI to be divided into sub-research fields.

Optimization is one of the related problems to which the AI is related intensely. Optimization, which we can define generally as the efforts on 'finding the most appropriate – optimum one' in the sense of the sources in hand, has found its way in AI after classical techniques have become unsatisfactory on solutions. Obtained successful results have caused designing techniques used for optimization problems and as a result, a sub-research environment related to AI based optimization techniques has appeared.

Typically, AI based optimization techniques are known as the algorithms formed in the context of various logical and mathematical solution approaches and mostly designed by inspiring from natural dynamics. The literature associated with such algorithms has gained momentum in especially recent years and taken part in scientists' interest areas.

Objective of this thesis study is to develop alternative AI based optimization algorithms, which can be coded easier. In this sense, in order to understand the background, essential information regarding to the subject of 'AI and optimization' was given, and then some algorithms that come into prominence in the literature were explained primarily, and after that, two different continuous optimization algorithms were developed. Called as Vortex Optimization Algorithm (VOA), and Cognitive Development Optimization Algorithm (CoDOA), the related algorithms include various mechanism and some additional simple mathematical processes that can be examined in the context of Swarm Intelligence. The developed algorithms were employed in some evaluation processes and the findings showed that they are successful enough in intelligent optimization processes.

Keywords: Artificial Intelligence, Optimization, Intelligent Optimization, Vortex Optimization Algorithm, Cognitive Development Optimization Algorithm, Continuous Optimization

ÖNSÖZ

Yapay Zekâ (YZ), farklı disiplinler kapsamında ve günlük yaşamımızda karşılaştığımız gerçek dünya temelli problemler için ürettiği 'zeki' çözümler sebebiyle popülaritesini her geçen gün giderek artırmaktadır. Geleceğin başlıca bilim alanlarından olan YZ, kendi içerisinde farklı çözüm kapsamlarına bölünerek, bilim dünyasına katkı sağlamaya devam etmektedir. Bu tez çalışmasında, optimizasyon problemlerine yönelik zeki çözümlere, daha kolay kodlanabilen alternatifler sunmak ve ilgili literatürün gelişmesine katkı sağlamak adına iki farklı YZ tabanlı sürekli optimizasyon algoritması geliştirilmiştir.

Doktora çalışmalarım süresince gösterdiği sabır ve özveri ile desteğini ve engin bilgilerini benden hiç bir zaman eksik etmeyen, çalışmalarımın her aşamasında önerileri ile beni yönlendiren değerli hocam ve danışmanım Sayın Prof. Dr. Ahmet ARSLAN'a, önerilerini ve bilgilerini benimle paylaşan meslektaşlarım; değerli hocam Sayın Prof. Dr. Ömer DEPERLİOĞLU ve Yrd. Doç. Dr. Gür Emre GÜRAKSIN'a ve son olarak, bugüne kadar yılmadan ve usanmadan her zaman yanımda olan sevgili ailem; Zafer Yılmaz KÖSE, Şaduman KÖSE ve Umut KÖSE ve Gamze KÖSE'ye sonsuz teşekkürlerimi ve şükranlarımı sunarım.

Utku KÖSE
KONYA-2017

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

SİMGELER VE KISALTMALAR ..... **x**

ŞEKİLLER DİZİNİ ..... **xiii**

RESİMLER DİZİNİ ..... **xiv**

ÇİZELGELER DİZİNİ ..... **xv**

1. GİRİŞ **1**

2. KAYNAK ARAŞTIRMASI **8**

2.1. Küresel Optimizasyon **8**
2.1.1. Sürekli optimizasyon 8
2.1.2. Sürekli optimizasyonun matematiksel ifadesi 11
2.2. Yapay Zekâ ve Optimizasyon **12**
2.2.1. Zeki optimizasyon kavramı 15
2.2.2. Zeki optimizasyonda temel yaklaşımlar 15
2.3. Zeki Optimizasyon Tekniklerinin Sınıflandırılması **18**
2.4. Sürü Zekâsı **21**
2.4.1. Sürü zekâsı tekniklerinde temel kabuller 22
2.4.2. Sürü zekâsı tekniklerinde alternatif durumlar 23
2.5. Başlıca Sürü Zekâsı Teknikleri – Algoritmaları **25**
2.5.1. Bakteriyel yiyecek arama optimizasyonu 26
2.5.2. Çiçek tozlaşma optimizasyonu 30
2.5.3. Diferansiyel evrim 33
2.5.4. Emperyalist rekabetçi algoritma 35
2.5.5. Genetik algoritma 39
2.5.6. Guguk kuşu araması 42
2.5.7. Klonal seçilim algoritması 46
2.5.8. Parçacık sürü optimizasyonu 48
2.5.9. Yapay arı kolonisi 51

3. MATERYAL VE YÖNTEM **55**

3.1. Tasarım – Geliştirme İlke Yaklaşım ve Ortamları **55**
3.1.1. Tasarım – geliştirme ilkeleri 55
3.1.2. Nesne yönelimli programlama 56
3.1.3. Geliştirme ve değerlendirme ortamları 59

3.2. Tasarım – Geliştirme Süreci 61
3.3. Test Fonksiyonları 62
3.3.1. Beale fonksiyonu 65
3.3.2. Booth fonksiyonu 66
3.3.3. Bukin (no. 6) fonksiyonu 67
3.3.4. Cross-in-tray fonksiyonu 68
3.3.5. Easom fonksiyonu 69
3.3.6. Eggholder fonksiyonu 70
3.3.7. Levy (no. 13) fonksiyonu 71
3.3.8. Matyas fonksiyonu 72
3.3.9. McCormick fonksiyonu 73
3.3.10. Schaffer (no. 2) fonksiyonu 74
3.3.11. Ackley fonksiyonu 75
3.3.12. Rastrigin fonksiyonu 76
3.3.13. Rosenbrock fonksiyonu 77
3.3.14. Sphere (küre) fonksiyonu 78
3.3.15. Styblinski–Tang fonksiyonu 79
3.3.16. Mishra'nın kuşu (Mishra's bird) fonksiyonu 80
3.3.17. Simionescu fonksiyonu 81
3.3.18. Townsend fonksiyonu 82
3.3.19. Döndürülmüş Cigar (rotated Cigar) fonksiyonu 83
3.3.20. Döndürülmüş yüksek kond. eliptik (rotated high cond. ellipt.) fonk. 84
3.3.21. Kaydırılmış ve dönd. Ackley (shifted and rotated Ackley) fonk. 85
3.3.22. Kaydırılmış ve dönd. Rastrigin (shifted and rotated Rastrigin) fonk. 86
3.3.23. Kaydırılmış ve dönd. Schwefel (shifted and rotated Schwefel) fonk. 87
3.3.24. Fonseca ve Fleming fonksiyonu 88
3.3.25. Kursawe fonksiyonu 89
3.3.26. Zitzler-Deb-Thiele (no. 1) fonksiyonu 90
3.3.27. Zitzler-Deb-Thiele (no. 3) fonksiyonu 91
3.3.28. Binh ve Korn fonksiyonu 92
3.3.29. Chakong ve Haimes fonksiyonu 93
3.3.30. Osyczka ve Kundu fonksiyonu 94
3.4. Gerçek Yaşam Tabanlı Problemler ve Uygulamalar 95
3.4.1. Sürekli optimizasyon odaklı problemler ve uygulamalar 95
3.5. Hibrit Kullanıma Dair Değerlendirmeler 98
3.5.1. Yapay sinir ağları ve hibrit kullanım 99
3.5.2. Adaptif sinirsel-bulanık çıkarım sistemi ve hibrit kullanım 100
3.5.3. Destek vektör makineleri ve hibrit kullanım 100
3.6. Karşılaştırmaya Tabi Algoritmalar – Teknikler 102
3.7. Değerlendirme Kuralları ve Yöntemleri 103

4. YAPAY ZEKÂ TABANLI OPTİMİZASYON İÇİN GELİŞTİRİLEN ALGORİTMALAR 108

4.1. Girdap Optimizasyon Algoritması (GOA) 108
4.1.1. Girdap optimizasyon algoritması kapsamında çözüm adımları 109
4.2. Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) 113
4.2.1. Bilişsel gelişim optimizasyon algoritması kapsamında çözüm adımları 115
4.3. GOA ve BiGOA Çok-Amaçlı Optimizasyon Revizyonu 120
4.4. Algoritmaların Karmaşıklık Analizi 120

5. DEĞERLENDİRME VE TARTIŞMA 122

5.1. Test Fonksiyonlarına Yönelik Bulgular 122
5.2. Gerçek Yaşam Tabanlı Problem ve Uygulamalara Yönelik Bulgular 138
5.3. Hibrit Kullanıma Yönelik Bulgular 143
5.4. Tartışma 153

6. SONUÇLAR – ÖNERİLER VE GELECEK ÇALIŞMALAR 156

6.1. Sonuçlar 156
6.2. Öneriler 157
6.3. Gelecek Çalışmalar 158

KAYNAKLAR 160

ÖZGEÇMİŞ 184

SİMGELER VE KISALTMALAR

Simgeler

$<$	**:** Küçüktür
$>$	**:** Büyüktür
$\leq$	**:** Küçük eşittir
$\geq$	**:** Büyük eşittir
$=$	**:** Eşittir
$\neq$	**:** Eşit değildir
$\boldsymbol{\Sigma}$	**:** Toplam (sembolü)
$\prod$	**:** Çarpım (sembolü)
*	**:** Çarpım
$\times$	**:** Çarpım
.	**:** Çarpım
/	**:** Bölüm
+	**:** Toplam
$\gg$	**:** Çok daha büyük
π	**:** Pi sayısı
Δ	**:** Değişim
sin	**:** Sinüs
cos	**:** Kosinüs
dsax	**:** Deneyim değeri 0'ın altında parçacık [İlgi: *Bkz.* BiGOA (Kısalt.)]
dsüx	**:** Deneyim değeri 0'ın üstünde veya ona eşit parçacık [İlgi: *Bkz.* BiGOA (Kısalt.)]
f(…)	**:** (Amaç) fonksiyon
F(…)	**:** Kapsayıcı – genel (amaç) fonksiyon
g	**:** Geçerli (değer) – üst simge olarak
go	**:** Geçiş olasılığı [İlgi: *Bkz.* ÇTO (Kısalt.)]
gx	**:** Girdap statüsündeki parçacık [İlgi: *Bkz.* GOA (Kısalt.)]
g(…)	**:** (Amaç – Kısıt) Fonksiyon
h(…)	**:** (Amaç – Kısıt) Fonksiyon
i, j, k, l, m, n	**:** Parçacık, Değişken ya da Düğüm sırası – alt-simge olarak
j(…)	**:** (Amaç) fonksiyon
Levy	**:** Levy Uçuşları dağılımı
max{…}	**:** En yüksek değer
n	**:** Fonksiyonda boyut – değişken sayısı – fonksiyon sayısı
N	**:** Parçacık sayısı
ofix	**:** Amaç fonksiyon değeri (fitness) ortalamadan iyi parçacık [İlgi: *Bkz.* BiGOA (Kısalt.)]
olix	**:** Olgunlaşma limiti bağlamında daha iyi parçacık [İlgi: *Bkz.* BiGOA (Kısalt.)]
ort{…}	**:** Ortalama değer
tp	**:** Test – problem – alt-simge olarak

x **:** Parçacık / Değişken
y **:** Yeni (değer) – üst simge olarak / Değişken
z **:** Zaman – üst simge olarak

Kısaltmalar

ABC **:** Artificial Bee Colony (*İngilizce*) – YAK
ACO **:** Ant Colony Optimization (*İngilizce*) – KKO
AI **:** Artificial Intelligence (*İngilizce)* – Yapay Zekâ
AIS **:** Artificial Immune Systems (*İngilizce)* – Yapay Bağışıklık Sistemleri
ANFIS **:** Adaptive Neuro-Fuzzy Inference System (*İngilizce*) – ASBÇS
ANN **:** Artificial Neural Networks (*İngilizce*) – YSA
ASBÇS **:** Adaptif Sinirsel-Bulanık Çıkarım Sistemi
BYAO **:** Bakteriyel Yiyecek Arama Optimizasyonu
Bkz. **:** Bakınız
BFO **:** Bacterial Foraging Optimization (*İngilizce*) – BYAO
BIC **:** Bio-Inspired Computing (*İngilizce*) – BEH
BiGOA **:** Bilişsel Gelişim Optimizasyon Algoritması
BEH **:** Biyo-Esinli Hesaplama
BM **:** Bulanık Mantık
CoDOA **:** Cognitive Development Optimization Algorithm (*İngilizce)* – BiGOA
CS **:** Cuckoo Search (*İngilizce*) – GKA
CSA **:** Clonal Selection Algorithm (*İngilizce*) – KSA
ÇTO **:** Çiçek Tozlaşma Optimizasyonu
DEH **:** Doğa-Esinli Hesaplama
DE **:** Diferansiyel Evrim – Differential Evolution (*İngilizce*)
EC **:** Evolutionary Computing (*İngilizce*) – EH
ERA **:** Emperyalist Rekabetçi Algoritma – ICA
EH **:** Evrimsel Hesaplama
FL **:** Fuzzy Logic (*İngilizce*) – BM
FPO **:** Flower Pollination Optimization (*İngilizce*) – ÇTO
GA **:** Genetik Algoritma – Genetic Algorithm (*İngilizce*)
GKA **:** Guguk Kuşu Araması
GOA **:** Girdap Optimizasyon Algoritması
GPSD **:** Genel Performans Sıra Değeri
ICA **:** Imperialist Competetive Algorithm (*İngilizce*) – ERA
IWDs **:** Intelligent Water Drops Algorithm (*İngilizce*) – ZSD
KKO **:** Karınca Koloni Optimizasyonu
KSA **:** Klonal Seçilim Algoritması
NIC **:** Nature-Inspired Computing (*İngilizce*) – DEH
NYP **:** Nesne Yönelimli Programlama
OOP **:** Object Oriented Programming (*İngilizce*) – NYP

PSO : Parçacık Sürü Optimizasyonu – Particle Swarm Optimization (*İngilizce*)
SI : Swarm Intelligence (*İngilizce*) – SZ
SZ : Sürü Zekâsı
VOA : Vortex Optimization Algorithm (*İngilizce)* – GOA
YAK : Yapay Arı Kolonisi
YBS : Yapay Bağışıklık Sistemleri
YSA : Yapay Sinir Ağları
YZ : Yapay Zekâ
ZSD : Zeki Su Damlaları Algoritması

ŞEKİLLER DİZİNİ

Şekil 2. 1. Çözüm uzayında yerel ve küresel optimumlar (Weise, 2009: 26). 9
Şekil 2. 2. Sürekli optimizasyonda temel unsurlar.....10
Şekil 2. 3. YZ'nin optimizasyon için tercih edilmesine sebep başlıca durumlar.....14
Şekil 2. 4. Zeki optimizasyonda sıklıkla başvurulan temel yaklaşımlar. 18
Şekil 2. 5. SZ tabanlı bir sistemde temel özellik ve işlevler.....22
Şekil 2. 6. BYAO'ya esin kaynağı Koli Basili hareketleri (Egbert ve ark., 2010). 29
Şekil 2. 7. ÇTO'ya esin kaynağı tozlaşma süreçleri (AMRITA Learning, 2013). 32
Şekil 2. 8. DE algoritması genel işleyişi (Datta ve Kumar, 2015). 34
Şekil 2. 9. ERA'da örnek emperyalist ve koloni dağılımı (Sun ve Liang, 2014). 38
Şekil 2. 10. GA'da kritik evrimsel süreçler (Cheng Yu, 2017). 41
Şekil 2. 11. GKA tekniğinde guguk kuşları davranışları (Djalal, 2005). 45
Şekil 2. 12. KSA'ya esin kaynağı seçilim süreci (Bachmayer, 2008).....48
Şekil 2. 13. PSO'da parçacık hareketlerinin özeti (Bryson ve ark., 2016). 50
Şekil 2. 14. YAK'ta arı hareketleri genel yapısı (Karaboğa, 2014). 54

Şekil 3. 1. Optimizasyon algoritmalarının tasarım – geliştirme sürecinde ilkeler. 56
Şekil 3. 2. Tipik bir SZ tekniğinde NYP ile ilişkilendirilebilen unsurlar. 57
Şekil 3. 3. Tasarım – Geliştirme sürecinin kısa bir özeti. 62

Şekil 4. 1. GOA – genel işleyiş yaklaşımına yönelik bir illüstrasyon. 111
Şekil 4. 2. GOA özet akış şeması. 112
Şekil 4. 3. BiGOA özet akış şeması. 119

Şekil 5. 1. Tek-amaç. çok-boy. önemli f. en iyi sonuç veren ilk üç alg. yak. eğrileri...127
Şekil 5. 2. 7 çok-amaçlı fonksiyonda en iyi üç algoritmanın pareto optimumları. 136
Şekil 5. 3. Test fonk. kapsamında değerlendirilen algoritmaların GPSD grafiği. 138
Şekil 5. 4. Disk Freni Tasarımı prob. – en iyi üç alg. elde edilen pareto optimumlar. .140

RESİMLER DİZİNİ

Resim 3. 1. Beale Fonksiyonu (Wikipedia, 2017i). 65
Resim 3. 2. Booth Fonksiyonu (Wikipedia, 2017i)....66
Resim 3. 3. Bukin (no. 6) Fonksiyonu (Wikipedia, 2017i). 67
Resim 3. 4. Cross-in-Tray Fonksiyonu (Wikipedia, 2017i). 68
Resim 3. 5. Easom Fonksiyonu (Wikipedia, 2017i). 69
Resim 3. 6. Eggholder Fonksiyonu (Wikipedia, 2017i). 70
Resim 3. 7. Levy (no. 13) Fonksiyonu (Wikipedia, 2017i). 71
Resim 3. 8. Matyas Fonksiyonu (Wikipedia, 2017i)....72
Resim 3. 9. McCormick Fonksiyonu (Wikipedia, 2017i). 73
Resim 3. 10. Schaffer (no. 2) Fonksiyonu (Wikipedia, 2017i). 74
Resim 3. 11. Ackley Fonksiyonu (Wikipedia, 2017i). 75
Resim 3. 12. Rastrigin Fonksiyonu (Wikipedia, 2017g). 76
Resim 3. 13. Rosenbrock Fonksiyonu (Wikipedia, 2017h; 2017i). 77
Resim 3. 14. Sphere (Küre) Fonksiyonu (Wikipedia, 2017i). 78
Resim 3. 15. Styblinski–Tang Fonksiyonu (Wikipedia, 2017i)....79
Resim 3. 16. Mishra'nın Kuşu (Mishra's Bird) Fonksiyonu (Wikipedia, 2017i). 80
Resim 3. 17. Simionescu Fonksiyonu (Wikipedia, 2017i). 81
Resim 3. 18. Townsend Fonksiyonu (Wikipedia, 2017i). 82
Resim 3. 19. Döndürülmüş Cigar (Rotated Cigar) Fonk. (Liang ve ark., 2014). 83
Resim 3. 20. Dönd. Yük. Kond. Eliptik (Rot. H. C. Ellipt.) F. (Liang ve ark., 2014). ...84
Resim 3. 21. Kayd. ve Dönd. Ackley (Shift. Rot. Ackley) F. (Liang ve ark., 2014). 85
Resim 3. 22. Kayd. ve Dönd. Rastrigin (Shift. Rot. Rast.) F. (Liang ve ark., 2014). 86
Resim 3. 23. Kayd. ve Dönd. Schwefel (Shift. Rot. Schw.) F. (Liang ve ark., 2014). ...87
Resim 3. 24. Fonseca ve Fleming Fonksiyonu (Wikipedia, 2017i). 88
Resim 3. 25. Kursawe Fonksiyonu (Wikipedia, 2017i). 89
Resim 3. 26. Zitzler-Deb-Thiele (no. 1) Fonksiyonu (Wikipedia, 2017i). 90
Resim 3. 27. Zitzler-Deb-Thiele (no. 3) Fonksiyonu (Wikipedia, 2017i). 91
Resim 3. 28. Binh ve Korn Fonksiyonu (Wikipedia, 2017i). 92
Resim 3. 29. Chakong ve Haimes Fonksiyonu (Wikipedia, 2017i). 93
Resim 3. 30. Osyczka ve Kundu Fonksiyonu (Wikipedia, 2017i). 94
Resim 3. 31. Disk Freni Tasarım problemi pareto optimumlar (Yang ve Deb, 2013)....97

Resim 5. 1. Birinci yeniden görüntü oluşturma probleminde GOA çözüm süreci. 141
Resim 5. 2. Birinci yeniden görüntü oluşturma probleminde BiGOA çözüm süreci....141
Resim 5. 3. İkinci yeniden görüntü oluş. prob. GOA ve BiGOA çözüm süreçleri. 142
Resim 5. 4. YSA-BiGOA 1. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c)....146
Resim 5. 5. YSA-BiGOA 2. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c)....146
Resim 5. 6. YSA-BiGOA 3. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c)....147
Resim 5. 7. YSA-BiGOA 4. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c)....147
Resim 5. 8. ASBÇS-GOA 5. kaotik zaman serisi. tahmini (Köse ve Arslan, 2016b)...148
Resim 5. 9. ASBÇS-GOA 6. kaotik zaman serisi. tahmini (Köse ve Arslan, 2016b)...148
Resim 5. 10. ASBÇS-GOA 7. kaotik zaman seri. tahmini (Köse ve Arslan, 2016b). ..149
Resim 5. 11. ASBÇS-GOA 8. kaotik zaman seri. tahmini (Köse ve Arslan, 2016b). ..149

ÇİZELGELER DİZİNİ

Çizelge 3. 1. Beale Fonksiyonu küresel min. ve arama alanı. 65
Çizelge 3. 2. Booth Fonksiyonu küresel min. ve arama alanı. 66
Çizelge 3. 3. Bukin (no. 6) Fonksiyonu küresel min. ve arama alanı. 67
Çizelge 3. 4. Cross-in-Tray Fonksiyonu küresel min. ve arama alanı. 68
Çizelge 3. 5. Easom Fonksiyonu küresel min. ve arama alanı. 69
Çizelge 3. 6. Eggholder Fonksiyonu küresel min. ve arama alanı. 70
Çizelge 3. 7. Levy (no. 13) Fonksiyonu küresel min. ve arama alanı. 71
Çizelge 3. 8. Matyas Fonksiyonu küresel min. ve arama alanı. 72
Çizelge 3. 9. McCormick Fonksiyonu küresel min. ve arama alanı. 73
Çizelge 3. 10. Schaffer (no. 2) Fonksiyonu küresel min. ve arama alanı. 74
Çizelge 3. 11. Ackley Fonksiyonu küresel min. ve arama alanı. 75
Çizelge 3. 12. Rastrigin Fonksiyonu küresel min. ve arama alanı. 76
Çizelge 3. 13. Rosenbrock Fonksiyonu küresel min. ve arama alanı. 77
Çizelge 3. 14. Sphere (Küre) Fonksiyonu küresel min. ve arama alanı. 78
Çizelge 3. 15. Styblinski–Tang Fonksiyonu küresel min. ve arama alanı. 79
Çizelge 3. 16. Mishra'nın Kuşu (Mishra's Bird) Fonk. küresel min. ve arama alanı. 80
Çizelge 3. 17. Simionescu Fonksiyonu küresel min. ve arama alanı. 81
Çizelge 3. 18. Townsend Fonksiyonu küresel min. ve arama alanı. 82
Çizelge 3. 19. Döndürülmüş Cigar (Rotated Cigar) Fonk. kür. min. ve arama alanı. 83
Çizelge 3. 20. Dönd. Yük. Kon. Eliptik (Rot. H. C. El.) F. kür. min. ve arama alanı. ...84
Çizelge 3. 21. Kayd. ve Dönd. Ackley (S. R. Ackley) F. kür. min. ve arama alanı. 85
Çizelge 3. 22. Kayd. ve Dönd. Rastrigin (S. R. Rast.) F. kür. min. ve arama alanı. 86
Çizelge 3. 23. Kayd. ve Dönd. Schwefel (S. R. Schw.) F. kür. min. ve arama alanı. 87
Çizelge 3. 24. Fonseca ve Fleming Fonksiyonu – ilgili kabuller. 88
Çizelge 3. 25. Kursawe Fonksiyonu ve ilgili kabuller. 89
Çizelge 3. 26. Zitzler-Deb-Thiele (no. 1) Fonksiyonu ve ilgili kabuller. 90
Çizelge 3. 27. Zitzler-Deb-Thiele (no. 3) Fonksiyonu ve ilgili kabuller. 91
Çizelge 3. 28. Binh ve Korn Fonksiyonu ve ilgili kabuller. 92
Çizelge 3. 29. Chakong ve Haimes Fonksiyonu ve ilgili kabuller. 93
Çizelge 3. 30. Osyczka ve Kundu Fonksiyonu ve ilgili kabuller. 94
Çizelge 3. 31. Portföy seçimi optimizasyon problemi (Xu ve ark., 2007). 96
Çizelge 3. 32. Çok-amaç. kısıtlı Disk Freni Tas. opt. problemi (Yang ve Deb, 2013). ..96
Çizelge 3. 33. Karşılaştırmaya tabi algoritmalar – teknikler. 102

Çizelge 5. 1. Tek-amaçlı fonk. değerlendirilen alg. elde edilen temel bulgular. 123
Çizelge 5. 2. Tek-amaç. fonk. GOA ve diğer alg. arası Wilcoxon bul. (1. kısım). 128
Çizelge 5. 3. Tek-amaç. fonk. GOA ve diğer alg. arası Wilcoxon bul. (2. kısım). 129
Çizelge 5. 4. Tek-amaç. fonk. BiGOA ve diğer alg. arası Wilcoxon bul. (1. kısım). ...130
Çizelge 5. 5. Tek-amaç. fonk. BiGOA ve diğer alg. arası Wilcoxon bul. (2. kısım). ...131
Çizelge 5. 6. Tek-amaçlı fonksiyonlar için OHK, OHKK, ve OMHY bulguları. 133
Çizelge 5. 7. Çok-amaçlı fonksiyonlarda en iyi sonuç veren ilk üç algoritma. 134
Çizelge 5. 8. Test fonk. kapsam. değ. alg. arası başarı sıra. ve GPSD değerleri. 137
Çizelge 5. 9. Portföy Seçimi problemi – uygulaması için elde edilen bulgular. 139
Çizelge 5. 10. Disk Freni Tasarımı problemi – uyg. için elde edilen bulgular. 140
Çizelge 5. 11. Kaotik zaman serileri ve modeller (Köse ve Arslan, 2016b; 2017c). 144
Çizelge 5. 12. Kaotik zaman seri. tahmininde YSA modellerine yönelik OHHK. 145

Çizelge 5. 13. Kaotik zaman seri. tahmininde ASBÇS modellerine yönelik OHHK. ..145
Çizelge 5. 14. E-Öğrenme tabanlı uygulama – 1 için OHKK bulguları.....151
Çizelge 5. 15. E-Öğrenme tabanlı uygulama – 2 için OHKK bulguları.....151
Çizelge 5. 16. Biyomedikal veri tahmini – teşhisi için OHKK bulguları.....152
Çizelge 5. 17. DVM sigma ve diyabet teşhis doğruluk. (Köse ve ark., 2015; 2016)....153

1. GİRİŞ

Teknolojik gelişmeler insanoğlunun yaşam standartları üzerinde büyük etkilere sahiptir. Kuşkusuz ki, insanoğlunun yaşamda meydana gelen değişmelere adapte olabilmesi ve toplumsal anlamda kendini değişimlere uyumlu tutabilmesi, teknolojiye olan bağlılığı sayesinde gerçekleşebilmiştir. Önceleri insanoğlunun sadece yaşamını devam ettirebilme ve neslini sürdürebilme adına yerine getirdiği 'adaptasyon' faaliyetleri, özellikle bilginin daha önemli bir hale gelmesiyle birlikte daha farklı amaçlar doğrultusunda da kullanılmaya başlanmıştır. Gerçekleşen değişimlerin ve nihayetinde gelişmelerin, esasında birbirini tetikleyen sonsuz bir zaman-olgu düzleminde meydana geldiği ortaya çıkmıştır (Yani genel hatlarıyla yaşantımızdaki bütün değişim ve gelişmelerin kendilerine bağlı çeşitli sebepler – öncüller içerdiğini, kendilerinin de sonraki bazı değişim ve gelişmelerin sebebi – öncülü olduğunu ifade etmemiz mümkündür). İlgili süreç içerisinde bilgi tabanlı teknolojik gelişmelerin ön plana çıkmış olması, insanoğlunun yaşamını devam ettirebilme ve neslini sürdürebilme ihtiyaçlarının ötesinde, yaşam standartlarını yükseltme ve merakını giderme ihtiyaçları dâhilinde bir yol izlemeye başladığının önemli bir işaretidir. Bu noktada, bilgi tabanlı teknolojik gelişmelerden en önemlisi bilgisayarın icat edilmesidir.

Bilgisayarın icadı, bilgiye erişip, onu daha etkin ve verimli işleyebilme ihtiyacının kaçınılmaz sonuçlarından birisi olarak düşünülebilmektedir. Elbette bilgisayar teknolojisinin gerek yazılım bağlamında, gerek donanım bağlamında günümüz formlarına erişmesi, farklı teknolojilerde meydana gelen destekleyici gelişmelerin (elektronik teknolojisi ile ilgili olarak entegrelerin icadı, iletişim teknolojileri ile ilgili olarak Internet teknolojisinin icadı… vb.) sayesinde olmuştur. Ancak detaylardan arınmak suretiyle, bilgisayar teknolojisinin bilgiye erişme ve bilgiyi işleyip iletme, depolama gibi işlevlerinin değeri aşikârdır. Bilgisayar teknolojisi öyle hızlı gelişmiştir ki, günümüz toplum formu bile –kaçınılmaz bir sonuç olarak– 'bilişim – bilgi toplumu' formuna geçiş yapmıştır. Bilişim – bilgi toplumunun bir gereği olarak, 'bilgisayar – bilişim okuryazarı olmak', günümüzde her bireyin sahip olması gereken önemli niteliklerden birisi olarak dikkat çekmektedir.

Bilgisayarlar söz konusu olduğunda, en basit haliyle akla gelen bilgiyi işleme ve depolama görevinin yanında, insanlığın yaşamında büyük rol oynayacak olan 'zeki olma' potansiyeli, esasında bundan uzun zaman önce sorgulanır olmuştur. Nasıl ki bilgisayarın ve sayısal hesaplamanın tarihçesine baktığımızda karşımıza 'abaküs' gibi ilkel hesap düzenekleri çıkıyorsa (Ifrah ve ark., 2000; Meyers, 1993; Ören, 2001: 64), bilgisayarların veya basit tabirle makinelerin zeki olma potansiyeli de tarihsel anlamda bizleri mekanik yapıdaki otomatonlara kadar götürmektedir (Aydın, 2013; Buchanan, 2005; Glaeser ve Strouhal, 2000; McCorduck ve ark., 1977). Ancak modern anlamda zeki makinelere dair ilk düşünceleri tetikleyen, Alan Mathison Turing olmuştur. Turing, 'Computing Machinery and Intelligence (Hesaplama Makineleri ve Zekâ)' başlıklı makalesinde *'Makineler düşünebilir mi?'* sorusunu sormuştur (Turing, 1950; Turing, 1995; Turing, 2009). Bu soruyu takiben, gerek Turing tarafından yapılan diğer çalışmalar, gerekse bilgisayar tabanlı sistemlerin insanlar gibi düşünebilmesi ve zeki olması potansiyelleri üzerine düşünen birçok bilim insanı tarafından farklı çalışmalar yapılarak yeni bir araştırma alanının doğması sağlanmıştır [Konuya ilişkin olarak, Turing ile yakın zamanda gerçekleştirilmiş olan diğer çalışmalardan bazıları: (Armer, 1960; De Latil, 1956; Hovland, 1960; Kemeny, 1955; Mays, 1952; Pinsky, 1951; Shannon, 1950; Wilkes, 1953)]. Aynı zamanda modern Bilgisayar Bilimleri'nin de babası olarak anılan Turing'in (Daylight, 2015; Nerode, 2016: x) ateşlediği fitilin ulaştığı noktada, Yapay Zekâ adı verilen bir araştırma alanı bilim dünyasına kazandırılmıştır.

Yapay Zekâ (Artificial Intelligence – AI; çalışmanın bundan sonraki paragraflarında kısaca YZ olarak anılacaktır), literatürde alternatif tanımlarla ifade edilmiş, esasında 'zeki sayısal sistemler' yaklaşımı çevresinde incelediğimizde yeteri kadar anlaşılır olan, Bilgisayar Bilimleri'nin bir alt-araştırma alanı olarak kabul edilmektedir. Bilgisayar Bilimleri'nin olduğu kadar, temellerini atmış olması bakımından öncüsü olarak düşünülen Turing'in dışında, YZ kavramını literatüre ilk kazandıran isim John McCarthy olmuştur. McCarthy (2007) ilgili kavramı zeki makineler yapma bilimi ya da mühendisliği şeklinde, Nabiyev (2012), bir bilgisayarın ya da bilgisayar denetimli bir makinenin, genellikle insana özgü nitelikler olduğu varsayılan akıl yürütme, anlam çıkartma, genelleme ve geçmiş tecrübelerden öğrenme gibi yüksek zihinsel süreçlerle bağlantılı görevleri yerine getirebilme yeteneği olarak, Luger (2004) ise, zeki davranışın otomasyonu ile uğraşan bir Bilgisayar Bilimleri dalı

olarak ifade etmiştir (Literatür incelendiğinde yine pek çok alternatif tanıma ulaşılabilmektedir). İlk olarak Amerika Birleşik Devletleri – Dartmouth'daki bir etkinlikte literatüre kazandırılan (McCarthy ve ark., 2006:1955) YZ kavramı, çıkış noktası olarak insanların düşünce ve davranış şekillerini taklit etmeyi temel aldıysa da, zaman içerisinde doğada zeki mekanizmalar çerçevesinde mantıksal ve matematiksel kalıplara sokulabilecek her türlü dinamik de bu alanın kapsamında dikkate alınır olmuştur. Bu noktada, literatüre de katkısı olması adına YZ ile ilgili iki farklı tanımlama daha yapabiliriz: Bir araştırma alanı olarak tanımlamak gerekirse YZ'yi, bilgisayar tabanlı sistemlerde insan düşünce ve davranış şekilleri ile doğal dinamiklerin benzetimi üzerine çalışmalar yapan, Bilgisayar Bilimleri'nin alt-araştırma alanı olarak; soyut bir kavram olarak tanımlamak gerektiğinde ise, insan düşünce ve davranış şekilleri ile doğal dinamiklerin, bilgisayar tabanlı sistemler tarafından mümkün olduğunca yerine getirilmesi olarak tanımlamamız mümkündür.

YZ alanı, özellikle farklı disiplinlerdeki gerçek dünya tabanlı problemlere sunduğu etkin çözümler ile kısa sürede gelişme göstermiştir. Gücünü mantıksal ve matematiksel düzlemde ifade edilebilen her türlü probleme çözüm sunabilmesinden almaktadır. Zaman içerisinde geliştirilen farklı YZ yaklaşımları, yöntemleri ve teknikleri, ileri matematiksel çözümlemeler yapma, öngörü – tahmin yapma, örüntü tanıma, görsel – işitsel – dilsel tanıma ve tanımlama, adaptif kontrol ve yorumlama – teşhis koyma gibi konularda başarılı çözüm yolları sunmuşlardır (Allahverdi, 2002; Aydın, 2013; Elmas, 2011; Nabiyev, 2012; Strong, 2016). Bu durum ve farklı problemlere kolayca adapte edilip uygulanabilme durumu, alanın bütün disiplinler tarafından rağbet görmesine sebebiyet vermiştir. Hatta sadece disiplinler içi spesifik problemlerin bilimsel çözümünde değil, günlük yaşantımızdaki birçok faaliyetin pratikleşmesi ve 'otomatikleşmesi' adına YZ'nin yaygın bir kullanım alanı elde etmiş olduğunu rahatlıkla ifade edebiliriz. İlerleme öyle bir noktaya ulaşmıştır ki, zeki sistemlerin insanlığın sonunu getirip getirmeyeceği, çalışan bireylerin işlerini kaybetmelerine sebep olup olmayacakları sıklıkla tartışılır hale gelmiştir (Brougham ve Haar, 2017; Brynjolfsson ve McAfee, 2015; Makridakis, 2017; Niewiadomski ve Anderson, 2016). Bu noktada, YZ'nin çözümler bağlamında bu kadar hızlı yükselmesini genel olarak iki nedene bağlayabiliriz: Birincil neden olarak söz konusu alan, çözümü bulunamamış problemleri başarılı bir şekilde çözebildiğinden dolayı uygulanmaktadır. İkincil neden olarak ise, klasik yöntemlerle istendik etkinlik ve verimlilik düzeyinde

çözüm elde edilemeyen problemlere oldukça iyi çözümler verebildiğinden dolayı uygulama sürecinde tercih edilir hale gelmiştir. Farklı disiplinler açısından ele alındığı zaman, problem çözümlerini hedef disiplinleri de dikkate almak suretiyle daha spesifik ifade edebiliriz. Diğer yandan klasik yaklaşım, yöntem ve tekniklerin yetersiz kaldığı ve YZ'nin çözüm bayrağını devraldığı, farklı disiplinlerin hep beraber ihtiyaç duyduğu çözüm yolları da bulunmaktadır. Optimizasyon bu çözüm yollarından birisidir.

Optimizasyon kavramını kısaca belirli alternatifler kümesinden en iyi elemanın (belirli kriterlere de bağlı olmak şartıyla) seçilmesi olarak tanımlamak mümkündür (INFORMS Computing Society, 2016; MedLibrary, 2016). Bu noktada, bir optimizasyon türü olan küresel optimizasyon (global optimization) ise spesifik anlamda, çözüm uzayındaki 'bütün' muhtemel çözümler arasındaki yegane optimum değeri bulmayı amaçlayan ve özellikle gerçek yaşam tabanlı problem için yaygın olarak kullanılan bir yaklaşım olarak karşımıza çıkmaktadır. Matematiksel açıdan baktığımız zaman küresel optimizasyon odaklı problemlerde, hedef optimizasyon problemi için ele alınmış, belirli kurallar ve kısıtlara tabi, bir veya daha fazla optimize edilmesi gereken fonksiyon söz konusu olmaktadır (Demir ve Köse, 2016; Güler ve Köse, 2016: 1; Weise, 2009; Yang, 2010b). Genel olarak küresel optimizasyon, kısıtlar içerisinde karar süreçlerinin ve tutarlı hesaplamaların gerekli olduğu her türlü problem için ihtiyaç duyulan bir hesaplama yaklaşımı olarak etkinliğini korumuştur. Söz konusu problemler yaşamın neredeyse bütün alanlarında söz konusu olabildiğinden dolayı, küresel optimizasyonun uygulama kapsamı oldukça büyüktür. Küresel optimizasyon, uğraştığı sayısal değerlerin niteliği anlamında kendi içerisinde alt-türlere de ayrılmaktadır. Sürekli – reel değerlerle uğraşan küresel optimizasyon, sürekli optimizasyon (continuous optimization) kavramıyla karşılanırken, sonlu – ayrık (discrite) değerler söz konusu olduğunda ise, ayrık optimizasyon (discrete optimization) kavramı kullanılmaktadır. Bu noktada, ayrık optimizasyonun bir kolu olan kombinasyonel optimizasyon da (combinatorial optimization), sınırlı çözüm seçeneklerinin kombinasyonlarının dikkate alınmasını gerekli kılan, gerçek yaşam tabanlı problemlerde kullanılabilmektedir (Hoffman, 2000; Kallrath ve Schreieck, 1995; Papadimitriou ve Steiglitz, 1982:1998). Anlaşılacağı üzere optimizasyon yaklaşımı, detaylarına inildiğinde geniş bir uygulama alternatiflerini de beraberinde getirmektedir.

Temelinde klasik yaklaşımların var olduğu ve zaman içerisinde duyulan ihtiyaçlar neticesinde ortaya çıkan farklı klasik optimizasyon tekniklerin de işe koşularak çözümlemelerin yapıldığı ilgili optimizasyon yaklaşımları, bir süre sonra daha etkin çözüm yollarına ihtiyaç duymaya başlamıştır. Özellikle ileri düzey optimizasyon problemlerine karşı klasik yöntem ve teknikler yetersiz kalmış, çözümü daha hassas ele alan, ancak bunu da zaman ve tutarlılık yönünden daha etkin gerçekleştirebilen alternatiflere olan ihtiyaç iyice gün yüzüne çıkmıştır. İşte bu noktada imdada YZ yetişmiş ve optimizasyon problemlerinde yaygınlaşmaya başlamıştır (Agrawal, 2016; Binitha ve Sathya, 2012; Karaboğa, 2014; Yang, 2010b). Zeki çözümlerin optimizasyon problemlerinde kullanımı o kadar yaygınlaşmıştır ki, optimizasyon tekniklerinin tasarlanıp geliştirildiği ve inceleme konusu olduğu alt-araştırma ilgileri ve alanları da zaman içerisinde birer birer ortaya çıkmıştır (Ahmed ve Glasgow, 2012; Yang ve ark., 2013: xix; Zhongzhi, 2001).

Açıklamalar kapsamında bu tez çalışmasının amacı, YZ optimizasyon teknikleri ile ilişkili literatüre, kodlanması daha kolay, alternatif algoritmalar sunmaktır. Bu amaç kapsamında iki farklı sürekli optimizasyon algoritmasının tasarlanması ve geliştirilmesi üzerinde durulmuş olunacak ve böylece optimizasyon problemlerine yönelik alternatif çözüm teknikleri elde edilmeye çalışılacaktır. Durum daha geniş bir bakış açısıyla ele alındığında ise bu çalışma ile elde edilecek çıktıların, farklı alanlar altında söz konusu olan ve optimizasyon ile çözülebilen, gerçek dünya tabanlı problemleri için alternatif çözüm yolları sağlamış olacağını da ifade etmek gerekmektedir.

Temel amaç kapsamında olmak üzere, çalışmanın alt-amaçlarını da şöyle açıklamak mümkündür:

- YZ ve YZ ile optimizasyon – zeki optimizasyon kavramları konusunda temel bilgiler sunmak ve okuyucuda bu konulara ilişkin genel bir farkındalık düzeyinin şekillenmesini sağlamak,
- Başlıca zeki optimizasyon algoritmalarının (tekniklerinin) neler olduğu ve bu algoritmaların genel özellik – işlevleri konusunda temel düzeyde bilgiler vermek,
- İlgili literatüre, mantıksal ve matematiksel anlamda modellenmesi, kodlanması ve uygulanması kolay, basit matematiksel denklemler kullanırken

yeterince etkin çözümler de sunan sürekli optimizasyon algoritmalarının tasarlanıp geliştirilmesini sağlamak,

- Bir önceki alt-amaç ve temel amaç doğrultusunda olmak üzere; çalışma kapsamında sunulan, Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) tekniklerinin gelişim süreci konusunda bilgi vererek, benzeri algoritma geliştirme süreçlerinde ne gibi yaklaşımlar izlenebileceğine dair fikirler sunmak,

- Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) tekniklerinin başarımı, etkinliği ve verimliliği konularında bilgi sahibi olabilmek adına çeşitli değerlendirme çalışmaları yapmak ve bu çalışmalar ile elde edilen bulguları yorumlamak.

İfade edilen temel amaç ve alt-amaçlar dikkate alınmak suretiyle, tez çalışmasının ilerleyen bölümleri şu şekilde organize edilmiştir:

Bir sonraki bölüm altında, çalışmanın konusu kapsamındaki kavramlara değinilmiş, bu yolla okuyucunun tez çalışması süreci içerisinde yapılan algoritma tasarım ve geliştirme süreçleri öncesinde yeter düzeyde bir teorik altyapı kazanması sağlanmıştır. Söz konusu bölüm altında ifade edilenler, öncelikli olarak 'zeki optimizasyon' süreçlerine yönelik temel konulara odaklanmakta, akabinde ise bu konuyla yüksek oranda ilişkili olan Sürü Zekâsı (Swarm Intelligence) alt-araştırma alanı dikkate alınmaktadır. Konuyla ilişkili teorik altyapının güncel ve geçerli bilgilerle donatılması adına, bu bölüm altında sunulan bilgilerin ilgili literatürdeki son çalışmalar ile bağlantılı olmasına ayrıca dikkat edilmiştir.

İkinci bölümü takiben, üçüncü bölüm altında, çalışmaya yönelik olarak açıklanması gereken materyal ve metotlara odaklanılmıştır. Burada, materyal ve metotlar ile dile getirilenler, çalışma kapsamında tasarlanıp geliştirilmiş algoritmaların başarımını değerlendirmek için işe koşulan test fonksiyonları (benchmark functions), gerçek yaşam tabanlı problemler – uygulamalar, geliştirilen algoritmaların mevcut alternatifler çerçevesindeki konumunu tespit etmek adına dikkate alınan diğer algoritmalar – teknikler ve algoritmaların tasarlanıp geliştirilmesi aşamalarıyla birlikte değerlendirme süreçlerine yönelik detaylardır. Okuyucunun bu bölüm altındaki

açıklamalar sayesinde, çalışmanın uygulama sürecine yönelik ön-bilgiler edinmesi amaçlanmıştır.

Dördüncü bölüm bağlamında, çalışmanın temel amacı ve alt-amaçları doğrultusunda geliştirilen algoritmaların, genel hatlarıyla tanıtılması yoluna gidilmiştir. Bu bölüm sayesinde okuyucuların çalışma kapsamında ortaya konulan, Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) tekniklerinin ortaya konulması aşamasındaki arka-plan konusunda bilgi sahibi olması amaçlanmış, ilgili tekniklerin temel işlev – özellikleri ve algoritma adımlarının anlatımı yapılmıştır.

Beşinci bölümde, ortaya konulan algoritmalara yönelik yapılan ve üçüncü bölümde kısaca ifade edilmiş olan değerlendirme yaklaşım, yöntem ve teknikleri dikkate alınarak yapılan çalışmalardan elde edilen bulgular okuyuculara sunulmuştur. Bu değerlendirme süreçlerinde algoritmaların test fonksiyonlarını ve gerçek yaşam tabanlı çeşitli problem – uygulamaları çözme konusundaki performansları, alternatiflerle karşılaştırmalı olarak ifade edilmiş, bulgular ışığında, Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) tekniklerinin başarımları ve literatürdeki yerleri konusunda objektif yorumlamalar yapılmıştır. Bu bölümün özellikle gerçekleştirilen çalışmanın, ifade edilen temel amaç ve alt-amaçlara ulaşıp ulaşmadığını anlamak adına gerekli değerlendirmeyi sunacağı ve yine –her ne kadar son bölüm altında daha açık ele alınmış olsa da– benzeri çalışmalara yönelik faydalı bilgileri – önerileri ortaya koyacağı düşünülmüştür.

Çalışmanın son bölümü olan; altıncı bölüm kapsamında, çalışma süreci ile elde edilen sonuçlar toparlanarak özetlenmiş, çalışma sürecinde elde edilen tecrübeler, bağlı literatürün mevcut durumu ve sunulan algoritmaların uygulanması sonucu elde edilmiş olan bulgular dikkate alınmak suretiyle çeşitli öneriler dile getirilmiş, son olarak da, ilerleyen dönemlerde yapılması planlanan ‘gelecek çalışmalar’ konusunda bilgi verilmiştir.

2. KAYNAK ARAŞTIRMASI

Çalışmanın konusu doğrultusunda, ilgili optimizasyon algoritmalarına yönelik tasarım – geliştirme ve değerlendirme süreçlerine değinmeden önce, konuyu destekleyen arkaplana da odaklanmak gerekmektedir. Nitekim bu bölüm, zeki optimizasyon konuları ile birlikte Sürü Zekâsı (Swarm Intelligence) alt-araştırma alanı kapsamında temel açıklamalarla donatılmıştır.

2.1. Küresel Optimizasyon

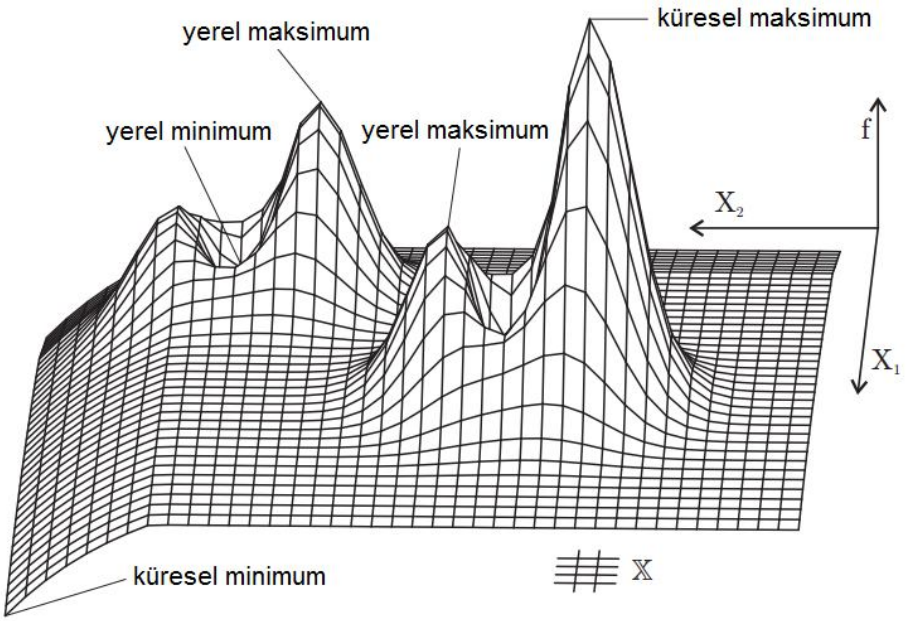
Çalışmaya konu olan optimizasyon algoritmaları, küresel optimizasyon (global optimizasyon) adı altında inceleyebildiğimiz, yani çözüm kümesindeki bütün muhtemel çözümler arasındaki dahilindeki optimumu bulmayı amaçlayan optimizasyon problemlerine yönelik olmak üzere literatüre sunulmuştur. Küresel optimizasyon süreçleri, genellikle bir ya da daha fazla fonksiyonun, belirli şartlar altında optimize edilmesi yönünde kullanılmaktadır (Horst ve Tuy, 2013; Neumaier, 1996; Torn ve Zilinskas, 1989). Bu bağlamda, özellikle gerçek yaşamda karşılaşılan ve çözümlenmesi zor optimizasyon problemlerinin matematiksel düzlemde çözülmesi için küresel optimizasyon yaygın bir şekilde işe koşulmaktadır. Burada öncelikli olarak, ‘model’ adı verilen ve ilgili gerçek yaşam tabanlı problemin optimizasyon sürecine uygulanabilmesine imkân verecek şekilde; matematiksel olarak temsil edilmesini sağlayan bir gösterimin yapılması gerekmektedir (Mucherino ve Seref, 2009).

2.1.1. Sürekli optimizasyon

Sürekli optimizasyon, küresel optimizasyonda söz konusun olan; eldeki problem kapsamında minimum ya da maksimum değer(ler)in bulunması işlevleriyle (Wikipedia, 2017a) ilişkili olmakla birlikte, ‘sürekli’ değerlerle uğraşması bakımından ise daha spesifik bir hale gelmektedir. Bu noktada, eğer bir optimizasyon probleminde amaç sonucu düşüren en küçük değer(ler)i bulmak ise (Örneğin; zararı en aza indirgemek, masrafı en aza indirgemek, ihtiyaç duyulan herhangi bir şeyi en aza indirgemek… vb.), o problemde minimizasyon (minimization) yapılmaktadır. Tersi olarak; eğer problemde amaç sonucu artıran en büyük değer(ler)i bulmak ise (Örneğin; kârı en yükseğe

çıkarmak, başarımı artırmak, elde edilen çıktıdan en yüksek verimi elde etmek… vb.), ilgili problemde maksimizasyon (maximization) gerçekleştirilmektedir.

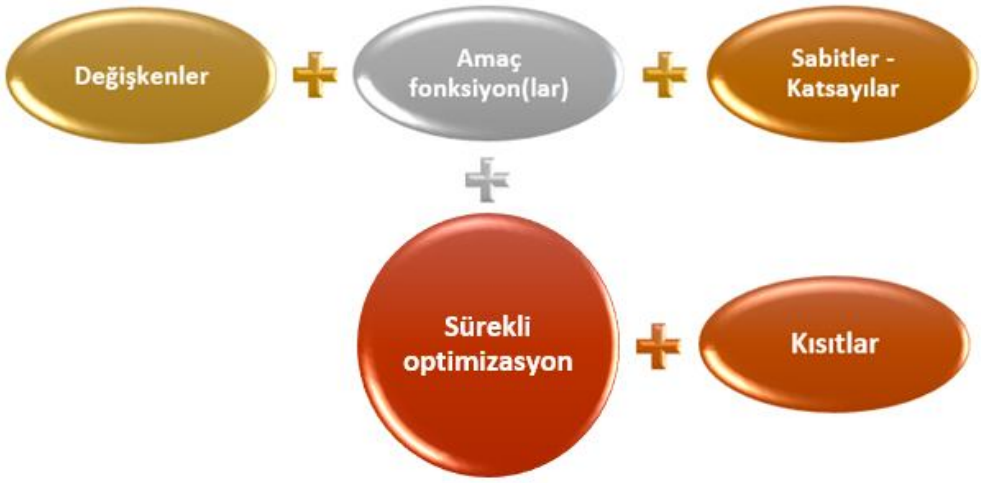
Bir problemde çözüme ulaştıran olabilecek optimum – en uygun (yani optimum minimum ya da maksimum) değer(ler) küresel (global: global minimum ya da global maksimum) olarak adlandırılırken, bu değer(ler)e ulaşılırken söz konusu olabilen 'optimuma yakın' durumdaki bütün değer(ler) ise yerel (lokal-local: local minimum ya da local maksimum) optimum olarak adlandırılmaktadır. Aslında küresel optimum değer(ler)i, yerel bütün optimuma yakınlar arasındaki 'en iyi' olan(lar)ı olarak ifade edebiliriz. Bu kapsamda, sürekli optimizasyon sürecinin gerçekleştirildiği (optimumun arandığı), aday çözümlerin bulunduğu düzlemlere çözüm uzayı (solution space) adı verilmektedir [Şekil 2.1 – (Weise, 2009: 26)].



Şekil 2. 1. Çözüm uzayında yerel ve küresel optimumlar (Weise, 2009: 26).

Tipik olarak sürekli optimizasyon problemlerinde şu temel unsurlar yer almaktadır (Horst ve Tuy, 2013; Talbi, 2009; Torn ve Zilinskas, 1989; Weise, 2009):

- Amaç fonksiyon (ya da fonksiyonlar),
- Fonksiyon(lar) bağlamında problemle ilişkili değişkenler,
- Fonksiyon(lar) bağlamında problemle ilişkili sabitler – katsayılar,
- Söz konusu değişkenlerle ilişkili çeşitli kısıtlar.



Şekil 2. 2. Sürekli optimizasyonda temel unsurlar.

İfade edilen unsurlardan amaç fonksiyon ve kısıtların durumuna göre de bir sürekli optimizasyon problemini farklı şekilde açıklayabiliriz. Buna göre:

- Eğer sürekli optimizasyon probleminde bir tane amaç fonksiyon varsa, o problem tek-amaçlı / bir-amaçlı (single-objective) optimizasyon (Örneğin, tek amaç olarak; kârı yükseltmek, zararı azaltmak, başarımı artırmak… vb.); birden fazla amaç fonksiyon varsa, çok-amaçlı (multi-objective) optimizasyon olarak (Örneğin, çakışan birden fazla amaç varken; maliyet düşürürken başarım artırmak, kârı artırırken başarımı da artırmak ancak malzeme oranını azaltmak… vb.) – (Weise, 2009: 25-27),

- Eğer sürekli optimizasyon probleminde karar değişken(ler)i üzerinde kısıt fonksiyonu (ya da fonksiyonları) varsa, o problem kısıtlı (constrained) optimizasyon (Örneğin, probleme dahil olan başarım değişkeninin 10'dan küçük 100'den büyük değer almaması, bir makinenin çalışma sayısının negatif olamaması… vb.); aksi halde, kısıtsız (unconstrained) optimizasyon olarak (NEOS Guide, 2017b; 2017e)

ifade edilebilmektedir. Tahmin edileceği üzere, optimizasyon problemini ilgili faktörlere göre doğru tanımlamak, doğru çözüm elde edebilmek adına oldukça kritiktir.

2.1.2. Sürekli optimizasyonun matematiksel ifadesi

Bir önceki alt-başlık kapsamındaki açıklamaları ve bağlı unsurları dikkate almak suretiyle, tipik bir sürekli optimizasyon probleminin matematiksel düzlemde ifade ediliş şekli kısaca şu şekilde olmaktadır (Mucherino ve Seref, 2009; Weise, 2009; Yang, 2010b):

$f_i: x \in A \rightarrow y \in B$ olmak üzere;

$x = (x_1, x_2, x_3, \dots, x_n)^T$ için her bir x_i, f_i'yi y vektörüne haritalayan birer değişkeni temsil etmektedir.

f_i kısaca, $i = 1,2,3, \dots, Z$ için amaç fonksiyonlar olmakla birlikte, bir problemde i = 1 için tek / bir amaç fonksiyon, $i \geq 2$ için çok amaç fonksiyon söz konusudur.

İfadeyi basitleştirmek adına i = 1 kabul edersek; B'deki küresel minimum değere karşılık gelen bir $x^o \in A$ arayışı, (yani):

$x^o \in A : f(x) \geq f(x^o)\ \forall x \in A$ bir minimizasyon problemine tekabül eden küresel optimizasyonunu tanımlamakta ve genellikle $min\, f(x)$ olarak gösterilebilmekte; tersi durum olarak, B'deki küresel maksimum değere karşılık gelen bir $x^o \in A$ arayışı, (yani):

$x^o \in A : f(x) \leq f(x^o)\ \forall x \in A$ bir maksimizasyon problemine tekabül eden küresel optimizasyonu tanımlamakta ve genellikle $max\, f(x)$ olarak gösterilebilmektedir.

Her f_i bağlamında, x vektörüne ait olmayan sayısal değerler, sabit olarak adlandırılmaktadır.

İfade edilen minimizasyon veya maksimizasyon problemlerinde olmak üzere;

$k_i(x) = 0\ \forall i$ ve $k_j(x) \leq 0\ \forall j$ gibi, A kümesinin sınırlarını düzenleyen, k_i ve k_j şeklindeki fonksiyonlar da, kısıt(lar) olarak ifade edilmektedir.

Söz konusu matematiksel yapılar bağlamında, modeli çıkartılabilen ve sürekli değerler kapsamında optimizasyon gerektiren her türlü gerçek yaşam tabanlı problem

sürekli optimizasyon konusu içerisine girmektedir. Sürekli optimizasyon odaklı problemlerin çözümlenmesi için geliştirilmiş olan yaklaşım, yöntem ve teknikler, gerçek yaşam tabanlı problemlere karşı elde edilen sonuçların vereceği fikirlerle olduğu kadar, bu tür yaklaşım, yöntem ve tekniklerin etkinliğini değerlendirmek için kullanılabilecek çeşitli test fonksiyonları da, araştırmacılar tarafından sıklıkla tercih edilmektedir. Bu tez çalışmasındaki algoritmaları değerlendirmek için de kullanılan test fonksiyonlarına yönelik detaylar üçüncü bölüm, alt-başlık 3.3.'de verilmiştir.

Daha önce de sıklıkla ifade edildiği üzere küresel optimizasyon ve bu bağlamdaki sürekli optimizasyon, gerçek yaşam temelli optimizasyon problemlerinin çözümünde oldukça faydalı olmuşlardır. Tabi ki ortaya çıkan teknolojik gelişmeler ve doğal olarak değişen modern hayat, optimizasyon problemlerinin düzeyini de artırmıştır. Bu noktada, ileri düzey optimizasyon problemlerine karşı varsayılan çözüm yaklaşımları yetersiz kalan optimizasyon süreçlerinin başarımı artırmak adına YZ alanından faydalanılmaya başlanmıştır.

2.2. Yapay Zekâ ve Optimizasyon

YZ'nin ilk temellerinin ortaya atılışından bu yana kadar geçen süre, bu araştırma alanının kendisini farklı problem çözüm konuları çerçevesinde geliştirilmesi doğrultusunda geçmiş durumdadır. Alanın farklı problem çözüm konuları içerisine nüfuz etmesi, çok-disiplinli yönünün daha fazla keşfedilmesine ve daha fazla alt-uzmanlıkların oluşmasına da sebep olmuştur. Bu alt-uzmanlıklar kimi zaman Makine Öğrenmesi (Machine Learning) gibi kendine özgü isimlerle anılır duruma gelmişken, kimi zaman da temel problem konusuna göre anılmıştır.

Kendine özgü isimlerle anılanları şu an için bir kenara bırakacak olursak, YZ'nin genel çerçevede şu problem konuları üzerinde yoğunlaştığını ifade edebiliriz (Allahverdi, 2002; Alpaydın, 2011; Aydın, 2013; Cohen ve Feigenbaum, 2014; Elmas, 2003a; 2003b; 2011; Hamet ve Tremblay, 2017; Holland, 1992:1975; Jaeger, 2016; Karaboğa, 2014; Korbicz ve ark., 2012; Michalski ve ark., 2013; Nabiyev, 2012; Nilsson, 2014; O'Hare ve Jennings, 1996; Strong, 2016): Öngörü – tahminler, görsel ve/veya işitsel tanıma, dilsel analiz, örüntü tanıma, optimizasyon, adaptif kontrol, teşhis, yorumlama, bilgi işleme ve bilgi modelleme.

İfade edilen problem konuları kuşkusuz ki birbirleriyle kesişebilmekte ve yine bu problem konularından alternatif konular türetebiliriz. Ancak esas önemli nokta; alanın, mantıksal ve matematiksel düzenekleri sayısal sistemlerin avantajlarıyla birleştirmek suretiyle, ileri düzey problemlere etkin ve verimli çözümler üretebiliyor olmasıdır. Bağlı literatürde günümüze kadar yaygınlığını korumuş olan optimizasyon uygulamaları da, ilgili problem konuları içerisinden ayrılarak kaleme alınan bu çalışmada kendisine yer bulmuştur.

Çalışmanın Giriş bölümü altında da değindiğimiz üzere, optimizasyonda adaylar arasından problemimize optimum çözümün seçilmesi ve bunun belirli kısıtlar ve kabuller altında yapmamız söz konusudur. Yang (2010b: 2-3), optimizasyonu genel olarak bir tür 'hazine avı' sürecine benzetmekte ve burada özellikle 'hazineyi (optimumu) arayan avcıların' rolüne odaklanmaktadır. Optimizasyon içerisindeki süreçleri dikkate aldığımızda, esasında ne kadar insani yönü olan işlevler – davranışlarla ilişkili olduklarını anlayabiliriz. İşte aslında bu nedenle YZ, optimizasyon problemleri için bir tür biçilmiş kaftan olmuştur. Özellikle optimizasyonun, karar süreçleri yönünde işe koşulan bir çözüm yolu olması ve yine insani (ya da en azından kabul edilebilir düşünebilme yetisine sahip bir canlı – varlık tarafından gerçekleştirilebilen) bir yaklaşım olan karar vermenin mevzu bahis olması, YZ ve optimizasyonun kesiştiği noktayı anlamamıza yeterli olmaktadır.

İfade edilen ilişkiyi biraz daha teknik açıdan ele almak ve yorumlamak da pekâlâ mümkündür. Buna göre, gerek literatür bilgisine sadık kalarak, gerekse bu çalışma ile elde ettiğimiz tecrübeler neticesinde YZ alanının optimizasyon problemlerinde tercih edilmesine sebep başlıca durumları şöyle sıralayabiliriz (Geem ve ark., 2001; Karaboğa, 2014; Lee ve El-Sharkawi, 2008: xxi; Weise, 2009; Yang, 2010b; Zelinka ve ark., 2012):

- Klasik optimizasyon yaklaşım, yöntem ve tekniklerinin ileri düzey optimizasyon problemlerinde yetersiz kalması [Örneğin; problemlerde yer alan yerel optimum değerlere takılma durumu, uzun hesap ve işlem süreçlerine ihtiyaç duyulması, optimum değerlerin büyük farklarla ıskalanması… vb. – Klasik optimizasyon konusunda daha fazla bilgi için okuyucular şu kaynaklara başvurabilirler: (Bayraktar ve Çebi, 2003; Floudas,

2013; Kaplan, 2011; Kaymaz, 2016; Kearfott, 2013; Miller, 2011; Rao, 2009; Rao ve Rao, 2009)],

- Söz konusu alanın optimizasyon problemlerinin çözümünde etkinlik, hız ve verimlilik avantajlarının beraberinde getirmesi,
- İleri düzey optimizasyon problemlerinde insanlara (ve kimi zaman diğer canlılara) has olan çeşitli yaklaşımların (sezgilerin kullanılması, rastgele davranışlar…vb.) benzetiminin ilgili alan tarafından kolaylıkla ve daha etkin ölçekte yapılabiliyor olması,
- YZ'nin her zaman geliştirilebilir bir yapıda olmak üzere, oldukça esnek, uygulanabilir çözüm yollarını sunma konusunda daha etkili bir yaklaşım olması,
- YZ'nin, tasarlanan mantıksal ve matematiksel süreçleri, uygulamaya pratik ve esnek bir şekilde aktarabilen geniş yaklaşımlar ve yöntemler kümesi üzerine kurulu olması.



Şekil 2. 3. YZ'nin optimizasyon için tercih edilmesine sebep başlıca durumlar.

Sonuç itibariyle YZ, optimizasyon problemlerine de hızlı bir şekilde katılmış ve günümüzde aktif ve oldukça geniş bir literatürün oluşmasına da sebep olmuştur. Bu

bağlamda, klasik optimizasyon kavramına karşılık olarak, ‘zeki optimizasyon’ kavramından bahsetmek sanırız ki yanlış olmayacaktır.

2.2.1. Zeki optimizasyon kavramı

‘Zeki optimizasyon’, kısa ve öz bir ifadeyle, YZ ile gerçekleştirilen optimizasyon sürecine yönelik olarak kullanabileceğimiz bir kavramdır. Zeki optimizasyon sayesinde, insan düşünce ve davranış şekilleri ile doğal dinamiklere bağlı olayların benzetimi yönündeki çözüm yolları, optimizasyon süreçlerine uygulanmaktadır. Zeki optimizasyon kavramı bu nedenle metin içerisinde de sıklıkla kullanılmış olacaktır.

Literatürü incelediğimiz zaman, ilerleyen paragraflar altında da açıklayacağımız tekniklerin – algoritmaların, bazı temel yaklaşımlar üzerine kurulu bir şekilde tasarlanmış ve geliştirilmiş olduğunu gözlemleyebiliriz. Zira geçen süre zarfında literatüre kazandırılmış tekniklerin – algoritmaların da, –istisnai durumlar hariç olmak üzere– genellikle bu yaklaşımlara yönelmiş olduklarını ifade edebiliriz. Bunun temel sebebi, söz konusu yaklaşımların zeki optimizasyonu gerçekleştirecek mekanizmaları sağlayan, kendini kanıtlamış bileşenler olmasıdır.

2.2.2. Zeki optimizasyonda temel yaklaşımlar

Zeki optimizasyonu daha iyi anlamak adına, söz konusu algoritmalar – teknikler içerisinde sıklıkla başvurulan temel yaklaşımları, detaylardan arınmak suretiyle kısaca şöyle açıklayabiliriz:

- **İteratif – Döngüsel Çözüm Süreci:** Nasıl ki gerçek dünya tabanlı problemlerin çözümü noktasında insanoğlu olarak kimi zaman deneme – yanılma yöntemleri kullanabiliyor ya da bazı problemleri benzer işlem adımlarını tekrar ederek çözebiliyorsak, bilgisayarların programlanması süreçlerinde sık kullanılan yaklaşımlardan birisi de iteratif – döngüsel çözüm süreçlerini kullanmaktır. Döngü adı verilen özel kod yapılarıyla gerçekleştirilen bu çözüm süreçleri, zeki optimizasyon içerisinde de kullanılmaktadır. Burada amaç, aynı mantıksal ve matematiksel işlemlerin üst üste devreye sokulması suretiyle çözümsüz problem yapısının aşındırmak

suretiyle istendik çözüm düzeyine ulaşılmasıdır. Zeki optimizasyonda, iteratif – döngüsel çözüm süreçleri ya 'belirli sayı kadar' aynı algoritmik adımların 'tekrar edilmesi' suretiyle gerçekleştirilmekte ya da 'arzu edilen bir çözüm noktasını (ya da en azından yakın değerleri) durma noktası kabul ederek' yine söz konusu algoritmik adımların 'tekrar edilmesi' şeklinde gerçekleştirilmektedir. Bunlar dışında, iteratif – döngüsel çözüm sürecini, algoritmada yer alan herhangi bir parametrenin belirli bir değere ulaşmasına kadar gerçekleştirilmesi yaklaşımı da kimi zaman izlenebilmektedir.

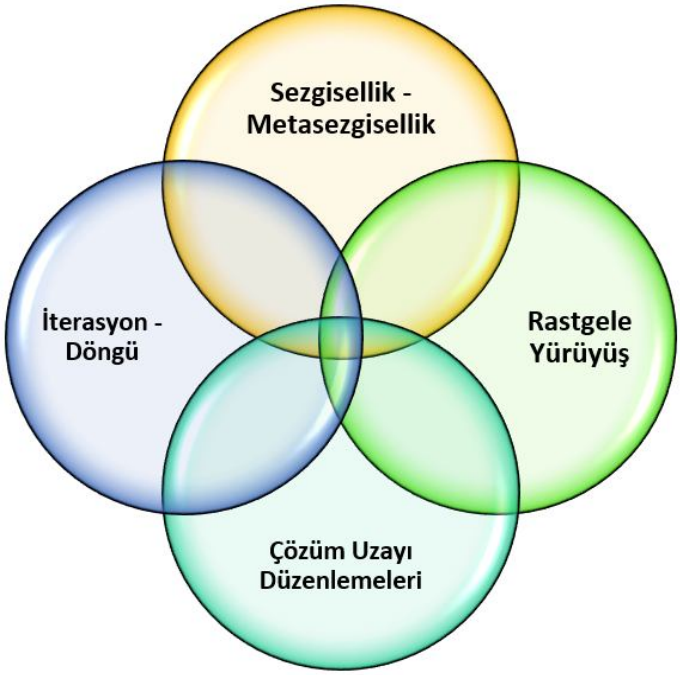
- **Çözüm Nesneleri Kullanımı:** Zeki optimizasyonun etkin çözüm bulma konusundaki yetenekleri kuşkusuz ki 'bir elin nesi var; iki elin sesi var' mantığı çevresinde birleşen, çok sayıda çözüm nesnesinin çözüm uzayında kullanılması yaklaşımıyla ilişkilidir. Günümüz zeki optimizasyon çalışmalarına baktığımız zaman, söz konusu tekniklerin – algoritmaların, çeşitli yardımcı parametrelere sahip, belirli sayıda çözüm nesnelerini çözüm sürecinde kullanma yoluna gittiklerini görebiliriz. Burada amaç, optimize edilecek problem üzerinde çok sayıda çözüm nesnesinin uğraşması ve iteratif – döngüsel süreç sayesinde adım adım, en iyi çözüme sahip nesne ya da nesnelerin elde edilmesidir. Çözüm nesnelerinin yapıları genellikle dayalı olduğu (ya da esinlenmiş olduğu) benzetim yaklaşımına göre farklılık göstermekle beraber, amaçları değişmemektedir.

- **Rastgele Yürüyüş (Random Walk):** Yine insan düşünce ve davranış şekillerine odaklanmak gerekirse, nasıl ki bazen problem çözümlerinde rastgele kararlar ve davranışlar bizi bir şekilde çözüme ulaştırabiliyorsa, zeki optimizasyonda da bu 'rastgelelik' işe yaramaktadır. Bu durumu yine doğal dinamiklerin özünde gözlemleyebildiğimiz ama geniş çerçevede bir düzene karşılık gelen rastgele olgular (Kaos) ile de açıklayabiliriz. Zeki optimizasyona dönecek olursak, bu rastgelelik, bir önceki madde altında açıkladığımız çözüm nesnelerinin rastgele yürüyüş (random walk) yaklaşımı altında, -günümüz algoritmalarında da sıklıkla görülen- çözüm uzayında rastgele hareketleri ile açıklanabilmektedir (Hassanien ve Emary, 2016; Yang, 2010b). İfade edilen rastgele hareketler, çözüm elemanlarının konumlarının genellikle rastgele üretilen sayılar aracılığıyla değiştirilmesi

şekilde olmaktadır. Bu rastgele yürüyüş esnasında ulaşılan noktalar optimum çözüm bağlamında değerlendirilmekte ve ulaşılan noktalar, çözüm nesnelerinin sonraki rastgele hareketleri üzerinde de etki sahibi olabilmektedir. Bu varsayılan konum değiştirme dışında, zeki optimizasyonda kendine yer edinen diğer rastgele yürüyüş şekli de 'Levy Uçuşları (Levy Flights)' olarak adlandırılmaktadır (Mandelbrot, 1982: 289). Levy Uçuşları'nda, varsayılan rastgele yürüyüş yaklaşımından farklı olarak, hareketin yapılması aşamasında olasılık dağılımlarından faydalanılması söz konusu olmaktadır. Burada kısaca ifade edilen, her iki rastgele yürüyüş yaklaşımına yönelik detaylar, ilerleyen bölümler / alt-başlıklar kapsamında daha sağlıklı anlaşılabilecektir.

- **Sezgisellik / Metasezgisellik:** İnsanoğlunun genel olarak yaşamında, özel olarak ise karar verme süreçlerinde sıklıkla izlediği yaklaşımlardan birisi de sezgilerini kullanmaktır. Bu bağlamda, sezgilerine güvenmek ifadesi de eylemlerinde sezgilerine sıklıkla başvuran insanlar için kullanılabilmektedir. Bu yaklaşıma benzer olarak zeki optimizasyonda problem çözümlerinde sezgisel yaklaşımlar yoğun bir şekilde kullanılmaktadır. Buradaki sezgisellik, çözümün nerede bulunacağından emin olamamakla birlikte, çözüm nesnelerinin, algoritma içi yapılardan güç almak suretiyle, kısıtlı mantıksal ve matematiksel bileşenlerle birlikte, çözümün bulunmasının garanti olmadığı şartlar altında, emin adımlarla işe koşulmasıyla bağlantılıdır (Karaboğa, 2014; Weise, 2009; Yang, 2010b: 4). Problem odaklı olan sezgiselliğin mantıksal yönünün yanında matematiksel karşılığı da, bir önceki madde altında anlatılan rastgele yürüyüş ile gerçekleştirilebilmektedir. Diğer yandan, problem odaklı olmamak (sezgisel yaklaşımlar gibi problemin spesifik avantajlarına dayanma eğilimi göstermemek) ile birlikte, yerel optimumlara yakalanma gibi sorunlardan kurtulabilen, dolayısıyla daha karmaşık optimizasyon problemlerinin çözümü noktasında işe koşulan, sezgiselliğin bir üst noktası olan metasezgisellik de, zeki optimizasyonun temel yaklaşımlarından birisi olarak ifade edilmelidir (Gendreau ve Potvin, 2010; Talbi, 2009; Weise, 2009; Yang, 2010b).

- **Çözüm Uzayı Düzenlemeleri:** Esasında zeki optimizasyonun söz konusu olduğu çözüm uzayları, optimum değer(ler)in bulunacağı amaç fonksiyonu ya da fonksiyonları ile ilişkili olmakla beraber, bazı tekniklerde – algoritmalarda özel düzenlemelere de tabi olmaktadır. Ele alınan problemin klasik düzenine göre bazen çözüm uzayında çeşitli düzenlemeler (Örneğin; çözüm uzayının kısıtlarla kontrol edilmesi, sınır değerlerin normalizasyonu…vb.) yapılabilmektedir. Yine konuyla ilişkili olarak, ayrık optimizasyon ile ve daha spesifik anlamda kombinasyonel optimizasyon ile ilgilenen bazı zeki optimizasyon tekniklerinin – algoritmalarının graf (graph) tabanlı çözüm uzayları kullanması da bu düzenlemeler içerisinde ifade edilebilmektedir. Kısaca çözüm uzayı düzenlemelerini zeki optimizasyon süreçlerinin ilgili problemlere başarılı bir şekilde uygulanabilmesi adına yapılan, 'ön-optimizasyon' yaklaşımlarından birisi olarak kabul edebiliriz.



Şekil 2. 4. Zeki optimizasyonda sıklıkla başvurulan temel yaklaşımlar.

2.3. Zeki Optimizasyon Tekniklerinin Sınıflandırılması

Zeki optimizasyon teknikleri – algoritmaları konusunu detaylı irdelediğimizde, birçok farklı uygulama yaklaşımının söz konusu olduğunu görebiliriz. En basitinden, zeki optimizasyon süreçlerine kimi genel teknikleri (Yapay Sinir Ağları, Bulanık

Mantık – Bulanık Programlama, Genetik Algoritmalar) uygulamak söz konusu olabildiği gibi (Chen, 2003; Cochocki ve Unbehauen, 1993; Horn ve ark., 1994; Karaboğa, 2014; Weile ve Michielssen, 1997; Zhang, 2013b; Zhou, 1998; Zimmermann, 1978), sadece optimizasyon problemleri için geliştirilmiş olan teknikler de, taşıdıkları farklı özellikler ve işlevler nedeniyle farklı sınıflandırmalar altına almak mümkün olmuştur. Literatüre sadık kalmak suretiyle başlıca sınıflandırmaları şu şekilde açıklayabiliriz:

- **Doğa-Esinli Hesaplama (Nature-Inspired Computing):** Doğal dinamiklerden esinlenilerek oluşturulmuş olan tekniklerin bünyesinde toplayan Doğa-Esinli Hesaplama (DEH-NIC), zeki optimizasyon tekniklerinin – algoritmalarının da içerisinde toplandığı bir sınıflandırma türüdür. Bu sınıf altındaki teknikler –optimizasyon yönelimli olsun veya olmasın– doğada yer alan rutin dinamiklerden ve zeki varsayılacak olgulardan cesaret alarak, daha karmaşık problemleri çözme yeterliğine sahip, çeşitli mantıksal ve matematiksel işlem adımları ile tasarlanmış olan YZ teknikleri olarak değerlendirilmektedir (Liu ve Tsui, 2006; Marrow, 2000; Shadbolt, 2004; Yang, 2010b).

- **Biyo-Esinli Hesaplama (Bio-Inspired Computing):** Biyolojik anlamda esinlenilmiş hesaplama yaklaşım, yöntem ve tekniklerine dayalı olarak ortaya çıkmış olan Biyo-Esinli Hesaplama (BEH-BIC), içerisinde kimi zeki optimizasyon tekniklerinin de yer aldığı ve genel olarak biyolojik unsurlardan yola çıkarak geliştirilmiş olan teknikleri içeren bir sınıflandırmadır (Mange ve Tomassini, 1998; Mohan, 2008; Pintea, 2014). BEH, tamamen zeki optimizasyon tekniklerine adanmamış bir sınıflandırma olsa da, bünyesinde bu tür tekniklerden de barındırdığı için ifade edilmesi gerekmektedir. Literatürde DEH ve BEH'in birbiri ile denk düzeyde veya birbirinin yerine kullanıldığını görmek mümkündür (Ancak bu çalışmada BEH, DEH'in bir alt-alanı olarak kabul edilmiştir). Çünkü biyolojik olgu ve olayların doğanın birer parçası olduğu ve doğadaki dinamiklerin, bu olgu – olayların üzerinde daha kapsayıcı nitelikte güçler içerebildiği düşünülmektedir.

- **Sürü Zekâsı (Swarm Intelligence):** Sürü Zekâsı (SZ-SI), zeki optimizasyon tekniklerinin sınıflandırılmasında belki de en kapsayıcı olan, DEH'in bir

başka alt-alanı olarak bilinmektedir. Hatta BEH bağlamında da değerlendirilebilen bir alt-alandır (Mohan, 2008; Yang ve ark., 2013). Doğadaki canlı sürülerinin merkezci olmayan, ortak davranışlarından esinlenerek geliştirilmiş olan, esnek ve etkin yapıdaki teknikleri içeren SZ (Bonabeau ve ark., 1999; Garnier ve ark., 2007; Hassanien ve Emary, 2016; Kennedy ve ark., 2001; Köse, 2015), bu tez çalışmasındaki algoritmalar için de taban vazifesi görmüştür (SZ bir sonraki alt-başlıklar bünyesinde daha detaylı ele alınmıştır).

- **Evrimsel Hesaplama (Evolutionary Computation):** Evrimsel Hesaplama (EH-EC), adından da anlaşılacağı üzere, Charles Robert Darwin'in Evrim Teorisi'ne (Darwin, 1859; 2015 (Çev.)) dayalı unsurlardan ve olgulardan esinlenerek geliştirilmiş olan teknikleri içeren bir sınıflandırma türüdür (Bäck ve ark., 1997; Fogel, 2000; 2006: 33-51; Mohan, 2008). Tıpkı SZ gibi genellikle zeki optimizasyon tekniklerini bünyesinde toplayan bu sınıf içerisinde yer alan çözüm tekniklerinde, işe koşulan çözüm nesnelerinin çeşitli evrimsel mekanizmalarla değiştirilmesi ve geliştirilip iyileştirilmesi gibi süreçler yer almaktadır (Bäck ve ark., 2000; Fogel, 2006: 59-246; Karaboğa, 2014). Burada, doğada ortaya çıkan ve 'kendiliğinden optimizasyon' süreci bağlamında değerlendirebileceğimiz evrimin (Boettcher ve Percus, 2000; Darwin, 1859; 2015 (Çev.); Mayr, 1988), zeki optimizasyon tekniklerinin inşasındaki kaçınılmaz kullanımı dikkat çekmektedir. Sonuç itibariyle EH'de amaç, arzu edilen çözüme yakın çözüm unsurlarının daha iyi hale getirilmesi suretiyle tutarlı sonuçların olasılığını da artırmaktır (Benzer biçimde nispeten kötü çözümlere ulaşan çözüm elemanlarının da elimine edilmesi ya da yenilerle değiştirilmesi suretiyle de çözüm elemanları bütününde kalite artışı çabaları da işe koşulabilmektedir). EH'i DEH'in bir alt-alanı olmasının yanında, SZ kapsamı içerisinde ve BEH ile ilişkili bir yapıda da kabul edebiliriz.

Anlaşılacağı üzere zeki optimizasyon teknikleri, aynı anda birden fazla sınıflandırma türüne tabi olabilmektedir. Çalışma kapsamında tasarlanıp geliştirilen YZ tabanlı optimizasyon algoritmalarını daha iyi anlamak adına, detaylarına daha fazla değineceğimiz sınıflandırma türü SZ olacaktır.

2.4. Sürü Zekâsı

SZ, ilk olarak Gerardo Beni ve Jing Wang tarafından (1993) hücresel robotik sistemler bağlamında önerilmiş olan bir kavram olmakla beraber, YZ araştırma alanı içerisinde, merkezi bir kontrole bağlı olmayan, bireysel zekânın ötesini gerektiren problemlere karşı ortak zekâda davranışlar sergileyen ve yine kendi kendine organize olabilen, doğal ya da yapay sistemler için kullanılan bir yaklaşımı ifade etmektedir (Blum ve Li, 2008; Engelbrecht, 2006; Garnier ve ark., 2007; Köse, 2015; Krause ve ark., 2010; Li ve ark., 2017). Bu yaklaşım içerisinde incelenebilen sistemler genel olarak doğadan esinlenilmiş olmakla beraber, çok sayıda bireyin problemlere karşı ortak davranış yaklaşımı içerisinde birleştiği, birbirleriyle ve içerisinde bulundukları karmaşık ortamla etkileşimin söz konusu olduğu bir bütüne karşılık gelmektedir (Andreasik, 2009; Bonabeau ve ark., 1999; Kennedy ve ark., 2001; Pan, 2016). SZ ile alakalı olarak değerlendirebileceğimiz, doğada yer alan sistemler genelde kuş sürüleri, balık sürüleri, karıncalar, arılar ve ortak davranış yönünde kabul edilebilecek her türlü canlı – cansız gruplar (Örneğin; bakteriler, akarsular… vb.) olabilmektedir (Blum ve Li, 2008; Garnier ve ark., 2007; Kennedy, 2006; Parpinelli ve Lopes, 2011).

Bir sistemin SZ kapsamında kabul edilebilmesi için taşıması gereken temel özellik ve işlevleri, literatürü ve araştırma tecrübelerimizi dikkate almak suretiyle, kısaca şöyle sıralayabiliriz (Ahmed ve Glasgow, 2012; Dorigo ve Birattari, 2007; Engelbrecht, 2006; Hassanien ve Emary, 2016; Köse, 2015; Yang, 2010b; Yang ve ark., 2013):

- Tipik bir sürüyü oluşturan, çok sayıda doğal veya yapay nesne – unsur,
- Sistem içerisinde yer alan nesne – unsurlar arası veya nesneler – unsurlar ile bulundukları ortam arası etkileşimsel süreçler,
- Spesifik anlamda anlaşılamayan, ancak sistem bütününe baktığında ‘zeki’ kabul edebileceğimiz davranışlar – hareketler,
- Sistemde yer alan nesne – unsurları kontrol eden bir merkezi iradenin olmaması,

- Sistemdeki ortak davranışlar – hareketlere yönelik kendiliğinden ortak kurallar bütünü.



Şekil 2. 5. SZ tabanlı bir sistemde temel özellik ve işlevler.

Belirtilen bütün özellik ve işlevler zeki optimizasyon ile ilişkili temel yaklaşımlarla harmanlandığında, ortaya etkin optimizasyon algoritmaları çıkmaktadır. Ayrıca, SZ tabanlı zeki optimizasyon tekniklerinin etkin ve verimli çözümler üretmesi noktasında, zaman içerisinde 'olmazsa olmaz' hale gelmiş çeşitli kabuller de bulunmaktadır.

2.4.1. Sürü zekâsı tekniklerinde temel kabuller

Tecrübelerimizi dikkate almak suretiyle, ilgili yaklaşıma dâhil tekniklerin genellikle tipik özellik ve işlevleri olarak da görebileceğimiz ve yeni tekniklerin tasarlanıp geliştirilmesinde dikkate alınabilecek temel kabulleri şöyle açıklayabiliriz:

- Başlangıç parametreleri (*N* adet çözüm nesneleri – parçacıklar, tekniğe özgü değişkenler),

- Başlangıçta belirlenmiş çözüm uzayı sınırları (Örneğin; çözümün -100 ile +100 değer aralığını içerisinde aranması),

- Optimizasyon sürecine yönelik durma kriteri (Örneğin; iterasyon sayısı, arzu edilen bir çözüm değeri… vb.),

- Çözüm sürecinde, optimum değere ulaşma noktasında çözüm nesnelerinin (metnin bundan sonraki kısımlarında parçacık olarak anılacaktır.) arzu edilen değerler çevresinde hareket halinde olan çözüm nesnelerine mantıksal ve matematiksel yollarla yaklaştırılması,

- Çözüm sürecine sürülen parçacıkların, birbirleriyle ve çözüm uzayıyla etkileşim içerisine girmesine imkân sağlayacak parametreler (Örneğin; her bir parçacığın hız, parlaklık gibi, esinlenilen dinamiklerin metaforları niteliğinde parametrelerinin olması),

- Çözüm sürecinde parçacıkların rastgele yürüyüş yapabilmelerini sağlayacak, hareket yönelimli matematiksel yaklaşımlar,

- Çözüm sürecinde kimi başarılı parçacıkların arzu edilen çözüme daha hızlı ulaşabilmesi adına parametrelerini güncelleyen gerekli matematiksel düzenlemeler,

- Evrimsel hesaplama yönünde tasarlandıkları takdirde, parçacıkların buldukları çözümlere göre geliştirilmesini veya elimine edilmesini sağlayacak gerekli matematiksel ifadeler.

Bu kabuller aslında SZ algoritmalarının iskeletini oluşturan temel kabuller olmakla beraber; bir algoritmanın literatürde farklı bir yaklaşım olarak kabul edilmesine sebep olarak, ortaya koyduğu performans ve çalışma mekanizmasıyla alakalı alternatif durumlar gibi çeşitli faktörler de bulunmaktadır.

2.4.2. Sürü zekâsı tekniklerinde alternatif durumlar

İlgili literatürü incelediğimiz zaman, çeşitli alternatif durumların teknikler içerisinde ve çözüm sürecinde kullanıldığını görebiliriz. Bu çalışmada ortaya konulan algoritmaların tasarlanmasında hangi kıstaslara dikkat edildiğini daha iyi anlamak adına bu alternatif durumları kısaca irdelemekte fayda vardır. Yaygın görülen alternatif durumlar, güncel teknik – algoritma örnekleriyle birlikte şu şekildedir:

- **Rol Dağılımlı Çözüm:** Bazı SZ tekniklerinde, parçacıkların rol dağılımları söz konusu olmaktadır. Bu tür tekniklerde elde ettikleri çözüm değerlerine ya da çözüm uzayında bulundukları konumlara göre parçacıklar arası rol dağılımları gerçekleştirilmektedir. Bu çözüm yöntemine en iyi örnek, Yapay Arı Kolonisi [YAK – Artificial Bee Colony (ABC)] verilebilir (*Bkz.* ikinci bölüm, alt-başlık 2.5.9.).

- **Evrimsel Süreçli Çözüm:** Oldukça sık tercih edilmekle birlikte, zeki optimizasyon tekniklerinin sınıflandırılması noktasında da değindiğimiz EH altında da incelenebilen tekniklerde, çözümü arama süreci sırasında parçacıkların buldukları değerlerin kalitesine, konumlarına veya sürecin gerekliliklerine göre evrimleşmesi söz konusu olmaktadır. Bu noktada, parçacıkların bazıları çaprazlama ile yeni parçacıklar oluşturabilmekte, bazı parçacıklar mutasyon ile farklı özellikler içeren parçacıklara dönüşebilmekte, ya da en basitinden bazı parçacıkların elimine edilip, bazılarının da geliştirilmesi yaklaşımları izlenebilmektedir. Bu çözüm yöntemine yönelik örnek verebileceğimiz başlıca teknikler Diferansiyel Evrim Algoritması [DE – Differential Evolution (DE)], Genetik Algoritma [GA – Genetic Algorithm (GA)] ve Klonal Seçilim Algoritması [KSA – Clonal Selection Algorithm (CSA)] şeklindedir (*Bkz.* ikinci bölüm, sırasıyla alt-başlık 2.5.3., 2.5.5., ve 2.5.7.). Ayrıca, bir önceki madde altında ifade edilen YAK ile birlikte Karınca Koloni Optimizasyonu [KKO – Ant Colony Optimization (ACO)] (Dorigo ve ark., 1991; Dorigo ve ark., 1996) ve Emperyalist Rekabetçi Algoritma [ERA – Imperialistic Competitive Algorithm (ICA)] (*Bkz.* ikinci bölüm, alt-başlık 2.5.4.) gibi bazı teknikler de evrimsel süreçli çözüm kapsamında kabul edilebilen, ancak bizzat Evrim Teorisi'ne yönelik özellik ve mekanizmalardan esinlenmeden; çeşitli seçilim, geliştirme, eliminasyon mekanizmaları kullanan teknikler olarak öne çıkmaktadır.

- **Graf Tabanlı Çözüm:** Graf tabanlı çözüm, esasında tekniğin kendi doğasında da birtakım farklılıkları gerektiren bir alternatif durumdur. Bunun en önemli sebebi kuşkusuz ki yöntemin temelinde kombinasyonel optimizasyonun yer almasıdır. İlgili yöntem dâhilinde çözüm uzayı, belirli düğümlerden oluşan bir graf yapısı altında tanımlanmaktadır. Dolayısıyla bu çözüm uzayına yönelik problemler ve çözüm süreci de farklılıklar taşımaktadır. Bu tez çalışmasının kapsamı dışında kalan, graf tabanlı çözüme dayalı SZ teknikleri için KKO ve Zeki Su Damlaları Algoritması [ZSD – Intelligent Water Drops Algorithm (IWDs)] (Shah-Hosseini, 2009) başlıca örnekler olarak verilebilir.

- **Gruplar Arası Mücadeleli Çözüm:** Literatürdeki bazı teknikler içerisinde, parçacıkların gruplara ayrılması ve çözüm sürecinin gruplar arası mücadeleye dayalı olarak devam ettirilmesi şeklinde bir yöntem de izlenmektedir. Bu yönteme en iyi örnekler Emperyalist Rekabetçi Algoritma [ERA – Imperialistic Competitive Algorithm (ICA)] ve Guguk Kuşu Araması [GKA – Cuckoo Search (CS)] teknikleri verilebilmektedir (*Bkz.* ikinci bölüm, alt-başlık 2.5.4., ve 2.5.6.). Gruplar arası mücadeleli çözüm, rol dağılımlı çözüme benzer yapıda olsa da, bu yöntemde çözüme gruplar arası değerlendirmeler neticesinde (mücadele ile ulaşılması) söz konusu olmaktadır.

SZ'ye yönelik olarak, okuyucunun bilgilendirilmesinin önemli olduğunu düşündüğümüz konuların ardından, bir sonraki alt-başlık kapsamında, ilgili literatürde yer alan başlıca teknikler kısaca açıklanmıştır.

2.5. Başlıca Sürü Zekâsı Teknikleri – Algoritmaları

SZ, araştırmacıların oldukça ilgisini çeken bir zeki optimizasyon alanı olarak değerlendirilebilmektedir. Temmuz 2017 itibariyle ilgili literatüre kazandırılmış olan çok sayıda farklı tekniğin olduğunu görebiliriz [Tekniklerin Temmuz 2017'deki yaklaşık sayısını ve hangileri olduğunu ifade eden kapsamlı, ortak yapıda bir kaynağa henüz rastlanmasa da, önde gelenlerini anlamak adına (Hassanien ve Emary, 2016; Köse, 2015; Parpinelli ve Lopes, 2011; Yang ve ark., 2013) gibi güncel referanslara başvurulabilir]. Bu tekniklerin tümünü bu metin içerisinde ele almak, metni çalışmanın amacı dışına taşıyacağı gibi, verimli bir inceleme de sunmuş olmayacaktır. Bunun yerine, ilgili literatürde çalışmalara sıklıkla konu olan tekniklerden önde gelenlere ve son dönemde geliştirilmiş olan tekniklerden bazılarına değinmek yeterlidir. Değinilen teknikler (ve hatta bunların çeşitli varyasyonları), bu tez çalışmasında geliştirilen algoritmaların değerlendirme süreçlerinde kullanılan 'rakip' teknikler olarak da kabul edilmiştir. Bu bağlamda söz konusu teknikleri şöyle listeleyebiliriz:

- Bakteriyel Yiyecek Arama Optimizasyonu [BYAO – Bacterial Foraging Optimization (BFO)]

- Çiçek Tozlaşma Optimizasyonu [ÇTO – Flower Pollination Optimization (FPO)]
- Diferansiyel Evrim [DE – Differential Evolution (DE)]
- Emperyalist Rekabetçi Algoritma [ERA – Imperialistic Competitive Algorithm (ICA)]
- Genetik Algoritma [GA – Genetic Algorithm (GA)]
- Guguk Kuşu Araması [GKA – Cuckoo Search (CS)]
- Klonal Seçilim Algoritması [KSA – Clonal Selection Algorithm (CSA)]
- Parçacık Sürü Optimizasyonu [PSO – Particle Swarm Optimization (PSO)]
- Yapay Arı Kolonisi [YAK – Articial Bee Colony (ABC)]

2.5.1. Bakteriyel yiyecek arama optimizasyonu

Bakteriyel Yiyecek Arama Optimizasyonu [BYAO – Bacterial Foraging Optimization (BFO)], ilk temelleri Liu ve Passino (2002) tarafından ortaya konulmuş olan ve E. Coli (Koli Basili) adlı bakterinin yiyecek kaynaklarını arama süreçlerine dayanan bir SZ tekniğidir (Passino, 2002). BYAO genel olarak Koli Basili bakterilerinin yiyecek kaynaklarını arama süreçlerine yönelik farklı hareketlerinden esinlenmekte ve tabi ki her tek hücreli canlı gibi Koli Basili de yaşamını sürdürme bağlamında tabi olduğu üreme (bölünme) ya da elimine olma gibi durumları da dikkate almaktadır (Gazi ve Passino, 2011; Passino, 2012).

BYAO'nun temellerini daha iyi anlamak adına, Koli Basili bakterilerinin hareketlerine ve yaşam süreçlerine kısaca değinmekte fayda vardır. Buna göre, bir Koli Basili bakterisi, içerisinde bulunduğu ortamın (maddenin) yoğunluğuna ve kendi yaşam koşullarına uygunluğuna göre çeşitli kemotaksislerde bulunabilmektedir. Üzerlerinde kamçılar taşıyan Koli Basili bakterileri, yiyecek kaynakları açısından uygun olan ortamlarda kamçılarını saatin tersi yönünde döndürerek, daha çok yüzmeye (swimming), daha az yuvarlanmalara (tumbling) dayalı hareketler etmekte, tersi durumdaki ortamlarda ise –daha uygun ortamlara geçme eğiliminden dolayı–

kamçılarını saat yönünde döndürerek söz konusu hareketlerini daha çok yuvarlanma içerecek şekilde yerine getirmektedir (Das ve ark., 2009; Passino, 2012). Bu süreçler sırasında Koli Basili bakterileri ikiye bölünerek üreyebilmekte veya ortamdaki zararlı kaynakların varlığına ya da ortalama yaşam süreçlerine göre bu bakterilerin yaşamları sona erebilmektedir. Yine Koli Basili bakterileri, uygun yaşam ortamları içerisinde bulunduklarında diğer Koli Basili bakterilerini de ilgili ortama çekmek adına bir madde salgılamakta ve ortamın bakteri yoğunluğu bu yayılma şekliyle artmaktadır (Das ve ark., 2009; Passino, 2012). BYAO da işte bütün bu süreçlerden esinlenerek ve birer bakteri olarak kabul ettiği parçacıkları çözüm uzayında sürü odaklı bir biçimde kullanarak, çeşitli çözüm süreçlerini işe koşmaktadır. Bu bağlamda, Kemotaksis, Yuvarlanma, Yüzme, Üreme ve Eliminasyon – Dağılım gibi farklı fazların karışımını içermektedir.

Açıklanan özellik ve işlevler bağlamında tasarlanmış olan BYAO'nun algoritmik çözüm adımlarını şu şekilde ifade edebiliriz (Passino, 2012):

Adım 1 (Kurulum Fazı). N adet bakteri parçacığını (potansiyel çözüm değişkenlerini) çözüm uzayına rastgele dağıt. Algoritma parametreleri değerlerini (Örneğin; N_k: maksimum kemotaksis sayısı, $N_{yüz}$: yüzme hareket sayısı, $N_{ü}$: üreme sayısı, N_e: maksimum eliminasyon sayısı, çekme ve itme katsayıları, o_{ed}: eliminasyon-dağılım olasılığı) ata. Çözülecek probleme dair gerekli düzenlemeleri gerçekleştir.

Adım 2. Bakterilerin (potansiyel çözüm değişkenlerinin) konumlarına göre amaç fonksiyon değerini (fitness) hesapla.

Adım 3. Aşağıdaki adımları, N_e kadar tekrarla:

(Her amaç fonksiyon boyutu bağlamında;)

Adım 3.1 (Kemotaksis Fazı). Aşağıdaki adımları N_k değeri kadar, her bakteri için gerçekleştir:

Adım 3.1.1. Sıradaki bakterinin (potansiyel çözüm değişkeni) konumuna göre amaç fonksiyon değerini (fitness) hesapla.

Adım 3.1.2. *Denklem 2.1.* ile ilgili bakterinin amaç fonksiyon değerini (fitness) hücreden hücreye çekici etkisi ile güncelle. Yüzme Fazı'na kadar bu değeri tut.

Adım 3.1.3 (Yuvarlanma Fazı). [-1, 1] aralığında amaç fonksiyon boyutu kadar rastgele sayı üret. İlgili bakteri için *Denklem 2.2.*'ye göre yuvarlanma işlemini gerçekleştir.

Adım 3.1.4. Bakterinin (potansiyel çözüm değişkeni) konumuna göre amaç fonksiyon değerini (fitness) hesapla ve *Denklem 2.1.* aracılığıyla, ilgili bakterinin amaç fonksiyon değerini (fitness) hücreden hücreye çekici etkisi ile güncelle.

Adım 3.1.5 (Yüzme Fazı). Aşağıdaki adımları $N_{yüz}$ değeri kadar, ilgili bakteri için gerçekleştir:

Adım 3.1.5.1. Eğer bakterinin son amaç fonksiyon değeri (fitness), Yüzme Fazı'ndan önce saklanandan daha iyi ise artık bu yeni değeri tut.

Adım 3.1.5.2. *Denklem 2.2.* ile hesaplanacak yer değiştirme değerine göre ilgili bakterinin – parçacığın tutulan amaç fonksiyon değerini (fitness) güncelle.

Adım 3.1.6. Eğer bütün bakteriler henüz işleme tabi tutulmadıysa sıradaki bakteriye geç ve Adım 3.1.1.'e dön.

Adım 3.2 (Üreme Fazı). *Denklem 2.3.* ile her bakterinin sağlık durumunu hesapla ve bu değerlere göre büyükten küçüğe doğru hepsini sırala.

Adım 3.3. Belirlenen kriter değere göre en kötü bakterileri elimine (yok) et. En iyi durumdaki bakterilerin ise üremesini (bölünmesini) sağla. Yeni bakteriler ebeveynlerinin yerlerinde yer almaktadır.

Adım 3.4. $N_{üi}$ değerine henüz ulaşılmadıysa, bu değere ilişkin sayacı bir artır ve Adım 3.1.'e dön ve yeni nesil ile işlemlere devam et.

Adım 3.5 (Eliminasyon – Dağılım Fazı). o_{ed} değerine göre her bir bakteriyi rastgele yeni bir konuma aktar.

Adım 4. Süreçler sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

$$f_{hh}\left(bakteri, B(j,k,l)\right) = \sum_{i=1}^{N} Fe_{hh}\left(bakteri, bakteri_i(j,k,l)\right) = \sum_{i=1}^{N}[-b_{çekici} * \exp(-a_{çekici} * \sum_{m=1}^{B}(bakteri_m - bakteri_{m_i})^2)] + \sum_{i=1}^{N}[-c_{itici} * \exp(-a_{itici} * \sum_{m=1}^{B}(bakteri_m - bakteri_{m_i})^2)] \quad (2.1.)$$

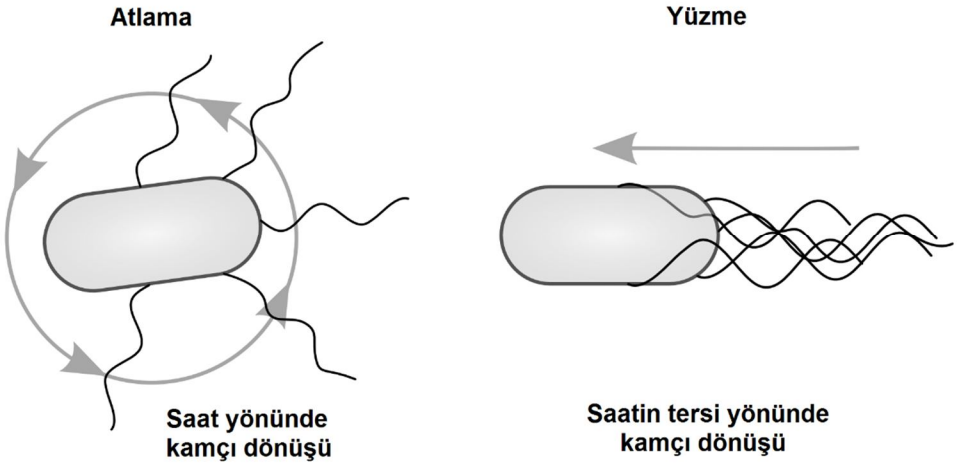
Denklem 2.1. kapsamında, $f_{hh}\left(bakteri, B(j,k,l)\right)$ amaç fonksiyona eklenecek olan değeri, *N* toplam bakteri sayısını, *B* toplam optimize edilecek değişken sayısını, $a_{çekici}$, $b_{çekici}$, a_{itici}, c_{itici} ise hücreden hücreye çekici etkisi hesaplamasına yönelik katsayıları ifade etmektedir.

$$bakteri_i(j+1,k,l) = bakteri_i(j,k,l) + Y(i) * \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}} \quad (2.2.)$$

Denklem 2.2.'de, *i* bakteri sıra numarasını, *j, k,* ve *l* içerisinde bulunulan (sırasıyla) kemotaksis, üreme, ve eliminasyon – dağılım faz sırasını ifade etmektedir. Diğer yandan, $Y(i)$ rastgele bir yöne yapılan yuvarlanmayı, Δ ise [-1, 1] aralığında rastgele değer vektörüne karşılık gelmektedir.

$$f_{sağlık}^{i} = \sum_{j=1}^{N_k+1} f(i,j,k,l) \quad (2.3.)$$

Denklem 2.3.'te, *k* kemotaksis işlem sıra numarasını, *i, j, k,* ve *l* içerisinde bulunulan (sırasıyla) bakteri numarası, kemotaksis, üreme, ve eliminasyon – dağılım faz sırasını ifade etmektedir.



Şekil 2. 6. BYAO'ya esin kaynağı Koli Basili hareketleri (Egbert ve ark., 2010).

BYAO tekniğine ilişkin olarak geliştirilen varyasyonlardan literatürde güncel olanları ve dikkat çekenleri şöyle belirtebiliriz: (Cao ve ark., 2016; Chen ve ark., 2011;

Chen ve ark., 2017; Datta ve ark., 2008; Diao ve ark., 2016; Franklin ve Kumar, 2016; Hou ve Zhang, 2016; Kora ve Kalva, 2017; Niu ve ark., 2013a; Panda ve ark., 2017; Pandit ve ark., 2012; Tang ve ark., 2017; Wu ve ark., 2007; Yang ve ark., 2016).

2.5.2. Çiçek tozlaşma optimizasyonu

Çiçek Tozlaşma Optimizasyonu [ÇTO – Flower Pollination Optimization (FPO)], Yang (2012) tarafından geliştirilmiş bir diğer SZ tekniğidir [Yang, GKO ve ÇTO dışında Ateş Böceği Algoritması (Firefly Algorithm) ve Yarasa Algoritması (Bat Algorithm) tekniklerini de geliştirmiştir: (Yang, 2009; 2010a; 2010b). Bu çalışmada sadece GKO ve ÇTO'nun içerilmesi, farklı tekniklere yeter düzeyde yer ayrılması, GKO'nun kendini kanıtlamış başarıları ve ayrıca ÇTO'nun da Yang'ın diğer tekniklerine göre daha yeni olmasıdır.]. ÇTO'nun temel esin kaynağı, anlaşılacağı üzere doğadaki tozlaşma olaylarıdır. Benzeri tekniklerde olduğu gibi bu teknik de odak noktası olarak aldığı tozlaşma olayıyla ilişkili kavramlar ve olgulardan faydalanmaktadır. ÇTO ile ilgili bu kavram ve olguları kısaca şöyle toparlayabiliriz:

Doğada yer alan bitki türlerinden yaklaşık %80 kadarı çiçekli bitkiler olarak bilinmektedir ve bu noktada çiçekli bitkilerde söz konusu olan; tozlaşma adı verilen süreç, çeşitli alternatif yollarla gerçekleşen ve ilgili bitkilerin yeniden üretilerek, türlerinin devamlılığını sağlayan bir dizi doğal dinamik olarak açıklanabilmektedir (Yang, 2012; Yang ve ark., 2014). Bu bağlamda tozlaşma sürecini, biyotik ve abiyotik olmak üzere iki farklı şekillerde gözlemlemek mümkün olmaktadır. Biyotik tozlaşma süreçleri, böcekler ve hayvanlar gibi çiçeklere konan (veya yakınlarından geçen) 'taşıyıcı' nitelikteki unsurlarla gerçekleşirken, abiyotik tozlaşma ise su, rüzgar gibi polenleri doğal dinamiklerle yayılmasını sağlayan unsurlarla gerçekleşen tozlaşma süreci olarak tanımlanabilmektedir (Glover, 2007; Yang ve ark., 2014). İlgili unsurlar dikkate alınmak suretiyle, uzun mesafede, biyotik unsurlar sayesinde meydana gelen ve farklı çiçekler arasında olabilen tozlaşma çapraz tozlaşma (cross-pollination) olarak, aynı çiçekte veya yakındaki aynı tür çiçekler arasında olan tozlaşma ise kendi kendine tozlaşma (self-pollination) olarak adlandırılmaktadır (Yang, 2012).

Açıklanan doğal dinamikler dikkate alınmak suretiyle geliştirilen ÇTO'da şu süreçler kapsamında bir algoritmik çözüm süreci işe koşulmaktadır (Yang, 2012; Yang ve ark., 2014):

- Küresel tozlaşma olarak adlandırılan ve biyotik unsurlar sebebiyle gerçekleştiği için Levy Uçuşları içeren çapraz tozlaşma işlemleri gerçekleştirilmektedir. Diğer yandan kendi kendine tozlaşma da yerel tozlaşma işlemi olarak yerine getirilmektedir.

- Çiçek istikrarı (flower constancy) adı verilen bir kriter ile çiçekler arası benzerlik ve dolayısıyla yeniden üretim olasılıkları dikkate alınmaktadır.

- Genelde yerel tozlaşmaya eğilimli olunmakla birlikte, yerel ve küresel tozlaşma süreçleri arası geçişler olasılıksal olarak yerine getirilmektedir.

İlk literatüre kazandırılmasından (Yang, 2012) sonra çok-amaçlı optimizasyon düzeyine getirilen (Yang ve ark., 2014) ÇTO tekniğinin algoritma detayları şu şekildedir:

Adım 1 (Kurulum Fazı). *N* adet çiçeği – parçacığı (potansiyel çözüm değişkenlerini) çözüm uzayına rastgele dağıt. Algoritma parametreleri değerlerini ata, geçiş olasılığı parametresini (*go*) belirle. Çözülecek probleme dair gerekli düzenlemeleri gerçekleştir.

Adım 2. Çiçeklerin – parçacıkların (potansiyel çözüm değişkenlerinin) konumlarına göre amaç fonksiyon değerini (fitness) hesapla. En iyi olanı tespit et.

Adım 3. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her parçacık için; her amaç fonksiyon boyutu için;)

Adım 3.1 (Küresel – Yerel Tozlaşma Fazı). Rastgele bir değer üret. Eğer üretilen değer *go* değerinden düşük ise *Denklem 2.4.* ve Levy Uçuşları (adım vektörü: *L*) kapsamında küresel tozlaşma sürecini gerçekleştir. Eğer üretilen değer *go* değerine eşit veya bu değerden büyük ise [0, 1] aralığında uniform dağılım ve *Denklem 2.5.* bağlamında yerel tozlaşma sürecini işe koş.

Adım 3.2. Çiçeklerin – parçacıkların (potansiyel çözüm değişkenlerinin) güncellenmiş konumlarına göre amaç fonksiyon değerini (fitness) hesapla.

Adım 3.2. O andaki en iyi amaç fonksiyon değerinden daha iyisi bulunmuşsa küresel en iyi değeri (ve dolayısıyla değişken konumlarını) güncelle.

Adım 4. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

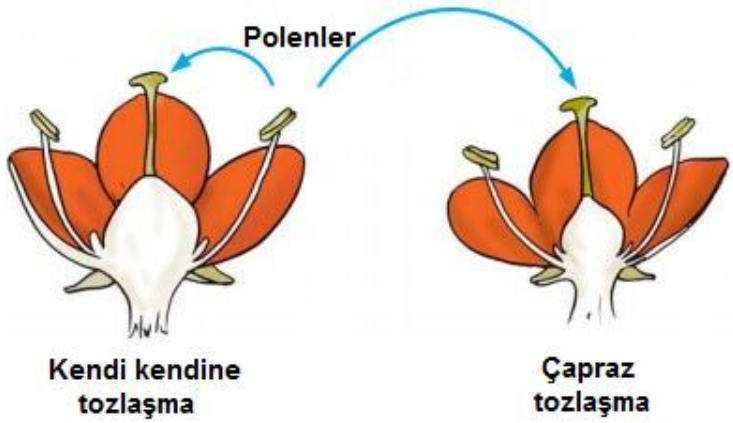
$$x_i^{z+1} = x_i^z + L * (x_i^z - k_*) \tag{2.4.}$$

$$L \backsim \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}, \qquad (s \gg s_0 > 0)$$

Denklem 2.4.'te *i* değişken sırasını temsil ederken, *z* zaman değerini (iterasyon – döngü anlamında) temsil etmektedir. Yine *L* parametresi de Levy Uçuşları'ndan kuvvetle tozlaşma gücüne karşılık gelmektedir.

$$x_i^{z+1} = x_i^z + \in * (x_j^z - x_k^z) \tag{2.5.}$$

Denklem 2.5. kapsamında, *i*, *j*, ve *k* farklı çiçekleri temsilen sıra numaralarına karşılık gelmekte, $\in$ ise rastgele yürüyüşü ortaya çıkaracak dağılım işleci anlamında kullanılmaktadır.



Şekil 2. 7. ÇTO'ya esin kaynağı tozlaşma süreçleri (AMRITA Learning, 2013).

ÇTO ile ilgili literatüre kazandırılmış çeşitli varyasyonlar da bulunmaktadır [Bunlardan güncel olanlar ve dikkat çekenler için: (Abdel-Baset ve Hezam, 2015; Agarwal ve Mehta, 2016; Dubey ve ark., 2015; El-henawy ve Ismail, 2014; Meng ve ark., 2017; Nabil, 2016; Salgotra ve Singh, 2017; Valenzuela ve ark., 2017; Wang ve Zhou, 2014; Zhang ve ark., 2016; Zhou ve Wang, 2016)].

2.5.3. Diferansiyel evrim

Diferansiyel Evrim [DE – Differential Evolution (DE)], Storn ve Price (1995; 1997) tarafından literatüre kazandırılmıştır bir SZ tekniğidir. Algoritmik yapısı itibariyle basit ve hızlı bir teknik olarak bilinmektedir (Price, 1996). Aynı zamanda tıpkı GA gibi bir EH tekniği olan DE, uyguladığı mutasyon süreci ile etkili sonuçlar elde edilmesinin önünü açmaktadır. DE'de uygulanan tipik mutasyon süreçlerinde rastgele seçilmiş amaç vektör (değişken değeri) çiftlerinin farklarına dayanan bir değerlendirme gerçekleştirilmektedir (Karaboğa, 2014). Mutasyon süreçlerinin yanında rekombinasyon ve seçilim işlemleri de uygulanarak evrimsel bir çözüm tercih edilmektedir.

DE'nin çözüm arama sürecinde öncelikli olarak, gerçek sayılardan oluşan kromozom yapılarıyla temsil edilen bireyler, yani rastgele üretilmiş amaç vektör (değişken) değerleri, yeni donör (donor) bireyler elde etmek için mutasyon işlemi içerisine dâhil edilmekte ve bu yolla arama uzayının da genişlemesi sağlanmaktadır. Mutasyon ile oluşturulan donör bireylerin ebeveynleriyle (target: oluşturuldukları bireyler – amaç vektör değişken değerleri) değerlendirilmelerini içeren rekombinasyon süreci sayesinde de, deneme (trial) bireyleri elde edilmektedir. Akabinde, deneme bireylerinin ebeveynlerle karşılaştırılarak –en iyilerin süreçte kalmaları adına– seçilim süreci gerçekleştirilmektedir (Arunachalam, 2008; Das ve Suganthan, 2011).

Açıklamalar eşliğinde, DE tekniğinin algoritma yapısını kısaca şu şekilde ifade etmemiz mümkündür (Arunachalam, 2008; Das ve Suganthan, 2011; Karaboğa, 2014):

Adım 1 (Kurulum Fazı). *N* bireyli popülasyonu temsilen, her bir bireyi (amaç vektör – değişken değerlerini) ilgili probleme göre, kromozom düzeninde oluştur. Başlangıç parametre değerlerini (Örneğin; ölçekleme faktörü, çaprazlama oranı) belirle.

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her birey için; her amaç fonksiyon boyutu için;)

Adım 2.1 (Mutasyon Fazı). Bireyin kromozom yapısı üzerinde *Denklem 2.6.*'daki mutasyon yaklaşımını uygulamak suretiyle donör bireyleri oluştur.

Adım 2.2 (Rekombinasyon Fazı). *Denklem 2.7.*'de ifade edilen rekombinasyon sürecini dikkate almak suretiyle deneme bireyleri oluştur.

Adım 2.3 (Seçilim Fazı). Ebeveyn ve deneme bireyleri için amaç fonksiyon değerleri (fitness) hesapla**.** Hesaplanan değerleri dikkate almak suretiyle, daha iyi – optimum bireyleri tercih ederek seçilim sürecini gerçekleştir.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

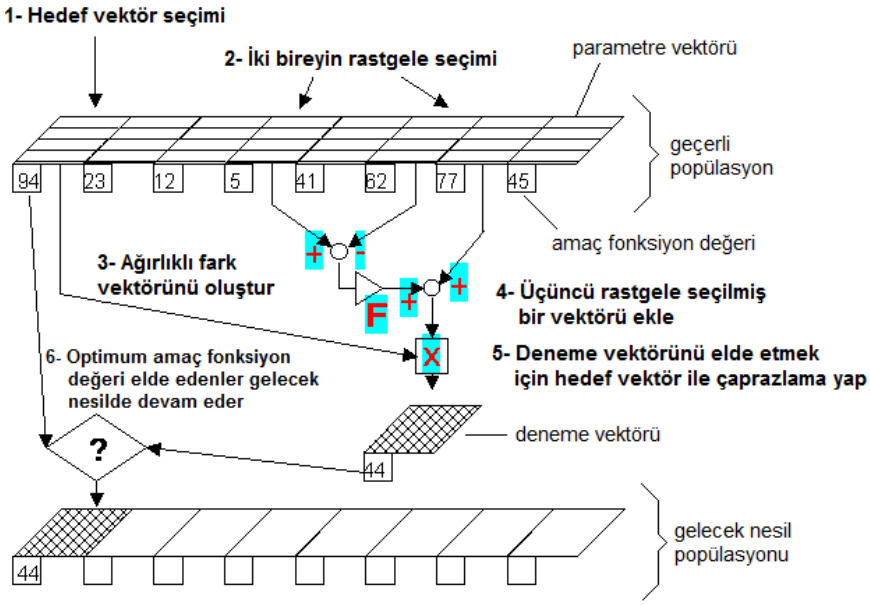
$$d_{i,Nes+1} = x_{r_1,Nes} + F * (x_{r_2,Nes} - x_{r_3,Nes}) \tag{2.6.}$$

Denklem 2.6.'da *d* donör bireyi, *i* gen sırasını – vektör indisini, *Nes* nesil sırasını, *r1*, *r2*, ve *r3* rastgele kromozom sırasını, *x* geni, ve *F* farklılık amlifikasyonunu sağlayan değişkeni temsil etmektedir.

$$d_{i,Nes+1} = (d_{1i,Nes+1}, d_{2i,Nes+1}, \dots, d_{Di,Nes+1}) \tag{2.7.}$$

$$d_{ji,Nes+1} = \begin{cases} d_{ji,Nes+1} & Eğer\ (rastb(j) \le ÇO)\ ya\ da\ j = rasti(i) \\ x_{ji,Nes} & Eğer\ (rastb(j) > ÇO)\ ve\ j \ne rasti(i) \end{cases}$$

Denklem 2.7.'de *j = 1, 2, ..., D* olmak üzere boyut sırasına, *rastb j.* sırada [0, 1] aralığından rastgele bir değere, *rasti* rastgele indis numarasına, ve *ÇO* çaprazlama sabitine karşılık gelmektedir.



Şekil 2. 8. DE algoritması genel işleyişi (Datta ve Kumar, 2015).

Literatürde DE'nin farklı varyasyonlarını da görmek mümkündür. Bu bağlamda, en güncel ve dikkat çeken çalışmaları şöyle sıralayabiliriz: (Das ve ark., 2008; Elsayed ve ark., 2013; Ghosh ve ark., 2011; Ho ve ark., 2010; Jia ve ark., 2011; Qin ve ark., 2009; Rahnamayan ve ark., 2008; Shih ve ark., 2017; Sun ve ark., 2012; Xue ve ark., 2003; Zhang ve Sanderson, 2009).

2.5.4. Emperyalist rekabetçi algoritma

Emperyalist Rekabetçi Algoritma [ERA – Imperialist Competetive Algorithm (ICA)], Atashpaz-Gargari ve Lucas'ın çalışmaları neticesinde (2007) literatüre kazandırılmış SZ odaklı bir tekniktir. ERA için literatürde insanlığın sosyal evrimiyle ilişkili bir EH tekniği gözüyle de bakılmaktadır (Marzband ve ark., 2015; Mozafari ve ark., 2012; Yousefikhoshbakht ve Sedighpour, 2013). Adından da anlaşılacağı üzere ERA, ülkelerin başka ülkeler üzerinde söz sahibi olarak, nüfuz etmesinden yola çıkarak tasarlanmıştır. Buna göre, ERA içerisinde çözüm uzayına dağıtılan ve birer ülke olarak kabul edilen parçacıklar elde ettikleri amaç fonksiyon değer(ler)ine (fitness) göre değerlendirilmekte ve iyi durumda olanlar diğerlerini koloni haline getirmektedir. Bu duruma ek olarak çözüm arayan ülkeler yine farklı etkilere maruz kalmakta ve bu etkiler çözüm sürecini de yönlendirmektedir.

ERA'da genel olarak algoritmik adımlar kapsamında şu alt-süreçlerden geçilmektedir (Atashpaz-Gargari ve Lucas, 2007; Nazari-Shirkouhi ve ark., 2010):

- **Asimilasyon:** Koloni niteliğinde olan ülkeler, emperyalist niteliğinde olmak üzere bağlı oldukları ülkeye (parçacığa) doğru çözüm uzayında hareket ederler. Bu süreçte daha iyi duruma geçen koloniler emperyalist konumuna geçebilirler.

- **Devrim:** Bazı koloni ülkelerde beklenmedik ani değişimler meydana gelebilir ve bu durum ilgili ülkelerin çözüm uzayındaki konumlarının ani değişimiyle ifade edilebilmektedir. Tıpkı asimilasyon durumunda olduğu gibi, devrim de daha iyi duruma geçen kolonilerin emperyalist konumundaki ülkelerle yer değiştirmesine sebep olabilir.

- **Rekabet:** Süreç sırasında çözüm uzayındaki emperyalist ülkeler birbirleriyle bir tür yarışma – rekabete sokulur ve sonuç itibariyle zayıf durumdaki

emperyalist ülkelerin kolonileri güçlü emperyalist ülkelerin safına geçebilmektedir.

Açıklamalardan da yola çıkmak suretiyle, ERA'nın algoritma yapısını genel olarak şöyle ifade edebiliriz (Atashpaz-Gargari ve Lucas, 2007; Xing ve Gao, 2014):

Adım 1 (Kurulum Fazı). N adet ülkeyi, çözüm uzayında rastgele oluştur. ERA'ya ilişkin temel algoritma parametrelerini (Örneğin; N_{emp}: emperyalist ülke sayısı, ξ: koloni etkisi) belirle. Probleme yönelik gereken düzenlemeleri yap.

Adım 2. Aşağıdaki adımları, çözüm uzayında tek bir emperyalist ülke ve ona bağlı koloniler kalıncaya kadar tekrarla tekrarla:

(Her amaç fonksiyon boyutu bağlamında;)

Adım 2.1 (Emperlist Oluşturma Fazı). Her ülke için amaç fonksiyon değeri (fitness) hesapla. N_{emp} kadar en iyi ülkeyi emperyalist ülke olarak belirle. *Denklem 2.8.* ve *Denklem 2.9.* yardımıyla her emperyalist ülkenin gücünü tespit et. Tespit edilen güçlerden yola çıkarak, *Denklem 2.10.* kapsamında her bir emperyalist ülkenin sahip olabileceği koloni ülke sayısını belirle ve ilgili emperyalist ülkelere sahip olabilecekleri sayı kadar koloni ülkeleri bağla.

Adım 2.2 (Asimilasyon Fazı). *Denklem 2.11.* aracılığıyla koloni ülkeleri bağlı oldukları emperyalist ülkelere doğru hareket ettir.

Adım 2.3 (Devrim Fazı). Asilimasyon Fazı sırasında bağlı oldukları emperyalist ülkeden daha iyi konuma gelen [amaç fonksiyon değeri (fitness) hesaplama] koloni ülkelerin konumları ile bağlı oldukları emperyalist ülkelerin konumlarını karşılıklı değiştir.

Adım 2.4 (Rekabet Fazı). *Denklem 2.12.* ile bütün emperyalist ülkelerin sahiplik gücünü tespit et. *Denklem 2.13.*, *Denklem 2.14.* ve *Denklem 2.15.*'i dikkate alarak zayıf durumdaki emperyalist ülkelerin kolonilerinden güçlü emperyalist ülkelere aktarma yap. Bütün kolonilerini kaybetmiş emperyalist ülkelerini elimine (yok) et.

Adım 3. Süreç sonunda emperyalist ülkenin sahip olduğu değer(ler) optimum değer(ler) olarak kabul edilmektedir.

$$F_n = f_n - \max_i\{f_i\} \tag{2.8.}$$

$$g_n = \left| \frac{F_n}{\sum_{i=1}^{N_{emp}} F_i} \right| \tag{2.9.}$$

Denklem 2.8. ve *Denklem 2.9.* bağlamında *F n.* emperyalist ülkenin gücünü, yani başka bir deyişle, normalize edilmiş amaç fonksiyon değerini, *f* ise yine aynı ülkenin (normalize öncesi) amaç fonksiyon değerini (fitness) temsil etmektedir. Yine N_{emp} tanımlanmış emperyalist ülke sayısı iken, *max* ifadesi ilgili amaç fonksiyon değerlerinden en büyüğü anlamındadır.

$$K_n = yuvarla\{g_n N_{kol}\} \tag{2.10.}$$

Denklem 2.10.'da N_{kol} belirlenmiş olan toplam koloni sayısına karşılık gelmekte ve bu denkleme ile K_n simgesi temsilinde, *n.* emperyalist ülkenin koloni sayısını tespit edilmektedir.

$$x \sim U(0, \beta \times m) \tag{2.11.}$$

$$\theta \sim U(-\gamma, \gamma)$$

Denklem 2.11. kapsamında kolonilerin emperyalist ülkelerine hareketi ifade edilmekle birlikte, *U* üniform dağılımı, β 1'den büyük bir sayıyı, m koloni ülke ve emperyalist ülke arası mesafeyi ifade etmekte; ayrıca üniform dağılımlı bir rastgele sayı olarak θ ve orijinal konumdan sapmayı sağlayacı olarak da γ ilgili denklem kapsamında dikkate alınmaktadır.

$$SF_n = F(emperyalist_n) + \xi * ort\{F(emperyalistkolonileri_n)\} \tag{2.12.}$$

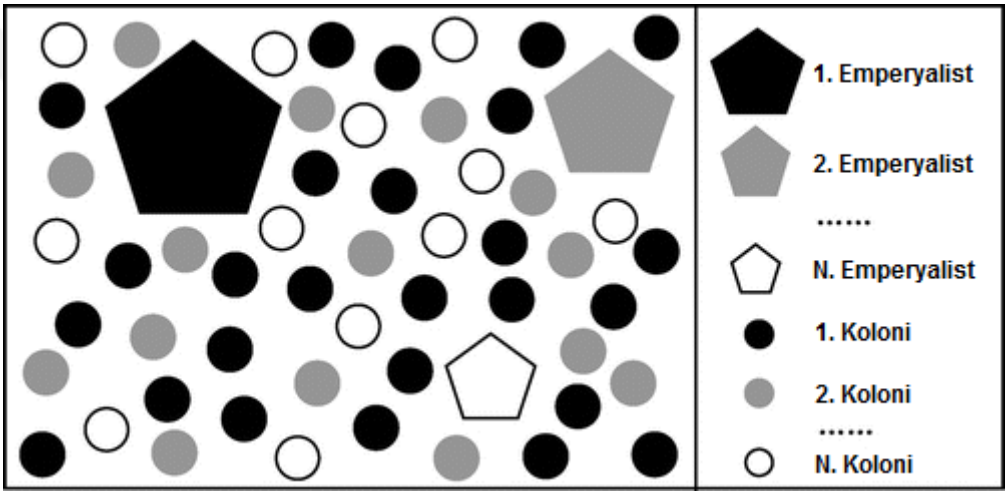
Denklem 2.12. bağlamında *TF* ilgili emperyalist ülkenin toplam sahiplik gücünü, yani *n.* emperyalist ülkenin toplan fonksiyon değerini (fitness), ξ koloni etkisi değerini ve *emperyalist* ifadesi, *n.* emperyalist ülkeyi temsil etmektedir. Diğer yandan *ort* ifadesi ise, *n.* emperyalist ülkenin kolonilerinin (burada her biri *emperyalistkolonileri* ile temsil ediliyor) fonksiyon değerinin ortalaması anlamına gelmektedir.

$$NSF_n = SF_n - \max_i\{SF_i\} \tag{2.13.}$$

$$o_{o_n} = \left| \frac{NSF_n}{\sum_{i=1}^{N_{emp}} NSF_i} \right| \quad (2.14.)$$

$$B = O - R \quad (2.15.)$$

Denklem 2.13., *Denklem 2.14.* ve *Denklem 2.15.* dikkate alındığında, *NSF* normalize edilmiş toplam fonksiyon değerini (sahiplik gücü), *max* ifadesi, *n.* emperyalistin *SF'*lerinden en yükseği, o_{o_n}, *n.* emperyalisting sahiplik olasılığını ifade etmektedir. Son olarak *Denklem 2.15.*'te gösterilen *B,* ilgili emperyalist ülkeye ait olan kolonilerin vektörünü, *O,* sahiplik olasılığı vektörünü, ve *R* üniform (0, 1) aralığında rastgele sayılar vektörünü tanımlamaktadır.



Şekil 2. 9. ERA'da örnek emperyalist ve koloni dağılımı (Sun ve Liang, 2014).

ERA tekniği için ilgili literatürde gerçekleştirilmiş olan geliştirme çalışmaları dikkate alındığında, özellikle kaotik sistemlerle ERA'nın birleştirildiği çalışmalar dikkat çekmektedir. Tahmin edileceği üzere, ERA'nın varyasyonlarına yönelik çalışmalar, –SZ tekniklerinin genelinde olduğu gibi– farklı problemlere dayalı ve algoritmik etkinliği – başarımı artırmaya dayalı yaklaşımlar çerçevesinde hayata geçirilmiştir [İlgili çalışmalardan bazı güncel olanlar ve dikkat çekenler için: (Abdechiri ve ark., 2010a; 2010b; Bahrami ve ark., 2010; Duan ve ark., 2010; Enayatifar ve ark., 2013; Mirhoseini ve ark., 2014; Niknam ve ark., 2011; Nourmohammadi ve ark., 2013; Talatahari ve ark., 2012a; Talatahari ve ark., 2012b; Zhang ve ark., 2009)].

2.5.5. Genetik algoritma

Genetik Algoritma [GA – Genetic Algorithm (GA)], Charles Robert Darwin (1859; 2015 (Çev.)) tarafından bilim dünyasına kazandırılan Evrim Teorisi'nin temel unsurları ve olgularından esinlenerek geliştirilmiş olan ilk YZ tekniğidir. Holland (1992:1975) tarafından literatüre kazandırılan GA, yapısı itibariyle genetik kavramları temel alan ve Evrim Teorisi ile açıklanan, seçilim (seleksiyon – selection), mutasyon (mutation) ve çaprazlama (crossover) süreçlerini gerçek yaşam tabanlı problemlerin çözümü için kullanan bir EH tekniği olarak da bilinmektedir. Esasında GA'da çözüm sürecinde kullanılan parçacıkların temsilinde –şu ana kadar açıklanan tekniklerden farklı olarak– kodlama yoluna gidilmektedir (Holland, 2012; Kramer, 2017; Mitchell, 1998). Bu kodlama, Holland'ın da (1995; 2012) ifade ettiği gibi, Ronald Aylmer Fisher tarafından ortaya konulan ve genlerin bir popülasyon boyunca aktarımını matematiksel yapıda açıklayan yaklaşımlardan (Fisher, 1958:1930) esinlenmektedir.

Genel olarak tipik bir GA, başlangıçta çözüm için kullanılacak ve seçilen bir kodlamayla kromozom şeklinde temsil edilen parçacıkların (bu alt-başlık altında bu noktadan sonra birey olarak ifade edilecektir) ilgili amaç fonksiyonu (ya da fonksiyonları) çerçevesinde elde ettikleri değerlere (fitness) göre çeşitli genetik işlemlere tabi tutulmasına dayanmaktadır. Buna göre, başlangıçtaki popülasyon elde edilen sonuçlara göre çaprazlama, mutasyon gibi süreçler işe koşularak, nispeten daha iyi durumdaki bireyler üzerinden yeni bireylerin elde edilmesi ve geliştirilmesi yoluna gidilmekte, her yeni nesille yapılan değerlendirmeler, istendik sonuç elde edilinceye kadar ya da belli bir durma kriteri sağlanıncaya kadar sürdürülmektedir (Goldberg, 2006; Karaboğa, 2014; Kramer, 2017). Karaboğa'ya (2014) göre bir GA'nın etkinliğini, çözümlerin nasıl temsil edildiği, bireylerin ve dolayısıyla başlangıç popülasyonunun nasıl kodlandığı, amaç fonksiyonu (ya da fonksiyonları), uygulanan genetik işlemler ve algoritma parametreleri belirlemektedir.

GA'da çözüm sürecinde kullanılan bireylerin her biri ilk başta çeşitli kodlama şekilleri kullanılarak temsil edilmektedir. Bu noktada bireyleri temsil eden kromozomların içeriği gen adı verilen kodlarla doldurulmaktadır. Kodlama için genellikle ikilik sayı sistemi (binary – bitler – 0 ve 1) kullanılabildiği gibi, alfabenin harfleri, ayrık sayılar, alternatif sayı kodlama biçimleri ya da geliştirilen alternatif

temsiller tercih edilebilmektedir (Balland ve ark., 2000; Gen ve Cheng, 2000; Janikow ve Michalewicz, 1991; Karaboğa, 2014; Patrascu ve ark., 2014; Whitley, 1994). Burada önemli olan bir husus da bu kodların amaç fonksiyon değeri (fitness) hesaplaması aşamasına gelindiğinde ilgili fonksiyon(lar)a nasıl dâhil edileceğinin iyi belirlenmiş olmasıdır (Örneğin alfabenin harfleriyle temsil edilen bir kromozom problemin amaç fonksiyonuna dâhil edilmeden önce bir ön işlemden geçerek sayısal yapıya kavuşturulabilir ya da ikilik sayı sistemiyle temsil edilen kromozomlar onluk sayı sistemine (decimal) dönüştürülerek fonksiyonda kullanılabilir). Bu durum aslında Karaboğa'nın ifade ettiği, çözüm temsili, popülasyonun kodlaması ve amaç fonksiyonu unsurları arası ilişkinin önemini göstermektedir.

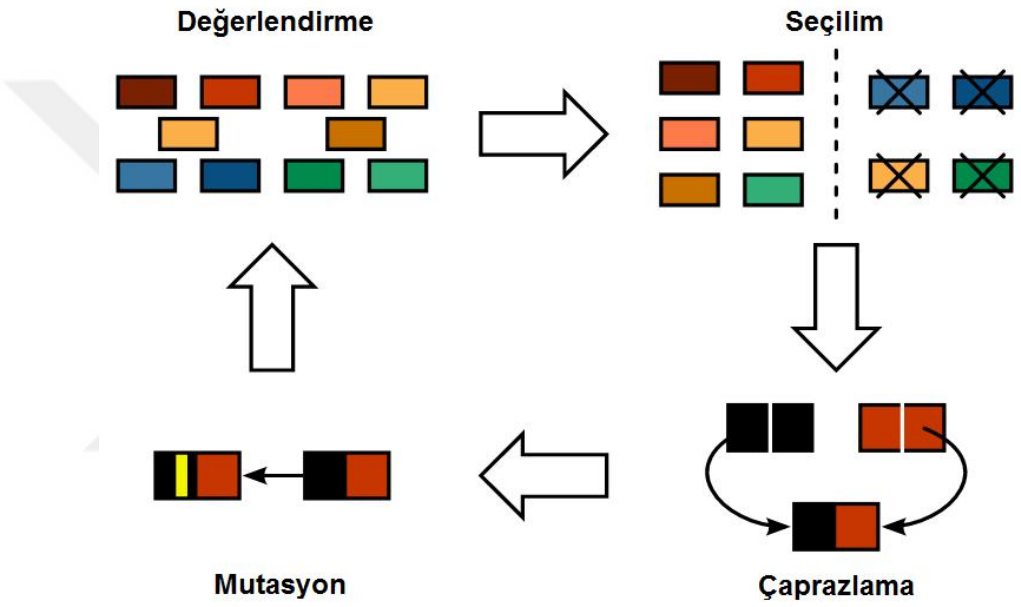
GA tekniği çözüm sürecinde, belirlenen popülasyonların genetik anlamda değiştirilip geliştirilmesi ve böylece çözüme ulaşma sürecinin pozitif yönde ilerlemesi için uygulanan mutasyon ve çaprazlama gibi genetik işlemlerin kodlanmış bireylere uygulanması şu şekilde olmaktadır (Holland, 2012; Kramer, 2017; Mitchell, 1998):

- **Çaprazlama (Crossover):** Bu işlemde, önceden belirlenmiş olan bireylerin bir bakıma üreyerek yeni bir birey oluşturması söz konusudur. Bunun için GA'nın ilgili parametreleri ve belirlenen çaprazlama şekline göre karşılıklı gen – kod alışverişi yaparak yeni bireyler elde edilmektedir. Bu bağlamda, örneğin simetrik anlamda birbirine ters düşen genlerin – kodların değiştirilmesi ya da belli aralıklarla değişim gerçekleştirilmesi gibi yaklaşımlar izlenebilmektedir.

- **Mutasyon (Mutation):** Algoritma dahilinde başarılı bireylerin bir şekilde daha başarılı çözümlere ulaşma potansiyelini artırmak için mutasyon işlemi de uygulanmaktadır. Mutasyon için seçilen bireylerde, yine GA'nın ilgili parametreleri eşliğinde kodların değiştirilmesi işlemi gerçekleştirilmektedir. Bu işleme dair en basit örnek 0 ve 1'lerle kodlanmış bir bireyde, bazı 0'ların 1, bazı 1'lerin de 0 yapılmasıdır.

GA'nın algoritmik sürecinde önemli olan diğer bir husus da çaprazlama ve mutasyon işlemlerinin hangi bireylere uygulanacağıdır. Bu durum, ilgili genetik işlemler uygulanmadan hemen önce seçilim sürecinin gerçekleştirilmesi ile çözülmeye çalışılmaktadır. Bu amaçla, geçerli popülasyondan, problemin amaç fonksiyon

yapısından başarılı sonuçlarla dönen bireyler arasından farklı yollarla seçilim yapılmaktadır. Liteatürü incelediğimizde, bayaslanmış rulet tekerleği (biased roulette wheel), orantılı yeniden üretim (proportionate reproduction), turnuva seçilimi (tournament selection), ve sıralama seçilimi (ranking selection) gibi alternatif yolların olduğunu görmek mümkündür (Baker, 1985; Goldberg ve Deb, 1991). Bunlar arasından çalışmalarda sık tercih edilen bayaslanmış rulet tekerleği ile birlikte bireylerin çözümlerinin, popülasyon ortalama çözümüne karşı kalitesine göre belli bir oranda temsili tekerlek üzerine konulmakta ve bu tekerleğin döndürülmesi suretiyle üremeye tabi tutulacak bireyler şans faktörüyle belirlenmektedir (Karaboğa, 2014).



Şekil 2. 10. GA'da kritik evrimsel süreçler (Cheng Yu, 2017).

GA'ın algoritmik çözüm adımlarını genel hatlarıyla şöyle ifade edebiliriz (Holland, 2012; Karaboğa, 2014; Kramer, 2017):

Adım 1 (Kurulum Fazı). *N* adet bireyden oluşan popülasyonu ilgili probleme ve tercih edilen kodlama düzenine göre oluştur. Başlangıç algoritma parametre değerlerini (Örneğin; çaprazlama oranı, mutasyon oranı) ve seçilim, çaprazlama, mutasyon gibi işlemlerde izlenecek yöntemleri belirle.

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her birey için; her amaç fonksiyon boyutu için;)

Adım 2.1. İlgili yapı kapsamında amaç fonksiyon değeri (fitness) hesapla.

Adım 2.2. Hesaplanmış olan amaç fonksiyon değeri (fitness) değerleri eşliğinde, üreme sürecine sokulacak bireyleri tespit için tercih edilen yöntemler bağlamında seçilim işlemini uygula.

Adım 2.3. Tespit edilen bireylere ilgili algoritma parametreleri ve tercihler kapsamında çaprazlama işlemini uygula.

Adım 2.4. Algoritma parametreleri ve tercihler kapsamında, ilgili bireylere mutasyon işlemini uygula.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

Tıpkı diğer YZ tabanlı optimizasyon tekniklerinde olduğu gibi, GA'nın da zaman içerisinde birçok farklı varyasyonu ilgili literatüre kazandırılmıştır [Bunlardan bazı güncel olanlar ve dikkat çekenler için: (Chiang, 2005; Dandy ve ark., 1996; Deb ve ark., 2002; Konar ve ark., 2017; Ling ve Leung, 2007; Mashwani ve ark., 2017; Ni ve ark., 2016; Palomo-Romero ve ark., 2017; Potts ve ark., 1994; Rajarathinam ve ark., 2017; Shi ve ark., 2003; Tan ve ark., 2016; Toğan ve Daloğlu, 2008)].

2.5.6. Guguk kuşu araması

Guguk Kuşu Araması [GKA – Cuckoo Search (CS)], Yang ve Deb (2009) tarafından geliştirilmiş, etkili bir SZ tekniğidir. Adında da anlaşılacağı üzere, guguk kuşlarının davranışlarından esinlenerek geliştirilmiş olan bu algoritmada, YAK'ta olduğu gibi çözüme yönelik muhtemel değişken değerlerinin rastgele tespit edilerek değerlendirilmesi yaklaşımını görebildiğimiz gibi, çözüm uzayında ortaya çıkan rastgele yürüyüşleri ve yine EH kapsamında inceleyebileceğimiz, nispeten kötü durumdaki unsurların elenmesi gibi süreçleri de GKA'da görmemiz mümkündür. GKA'nın iyi anlaşılması adına iki temel konuyu anlamak yeterlidir:

- **Guguk Kuşu Üremesi (Cuckoo Breeding):** Yumurtlayarak üreyen guguk kuşları, üremelerini doğada birtakım 'parazitsel' eylemlerine borçludurlar.

Buna göre, guguk kuşları genel olarak yumurtalarını ortak yapıdaki yuvalarda muhafaza ettiklerinden dolayı, yumurtalarının devamlılığını (dolayısıyla yavrularının yumurtalardan çıkmaları şansını) artırmak adına kendi yumurtalarını başkalarının yuvalarına bırakabilmekte ve yuva sahiplerine ait kendilerine yabancı yumurtaları parazitsel eylemle yaklaştıkları yuvalardan atabilmektedir. Diğer yandan, yabancı yuvada yumurtadan çıkan yavru guguk kuşları da, –yuvada kendilerine daha fazla yiyecek kaynağı düşebilmesi ve kendi türlerinin devamı için– yuva sahibinin yumurtalarını atabilmektedir. Bu noktada, yumurta görünüşleri diğerlerininkine benzeyecek şekilde evrimleşebilmiş bazı guguk kuşlarının yumurtalarının yuva sahipleri tarafından tespit edilip atılma olasılığı da düşebilmektedir. Payne (2005) tarafından ifade edilen bu doğal süreçler, GKA için de esin kaynağı olmuştur.

- **Levy Uçuşları (Levy Flights):** Metnin önceki bölümleri (*Bkz.* ikinci bölüm, alt-başlık 2.3.2.) kapsamında ifade edilen ve rastgele yürüyüş yaklaşımlarından birisi olan Levy Uçuşları, canlıların hareketleri esnasında ortaya çıkan konum değiştirmelerinde olasılıksal dağılıma dayanan bir yaklaşımdır. İlk olarak Mandelbrot (1982: 289) tarafından literatüre kazandırılmış olan Levy Uçuşları'nda, hareket halindeki unsurun atacağı adımın (konum değiştirmenin) boyutu değişkenlik göstermektedir. Bu değişkenlik, konum değişikliği süreci boyunca fraktal ve fraktal olmayan bir akış seyri ortaya koymaktadır (Cheng ve Savit, 1987). Araştırmalar genel olarak bu davranışın doğadaki birçok canlı tarafından ortaya konulduğunu göstermiştir (Brown ve ark., 2007; Pavlyukevich, 2007; Reynolds ve Frye, 2007). Yang ve Deb, GKA'yı tasarlama aşamalarında, doğadaki rastgele canlı hareketlerini daha hassas açıklayan bu yaklaşımı, varsayılan rastgele yürüyüş yerine tercih etmişlerdir [Levy Uçuşları'na ilişkin daha detaylı bilgi için: (Cheng ve Savit, 1987; Shlesinger ve ark., 1995; Viswanathan ve ark., 2002)].

GKA'da algoritmik süreçler kapsamında dikkate alınan bazı kabulleri de şu şekilde ifade edebiliriz (Yang ve Deb, 2009; 2010; 2014):

- Çözüm sürecinde işe koşulan her bir guguk kuşu – parçacık, aynı anda sadece bir tane yumurta üretebilir ve ürettiği yumurtayı da rastgele bir yuvaya bırakabilir (çözüm uzayında arama).

- Algoritma boyunca nispeten daha iyi sonuçlar üretilmesini sağlayan yumurtaları içeren yuvalar (potansiyel çözüm değişkenleri) gelecek nesillere aktarılır.

- Başka bir guguk kuşu tarafından bırakılan yumurtaların, yuva sahibi guguk kuşu tarafından tespiti olasılıksal olarak gerçekleşir (daha kötü durumdaki potansiyel çözümlerin tespiti). Tespit olması durumunda yumurta yuvadan atılır ya da başka bir deyişle yuva sahibi guguk kuşu ilgili yuvayı ortadan kaldırarak başka yuva arayışlarına geçer.

GKA tekniğinin algoritmik işlem adımlarını kısaca şöyle ifade edebiliriz (Yang ve Deb, 2009; 2010):

Adım 1 (Kurulum Fazı). *N* adet yuvayı (amaç vektör – değişken değerlerini) ilgili probleme göre, çözüm uzayında rastgele oluştur. GKA fraksiyon parametresi değerini belirle (Her yuvaya karşılık birer adet guguk kuşu olduğunu kabul et).

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her amaç fonksiyon boyutu bağlamında;)

Adım 2.1 (Levy Uçuşları Fazı). Her bir guguk kuşu için *Denklem 2.16.*'ya göre Levy Uçuş'u gerçekleştir ve ulaştığı noktaya göre amaç fonksiyon değeri (fitness) hesapla.

Adım 2.2 (Yuva Tespit ve Değerlendirme Fazı). İlgili guguk kuşu için yuvalardan birisini rastgele seç ve karşılıklı amaç fonksiyon değeri (fitness) değerlerini değerlendir. Eğer guguk kuşunun daha optimum bir amaç fonksiyon değeri (fitness) varsa, guguk kuşu ile yuva değerlerini yer değiştir.

Adım 2.3 (Seçilim Fazı). Fraksiyon değerine göre kötü durumdaki yuvaları kaldır ve rastgele yeni yuvalar oluştur. İyi durumdaki yuvaları gelecek nesillere aktarmak üzere koru.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

$$x_i^{z+1} = x_i^z + \alpha \oplus Levy(\lambda) \qquad (2.16.)$$

$$Levy \sim d = z^{-\lambda}, \qquad (1 < \lambda \leq 3)$$

Denklem 2.16.'da x değişken – konum, z zaman (iterasyon – döngü anlamında), i sıra numarası – indisi ve α adım boyutu anlamında gelmektedir. Yine Levy ifadesi kapsamında Levy Uçuşları tanımlanmakta ve $\oplus$ bu bağlamda değişken – konum değişikliği toplam işlemini tanımlamaktadır.



Şekil 2. 11. GKA tekniğinde guguk kuşları davranışları (Djalal, 2005).

GKA özellikle basit yapısı ve uygulamadaki verimliliği ile dikkatlerini üzerine çeken bir SZ tekniği olmuştur. Ancak diğer yandan, tekniğe yönelik farklı varyasyonlar da yine literatüre kazandırılmıştır. Varyasyonlara dair, en güncel ve dikkat çeken çalışmaları şöyle ifade edebiliriz: (Dhal ve ark., 2017; Dutta ve ark., 2016; Fister Jr ve ark., 2013; Li ve Wang, 2017; Liao ve ark., 2017; Marichelvam ve ark., 2014; Mohapatra ve ark., 2015; Ouaarab ve ark., 2014; Qu ve He, 2016; Shehab ve ark., 2017; Valian ve ark., 2011a; 2011b; Valian ve ark., 2013; Walia ve Kapoor, 2014; Yang ve Deb, 2013).

2.5.7. Klonal seçilim algoritması

Klonal Seçilim Algoritması [KSA – Clonal Selection Algorithm (CSA)], YZ altındaki Yapay Bağışıklık Sistemleri [YBS – Artificial Immune Systems (AIS)] yöntemi içerisinde incelenebilen tekniklerinden birisidir. YBS altındaki teknikleri zeki optimizasyon açısından değerlendirdiğimizde, daha önce ifade edilen sınıflandırmaların (*Bkz.* ikinci bölüm, alt-başlık 2.4.) tümüne yönelik özellik ve işlevler içerebildiklerini ifade edebiliriz. YBS kısaca Doğal Bağışıklık Sistemi'ne ait unsurlardan ve bu unsurlar bağlamında, bir organizmanın zararlı etkenlere karşı korunması adına işe koşulan çeşitli mekanizmalardan esinlenen YZ yöntemi olarak tanımlanabilmektedir (Aickelin ve ark., 2014; De Castro ve Timmis, 2003; Greensmith ve ark., 2010). Esin kaynağı olan unsurlar genellikle bağışıklık sürecinin parçaları olan B ve T hücreleri, antikorlar, antijenler ve dendritik hücrelerdir (Bachmayer, 2008; Greensmith ve ark., 2005).

YBS'nin daha iyi anlaşılması adına bağışıklık sürecindeki temel işleyişi de kısaca ifade etmekte fayda vardır. Genel olarak bağışıklık süreci, antijen adı veren zararlı hücrelerin, Doğal Bağışıklık Sistemi unsurları olan hücreler tarafından tespiti ve yok edilmesine dayanmakta ve söz konusu süreç sırasında antijenlere karşı dayanıklılığın artırılması için de çeşitli mekanizmalar işletilmektedir. Bu mekanizmalar, (bir tür evrimleşmeye karşılık gelen) hücrelerin olgunlaşması ve hafıza hücrelerinin devreye girmesi gibi ön plana çıkan işlevlerle tanımlanmaktadır (De Castro ve Von Zuben, 1999: 6-12). İşte bu süreçler ve ilgili unsurlardan esinlenerek geliştirilmiş olan YBS yöntemi dâhilindeki başlıca teknikler: Negatif Seçilim Algoritması (Negative Selection Algorithm), Klonal Seçilim Algoritması [KSA – Clonal Selection Algorithm (CSA)], Yapay Bağışıklık Ağı (Artificial Immune Network) olarak bilinmektedir (Bachmayer, 2008). Bu tez çalışması kapsamında ilgili tekniklerden KSA, aynı zamanda bir SZ tekniği olarak dikkate alınmıştır.

De Castro ve Von Zuben (2002) tarafından literatüre kazandırılmış olan KSA, antijen varlığında B hücrelerinin ürettiği antikorların antijenlere karşı olan eğilimi, bağışıklık sistemini güçlendirmek adına klonlanmaları, mutasyona uğramaları – olgunlaşmaları ve hafıza hücrelerinin istikrarı sağlaması gibi mekanizmalardan esinlenmekte ve anlaşılacağı üzere, optimizasyon tabanlı çözümlemeler de antikor ve

antijen rolünde kabul edilen parçacık – çözüm uzayı etkileşimleri neticesinde cereyan etmektedir.

KSA'nın algoritmik çözüm sürecini genel olarak şöyle açıklayabiliriz (De Castro ve Von Zuben, 2002; Karaboğa, 2014; Ulutaş ve Kültürel-Konak, 2011):

Adım 1 (Kurulum Fazı). *M* adet antijeni, çözüm uzayında rastgele belirle ve *N* adet antikora sun. KSA'ya dair temel algoritma parametrelerini (Örneğin; *k*: klonlama oranı, *m*: mutasyon oranı) belirle. Probleme yönelik gereken düzenlemeleri yap.

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her amaç fonksiyon boyutu bağlamında;)

Adım 2.1 (Yakınlık Fazı). İlgili antijenler ve antikorları dikkate almak suretiyle, yakınlık değerlerini (affinity), yani amaç fonksiyon değerlerini (fitness) hesapla. En iyi (minimizasyon ise en düşük, maksimizasyon ise en yüksek) yakınlık değerini elde etmiş, belirli sayıdaki antikoru seç.

Adım 2.2 (Klonlama Fazı). Seçilmiş antikorlara algoritma parametreleri dahilinde klonlama işlemi uygula (Daha fazla antijenik yakınlığı olan antikorlardan daha fazla klon üret).

Adım 2.3 (Mutasyon Fazı). Klonlanmış antikorlara, antijenik yakınlık değerleriyle ters orantılı büyüklükte oranlarla mutasyon işlemi uygula.

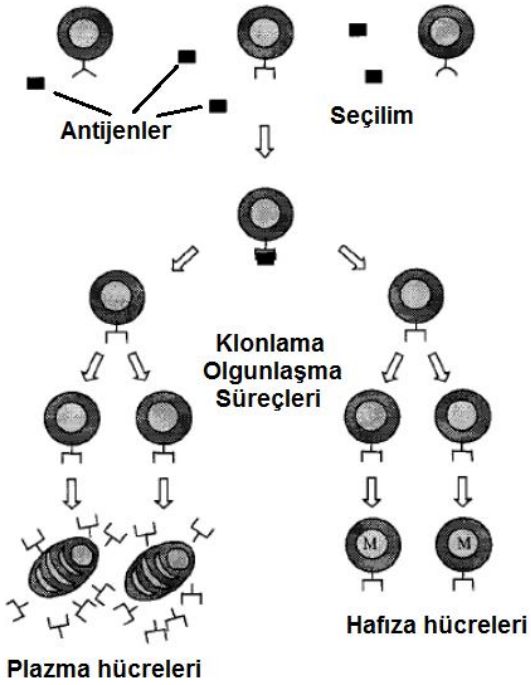
Adım 2.4 (Yakınlık Fazı). Klonlanmış ve mutasyona uğramış antikorlarla ilgili antijenleri, tekrar antijenik yakınlık değerleri hesaplama sürecine sok.

Adım 2.5 (Hafıza Fazı). Yüksek antijenik yakınlık elde eden antikorları hafıza hücresi kabul et. Bir antikorun antijenik yakınlık değeri daha önce kendisine göre oluşturulmuş hafıza hücresininkinden daha yüksekse, yüksek olanı önceki hafıza hücresinin yerine koy.

Adım 2.6 (Yenileme). Belirli sayıdaki düşük antijenik yakınlık değerine sahip antikorların yerine yeni antikorlar belirle.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

KSA'ya yönelik olarak ilgili literatüre kazandırılmış olan güncel ve dikkat çeken varyasyonları ise şu şekilde sıralayabiliriz: (Chitsaz ve ark., 2015; Dai ve ark., 2016; Das ve ark., 2017; Dong ve ark., 2007; Gong ve ark., 2010; Peng ve Lu, 2015; Rafiei ve ark., 2016; Xiao ve ark., 2016; Yin ve ark., 2016).



Şekil 2. 12. KSA'ya esin kaynağı seçilim süreci (Bachmayer, 2008).

2.5.8. Parçacık sürü optimizasyonu

Literatürü incelediğimizde, SZ yaklaşımını en basit ve pratik haliyle tanımlayan tekniklerden birisi de Parçacık Sürü Optimizasyonu'dur [PSO – Particle Swarm Optimization (PSO)]. PSO, genel olarak, bağımsız parçacıkların basit matematiksel yollarla yönlendirildiği ancak büyük resme bakıldığında ortak bir arayış hareketinin ortaya çıktığı bir zeki optimizasyon algoritmasıdır. Arka plandaki sürece bakıldığında, PSO'nun geliştirilmesinde, Lucas Film'de çalışmakta olan Reeves'in (1983) akt. (Karaboğa, 2014), patlama ve bulut efektleri için işe koşulmak üzere tasarlamış olduğu, çok sayıda parçacıktan oluşan hareket sisteminin rolü önemlidir. Bu sistem sonradan

Reynolds (1987) akt. (Karaboğa, 2014) tarafından, sürü içi kurallarla (birbirine çarpmama, birbirlerine yakın kalma… vb.) donanmış daha gelişmiş bir modele dönüşmüş ve nihayet, Eberhart ve Kennedy'nin (1995), bu modeli sosyal davranışları tanımlamaya yönelik genişletme çalışmaları sonucunda PSO ortaya çıkmıştır [(Karaboğa, 2014; Kennedy, 2011), akt. Karaboğa].

Çıkış noktası açısından bakıldığında, doğadaki kuş veya balık sürülerinin hareketlerinin (Örneğin; yiyecek arama eylemlerinin) benzetiminden yola çıkarak (Karaboğa, 2014; Kennedy, 2011; Yang, 2010b) oluşturulan PSO'da, *N* adet parçacıktan her birinin konumu (position), sahip olduğu hız (velocity) parametresi ve algoritma içi bazı parametrelerin etkisinde, iteratif çözüm süreci içerisinde değişmektedir. Parçacıkların çözüm uzayında hareketini sağlayan konum değişikliklerinde aynı zamanda her parçacığın o ana kadarki kişisel en iyi konumu (kendi çapında optimum değeri verdiği konum) ve iteratif süreçte o ana kadar elde edilmiş küresel en iyi konum (bütün parçacıklar arasında o ana kadar elde edilmiş optimum değeri veren konum) tarafından etkilenmektedir (Karaboğa, 2014; Kennedy, 2011; Yang, 2010b).

Tipik bir PSO algoritma yapısı şöyledir (Karaboğa, 2014; Kennedy, 2011):

Adım 1 (Kurulum Fazı). *N* adet parçacığı çözüm uzayına rastgele dağıt, her bir parçacık için başlangıç *Hız* (*v: velocity*) değerini ata. Her parçacığın başlangıç konumunu aynı zamanda kişisel en iyi konumu yap. Algoritma parametreleri değerlerini ata. Çözülecek probleme yönelik düzenlemeleri yap.

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her parçacık için; her amaç fonksiyon boyutu için;)

Adım 2.1. Parçacığın konumuna göre amaç fonksiyon değerini (fitness) hesapla.

Adım 2.2. Parçacık kişisel en iyi konumundan daha iyi bir konum bulmuşsa, kişisel en iyi konumunu güncelle. Eğer bu konum aynı zamanda küresel en iyi konumdan daha iyi ise, küresel en iyi konumu da parçacığın konumu yap.

Adım 2.3. Parçacığın hız parametresini *Denklem 2.17.* ile güncelle.

Adım 2.4. Parçacığın konumunu *Denklem 2.18.* ile güncelle.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

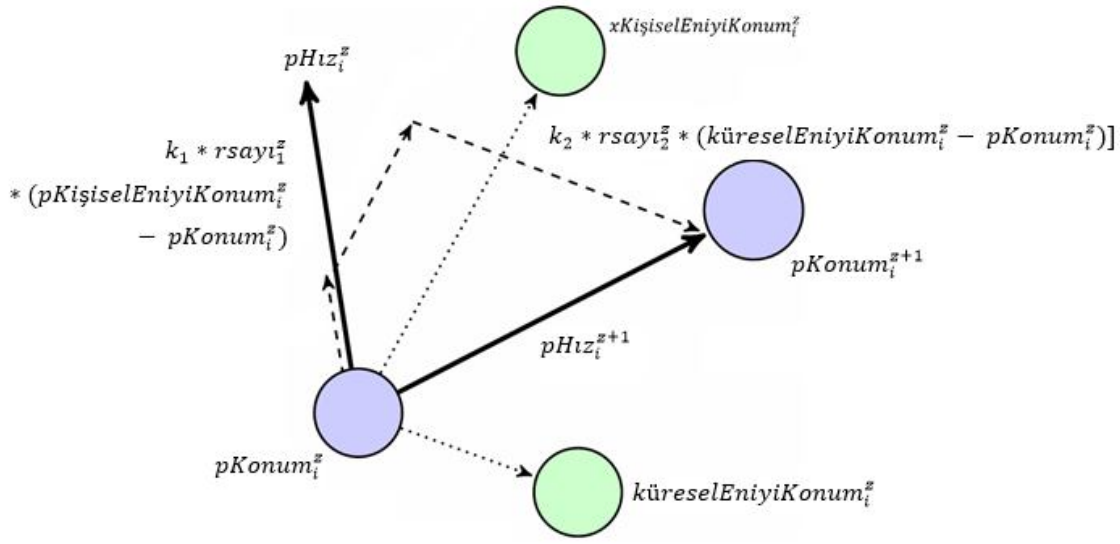
PSO'da her bir parçacığın hız parametresi; parçacığın geçerli hız değeri, kişisel en iyi konumu ile çözüm sürecindeki küresel en iyi konumun etkisi ve 0 ile 1 arasında üretilen rastgele sayıların etkisi bir araya getirilerek oluşturulan bir denklem ile güncellenmektedir:

$$pHız_i^{z+1} = pHız_i^z + [k_1 * rsayı_1^z * (pKişiselEniyiKonum_i^z - pKonum_i^z) + k_2 * rsayı_2^z * (küreselEniyiKonum_i^z - pKonum_i^z)] \quad (2.17.)$$

Denklem 2.17.'de, *i* parçacık numarasını temsil etmekte, *z* zaman için (iterasyon – döngü anlamında) kullanılmakta, k_1 ve k_2 ise sırasıyla kişisel ve küresel en iyi konumların etkisini ayarlayan algoritma içi parametrelere karşılık gelmektedir.

Elde edilen yeni hız değerleri parçacıkların geçerli konumuna eklenerek yeni konumlar hesaplanmakta ve parçacığın hareketi gerçekleştirilmiş olmaktadır:

$$pKonum_i^{z+1} = pKonum_i^z + pHız_i^{z+1} \quad (2.18.)$$



Şekil 2. 13. PSO'da parçacık hareketlerinin özeti (Bryson ve ark., 2016).

PSO, sonraları Shi ve Eberhart (1998) tarafından modifiye edilmiş, hatta literatürde birçok farklı PSO varyasyonu da geliştirilmiştir [Bunlardan bazı güncel olanlar ve dikkat çekenler için: (Chen ve Li, 2007; Delice ve ark., 2017; Hoang ve Pham, 2016; Rehman ve ark., 2017; Zhang ve Ni, 2017)]. Ayrıca, Poli (2008) ile Bonyadi ve Michalewicz (Bonyadi ve Michalewicz, 2017) tarafından kaleme alınmış çalışmalar, PSO'ya yönelik çalışmalara – çözümlere ışık tutan dikkate değer kaynaklar arasındadır.

2.5.9. Yapay arı kolonisi

Yapay Arı Kolonisi [YAK – Artificial Bee Colony (ABC)], temelde Karaboğa'nın arıların yiyecek arama davranışlarının modelleme fikri (Karaboğa, 2005) üzerinden şekillenen ve Karaboğa ile Baştürk (2007) tarafından literatüre kazandırılmış olan bir YZ tabanlı optimizasyon algoritmasıdır. YAK, özellikle rol dağılımlı çözüm ve –doğrudan Evrim Teorisi işlevleri kullanılmasa da– evrimsel süreçli çözüm yaklaşımları içerisinde (*Bkz.* ikinci bölüm, alt-başlık 2.4.2.) optimizasyon süreci sunan, etkili tekniklerden birisi olarak bilinmektedir. YAK'ın temelleri incelendiğinde, Tereshko (2000) tarafından önerilen ve yiyecek kaynakları, görevli – görevsiz arılar gibi bileşenler kapsamında, yiyecek kaynağına yönelme, kaynağı bırakma gibi işlevlere dayanan minimal yiyecek arama modelinin ve yine bu bağlamda, Tereshko ve Loengarov'un (2005) öne sürdükleri arılardaki kolektif karar verme yaklaşımının etkileri söz konusu olmaktadır (Karaboğa, 2014).

YAK'ın algoritmik yapısı incelendiğinde, temel olarak üç farklı role: işçi arı, gözcü arı ve kâşif arıya dönüşebilen parçacıkların çözüm uzayına dâhil edildiği bir çözüm süreci göze çarpmaktadır (Karaboğa, 2014; Karaboğa ve Baştürk, 2007). Bu rollerdeki parçacıklar genel olarak arıların yiyecek arama davranışlarının benzetimini gerçekleştirmektedir. Diğer yandan, çözüm sürecini matematiksel ve mantıksal anlamda basitleştirmek için algoritmada şu kabullere de uyulmaktadır (Karaboğa, 2014):

- Çözüm sürecinde arıların yiyecek aramasında ulaştıkları kaynaklar olası optimum değerlere karşılık gelmektedir.

- Algoritmada, ilgili kaynaklardaki nektar miktarları dikkate alınmaktadır. Buradaki nektar kavramı da kaynaklardaki değerlerle elde edilen çözüm değerleri kalitesine karşılık kullanılmaktadır.

- Her bir kaynaktaki nektar, sadece bir işçi arı tarafından alınabilmektedir. Bu durumda, toplam yiyecek kaynağı ile işçi arıları sayısı eşit olarak kabul edilmektedir.

- Genellikle işçi arı sayısı da gözcü arı sayısına eşit olarak kabul edilmektedir.

YAK'ın genel algoritma yapısını kısaca şöyle ifade edebiliriz (Karaboğa, 2010; 2014; Karaboğa ve Baştürk, 2007):

Adım 1 (Kurulum Fazı). *N* adet yiyecek kaynağını *Denklem 2.6.*'ya göre oluştur. İlgili arı parçacıklarını da bu bağlamda sayı olarak belirle. Başlangıç algoritma parametre değerlerini (Örneğin; kaynak geliştirilememe sayacı, tükenme limiti) ayarla. Probleme yönelik düzenlemeleri yap.

Adım 2. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her arı için; her amaç fonksiyon boyutu için;)

Adım 2.1 (İşçi Arı Fazı). İşçi arılarının *Denklem 2.19.*'u dikkate almak suretiyle yiyecek kaynağı bölgeleri seçmelerini ve gönderilmelerini sağla.

Adım 2.2. Yiyecek kaynağı konumlarına bağlı olarak amaç fonksiyon değeri (fitness) hesaplamasını yap ve bulunan amaç fonksiyon değerleri (fitness) eşliğinde yiyecek kaynaklarındaki nektarı (kaliteyi) belirle (*Denklem 2.20.* ve *Denklem 2.21.*). Nektar değerlerine göre yiyecek kaynaklarının geliştirilememe sayaçlarını bir artır ya da sıfırla.

Adım 2.3 (Gözcü Arı Fazı). İşçi arılardan gözcü arılara, yiyecek kaynaklardaki nektarlara ilişkin olarak gelen bilgilere göre, gözcü arıların seçecekleri yiyecek kaynakları (çözümler) için olasılıksal seçilim sürecini gerçekleştir. Bu amaçla her bir kaynağın seçilim olasılıkları belirlemek için *Denklem 2.22.*'yi kullan. Gözcü arılar

yiyecek kaynağı bölgeleri seçip, işçi arı olarak bu bölgelere gönderildikten sonra, Adım 2.2.'de ifade edilen işlem süreçlerini gerçekleştir.

Adım 2.4 (Kaşif Arı Fazı). Yiyecek kaynaklarındaki geliştirilememe sayaçlarını kontrol et ve ilgili sayaç değeri tükenme limitini aşan kaynaktaki işçi arıyı kaşif arıya dönüştür (Genellikle her döngüde bir kaşif arı oluşturulmaktadır). Kaşif arının yiyecek kaynağı aramasını sağla.

Adım 3. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

$$x_{ij} = x_j^{min.} + rastgele(0,1) * (x_j^{maks.} - x_j^{min.}) \quad (2.19.)$$

Denklem 2.19. kapsamında *i* yiyecek kaynağı – değişken sırasına, *j* ise boyut sırasına karşılık gelmekte, *rastgele* işlevi 0 ile 1 aralığında rastgele bir sayıyı, *maks.* ve *min.* gibi ifadeler de ilgili değişkenlerin üst ve alt-limitlerini ifade etmektedir.

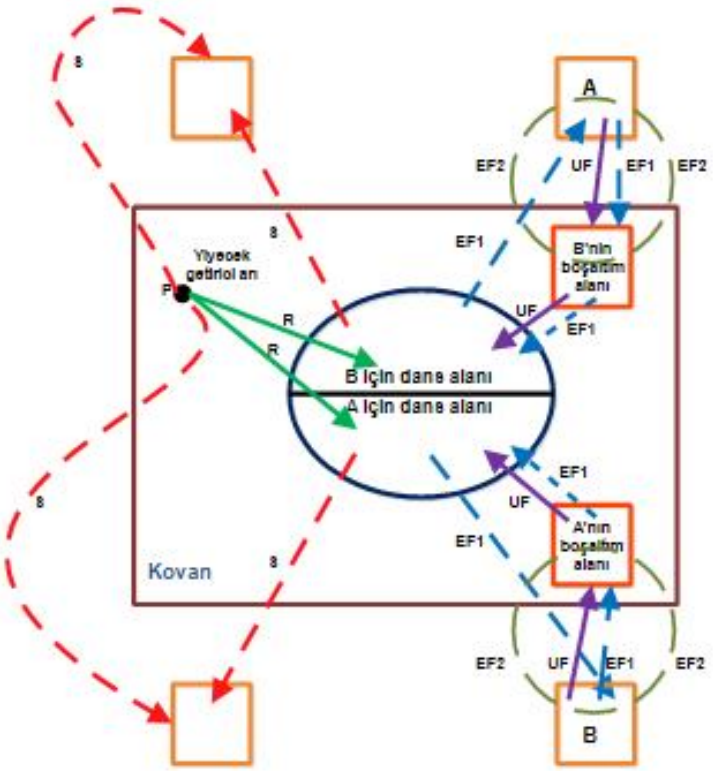
$$d_{ij} = x_{ij} + \phi_{ij} * (x_{ij} - x_{kj}) \quad (2.20.)$$

$$fonkdeğeri_i = \begin{cases} 1/(1 + fonksdeğeri_i), & Eğer\ fonkdeğeri_i \geq 0 \\ 1 + mutlak(fonkdeğeri_i), & Eğer\ fonkdeğeri_i < 0 \end{cases} \quad (2.21.)$$

Denklem 2.20. ve *Denklem 2.21.* bağlamında, *d* uygunluk değerini temsil etmekte, ϕ [-1, 1] aralığında rastgele bir sayıyı, *fonkdeğeri* hesaplanan amaç fonksiyon değeri (fitness), *mutlak* işlevi ise sayının mutlak değerinin alınmasını ifade etmektedir.

$$o_i = \frac{fonkdeğeri_i}{\sum_{j=1}^{İS} fonkdeğeri_i} \quad (2.22.)$$

Denklem 2.22.'de *o* olasılık değerine karşılık gelmekte, ayrıca *fonkdeğeri* hesaplanan amaç fonksiyon değerini (fitness), *j* işçi arı sıra numarasını ve *İS* toplam işçi arı sayısını ifade etmektedir.



Şekil 2. 14. YAK'ta arı hareketleri genel yapısı (Karaboğa, 2014).

YAK'ın literatürdeki diğer tekniklerle karşılaştırılmalarına dayanan kapsamlı çalışmalar (Karaboğa ve Akay, 2009; Karaboğa ve Baştürk, 2008; Karaboğa ve ark., 2014), algoritmanın başarımını ortaya koymuş, bununla birlikte YAK'ın işe koşulduğu birçok çalışmanın da bir bakıma önünü açmıştır. Literatürü incelediğimizde YAK'ın farklı optimizasyon türlerine olduğu kadar daha etkin ve verimli çözüm süreçlerine de izin veren çeşitli varyasyonlarını da görmek mümkündür [Bunlardan bazı güncel olanlar ve dikkat çekenler için: (Akbari ve ark., 2012; Bacanin ve Tuba, 2012; Gao ve Liu, 2011; Karaboğa ve Akay, 2011; Karaboğa ve Görkemli, 2011; Karaboğa ve Öztürk, 2011; Lei ve ark., 2010; Liu ve ark., 2013; Quan ve Shi, 2008; Wu ve Fan, 2011; Yıldız, 2013; Zhang ve ark., 2011)].

Bu bölüm altında açıklanan SZ teknikleri ve bu tekniklerden varyasyonlar, çalışmada geliştirilen algoritmaların değerlendirilmesi amacıyla karşılaştırma süreçlerine de katılmıştır. Konuya dair daha fazla detay üçüncü bölümde ele alınmıştır.

3. MATERYAL VE YÖNTEM

Bu bölüm altında, gerçekleştirilen çalışmayla ilişkili materyal ve metotlar açıklanmıştır. Tahmin edileceği üzere, çalışmanın temel amacına uygun olarak ortaya konulan algoritmaların geliştirilme sürecine yönelik izlenilen ilkeler, yaklaşımlar, geliştirmelerin yapıldığı ortam(lar), tasarım – geliştirme süreci, söz konusu algoritmaların başarımını değerlendirmek için izlenen yaklaşımlar, yöntemler ve karşılaştırmalı değerlendirme yapabilmek için dikkate alınan alternatif algoritmalar – teknikler önemlidir. Bu nedenle, algoritmalara ilişkin bilgilere, yapılmış olan uygulamalara ve elde edilen bulgulara değinmeden önce söz konusu konulara yönelik kısa bilgiler sunulması tercih edilmiştir.

3.1. Tasarım – Geliştirme İlke Yaklaşım ve Ortamları

Çalışma ile birlikte ortaya konulan optimizasyon algoritmalarının tasarımı ve geliştirilmesi süreçlerinde, kabul edilmiş çeşitli ilkeler, odaklanılmış yaklaşımlar ve neticede algoritmaların uygulamaya konulduğu ortamlar söz konusudur. Bu bağlamdaki açıklamaları kısaca şu şekilde sunabiliriz:

3.1.1. Tasarım – geliştirme ilkeleri

İlgili literatüre alternatif algoritmaların kazandırılması süreci kapsamında, özellikle tasarım – geliştirme süreci öncesinde kabul edilmiş olan bazı ilkeler söz konusudur. Algoritmaların ortaya konulması noktasında bu ilkelere uyulmasına dikkat edilmiş ve böylece hem literatürden elde edilmiş olan konulara bağlı kalınması, hem de alternatif zeki optimizasyon tekniklerinin elde edilmesi sağlanmıştır. Söz konusu tasarım – geliştirme ilkeleri kısaca şöyle listelenebilir:

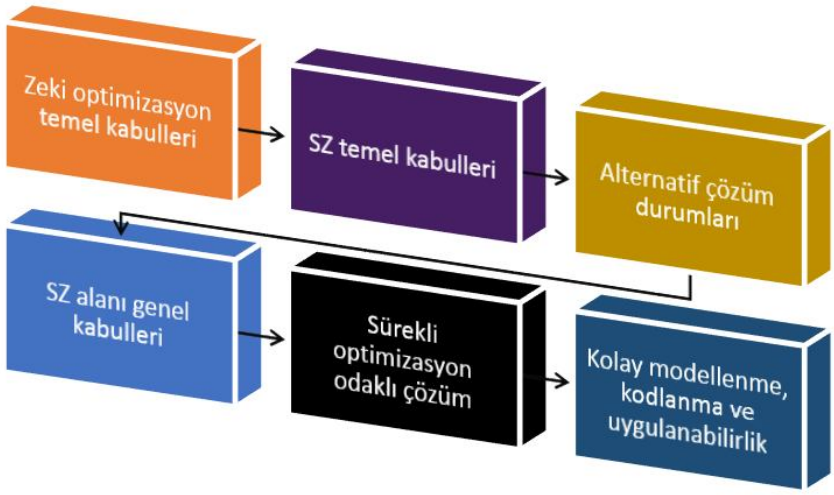
- Zeki optimizasyon ile ilgili temel yaklaşımlara sadık kalmak (*Bkz.* ikinci bölüm, alt-başlık 2.2.2.).
- Var olan teknikler ile ilgili temel kabullere mümkün olduğunca uymak (*Bkz.* ikinci bölüm, alt-başlık 2.4.1.).

- Neticede, sürekli optimizasyon için kullanılabilen algoritmalar geliştirmiş olmak.

- Teknikler ile ilgili alternatif durumlardan, ‘rol dağılımlı çözüm’ ve ‘evrimsel süreçli çözüm’ yöntemlerini içeren algoritmalar ortaya koymak (*Bkz.* ikinci bölüm, alt-başlık 2.4.2. – algoritmaları karmaşık hale getirme riskine karşı, ‘gruplar arası mücadeleli çözüm’ yöntemi pek tercih edilmemiştir. Sürekli optimizasyona odaklanıldığından dolayı da, ‘graf tabanlı çözüm’ kullanılmamıştır).

- SZ yaklaşımı genel kabullerine uygun algoritmalar ortaya koymak.

- Mantıksal ve matematiksel anlamda modellenmesi, kodlanması ve uygulanması kolay ve yeterince etkin çözümler sunan sürekli optimizasyon algoritmalarının ortaya konulması (*Bkz.* giriş bölümü altında ifade edilen tez çalışmasının alt-amaçları).



Şekil 3. 1. Optimizasyon algoritmalarının tasarım – geliştirme sürecinde ilkeler.

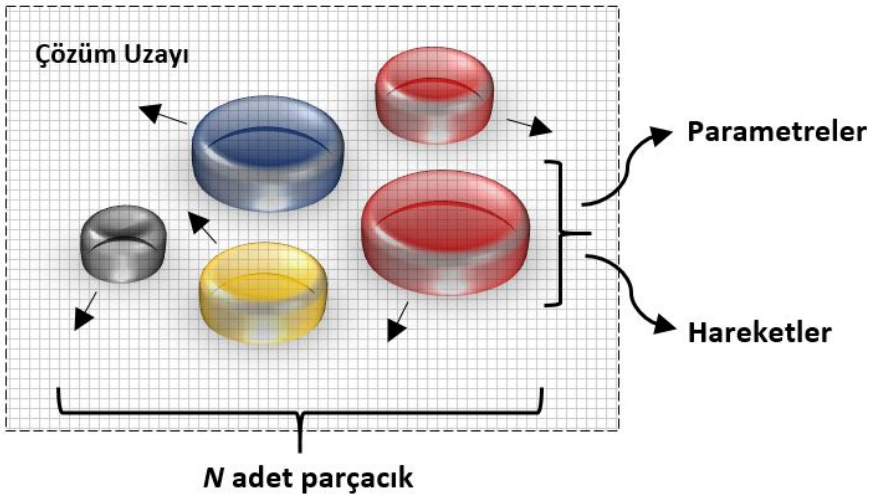
3.1.2. Nesne yönelimli programlama

Her ne kadar çalışma ile birlikte ortaya konulacak algoritmaların ilerleyen süreçte farklı sistemler için, farklı bilgisayar programlama dilleriyle yazılması söz konusu olacaksa da; tasarım – geliştirme süreçlerinde verimli ve hızlı programlama yaklaşımları içerisinde yazılması oldukça önemlidir. Böylelikle, hem çalışma sürecinin etkinliğini artacak, hem de tasarlanan algoritmaların uygulamaya çabuk dökülerek, daha

tutarlı ve sağlıklı uygulanmalarına imkân verilecektir. Bu nedenle çalışma bağlamında, günümüz popüler bilgisayar programlama yaklaşımlarından birisi olan Nesne Yönelimli Programlama [NYP – Object Oriented Programming – OOP] tercih edilmiştir.

Çalışmada NYP yaklaşımının neden tercih edildiğini anlamak adına SZ tekniklerinden yola çıkmak gerekmektedir. Öncelikli olarak, tipik bir teknikle bağlantılı olarak şu unsurlardan bahsedebiliriz:

- Bir teknikte *N* adet parçacık çözüm sürecinde işe koşulmaktadır.
- Her bir parçacığın, çözüm süresini etkileyen ve bu çözüm süreci sırasında değişip – güncellenen çeşitli parametreleri vardır.
- İlgili parçacıklar, çözüme ulaşmaya çalışırken, çözüm uzayı ortamında çeşitli hareketlerde bulunmaktadır.



Şekil 3. 2. Tipik bir SZ tekniğinde NYP ile ilişkilendirilebilen unsurlar.

Bahsedilen bu üç durum, bilgisayar ortamında en iyi NYP ile yazılabilecektir. Çünkü ifade edilen durumlar, NYP yaklaşımında söz konusu olan, nesnelerin 'özellikleri' ve 'olayları' ile neredeyse birebir örtüşebilmektedir. Konuyu daha iyi anlamak adına NYP yaklaşımından da kısaca bahsetmekte fayda vardır:

NYP, 1960'lı yılların sonlarına doğru ortaya çıkan ve o zamanlar, sadece işlev tabanlı programlama (procedural programming: prosedür – fonksiyon – işlev adı verilen yapılar altında işlem adımları içeren) yaklaşımının sebep olduğu maliyet – çabaya karşılık karmaşıklık – boyut dengesizliğini ve zamanla ortaya çıkan birtakım zorlukları

giderme yolunda geliştirilmiş olan bir bilgisayar programlama yaklaşımıdır (Kalemis, 2013; Nino, 2007; Rentsch, 1982; Vikipedi, 2016a; 2016b; Wikipedia, 2017e; 2017f). NYP'nin temelindeki düşünce, nesne (object) adı verilen ve birbirleriyle etkileşim içerisine girebilen, aralarındaki etkileşimlerle problem çözümüne yön verebilen elemanlar aracılığıyla bilgisayar programları tasarlanabilmesidir. Bu noktada, söz konusu nesneler, problem çözümünde kullanılmak üzere çeşitli özelliklerle (properties) ve işlevlerle – olaylarla (events – methods) donatılmakta ve hatta sınıf (class) adı verilen bir yapı altında hiyerarşik bir şekilde, aynı program üzerinde birden fazla geliştiricinin çalışabilmesine imkân veren bir programlama dili sisteminin oluşturulması sağlanmaktadır (Kalemis, 2013; Karaçay, 2016; Vikipedi, 2016a). Farklı nesneler farklı sınıflara ait olabilmekte ve bu duruma göre farklı problemlerde yalın veya farklı sınıflarla (nesnelerle) işbirliği içerisinde işe koşulabilmektedir. Sınıf yapısı ile ilgili olmak üzere, NYP'nin temelinde yer alan kalıtım (inheritance) yöntemi sayesinde, bir nesneye ait yeni alt-nesneler oluşturulabilmektedir. Böylelikle, ilgili nesnenin özellik ve işlevlerinin yeni alt-nesnelere geçirilmesinin yanında, yeni özellik ve işlevlerle birlikte yepyeni bir nesne (Örneğin; taşıtlar adlı bir sınıf üzerinden, otomobiller ve kamyonlar olarak iki yeni nesne sınıfı üretilmesi gibi) tasarlanabilmektedir. Nesne mantığı sayesinde, aynı isimde olan ancak farklı nesnelere ait olan işlevlerle aynı parametrelere alternatif çözüm süreçleri işletilebilmektedir [Çok biçimlilik (Polymorphism)]. Genel olarak NYP'nin soyutlama (abstraction) adı verilen felsefesi, gerçek dünya tabanlı unsurların özelliklere ve işlevlere sahip programlama nesneleri şeklinde tasarlanması mantığını açıklamaktadır (Kalemis, 2013; Karaçay, 2016; Stefik ve Bobrow, 1985; Stroustrup, 1988).

Açıklamalardan da anlaşılacağı üzere, bir SZ tekniğini tasarlayıp, pratik bir şekilde uygulayabilmenin en etkin ve kolay yolu günümüz programlama yaklaşımları kapsamında NYP'den geçmektedir. Bu nedenle, çalışmada geliştirilen tekniklerin – algoritmaların tasarımı aşamasında NYP yaklaşımına dayalı bir programlama dilinin kullanılması yoluna gidilmiştir. Bilgisayar programlama literatüründe, yaygınlıkları bilinen Java, C++, Python, Perl ve Ruby gibi programlama dilleri olsa da, Microsoft'un gücünü arkasına alan, gerek bu tez çalışması kaleme alınırken tıpkı diğer BYP dilleri gibi yaygın olan, gerekse Microsoft Visual Studio adlı pratik görsel programlama platformunu kullanan C# programlama dili bu aşamada tercih edilmiştir. Bu noktada,

çalışmalara konu olan tasarım, geliştirme ve değerlendirme ortamlarından da bahsetmekte fayda bulunmaktadır.

3.1.3. Geliştirme ve değerlendirme ortamları

Bir sonraki bölüm altında detaylarına değinilmiş olan tasarım, geliştirme ve değerlendirme süreçlerinde üç farklı yazılım ortamından faydalanılmıştır. Bunlar sırasıyla:

- Microsoft Visual Studio,
- MathWorks MATLAB
- IBM SPSS

şeklindedir. İlgili yazılım ortamlarının temel özelliklerini kısaca şöyle açıklayabiliriz:

- **Microsoft Visual Studio:** Visual Studio programı, Microsoft yazılım firması tarafından 1995 yılından bu yana piyasaya sürülen ve farklı programlama dilleri kapsamında Windows Form (Pencere mantığına dayalı – masaüstü) uygulamaları, Web uygulamaları, Web servisleri, mobil platform uygulamaları ve konsol (komut satırı mantığına dayalı) uygulamaları tasarlayıp geliştirmeye yarayan bir kodlama – tasarım yazılımıdır (Johnson ve ark., 2003; Ritchie, 2016; Uzunköprü, 2015; Wikipedia, 2017d). Genel hatlarıyla .Net olarak isimlendirilen, Microsoft'un platform bağımsız programlama çerçevesine (framework) dayalı uygulamaların temel geliştirme platformu da Visual Studio'dur. Günümüz görsel programlama süreçlerinin aktif aktörlerinden birisi olan Visual Studio platformunun desteklediği temel programlama dilleri C#, C++ / C, ASP / ASP.Net ve VB (Visual Basic) olarak sayılabilmektedir (Johnson ve ark., 2003; Ritchie, 2016; Uzunköprü, 2015; Wikipedia, 2017d). Visual Studio platformu, özellikle NYP yaklaşımına (*Bkz.* üçüncü bölüm, alt-başlık 3.1.2.) yönelik pratik ve etkin kodlama – tasarım çözümleri nedeniyle, bu çalışma kapsamında geliştirilen algoritmaların tasarlanması aşamalarında oldukça faydalı olmuştur (*Bkz.* dördüncü bölüm, alt-başlık 4.1.). Çalışma sürecinde söz konusu yazılımın 2012 sürümü (11.0) işe koşulmuştur.

- **MathWorks MATLAB:** MATLAB yazılımı, 1984 yılından bu yana, MathWorks firması tarafından piyasaya sürülen, sayısal hesaplama (numerical computing) yazılımıdır. İsmi MATrix LABoratory (MATris LABoratuvarı) ifadelerinden oluşturulan yazılım, bilimsel ve mühendislik tabanlı çalışmalarda yaygın bir şekilde kullanılmaktadır. MATLAB genel hatlarıyla içerisinde kendine has programlama dilini de barındıran ve komut tabanlı yönlendirmelerle eldeki veriler üzerinde ileri düzey hesaplamalar yapılmasına, görsel arayüz destekli özel sistemler geliştirilmesine, algoritmalar tasarlanıp kodlanmasına, yapılan çalışmalara yönelik görsel bulguların (Örneğin; grafikler) elde edilmesine ve farklı dillerle (C#, C / C++, Java, Python…vb.) kodlanmış uygulamalarla entegre özel çözüm yaklaşımlarının ortaya konulmasına imkan veren, oldukça kapsamlı bir yazılım olarak değerlendirilmektedir (Çiftçi, 2011; Gilat, 2016; Kubat, 2016; Wikipedia, 2017c). Bu çalışmada ortaya konulan algoritmaların genel tasarım ve geliştirme süreçlerini takiben test fonksiyonları, problemleri ve bütün gerçek yaşam tabanlı problem ve uygulamaların gerçekleştirilmesinde (*Bkz.* beşinci bölüm) MATLAB yazılımı ortamından faydalanılmıştır. Böylelikle tasarım ve geliştirme süreçlerini NYP destekli, pratik geliştirme ortamı ile destekleyen Visual Studio ortamına ek olarak, hızlı hesaplama ve bulguların kolay anlamlandırılması yönünde avantajları beraberinde getiren MATLAB ortamı da, yerinde bir tercih olarak çalışmalarda rol üstlenmiştir. Bu bağlamda, çalışma sürecinde MATLAB yazılımının 2010a sürümü (7.10) kullanılmıştır.

- **IBM SPSS:** SPSS yazılımı, 1968 yılından bu yana istatistiksel analiz ve yorumlamalar için oldukça yaygın bir şekilde kullanılan, güçlü bir platformdur [İsmini, Statistical Package for Social Sciences (Sosyal Bilimler için İstatistiksel Paket) ifadesinin kısaltmasından almaktadır]. Önceleri kendi adını taşıyan yazılım firması tarafından piyasaya sürülürken, 2009 yılında IBM tarafından satın alınan SPSS, araştırmacıların verilerini bir arada tutabilmesi, manipüle edebilmesi, analiz edebilmesi ve farklı şekillerde sunabilmesi adına bünyesinde çeşitli alt-programlar sunmaktadır (Landau ve Everitt, 2004). İsminde ‘Sosyal Bilimler’ kelime bütünü geçse de, verilerin uygunluğuna göre pekâlâ bütün bilim alanlarında analiz ve yorumlama

süreçleri için kullanılabilmektedir. Benzer olarak, bu tez çalışmasında değerlendirme süreçleriyle elde edilen verilerin analiz edilerek yorumlanması için de SPSS yazılımının 19.0 sürümü kullanılmıştır.

3.2. Tasarım – Geliştirme Süreci

Önceki başlıklar altında bahsedilen ilkeler, yaklaşımlar ve ortamlar dahilinde gerçekleştirilen çalışmalar, neticede iki sürekli optimizasyon algoritmasının literatüre kazandırılmasını sağlamıştır. Yaklaşık olarak 2013'ün üçüncü çeyreğinde başlamış olan sürece algoritmalar ile gerçekleştirilen uygulamalar ve çalışmaların literatürde yayınlanması süreçleri de eklendiğinde, –şu an için– 2017 Temmuz dönemine kadar sürmüş olan bir süreçten bahsetmemiz mümkündür. Bu bağlamda, ilgili tasarım – geliştirme sürecini biraz daha geniş bir perspektiften incelemek de çalışmanın anlaşılması adına yararlı olacaktır.

YZ tabanlı optimizasyon algoritmalarının tasarım – geliştirme sürecini genel olarak şu başlıklar altında tanımlamamız mümkündür:

- Algoritmaların geliştirilmesine yönelik yeterliklerin değerlendirilmesi ve hazırlık dönemi,
- Algoritmaların temelinde yer alacak doğa temelli düşüncelerin netleştirilmesi ve bu düşüncelerin uygunluğuna yönelik olarak literatür incelemesinin yapılması,
- Var olan matematiksel ve mantıksal kabullerin, düşünülen algoritmaların geliştirilmesi aşamasında kullanılabilirliklerinin değerlendirilmesi,
- Algoritmaların optimizasyon test problemleriyle değerlendirilmesi eşliğinde genel hatlarının tasarlanması,
- Genel hatlarıyla tasarlanmış olan algoritmaların daha geniş çapta problemlerde değerlendirilmesi ve gerekli noktalarının geliştirilmesi (Ayrıca bu noktada algoritmaların temellerine ilişkin bazı temel literatür yayınlarının gerçekleştirilmeye başlanması).

- Söz konusu algoritmalarla yapılabilecek çok-disiplinli uygulamaların genişletilmesi ve literatürde yüksek değeri olan, farklı nitelikte yayınların (Örneğin; Science Citation Index – Web of Science tabanlı) gerçekleştirilmesi.



Şekil 3. 3. Tasarım – Geliştirme sürecinin kısa bir özeti.

Kısaca açıklanmış olan süreç ile birlikte geliştirilen algoritmaların (GOA ve BiGOA: *Bkz.* dördüncü bölüm) başarımını anlamak ve literatürdeki yerini tespit etmek adına farklı değerlendirme yaklaşımları içerisinde bulunulmuştur. Buna göre ilk olarak, algoritmaların literatürde kendilerine ifade bulan çeşitli test fonksiyonları ile değerlendirilmesi yoluna gidilmiştir.

3.3. Test Fonksiyonları

Herhangi bir YZ tabanlı optimizasyon algoritmasının geliştirilmesi aşamasında dikkate alınan en önemli kıstas hiç kuşkusuz ki algoritmanın farklı optimizasyon problemlerindeki başarımıdır. Algoritmanın muhtemel farklı optimizasyon problemlerinde ne derece başarılı olacağı yüzde yüz kestirilemez olsa da, literatürde yer alan farklı yaklaşımlarla yeter düzeyde fikir sahibi olabiliriz. Bu amaçla literatürde yapılan çalışmalar neticesinde optimizasyon yönelimli çözümlerin etkinliğine dair fikir veren farklı test (benchmark) fonksiyonları ortaya konulmuştur.

Test fonksiyonları farklı algoritmaların – tekniklerin karşılaştırılması ve gerçek dünya tabanlı problemlere yönelik potansiyellerinin önceden değerlendirilebilmesi adına kullanılan çeşitli unsurlar olarak kabul edilmektedir (Dieterich ve Hartke, 2012: 1552; Jamil ve Yang, 2013; Wikipedia, 2017i). Genel olarak bu fonksiyonlar boyutlarına göre (toplam değişken sayısına göre) tek-boyutlu, iki-boyutlu veya çok-boyutlu olarak sınıflandırılırken, yine sahip oldukları kısıtlara göre kısıtlı kısıtsız ve amaç fonksiyonun sayısına göre tek-amaçlı veya çok-amaçlı test fonksiyonları şeklinde çeşitli sınıflara ayrılabilmektedir. Literatürde test – değerlendirme çalışmalarının konu olan birçok farklı test fonksiyonu bulunmaktadır (Ali ve ark., 2005; Molga ve Smutnicki, 2005; Surjanovic ve Bingham, 2015). Ancak bunlardan bazıları nitelikleri ve çözüm yönünden zorlukları nedeniyle daha çok kullanılagelir hale gelmiştir.

Geliştirilen algoritmalar test fonksiyonlarına uygulanırken, şu kıstaslara göre düzenlemeler yapılmaktadır:

- Amaç fonksiyon(lar),
- Varsa kısıtlar,
- Boyut limitleri (Değişken sayısı): İki boyutluluk ya da çok-boyutluluk,
- Çözüm uzayı arama alanı sınırları (minimum ve maksimum sınır değerleri),
- Küresel optimumu (Örneğin; minimumu) sağlayan değerler.

Bu tez çalışması kapsamında gerçekleştirilen algoritmaların değerlendirilmesi için kısıtsız tek-amaçlı, kısıtlı tek-amaçlı, kısıtsız çok-amaçlı, ve kısıtlı çok-amaçlı fonkiyon(lar)a sahip toplam 30 adet test fonksiyonu kullanılmıştır. Test fonksiyonlarının seçimine yönelik kriterler ise kısaca şu şekildedir:

- Toplam 15 kısıtsız test fonksiyonu, Surjanovic ve Bingham (2015) tarafından belirtilmiş olan kategoriler: ‘Çok Sayıda Yerele Sahip (Many Local)’, ‘Çanak Şekilli (Bowl-Shaped)’, ‘Yayvan Şekilli (Plate-Shaped)’, ‘Vadi Şekilli (Valley-Shaped)’, ‘Dik Tepeli / Damlalı (Steep Ridges / Drops)’ ve ‘Diğer (Others) / Diğer Kategorilere Girmeyen’ kapsamında dikkate alınmıştır. İlgili fonksiyonlar belirlenirken özellikle kısıtsız olanlarda, literatürde de sıklıkla ifade edilen yerel optimum(lar)a takılma probleminin (Chiang ve ark., 2001;

Cutello ve ark., 2014; Deng ve ark., 2012; Diwei ve ark., 2007; Ho, 2013; Hong, 2009; Hong ve ark., 2010; Korenaga ve ark., 2007; Li ve Li, 2013; Lin ve Li, 2009) yeterince çözülüp çözülmediğini anlamak için 'Çok Sayıda Yerele Sahip (Many Local)' kategorisinde 5 fonksiyon çalışmaya dâhil edilmiştir. Yine bu fonksiyonlardan 5 tanesi 'Çok Sayıda Yerele Sahip (Many Local)' olmayan ancak iki-boyutlu, geriye kalan 5 tanesi de, literatürde yaygın bir şekilde bilinen, çok-boyutlu ve yine 2 tanesi çok sayıda yerele sahip olan test fonksiyonları olmuştur.

- Çok-boyutlu test durumları için Surjanovic ve Bingham (2015) tarafından belirtilmiş fonksiyonlar arasından seçilenlere ek olarak, 2015 IEEE Evrimsel Hesaplama Kongresi'ndeki (IEEE Congress on Evolutionary Computation) optimizasyon yarışmasında sunulan (Liang ve ark., 2014) 5 adet test fonksiyonu, çok-boyutlu test durumları kapsamında çalışmada yer almıştır.

- Kısıtlı problem durumlarını yorumlamak adına 3 fonksiyon daha, kısıtlı tek-amaçlı test fonksiyonları olarak çalışmaya dâhil edilmiştir.

- Buraya kadar ifade edilen tek-amaçlı test fonksiyonlarının yanında, çok-amaçlı optimizasyon çözümlemelerini değerlendirmek için 4 adet çok-amaçlı kısıtsız, 3 adet de çok-amaçlı kısıtlı olmak üzere toplam 7 çok-amaçlı test fonksiyonu da çalışmaya dâhil edilmiştir.

İlgili fonksiyonlara dair detaylar ilerleyen alt-başlıklar altında (alfabetik sıra düzeniyle birlikte aşağıdaki kıstaslar dikkate alınarak) kısaca ifade edilmiştir:

- Surjanovic ve Bingham (2015) ve genel literatürden seçilen tek-amaçlı kısıtsız ve kısıtlı fonksiyonlar,

- Surjanovic ve Bingham (2015) ve genel literatürden seçilen tek-amaçlı çok-boyutlu fonksiyonlar,

- 2015 IEEE Evrimsel Hesaplama Kongresi'ndeki (IEEE Congress on Evolutionary Computation) optimizasyon yarışmasından (Liang ve ark., 2014) seçilen tek-amaçlı çok-boyutlu fonksiyonlar,

- Genel literatürden seçilen çok-amaçlı kısıtsız ve kısıtlı fonksiyonlar.

3.3.1. Beale fonksiyonu

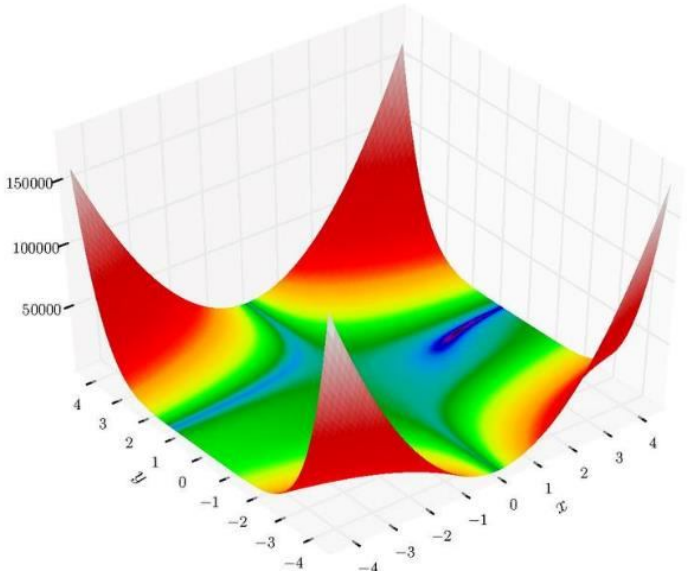
Beale Fonksiyonu, keskin tepe noktalarına sahip, çok modlu, iki-boyutlu – değişkenli bir test fonksiyonudur (Surjanovic ve Bingham, 2015).

İlgili fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.1.'de verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 1. Beale Fonksiyonu küresel min. ve arama alanı.

Beale Fonksiyonu	$f(x,y) = (1{,}5\ - x + x * y)^2 + (2{,}25\ - x + x * y^2)^2 + (2{,}625\ - x + x * y^3)^2$	(3.1.)
Küresel Minimum	$f(3, 0{,}5) = 0$	
Arama Alanı	$-4{,}5 \leq x,y \leq 4{,}5$	

Beale Fonksiyonu'nun ilgili kabuller eşliğinde varsayılan grafiği Resim 3.1.'deki gibidir (Wikipedia, 2017i).



Resim 3. 1. Beale Fonksiyonu (Wikipedia, 2017i).

3.3.2. Booth fonksiyonu

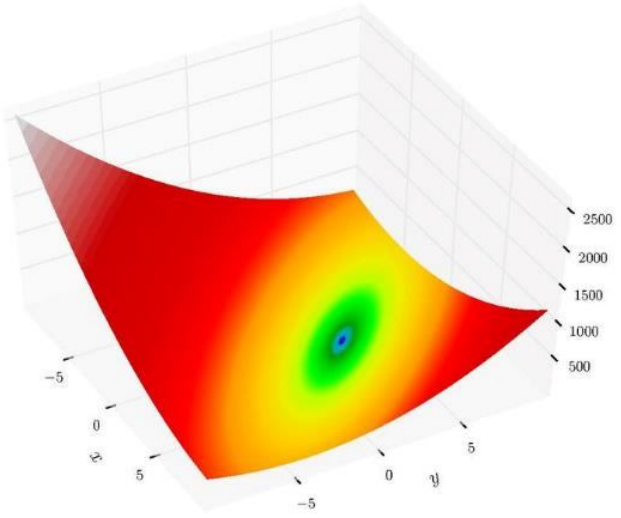
Booth Fonksiyonu, yayvan yapıda şekle sahip, iki-boyutlu – değişkenli bir test fonksiyonu olarak bilinmektedir.

Söz konusu fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.2.'de verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 2. Booth Fonksiyonu küresel min. ve arama alanı.

Booth Fonksiyonu	$f(x,y) = (x + 2 * y - 7)^2 + (2 * x + y - 5)^2$	(3.2.)
Küresel Minimum	$f(1,3) = 0$	
Arama Alanı	$-10 \leq x,y \leq 10$	

Booth Fonksiyonu'nun ifade edilen kabuller eşliğinde varsayılan grafiği Resim 3.2.'deki gibidir (Wikipedia, 2017i).



Resim 3. 2. Booth Fonksiyonu (Wikipedia, 2017i).

3.3.3. Bukin (no. 6) fonksiyonu

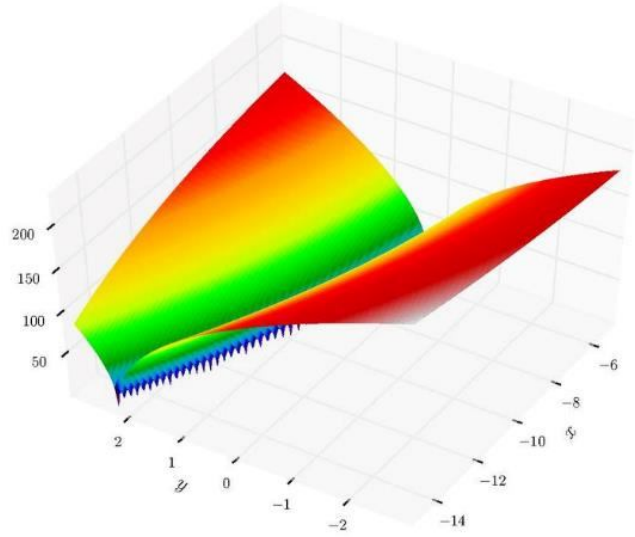
Bukin (no. 6) Fonksiyonu (6. Bukin Fonksiyonu), uzun bir sırada çok sayıda yerel minimumu içeren bir fonksiyondur. Bu kapsamda, iki-boyutlu – değişkenli olarak karşımıza çıkmaktadır.

İlgili fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.3.'te verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 3. Bukin (no. 6) Fonksiyonu küresel min. ve arama alanı.

Bukin (no. 6) Fonksiyonu	$f(x,y) = 100 * \sqrt{\lvert y - 0{,}01 * x^2 \rvert} + 0{,}01 * \lvert x + 10 \rvert$	(3.3.)
Küresel Minimum	$f(-10,1) = 0$	
Arama Alanı	$-15 \leq x \leq -5,$ $-3 \leq y \leq 3$	

Bukin (no. 6) Fonksiyonu'nun ilgili kabuller kapsamında varsayılan grafiği Resim 3.3.'teki gibidir (Wikipedia, 2017i).



Resim 3. 3. Bukin (no. 6) Fonksiyonu (Wikipedia, 2017i).

3.3.4. Cross-in-tray fonksiyonu

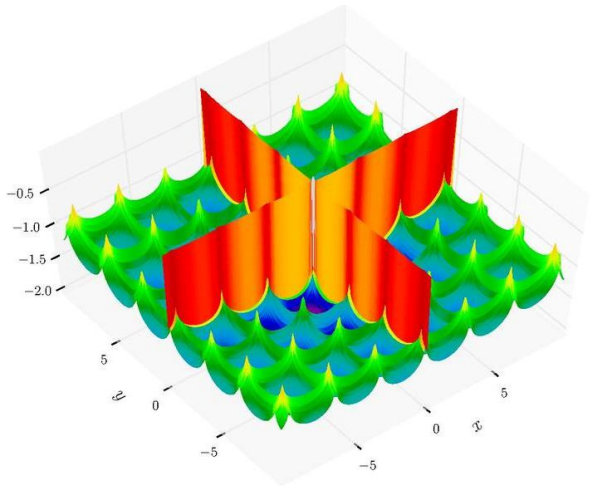
Cross-in Tray Fonksiyonu, iki-boyutlu – değişkenli bir test fonksiyonu olmakla beraber, aynı küresel minimumu aynı değerlerin farklı işaretlileriyle birlikte elde eden (çoklu küresel minimuma sahip) bir yapıya sahiptir. İsmindeki 'cross' ifadesi de buradan gelmektedir (Surjanovic ve Bingham, 2015).

İlgili fonksiyon ve fonksiyonun küresel minimumu ile arama alanına dair kabuller Çizelge 3.4.'te verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 4. Cross-in-Tray Fonksiyonu küresel min. ve arama alanı.

Cross-in-Tray Fonksiyonu	$f(x,y) = -0{,}0001 * \left(\left\lvert \sin(x) * \sin(y) * exp\left(\left\lvert 100 - \frac{\sqrt{x^2+y^2}}{\pi}\right\rvert\right)\right\rvert + 1\right)^{0{,}1}$	(3.4.)
Küresel Minimum	$K.M. \begin{cases} f(1{,}34941, 1{,}34941) = -2{,}06261, \\ f(-1{,}34941, 1{,}34941) = -2{,}06261, \\ f(1{,}34941, -1{,}34941) = -2{,}06261, \\ f(-1{,}34941, -1{,}34941) = -2{,}06261 \end{cases}$	
Arama Alanı	$-10 \leq x,y \leq 10$	

Cross-in-Tray Fonksiyonu'nun söz konusu kabuller eşliğinde varsayılan grafiği Resim 3.4.'teki gibidir (Wikipedia, 2017i).



Resim 3. 4. Cross-in-Tray Fonksiyonu (Wikipedia, 2017i).

3.3.5. Easom fonksiyonu

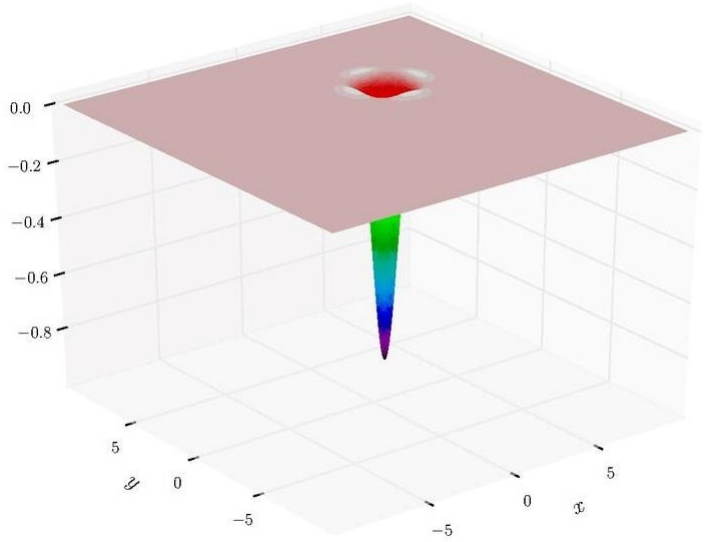
Easom Fonksiyonu, tek modlu, küresel minimumu küçük bir bölgede gözlemlenen, birkaç yerel minimuma sahip bir test fonksiyonudur (Surjanovic ve Bingham, 2015).

Easom Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.5.'te verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 5. Easom Fonksiyonu küresel min. ve arama alanı.

Easom Fonksiyonu	$f(x,y) = -\cos(x) * \cos(y) * exp\left(-((x-\pi)^2 + (y-\pi)^2)\right)$	(3.5.)
Küresel Minimum	$f(\pi,\pi) = -1$	
Arama Alanı	$-100 \leq x,y \leq 100$	

Easom Fonksiyonu'nun ilgili kabuller kapsamında varsayılan grafiği Resim 3.5.'teki gibidir (Wikipedia, 2017i).



Resim 3. 5. Easom Fonksiyonu (Wikipedia, 2017i).

3.3.6. Eggholder fonksiyonu

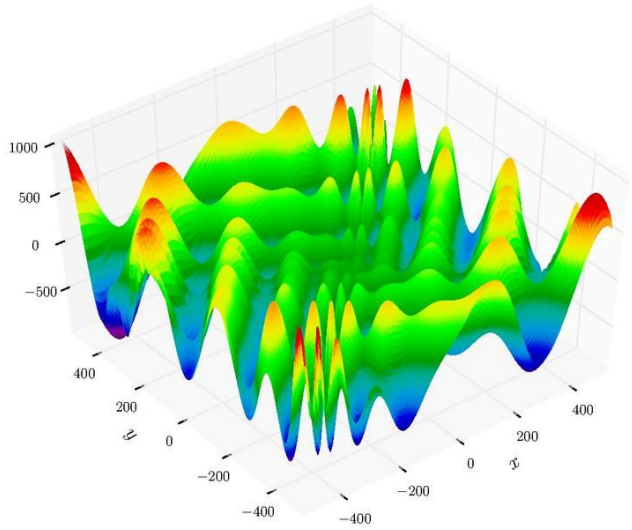
İki boyutlu – değişkenli bir diğer fonksiyon olarak Eggholder Fonksiyonu, bünyesinde çok sayıda yerel minimum içermektedir. Bu açıdan ele alındığında, çözülmesi zor fonksiyonlardan birisi olarak görülmektedir.

Eggholder Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.6.'da verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 6. Eggholder Fonksiyonu küresel min. ve arama alanı.

Eggholder Fonksiyonu	$f(x,y) = -(y+47) * sin\left(\sqrt{\left\lvert\frac{x}{2}+(y+47)\right\rvert}\right) - x * sin\left(\sqrt{\lvert x-(y+47)\rvert}\right)$	(3.6.)
Küresel Minimum	$f(512,404{,}2319) = -959{,}6407$	
Arama Alanı	$-512 \leq x,y \leq -512$	

Eggholder Fonksiyonu'nun ilgili kabuller kapsamında varsayılan grafiği Resim 3.6.'daki gibidir (Wikipedia, 2017i).



Resim 3. 6. Eggholder Fonksiyonu (Wikipedia, 2017i).

3.3.7. Levy (no. 13) fonksiyonu

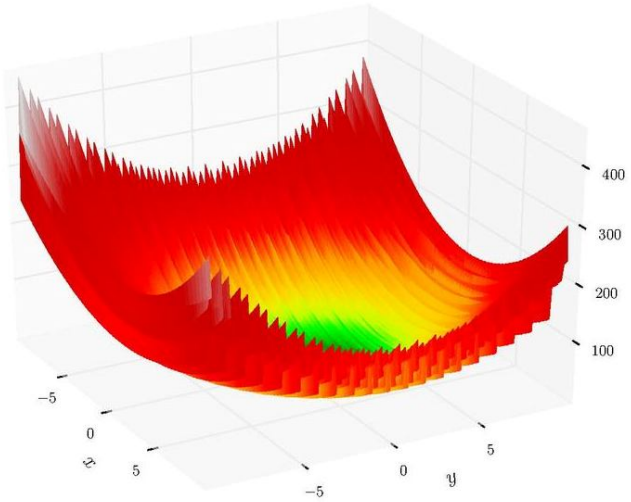
İki boyutlu – değişkenli Levy (no. 13) Fonksiyonu da (13. Levy Fonksiyonu) çok sayıda yerel minimuma sahip bir test fonksiyonu olarak karşımıza çıkmaktadır.

Levy (no. 13) Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.7.'de sunulmuştur (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 7. Levy (no. 13) Fonksiyonu küresel min. ve arama alanı.

Levy (no. 13) Fonksiyonu	$f(x,y) = sin^2 * (3 * \pi * x) + (x-1)^2 * \left(1 + sin^2 * (3 * \pi * y)\right) + (y-1)^2 * (1 + sin^2 * (2 * \pi * y))$	(3.7.)
Küresel Minimum	$f(1,1) = 0$	
Arama Alanı	$-10 \leq x,y \leq 10$	

Levy (no. 13) Fonksiyonu'nun ilgili kabuller eşliğinde varsayılan grafiği Resim 3.7.'deki gibidir (Wikipedia, 2017i).



Resim 3. 7. Levy (no. 13) Fonksiyonu (Wikipedia, 2017i).

3.3.8. Matyas fonksiyonu

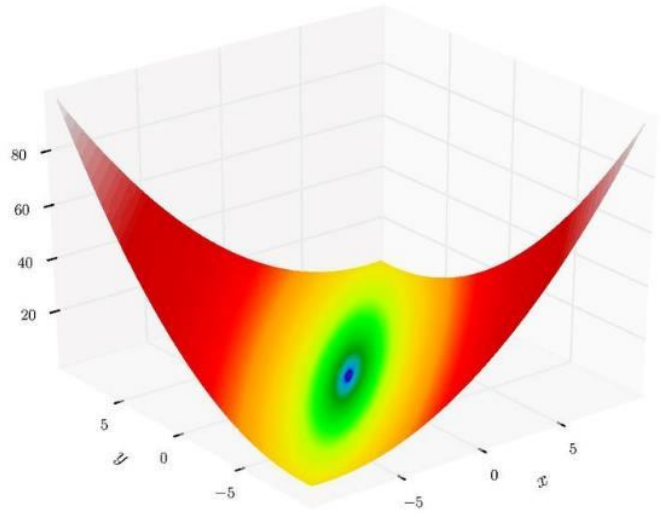
Genel olarak hiç yerel minimuma sahip olmayan Matyas Fonksiyonu, yayvan yapıdaki, iki-boyutlu – değişkenli bir başka test fonksiyonudur (Surjanovic ve Bingham, 2015).

Matyas Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.8.'de verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 8. Matyas Fonksiyonu küresel min. ve arama alanı.

Matyas Fonksiyonu	$f(x,y) = 0{,}26 * (x^2 + y^2) - 0{,}48 * x * y$	(3.8.)
Küresel Minimum	$f(0,0) = 0$	
Arama Alanı	$-10 \leq x,y \leq 10$	

Matyas Fonksiyonu'nun ilgili kabuller bağlamında varsayılan grafiği Resim 3.8.'deki gibidir (Wikipedia, 2017i).



Resim 3. 8. Matyas Fonksiyonu (Wikipedia, 2017i).

3.3.9. McCormick fonksiyonu

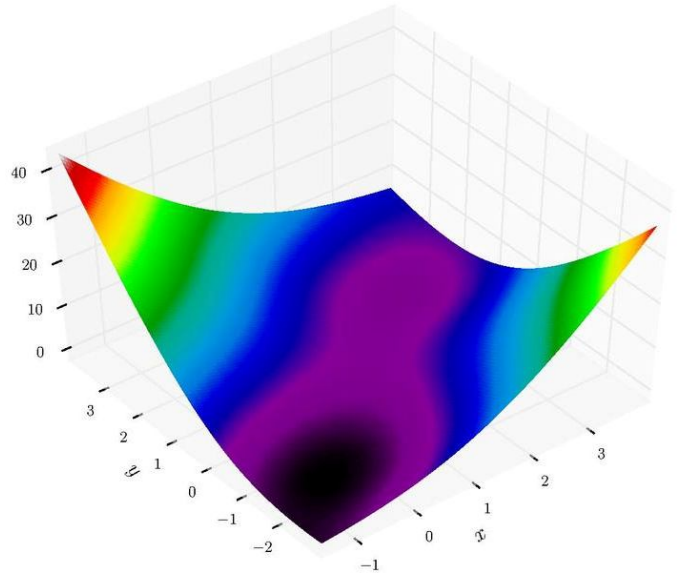
McCormick Fonksiyonu iki-boyutlu – değişkenli, yayvan şekilde yapıya sahip test fonksiyonlarından bir diğeri olarak yine ilgili literatürde yer almaktadır.

McCormick Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanı bağlamındaki kabuller Çizelge 3.9.'da verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 9. McCormick Fonksiyonu küresel min. ve arama alanı.

McCormick Fonksiyonu	$f(x,y) = \sin(x+y) + (x-y)^2 - 1{,}5 * x + 2{,}5 * y + 1$	(3.9.)
Küresel Minimum	$f(-0{,}54719, -1{,}54719) = -1{,}9133$	
Arama Alanı	$-1{,}5 \leq x \leq 4,$ $-3 \leq y \leq 4$	

McCormick Fonksiyonu'nun söz konusu kabuller eşliğinde varsayılan grafiği Resim 3.9.'daki gibidir (Wikipedia, 2017i).



Resim 3. 9. McCormick Fonksiyonu (Wikipedia, 2017i).

3.3.10. Schaffer (no. 2) fonksiyonu

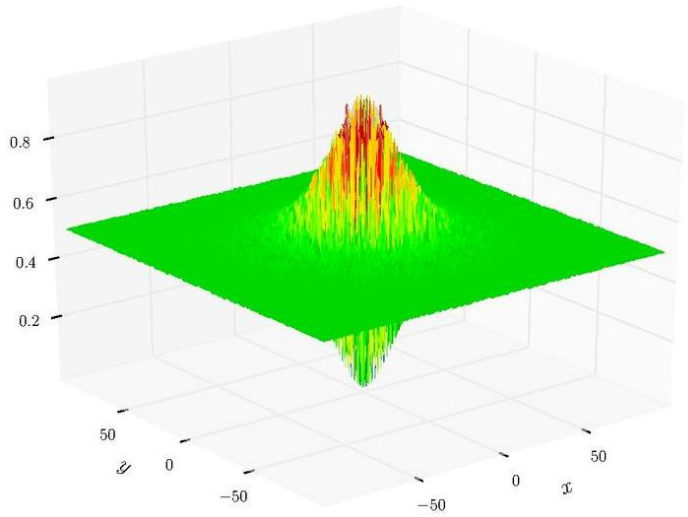
Schaffer (no. 2) Fonksiyonu (2. Schaffer Fonksiyonu) iki-boyutlu – değişkenli, yoğun orta alandan dışa doğru yerel minimumlarla kendini gösteren bir test fonksiyonudur.

İlgili fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.10.'da gösterilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 10. Schaffer (no. 2) Fonksiyonu küresel min. ve arama alanı.

Schaffer (no. 2) Fonksiyonu	$f(x,y) = 0{,}5 + \frac{sin^2(x^2 - y^2) - 0{,}5}{(1 + 0{,}001 * (x^2 + y^2))^2}$	(3.10.)
Küresel Minimum	$f(0,0) = 0$	
Arama Alanı	$-100 \leq x,y \leq 100$	

Schaffer (no. 2) Fonksiyonu'nun ilgili kabuller bağlamında varsayılan grafiği Resim 3.10.'daki gibidir (Wikipedia, 2017i).



Resim 3. 10. Schaffer (no. 2) Fonksiyonu (Wikipedia, 2017i).

3.3.11. Ackley fonksiyonu

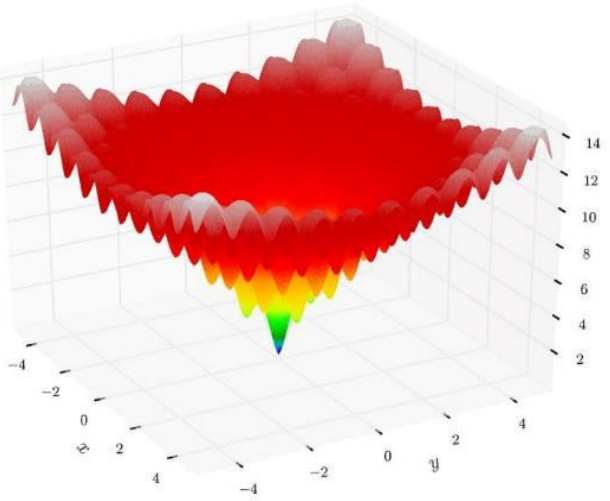
Ackley (1987:2012) tarafından literatüre kazandırılan ve yine sonraları geliştirilen (Back, 1996) Ackley Fonksiyonu, ortasında küresel minimuma ulaşan bir bölge ile birlikte, optimizasyon amaçlı geliştirilen çözüm teknikleri için etkin bir değerlendirme faktörünü oluşturan, çok sayıda yerel minimuma sahip dış bölgeden oluşan bir fonksiyondur (Surjanovic ve Bingham, 2015).

Ackley Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.11.'de verilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 11. Ackley Fonksiyonu küresel min. ve arama alanı.

Ackley Fonksiyonu	$f(x) = -20 * exp\left(-0{,}2 * \sqrt{\frac{1}{n} * \sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n} * \sum_{i=1}^{n} \cos(2 * \pi * x_i)\right) + 20 + \exp(1)$	(3.11.)
Küresel Minimum	$f(0, \dots, 0) = 0$	
Arama Alanı	$-32 \leq x_i \leq 32,$ $1 \leq i \leq n$	

Ackley Fonksiyonu'nun ilgili kabuller eşliğinde varsayılan grafiği Resim 3.11.'deki gibidir (Wikipedia, 2017i).



Resim 3. 11. Ackley Fonksiyonu (Wikipedia, 2017i).

3.3.12. Rastrigin fonksiyonu

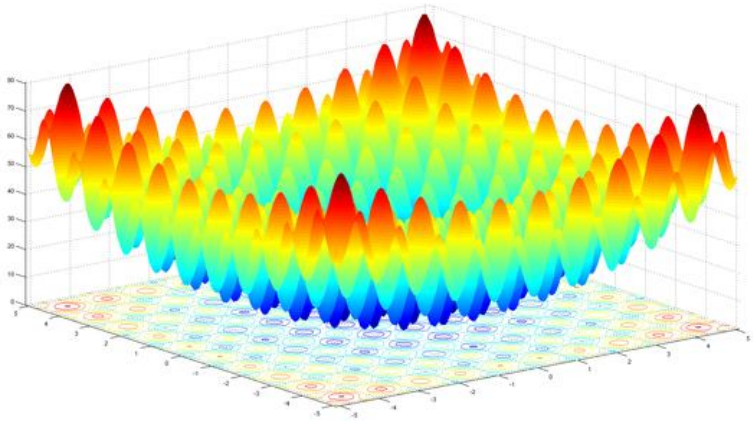
İlk olarak Rastrigin (1974) tarafından iki-boyutlu hali literatüre kazandırılan, akabinde ise genişletilen (Mühlenbein ve ark., 1991) Rastrigin Fonksiyonu, konveks ve lineer olmayan, çok modlu bir test fonksiyonudur (Surjanovic ve Bingham, 2015; Wikipedia, 2017g).

Çok sayıda ancak düzenli bir şekilde yerleşmiş yerel minimumlara sahip olan (Surjanovic ve Bingham, 2015; Wikipedia, 2017g) Rastrigin Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.12.'de kısaca sunulmuştur (Surjanovic ve Bingham, 2015; Wikipedia, 2017g; 2017i).

Çizelge 3. 12. Rastrigin Fonksiyonu küresel min. ve arama alanı.

Rastrigin Fonksiyonu	$f(x) = 10 * n + \sum_{i=1}^{n} [x_i^2 - 10 * \cos(2\pi * x_i)]$	(3.12.)
Küresel Minimum	$f(0, \ldots, 0) = 0$	
Arama Alanı	$-5{,}12 \leq x_i \leq 5{,}12,$ $1 \leq i \leq n$	

Rastrigin Fonksiyonu'nun ilgili kabuller bağlamında varsayılan grafiği Resim 3.12.'deki gibidir (Wikipedia, 2017g; 2017i).



Resim 3. 12. Rastrigin Fonksiyonu (Wikipedia, 2017g).

3.3.13. Rosenbrock fonksiyonu

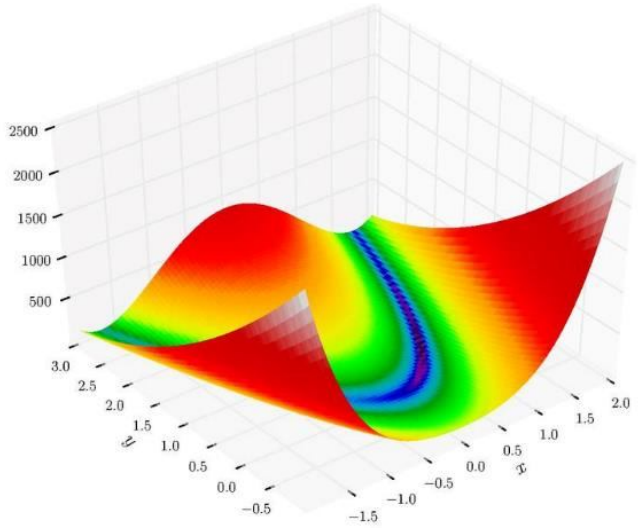
Rosenbrock Fonksiyonu, 1960 yılında Howard Harry Rosenbrock tarafından literatüre kazandırılan, konveks olmayan, sürekli bir fonksiyondur (Karaboğa, 2014; Rosenbrock, 1960; Wikipedia, 2017h). Fonksiyonda küresel minimumun uzun, dar ve parabolik yapıda düz bir vadide olması, fonksiyonu, minimum konumu (vadi) bulunması kolay ancak küresel minimumu konumlandırmanın zor olduğu bir yapı haline getirmektedir (Karaboğa, 2014; Picheny ve ark., 2013; Wikipedia, 2017h).

Rosenbrock Fonksiyonu ve bu fonksiyona dair küresel minimum ile arama alanına yönelik kabuller Çizelge 3.13.'te sunulmuştur (Rosenbrock, 1960; Surjanovic ve Bingham, 2015; Wikipedia, 2017h).

Çizelge 3. 13. Rosenbrock Fonksiyonu küresel min. ve arama alanı.

Rosenbrock Fonksiyonu	$f(x) = \sum_{i=1}^{n-1} [100 * (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(3.13.)
Küresel Minimum	$f(1, \dots, 1) = 0$	
Arama Alanı	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$	

Rosenbrock Fonksiyonu'nun söz konusu kabuller eşliğinde varsayılan grafiği Resim 3.13.'teki gibidir (Wikipedia, 2017h; 2017i).



Resim 3. 13. Rosenbrock Fonksiyonu (Wikipedia, 2017h; 2017i).

3.3.14. Sphere (küre) fonksiyonu

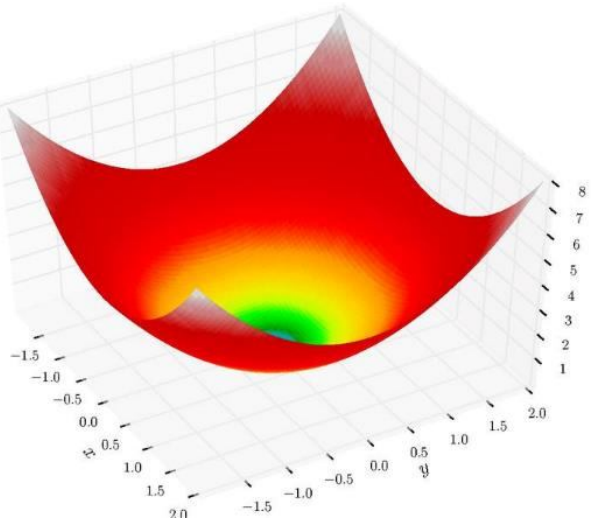
Sphere (Küre) Fonksiyonu, konveks yapıda, düzgün, simetrik ve küresel minimumu dışında yerel minimumlarıyla ifade edilen bir fonksiyondur (Karaboğa, 2014; Surjanovic ve Bingham, 2015).

İlgili fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.14.'te kısaca gösterilmiştir (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 14. Sphere (Küre) Fonksiyonu küresel min. ve arama alanı.

Sphere (Küre) Fonksiyonu	$f(x) = \sum_{i=1}^{n} x_i^2$	(3.14.)
Küresel Minimum	$f(0, \dots, 0) = 0$	
Arama Alanı	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$	

Sphere (Küre) Fonksiyonu'nun ilgili kabuller eşliğinde varsayılan grafiği Resim 3.14.'teki gibidir (Wikipedia, 2017i).



Resim 3. 14. Sphere (Küre) Fonksiyonu (Wikipedia, 2017i).

3.3.15. Styblinski–Tang fonksiyonu

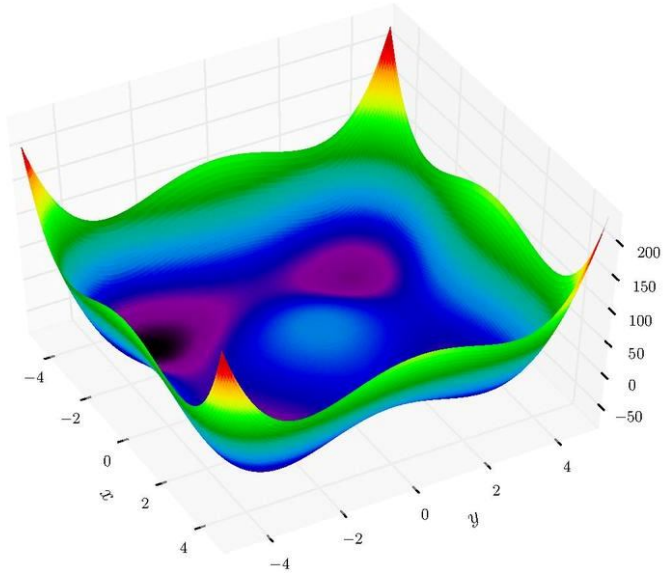
Styblinski–Tang Fonksiyonu, çok-boyutlu – değişkenli yapıda da karşılığı olan bir test fonksiyonudur.

Söz konusu fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.15.'te sunulmuştur (Surjanovic ve Bingham, 2015; Wikipedia, 2017i).

Çizelge 3. 15. Styblinski–Tang Fonksiyonu küresel min. ve arama alanı.

Styblinski–Tang Fonksiyonu	$f(x) = \frac{\sum_{i=1}^{n} x_i^4 - 16 * x_i^2 + 5 * x_i}{2}$	(3.15.)
Küresel Minimum	$f(-2{,}903534, \ldots, -2{,}903534) = -39{,}16599 * n$	
Arama Alanı	$-5 \leq x_i \leq 5,$ $1 \leq i \leq n$	

Styblinski–Tang Fonksiyonu'nun ifade edilen kabuller eşliğinde varsayılan grafiği Resim 3.15.'teki gibidir (Wikipedia, 2017i).



Resim 3. 15. Styblinski–Tang Fonksiyonu (Wikipedia, 2017i).

3.3.16. Mishra'nın kuşu (Mishra's bird) fonksiyonu

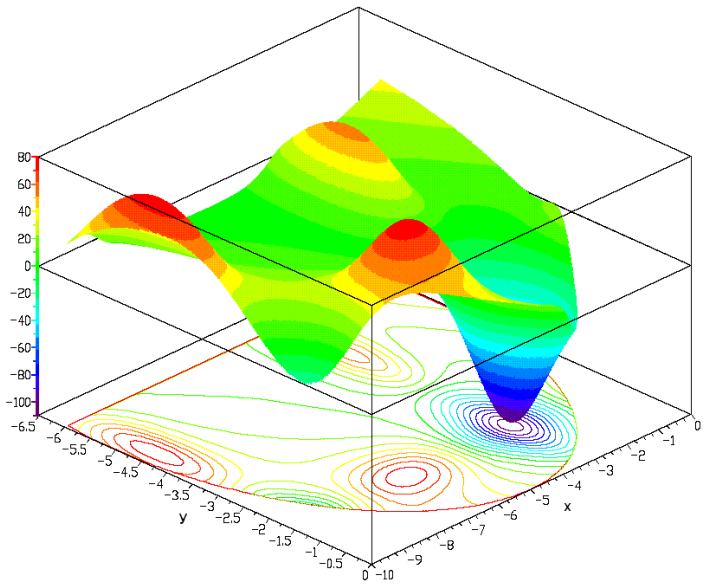
Mishra'nın Kuşu (Mishra's Bird) Fonksiyonu, birkaç yerel ve bir küresel minimuma sahip olan, iki-boyutlu – değişkenli, kısıtlı bir test fonksiyonudur (Phoenix Integration, 2017).

Fonksiyon ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.16.'da gösterilmiştir (Mishra, 2006; Phoenix Integration, 2017; Wikipedia, 2017i).

Çizelge 3. 16. Mishra'nın Kuşu (Mishra's Bird) Fonk. küresel min. ve arama alanı.

Mishra'nın Kuşu (Mishra's Bird) Fonksiyonu	$f(x,y) = \sin(y) * exp[(1-\cos(x))^2] + \cos(x) * exp[(1-\sin(y))^2] + (x-y)^2$ Kısıt $(x+5)^2 + (y+5)^2 < 25$	(3.16.)
Küresel Minimum	$f(-3{,}1302468, -1{,}5821422) = -106{,}7645367$	
Arama Alanı	$-10 \leq x \leq 0,$ $-6{,}5 \leq y \leq 0$	

Mishra'nın Kuşu (Mishra's Bird) Fonksiyonu'nun ilgili kabuller eşliğinde varsayılan grafiği Resim 3.16.'daki gibidir (Wikipedia, 2017i).



Resim 3. 16. Mishra'nın Kuşu (Mishra's Bird) Fonksiyonu (Wikipedia, 2017i).

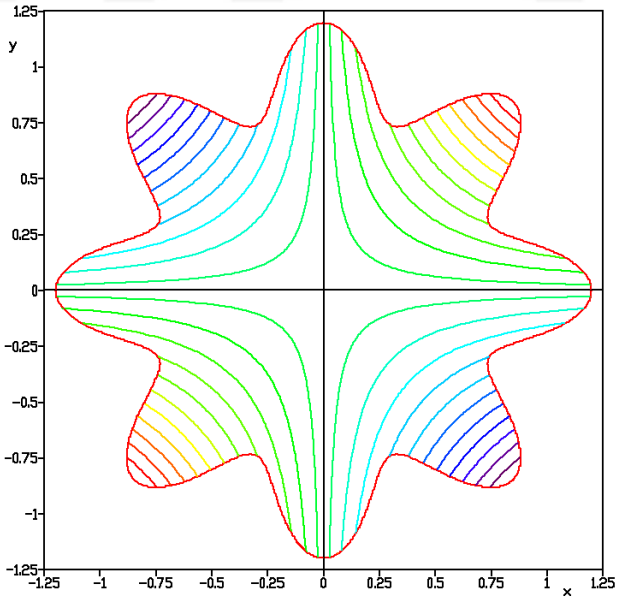
3.3.17. Simionescu fonksiyonu

Kısıtlı bir test fonksiyonu olan Simionescu Fonksiyonu (Simionescu, 2014) ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.17.'de gösterilmiştir (Wikipedia, 2017i).

Çizelge 3. 17. Simionescu Fonksiyonu küresel min. ve arama alanı.

Simionescu Fonksiyonu	$f(x,y) = 0{,}1 * x * y$ Kısıt $x^2 + y^2 \leq \left(r_T + r_S * cos\left(n * arctan\frac{x}{y}\right)\right)^2$ $r_T = 1 \quad r_S = 0{,}2 \quad n = 8$	(3.17.)
Küresel Minimum	$f(\pm 0{,}85586214, \mp 0{,}85586214) = -0{,}072625$	
Arama Alanı	$-1{,}25 \leq x,y \leq 1{,}25$	

Simionescu Fonksiyonu'nun ifade edilen kabuller kapsamında varsayılan grafiği Resim 3.17.'deki gibidir (Wikipedia, 2017i).



Resim 3. 17. Simionescu Fonksiyonu (Wikipedia, 2017i).

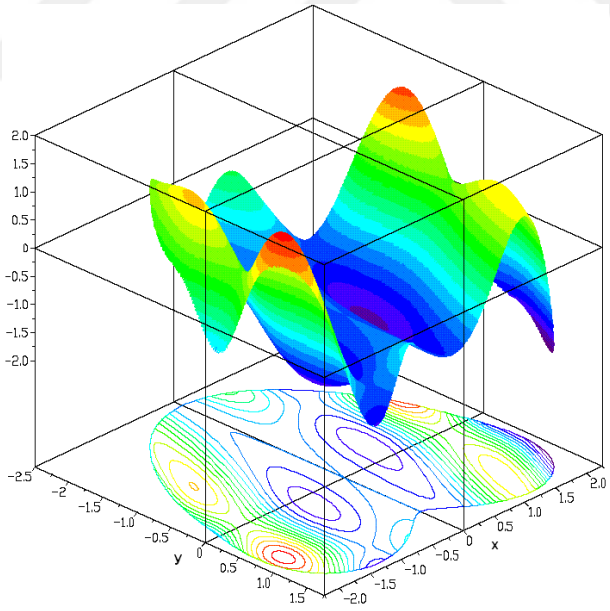
3.3.18. Townsend fonksiyonu

Çalışmaya konu olan son kısıtlı test fonksiyonu olarak Townsend Fonksiyonu ve fonksiyonun küresel minimumu ile arama alanına yönelik kabuller Çizelge 3.18.'de verilmiştir (Townsend, 2014; Wikipedia, 2017i).

Çizelge 3. 18. Townsend Fonksiyonu küresel min. ve arama alanı.

Townsend Fonksiyonu	$f(x,y) = -[\cos((x-0{,}1)*y)]^2 - x*\sin(3*x+y)$ Kısıt $x^2+y^2 < [2*\cos(t) - 0{,}5*\cos(2*t) - 0{,}25*\cos(3*t) - 0{,}125*\cos(4*t)]^2 + [2*\sin(t)]^2$ $t = Atan2(\frac{y}{x})$	(3.18.)
Küresel Minimum	$f(2{,}0052938, 1{,}1944509) = 2{,}0239884$	
Arama Alanı	$-2{,}25 \le x \le 2{,}5,$ $-2{,}5 \le y \le 1{,}75$	

Fonksiyonun varsayılan grafiği Resim 3.18.'deki gibidir (Wikipedia, 2017i).



Resim 3. 18. Townsend Fonksiyonu (Wikipedia, 2017i).

Bu noktadan sonra ifade edilen 5 adet fonksiyon da IEEE Evrimsel Hesaplama Kongresi'ndeki bir yarışma kapsamında sunulan fonksiyonlardır. Etkinlik ve fonksiyonlar ve diğer detaylara ilişkin bilgi için okuyucular (Liang ve ark., 2014)'e başvurabilirler.

3.3.19. Döndürülmüş Cigar (rotated Cigar) fonksiyonu

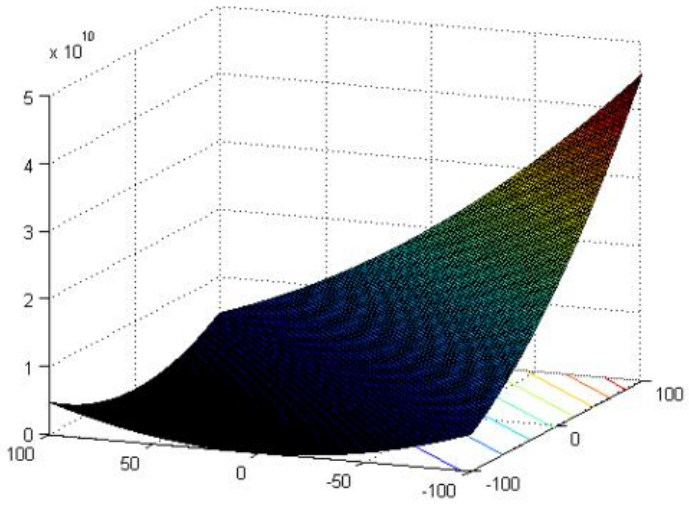
Döndürülmüş Cigar (Rotated Cigar) Fonksiyonu, ayrıştırılamaz nitelikte, çok-boyutlu bir test fonksiyonudur. Diğer yandan, çözüm uzayındaki yapısı, –optimum noktalar bağlamında– düz bir yüzeye ancak dar kapsamda bir akışa sahiptir.

Döndürülmüş Cigar Fonksiyonu bağlamında söz konusu kabuller Çizelge 3.19.'da sunulmuştur (Liang ve ark., 2014).

Çizelge 3. 19. Döndürülmüş Cigar (Rotated Cigar) Fonk. küresel min. ve arama alanı.

Döndürülmüş Cigar (Rotated Cigar) Fonksiyonu	Cigar Fonksiyonu standart formunu (*f*) içermek suretiyle (Liang *et al.*, 2014) $F(x) = f\left(M_2 * (x - o_2)\right) + F^*$	(3.19.)
Arama Alanı	$-100 \leq x \leq 100$	

Döndürülmüş Cigar (Rotated Cigar) Fonksiyonu'nun söz konusu kabuller kapsamında varsayılan grafiği Resim 3.19.'da verilmiştir (Liang ve ark., 2014).



Resim 3. 19. Döndürülmüş Cigar (Rotated Cigar) Fonk. (Liang ve ark., 2014).

3.3.20. Döndürülmüş yüksek kond. eliptik (rotated high cond. ellipt.) fonk.

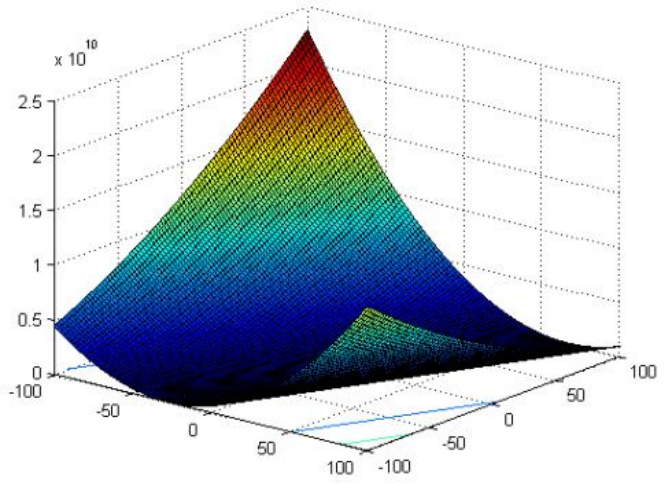
Döndürülmüş Yüksek Kondisyonlu Eliptik (Rotated High Conditioned Elliptic) Fonksiyonu, ayrıştırılamaz ve çok-boyutlu yapıda, ikinci dereceden bir test fonksiyonudur. Girişteki ufak değişiklikler, çıkışta büyük değişikliklere yol açmaktadır.

Fonksiyon ve fonksiyona dair kabuller Çizelge 3.20.'de verilmiştir (Liang ve ark., 2014).

Çizelge 3. 20. Döndürülmüş Yüksek Kondis. Eliptik (Rot. H. C. El.) Fonk. küresel min. ve arama alanı.

Döndürülmüş Yüksek Kondisyonlu Eliptik (Rotated High Conditioned Elliptic) Fonksiyon	Yüksek Kondisyonlu Eliptik Fonksiyon standart formunu (f) içermek suretiyle (Liang *et al.*, 2014) $F(x) = f\left(M_1 * (x - o_1)\right) * F^*$	(3.20.)
Arama Alanı	$-100 \leq x \leq 100$	

İlgili fonksiyonun ifade edilen kabullere göre varsayılan grafiği Resim 3.20.'deki gibidir (Liang ve ark., 2014).



Resim 3. 20. Döndürülmüş Yüksek Kondisyonlu Eliptik (Rot. H. C. Ellipt.) Fonk. (Liang ve ark., 2014).

3.3.21. Kaydırılmış ve dönd. Ackley (shifted and rotated Ackley) fonk.

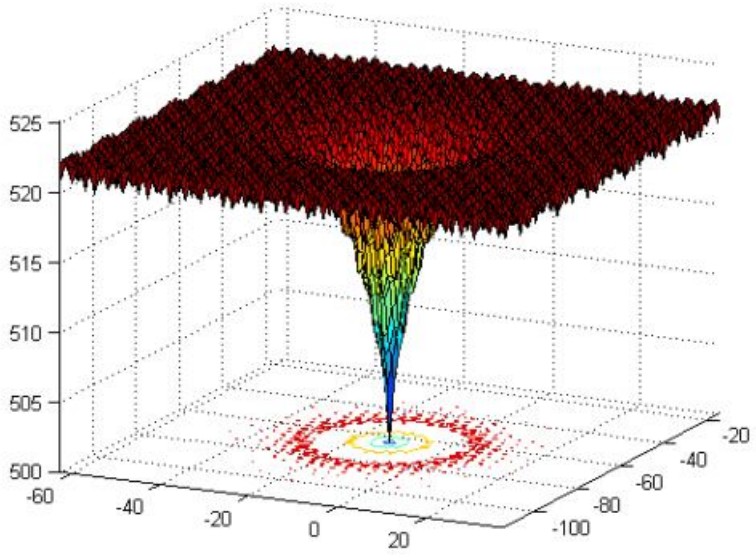
Kaydırılmış ve Döndürülmüş Ackley (Shifted and Rotated Ackley) Fonksiyonu, çok-boyutlu, ayrıştırılamaz yapıdaki bir başka test fonksiyonudur.

Kaydırılmış ve Döndürülmüş Ackley Fonksiyonu ve fonksiyonun geneline yönelik kabuller Çizelge 3.21.'de verilmiştir (Liang ve ark., 2014).

Çizelge 3. 21. Kaydırılmış ve Döndürülmüş Ackley (S. R. Ackley) Fonk. küresel min. ve arama alanı.

Kaydırılmış ve Döndürülmüş Ackley (Shifted and Rotated Ackley) Fonksiyonu	Ackley Fonksiyonu standart formunu (f) içermek suretiyle (Liang *et al.*, 2014) $F(x) = f\left(M_3 * (x - o_3)\right) + F^*$	(3.21.)
Arama Alanı	$-100 \leq x \leq 100$	

Kaydırılmış ve Döndürülmüş Ackley Fonksiyonu'nun kabullere göre varsayılan grafiği Resim 3.21.'de verilmiştir (Liang ve ark., 2014).



Resim 3. 21. Kaydırılmış ve Döndürülmüş Ackley (Shift. Rot. Ackley) Fonk. (Liang ve ark., 2014).

3.3.22. Kaydırılmış ve dönd. Rastrigin (shifted and rotated Rastrigin) fonk.

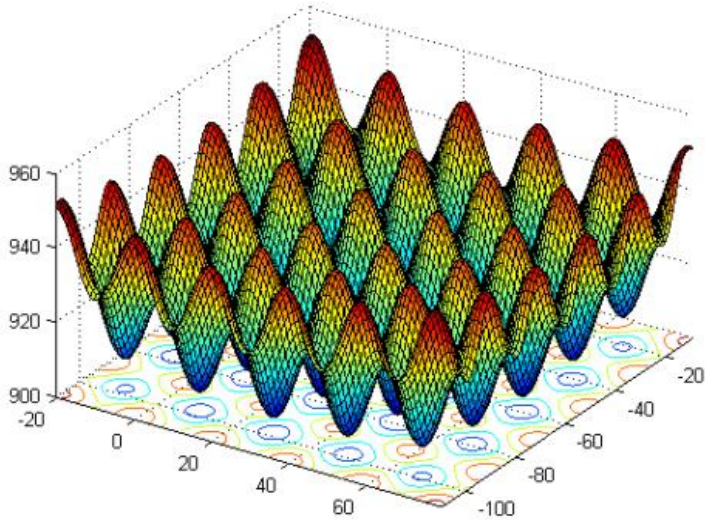
Kaydırılmış ve Döndürülmüş Rastrigin (Shifted and Rotated Rastrigin) Fonksiyonu, ayrıştırılamaz ve çok sayıda yerel optimumlara sahip, çok-boyutlu bir test fonksiyonudur.

Fonksiyon ve fonksiyona yönelik kabuller Çizelge 3.22.'da sunulmuştur (Liang ve ark., 2014).

Çizelge 3. 22. Kaydırılmış ve Döndürülmüş Rastrigin (S. R. Rast.) Fonk. küresel min. ve arama alanı.

Kaydırılmış ve Döndürülmüş Rastrigin (Shifted and Rotated Rastrigin) Fonksiyonu	Rastrigin Fonksiyonu standart formunu (f) içermek suretiyle (Liang *et al.*, 2014) $F(x) = f\left(M_4 * \left(\frac{5{,}12 * (x - o_4)}{100}\right)\right) + F^*$	(3.22.)
Arama Alanı	$-100 \leq x \leq 100$	

Kaydırılmış ve Döndürülmüş Rastrigin Fonksiyonu'nun kabul eşliğindeki varsayılan grafiği Resim 3.22.'deki gibidir (Liang ve ark., 2014).



Resim 3. 22. Kaydırılmış ve Döndürülmüş Rastrigin (Shift. Rot. Rast.) Fonk. (Liang ve ark., 2014).

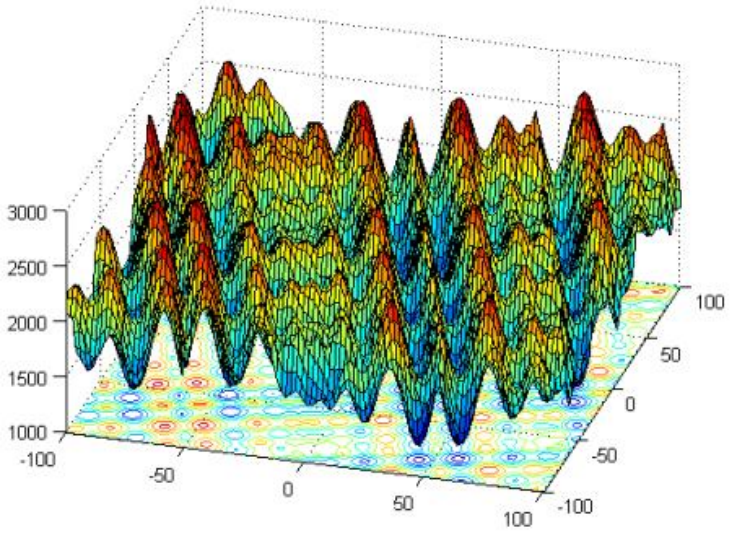
3.3.23. Kaydırılmış ve dönd. Schwefel (shifted and rotated Schwefel) fonk.

Kaydırılmış ve Döndürülmüş Schwefel (Shifted and Rotated Schwefel) Fonksiyonu, –bir önceki fonksiyona benzer olarak– ayrıştırılamaz ve çok sayıda yerel optimumlara sahip, çok-boyutlu bir test fonksiyonudur. Kaydırılmış ve Döndürülmüş Schwefel Fonksiyonu ve fonksiyon kapsamında genel kabuller Çizelge 3.23.'te gösterilmiştir (Liang ve ark., 2014).

Çizelge 3. 23. Kaydırılmış ve Döndürülmüş Schwefel (S. R. Schw.) Fonk. küresel min. ve arama alanı.

Kaydırılmış ve Döndürülmüş Schwefel (Shifted and Rotated Schwefel) Fonksiyonu	Schwefel Fonksiyonu standart formunu (f) içermek suretiyle (Liang *et al.*, 2014) $F(x) = f\left(M_5 * \left(\frac{1000 * (x - o_5)}{100}\right)\right) + F^*$	(3.23.)
Arama Alanı	$-100 \leq x \leq 100$	

Kaydırılmış ve Döndürülmüş Schwefel (Shifted and Rotated Schwefel) Fonksiyonu'nun söz konusu kabuller bağlamında varsayılan grafiği Resim 3.23.'te verilmiştir (Liang ve ark., 2014).



Resim 3. 23. Kaydırılmış ve Döndürülmüş Schwefel (Shift. Rot. Schw.) Fonk. (Liang ve ark., 2014).

Bu noktadan sonra, ilerleyen alt-başlıklar kapsamındaki fonksiyonlar, çok-amaçlı optimizasyon yönelimli olarak dikkate alınan test fonksiyonlarıdır.

3.3.24. Fonseca ve Fleming fonksiyonu

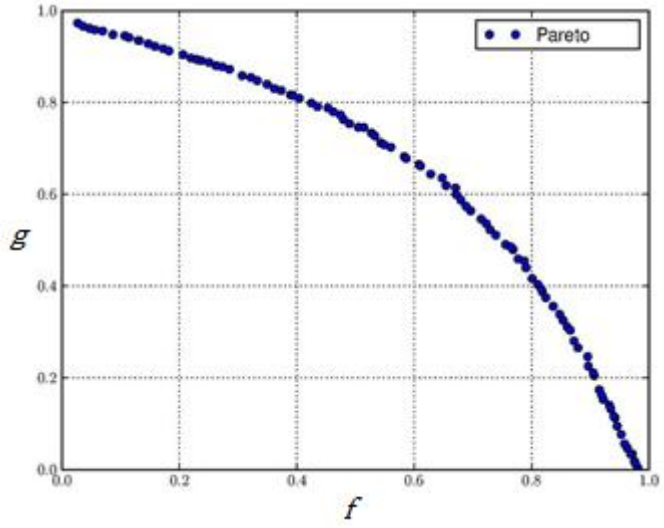
Fonseca ve Fleming Fonksiyonu, iki amaç fonksiyon üzerinden kurulmuş, kısıtsız çok-amaçlı bir test fonksiyonudur.

İlgili fonksiyon bağlamındaki kabuller Çizelge 3.24.'te sunulmuştur (Fonseca ve Fleming, 1995; Wikipedia, 2017i).

Çizelge 3. 24. Fonseca ve Fleming Fonksiyonu – ilgili kabuller.

Fonseca ve Fleming Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x) = 1 - exp\left[-\sum_{1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right]$ $g(x) = 1 - exp\left[-\sum_{1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right]$	(3.24.)
Arama Alanı	$-4 \le x_i \le 4$ $1 \le i \le n$	

Fonseca ve Fleming Fonksiyonu'nun ifade edilen kabuller kapsamında, pareto optimumlarının varsayılan grafiği Resim 3.24.'teki gibidir (Wikipedia, 2017i).



Resim 3. 24. Fonseca ve Fleming Fonksiyonu (Wikipedia, 2017i).

3.3.25. Kursawe fonksiyonu

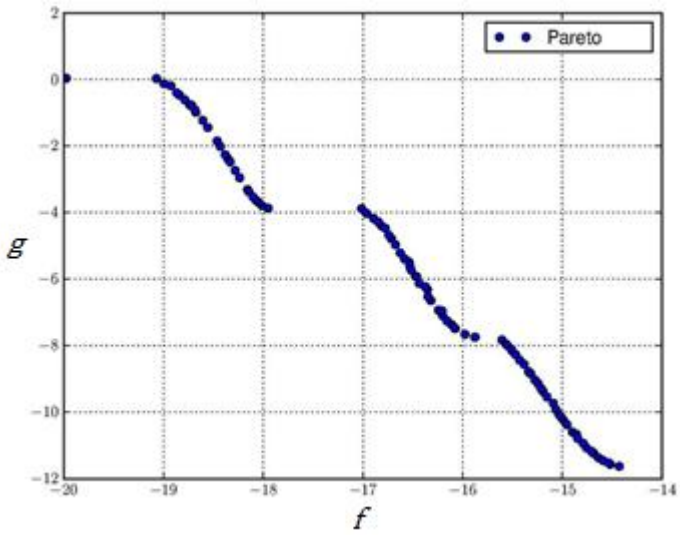
Kursawe Fonksiyonu, iki amaç fonksiyon çerçevesinde oluşturulmuş, kısıtsız çok-amaçlı bir test fonksiyonudur.

Kursawe Fonksiyonu'na dair kabuller Çizelge 3.25.'te gösterilmiştir (Kursawe, 1990; Wikipedia, 2017i).

Çizelge 3. 25. Kursawe Fonksiyonu ve ilgili kabuller.

	Minimize (bütün fonksiyonlar):	
Kursawe Fonksiyonu	$f(x) = \sum_{i=1}^{2}\left[-10 * \exp\left(-0{,}2 * \sqrt{x_i^2 + x_{i+1}^2}\right)\right]$ $g(x) = \sum_{i=1}^{3}[\lvert x_i\rvert^{0,8} + 5 * sin(x_i^3)]$	(3.25.)
Arama Alanı	$-5 \leq x_i \leq 5$ $1 \leq i \leq 3$	

Kursawe Fonksiyonu'nun söz konusu kabuller bağlamında ve pareto optimumlara göre varsayılan grafiği Resim 3.25.'teki gibidir (Wikipedia, 2017i).



Resim 3. 25. Kursawe Fonksiyonu (Wikipedia, 2017i).

3.3.26. Zitzler-Deb-Thiele (no. 1) fonksiyonu

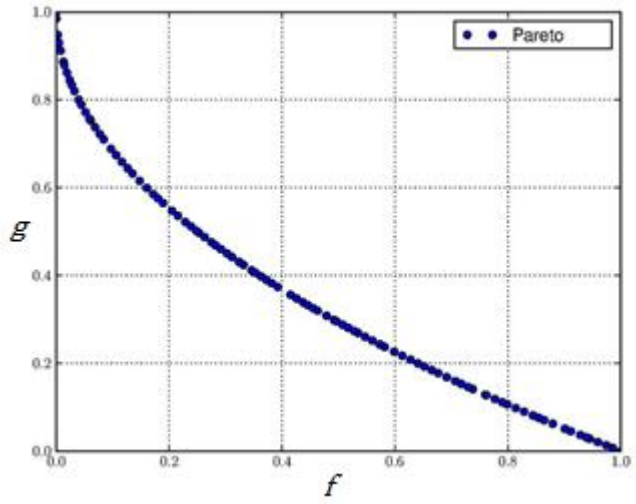
Zitzler-Deb-Thiele (no. 1) Fonksiyonu (1. Zitzler-Deb-Thiele Fonksiyonu), (birisi birinci ve üçüncü fonksiyonu içeren) dört amaç fonksiyonu içeren, kısıtsız çok-amaçlı bir test fonksiyonudur.

İlgili fonksiyon ve fonksiyona yönelik kabuller Çizelge 3.26.'da gösterilmiştir (Wikipedia, 2017i).

Çizelge 3. 26. Zitzler-Deb-Thiele (no. 1) Fonksiyonu ve ilgili kabuller.

Zitzler-Deb-Thiele (no. 1) Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x) = x_1$ $g(x) = h(x) * j\big(f(x), h(x)\big)$ $h(x) = 1 + \frac{9}{29} * \sum_{i=2}^{30} x_,$ $j\big(f(x), h(x)\big) = 1 - \sqrt{\frac{f(x)}{h(x)}}$	(3.26.)
Arama Alanı	$0 \leq x_i \leq 1$ $1 \leq i \leq 30$	

Zitzler-Deb-Thiele (no. 1) Fonksiyonu'nun ifade edilen kabullere göre, pareto optimumlar kapsamındaki varsayılan grafiği Resim 3.26.'daki gibidir (Wikipedia, 2017i).



Resim 3. 26. Zitzler-Deb-Thiele (no. 1) Fonksiyonu (Wikipedia, 2017i).

3.3.27. Zitzler-Deb-Thiele (no. 3) fonksiyonu

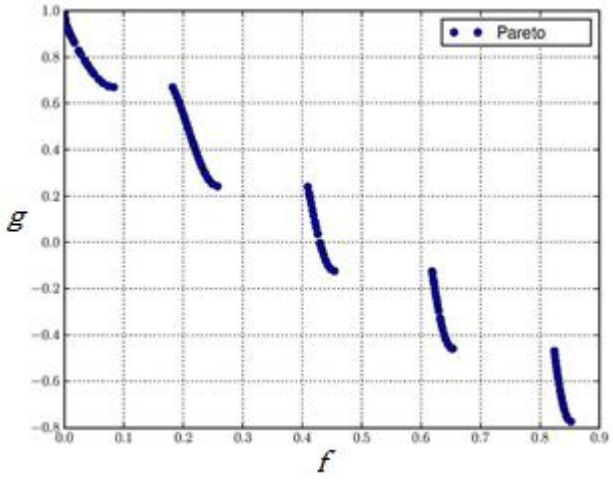
Zitzler-Deb-Thiele (no. 3) Fonksiyonu (2. Zitzler-Deb-Thiele Fonksiyonu), (birisi birinci ve üçüncü fonksiyonu içeren) dört amaç fonksiyonu içeren, kısıtsız çok-amaçlı bir test fonksiyonudur. Aynı isimdeki birinci fonksiyondan farkı, kapsam içerisindeki dördüncü fonksiyondan kaynaklanmaktadır.

İlgili fonksiyon ve fonksiyona yönelik kabuller Çizelge 3.27.'de gösterilmiştir (Wikipedia, 2017i).

Çizelge 3. 27. Zitzler-Deb-Thiele (no. 3) Fonksiyonu ve ilgili kabuller.

Zitzler-Deb-Thiele (no. 3) Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x) = x_1$ $g(x) = h(x) * j\big(f(x), h(x)\big)$ $h(x) = 1 + \frac{9}{29} * \sum_{i=2}^{30} x_{,}$ $j\big(f(x), h(x)\big) = 1 - \sqrt{\frac{f(x)}{h(x)}} - \left(\frac{f(x)}{h(x)}\right) * \sin\big(10 * \pi * f(x)\big)$	(3.27.)
Arama Alanı	$0 \le x_i \le 1$ $1 \le i \le 30$	

Zitzler-Deb-Thiele (no. 3) Fonksiyonu'nun ifade edilen kabullere göre, pareto optimumlara dair varsayılan grafiği Resim 3.27.'deki gibidir (Wikipedia, 2017i).



Resim 3. 27. Zitzler-Deb-Thiele (no. 3) Fonksiyonu (Wikipedia, 2017i).

3.3.28. Binh ve Korn fonksiyonu

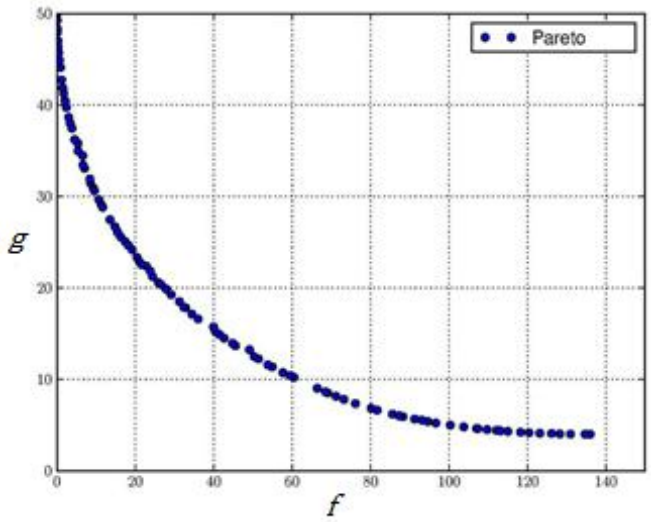
Binh ve Korn Fonksiyonu, iki amaç fonksiyon ve iki kısıttan meydana gelen çok-amaçlı bir test fonksiyonudur.

İlgili fonksiyon ve fonksiyona dair kabuller Çizelge 3.28.'de sunulmuştur (Wikipedia, 2017i).

Çizelge 3. 28. Binh ve Korn Fonksiyonu ve ilgili kabuller.

Binh ve Korn Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x,y) = 4 * x^2 + 4 * y^2$ $g(x,y) = (x-5)^2 + (y-5)^2$ Kısıt $= \begin{cases} h_1(x,y) = (x-5)^2 + y^2 \leq 25 \\ h_2(x,y) = (x-8)^2 + (y+3)^2 \geq 7{,}7 \end{cases}$	(3.28.)
Arama Alanı	$0 \leq x \leq 5$ $0 \leq y \leq 3$	

Binh ve Korn Fonksiyonu'nun söz konusu kabuller kapsamında, pareto optimumlara yönelik varsayılan grafiği Resim 3.28.'deki gibidir (Wikipedia, 2017i).



Resim 3. 28. Binh ve Korn Fonksiyonu (Wikipedia, 2017i).

3.3.29. Chakong ve Haimes fonksiyonu

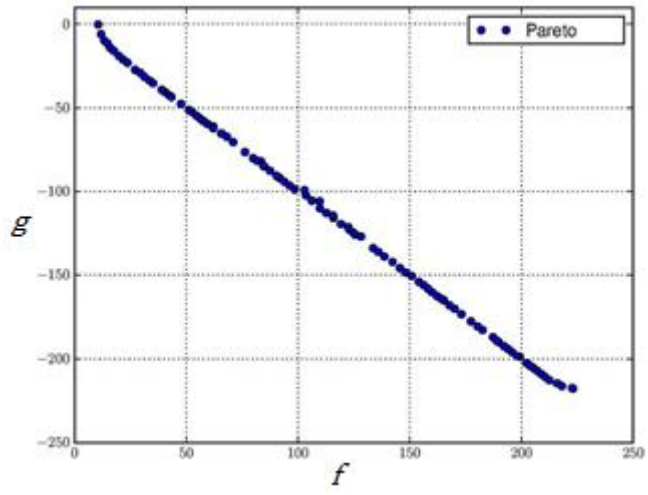
Chakong ve Haimes Fonksiyonu, iki amaç fonksiyon ve iki kısıt kapsamında değerlendirilen bir diğer çok-amaçlı test fonksiyonudur.

İlgili fonksiyon ve fonksiyona yönelik kabuller Çizelge 3.29.'da gösterilmiştir (Wikipedia, 2017i).

Çizelge 3. 29. Chakong ve Haimes Fonksiyonu ve ilgili kabuller.

Chakong ve Haimes Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x,y) = 2 + (x-2)^2 + (y-1)^2$ $g(x,y) = 9 * x - (y-1)^2$ Kısıt $= \begin{cases} h_1(x,y) = x^2 + y^2 \leq 225 \\ h_2(x,y) = x - 3 * y + 10 \leq 0 \end{cases}$	(3.29.)
Arama Alanı	$-20 \leq x, y \leq 20$	

Chakong ve Haimes Fonksiyonu'nun ifade edilen kabuller bağlamında, pareto optimumlara yönelik varsayılan grafiği Resim 3.29.'daki gibidir (Wikipedia, 2017i).



Resim 3. 29. Chakong ve Haimes Fonksiyonu (Wikipedia, 2017i).

3.3.30. Osyczka ve Kundu fonksiyonu

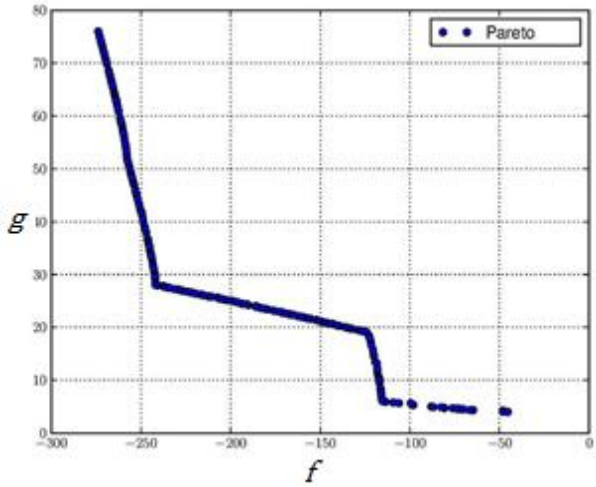
Osyczka ve Kundu Fonksiyonu, temelde iki fonksiyon ve altı farklı kısıta sahip çok-amaçlı test fonksiyonudur.

İlgili fonksiyona dair kabuller Çizelge 3.30.'da verilmiştir (Wikipedia, 2017i).

Çizelge 3. 30. Osyczka ve Kundu Fonksiyonu ve ilgili kabuller.

Osyczka ve Kundu Fonksiyonu	Minimize (bütün fonksiyonlar): $f(x) = -25 * (x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2$ $g(x) = \sum_{i=1}^{6} x_i^2$ Kısıt $= \begin{cases} h_1(x) = x_1 + x_2 - 2 \geq 0 \\ h_2(x) = 6 - x_1 - x_2 \geq 0 \\ h_3(x) = 2 - x_2 + x_1 \geq 0 \\ h_4(x) = 2 - x_1 + 3 * x_2 \geq 0 \\ h_5(x) = 4 - (x_3 - 3)^2 - x_4 \geq 0 \\ h_6(x) = (x_5 - 3)^2 + x_6 - 4 \geq 0 \end{cases}$ (3.30.)
Arama Alanı	$0 \leq x_1, x_2, x_6 \leq 10$ $1 \leq x_3, x_5 \leq 5$ $0 \leq x_4 \leq 6$

Osyczka ve Kundu Fonksiyonu'nun ilgili kabullere göre, pareto optimumlara yönelik varsayılan grafiği Resim 3.30.'daki gibidir (Wikipedia, 2017i).



Resim 3. 30. Osyczka ve Kundu Fonksiyonu (Wikipedia, 2017i).

SZ tekniklerinin değerlendirilmesi aşamasında, açıklaması yapılan test fonksiyonlarının yanında, tabi ki gerçek yaşam tabanlı problemler – uygulamalar da dikkate alınmaktadır. Zaten test fonksiyonlarında görülen başarımlar, SZ tekniklerinin gerçek yaşamdaki problem koşullarına uygulanabilirliğini gösteren ölçütler olarak da değerlendirilebilmektedir. Bu ölçütlerin sonrasındaki süreç, SZ tekniklerinin gerçek yaşamdaki problemlerde, sahaya sürülmesi ile ilişkili olmaktadır. Çalışmada ortaya konulan GOA ve BiGOA tekniklerinin ilgili test fonksiyonları değerlendirmelerini takiben, literatürden seçilmiş çeşitli gerçek yaşam tabanlı çok-disiplinli problemlere ve uygulamalara yönelik uygulama tabanlı değerlendirmeler de gerçekleştirilmiştir.

3.4. Gerçek Yaşam Tabanlı Problemler ve Uygulamalar

Ortaya konulan YZ tabanlı optimizasyon tekniklerinin temel niteliklerine uygun olarak değerlendirme kriterleri bağlamında çalışmaya dahil edilen gerçek yaşam tabanlı uygulamaları da iki kategori altında sunabiliriz. Burada dikkat edilmesi gereken diğer önemli bir husus, –ifade edildiği üzere–, problem ve uygulamaların bir bütün olarak ele alındığında çok-disiplinli bir çerçeveye ışık tutuyor olmasıdır.

3.4.1. Sürekli optimizasyon odaklı problemler ve uygulamalar

Sürekli optimizasyon odaklı olmak üzere, gerçek yaşam tabanlı uygulamalar açısından üç farklı problem dikkate alınmıştır. Çalışmada geliştirilen GOA ve BiGOA'nın diğer algoritmalarla – tekniklerle karşılaştırılması önceki sayfalarda açıklanan test fonksiyonlarıyla yoğun bir şekilde yapılacağından dolayı, buradaki gerçek yaşam tabanlı problemlerde GOA ve BiGOA'nın, sadece belirli algoritmalarla ya da tespit edilebilen optimum çözümlere karşı performansı değerlendirilmiş olacaktır. Söz konusu problemleri – uygulamaları kısaca şu şekilde açıklayabiliriz:

- **Portföy Seçimi:** Xu ve arkadaşları (2007) tarafından çözümlenen 'Portföy Seçimi' problemi, sürekli optimizasyon ile alakalı olarak ele alınan ilk gerçek yaşam tabanlı problem – uygulama olmuştur. Problem kısaca literatürde dikkate alınan, standard Markowitz modeline alternatif olarak; işlem maliyeti ve miktar sabitleri gibi çeşitli karmaşık unsurları da dikkate alan ve geleneksel tekniklerle çözümlenemeyen bir model sunmaktadır (Xu ve ark.,

2007). Optimizasyon bağlamında ele alındığında, *dön:* beklenen varlık dönüş değerlerini, *x:* incelenen varlığın portföydeki oranı gibi değişkenler ve tespit edilen kısıtlarla birlikte riskin minimizasyonu üzerine kurulan problemin formülizasyonu Çizelge 3.31.'de gösterilmiştir (Xu ve ark., 2007).

Çizelge 3. 31. Portföy seçimi optimizasyon problemi (Xu ve ark., 2007).

Portföy Seçimi	$f(x) = (1 - iş_g)^2 * x' * d * x$ Kısıt $(1 - iş_g) * \sum_{i=1}^{n} dön_i * x_i - iş_0 * \sum_{i=1}^{n} \lvert x_i - x_i^0 \rvert \geq dön_0$ $e' * x = 1$ $x \geq 0$	(3.31.)
Arama Alanı	$l_i \leq x_i \leq u_i,$ $1 \leq i \leq n$	

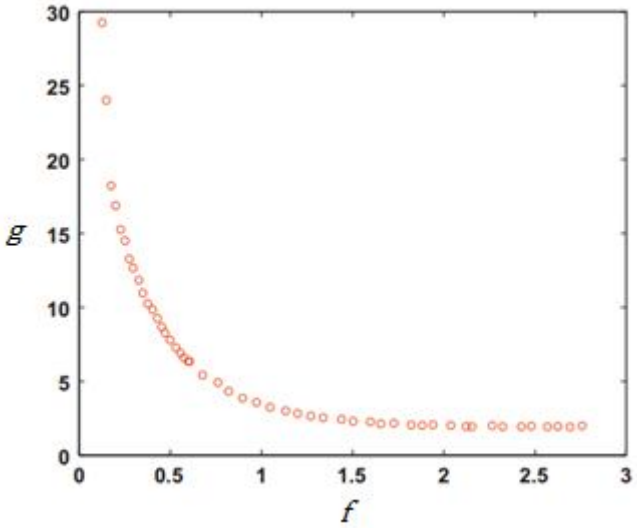
- **Disk Freni Tasarımı:** Literatürde çok-amaçlı kısıtlı optimizasyon için kullanılan (Gong ve ark., 2009; Osyczka ve Kundu, 1995; Ray ve Liew, 2002; Yang ve Deb, 2013) ve gerçek yaşam tabanlı sürekli optimizasyon problemlerinden olan 'Disk Freni Tasarımı', optimum tasarım değişkenlerinin tespit edilerek, toplam kütlenin ve ayrıca frenleme zamanının minimize edilmeye çalışıldığı bir problem olarak bilinmektedir. Tasarım değişkenleri *r*: disk iç yarıçapı, *R*: disk dış yarıçapı, *F*: çekiş kuvveti, ve *S*: sürtünme yüzeyi sayısı olmak üzere ve basınç, sıcaklık, fren uzunluğu, tork gibi sabitler altında ilgili problemin genel formülizasyonu Çizelge 3.32.'de verilmiştir (Yang ve Deb, 2013).

Çizelge 3. 32. Çok-amaçlı kısıtlı Disk Freni Tasarımı optimizasyon problemi (Yang ve Deb, 2013).

Disk Freni Tasarımı	$f(x) = 4{,}9 * 10^{-5} * (R^2 - r^2) * (s - 1)$ $g(x) = \frac{9{,}82 * 10^6 * (R^2 - r^2)}{F * s * (R^3 - r^3)}$ Kısıt $= \begin{cases} h_1(x) = 20 - (R - r) \leq 0, \\ h_2(x) = 2{,}5 * (s + 1) - 30 \leq 0, \\ h_3(x) = \frac{F}{3{,}14 * (R^2 - r^2)} - 0{,}4 \leq 0, \\ h_4(x) = \frac{2{,}22 * 10^{-3} * F * (R^3 - r^3)}{(R^2 - r^2)^2} - 1 \leq 0, \\ h_5(x) = 900 - \frac{0{,}0266 * F * s * (R^3 - r^3)}{(R^2 - r^2)} \leq 0 \end{cases}$	(3.32.)

Arama Alanı	$55 \le r \le 80,$ $75 \le R \le 110,$ $1000 \le F \le 3000,$ $2 \le s \le 20$

Söz konusu çok-amaçlı optimizasyon probleminin varsayılan pareto optimumların grafiği ise Resim 3.31'de gösterilmiştir (Yang ve Deb, 2013).



Resim 3. 31. Disk Freni Tasarım problemi pareto optimumlar (Yang ve Deb, 2013).

- **Görüntü Yeniden Oluşturma:** Bu problemde, geliştirilen sürekli optimizasyon algoritmalarının – tekniklerinin, kendilerine verilen bir görüntüyü, parçacık mantığında yeniden oluşturması amaçlanmıştır. İngilizce 'image reconstruction' olarak ifade edebileceğimiz bu problem de iki farklı şekilde çözüm sürecine dâhil edilmiştir.

  'Birinci görüntü yeniden oluşturma problemi', orijinal, sabit bir görüntüyü temel alarak, oluşturulacak görüntü ortamına rastgele dağıttığı parçacıkların, orijinal görüntüde karşılık geldiği piksel ile benzerlik hatalarına dayanan ve *Denklem 3.33*'te verilen Görüntü Hatası (GH_1) değerini minimize etmeye çalışmaktadır. Herhangi bir paçacığın temsil ettiği piksel, konum, büyüklük ve renk açısından orijinal konumdaki piksele ne kadar yakınsa, aralarındaki hata sıfıra o kadar yakın olacağından dolayı, bütün pikseller kapsamında hesaplanan GH değeri de mümkün olduğunca düşürülmeye (minimizasyon) çalışılmaktadır.

$$GH_1 = \sum_{i=1}^{n}(orijinal_piksel_kbr_i - oluşturulan_piksel_kbr_i)^2 \qquad (3.33.)$$

'İkinci görüntü yeniden oluşturma problemi', ilk problemden farklı olarak, sabit görüntülerin yanında video (hareketli) görüntülerin anlık yeniden oluşturulması problemi – uygulamasına da dayanmaktadır. Çözülebilirliği kolaylaştırmak adına, orijinal görüntüde öncelikle kenar (edge) tespitleri yapılarak (görüntü işleme – image processing), tespit edilen kenarların optimizasyon algoritmaları parçacıkları tarafından tutarlı bir şekilde belirlenmesi amaçlanmaktadır. *Denklem 3.33*'te verilen GH_1 hesaplamasına benzer yapıdaki hata hesaplaması (GH_2), burada sadece piksellerin konum hatalarının değerlendirmeye alınmasını sağlamaktadır (*Denklem 3.34.)*.

$$GH_2 = \sum_{i=1}^{n}(orijinal_piksel_k_i - oluşturulan_piksel_k_i)^2 \qquad (3.34.)$$

Collingridge (2017), Johansson (2008), ve Marquer (2016) tarafından yapılan uygulamalar, Görüntü Yeniden Oluşturma problemlerine esin kaynağı olmuştur. Bu metin kapsamında sadece geliştirilen YZ tabanlı algoritmalarının çalıştırılmasıyla elde edilen görseller üzerinden bulgulara yer verilmiş ve ilgili yorumlamalar yapılmıştır.

3.5. Hibrit Kullanıma Dair Değerlendirmeler

İlgili literatürü incelediğimizde, birden fazla farklı YZ yöntem ve tekniklerinin bir araya getirilerek hibrit sistemler oluşturulması ve bu sistemlerin de problem çözümlemelerinde kullanılması yaklaşımlarının sıklıkla izlendiğini görebiliriz. Bu çalışmalar, problemlerin alternatif yollarla ve başarımı daha yüksek boyutlarda çözülmesini amaçlıyor olsa da, arkaplanda yöntem ve tekniklerin hibrit kullanım yeterliklerini de ortaya koymaktadır. Alan ile ilgili günümüzde ortaya çıkan gelişmeler neticesindeki düşüncelerimizden birisi de, gelecekteki zeki sistemlerin sıklıkla hibrit yapıda kullanılacağıdır. Bu nedenle, tez çalışmasında geliştirilen algoritmaların – tekniklerin değerlendirilmesi aşamasında işe koşulan yaklaşımlardan birisi de ilgili algoritmaların farklı tekniklerle adaptasyonunun nasıl sonuçlandığını görmek olmuştur.

Geliştirilen algoritmalar birçok farklı teknikle hibrit yapıda kullanılmış ve yeter düzeyde sonuçlar alındığı gözlemlenmiştir. Ancak metnin karmaşık bir organizasyona kavuşmaması adına, burada sadece Yapay Sinir Ağları [YSA – Artificial Neural

Networks (ANN)], Adaptif Sinirsel-Bulanık Çıkarım Sistemi [ASBÇS – Adaptive Neuro-Fuzzy Inference System (ANFIS)] ve Destek Vektör Makineleri [DVM – Support Vector Machines (SVM)] ile yapılan hibrit sistemlere dair bulgular detaylı bir şekilde verilmiştir. YSA, ASBÇS, ve DVM'nin seçilmesinin nedeni ise bu YZ (Makine Öğrenmesi) tekniklerinin diğer tekniklere göre daha yaygın ve çok-disiplinli bir kapsamda etkin kullanılıyor olmasıdır. Yeri gelmişken, okuyucuların ilgili teknikler hakkında kısaca bilgilendirilmesi de önemlidir:

3.5.1. Yapay sinir ağları ve hibrit kullanım

YSA, çeşitli paralel dağıtık işlem elemanlarının bir araya gelmesi suretiyle oluşturulan ve insan beyninin işleyişinden esinlenen bir YZ tekniğidir. YSA, eldeki verilerle ve ortamdaki verilerle eğitilerek, farklı problemlerde farklı durumlara cevaplar – çözümler üretebilen, etkili bir teknik olarak bilinmektedir (Elmas, 2003b; Hassoun, 1995; Nabiyev, 2012; Öztemel, 2003). Literatürde farklı çalışma şekillerine ve yapılarına sahip YSA modelleri var olmakla birlikte (Hassoun, 1995; Krenker ve ark., 2011; Öztemel, 2003), tipik bir YSA modeli en iyi çok-katmanlı, ileri beslemeli mimari ile açıklanabilmektedir (Hornik ve ark., 1989; Öztemel, 2003; Yegnanarayana, 2009). Tipik bir çok-katmanlı, ileri beslemeli mimari modelinde, yapay sinir hücreleri olarak adlandırılan ve kendilerine giriş olarak gelen verileri toplayıp, özel aktivasyon fonksiyonlarından geçiren ve elde edilen çıktıları da kendilerinden sonra başka katmanlar altında bağlandıkları yapay sinir hücrelerine ileten ve nihayetinde giriş verilerine karşılık çıkış katmanında sonuçlar – dönütler üreten bir sistem söz konusudur (Anderson ve McNeill, 1992; Krenker ve ark., 2011; Öztemel, 2003; Uğur ve Kınacı, 2010). Bu noktada, her hücre bağlantısı birer ağırlık değeri ile temsil edilmekte ve modelin kendisine giriş olarak verilen verilere karşılık çıkışta istenilen çözümleri üretmesi için de eğitilmesi gerekmektedir. Çünkü eğitilen bir YSA, artık kendisine daha önce gösterilmemiş olan giriş verilerine karşılık doğru çıktılar – sonuçlar üretebilmekte ve böylece hedef problem kapsamında kullanılabilmektedir. Eğitim süreci için geleneksel olarak Back-Propagation (Geri Yayılım) Algoritması (Hagan ve Menhaj, 1994; Öztemel, 2003; Phansalkar ve Sastry, 1994; Yegnanarayana, 2009) veya Marquardt Algoritması (Hagan ve Menhaj, 1994) gibi algoritmalar kullanılıyor olsa da, günümüzde farklı algoritmalar ya da YZ tabanlı optimizasyon teknikleri de daha hızlı ve pratik sonuçlar vermeleri sebebiyle kullanılmaktadır (Gudise ve Venayagamoorthy,

2003; Ilonen ve ark., 2003; Karaboğa ve ark., 2007; Montana ve Davis, 1989; Valian ve ark., 2011a).

3.5.2. Adaptif sinirsel-bulanık çıkarım sistemi ve hibrit kullanım

İlk olarak Jang (1992; 1993) tarafından literatüre önerilen Adaptif Sinirsel-Bulanık Çıkarım Sistemi [ASBÇS – Adaptive Neuro-Fuzzy Inference System (ANFIS)], YSA ile Bulanık Mantık [BM – Fuzzy Logic (FL)] tekniklerinin hibrit yapıda kullanıldığı bir YZ tekniği formudur. Keskin Mantık'tan (Crisp Logic) farklı olarak, insan mantığına benzer bir şekilde, Zadeh'in (1965; 1978) Bulanık Küme Teorisi'ne sadık kalarak, bir durumun ara değerleri üzerinden yorumlamalara (ve çözümlemelere) ulaşılmasını sağlayan BM (Baykal ve Beyan, 2004; Klir ve Yuan, 1995), YSA'nın eğitim sürecine 'EĞER…İSE' kuralları aracılığıyla, Takagi-Sugeno modeli aracılığıyle 'dilsel' bir yaklaşım getirmektedir (Köse ve Arslan, 2016b; Yun ve ark., 2008) [BM ve ASBÇS ile ilgili daha detaylı bilgi için: (Güler ve Übeyli, 2005; Jang ve ark., 1997; Klir ve Yuan, 1995; Ross, 2009; Yager ve Zadeh, 2012; Zadeh, 1994)].

Tıpkı varsayılan YSA modellerinde olduğu gibi, ASBÇS modellerinin eğitimi süreçlerinde de alternatif algoritmalar – teknikler kullanılabilmektedir. Bu bağlamda, ASBÇS modellerinin eğitilmesinde 'öncül (premise)' ve 'izleyen (consequent)' adlı iki parametrenin optimize edilmesi yaklaşımı izlenmektedir (Güler ve Übeyli, 2005; Jang, 1992; 1993). Anlaşılacağı üzere, bir önceki alt-başlıkta da ifade edilen ve YSA modelleri için elde edilen avantajlar kapsamında, ASBÇS modellerinin eğitiminde de alternatif algoritmalar – teknikler kullanılabilmektedir.

3.5.3. Destek vektör makineleri ve hibrit kullanım

Genel anlamda DVM, regrasyon ile sınıflandırma amaçlı işe koşulan bir tekniktir (Cortes ve Vapnik, 1995; Vapnik ve ark., 1997). Literatüre bakıldığında bu tekniğin hem bir YZ – Makine Öğrenmesi tekniği olduğunu, hem de bir Veri Madenciliği tekniği olduğunu ifade etmek mümkündür (Alpaydın, 2011; Özkan, 2016). DVM'deki temel mantık, eldeki veri gruplarını birbirinden ayrıracak bir biçimde, en öndeki elemanlar arasındaki mesafeyi maksimize eden ve böylece grupları sınıflandıran hiper-düzlemi (hyper-plane) tespit etmektir (Köse ve ark., 2016). Burada söz konusu

süreci gerçekleştirebilmek için DVM modelinin danışmanlı öğrenme yaklaşımı içerisinde, destek vektörü adı verilen verileri içeren eğitim kümeleri yardımıyla eğitilmesi gerekmektedir (Javed ve ark., 2007; Santhanam ve Padmavathi, 2015; Tamura ve Tanno, 2009). İşte bu özelliğinden dolayı DVM bir Makine Öğrenmesi tekniği olarak kabul görmekte, verilerin analizi ve istendik doğrultuda ayrıştırılması anlamında ise Veri Madenciliği içerisine girmektedir.

Genel anlamda çeşitli matematiksel fonksiyonların kullanımı ve hesaplamalar (özellikle optimizasyon odaklı) neticesinde sonuca ulaşan DVM modelleri sınıflandırmanın ne şekilde yapılacağına göre de 'Doğrusal (Linear)' ve 'Doğrusal Olmayan (Non-Linear)' DVM'ler olmak üzere iki kategori altında incelenebilmektedir. Doğrusal bir DVM'de sınıflandırmayı yapan düzlem doğrusal bir yapıda gözlemlenirken, Doğrusal Olmayan DVM'de ise daha yüksek boyutlu uzayda anlamlandırılabilen düzlemler ortaya çıkmaktadır. Doğrusal olmayan düzlemlerin elde edilmesi noktasında ise çekirdek (kernel) fonksiyonu adı verilen ve problem çözümünü kolaylaştıran fonksiyonlar işe koşulmaktadır. DVM modelleri için literatürde yaygın bir şekilde kullanılan çekirdek fonksiyonları Linear – Lineer, Gaussian (RBF), ve Polynomial – Polinomal fonksiyonlar olarak bilinmektedir (Cortes ve Vapnik, 1995; Köse ve ark., 2016; Tamura ve Tanno, 2009). Çözüm süreçleri esnasında bu fonksiyonlara ait çeşitli parametrelerin optimize edilmesi problemi de, hibrit modellemenin kapılarını özellikle SZ tekniklerine açmış ve literatürün bu araştırma konusu üzerinden ilerlemesine de imkân vermiştir.

Bu tez çalışma içerisindeki değerlendirme yaklaşımı kapsamında da, YSA, ve ASBÇS modellerinin, geliştirilen YZ tabanlı (sürekli) optimizasyon algoritmaları ile eğitilmesi yoluna gidilmiş, DVM modelleri için ise çekirdek fonksiyonu parametresi optimizasyonu gerçekleştirilmiştir. Böylelikle, özelde hedef alınan problemlerin çözülmesi, genelde ise modellerin etkin uygulanabilir yapıya kavuşturulması açısından başarımlara odaklanılmıştır. Bu noktada kısaca dikkat çekilen diğer bir husus da, ilgili YSA, ASBÇS, ve DVM gibi modellerin kullanıldığı araştırmalar – uygulamalar olmuştur. Söz konusu araştırmalar – uygulamalar, çok-disiplinli uygulanabilirliği gözler önüne sermektedir.

Geliştirilen tekniklerin değerlendirilmesi sürecinde, buraya kadar açıklanan unsurların uygulanması aşamasında, benzeri algoritmalarla – tekniklerle yapılacak karşılaştırmalar da önemlidir.

3.6. Karşılaştırmaya Tabi Algoritmalar – Teknikler

Çalışma kapsamında tasarlanıp geliştirilen algoritmaların, buraya kadar ifade edilen unsurlar kapsamında değerlendirilmesi aşamalarına farklı algoritmalar – teknikler de dâhil edilmiştir. Bu bağlamda, metnin ikinci bölümü, alt-başlık 2.6.'da detayları verilen 9 farklı SZ tekniği ve bu tekniklere ait ikişer güncel varyasyon, geliştirilen algoritmalar – teknikler ile farklı kriterler açısından karşılaştırılmıştır. İlgili algoritmaların – tekniklerin birinci varyasyonları tek-amaçlı optimizasyon değerlendirmelerinde kullanılırken, ikinci varyasyonlar ise çok-amaçlı optimizasyon değerlendirme süreçlerinde kullanılmıştır.

Karşılaştırmaya tabi algoritmalar – teknikler, kategorileri, yine karşılaştırmaya katılmış varyasyonları (referansları ile birlikte) ve varyasyonlarının pozitif yönlerine dair özet bilgiler, Çizelge 3.33. altında verilmiştir.

Çizelge 3. 33. Karşılaştırmaya tabi algoritmalar – teknikler.

Algoritma – Teknik[a]	**Varyasyon – 1 (Tek-Amaçlı)**	**Varyasyon – 1 Pozitif Yönü / Yönleri**	**Varyasyon – 2 (Çok-Amaçlı)**	**Varyasyon – 2 Pozitif Yönü / Yönleri**
BYAO	(Panda ve ark., 2017)	Geliştirilmiş yakınsama ve güncellik	(Niu ve ark., 2013a)	Çok-amaçlı optimizasyon için eklenmiş pareto dominant mekanizması
ÇTO	(Valenzuela ve ark., 2017)	Bulanık Mantık ile etkin bir yaklaşım uygulanmış olması ve güncellik	(Yang ve ark., 2014)	Çok-amaçlı optimizasyon için geliştirilmiş yaklaşım ve etkin yakınsama
DE	(Shih ve ark., 2017)	Gelişmiş çözüm, hız ve yakınsama, güncellik	(Xue ve ark., 2003)	Pareto optimumları bulmada etkinlik ve verimlilik
ERA	(Talatahari ve ark., 2012a)	Kaotik haritalar ile geliştirilmiş adımlama – arama	(Nourmohammadi ve ark., 2013)	Gelişmiş çözüm yeteneği. Taguchi tasarımı ile parametre ayarlama
GA	(Rajarathinam ve ark., 2017)	Arama sınırlarına müdahale ile gelişen çözüm yaklaşımı ve güncellik	(Mashwani ve ark., 2017)	Çok-amaçlı optimizasyon için geliştirilmiş mekanizmalar ve güncellik

Algoritma – Teknik[a]	Varyasyon – 1 (Tek-Amaçlı)	Varyasyon – 1 Pozitif Yönü / Yönleri	Varyasyon – 2 (Çok-Amaçlı)	Varyasyon – 2 Pozitif Yönü / Yönleri
GKA	(Dhal ve ark., 2017)	Kaotik ve popülasyon çeşitliliğine dayalı geliştirmeler	(Yang ve Deb, 2013)	Çok-amaçlı optimizasyon için elde edilmiş iyi sonuçlar
KSA	(Peng ve Lu, 2015)	Baldwinian ve Ortogonal öğrenme ile artırılmış yetenek	(Rao ve Vaisakh, 2013)	Çok-amaçlı optimizasyon için adaptif yaklaşım
PSO	(Rehman ve ark., 2017)	Quantum mekaniği ile geliştirilmiş arama ve çözümleme	(Tripathi ve ark., 2007)	Modifiye edilmiş parametrelerle daha iyi arama
YAK	(Gao ve Liu, 2011)	Artırılmış çözüm yeteneği ve hız	(Akbari ve ark., 2012)	Harici arşiv adı verilen yaklaşım ile başarılı çok-amaçlı optimizasyon

[a] Tekniklerin detayı için *Bkz.* ikinci bölüm, alt-başlık 2.6.

Çalışmaya dahil edilen teknikler genel anlamda literatürde adından sıklıkla bahsedilen ya da güncelliği bakımından literatüre yeni kazandırılmış zeki optimizasyon teknikleri olarak çalışmaya konu olmuştur. Bu tez çalışmasında geliştirilen algoritmaların yeterliklerinin değerlendirilmesi aşamasında ifade edilen teknikler yeterli olarak kabul edilmiştir. Çünkü SZ ile ilgili literatür, bu satırlar kaleme alınırken bile yeni tekniklerin geliştirilmesine sahne olan, dinamik bir yapıya sahiptir. Bu yüzden, çalışma içerisinde geliştirilen tekniklerin alternatif teknikler ya da literatüre yeni girecek teknikler bağlamındaki başarımları da gelecek çalışmalara konu olabilecektir.

3.7. Değerlendirme Kuralları ve Yöntemleri

İfade edilen değerlendirme aşamalarında, sonuçların tutarlılığına emin olmak adına çeşitli kurallara uymak da oldukça önemlidir. Ayrıca, çalışmayla birlikte geliştirilen tekniklerin – algoritmaların başarımlarını ve literatürdeki yerlerini doğru bir şekilde tespit edebilmemiz için de, objektif yapıda, farklı perspektiflerden değerlendirme tekniklerini uygulamamız gerekmektedir.

Öncelikli olarak, değerlendirme süreçlerinde uyulan başlıca kuralları kısaca şöyle sıralayabiliriz:

- Değerlendirme işlemlerinde aynı konfigürasyondaki bilgisayar sistemi kullanılmış ve yine test süreçlerinin tümünde objektif sayısal bulgular elde

edilmesi adına sistemin harcamakta olduğu kritik kaynakların (Örneğin; kullanılabilir ana bellek – RAM miktarı) sürekli aynı düzeyde olması sağlanmıştır.

- Kullanılan sürekli optimizasyon algoritmaların – tekniklerin tümü MATLAB ortamında, standart ve ilgili varyasyon yapılarına uygun bir şekilde kodlanmıştır.

- Değerlendirme süreçlerine dahil edilen sürekli optimizasyon tabanlı algoritmaların test fonksiyonları süreçleri ve diğer uygulamalarda (hibrit kullanım ve görüntü yeniden oluşturma çalışmaları hariç) süreçlerinde 50'şer çözüm elemanı – parçacık ile çalıştırılması sağlanmış, yine bütün algoritmaların parametreleri, literatürde standart olarak kabul edilen değerler çerçevesinde seçilmiştir. Hibrit kullanım uygulamalarındaki parçacık sayıları eğitilen hedef sistemlerin ihtiyaçlarına göre belirlenmiş, görüntü yeniden oluşturma çalışmalarında ise problem çözümlerine her ne kadar 50'şer parçacık ile başlanmışsa da, hesaplanan hata değerleri ve piksel durumlarına göre var olan parçacıkların ek parçacıklarla desteklenmesi şeklinde bir eklenti, ilgili algoritmalara (GOA ve BiGOA) eklenmiştir.

- Tutarlılığın ölçülmesi adına, optimizasyon algoritmalarının aynı problem üzerinde defalarca çalıştırılması ve bu çalıştırmalar netices elde edilen sonuçların bütünü üzerinden yargılara ulaşılması yolu tercih edilmiştir. Buna göre, her bir algoritmanın test fonksiyonları için 50'şer kez, diğer uygulamalarda (gerçek yaşam tabanlı problemler; görüntü yeniden oluşturma çalışmaları hariç) ise 30'ar kez çalıştırılması yoluna gidilmiştir.

- Başta geliştirilen algoritmalar olmak üzere, değerlendirme süreçlerine dâhil olan algoritmaların başarım durumlarını anlamak adına şu kriterler çevresinde bulgular elde edilmiştir:

    - Bulunan optimum değerlerin isabetliliği,

    - Farklı test – problemler kapsamında genel istikrar.

    - Hata durumları ve yakınsama performansları,

- Sürekli optimizasyon test fonksiyonlarının çözümlenmesi aşamasında algoritmaların durma kriteri olarak belirli iterasyon sayılarının kullanılması tercih edilmiştir. Buna göre, tek-amaçlı (kısıtlı-kısıtsız) iki-boyutlu fonksiyonlarda her çalıştırmada 1.000 iterasyon seçilmiştir. Tek-amaçlı, (kısıtlı-kısıtsız) çok-boyutlu ve IEEE CEC 2015 test fonksiyonları çözümlemelerinde ise, tercih edilen 30 boyut kapsamında 10.000 iterasyon gerçekleştirilmiştir. Çok-amaçlı fonksiyonlarda ve gerçek yaşam tabanlı problemlerin çözümlenmesi süreçlerinde ise iterasyonun 20 ve altı boyutlar için 20.000, üstü boyutlar için ise 30.000 olarak belirlenmiştir.

İfade edilen kurallar ve kriterler çerçevesinde, ihtiyaç duyulan bulguların elde edilmesi amacıyla şu yöntemler izlenmiştir:

- Bulguları geniş çerçevede kolayca yorumlayabilmek adına yeri geldiğinde grafiklerden de faydalanılmıştır.

- Test fonksiyonlarının çözülmesinde, algoritmaların kendi içindeki istikrarlarını anlamak için, gerçekleştirilen 50 çalıştırma aşamasının elde edilen en iyi sonuçları, ortalama değeri ve standart sapma gibi istatistiksel bulgular kullanılmıştır (10^{-15}'ten küçük değerler 0 olarak kabul edilmiştir. Ancak 10^{-15}'ten küçük değerlerin büyüklüğü de algoritmalar arası sıralamayı belirlemede dikkat edilen bir kriter olmuştur).

- Her farklı test fonksiyonu içerisinde ilgili algoritmalar için elde edilen sonuçların istatistiksel anlamda kanıtlanması ve çalışmada geliştirilen algoritmaların başarımı hakkında ayakları yere basan yorumlara ulaşabilmek için ikili kıyaslamalar yapılmıştır. Bu amaçla, geliştirilen her bir algoritma değerlendirmelere konu olan 11 farklı algoritma – teknik ile sırayla karşılaştırılmış ve parametrik olmayan veriler kapsamında kullanılabilen Wilcoxon İşaretli Sıralama Testi (Wilcoxon Signed-Rank Test) (Woolson, 2008), hangi algoritmanın daha iyi olduğu ve farkın istatiksel bağlamda önemli olup olmadığı (*H0* – null hipotezine göre önemli değil; *H1* – alternatif hipoteze göre önemli) sonuçlarına ulaşmak için gerçekleştirilmiştir. Burada, $\alpha = 0,05$ anlamlılık düzeyi, yani %95 güven aralığında tercih edilmiştir.

- Farklı test fonksiyonları kapsamında algoritmaların genel istikrarını yorumlayabilmek adına, her test – problem için 50 farklı çalıştırma içerisinde bulunan (en iyi) optimum değer ve arzu edilen değer arası farkların genelinden faydalanılarak *Denklem 3.35.*, *Denklem 3.36.*, ve *Denklem 3.37.*'de sırasıyla verilen, Ortalama Hataların Karesi [OHK – Mean Square Error (MSE)], Ortalama Hataların Karesinin Kökü [OHKK – Root Mean Square Error (RMSE)], ve Ortalama Mutlak Hata Yüzdesi [OMHY – Mean Absolute Percentage Error (MAPE)] değerleri hesaplanmıştır. Burada hesaplanan değerler, geliştirilen algoritmaların – tekniklerin genel istikrarı konusunda fikir sahibi olmamızı sağlayan bulgular olarak kabul edilmiştir.

$$OHK = \frac{\sum_{tp=1}^{n}(bulunan_{tp} - arzu_edilen_{tp})^2}{n} \quad (3.35.)$$

$$OHKK = \sqrt{\frac{\sum_{tp=1}^{n}(bulunan_{tp} - arzu_edilen_{tp})^2}{n}} \quad (3.36.)$$

$$OHMY = \frac{\sum_{tp=1}^{n}|(bulunan_{tp} - arzu_edilen_{tp})/arzu_edilen_{tp}|}{n} \quad (3.37.)$$

Denklem 3.35., *Denklem 3.36.*, ve *Denklem 3.37.* kapsamında *tp* dikkate alınan test – problemi, *n* toplam test – problem sayısını, $bulunan_{tp}$ dikkate alınan test – problem için 50 farklı çalıştırma içerisinde bulunan (en iyi) optimum değeri ve $arzu_edilen_{tp}$ ise dikkate alınan test – problem için bulunması beklenen optimum değeri temsil etmektedir.

- Test fonksiyonları kapsamında işe koşulan son değerlendirme kriteri, algoritmaların bu çalışmadaki genel başarı sıralamalarını belirlemek için araştırmacı tarafından düşünülmüş olan, basit bir Genel Performans Sıra Değeri (GPSD) sistemi olmuştur. Bu sistemde her bir algoritma, çözülen her bir test fonksiyonundaki başarı sırasını puan olarak almakta ve toplam GPSD'si hesaplanmaktadır. Anlaşılacağı üzere, daha düşük GPSD'ye sahip algoritma – teknik daha başarılı sayılmaktadır.

- Gerçek yaşam tabanlı problemler – uygulamalarda, ilgili problemlerin başarılı çözülüp çözülmediğini gösteren, bilinen optimum kriterler ya da belirli algoritmalara karşı başarım bulgularından faydalanılmıştır.

- Hibrit sistemlere dair değerlendirmeler kapsamında YSA, ASBÇS, ve DVM modellerine dayalı çalışmalar da, başarımı dikkate çeken algoritmaların sürece dâhil edilmesi ve genelde önceki maddelerde açıklanan ölçütlerin kullanılması yaklaşımına dayanmıştır.

- Kimi zaman bazı uygulamalarda sayısal bulguların yanında, ulaşılması arzu edilen ürünlere yönelik elde edilen çıktılar da tekniklerin başarımı konusunda bizlere bilgi verebilmektedir. Bu bağlamda, görüntü yeniden oluşturma, eğitilen YSA modellerinin çok-disiplinli problem çözümleri, oyun eğitimi gibi problemlerdeki tecrübeler de geliştirilen algoritmaların başarımı hakkında fikir sahibi olunması için kullanılmıştır.

4. YAPAY ZEKÂ TABANLI OPTİMİZASYON İÇİN GELİŞTİRİLEN ALGORİTMALAR

Önceki bölümler altında elde edilmiş olan teorik alt-yapı ile beraber bu çalışma bünyesinde tasarlanıp geliştirilmiş olan algoritmaların anlaşılması daha kolay olmaktadır. Söz konusu teorik alt-yapının kurulması akabinde bu bölüm altında, ilgili algoritmalar: Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Optimizasyon Algoritması (BiGOA) sırasıyla tanıtılmış – açıklanmıştır.

4.1. Girdap Optimizasyon Algoritması (GOA)

Girdap Optimizasyon Algoritması [GOA – Vortex Optimization Algorithm (VOA)], temel olarak doğadaki girdaplardan esinlenilerek geliştirilmiş bir YZ tabanlı optimizasyon algoritmasıdır. Düşünce anlamında ilk olarak 2013 yılı sonlarına doğru ortaya çıkmış, akabinde gerçekleştirilen çalışmalar neticesinde algoritmaya yönelik ilk yayın 2014 yılında yapılmıştır (Köse ve Arslan, 2014). 'Hızlı bir düzeyde zorlaşan gerçek yaşam tabanlı problemlerin literatürdeki alternatif algoritmaların aksine, basit matematiksel yapılarla çözülmesi' amacı ile çıkılan yolda; amaca uygun, basit bir matematiksel yapı ile sunulan ilk format ile çok çeşitli uygulamalar gerçekleştirilmiş, bu uygulamalara eş zamanlı olarak da, algoritma içerisinde çeşitli düzenlemeler yapılmıştır. Algoritmanın son hali, 'sistem-içi optimizasyon' yaklaşımıyla ve hatta 'çok-amaçlı optimizasyon' odaklı eklentileriyle (Köse ve Arslan, 2017b) birlikte girdap dinamiğini ve etkin çözüm sürecine yeter düzeyde kavuşmuştur.

GOA'nın çözüm adımları, doğada ortaya çıkan girdap adlı dinamiklerin, çevresindeki nesneleri kendilerine çekmesi ve bu çekim noktasının çözüm uzayında optimum noktaların temsil edebileceği düşüncesinin ortaya çıkması sonucunda tasarlanmıştır. İlk aşamada, gerçek anlamda girdap dinamiklerinin hareketlerini belirten hâlihazırdaki denklemler üzerinden bir algoritma yapısı inşa edilmeye çalışılsa da, uygulamalarda istendik sonuçlar elde edilememiş olması neticesinde, girdap dinamikleri fikrinden ayrılmadan, özgün ve basit denklemler ile tasarlanmış bir algoritmanın literatüre kazandırılması yoluna gidilmiştir. Burada amaç, çok basit denklemlerle zeki optimizasyonun yapılabileceği düşüncesine yoğunlaşmak olmuştur. Bu çalışmalar

sırasında GOA'nın daha stabil hale gelmesi için 'sistem-içi optimizasyon' adı verilen bir yaklaşım daha eklenmiş ve bu yaklaşımda girdap dinamiklerinden bir denklem de içerilerek hem algoritmanın doğa-esinli temelleri sağlama alınmış, hem de başarımı istendik düzeye taşınmıştır. GOA, SZ teknikleri ile ilgili alternatif durumlardan, 'rol dağılımlı çözüm', ve 'evrimsel süreçli çözüm' durumlarına (*Bkz.* ikinci bölüm, alt-başlık 2.4.2.) dayalı olmak suretiyle, bu tez çalışması sürecinde geliştirilmiş olan ilk algoritmadır. Algoritmanın çözüm adımlarının ortaya konulması aşamasında, mümkün olduğunca kısa, –tıpkı PSO'da olduğu gibi (*Bkz.* ikinci bölüm, alt-başlık 2.6.1.)– basit denklemlerden oluşan, pratik ama etkili bir yapının elde edilmesi amaçlanmıştır.

4.1.1. Girdap optimizasyon algoritması kapsamında çözüm adımları

GOA ile birlikte literatüre kazandırılmış olan YZ tabanlı optimizasyon çözüm adımlarını şöyle ifade edebiliriz (Köse ve Arslan, 2014; 2016b; 2017a; 2017b):

Adım 1 (Kurulum Fazı). *N* adet parçacığı oluştur ve her bir parçacık için başlangıç *Girdap* (*v: vorticity*) değerini belirle. Ayrıca algoritma ile ilgili *Maksimum_Girdap* ve *Minimum_Girdap* (*max_v, min_v*) değerlerini ve *Eleme_Oranı* (*e: elimination_rate*) değerini belirle. Çözülecek probleme yönelik diğer düzenlemeleri de yap.

Adım 2. Parçacıkları çözüm uzayına rastgele dağıt ve her parçacığın konumuna göre amaç fonksiyon değerini (fitness) hesapla. Optimum değeri hesaplayan, o anda en iyi konumdaki parçacığın *Girdap* değerini *Denklem 4.1.*'e göre güncelle. Bu parçacığı 'girdap' statüsüne yükselt ve konumunu en iyi konum olarak kabul et.

Adım 3. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her parçacık için; her amaç fonksiyon boyutu için;)

Adım 3.1. Bütün parçacıkların hesapladığı amaç fonksiyon değerinin (fitness) ortalamasına eşit veya ondan daha iyi (mininizasyon ise altında, maksimizasyon ise üstünde) olan parçacıkları 'girdap' statüsüne yükselt. Diğer parçacıkları 'normal' parçacık statüsünde kabul et.

Adım 3.2. Her parçacığın *Girdap* değerini *Denklem 4.2.*'ye göre güncelle.

Adım 3.3. En iyi durumdaki parçacık dışında kalan ve 'girdap' statüsünde olan parçacıkların *Girdap* değerini *Denklem 4.3.*'e göre güncelle.

Adım 3.4. En iyi durumdaki parçacık dışında kalan bütün parçacıkların konumunu *Denklem 4.4.*'e göre güncelle.

Adım 3.5. Yeni konumlarıyla birlikte bütün parçacıklar için amaç fonksiyon değerini (fitness) hesapla. Optimum değeri hesaplayan, o anda en iyi konumdaki parçacığın (eğer henüz değilse) 'girdap' statüsüne yükselt ve konumunu en iyi konum olarak kabul et.

Adım 3.6. Toplam 'normal' statüdeki parçacık sayısı *Eleme_Oranı* değerine eşit veya bu değerin altında ise, bütün 'normal' statüdeki parçacıkları çözüm uzayından sil ve sayıları kadar yeni parçacığı (algoritmada kabul edilen başlangıç değerlerini vererek) çözüm uzayına rastgele dağıt.

Adım 3.7. Çözüme yakınsama istendik düzeyde değilse sistem-içi optimizasyon uygula.

Adım 4. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

GOA'da algoritma başlangıcında en iyi durumdaki parçacığın *Girdap* değeri için yapılan güncelleme, tipik bir rastgele sayı (0 ile 1 arasında) ile çarpım ve elde edileni ekleme işlemine dayanmaktadır:

$$EnİyixGirdap^{y} = EnİyixGirdap^{g} + (EnİyixGirdap^{g} * rsayı) \qquad (4.1.)$$

Denklem 4.1.'de, y ve g simgeleri, en iyi durumdaki parçacığın geçerli (g) ve yeni (y) durumdaki *Girdap* değerlerini belirtmek için kullanılmaktadır.

Süreç esnasında her bir parçacığın *Girdap* değerini güncellerken, her bir parçacığın geçerli *Girdap* değerleriyle en iyi durumdaki parçacığın *Girdap* değeri, rastgele sayılar ile birlikte hesaplama işlemi içerisine dâhil edilmektedir. Burada en iyi durumdaki parçacığın *Girdap* değerine yakınsama sağlanabilmesi noktasında en iyi *Girdap* değeri ile ilgili parçacığın geçerli *Girdap* değeri oranlanmaktadır:

$$xGirdap_i^{z+1} = xGirdap_i^z + [rsayı_i^z * (Enİyix Girdap / xGirdap_i^z)] \quad (4.2.)$$

Denklem 4.2.'de, *i* parçacık numarasını temsil etmekte, *z* zaman için (iterasyon – döngü anlamında) kullanılmaktadır.

Algoritmada, Adım 3.3.'deki güncelleme işlemi, algoritma başında gerçekleştirilmiş olan (*Denklem 4.1.*) güncellemeye benzerlik taşımakta, ancak bu sefer 'girdap' statüsündeki (en iyi durumdaki parçacık dışındaki) bütün parçacıkları etkilemekte ve *Girdap* değerinin üzerine ekleme yapılmadan yerine getirilmektedir:

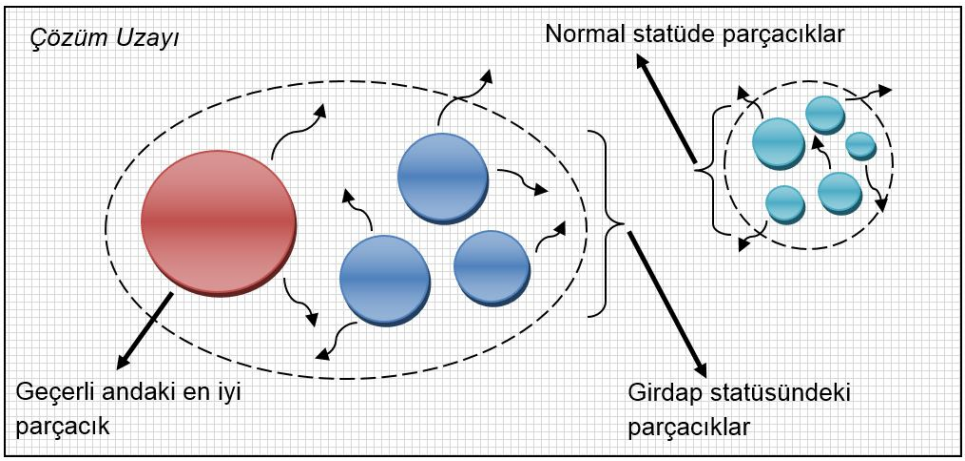
$$gxGirdap_i^{z+1} = gxGirdap_i^z * rsayı_i^z \quad (4.3.)$$

Denklem 4.3.'de, diğer temsillere ek olarak, işlemin (en iyi olan dışında) 'girdap' statüsündeki parçacıkları içerdiğini belirtmek adına *x* simgesi yerine *gx* kullanılmıştır.

Algoritmadaki (en iyi olan dışında) bütün parçacıklar için uygulanan konum güncellenmesi, parçacıkların, optimum değer(ler)in olduğu konum(lar)a yakınsanması üzerine kurulmuştur:

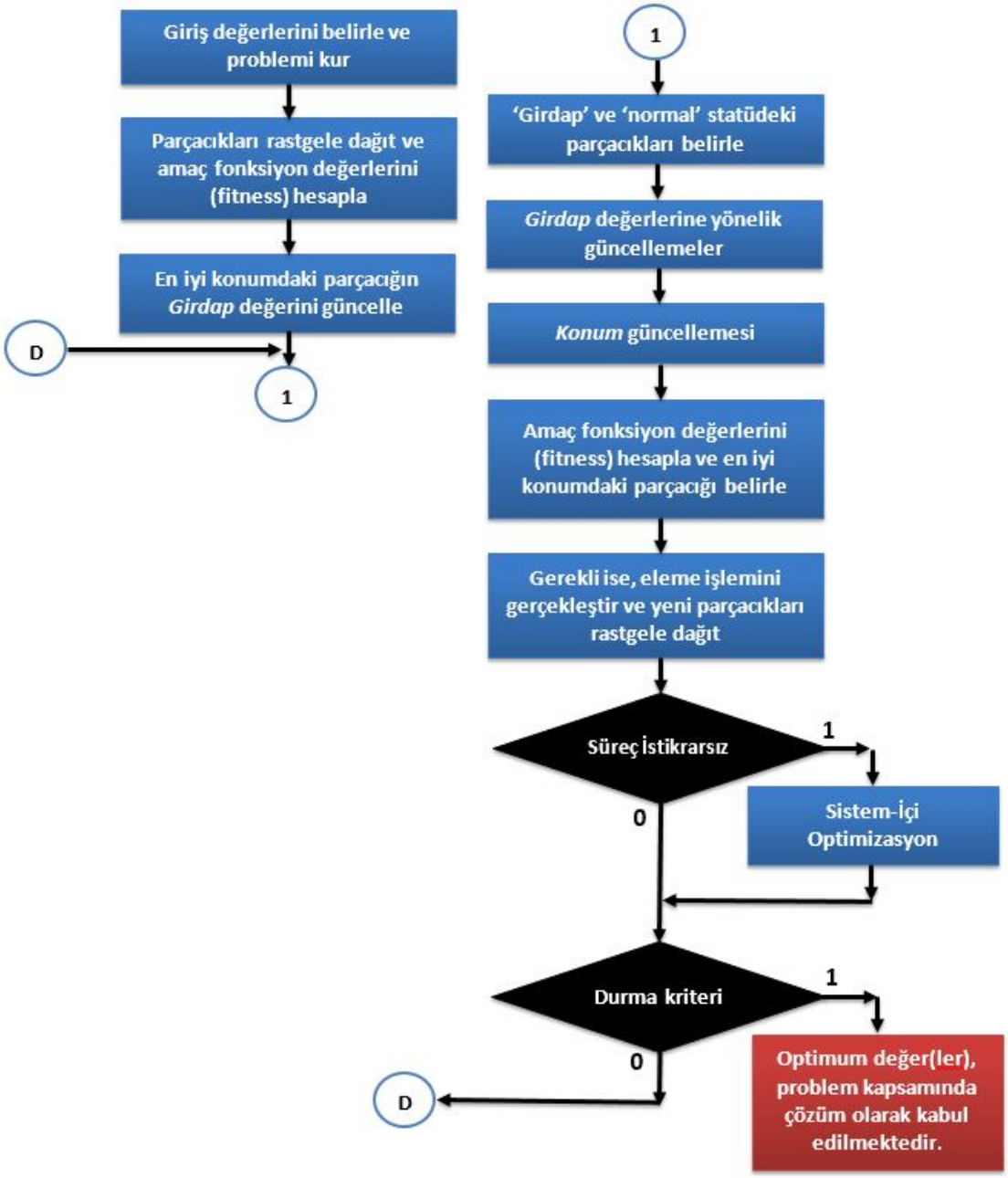
$$xKonum_i^{z+1} = xKonum_i^z + [rsayı_i^z * [xGirdap_i^z * (Enİyix Konum - xKonum_i^z)]] \quad (4.4.)$$

GOA'daki genel yaklaşıma yönelik bir illüstrasyon Şekil 4.1.'de gösterilmiştir.



Şekil 4. 1. GOA – genel işleyiş yaklaşımına yönelik bir illüstrasyon.

Söz konusu çözüm adımları bağlamında GOA'ya yönelik özet akış şeması Şekil 4.2.'de verilmiştir.



Şekil 4. 2. GOA özet akış şeması.

GOA'daki sistem-içi optimizasyon yaklaşımını da kısaca şöyle açıklayabiliriz (Köse ve Arslan, 2017b):

- Sistem-içi optimizasyon genel olarak bir optimizasyon sürecinin, kaotik unsurlar içerse bile belli bir stabilite (burada girdap mantığında; bir akışkandaki duruluk veya girdap sebepli dalgalanma homojenliği stabiliteye karşılık olarak düşünülmüştür) içerisinde gerçekleşeceği düşünülmüştür. Bu nedenle ilgili işlem adımına gelindiği zaman elde edilen fonksiyon değerleri (fitness) ve parçacık konumlarının (değişkenler) o ana kadar ki stabilitesi

ölçülmekte, eğer stabilitasyon sağlanmıyorsa, girdap statüsündeki parçacıkların açısal hızları *Denklem 4.5*. (Siddique, 2015) ile bulunmaktadır.

$$ahxGirdap_i = \frac{2*\pi*xGirdap_i}{\text{ÇözümUzayıMaks}} \quad (4.5.)$$

- *Denklem 4.5.* ile bulunan her açısal hız ile normal statüdeki ve stabilitasyon dışında kalan parçacıklar, girdap statüsündekilere rastgele sayılar yardımıyla, 'rastgele loxodrome yürüyüş' gerçekleştirilerek yaklaştırılmaktadır. Loxodrome yürüyüş, temelleri *Denklem 4.6.*'ya (Köller, 2017)dayanan bir matematiksel yaklaşımdır. *Denklem 4.5.* ve *Denklem 4.6.* öncesi gerçekleştirilen işlemler, sistem-içi optimizasyona gerek olmadığında, basit girdapların yer aldığı akışkanlardaki olaylara benzerken, sistem-içi optimizasyon girdap akışı ve ilgili denklemlere dayalı gerçek dinamikler üzerine kurulu olmaktadır.

$$x = \cos(ahxGirdap_i) * \cos\left(\frac{1}{\tan(ahxGirdap_i * xGirdap_i)}\right) \quad (4.6.)$$

$$y = \sin(ahxGirdap_i) * \cos\left(\frac{1}{\tan(ahxGirdap_i * xGirdap_i)}\right)$$

$$z = -sin\left(\frac{1}{\tan(ahxGirdap_i * xGirdap_i)}\right)$$

4.2. Bilişsel Gelişim Optimizasyon Algoritması (BiGOA)

Bilişsel Gelişim Optimizasyon Algoritması [BiGOA – Cognitive Development Optimization Algorithm (CoDOA)], GOA tekniğini takiben, çalışma süreci içerisinde geliştirilmiş ikinci YZ tabanlı optimizasyon algoritmasıdır. BiGOA'nın tasarlanması sürecinde GOA'da uygulanan basit yapıdaki matematiksel ve mantıksal yaklaşımların başarımı etkili olmuş ve yine literatürde hâlihazırda bulunan tekniklerin karmaşıklığından kaçınarak, –tıpkı GOA'da olduğu gibi– etkin ama uygulanabilirliği basit bir algoritmanın – tekniğin literatüre kazandırılması sağlanmıştır. BiGOA'nın esin kaynağı, İsviçreli psikolog Jean Piaget'in Bilişsel Gelişim Kuramı'dır (Piaget, 1964; Piaget, 2000).

GOA ile karşılaştırıldığında biraz daha hacimli çözüm adımları bütününe sahip olan BiGOA'ya esin kaynağı olan Bilişsel Gelişim Kuramı'nı (The Theory of Cognitive Development) kısaca şöyle açıklayabiliriz (Piaget, 1964; 1973; 2000; Singer ve Revenson, 1997): Bilişsel gelişim genel olarak bireylerin yeni bilgiler edinirken ve kendini bu bağlamda geliştirirken tecrübe ettiği bir süreçtir. Piaget'nin Bilişsel Gelişim Kuramı'na göre, kendimizi bilişsel anlamda geliştirirken; olgunlaşma, sosyal etkileşim ve yeni kavramların öğrenilmesi aşamasında bildiklerimizi dengelemek gibi çeşitli alt-süreçlerden de geçmiş oluruz. Bu süreçlerin bir bileşkesi olarak, öğrenme ve bilişsel gelişim aşamalarını defalarca tecrübe etmiş oluruz.

Çalışma kapsamında tasarlanıp geliştirilen BiGOA'da, söz konusu kuramın ifade ettiği aşamalar esin kaynağı olmak üzere; çözüm uzayına rastgele dağıtılan parçacıkların, çeşitli gelişme süreçlerinden geçerek optimum değer(ler)e ulaşmaya çalıştığı bir yaklaşım vardır. Başlarda temel optimizasyon denemelerine tabi tutulan BiGOA'nın son hali, etkinliği GOA'da görülmüş olan sistem-içi optimizasyon sürecinin bir benzerinin eklenmesiyle elde edilmiş ve hatta yine çok-amaçlı optimizasyon desteği de BiGOA'ya bu format ile birlikte gelmiştir (Köse ve Arslan, 2017b).

BiGOA'nın çözüm adımlarını genel olarak şu fazlar altında gruplandırmak mümkündür (Köse ve Arslan, 2015; 2016a):

- Kurulum Fazı (Initialization Phase),
- Sosyalleşme Fazı (Socialization Phase),
- Olgunlaşma Fazı (Maturation Phase),
- Rasyonelleşme Fazı (Rationalizing Phase),
- Dengeleme Fazı (Balancing Phase).
- Sistem-içi Optimizasyon Fazı (*Eğer gerekli ise*)

İfade edilen fazlar kapsamında çözüm uzayındaki parçacıkların süreç içerisinde 'gelişmesi' ve bu yolla optimum çözüm(ler)e ulaşması amaçlanmıştır. GOA'dan farklı olarak BiGOA'da, parçacıklar arası roller söz konusu olmadığı gibi, çözüm sürecinde dezavantajlı duruma düşen parçacıkların elenmesi gibi bir yaklaşım da yoktur.

4.2.1. Bilişsel gelişim optimizasyon algoritması kapsamında çözüm adımları

BiGOA ile işe koşulan YZ tabanlı optimizasyon çözüm adımlarını şu şekilde ifade etmemiz mümkündür (Köse ve Arslan, 2015; 2016a; Köse ve ark., 2016):

Adım 1 (Kurulum Fazı). *N* adet parçacığı oluştur ve her bir parçacık için başlangıç *Etkileşim_Oranı* (*ir: interactivity rate*) ve *Deneyim* (*ex: experience*) değerlerini belirle. Algoritma ile ilgili olarak da, *Maksimum_Etkileşim_Oranı* ve *Minimum_Etkileşim_Oranı* (*max_ir, min_ir*) değerlerini, *Olgunlaşma_Limiti* (*ml: maturity_limit*) değerini ve *Rasyonallik_Oranı* (*r: rationality_rate*) değerini belirle. Çözülecek probleme yönelik diğer düzenlemeleri de yap.

Adım 2. Parçacıkları çözüm uzayına rastgele dağıt ve her parçacığın konumuna göre amaç fonksiyon değerini (fitness) hesapla. Optimum değeri hesaplayan, o anda en iyi konumdaki parçacığın *Etkileşim_Oranı* değerini *Denklem 4.7.*'ye göre güncelle ve yine aynı parçacığın *Deneyim* değerini 1 artır (*Denklem 4.8.*).

Adım 3. Aşağıdaki adımları, iteratif süreç boyunca (Örneğin; belirli bir iterasyon sayısına ulaşıncaya kadar ya da amaç fonksiyonda istendik bir değere ulaşıncaya kadar) tekrarla:

(Her parçacık için; her amaç fonksiyon boyutu için;)

Adım 3.1 (Sosyalleşme Fazı). Bütün parçacıkların hesapladığı amaç fonksiyon değerinin (fitness) ortalamasına eşit veya ondan daha kötü (mininizasyon ise üstünde, maksimizasyon ise altında) olan parçacıkların *Deneyim* değerini 1 düşür (*Denklem 4.9.*). Tersi durumda değere sahip (yani daha iyi: mininizasyon ise ortalamanın altında, maksimizasyon ise ortalamanın üstünde) olan parçacıkların *Deneyim* değerini ise 1 artır (*Denklem 4.10.*). Söz konusu bu parçacıkların *Etkileşim_Oranı* değerini *Denklem 4.11.*'e göre güncelle.

Adım 3.2. Her parçacığın *Etkileşim_Oranı* değerini *Denklem 4.12.*'ye göre güncelle.

Adım 3.3. En iyi durumdaki parçacık dışında kalan bütün parçacıkların konumunu *Denklem 4.11.*'e göre güncelle.

Adım 3.4. Her parçacığın yeni konumuna göre amaç fonksiyon değerini (fitness) hesapla. Optimum değeri hesaplayan, o anda en iyi konumdaki parçacığın *Etkileşim_Oranı* değerini *Denklem 4.7.*'ye göre güncelle ve yine aynı parçacığın *Deneyim* değerini 1 artır (*Denklem 4.8.*).

Adım 3.5 (Olgunlaşma Fazı). *Deneyim* değeri *Olgunlaşma_Limiti* değerine eşit veya ondan daha iyi (mininizasyon ise altında, maksimizasyon ise üstünde) olan parçacıkların *Etkileşim_Oranı* değerini *Denklem 4.13.*'e göre güncelle. Optimum değeri hesaplayan (Adım 3.4.), o anda en iyi konumdaki parçacığın *Etkileşim_Oranı* değerini *Denklem 4.7.*'e göre güncelle ve yine aynı parçacığın *Deneyim* değerini 1 artır (*Denklem 4.8.*).

Adım 3.6 (Rasyonelleşme Fazı). *Deneyim* değeri 0'ın altında olan parçacıkların *Etkileşim_Oranı* değerini ve konumlarını sırasıyla *Denklem 4.14.* ve *Denklem 4.15.*'e göre güncelle. *Deneyim* değeri 0'a eşit veya ondan yüksek olan parçacıkların *Etkileşim_Oranı* değerlerini ise *Denklem 4.16.*'ya göre, *Rasyonallik_Oranı* değeri kadar güncelle.

Adım 3.7 (Dengeleme Fazı). Her parçacığın *Etkileşim_Oranı* değerini *Denklem 4.10.*'a göre güncelle. Her parçacığın yeni konumuna göre amaç fonksiyon değerini (fitness) hesapla. Optimum değeri hesaplayan, o anda en iyi konumdaki parçacığın *Etkileşim_Oranı* değerini *Denklem 4.7.*'ye göre güncelle ve yine aynı parçacığın *Deneyim* değerini 1 artır (*Denklem 4.8.*).

Adım 3.7. Çözüme yakınsama istendik düzeyde değilse sistem-içi optimizasyon uygula.

Adım 4. İterasyon – döngü sonunda küresel en iyi konuma göre elde edilen değer(ler) optimum değer(ler) olarak kabul edilmektedir.

BiGOA'da algoritma başlangıcında en iyi durumdaki parçacığın *Etkileşim_Oranı* değerine yönelik güncelleme, rastgele bir sayı (0 ile 1 arasında) ile çarpım ve elde edileni ekleme işlemine dayanmaktadır:

$$EnİyixEtkileşimOranı^{y} = EnİyixEtkileşimOranı^{g} + (EnİyixEtkileşimOranı^{g} * rsayı) \quad (4.7.)$$

Denklem 4.7.'de, *y* ve *g* simgeleri, en iyi durumdaki parçacığın geçerli (*g*) ve yeni (*y*) durumdaki *Etkileşim_Oranı* değerlerini göstermektedir.

İlgili parçacığın *Deneyim* değerinin geliştirilmesi, geçerli değerin 1 artırılması ile gerçekleştirilmektedir:

$$EnİyixDeneyim^{y} = EnİyixDeneyim^{g} + 1 \quad (4.8.)$$

Denklem 4.8.'de, *y* ve *g* simgeleri, en iyi durumdaki parçacığın geçerli (*g*) ve yeni (*y*) durumdaki *Deneyim* değerlerine karşılık gelmektedir.

Sosyalleşme Fazı – Adım 3.1.'de söz konusu olan *Deneyim* ve *Etkileşim_Oranı* değerlerine yönelik güncellemeler üç denklem kapsamında yerine getirilmektedir:

$$ofkxDeneyim_i^{z+1} = ofkxDeneyim_i^{z} - 1 \quad (4.9.)$$

$$ofixDeneyim_j^{z+1} = ofixDeneyim_j^{z} + 1 \quad (4.10.)$$

$$ofixEtkileşimOranı_j^{z+1} = ofixEtkileşimOranı_j^{z} + (ofixEtkileşimOranı_j^{z} * rsayı_j^{z}) \quad (4.11.)$$

Denklem 4.9., Denklem 4.10. ve *Denklem 4.11.*'de *z* zaman için (iterasyon – döngü anlamında) kullanılmaktadır. Diğer yandan, daha önce ifade edilen diğer temsiller dışında; ortalama amaç fonksiyon değerinden (fitness) kötü durumdaki parçacıkları belirtmek için *x* simgesi yerine *ofkx* simgesi ve parçacık numarası belirtimi için *i* (*Denklem 4.9.*), ortalama amaç fonksiyon değerinden (ortalama fitness) iyi durumdaki parçacıkları belirtmek için ise, *x* simgesi yerine *ofix* simgesi ve parçacık numarası belirtimi için *j* kullanılmıştır (*Denklem 4.10.* ve *Denklem 4.11.*).

Sosyalleşme Fazı – Adım 3.2.'de bütün parçacıkların *Etkileşim_Oranı* değerlerinin rastgele sayı(lar) (0 ile 1 arasında) geliştirilmesi işlemi gerçekleştirilmektedir:

$$xEtkileşimOranı_k^{z+1} = xEtkileşimOranı_k^{z} * rsayı_k^{z} \quad (4.12.)$$

Denklem 4.12.'de bütün parçacıklar kapsamında parçacık numarası belirtimi için *k* kullanılmaktadır.

Olgunlaşma Fazı – Adım 3.5.'te *Deneyim* değeri *Olgunlaşma_Limiti* değeri bağlamında daha iyi olan parçacıklara yönelik *Etkileşim_Oranı* güncellemesi, rastgele bir sayı (0 ile 1 arasında) ile çarpım ve elde edileni ekleme işlemine dayanmaktadır:

$$olixEtkileşimOranı_l^{z+1} = olixEtkileşimOranı_l^z + (olixEtkileşimOranı_l^z * rsayı_l^z) \quad (4.13.)$$

Denklem 4.13. kapsamında *l* simgesi ilgili parçacıkların numarasını belirtmek için kullanılmaktadır. *Olgunlaşma_Limiti* değeri bağlamında daha iyi olan parçacıkları belirtmek için ise, *x* simgesi yerine *olix* simgesi tercih edilmiştir.

Rasyonelleşme Fazı – Adım 3.6.'da *Deneyim* değeri 0'ın altında olan parçacıkların *Etkileşim_Oranı* değerleri ve konumları, o anda en iyi durumdaki parçacığın değerleri, bu parçacık bağlamında optimum değer(ler)in olduğu konum(lar) ve rastgele sayılar (0 ile 1 arasında) temel alınarak, yakınsamayla güncellenmektedir:

$$dsaxEtkileşimOranı_m^{z+1} = dsaxEtkileşimOranı_m^z + [(EnİyixEtkileşimOranı / dsaxEtkileşimOranı_m^z) * rsayı_m^z] \quad (4.14.)$$

$$dsaxKonum_m^{z+1} = dsaxKonum_m^z + \left[\left(dsaxKonum_m^z * (EnİyixKonum - dsaxKonum_m^z)\right) * rsayı_m^z\right] \quad (4.15.)$$

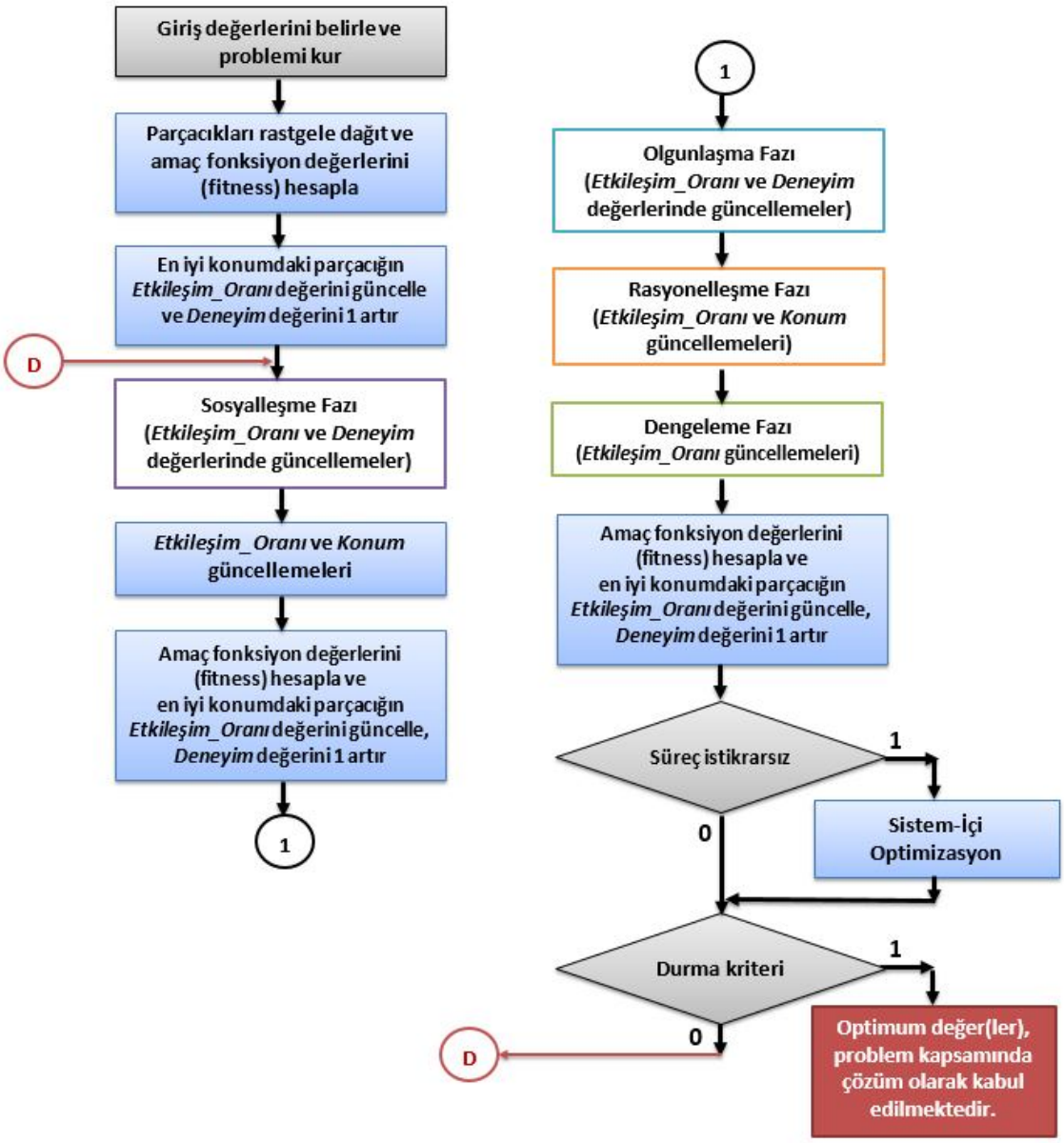
Denklem 4.14. ve *Denklem 4.15.*'te *m* simgesi söz konusu parçacıkların numarasını belirtmek için kullanılmaktadır. *Deneyim* değeri 0'ın altında olan parçacıkları belirtmek için ise, *x* simgesi yerine *dsax* simgesi tercih edilmiştir.

Deneyim değeri 0'ın üstünde veya ona eşit olan parçacıklar için *Rasyonallik_Oranı* değeri kadar uygulanan güncelleme işlemi de esasında *Denklem 4.14.*'teki güncellemeyle aynı fakat hedef kitlesi ve etki oranı farklı bir işlemdir:

$$dsüxEtkileşimOranı_n^{z+1} = dsüxEtkileşimOranı_n^z + [(EnİyixEtkileşimOranı / dsüxEtkileşimOranı_n^z) * rsayı_n^z] \quad (4.16.)$$

Denklem 4.16.'da *n* simgesi hedef parçacıkların numarasını belirtmek için; *dsüx* simgesi ise, *Deneyim* değeri 0'ın üstünde veya ona eşit parçacıkları belirtmek için, *x* simgesi yerine kullanılmıştır.

İfade edilen çözüm adımları dikkate alınmak suretiyle, BiGOA'nın özet akış şeması Şekil 4.3.'te gösterilmiştir.



Şekil 4. 3. BiGOA özet akış şeması.

BiGOA'daki sistem-içi optimizasyon, GOA'dakinden esinlenmekle birlikte kısaca şöyledir (Köse ve Arslan, 2017b):

- Söz konusu işlem adımına gelindiğinde, fonksiyon değerleri (fitness) ve parçacık konumlarının (değişkenler) stabilitesi ölçülmekte; stabilitasyon yoksa parçacıkların konum (değişken) ağırlıkları bulunmaktadır. BiGOA'daki stabilite bir bireyin zihinsel karmaşa oranı ile ilişkilendirilmiştir.

- Konum (değişken) ağırlıkları *a:* ağırlık sınırı parametresinin altına düşmeyen parçacıklara, diğer parçacıklar rastgele sayılar yardımıyla, rastgele yürüyüş gerçekleştirilerek yaklaştırılmaktadır. Anlaşılacağı üzere buradaki süreç GOA'dakine benzemekte, ancak daha basit bir matematiğe dayanmaktadır.

4.3. GOA ve BiGOA Çok-Amaçlı Optimizasyon Revizyonu

Literatürdeki SZ tabanlı algoritmalara baktığımızda farkedebileceğimiz önemli geliştirmelerden birisi, ilgili algoritma – tekniklerin genellikle tek-amaçlı optimizasyon problemlerine yönelik tasarlanıp geliştirilmesi, ardından ise çok-amaçlı optimizasyon için varyasyonlarının geliştirilmesidir. Benzer şekilde, bu tez çalışmasında tasarlanıp geliştirilen sürekli optimizasyon algoritmaları GOA ve BiGOA için de ilk başta basit tasarımlardan yola çıkılmış, ardından sistem-içi optimizasyon eklenmiş, son olarak da çok-amaçlı optimizasyon için bir 'anahtar' (switch) mekanizması kullanılmıştır.

Anahtar mekanizması ile çok-amaçlı optimizasyonlarda şu süreç işletilmektedir:

- Eğer problem çok-amaçlı bir optimizasyon ise, anahtar bayrak değişkeni (ç) çözüm süreci başlangıcında 1'e ayarlanır. Bu değişken tek-amaçlı ya da çok-amaçlı problemler arası süreç yönlendiren basir bir değişkendir.

- Süreç sırasında, pareto optimum çizmek için 'hafıza parçacıkları' kullanılır.

- Hafıza parçacıkları, GOA veya BiGOA algoritma yapısı detaylarına (Örneğin; GOA'daki parçacık statüleri) bakmaksızın pareto optimumları oluşturabilecek parçacıklar kümesini oluşturur.

- Süreç tıpkı PSO'da olduğu gibi, parçacıkların geçmiş değerlerini kullanması yönünde işlemektedir. Ama buradaki amaç pareto optimum oluşturmaktır.

- İteratif süreç sonunda pareto optimumları oluşturacak parçacıklar, çok-amaçlı optimizasyon probleminin de çözüm değerleri olarak kabul edilmektedir.

4.4. Algoritmaların Karmaşıklık Analizi

GOA ve BiGOA'nın algoritma karmaşıklığını bilmek, çözüm adımlarının ne ölçüde tasarlandığını ve algoritmaların eldeki sayısal kaynakları nasıl kullanacağına dair

temel bilgi sahibi olmamızı sağlayacaktır. Algoritmaların – tekniklerin karmaşıklığını, Big-O notasyonu (Danziger, 2015) üzerinden kısaca şöyle açıklayabiliriz:

- **GOA:** Algoritmadaki döngüler haricinde yer alan ve tek veya çok-amaçlı olması farketmeksizin, iteratif süreç – döngüden önce gerçekleşen işlem sayısını sabit sayı *A*, döngüden sonra gerçekleşen işlem sayısını ise sabit sayı *B* ile temsil edelim. Tipik olarak algoritmamızda döngü harici toplam işlem sayımız *A + B* ile ifade edilir. Ancak genel anlamda GOA ile çözülecek optimizasyon problemleri büyük çoğunlukla en az 2 boyut ve üzeri olacağından dolayı, *N* kere dönüş sağlayan bir döngü yapısı söz konusu olmaktadır. Döngü içi işlemleri de *C* temsil edersek, bu durumda GOA için toplam işlem sayısı *A + B + (N*C)* olmaktadır. Big-O notasyonuna göre GOA'nın karmaşıklığı böylelikle $O(N^2)$ ile gösterilmektedir. Bu durumda GOA için karesel bir karmaşıklıktan söz edebiliriz. GOA'nın karmaşıklığı maliyet anlamında kübik, üssel veya daha büyük yapılara göre daha az, logaritmik, doğrusal ve N-logaritmik olanlara göre ise daha fazladır.

- **BiGOA:** GOA'da olduğu gibi, algoritmadaki döngüler haricinde yer alan, iteratif süreç – döngüden önce gerçekleşen işlem sayısını *A*, döngüden sonrakileri ise *B* ile temsil edelim. BiGOA ile çözülecek optimizasyon problemlerinin de büyük çoğunlukla en az 2 boyut ve üzeri olacağını bildiğimizden dolayı, *N* kere dönüş sağlayan bir döngü yapısı söz konusu olmaktadır. Diğer yandan, algoritmaya özgü olan, her parçacığın diğer parçacıklarla olan kendi iç değerlendirmesi de ayrı bir iç döngüyü daha gerektirmektedir. Bu durumda, iç içe üç döngünün varlığı, Big-O notasyonuna göre BiGOA'nın karmaşıklığını $O(N^3)$ ile göstermemizi gerektirmektedir. GOA'dan farklı olarak BiGOA bu kabullerle birlikte kübik bir karmaşıklığı beraberinde getirmektedir. BiGOA'nın karmaşıklığı maliyet anlamında üssel veya daha büyük yapılara göre daha az, logaritmik, doğrusal, N-logaritmik, ve karesel olanlara göre ise daha fazladır.

5. DEĞERLENDİRME VE TARTIŞMA

Bu bölüm ile birlikte, ortaya konulan iki farklı optimizasyon algoritmasının başarımını değerlendirmeye yönelik olarak yapılmış olan çalışmalar ile elde edilen bulgular okuyuculara sunulmuş, bulguların rapor edilmesi sonrasında ise, –söz konu bulgulara dayanarak– çeşitli yorumlamalarda bulunulmuştur.

5.1. Test Fonksiyonlarına Yönelik Bulgular

GOA ve BiGOA'nın, ilgili algoritma – tekniklerle birlikte, test fonksiyonlarıyla değerlendirilmesine yönelik bulgular bu alt-bölüm kapsamında sunulmuştur. Bu bağlamda, her bir algoritma – teknik için temel bulgular adı altında: en iyi elde edilen değer, ortalama değer ve standart sapma bulgularına, yakınsama eğrilerine, Wilcoxon İşaretli Sıralama Testi bulgularına ve nihayetinde Ortalama Hataların Karesi (OHK), Ortalama Hataların Karesinin Kökü (OHKK), ve Ortalama Mutlak Hata Yüzdesi (OMHY) değerlerine, çok-amaçlı optimizasyon test fonksiyonları için ise pareto optimum grafiklerine yer verilmiş, son olarak da algoritmalar – teknikler arası gerçekleştirilen başarı sıralaması sunulmuştur.

Değerlendirmeye dâhil olan tek-amaçlı bütün test fonksiyonlar için ilgili algoritmalar – teknikler tarafından, ifade edilen şartlar altında, 50 çalıştırma sonrası elde edilen 'en iyi değer', 'ortalama değer', ve 'standart sapma' temel bulguları Çizelge 5.1.'de verilmiştir.

Çizelge 5.1., takip eden sayfadadır.

Çizelge 5. 1. Tek-amaçlı fonksiyonlarda değerlendirilen algoritmalarla elde edilen temel bulgular.

Fonksiyon	Bulgu[a]	GOA	BiGOA	BYAO	BYAO$_{V1}$	ÇTO	ÇTO$_{V1}$	DE	DE$_{V1}$	ERA	ERA$_{V1}$	GA	GA$_{V1}$	GKA	GKA$_{V1}$	KSA	KSA$_{V1}$	PSO	PSO$_{V1}$	YAK	YAK$_{V1}$
Beale	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
Booth	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
Bukin (no. 6)	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	4,180E-14	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
Cross-in Tray	**En İyi**	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00
	Ortalama	-2,063E+00	**-2,063E+00**	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,062E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,046E+00	-2,060E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,063E+00	-2,187E+00	-2,063E+00	-2,063E+00	-2,063E+00
	Std. Sapma	1,417E-10	2,204E-11	3,982E-09	5,103E-10	7,396E-06	3,338E-09	0,501E+00	0,338E-05	2,388E-09	0,101E-11	3,531E+00	1,501E+00	1,590E-09	1,225E-09	1,610E-08	0,119E-08	3,110E+00	1,417E-10	3,111E-07	1,590E-09
Easom	**En İyi**	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00
	Ortalama	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	0,241E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	-1,000E+00	0,008E+00	-1,000E+00	-1,000E+00	**-1,000E+00**
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	1,049E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,381E+00	0,000E+00	0,000E+00	0,000E+00
Eggholder	**En İyi**	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-3,165E+01	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02
	Ortalama	-9,597E+02	-9,611E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-2,280E+01	-9,211E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,596E+02	-9,597E+02	-9,596E+02	-9,596E+02	**-9,596E+02**
	Std. Sapma	0,086E+00	0,369E+00	0,000E+00	0,000E+00	0,714E-10	1,017E-12	0,000E+00	0,000E+00	0,000E+00	0,000E+00	3,792E+00	1,301E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,073E+00	0,000E+00	0,000E+00	0,000E+00
Levy (no. 13)	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,603E-06	0,000E+00	0,000E+00	0,000E+00	0,000E+00	1,043E-08	1,010E-08	0,000E+00	0,000E+00	0,088E+00	0,017E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	2,179E-09	0,000E+00	0,000E+00	0,000E+00
Matyas	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	1,031E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	2,542E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
McCormick	**En İyi**	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00
	Ortalama	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	**-1,913E+00**	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,910E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00	-1,913E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	1,053E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00

Çizelge 5.1. ***Devamı***

Fonksiyon	Bulgu[a]	GOA	BiGOA	BYAO	BYAO$_{V1}$	ÇTO	ÇTO$_{V1}$	DE	DE$_{V1}$	ERA	ERA$_{V1}$	GA	GA$_{V1}$	GKA	GKA$_{V1}$	KSA	KSA$_{V1}$	PSO	PSO$_{V1}$	YAK	YAK$_{V1}$
Schaffer (no. 2)	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Ortalama	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
	Std. Sapma	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00
Ackley 30 b.	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	5,306E-07	6,117E-09	7,274E-08	4,110E-10	2,511E+01	7,104E-01	0,000E+00	0,000E+00	4,393E-02	3,196E-04	8,231E-03	7,191E-05	0,000E+00	0,000E+00
	Ortalama	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	6,422E-07	7,021E-09	7,275E-08	4,303E-10	2,522E+01	7,107E-01	0,000E+00	**0,000E+00**	4,401E-02	3,201E-04	9,517E-03	8,303E-05	0,000E+00	0,000E+00
	Std. Sapma	3,111E-09	5,181E-07	7,001E-08	3,447E-07	5,502E-10	3,748E-12	7,330E+00	5,619E+00	4,388E+00	2,481E+00	4,080E+00	5,320E+00	6,511E-10	2,940E-11	2,184E+00	5,104E+00	3,029E+00	2,100E+00	4,890E-12	2,110E-14
Rastrigin 30 b.	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	3,299E-01	3,100E-06	0,000E+00	0,000E+00	4,114E-05	2,100E-06	4,091E-03	8,111E-08	0,000E+00	0,000E+00
	Ortalama	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	3,311E-01	4,099E-06	0,000E+00	**0,000E+00**	5,040E-05	2,233E-06	5,131E-03	9,030E-08	0,000E+00	0,000E+00
	Std. Sapma	4,567E-10	4,101E-04	6,560E-08	7,310E-09	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	9,021E+00	4,732E+00	0,000E+00	0,000E+00	3,110E+00	4,027E+00	4,281E+00	3,107E+00	0,000E+00	0,000E+00
Rosenbrock 30 b.	**En İyi**	6,931E+00	4,105E+01	3,012E+01	1,103E+01	6,101E+01	5,001E+00	5,751E+01	3,003E+01	3,010E+01	2,108E+01	6,110E+05	6,111E+02	3,509E+01	4,910E+01	7,800E+01	5,111E+01	3,690E+06	3,820E+03	7,011E+01	4,701E+01
	Ortalama	7,511E+00	4,105E+01	3,022E+01	1,201E+01	6,101E+01	**5,101E+00**	5,851E+01	4,101E+01	3,011E+01	2,108E+01	6,110E+05	6,111E+02	3,509E+01	4,910E+01	7,809E+01	6,111E+01	3,690E+06	4,910E+03	7,191E+01	5,091E+01
	Std. Sapma	4,561E-02	5,120E+01	7,011E-02	5,111E-01	5,120E+00	7,011E-05	5,120E+00	9,458E-02	5,120E+01	7,011E-02	5,120E+02	7,011E-02	5,120E+02	6,081E-01	6,133E-06	1,179E-05	5,179E+00	6,039E+00	9,591E+03	1,009E+00
Sphere (Küre) 30 b.	**En İyi**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,200E+01	0,008E+00	0,000E+00	0,000E+00
	Ortalama	**0,000E+00**	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,210E+01	0,060E+00	0,000E+00	0,000E+00
	Std. Sapma	3,361E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	4,101E-02	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	0,000E+00	1,012E+00	5,011E+00	0,000E+00	0,000E+00
Styblinski–Tang 30 b.	**En İyi**	-3,166E+04	-1,407E+05	-4,822E+06	-2,901E+04	-1,997E+04	-6,600E+05	-9,503E+06	-2,009E+04	-2,429E+04	-1,718E+04	-4,302E+05	-7,237E+05	-3,943E+04	-5,003E+04	-4,517E+05	-4,107E+05	-2,111E+04	-3,701E+04	-1,944E+04	-5,117E+04
	Ortalama	-3,116E+04	-1,007E+05	-4,501E+06	-2,901E+04	-1,997E+04	-6,401E+05	-9,401E+06	-1,999E+04	-2,429E+04	-1,718E+04	-4,302E+05	-7,237E+05	-3,943E+04	**-4,903E+04**	-4,411E+05	-4,107E+05	-2,111E+04	-3,701E+04	-1,944E+04	-5,116E+04
	Std. Sapma	6,121E-06	4,154E-05	3,081E-06	3,487E-04	1,006E-06	2,778E-05	5,330E+00	5,619E+00	4,378E+00	6,481E+00	4,080E+00	5,320E+00	8,581E-10	2,940E-10	2,184E+00	8,114E+00	8,039E+00	8,100E-02	6,890E-08	4,110E-07
Mishra'nın Kuşu (Mishra's Bird)	**En İyi**	-4,019E+01	-5,329E+00	-3,367E+01	-7,098E+01	-4,100E+00	-7,10E+01	-8,067E+01	-7,130E+01	-6,910E+01	-7,121E+01	-3,149E+01	-7,099E+01	-8,406E+01	-3,149E+01	-3,339E+01	-1,009E+02	-3,149E+01	-7,989E+01	-1,006E+02	-1,078E+02
	Ortalama	-3,119E+01	-4,329E+00	-3,367E+01	-7,098E+01	-4,100E+00	-7,10E+01	-8,067E+01	-6,910E+01	-6,910E+01	-7,111E+01	-3,149E+01	-6,999E+01	-8,406E+01	-3,149E+01	-3,339E+01	-1,009E+02	-3,149E+01	-7,989E+01	-1,006E+02	**-1,077E+02**
	Std. Sapma	9,571E+00	7,110E+00	9,012E-03	4,111E-01	5,120E+00	7,011E-05	5,120E+00	4,929E+00	5,120E+01	7,011E-02	5,120E+02	8,021E-02	5,120E+02	9,012E-03	6,133E-06	2,179E-04	5,179E+00	6,039E+00	8,591E+03	2,028E+00

Çizelge 5.1. ***Devamı***

Fonksiyon	Bulgu[a]	GOA	BiGOA	BYAO	BYAO$_{V1}$	ÇTO	ÇTO$_{V1}$	DE	DE$_{V1}$	ERA	ERA$_{V1}$	GA	GA$_{V1}$	GKA	GKA$_{V1}$	KSA	KSA$_{V1}$	PSO	PSO$_{V1}$	YAK	YAK$_{V1}$
Simionescu	**En İyi**	-3,616E-01	-1,009E+01	-4,521E+01	-3,001E+00	-2,007E+00	-6,441E-01	-9,401E+06	-2,111E+04	-2,431E+04	-1,720E+04	-4,302E+05	-7,237E+05	-8,043E-01	-7,289E-02	-3,221E-01	-5,666E-02	-7,018E+01	-7,226E-01	-8,129E-02	-9,848E-02
	Ortalama	-3,116E-01	-1,007E+01	-4,501E+01	-2,901E+00	-1,997E+00	-6,401E-01	-9,401E+06	-1,999E+04	-2,429E+04	-1,718E+04	-4,302E+05	-7,237E+05	-8,043E-01	**-7,271E-02**	-3,216E-01	-5,666E-02	-7,006E+01	-7,226E-01	-8,126E-02	-9,236E-02
	Std. Sapma	6,121E-05	4,154E-02	8,081E-01	7,487E-03	4,027E-06	9,718E-01	8,130E+00	7,619E+00	8,378E+00	6,481E+00	4,080E+00	8,530E+01	6,919E+00	2,308E+01	3,081E+00	7,330E+00	5,079E+01	4,100E-01	3,890E+00	6,210E+00
Townsend	**En İyi**	2,031E+00	5,015E+01	6,800E+00	9,005E+02	4,110E+00	7,480E+00	5,820E+00	7,400E+01	2,888E+00	6,929E+01	8,021E+00	6,339E+00	2,712E+00	3,719E+00	7,021E+00	4,911E+00	2,911E+02	5,899E+01	2,781E+00	2,517E+00
	Ortalama	**2,049E+00**	5,055E+01	7,100E+00	9,008E+02	4,114E+00	7,980E+00	5,821E+00	7,449E+01	3,022E+00	6,929E+01	8,021E+00	6,339E+00	2,722E+00	3,719E+00	7,021E+00	4,911E+00	2,911E+02	6,722E+01	3,011E+00	2,811E+00
	Std. Sapma	1,227E-01	5,082E+00	5,037E+01	4,792E+00	8,747E+00	1,696E+01	5,027E+00	5,972E+00	2,722E+00	2,272E+01	3,890E+00	3,190E+00	2,272E+01	4,797E+00	1,099E+00	5,273E+01	4,739E+00	3,739E+00	3,277E-01	8,797E+00
Dönd. Cigar (Rot. Cigar) 30 b.	**En İyi**	4,811E+02	6,019E+02	5,829E+02	4,799E+05	3,792E+03	4,007E+09	3,990E+05	5,792E+02	3,729E+06	2,827E+05	5,992E+05	2,927E+05	4,827E+05	6,203E+02	2,027E+05	8,179E+02	4,999E+06	2,999E+03	6,811E+03	6,207E+03
	Ortalama	**4,929E+02**	6,039E+02	6,229E+02	4,799E+05	3,792E+03	4,097E+09	3,999E+05	5,792E+02	3,729E+06	2,827E+05	6,222E+05	2,927E+05	4,827E+05	6,222E+02	2,927E+05	8,779E+02	4,999E+06	2,999E+03	7,220E+03	6,297E+03
	Std. Sapma	2,009E+00	3,982E+00	3,097E+00	4,792E+00	7,747E+00	2,676E-02	3,027E+00	4,892E+00	2,722E+00	2,272E-02	4,797E+00	2,293E+00	2,272E-02	4,797E+00	2,293E+00	3,273E+01	4,739E+00	6,739E+00	6,277E-01	7,797E+00
Dönd. Yük. Kond. Elip. (R. H. C. El.) 30 b.	**En İyi**	4,910E+03	3,171E+05	5,099E+05	4,021E+07	5,490E+08	1,081E+04	3,929E+05	5,621E+04	5,813E+06	5,149E+05	4,999E+08	2,509E+06	5,708E+05	8,001E+03	2,910E+05	3,709E+05	5,922E+06	4,200E+07	5,120E+04	9,170E+03
	Ortalama	5,510E+03	2,611E+05	5,099E+05	4,021E+07	5,490E+08	1,081E+04	3,929E+05	5,721E+04	4,019E+06	5,449E+05	5,819E+08	2,509E+06	5,810E+05	**8,001E+03**	2,910E+05	3,709E+05	5,922E+06	4,200E+07	5,120E+04	9,320E+03
	Std. Sapma	5,421E+01	3,126E-01	5,021E-01	7,057E-01	9,676E-03	7,778E-03	4,310E+00	5,821E+00	2,711E+00	5,921E+00	3,111E+00	5,820E+00	1,606E-04	3,778E-03	7,330E+00	9,426E-03	5,039E+00	3,100E-01	4,830E+01	5,110E+01
Kayd. ve Dönd. Ackley (S. R. Ackley) 30 b.	**En İyi**	2,500E+02	2,211E+05	2,099E+05	4,001E+07	1,990E+08	1,081E+05	3,929E+05	2,721E+05	4,019E+02	2,849E+05	2,819E+08	4,209E+02	2,810E+05	7,881E+05	2,910E+05	3,709E+05	5,922E+02	4,200E+07	8,070E+02	6,990E+02
	Ortalama	**2,519E+02**	2,211E+05	2,099E+05	4,021E+07	2,490E+08	1,081E+05	3,929E+05	2,721E+05	4,019E+02	2,849E+05	2,819E+08	4,209E+02	2,810E+05	8,111E+05	2,910E+05	3,709E+05	5,922E+02	4,200E+07	8,120E+02	7,320E+02
	Std. Sapma	2,421E+01	8,022E-02	1,021E-01	7,987E-02	6,272E-03	2,778E-03	3,310E+00	5,821E+00	2,211E+00	5,921E+00	8,011E+00	2,820E+00	1,171E-05	5,090E+00	2,124E+00	8,004E+01	2,039E+00	8,000E-02	4,830E+01	2,110E+01
Kayd. ve Dönd. Rastrigin (S. R. Rast.) 30 b.	**En İyi**	6,069E+04	2,421E+05	6,049E+05	4,021E+07	6,489E+08	1,033E+04	3,929E+05	6,721E+04	4,019E+06	6,849E+05	5,819E+08	2,709E+06	6,810E+05	8,111E+04	2,910E+05	3,709E+04	5,922E+06	4,200E+07	6,120E+05	6,320E+05
	Ortalama	6,519E+04	2,611E+05	6,099E+05	4,021E+07	6,490E+08	**1,081E+04**	3,929E+05	6,721E+04	4,019E+06	6,849E+05	6,819E+08	2,709E+06	6,810E+05	8,111E+04	2,910E+05	3,709E+04	5,922E+06	4,200E+07	6,120E+05	7,320E+05
	Std. Sapma	6,421E+01	3,126E-02	6,021E-01	7,987E-02	9,676E-03	7,778E-03	4,310E+00	5,821E+00	2,711E+00	5,921E+00	3,111E+00	6,820E+00	1,171E-05	5,090E+00	2,124E+00	3,104E+01	6,039E+00	3,100E-02	4,830E+01	6,110E+01
Kayd. ve Dönd. Schw. (S. R. Schw.) 30 b.	**En İyi**	5,209E+02	5,001E+04	8,079E+04	5,011E+03	7,449E+03	3,022E+02	7,929E+05	9,921E+02	5,719E+03	8,349E+03	5,819E+03	3,889E+04	5,919E+04	5,091E+05	2,120E+03	5,709E+03	1,032E+04	7,160E+04	8,020E+02	7,020E+02
	Ortalama	5,219E+02	6,911E+04	8,089E+04	5,911E+03	7,449E+03	**3,056E+02**	7,929E+05	9,921E+02	5,919E+03	8,349E+03	6,119E+03	3,999E+04	5,919E+04	5,091E+05	2,910E+03	5,709E+03	1,922E+04	7,200E+04	8,120E+02	7,210E+02
	Std. Sapma	4,121E+00	1,146E-03	5,061E+00	6,987E-03	1,606E-04	3,778E-03	7,330E+00	6,919E+00	1,308E+00	4,981E+00	4,080E+00	3,820E+00	3,181E-05	7,820E+00	3,124E+00	6,104E+01	8,039E+00	8,100E-02	7,090E+00	3,010E+00

Çizelge 5.1.'teki bulgular ışığında, bütün tek-amaçlı test fonksiyonları için değerlendirmeye dâhil olan algoritmaların – tekniklerin başarımları konusunda şu genel yorumları yapabiliriz:

- 10 kısıtsız tek-amaçlı fonksiyon için GOA, GKA_{v1}, $ÇTO_{v1}$, BiGOA ve YAK_{v1} genel olarak başarılı sonuçlar sergilemiş, bu algoritmaları diğerleri izlemiştir. Sıralamaya tabi tutulduğunda en çok başarılı sonuç elde eden algoritma GOA ve GKA_{v1} olurken, bu algoritmayı $ÇTO_{v1}$ ve YAK_{v1} ve BiGOA birbirine yakın başarımlarla takip etmektedir.

- 3 kısıtlı tek-amaçlı fonksiyonlar için sırasıyla GOA, GKA_{v1}, YAK_{v1} ve $ÇTO_{v1}$ yüksek başarılar göstermiştir.

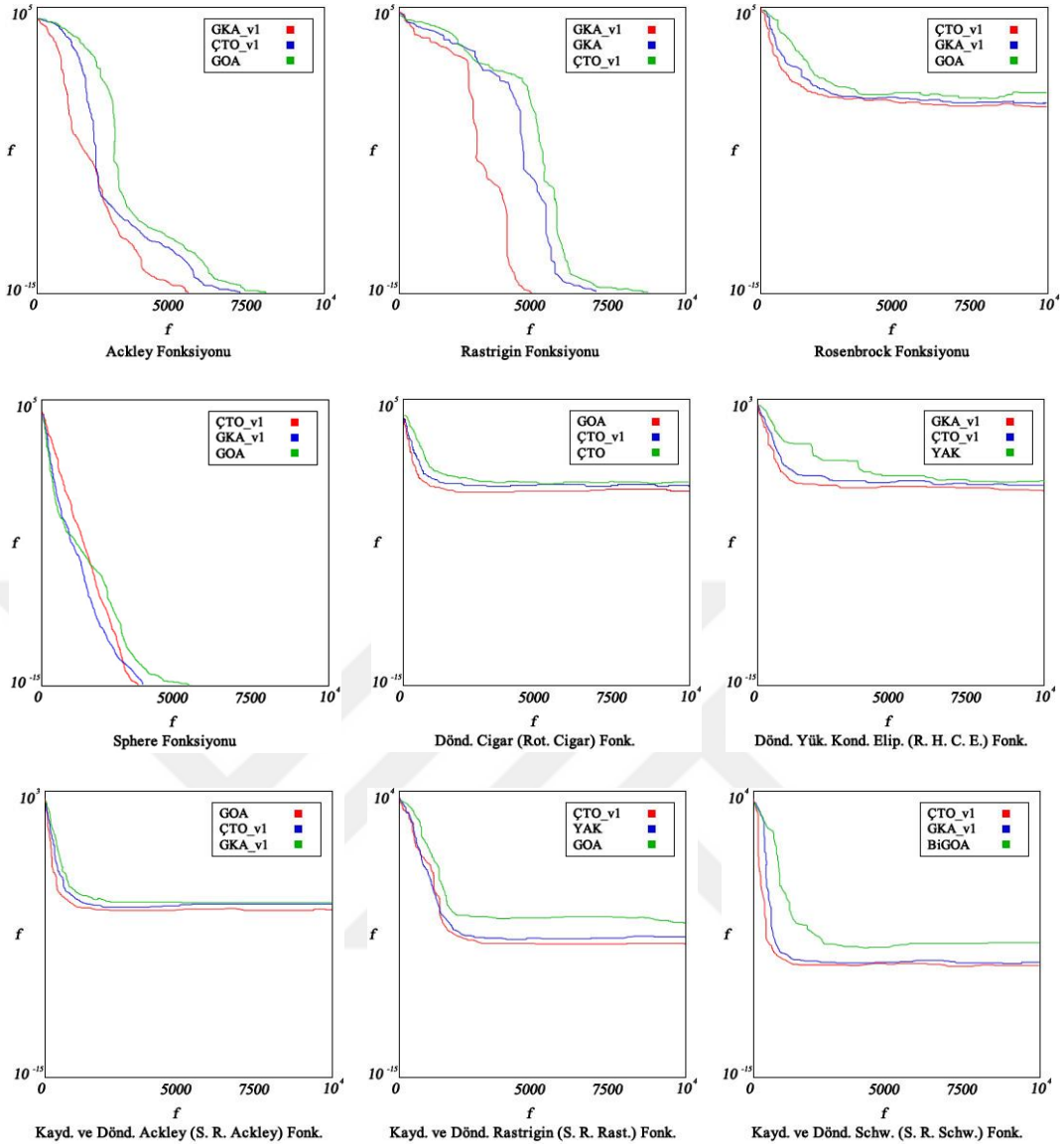
- 5 kısıtsız tek-amaçlı ve çok-boyutlu fonksiyon için GKA_{v1} en iyi başarımı gösteren algoritma olmuş, bu algoritmayı $ÇTO_{v1}$ GOA, BiGOA, GKA ve YAK_{v1} gibi algoritmalar takip etmiştir. Buradan görülmektedir ki GOA ve BiGOA, fonksiyonların boyutu yükselse bile diğer algoritmalarla yakın bulgulara ulaşma eğilimi göstermiştir.

- IEEE CEC 2015 etkinliğinden seçilen 5 çok-boyutlu fonksiyon için YAK_{v1} en iyi başarımı gösterirken, GOA, YAK, GKA_{v1}, BiGOA $ÇTO_{v1}$'nun da genel anlamda diğerlerinden daha iyi sonuçlar ortaya koyduğu görülmektedir.

Çizelge 5.1.'de 10^{-15}'ten küçük değerler 0 olarak gösterilmiştir. Ancak tabi ki aynı test fonksiyonu için farklı algoritmalar arası başarı sıralaması değişkenlik göstermektedir. Bu konuda daha net bilgiler ilerleyen çizelgelerde mevcuttur.

İlgili tek-amaçlı test fonksiyonlarından, çözümü bütün algoritmalar tarafından iyi düzeyde yapılmış olanlar hariç olmak üzere, önemli olduğu düşünülen fonksiyonlar için, en iyi sonuç veren üç algoritmanın – tekniğin çizmiş olduğu yakınsama eğrileri, Şekil 5.1.'de sunulmuştur.

Şekil 5.1., takip eden sayfadadır.



Şekil 5. 1. Tek-amaçlı çok-boyutlu önemli fonksiyonlarda en iyi sonuç veren ilk üç algoritmanın yakınsama eğrileri.

Temel bulgular ve yakınsama eğrilerine ek olarak, bütün tek-amaçlı fonksiyonlar için, GOA, BiGOA ve diğer algoritmalar – teknikler arası Wilcoxon İşaretli Sıralama Testi'ne (α = 0,05 anlamlılık düzeyi, yani %95 güven aralığında olmak üzere) yönelik olarak elde edilmiş olan bulgular da Çizelge 5.2., Çizelge 5.3., Çizelge 5.4. ve Çizelge 5.5.'te verilmiştir.

Çizelge 5.2., Çizelge 5.3., Çizelge 5.4. ve Çizelge 5.5. takip eden sayfalardadır.

Çizelge 5. 2. Tek-amaçlı fonksiyonlar için GOA ve diğer algoritmalar arası Wilcoxon İşaretli Sıralama Testi bulguları (1. kısım).

Fonksiyon	GOA – BYAO			GOA – BYAO$_{V1}$			GOA – ÇTO			GOA – ÇTO$_{V1}$			GOA – DE			GOA – DE$_{V1}$			GOA – ERA			GOA – ERA$_{V1}$			GOA – GA		
	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W
Beale	4,48 E-06	0	●	4,02 E-06	0	●	8,83 E-07	3	●	3,34 E-06	0	●	4,21 E-01	0	●	1,12 E-05	0	●	2,62 E-06	0	●	4,08 E-05	0	●	2,56 E-04	1	●
Booth	5,44 E-05	0	●	4,28 E-05	6	●	1,87 E-01	54	●	2,72 E-01	47	●	4,17 E-03	7	●	7,88 E-07	2	●	6,68 E-04	0	●	4,58 E-06	0	●	6,16 E-02	0	●
Bukin (no. 6)	1,12 E-03	0	●	7,86 E-07	0	●	7,36 E-01	33	=	3,47 E+00	0	=	6,07 E-05	0	●	2,76 E-07	0	●	7,16 E-06	0	●	8,83 E-06	0	●	5,25 E-04	0	●
Cross-in Tray	3,59 E-08	0	●	2,88 E-01	2	●	7,56 E-05	0	●	2,36 E-06	2	●	3,41 E-01	6	●	1,12 E-01	0	●	1,46 E-02	5	●	6,45 E+00	0	=	3,34 E-03	0	●
Easom	9,56 E-02	0	●	1,41 E-07	0	●	6,16 E-03	0	●	8,66 E+00	0	=	5,21 E-02	0	●	1,45 E+00	0	=	3,87 E-02	0	●	3,26 E-01	0	●	8,54 E-01	0	●
Eggholder	6,17 E-09	0	●	1,85 E-01	76	=	2,66 E+00	0	=	7,11 E+00	0	=	8,86 E-07	5	●	2,74 E-04	0	●	2,15 E-06	0	●	6,21 E-02	0	●	2,45 E-04	66	●
Levy (no. 13)	2,57 E-04	0	●	6,33 E-06	0	●	4,86 E-01	111	=	1,54 E-01	133	=	2,20 E-02	0	●	8,58 E-05	122	●	6,85 E-03	0	●	2,60 E+00	0	=	5,48 E-06	0	●
Matyas	1,30 E-06	1	●	1,61 E-05	4	●	3,88 E-01	0	●	6,16 E-05	0	●	5,37 E-05	112	●	8,51 E-02	0	●	2,56 E-07	139	●	1,41 E-04	1	●	8,86 E-06	0	●
McCormick	1,69 E-04	0	●	6,25 E-03	0	●	1,63 E+00	0	=	1,12 E-01	0	=	8,87 E-06	0	●	7,56 E-03	0	●	6,10 E-04	0	●	6,82 E-07	0	●	6,51 E-03	0	●
Schaffer (no. 2)	1,07 E-04	5	●	4,46 E-01	0	●	1,41 E-07	1	●	3,34 E-01	43	=	4,21 E-04	45	●	1,42 E-04	0	●	7,55 E-04	0	●	4,67 E-03	0	●	6,05 E-01	3	●
Ackley 30 b.	7,12 E-01	157	=	1,46 E-01	232	=	3,07 E-05	0	●	2,12 E+00	0	=	4,76 E-05	0	●	1,86 E-01	55	=	1,88 E-05	8	●	4,57 E-07	0	●	7,55 E-04	3	●
Rastrigin 30 b.	3,10 E-04	0	●	1,43 E-01	187	=	5,64 E-07	0	●	7,77 E-01	16	=	6,22 E+00	0	=	3,22 E-02	0	●	6,07 E-06	0	●	4,78 E-07	0	●	2,21 E-05	0	●
Rosenbrock 30 b.	2,28 E-03	0	●	3,65 E-01	32	=	7,88 E+00	0	=	1,44 E-02	0	○	2,00 E-04	0	●	1,70 E-03	0	●	4,15 E-02	0	●	4,76 E-03	231	○	1,71 E-06	5	●
Sphere (Küre) 30 b.	8,43 E-08	0	●	2,30 E-05	0	●	2,84 E-07	0	●	3,15 E+00	0	=	1,46 E-02	4	●	4,70 E-05	0	●	6,14 E-07	0	●	7,62 E-04	0	●	1,28 E-06	0	●
Styblinski–Tang 30 b.	6,07 E-07	2	●	3,82 E-04	3	●	8,32 E-02	1	●	7,87 E-05	8	●	8,45 E-05	6	●	6,06 E-05	6	●	3,37 E-01	56	=	3,44 E-01	54	=	3,68 E-06	0	●
Mishra'nın Kuşu (Mishra's Bird)	2,11 E-03	0	●	4,53 E-04	0	●	3,67 E-03	0	●	6,13 E-06	56	●	6,34 E-01	0	●	5,86 E-06	0	●	6,02 E+00	0	=	7,67 E-01	76	=	4,43 E-04	6	●
Simionescu	3,75 E-03	0	●	5,83 E-07	0	●	4,82 E-07	0	●	3,23 E+00	0	=	4,23 E-07	0	●	2,83 E-06	0	●	4,42 E-01	19	=	5,54 E-02	0	○	5,73 E-02	0	●
Townsend	3,48 E-05	0	●	3,66 E-03	0	●	3,51 E+00	0	=	1,87 E-01	132	=	1,86 E-01	0	●	7,01 E-05	0	●	2,63 E-06	1	●	7,77 E-01	33	=	2,65 E-01	0	●
Dönd. Cigar (R. Cigar) 30 b.	4,71 E-01	101	=	1,50 E-01	33	=	3,35 E+00	0	=	8,13 E-04	4	○	1,86 E-06	44	●	4,70 E-02	0	●	8,71 E-02	0	●	3,23 E-03	0	●	8,50 E-06	119	●
Dönd. Yük. Kond. Elip. (R. H. C. El.) 30 b.	1,48 E-02	0	●	1,04 E-04	57	○	1,04 E-07	8	●	5,30 E-01	11	●	8,32 E-05	12	●	3,61 E-01	100	=	2,54 E-06	0	●	4,81 E-04	0	●	6,72 E-03	0	●
Kayd. ve Dönd. Ackley (S. R. Ackley) 30 b.	7,71 E-09	0	●	5,56 E-06	0	●	3,41 E-03	0	●	8,64 E+00	0	=	2,87 E-01	136	=	5,27 E+00	0	=	1,24 E-01	0	●	7,56 E-02	6	●	8,33 E-06	0	●
Kayd. ve Dönd. Rast.(S. R. Rast.) 30 b.	6,84 E-01	86	=	5,25 E-07	165	○	1,10 E-01	89	=	5,34 E-02	2	●	4,03 E-06	4	●	7,77 E-01	0	●	1,44 E-01	1	●	1,32 E-07	0	●	5,04 E-06	56	●
Kayd. ve Dönd. Schw. (S. R. Schw.) 30 b.	3,18 E-05	7	●	5,12 E-01	2	=	8,77 E-01	120	=	5,13 E-06	66	○	5,87 E-07	0	●	6,45 E-03	0	●	5,68 E-01	0	●	2,03 E-01	1	=	1,78 E-04	0	●
● / = / ○	20/3/0			15/6/2			16/7/0			8/12/3			21/2/0			19/4/0			20/3/0			16/6/1			23/0/0		
(α = 0,05)																											

Çizelge 5. 3. Tek-amaçlı fonksiyonlar için GOA ve diğer algoritmalar arası Wilcoxon İşaretli Sıralama Testi bulguları (2. kısım).

Fonksiyon	GOA – GA_{V1}			GOA – GKA			GOA – GKA_{V1}			GOA – KSA			GOA – KSA_{V1}			GOA – PSO			GOA – PSO_{V1}			GOA – YAK			GOA – YAK_{V1}		
	p	T	W	P	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W	p	T	W
Beale	4,73 E-07	1	●	4,57 E-04	0	●	8,57 E-01	0	●	1,80 E-03	0	●	2,05 E-07	0	●	5,42 E-07	0	●	6,38 E-07	0	●	5,55 E-03	0	●	1,25 E-02	0	●
Booth	4,40 E-03	0	●	3,65 E-06	0	●	8,71 E-02	7	●	1,25 E-04	0	●	2,33 E-03	1	●	1,30 E-06	0	●	8,55 E-04	43	●	4,07 E-01	56	=	3,42 E-01	0	●
Bukin (no. 6)	8,50 E-03	1	●	4,72 E-01	17	=	1,42 E-07	0	●	2,86 E-01	218	=	3,44 E-01	43	=	8,50 E-07	9	●	1,66 E-05	0	●	2,20 E+00	0	=	3,55 E+00	0	=
Cross-in Tray	3,47 E-07	6	●	3,80 E-07	8	●	8,47 E-01	99	○	4,05 E-07	0	●	8,53 E-07	0	●	3,18 E-05	0	●	1,85 E-04	0	●	1,10 E-05	196	○	8,08 E-04	0	●
Easom	8,00 E-03	0	●	7,88 E+00	0	=	2,82 E-01	2	=	6,50 E-03	0	●	8,34 E-05	0	●	2,51 E-04	0	●	4,35 E-07	0	●	4,31 E+00	0	=	3,33 E-07	0	●
Eggholder	3,57 E-04	27	●	3,12 E-02	0	●	4,57 E-05	0	●	1,71 E-01	49	●	3,42 E-04	7	●	7,78 E-06	2	●	7,75 E-06	0	●	1,65 E+00	0	=	6,46 E-04	0	●
Levy (no. 13)	7,35 E-07	0	●	5,58 E-06	0	●	3,06 E-03	0	●	7,14 E-05	0	●	8,61 E-05	0	●	3,00 E-03	0	●	3,36 E-03	0	●	1,51 E-05	0	●	5,35 E+00	0	=
Matyas	8,52 E-05	6	●	6,36 E-07	0	●	6,73 E-05	1	●	8,03 E-07	7	●	8,60 E-03	0	●	4,86 E-07	0	●	1,13 E-06	0	●	5,17 E+00	0	=	5,11 E-01	99	=
McCormick	2,87 E-05	0	●	1,61 E-07	0	●	6,77 E-07	0	●	1,13 E-07	0	●	1,12 E-05	50	●	3,18 E-03	4	●	7,52 E-01	0	●	1,38 E-02	6	●	6,46 E-04	0	●
Schaffer (no. 2)	8,87 E-02	0	●	2,41 E-06	0	●	2,65 E-06	1	●	3,12 E-01	0	●	1,51 E-02	0	●	4,61 E-05	0	●	3,66 E-02	0	●	3,53 E-07	0	●	3,78 E-01	0	●
Ackley 30 b.	8,33 E-01	33	●	3,02 E+00	0	=	4,86 E+00	0	=	5,01 E+00	0	=	5,83 E-01	137	=	7,66 E-02	0	●	5,71 E+00	141	=	5,21 E-04	0	●	2,43 E-04	0	●
Rastrigin 30 b.	3,07 E-01	171	=	7,64 E-06	22	●	3,55 E+00	0	=	5,00 E-02	167	○	4,03 E-01	72	○	5,22 E-07	0	●	7,24 E-04	0	●	4,45 E-03	0	○	7,17 E-02	8	●
Rosenbrock 30 b.	6,51 E-01	0	●	6,70 E+00	0	=	4,77 E-01	55	=	2,06 E+00	0	=	1,07 E-02	0	●	1,37 E-07	0	●	7,88 E-06	0	●	7,45 E-01	55	=	3,52 E-05	0	○
Sphere (Küre) 30 b.	5,71 E-03	0	●	6,24 E+00	0	=	4,68 E+00	0	=	7,48 E-01	45	=	1,66 E-07	0	○	5,45 E-02	0	●	4,31 E-01	33	=	6,04 E-04	0	●	7,36 E-01	18	=
Styblinski–Tang 30 b.	3,57 E-03	6	●	8,63 E+00	0	=	6,08 E-01	122	○	1,80 E-04	101	●	4,73 E-01	4	=	7,06 E-02	0	●	2,81 E-02	0	●	5,01 E+00	0	=	1,20 E-06	0	○
Mishra'nın Kuşu (Mishra's Bird)	1,63 E-02	0	●	5,23 E-01	0	●	5,30 E-03	80	○	7,58 E-07	0	●	1,33 E-04	0	●	1,34 E-02	9	●	7,52 E-05	0	●	8,30 E+00	0	=	8,14 E+00	0	=
Simionescu	2,21 E-02	0	●	1,07 E-01	3	●	3,18 E-04	0	●	6,85 E-05	0	●	4,38 E-07	0	●	7,78 E-04	0	●	1,06 E-07	0	●	2,66 E-06	1	●	6,70 E-01	121	=
Townsend	1,10 E-03	0	●	3,46 E-06	0	●	6,25 E-05	0	●	6,54 E-01	0	●	4,71 E-07	1	●	7,37 E-04	0	●	7,72 E-06	0	●	7,55 E+00	0	=	2,36 E-04	0	○
Dönd. Cigar (R. Cigar) 30 b.	1,44 E-04	0	●	4,75 E-03	0	●	1,78 E+00	0	=	4,07 E-01	28	●	5,46 E-02	0	●	1,51 E-05	0	●	6,44 E-06	0	●	3,61 E-03	6	○	1,20 E-07	76	○
Dönd. Yük. Kond. Elip. (R. H. C. El.) 30 b.	5,08 E-03	0	●	5,31 E+00	0	=	4,75 E-05	6	○	4,83 E-05	0	●	1,61 E+00	0	=	3,23 E-01	0	●	3,13 E-04	0	●	3,25 E+00	0	=	6,36 E+00	0	●
Kayd. ve Dönd. Ackley (S. R. Ackley) 30 b.	6,26 E-06	0	●	2,42 E-01	2	=	3,77 E-01	7	=	1,75 E-05	1	●	2,57 E-06	0	●	7,50 E-05	68	●	3,06 E-03	1	●	6,65 E-04	101	○	6,14 E-02	0	○
Kayd. ve Dönd. Rast.(S. R. Rast.) 30 b.	5,70 E-01	44	=	4,78 E-06	4	○	6,57 E-06	0	○	1,37 E-04	1	●	3,02 E-04	0	●	8,07 E-01	0	●	6,03 E+00	0	=	1,28 E+00	0	=	3,48 E-01	61	=
Kayd. ve Dönd. Schw. (S. R. Schw.) 30 b.	6,41 E-02	3	●	6,77 E-01	77	=	8,40 E-03	0	●	7,04 E-01	0	●	2,15 E-01	33	=	4,26 E-06	0	●	6,43 E-01	14	=	8,40 E-07	0	●	8,45 E+00	0	=
● / = / ○	21/2/0			13/9/1			11/7/5			18/4/1			17/4/2			23/0/0			19/4/0			8/11/4			10/8/5		
($\alpha = 0,05$)																											

Çizelge 5. 4. Tek-amaçlı fonksiyonlar için BiGOA ve diğer algoritmalar arası Wilcoxon İşaretli Sıralama Testi bulguları (1. kısım).

Fonksiyon	BiGOA – BYAO			BiGOA – BYAO$_{V1}$			BiGOA – ÇTO			BiGOA – ÇTO$_{V1}$			BiGOA – DE			BiGOA – DE$_{V1}$			BiGOA – ERA			BiGOA – ERA$_{V1}$			BiGOA – GA		
	p	T	W	P	T	W	p	T	W	P	T	W	p	T	W	P	T	W	p	T	W	P	T	W	p	T	W
Beale	4,07 E-02	5	●	2,44 E-04	0	●	4,41 E-02	0	●	3,51 E-07	0	●	3,44 E-02	0	●	6,26 E-07	0	●	1,61 E-01	0	●	4,43 E-06	0	●	3,31 E-04	4	●
Booth	7,07 E-04	0	●	3,31 E-07	0	●	7,34 E-01	128	=	2,32 E+00	0	=	2,35 E-05	2	●	3,28 E-04	0	●	2,61 E-03	47	●	2,12 E-03	118	●	3,88 E-04	0	●
Bukin (no. 6)	1,27 E-07	1	●	7,88 E-07	0	●	7,02 E-02	0	=	7,51 E+00	0	=	3,22 E-02	0	●	4,32 E-04	161	●	4,85 E-07	0	●	5,83 E-05	0	●	4,51 E-01	0	●
Cross-in Tray	4,00 E-06	0	●	7,48 E-02	18	●	3,58 E-07	7	●	4,87 E-07	2	●	8,00 E-07	1	●	6,22 E-06	0	●	3,75 E-02	111	●	3,31 E-01	71	=	5,22 E-06	101	●
Easom	7,54 E-05	1	●	8,31 E-06	6	●	5,41 E-05	0	●	8,54 E+00	0	=	5,41 E-05	1	●	2,67 E-02	0	●	8,11 E-06	0	●	4,73 E-07	0	●	2,72 E-01	0	●
Eggholder	5,82 E-01	69	=	5,45 E+00	0	=	5,01 E-01	67	=	7,08 E+00	0	=	8,82 E-04	0	●	4,46 E-07	0	●	2,00 E-02	0	●	8,27 E-03	6	●	4,12 E-05	33	●
Levy (no. 13)	6,27 E-03	0	●	3,05 E-07	0	●	8,88 E+00	0	=	2,71 E+00	0	=	1,48 E-05	47	●	1,01 E-07	3	●	2,31 E-01	78	=	3,53 E-02	0	○	6,14 E-02	0	●
Matyas	1,00 E-05	5	●	3,58 E-06	0	●	2,71 E-01	1	●	2,55 E-03	1	●	5,76 E-06	0	●	3,17 E-05	0	●	7,86 E-05	0	●	1,25 E-03	0	●	7,14 E-02	1	●
McCormick	3,83 E-01	0	●	4,18 E-07	0	●	6,80 E+00	0	=	5,17 E-06	0	=	1,75 E-04	0	●	5,05 E-04	2	●	4,31 E-03	41	●	6,48 E-05	1	●	8,21 E-04	0	●
Schaffer (no. 2)	3,46 E-01	0	●	8,18 E-04	0	●	6,05 E-02	0	●	3,84 E+00	0	=	6,01 E-05	19	●	4,27 E-06	0	●	5,33 E-06	0	●	8,44 E-01	0	●	5,45 E-06	0	●
Ackley 30 b.	5,40 E-01	179	=	6,54 E-03	4	○	4,36 E-01	1	●	1,54 E-01	109	=	4,63 E-02	0	●	2,14 E+00	0	=	8,62 E-03	30	●	2,75 E-02	1	●	5,43 E-02	0	●
Rastrigin 30 b.	4,01 E-06	0	●	2,27 E-01	41	=	6,07 E-06	1	●	3,83 E+00	0	=	7,16 E-01	151	=	2,60 E-03	1	●	3,13 E-05	0	●	3,43 E-01	0	●	7,32 E-06	2	●
Rosenbrock 30 b.	2,56 E-05	0	●	7,60 E+00	0	=	2,16 E-01	59	=	4,63 E-07	0	○	3,83 E-03	0	●	3,70 E-01	0	●	6,38 E-05	0	●	6,50 E-03	87	○	4,00 E-05	0	●
Sphere (Küre) 30 b.	7,11 E-07	45	=	3,56 E-04	0	○	5,71 E-03	0	●	5,57 E-01	32	=	1,25 E-02	0	●	3,36 E-01	6	●	1,15 E-02	0	●	2,40 E-05	0	●	5,70 E-04	2	●
Styblinski–Tang 30 b.	1,48 E+00	0	=	8,04 E-07	0	●	2,44 E-07	0	●	1,83 E-05	0	●	3,08 E-03	0	●	6,44 E-04	0	●	2,77 E+00	0	=	8,08 E+00	0	=	8,24 E-07	0	●
Mishra'nın Kuşu (Mishra's Bird)	5,15 E-03	0	●	6,15 E-04	1	●	6,73 E-04	16	●	2,10 E-05	3	●	3,48 E-02	0	●	4,48 E-04	0	●	5,34 E-01	48	=	3,45 E+00	0	=	1,00 E-04	0	●
Simionescu	1,46 E-03	0	●	2,38 E-03	0	●	6,32 E-01	0	●	4,06 E+00	0	=	8,20 E+00	0	=	7,84 E-01	43	=	3,54 E+00	0	=	6,17 E-04	259	○	3,72 E-04	76	●
Townsend	5,77 E-06	6	●	7,76 E-01	0	●	7,66 E-03	0	○	1,35 E-04	0	○	4,17 E-01	0	●	3,36 E-03	0	●	8,56 E-05	0	●	6,27 E+00	0	=	8,40 E-01	0	●
Dönd. Cigar (R. Cigar) 30 b.	2,44 E+00	0	=	4,32 E-05	65	○	3,18 E-01	211	=	2,61 E-04	154	○	1,04 E-06	5	●	7,55 E-07	0	●	7,22 E-07	1	●	7,87 E-02	7	●	1,10 E-07	2	●
Dönd. Yük. Kond. Elip. (R. H. C. El.) 30 b.	8,08 E+00	0	=	4,03 E-05	0	○	3,50 E-03	0	●	1,34 E+00	0	=	6,07 E-05	0	●	4,24 E-01	190	=	6,83 E-01	1	●	2,18 E-02	0	●	1,53 E-03	0	●
Kayd. ve Dönd. Ackley (S. R. Ackley) 30 b.	2,37 E-02	0	●	2,86 E-02	0	●	4,71 E-03	0	○	2,32 E-06	173	○	7,07 E-06	0	●	8,40 E+00	0	=	3,71 E-03	0	○	4,00 E-06	0	○	1,54 E-04	1	●
Kayd. ve Dönd. Rast.(S. R. Rast.) 30 b.	7,54 E+00	0	=	7,22 E-05	232	○	2,14 E-01	51	=	4,40 E-01	0	●	5,27 E-03	13	●	7,10 E+00	0	=	7,87 E-01	0	●	7,21 E-07	0	●	4,67 E-03	0	●
Kayd. ve Dönd. Schw. (S. R. Schw.) 30 b.	3,28 E-02	0	●	5,56 E+00	0	=	2,57 E+00	0	=	2,27 E-01	0	○	4,45 E-02	0	●	5,43 E-05	0	●	1,05 E-07	0	●	1,26 E-07	34	○	6,86 E-05	0	●
● / = / ○	16/7/0			14/4/5			12/9/2			6/12/5			21/2/0			18/5/0			18/4/1			14/4/5			23/0/0		
($\alpha = 0,05$)																											

Çizelge 5. 5. Tek-amaçlı fonksiyonlar için BiGOA ve diğer algoritmalar arası Wilcoxon İşaretli Sıralama Testi bulguları (2. kısım).

Fonksiyon	BiGOA – GA_{V1}			BiGOA – GKA			BiGOA – GKA_{V1}			BiGOA – KSA			BiGOA – KSA_{V1}			BiGOA – PSO			BiGOA – PSO_{V1}			BiGOA – YAK			BiGOA – YAK_{V1}		
	P	T	W	p	T	W	P	T	W	p	T	W	P	T	W	P	T	W	P	T	W	p	T	W	P	T	W
Beale	7,14 E-01	0	●	1,80 E-06	54	●	3,86 E-02	0	●	7,65 E-01	0	●	8,58 E-05	0	●	5,41 E-01	0	●	8,55 E-01	0	●	1,06 E-05	0	●	8,10 E-03	0	●
Booth	2,78 E-06	0	●	4,64 E-05	0	●	5,57 E-07	41	●	4,16 E-04	0	●	6,50 E-03	0	●	1,15 E-04	6	●	2,83 E-06	0	●	1,14 E-01	76	=	2,32 E-02	0	●
Bukin (no. 6)	7,26 E-03	0	●	2,12 E-01	175	=	3,53 E-04	0	●	6,32 E-01	91	=	6,45 E-01	78	=	1,14 E-05	0	●	2,47 E-05	0	●	7,76 E-07	0	○	3,08 E-06	0	○
Cross-in Tray	2,58 E-06	0	●	4,46 E-04	0	●	5,60 E-05	0	○	7,14 E-03	0	●	5,71 E-03	0	●	1,27 E-06	4	●	5,32 E-06	1	●	4,82 E-02	0	○	2,23 E-05	6	●
Easom	6,67 E-05	4	●	5,55 E-01	0	●	3,81 E+00	0	=	4,38 E-01	0	●	3,28 E-04	0	●	5,83 E-01	0	●	8,67 E-07	1	●	3,72 E-01	0	=	7,80 E-03	0	●
Eggholder	6,60 E-03	4	●	1,34 E-01	1	●	5,48 E-03	0	●	5,33 E-07	7	●	8,25 E-01	0	●	4,42 E-02	0	●	2,77 E-01	0	●	7,04 E-01	0	=	7,78 E-07	0	●
Levy (no. 13)	4,78 E-06	0	●	5,41 E-02	0	●	5,05 E-02	7	●	1,73 E+00	0	=	1,83 E-04	113	○	4,22 E-07	8	●	8,05 E-03	6	●	2,38 E-02	0	●	4,22 E-01	54	=
Matyas	7,74 E+00	0	=	6,61 E-04	0	●	8,04 E-07	0	●	6,36 E-01	0	●	6,40 E-03	42	●	4,56 E-07	1	●	1,64 E-05	0	●	3,34 E+00	0	=	7,82 E+00	0	=
McCormick	6,57 E+00	0	=	3,27 E-02	0	●	2,23 E-07	1	●	5,25 E-07	4	●	1,64 E-05	0	●	6,04 E-01	0	●	3,88 E-03	0	●	3,34 E-03	7	●	1,43 E-06	0	●
Schaffer (no. 2)	5,23 E+00	0	=	4,37 E-05	1	●	4,54 E-03	0	●	6,13 E-01	6	●	3,83 E-03	5	●	7,12 E-06	0	●	2,00 E-07	0	●	8,03 E-07	1	●	3,00 E-04	0	○
Ackley 30 b.	6,71 E-07	11	●	3,88 E+00	0	=	7,87 E+00	0	=	1,06 E+00	0	=	7,22 E-07	0	○	2,05 E-05	0	●	4,63 E-01	211	=	4,41 E-01	0	●	1,30 E-07	1	●
Rastrigin 30 b.	7,12 E+00	0	=	4,84 E-01	0	●	4,52 E+00	43	=	3,51 E-06	79	○	1,42 E-02	0	○	6,50 E-05	0	●	3,02 E-04	0	●	2,35 E-06	0	○	4,47 E-04	4	●
Rosenbrock 30 b.	3,08 E-03	0	●	6,61 E+00	0	=	6,63 E-02	0	○	8,73 E+00	0	=	5,27 E-04	7	●	4,37 E-05	4	●	1,17 E-06	0	●	1,78 E+00	0	=	3,16 E-04	0	○
Sphere (Küre) 30 b.	2,17 E-02	62	●	7,76 E+00	0	=	5,27 E-03	0	○	6,77 E+00	0	=	1,08 E-07	0	○	2,22 E-01	0	●	4,53 E-01	141	=	2,62 E-03	0	●	6,86 E+00	0	=
Styblinski–Tang 30 b.	3,15 E-02	0	●	6,14 E+00	0	=	7,37 E-06	0	○	5,43 E-03	51	●	8,84 E+00	0	=	3,58 E-06	0	●	7,48 E-03	0	●	8,75 E-01	33	○	3,45 E-05	65	○
Mishra'nın Kuşu (Mishra's Bird)	1,52 E-06	0	●	7,06 E-01	0	●	2,78 E-04	0	○	8,21 E-02	0	●	5,44 E-07	0	○	8,66 E-06	0	●	5,04 E-05	11	●	6,14 E+00	0	=	6,31 E-05	0	○
Simionescu	5,77 E-04	0	●	6,58 E-05	0	●	5,63 E-02	1	●	4,00 E-01	0	●	5,38 E-03	0	●	5,81 E-05	166	●	6,52 E-01	0	●	5,06 E+00	0	=	6,24 E+00	0	=
Townsend	1,42 E-05	6	●	2,44 E-07	9	●	1,26 E-01	0	●	8,41 E-01	0	●	2,67 E-07	1	●	4,67 E-02	0	●	3,32 E-03	7	●	7,21 E+00	0	=	5,43 E-02	188	○
Dönd. Cigar (R. Cigar) 30 b.	5,23 E+00	0	=	2,66 E-05	0	●	8,28 E+00	0	=	3,44 E-07	0	●	7,28 E-06	0	●	3,01 E+00	0	=	3,42 E+00	0	=	2,45 E-02	127	○	4,81 E-02	6	○
Dönd. Yük. Kond. Elip. (R. H. C. El.) 30 b.	6,80 E-02	0	●	6,15 E-04	0	●	4,15 E-06	88	○	1,84 E-07	0	●	7,80 E+00	0	=	2,47 E-02	0	●	2,46 E-07	0	●	4,01 E+00	0	=	3,18 E+00	0	=
Kayd. ve Dönd. Ackley (S. R. Ackley) 30 b.	5,33 E-07	0	●	2,01 E+00	0	=	5,84 E+00	0	=	8,77 E-03	0	●	6,67 E-04	0	●	8,00 E-05	56	●	4,45 E-06	0	●	6,82 E-05	0	○	8,64 E-07	112	○
Kayd. ve Dönd. Rast.(S. R. Rast.) 30 b.	7,24 E+00	0	=	3,54 E-01	98	○	5,01 E-04	56	○	6,16 E-04	126	●	8,34 E-07	8	●	2,73 E-06	0	●	7,12 E-02	39	○	3,25 E+00	0	=	1,82 E+00	0	=
Kayd. ve Dönd. Schw. (S. R. Schw.) 30 b.	6,50 E-07	0	●	4,18 E+00	0	○	1,64 E-04	0	○	7,76 E+00	0	=	4,17 E-05	0	○	4,45 E-06	0	●	5,76 E+00	0	=	6,01 E-01	0	●	2,35 E+00	0	=
● / = / ○	17/6/0			15/6/2			10/5/8			16/6/1			14/3/6			22/1/0			18/4/1			7/10/6			8/7/8		
(α = 0,05)																											

Çizelge 5.2., Çizelge 5.3., Çizelge 5.4. ve Çizelge 5.5.'teki bulguları şu şekilde yorumlayabiliriz:

- GOA, 10 kısıtsız test fonksiyon için rakip durumdaki YAK_{v1}, $ÇTO_{v1}$ ve GKA_{v1} ile karşılıklı üstünlükler sergilemekte, bunun dışında diğer algoritmalara belirgin bir üstünlüğe sahip olmaktadır. Diğer yandan, Çizelge 5.4. ve Çizelge 5.5.'deki bulgulara göre, BiGOA da GOA gibi yine benzer algoritmalar ile (YAK_{v1}, $ÇTO_{v1}$ ve GKA_{v1} gibi) mücadele içindedir. Genel durumu GOA kadar iyi olmasa da, bu algoritma da üst sıralarda bir konuma yerleşmektedir.
- 3 kısıtlı tek-amaçlı fonksiyon için GOA'nın, YAK, YAK_{v1}, GKA_{v1} ve ERA_{v1} dışındaki diğer algoritmalara genelde üstünlük sağladığı anlaşılmaktadır. BiGOA ise, GOA kadar başarılı olamasa da (Örneğin, YAK, YAK_{v1}'e belirgin üstünlüğü yoktur), yine belirli bir kesim algoritmadan daha iyi sonuçlar üretmiştir.
- 5 tek-amaçlı çok-boyutlu fonksiyon kapsamında GOA, $ÇTO_{v1}$, GKA_{v1} ve KSA dışındaki algoritmalara karşı genellikle iyi performanslar göstermiş, BiGOA'nın ise, GKA_{v1}, $BYAO_{v1}$, YAK ve $ÇTO_{v1}$ algoritmaları ile rekabette olduğu görülmüştür.
- IEEE CEC 2015 etkinliğinden seçilen 5 çok-boyutlu fonksiyon için GOA diğer algoritmaların çoğunluğuna üstünlük sağlamış, ancak GKA_{V1} ve YAK_{V1} algoritmalarıyla da oldukça yakın rekabet içerisine girmiştir. Genel çerçevede GOA'nın takipçisi durumunda olan BiGOA ise, $ÇTO_{v1}$, GKA_{V1}, YAK ve YAK_{v1} dışındaki diğer algoritmalara genellikle üstünlük sağlamıştır.

Değerlendirmeye alınan tek-amaçlı fonksiyonlar bağlamında, sürece dâhil bütün algoritmaların elde ettiği OHK, OHKK, ve OMHY değerleri Çizelge 5.6.'da verilmiştir.

Çizelge 5.6., takip eden sayfadadır.

Çizelge 5. 6. Tek-amaçlı fonksiyonlar için OHK, OHKK, ve OMHY bulguları.

Alg. – Tek.	OHK				OHKK				OMHY			
	Test Fonksiyonu Grubu				Test Fonksisyonu Grubu				Test Fonksiyonu Grubu			
	10 kısıtsız	3 Kısıtlı	5 kısıtsız çok-amaç	5 IEEE CEC '15	10 kısıtsız	3 Kısıtlı	5 kısıtsız çok-amaç	5 IEEE CEC '15	10 kısıtsız	3 Kısıtlı	5 kısıtsız çok-amaç	5 IEEE CEC '15
GOA	**5,173E+13**	2,331E+16	**3,701E+18**	5,282E+19	**7,269E+05**	4,571E+07	**2,630E+06**	3,127E+06	**4,037E+05**	6,068E+07	**4,006E+05**	6,097E+04
BiGOA	7,701E+14	5,011E+16	6,119E+18	5,441E+20	5,536E+07	3,389E+08	6,329E+08	4,059E+07	6,460E+08	8,220E+08	3,463E+06	5,380E+05
BYAO	8,514E+15	2,533E+17	9,833E+18	6,068E+21	3,332E+08	1,709E+09	7,722E+09	6,690E+08	6,609E+08	3,083E+09	7,044E+07	2,648E+05
BYAOv1	6,504E+15	9,201E+16	9,047E+18	6,631E+20	4,726E+08	7,636E+08	8,481E+08	5,331E+07	4,261E+08	2,803E+08	3,261E+06	6,350E+05
ÇTO	9,511E+14	8,255E+16	8,331E+18	3,306E+20	7,824E+07	7,378E+08	7,208E+09	4,631E+09	7,127E+07	1,067E+08	7,604E+08	7,929E+08
ÇTOv1	5,419E+14	7,110E+16	7,580E+18	3,292E+20	6,450E+07	4,445E+07	2,392E+08	4,492E+07	4,700E+07	6,351E+06	5,210E+09	3,403E+03
DE	8,361E+15	1,701E+17	3,740E+20	7,607E+10	7,242E+08	3,290E+08	1,468E+11	8,717E+08	9,886E+08	5,949E+09	8,030E+08	6,145E+06
DEv1	1,233E+15	1,098E+17	2,619E+20	8,210E+10	2,541E+08	2,189E+07	6,500E+11	9,323E+08	7,042E+07	8,610E+08	5,706E+07	7,261E+07
ERA	8,677E+14	9,711E+16	4,801E+19	5,604E+20	4,707E+07	5,576E+08	2,614E+09	7,735E+07	5,122E+07	4,120E+08	7,921E+09	2,470E+04
ERAv1	8,557E+14	9,611E+16	3,097E+19	6,064E+20	5,830E+07	6,752E+07	5,340E+07	5,186E+07	4,288E+06	1,037E+06	2,346E+09	3,595E+05
GA	7,422E+16	5,099E+18	8,055E+21	8,901E+22	6,534E+09	3,400E+09	1,436E+12	4,236E+11	3,870E+08	3,292E+08	4,470E+10	6,646E+07
GAv1	4,111E+16	9,807E+17	4,474E+20	8,521E+21	3,262E+09	5,449E+08	2,887E+10	3,453E+10	2,205E+08	2,353E+09	3,702E+09	4,963E+08
GKA	3,190E+15	8,527E+16	9,802E+18	5,189E+20	2,305E+07	4,611E+08	7,753E+09	6,032E+08	1,337E+07	1,659E+08	1,989E+06	5,143E+06
GKAv1	5,809E+13	6,710E+15	7,011E+18	4,210E+19	5,530E+06	2,015E+07	5,282E+08	5,325E+07	1,710E+06	1,047E+09	6,452E+05	5,105E+04
KSA	5,173E+13	8,288E+17	2,990E+19	5,305E+20	5,551E+07	5,860E+08	6,448E+08	2,647E+08	3,607E+07	6,904E+07	5,207E+09	3,773E+05
KSAv1	8,249E+14	8,520E+17	3,113E+19	5,083E+20	6,460E+07	4,764E+08	5,070E+08	3,499E+08	2,514E+07	3,816E+06	5,474E+09	3,609E+06
PSO	3,190E+16	2,333E+18	2,589E+20	7,807E+23	2,421E+08	1,105E+09	3,260E+10	7,264E+11	8,426E+09	7,137E+07	8,730E+10	6,460E+09
PSOv1	1,056E+16	9,361E+17	5,188E+19	6,710E+22	2,277E+08	8,296E+09	4,191E+07	8,055E+10	7,330E+08	4,170E+08	8,402E+09	4,351E+08
YAK	6,183E+13	**3,047E+15**	8,817E+18	4,504E+20	5,422E+06	**2,338E+07**	5,854E+07	3,820E+07	3,248E+07	**2,059E+05**	3,066E+07	6,014E+05
YAKv1	6,013E+13	3,062E+16	6,407E+18	**3,349E+19**	5,349E+06	2,670E+08	8,640E+06	**5,261E+04**	2,267E+06	2,892E+06	2,829E+05	**2,139E+03**

Çizelge 5.6. kapsamındaki bulgulara göre şu yorumları yapabiliriz:

- 10 kısıtsız fonksiyon için GOA bütün hata değerlerinde en iyi durumda iken, BiGOA ise ilk beş içerisinde kendisine yer edinmiştir.
- 3 kısıtlı fonksiyon için hata değerleri YAK ve ardından GKA_{V1} lehineyken, GOA da bu algoritmalara yakın bir konuma yerleşmiş, BiGOA da ilk beş arasında konumunu almıştır.
- 5 kısıtsız tek-amaçlı çok-boyutlu fonksiyonlarda GOA en iyi performansı göstermiş, BiGOA da bu algoritmayı takip etmek üzere, YAK_{V1}, GKA_{V1}, ve $ÇTO_{V1}$ gibi algoritmalarla üst sıralarda yerini almıştır.
- IEEE CEC 2015 5 çok-boyutlu fonksiyon için YAK_{V1}'ın performansı diğer algoritmalara göre en iyi durumda iken, bu algoritmayı GKA_{V1}, GOA, BiGOA, ve $ÇTO_{V1}$ algoritmaları ve geriye kalanlar izlemiştir.

Buradaki bulgunun, Wilcoxon İşaretli Sıralama Testi'ndeki GOA üstünlüğünden farklı olmasının sebebi, teke tek karşılaştırmalarda GOA'nın başarı durumlarının YAK_{V1}'dan nispeten daha iyi olması gösterilebilir. Ancak IEEE CEC 2015 fonksiyonlarına karşı başarım sıralamalarında YAK_{V1}'ın istikrarı GOA'dan daha iyi olmuştur. Teke tek karşılaştırmalar dikkate alınan algoritmalar sebebiyle esnek olduğundan, fonksiyon başarımı bulgusu daha anlamlı olarak kabul edilebilir.

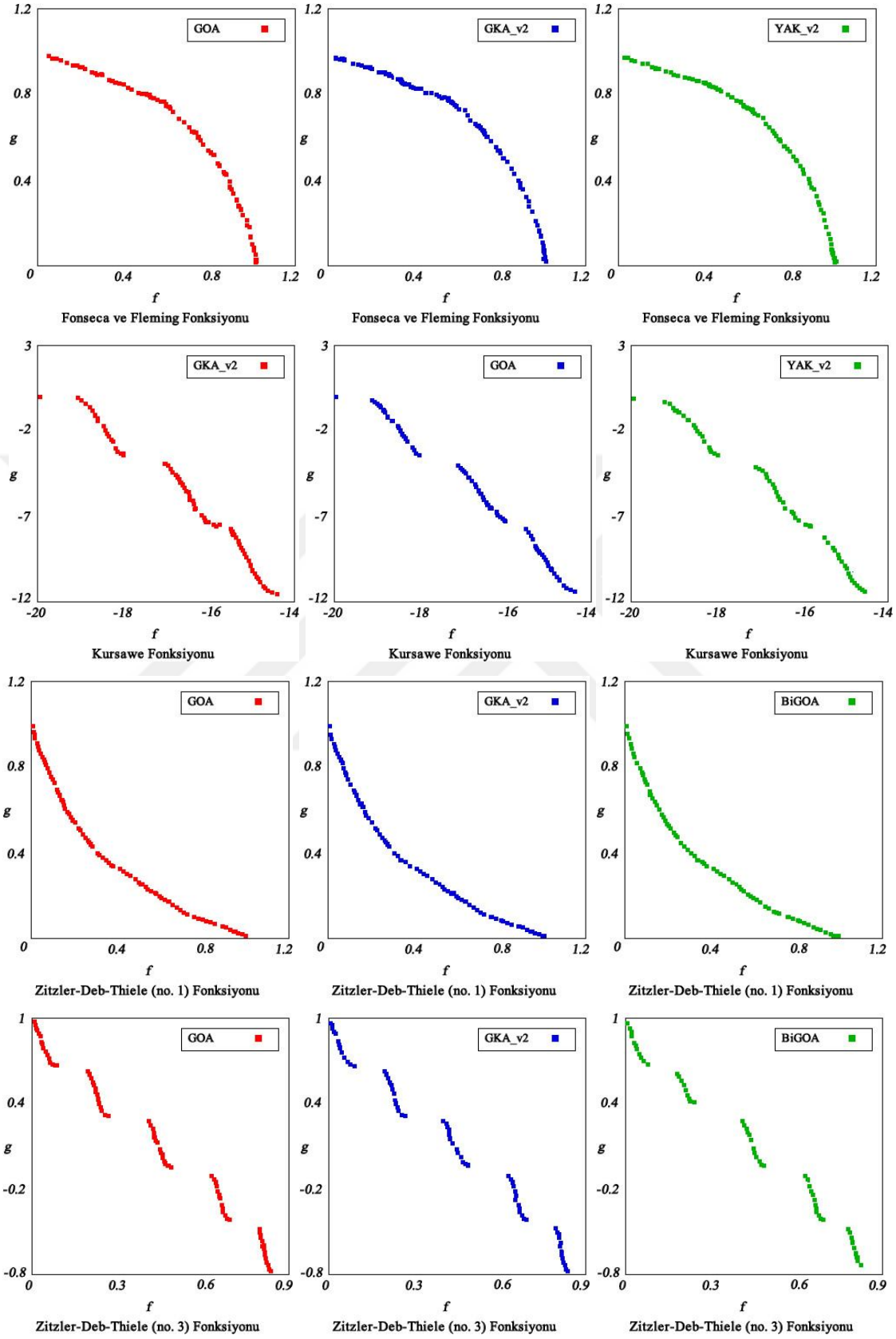
Test fonksiyonları olarak değerlendirme sürecine konu olan son fonksiyonlar da çok-amaçlı fonksiyon grubuna girmektedir. Kendi içerisinde kısıtlı ve kısıtsız olarak da ayrılan bu fonksiyonlar, GOA ve BiGOA başta olmak üzere, karşılaştırmaya tabi diğer algoritmaların ikinci varyasyonları ile birlikte yorumlanmıştır. Sayısal bulgular ile birlikte pareto optimumlara yönelik elde edilen grafikler de algoritmaların – tekniklerin başarımları konusunda bizlere etkin fikir verebilmektedir (Yine gerçek yaşam tabanlı uygulamalardan birisinde izlenmiş olan hata oranları değerleri kullanımı da bu yönde işe yaramaktadır). Metnin karmaşık hale gelerek okunurluğu olumsuz etkilememesi adına, her bir çok-amaçlı fonksiyon için en iyi başarımı – performansı gösteren algoritmalar daha çok dikkate alınmıştır. Bu bağlamda, Çizelge 5.7.'de her bir çok-amaçlı fonksiyon için en iyi üç performansı gösteren algoritma belirtilmiş; Şekil 5.2.'de ise, ilgili algoritmalar için elde edilen pareto optimum grafikleri sunulmuştur.

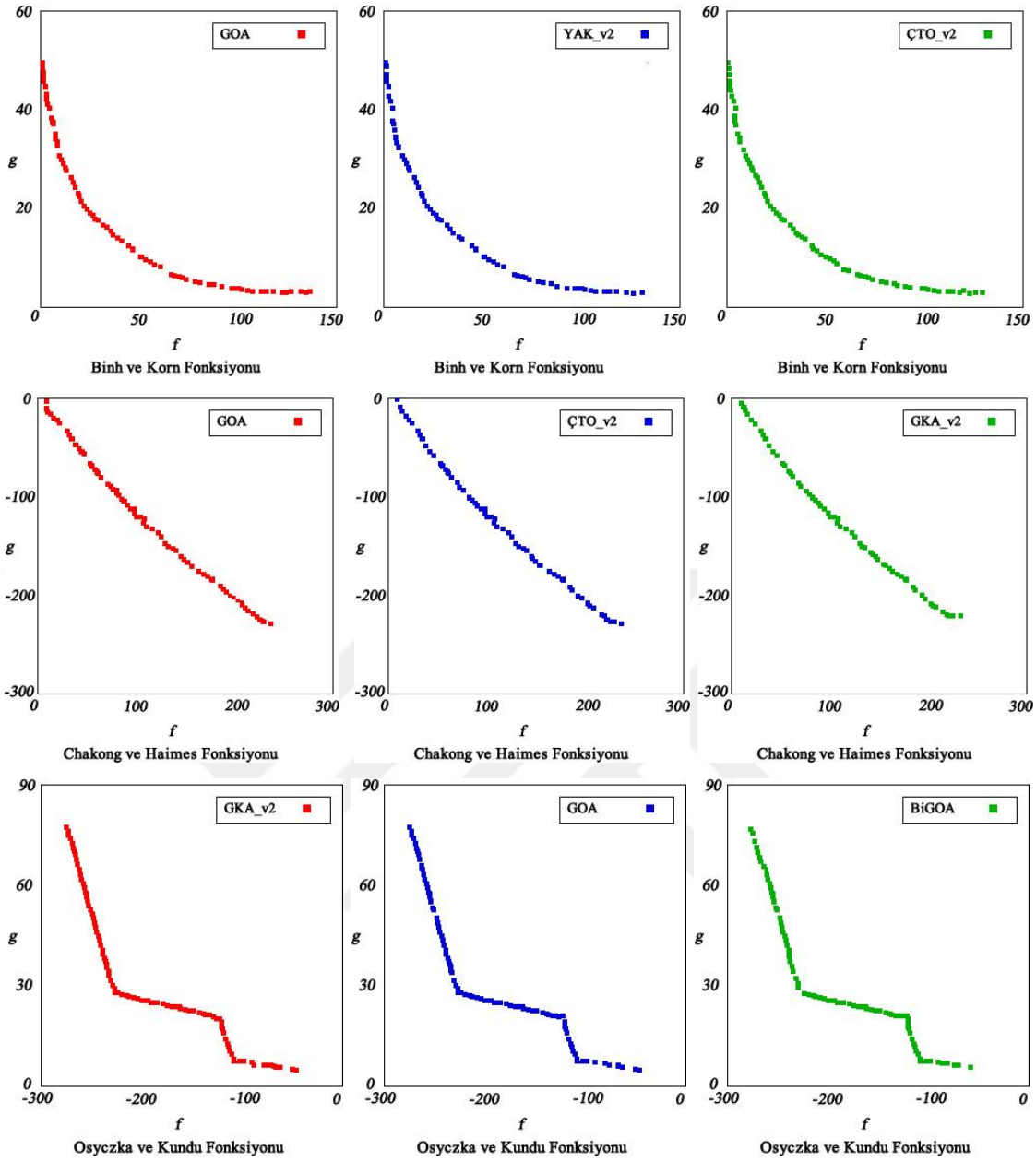
Çizelge 5. 7. Çok-amaçlı fonksiyonlarda en iyi sonuç veren ilk üç algoritma.

Fonseca ve Fleming Fonk.		**Kursawe Fonk.**	
1. Algoritma	GOA	**1. Algoritma**	GKA_{V2}
2. Algoritma	GKA_{V2}	**2. Algoritma**	GOA
3. Algoritma	YAK_{V2}	**3. Algoritma**	YAK_{V2}
Zitzler-Deb-Thiele (no. 1) Fonk.		**Zitzler-Deb-Thiele (no. 3) Fonk.**	
1. Algoritma	GOA	**1. Algoritma**	GOA
2. Algoritma	GKA_{V2}	**2. Algoritma**	GKA_{V2}
3. Algoritma	BiGOA	**3. Algoritma**	BiGOA
Binh ve Korn Fonk.		**Chakong ve Haimes Fonk.**	
1. Algoritma	GOA	**1. Algoritma**	GOA
2. Algoritma	YAK_{V2}	**2. Algoritma**	$ÇTO_{V2}$
3. Algoritma	$ÇTO_{V2}$	**3. Algoritma**	GKA_{V2}

Osyczka ve Kundu Fonksiyonu	
1. Algoritma	GKA_{V2}
2. Algoritma	GOA
3. Algoritma	BiGOA

Şekil 5.2. *Başlangıcı*





Şekil 5. 2. 7 çok-amaçlı fonksiyonda en iyi üç algoritmanın pareto optimumları.

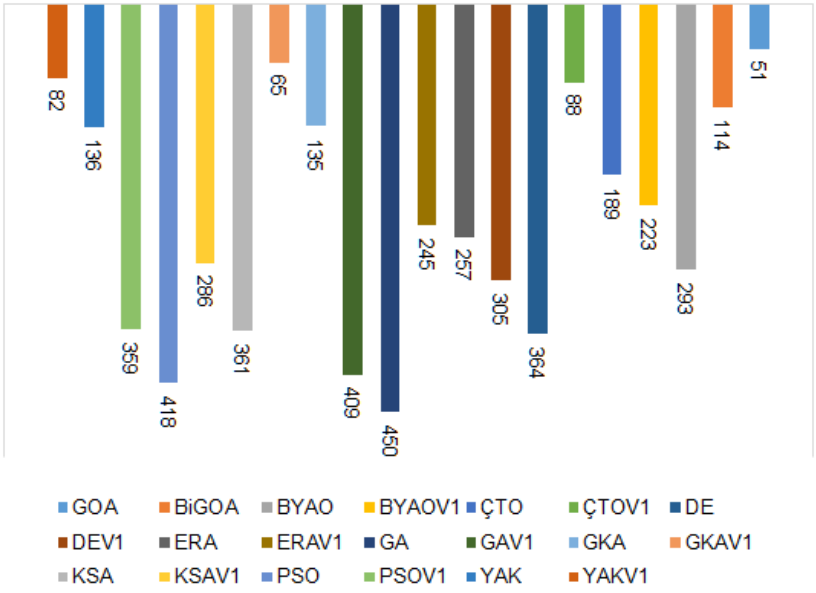
Şekil 5.2.'de, bulunmuş olan pareto optimumlardaki eksiklik ve / veya kaymalara göre algoritma başarımlarını görebiliriz. GOA çok-amaçlı fonksiyonlarda dikkate değer başarı gösterirken, BiGOA da bazılarında ilk üçe girmiştir.

GPSD değerlerinin (GPSD'ye dair bilgi için *Bkz.* üçüncü bölüm, alt-başlık 3.7.) hesaplanması aşamasında dikkate alınan ve test fonksiyonlarının çözümlenmesi süreci sonrasında oluşan algoritmalar – teknikler arası başarı sıralamaları ve elde edilen GPSD değerleri Çizelge 5.8.'de gösterilmiştir.

Çizelge 5. 8. Test fonksiyonları kapsamındaki değerlendirmede algoritmalar arası başarı sıralamaları ve GPSD değerleri.

Fonksiyon	Algoritma / Başarı Sırası																			
	GOA	BiGOA	BYAO	$BYAO_{VI}$	ÇTO	$ÇTO_{VI}$	DE	DE_{VI}	ERA	ERA_{VI}	GA	GA_{VI}	GKA	GKA_{VI}	KSA	KSA_{VI}	PSO	PSO_{VI}	YAK	YAK_{VI}
Beale	1	6	13	7	9	5	16	14	10	11	20	18	3	2	15	12	19	17	8	4
Booth	1	7	12	9	10	2	15	13	8	11	20	16	4	3	17	14	19	18	6	5
Bukin (no. 6)	3	4	13	11	8	1	20	15	12	9	19	18	7	2	14	10	17	16	5	6
Cross-in Tray	3	2	14	7	10	5	19	17	9	13	20	16	6	4	12	11	18	15	8	1
Easom	3	2	10	12	11	5	13	9	8	15	20	19	7	1	17	14	18	16	6	4
Eggholder	3	2	12	8	9	4	14	13	11	10	20	19	5	1	18	16	17	15	7	6
Levy (no. 13)	2	7	14	8	12	1	18	13	9	16	19	17	6	4	20	10	15	11	5	3
Matyas	1	2	11	8	10	5	15	12	9	16	20	18	6	4	17	19	14	13	7	3
McCormick	2	1	13	6	9	4	15	14	10	11	20	19	7	5	16	12	18	17	8	3
Schaffer (no. 2)	1	6	10	11	7	5	19	16	9	12	20	18	3	2	14	13	17	15	8	4
Ackley 30 b.	3	4	15	11	8	2	14	12	13	9	19	16	6	1	18	10	20	17	7	5
Rastrigin 30 b.	4	8	12	9	7	3	14	13	11	10	19	18	2	1	20	16	17	15	5	6
Rosenbrock 30 b.	3	5	11	7	10	1	15	17	9	12	20	16	4	2	14	13	19	18	6	8
Sphere (Küre) 30 b.	1	3	12	11	7	2	14	10	13	8	19	18	6	5	16	15	20	17	9	4
Styblinski–Tang 30 b.	2	5	11	10	9	4	15	14	13	12	20	17	6	1	16	8	19	18	7	3
Mishra'nın Kuşu (Mishra's Bird)	2	8	10	13	6	5	16	12	11	9	20	19	4	3	18	14	17	15	7	1
Simionescu	2	7	12	10	6	1	18	16	14	9	20	19	8	3	13	11	17	15	4	5
Townsend	1	6	13	11	8	4	17	9	12	10	19	18	7	2	15	14	20	16	5	3
Dönd. Cigar (R. Cigar) 30 b.	1	8	15	12	3	6	16	14	13	5	19	18	9	7	17	10	20	11	4	2
Dönd. Yük. Kon. Elip. (R. H. C. El.) 30 b.	4	8	16	11	7	6	14	13	15	9	18	17	5	1	12	10	20	19	3	2
Kayd. ve Dönd. Ackley (S. R. Ac.) 30 b.	1	6	16	12	8	5	15	14	13	9	20	18	7	3	11	10	19	17	4	2
Kayd. ve Dönd. Rast.(S. R. Rast.) 30 b.	3	4	15	13	7	5	16	10	14	9	19	18	8	6	17	12	20	11	2	1
Kayd. ve Dönd. Schw. (S. R. Sc.) 30 b.	4	3	13	6	8	7	16	15	11	10	20	19	9	2	14	12	18	17	5	1
ELDE EDİLEN TOPLAM GPSD	*51*	*114*	*293*	*223*	*189*	*88*	*364*	*305*	*257*	*245*	*450*	*409*	*135*	*65*	*361*	*286*	*418*	*359*	*136*	*82*

İlgili GPSD değerleri Şekil 5.5.'te de grafiksel olarak verilmiştir.



Şekil 5. 3. Test fonksiyonları kapsamında değerlendirilen algoritmaların GPSD grafiği.

Çizelge 5.8. ve Şekil 5.3.'ten anlaşılacağı üzere, bu tez çalışması kapsamında gerçekleştirilen GOA ve BiGOA algoritmaları literatüre başarılı bir giriş yapmaktadır. GOA genel anlamda diğer algoritmalardan daha iyi durumda iken, BiGOA ise üst sıralarda kendine yer bulmaktadır. Ayrıca Çizelge'den de anlaşılacağı üzere, SZ alanı dâhilinde zaman içerisinde geliştirilen her yeni algoritma, istisnai durumlar haricinde önceki algoritmalar – tekniklerden daha iyi konuma yerleşmektedir. İstisnai durumlar ise genelde yeni geliştirilen kimi algoritmalardan halen iyi performans gösterebilen ve algoritmik yapısı iyi tasarlanmış SZ algoritmaları – teknikleri ile alakalıdır.

GPSD değerlendirmesinde ilk on içerisine giren algoritmalar – teknikler, GOA ve BiGOA'nın YSA ve ASBÇS'nin eğitimi amaçlı, 'hibrit kullanım' süreçlerinde de karşılaştırma yaklaşımına dâhil edilmiştir.

5.2. Gerçek Yaşam Tabanlı Problem ve Uygulamalara Yönelik Bulgular

Bu alt-bölümde, GOA ve BiGOA'nın, ifade edilen gerçek yaşam tabanlı problem ve uygulamalar kapsamında değerlendirilmesiyle elde edilen bulgular sunulmuştur.

GOA ve BiGOA'nın gerçek yaşam tabanlı olarak kullanıldığı problemlerden ilki 'Portföy Seçimi' olmuştur. Bu problemde her iki algoritmanın değerlendirilmesi sürecinde GPSD'de ilk on içerisinde yer alan algoritmalar – teknikler dikkate alınmış ve böylece bir bakıma GOA ve BiGOA'nın değerlendirilmesinde 8 farklı tekniğin sonuçları da kullanılmıştır. Uygulamada optimizasyon sürecine yönelik veriler, Markowitz (1959) tarafından literatüre kazandırılan, yine Xu ve ark. (2007) tarafından da dikkate alınan, 1937-1954 yılları arası çeşitli şirketlerin verileri ile bağlantılı olmuştur. Daha detaylı ifade edilecek olursa, American Tobacco, AT&T, United States Steel, General Motors, Atchison, Topeka and Santa Fe, Coca-Cola, Borden, ve Firestone and Sharon Steel adlı dokuz şirketin hisse senedinin ifade edilen yıllar dahilinde yıllık net dönüş değerleri (Markowitz, 1959; Xu ve ark., 2007) optimizasyon probleminin tabanını oluşturmuştur.

Xu ve ark. (2007) tarafından kabul edilen $dön_0$=0,08 olmak üzere GPSD'de ilk ona giren algoritmaların – tekniklerin 30 kez çalıştırılması sonucunda elde edilen optimum değerlerin ortalamasına yönelik bulgular Çizelge 5.9.'da gösterilmiştir.

Çizelge 5. 9. Portföy Seçimi problemi – uygulaması için elde edilen bulgular.

GOA	**GKA_{V1}**	**YAK_{V1}**	**$ÇTO_{V1}$**	**BiGOA**	**GKA**	**YAK**	**ÇTO**	**$BYAO_{V1}$**	**ERA_{V1}**
12,878E-03	13,013E-03	12,976E-03	13,844E-03	13,293E-03	13,366E-03	13,310E-03	14,018E-03	14,911E-03	14,557E-03

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Çizelge 5.9.'daki bulgulardan, ilgili algoritmaların – tekniklerin, birbirlerine yakın ortalama değerler elde ettiğini ve (Xu ve ark., 2007)'deki ortalama 13,084E-03 değerine yakın sonuçlar ortaya koyduğunu ifade edebiliriz. GOA en iyi bulguya ulaşırken, BiGOA en iyi dördüncü değere ulaşmıştır.

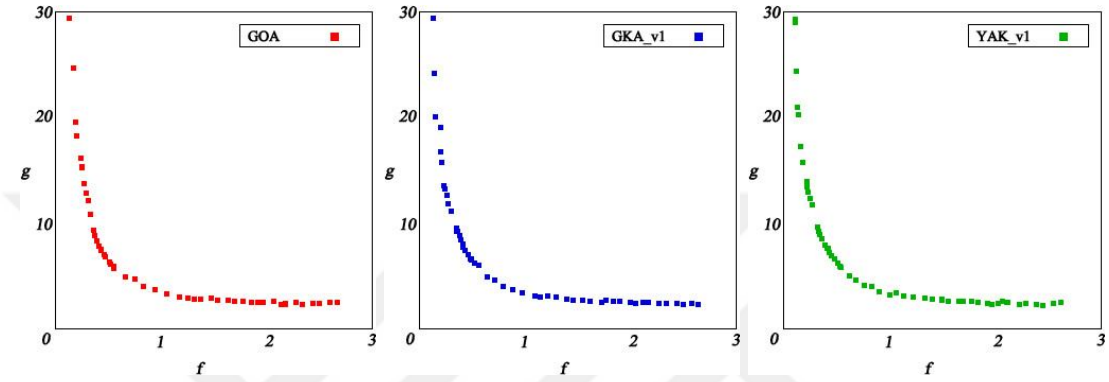
İkinci gerçek yaşam tabanlı problem – uygulama olan 'Disk Freni Tasarımı' problemi ile bir tür tasarım optimizasyonu süreci ele alınmış ve bu kapsamda, GOA ve BiGOA'nın da dâhil olduğu GPSD'deki en iyi on algoritma – tekniğe yönelik bulgular gözlemlenmiştir. Her bir algoritma 10 kez çalıştırılmış ancak bu sefer (Yang ve Deb, 2013)'deki gibi, hata değerleri ve pareto optimum grafikleri üzerinden değerlendirmeler yapılmıştır. Çizelge 5.10., hata değerleri ortalamasına yönelik bulguları göstermektedir.

Çizelge 5. 10. Disk Freni Tasarımı problemi – uygulaması için elde edilen bulgular.

GOA	GKA_{V1}	YAK_{V1}	$ÇTO_{V1}$	BiGOA	GKA	YAK	ÇTO	$BYAO_{V1}$	ERA_{V1}
2,521E -33	1,190E -30	3,274E -29	2,211E -23	2,062E -24	1,284E -18	2,759E -15	2,305E -13	1,807E -10	1,216E -12

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Şekil 5.4., elde edilen hata değerlerine eşlik etmek üzere, en iyi performans gösteren ilk üç algoritmanın – tekniğin pareto optimumlar grafiğini göstermektedir.

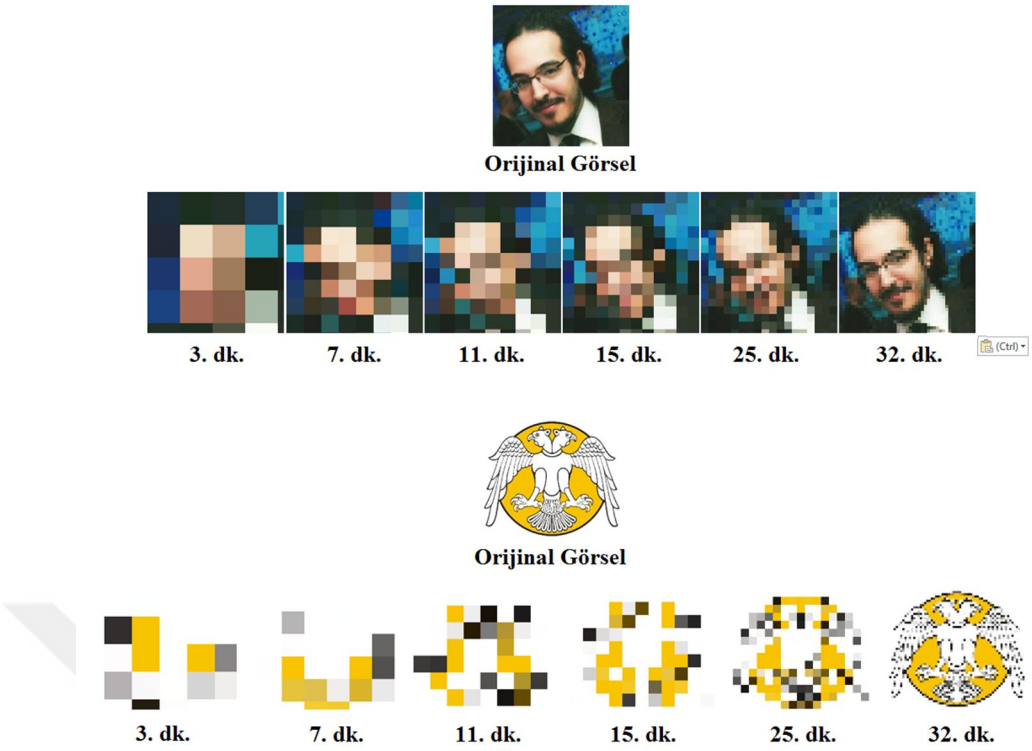


Şekil 5. 4. Disk Freni Tasarımı problemi – en iyi üç algoritmayla elde edilen pareto optimumlar.

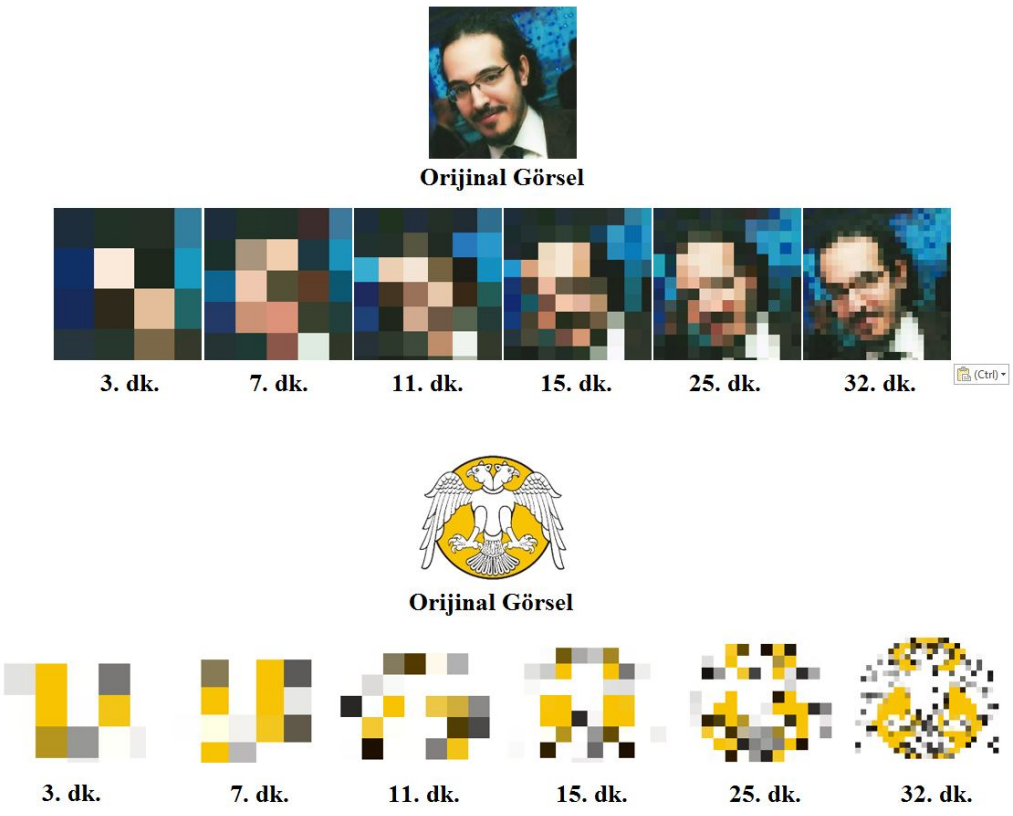
Çizelge 5.10.'dan anlaşılacağı üzere, GOA bu problem için de en iyi performansı gösterirken, kendisini GKA_{V1} ve YAK_{V1} gibi algoritmalar takip etmektedir. BiGOA da problem kapsamında en iyi dördüncü sonucu vermiştir. Diğer yandan, Şekil 5.6 da, GOA'nın kendisini takip eden iki algoritma – teknik ile beraber, literatürde rapor edilen pareto optimumlara benzer yapıda başarım ortaya koyduğunu göstermektedir.

Çözümlemeleri açıklanan uygulamalar dışında, 'yeniden görüntü oluşturma' problemi – uygulamasına yönelik iki farklı sürece ilişkin bulgular da kısaca şöyledir:

- 'Birinci yeniden görüntü oluşturma problemi', piksellerin orijinal görüntüdekilere, renk, boyut ve konum gibi faktörlerden uyumlanması yaklaşımına dayanmakla birlikte, *Denklem 3.33.* üzerinden devreye sokulan bir optimizasyon süreciydi. Örnek olması adına; GOA ve BiGOA'nın 50 parçacık ile iki farklı görseller üzerinde zamana bağlı ilerlemeleri Resim 5.1.' ve Resim 5.2.'de sunulmuştur.



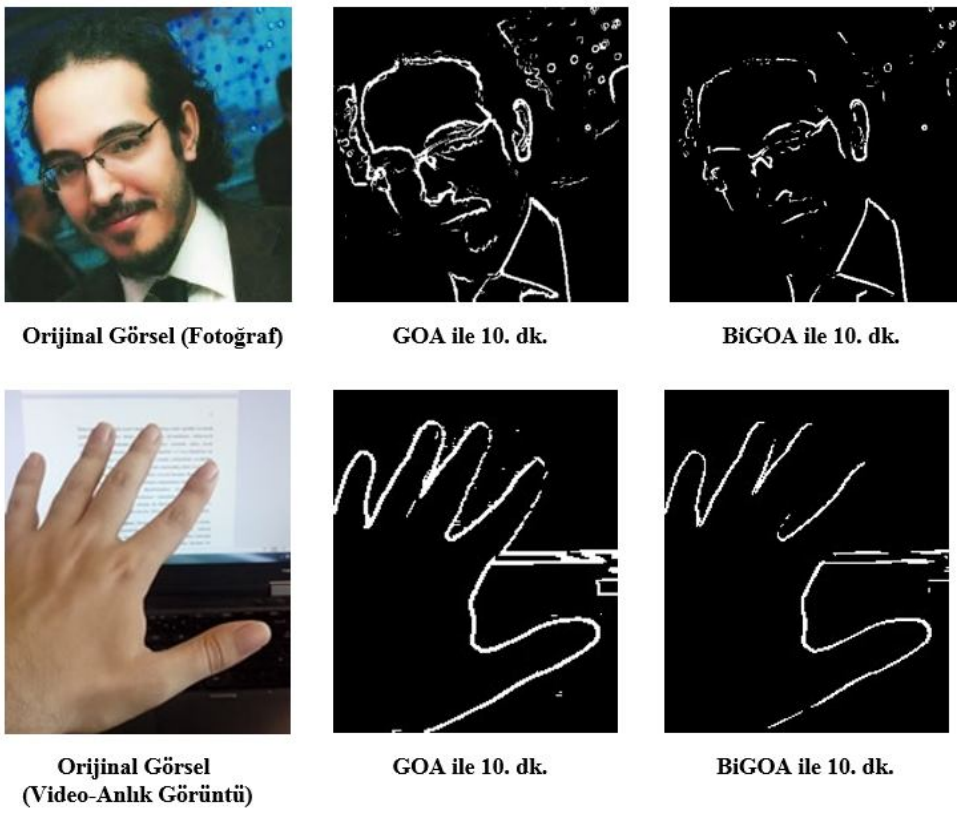
Resim 5. 1. Birinci yeniden görüntü oluşturma probleminde GOA çözüm süreci.



Resim 5. 2. Birinci yeniden görüntü oluşturma probleminde BiGOA çözüm süreci.

Görsel bulgulardan görüleceği üzere, sabit görseller üzerinde yeniden görüntü oluşturma konusunda GOA'nın performansı BiGOA'dan daha iyi durumdadır. Bunun dışında, her iki algoritma da, yeniden görüntü oluşturma konusunda yeter düzeyde başarım sergilemektedir. Burada dikkat çekilmesi gereken önemli bir husus da, hem GOA, hem de BiGOA da çözüm ortamının ek parçacıklarla desteklenmesi yoluna gidilmesidir. Buna göre, çözümlemeler 50 parçacık ile başlamış olsa da, hata oranlarına göre parçacık sayılarının artırılması yoluna gidilmiştir. Bu çözüm yaklaşımı problemin niteliği itibariyle, nitelikli tespit ve görsel sonucunun elde edilmesi için gereklidir.

- 'İkinci yeniden görüntü oluşturma problemi', hatırlanacağı üzere diğer problemden farklı olarak hareketli görüntülere dayalı ve kenarlarının tespitine odaklanan bir yapıdadır. Bu noktada, *Denklem 3.34.* ile piksel konumları hataları üzerinden optimizasyon çözümüne yönlenilmektedir. Bazı örnek hareketli görüntülerden faydalanmak suretiyle, GOA ve BiGOA'nın bu probleme yönelik performansları Resim 5.3.'te gösterilmiştir.



Resim 5. 3. İkinci yeniden görüntü oluşturma probleminde GOA ve BiGOA çözüm süreçleri.

Resim 5.3.'ten anlaşılacağı üzere, hareketli görüntü (video) üzerinden yeniden görüntü oluşturma problemi de hem GOA, hem de BiGOA tarafından kabul edilebilir düzeylerde çözümlenebilmektedir. Birinci yeniden görüntü oluşturma probleminde olduğu gibi, bu görüntü oluşturma probleminde de ek parçacıkların kullanılması yaklaşımı izlenmiş, böylelikle bir bakıma görseldeki kenarların daha belirgin tespiti sağlanması amaçlanmıştır.

5.3. Hibrit Kullanıma Yönelik Bulgular

Sürekli optimizasyon teknikleri GOA ve BiGOA, özellikle günümüzde oldukça yaygın bir araştırma yaklaşımı olan hibrit model kullanımı açısından yeterlikleri değerlendirme adına YSA, ASBÇS, ve SVM modellerinin eğitiminde kullanılmıştır. Bu amaçla aşağıdaki uygulamalar da gerçekleştirilmiştir:

- Kaotik zaman serilerinin tahmini (Köse ve Arslan, 2016b; 2017c),
- E-Öğrenme tabanlı sistemde öğrencilere uygun materyallerin sunulması (Köse ve Arslan, 2016a; 2017a),
- Biyomedikal veri tahmini – teşhis (Diyabet uygulaması) – (Köse ve ark., 2015; 2016).

YSA ve ASBÇS'nin eğitimi açısından dikkate alınan araştırmalar – uygulamalar, öncelikli olarak kaotik zaman serilerinin analizi üzerine kurulmuştur. Bu konunun seçilmesi, zaman serilerinin tahminine olan araştırma ilgisinin dışında, kaotik zaman serilerinin özellikle bu açıdan ilgili literatürde önem arz etmesidir. Okuyucular (Köse ve Arslan, 2016b) ve (Köse ve Arslan, 2017c)'den ilgili uygulamalar konusunda detaylı bilgi sahip olabileceğinden dolayı, burada uygulamalar öncesinde YSA ve ASBÇS modellerinin GOA ve BiGOA'nın da dâhil olduğu, GPSD'de başarılı olan ilk on algoritma ile eğitilme performansları konusunda bulgulara yer verilmiştir.

Kaotik zaman serileri tahmini kapsamında YSA ve ASBÇS için 4'er tane olmak üzere toplam 8 farklı kaotik zaman serisi tahmin edilmeye çalışılmıştır. Bu noktada YSA'nın eğitimi için BiGOA kullanılırken, ASBÇS'nin eğitimi için de GOA tercih edilmiştir. Söz konusu zaman serileri ve tahmine konu hibrit model bilgileri Çizelge 5.11.'de verilmiştir (Köse ve Arslan, 2016b; 2017c).

Çizelge 5. 11. Kaotik zaman serileri ve modeller (Köse ve Arslan, 2016b; 2017c).

K. S. No	Kaotik Seri	Kullanılan Model
1	1981-1990 periyodunda Avustralya, Melbourne'de günlük maksimum hava sıcaklıkları	YSA-BiGOA
2	1781-1988 periyodunda kış mevsiminde negatif hava sıcaklıkları toplamları	YSA-BiGOA
3	1871-1970 periyodunda Nil'deki yıllık ortalama akış değerleri	YSA-BiGOA
4	20dB gürültü ile modifiye edilmiş Mackey Glass zaman serisi	YSA-BiGOA
5	Z023 (Bonn Üniversitesi kaotik zaman serisi)	ASBÇS-GOA
6	N052 (Bonn Üniversitesi kaotik zaman serisi)	ASBÇS-GOA
7	MATLAB—EEGLAB (kaotik zaman serisi – 1)	ASBÇS-GOA
8	MATLAB—EEGLAB (kaotik zaman serisi – 2)	ASBÇS-GOA

Uygulama – problem kapsamında kurulan YSA ve ASBÇS yapılarına ilişkin genel kabulleri kısaca şu şekilde ifade edebiliriz (Köse ve Arslan, 2016b; 2017c):

- YSA modeli çok-katmanlı, ileri beslemeli mimari üzerine kurulu olmakla birlikte, her birinde 8 yapay sinir hücresi bulunan 2 gizli katmana sahiptir. Giriş katmanında 4, çıkış katmanında ise 1 yapay sinir hücresi bulunan modelin amacı, hedef zaman serisinde, $x(z + 3)$ zamanındaki noktayı, $x(z)$, $x(z - 3)$, $x(z - 6)$ ve $x(z - 9)$ zamanlarındaki noktalar aracılığıyla tahmin etmektir. YSA modelindeki bütün yapay sinir hücreleri sigmoid aktivasyon fonksiyonun kullanmaktadır.

- ASBÇS modeli 8 kural içermekte ve hedef zaman serisinde, $x(z + 3)$ zamanındaki noktayı $x(z)$, $x(z - 3)$ ve $x(z - 6)$ zamanlarındaki noktalar aracılığıyla tahmin etmek üzere, 3 giriş yapay sinir hücresi ve 1 çıkış yapay sinir ücresi içerecek şekilde tasarlanmıştır.

Çizelge 5.11.'de ifade edilen modellerin kurulması öncesinde, her bir seri için GPSD'deki ilk on algoritma – teknik ile birlikte toplamda 30 defa yapılan tahmin süreçlerine tümüne dair OHKK (*Bkz. Denklem 3.36.*) değerlerinin ortalaması bağlamındaki bulgular Çizelge 5.12. ve Çizelge 5.13.'de sunulmuştur.

Çizelge 5. 12. Kaotik zaman serileri tahmininde YSA modellerine yönelik OHHK bulguları.

K.S. No	OHKK Bazlı Bulgular[a]									
	YSA-GOA	**YSA-GKA$_{V1}$**	**YSA-YAK$_{V1}$**	**YSA-ÇTO$_{V1}$**	**YSA-BiGOA**	**YSA-GKA**	**YSA-YAK**	**YSA-ÇTO**	**YSA-BYAO$_{V1}$**	**YSA-ERA$_{V1}$**
1	**6,450E+03**	8,191E+03	9,518E+03	8,025E+03	2,451E+04	1,553E+04	9,616E+03	8,421E+03	3,670E+04	2,442E+04
2	7,121E+04	**5,083E+04**	8,101E+04	9,253E+04	4,472E+05	7,641E+04	5,899E+05	1,097E+05	7,330E+04	8,292E+04
3	**5,153E+04**	7,413E+04	7,949E+04	7,307E+04	7,609E+04	8,019E+04	3,408E+05	2,709E+05	8,511E+04	1,796E+05
4	**7,811E+05**	3,214E+06	4,233E+06	9,471E+05	5,703E+06	2,813E+07	2,420E+07	3,305E+06	6,201E+06	5,828E+06

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

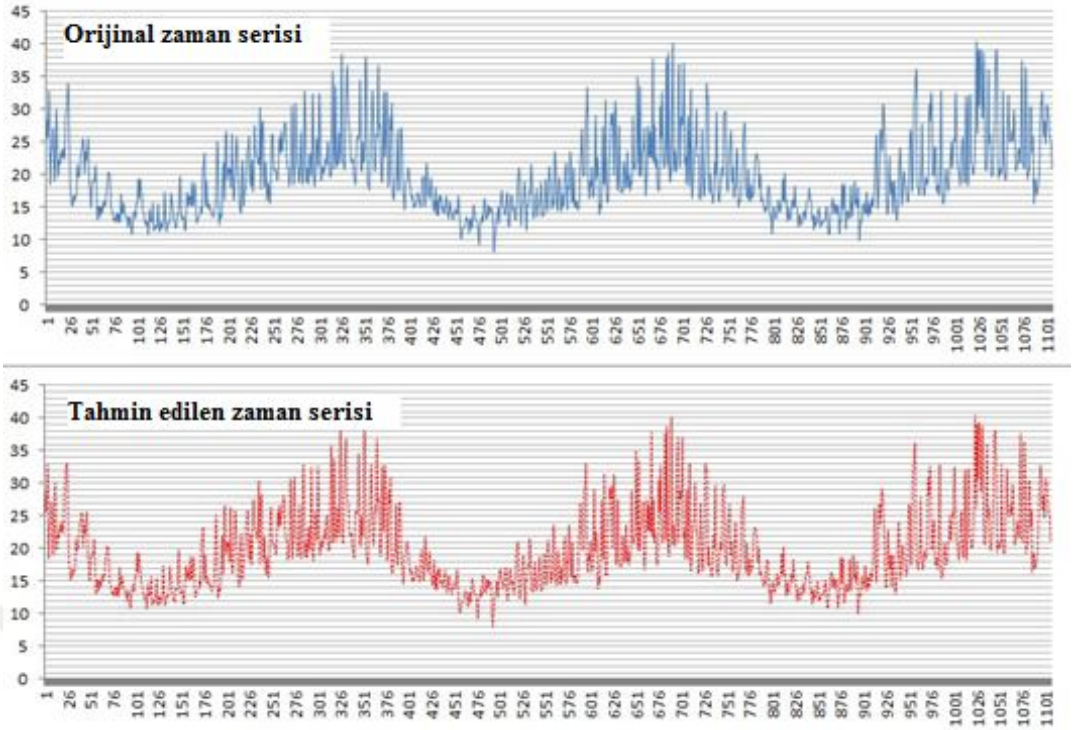
Çizelge 5. 13. Kaotik zaman serileri tahmininde ASBÇS modellerine yönelik OHHK bulguları.

K.S. No	OHKK Bazlı Bulgular[a]									
	ASBÇS-GOA	**ASBÇS-GKA$_{V1}$**	**ASBÇS-YAK$_{V1}$**	**ASBÇS-ÇTO$_{V1}$**	**ASBÇS-BiGOA**	**ASBÇS-GKA**	**ASBÇS-YAK**	**ASBÇS-ÇTO**	**ASBÇS-BYAO$_{V1}$**	**ASBÇS-ERA$_{V1}$**
5	**3,306E+04**	4,259E+04	6,137E+04	5,282E+04	7,470E+04	6,771E+04	7,843E+04	6,337E+04	6,117E+04	6,414E+04
6	**5,613E+05**	7,421E+05	3,453E+06	1,199E+06	5,608E+06	4,963E+06	6,021E+06	9,122E+05	4,307E+06	4,337E+06
7	6,514E+05	8,111E+05	**8,180E+04**	2,159E+06	3,523E+05	5,221E+05	9,985E+04	7,126E+05	5,237E+05	6,311E+05
8	**6,927E+05**	8,038E+05	9,537E+05	8,510E+05	9,246E+05	9,236E+05	8,691E+05	7,917E+05	7,383E+05	7,701E+05

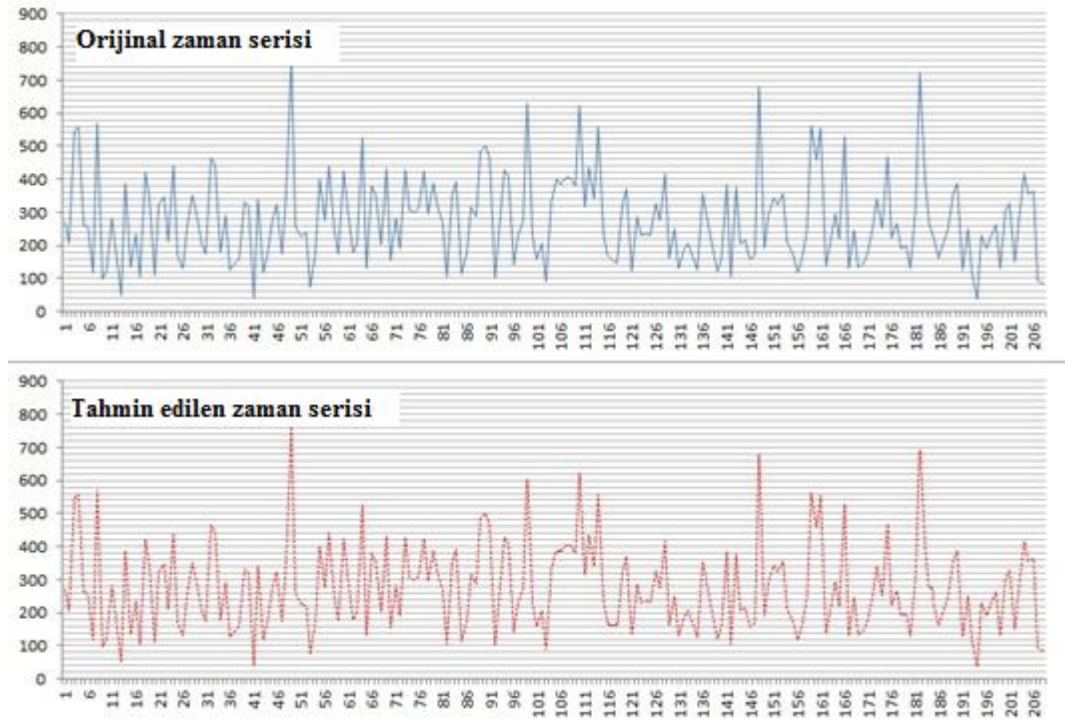
[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Çizelge 5.12. ve Çizelge 5.13.'ten anlaşılacağı üzere, GOA'nın hem YSA, hem de ASBÇS üzerindeki performansı diğer algoritmaların çoğuna göre daha iyi sonuç vermiştir. Diğer yandan BiGOA da, kabul edilebilir düzeyde performanslar sergilediği için ilgili kaotik zaman serisi tahmini uygulamalarında bu iki algoritmanın, iki farklı model için ayrı ve detaylı bir şekilde kullanılmasına devam edilmiştir. Tahmin süreçleri ile genel anlamda olumlu yönde bulgulara ulaşılmış, dolayısıyla GOA ve BiGOA'nın YSA ve ASBÇS'yi eğitme açısından başarılı oldukları da böylelikle görülmüştür. Resim 5.4., Resim 5.5., Resim 5.6. ve Resim 5.7. YSA-BiGOA sistemi ile, Resim 5.8., Resim 5.9., Resim 5.10. ve Resim 5.11. ise ASBÇS-GOA sistemi ile yapılan kaotik zaman serisi tahmin süreçlerinin grafiklerini göstermektedir (Köse ve Arslan, 2016b; 2017c).

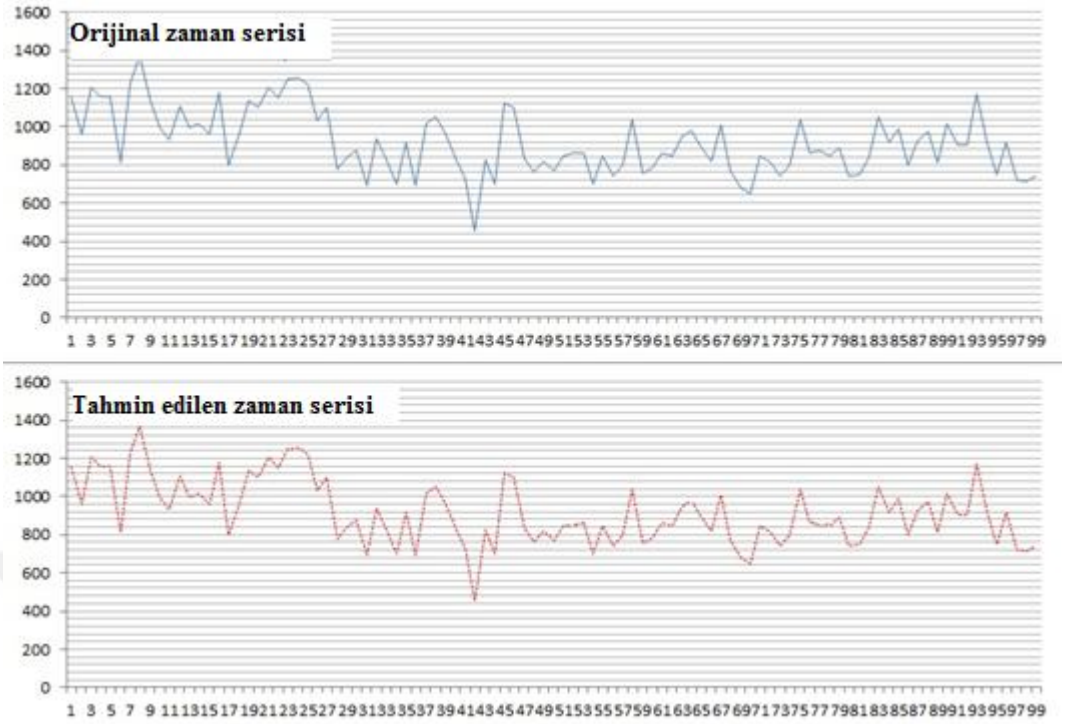
Resim 5.4. – Resim 5.11. takip eden sayfalardadır.



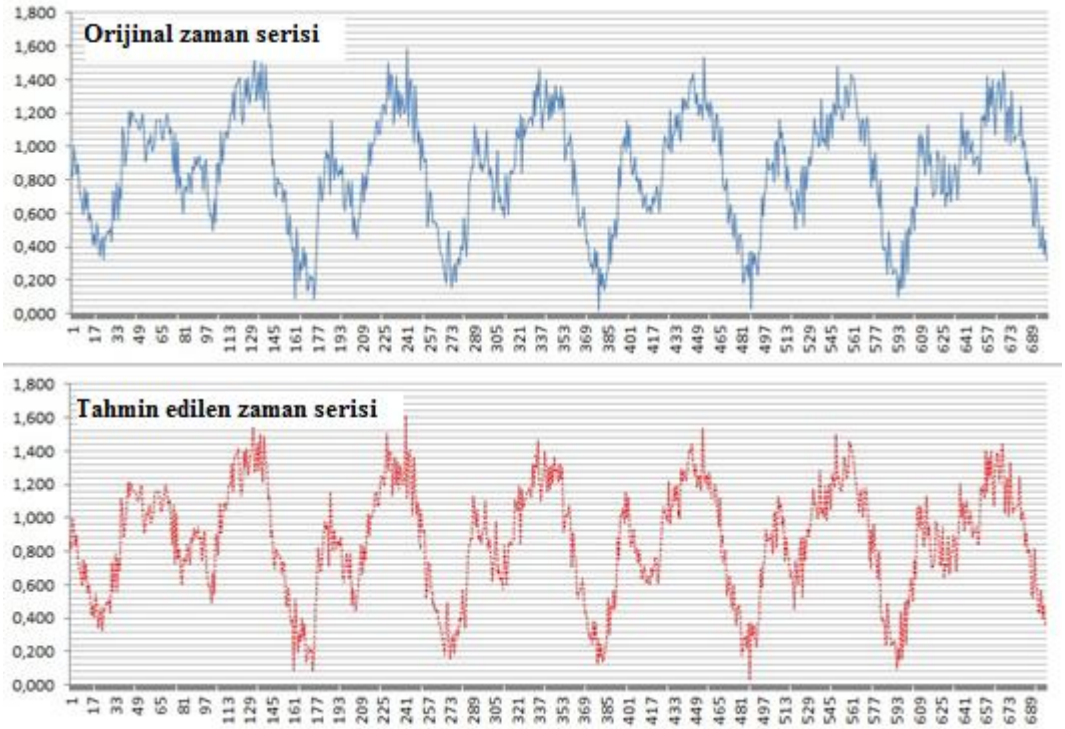
Resim 5. 4. YSA-BiGOA ile 1. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c).



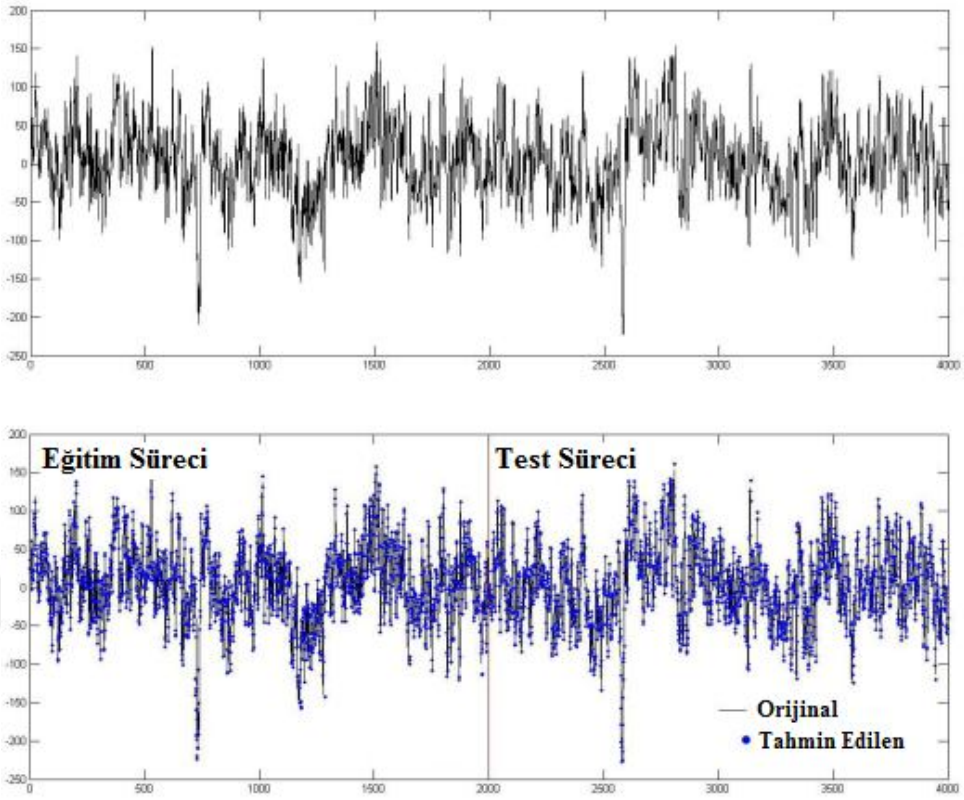
Resim 5. 5. YSA-BiGOA ile 2. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c).



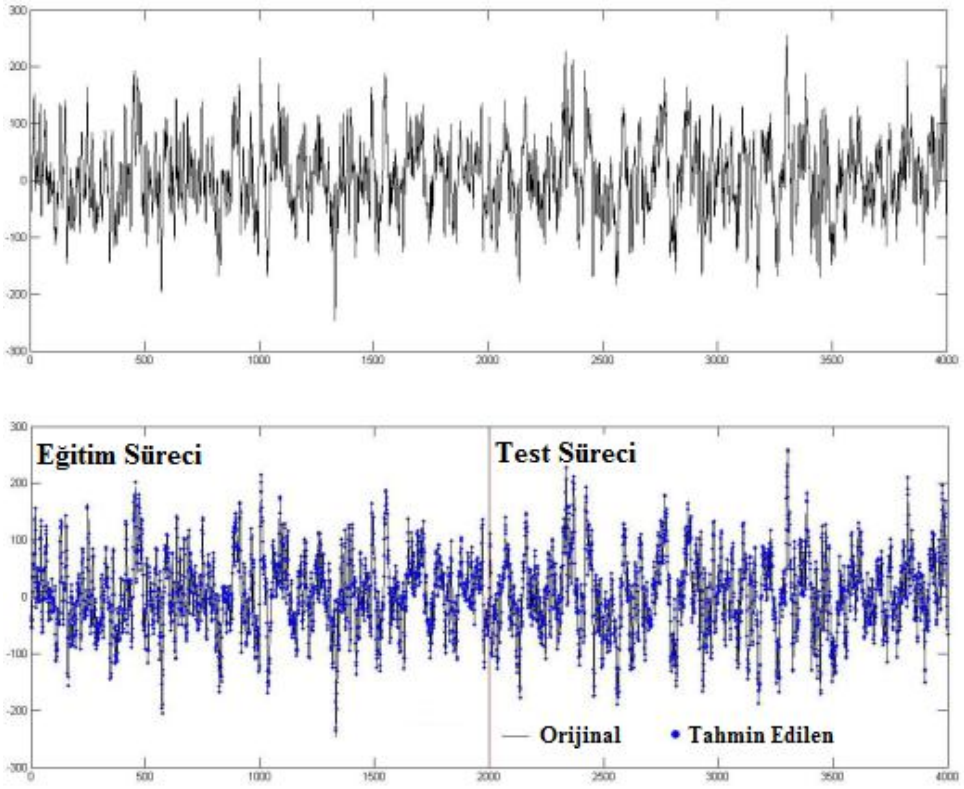
Resim 5. 6. YSA-BiGOA ile 3. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c).



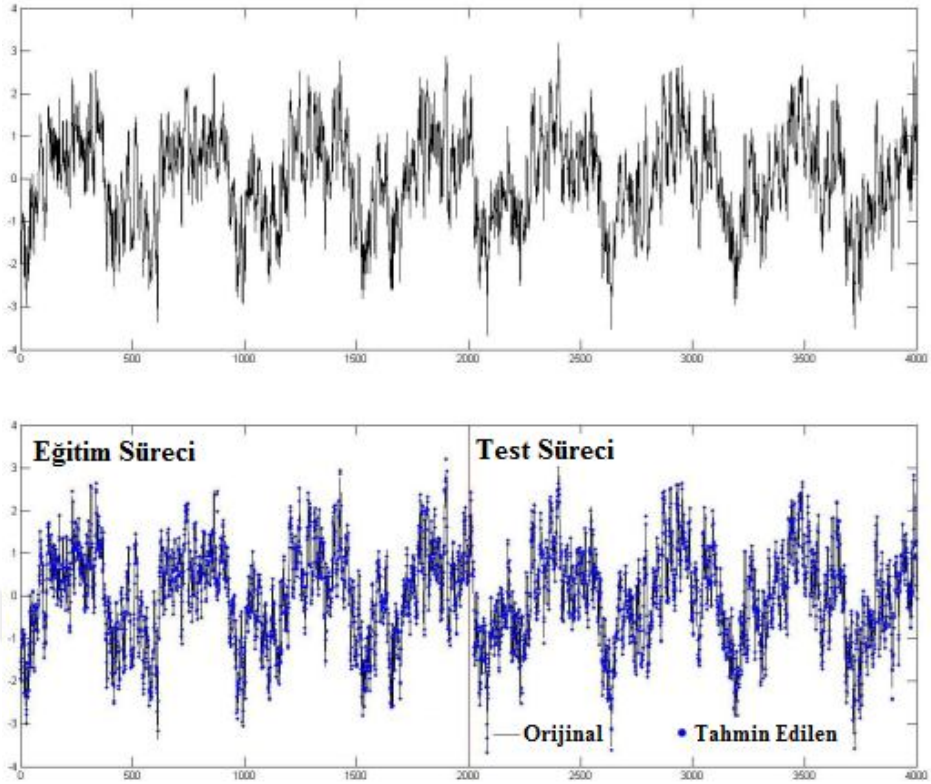
Resim 5. 7. YSA-BiGOA ile 4. kaotik zaman serisi tahmini (Köse ve Arslan, 2017c).



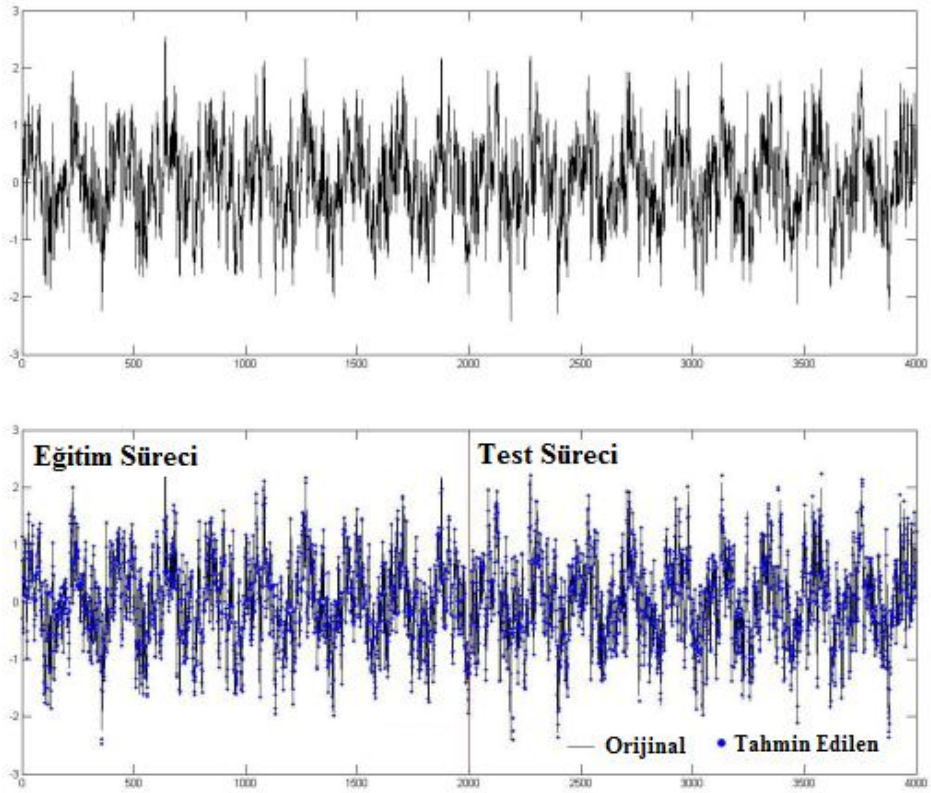
Resim 5. 8. ASBÇS-GOA ile 5. kaotik zaman serisi tahmini (Köse ve Arslan, 2016b).



Resim 5. 9. ASBÇS-GOA ile 6. kaotik zaman serisi tahmini (Köse ve Arslan, 2016b).



Resim 5. 10. ASBÇS-GOA ile 7. kaotik zaman serisi tahmini (Köse ve Arslan, 2016b).



Resim 5. 11. ASBÇS-GOA ile 8. kaotik zaman serisi tahmini (Köse ve Arslan, 2016b).

Eğitimsel süreçlerin kalitesinin artırılması konusunda yapılan çalışmalarla ilgili olmak üzere, GOA ve BiGOA'nın hibrit model içerisinde kullanıldığı diğer bir uygulama da E-Öğrenme sistemleri üzerinde zeki YSA tabanlı materyal seçim mekanizmalarının oluşturulması üzerine kurulmuştur. Uygulamalardan ilki (Uygulama – 1) standart öğrenci faaliyetleri (sistem üzerinde ders takibi, sınav puanları… vb.) ile materyallerin eşleştirilmesi üzerine kurulmuşken (Köse ve Arslan, 2016a), ikincisi ise (Uygulama – 2) Gardner'ın Çoklu Zekâ Teorisi (Gardner, 1983; 1991) üzerine kurularak, öğrencilerin çoklu zekâ durumları ile materyallerin eşleştirilmesi şeklinde bir yaklaşıma dayanmıştır (Köse ve Arslan, 2017a). Uygulamalar için YSA modelleri çok-katmanlı, ileri beslemeli mimari kapsamında tasarlanmış ve eğitim süreçlerinde SZ algoritmalarının kullanılması yaklaşımı tercih edilmiştir. Bu noktada, birinci uygulamadaki YSA modeli, E-Öğrenme sistemi üzerindeki çalışma sürecinde gizli katmanlardaki ve hatta giriş ile çıkış katmanlarındaki yapay sinir hücreleri sayıları değişebilir yapıda kullanılmış olsa da, eğitim ve buradaki algoritma değerlendirmesi için kullanılan modelin, toplamda 28 farklı eğitimsel veriyi (sınav, quiz verileri, materyal istatistikleri, ders takibi… vb.) kabul edecek şekilde giriş katmanına sahip, 25'er yapay sinir hücreinden oluşan iki gizli katmanı içeren, çıkışta da ön-tanımlı 5 farklı öğrenme düzeyi – kategorisi sınıflandırmasına karşılık gelen hücrelerden oluşan bir yapıda tasarlanması yolu izlenmiştir. Her ne kadar ikinci uygulamadaki YSA modeli de sistem kullanımı sırasında değişebilir bir yapıda olsa da, eğitim ve algoritma değerlendirmesi için işe koşulan modelin, –kullanılan Çoklu Zekâ testine uygun olarak– giriş katmanında 56 yapay sinir hücresi ve çıkış katmanında Çoklu Zeka gruplarına uygun olarak 8 yapay sinir hücresi ve yine 30'ar yapay sinir hücresi içeren 2 gizli katmanda kurulu olması sağlanmıştır.

Tıpkı kaotik zaman serileri tahmini uygulamasında olduğu gibi, E-Öğrenme yönelimli uygulamalarda da, GPSD'deki ilk on algoritma – tekniğin ön-değerlendirmede denenmesi ve böylelikle, hedef uygulamalardaki iki farklı YSA modelinin, toplamda 30 defa eğitilmesi süreci gerçekleştirilmiştir. İlgili süreçteki verilerin tespitine yönelik olarak da yine OHKK (*Bkz. Denklem 3.36.*) değerlerinin ortalamaları dikkate alınmıştır. Bu kapsamdaki bulgular, Çizelge 5.14. ve Çizelge 5.15.'te sunulmuştur.

Çizelge 5. 14. E-Öğrenme tabanlı uygulama – 1 için OHKK bulguları.

Uygulama – 1 için OHKK Bazlı Bulgular[a]									
YSA-GOA	**YSA-GKA$_{V1}$**	**YSA-YAK$_{V1}$**	**YSA-ÇTO$_{V1}$**	**YSA-BiGOA**	**YSA-GKA**	**YSA-YAK**	**YSA-ÇTO**	**YSA-BYAO$_{V1}$**	**YSA-ERA$_{V1}$**
7,127E+06	5,222E+06	6,436E+06	7,419E+06	**3,584E+06**	7,031E+06	7,215E+06	8,871E+06	9,283E+06	8,904E+06

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Çizelge 5. 15. E-Öğrenme tabanlı uygulama – 2 için OHKK bulguları.

Uygulama – 2 için OHKK Bazlı Bulgular[a]									
YSA-GOA	**YSA-GKA$_{V1}$**	**YSA-YAK$_{V1}$**	**YSA-ÇTO$_{V1}$**	**YSA-BiGOA**	**YSA-GKA**	**YSA-YAK**	**YSA-ÇTO**	**YSA-BYAO$_{V1}$**	**YSA-ERA$_{V1}$**
4,471E+05	6,320E+05	5,129E+05	5,116E+05	9,699E+05	2,327E+06	9,915E+05	1,352E+06	1,522E+06	9,946E+05

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Çizelge 5.14. ve Çizelge 5.15'ten görüleceği üzere, GOA ve BiGOA ayrı YSA modelleri için diğer algoritmalara – tekniklere göre daha iyi performanslar göstermiştir. Bu nedenle GOA ve BiGOA'nın uygulamalarda ayrı YSA'larda kullanılarak uygulamalara devam edilmesi yönünde karar alınmıştır.

Gerçekleştirilen E-Öğrenme uygulamalarının (sistemlerin), eğitimsel bağlamda etkin oldukları, deney – kontrol grupları odaklı süreçlerden, yapılan öğrenci anketlerinden ve sistemler üzerinden gerçekleştirilen, doğru materyallerin sunulup sunulmadığını ölçen süreçlerden elde edilen bulgularla anlaşılmıştır (Köse ve Arslan, 2016a; 2017a). Dolayısıyla, GOA ve BiGOA'nın YSA'yı eğitme konusunda olumlu yöndeki başarımları, bir kez daha görülmüştür.

GOA ve BiGOA'nın hibrit model kullanımı yönündeki uygulamalardan bir diğer de DVM ile bağlantılı olarak gerçekleştirilmiştir. Buna göre, biomedikal veri tahmini – teşhis üzerine kurulu uygulamada her iki algoritma da, diyabet veri seti üzerinden bireylerin diyabet olup olmadıklarının tespiti amaçlı tasarlanan DVM'nin çekirdek fonksiyonu parametre optimizasyonu için kullanılmıştır. Sırasıyla (Köse ve ark., 2015) ve (Köse ve ark., 2016)'da raporlanan uygulamalarda, *Denklem 5.1*'de verilen Gaussian (RBF) fonksiyonunun sigma (σ) parametresi optimize edilerek, tahmin – teşhis sürecinde kullanılacak en uygun DVM'nin belirlenmesi yoluna gidilmiştir.

$$f(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2*\sigma^2}} \quad (5.1)$$

Diğer uygulamalarda olduğu gibi, GPSD'deki ilk on algoritma – teknik, buradaki DVM için kullanılmış ve toplam 30 defa yapılan sigma (σ) parametresi optimizasyonlarından elde edilen en iyi parametre değerleri dikkate alınmıştır. Bu bağlamda, çekirdek fonksiyonu en iyi değerle optimize edilmiş her bir hibrit DVM modellerinden veri tahmini – teşhis kapsamında elde edilen OHKK (*Bkz. Denklem 3.36.*) değerleri başarım kriteri kapsamında incelenmiştir. Söz konusu OHKK bulguları Çizelge 5.16.'da verilmiştir.

Çizelge 5. 16. Biyomedikal veri tahmini – teşhisi için OHKK bulguları.

Biyomedikal veri tahmini – teşhis uygulaması için OHKK Bazlı Bulgular[a]									
DVM-GOA	**DVM - GKA$_{V1}$**	**DVM - YAK$_{V1}$**	**DVM - ÇTO$_{V1}$**	**DVM - BiGOA**	**DVM – GKA**	**DVM - YAK**	**DVM - ÇTO**	**DVM - BYAO$_{V1}$**	**DVM-ERA$_{V1}$**
5,249E +04	6,844E +04	7,081E +04	7,116E +04	6,549E +04	8,472E +04	1,024E +05	1,658E +05	1,392E +05	9,714E +04

[a] Koyu gösterilen değer, bütün algoritmalar – teknikler arasında elde edilen en iyi değer demektir.

Çizelge 5.16.'dan anlaşılacağı gibi, başta GOA olmak üzere, bu tez çalışması kapsamında ortaya konulan her iki sürekli optimizasyon algoritması da, ilgili parametre optimizasyonu açısından diğer algoritmalara – tekniklere göre daha iyi performans sergilediğinden dolayı, tahmin – teşhis açısından daha iyi DVM modelleri elde edilebilmektedir.

GOA ve BiGOA'nın DVM optimizasyonu sürecinde kullanıldığı biyomedikal tabanlı çalışmada, teşhis adına UCI Machine Learning Repository'den (Asuncion ve Newman, 2007) alınmış olan, Pima Yerlileri'ne ait diyabet veri seti kullanılmıştır. Toplamda 8 nitelik kapsamında, 768 bireye ait verilerden 500 tanesi diyabet olmayan (diyabet negatif), geriye kalan 268 tanesi ise diyabetli (diyabet pozitif) bireylere aittir. Yapılan çalışmalar neticesinde farklı hibrit modellerle ulaşılan sigma (σ) değerleri ve teşhis doğruluğu yönündeki bulgular Çizelgel 5.17.'de verilmiştir (Köse ve ark., 2015; 2016).

Çizelge 5.17., takip eden sayfadadır.

Çizelge 5. 17. DVM sigma değerleri ve diyabet teşhis doğrulukları (Köse ve ark., 2015; 2016).

Kullanılan Model	Bulunan Sigma (σ)	Teşhis Doğruluğu (%)
DVM-GOA	8,6144	76,8
DVM-GOA	4,4587	74,2
DVM-GOA	0,9028	77,1
DVM-GOA	0,7017	90,4
DVM-GOA	0,6618	92,7
DVM-BiGOA	9,5062	76,6
DVM-BiGOA	7,2388	75,3
DVM-BiGOA	3,9547	74,5
DVM-BiGOA	0,9107	76
DVM-BiGOA	0,8377	81,3
DVM-BiGOA	0,7405	87,5

Sonuç olarak bu biyomedikal odaklı uygulama süreciyle ulaşılan bulgular da, GOA ve BiGOA'nın hibrit kullanımdaki etkinliğini ve bu doğrultuda DVM parametre optimizasyonu yönündeki başarılarını göstermiştir.

5.4. Tartışma

Elde edilen bütün bulgulardan yola çıkarak, çalışmanın geneliyle ilgili olarak şu tespitlerde bulunmamız mümkündür:

- Geliştirilen GOA ve BiGOA, basit matematiksel yapılara dayanıyor olsa da, optimizasyon problemleri için etkin sonuçlar ortaya koyabilmiştir.
- GOA'nın literatürdeki rakiplerine göre durumuna baktığımızda, yapılan değerlendirme çalışmalarının büyük çoğunluğunda üstünlükler sağladığını ifade edebiliriz. GOA bu açıdan bakıldığında BiGOA'dan daha iyi performanslar sergileyebilmektedir.

- Elde edilen bulgulardan yola çıkarak, BiGOA'nın rakipleri yanında daha ortalama performanslar sergileyebildiğini yorumlayabiliriz. Yine işlem süreleri açısından nispeten biraz daha yavaştır.

- Genel süreçlerde GOA'nın yanında diğer tekniklerin varyasyonlarının ön plana çıktığını görebiliriz. Bunun sebebi, varyasyonların zaten algoritmaların – tekniklerin eksiklerini kapatma yönünde yapılan çalışmaların ürünleri olmaları olarak yorumlayabiliriz.

- SZ literatüründe yeni çıkan algoritmalar öncekilere göre daha iyi performanslar ortaya koyabilmektedir. Bunun nedeni olarak da, literatüre önceleri kazandırılmış olan algoritmalarda – tekniklerde söz konusu olan çeşitli eksiklerin yeni algoritmalar ile kapatılmaya çalışılması olarak yorumda bulunabiliriz.

- Her ne kadar yeni algoritmaların ön plana çıkmalarından bahsediyorsak da, zamanında iyi geliştirilmiş olan GKA, YAK, ve ÇTO gibi algoritmaların da istikrarı dikkate çekici bir durum olarak yorumlanabilir.

- Tekrar GOA ve BiGOA'ya odaklandığımızda, bu algoritmaların 'en iyi' performans – sonuç ortaya koyamadıkları problemlerde – uygulamalarda bile kabul edilebilir düzeyde sonuçlar ürettiğini görebiliriz. Bu durumu en iyi GPSD sonuçlarında görmekteyiz.

- GOA ve BiGOA tek-amaçlı optimizasyon problemlerinde de, çok-amaçlı optimizasyon problemlerinde de etkin sonuçlar elde edebilmiştir.

- GOA ve BiGOA'nın etkin sonuçlar yönünden performansı yüksek boyutlu fonksiyonlar için de geçerlidir. Söz konusu algoritmalar, problemlerdeki fonksiyonların boyutları yükselse bile başarılı ya da kabul edilir düzeylerde sonuçlar vermeye devam etmiştir.

- GOA ve BiGOA'nın genel performansları sadece test amaçlı tasarlanmış problemlerde değil, gerçek yaşam tabanlı ya da alternatif yapıdaki uygulamalarda da iyi düzeylerde olmuştur.

- GOA ve BiGOA, YSA, ASBÇS, ve DVM gibi tekniklerle hibrit yapıda sistemler kurma konusunda başarılı sonuçlar vermiştir. Bu durum, ilgili algoritmaların YZ tabanlı hibrit yapıların oluşturulmasında farklı tekniklere entegre edilebilecekleri yorumuna ulaşmamızı sağlamaktadır.

- Özellikle çok-disiplinli kapsamda gerçekleştirilen çalışmalar, her iki algoritmanın farklı yapıda ve karmaşıklıkta optimizasyon problemleri için bile iyi tasarlanmış olduklarını göstermiştir.

- SZ ve zeki optimizasyon ile ilgili literatürler her zaman için ilerleme durumunda olduğundan dolayı, her iki algoritmanın da gelişime açık bir yapıda olduğunu ifade etmek gerekmektedir. Özellikle kimi problemlerde en iyi performansı gösterememe durumları, algoritmalardaki işlem süreçlerinin detaylı incelenmesini de beraberinde getirmelidir. Bu durum kuşkusuz ki algoritmaların çeşitli varyasyonlarının geliştirilmesi için de önemlidir.

- Literatürde dikkat çeken çalışmalardan birisi de, 'No Free Lunch Teorisi'dir (Ho ve Pepyne, 2002; Wolpert ve Macready, 1995). Bu teori, hangi YZ tabanlı optimizasyon algoritması olursa olsun, herhangi iki algoritmanın – tekniğin asla birbirlerine üstünlük sağlamayacağını, bu algoritmalardan her birinin diğerine üstünlük sağladığı problem çözümlemelerinin dengeli olacağını savunmaktadır. Bu durumda, YZ tabanlı optimizasyon algoritmaları için literatürde gerçekleştirilen 'karşılaştırma' odaklı değerlendirmelerin geçerliliği tartışılabilir bir niteliğe bürünmektedir. Bu durum akıllarda soru işareti yaratsa da, özellikle çok-disiplinli ve farklı kapsamdaki problemler için yapılan optimizasyon çözümlemelerinin, bu tür algoritmaların – tekniklerin sağlıklı analiz edilebilmesi adına birincil öncelik olması gerektiği yorumu yapılabilmektedir.

6. SONUÇLAR – ÖNERİLER VE GELECEK ÇALIŞMALAR

Bu tez çalışması ile birlikte, YZ araştırma alanı ve zeki optimizasyon teknikleri ile ilişkili olan literatürlere iki farklı sürekli optimizasyon algoritması kazandırılmıştır. Bu kapsamda, sırasıyla Girdap Optimizasyon Algoritması (GOA) ve Bilişsel Gelişim Algoritması (BiGOA) adıyla anılan algoritmaların tasarım – geliştirme süreçleri, temel özellik – işlevleri ve algoritma adımları açıklanmış, nihayetinde bu algoritmaların başarımlarını değerlendirmeye yönelik uygulamalara odaklanılmıştır. Çalışma içerisinde ayrıca alanla ilgili veya alan kapsamındaki çalışmalara yeni okuyucuların yeterli bilgi birikimi elde etmeleri adına gereken temel teorik bilgiler de sunulmuştur.

Çalışma sürecinde edinilen tecrübeler, çalışmanın temel amacı ile alt-amaçları doğrultusunda tamamlanmış aşamalar ve değerlendirme odaklı uygulamalar ile elde edilmiş bulgulardan yola çıkılarak, çalışma ile ulaşılmış olan başlıca sonuçlardan, öne sürülebilecek çeşitli önerilerden ve yine gelecek süreç için planlanan yeni çalışmalardan kısaca şu şekilde bahsetmek mümkündür:

6.1. Sonuçlar

Bu tez çalışması ile ulaşılan temel sonuçları kısaca şöyle özetleyebiliriz:

- Çalışma ile birlikte, hedef alınan literatürlere iki farklı algoritma – teknik kazandırılmıştır. Bu algoritmalar sürekli optimizasyon problemlerinde etkin olarak kullanılabilmektedir.
- Değerlendirme odaklı yapılan ve çeşitliliği yüksek çalışmalar, tanıtılan algoritmaların literatür için uygun niteliklerde olduklarını göstermiştir.
- Tasarlanıp geliştirilen algoritmalar, basit matematiksel yapılarla kodlanması kolay, etkin optimizasyon algoritmaları oluşturulabileceğini göstermiştir.
- GOA, sürekli optimizasyon açısından literatürde dikkat çeken bir konuma yerleşmiş, ikinci geliştirilen algoritma olan BiGOA da, GOA'yı takiben diğer alternatif algoritmalar – teknikler arasında yerini almıştır.

- Geliştirilen algoritmalar, gerçek yaşam tabanlı problemlerin çözümlenebilmesi aşamasında doğadan esinlebileceğimiz, belki de sonsuz nitelikte unsurlar olduğunu bizlere işaret etmektedir.

6.2. Öneriler

Bilimin en önemli özellikleri içerisinde birikimli ilerlemesini ve sürekli değişkenliğe açık olmasını sayabiliriz. İşte bu nedenle, her bilimsel alanda olduğu gibi, YZ alanı kapsamında yapılan bu tez çalışmasıyla ulaşılan sonuçların da alternatiflerle desteklenmesi, daha fazla yorumlanması ve hatta bu sonuçların üzerine yenilerin eklenmesi doğal olacaktır. Bu doğrultuda ifade edilebilecek çeşitli öneriler şöyledir:

- Çalışmada ortaya konulan GOA ve BiGOA tekniklerinin başarımına yönelik düşünceler, ne kadar fazla değerlendirme çalışması ve problem çözümü – uygulama gerçekleştirilirse o kadar netleşecektir ve 'mümkün olduğunca' sabitleşecektir. Bu nedenle ilgili algoritmalara ilgi duyacak araştırmacılar farklı disiplinler, farklı problemler, farklı problem koşulları ve hatta farklı alternatif teknikler dikkate alınmak suretiyle çalışmaları geliştirebilirler.

- Nasıl ki alternatif optimizasyon algoritmalarının – tekniklerinin zaman içerisinde farklı sürümleri ve gelişmiş formları tasarlandıysa, tekniklere ilgi duyan araştırmacılar, 'algoritmaların geliştirilmesi' yönünde çalışmaları literatüre kazandırabilirler.

- Gerek bir önceki madde altında belirtilen geliştirme yaklaşımları için, gerekse algoritmaların başarımlarını inceleme yönünde yapılacak çalışmalarda, dezavantajlı görülen yönlerin tespiti son derece önemlidir. Bu bağlamda, geliştirilen GOA ve BiGOA tekniklerinin çözüm süreçlerinin farklı faktörler açısından detaylı incelenmesi (Örneğin; parametre değişimlerinin ele alınması, farklı başlangıç parametre değerlerinin etkisinin analizi… vb.) izlenebilecek yaklaşımlar olarak önerilebilmektedir.

- Çalışmayla elde edilen bulgulardan (ve nihayetinde sonuçlardan) anlaşılacağı üzere, geliştirilen algoritmalar kimi optimizasyon süreçlerinde 'en iyi' olamamışlardır. Durumu yersine çevirmenin yolu yine önceki maddelerde

açıklanan detaylı çalışmaların yürütülmesinden geçmekle beraber, bu doğrultuda iyileştirmelerin yapılması da ayrıca önerilebilir.

- Bu tez çalışması kaleme alındığı zaman itibariyle popülaritesi yükselmekte olan YZ Güvenliği ve Makine Etiği gibi konularla ilişkin çeşitli optimizasyon problemleri tasarlanabilir ve bu konularla ilişkin literatürlerin geliştirilmeye devam etmesi noktasında GOA ve BiGOA gibi SZ teknikleri kullanılabilir.

6.3. Gelecek Çalışmalar

Ulaşılmış olunan sonuçlar ve elde edilen yeter düzeydeki başarımlar, çalışmada ortaya konulan algoritmalar ekseninde çeşitli 'gelecek çalışmaların' planlanması – düşünülmesi yönünde motivasyon oluşturmuştur. Buna göre, bu metin tamamlandığı an itibariyle ifade edilebilecek başlıca gelecek çalışmaları kısaca şu şekilde açıklayabiliriz:

- Daha önce de ifade edildiği üzere (*Bkz.* üçüncü bölüm, alt-başlık 3.6.) SZ ile ilgili literatür, bu satırlar kaleme alınırken bile yeni tekniklerin geliştirilmesine sahne olan, dinamik bir yapıya sahiptir. Bu yüzden, çalışma içerisinde geliştirilen tekniklerin alternatif teknikler ya da literatüre yeni girecek teknikler bağlamındaki başarımları da gelecek çalışmalara konu olabilecektir.

- Tahmin edileceği üzere, geliştirilen algoritmaların gerçek yaşam tabanlı, çok-disiplinli uygulamalarına devam edilecektir. Buna göre, algoritmaların hibrit YZ sistemleri içerisinde kullanılması olduğu kadar, doğrudan doğruya uygulanması yolları da tercih edilecek, böylece algoritmaların çok-disiplinli uygulamalardaki etkinlikleri değerlendirilmeye devam edecektir (Bu durum, YZ'nin gerçek yaşam tabanlı problemlerdeki etkinliğine ve çok-disiplinli yönünün değerlendirilmesine de katkı sağlayacak bir yaklaşımdır).

- GOA ve BiGOA'nın diğer tekniklerle birlikte hibrit sistem kurulumuna dâhil edilmesi yaklaşımı izlenmeye devam edilecek ve böylelikle algoritmaların, farklı teknikler dâhilinde problem çözümlemelerindeki performansları hakkında çalışmalar yapılacaktır.

- Bir önceki alt-başlık: 6.2. altında da konuyla ilgili araştırmacılara tavsiye edilen ‘yeni algoritmaların geliştirilmesi’ çalışmaları (Örneğin; alternatif bir kombinasyonel optimizasyon algoritması) gerçekleştirilmeye devam edilecektir.

- Yine bir önceki alt-başlık: 6.2.’de odaklanılan ve geliştirilen algoritmaların kimi süreçlerde ‘en iyi’ konuma yükseltilebilmesi için yapılabilecek iyileştirmeler bağlamında, çeşitli çalışmalar da yürütülecek ve bu esnada algoritmaların çözüm süreçleri daha detaylı irdelenecektir. Bu irdelemeler, algoritmaların hem çözüm üretme açısından, hem de çözüme ulaşma süreleri (çalışma zamanları) açısından iyileştirme – geliştirmeleri içerecektir.

- İlgili literatürde dikkat çeken araştırma yaklaşımlarından birisi de, sürekli optimizasyon algoritmaları – tekniklerinin kombinasyonel optimizasyona uyarlanacak şekilde revize edilmesidir. GOA ve BiGOA için bu yönde revizyonlar gerçekleştirilecektir.

- Bir önceki alt-başlık: 6.2. kapsamında da ifade edilen öneriyi izlemek suretiyle, GOA ve BiGOA üzerinden YZ Güvenliği ve Makine Etiği konularına yönelik optimizasyon odaklı çalışmalar (Örneğin; zeki ajanların davranış – çözüm optimizasyonları) gerçekleştirilecektir.

- Bilindiği üzere doğa ve yaşam, YZ’nin esinlenebileceği daha birçok dinamiği bünyesinde barındırmaktadır. Bu düşünceden hareketle, farklı doğal dinamiklerden ve canlı davranışlarından esinlenmek suretiyle alternatif algoritmaların tasarlanması ve geliştirilmesi çalışmalarına devam edilecektir. Hatta bu noktada, yapılacak çalışmaların, sadece optimizasyona odaklanan algoritmalar – teknikler olarak değil, farklı problem kapsamlarına karşı geliştirilebilecek algoritmalar – teknikler yönünde sürdürülmesine de dikkat edilecektir.

KAYNAKLAR

Abdechiri, M., Faez, K. ve Bahrami, H., 2010a, Neural network learning based on chaotic imperialist competitive algorithm, *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, 1-5.

Abdechiri, M., Faez, K. ve Bahrami, H., 2010b, Adaptive imperialist competitive algorithm (AICA), *Cognitive informatics (ICCI), 2010 9th IEEE international conference on*, 940-945.

Abdel-Baset, M. ve Hezam, I. M., 2015, An improved flower pollination algorithm for ratios optimization problems, *Applied Mathematics & Information Sciences Letters An International Journal*, 3 (2), 83-91.

Ackley, D., 1987:2012, A Connectionist Machine for Genetic Hillclimbing, (1987: Kluwer Boston Inc.) Springer Science & Business Media.

Agarwal, P. ve Mehta, S., 2016, Enhanced flower pollination algorithm on data clustering, *International Journal of Computers and Applications*, 38 (2-3), 144-155.

Agrawal, G., 2016, Artificial Intelligence-based Optimization Demystified. LinkedIn, https://www.linkedin.com/pulse/artificial-intelligence-based-optimization-gaurav-agrawal: [Erişim: 06.04.2016].

Ahmed, H. ve Glasgow, J., 2012, Swarm Intelligence: Concepts, Models and Applications. School Of Computing, Queens University Technical Report, https://pdfs.semanticscholar.org/116b/67cf2ad2c948533e6890a9fccc5543dded89.pdf: [Erişim: 18.03.2016].

Aickelin, U., Dasgupta, D. ve Gu, F., 2014, Artificial immune systems, In: Search Methodologies, Eds: Springer, p. 187-211.

Akbari, R., Hedayatzadeh, R., Ziarati, K. ve Hassanizadeh, B., 2012, A multi-objective artificial bee colony algorithm, *Swarm and Evolutionary Computation*, 2, 39-52.

Ali, M. M., Khompatraporn, C. ve Zabinsky, Z. B., 2005, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, *Journal of global optimization*, 31 (4), 635-672.

Allahverdi, N., 2002, Uzman Sistemler: Bir Yapay Zeka Uygulaması, Atlas Yayın Dağıtım.

Alpaydın, E., 2011, Yapay Öğrenme, *İstanbul*, Boğaziçi Üniversitesi Yayınevi.

AMRITA Learning, 2013, Sexual Reproduction in Plants, http://aven.amritalearning.com/index.php?sub=100&brch=293&sim=1484&cnt=3420: [Erişim: 15.02.2016].

Anderson, D. ve McNeill, G., 1992, Artificial neural networks technology, *Kaman Sciences Corporation*, 258 (6), 1-83.

Andreasik, J., 2009, The Knowledge Generation about an Enterprise in the KBS-AE (Knowledge-Based System-Acts of Explanation), In: New Challenges in Computational Collective Intelligence, Eds: Springer, p. 85-94.

Armer, P., 1960, Attitudes toward intelligent machines (pp. 3-19), http://aitopics.org/sites/default/files/classic/Feigenbaum_Feldman/Computers_And_Thought-Part_3_Armer.pdf: [Erişim: 10.12.2015].

Arunachalam, V., 2008, Optimization Using Differential Evolution. Water Resources Research Report. Unıversıty of Western Ontario – Dept. of Civil and Env. Eng., http://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=1022&context=wrrr: [Erişim: 28.01.2016].

Asuncion, A. ve Newman, D., 2007, UCI machine learning repository.

Atashpaz-Gargari, E. ve Lucas, C., 2007, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, *Evolutionary computation, 2007. CEC 2007. IEEE Congress on*, 4661-4667.

Aydın, A. O., 2013, Yapay Zekâ: Bütünleşik Bilişe Doğru, *İstanbul*, İstanbul Gelişim Üniversitesi Yayınları.

Bacanin, N. ve Tuba, M., 2012, Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators, *Studies in Informatics and Control*, 21 (2), 137-146.

Bachmayer, S., 2008, Artificial immune systems, *Artificial Immune Systems*, 5132, 119-131.

Back, T., 1996, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press.

Bäck, T., Fogel, D. B. ve Michalewicz, Z., 1997, Handbook of evolutionary computation, *New York: Oxford*.

Bäck, T., Fogel, D. B. ve Michalewicz, Z., 2000, Evolutionary Computation 1: Basic Algorithms and Operators, CRC Press.

Bahrami, H., Faez, K. ve Abdechiri, M., 2010, Imperialist competitive algorithm using chaos theory for optimization (CICA), *Computer modelling and simulation (UKSim), 2010 12th international conference on*, 98-103.

Baker, J. E., 1985, Adaptive selection methods for genetic algorithms, *Proceedings of an International Conference on Genetic Algorithms and their applications*, 101-111.

Balland, L., Estel, L., Cosmao, J.-M. ve Mouhab, N., 2000, A genetic algorithm with decimal coding for the estimation of kinetic and energetic parameters, *Chemometrics and Intelligent Laboratory Systems*, 50 (1), 121-135.

Baykal, N. ve Beyan, T., 2004, Bulanık Mantık İlke ve Temelleri, Bıçaklar Kitabevi.

Bayraktar, D. ve Çebi, F., 2003, Yöneylem Araştırması-1 – Ders Notları (İstanbul Teknik Üniversitesi – İşletme Fakültesi), http://web.itu.edu.tr/~cebife/TUM_YA1notlari.pdf: [Erişim: 10.12.2016].

Beni, G. ve Wang, J., 1993, Swarm intelligence in cellular robotic systems, In: Robots and Biological Systems: Towards a New Bionics?, Eds: Springer, p. 703-712.

Binitha, S. ve Sathya, S. S., 2012, A survey of bio inspired optimization algorithms, *International Journal of Soft Computing and Engineering*, 2 (2), 137-151.

Blum, C. ve Li, X., 2008, Swarm intelligence in optimization, In: Swarm Intelligence, Eds: Springer, p. 43-85.

Boettcher, S. ve Percus, A., 2000, Nature's way of optimizing, *Artificial Intelligence*, 119 (1-2), 275-286.

Bonabeau, E., Dorigo, M. ve Theraulaz, G., 1999, Swarm Intelligence: From Natural to Artificial Systems, 1, Oxford University Press.

Bonyadi, M. R. ve Michalewicz, Z., 2017, Particle swarm optimization for single objective continuous space problems: a review, MIT Press.

Brougham, D. ve Haar, J., 2017, Smart Technology, Artificial Intelligence, Robotics, and Algorithms (STARA): Employees' perceptions of our future workplace, *Journal of Management & Organization*, 1-19.

Brown, C. T., Liebovitch, L. S. ve Glendon, R., 2007, Lévy flights in Dobe Ju'hoansi foraging patterns, *Human Ecology*, 35 (1), 129-138.

Brynjolfsson, E. ve McAfee, A., 2015, The Second Machine Age: Akıllı Teknolojiler Devrinde Çalışma İlerleme ve Refah, *İstanbul*, Türk Hava Yolları Yayınları.

Bryson, J. T., Jin, X. ve Agrawal, S. K., 2016, Optimal design of cable-driven manipulators using particle swarm optimization, *Journal of mechanisms and robotics*, 8 (4), 041003.

Buchanan, B. G., 2005, A (Very) Brief History of Artificial Intelligence. AI Magazine, 26(4), 25th Anniversary Edition. 53-60, http://www.aaai.org/ojs/index.php/aimagazine/article/view/1848/1746: [Erişim: 20.01.2016].

Cao, J., Li, M., Wang, H., Huang, L. ve Zhao, Y., 2016, An Improved Bacterial Foraging Algorithm with cooperative learning for eradicating cancer cells using nanorobots, *Robotics and Biomimetics (ROBIO), 2016 IEEE International Conference on*, 1141-1146.

Chen, H., Zhu, Y. ve Hu, K., 2011, Adaptive bacterial foraging optimization, *Abstract and Applied Analysis*.

Chen, L., 2003, Real coded genetic algorithm optimization of long term reservoir operation, *JAWRA Journal of the American Water Resources Association*, 39 (5), 1157-1165.

Chen, X. ve Li, Y., 2007, A modified PSO structure resulting in high exploration ability with convergence guaranteed, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37 (5), 1271-1289.

Chen, Y.-P., Li, Y., Wang, G., Zheng, Y.-F., Xu, Q., Fan, J.-H. ve Cui, X.-T., 2017, A novel bacterial foraging optimization algorithm for feature selection, *Expert Systems with Applications*, 83, 1-17.

Cheng Yu, J., 2017, Numerical Optimization. Jade-Cheng.com, http://www.jade-cheng.com/au/coalhmm/optimization/: [Erişim: 19.01.2016].

Cheng, Z. ve Savit, R., 1987, Fractal and nonfractal behavior in Levy flights, *Journal of mathematical physics*, 28 (3), 592-597.

Chiang, C.-L., 2005, Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels, *IEEE transactions on power systems*, 20 (4), 1690-1699.

Chiang, S.-S., Chang, C.-I. ve Ginsberg, I. W., 2001, Unsupervised target detection in hyperspectral images using projection pursuit, *IEEE Transactions on Geoscience and Remote Sensing*, 39 (7), 1380-1391.

Chitsaz, H., Amjady, N. ve Zareipour, H., 2015, Wind power forecast using wavelet neural network trained by improved Clonal selection algorithm, *Energy Conversion and Management*, 89, 588-598.

Cochocki, A. ve Unbehauen, R., 1993, Neural Networks for Optimization and Signal Processing, John Wiley & Sons.

Cohen, P. R. ve Feigenbaum, E. A., 2014, The Handbook of Artificial Intelligence, Butterworth-Heinemann.

Collingridge, P., 2017, Evolving Images. Peter's Website, http://www.petercollingridge.co.uk/evolving-images: [Erişim: 11.01.2017].

Cortes, C. ve Vapnik, V., 1995, Support-vector networks, *Machine learning*, 20 (3), 273-297.

Cutello, V., De Michele, A. G. ve Pavone, M., 2014, Escaping local optima via parallelization and migration, In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), Eds: Springer, p. 141-152.

Çiftçi, S., 2011, MATLAB, *İstanbul*, Kodlab Yayıncılık.

Dai, H., Yang, Y. ve Li, C., 2016, Hybrid crossover based clonal selection algorithm and its applications. International Conference on Intelligent Data Engineering and Automated Learning, Springer**:** 468-475.

Dandy, G. C., Simpson, A. R. ve Murphy, L. J., 1996, An improved genetic algorithm for pipe network optimization, *Water resources research*, 32 (2), 449-458.

Danziger, P., 2015, Big O Notation. MTH607, http://www.math.ryerson.ca/~danziger/professor/MTH607/Handouts/bigO.pdf: [Erişim: 18.03.2017].

Darwin, C., 1859, On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life (On the Origin of Species), *London*, John Murray.

Darwin, C., 2015 (Çev.), Türlerin Kökeni, *İstanbul*, Evrensel Basım Yayın.

Das, R. K., Dash, S., Mishra, R. K. ve Panigrahy, A., 2017, E-CLONALG: A classifier based on Clonal Selection Algorithm, *Transactions on Machine Learning and Artificial Intelligence*, 4 (6), 88.

Das, S., Abraham, A. ve Konar, A., 2008, Automatic clustering using an improved differential evolution algorithm, *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 38 (1), 218-237.

Das, S., Biswas, A., Dasgupta, S. ve Abraham, A., 2009, Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications, In: Foundations of Computational Intelligence, Eds: Springer, p. 23-55.

Das, S. ve Suganthan, P. N., 2011, Differential evolution: A survey of the state-of-the-art, *IEEE transactions on evolutionary computation*, 15 (1), 4-31.

Datta, D. ve Kumar, S., 2015, Modeling using response surface methodology and optimization using differential evolution of reactive extraction of glycolic acid, *Chemical Engineering Communications*, 202 (1), 59-69.

Datta, T., Misra, I. S., Mangaraj, B. B. ve Imtiaj, S., 2008, Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence, *Progress In Electromagnetics Research C*, 1, 143-157.

Daylight, E. G., 2015, Towards a Historical Notion of 'Turing—the Father of Computer Science', *History and Philosophy of Logic*, 36 (3), 205-228.

De Castro, L. N. ve Von Zuben, F. J., 1999, Artificial immune systems: Part I–basic theory and applications, *Universidade Estadual de Campinas, Dezembro de, Tech. Rep*, 210 (1).

De Castro, L. N. ve Von Zuben, F. J., 2002, Learning and optimization using the clonal selection principle, *IEEE transactions on evolutionary computation*, 6 (3), 239-251.

De Castro, L. N. ve Timmis, J. I., 2003, Artificial immune systems as a novel soft computing paradigm, *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 7 (8), 526-544.

De Latil, P., 1956, Thinking by Machine: A Study of Cybernetics, Houghton Mifflin.

Deb, K., Pratap, A., Agarwal, S. ve Meyarivan, T., 2002, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE transactions on evolutionary computation*, 6 (2), 182-197.

Delice, Y., Aydoğan, E. K., Özcan, U. ve İlkay, M. S., 2017, A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing, *Journal of Intelligent Manufacturing*, 28 (1), 23-36.

Demir, A. ve Köse, U., 2016, Solving optimization problems via vortex optimization algorithm and cognitive development optimization algorithm, *Broad Research in Artificial Intelligence and Neuroscience*, 7 (4), 23-42.

Deng, Y., Xiang, J. ve Ou, Z., 2012, SVR with hybrid chaotic genetic algorithm for short-term traffic flow forecasting, *Natural Computation (ICNC), 2012 Eighth International Conference on*, 708-712.

Dhal, K. G., Quraishi, M. I. ve Das, S., 2017, An Improved Cuckoo Search based Optimal Ranged Brightness Preserved Histogram Equalization and Contrast Stretching Method, *International Journal of Swarm Intelligence Research (IJSIR)*, 8 (1), 1-29.

Diao, X., Zhu, H. ve Zhang, D., 2016, Speed and Displacement Control System of Bearingless Brushless DC Motor Based on Improved Bacterial Foraging Algorithm, *MATEC Web of Conferences*, 07005.

Dieterich, J. M. ve Hartke, B., 2012, Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization, *Applied Mathematics*, 3, 1552-1564.

Diwei, Y., Shaohua, L. ve Yuantong, S., 2007, Wavelet neural network learning algorithm based on particle swarm optimization, *Chinese Journal of Engineering Geophysics*, 6, 003.

Djalal, M. R., 2005, Algoritma Cerdas Part 1: Cuckoo Search Algorithm (CSA). Catatan Kecil – Wandy Notes, http://www.wandynotes.com/2016/07/algoritma-cerdas-part-1-cuckoo-search_15.html: [Erişim: 02.02.2016].

Dong, W., Shi, G. ve Zhang, L., 2007, Immune memory clonal selection algorithms for designing stack filters, *Neurocomputing*, 70 (4), 777-784.

Dorigo, M., Maniezzo, V. ve Colorni, A., 1991, Positive Feedback as a Search Strategy. Technical Report 91-016, *Dipartimento di Elettronica, Politecnico di Milano, Italy*.

Dorigo, M., Maniezzo, V. ve Colorni, A., 1996, Ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26 (1), 29-41.

Dorigo, M. ve Birattari, M., 2007, Swarm Intelligence. Scholarpedia, 2(9), 1462, http://www.scholarpedia.org/article/Swarm_intelligence: [Erişim: 16.03.2016].

Duan, H., Xu, C., Liu, S. ve Shao, S., 2010, Template matching using chaotic imperialist competitive algorithm, *Pattern Recognition Letters*, 31 (13), 1868-1875.

Dubey, H. M., Pandit, M. ve Panigrahi, B. K., 2015, A biologically inspired modified flower pollination algorithm for solving economic dispatch problems in modern power systems, *Cognitive Computation*, 7 (5), 594-608.

Dutta, A., Das, S., Tudu, B. ve Mandal, K. K., 2016, A novel improved algorithm using Cuckoo Search for Economic Load Dispatch, *Power Electronics, Intelligent Control and Energy Systems (ICPEICES), IEEE International Conference on*, 1-6.

Eberhart, R. ve Kennedy, J., 1995, A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 39-43.

Egbert, M. D., Barandiaran, X. E. ve Di Paolo, E. A., 2010, A minimal model of metabolism-based chemotaxis, *PLoS Comput Biol*, 6 (12), e1001004.

El-henawy, I. ve Ismail, M., 2014, An improved chaotic flower pollination algorithm for solving large integer programming problems, *International Journal of Digital Content Technology and Its Applications*, 8 (3), 72.

Elmas, Ç., 2003a, Bulanık Mantık Denetleyiciler (Kuram, Uygulama, Sinirsel Bulanık Mantık), *Ankara*, Seçkin Yayıncılık.

Elmas, Ç., 2003b, Yapay Sinir Ağları (Kuram, Mimari, Uygulama), *Ankara*, Seçkin Yayıncılık.

Elmas, Ç., 2011, Yapay Zekâ Uygulamaları, *Ankara*, Seçkin Yayıncılık.

Elsayed, S. M., Sarker, R. A. ve Essam, D. L., 2013, An improved self-adaptive differential evolution algorithm for optimization problems, *IEEE Transactions on Industrial Informatics*, 9 (1), 89-99.

Enayatifar, R., Abdullah, A. H. ve Lee, M., 2013, A weighted discrete imperialist competitive algorithm (WDICA) combined with chaotic map for image encryption, *Optics and Lasers in Engineering*, 51 (9), 1066-1077.

Engelbrecht, A. P., 2006, Fundamentals of Computational Swarm Intelligence, John Wiley & Sons.

Fisher, R. A., 1958:1930, The Genetical Theory of Natural Selection, (1930: Clarendon Press) Oxford University Press.

Fister Jr, I., Fister, D. ve Fister, I., 2013, A comprehensive review of cuckoo search: variants and hybrids, *International Journal of Mathematical Modelling and Numerical Optimisation*, 4 (4), 387-409.

Floudas, C. A., 2013, Deterministic Global Optimization: Theory, Methods and Applications, Springer Science & Business Media.

Fogel, D. B., 2000, What is evolutionary computation?, *IEEE Spectrum*, 37 (2), 26-32.

Fogel, D. B., 2006, Evolutionary Computation: Toward A New Philosophy of Machine Intelligence, John Wiley & Sons.

Fonseca, C. M. ve Fleming, P. J., 1995, An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary computation*, 3 (1), 1-16.

Franklin, J. V. ve Kumar, K. S., 2016, Modified Bacterial Foraging Optimization (MBFO) to Improve Quality of Experience in LTE Networks, *Asian Journal of Research in Social Sciences and Humanities*, 6 (8), 550-562.

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

Gardner, H., 1983, Frames of Mind: The Theory of Multiple Intelligences, *New York, NY*, Basic Books.

Gardner, H., 1991, The Unschooled Mind: How Children Think and How Schools Should Teach, *New York, NY*, Basic Books.

Garnier, S., Gautrais, J. ve Theraulaz, G., 2007, The biological principles of swarm intelligence, *Swarm Intelligence*, 1 (1), 3-31.

Gazi, V. ve Passino, K. M., 2011, Bacteria Foraging Optimization, In: Swarm Stability and Optimization, Eds: Springer, p. 233-249.

Geem, Z. W., Kim, J. H. ve Loganathan, G., 2001, A new heuristic optimization algorithm: harmony search, *Simulation*, 76 (2), 60-68.

Gen, M. ve Cheng, R., 2000, Genetic Algorithms and Engineering Optimization, John Wiley & Sons.

Gendreau, M. ve Potvin, J.-Y., 2010, Handbook of Metaheuristics, Springer.

Ghosh, A., Das, S., Chowdhury, A. ve Giri, R., 2011, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, *Information Sciences*, 181 (18), 3749-3765.

Gilat, A., 2016, MATLAB: An introduction with Applications, John Wiley & Sons.

Glaeser, G. ve Strouhal, E., 2000, Von Kempelen's Chess-Playing Pseudo-Automaton and Zu Racknitz's Explanation of Its Controls, in 1789, *International Symposium on History of Machines and Mechanisms Proceedings HMM 2000*, 351-360.

Glover, B. J., 2007, Understanding Flowers and Flowering: An Integrated Approach, Oxford University Press.

Goldberg, D. E. ve Deb, K., 1991, A comparative analysis of selection schemes used in genetic algorithms, *Foundations of genetic algorithms*, 1, 69-93.

Goldberg, D. E., 2006, Genetic Algorithms, Pearson Education India.

Gong, M., Jiao, L. ve Zhang, L., 2010, Baldwinian learning in clonal selection algorithm for optimization, *Information Sciences*, 180 (8), 1218-1236.

Gong, W., Cai, Z. ve Zhu, L., 2009, An efficient multiobjective differential evolution algorithm for engineering design, *Structural and Multidisciplinary Optimization*, 38 (2), 137-157.

Greensmith, J., Aickelin, U. ve Cayzer, S., 2005, Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection, *International Conference on Artificial Immune Systems*, 153-167.

Greensmith, J., Whitbrook, A. ve Aickelin, U., 2010, Artificial immune systems, In: Handbook of Metaheuristics, Eds: Springer, p. 421-448.

Gudise, V. G. ve Venayagamoorthy, G. K., 2003, Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks, *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, 110-117.

Güler, G. ve Köse, U., 2016, Intelligent Optimization for Logistics. Proceedings of the ScieConf 2016 – International Virtual Conference on Advanced Scientific Results**:** 131-137.

Güler, I. ve Übeyli, E. D., 2005, Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients, *Journal of neuroscience methods*, 148 (2), 113-121.

Hagan, M. T. ve Menhaj, M. B., 1994, Training feedforward networks with the Marquardt algorithm, *IEEE transactions on Neural Networks*, 5 (6), 989-993.

Hamet, P. ve Tremblay, J., 2017, Artificial intelligence in medicine, *Metabolism*, 69, S36-S40.

Hassanien, A. E. ve Emary, E., 2016, Swarm Intelligence: Principles, Advances, and Applications, CRC Press.

Hassoun, M. H., 1995, Fundamentals of Artificial Neural Networks, MIT Press.

Ho, R., 2013, Escape Local Optimum Trap. Pragmatic Programming Techniques – Ho Ricky Blog, http://horicky.blogspot.com.tr/2013/12/escape-local-optimum-trap.html: [Erişim: 20.11.2015].

Ho, W.-H., Chou, J.-H. ve Guo, C.-Y., 2010, Parameter identification of chaotic systems using improved differential evolution algorithm, *Nonlinear Dynamics*, 61 (1-2), 29-41.

Ho, Y.-C. ve Pepyne, D. L., 2002, Simple explanation of the no-free-lunch theorem and its implications, *Journal of optimization theory and applications*, 115 (3), 549-570.

Hoang, A.-Q. ve Pham, M.-T., 2016, Comparing Modified PSO Algorithms for MRS in Unknown Environment Exploration, *International Conference on Advances in Information and Communication Technology*, 207-216.

Hoffman, K. L., 2000, Combinatorial optimization: Current successes and directions for the future, *Journal of computational and applied mathematics*, 124 (1), 341-360.

Holland, J., 1995, Hidden Order: How Adaptation Builds Complexity, Basic Books.

Holland, J. H., 1992:1975, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, (1975: University of Michigan Press) MIT Press.

Holland, J. H., 2012, Genetic Algorithms. Scholarpedia, 7(12), 1482, http://www.scholarpedia.org/article/Genetic_algorithms: [Erişim: 17.01.2016].

Hong, W.-C., 2009, Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model, *Energy Conversion and Management*, 50 (1), 105-117.

Hong, W.-C., Lai, C.-Y., Hung, W.-M. ve Dong, Y., 2010, Electric load forecasting by SVR with chaotic ant swarm optimization, *Cybernetics and Intelligent Systems (CIS), 2010 IEEE Conference on*, 102-107.

Horn, J., Nafpliotis, N. ve Goldberg, D. E., 1994, A niched Pareto genetic algorithm for multiobjective optimization, *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 82-87.

Hornik, K., Stinchcombe, M. ve White, H., 1989, Multilayer feedforward networks are universal approximators, *Neural networks*, 2 (5), 359-366.

Horst, R. ve Tuy, H., 2013, Global optimization: Deterministic approaches, Springer Science & Business Media.

Hou, W. ve Zhang, Z., 2016, A method of test points optimization selection based on improved bacterial foraging algorithm, *Prognostics and System Health Management Conference (PHM-Chengdu), 2016*, 1-5.

Hovland, C. I., 1960, Computer simulation of thinking, *American Psychologist*, 15 (11), 687.

Ifrah, G., Harding, E. F., Bellos, D. ve Wood, S., 2000, The universal history of computing: From the abacus to quantum computing, John Wiley & Sons, Inc..

Ilonen, J., Kamarainen, J.-K. ve Lampinen, J., 2003, Differential evolution training algorithm for feed-forward neural networks, *Neural Processing Letters*, 17 (1), 93-105.

INFORMS Computing Society, 2016, The nature of mathematical programming, *Mathematical Programming Glossary*.

Jaeger, H., 2016, Artificial intelligence: Deep neural reasoning, *Nature*, 538 (7626), 467-468.

Jamil, M. ve Yang, X.-S., 2013, A literature survey of benchmark functions for global optimisation problems, *International Journal of Mathematical Modelling and Numerical Optimisation*, 4 (2), 150-194.

Jang, J.-S., 1992, Self-learning fuzzy controllers based on temporal backpropagation, *IEEE transactions on Neural Networks*, 3 (5), 714-723.

Jang, J.-S., 1993, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE transactions on systems, man, and cybernetics*, 23 (3), 665-685.

Jang, J. S. R., Sun, C. T. ve Mizutani, E., 1997, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, *New Jersey*, Prentice Hall.

Janikow, C. Z. ve Michalewicz, Z., 1991, An experimental comparison of binary and floating point representations in genetic algorithms, *ICGA*, 31-36.

Javed, I., Ayyaz, M. ve Mehmood, W., 2007, Efficient training data reduction for SVM based handwritten digits recognition, *Electrical Engineering, 2007. ICEE'07. International Conference on*, 1-4.

Jia, D., Zheng, G. ve Khan, M. K., 2011, An effective memetic differential evolution algorithm based on chaotic local search, *Information Sciences*, 181 (15), 3175-3187.

Johansson, R., 2008, Genetic Programming: Evolution of Mona Lisa. Roger Johansson Blog, https://rogerjohansson.blog/2008/12/07/genetic-programming-evolution-of-mona-lisa/: [Erişim: 11.01.2017].

Johnson, B., Skibo, C. ve Young, M., 2003, Inside Visual Studio .Net, Microsoft Press.

Kalemis, D., 2013, The Fundamental Concepts of Object-Oriented Programming, CreateSpace Independent Publishing.

Kallrath, J. ve Schreieck, A., 1995, Discrete optimisation and real world problems, *High-performance computing and networking*, 351-359.

Kaplan, W., 2011, Maxima and minima with applications: practical optimization and duality, John Wiley & Sons.

Karaboğa, D., 2005, An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, *Faculty of Engineering, Dept. of Computer Engineering, Erciyes University*, *Kayseri*.

Karaboğa, D., Akay, B. ve Öztürk, C., 2007, Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks, *International Conference on Modeling Decisions for Artificial Intelligence*, 318-329.

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

Karaboğa, D. ve Akay, B., 2009, A comparative study of artificial bee colony algorithm, *Applied mathematics and computation*, 214 (1), 108-132.

Karaboğa, D., 2010, Artificial Bee Colony Algorithm. Scholarpedia, 5(3), 6915, http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm: [Erişim: 14.01.2016].

Karaboğa, D. ve Akay, B., 2011, A modified artificial bee colony (ABC) algorithm for constrained optimization problems, *Applied Soft Computing*, 11 (3), 3021-3031.

Karaboğa, D. ve Görkemli, B., 2011, A combinatorial artificial bee colony algorithm for traveling salesman problem, *Innovations in intelligent systems and applications (inista), 2011 international symposium on*, 50-53.

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

Karaboğa, D., 2014, Yapay Zekâ Optimizasyon Algoritmaları, *Ankara*, Nobel Akademik Yayıncılık.

Karaboğa, D., Görkemli, B., Öztürk, C. ve Karaboğa, N., 2014, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, *Artificial Intelligence Review*, 42 (1), 21-57.

Karaçay, T., 2016, Java ile Nesne Programlama, *Ankara*, Seçkin Yayıncılık.

Kaymaz, İ., 2016, Optimizasyon Teknikleri – Ders Notları (Atatürk Üniversitesi – Fen Bilimleri Enstitüsü), http://194.27.49.11/makine/ikaymaz/optimizasyon/: [Erişim: 29.12.2016].

Kearfott, R. B., 2013, Rigorous global search: continuous problems, Springer Science & Business Media.

Kemeny, J. G., 1955, Man Viewed as a Machine, *Scientific American*, 192, 58-67.

Kennedy, J., 2006, Swarm intelligence, In: Handbook of nature-inspired and innovative computing, Eds: Springer, p. 187-219.

Kennedy, J., 2011, Particle swarm optimization, In: Encyclopedia of machine learning, Eds: Springer, p. 760-766.

Kennedy, J. F., Kennedy, J., Eberhart, R. C. ve Shi, Y., 2001, Swarm Intelligence, Morgan Kaufmann.

Klir, G. ve Yuan, B., 1995, Fuzzy Sets and Fuzzy Logic, *New Jersey*, Prentice Hall.

Konar, D., Bhattacharyya, S., Sharma, K., Sharma, S. ve Pradhan, S. R., 2017, An Improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for Scheduling of Real-Time Task in Multiprocessor System, *Applied Soft Computing*.

Kora, P. ve Kalva, S. R. K., 2017, Detection of Bundle Branch Block using Adaptive Bacterial Foraging Optimization and Neural Network, *Egyptian Informatics Journal*, 18 (1), 67-74.

Korbicz, J., Kościelny, J. M., Kowalczuk, Z. ve Cholewa, W., 2012, Fault diagnosis: models, artificial intelligence, applications, Springer Science & Business Media.

Korenaga, T., Hatanaka, T. ve Uosaki, K., 2007, Performance improvement of particle swarm optimization for high-dimensional function optimization, *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 3288-3293.

Köller, J., 2017, Spirals. Mathematische Basteleien, http://www.mathematische-basteleien.de/spiral.htm: [Erişim: 14.11.2016].

Köse, U. ve Arslan, A., 2014, On the idea of a new artificial intelligence based optimization algorithm inspired from the nature of vortex, *Broad Research in Artificial Intelligence and Neuroscience*, 5 (1-4), 60-66.

Köse, U., 2015, Present State of Swarm Intelligence and Future Directions, In: Encyclopedia of Information Science and Technology, Third Edition, Eds: IGI Global, p. 239-252.

Köse, U. ve Arslan, A., 2015, Realizing an optimization approach inspired from piaget's theory on cognitive development, *Broad Research in Artificial Intelligence and Neuroscience*, 6 (1-2), 14-21.

Köse, U., Güraksın, G. E. ve Deperlioğlu, Ö., 2015, Diabetes determination via vortex optimization algorithm based support vector machines, *Medical Technologies National Conference (TIPTEKNO), 2015*, 1-4.

Köse, U., Güraksın, G. E. ve Deperlioğlu, Ö., 2016, Cognitive development optimization algorithm based support vector machines for determining diabetes, *Broad Research in Artificial Intelligence and Neuroscience*, 7 (1), 80-90.

Köse, U. ve Arslan, A., 2016a, Intelligent E-Learning System for Improving Students' Academic Achievements in Computer Programming Courses, *International Journal of Engineering Education*, 32 (1), 185-198.

Köse, U. ve Arslan, A., 2016b, Forecasting chaotic time series via ANFIS supported by vortex optimization algorithm: Applications on electroencephalogram time series, *Arabian Journal for Science and Engineering*, (Online Early View), 1-12.

Köse, U. ve Arslan, A., 2017a, Optimization of self-learning in computer engineering courses: An intelligent software system supported by artificial neural network and vortex optimization algorithm, *Computer Applications in Engineering Education*, 25 (1), 142-156.

Köse, U. ve Arslan, A., 2017b, Global optimization algorithms with in-system optimization approach, *International Journal of Swarm Intelligence Research*, (Basımda).

Köse, U. ve Arslan, A., 2017c, Time series prediction with a hybrid system formed by artificial neural network and cognitive development optimization algorithm, *Scientia Iranica*, (Basımda).

Kramer, O., 2017, Genetic Algorithm Essentials, Springer.

Krause, J., Ruxton, G. D. ve Krause, S., 2010, Swarm intelligence in animals and humans, *Trends in ecology & evolution*, 25 (1), 28-34.

Krenker, A., Kos, A. ve Bešter, J., 2011, Introduction to the artificial neural networks, INTECH Open Access Publisher.

Kubat, C., 2016, MATLAB – Yapay Zekâ ve Mühendislik Uygulamaları, *İstanbul*, Abaküs Kitap.

Kursawe, F., 1990, A variant of evolution strategies for vector optimization, *International Conference on Parallel Problem Solving from Nature*, 193-197.

Landau, S. ve Everitt, B., 2004, A handbook of statistical analyses using SPSS, Chapman & Hall/CRC Boca Raton, FL.

Lee, K. Y. ve El-Sharkawi, M. A., 2008, Modern heuristic optimization techniques: theory and applications to power systems, John Wiley & Sons.

Lei, X., Huang, X. ve Zhang, A., 2010, Improved artificial bee colony algorithm and its application in data clustering, *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, 514-521.

Li, B.-h., Qu, H.-y., Lin, T.-y., Hou, B.-c., Zhai, X., Shi, G.-q., Zhou, J.-h. ve Ruan, C., 2017, A swarm intelligence design based on a workshop of meta-synthetic engineering, *Frontiers of Information Technology & Electronic Engineering*, 18 (1), 149-152.

Li, H. ve Li, P., 2013, Self-adaptive ant colony optimization for construction time-cost optimization, *Kybernetes*, 42 (8), 1181-1194.

Li, S.-X. ve Wang, J.-S., 2017, Improved Cuckoo Search Algorithm with Novel Searching Mechanism for Solving Unconstrained Function Optimization Problem, *IAENG International Journal of Computer Science*, 44 (1), 8-12.

Liang, J., Qu, B., Suganthan, P. ve Chen, Q., 2014, Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Technical Report, *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, Nanyang Technological University, Singapore*.

Liao, Q., Zhou, S., Shi, H. ve Shi, W., 2017, Parameter estimation of nonlinear systems by dynamic cuckoo search, *Neural Computation*, 29 (4), 1103-1123.

Lin, Z.-Y. ve Li, Y.-X., 2009, Hybrid differential evolutionary algorithm for solving function optimization problems, *Journal of System Simulation*, 13, 009.

Ling, S.-H. ve Leung, F. H., 2007, An improved genetic algorithm with average-bound crossover and wavelet mutation operations, *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11 (1), 7-31.

Liu, H., Gao, L., Kong, X. ve Zheng, S., 2013, An improved artificial bee colony algorithm, *Control and Decision Conference (CCDC), 2013 25th Chinese*, 401-404.

Liu, J. ve Tsui, K. C., 2006, Toward nature-inspired computing, *Communications of the ACM*, 49 (10), 59-64.

Liu, Y. ve Passino, K., 2002, Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors, *Journal of optimization theory and applications*, 115 (3), 603-628.

Luger, G. F., 2004, Inteligência Artificial-: Estruturas e estratégias para a solução de problemas complexos, Bookman.

Makridakis, S., 2017, The Forthcoming Artificial Intelligence (AI) Revolution: Its Impact on Society and Firms. Neapolis University - HEPHAESTUS Repository, http://hephaestus.nup.ac.cy/bitstream/handle/11728/9254/THE_FORTHCOMING...2017_Full.pdf?sequence=1&isAllowed=y: [Erişim: 17.01.2017].

Mandelbrot, B. B., 1982, The Fractal Geometry of Nature, *New York*, W. H. Freeman.

Mange, D. ve Tomassini, M., 1998, Bio-inspired computing machines: Towards novel computational architectures, PPUR presses polytechniques.

Marichelvam, M., Prabaharan, T. ve Yang, X.-S., 2014, Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan, *Applied Soft Computing*, 19, 93-101.

Markowitz, H., 1959, Portfolio Selection: Efficient Diversification of Investments, *New York*, Wiley.

Marquer, F., 2016, Projet D'animation Numérique. Frédéric Marquer Portfolio, https://fredericmarquer.com/2016/04/18/projet-danimation-numerique/: [Erişim: 11.01.2017].

Marrow, P., 2000, Nature-inspired computing technology and applications, *BT Technology Journal*, 18 (4), 13-23.

Marzband, M., Parhizi, N. ve Adabi, J., 2015, Optimal energy management for stand-alone microgrids based on multi-period imperialist competition algorithm considering uncertainties: experimental validation, *International transactions on electrical energy systems*.

Mashwani, W. K., Salhi, A., Yeniay, O., Hussian, H. ve Jan, M., 2017, Hybrid non-dominated sorting genetic algorithm with adaptive operators selection, *Applied Soft Computing*, 56, 1-18.

Mayr, E., 1988, Toward a new philosophy of biology: Observations of an evolutionist, 211, Harvard University Press.

Mays, W., 1952, Can machines think?, *Philosophy*, 27 (101), 148-162.

McCarthy, J., Minsky, M. L., Rochester, N. ve Shannon, C. E., 2006:1955, A proposal for the dartmouth summer research project on artificial intelligence, 31 Ağustos 1955. AI Magazine, 27(4), 12, http://www.aaai.org/ojs/index.php/aimagazine/article/view/1904/1802: [Erişim: 15.12.2015].

McCarthy, J., 2007, What is Artificial Intelligence?. Stanford – Formal Reasoning Group, http://www-formal.stanford.edu/jmc/whatisai/whatisai.html: [Erişim: 10.12.2015].

McCorduck, P., Minsky, M., Selfridge, O. ve Simon, H. A., 1977, History of Artificial Intelligence. Proceedings of the 5th International Joint Conference on Artificial Intelligence, Volume 2 (pp. 951-954). Morgan Kaufmann Publishers Inc., http://ijcai.org/Proceedings/77-2/Papers/083.pdf: [Erişim: 16.01.2016].

MedLibrary, 2016, Mathematical Optimization. MedLibrary – Open Source Encyc., http://medlibrary.org/medwiki/Mathematical_optimization: [Erişim: 13.12.2015].

Meng, O. K., Pauline, O., Kiong, S. C., Wahab, H. A. ve Jafferi, N., 2017, Application of Modified Flower Pollination Algorithm on Mechanical Engineering Design Problem, *IOP Conference Series: Materials Science and Engineering*, 012032.

Meyers, J., 1993, A Short History of the Computer, https://classes.soe.ucsc.edu/cmps060n/Spring07/shorthistory.pdf: [Erişim: 15.01.2016].

Michalski, R. S., Carbonell, J. G. ve Mitchell, T. M., 2013, Machine learning: An artificial intelligence approach, Springer Science & Business Media.

Miller, H. R., 2011, Optimization: foundations and applications, John Wiley & Sons.

Mirhoseini, S. H., Hosseini, S. M., Ghanbari, M. ve Ahmadi, M., 2014, A new improved adaptive imperialist competitive algorithm to solve the reconfiguration problem of distribution systems for loss reduction and voltage profile improvement, *International Journal of Electrical Power & Energy Systems*, 55, 128-143.

Mishra, S., 2006, Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method. Munich Personal RePEc Archive, https://mpra.ub.uni-muenchen.de/2718/1/MPRA_paper_2718.pdf: [Erişim: 17.02.2016].

Mitchell, M., 1998, An Introduction to Genetic Algorithms, MIT Press.

Mohan, U., 2008, Bio Inspired Computing, *Division of computer science SOE. CUSAT*.

Mohapatra, P., Chakravarty, S. ve Dash, P. K., 2015, An improved cuckoo search based extreme learning machine for medical data classification, *Swarm and Evolutionary Computation*, 24, 25-49.

Molga, M. ve Smutnicki, C., 2005, Test Functions for Optimization Needs, http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf: [Erişim: 16.02.2016].

Montana, D. J. ve Davis, L., 1989, Training feedforward neural networks using genetic algorithms, *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 1*, 762-767.

Mozafari, H., Abdi, B. ve Ayob, A., 2012, Optimization of adhesive-bonded fiber glass strip using imperialist competitive algorithm, *Procedia Technology*, 1, 194-198.

Mucherino, A. ve Seref, O., 2009, Modeling and solving real-life global optimization problems with meta-heuristic methods, In: Advances in Modeling Agricultural Systems, Eds: Springer, p. 403-419.

Mühlenbein, H., Schomisch, M. ve Born, J., 1991, The parallel genetic algorithm as function optimizer, *Parallel computing*, 17 (6-7), 619-632.

Nabil, E., 2016, A modified flower pollination algorithm for global optimization, *Expert Systems with Applications*, 57, 192-203.

Nabiyev, V. V., 2012, Yapay Zekâ (Problemler – Yöntemler – Algoritma), *Ankara*, Seçkin Yayıncılık.

Nazari-Shirkouhi, S., Eivazy, H., Ghodsi, R., Rezaie, K. ve Atashpaz-Gargari, E., 2010, Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm, *Expert Systems with Applications*, 37 (12), 7615-7626.

NEOS Guide, 2017b, Constrained Optimization. NEOS Guide.org, https://neos-guide.org/content/constrained-optimization: [Erişim: 10.01.2016].

NEOS Guide, 2017e, Unconstrained Optimization. NEOS Guide.org, https://neos-guide.org/content/unconstrained-optimization: [Erişim: 10.01.2016].

Nerode, A., 2016, Foreword. Advances in Comp. Sci. and Eng.: Texts – Vol.9: The Human Face of Computing (Ed. C. S. Calude). Imperial College Press, London, http://www.worldscientific.com/doi/pdf/10.1142/9781783266449_fmatter: [Erişim: 09.12.2015].

Neumaier, A., 1996, Global Optimization, http://www.mat.univie.ac.at/~neum/glopt.html: [Erişim: 10.01.2016].

Ni, J., Wang, K., Huang, H., Wu, L. ve Luo, C., 2016, Robot path planning based on an improved genetic algorithm with variable length chromosome, *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, 145-149.

Niewiadomski, R. ve Anderson, D., 2016, The Rise of Artificial Intelligence: Its Impact on Labor, *Strategic Imperatives and Core Competencies in the Era of Robotics and Artificial Intelligence*, 29.

Niknam, T., Fard, E. T., Pourjafarian, N. ve Rousta, A., 2011, An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering, *Engineering Applications of Artificial Intelligence*, 24 (2), 306-317.

Nilsson, N. J., 2014, Principles of Artificial Intelligence, Morgan Kaufmann.

Nino, J., 2007, An introduction to programming and object-oriented design using JAVA, John Wiley & Sons, Inc..

Niu, B., Wang, H., Wang, J. ve Tan, L., 2013a, Multi-objective bacterial foraging optimization, *Neurocomputing*, 116, 336-345.

Nourmohammadi, A., Zandieh, M. ve Tavakkoli-Moghaddam, R., 2013, An imperialist competitive algorithm for multi-objective U-type assembly line design, *Journal of Computational Science*, 4 (5), 393-400.

O'Hare, G. M. ve Jennings, N., 1996, Foundations of distributed artificial intelligence, John Wiley & Sons.

Osyczka, A. ve Kundu, S., 1995, A genetic algorithm-based multicriteria optimization method, *Proc. 1st World Congr. Struct. Multidisc. Optim*, 909-914.

Ouaarab, A., Ahiod, B. ve Yang, X.-S., 2014, Discrete cuckoo search algorithm for the travelling salesman problem, *Neural Computing and Applications*, 24 (7-8), 1659-1669.

Ören, T., 2001, Advances in computer and information sciences: From abacus to holonic agents, *Turkish Journal of Electrical Engineering and Computer Sciences*, 9 (1), 63-70.

Özkan, Y., 2016, Veri Madenciliği Yöntemleri, *İstanbul*, Papatya Yayıncılık.

Öztemel, E., 2003, Yapay Sinir Ağları, *İstanbul*, Papatya Yayıncılık.

Palomo-Romero, J. M., Salas-Morera, L. ve García-Hernández, L., 2017, An island model genetic algorithm for unequal area facility layout problems, *Expert Systems with Applications*, 68, 151-162.

Pan, Y., 2016, Heading toward Artificial Intelligence 2.0, *Engineering*, 2 (4), 409-413.

Panda, A., Tripathy, M., Barisal, A. ve Prakash, T., 2017, A modified bacteria foraging based optimal power flow framework for Hydro-Thermal-Wind generation system in the presence of STATCOM, *Energy*, 124, 720-740.

Pandit, N., Tripathi, A., Tapaswi, S. ve Pandit, M., 2012, An improved bacterial foraging algorithm for combined static/dynamic environmental economic dispatch, *Applied Soft Computing*, 12 (11), 3500-3513.

Papadimitriou, C. H. ve Steiglitz, K., 1982:1998, Combinatorial optimization: algorithms and complexity, (1982 Courier Corporation) Dover Publications.

Parpinelli, R. S. ve Lopes, H. S., 2011, New inspirations in swarm intelligence: a survey, *International Journal of Bio-Inspired Computation*, 3 (1), 1-16.

Passino, K. M., 2002, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE control systems*, 22 (3), 52-67.

Passino, K. M., 2012, Bacterial foraging optimization, *Innovations and Developments of Swarm Intelligence Applications*, 219-233.

Patrascu, M., Stancu, A. F. ve Pop, F., 2014, HELGA: a heterogeneous encoding lifelike genetic algorithm for population evolution modeling and simulation, *Soft Computing*, 18 (12), 2565-2576.

Pavlyukevich, I., 2007, Lévy flights, non-local search and simulated annealing, *Journal of Computational Physics*, 226 (2), 1830-1844.

Payne, R. B., 2005, The Cuckoos, Oxford University Press.

Peng, Y. ve Lu, B.-L., 2015, Hybrid learning clonal selection algorithm, *Information Sciences*, 296, 128-146.

Phansalkar, V. ve Sastry, P., 1994, Analysis of the back-propagation algorithm with momentum, *IEEE transactions on Neural Networks*, 5 (3), 505-506.

Phoenix Integration, 2017, Bird Problem (Constrained). Phoenix Integration – Benchmark Report, http://www.phoenix-int.com/software/benchmark_report/bird_constrained.php: [Erişim: 17.02.2016].

Piaget, J., 1964, Part I: Cognitive development in children: Piaget development and learning, *Journal of research in science teaching*, 2 (3), 176-186.

Piaget, J., 1973, Main trends in psychology, *London*, George Allen & Unwin.

Piaget, J., 2000, Piaget's Theory of Cognitive Development, In: Childhood cognitive development: The essential readings, Eds, p. 33-47.

Picheny, V., Wagner, T. ve Ginsbourger, D., 2013, A benchmark of kriging-based infill criteria for noisy optimization, *Structural and Multidisciplinary Optimization*, 48 (3), 607-626.

Pinsky, L., 1951, Do machines think about machines thinking?, *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 143.

Pintea, C.-M., 2014, Bio-inspired computing, In: Advances in Bio-inspired Computing for Combinatorial Optimization Problems, Eds: Springer, p. 3-19.

Poli, R., 2008, Analysis of the publications on the applications of particle swarm optimisation, *Journal of Artificial Evolution and Applications*, 2008.

Potts, J. C., Giddens, T. D. ve Yadav, S. B., 1994, The development and evaluation of an improved genetic algorithm based on migration and artificial selection, *IEEE transactions on systems, man, and cybernetics*, 24 (1), 73-86.

Price, K. V., 1996, Differential evolution: a fast and simple numerical optimizer, *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*, 524-527.

Qin, A. K., Huang, V. L. ve Suganthan, P. N., 2009, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE transactions on evolutionary computation*, 13 (2), 398-417.

Qu, C. ve He, W., 2016, A Cuckoo Search Algorithm with Complex Local Search Method for Solving Engineering Structural Optimization Problem, *MATEC Web of Conferences*.

Quan, H. ve Shi, X., 2008, On the analysis of performance of the improved artificial-bee-colony algorithm, *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, 654-658.

Rafiei, M., Niknam, T. ve Khooban, M. H., 2016, Probabilistic electricity price forecasting by improved clonal selection algorithm and wavelet preprocessing, *Neural Computing and Applications*, 1-13.

Rahnamayan, S., Tizhoosh, H. R. ve Salama, M. M., 2008, Opposition-based differential evolution, *IEEE transactions on evolutionary computation*, 12 (1), 64-79.

Rajarathinam, K., Gomm, J. B., Yu, D. ve Abdelhadi, A. S., 2017, An improved search space resizing method for model identification by standard genetic algorithm, *Systems Science & Control Engineering*, 5 (1), 117-128.

Rao, B. S. ve Vaisakh, K., 2013, Multi-objective adaptive Clonal selection algorithm for solving environmental/economic dispatch and OPF problems with load uncertainty, *International Journal of Electrical Power & Energy Systems*, 53, 390-408.

Rao, S. S., 2009, Classical optimization techniques, In: Engineering Optimization: Theory and Practice, Eds, p. 63-118.

Rao, S. S. ve Rao, S. S., 2009, Engineering optimization: theory and practice, John Wiley & Sons.

Rastrigin, L. A., 1974, Extremal control systems, *Theoretical foundations of engineering cybernetics series*, 3.

Ray, T. ve Liew, K. M., 2002, A swarm metaphor for multiobjective design optimization, *Engineering optimization*, 34 (2), 141-153.

Reeves, W. T., 1983, Particle systems—a technique for modeling a class of fuzzy objects, *ACM Transactions on Graphics*, 2 (2), 91-108.

Rehman, O. U., Yang, S. ve Khan, S. U., 2017, A modified quantum based particle swarm optimization for engineering inverse problem, *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 36 (1).

Rentsch, T., 1982, Object oriented programming, *ACM Sigplan Notices*, 17 (9), 51-57.

Reynolds, A. M. ve Frye, M. A., 2007, Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search, *PloS one*, 2 (4), e354.

Reynolds, C. W., 1987, Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH computer graphics. 21**:** 25-34.

Ritchie, P., 2016, Introduction to Visual Studio 2015, In: Practical Microsoft Visual Studio 2015, Eds: Springer, p. 1-25.

Rosenbrock, H. H., 1960, An automatic method for finding the greatest or least value of a function, *The Computer Journal*, 3 (3), 175-184.

Ross, T. J., 2009, Fuzzy logic with engineering applications, John Wiley & Sons.

Salgotra, R. ve Singh, U., 2017, Application of mutation operators to flower pollination algorithm, *Expert Systems with Applications*, 79, 112-129.

Santhanam, T. ve Padmavathi, M. S., 2015, Application of k-means and genetic algorithms for dimension reduction by integrating SVM for diabetes diagnosis, *Procedia Computer Science*, 47, 76-83.

Shadbolt, N., 2004, Nature-inspired computing, *IEEE Intelligent Systems*, 19 (1), 2-3.

Shah-Hosseini, H., 2009, The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, *International Journal of Bio-Inspired Computation*, 1 (1-2), 71-79.

Shannon, C. E., 1950, A chess-playing machine, *Scientific American*, 182 (2), 48-51.

Shehab, M., Khader, A. T. ve Al-Betar, M. A., 2017, A survey on applications and variants of the cuckoo search algorithm, *Applied Soft Computing*, Online: 11 March/Mart 2017 (Basımda).

Shi, X., Wan, L., Lee, H., Yang, X., Wang, L. ve Liang, Y., 2003, An improved genetic algorithm with variable population-size and a PSO-GA based hybrid evolutionary algorithm, *Machine Learning and Cybernetics, 2003 International Conference on*, 1735-1740.

Shi, Y. ve Eberhart, R., 1998, A modified particle swarm optimizer, *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 69-73.

Shih, M. Y., Enríquez, A. C., Hsiao, T.-Y. ve Treviño, L. M. T., 2017, Enhanced differential evolution algorithm for coordination of directional overcurrent relays, *Electric Power Systems Research*, 143, 365-375.

Shlesinger, M. F., Klafter, J. ve Zumofen, G., 1995, Levy Flights: Chaotic, Turbulent, and Relativistic, *Fractals*, 3 (03), 491-497.

Siddique, M., 2015, Fluid Mechanics - Vortex flow and impulse momentum. Slideshare.net, https://www.slideshare.net/yourmohsin/fluid-mechanicsvortex-flow-and-impulse-momentum: [Erişim: 12.11.2016].

Simionescu, P. A., 2014, Computer-aided Graphing and Simulation Tools for AutoCAD Users, CRC Press.

Singer, D. G. ve Revenson, T. A., 1997, A Piaget primer: How a child thinks, ERIC.

Stefik, M. ve Bobrow, D. G., 1985, Object-oriented programming: Themes and variations, *Ai Magazine*, 6 (4), 40.

Storn, R. ve Price, K., 1995, Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces. Technical Report, *International Computer Science Institute*, *Berkeley, CA.*

Storn, R. ve Price, K., 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, 11 (4), 341-359.

Strong, A., 2016, Applications of artificial intelligence and associated technologies, *Science [ETEBMS-2016]*, 5, 6.

Stroustrup, B., 1988, What is object-oriented programming?, *IEEE software*, 5 (3), 10-20.

Sun, C., Zhou, H. ve Chen, L., 2012, Improved differential evolution algorithms, *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, 142-145.

Sun, W. ve Liang, Y., 2014, Least-squares support vector machine based on improved imperialist competitive algorithm in a short-term load forecasting model, *Journal of Energy Engineering*, 141 (4), 04014037.

Surjanovic, S. ve Bingham, D., 2015, Optimization Test Problems. Virtual Library of Simulation Experiments: Test Functions and Datasets, https://www.sfu.ca/~ssurjano/optimization.html: [Erişim: 16.02.2016].

Talatahari, S., Azar, B. F., Sheikholeslami, R. ve Gandomi, A. H., 2012a, Imperialist competitive algorithm combined with chaos for global optimization, *Communications in Nonlinear Science and Numerical Simulation*, 17 (3), 1312-1319.

Talatahari, S., Kaveh, A. ve Sheikholeslami, R., 2012b, Chaotic imperialist competitive algorithm for optimum design of truss structures, *Structural and Multidisciplinary Optimization*, 46 (3), 355-367.

Talbi, E.-G., 2009, Metaheuristics: from design to implementation, John Wiley & Sons.

Tamura, H. ve Tanno, K., 2009, Midpoint validation method for support vector machines with margin adjustment technique, *International Journal of Innovative Computing, Information and Control*, 5 (11), 4025-4032.

Tan, L.-Z., Tan, Y.-Y., Yun, G.-X. ve Zhang, C., 2016, An improved genetic algorithm based on k-means clustering for solving traveling salesman problem, *Computer Science, Technology and Application: Proceedings of the 2016 International Conference on Computer Science, Technology and Application (CSTA2016)*, 334.

Tang, K., Xiao, X., Wu, J., Yang, J. ve Luo, L., 2017, An improved multilevel thresholding approach based modified bacterial foraging optimization, *Applied Intelligence*, 46 (1), 214-226.

Tereshko, V., 2000, Reaction-diffusion model of a honeybee colony's foraging behaviour, *International Conference on Parallel Problem Solving from Nature*, 807-816.

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

Toğan, V. ve Daloğlu, A. T., 2008, An improved genetic algorithm with initial population strategy and self-adaptive member grouping, *Computers and Structures*, 86 (11), 1204-1218.

Torn, A. ve Zilinskas, A., 1989, Global Optimization, *New York*, Springer-Verlag.

Townsend, A., 2014, Constrained Optimization in Chebfun. Chebfun, http://www.chebfun.org/examples/opt/ConstrainedOptimization.html: [Erişim: 17.02.2016].

Tripathi, P. K., Bandyopadhyay, S. ve Pal, S. K., 2007, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, *Information Sciences*, 177 (22), 5033-5049.

Turing, A. M., 1950, Computing machinery and intelligence, *Mind*, 59 (236), 433-460.

Turing, A. M., 1995, Computing machinery and intelligence, *Brian Physiology and Psychology*, 213.

Turing, A. M., 2009, Computing machinery and intelligence, In: Parsing the Turing Test, Eds: Springer, p. 23-65.

Uğur, A. ve Kınacı, A. C., 2010, A web-based tool for teaching neural network concepts, *Computer Applications in Engineering Education*, 18 (3), 449-457.

Ulutaş, B. H. ve Kültürel-Konak, Ş., 2011, A review of clonal selection algorithm and its applications, *Artificial Intelligence Review*, 36 (2), 117-138.

Uzunköprü, S., 2015, Visual Studip 2015, *İstanbul*, Kodlab Yayıncılık.

Valenzuela, L., Valdez, F. ve Melin, P., 2017, Flower Pollination Algorithm with Fuzzy Approach for Solving Optimization Problems, In: Nature-Inspired Design of Hybrid Intelligent Systems, Eds: Springer, p. 357-369.

Valian, E., Mohanna, S. ve Tavakoli, S., 2011a, Improved cuckoo search algorithm for feedforward neural network training, *International Journal of Artificial Intelligence and Applications*, 2 (3), 36-43.

Valian, E., Mohanna, S. ve Tavakoli, S., 2011b, Improved cuckoo search algorithm for global optimization, *International Journal of Communications and Information Technology*, 1 (1), 31-44.

Valian, E., Tavakoli, S., Mohanna, S. ve Haghi, A., 2013, Improved cuckoo search for reliability optimization problems, *Computers and Industrial Engineering*, 64 (1), 459-468.

Vapnik, V., Golowich, S. E. ve Smola, A. J., 1997, Support vector method for function approximation, regression estimation and signal processing, *Advances in Neural Information Processing Systems*, 281-287.

Vikipedi, 2016a, Nesne Yönelimli Programlama, Vikipedi, Özgür Ansiklopedi, https://tr.wikipedia.org/wiki/Nesne_yönelimli_programlama: [Erişim: 06.02.2016].

Vikipedi, 2016b, Yordamsal Programlama, Vikipedi, Özgür Ansiklopedi, https://tr.wikipedia.org/wiki/Yordamsal_programlama: [Erişim: 06.02.2016].

Viswanathan, G., Bartumeus, F., Buldyrev, S. V., Catalan, J., Fulco, U., Havlin, S., Da Luz, M., Lyra, M., Raposo, E. ve Stanley, H. E., 2002, Lévy flight random

searches in biological phenomena, *Physica A: Statistical Mechanics and Its Applications*, 314 (1), 208-213.

Walia, G. S. ve Kapoor, R., 2014, Intelligent video target tracking using an evolutionary particle filter based upon improved cuckoo search, *Expert Systems with Applications*, 41 (14), 6315-6326.

Wang, R. ve Zhou, Y., 2014, Flower pollination algorithm with dimension by dimension improvement, *Mathematical Problems in Engineering*, 2014.

Weile, D. S. ve Michielssen, E., 1997, Genetic algorithm optimization applied to electromagnetics: A review, *IEEE Transactions on Antennas and Propagation*, 45 (3), 343-353.

Weise, T., 2009, Global Optimization Algorithms – Theory and Application, http://www.it-weise.de/projects/book.pdf: [Erişim: 11.01.2016].

Whitley, D., 1994, A genetic algorithm tutorial, *Statistics and computing*, 4 (2), 65-85.

Wikipedia, 2017a, Global Optimization, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Global_optimization: [Erişim: 12.01.2016].

Wikipedia, 2017c, MATLAB, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/MATLAB: [Erişim: 08.02.2016].

Wikipedia, 2017d, Microsoft Visual Studio, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Microsoft_Visual_Studio: [Erişim: 09.02.2016].

Wikipedia, 2017e, Object-Oriented Programming, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Object-oriented_programming: [Erişim: 06.02.2016].

Wikipedia, 2017f, Procedural Programming, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Procedural_programming: [Erişim: 06.02.2016].

Wikipedia, 2017g, Rastrigin Function, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Rastrigin_function: [Erişim: 12.02.2016].

Wikipedia, 2017h, Rosenbrock Function, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Rosenbrock_function: [Erişim: 12.02.2016].

Wikipedia, 2017i, Test Functions for Optimization, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Test_functions_for_optimization: [Erişim: 12.02.2016].

Wilkes, M., 1953, Can machines think? Proceedings of the IRE. 41**:** 1230-1234.

Wolpert, D. H. ve Macready, W. G., 1995, No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, *Santa Fe Institute*.

Woolson, R., 2008, Wilcoxon Signed-Rank Test, *Wiley encyclopedia of clinical trials*.

Wu, B. ve Fan, S.-h., 2011, Improved artificial bee colony algorithm with chaos, *Computer Science for Environmental Engineering and EcoInformatics*, 51-56.

Wu, C., Zhang, N., Jiang, J., Yang, J. ve Liang, Y., 2007, Improved bacterial foraging algorithms and their applications to job shop scheduling problems, *International Conference on Adaptive and Natural Computing Algorithms*, 562-569.

Xiao, J.-k., Li, W.-m., Xiao, X.-r. ve Lv, C.-z., 2016, Improved Clonal Selection Algorithm Optimizing Neural Network for Solving Terminal Anti-missile

Collaborative Intercepting Assistant Decision-Making Model, *Asian Simulation Conference*, 216-231.

Xing, B. ve Gao, W.-J., 2014, Imperialist competitive algorithm, In: Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms, Eds: Springer, p. 203-209.

Xu, F., Chen, W. ve Yang, L., 2007, Improved particle swarm optimization for realistic portfolio selection, *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, 185-190.

Xue, F., Sanderson, A. C. ve Graves, R. J., 2003, Pareto-based multi-objective differential evolution. Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, IEEE. 2**:** 862-869.

Yager, R. R. ve Zadeh, L. A., 2012, An Introduction to Fuzzy Logic Applications in Intelligent Systems, Springer Science & Business Media.

Yang, C., Ji, J., Liu, J. ve Yin, B., 2016, Bacterial foraging optimization using novel chemotaxis and conjugation strategies, *Information Sciences*, 363, 72-95.

Yang, X.-S., 2009, Firefly algorithms for multimodal optimization, *International Symposium on Stochastic Algorithms*, 169-178.

Yang, X.-S. ve Deb, S., 2009, Cuckoo search via Lévy flights, *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 210-214.

Yang, X.-S. ve Deb, S., 2010, Engineering optimisation by cuckoo search, *International Journal of Mathematical Modelling and Numerical Optimisation*, 1 (4), 330-343.

Yang, X.-S., 2010a, A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010)**:** 65-74.

Yang, X.-S., 2010b, Nature-Inspired Metaheuristic Algorithms, Luniver Press.

Yang, X.-S., 2012, Flower pollination algorithm for global optimization, *International Conference on Unconventional Computing and Natural Computation*, 240-249.

Yang, X.-S., Cui, Z., Xiao, R., Gandomi, A. H. ve Karamanoglu, M., 2013, Swarm intelligence and bio-inspired computation: theory and applications, Newnes.

Yang, X.-S. ve Deb, S., 2013, Multiobjective cuckoo search for design optimization, *Computers and Operations Research*, 40 (6), 1616-1624.

Yang, X.-S. ve Deb, S., 2014, Cuckoo search: Recent advances and applications, *Neural Computing and Applications*, 24 (1), 169-174.

Yang, X.-S., Karamanoglu, M. ve He, X., 2014, Flower pollination algorithm: A novel approach for multiobjective optimization, *Engineering optimization*, 46 (9), 1222-1237.

Yegnanarayana, B., 2009, Artificial neural networks, PHI Learning Pvt. Ltd..

Yıldız, A. R., 2013, A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing, *Applied Soft Computing*, 13 (5), 2906-2912.

Yin, C., Ma, L. ve Feng, L., 2016, A feature selection method for improved clonal algorithm towards intrusion detection, *International Journal of Pattern Recognition and Artificial Intelligence*, 30 (05), 1659013.

Yousefikhoshbakht, M. ve Sedighpour, M., 2013, New imperialist competitive algorithm to solve the travelling salesman problem, *International Journal of Computer Mathematics*, 90 (7), 1495-1505.

Yun, Z., Quan, Z., Caixin, S., Shaolan, L., Yuming, L. ve Yang, S., 2008, RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment, *IEEE transactions on power systems*, 23 (3), 853-858.

Zadeh, L. A., 1965, Fuzzy sets, *Information and control*, 8 (3), 338-353.

Zadeh, L. A., 1978, Fuzzy sets as a basis for a theory of possibility, *Fuzzy sets and systems*, 1 (1), 3-28.

Zadeh, L. A., 1994, Soft computing and fuzzy logic, *IEEE software*, 11 (6), 48-56.

Zelinka, I., Snasel, V. ve Abraham, A., 2012, Handbook of optimization: from classical to modern approach, Springer Science & Business Media.

Zhang, J. ve Sanderson, A. C., 2009, JADE: Adaptive differential evolution with optional external archive, *IEEE transactions on evolutionary computation*, 13 (5), 945-958.

Zhang, M., Dai, J., Zheng, J. ve Zhang, G., 2016, An Improved Flower Pollination Algorithm, *Web Information Systems and Applications Conference, 2016 13th*, 179-183.

Zhang, X.-S., 2013b, Neural networks in optimization, Springer Science & Business Media.

Zhang, Y., Wang, Y. ve Peng, C., 2009, Improved imperialist competitive algorithm for constrained optimization, *Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on*, 204-207.

Zhang, Y., Wu, L. ve Wang, S., 2011, Magnetic resonance brain image classification by an improved artificial bee colony algorithm, *Progress In Electromagnetics Research*, 116, 65-79.

Zhang, Y. ve Ni, Q., 2017, A coordinated traffic control on urban expressways with modified particle swarm optimization, *KSCE Journal of Civil Engineering*, 21 (2), 501-511.

Zhongzhi, S., 2001, A clustering algorithm based on swarm intelligence, *Info-tech and Info-net, 2001. Proceedings. ICII 2001-Beijing. 2001 International Conferences on*, 58-66.

Zhou, M., 1998, Fuzzy logic and optimization models for implementing QFD, *Computers and Industrial Engineering*, 35 (1-2), 237-240.

Zhou, Y. ve Wang, R., 2016, An improved flower pollination algorithm for optimal unmanned undersea vehicle path planning problem, *International Journal of Pattern Recognition and Artificial Intelligence*, 30 (04), 1659010.

Zimmermann, H.-J., 1978, Fuzzy programming and linear programming with several objective functions, *Fuzzy sets and systems*, 1 (1), 45-55.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Utku KÖSE
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Afyonkarahisar / 26.03.1985
Telefon	**:**	0532 590 83 26
Faks	**:**	-
E-Posta	**:**	utkukose@gmail.com
Kişisel Web Sitesi	**:**	http://www.utkukose.com

EĞİTİM

Derece	**Adı, İl**	**Bitirme Yılı**
Yüksek Lisans :	Afyon Kocatepe Üniversitesi, Afyonkarahisar	2010
Lisans :	Gazi Üniversitesi, Ankara	2008
Lise :	Gazi Anadolu Meslek Lisesi, Afyonkarahisar	2003

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2012 - …	Uşak Üniversitesi	Öğretim Görevlisi
2011 - 2012	Afyon Kocatepe Üniversitesi	Öğretim Görevlisi
2009 - 2011	Afyon Kocatepe Üniversitesi	Araştırma Görevlisi

UZMANLIK ALANI

Yapay Zekâ, Optimizasyon, Bilgisayar Bilimleri, Kaos, Uzaktan Eğitim, E-Öğrenme, Bilgisayar Eğitimi, Bilgisayar Yazılımları

YABANCI DİLLER

İngilizce, Almanca, İtalyanca

YAYINLAR

Doktora Tez Çalışmasından Yapılan Yayınlar:

SCI İndeksli Makale:

Köse, U. ve Arslan, A., 2017, Time series prediction with a hybrid system formed by artificial neural network and cognitive development optimization algorithm, *Scientia Iranica*, (Basımda), 2017.

Köse, U. ve Arslan, A., 2017, Optimization of self-learning in computer engineering courses: An intelligent software system supported by artificial neural network and

vortex optimization algorithm, *Computer Applications in Engineering Education*, 25(1), 142-156, 2017.

Köse, U. ve Arslan, A., 2016, Forecasting chaotic time series via ANFIS supported by vortex optimization algorithm: Applications on electroencephalogram time series, Arabian Journal for Science and Engineering, (Online First View, DOI:10.1007/s13369-016-2279-z).

Köse, U. ve Arslan, A., 2016, Intelligent e-learning system for improving students' academic achievements in computer programming courses, *International Journal of Engineering Education*, 32(1-A), 185-198.

SCI Dışı İndeksli Makale:

Köse, U. ve Arslan, A., 2017, Global Optimization Algorithms with In-System Optimization Approach, *International Journal of Swarm Intelligence Research*, (Basımda), 2017.

Boz, H. ve Köse, U., 2017, Emotion Extraction from Facial Expressions by Using Artificial Intelligence Techniques, *Broad Research in Artificial Intelligence and Neuroscience*, (Basımda), 2017.

Demir, A. ve Köse, U., 2016, Solving optimization problems via vortex optimization algorithm and cognitive development optimization algorithm, *Broad Research in Artificial Intelligence and Neuroscience*, 7(4), 23-42.

Köse, U., Güraksın, G. E. ve Deperlioğlu, Ö., 2016, Cognitive development optimization algorithm based support vector machines for determining diabetes, *Broad Research in Artificial Intelligence and Neuroscience*, 7(1), 80-90.

Köse, U. ve Arslan, A., 2015, Realizing an optimization approach inspired from Piaget's theory on cognitive development, *Broad Research in Artificial Intelligence and Neuroscience*, 6(1-4), 15-22.

Köse, U. ve Arslan, A., 2015, E-learning experience with artificial intelligence supported software: An international application on English language courses, *GLOKALde*, 1(3), 61 – 75.

Köse, U. ve Arslan, A., 2014, On the idea of a new artificial intelligence based optimization algorithm inspired from the nature of vortex, *Broad Research in Artificial Intelligence and Neuroscience*, 5(1-4), 60 – 66.

Uluslararası Tebliğ:

Köse, U. ve Gezer, B., 2017, Optimum Conditions for Obtaining Boric Acid from Tincal with UAE and Swarm Intelligence Based Optimization, *International Scientific and Vocational Studies Congress 2017 (BILMES 2017),* Türkiye.

Gezer, B. ve Köse, U., 2017, Optimization of Design Parameters for Producing Boric Acid from Colemanite with UAE and Artificial Intelligence, *International Scientific and Vocational Studies Congress 2017 (BILMES 2017),* Türkiye.

Köse, U., Uğur, S. ve Vasant, P., 2017, Design and Development of a Diagnosis System with a SVM-VOA Hybrid Model Approach, *World Conference on Computer Engineering 2017 (COMENG 2017),* Türkiye.

Köse, U., Arslan, A. ve Vasant, P., 2017, Solving Real-World Problems via Novel Intelligent Algorithms, *EAI International Conference on Computer Science and Engineering 2017 (COMPSE 2017),* Tayland.

Köse, U., Güraksın, G. E. ve Deperlioğlu, Ö., 2016, Improving Underwater Image Quality via Vortex Optimization Algorithm, *International Multidisciplinary Conference 2016 (IMUCO 2016)*, Antalya, Turkey, 327-333.

Yüksek Lisans Tez Çalışmasından Yapılan Yayınlar:

SCI İndeksli Makale:

Deperlioğlu, Ö. ve Köse, U., 2011, An educational tool for artificial neural networks, *Computers and Electrical Engineering*, 37(3), 392-402.

SCI Dışı İndeksli Makale:

Köse, U. ve Deperlioğlu, Ö., 2015, FL-LAB v2: Design and development of an easy-to-use, interactive fuzzy logic control software system, *Applied Mathematics & Information Sciences*, 9(2), 883-897.

Uluslararası Tebliğ:

Köse, U. ve Deperlioğlu, Ö., 2010, An Educational, Virtual Laboratory System for Fuzzy Logic, *International Symposium on Computing in Science and Engineering 2010 (ISCSE 2010)*, Turkey, 1335 – 1342, 2010.