**T.C.**
**ERCİYES ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

# SÜRÜ TABANLI KARINCA ASLANI VE BALİNA OPTİMİZASYONU ALGORİTMALARININ FİZİK TABANLI ALGORİTMALARLA HİBRİTLEŞTİRİLMESİ

**(Yüksek lisans Tezi)**

**Hazırlayan**
**Bahadur ALIZADA**

**Danışman**
**Dr. Öğr. Üyesi Mustafa DANACI**

**Temmuz 2019**
**KAYSERİ**

# BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Bahadur ALIZADA

İmza : 

# YÖNERGEYE UYGUNLUK

Sürü Tabanlı Karınca Aslanı ve Balina Optimizasyonu Algoritmalarının Fizik Tabanlı Algoritmalarla Hibritleştirilmesi adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi 'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan

Bahadur ALIZADA

Danışman

Dr. Öğr. Üyesi Mustafa DANACI

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Veysel ASLANTAŞ

Dr. Öğr. Üyesi. Mustafa DANACI danışmanlığında **Bahadur ALIZADA** tarafından hazırlanan "Sürü Tabanlı **Karınca Aslanı ve Balina Optimizasyonu Algoritmalarının Fizik Tabanlı Algoritmalarla Hibritleştirilmesi**" adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında **Yüksek lisans** tezi olarak kabul edilmiştir.

03 /07 / 2019

(Tez savunma sınav tarihi yazılacaktır.)

**JÜRİ:**

| | | |
|---|---|---|
| Danışman | : Dr. Öğr. Üyesi. Mustafa DANACI | ..... |
| Üye | : Doç. Dr. Orhan ER | ..... |
| Üye | : Dr. Öğr. Üyesi Fehim KÖYLÜ | ..... |

ONAY:
Bu tezin kabulü Enstitü Yönetim Kurulunun 04/07/2019 tarih ve 2019/38-03 sayılı kararı ile onaylanmıştır.

T.C.

04 / 07 / 2019

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

# TEŞEKKÜRLER

Çalışmalarım boyunca farklı bakış açıları ve bilimsel katkılarıyla beni yönlendiren, yakın ilgi ve desteği olan danışman hocam Dr. Öğr. Üyesi Mustafa DANACI 'ya teşekkürü bir borç bilirim.

Yüksek lisans eğitimimi maddi açıdan destekleyen Türkiye Bursları programına teşekkür ederim.

Ayrıca çalışmalarım süresince sabır göstererek beni daima destekleyen aileme, yurt arkadaşlarıma, sınıf arkadaşlarıma en içten teşekkürlerimi sunarım.

Bahadur ALIZADA

Kayseri, Temmuz 2019

# SÜRÜ TABANLI KARINCA ASLANI VE BALİNA OPTİMİZASYONU ALGORİTMALARININ FİZİK TABANLI ALGORİTMALARLA HİBRİTLEŞTİRİLMESİ

**Bahadur ALIZADA**
**Erciyes Üniversitesi, Fen Bilimleri Enstitüsü**
**Yüksek Lisans Tezi, Temmuz 2019**
**Danışman: Dr. Öğr. Üyesi. Mustafa DANACI**

## ÖZET

Mühendislik problemlerinin çözümünün zor olması kullanılacak yöntemlerin seçiminde zorluklar oluşturmaktadır. Doğadan esinlenen sürü zekâsı tabanlı meta-sezgisel optimizasyon teknikleri bu tür problemlerin çözümünde son zamanlarda en çok tercih edilen popüler algoritmalardır.

Bu tez çalışmasında çeşitli problemlere uyarlanmak üzere 2 adet yeni hibrit algoritma modeli geliştirilmiştir. Geliştirilen modeller literatürde yer alan 23 adet Benchmark test problemline uyarlanarak meta-sezgisel algoritmalarla kıyaslanıp başarılı olduğu gözlenmiştir.

Algoritmaların amacı optimizasyon süreçleri olan keşif ve sömürü arasındaki dengeyi sağlamaktır. Meta-sezgisel algoritma geliştirmede stokastik yapısından dolayı dengenin sağlanması oldukça zordur. Bu çalışmada Sinüs Kosinüs Algoritması (SKA) kullanılarak geliştirilen Karınca Aslanının hibrit modeli HSKKAO, her iki algoritmanın eksik yönlerini tamamlayarak iyileştirip oldukça başarılı sonuçlar alınmasını sağlamıştır. Aynı şekilde literatürde mevcut olan Sinüs Kosinüs Balina Optimizasyon Algoritması (SKBOA) hibrit modeli üzerinde Çoklu Evren Optimizasyonu (ÇEO) ile iyileştirilen yeni hibrit modeli test problemlerindeki başarıyı daha da arttırmıştır.

Literatürde yer alan Genetik Algoritma (GA), Parçacık Sürü Optimizasyonu (PSO), Yarasa Algoritması (YA), Çiçek Tozlama Algoritması (ÇTA), Guguk Arama (GuA), Ateş Böceği Algoritması (ABA), Madde Arama Durumları (MAD), Yerçekimsel Arama Algoritması (YAA) ve Hızlı Evrimsel Programlama (HEP) algoritmalarına ait Benchmark test sonuçları ile hibrit modellerin sonuçları karşılaştırılmış ve oldukça başarılı ve rekabetçi sonuçlar elde edilmiştir.

**Anahtar Kelimeler:** Benchmark, meta-sezgisel algoritmalar, Sinüs Kosinüs Algoritması, Karınca Aslanı Optimizasyonu, Balina Optimizasyonu Algoritması, Çoklu Evren Optimizasyonu

# HYBRIDIZATION OF SWARM-BASED ANT LION AND WHALE OPTIMIZATION ALGORITHMS WITH PHYSICS-BASED ALGORITHMS

**Bahadur ALIZADA**
**Erciyes University, Graduate School of Natural and Applied Sciences**
**M.Sc. Thesis, July, 2019**
**Supervisor: Asst. Prof. Mustafa DANACI**

## ABSTRACT

The difficulty in solving engineering problems creates difficulties in the selection of the methods to be used. Nature-inspired herd intelligence-based meta-heuristic optimization techniques have recently become the most popular algorithms for solving such problems.

In this thesis, two new hybrid algorithm models have been developed to adapt to various problems. The developed models were adapted to 23 Benchmark test problems in the literature and compared with meta-heuristic algorithms.

The aim of the algorithms is to balance the optimization processes of exploration and exploitation. In the development of a meta-heuristic algorithm, it is very difficult to achieve a balance due to its stochastic structure. In this study, HSKKAO hybrid model of Ant Lion developed by using Sine Cosine Algorithm (SCA) has improved the missing aspects of both algorithms and achieved very successful results. Similarly, the new hybrid model improved by Multi-Verse Optimization (MVO) on the Sine Cosine Whale Optimization Algorithm (SCBOA) hybrid model, which is available in the literature, has increased the success of test problems.

In the literature Benchmark test results of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Bat Algorithm (BA), Flower Pollination Algorithm (FPA), Cuckoo Search (CS), Firefly Algorithm (FA), States of Matter Search (SMS), Gravity Search Algorithm (GSA) and Fast Evolutionary Programming (FEP) were compared with the results of the hybrid models, and very successful and competitive results were obtained.

**Keyword**: Benchmark, meta-heuristic algorithms, Sinus Cosine Algorithm, The Ant Lion Optimization, Whale Optimization Algorithm, Multi-Verse Optimization

**İÇİNDEKİLER**

**SÜRÜ TABANLI KARINCA ASLANI VE BALİNA OPTİMİZASYONU ALGORİTMALARININ FİZİK TABANLI ALGORİTMALARLA HİBRİTLEŞTİRİLMESİ**

Sayfa

BİLİMSEL ETİĞE UYGUNLUK SAYFASI ..... ii
YÖNERGEYE UYGUNLUK SAYFASI ..... iii
KABUL VE ONAY SAYFASI ..... iv
TEŞEKKÜR ..... v
ÖZET ..... vi
ABSTRACT ..... vii
İÇİNDEKİLER ..... viii
KISALTMALAR VE SİMGELER ..... xii
TABLOLAR LİSTESİ ..... xvi
ŞEKİLLER LİSTESİ ..... xvii

**GİRİŞ ..... 1**

**1. BÖLÜM**

**GENEL BİLGİLER**

**1.1. Benchmark test problemleri ..... 3**
**1.2. Benchmark fonksiyonları çeşitleri ..... 3**
**1.2.1. Tek modlu Benchmark fonksiyonları ..... 3**
**1.2.2. Çok modlu Benchmark fonksiyonları ..... 4**
**1.2.3. Sabit boyutlu çok modlu Benchmark fonksiyonları ..... 6**
**1.3. Benchmark Fonksiyonları Literatür Taraması ..... 7**
**1.3.1. Fonksiyon özellikleri ..... 7**
**1.3.1.1. Modallık ..... 7**
**1.3.1.2. Vadi ..... 7**
**1.3.1.3. Ayrıştırılabilirlik ..... 7**
**1.3.1.4. Boyutsallık ..... 8**

**1.4. Fonksiyon bilgileri ..... 8**
**1.4.1. F1-Sphere fonksiyonu ..... 8**
**1.4.2. F2-Schwefel2.22 fonksiyonu ..... 9**
**1.4.3. F3-Schwefel1.2 fonksiyonu ..... 9**
**1.4.4. F4-Schwefel2.21 fonksiyonu ..... 10**
**1.4.5. F5-Rosenbrock fonksiyonu ..... 10**
**1.4.6. F6-Step2 fonksiyonu ..... 11**
**1.4.7. F7-Quartic fonksiyonu ..... 11**
**1.4.8. F8-Generalized Schwefel2.26 fonksiyonu ..... 12**
**1.4.9. F9-Rastrigin fonksiyonu ..... 12**
**1.4.10. F10-Ackley fonksiyonu ..... 13**
**1.4.11. F11-Griewank fonksiyonu ..... 13**
**1.4.12. F12-Penalized fonksiyonu ..... 14**
**1.4.13. F13-Penalized2 fonksiyonu ..... 14**
**1.4.14. F14-Shekel`s Foxholes fonksiyonu ..... 15**
**1.4.15 F15-Kowalik fonksiyonu ..... 15**
**1.4.16. F16-Six-Hump Camel-Back fonksiyonu ..... 16**
**1.4.17. F17-Branin RCOS fonksiyonu ..... 16**
**1.4.18. F18-GoldStein-Price fonksiyonu ..... 17**
**1.4.19. F19-Hartman3 fonksiyonu ..... 17**
**1.4.20. F20-Hartman6 fonksiyonu ..... 18**
**1.4.21. F21-Shekel5 fonksiyonu ..... 18**
**1.4.22. F22-Shekel7 fonksiyonu ..... 19**
**1.4.23. F23-Shekel10 fonksiyonu ..... 19**

**2. BÖLÜM**

**META-SEZGİSEL OPTİMİZASYON TEKNİKLERİ**

**2.1. Giriş ..... 21**
**2.2. Meta-sezgisel algoritmalar ..... 23**
**2.2.1. Evrimsel Algoritmalar ..... 24**
**2.2.2 Fizik Tabanlı Algoritmalar ..... 25**
**2.2.3 Sürü Tabanlı Algoritmalar ..... 26**
**2.2.4. İnsan Tabanlı Algoritmalar ..... 27**

**2.3. No Free Lunch (NFL) teoremi ..... 28**
**2.4. Tezde Kıyaslanan Algoritmalar ..... 29**
**2.4.1. Genetik Algoritma (GA) ..... 29**
**2.4.2. Parçacık Sürü Optimizasyonu (PSO) ..... 29**
**2.4.3. Yarasa Algoritması (YA) ..... 30**
**2.4.4. Çiçek Tozlama Algoritma (ÇTA) ..... 30**
**2.4.5. Guguk Arama (GuA) ..... 30**
**2.4.6. Ateş Böceği Algoritması (ABA) ..... 31**
**2.4.7. Madde Arama Durumları (MAD) ..... 31**
**2.4.8. Yerçekimsel Arama Algoritması (YAA) ..... 31**
**2.4.9. Hızlı Evrimsel Programlama (HEP) ..... 32**

**3. BÖLÜM**
**HİBRİTLEŞTİRMEDE KULLANILAN OPTİMİZASYON ALGORİTMALARI**

**3.1 Giriş ..... 33**
**3.2. Sinüs Kosinüs Algoritması (SKA) ..... 33**
**3.2.1 Sinüs Kosinüs Algoritmasının kaba kodu. ..... 35**
**3.3. Karınca Aslanı Optimizasyonu (KAO) ..... 36**
**3.3.1 Esinlenme. ..... 36**
**3.3.2. Karınca Aslanı Optimizasyonu algoritmasının operatörleri ..... 37**
**3.3.2.1. Karıncaların rastgele yürüşü ..... 39**
**3.3.2.2. Karınca aslanlarının çukurlarında tuzağa düşürme ..... 40**
**3.3.2.3. Tuzağın inşası ..... 40**
**3.3.2.4. Karınca aslanlarına doğru kayan karıncalar ..... 40**
**3.3.2.5. Avları yakalamak ve çukuru yeniden inşa etmek ..... 41**
**3.3.2.6. Elitizm ..... 41**
**3.3.3. Karınca Aslanı Optimizasyonu algoritmasının kaba kodu ..... 42**
**3.4. Balina Optimizasyon Algoritması (BOA) ..... 43**
**3.4.1. Esinlenme. ..... 43**
**3.4.2. Matematiksel model. ..... 43**
**3.4.2.1. Av kuşatma ..... 43**
**3.4.2.2. Kabarcıklı saldırı (sömürü aşaması). ..... 44**
**3.4.2.2.1. Daralan çevreleme mekanizması. ..... 44**

**3.4.2.2.1. Konumun spiral güncellenmesi..... 45**
**3.4.2.3. Av arama (çözüm alanını keşif etmek)..... 45**
**3.4.3. Balina Optimizasyonu Algoritmasının kaba kodu. ..... 46**
**3.5. Çoklu Evren Algoritması (ÇEA) ..... 47**
**3.5.1. Esinlenme. ..... 47**
**3.5.2.Matematiksel model. ..... 48**
**3.5.3. Çoklu Evren Optimizasyonu algoritmasının kaba kodu. ..... 49**

**4. BÖLÜM**
**HİBRİT UYGULAMA GELİŞTİRME 1**

**4.1. Sinüs Kosinüs ile Karınca Aslanı Optimizasyonu (HSKKAO) ..... 50**
**4.2. Çalışmanın motivasyonu ..... 50**
**4.2. Matematiksel model ..... 51**
**4.2. Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyon algoritmasının kaba kodu ... 52**
**4.2. Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyonu Performans Analizi ..... 53**

**5. BÖLÜM**
**HİBRİT UYGULAMA GELİŞTİRME 2**

**5.1. Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu (ÇESKBO) hibrit algoritması model..... 56**
**5.2. Çalışmanın motivasyonu ..... 56**
**5.3. Matematiksel model ..... 56**
**5.4. Çok Evrenli Sinüs Kosinüs Balina Optimizasyon algoritmasının kaba kodu..... 57**
**5.5. Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu Performans Analizi..... 58**

**6. BÖLÜM**
**SONUÇ, TARTIŞMA ve ÖNERİLER**

**6.1. Sonuç ..... 61**
**6.2 Tartışma ve Öneriler..... 62**
**KAYNAKLAR..... 63**
**ÖZGEÇMİŞ..... 70**

# KISALTMALAR VE SİMGELER

| | |
|---|---|
| GA | Genetik Algoritma |
| GP | Genetik Programlama |
| ES | Evrimsel Strateji |
| BTO | Biyocoğrafya Tabanlı Optimize |
| EP | Evrimsel Programlama |
| DG | Diferansiyel Gelişim |
| GAP | Gen Anlatım Programlama |
| BT | Benzetimli Tavlama |
| YYA | Yerçekimsel Yerel Arama |
| BPBÇ | Büyük Patlama Büyük Çöküş |
| YAA | Yerçekimsel Arama Algoritması |
| YSA | Yüklü Sistem Araması |
| MKO | Merkezi Kuvvet Optimizasyonu |
| YKROA | Yapay Kimyasal Reaksiyon Optimizasyon Algoritması |
| KDA | Kara Delik Algoritması |
| IO | Işın Optimizasyonu |
| KDOA | Küçük Dünya Optimizasyon Algoritması |
| GTAA | Galaksi Tabanlı Arama Algoritması |
| KUO | Kavisli Uzay Optimizasyonu |
| PSO | Parçacık Sürü Optimizasyonu |

| | |
|---|---|
| KKO | Karınca Kolonisi Optimizasyonu |
| BAEOA | Bal Arıları Evlilik Optimizasyon Algoritmaları |
| YBSA | Yapay Balık Sürüsü Algoritması |
| TA | Termit Algoritması |
| YAK | Yapay Arı Kolonisi |
| YASA | Yaban Arısı Sürüsü Algoritması |
| MA | Maymun Arama |
| KSAA | Kurt Sürüsü Arama Algoritması |
| APTA | Arı Polen Toplama Algoritması |
| GuA | Guguk Arama |
| YPA | Yunus Partner Algoritması |
| YeA | Yarasa-esinlenme Algoritması |
| ABA | Ateş Böceği Algoritması |
| AvA | Av Arama |
| KÇO | Kuş Çiftleşme Optimizasyonu |
| KS | Kril Sürüsü |
| MSOA | Meyve Sineği Optimizasyon Algoritması |
| YE | Yunus Ekolokasyonu |
| ÖÖBO | Öğretme Öğrenme Bazlı Optimizasyon |
| AhA | Ahenk Arama |
| TA | Tabu Arama |
| GAO | Grup Arama Optimizasyonu |

| | |
|---|---|
| ERA | Emperyalist Rekabetçi Algoritma |
| LŞA | Lig Şampiyonluğu Algoritması |
| HFA | Havai Fişek Algoritması |
| ÇCO | Çarpışan Cisimler Optimizasyonu |
| İAA | İç Arama Algoritması |
| MPA | Mayın Patlaması Algoritması |
| FLYA | Futbol Ligi Yarışması Algoritması |
| AOA | Arayıcı Optimizasyon Algoritması |
| STA | Sosyal Tabanlı Algoritma |
| DPA | Döviz Piyasası Algoritması |
| GDO | Grup Danışmanlık Optimizasyonu |
| NFL | No Free Lunch teoremi |
| YA | Yarasa Algoritması |
| ÇTA | Çiçek Tozlama Algoritması |
| MAD | Madde Arama Durumları |
| HEP | Hızlı Evrimsel Programlama |
| CEP | Cauchy Evrimsel Programlama |
| SKA | Sinüs Kosinüs Algoritması |
| KAO | Karınca Aslanı Optimizasyonu |
| BOA | Balina Optimizasyonu Algoritması |
| ÇEO | Çoklu Evren Optimizasyonu |
| HSKKAO | Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyonu |

| | |
|---|---|
| ÇESKBO | Çoklu Evren Sinüs Kosinüs Balina Optimizasyonu |
| SKBOA | Sinüs Kosinüs Balina Optimizasyonu Algoritması |
| M | mod |
| B | boyut |
| $f_{min}$ | minimum değer |
| + | var |
| - | yok |
| x | bilgi yok |
| WEP | solucan deliği varlık olasılığı |
| TDR | seyahat mesafesi oranı |
| F | fonksiyonlar |
| ort | aritmetik ortalama |
| sts | standart sapma |
| GSP | Gezgin Satıcı Problemi |

# TABLOLAR LİSTESİ

Tablo 1.1. Tek modlu Benchmark fonksiyonları……..……4

Tablo 1.2. Çok modlu Benchmark fonksiyonları……..……..…5

Tablo 1.3. Sabit boyutlu çok modlu Benchmark fonksiyonları…….6

Tablo 1.4. Fonksiyonların çeşitli özellikleri……...…….20

Tablo 2.1. Literatürdeki bir kısım evrimsel algoritmalar……25

Tablo 2.2. Literatürdeki popüler fizik tabanlı algoritmalar…….25

Tablo 2.3. Literatürdeki geliştirilen sürü tabanlı optimizasyon algoritmaları…….27

Tablo 2.4. Literatürdeki popüler insan tabanlı algoritmalar……28

Tablo 4.1. HSKKAO algoritmasının tek modlu ve çok modlu Benchmark fonksiyon sonuçları ile kıyaslama...……54

Tablo 4.2. HSKKAO algoritmasının sabit boyutlu çok modlu Benchmark fonksiyon sonuçları ile kıyaslama……...……..…55

Tablo 5.1. ÇESKBO algoritmasının tek modlu ve çok modlu Benchmark fonksiyon sonuçları ile kıyaslama...……59

Tablo 5.2. ÇESKBO algoritmasının sabit boyutlu çok modlu Benchmark fonksiyon sonuçları ile kıyaslama……60

# ŞEKİLLER LİSTESİ

Şekil.1.1. F1-Sphere fonksiyonu grafiği ve denklemi……..…….8

Şekil.1.2. F2-Schwefel2.22 fonksiyonu grafiği ve denklemi ……...9

Şekil.1.3. F3-Schwefel1.2 fonksiyonu grafiği ve denklemi …….……9

Şekil.1.4. F4-Schwefel2.21 fonksiyonu grafiği ve denklemi …….……10

Şekil.1.5. F5-Rosenbrock fonksiyonu grafiği ve denklemi ……...10

Şekil.1.6. F6-Step2 fonksiyonu grafiği ve denklemi ……..11

Şekil.1.7. F7-Quartic fonksiyonu grafiği ve denklemi ……...11

Şekil.1.8. F8-Generalized Schwefel2.26 fonksiyonu grafiği ve denklemi …….12

Şekil.1.9. F9-Rastrigin fonksiyonu grafiği ve denklemi ……12

Şekil.1.10. F10-Ackley fonksiyonu grafiği ve denklemi ……13

Şekil.1.11. F11-Generalized Schwefel2.26 fonksiyonu grafiği ve denklemi ...……..13

Şekil.1.12. F12-Penalized fonksiyonu grafiği ve denklemi ……....14

Şekil.1.13. F13-Penalized2 fonksiyonu grafiği ve denklemi ...……...14

Şekil.1.14. F14-Shekel`s Foxholes fonksiyonu grafiği ve denklemi ……..15

Şekil.1.15. F15-Kowalik fonksiyonu grafiği ve denklemi …….15

Şekil.1.16. F16-Six-Hump Camel-Back fonksiyonu grafiği ve denklemi …….16

Şekil.1.17. F17-Branin RCOS fonksiyonu grafiği ve denklemi ……..……..16

Şekil.1.18. F18-GoldStein Price fonksiyonu grafiği ve denklemi ……..17

Şekil.1.19. F19Hartman3 fonksiyonu grafiği ve denklemi ……17

Şekil.1.20. F20-Hartman6 fonksiyonu grafiği ve denklemi ...……18

Şekil.1.21. F21-Shekel5 fonksiyonu grafiği ve denklemi ...……...18

Şekil.1.22. F22-Shekel7 fonksiyonu grafiği ve denklemi …...……...19

Şekil.1.23. F23-Shekel10 fonksiyonu grafiği ve denklemi ...…….19

Şekil.2.1. Meta-Sezgisel Algoritmaların Sınıflandırılmasın……...23

Şekil.3.1.Şimdiye kadar elde edilen en iyi çözüme doğru güncelleme veya uzaklaşma.34

Şekil.3.2. İterasyonlar (yinelemeler) boyunca sinüs ve kosinüs aralığını azaltmak……35

Şekil.3.3. Koni şeklindeki tuzaklar ve karınca aslanlarının avlanma davranışları……..37

Şekil.3.4. Daralan çevreleme mekanizması…….44

Şekil.3.5. Konumun spiral güncellenmesi……...45

Şekil.3.6. Beyaz delik, kara delik ve solucan deliği……47

# GİRİŞ

Optimizasyon algoritmaları hayatımızın hemen hemen her alanında çeşitli problemleri çözmek, minimize veya maksimize etmek için kullanılmak üzere vardır.

**Tezin Konusu**

Benchmark fonksiyonları çeşitli özelliklerle yeni oluşturulan optimizasyon algoritmalarının denenmesinde ve onaylanmasında işe yarar.

Optimizasyon problemleri çeşitli yöntemlerle çözülebilir. Ama hesaplama zahmeti, fazladan zaman kaybı, maliyet ve diğer etkenlerden dolayı bazı çözümleri kullanılmaz. Optimizasyon algoritmaları ise bunların çoğunda sağladığı avantajlardan dolayı çok tercih edilmektedir.

Optimizasyon algortimalarının verimliliğini arttırmak için hibritleştirme, iyileştirme ve modifikasyon gibi çeşitli algoritmalar geliştirilir. İki veya daha fazla algortimanın farklı ve avantajlı özellikleri birleştirilerek bir optimizasyon algortimasında toplandığında genelde iyi sonuçlar elde edilimektedir.

**Tezin Amacı**

Tezin amacı yeni hibrit algortimalar oluşturularak orjinal algoritmalara göre verimliliğin hangi yönde olduğunun tespit edilmesi, bilim camiasında farklı problemlere uyarlanmak üzere literatüre hibritlerin kazandırılması, yeni hibrit, iyileştirme modellerinin oluşturulmasında bilim insanlarının motive edilmesidir.

Hibritlerin test edilerek onaylanması ve literatüre kazandırılması sayesinde çözümü zor gerçek hayat problemlerinde uyarlanabilecek tekniklere yenilerinin eklenmesini sağlamak beklenilen amaçlardandır.

## Tezin İçeriği

Bölüm 1`de, optimizasyon tekniklerinin test edileceği Benchmark fonksiyonlarının özellikleri, parametreleri, grafikleri detaylı şekilde verilmiştir.

Bölüm 2`de, meta-sezgisel optimizasyon algortimalarının türleri, özellikleri ve kıyaslanacak algoritmalar hakkında kısa bilgiler verilmiştir.

Bölüm 3`te, hibritleştirmede kullanılan algoritmaların özellikleri ve matematiksel modelleri hakkında bilgi verilmiştir.

Bölüm 4`te, Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyonu modelinin motivasyon kaynağı, matematiksel modeli, kaba kodu ve de performans analizi bulunmaktadır.

Bölüm 5`te, Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu modelinin motivasyon kaynağı, matematiksel modeli, kaba kodu ve de performans analizi bulunmaktadır.

Bölüm 6`da, tartışma, sonuç ve öneriler verilmiştir.

# 1. BÖLÜM

# GENEL BİLGİLER

## 1.1. Benchmark test problemleri

Benchmark fonksiyonları herhangi bir optimizasyon yaklaşımını test etmek amaçlı kullanılan fonksiyonlardır. Bu toplu fonsiyonlar ayrıştırılabilir, türevlenebilir, sürekli, devamsız, ölçeklenebilir, tek modlu ve çok modlu gibi özelliklere sahip olduğu için algoritmanın hangi tür problemlerde verimli olabileceğine ilişkin bilgileri bu fonksiyonlara uygulanarak elde edilebilir. Benchmark test problemlerine çeşitli sitelerden ve kaynaklardan ulaşılabilinir.

Tez çalışmasında geliştirilen hibrit algoritmalar farklı karakteristik özelliklere sahip üç grup Benchmark fonksiyonları üzerinde test edilerek literatürde yer alan diğer optimizasyon algoritmaları ile kıyaslanmıştır.

## 1.2. Benchmark fonksiyonları çeşitleri

Çalışmada üç kategoriye ayrılan 23 adet Benchmark fonksiyonu kullanılmıştır:

- n boyutlu tek modlu
- n boyutlu çok modlu
- sabit boyutlu çok modlu .

Tez çalışmasında N boyutlu problemlerde boyut 30 olarak belirlenmiştir.

### 1.2.1. Tek modlu Benchmark fonksiyonları

Tek modlu Benchmark fonksiyonları yerel optimuma sahip değildir ve sadece bir global optimumu var. Bu, yakınsama hızını test etmek ve algoritmalardan yararlanmak için oldukça uygun olmalarını sağlar [1]. Detalı bilgiler Tablo 1.1`de verilmiştir.

Tablo1.1. Tek modlu Benchmark fonksiyonları

| Matematiksel formülasyon | Ad | B | Arama alanı | $f_{min}$ |
|---|---|---|---|---|
| F1(x)=$\sum_{i=1}^{n} x_i^2$ | Sphere | 30 | [-100,100] | 0 |
| F2(x)=$\sum_{i=1}^{n}\|x_i\| + \prod_{i=1}^{n}\|x_i\|$ | Schwefel2.22 | 30 | [-10,10] | 0 |
| F3(x)=$\sum_{i=1}^{n}(\sum_{j-1}^{i} x_j)^2$ | Schwefel1.2 | 30 | [-100,100] | 0 |
| F4(x)=$max_i\{\|x_i\|, 1 \le i \le n\}$ | Schwefel2.21 | 30 | [-100,100] | 0 |
| $F5(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | Rosenbrock | 30 | [-30,30] | 0 |
| F6(x)=$\sum_{i=1}^{n}([x_i + 0.5])^2$ | Step | 30 | [-100,100] | 0 |
| F7(x)=$\sum_{i=1}^{n} ix_i^4 + random[0,1)$ | Quartic | 30 | [-1.28,1.28] | 0 |

### 1.2.2. Çok modlu Benchmark fonksiyonları

Çok modlu test fonksiyonları birden fazla optimaya sahiptir, bu durum onları tek modlu fonksiyonlardan daha zorlu kılar. Optimalardan birine global optimum, geri kalanına ise yerel optimum denir. Bir algoritma, küresel optimuma yaklaşmak için tüm yerel optimumlardan kaçınmalıdır. Tüm tek ve çok modlu test fonksiyonlarının minimum değeri F8 işlevi hariç 0'a eşittir. F8`de değişkenlerin sayısına bağlı olarak minimumun değeri değişmektedir [1]. Fonksiyonların özellikleri Tablo 1.2`de verilmiştir.

Bir algoritmanın keşif işlemi kötü tasarlanmışsa, o zaman geniş açıda etkili bir arama yapamaz. Bu da, bir algoritmanın yerel optimada sıkışıp kalmasına yol açar. Bu yüzden çok sayıda yerel optima içeren çok modlu fonksiyonlar, birçok algoritma için en zor problem sınıfları arasındadır [2].

Tablo1.2. Çok modlu Benchmark fonksiyonları

| Matematiksel formülasyon | Ad | B | Arama alanı | $f_{min}$ |
|---|---|---|---|---|
| F8(x)=$\sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$ | Schwefel | 30 | [-500,500] | -418.9829× $B$ |
| F9(x)=$\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$ | Rastrigin | 30 | [-5.12,5.12] | 0 |
| F10(x)=-20exp (-0.2$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$) -exp($\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)$)+20+e | Ackley | 30 | [-32,32] | 0 |
| F11(x)=$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | Griewank | 30 | [-600,600] | 0 |
| F12(x)=$\frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$ <br> $y_i = 1 + \frac{x_i+1}{4}$ <br> $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ | Penalized | 30 | [-50,50] | 0 |
| F13(x)= $0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$ | Penalized2 | 30 | [-50,50] | 0 |

### 1.2.3. Sabit boyutlu çok modlu Benchmark fonksiyonları

Çok modlu fonksiyonlarla kıyasta sabit boyutlu çok modlu fonksiyonlar matematiksel formülasyonları nedeniyle tasarım değişkenlerinin sayısının ayarlanmasına izin vermez, ancak farklı bir arama alanı sağlarlar [3]. Tablo 1.3`te fonksiyon özellikleri verilmiştir.

Tablo 1.3. Sabit boyutlu çok modlu Benchmark fonksiyonları

| Matematiksel formülasyon | Ad | B | Arama alanı | $f_{min}$ |
|---|---|---|---|---|
| F14(x)= $(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$ | Foxholes | 2 | [-65,65] | 1 |
| F15(x)= $\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$ | Kowalik | 4 | [-5,5] | 0.00030 |
| F16(x)=$4x_1^2$-2.1$x_1^4$+$\frac{1}{3}x_1^6$+$x_1x_2$-$4x_2^2$+$4x_2^4$ | Camel Six Hump | 2 | [-5,5] | -1.0316 |
| F17(x)=$(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$ | Branin RCOS | 2 | [-5,5] | 0.398 |
| F18(x)=$[1+(x_1+x_2+1)^2(19-14x_1+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$ | GoldStein Price | 2 | [-2,2] | 3 |
| F19(x)=$-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$ | Hartman3 | 3 | [1,3] | -3.86 |
| F20(x)=$-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$ | Hartman6 | 6 | [0,1] | -3.32 |
| F21(x)=$-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$ | Shekel5 | 4 | [0,10] | -10.1532 |
| F22(x)=$-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$ | Shekel7 | 4 | [0,10] | -10.4028 |
| F23(x)=$-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$ | Shekel10 | 4 | [0,10] | -10.5363 |

## 1.3. Benchmark Fonksiyonları Literatür Taraması

Literatür araştırmasına göre Benchmark amaç fonksiyonları sürekli, kesintili, doğrusal, doğrusal olmayan, dışbükey, dışbükey olmayan, tek modlu, çok modlu, ayrıştırılabilir ve ayrıştırılmaz olarak tanımlanabilir [2].

### 1.3.1. Fonksiyon özellikleri

Fonksiyon alanının hangi yönlerinin optimizasyon işlemini zorlaştırdığını ve ne tür bilgilerin belirli fonksiyon türlerini aramak için etkili olduğunu anlamak için Benchmark fonksiyonları; modallık, havzalar, vadiler, ayrıştırılabilir ve boyutluluk gibi çeşitli özellikleri açısından değerlendirilmesi gerekir [4].

#### 1.3.1.1. Modallık

Arama alanındaki belirsiz tepe noktaları sayısı bir fonksiyonun modallığına karşılık gelir. Algoritmalar arama işlemi sırasında bu tepe noktalarla karşılaşırsa algoritmanın bu noktalardan birinde sıkışıp kalma ihtimali vardır. Bu, arama sürecini olumsuz yönde etkileyecektir, çünkü aramayı gerçek optimal çözümlerden uzağa yönlendirebilir [2].

#### 1.3.1.2. Vadi

Vadi, küçük değişikliklerin dar alanları dik iniş bölgeleriyle çevrili olduğunda oluşur [5]. Havzalarda da olduğu gibi minimumlar başlangıçta bu alanlara çekilmektedir. Bir algoritmanın arama işleminin ilerlemesi, vadi tabanında önemli ölçüde yavaşlamalara sebep olabilir.

#### 1.3.1.3. Ayrıştırılabilirlik

Ayrıştırılabilirlik farklı Benchmark fonksiyonlarının zorluk ölçüsüdür. Genel olarak, ayrıştırılabilir fonksiyonlar ayrıştırılmaz emsalleriyle kıyaslandığında çözülmesi nispeten kolaydır, çünkü fonksiyonun her bir değişkeni diğer değişkenlerden bağımsızdır. Tüm parametreler veya değişkenler bağımsızsa, ***n*** tane bağımsız optimizasyon işlem dizisi gerçekleştirilebilir [2]. Salomon'a göre sonuç olarak, her bir tasarım değişkeni veya parametresi bağımsız olarak optimize edilebilir [6].

#### 1.3.1.4. Boyutsallık

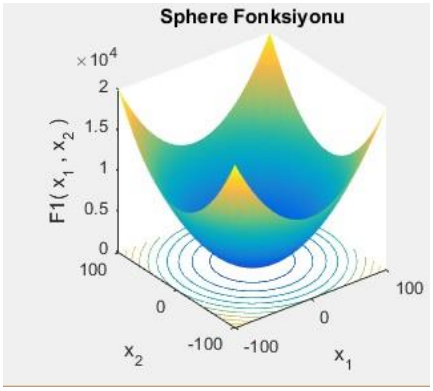
Bir problemin zorluğu genellikle boyutsallığı ile artar. Winston'a [4], Yao ve Liu'ya [7] göre parametre veya boyut sayısı arttıkça arama alanı da üstel olarak artar. Yüksek derecede doğrusal olmayan problemler için bu boyutluluk hemen hemen tüm optimizasyon algoritmaları için önemli bir engel olabilir.

### 1.4. Fonksiyon bilgileri

Benchmark fonksiyonlarının formülleri Tablo1.1-1.3`te, çeşitli özelliklere göre kıyaslanması Tablo1.4`te, fonksiyon grafikleri ise aşağıda detayları belirtilen her fonksiyonun altında Şekil1.1-1.23`e kadar verilmiştir.

#### 1.4.1. F1-Sphere fonksiyonu

Sphere [8] fonksiyonu farklı arama alanlarında tanımlanabilir. Örneğin Jamil ve Yang [2] bunu $0 \leq x_i \leq 10$ olarak belirtmiştir. Bazı kaynaklarda arama alanı $-5.12 \leq x_i \leq 5.12$ olarak verilmiştir [9]. Literatürdeki çalışmaların çoğunda arama alanı $-100 \leq x_i \leq 100$ ve tek modlu olarak belirtilmiş [1] ve tez çalışmasında da bu aralık tercih edilmiştir.

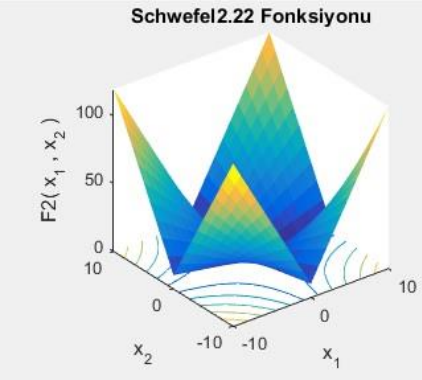



F1(x)=$\sum_{i=1}^{n} x_i^2$

Şekil 1.1. F1-Sphere fonksiyonu grafiği ve denklemi ve denklemi

### 1.4.2. F2-Schwefel2.22 fonksiyonu

Schwefel2.22 [10] fonksiyonunun arama alanını Jamil ve Yang [2] $-100 \leq x_i \leq 100$ olarak belirtmiştir. Çalışmada ise arama alanı $-10 \leq x_i \leq 10$ olarak seçilmiştir [1]

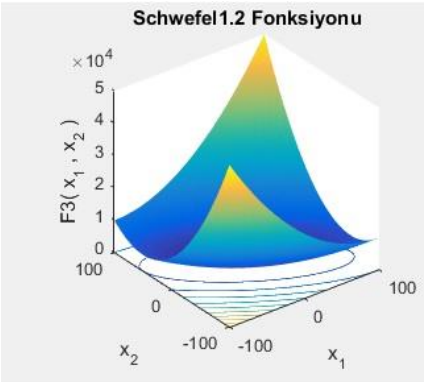



F2(x)=$\sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$

Şekil 2.2. F2-Schwefel2.22 fonksiyonu grafiği ve denklemi ve denklemi

### 1.4.3. F3-Schwefel1.2 fonksiyonu

Schwefel1.2 [10] fonksiyonunun arama alanını Jamil ve Yang [2] ve çalışmada [1] $-100 \leq x_i \leq 100$ olarak belirtilmiştir.

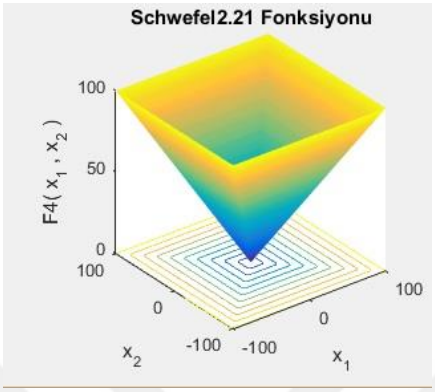



F3(x)=$\sum_{i=1}^{n}(\sum_{j-1}^{i} x_j)^2$

Şekil 1.3. F3-Schwefel1.2 fonksiyonu grafiği ve denklemi

### 1.4.4. F4-Schwefel2.21 fonksiyonu

Schwefel2.21 [10] fonksiyonunun arama alanını Jamil ve Yang [2] ve çalışmada [1] $-100 \leq x_i \leq 100$ olarak belirtilmiştir.

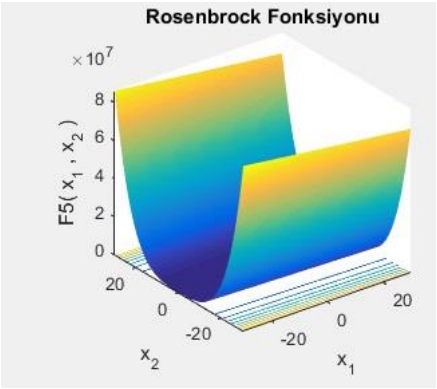



F4(x)=$max_i\{|x_i|, 1 \leq i \leq n\}$

Şekil 1.4. F4-Schwefel2.21 fonksiyonu grafiği ve denklemi

### 1.4.5. F5-Rosenbrock fonksiyonu

Rosenbrock [11] fonksiyonunun arama alanını Jamil ve Yang [2] ve çalışmada [1] $-30 \leq x_i \leq 30$ olarak belirtilmiştir. Bunun dışında başka bir kaynakta [12] bu alan $-5 \leq x_i \leq 10$ olarak olarak gösteriliyor.

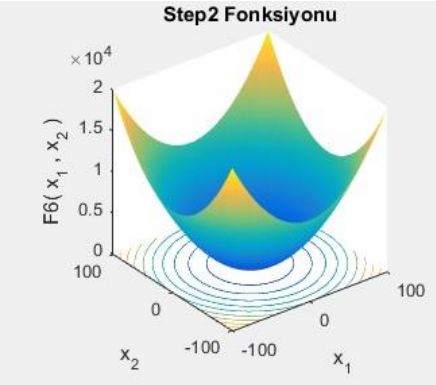



$$F5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

Şekil 1.5. F5-Rosenbrock fonksiyonu grafiği ve denklemi

### 1.4.6. F6-Step2 fonksiyonu

Step2 [13] fonksiyonunun arama alanı $-100 \leq x_i \leq 100$ olarak tanımlanmıştır [2].

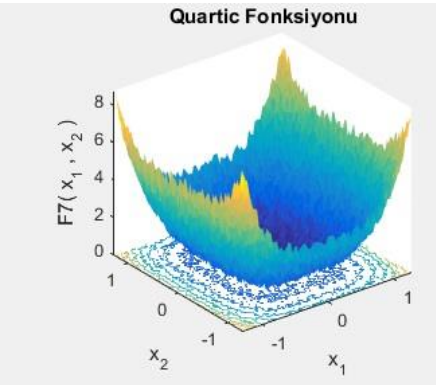



$$F6(x)=\sum_{i=1}^{n}([x_i + 0.5])^2$$

Şekil 1.6. F6-Step2 fonksiyonu grafiği ve denklemi

### 1.4.7. F7-Quartic fonksiyonu

Quartic [14] fonksiyonunun arama alanı $-1.28 \leq x_i \leq 1.28$ olarak tanımlanıyor.

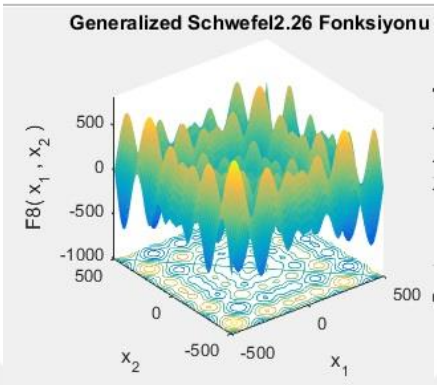



$$F7(x)=\sum_{i=1}^{n} ix_i^4 + random[0,1)$$

Şekil 1.7. F7-Quartic fonksiyonu grafiği ve denklemi

### 1.4.8. F8-Generalized Schwefel2.26 fonksiyonu

Generalized Schwefel2.26 [10] fonksiyonunun arama alanını Jamil ve Yang [2] ve çalışmamızda [1] $-500 \leq x_i \leq 500$ olarak belirtilmiştir.

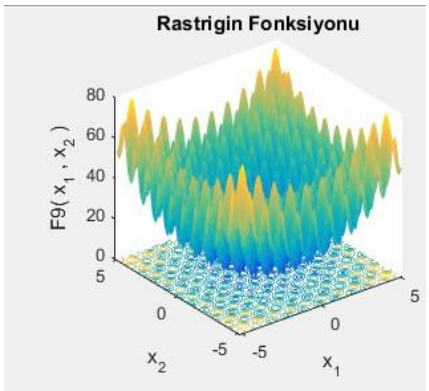



$$F8(x)=\sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})$$

Şekil 1.8. F8-Generalized Schwefel2.26 fonksiyonu grafiği ve denklemi

### 1.4.9. F9-Rastrigin fonksiyonu

Rastrigin fonksiyonu $-5.12 \leq x_i \leq 5.12$ arama alanında belirlenmiştir [1].

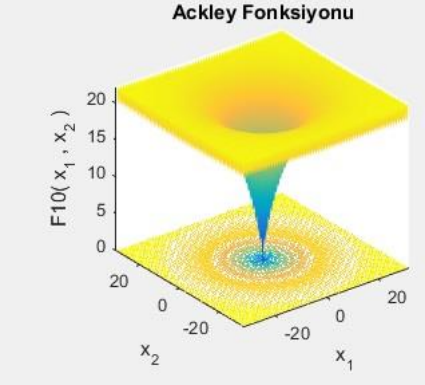



$$F9(x)=\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$$

Şekil 1.9. F9-Rastrigin fonksiyonu grafiği ve denklemi

### 1.4.10. F10-Ackley fonksiyonu

Ackley [12] fonsiyonu Jamil ve Yang`ın [2] çalışmasında arama alanını $-35 \leq x_i \leq 35$, Mirjalili [1] ve Yao ve Lui`de [7] ise $-32 \leq x_i \leq 32$ olarak belirlenmiştir. Fonksiyonun özelliklerinden süreklilik, türevlenebilir, ayrıştırılabilir ve çok modlu olmasına ilişkin özelliklerine ulaşılmıştır [15].

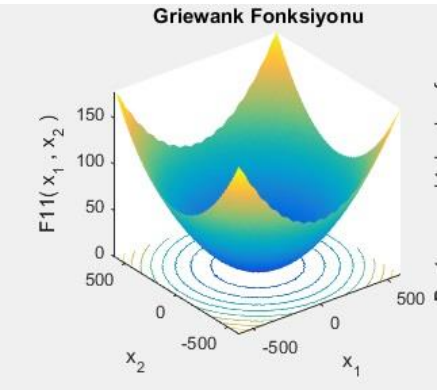



$$F10(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$$

Şekil 1.10. F10-Ackley fonksiyonu grafiği ve denklemi

### 1.4.11. F11-Griewank fonksiyonu

Griewank [16] fonksiyonunun arama alanı Jamil ve Yang`da [2] $-100 \leq x_i \leq 100$, Mirjalili [1] ve Yao ve Lui`de [7] ise $-600 \leq x_i \leq 600$ olarak tanımlanmıştır.

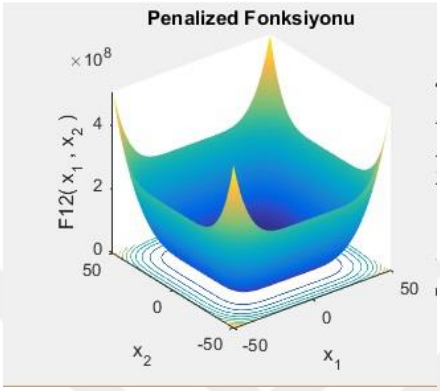



$$F11(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

Şekil 1.11. F11-Generalized Schwefel2.26 fonksiyonu grafiği ve denklemi

### 1.4.12. F12-Penalized fonksiyonu

Penalized fonksiyonunun arama alanı Mirjalili [1] ve Yao ve Lui`de [7] $-50 \leq x_i \leq 50$ olarak belirlenmiştir. Fonksiyonun özelliklerinden sadece çok modlu ve ayrıştırılamaz özelliklerine ulaşmıştır [17].




$$F12(x)= \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$$

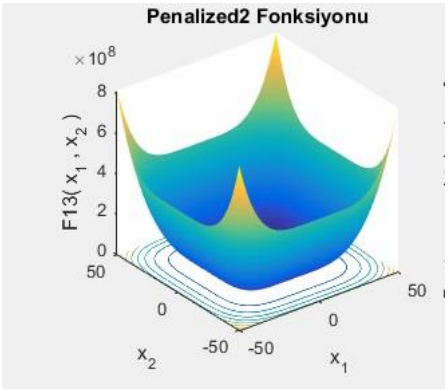
$$y_i = 1 + \frac{x_i+1}{4}$$

$$u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$$

Şekil 1.12. F12-Penalized fonksiyonu grafiği ve denklemi

### 1.4.13. F13-Penalized2 fonksiyonu

Penalized2 fonksiyonunun arama alanı Mirjalili [1] ve Yao ve Lui`de [7] $-50 \leq x_i \leq 50$ olarak belirlenmiştir. Fonksiyon çok modlu ve ayrıştırılamaz özelliklerine sahiptir [17].

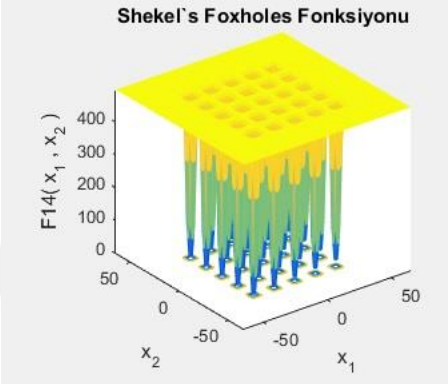



$$F13(x)= 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2[1 + sin^2(3\pi x_i+1)] + (x_n-1)^2[1+sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$$

Şekil 1.13. F13-Penalized fonksiyonu grafiği ve denklemi

### 1.4.14. F14-Shekel`s Foxholes fonksiyonu

Shekel`s Foxholes fonksiyonu süreklil, dış bükey olmayan, karesel olmayan, iki boyutlu 25 yerel optimuma sahip çok modlu Benchmark fonksiyonudur [18]. Fonksiyon hakkında diğer bir bilgi ayrıştırılabilen ve çok modlu olmasıdır [17]. Bazı çalışmalarda [7] arama alanı $-65.536 \leq x_i \leq 65.536$ bazılarında [19] ise $-65 \leq x_i \leq 65$ olarak belirlenmiştir.

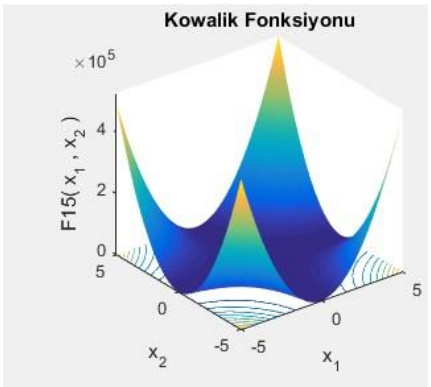



$$F14(x)= \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$$

Şekil 1.14. F14-Shekel`s Foxholes fonksiyonu grafiği ve denklemi

### 1.4.15. F15-Kowalik fonksiyonu

Kowalik fonksiyonu 4 boyutlu, ayrıştırılamayan, $-5 \leq x_i \leq 5$ arama alanına sahip çok modlu Benchmark fonksiyonudur [17].

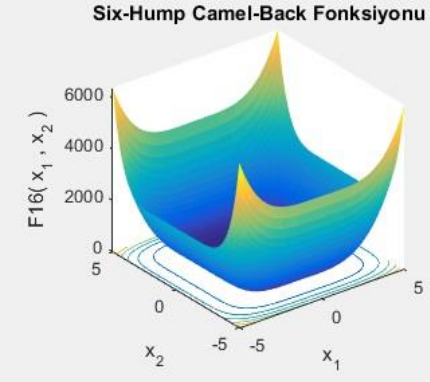



$$F15(x)= \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$$

Şekil 1.15. F15-Kowalik fonksiyonu grafiği ve denklemi

### 1.4.16. F16-Six-Hump Camel-Back fonksiyonu

Six-Hump Camel-Back [20] fonksiyonu $-5 \leq x_i \leq 5$ arama alanına sahip sabit boyutlu çok modlu Benchmark fonksiyonudur [19].

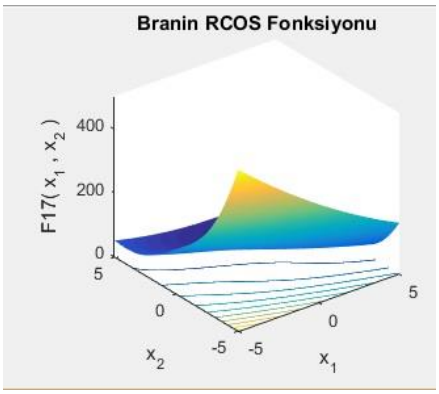



$$F16(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$$

Şekil 1.16. F16-Six-Hump Camel-Back fonksiyonu grafiği ve denklemi

### 1.4.17. F17-Branin RCOS fonksiyonu

Branin RCOS [20] fonksiyonu Jamil ve Yang [2] çalışmasında $-5 \leq x_1 \leq 10,\ 0 \leq x_1 \leq 15$ arama alanına sahip ayrıştırılamaz olarak gösterilirken, Civicioglu [17] çalışmasında $-5 \leq x_i \leq 10$ arama alanına sahip ayrıştırılabilen fonksiyon olarak belirtilmiştir. Kıyaslamalarda kullanılan çalışmada [19] ise arama alanı $-5 \leq x_i \leq 5$ olarak kabul edilmiştir.

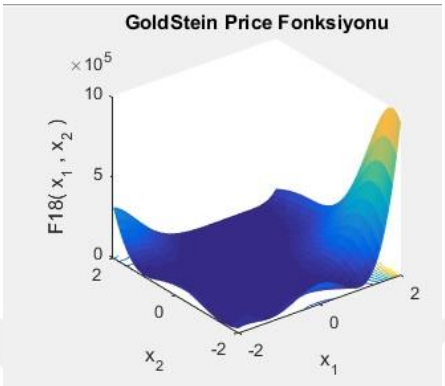



$$F17(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$$

Şekil 1.17. F17-Branin RCOS fonksiyonu grafiği ve denklemi

### 1.4.18. F18-GoldStein-Price fonksiyonu

GoldStein-Price [21] fonksiyonu $-2 \leq x_i \leq 2$ arama alanına sahip ayrıştırılamaz 2 boyutlu çok modlu Benchmark fonksiyoudur [17].

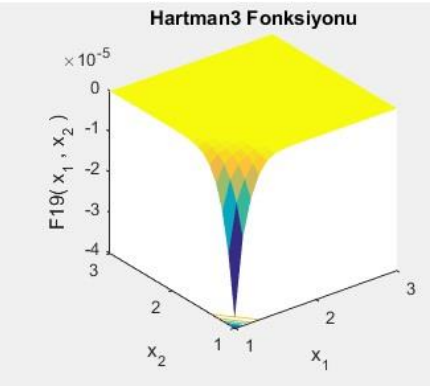



$$F18(x)=[1+(x_1+x_2+1)^2(19-14x_1+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$$

Şekil 1.18. F18-GoldStein Price fonksiyonu grafiği ve denklemi

### 1.4.19. F19-Hartman3 fonksiyonu

Hartman3 [22] $0 \leq x_i \leq 1$ arama alanına sahip ayrıştırılamaz, 3 boyutlu, çok modlu Benchmark fonksiyonudur [17]. Yao ve Lui [7] çalışmasında 4 boyutlu olarak belirtilmiştir.

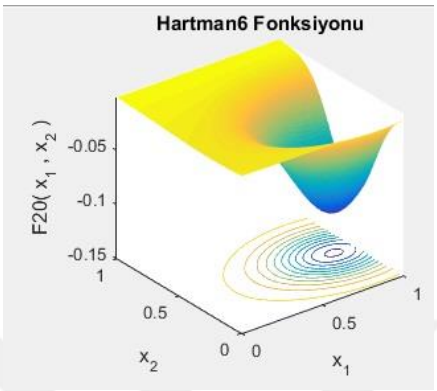



$$F19(x)=-\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j-p_{ij}\right)^2\right)$$

Şekil 1.19. F19-Hartman3 fonksiyonu grafiği ve denklemi

### 1.4.20. F20-Hartman6 fonksiyonu

Hartman6 [22] $0 \leq x_i \leq 1$ arama alanına sahip ayrıştırılamaz, türevlenebilir, ölçeklenemez, sürekli özelliklerine sahip çok modlu Benchmark fonksiyonudur [2].

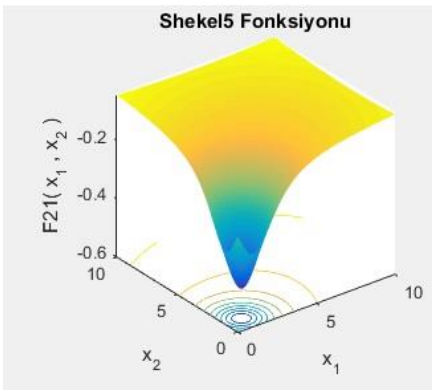



$$F20(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$$

Şekil 1.20. F20-Hartman6 fonksiyonu grafiği ve denklemi

### 1.4.21. F21-Shekel5 fonksiyonu

Shekel5 [23] arama alanı $0 \leq x_j \leq 10$, sürekli, türevlenebilir, ayrıştırılamaz, ölçeklenebilir, çok modlu Benchmark fonksiyonudur [2].

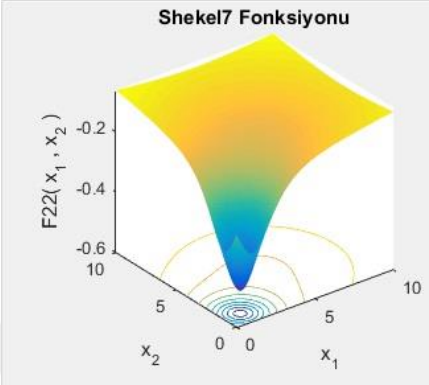



$$F21(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$$

Şekil 1.21. F21-Shekel5 fonksiyonu grafiği ve denklemi

### 1.4.22. F22-Shekel7 fonksiyonu

Shekel7 [23] arama alanı $0 \leq x_j \leq 10$, ayrıştırılamaz, 4 boyutlu, çok modlu Benchmark fonksiyonudur [17].

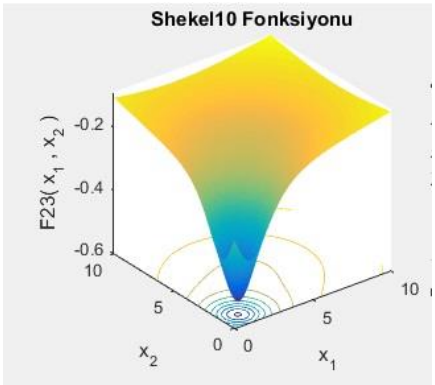



$$F22(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$$

Şekil 1.22. F22-Shekel7 fonksiyonu grafiği ve denklemi

### 1.4.23. F23-Shekel10 fonksiyonu

Shekel10 [23] arama alanı $0 \leq x_j \leq 10$, 4 boyutlu çok modlu Shekel ailesine ait Benchmark fonksiyonudur [7].




$$F23(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$$

Şekil 1.23. F23-Shekel10 fonksiyonu grafiği ve denklemi

Tablo1.4. Fonksiyonların çeşitli özellikleri.

| Fonksiyon | Sürekli | Kesintili | Türevlenebilir | Türevlenemeyen | Ayrıştırılabilir | Ayrıştırılamaz | Kısmen ayrıştırılabilir | Ölçeklenebilir | Ölçeklenemez | Dışbükey | Dışbükey olmayan | Tek modlu | Çok modlu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sphere | + | - | + | - | + | - | - | + | - | + | - | + | - |
| Schwefel2.22 | + | - | + | - | - | + | - | + | - | + | - | + | - |
| Schwefel1.2 | + | - | + | - | - | + | - | + | - | x | x | + | - |
| Schwefel2.21 | + | - | - | + | + | - | - | + | - | + | - | + | - |
| Rosenbrock | + | - | + | - | - | + | - | + | - | - | + | + | - |
| Step | - | + | - | + | + | - | - | + | - | x | x | + | - |
| Quartic | + | - | + | - | + | - | - | + | - | - | + | + | - |
| Generalized Schwefel2.26 | + | - | + | - | - | - | + | + | - | - | + | - | + |
| Rastrigin | + | - | + | - | + | - | - | x | x | + | - | - | + |
| Ackley | + | - | + | - | - | + | - | + | - | - | + | - | + |
| Griewank | + | - | + | - | - | + | - | + | - | - | + | - | + |
| Penalized | x | x | x | x | - | + | - | x | x | x | x | - | + |
| Penalized2 | x | x | x | x | - | + | - | x | x | x | x | - | + |
| Shekel`s Foxholes | + | - | x | x | + | - | - | x | x | - | + | - | + |
| Kowalik | x | x | x | x | - | + | - | x | x | x | x | - | + |
| Six-Hump Camel-Back | + | - | + | - | - | + | - | - | + | x | x | - | + |
| Branin RCOS | + | - | + | - | + | + | - | - | + | x | x | - | + |
| GoldStein-Price | + | - | + | - | - | + | - | - | + | - | + | - | + |
| Hartman3 | + | - | + | - | - | + | - | - | + | x | x | - | + |
| Hartman6 | + | - | + | - | - | + | - | - | + | x | x | - | + |
| Shekel5 | + | - | + | - | - | + | - | + | - | x | x | - | + |
| Shekel7 | + | - | + | - | - | + | - | + | - | x | x | - | + |
| Shekel10 | + | - | + | - | - | + | - | + | - | x | x | - | + |

# 2. BÖLÜM

# META-SEZGİSEL OPTİMİZASYON TEKNİKLERİ

## 2.1. Giriş

Optimizasyon, bir problemin ya iyileştirilmesi ya da en iyi sonucunun bulunması anlamına gelir. Gerçek dünya problemleri çözülmesi zor problemlerdir. Bu problemleri çözmek için çeşitli yöntemler geliştirilmiştir. Son yirmi beş seneye bakıldığında meta-sezgisel teknikler gerçek mühendislik problemlerinin çözülmesinde daha popüler hale geldiği görülmüştür [24-26]. Çünkü oldukça basit konseptte ve uygulaması kolay, gradyan bilgisi gerektirmeyen, yerel optimadan kaçabilen özellikleriyle farklı alanlara ait çok çeşitli problemlere uyarlanmaktadır [3]. Mirjalili [19] çalışmalarında bunu dört kategoride yorumlamıştır:

- Basitlik
- Esneklik
- Türevsiz mekanizma
- Yerel optimadan kaçınma

Öncelikle meta-sezgisel yöntemler basittir. Basit olmasının sebebi fiziksel olaylar, hayvanların davranışları, evrimsel kavramlar gibi tipik, basit kavramlardan esinlenmesidir. Bu özellik bilgisayar bilimcilerinin işini kolaylaştırmaktadır. Bunun sayesinde bilim insanları farklı doğa kavramlarından esinlenerek yeni meta-sezgisel yöntemler ortaya çıkarmaktadır. Bununla yetinmeyip onları hibritleştirerek veya mevcut olanı iyileştirerek daha iyi sonuçlar elde edilmektedir. Esinlenmenin basitliği bilim insanlarının bunu daha kolay ve hızlı şekilde benimseyip ilgili alanlara uyarlamasına yardımcı olur [19].

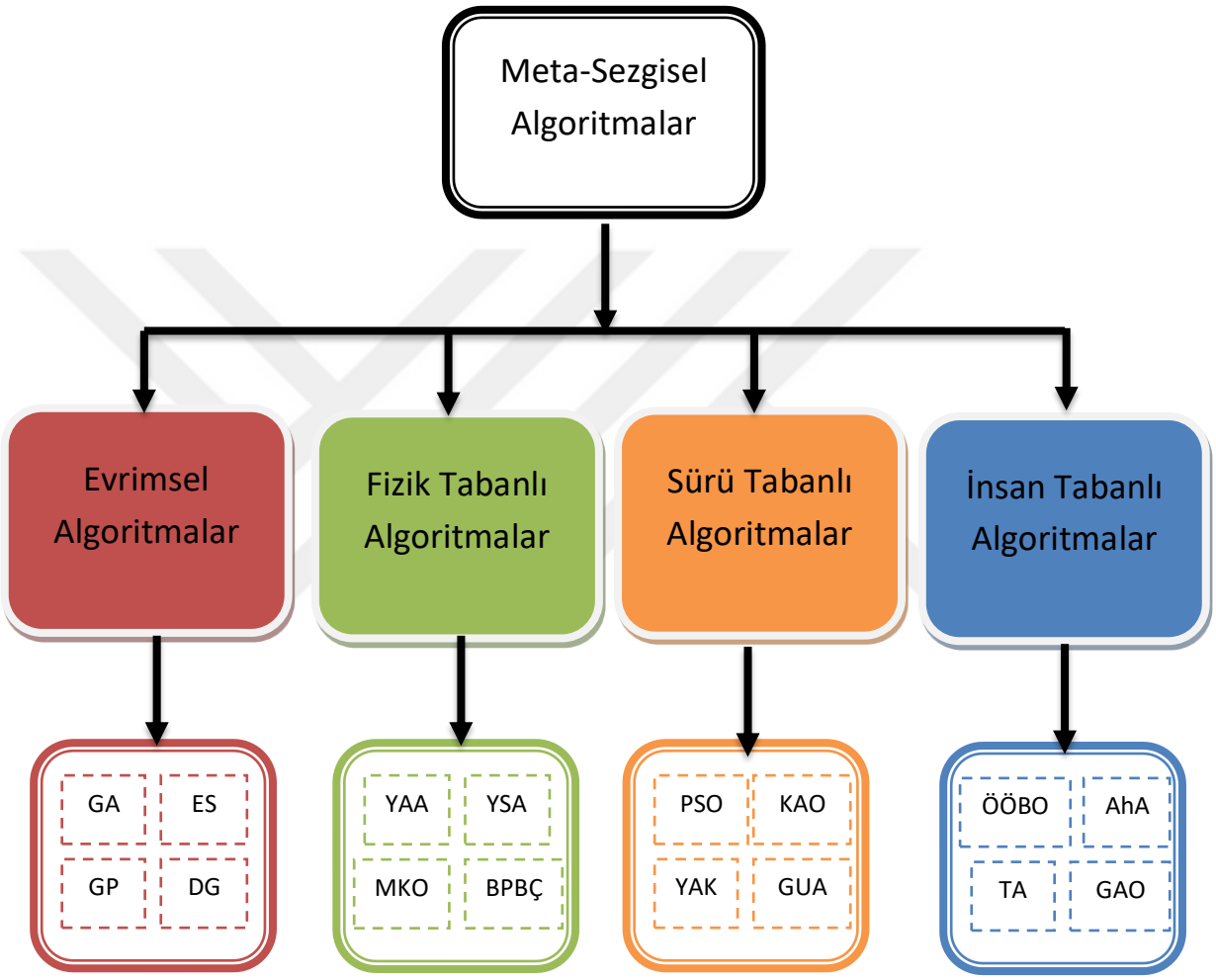
Esneklik kavramı yukarıda da belirtildiği gibi farklı alanlara uyarlanmayı mümkün kılar. Meta-sezgisel tekniklerde sadece giriş ve çıkışlar önemli olduğu için tasarımcının tek işi problemini meta-sezgisel teknik için nasıl sunacağıdır [19].

Türevsiz mekanizma sayesinde meta-sezgisel teknikler gradyan temelli optimizasyon yaklaşımlarının aksine problemleri stokastik olarak optimize eder. Arama alanlarının pahalı veya bilinmeyen türevlerini hesaplama gereği duymadan işlemi rastgele çözümlerle başlayarak optimize eder. Bu da gerçek problemlere uyarlamada büyük bir avantaj sağlar [19].

Meta-sezgisel teknikler geleneksel yöntemlere kıyasla yerel optimaya takılıp kalmamak için özel tekniklere sahiptir. Bunun sebebi meta-sezgisellerin stokastik yapıya sahip olmasıdır. Bu sayede yerel optimadan kaçarak daha geniş bir alanda tarama yapılmasını sağlar. Gerçek dünya problemlerinde arama alanı çoğu zaman bilinmediği ve çok sayıda yerel optimum olduğu için çözüm zorlaşır. Bu yüzden meta-sezgisel tekniklerle optimize etmek en iyi seçeneklerdendir [19].

## 2.2. Meta-sezgisel algoritmalar

Meta-sezgisel algoritmalar farklı doğa olaylarından esinlenen doğa olaylarının biyolojik yönlerini, fiziksel davranışlarını taklit ederek geliştirilen optimizasyon algoritmalarıdır. Mirjalili`ye [3] göre esinlenme dört farklı kategoride gösterilebilir (Şekil 2.1).




Şekil 2.1. Meta-Sezgisel Algoritmaların Sınıflanırılması.

Kaynak: [3]

Nüfusa dayalı meta-sezgisel optimizasyon algoritmaları, niteliklerinden bağımsız olarak ortak bir özelliği paylaşır. Arama süreci iki aşamaya ayrılır: keşif ve sömürü [27-29]. Keşif, bir arama alanının tamamen yeni bölgelerini ziyaret etme işlemi iken, sömürü ise daha önce ziyaret edilen noktaların mahallesindeki bir arama alanının bu bölgelerini ziyaret etme işlemidir [30]. Optimize edici arama alanını global olarak araştırabilmek için

operatörler içermelidir: bu aşamada hareketler (yani tasarım değişkenlerinin karışıklığı) mümkün olduğunca rastgele seçilmelidir. Sömürü aşaması keşif aşamasını izler ve arama alanının gelecek vaat eden alanlarını ayrıntılı olarak araştırır. Dolayısıyla keşif arama aşamasında bulunan gelecek vaat eden tasarım alanı bölgelerinde yerel arama kabiliyeti ile ilgilidir. Keşif ve sömürü arasında uygun bir denge bulmak, optimizasyon sürecinin stokastik yapısından dolayı, herhangi bir meta-sezgisel algoritma geliştirmede en zor görevdir [3].

Mirjali`ye [19] göre meta-sezgisel teknikler iki ana sınıfa ayrılabilir: tek çözüm tabanlı ve popülasyon temelli. Örneğin BT [31], arama işlemi bir aday çözümle başlar. Bu tek aday çözümü, yinelemeler boyunca geliştirilir. Popülasyon temelli meta-sezgisel tarama, bir dizi çözümü (popülasyon) kullanarak optimizasyonu gerçekleştirir. Bu durumda, arama işlemi rastgele bir başlangıç popülasyonuyla (çoklu çözümler) başlar ve bu popülasyon, yinelemeler boyunca geliştirilir.

Nüfus temelli meta-sezgisel taramaların tek çözüme dayalı algoritmalara göre bazı avantajları aşağıda listelenmiştir:

- Birden fazla aday çözüm arama alanıyla ilgili bilgileri paylaşır ve bu da arama alanının ümit verici kısmına doğru ani sıçramalarla sonuçlanır.
- Birden fazla aday çözüm, yerel olarak en uygun çözümlerden kaçınmak için birbirlerine yardımcı olur.
- Sürü bazlı meta-sezgisel teknikler genellikle tek bir çözüme dayalı algoritmalara kıyasla daha fazla araştırmaya sahiptir.

### 2.2.1. Evrimsel Algoritmalar

Evrimsel algoritmalar adın da anlaşılacağı gibi evrime dayalı yasalardan esinlenmiştir. Genelde arama işlemi sonraki nesilleri oluşturucak bir nüfusla başlar. Bu algoritmaların başarılı yanlarından biri en iyi bireylerin bir sonraki nesli oluşturmak için her zaman bir araya getirilmesidir. Bu da nüfusu nesiller boyunca optimize eder [3]. Evrimsel algoritmalardan literatürde olanlara örnekler Tablo 2.1`de listelenmiştir.

Tablo 2.1. Literatürdeki bir kısım evrimsel algoritmalar

| Algoritm | Esinlenme | Öneri yılı |
|---|---|---|
| Genetik Algortima (GA) [32] | Darvinist evrimi (gen) | 1992 |
| Genetik Programlama (GP)[33] | Popülasyon genetiği | 1992 |
| Evrimsel Strateji (ES) [34] | Doğal seleksiyon yoluyla Evrim teorisinden | 1978 |
| Biyocoğrafya Tabanlı Optimize (BTO)[35] | Biyolojik organizmaların coğrafi dağlımı | 2008 |
| Evrimsel Programlama (EP)[36] | Doğal seleksiyon yoluyla Evrim teorisinden | 1962 |
| Diferansiyel Gelişim (DG) [37] | Biyolojik ve sosyolojik durumlar | 1995 |
| Gen Anlatım Programlama (GAP) [38] | DNA molekülleri | 2001 |

### 2.2.2. Fizik Tabanlı Algoritmalar

Fizik tabanlı algoritmalar evrendeki fizik kurallarından esinlenir. Örnek algoritmalar Tablo 2.2`de listelenmiştir.

Tablo 2.2. Literatürdeki popüler fizik tabanlı algoritmalar

| Algoritm | Esinlenme | Öneri yılı |
|---|---|---|
| Benzetimli Tavlama (BT) [31] | Metalurjide tavlama | 1983 |
| Yerçekimsel Yerel Arama (YYA) [39] | Yerçekimi yasası | 2003 |
| Büyük Patlama Büyük Çöküş (BPBÇ) [40] | Büyük Patlama ve Büyük Çöküş teorisi | 2006 |
| Yerçekimsel Arama Algoritması (YAA) [41] | Yerçekimi yasası | 2009 |
| Yüklü Sistem Araması (YSA) [42] | Coulomb yasası ve hareket yasaları | 2010 |
| Merkezi Kuvvet Optimizasyonu (MKO) [43] | Yerçekimi kinematiği | 2007 |

| | | |
|---|---|---|
| Yapay Kimyasal Reaksiyon Optimizasyon Algoritması (YKROA) [44] | Kimyasal reaksiyon prosesi | 2011 |
| Kara Delik Algoritması (KDA) [45] | Kara deliklerden | 2013 |
| Işın Optimizasyonu (IO) [46] | Işın geçişi | 2012 |
| Küçük Dünya Optimizasyon Algoritması (KDOA) [47] | Küçük dünya fenomeninin mekanizması | 2006 |
| Galaksi Tabanlı Arama Algoritması (GTAA) [48] | Kuyruklu yıldızlar | 2011 |
| Kavisli Uzay Optimizasyonu (KUO) [49] | Genel İzafiyet teorisi | 2012 |

### 2.2.3. Sürü Tabanlı Algoritmalar

Üçüncü grup farklı hayvan gruplarının çeşitli sosyal davranışlarından esinlenen sürü tabanlı algoritmalardır. En popülerlerinden biri Kennedy ve Eberhart [50] tarafından geliştirilen Parçacık Sürüsü Optimizasyonu'dur. PSO kuş sürüsünün sosyal davranışından ilham almıştır. En iyi çözümü bulmak için arama sahasında dolaşan birkaç parçayı (aday çözümler) kullanır (yani en uygun konum). Bu arada hepsi yollarında en iyi yolu (en iyi çözüm) izlerler. Başka bir deyişle, parçacıklar sürünün şimdiye kadar elde ettiği en iyi çözüm kadar kendi en iyi çözümlerini de düşünürler. Başka popüler bir sürü tabanlı algoritma ise ilk önce Dorigo vd. [51] tarafından geliştirilen Karınca Kolonisi Optimizasyonudur. KKO karıncaların yuvadan en yakın yolu ve yiyecek kaynağını bulmadaki sosyal zekâsını kullanmasından esinlenmiştir. Aday çözümler tarafından yineleme sürecinde feromon matrisi geliştirilmiştir [3].

Diğer sürü temelli teknikler Tablo 1'de listelenmiştir. Bu meta-sezgisel yöntem grubu PSO'nun evrim temelli ve fiziksel temelli algoritmalarla çok rekabetçi olduğu kanıtlandığından beri oldukça çekici olmaya başlamıştır. Çünkü sürü tabanlı algoritmaların evrim tabanlı algoritmalara göre bazı avantajları vardır. Örneğin, sürü tabanlı algoritmalar sonraki yinelemelere karşı arama alanı bilgilerini korurken evrim tabanlı algoritmalar yeni bir nüfus oluşturulduktan hemen sonra herhangi bir bilgiyi atar. Bunun dışında evrimsel yaklaşımlara (seçim, çaprazlama, mutasyon, seçkinlik vb.) kıyasla daha az operatör içerir ve bu nedenle uygulanması daha kolaydır.

Tablo 2.3. Literatürdeki geliştirilen sürü tabanlı optimizasyon algoritmaları.

| Algoritm | Esinlenme | Öneri yılı |
|---|---|---|
| Parçacık Sürü Optimizasyonu (PSO) [50] | Kuş sürüsü | 1995 |
| Bal Arıları Evlilik Optimizasyon Algoritması (BAEOA) [52] | Bal arıları | 2001 |
| Yapay Balık Sürüsü Algoritması (YBSA)[53] | Balık sürüsü | 2003 |
| Termit Algoritması (TA) [54] | Termit sürüsü | 2005 |
| Karınca Kolonisi Optimizasyonu (KKO) [51] | Karınca kolonisi | 2006 |
| Yapay Arı Kolonisi (YAK) [55] | Bal arısı | 2006 |
| Yaban Arısı Sürüsü Algoritması (YASA)[56] | Parazit yaban arısı | 2007 |
| Maymun Arama (MA) [57] | Maymun | 2007 |
| Kurt Sürüsü Arama Algoritması (KSAA) [58] | Kurt sürüsü | 2007 |
| Arı Polen Toplama Algoritması (APTA) [59] | Arılar | 2008 |
| Guguk Arama (GuA) [60] | Guguk | 2009 |
| Yunus Partner Algoritması (YPA) [61] | Yunus | 2009 |
| Yarasa-esinlenme Algoritması (YeA) [62] | Yarasa sürüsü | 2010 |
| Ateş Böceği Algoritması (ABA) [63] | Ateş böceği | 2010 |
| Av Arama (AvA) [64] | Hayvan sürüleri | 2010 |
| Kuş Çiftleşme Optimizasyonu (KÇO) [65] | Kuş çiftleşmesi | 2012 |
| Krill Sürüsü (KS) [66] | Krill sürüsü | 2012 |
| Meyve Sineği Optimizasyon Algoritması (MSOA) [67] | Meyve sineği | 2012 |
| Yunus Ekolokasyonu (YE) [68] | Yunus | 2013 |

### 2.2.4. İnsan Tabanlı Algoritmalar

Literatürdeki insan davranışlarından ilham alan meta-sezgisel yöntemleri de belirtmekte fayda var. Popüler olanlarından bir kısmı Tablo 2.4`te belirtilmiştir.

Tablo 2.4. Literatürdeki popüler insan tabanlı algoritmalar

| Algoritm | Esinlenme | Öneri yılı |
|---|---|---|
| Öğretme Öğrenme Bazlı Optimizasyon (ÖÖBO) [69] | Sınıf (öğretmen-öğrenci) | 2012 |
| Ahenk Arama (AhA) [70] | Müzik | 2001 |
| Tabu Arama (TA) [71] | Hafıza ve keşif | 2009 |
| Grup Arama Optimizasyonu (GAO) [72] | Hayvanların arama yeteneği | 2006 |
| Emperyalist Rekabetçi Algoritma (ERA) [73] | Emperyalist yönetimi | 2007 |
| Lig Şampiyonluğu Algoritması (LŞA) [74] | Spor ligi | 2009 |
| Havai Fişek Algoritmasl (HFA) [75] | Havai fişek patlaması | 2010 |
| Çarpışan Cisimler Optimizasyonu (ÇCO)[76] | İki cisim çarpışması | 2014 |
| İç Arama Algoritması (İAA) [77] | İç tasarım ve dekorasyon | 2014 |
| Mayın Patlaması Algoritması (MPA) [78] | Mayın patlaması | 2013 |
| Futbol Ligi Yarışması Algoritması(FLYA)[79] | Futbol | 2013 |
| Arayıcı Optimizasyon Algoritması(AOA)[80] | İnsan gruplarının araması | 2007 |
| Sosyal Tabanlı Algoritma (STA) [81] | İnsan | 2013 |
| Döviz Piyasası Algoritması (DPA) [82] | Döviz piyasası kuralları | 2014 |
| Grup Danışmanlık Optimizasyonu (GDO) [83] | İnsanın grup içinde problem çözme yeteneği | 2014 |

## 2.3. No Free Lunch (NFL) teoremi

No Free Lunch (NFL) teoremi [84] tüm optimizasyon problemlerini çözmek için en uygun meta-sezgisel teknik olmadığını kanıtlamıştır. Başka bir deyişle, belirli bir meta-sezgisel tarama bir dizi problemde çok umut verici sonuçlar gösterebilir ancak aynı algoritma farklı bir problem setinde düşük performans gösterebilir. NFL teoremi çalışma alanını aktif hale getirir, mevcut yaklaşımların iyileştirilmesine ve her yıl yeni meta-sezgisel yöntemlerin önerilmesine neden olur.

## 2.4. Tezde Kıyaslanan Optimizasyon Algoritmaları

Çalışmada yapılan hibritler farklı algoritmaların sonuçları ile kıyaslanarak değerlendirilmiştir. Her iki hibrit makalelerde bulunan hazır sonuçlarla kıyaslanmıştır. Kıyaslamada GA, PSO, YA, ÇTA, GuA, ABA, MAD, YAA, HEP algoritmaları kullanılmıştır.

### 2.4.1. Genetik Algoritma (GA)

GA [32] uyarlanabilir bir strateji ve global optimizasyon tekniğidir. Evrimsel Algoritmadır ve daha geniş bir evrimsel hesaplama çalışmasına aittir. Genetik Programlama, Evrim Stratejiler, Evrimsel Programlama ve Öğrenme Sınıflandırıcı Sistemler gibi diğer Evrimsel Algoritmaların bir kardeşidir. GA listelenemeyecek kadar çok sayıda değişken teknik ve alt alanların ebeveynidir [85]

Genetik Algoritma, nüfus genetiğinden (kalıtım ve gen frekansları dahil) ve nüfus seviyesindeki evrimin yanı sıra, Mendel yapısının (kromozomlar, genler, aleller) ve mekanizmaların (rekombinasyon ve mutasyon gibi) anlaşılmasını da içerir. Bu evrimsel biyolojinin yeni ya da modern sentezidir [85].

### 2.4.2. Parçacık Sürü Optimizasyonu (PSO)

Parçacık Sürü Optimizasyonu sürü zekâsı alanına aittir ve hesaplamalı zekânın bir alt alanıdır. PSO, KKO gibi diğer sürü zekâlı algoritmalarla ilişkilidir ve listelenmek için çok fazla sayıda varyasyon için temel bir algoritmadır [86]. PSO kuşların sürü davranışları ve balıklar gibi bazı hayvanların sosyal arama davranışlarından esinlenmiştir.

Algoritmanın amacı tüm parçacıkların optimayı çok boyutlu bir hiper hacimde konumlandırmasıdır. Başlangıçta uzaydaki tüm parçacıklara rastgele konumlar ve küçük başlangıç rastgele hızları atanarak elde edilir. Algoritma her parçanın konumunu hızına, problem uzayındaki en iyi bilinen küresel pozisyona ve bir parçacık için bilinen en iyi pozisyona dayalı olarak ilerleten bir simülasyon gibi yürütülür. Amaç fonksiyonu her pozisyon güncellemesinden sonra örneklenir. Zaman içinde, arama alanındaki bilinen iyi pozisyonların keşfedilmesi ve sömürülmesinin bir kombinasyonu yoluyla, parçacıklar bir optima veya birkaç optima etrafında toplanır veya birleşir.

### 2.4.3. Yarasa Algoritması (YA)

2010 yılında Xin-She Yang tarafından geliştirilen YA mikro yarasaların yankı veya biyosonar özelliklerine dayanıyor [87]. Mikro yarasalar tipik olarak avı tespit etmek, engelleri önlemek ve karanlıkta çatlak yarıklarını bulmak için yankı adı verilen bir tür sonar kullanır. Çok yüksek bir ses darbesi yayabilir ve çevreleyen nesnelerden geri dönen yankıları dinleyebilirler [88].

Gerçekte mikro yarasalar üç boyutlu çevreyi algılamak için kulakları ve ses yüksekliği değişimleri arasındaki zaman gecikmesini de kullanabilmektedir. Ama temel olarak yankıların bazı özellikleri ile ilgilenilmektedir [89].

### 2.4.4. Çiçek Tozlama Algoritması (ÇTA)

Çiçek tozlaşma doğal dünyada ilginç bir işlemdir. Evrimsel özellikleri yeni optimizasyon algoritmaları tasarlamak için kullanılmıştır. Bir çiçeğin temel amacı tozlaşma yoluyla çoğaltılmasıdır. Çiçek tozlaşması tipik olarak polenin transferiyle ilişkilidir ve bu transfer genellikle böcekler, kuşlar, yarasalar ve diğer hayvanlar gibi tozlayıcılarla bağlantılıdır.

Her bitkinin birden fazla çiçeği olabilir ve her bir çiçek yaması genellikle milyonlarca ve hatta milyarlarca polen gameti salgılar. Ancak sadelik için algortima haline getirirken her bitkinin sadece bir çiçeğe sahip olduğunu ve her çiçeğin sadece bir polen gameti ürettiği varsayılıyor. Bu nedenle, bir polen gameti, çiçeği, bir bitkiyi veya bir çözümü bir probleme ayırmaya gerek yoktur [90]

### 2.4.5. Guguk Arama (GuA)

Guguklu büyüleyici kuşlar sadece yapabilecekleri güzel sesler yüzünden değil aynı zamanda agresif üreme stratejileri nedeniyle Ani ve Guira gugukluları gibi bazı türler yumurtalarını ortak yuvalara koyarlar ancak kendi yumurtalarının çıkım olasılığını artırmak için diğerlerinin yumurtalarını çıkarabilirler [91].

Algoritma oluşturulurken sadelik için bazı kriterler kabul edilir. Her guguk kuşu bir kerede bir yumurta bırakır ve rastgele seçilen bir yuvaya atar. Yüksek kalitede yumurta içeren en iyi yuvalar (solüsyonlar) gelecek nesillere taşınır. Kullanılabilir konak yuvalarının sayısı sabittir ve bir ev sahibi belli olasılıklı bir yabancı yumurtayı

keşfedebilir. Bu durumda, ev sahibi kuş yumurtayı fırlatıp atabilir veya tamamen yeni bir yuva inşa etmek için yuvayı terk edebilir [92].

### 2.4.6. Ateş Böceği Algoritması (ABA)

Ateşböceklerinin yanıp sönen ışığı tropikal ve ılıman bölgelerde yaz göğünde muhteşem bir manzaradır. Yaklaşık iki bin ateş böceği türü var ve çoğu ateşböceği kısa ve ritmik flaşlar üretiyor. Yanıp sönme şekli genellikle belirli bir tür için benzersizdir. Flaşların iki temel eşleşen ortakları (iletişim) çekmek ve potansiyel avları çekmek işlevi vardır. Ek olarak yanıp sönme koruyucu bir uyarı mekanizması olarak da işlev görebilir. Ritmik flaş yanıp sönme hızı ve zaman miktarı her iki cinsiyeti bir araya getiren sinyal sisteminin bir parçasını oluşturur.

Yanıp sönen ışık optimize edilecek olan amaç fonksiyonuyla ilişkili olacak şekilde formüle edilebilir, bu da yeni optimizasyon algoritmalarının formüle edilmesini mümkün kılar [93].

### 2.4.7. Madde Arama Durumları (MAD)

Maddenin gaz, sıvı, katı hallerindeyken moleküllerin hareketlerine dayalıdır. Bu tür moleküllerin hareketi termal enerjinin hareketine benzetme ile motive edilir. Her bir molekülün hareketinin hızı ve yönü çarpışma, çekim kuvvetleri ve molekül setinin deneyimlediği rastgele fenomenler göz önüne alınarak belirlenir [94]. Yaklaşımda bu tür davranışlar hepsi gerçek fizik yasalarının davranışını taklit eden yön vektörü, çarpışma ve rastgele konum operatörleri gibi birkaç operatör tanımlanarak uygulanmıştır [95].

### 2.4.8. Yerçekimsel Arama Algoritması (YAA)

YAA`da ajanlar nesne olarak kabul edilir ve performansları kütleleri ile ölçülür. Tüm bu nesneler yerçekimi kuvveti ile birbirlerini çeker ve bu kuvvet tüm nesnelerin ağır kütleli nesnelere doğru küresel bir hareketine neden olur. Dolayısıyla kütleler kütleçekim kuvveti vasıtasıyla doğrudan bir iletişim şekli kullanarak işbirliği yapar. İyi çözümlere karşılık gelen ağır kütleler, hafif olanlardan daha yavaş hareket eder, bu da algoritmanın kullanılma adımını garanti eder. YAA'da her kütle (ajan) dört spesifikasyona sahiptir: pozisyon, atalet kütlesi, aktif kütleçekim kütlesi ve pasif kütle kütlesi. Kütlenin konumu sorunun çözümüne karşılık gelir ve yerçekimi ve atalet kütleleri bir uygunluk işlevi kullanılarak belirlenir. Başka bir deyişle her kütle bir çözüm sunar ve algoritma yerçekimi

ve atalet kütlelerini uygun şekilde ayarlayarak yönlendirilir. Zaman atlamalı olarak kitlelerin en ağır kütle tarafından çekilmesi beklenir. Bu kütle arama alanında optimum bir çözüm sunar [96].

### 2.4.9. Hızlı Evrimsel Programlama (HEP)

Evrimsel Programlamanın multimodal optimizasyon problemlerinin bazılarını çözmedeki bir dezavantajı yavaşça yakınlaşmasının iyi bir optimum seviyeye yakınlaşmasıdır. EP'nin üret ve test et formülasyonu mutasyonun mevcut olanlardan yeni çözümler üreten anahtar bir arama operatörü olduğunu gösterir. Cauchy rastgele sayılarına dayanan yeni bir mutasyon operatörü 23 işlevden oluşan Benchmark problemleri üzerinde test edilmiştir. Cauchy mutasyonlu yeni EP, Gauss mutasyonunu kullanan klasik EP (GEP), birkaç yerel minima ile unimodal ve multimodal fonksiyonlar için GEP ile karşılaştırılabilirken, birçok yerel minima ile bir dizi multimodal fonksiyon üzerinde kullanılır. Yeni EP "hızlı EP" (HEP) olarak belirtilmiştir.

Farklı problemler için HEP ve GEP'in göreceli gücünü ve zayıflığını değerlendirmek için hem HEP hem de GEP'nin kapsamlı deneysel çalışmaları yapılmıştır. Sonuçlar, Cauchy mutasyonunun büyük bir multimodal fonksiyon optimizasyon problemleri sınıfı için etkili bir arama operatörü olduğunu göstermektedir. HEP'de kullanılan tüm parametreler GEP'de kullanılanlara eşit olarak ayarlandığı için HEP'in performansının daha da iyileştiği gözlemlenmiştir [97].

# 3. BÖLÜM

# HİBRİTLEŞTİRMEDE KULLANILAN OPTİMİZASYON ALGORİTMALARI

## 3.1. Giriş

Önceki bölümlerde de bahs edildiği gibi literatüre yeni bir optimizasyon tekniği kazandırmak için onu belli test problemlerine tabi tutmak gerekmektedir. Bunun dışında algoritmalar ya iyileştirilebilir ya da iki veya daha fazla algoritmanın hibridi yapılarak daha iyi sonuçlar elde edilebilir. Çalışmada fizik tabanlı SKA ve sürü tabanlı KAO algoritmalarının ve de SKA, BAO ve ÇEO algoritmalarının hibritleri yapılarak çeşitli Benchmark fonksiyonlarıyla testlere tabi tutulmuştur. Bu bölümde hibritleştirmede kullanılacak algoritmaların özellikleri ile ilgili detaylı bilgi verilecektir.

## 3.2. Sinüs Kosinüs Algoritması (SKA)

Sinüs-Kosinüs algoritması Mirjalili [98] tarafından 2016 yılında önerilen bir popülasyon bazlı metasezgisel algoritmadır. İlk olarak SKA rastgele bir dizi çözüm oluşturur. Sonra, kendi amaç fonksiyonu değerine göre başka çözümler için hedef olacak şekilde en iyi bireysel çözümü seçer. Daha sonra ilk popülasyondaki her birey (3.1) ve (3.2) denklemleri kullanılarak en iyi çözüme ilişkin olarak konumunu günceller.

$$X_i^{t+1} = X_{\text{İ}}^T + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_{\text{İ}}^T \right| \tag{3.1}$$

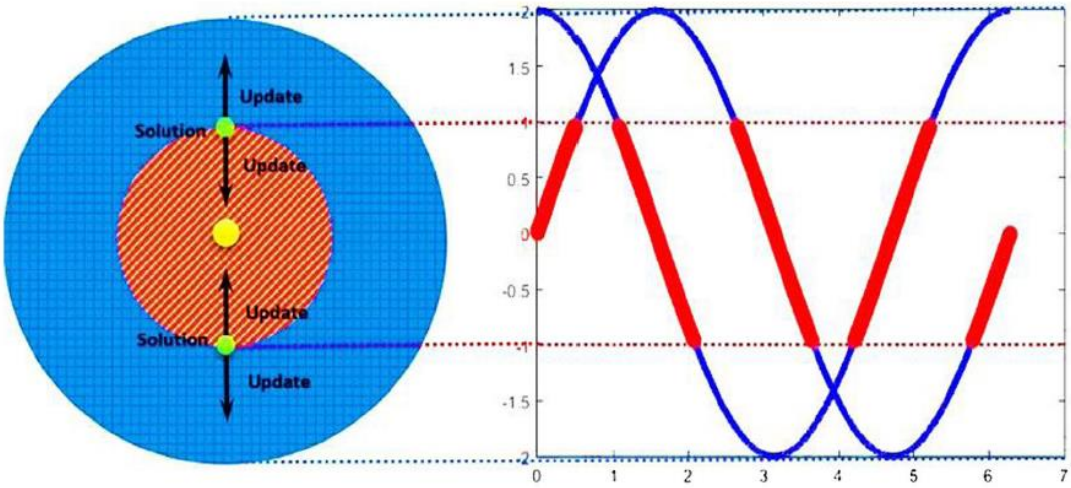
$$X_i^{t+1} = X_{\text{İ}}^T + r_1 \times \cos(r_2) \times \left| r_3 P_i^t - X_{\text{İ}}^T \right| \tag{3.2}$$

$X_{\text{İ}}^T$ t. iterasyonda populasyonda bulunan bireydir. $P_i^t$ o ana kadar elde edilen en iyi bireyin pozisyonudur ve $r_1, r_2, r_3$ rastgele sayılardır. (3.1) ve (3.2) denklemleri birleştirilerek son güncelleme denklemi (3.3) önerilmiştir.

$$X_i^{t+1} = \begin{cases} X_i^T + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^T \right|, & r_4 < 0.5 \\ X_i^T + r_1 \times \cos(r_2) \times \left| r_3 P_i^t - X_i^T \right|, & r_4 \geq 0.5 \end{cases} \quad (3.3)$$

Yukarıdaki denklemde $\boldsymbol{r_1}$ parametresi şimdiye kadar elde edilen en iyi çözüm ile çözüm arasındaki veya dışındaki boşlukta olan güncelleme yönünü belirler. $\boldsymbol{r_2}$ parametresi şimdiye kadar elde edilen en iyi çözüme göre güncelleme mesafesini belirler [99]. $\boldsymbol{r_3}$ parametresi hedefin mesafeyi tanımlamadaki etkisini stokastik olarak vurgulamak **($r_3$ > 1)** veya önemini azaltmak **($r_3$ < 1)** için rastgele bir ağırlık getirir. Son olarak $\boldsymbol{r_4}$ parametresi denklem (3.3) 'te sinüs ve kosinüs bileşenleri arasında eşit şekilde geçiş yapar [94].

Şekil 3.1 sinüs kosinüs arasındaki değişimin etkilerini [−2, 2] aralığında göstermektedir. Bu seri SKA'da keşif yapmayı garanti eder çünkü her birey bugüne kadar elde edilen en iyi çözüm ve kendi arasındaki alan dışındaki konumunu güncelleyebilir.



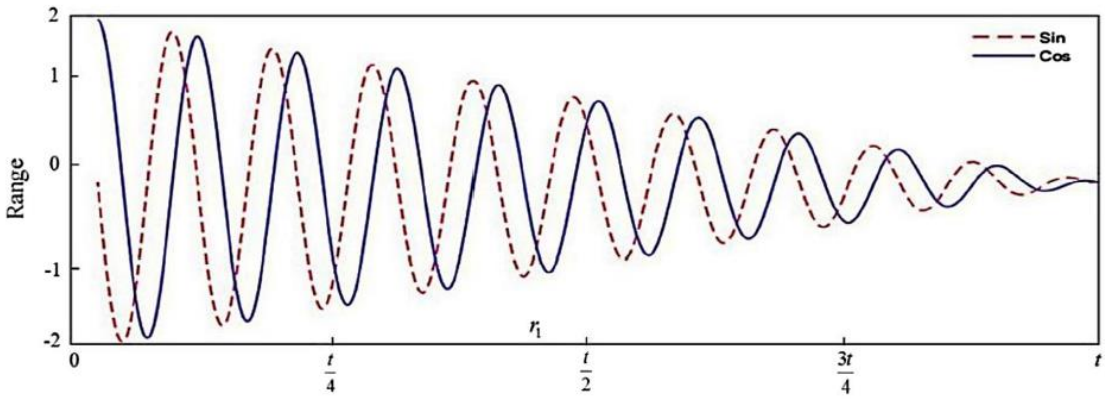

Şekil 3.1. Şimdiye kadar elde edilen en iyi çözüme doğru güncelleme veya uzaklaşma.

Kaynak: [99]

Verimli bir meta-sezgisel algoritma hem keşif hem de sömürü kabiliyeti arasında uygun bir denge sağlayabilen bir algoritmadır. SKA'da hem keşif hem de sömürü arasında uygun bir denge sağlamak için, denklem (3.4) kullanılarak optimizasyon sırasında sinüs ve kosinüs aralığı azaltılır.

$$r_1 = a - t\frac{a}{T} \quad (3.4)$$

***t*** o anki iterasyonu, ***T*** ise maksimum iterasyonu gösteriyor. Şekil 3.2 sinüs ve kosinüs aralığının iterasyonlar (yinelemeler) boyunca nasıl azaldığını göstermektedir. Başka bir deyişle SKA problemin çözüm uzayını [1, 2] ve [−2, −1] 'de sinüs ve kosinüs aralıklarıyla araştırır. SKA'nın araştırılması, sinüs ve kosinüs aralığı [−1, 1] olduğunda gerçekleşir.




Şekil 3.2. İterasyonlar (yinelemeler) boyunca sinüs ve kosinüs aralığını azaltmak.

Kaynak: [99]

### 3.2.1. Sinüs Kosinüs Algoritmasının kaba kodu

Popülasyonunu rastgele **başlat**.

**Do**

Her bir arama temsilcisinin **uygunluğunu hesapla**.

Şimdiye kadar ki en iyi çözümü **güncelle *( P = X*)*** .

$r_1$ $r_2$, $r_3$ ve $r_4$`ü **güncelle.**

arama temsilcilerini **güncelle** denk. **(3.3)**

***while*** ( ***t <*** maksimum iterasyon sayısı )

***Return*** Küresel optimum olarak şimdiye kadar elde edilen en iyi çözüm

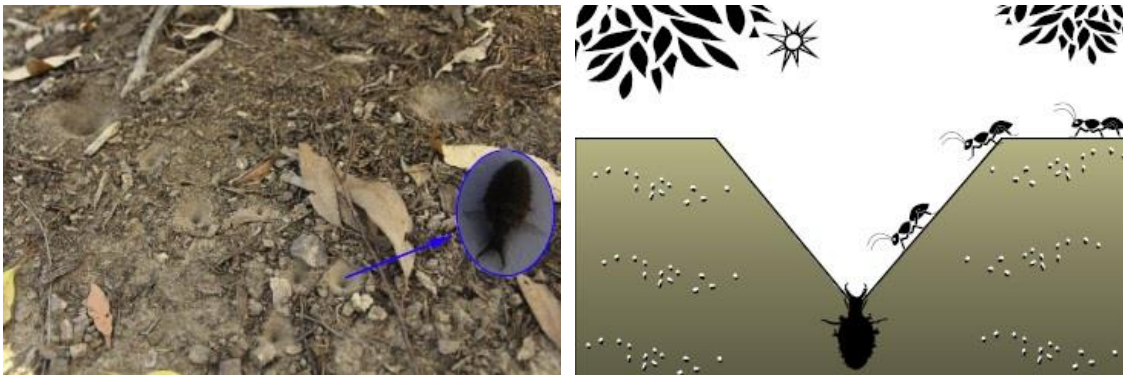
## 3.3. Karınca Aslanı Optimizasyonu (KAO)

KAO [1] 2015 yılında ilk defa Mirjalili tarafından önerilmiştir. Daha sonra algoritmanın çok amaçlı sürümü olan ÇAKAO [100] Mirjalili tarafından 2016 yılında literatüre kazandırılmıştır. Sırasıyla algoritmanın esinlenmesi, operatörleri, kaba kodu anlatılacaktır.

### 3.3.1. Esinlenme

Karınca aslanı Myrmeleontidae familyasına ve Neuroptera ağ kanatlı böcekler takımındandır. Yaşam döngüsü larva ve yetişkin olarak iki ana aşamadan oluşur. Yaşam süreleri toplamda 3 yıla kadar sürebiliyor. Bu süre larva ve sonlara doğru 3 ila 5 hafta arası yetişkinlik sürecini kapsar. Yetişkin olmak için larva bir kozada değişime maruz kalmaktadır. Çoğunlukla larva döneminde avlanır, yetişkinlik döneminde ise ürerler [1].

İsimleri benzersiz avlanma davranışlarından ve en sevdikleri av olan karıncadan geliyor. Larva dairesel bir yol boyunca hareket ederek ve çenesiyle kumları fırlatarak koni biçimli çukur kazmaktadır [101,102]. Şekil 3.3 (a), farklı ebatlardaki koni biçimli çukurları göstermektedir. Tuzağı kazdıktan sonra larva koninin altına gizlenir ve Şekil 3.3b'de gösterildiği gibi çukurun [103] içinde karıncanın tuzağa düşmesini bekler. Koninin kenarı, böceklerin tuzağın dibine kolayca düşmesi için yeterince keskindir. Karınca aslanı bir avın tuzağa düştüğünü fark ettiğinde onu yakalamaya çalışır. Böcekler genellikle hemen yakalanmaz ve tuzaktan kaçmaya çalışırlar. Bu durumda karınca aslanları avı çukurun dibine kaydırmak için akıllıca çukurun kenarına doğru kum atar. Av çenenin içine girdiğinde toprağın altına çekilir ve tüketilir. Avları tükettikten sonra, karınca aslanları artıkları çukurun dışına atar ve çukuru bir sonraki av için değiştirir [1]. KAO`nun ilham kaynağı, larvalarının yiyecek arama davranışından esinlenmiştir.



(a) (b)

Şekil 3.3. Koni şeklindeki tuzaklar ve karınca aslanlarının avlanma davranışları.

Kaynak: [1]

### 3.3.2. Karınca Aslanı Optimizasyon algoritmasının operatörleri

KAO algoritması, tuzağın içinde bulunan karınca aslanları ve karıncalar arasındaki etkileşimi taklit eder. Bu tür etkileşimleri modellemek için karıncaların arama alanı üzerinde hareket etmeleri gerekir ve karınca aslanlarının onları avlamalarına ve tuzakları kullanarak zinde olmalarına izin verilir. Yiyecek ararken, karıncalar stokastik olarak hareket ettiklerinden karıncaların hareketini modellemek için rastgele bir yürüyüş seçilir:

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \dots, cumsum(2r(t_n) - 1)] \tag{3.5}$$

cumsum kümülatif toplamı hesaplar, n maksimum yineleme sayısıdır, t rastgele iterasyon sayısını gösterir ve r (t) aşağıdaki gibi tanımlanmış bir stokastik fonksiyondur:

$$r(t) = \begin{cases} 1 \; if \; rand > 0.5 \\ 0 \; if \; rand \leq 0.5 \end{cases} \tag{3.6}$$

Burada t rastgele yürüme basamağını gösterir (iterasyon) ve rand [0, 1] aralığında tekdüze dağılımla oluşturulan rastgele bir sayıdır.

Karıncaların konumu, aşağıdaki matristeki optimizasyon sırasında kaydedilir ve kullanılır:

$$M_{karınca} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & \dots & A_{2,d} \\ : & : & : & : & : \\ : & : & : & : & : \\ A_{n,1} & A_{n,2} & & & A_{n,d} \end{bmatrix} \quad (3.7)$$

Burada $M_{karınca}$ her bir karıncanın konumunu kaydetme matrisidir. $A_{i,j}$ i. karıncanın j. değişkeninin (boyut) değerini gösterir, n karıncaların sayısıdır ve d değişkenlerin sayısıdır.

Bir karıncanın konumu belirli bir çözelti için parametreleri belirtir. Matrix $M_{Ant}$ optimizasyon sırasında tüm karıncaların konumunu kaydettiği düşünülmüştür.

Her karıncayı değerlendirmek için, optimizasyon sırasında bir zindelik (objektif) fonksiyonu kullanılır ve aşağıdaki matris tüm karıncaların zindelik değerini depolar:

$$M_{ZK} = \begin{bmatrix} f([A_{1,1}, A_{1,2}, \dots, A_{1.d}]) \\ f([A_{2,1}, A_{2,2}, \dots, A_{2.d}]) \\ \cdot \\ \cdot \\ \cdot \\ f([A_{n,1}, A_{n,2}, \dots, A_{n.d}]) \end{bmatrix} \quad (3.8)$$

$M_{ZK}$ her bir karıncanın formunu korumak için kullanılan matristir. Ai, j, i. karıncanın j. boyutunun değerini gösterir, n, karıncaların sayısıdır ve f, amaç fonksiyonudur.

Karıncalara ek olarak karınca aslanlarının da arama alanında bir yerlerde saklandıkları varsayılır. Konumlarını ve uygunluk değerlerini kaydetmek için aşağıdaki matrisler kullanılır:

$$M_{karınca\ aslanı} = \begin{bmatrix} AL_{1,1} & AL_{1,2} & \dots & \dots & AL_{1,d} \\ AL_{2,1} & AL_{2,2} & \dots & \dots & AL_{2,d} \\ : & : & : & : & : \\ : & : & : & : & : \\ AL_{n,1} & AL_{n,2} & & & AL_{n,d} \end{bmatrix} \quad (3.9)$$

$M_{karınca\ aslanı}$ karınca aslanlarının konumunu kaydetme matrisidir.

$$M_{ZKA} = \begin{bmatrix} f([AL_{1,1}, AL_{1,2}, \dots, AL_{1.d}]) \\ f([AL_{2,1}, AL_{2,2}, \dots, AL_{2.d}]) \\ \vdots \\ f([AL_{n,1}, AL_{n,2}, \dots, AL_{n.d}]) \end{bmatrix} \quad (3.10)$$

$M_{ZKA}$ her bir karınca aslanının formunu korumak için kullanılan matristir.

Optimizasyon sırasında, aşağıdaki koşullar uygulanır:

- Karıncalar farklı rastgele yürüyüşleri kullanarak arama alanı içinde hareket eder.
- Karıncaların tüm boyutlarına rastgele yürüyüşler uygulanır.
- Rastgele yürüyüşler karınca aslanlarının tuzaklarından etkilenir.
- Karınca aslanları zindelikleriyle orantılı çukurlar oluşturabilir (daha yüksek zindelik, daha büyük çukur).
- Daha büyük çukurlara sahip karınca aslanları karıncaları yakalama ihtimalinin daha yüksek olduğunu gösterir.
- Her bir karınca, her bir yinelemede bir karınca aslanı ve elit (en uygun karınca aslanı) tarafından yakalanabilir.
- Kayan karıncaları karınca aslanlarına doğru simüle etmek için rastgele yürüyüş aralığında azaltılır.
- Bir karınca bir karınca aslanından daha kurnaz hale gelirse bu karınca aslanı tarafından kumun altında yakalandığı ve çekildiği anlamına gelir.
- Bir karınca aslanı kendisini en son yakalanan avına yerleştirir ve her avdan sonra başka bir av yakalama değişikliğini geliştirmek için bir çukur oluşturur.

#### 3.3.2.1. Karıncaların rastgele yürüşü

Rastgele yürüyüşlerin tümü (3.5) denklemine dayanıyor. Karıncalar pozisyonlarını optimizasyonun her aşamasında rastgele yürüyüşle günceller. Ancak, her arama alanının bir sınırı vardır (değişken aralığı), denklem (3.5) karıncaların pozisyonlarını güncellemek için doğrudan kullanılamaz. Rastgele yürüyüşleri arama alanı içinde tutmak için, aşağıdaki denklem kullanılarak normalleştirilirler (minimum – maksimum normalizasyon):

$$X_i^t = \frac{(X_i^t - a_i) \times (d_i - c_i^t)}{(d_i^t - a_i)} + c_i \quad (3.11)$$

$a_i$ i. değişkenin minimum rastgele yineleme yürüyüşü, $b_i$ i. değişkenin maksimum rastgele yineleme yürüyüşü, $c_i$ t. iterasyonda i. değişkenin minimumu, $d_i$ t. iterasyonda i. değişkenin maksimumunu belirtiyor.

Denklem (3.11) arama alanı içinde rastgele yürüyüşlerin meydana gelmesini sağlamak için her yinelemede uygulanmalıdır.

#### 3.2.2.2. Karınca aslanlarının çukurlarında tuzağa düşürme

Yukarıda da belirtildiği gibi rastgele karınca yürüyüşleri karınca aslanlarının tuzaklarından etkilenir. Bu varsayımı matematiksel olarak modellemek için aşağıdaki denklemler önerilmiştir:

$$c_i^t = Karınca\ aslanı_j^t + c^t \tag{3.12}$$

$$d_i^t = Karınca\ aslanı_j^t + d^t \tag{3.13}$$

$c^t$ t. iterasyonda bütün değişkenlerin minimunu, $d^t$ ise t. iterasyondaki bütün değişkenlerin maksimumunu içeren vektördür, $c_i^t$ i. karınca için bütün değişkenlerin minimumunu, $d_i^t$ i. karınca için bütün değişkenlerin maksimumunu ve $Karınca\ aslanı_j^t$ t. iterasyonda seçilen j. karınca aslanının pozisyonunu göstermektedir.

Denklem. (3.12) ve (3.13), karıncaların rastgele seçilmiş bir karınca aslanları etrafında c ve d vektörleri tarafından tanımlanan bir hiper kürede yürüdüğünü göstermektedir.

#### 3.3.2.3. Tuzağın inşası

Karınca aslanının avlanma tekniğini modellemek için bir rulet çarkı kullanılır. Karıncaların seçilen tek bir karınca aslanına yakalandıkları varsayılmaktadır. KAO algoritması optimizasyon sırasındaki zindeliğine bağlı bölgeleri seçmek için bir rulet tekerleği operatörü kullanır. Bu mekanizma karıncaların yakalanması için en güçlü karınca aslanına yüksek şans verir.

#### 3.3.2.4. Karınca aslanlarına doğru kayan karıncalar

Bugüne kadar önerilen mekanizmalarla karınca aslanları zindelikleriyle orantılı olarak tuzaklar kurabiliyorlar ve karıncaların rastgele hareket etmeleri gerekiyor. Ancak karınca aslanları bir karıncanın tuzağa düştüğünü anladıklarında çukurun merkezinden dışarıya

doğru kum atarlar. Bu davranış kaçmaya çalışan kapana kısılmış karıncanın aşağı kaymasına sebep oluyor. Bu davranışın matematiksel olarak modellenmesi için karıncaların rastgele yürüme hiper küresinin yarıçapı adaptif olarak azaltılır. Bu konuda aşağıdaki denklemler önerilmiştir:

$$c^t = \frac{c^t}{I} \tag{3.14}$$

$$d^t = \frac{d^t}{I} \tag{3.15}$$

Burada $I$ bir orandır, $c^t$ t. iterasyonda bütün değişkenlerin minimunu, $d^t$ ise t. iterasyondaki bütün değişkenlerin maksimumunu içeren vektördür.

(3.14) ve (3.15) denklemlerine göre $I = 10^w \frac{t}{T}$, burada t anlık iterasyon, T ise maksimum iterasyondur, w ise anlık iterasyona göre belirlenen 12/tir ($t > 0.1T, w = 2;\ t > 0.5T, w = 3;\ t > 0.75T, w = 4;\ t > 0.9T, w = 5; t > 0.95T, w = 6;$). Temel olarak w sabiti sömürünün doğruluk seviyesini ayarlayabilir. Bu denklemler, karıncaların konumlarını güncelleme yarıçapını küçültür ve karıncaların çukurların içindeki kayma işlemini taklit eder. Bu arama alanının kullanılmasını garanti eder.

#### 3.3.2.5. Avları yakalamak ve çukuru yeniden inşa etmek

Avın son aşaması, bir karınca çukurun dibine ulaştığında ve karınca aslanının çenesine yakalandığında gerçekleşir. Bu aşamadan sonra karınca aslanı karıncaları kumun içine çeker ve tüketir. Bu işlemi taklit etmek için avların karıncaların karşılık gelen karınca aslanlarına göre daha zayıf hale geldiğinde (kumun içine girdiğinde) avın gerçekleştiği kabul edilir. Daha sonra yeni bir av yakalama şansını arttırmak için avlanan karıncanın konumuna göre konumunu günceller. Bu konuda aşağıdaki denklem önerilmiştir:

$$Karınca\ aslanı_j^t = Karınca_i^t \text{ eğer } f(Karınca_i^t) > f(Karınca\ aslanı_j^t) \tag{3.16}$$

t anlık iterasyonu, $Karınca\ aslanı_j^t$ t. iterasyonda seçilmiş j. karınca aslanını ve $Karınca_i^t$ t. iterasyondaki i. karıncayı temsil etmektedir.

#### 3.3.2.6. Elitizm

Elitizm, optimizasyon sürecinin herhangi bir aşamasında elde edilen en iyi çözümü veya çözümleri sürdürmelerini sağlayan evrimsel algoritmaların önemli bir özelliğidir. Her bir

iterasyonda şimdiye kadar elde edilen en iyi karınca aslanları kaydedilir ve elit olarak kabul edilir. Seçkinler en uygun karınca aslanı olduğundan yinelemeler sırasında tüm karıncaların hareketlerini etkileyebilmelidir. Bu nedenle her karıncanın rulet tekerleği yöntemiyle seçilmiş bir karınca aslanı etrafında rastgele yürüdüğü varsayılmaktadır:

$$Karınca_i^t = \frac{R_{KA}^t + R_E^t}{2} \tag{3.17}$$

$R_{KA}^t$ t. iterasyonda rulet tekerleğiyle seçilmiş karınca aslanının etrafındaki rastgele yürüşü, $R_E^t$ t. iterasyonda elit etrafındaki rastgele yürüşü ve $Karınca_i^t$ t. iterasyondaki i. karıncanın konumunu temsil etmektedir.

### 3.3.3. Karınca Aslanı Optimizasyon algoritmasının kaba kodu

İlk karınca ve karınca aslanları popülasyonunu rastgele başlat.
Karıncaların ve karınca aslanlarının uygunluğunu hesapla.
En iyi karınca aslanlarını bul ve seçkin olarak kabul et.
***while*** son kriter karşılanmadığında
***for*** her karınca için
    Rulet tekerleğiyle karınca aslanı seç
    $c_i^t$ ve $d_i^t$`ni (3.12) ve (3.13)`a göre güncelle
    c ve d`ni (3.14) ve (3.15)`e göre güncelle
    Ratgele bir yürüş oluştur ve (3.5) ve (3.11) kullanarak normalleştir
    (3.17)`yi kullanarak karıncanın konumunu güncelle
***end for***
    Tüm karıncaların zindeliğini hesapla
    Bir karınca aslanını soluk hale gelirse uygun gelen karınca ile değiştir (Denk. (3.16))
    Eğer bir karınca aslanı seçkinlerden daha zinde olursa elitleri güncelle
***end while***
***Return*** elit

## 3.4. Balina Optimizasyon Algoritması (BOA)

Balina Optimizasyonu Algoritması (BOA) [3] 2016 yılında Mirjalili ve Lewis tarafından önerilmiştir. BOA üzerinde çeşitli hibritler yapılarak başarılı sonuçlar elde edilmiştir. [99,104-106] Sırasıyla algoritmanın esinlenmesi, matematiksel modeli ve kaba kodu anlatılacaktır.

### 3.4.1. Esinlenme

Balinalar, dünyanın en büyük memelilerinden biri olarak sınıflandırılır. Katil, sağ, kambur, finback, Sei, mavi balinalar ve Minke adında yedi ana balina türü vardır. Kambur balina balinalar arasında en büyüğüdür. Kambur balinalar kril ve küçük balıkları bulmak için harika bir av yöntemine sahiptir [3]. Kambur balinaların arama paternine, balinalar balıkların etrafında yukarı doğru spiral bir yüzme yolunda kabarcıklar oluşturduğu kabarcık ağı beslemesi [107] denir. BOA kambur balinaların bu davranışından esinlenerek geliştirilmiştir.

### 3.4.2. Matematiksel model

Kambur balinanın av kuşatma, kabarcıklı saldırı ve av arama davranışları modellenerek algoritma oluşturulmuştur. Bu modelleme üç aşamada anlatılacaktır.

#### 3.4.2.1. Av kuşatma

Kambur balinalar avın yerini tanıyabilir ve çevreleyebilir. Arama alanındaki optimal tasarımın konumu önceden bilinmediğinden, BOA algoritması mevcut en iyi aday çözümün hedef av olduğunu veya optimuma yakın olduğunu varsayar. En iyi arama ajanı tanımlandıktan sonra diğer arama ajanları bu yüzden konumlarını en iyi arama ajanına doğru güncellemeye çalışacaktır. Bu davranış aşağıdaki denklemlerle temsil edilir:

$$\vec{D} = \left|\vec{C}.\vec{X}^{*}(t) - \vec{X}(t)\right| \tag{3.18}$$

$$\vec{X}(t+1) = \vec{X}^{*}(t) - \vec{A}.\vec{D} \tag{3.19}$$

Mevcut yinelemenin $\boldsymbol{t}$ olduğu yerlerde $\vec{\boldsymbol{X}}^{*}$ şu ana kadar elde edilen en iyi kambur balinanın konumunu sunar. $\vec{\boldsymbol{A}}$ ve $\vec{\boldsymbol{C}}$ aşağıda gösterilen spesifik katsayılardır.

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \tag{3.20}$$

$$\vec{C} = 2.\vec{r} \tag{3.21}$$

$$a = 2 - m.((2)/M) \tag{3.22}$$

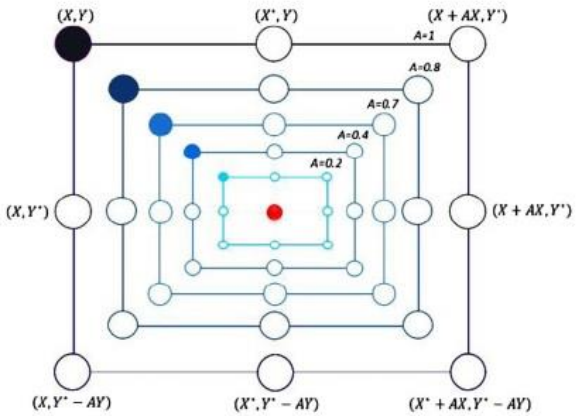
$\vec{\boldsymbol{a}}$ başlangıç değeri 2 olan bir parametredir. Bu parametre BOA'nın yinelemeleri boyunca doğrusal olarak 0'a düşer. $\boldsymbol{m}$ geçerli anlık iterasyonu ve $\boldsymbol{M}$ maksimum iterasyonu gösterir. Ek olarak $\vec{\boldsymbol{r}}$ düzgün bir dağılım kullanılarak oluşturulan 0 ile 1 arasında rastgele bir sayıdır. (3.18) ve (3.19) denklemleri kambur balinaların optimizasyon probleminin n boyutlu çözüm uzayını daha verimli olarak aramasını sağlar.

#### 3.4.2.2. Kabarcıklı saldırı (sömürü aşaması)

WOA'da kambur balinaların davranışını matematiksel olarak gösteren iki ana işlem olduğu düşünülmektedir. Bu iki işlem daralan çevreleme mekanizması ve konumun spiral güncellemesi olarak adlandırılır [99].

##### 3.4.2.2.1. Daralan çevreleme mekanizması

Önceki bölümde belirtildiği gibi, $\vec{\boldsymbol{a}}$'nın değeri yinelemelere göre doğrusal olarak azalır. $\vec{\boldsymbol{a}}$ parametresinin değerinde azalma, $\vec{\boldsymbol{A}}$'nın değerini azaltır. Bu, kambur balinaların konumlarını en iyi balina çevresinde herhangi bir yerde güncellemelerini sağlar. Şekil 3.4 en iyi kambur balinanın konumu (kırmızı nokta) etrafındaki konumunu güncellemeye çalışan bir kambur balinanın olası pozisyonlarını göstermektedir.




Şekil 3.4. Daralan çevreleme mekanizması

Kaynak:[99]

#### 3.4.2.2.2. Konumun spiral güncellenmesi

Kambur balina av için yukarı doğru spiral veya helis şeklinde yüzer. Kambur balinaların bu davranışını taklit etmek için logaritmik spiral fonksiyon kullanılır.

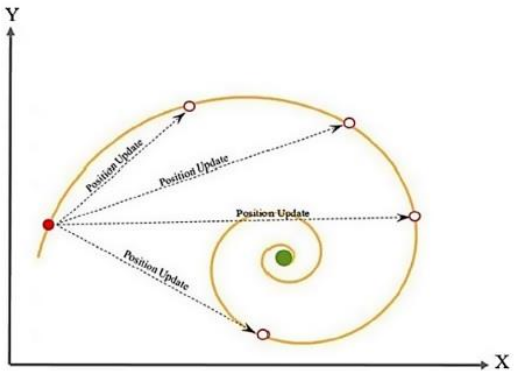
$$\vec{X}^*_{(t+1)} = \left|\overrightarrow{X^*_{(t)} - X_{(t)}}\right|.e^{bt}.\cos(2\pi l) + X^*_{(t)} \tag{3.23}$$

$$l = (a_2 - 1).rand + 1 \tag{3.24}$$

$$a_2 = -1 + m.((-1)/M) \tag{3.25}$$

***b*** parametresinin spiral fonksiyonun şeklini belirlediği ve ***l***'nin -1 ve 1 arasında rastgele oluşturulmuş bir sayı olduğu durumlarda Şekil 3.5 kambur balinaların spiral yüzme yamasını göstermektedir. *m* geçerli anlık iterasyonu ve *M* maksimum iterasyonu gösterir. Doğada kambur balinalar daralan çevreleme tekniğini ve spiral yüzmeyi aynı anda gerçekleştirir. Bu nedenle BOA her davranışı yüzde 50 olasılıkla kullanır. Başka bir deyişle her kambur balina en iyi kambur balinaya göre konumunu güncellemek için rastgele daralan çember veya spiral yüzmeyi seçebilir.

$$\vec{X}_{(t+1)} = \begin{cases} \vec{X}^*_{(t)} - \vec{A}.\overrightarrow{D} \ eğer & p < 0.5 \\ \vec{X}^*_{(t+1)} = \vec{D}.e^{bt}.\cos(2\pi l) + X^*_{(t)} \ eğer & p \geq 0.5 \end{cases} \tag{3.26}$$




Şekil 3.5. Konumun spiral güncellenmesi

Kaynak:[99]

#### 3.4.2.3. Av arama (çözüm alanını keşf etmek)

Sömürü aşamasında her balina konumunu doğrusal olarak azalan bir parametrenin (A) düşünüldüğü en iyi balinaya doğru günceller. Keşif aşamasında her kambur balina

rastgele seçilen balinaya göre konumunu günceller. Bu aşamada, eğer (A) parametresi -1 ve 1 arasında bir değer alırsa kambur balinanın bir sonraki pozisyonu mevcut pozisyonu ile rastgele seçilen bir kambur balinanın pozisyonu arasında olacaktır. BOA'nın keşif evresinin matematiksel formülasyonları (18) ve (19) denklemlerinde sunulmuştur.

$$\vec{D} = \left|\vec{C}.\vec{X}_{rand} - \vec{X}\right| \tag{3.27}$$

$$\vec{X}(t+1) = \vec{X}_{rand} - \vec{A}.\vec{D} \tag{3.28}$$

### 3.4.3. Balina Optimizasyon Algoritmasının kaba kodu

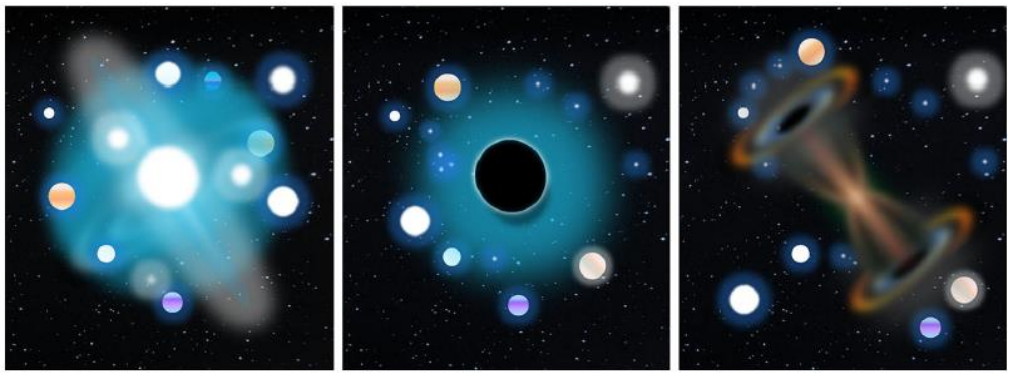
Balina popülasyonunu rastgele başlat.
Her bir arama temsilcisinin uygunluğunu hesapla.
En iyi arama temsilcisi = ***X**** .
***while*** son kriter karşılanmadığında
***for*** her arama temsilcisi için
    ***a***, ***A***, ***C***, ***l*** ve ***p***`yi güncelle
    ***if1(p<0.5)***
        ***if2(|A|<1)***
            o anki arama temsilcisininin konumunu denk. (3.18)`le güncelle
        ***else if2(|A≥ 1|)***
            rastgele arama temsilcisini seç (rand)
            o anki arama temsilcisininin konumunu denk. (3.28)`le güncelle
        ***end if2***
    ***else if1(p≥0.5)***
        o anki arama temsilcisininin konumunu denk. (3.23)`le güncelle
    ***end if1***
***end for***
    Herhangi bir arama temsilcisinin arama alanının ötesine geçip geçmediğini kontrol et ve değiştir
    Her bir arama temsilcisinin uygunluğunu hesapla
    Daha iyi bir çözüm varsa ***X**** güncelle
    ***t=t+1***
***end while***
***Return X****

## 3.5. Çoklu Evren Optimizasyonu (ÇEO)

Büyük patlama teorisi [108] evrenin büyük bir patlama ile başladığını kabul eder. Bu teoriye göre, büyük patlama bu dünyadaki her şeyin kökenidir ve bundan önce hiçbir şey yoktur. Çoklu evren teorisi, fizikçiler arasında yakın zamanda bilinen bir teoridir [109]. Bu teoride birden fazla büyük patlama olduğu ve her büyük patlamanın bir evrenin doğuşuna neden olduğuna inanılmaktadır. Çoklu evren terimi, içinde yaşanılan evrene ek olarak diğer evrenlerin varlığına atıfta bulunan evrenin karşıtıdır [109]. Çoklu evrenler etkileşir ve hatta çoklu evren teorisinde birbirleriyle çarpışabilir. Çoklu evren teorisi aynı zamanda her bir evrende farklı fiziksel yasalar olabileceğini öne sürer.

### 3.5.1. Esinlenme

ÇEO algoritması için ilham kaynağı olarak çoklu evren teorisinin üç ana kavramı seçilmiştir: beyaz delikler, kara delikler ve solucan delikleri. Evrende hiç beyaz bir delik görülmediği, ancak fizikçiler büyük patlamanın beyaz bir delik olarak değerlendirilebileceğini ve bir evrenin doğuşunun ana bileşeni olabileceğini düşünüyorlar [110]. Çoklu evren teorisinin [111] döngüsel modelinde, paralel evrenler arasındaki çarpışmaların gerçekleştiği yerde büyük patlamaların (beyaz deliklerin) yaratıldığı da tartışılmaktadır. Sıkça görülen kara delikler beyaz deliklerin aksine tamamen aksidirler. Işık huzmeleri dâhil her şeyi son derece yüksek çekim kuvveti ile çekiyorlar [112]. Solucan delikleri, bir evrenin farklı kısımlarını birbirine bağlayan deliklerdir. Çoklu evren teorisinde yer alan solucan delikleri nesnelerin bir evrenin herhangi bir köşesinde (hatta bir evrenden diğerine) anında seyahat edebildiği zaman uzay yolculuğu tünelleri gibi hareket eder [113]. Çoklu evren teorisinin bu üç temel bileşeninin kavramsal modelleri Şekil 3.6'da gösterilmektedir.



Şekil 3.6. Beyaz delik, kara delik ve solucan deliği

Kaynak: [114]

### 3.5.2. Matematiksel model

Beyaz / kara delik tünellerini matematiksel olarak modellemek ve evren nesnelerini değiştirmek için rulet tekerleği mekanizması kullanılır. Her yinelemede evrenler enflasyon oranlarına göre sıralanır ve bunlardan birine rulet tekerleği ile beyaz bir delik açması için seçilir. Bunu yapmak için aşağıdaki adımlar uygulanır. Farz edilmiştir ki:

$$U = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix}$$

$\boldsymbol{d}$ parametre sayısı (değişkenler) ve $\boldsymbol{n}$ evren sayısıdır (aday çözümler):

$$x_i^j = \begin{cases} x_k^j \; r_1 < NI(Ui) \\ x_i^j \; r_1 \geq NI(Ui) \end{cases} \tag{3.29}$$

$\boldsymbol{x_i^j}$ **i.** evrenin **j.** parametresini gösterir, ***Ui*** evren gösterir, ***NI(Ui)*** evrenin normalize edilmiş enflasyon oranıdır, $\boldsymbol{r_1}$ [0,1]'de rastgele bir sayıdır ve $\boldsymbol{x_k^j}$ **k.** evreninin **j.** parametresini gösterir ve bir rulet tekerleği seçme mekanizması tarafından seçilir.

Her evren için yerel değişiklikler sağlamak ve solucan deliklerini kullanarak enflasyon oranını iyileştirme ihtimalinin yüksek olması için solucan deliği tünellerinin her zaman bir evren ile şimdiye kadar oluşturulan en iyi evren arasında kurulduğu varsayılır. Bu mekanizmanın formülasyonu aşağıdaki gibidir:

$$x_i^j = \begin{cases} \begin{cases} X_j + TDR \times \left( \left( ub_j - lb_j \right) \times r_4 + lb_j \right) \; r_3 < 0.5 \\ X_j - TDR \times \left( \left( ub_j - lb_j \right) \times r_4 + lb_j \right) \; r_3 \geq 0.5 \end{cases} & r_2 < WEP \\ x_i^j & r_2 \geq WEP \end{cases} \tag{3.30}$$

$\boldsymbol{X_j}$ şu ana kadar oluşturulan en iyi evrenin **j.** parametresini gösterir, TDR bir katsayıdır, WEP başka bir katsayıdır, $\boldsymbol{lb_j}$ **j.** değişkeninin alt sınırını gösterir, $\boldsymbol{ub_j}$ **j.** değişkeninin üst sınırıdır, $\boldsymbol{x_i^j}$ **i.** evrenin **j.** parametresini gösterir ve $\boldsymbol{r_2}$, $\boldsymbol{r_3}$, $\boldsymbol{r_4}$ [0,1]'de rastgele sayılardır.

Buradaki iki ana katsayı olduğu matematiksel formülasyondan çıkarılabilir: solucan deliği varlık olasılığı (WEP) ve seyahat mesafesi oranı (TDR). Önceki katsayı solucan

deliğinin evrende var olma olasılığını tanımlamak içindir. En iyi duruma getirme sürecinin ilerlemesi olarak sömürüyü vurgulamak için yinelemelerde doğrusal olarak artması gerekir. Seyahat mesafesi oranı aynı zamanda, bir nesnenin şimdiye kadar elde edilen en iyi evren etrafındaki bir solucan deliği tarafından ışınlanabileceği mesafeyi (varyasyon) tanımlayan bir faktördür. WEP'in tersine, TDR, elde edilen en iyi evrende daha kesin bir sömürü / yerel arama yapabilmek için yinelemelerde artar. Her iki katsayı için adaptif formül aşağıdaki gibidir:

$$WEP = min + l \times (\frac{max-min}{L}) \tag{3.31}$$

min (minimum), maksimum (maksimum), ***l*** geçerli anlık iterasyonu ve ***L*** maksimum iterasyonu gösterir.

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{3.32}$$

burada ***p*** iterasyonlar üzerindeki sömürü doğruluğunu tanımlar. Ne kadar yüksek olursa o kadar çabuk ve doğru bir şekilde sömürü / yerel arama yapar.

### 3.5.3. Çoklu Evren Optimizasyonu algortimasının kaba kodu

```
for i tarafından indekslenen her evren
for j tarafından indekslenen her nesne
        r2=random([0,1]);
        if r2<WEP
                r3=random([0,1]);
                r4=random([0,1]);
                if r3<0.5
                        U(i,j)=En_iyi_evren(j) + TDR * (( ub(j) - lb(j)) *
                        r4 + lb(j));
                else
                        U(i,j)=En_iyi_evren(j) - TDR * (( ub(j) - lb(j)) *
                         r4 + lb(j));
                end if
        end if
        end for
end for
```

# BÖLÜM 4

## HİBRİT UYGULAMA GELİŞTİRME 1

### 4.1. Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyonu (HSKKAO)

Çalışmamızda KAO algoritmasını geliştirmek amaçlı SKA`nın özellikleri ile hibrit model geliştirilerek yeni bir algoritma elde edilmiştir. Bu hibritleşmeyi yaparken çeşitli SKA ile yapılan hibritler [99] gözden geçirilmiştir. Oluşturulan hibritin motivasyon kaynağı, yapılan hibritleşme ve kaba kodu aşağıda detaylı olarak anlatılacaktır.

### 4.2. Çalışmanın motivasyonu

Temel SKA sömürü aşamasında iyi bir performans sergiler. SKA'da, bir sinüs veya kosinüs işlevi 1'den büyük ve −1'den küçük bir değer döndürdüğünde keşif gerçekleşir. Ek olarak, konumunu güncellemeye çalışan mevcut bir çözüm, hedef noktadan (şu ana kadar elde edilen en iyi çözüm) uzak olamaz çünkü sinüs ve kosinüs aralığı [−2, 2] ile sınırlıdır. Bu nedenle SKA keşif aşamasında iyi performans gösteremez. Bu, yerel optimada takılma olasılığını arttırır. Bunun tersine, KAO dengeli keşif aramasına ve sömürü işletmecisine sahiptir. İkisinin hibriti yapılıp bu özellikleri daha da etkin hale getirilerek SKA ve KAO`dan daha iyi sonuçlar elde edilmiştir.

Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyonunun (HSKKAO) ana fikri, SKA ve KAO'nun her iki avantajına ve daha iyileştirilmiş yeni bir hibrit çözüm yöntemi geliştirmektir. Burada, KAO'nun karınca rastgele yürüş mekanizması yerine SKA'nın güncelleme operatörü eklenmiştir. Bu sayede hem keşif hem de sömürü arasındaki denge daha da iyileştirilmiştir.

### 4.3. Matematiksel model

Matematiksel model olarak KAO ile neredeyse aynı adımları tamamlasa da bazı fonksiyonlar değiştirilmiştir. Değiştirilmiş kısımlardan sonra kaba kod üzerinden tamamlanmış hali kısa olarak gösterilecektir.

KAO`da olan rastgele karınca yürüyüşlerini modellemek için kullanılan (3.12) ve (3.13) denklemleri SKA`da bulunan (3.1) ve (3.2) denklemleri haline getirilerek güncellenme şekli aşağıdaki yeni oluşturulmuş denklemlerle yapılmıştır:

$$c_i^t = c^t + r_1 \times \sin(r_2) \times r_3 \times Karınca\ aslanı_j^t \quad (4.1)$$

$$d_i^t = d^t + r_1 \times \cos(r_2) \times r_3 \times Karınca\ aslanı_j^t \quad (4.2)$$

$\boldsymbol{c^t}$ **t.** iterasyonda bütün değişkenlerin minimunu, $\boldsymbol{d^t}$ ise **t.** iterasyondaki bütün değişkenlerin maksimumunu içeren vektördür, $\boldsymbol{c_i^t}$ **i.** karınca için bütün değişkenlerin minimumunu, $\boldsymbol{d_i^t}$ **i.** karınca için bütün değişkenlerin maksimumunu ve $\boldsymbol{Karınca\ aslanı_j^t}$ **t.** iterasyonda seçilen **j.** karınca aslanının pozisyonunu gösterir**.** $\boldsymbol{r_1}$**,** $\boldsymbol{r_2}$**,** $\boldsymbol{r_3}$ rastgele sayılardır. $\boldsymbol{r_1}$ parametresi şimdiye kadar elde edilen en iyi çözüm ile çözüm arasındaki veya dışındaki boşlukta olan güncelleme yönünü belirler. $\boldsymbol{r_2}$ parametresi şimdiye kadar elde edilen en iyi çözüme göre güncelleme mesafesini belirler. $\boldsymbol{r_3}$ parametresi hedefin mesafeyi tanımlamadaki etkisini stokastik olarak vurgulamak **(**$\boldsymbol{r_3 > 1}$**)** veya önemini azaltmak **(**$\boldsymbol{r_3 < 1}$**)** için rastgele bir ağırlık getirir. Son olarak $\boldsymbol{r_4}$ parametresi (4.1) ve (4.2) denklemlerinde bulunan sinüs ve kosinüs bileşenleri arasında eşit şekilde geçiş yapar.

## 4.4. Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyon algoritmasının kaba kodu

İlk karınca ve karınca aslanları popülasyonunu rastgele başlat.

Karıncaların ve karınca aslanlarının uygunluğunu hesaplay.

En iyi karınca aslanlarını bulun ve seçkin olarak kabul et.

$r_1$ $r_2$, $r_3$ ve $r_4$`ü **güncelle.**

***while*** son kriter karşılanmadığında

***for*** her karınca için

Rulet tekerleğiyle karınca aslanı seç

***if*** $r_4$**<0.5**

$c_i^t$`ni (4.1)`e göre güncelle

***end***

***if*** $r_4 \geq$**0.5**

$d_i^t$`ni (4.2)`e göre güncelle

***end***

**c** ve **d**`ni (3.14) ve (3.15)`e göre güncelle

Ratgele bir yürüş oluştur ve (3.5) ve (3.11) kullanarak normalleştir

(3.17) kullanarak karıncanın konumunu güncelle

***end for***

Tüm karıncaların zindeliğini hesapla

Bir karınca aslanı soluk hale gelirse uygun gelen karınca ile değiştir (Denk. (3.16))

Eğer bir karınca aslanı seçkinlerden daha zinde olursa elitleri güncelle

***end while***

***Return*** elit

## 4.5. Hibrit Sinüs Kosinüs Karınca Aslanı Optimizasyon Performans Analizi

HSKKAO 23 Benchmark problemi kıyaslamasında Mirjalili`ye [1] ait KAO algoritması makalesinden KAO, PSO, MAD, YA, ÇTA, GuA, ABA, GA algoritmalarının aritmetik ortalama ve standart sapma sonuçları hazır olarak alınmış ve toplamda 9 algoritma ile kıyaslama yapılmıştır. Bu kıyaslamada boyutlar önceden belirtildiği şekilde her fonksiyonun karşısında tablolarda belirtilmiş durumdadır. Kıyaslanan makaledeki ile aynı parametreler ayarlanarak HSKKAO matlab üzerinden çalıştırılmıştır. Makaleye göre popülasyon sayısı tüm algoritmalarda 30, iterasyon ise 1000 üzeri şeklinde ifade edilmiştir. Çalışmada iterasyonun tam olarak ne kadar olduğu bilinmediği için bu 1050 olarak belirlenmiştir. HSKKAO, KAO ve SKA algoritmaları matlab üzerinden çalıştırıldı. SKA matlab kodu [115] Mirjali`ye ait olup kendi sitesinde kaynak şeklinde bulunmaktadır. F14-F23 Benchmark fonksiyonlarının kıyaslamasında ise Zhu vd. [116] gri kurt üzerine olan hibritleşme çalışmalarından PSO sonuçları alınarak kıyaslama yapılmıştır. Kıyaslama için HSKKAO, KAO ve SKA matlabda 23 Benchmark probleminin her birisi için 30 kere çalıştırılarak ortalaması elde edilmiştir. Sonuçlar Tablo 4.1 ve Tablo 4.2`de verilmiştir.

Sonuçlara bakıldığında HSKKAO algoritması F1-F4, F9 ve F11 fonksiyonlarında performansın üstü bir sonuç sergilemiştir. Aynı şekilde F7, F8, F10, F15 fonksiyonlarında iyileştirme yapmış ve diğer tüm algoritmalarla kıyasta en iyi sonuçları elde edilmiştir. Bunlar tabloda belirtilmiştir. F14 ve F19 fonksiyonlarında ise yine iyi sonuç sergileyerek ikinci olmuştur. Geri kalan problemlerde ise yaklaşık değerlerle ortalama bir performans sergilemiştir. Genel itibari ile HSKKAO iyileştirmede kendini kanıtlamıştır.

Tablo 4.1. HSKKAO algoritmasının tek modlu ve çok modlu Benchmark fonksiyon sonuçları ile kıyaslama

| M | F | $f_{min}$ | HSKKAO | | 1-KAO | | 2-SKA | | 3-PSO | | 4-MAD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts | ort | sts |
| TEK MODLU | F1 | 0 | 0 | 0 | 2.59E-10 | 1.65E-10 | 0.054556 | 0.181985 | 2.70E-09 | 1.00E-09 | 0.056987 | 0.014689 |
| | F2 | 0 | 0 | 0 | 1.84241E-06 | 6.58E-07 | 4.8425E-06 | 1.5831E-05 | 7.15E-05 | 2.26E-05 | 0.006848 | 0.001577 |
| | F3 | 0 | 0 | 0 | 6.06847E-10 | 6.34E-10 | 3071.3461 | 2817.8356 | 4.71E-06 | 1.49E-06 | 0.959865 | 0.82345 |
| | F4 | 0 | 0 | 0 | 1.36061E-08 | 1.81E-09 | 17.317533 | 9.851325 | 3.25E-07 | 1.02E-08 | 0.276594 | 0.005738 |
| | F5 | 0 | 11.215807 | 5.564887 | 0.346772393 | 0.109584 | 941.5811 | 3017.459 | 0.123401 | 0.216251 | 0.085348 | 0.140149 |
| | F6 | 0 | 0.072589 | 0.042064 | 2.56183E-10 | 1.09E-10 | 4.62801 | 0.508374 | 5.23E-07 | 2.74E-06 | 0.125323 | 0.084998 |
| | F7 | 0 | 5.91495E-05 | 4.04358E-05 | 0.004292492 | 0.005089 | 0.029997 | 0.026497 | 0.001398 | 0.001269 | 0.000304 | 0.000258 |
| ÇOK MODLU | F8 | -418.9829×B | -9842.76142 | 1414.29963 | -1606.27643 | 314.4302 | -4072.1282 | 335.8172 | -1367.01 | 146.4089 | -4.20735 | 9.36E-16 |
| | F9 | 0 | 0 | 0 | 7.71411E-06 | 8.45E-06 | 16.165658 | 30.671868 | 0.278588 | 0.218991 | 1.32512 | 0.326239 |
| | F10 | 0 | 8.8818E-16 | 4.01173E-31 | 3.73035E-15 | 1.5E-15 | 13.58369 | 8.659423 | 1.11E-09 | 2.39E-11 | 8.88E-06 | 8.56E-09 |
| | F11 | 0 | 0 | 0 | 0.018604494 | 0.009545 | 0.234076 | 0.256274 | 0.273674 | 0.204348 | 0.70609 | 0.907954 |
| | F12 | 0 | 0.000665 | 0.000485 | 9.74645E-12 | 9.33E-12 | 1.702542 | 4.158149 | 9.42E-09 | 2.31E-10 | 0.12334 | 0.040898 |
| | F13 | 0 | 0.010241 | 0.005265 | 2.00222E-11 | 1.13E-11 | 235.3133 | 1182.5945 | 1.35E-07 | 2.88E-08 | 1.35E-02 | 2.88E-04 |

| M | F | $f_{min}$ | 5-YA | | 6-YAA | | 7-GuA | | 8-ABA | | 9-GA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts | ort | sts |
| TEK MODLU | F1 | 0 | 0.773622 | 0.528134 | 1.06346E-07 | 1.27E-07 | 6.50E-03 | 2.05E-04 | 0.039615 | 0.01449 | 0.118842 | 0.125606 |
| | F2 | 0 | 0.334583 | 3.816022 | 0.000624246 | 0.000176 | 2.12E-01 | 3.98E-02 | 0.050346 | 0.012348 | 0.145224 | 0.053227 |
| | F3 | 0 | 0.115303 | 0.766036 | 5.6682E-08 | 3.9E-08 | 2.47E-01 | 2.14E-02 | 0.049273 | 0.019409 | 0.13902 | 0.121161 |
| | F4 | 0 | 0.192185 | 0.890266 | 0.003837885 | 0.002186 | 1.12E-05 | 8.25E-06 | 0.145513 | 0.031171 | 0.157951 | 0.862029 |
| | F5 | 0 | 0.334077 | 0.300037 | 0.781200043 | 0.366891 | 0.007197 | 0.007222 | 2.175892 | 1.447251 | 0.714157 | 0.972711 |
| | F6 | 0 | 0.778849 | 0.67392 | 1.08459E-07 | 1.25E-07 | 5.95E-05 | 1.08E-06 | 0.05873 | 0.014477 | 0.167918 | 0.868638 |
| | F7 | 0 | 0.137483 | 0.112671 | 0.003105276 | 0.001367 | 0.001321 | 0.000728 | 0.000853 | 0.000504 | 0.010073 | 0.003263 |
| ÇOK MODLU | F8 | -418.9829×B | -1065.88 | 858.498 | -1842.42621 | 50.42824 | -2094.91 | 0.007616 | -1245.59 | 353.2667 | -2091.64 | 2.47235 |
| | F9 | 0 | 1.233748 | 0.686447 | 0.273294621 | 0.068583 | 0.127328 | 0.002655 | 0.263458 | 0.182824 | 0.659271 | 0.815751 |
| | F10 | 0 | 0.129359 | 0.043251 | 0.007398721 | 0.007096 | 8.16E-09 | 1.63E-08 | 0.168306 | 0.050796 | 0.956111 | 0.807701 |
| | F11 | 0 | 1.451575 | 0.570309 | 0.085021659 | 0.040046 | 0.122678 | 0.049673 | 0.099815 | 0.024466 | 0.487809 | 0.217782 |
| | F12 | 0 | 0.395977 | 0.993325 | 0.000265711 | 0.000553 | 5.60E-09 | 1.58E-10 | 0.126076 | 0.263201 | 0.110769 | 0.002152 |
| | F13 | 0 | 3.86631E-01 | 0.121986 | 3.67E-06 | 3.51E-06 | 4.88E-06 | 6.09E-07 | 2.13E-03 | 0.001238 | 1.29E-01 | 0.068851 |

en iyi sonuç

ortalama sonuç

Tablo 4.2. HSKKAO algoritmasının sabit boyutlu çok modlu Benchmark fonksiyon sonuçları ile kıyaslama

| M | F | $f_{min}$ | HSKKAO | | 1-KAO | | 2-SKA | | 3-PSO | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts |
| SABİT BOYUTLU ÇOK MODLU | F14 | 1 | 2.125267667 | 1.233137242 | 2.416083 | 1.671098 | 1.925526 | 1.005242 | 3.5921 | 2.7273 |
| | F15 | 0.00030 | 0.000350748 | 3.16437E-05 | 0.002314 | 0.004919 | 0.001002 | 0.00041 | 0.00093 | 0.000252 |
| | F16 | -1.0316 | -1.03086 | 0.000616218 | -1.0316 | 6.7752E-16 | -1.03156 | 5.6832E-05 | -1.0316 | 6.4099E-16 |
| | F17 | 0.398 | 0.400891 | 0.003454392 | 0.39789 | 1.6938E-16 | 0.399486 | 0.001396 | 0.39789 | 0 |
| | F18 | 3 | 3.007433333 | 0.009568819 | 3 | 0 | 3.000093 | 0.000207 | 3 | 2.4659E-15 |
| | F19 | -3.86 | -0.30048 | 1.1292E-16 | -0.30048 | 1.1292E-16 | -0.30048 | 1.1292E-16 | -3.8628 | 2.6373E-15 |
| | F20 | -3.32 | -3.20209333 | 0.091319073 | -3.266213 | 0.06066 | -2.69072 | 1.114497 | -3.2586 | 0.059314 |
| | F21 | -10.1532 | -5.05279333 | 0.477902534 | -5.45086 | 2.848357 | -2.874981 | 2.004471 | -5.9587 | 3.1213 |
| | F22 | -10.4028 | -5.13214333 | 0.653032605 | -7.356093 | 3.392104 | -3.160536 | 1.548684 | -8.4055 | 2.8918 |
| | F23 | -10.5363 | -5.02233667 | 0.188030129 | -7.51072 | 3.546232 | -3.552445 | 1.267745 | -9.3733 | 2.3429 |

en iyi sonuç

ikinci en iyi sonuç

ortalama sonuç

# BÖLÜM 5

## HİBRİT UYGULAMA GELİŞTİRME 2

### 5.1 Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu (ÇESKBO) hibrit algoritması modeli

Literatüre önceden de kazandırılmış olan SKBOA, BOA`nın iyi arama özelliğini ve SKO`nun iyi sömürü özelliğini ele alarak her ikisini dengeleyen bir hibrit modeldir. Bu çalışmada Çoklu Evren Optimizasyonu özellikleri kullanılarak mevcut hibrit model üzerinde yeni bir iyileştirme yapılarak üçlü bir hibrit geliştirilmiştir. Modelin motivasyon kaynağı, matematiksel modeli ve kaba kodu sırasıyla aşağıda verilmiştir.

### 5.2. Çalışmanın motivasyonu

Önceden literatürde bulunan SKA ve BOA hibriti olan SKBOA [91] hem arama hem de sömürü özelliklerini dengelemeyi başarmış algoritmadır. Çalışmada ÇEO`nun bazı özellikleri SKBOA`da bulunan özelliklerle değiştirilmiştir. Çalışmada iki özellikten faydalanılmıştır. Bunlar çoklu evren algoritmasında bulunan solucan deliği varlık olasılığı (WEP) ve seyahat mesafesi oranı (TDR) özelliğidir. WEP SKA`da bulunan güncelleme yönünü belirleyen $\boldsymbol{r_1}$ parametresinin, TDR ise BOA`da bulunan elemanlar arası çarpma parametresi olan $\boldsymbol{l}$`nin güncellemesinde kullanılan $\boldsymbol{a_2}$(-1 ile -2 arasında değişir) yerine kullanılmıştır. Yeni geliştirilen hibrit modelden elde edilen sonuçlar SKBOA, BOA , SKA ve ÇEO`dan daha iyi sonuçlar sergileyerek arama ve sömürü özelliklerini daha etkin hale getirdiğini kanıtlamıştır.

### 5.3. Matematiksel modeli

Oluşturulan modelde SKA`nın güncellenme operatörünü BOA`nın daralan çevreleme mekanizmasıyla değiştirildiğinde ortaya çıkan fonksiyon aşağıda belirtilmiştir.

$$X_{\text{İ}}^{t+1} = \begin{cases} X_{\text{İ}}^{t} + r_1 \times \sin(r_2) \times \left| r_3 X_{(t)}^{*} - X_{\text{İ}}^{t} \right|, & q < 0.5 \\ X_{\text{İ}}^{t} + r_1 \times \cos(r_2) \times \left| r_3 X_{(t)}^{*} - X_{\text{İ}}^{t} \right|, & q \geq 0.5 \end{cases} \quad (5.1)$$

Yinelemeler üzerinden çalışan $\boldsymbol{r_1}$ ve $\boldsymbol{a_2}$ aynı şekilde yinelemeler üzerinden çalışan WEP ve TDR`le değiştirilmiştir. Bunlar daha net şekilde kaba kod üzerinden anlaşılmaktadır. Adımlarda kullanılan sıra bunu daha basit şekle getirmiştir.

## 5.4. Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu algoritmasının kaba kodu

Popülasyonunu rastgele başlat.
Her bir arama temsilcisinin uygunluğunu hesapla.
En iyi arama temsilcisi = ***X**** .
***while*** son kriter karşılanmadığında
***for*** her arama temsilcisi için
    ***A***, ***l, p, TDR*(3.31), *WEP*(3.32), *r2, r3 ve r4***`ü güncelle
    ***if1(p<0.5)***
        ***if2(r4<0.5)***
            o anki arama temsilcisininin konumunu denk. (5.1)`le güncelle
        ***else if2(r4≥ 0.5)***
            rastgele arama temsilcisini seç (rand)
            o anki arama temsilcisininin konumunu denk. (5.1)`le güncelle
        ***end if2***
    ***else if1(p≥0.5)***
        o anki arama temsilcisininin konumunu denk. (3.23)`le güncelle
    ***end if1***
***end for***
    Herhangi bir arama temsilcisinin arama alanının ötesine geçip geçmediğini kontrol et ve değiştir
    Her bir arama temsilcisinin uygunluğunu hesapla
    Daha iyi bir çözüm varsa ***X**** güncelle
    ***t=t+1***
***end while***

## 5.5. Çok Evrenli Sinüs Kosinüs Balina Optimizasyonu Performans Analizi

ÇESKBO deneyinde popülasyon sayısı 30, iterasyon sayısı 500, 23 Benchmark fonksiyonu boyutları ise önceden de Benchmark fonksiyon tablolarında belirtildiği şekilde ayarlanarak uyarlanmıştır. Her bir fonksiyon 30 kere çalıştırılıp elde edilen değerlerin ortalaması ve standart sapması hesaplanmıştır. Kıyaslama zamanı ÇESKBO, SKBOA, SKA, ÇEO algoritmaları çalıştırılmıştır. ÇEO matlab koduna Mirjalili`nin açık kaynak kodlarından ulaşılabilir. Kıyaslama zamanı aynı parametrelerle çalıştırılmış Mirjalili`nin [3] BOA ile ilgili makalesinden BOA, PSO, YAA ve HEP algoritmalarının sonuçları alınarak toplamda 7 algoritmayla kıyaslama yapılmıştır. Sonuçlar Tablo 5.1 ve Tablo 5.2`de gösterilmiştir.

ÇESKBO algoritmasının sonuçlarına bakıldığında F9 performans üstü, F1-F4, F7, F10, F11 fonksiyonlarında hibritleşme sayesinde iyileştirmenin çok iyi olduğu gözlemlenmiştir. Bunlardan F1-F3, F9-F10 kıyaslamada tüm algoritmalardan daha iyi sonuç verirken F7, F11, F21 kıyaslamada ikincilik göstermiştir. F5, F8, F12, F15-F22, F23 problemlerinde ise ortalama ve yaklaşık sonuçlar sergilemiştir. Genel olarak değerlendirdiğimizde ortalama bir iyileştirme yapılmıştır.

Tablo 5.1. ÇESKBO algoritmasının tek modlu ve çok modlu Benchmark fonksiyon sonuçları ile kıyaslama

| M | F | $f_{min}$ | ÇESKBO | | 1-SKBOA | | 2-BOA | | 3-SKA | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts |
| TEK MODLU | F1 | 0 | 3.6035E-65 | 1.887705 | 2.5662E-30 | 8.7236E-30 | 1.41E -30 | 4.91E -30 | 17.454462 | 23.343298 |
| | F2 | 0 | 1.9681E-34 | 5.3152E-34 | 1.2455E-16 | 2.3789E-16 | 1.06E -21 | 2.39E -21 | 0.018628 | 0.028121 |
| | F3 | 0 | 1.6432E-55 | 7.149E-55 | 4.8624E-27 | 1.9813E-26 | 5.39E -07 | 2.93E -06 | 10016.544 | 5884.8338 |
| | F4 | 0 | 1.6259E-32 | 7.1337E-32 | 7.3999E-16 | 1.424E-15 | 0.072581 | 0.39747 | 32.825737 | 8.726484 |
| | F5 | 0 | 27.91003 | 0.69361 | 27.2007 | 4.284303 | 27.86558 | 0.763626 | 48385.494 | 121848.76 |
| | F6 | 0 | 2.569737 | 0.635213 | 0.232776 | 0.130224 | 3.116266 | 0.532429 | 19.44915 | 23.02077 |
| | F7 | 0 | 0.000333 | 0.000334 | 0.000316 | 0.000218 | 0.001425 | 0.001149 | 0.09472 | 0.093542 |
| ÇOK MODLU | F8 | -418.9829×B | -5167.9574 | 493.98085 | -11515.5 | 1528.236 | -5080.76 | 695.7968 | -3712.787 | 241.8149 |
| | F9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 349.87198 | 1736.1781 |
| | F10 | 0 | 8.8818E-16 | 4.0117E-31 | 8.3489E-15 | 3.2787E-15 | 7.4043 | 9.897572 | 15.77695 | 7.96786 |
| | F11 | 0 | 4.996E-16 | 2.7364E-15 | 0 | 0 | 0.000289 | 0.001586 | 1.049631 | 0.340067 |
| | F12 | 0 | 0.191217 | 0.05168 | 0.010909 | 0.005131 | 0.339676 | 0.214864 | 548.78755 | 1452.7774 |
| | F13 | 0 | 1.934423 | 0.423003 | 0.1918 | 0.112834 | 1.889015 | 0.266088 | 69343.16 | 168080.93 |

| M | F | $f_{min}$ | 4-ÇEO | | 5-PSO | | 6-YAA | | 7-HEP | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts |
| TEK MODLU | F1 | 0 | 1.251655 | 0.417805 | 0.000136 | 0.000202 | 2.53E-16 | 9.67E-17 | 0.00057 | 0.00013 |
| | F2 | 0 | 9.913015 | 30.80453 | 0.042144 | 0.045421 | 0.055655 | 0.194074 | 0.0081 | 0.00077 |
| | F3 | 0 | 245.75308 | 106.89063 | 70.12562 | 22.11924 | 896.5347 | 318.9559 | 0.016 | 0.014 |
| | F4 | 0 | 1.782515 | 0.57194 | 1.086481 | 0.317039 | 7.35487 | 1.741452 | 0.3 | 0.5 |
| | F5 | 0 | 298.2843 | 446.541 | 96.71832 | 60.11559 | 67.54309 | 62.22534 | 5.06 | 5.87 |
| | F6 | 0 | 1.256098 | 0.325769 | 0.000102 | 8.28E -05 | 2.5E-16 | 1.74E-16 | 0 | 0 |
| | F7 | 0 | 0.037924 | 0.011749 | 0.122854 | 0.044957 | 0.089441 | 0.04339 | 0.1415 | 0.3522 |
| ÇOK MODLU | F8 | -418.9829×B | -7844.6521 | 486.6885 | -4841.29 | 1152.814 | -2821.07 | 493.0375 | -12554.5 | 52.6 |
| | F9 | 0 | 123.0991 | 28.65995 | 46 .70423 | 11 .62938 | 25 .96841 | 7 .470068 | 0.046 | 0.012 |
| | F10 | 0 | 1.951047 | 0.457449 | 0 .276015 | 0 .50901 | 0 .062087 | 0 .23628 | 0.018 | 0.0021 |
| | F11 | 0 | 0.853236 | 0.085029 | 0 .009215 | 0 .007724 | 27 .70154 | 5 .040343 | 0.016 | 0.022 |
| | F12 | 0 | 1.867438 | 1.287785 | 0 .006917 | 0 .026301 | 1 .799617 | 0 .95114 | 9.2E-06 | 3.6E-06 |
| | F13 | 0 | 0.213151 | 0.125645 | 0 .006675 | 0 .008907 | 8 .899084 | 7 .126241 | 0.00016 | 0.000073 |

en iyi sonuç

ikinci en iyi sonuç

ortalama sonuç

Tablo 5.2. ÇESKBO algoritmasının sabit boyutlu çok modlu Benchmark fonksiyon sonuçları ile kıyaslama

| M | F | $f_{min}$ | ÇESKBO | | 1-SKBOA | | 2-BOA | | 3-SKA | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts |
| SABİT BOYUTLU ÇOK MODLU | F14 | 1 | 5.718113 | 4.938023 | 1.626363 | 0.920508 | 2.111973 | 2 .498594 | 1.925526 | 1.005242 |
| | F15 | 0.00030 | 0.000593 | 0.00042 | 0.000674 | 0.000376 | 0.000572 | 0.000324 | 0.001002 | 0.00041 |
| | F16 | -1.0316 | -1.0316 | 6.7752E-16 | -1.0316 | 6.7752E-16 | -1.03163 | 4.2E-07 | -1.03156 | 5.6832E-05 |
| | F17 | 0.398 | 0.397949 | 0.000109 | 0.397968 | 0.00014 | 0.397914 | 2.7E-05 | 0.399486 | 0.001396 |
| | F18 | 3 | 3.000453 | 0.000696 | 3.00057 | 0.001288 | 3 | 4.22E-15 | 3.000093 | 0.000207 |
| | F19 | -3.86 | -0.30048 | 1.1292E-16 | -0.30048 | 1.1292E-16 | -3.85616 | 0.002706 | -0.30048 | 1.1292E-16 |
| | F20 | -3.32 | -3.23337 | 0.093944 | -3.21142 | 0.106658 | -2.98105 | 0.376653 | -2.69072 | 1.114497 |
| | F21 | -10.1532 | -8.22838 | 2.634873 | -8.41452 | 2.650293 | -7.04918 | 3.629551 | -2.874981 | 2.004471 |
| | F22 | -10.4028 | -8.34166 | 2.780134 | -8.8921 | 2.336948 | -8.18178 | 3.829202 | -3.160536 | 1.548684 |
| | F23 | -10.5363 | -8.677407 | 2.553253 | -9.51277 | 2.000822 | -9.34238 | 2.414737 | -3.552445 | 1.267745 |

| M | F | $f_{min}$ | 4-ÇEO | | 5-PSO | | 6-YAA | | 7-HEP | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ort | sts | ort | sts | ort | sts | ort | sts |
| SABİT BOYUTLU ÇOK MODLU | F14 | 1 | 0.998 | 4.5168E-16 | 3.627168 | 2.560828 | 5.859838 | 3.831299 | 1.22 | 0.56 |
| | F15 | 0.00030 | 0.004801 | 0.007922 | 0.000577 | 0.000222 | 0.003673 | 0.001647 | 0.0005 | 0.00032 |
| | F16 | -1.0316 | -1.0316 | 6.7752E-16 | -1.03163 | 6.25E-16 | -1.03163 | 4.88E-16 | -1.03 | 4.9E-07 |
| | F17 | 0.398 | 0.39789 | 1.6938E-16 | 0.397887 | 0 | 0.397887 | 0 | 0.398 | 1.5E-07 |
| | F18 | 3 | 11.1 | 24.71542 | 3 | 1.33E-15 | 3 | 4.17E-15 | 3.02 | 0.11 |
| | F19 | -3.86 | -0.30048 | 1.1292E-16 | -3.86278 | 2.58E-15 | -3.86278 | 2.29E-15 | -3.86 | 0.000014 |
| | F20 | -3.32 | -3.27712 | 0.060031 | -3.26634 | 0.060516 | -3.31778 | 0.023081 | -3.27 | 0.059 |
| | F21 | -10.1532 | -6.70704 | 2.98929 | -6.8651 | 3.019644 | -5.95512 | 3.737079 | -5.52 | 1.59 |
| | F22 | -10.4028 | -8.570013 | 3.160997 | -8.45653 | 3.087094 | -9.68447 | 2.014088 | -5.53 | 2.12 |
| | F23 | -10.5363 | -9.12424 | 2.65473 | -9.95291 | 1.782786 | -10.5364 | 2.6E-15 | -6.57 | 3.14 |

en iyi sonuç

ikinci en iyi sonuç

ortalama sonuç

# 6. BÖLÜM
# SONUÇ, TARTIŞMA VE ÖNERİLER

## 6.1. Sonuç

Bu çalışmada geliştirilen 2 adet hibrit algoritmanın test amaçlı 23 Benchmark fonksiyonuna uyarlanarak performans analizleri değerlendirilmiştir. Kullanılan test fonksiyonları özellikleri açısından birbirinden farklılıklar göstermekte olup geniş kapsamlı bir değerlendirme elde ediliyor. Çalışmada hibritleştirmede kullanılan algoritmalar ve tezde kıyaslanmada kullanılan bazı popüler optimizasyon algoritmaların sonuçları ile karşılaştırıldığında yeni hibritlerin oldukça başarılı oldukları gözlenmiştir.

Seçilen parametre ayarları problemleri daha zorlu yaparak hibrit algoritmaların performanslarını daha rahat sergilemelerini sağlamaktadır. Bu genel olarak n boyutlu Benchmark fonksiyonlarında boyutun 30 olarak seçilmesi ile değerlendirilebilir. Boyutlar, arama alanı, popülasyon büyüklüğü, iterasyon (yineleme) sayısı çeşitli araştırmalar ve seçilen makalelere göre belirlenmiştir. Algoritmalar rastgelelik mantığıyla çalıştığı için yapılan çalışmalara dayanarak her fonksiyon için 30 kere kod çalıştırılarak ortalamaları alınmıştır. Böylece sonuç daha belirgin hale gelmektedir.

SKA, KAO, BOA, ÇEO algoritmalarının her biri kendi optimizasyon sürecine göre sonuçlara etki etmektedir. SKA sömürüde iyi olsa da keşifte iyi değildir. BOA ise tam tersidir. KAO ise sömürü ve keşif arasındaki dengeyi sağlamaktadır. ÇEO hem sömürü hem keşif özelliklerine sahiptir. Tez çalımasında geliştirilen hibritler de bu özellikle birleştirilerek daha iyi sonuç alınması ya da iyileştirilmesi amaçlanmış ve oldukça başarılı ve rekabetçi sonuçlar ürettiği görülmüştür.

HSKKAO algoritması, SKA`nın sömürü aşamasında sergilediği iyi performans özelliğini ve KAO`nun hem sömürü hem de keşif arasında sağladığı denge özelliklerini birleştirmiştir. Bu sayede hibrit algoritma ile performans üstü sonuçlar elde edilmiştir.

SCBOA hibriti sömürü ve keşifi dengelese de ÇEO`nun solucan deliği varlık olasılığı (WEP) ve seyahat mesafesi oranı (TDR) ile modifiye edilerek oluşturulan ÇESKBO hibrit algoritması bu dengeyi daha da iyileştirerek başarı elde etmiştir.

Yapılan hibritleştirmeler sayesinde yeni oluşan HSKKAO ve ÇESKBO algoritmalarının performansları kıyaslanmıştır. HSKKAO 23 test probleminden 12`sinde, ÇESKBO ise 9`unda başarılı olmuştur. Test sonuçlarına göre her iki yeni algoritma iyileştirme elde etmiştir.

## 6.2. Tartışma ve Öneriler

Gelecek çalışmalarda yapılan deneyler farklı alanlarda örneğin Gezgin Satıcı Problemi (GSP) gibi test problemlerine, gerçek hayat problemlerine ve mühendislik problemlerine uyarlanarak hibrit algoritmaların performansı daha geniş sahada incelenebilir.

Hibritleştirmeler sömürü ve keşif aşamaları açısından farklı testler halinde de yapılabilir. Sömürü aşaması iyi olup keşif aşaması iyi olmayan bir algoritmayla tam tersi bir algoritmanın hibritleri yapılarak daha iyi sonuçlar elde edilebilir.

# KAYNAKLAR


[1] Mirjalili, S. (2015). The ant lion optimizer. **Advances in Engineering Software**, 83, 80-98.

[2] Jamil, M., & Yang, X. S. (2013). A literature survey of Benchmark functions for global optimization problems. arXiv preprint arXiv:1308.4008.

[3] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. **Advances in engineering software**, 95, 51-67.

[4] Winston, P.H. (1992) Artificial Intelligence, 3rd ed., Addison-Wesley, Boston, MA, USA.

[5] Chung, C.J. and Reynolds, R.G. (1998) 'CAEP: an evolution-based tool for real-valued function optimization using cultural algorithms', **International Journal on Artificial Intelligence Tool, Vol. 7**, No. 3, pp.239–291.

[6] Salomon, R. (1996) 'Re-evaluating genetic algorithm performance under corodinate rotation of Benchmark functions: a survey of some theoretical and practical aspects of genetic algorithms'**, BioSystems, Vol. 39**, No. 3, pp.263–278.

[7] Yao, X. and Liu, Y. (1996) 'Fast evolutionary programming', Proc. 5th Conf. on Evolutionary Programming.

[8] Schumer, M.A. and Steiglitz, K. (1968) 'Adaptive step size random search', **IEEE Transactions on Automatic Control, Vol. 13**, No. 3, pp.270–276.

[9] **http://benchmarkfcns.xyz/benchmarkfcns/spherefcn.html** (Haziran 2019)

[10] H. P. Schwefel, "Numerical Optimization for Computer Models," John Wiley Sons,1981.

[11] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or least Value of a Function," **Computer Journal, vol. 3**, no. 3, pp. 175-184, 1960. [Available Online]: **http://comjnl.oxfordjournals.org/content/3/3/175.full.pdf**

[12] http://benchmarkfcns.xyz/benchmarkfcns/rosenbrockfcn.html(Haziran 2019)

[13] T. B¨ack, H. P. Schwefel, "An Overview of Evolutionary Algorithm for Parameter Optimization," **Evolutionary Computation, vol. 1**, no. 1, pp. 1-23, 1993.

[14] R. Storn, K. Price, "Differntial Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," Technical

Report no. TR-95-012, International Computer Science Institute, Berkeley, CA, 1996. [Available Online] **http://www1.icsi.berkeley.edu/~storn/TR-95-012.pdf**

[15] **http://benchmarkfcns.xyz/benchmarkfcns/ackleyfcn.html**(Haziran 2019)

[16] A. O. Griewank, "Generalized Descent for Global Optimization," **Journal of Optimization Theory and Applications, vol. 34,** no. 1, pp. 11-39, 1981.

[17] Civicioglu, P., & Besdok, E. (2014). Comparative analysis of the cuckoo search algorithm. In Cuckoo Search and Firefly Algorithm (pp. 85-113). Springer, Cham.

[18] Montiel, O., Castillo, O., Melin, P., & Sepúlveda, R. (2007). Improving the Human Evolutionary Model: An intelligent optimization method. In International Mathematical Forum (**Vol. 2**, No. 1, pp. 21-44).

[19] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. **Advances in engineering software**, 69, 46-61.

[20] F. H. Branin Jr., "Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations," **IBM Journal of Research and Development, vol. 16,** no. 5, pp. 504-522, 1972.

[21] A. Goldstein, J. F. Price, "On Descent from Local Minima," **Mathematics and Comptutaion, vol. 25**, no. 115, pp. 569-574, 1971.

[22] J. K. Hartman, "Some Experiments in Global Optimization," [Available Online]:**http://ia701505.us.archive.org/9/items/someexperimentsi00hart/someexperimentsi00hart.pdf**

[23] J. Opaˇci´c, "A Heuristic Method for Finding Most extrema of a Nonlinear Functional," **IEEE Transactions on Systems, Man and Cybernetics, vol. 3**, no. 1, pp. 102-107, 1973.

[24] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: a survey. **Appl Soft Comput** 2011;11:4135–51.

[25] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. **Inform Sci** 2013;237:82–117.

[26] Gogna A, Tayal A. Metaheuristics: review and application. **J Exp Theor Artif Intell** 2013;25:503–26.

[27] Olorunda O , Engelbrecht AP . Measuring exploration/exploitation in particle swarms using swarm diversity. In: Proceedings of the 2008 IEEE congress on evolutionary computation, CEC (IEEE world congress on computational intel- ligence); 2008. p. 1128–34 .

[28] Alba E , Dorronsoro B . The exploration/exploitation tradeoffin dynamic cellu- lar genetic algorithms. **IEEE Trans Evol Comput** 2005;9:126–42 .

[29] Lin L , Gen M . Auto-tuning strategy for evolutionary algorithms: balancing be- tween exploration and exploitation. **Soft Comput** 2009;13:157–68

[30] Črepinšek, M., Liu, S. H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. **ACM Computing Surveys (CSUR)**, 45(3), 35.

[31] Kirkpatrick S, Jr. DG, Vecchi MP. Optimization by simulated annealing. **Science, vol. 220**; 1983. p. 671–80.

[32] Holland JH . Genetic algorithms. **Sci Am** 1992;267:66–72 .

[33] J.R. Koza, "Genetic programming,"1992.

[34] **http://www.cleveralgorithms.com/nature-inspired/evolution/evolution_strategies.html** (Haziran 2019)

[35] Simon D . Biogeography-based optimization. **IEEE Trans Evol Comput** 2008;12:702–13 .

[36] L. J. Fogel, "Autonomous automata", **Industrial Research**, 1962.

[37] R. Storn and K. Price, "Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces", Technical Report TR-95-012, **International Computer Science Institute**, Berkeley, CA, 1995.

[38] C. Ferreira, "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems", **Complex Systems**, 2001

[39] Webster B , Bernhard PJ . A local search optimization algorithm based on natural principles of gravitation. In: Proceedings of the 2003 interna- tional conference on information and knowledge engineering (IKE'03); 2003. p. 255–61 .

[40] Erol OK , Eksin I . A new optimization method: big bang–big crunch. **Adv Eng Softw** 2006;37:106–11 .

[41] Rashedi E , Nezamabadi-Pour H , Saryazdi S . GSA: a gravitational search algo- rithm. **Inf Sci** 2009;179:2232–48 .

[42] Kaveh A , Talatahari S . A novel heuristic optimization method: charged system search. **Acta Mech** 2010;213:267–89 .

[43] Formato RA . Central force optimization: A new metaheuristic with applica- tions in applied electromagnetics. **Prog Electromag Res** 2007;77:425–91 .

[44] Alatas B . ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization**. Expert Syst Appl** 2011;38:13170–80 .

[45] Hatamlou A . Black hole: a new heuristic optimization approach for data clus- tering. **Inf Sci** 2013;222:175–84 .

[46] Kaveh A , Khayatazad M . A new meta-heuristic method: ray optimization. **Comput Struct** 2012;112:283–94 .

[47] Du H , Wu X , Zhuang J . Small-world optimization algorithm for function opti- mization. Advances in natural computation**. Springer**; 2006. p. 264–73 .

[48] Shah-Hosseini H . Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. **Int J Comput Sci Eng** 2011;6:132–40 .

[49] Moghaddam FF, Moghaddam RF, Cheriet M. Curved space optimization: A random search based on general relativity theory. 2012. arXiv: 1208.2214 .

[50] Kennedy J , Eberhart R . Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks; 1995. p. 1942–8 .

[51] Dorigo M , Birattari M , Stutzle T . Ant colony optimization. **IEEE Comput Intell** 2006;1:28–39 .

[52] Abbass HA . MBO: Marriage in honey bees optimization –a haplometrosis polygynous swarming approach. In: Proceedings of the 2001 congress on evolutionary computation; 2001. p. 207–14 .

[53] Li X . A new intelligent optimization-artificial fish swarm algorithm [Doctor thesis]. China: Zhejiang University of Zhejiang; 2003 .

[54] Roth M , Stephen W . Termite: A swarm intelligent routing algorithm for mo- bilewireless Ad-Hoc networks. Stigmergic Optimization**. Springer Berlin Heidelberg**; 2006. p. 155–84 .

[55] Basturk B , Karaboga D . An artificial bee colony (ABC) algorithm for numeric function optimization. In: Proceedings of the IEEE swarm intelligence symposium; 2006. p. 12–14 .

[56] Pinto PC , Runkler TA , Sousa JM . Wasp swarm algorithm for dynamic MAX- SAT problems. Adaptive and natural computing algorithms. **Springer**; 2007. p. 350–7 .

[57] Mucherino A , Seref O . Monkey search: a novel metaheuristic search for global optimization. In: AIP conference proceedings; 2007. p. 162 .

[58] Yang C , Tu X , Chen J . Algorithm of marriage in honey bees optimization based on the wolf pack search. In: Proceedings of the 2007 international conference on intelligent pervasive computing, IPC; 2007. p. 462-7

[59] Lu X , Zhou Y . A novel global convergence algorithm: bee collecting pollen al- gorithm. Advanced intelligent computing theories and applications With as- pects of artificial intelligence. **Springer**; 2008. p. 518–25 .

[60] Yang X-S , Deb S . Cuckoo search via Lévy flights. In: Proceedings of the world congress on nature & biologically inspired computing, NaBIC 20 09; 20 09. p. 210–14 .

[61] Shiqin Y , Jianjun J , Guangxing Y . A dolphin partner optimization. In: Proceedings of the WRI global congress on intelligent systems, GCIS’09; 2009. p. 124–8 .

[62] Yang X-S . A new metaheuristic bat-inspired algorithm. In: Proceedings of the workshop on nature inspired cooperative strategies for optimization (NICSO 2010). **Springer**; 2010. p. 65–74 .

[63] Yang X-S . Firefly algorithm, stochastic test functions and design optimisation. **Int J Bio-Inspired Comput** 2010;2:78–84 .

[64] Oftadeh R , Mahjoob MJ , Shariatpanahi M . A novel meta-heuristic optimiza- tion algorithm inspired by group hunting of animals: hunting search. **Comput Math Appl** 2010;60:2087–98 .

[65] Askarzadeh A , Rezazadeh A . A new heuristic optimization algorithm for mod- eling of proton exchange membrane fuel cell: bird mating optimizer. **Int J Energy Res** 2012 .

[66] Gandomi AH , Alavi AH . Krill Herd: a new bio-inspired optimization algorithm. **Commun Nonlinear Sci Numer Simul** 2012;17(12):4831–45 .

[67] Pan W-T . A new fruit fly optimization algorithm: taking the financial distress model as an example. **Knowledge-Based Syst** 2012;26:69–74 .

[68] Kaveh A , Farhoudi N . A new optimization method: dolphin echolocation. **Adv Eng Softw** 2013;59:53–70 .

[69] Rao RV , Savsani VJ , Vakharia DP . Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. **Inf Sci** 2012;183:1–15 .

[70] Geem ZW , Kim JH , Loganathan G . A new heuristic optimization algorithm: harmony search. **Simulation** 2001;76:60–8 .

[71] Fogel D . Artificial intelligence through simulated evolution. **Wiley-IEEE Press**; 2009 .

[72] He S , Wu Q , Saunders J . A novel group search optimizer inspired by animal behavioural ecology. In: Proceedings of the 2006 IEEE congress on evolutionary computation, CEC; 2006. p. 1272–8 .

[73] Atashpaz-Gargari E , Lucas C . Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Proceedings of the 2007 IEEE congress on evolutionary computation, CEC; 2007. p. 4661–7 .

[74] Kashan AH . League championship algorithm: a new algorithm for numerical function optimization. In: Proceedings of the international conference on soft computing and pattern recognition, SOCPAR'09.; 2009. p. 43–8 .

[75] Tan Y , Zhu Y . Fireworks algorithm for optimization. Advances in swarm intel- ligence. **Springer**; 2010. p. 355–64 .

[76] Kaveh A . Colliding bodies optimization. Advances in metaheuristic algorithms for optimal design of structures. **Springer**; 2014. p. 195–232 .

[77] Gandomi AH . Interior search algorithm (ISA): a novel approach for global optimization. **ISA Trans** 2014 .

[78] Sadollah A , Bahreininejad A , Eskandar H , Hamdi M . Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. **Appl Soft Comput** 2013;13:2592–612 .

[79] Moosavian N , Roodsari BK . Soccer league competition algorithm: a new method for solving systems of nonlinear equations. **Int J Intell Sci** 2013;4:7.

[80] Dai C , Zhu Y , Chen W . Seeker optimization algorithm. Computational intelli- gence and security. **Springer**; 2007. p. 167–76 .

[81] Ramezani F , LotfiS . Social-based algorithm (SBA). **Appl Soft Comput** 2013;13:2837–56 .

[82] Ghorbani N , Babaei E . Exchange market algorithm. **Appl Soft Comput** 2014;19:177–87 .

[83] Eita MA , Fahmy MM . Group counseling optimization. **Appl Soft Comput** 2014;22:585–604 .

[84] Wolpert DH, Macready WG. No free lunch theorems for optimization. Evolut Comput, **IEEE Trans** 1997;1:67–82.

[85] **http://www.cleveralgorithms.com/natureinspired/evolution/genetic_algorithm.html** (Haziran 2019)

[86] **http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html** (Haziran 2019)

[87] Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74). **Springer**, Berlin, Heidelberg.

[88] Richardson, P., (2008). Bats. Natural History Museum, London.

[89] Yang, X. S. (2013). Bat algorithm: literature review and applications. arXiv preprint arXiv:1308.3900.

[90] Yang, X. S. (2012, September). Flower pollination algorithm for global optimization. In International conference on unconventional computing and natural computation (pp. 240-249). **Springer**, Berlin, Heidelberg.

[91] Payne, R. B., Sorenson, M. D., and Klitz, K.,2005. The Cuckoos, **Oxford University Press.**

[92] Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. arXiv preprint arXiv:1005.2908.

[93] Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169-178). **Springer**, Berlin, Heidelberg.

[94] Cengel YA, Boles MA (2005) Thermodynamics: an engineering approach, 5th edn. McGraw-Hill, New York

[95] Cuevas, E., Echavarría, A., & Ramírez-Ortegón, M. A.(2014). An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. **Applied intelligence**,40(2)

[96] Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. **Information sciences**, 179(13), 2232-2248.

[97] Yao, X., & Liu, Y. (1996). Fast Evolutionary Programming. Evolutionary programming, 3, 451-460.

[98] Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. **Knowledge-Based Systems**, 96, 120-133.

[99] Khalilpourazari, S., & Khalilpourazary, S. (2018). SCWOA: an efficient hybrid algorithm for parameter optimization of multi-pass milling process. **Journal of Industrial and Production Engineering**, 35(3), 135-147.

[100] Mirjalili, S., Jangir, P., & Saremi, S. (2017). Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. **Applied Intelligence**, 46(1), 79-95.

[101] Scharf I, Subach A, Ovadia O. Foraging behaviour and habitat selection in pitbuilding antlion larvae in constant light or dark conditions. **Anim Behav** 2008;76:2049–57.

[102] Griffiths D. Pit construction by ant-lion larvae: a cost-benefit analysis. **J Anim Ecol** 1986:39–57.

[103] Scharf I, Ovadia O. Factors influencing site abandonment and site selection in a sit-and-wait predator: a review of pit-building antlion larvae. **J Insect Behav** 2006;19:197–218.

[104] Singh, N., & Hachimi, H. (2018). A new hybrid whale optimizer algorithm with mean strategy of grey wolf optimizer for global optimization. **Mathematical and Computational Applications**, 23(1), 14.

[105] Doğan, C (2019). Balina Optimizasyon Algortitması ve Gri Kurt Optimizasyonu algoritmaları kullanılarak yeni hibrit optimizasyon algoritmalarının geliştirilmesi. Kayseri :Erciyes Üniversitesi Fen Bilimleri Enst.

[106] Danacı, M,.Doğan, C. (2019, Nisan). Improved Whale Optimization Algorithm. Talas,M *Uluslararası Erciyes Bilimsel Araştırmalar Kongresi* (s482-496)Yer: Erciyes üniversitesi

[107] Watkins WA, Schevill WE. Aerial observation of feding behavior in four baleen whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaean-gliae and Balaenoptera physalus. **J Mammal**. 1979;60:155–163.

[108] Khoury J, Ovrut BA, Seiberg N, Steinhardt PJ, Turok N (2002) From big crunch to big bang. **Phys Rev D** 65:086007

[109] Tegmark M (2004) Parallel universes. In: Barrow JD, Davies PCW, Harper CL Jr (eds) Science and ultimate reality: Quantum theory, cosmology, and complexity. **Cambridge University Press**, pp 459–491

[110] Eardley DM (1974) Death of white holes in the early Universe. **Phys Rev Lett** 33:442

[111] Steinhardt PJ, Turok N (2002) A cyclic model of the universe. **Science** 296:1436–1439

[112] Davies PC (1978) Thermodynamics of black holes. **Rep Prog Phys** 41:1313

[113] Morris MS, Thorne KS (1988) Wormholes in spacetime and theiruse for interstellar travel: a tool for teaching general relativity. **Am J Phys** 56:395–412Mirjalili,

[114] S.,Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. **Neural Computing and Applications**, 27(2), 495-513.

[115] **http://www.alimirjalili.com/SCA.html** (Haziran 2019)

[116] Zhu, A., Xu, C., Li, Z., Wu, J., & Liu, Z. (2015). Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. **Journal of Systems Engineering and Electronics**, 26(2), 317-328.

# ÖZGEÇMİŞ

## KİŞİSEL BİLGİLER

Adı, Soyadı: Bahadur ALIZADA

Uyruğu: Azerbaycan (A.C.)

Doğum Tarihi ve Yeri: 3 Nisan 1995, Bakü

Medeni Durumu: Bekar

Tel: +90 541 488 59 99

email: bahaduralizade@gmail.com

## EĞİTİM

| Derece | Kurum | Mezuniyyet |
|---|---|---|
| Lisans | Azerbaycan Devlet Petrol ve Sanayi Üniversitesi<br>Bilgisayar Mühendisliği | 2016 |
| Lise | Binagadi kasabası 182 numaralı lise<br>Bakü | 2012 |

## YABANCI DİL

İngilizce – iyi

Rusca – başlangıç

Fransızca - başlangıç