T.C.
BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ PROBLEMLERİNİN METASEZGİSEL ALGORİTMALARLA ÇÖZÜLMESİ VE SONUÇLARININ KARŞILAŞTIRILMASI

Samet PANDA

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

OCAK 2024

T.C.
BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ PROBLEMLERİNİN METASEZGİSEL ALGORİTMALARLA ÇÖZÜLMESİ VE SONUÇLARININ KARŞILAŞTIRILMASI

Samet PANDA

DANIŞMAN
Dr. Öğr. Üyesi Serhat KILIÇARSLAN

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

OCAK 2024

ONAY

Samet PANDA tarafından hazırlanan "**Makine Mühendisliği Problemlerinin Metasezgisel Algoritmalarla Çözülmesi Ve Sonuçlarının Karşılaştırılması** " adlı tez çalışması 29/01/2024 tarihinde yapılan sınavla aşağıdaki jüri tarafından oybirliği ile Bandırma Onyedi Eylül Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri		**İmza**
Dr. Öğr. Üyesi Serhat KILIÇARSLAN (Danışman)	(BANÜ)	_____
Doç. Dr. Abdullah ELEN (Üye)	(BANÜ)	_____
Dr. Öğr. Üyesi Mehmet Akif BÜLBÜL (Üye)	(NEVÜ)	_____

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …/…/…gün ve …/… sayılı kararı ile onaylanmıştır.

Doç. Dr. Abdullah YEŞİL

Enstitü Müdürü

BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS ÇALIŞMASI
ETİK BEYANI

Bandırma Onyedi Eylül Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre hazırlamış olduğum "**Makine Mühendisliği Problemlerinin Metasezgisel Algoritmalarla Çözülmesi Ve Sonuçlarının Karşılaştırılması** " adlı tezimin özgün bir çalışma olduğunu, tez hazırlanırken tüm aşamalarda bilimsel etik ilkelerine uygun davrandığımı, tez kapsamında sunulan tüm verileri bilimsel etik ilkelerine uygun elde ettiğimi, tezde faydalandığım tüm eserlere atıf yaptığımı ve kaynaklar kısmında bu eserleri gösterdiğimi beyan ederim. 29/01/2024

Samet PANDA
İmza

ÖNSÖZ

Optimizasyon, makine tasarımlarında önemli bir rol oynamakta ve mekanik bileşenlerin optimal şekilde tasarlanması gerekir. Bununla birlikte, mekanik sistemlerin ya da bileşenlerinin tasarım optimizasyonunda çok sayıda tasarım değişkeni ve karmaşık kısıtlamalar söz konusudur. Böylece, gereksinimlere dayalı bir mekanik sistem tasarlarken metasezgiseller gibi optimizasyon teknikleri, malzeme maliyetlerini düşürmek, bileşenlerin daha iyi çalışmasını sağlamak ve üretim hızını artırmak gibi hususlarda mühendislere birçok yönden yardımcı olur.

Bu tez çalışmasında, makine mühendisliği alanındaki gerçek dünya tasarım problemlerinin çözümünde metasezgisel yöntemlerden faydalanıldı. Deneysel çalışmalarda, dört metasezgisel algoritma ile sekiz mekanik tasarım problemi ele alınarak, farklı yineleme ve popülasyon sayılarına göre hem yakınsama değeri hem de çalışma süreleri bakımından karşılaştırılmalı analizi yapıldı.

Tez çalışması sürecinde, benden yardımlarını ve desteklerini esirgemeyen danışman hocam Sayın Dr. Öğr. Üyesi Serhat Kılıçarslan'a teşekkürü borç bilirim.

Bilgi ve tecrübelerinden yararlandığımız Doç. Dr. Abdullah Elen'e ve Dr. Öğr. Üyesi Cemil Közkurt'a teşekkürü borç bilirim.

İyi ve kötü günümde her zaman yanımda olan eşim Sezen'e, yüksek lisans öğrenim süresince beraber yol yürüdüğümüz arkadaşım Oğuzhan Dilber'e ve bu günlere gelmemde üzerimde emeği olan rahmetli babam Halil Panda'ya ve annem Ayşe Panda'ya sonsuz şükranlarımı sunarım.

ÖZET

YÜKSEK LİSANS TEZİ

MAKİNE MÜHENDİSLİĞİ PROBLEMLERİNİN METASEZGİSEL ALGORİTMALARLA ÇÖZÜLMESİ VE SONUÇLARININ KARŞILAŞTIRILMASI

Samet PANDA

Bandırma Onyedi Eylül Üniversitesi
Fen Bilimleri Enstitüsü
Mekatronik Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Serhat KILIÇARSLAN
Ocak 2024, 72 sayfa

Optimizasyonun amacı, bir dizi öncelikli koşul veya kısıtlamaya göre "en iyi" tasarımı elde etmektir. Bunlar arasında üretkenlik, güç, güvenilirlik, uzun ömür, verimlilik ve kullanım gibi faktörlerin en üst düzeye çıkarılması yer alır. Optimizasyon tekniklerinden biri olan metasezgisel algoritmalar, çözüm bulmadaki basitlikleri ve hızlılıkları nedeniyle son yıllarda mühendislik tasarımında giderek artan bir önem kazanmıştır. Bu durum, söz konusu metasezgisellerin yaygınlaşmasına ve yeni algoritmalar geliştirme eğiliminin artmasına yol açmıştır.

Bu tez çalışmasıda, metasezgisel optimizasyon yöntemlerinden Karınca Kolonisi Optimizasyonu, Yapay Arı Kolonisi, Salp Sürü Algoritması ve Sinüs Kosinüs Algoritması ile makine mühendisliği alanındaki gerçek dünya problemlerinden Basınçlı Kap Tasarımı, Robot Kavrayıcı Problemi, Makaralı Rulman Tasarım Problemi, Kademeli Konik Kasnak Problemi, Germe Sıkıştırma Yayı Tasarımı, Üç Çubuklu Kafes Kiriş Tasarım Problemi, Hız Düşürücüsünün Ağırlık Minimizasyonu ve Kaynaklı Kiriş Tasarımı'nın ele alınması ve bu metasezgisellerin performans analizlerinin yapılması konusunda bir araştırma yapılmıştır. Deneysel çalışmalarda, çeşitli sayıdaki yineleme ve popülasyon parametrelerine göre aşamalı

olarak altı farklı senaryo belirlenmiştir. Elde edilen sonuçlara göre, bu türdeki problemler için en uygun yöntemin hangisi olduğuna karar verilmiştir.

Anahtar kelimeler: Optimizasyon, Metasezgiseller, Gerçek Dünya Problemleri, Mühendislik Problemleri.

ABSTRACT

M.Sc. THESIS

SOLVING MECHANICAL ENGINEERING PROBLEMS WITH METAHEURISTIC ALGORITHMS AND COMPARING THE RESULTS

Samet PANDA

Bandırma Onyedi Eylül University
Graduate School of Natural and Applied Sciences
Department of Mechatronics Engineering

Supervisor: Assist. Prof. Dr. Serhat KILIÇARSLAN
January 2024, 72 pages

The goal of optimization is to achieve the "best" design according to a set of prior conditions or constraints. These include maximizing factors such as productivity, power, reliability, longevity, efficiency and availability. Metaheuristic algorithms, one of the optimization techniques, have gained increasing importance in engineering design in recent years due to their simplicity and rapidity in finding solutions. This has led to the proliferation of these metaheuristics and an increased tendency to develop new algorithms.

In this thesis, Ant Colony Optimization, Artificial Bee Colony, Salp Swarm Algorithm and Sine Cosine Algorithm, which are metaheuristic optimization methods, and Pressure Vessel Design, Robot Gripper Problem, Roller Bearing Design Problem, Gradual Conical Pulley Problem, Tension Compression Spring, which are real-world problems in the field of mechanical engineering, are used. A research has been conducted on Design, Three-Bar Truss Design Problem, Weight Minimization of Speed Reducer and Welded Beam Design and performance analysis of these metaheuristics. In experimental studies, six different scenarios were gradually determined according to various numbers of iterations and population parameters. According to the results obtained, it was decided which method is the most suitable for this type of problems.

Keywords: Optimization, Metaheuristics, Real-world Problems, Engineering Problems.

İÇİNDEKİLER

Sayfa

ÖZET ..... **I**

ABSTRACT ..... **III**

İÇİNDEKİLER ..... **IV**

ŞEKİL LİSTESİ ..... **VI**

TABLO LİSTESİ ..... **VII**

SİMGE VE KISALTMA LİSTESİ ..... **VIII**

1. GİRİŞ **1**

1.1 Tezin Amacı ve Katkısı **2**

1.2 Tezin Organizasyonu **2**

2. LİTERATÜR İNCELEMESİ **4**

3. METASEZGİSELLER **9**

3.1 KARINCA KOLONİSİ OPTİMİZASYONU **9**

3.2 YAPAY ARI KOLONİSİ **12**

3.3 SALP SÜRÜ ALGORİTMASI **14**

3.4 SİNÜS KOSİNÜS ALGORİTMASI **17**

4. GERÇEK DÜNYA PROBLEMLERİ **20**

4.1 BASINÇLI KAP TASARIMI **20**

4.2 ROBOT KAVRAYICI PROBLEMİ **22**

4.3 MAKARALI RULMAN TASARIM PROBLEMİ **23**

4.4 KADEMELİ KONİK KASNAK PROBLEMİ **25**

4.5 GERME SIKIŞTIRMA YAYI TASARIMI **27**

4.6 ÜÇ ÇUBUKLU KAFES KİRİŞ TASARIM PROBLEMİ **28**

4.7 HIZ DÜŞÜRÜCÜSÜNÜN AĞIRLIK MİNİMİZASYONU **29**

4.8 KAYNAKLI KİRİŞ TASARIMI **31**

5. DENEYSEL ÇALIŞMALAR 34

5.1 PERFORMANS ÖLÇÜM KRİTERLERİ 34

5.2 GERÇEK DÜNYA PROBLEMLERİNİN TESTLERİ 35

5.2.1 Deney I 37

5.2.2 Deney II 40

5.2.3 Deney III 43

5.2.4 Deney IV 46

5.2.5 Deney V 49

5.2.6 Deney VI 51

6. TARTIŞMA VE SONUÇ 55

7. KAYNAKLAR 58

8. EKLER 63

ÖZGEÇMİŞ ..... Hata! Yer işareti tanımlanmamış.

ŞEKİL LİSTESİ

Şekil **Sayfa**

Şekil 3-1: Olasılık dağılımları; (a) Ayrık olaslık dağılımı, (b) Sürekli olaslık dağılımı [36]. 10

Şekil 3-2: ACO$_R$'nin akış şeması. 12

Şekil 3-3: Yapay Arı Kolonisi algoritmasının akış şeması. 14

Şekil 3-4: Salp ve salp sürüsü veya zinciri. 15

Şekil 3-5: Salp Sürü Algoritmasının Akış Şeması. 16

Şekil 3-6: Sinüs ve kosinüs fonksiyonlarının bir sonraki pozisyona etkileri [45]. 18

Şekil 3-7: Sinüs Kosinüs Algoritmasının akış şeması. 19

Şekil 4-1: Basınçlı kap tasarımı problemi [49]. 21

Şekil 4-2: Robot Tutucu Probleminin Şematiği [48]. 22

Şekil 4-3: Makaralı rulman tasarım probleminin şematik görünümü [50]. 24

Şekil 4-4: Kademeli Konik Kasnak Probleminin Şematiği [51]. 26

Şekil 4-5: Germe Sıkıştırma Yayı Tasarımı [52] 27

Şekil 4-6: Üç Çubuklu Kafes Tasarımı Probleminin Şematiği [53]. 29

Şekil 4-7: Hız Düşürücü Sistemin Şematiği. [54]. 30

Şekil 4-8: Kaynaklı Kiriş Tasarımı [56]. 32

Şekil 5-1: Deney I'e göre mühendislik problemlerinin yakınsama grafikleri. 38

Şekil 5-2: Deney II'ye göre mühendislik problemlerinin yakınsama grafikleri. 41

Şekil 5-3: Deney III'e göre mühendislik problemlerinin yakınsama grafikleri. 44

Şekil 5-4: Deney IV'e göre mühendislik problemlerinin yakınsama grafikleri. 47

Şekil 5-5: Deney V'e göre mühendislik problemlerinin yakınsama grafikleri. 50

Şekil 5-6: Deney VI'ya göre mühendislik problemlerinin yakınsama grafikleri. 53

TABLO LİSTESİ

Tablo **Sayfa**

Tablo 5-1: Metasezgisel algoritmaların parametreleri. 34
Tablo 5-2: Mühendislik problemlerinin özellikleri. 35
Tablo 5-3: Deneylerin özellikleri. 36
Tablo 5-4: Deney I'e göre metasezgisellerin en iyi uygunluk değerleri. 37
Tablo 5-5: Deney I'e göre metasezgisellerin çalışma süreleri (sn). 37
Tablo 5-6: Deney II'ye göre metasezgisellerin en iyi uygunluk değerleri. 40
Tablo 5-7: Deney II'ye göre metasezgisellerin çalışma süreleri (sn). 40
Tablo 5-8: Deney III'e göre metasezgisellerin en iyi uygunluk değerleri. 43
Tablo 5-9: Deney III'e göre metasezgisellerin çalışma süreleri (sn). 43
Tablo 5-10: Deney IV'e göre metasezgisellerin en iyi uygunluk değerleri. 46
Tablo 5-11: Deney IV'e göre metasezgisellerin çalışma süreleri (sn). 46
Tablo 5-12: Deney V'e göre metasezgisellerin en iyi uygunluk değerleri. 49
Tablo 5-13: Deney V'e göre metasezgisellerin çalışma süreleri (sn). 49
Tablo 5-14: Deney VI'ya göre metasezgisellerin en iyi uygunluk değerleri. 51
Tablo 5-15: Deney VI'ya göre metasezgisellerin çalışma süreleri (sn). 51

SİMGE VE KISALTMA LİSTESİ

Simgeler	**Açıklama**
η	: Çözüm elemanlarına atanan buluşsal bir değer
α, β	: Feromon ile buluşsal bilginin ilişki etkisini belirleyen parametre
ρ	: Feromon buharlaşma hızı
$\Delta\tau$	: Belirli bir karınca üzerinde biriken feromon miktarı
ω	: Gauss fonksiyonlarıyla ilişkili ağırlıkların vektörü
μ	: Ortalamaların vektörü
σ	: Standart sapmaların vektörü
ϕ	: Besin kaynaklarını belirleyen rastgele bir sayı
T	: Maksimum yineleme sayısı

Kısaltmalar	**Açıklama**
AA	: Ateşböceği Algoritması (Firefly Algorithm)
ABC	: Yapay Arı Kolonisi (Artificial Bee Colony)
ACO	: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
AIS	: Yapay Bağışıklık Sistemini (Artificial Immune System)
BA	: Yarasa Algoritması (Bat Algorithm)
BBO	: Biyografik Tabanlı Optimizasyon (Biogeography-Based Optimization)
CSO	: Tavuk Sürüsü Optimizasyonu (Chicken Swarm Optimization)
CSS	: Şarjlı Sistem Araması (Charged System Search)
DE	: Diferansiyel Evrim (Differential Evolution)
FA	: Ateşböceği Algoritması (Firefly Algorithm)
GSA	: Yerçekimsel Arama Algoritması (Gravitational Search Algorithm)
GWO	: Gri Kurt Optimizasyon Algoritması (Grey Wolf Optimizer)
HGSO	: Henry Gaz Çözünürlüğü Optim. (Henry Gas Solubility Optimization)
MVO	: Çoklu Evren Optimizasyon Algoritması (Multiverse Optimizer)
PDF	: Olasılık Yoğunluk Fonksiyonu (Probability Density Function)
PSO	: Parçaçık Sürü Optimizasyonu (Particle Swarm Optimization)
SA	: Benzetimli Tavlama (Simulated Annealing)
SCA	: Sinüs Kosinüs Algoritması (Sine Cosine Algorithm)

SFS : Stokastik Fraktal Arama (Stochastic Fractal Search)

SHO : Benekli Sırtlan Optimizasyonu (Spotted Hyena Optimizer)

SM : Örümcek Maymun Algoritması (Spider Monkey)

SSA : Salp Sürü Algoritması (Salp Swarm Algorithm)

1. GİRİŞ

Dünyamızda nüfusun gelişip artmasıyla birlikte mevcut kaynakların optimum kullanımı insanlık için en önemli konulardan biri haline gelmiştir. Bu bağlamda mühendislik sistemleri (örneğin; yapısal, mekanik ve endüstriyel sistemler), kaynaklar ve tüketimler arasında gerekli dengenin sağlanmasında büyük rol oynamaktadır. Genel olarak mühendislik tasarımı, minimum kaynakları kullanarak tüm beklentileri karşılayan bir sistem elde etme sanatı olarak tanımlanabilir [1, 2]. Burada bahsedilen kaynak, genel anlamda zaman, malzeme, insan emeği gibi farklı kavramları kapsamaktadır. Bu tanıma dayanarak, çoğu mühendislik tasarım süreci bir optimizasyon problemine dönüştürülebilir ve bunları çözmek için sağlam bir optimizasyon tekniği gereklidir [3].

Gerçek dünyadaki mühendislik tasarım problemleri hem endüstri hem de endüstrideki çeşitli araştırma disiplinlerinde yaygındır. Bu tür problemleri çözmek için birçok optimizasyon algoritması kullanılmıştır. Ancak problemlerin ölçeği ve zorluğu arttıkça algoritmanın performansı önemli ölçüde azalmaktadır. Literatürdeki mühendislik tasarım problemlerini verimli bir şekilde ele almak için optimizasyon yöntemlerinin çeşitli versiyonları önerilmiştir [4].

Optimizasyon algoritmaları, arama uzayındaki aday çözümleri bulmak için kullanılan rastgele bir etmen popülasyonundan oluşur [5, 6]. Süreç, problem alanındaki etmenlerin başlatılmasıyla başlar. Daha sonra algoritma birçok yinelemeden geçer ve her yineleme, kriterin sonu karşılanana kadar uygun aday çözümü verir. Tüm yinelemeler arasından en uygun aday çözüm, optimal çözümdür [7]. Stokastik algoritmaların keşif ve kullanım gibi birkaç temel adımdan oluşması oldukça şaşırtıcıdır. Keşif, algoritmanın arama alanı anlamına gelir. Bu aşamada aday çözümler bir takım değişikliklere uğrar. Ayrıca, kullanım, farklı uygulanabilir çözümler etrafında yerel optimumu bulma yeteneğidir [8].

Genellikle doğal fenomenlerden esinlenerek geliştirilen Karınca Kolonisi Optimizasyonu, Genetik Algoritmalar ve Parçacık Sürüsü Optimizasyonu gibi metasezgisel arama teknikleri, karmaşık optimizasyon problemlerine umut verici çözümler bulabilme yetenekleri sayesinde popüler hale gelmiştir [9]. Ayrıca metasezgiseller hem ayrık hem de gerçek değerli değişkenleri ele alabilir ve çok çeşitli optimizasyon problemlerine etkili bir şekilde uygulanabilir. Temel olarak, hem yörünge hem de popülasyon temelli meta-sezgisel yaklaşımlar, rastgele hareketler yoluyla çözüm uzayında küresel optimumun yerini belirlemeyi amaçlamaktadır. Metasezgiseller arasındaki temel fark, çözüm uzayındaki bir sonraki hamleyi önerme şeklidir. Bu, optimizasyon algoritması geliştiricilerini, sağlam optimizasyon algoritmaları oluşturmak için daha verimli

metodolojiler bulmaya motive eder. Ancak bazen bu, anlaşılması ve uygulanması zor olan karmaşık yaklaşımlarla sonuçlanır [10].

Bu tez çalışmasında, gerçek-dünya mühendislik tasarım problemlerinin literatürdeki iyi bilinen bazı meta-sezgisel optimizasyon algoritmaları ile çözümlenmesi ve sonuçlarının karşılaştırılması amaçlanmıştır. Tez kapsamında ele alınan algoritmalar şunlardır: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization, ACO), Yapay Arı Kolonisi (Artificial Bee Colony, ABC), Salp Sürü Algoritması (Salp Swarm Algorithm, SSA) ve Sinüs Kosinüs Algoritması (Sine Cosine Algorithm, SCA). Deneysel çalışmalarda bu algoritmalar, Basınçlı Kap Tasarımı, Robot Kavrayıcı Problemi, Makaralı Rulman Tasarım Problemi, Kademeli Konik Kasnak Problemi, Germe Sıkıştırma Yayı Tasarımı, Üç Çubuklu Kafes Kiriş Tasarım Problemi, Hız Düşürücüsünün Ağırlık Minimizasyonu ve Kaynaklı Kiriş Tasarımı olmak üzere sekiz makine mühendisliği tasarımı problemi üzerinde test edilmiştir.

1.1 Tezin Amacı ve Katkısı

Bu tez çalışmasının amacı, literatürde iyi bilinen metasezgisel yöntemleri kullanarak, makine mühendisliği alanındaki gerçek dünya tasarım problemlerinin çözümlenmesi ve optimize eden algoritmaların hem yakınsama hem de çalışma süreleri bakımından karşılatırışmalı analizini kapsamaktadır. Elde edilen bulgular ışığında, mühendislik tasarım problemleri üzerine yapılan çalışmalar için bu tezin bir referans kaynağı olacağı düşünülmektedir.

1.2 Tezin Organizasyonu

Bu tez çalışması şu şekilde organize edilmiştir:

- Giriş bölümünde, tez konusunun tanıtımı yapılmış, mühendislik problemlerinin çözümü için hangi metasezgisel yöntemlerin kullanıldığı belirtilmiş, tez kapsamında yapılan çalışmaların literatüre katkıları sunulmuş ve tezin takip eden bölümleri tanıtılmıştır.
- İkinci bölümde, araştırmacılar tarafından önerilen metasezgisel optimizasyon algoritmaları ve gerçek-dünya mühendislik problemlerinin çözümünde kullanılan algoritmalar konusunda ayrıntılı bir literatür bilgisi sunulmuştur.
- Üçüncü bölümde, tez çalışmasında kullanılan metasezgisel optimizasyon algoritmaları, esin kaynakları ve metodolojileriyle birlikte detaylı bir şekilde tanıtılmıştır.

- Dördüncü bölümde, metasezgisel yöntemlerin performans testlerinin yapılması amacıyla sekiz farklı gerçek dünya mühendislik tasarım problemlerinin tanıtımına, uygunluk/amaç fonksiyonlarına ve bu problemlerin kısıt fonksiyonlarına yer verilmiştir.
- Beşinci bölümde, metasezgisel yöntemlerinin mühendislik problemleri üzerinde, yineleme ve popülasyon sayılarının değişen durumlarına göre altı farklı deney gerçekleştirilmiş ve ayrıntılı olarak sonuçlarından bahsedilmiştir.
- Altıncı ve son bölümde ise tez kapsamında yapılan çalışmalardan elde edilen sonuçlar değerlendirilmiş ve gelecekte yapılabilecek çalışmalara dair önerilere yer verilmiştir.

2. LİTERATÜR İNCELEMESİ

Metasezgisel, optimizasyon problemlerini çözmek için kullanılan bir algoritma türüdür. Genellikle problemin geleneksel yöntemlere göre oldukça karmaşık olduğu veya bu yöntemlerin çok yavaş olduğu durumlarda kullanılırlar. Çeşitli türden optimizasyon problemlerinin çözümü için literatürde çok sayıda metasezgisel teknikler önerilmiştir.

Farmer ve diğ. [11] optimizasyon problemlerini çözmek için biyolojik bağışıklık sisteminin yapısını ve işlevini simüle eden Yapay Bağışıklık Sistemini (*Artificial Immune System*, AIS) tanıttı. Bağışıklık sistemi, vücudu yabancı, tehlikeli hücrelere veya onu istila eden maddelere karşı korur. Aslında AIS, insan vücudunun hastalıklara karşı aşı kullanarak bağışıklık kazanması yöntemini kopyalıyor. AIS, belirli sayıda antikor ve antijen arasından en iyi çözümü belirlemek için VACCINE-AIS algoritmasını uygular. AIS'de karar değişkenleri ve çözüm üretmede kullanılan yapılar, optimizasyon problemlerini çözmek için kullanılan bağışıklık sistemindeki antikorlar ve antijenlerdir. Kennedy ve Eberhart [12] tarafından Parçacık sürüsü optimizasyonu (*Particle Swarm Optimization*, PSO) adını verdikleri, kuşların ve balıkların kaynaşma davranışını taklit eden, sürü zekâsına dayalı bir meta-sezgisel algoritma önerdiler. PSO'da sürünün parçacıkları rastgele oluşturulur ve yeni çözümler, mevcut konumlara, hızlara ve mevcut en iyi çözüme bağlı olan iki yinelemeli denklemlere göre yinelemeli olarak güncellenir. Bu popülasyona dayalı bir algoritmadır ve küresel optimuma, tasarım alanı birden fazla parçacık tarafından yeterince uzun bir sürede iyi bir şekilde araştırıldığında ulaşılabilmektedir. Storn ve Price [13] tarafından geliştirilen Diferansiyel Evrim (*Differential Evolution*, DE), küresel bir optimizasyon algoritmasıdır. Genetik algoritma ve evrimsel algoritma gibi diğer popüler yaklaşımlara benzer şekilde, diferansiyel evrim, aday çözümlerin başlangıç popülasyonuyla başlar. Bu aday çözümler, popülasyona çaprazlama, mutasyon ve seçim getirilerek ve en uygun aday çözümler tutularak yinelemeli olarak geliştirilir. Dan Simon [14] tarafından ilk olarak 2008 yılında önerilen Biyografik tabanlı optimizasyon (*Biogeography-Based Optimization*, BBO), popülasyon tabanlı bir evrimsel algoritmadır. BBO'daki cevaba habitat denir ve habitat, yaşadığı kişinin vektörü olarak kabul edilir. Ayrıca her bir habitatın değeri, habitat uygunluk indeksi ile tanımlanır. Bu indeksin yüksek değeri, habitatın yüksek uygunluk fonksiyonunu gösterir. BBO'nun üç ana operatörü göç, mutasyon ve elitizmdir. BBO'da her habitatın kendi göç oranı, göç oranı ve mutasyon oranı vardır. Yang [15] tarafından geliştirilen ateşböceği algoritması (*Firefly Algorithm*, FA) sürü zekâsı tabanlı bir algoritmadır. Bu algoritma, ateşböceklerinin çekim mekanizması ve yanıp sönme özelliklerini kullanır. Kısa

mesafeli çekim, uzun mesafeli çekimden daha güçlü olduğundan, sürü otomatik olarak birden fazla alt sürüye bölünebilir, dolayısıyla ateşböceği algoritması özellikle doğrusal olmayan çok modlu optimizasyon problemleri için uygundur. Kaveh ve Talatahari [16] tarafından önerilen Şarjlı Sistem Araması (*Charged System Search*, CSS), mekanikteki Newton yasalarına ve elektrostatikteki geçerli Coulomb ve Gauss yasalarına dayanarak geliştirilmiştir. Algoritma çoklu ajana sahiptir ve her ajana Yüklü Parçacık (CP) adı verilir. Her CP'nin bir elektrik kuvveti vardır ve Coulomb ve Gauss yasalarına göre diğer CP'leri etkiler. Bu kuvvet CP'ye ivme kazandırır ve her CP'nin hızı zamanla değişir. Hareket yasaları ve sonuçta ortaya çıkan kuvvetler, CP'lerin yeni konumunu (yeni çözümü) belirler. Bu pozisyonlar değerlendirilerek daha iyi olması durumunda eski pozisyonlarla değiştirilir. Bu eğilim, maksimum yineleme sayısına ulaşana kadar devam eder ve bu ölçüde en iyi sonucu elde eder. Yang [17] tarafından 2010 yılında Yarasa Algoritması (*Bat Algorithm*, BA) önerilmiştir. Algoritmanın ilkesi, mikro yarasaların eko-lokasyon davranışını değişen frekans, ses yüksekliği ve darbe emisyon oranlarıyla idealleştirmektir. Bu tür özellikler, keşif ve kullanım arasında denge kurmaya çalışmak amacıyla yarasa algoritmasının denklemlerini güncellemek için kullanılır. Mirjalili ve diğ. [18] tarafından önerilen Gri Kurt Optimizasyon Algoritması (*Grey Wolf Optimization*, GWO), gri kurdun avlanma politikalarından ilham alan sürü tabanlı bir meta-sezgiseldir. GWO, popülasyonu dört seviyeye ayırır: alfa, beta, delta ve omega. Alfalar kurtların yaşaması, avlanması ve hareket ettirilmesi konusunda karar veren liderlerdir. Beta kurtlar ise alfaya danışmanlık yapar. Delta, tehlike olduğunda uyarıda bulunmaktan, sürüyü korumaktan, hasta veya yaralı kurtlara yiyecek sağlamaktan ve bakımdan sorumludur. Son olarak omega, liderlere itaat etmek zorunda olan son kurttur. GWO; avlanma, arama, kuşatma ve ava saldırma olmak üzere dört aşamayı takip eder. Meng ve diğ. [19] tarafından geliştirilen Tavuk Sürüsü Optimizasyonu (*Chicken Swarm Optimization*, CSO), tavukların sosyal davranışlarından ilham alan bir algoritmadır. Bu yöntemde, her tavuğun kendine özgü hareket yasaları vardır. Tavukların uygunluk değerleri kendilerini horoz, tavuk ve civciv olarak tanımlayacaktır. Bu kimlikler onları gruplara ayıracaktır. En uygun, en zayıf ve orta seviyedeki tavuklar buna göre horoz, civciv ve tavuk olarak simüle edilir. Hiyerarşik düzen tavuk grubunda hayati bir rol oynar ve bu özellik CSO algoritmasını modellemek için kullanılır. Bansal ve diğ. [20] tarafından önerilen Örümcek Maymun (*Spider Monkey*, SM) algoritması, örümcek maymunlarının en uygun besin kaynaklarını arama konusundaki davranışlarından ilham alan, popülasyona dayalı bir optimizasyon tekniğidir. Bu yöntemde, besin kaynağının mükemmelliği çözümün uygunluğuna karşılık gelir. SM algoritmasının temel özellikleri ve stratejileri, yapay arı kolonisi algoritmasına benzer. Keşif ve kullanım fonksiyonları, SM

algoritmasının büyük arama alanı gerçekleştirmesine ve mümkün olan en iyi çözümleri üretmesine olanak tanımaktadır.

Makine mühendisliği alanındaki tasarım problemleri, oldukça zaman alıcı ve karmaşık hesaplamalar gerektirmektedir. Bu türdeki problemlerin birçoğu konvansiyonel hesaplama yöntemleriyle (türevi alınmak suretiyle) çözümleme işlemi yapılmaktadır. Ancak problemlerin türevlenebilir yapıda olmaları ve optimal noktasının tespit edilebilmesi için uygun bir başlangıç noktasından arama işleminin yapılması gerekir aksi halde küresel optimum yerine lokal optimum çözümler bulunabilir [21]. Bu problemlerin çözümünde, global optimum noktasını bulmada en etkili ve düşük maliyetli yöntem metasezgisel optimizasyon algoritmalarıdır. Literatürde, mühendislik alanındaki bu problemlerin çözümü için metasezgisel yöntemleri içeren çok sayıda çalışma bulunmaktadır. Balande ve diğ. [22] tarafından, kısıt yönetimi stratejilerine ilişkin bir araştırma sunulmuştur. Kısıtlı mühendislik optimizasyon problemlerinin çözümü için gelişmiş Ateşböceği Algoritmasına sahip Stokastik Sıralama (*Stochastic Ranking with Improved Firefly Algorithm*, SRIFA) olarak bilinen hibrit bir algoritma önerdiler. Stokastik sıralama tekniği, ağırlık fonksiyonları ile ceza arasındaki dengeyi korumak için yaygın olarak kullanılmaktadır. SRIFA, hedef ve ceza rolleri arasındaki dengeyi korumak için fizibilite ilkelerini kullanır. Önerdikleri yöntemin, performansını değerlendirmek amacıyla beş mühendislik problemi kullanmışlardır. SRIFA'nın sonuçlarının öncü sürümü olan FA'dan daha iyi olduğu gösterilmiştir. Yıldırım ve Karcı [23] tarafından, literatürde iyi bilinen bir mühendislik problemi olan Üç Çubuklu Kafes Kiriş Tasarımı (ÜÇKTK) probleminin çözümü için Yapay Atom Algoritması adı verilen bir yöntem önerilmiştir. Yapılan deneysel çalışmalarda, önerilen yöntemin Gugukkuşu Arama, Mayın Patlama, Yarasa ve Kriket algoritmaları da dâhil olmak üzere bir dizi algoritmadan daha iyi performans gösterdiği bildirilmiştir. Chagwiza ve diğ. [24], iki devreli su şebekesi problemi, 10 barlık kafes problemi vb. mühendislik optimizasyon problemlerini çözmek için Grotschel-Holland ve Max-Min Ant Sistemi yöntemlerini birleştiren GHMMAS yaklaşımını sunmuşlardır. Yapılan deneysel çalışmalarda, yüksek dereceli polinom problemlerinin çözümü söz konusu olduğunda GHMMAS etkili bir performans sergilediği bildirilmiştir. Rahman ve diğ. [25], çeşitli mühendislik problemlerininden çekme/bastırma yayı, basınçlı kap tasarımı ve kaynaklı kiriş tasarımı problemleri çözmek için Kaosla geliştirilmiş Stokastik Fraktal Arama (*Chaos-enhanced Stochastic Fractal Search*, CFS) yöntemini temel alan bir model önermişlerdir. Yakınsama oranı ve doğruluğu söz konusu olduğunda CFS modeli, standart Stokastik Fraktal Arama (*Stochastic Fractal Search*, SFS) algoritmasından daha iyi performans gösterdiğini ifade etmişlerdir. Hashim ve diğ. [26] tarafından Henry gaz kanunu göre davranışı

taklit eden yeni bir metasezgisel yaklaşım olan ve adını Henry Gaz Çözünürlüğü Optimizasyonu (*Henry gas solubility optimization,* HGSO) verdikleri bir metasezgisel yöntem tanıtılmaktadır. Henry yasası, belirli bir sıcaklık ve hacimde çözünmüş gaz miktarını ölçen bir gaz düşüklüğüdür. Sömürü ve keşif arasında bir denge sağlamak için HGSO, gaz toplama davranışını taklit eder. HGSO'nun performansı, PSO, GWO ve WOA dâhil olmak üzere bir dizi algoritmayla karşılaştırılmış ve oldukça rekabetçi sonuçlar alındığı belirtilmiştir. Kaleka ve diğ. [27], kaynaklı kiriş tasarımı, çekme/basma yayı tasarım problemi, makaralı rulman tasarım problemi, basınçlı kap tasarımı ve hız düşürücü tasarımı olarak bilinen mühendislik problemlerinin çözümünde, çeşitli metasezgisel algoritmaların performanslarını incelenmiştir. Çalışma kapsamında, Yerçekimsel Arama Algoritması (*Gravitational Search Algorithm,* GSA), Benekli Sırtlan Optimizasyonu (*spotted hyena optimizer,* SHO), GWO, Yarasa Algoritması (*bat algorithm,* BA), WOA, MFO, ACO ve PSO yer almaktadır. Deneysel sonuçlara göre makaralı rulman tasarımı problemi için hem GWO hem de SHO karşılaştırılabilir optimal çözümler sağladığı, PSO, ACO ve WOA'nın mühendislik tasarım problemlerinin çoğu için rekabetçi sonuçlar ürettiğini belirtmişlerdir. Zhi ve diğ. [28], içten yanmalı bir motorun titreşimini, gürültüsünü ve çarpmasını azaltarak dinamik performansını en üst düzeye çıkarmak amacıyla ACO algoritmasını kullandılar. MATLAB esas alınarak modele ilişkin optimizasyon programları geliştirilmiş ve simülasyonlarını yapmışlardır. Orijinal tasarımla karşılaştırıldığında, simülasyon sonuçları bolluk katsayısının %1,24 arttığını, dinamik maksimum pozitif ivmenin yükselişte %0,87 azaldığını ve alçalmada %5,23 arttığını ifade etmişlerdir. Savsani ve diğ. [29], düz dişli takımının toplam ağırlığını minimize etmek amacıyla optimal tasarım parametrelerini belirlemek için PSO ve Benzetimli Tavlama (*Simulated Annealing*, SA) algoritmalarını kullandılar. Yapılan çalışmada, dişli genişliği, pinyon mil çapı, dişli mil çapı, pinyon diş sayısı ve modül olmak üzere 5 tasarım değişkeni ve önceden tanımlanmış 5 kısıtlama (*pinyon ve dişli için millerin burulma mukavemeti, dişin bükülme mukavemeti, yüzey dayanıklılığı ve merkez uzaklığı*) dikkate almışlardır. Yapılan deneylerde, PSO ve SA'nın sonuçları birbirine çok yakın oluğu, GA ile elde edilen sonuçlardan ise daha iyi olduğunu gözlemlemişlerdir. Öte yandan PSO, yakınsama yoluyla optimal çözüme ulaşmak için daha az yinelemeye ihtiyaç duyduğundan SA ve GA'dan daha hızlı olduğunu beyan etmişlerdir. Abderazek ve diğ. [30] tarafından, dönen bir silindir takipçisi ile disk kam tasarımını optimize etmek için Çoklu Evren Optimizasyon Algoritması (*Multiverse Optimizer,* MVO), MFO, MBA, GWO, ALO, SSA ve ER algoritması olmak üzere yedi farklı metasezgisel teknik kullanıldı. Kam sıkışıklığını minimize etmek, performansı ve

mukavemet direncini en üst düzeye çıkarmak için tasarım problemi çok amaçlı bir optimizasyon olarak formüle etmişlerdir.

Literatür araştırmları genel olarak değerlendirildiğinde, tek veya çok amaçlı gerçek dünya mühendislik problemlerinin çözümü oldukça karmaşık hesaplar gerektirdiği, metasezgisel yöntemler sayesinde bu zorlukların üstesinden kolayca gelinebildiği görülmüştür. Ayrıca PSO, ACO, GWO gibi literatürde iyi bilinen algoritmalar, bu problemlerin çözümündeki başarılarından dolayı çoğu araştırmacının ilgi odağı haline gelmiştir. Bu konu üzerinde çalışan araştırmacılar, metasezgisel yöntemleri bir araya getirerek daha iyi sonuçlar elde etmeyi amaçlamışlardır.

3. METASEZGİSELLER

Metasezgiseller, buluşsal teknikleri kullanarak optimize edici metotlar oluşturmak için bir dizi talimatlar sunan, yüksek düzeyde ve problemden bağımsız bir çerçevedir [31]. Bu algoritmaların çoğu doğadan ilham alırken, uygunluğu artırmak amacıyla çeşitli kaynaklardan da ilham almaktadır [32]. Küresel optimumu ortaklaşa aramak için metasezgisel algoritmalar, sürüler, kuşlar, balıklar vb. gibi sosyal hayvanların ya da böceklerin kolektif zekâsını taklit eder. Bu grubun önde gelen üyeleri arasında karınca kolonisi optimizasyonu, guguk kuşu arama algoritması, parçacık sürü optimizasyonu ve yapay arı kolonisi gibi popüler yöntemler bulunmaktadır. Sürü tabanlı yaklaşımlar, olağanüstü bilgi işlem performanslarından dolayı otomotiv imalat endüstrisi, makine tasarımı, mekanik işleme, yapısal optimizasyon, havacılık ve uzay mühendisliği dahil olmak üzere birçok farklı alanda kullanılmaktadır [33]. Son yirmi yılda yapılan araştırmalar, bu alandaki uygulamaların muazzam bir şekilde büyüdüğünü göstermektedir [31].

Metasezgiseller, fiziksel tabanlı algoritmalar, sürü algoritmaları ve evrimsel algoritmalar olmak üzere üç ana kategoride incelenebilir. Diferansiyel Evrim (DE) ve Genetik Algoritma (GA) gibi evrimsel algoritmalar, güvenilir optimizasyon teknikleri geliştirmek amacıyla doğadaki evrimsel ilkeleri taklit eder. Parçacık sürüsü optimizasyonu ve balina optimizasyonu, çeşitli canlıların grup davranışlarını taklit eden sürü algoritmalarının iki örneğidir. Yerçekimi arama algoritmaları ve simüle edilmiş tavlama gibi gerçek dünyadaki fiziksel süreçler, fiziksel tabanlı algoritmalar için ilham kaynağı olarak hizmet etmektedir [34].

Bu çalışmada, gerçek dünya problemlerinin çözümünde test etmek amacıyla doğadan ilham alınan metasezgizel yöntemlerden Karınca Kolonisi Optimizasyonu, Yapay Arı Kolonisi, Salp Sürü Algoritması ve Sinüs Kosinüs Algoritması kullanıldı. Bu yöntemlere ait detaylı açıklamalar aşağıda verilmiştir.

3.1 KARINCA KOLONİSİ OPTİMİZASYONU

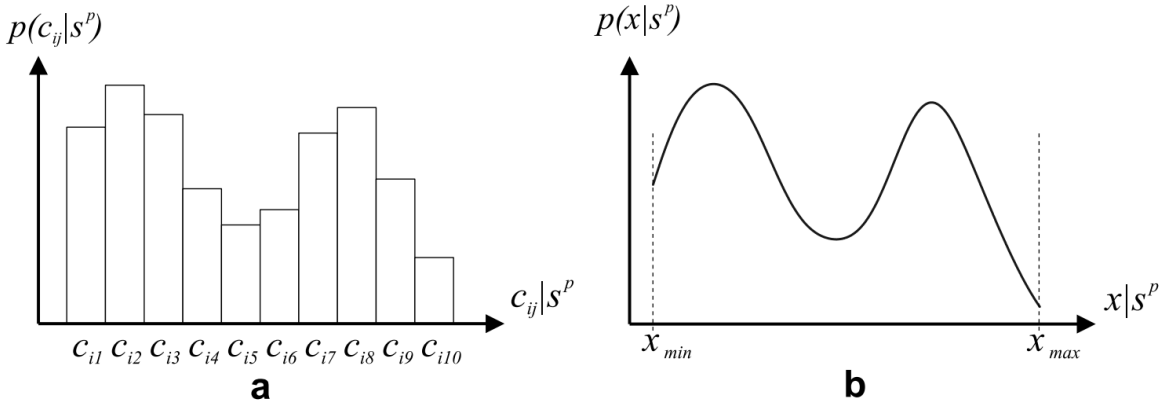
Karınca Kolonisi Optimizasyonu (*Ant Colony Optimization,* ACO) tekniği [35], ilk olarak 1990'lı yıllarda Marco Dorigo tarafından ortaya atılan karınca kolonilerinin yiyecek arama davranışlarından esinlenmiştir. Karıncalar, bireysel türler yerine topluluğun hayatta kalmasını ve sürdürülmesini tercih eden sosyal böceklerdir. Birbirleriyle ses, dokunma ve feromon kullanarak iletişim kurarlar. Feromonlar, karıncalar tarafından salgılanan ve aynı türün üyelerinde sosyal

tepkiyi tetikleyen organik kimyasal bileşiklerdir. Karıncalar yerde yaşadığından, diğer karıncaların takip edebileceği feromon izleri bırakmak için toprak yüzeyini kullanırlar.

ACO'nun temel prensibi, mümkün olan en kısa yoldan yiyecek aramak için karıncaların yuvalarından hareketlerini gözlemlemektir. Başlangıçta karıncalar, besin kaynağı bulmak için yuvalarının çevresinde rastgele hareket etmeye başlarlar. Bu rastgele arama, yuvadan yiyecek kaynağına kadar birden fazla rotanın oluşturulmasını sağlar. Besin kaynağının kalitesine ve miktarına bağlı olarak karıncalar, dönüş yolunda yiyeceğin bir kısmını gerekli feromon konsantrasyonuyla birlikte yuvaya geri taşırlar. Bu feromon denemelerine bağlı olarak, iz süren karıncaların belirli bir yolu seçme olasılığı, besin kaynağına yön veren bir faktör olacaktır. Bu olasılık feromonun buharlaşma hızına ve konsantrasyonuna bağlı olarak değişmektedir. Feromonun buharlaşma hızı belirleyici bir faktör olduğundan, besin kaynağına giden her bir yolun uzunluğunun kolaylıkla hesaplanabileceği de gözlemlenebilir.

Socha ve Dorigo [36], 2008 yılında yaptıkları çalışmada, başlangıçta kombinatoryal optimizasyon için metasezgisel olarak geliştirilen ACO'nun yapısında herhangi bir büyük kavramsal değişiklik olmadan sürekli optimizasyona (ACO_R) nasıl uyarlanabileceğini gösterdiler.

Kombinatoryal optimizasyon için ACO'da feromon bilgisi bir tablo olarak saklanmaktadır. Şekil 3.1(a)'da gösterildiği gibi her yinelemede, mevcut kısmi çözüme eklenecek bir bileşeni seçerken, bir karınca bu tablodaki bazı değerleri ayrık olasılık dağılımı olarak kullanır. Sürekli optimizasyon durumunda ise Şekil 3.1 (b)'de gösterildiği gibi karıncaların yaptığı seçim sonlu bir kümeyle sınırlı değildir. Dolayısıyla feromonu tablo şeklinde göstermek mümkün değildir.



Şekil 3-1: Olasılık dağılımları; (a) Ayrık olaslık dağılımı, (b) Sürekli olaslık dağılımı [36].

Bu sorunun üstesinden gelmek için Guntsch ve Middendorf [37] tarafından önerilen Nüfusa Dayalı Karınca Kolonisi Optimizasyonunda (*Population-based ant colony optimization*, Pb-ACO) feromon tablosu, tıpkı normal ACO'da olduğu gibi, bulunan iyi çözümlerin bileşenlerine göre güncellenir. Ancak geleneksel ACO'da karıncalar tarafından bulunan gerçek çözümler, feromon tablosu güncellendikten sonra atılır. Buna karşılık Pb-ACO algoritması, feromon tablosunu güncellemek için kullanılan belirli sayıda çözümün kaydını tutar. Feromon buharlaştırmayı kullanmak yerine, en eski çözümlerle ilişkili feromon, nihayetinde feromon tablosunda negatif bir güncelleme yapılarak ortadan kaldırılır, böylece etkisi iptal edilir.

ACO algoritması, parametrelerin belirlenmesiyle başlar ve her bir yinelemede feromonları günceller. ACO, tüm karıncaların çözümler oluşturduğu bir ana döngü boyunca yinelenir. Karıncaların *i*'den *j*'ye yer değiştirmesi, Eşitlik 3.1'de tanımlandığı gibi olasılıksal bir yaklaşım kullanılarak yapılır.

$$P^k\left(c_{ij}\right)=\begin{cases}\tau_{ij}^{\alpha}\cdot\eta\left(c_{ij}\right)^{\beta}\Big/\sum_{c_{ij}\in N_i^k}\tau_{ij}^{\alpha}\cdot\eta\left(c_{ij}\right)^{\beta}, & if\ \forall c_{ij}\in N_i^k\\ 0, & değilse\end{cases} \tag{3.1}$$

Burada $\eta(.)$, her bir adımda uygun çözüm elemanlarına ($c_{ij} \in N_i^k$) atanan buluşsal bir değerdir. α ve β, feromon bilgisi ile buluşsal bilginin ilişki etkisini belirleyen iki pozitif parametredir. N_i^k ise karınca k'nin henüz seçmediği n elemandan oluşan sonlu bir kümedir ve τ_{ij} ise c_{ij} elemanıyla ilişkili feromon değeri olup, τ_{ij} Eşitlik 3.2'deki gibi hesaplanır.

$$\tau_{ij}^{t+1}=\begin{cases}(1-\rho)\cdot\tau_{ij}^{t}+\rho\sum_{k=1}^{m}\Delta\tau_{ij}^{t,k}, & if\ \tau_{ij}\in s_{ch}\\ (1-\rho)\cdot\tau_{ij}^{t}, & değilse\end{cases} \tag{3.2}$$

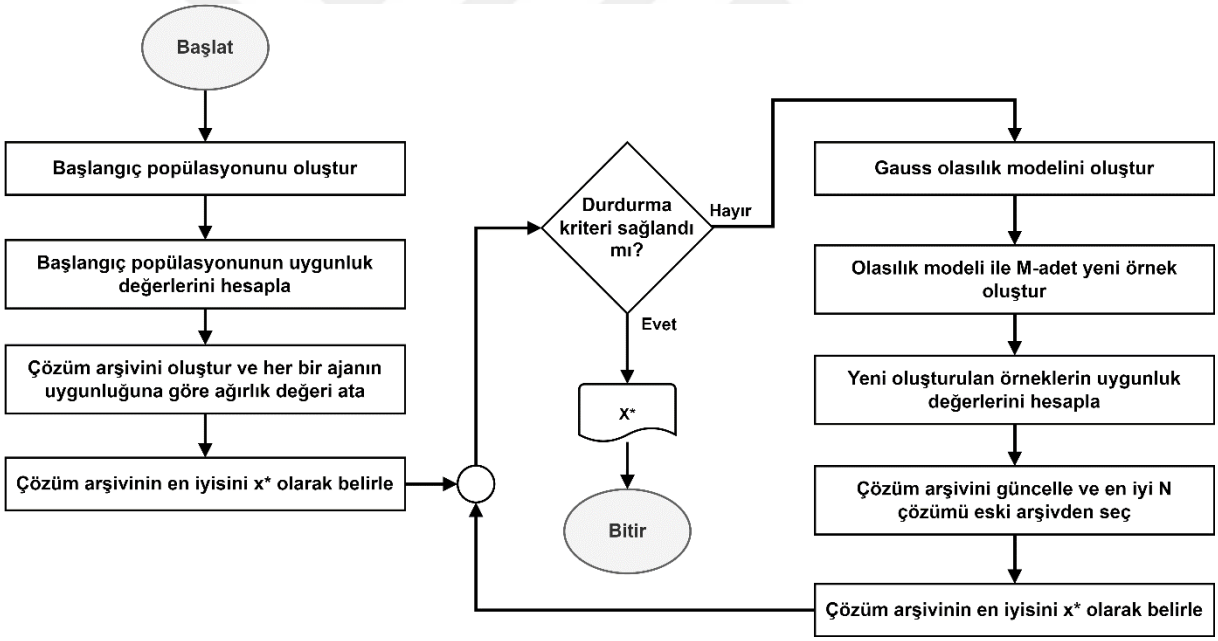
Burada $0 < \rho \leq 1$ feromonun buharlaşma hızıdır. Algoritmanın çok hızlı yakınsamasını önlemek için ρ parametresi kullanılır. $\Delta\tau_{ij}^{t,k}$ seçtiği k karıncası üzerinde biriken feromon miktarıdır. Son olarak, s_{ch} ise seçilen iyi çözümlerin kümesidir.

ACO'da karıncalar, her aşamada olasılıksal bir seçim yaparlar. Yani, ayrık bir olasılık dağılımı oluşturulur. Ancak bu sürekli optimizasyon problemleri için uygun bir yapıda değildir. Socha ve Dorigo tarafından önerilen ACO_R'daki temel fikir, ayrık bir olasılık dağılımı yerine sürekli bir Olasılık Yoğunluk Fonksiyonu (*Probability Density Function*, PDF) kullanmaya

geçiştir [38]. Bu amaçla, en popüler fonksiyonlardan biri olan Gauss fonksiyonu kullanılmaktadır. Bununla birlikte, tek bir Gauss fonksiyonu, yalnızca bir maksimuma sahip olduğundan, arama uzayının iki ayrı alanının umut verici olduğu bir durumu tanımlayamaz. Bu gerçeğe bağlı olarak, Socha ve Dorigo [36] Gauss çekirdeğini Eşitlik 3.3'te gösterildiği gibi birkaç tek boyutlu Gauss fonksiyonunun $g(x)_j^i$ ağırlıklı toplamı olarak tanımladı.

$$G^i(x) = \sum_{j=1}^{k} \omega_j g_j^i(x) = \sum_{j=1}^{k} \omega_j \frac{1}{\sigma_j^i \sqrt{2\pi}} \exp\left(-\frac{\left(x-\mu_j^i\right)^2}{2\left(\sigma_j^i\right)^2} \right) \tag{3.3}$$

Burada ω, tek Gauss fonksiyonlarıyla ilişkili ağırlıkların vektörünü, μ^i ortalamaların vektörünü ve σ^i standart sapmaların vektörünü temsil etmektedir. Böyle bir PDF, kolay örneklemeye izin verir ve tek bir Gauss fonksiyonuyla karşılaştırıldığında olası şekil açısından çok daha gelişmiş bir esnekliğe sahiptir. Şekil 3.2'de ACO_R'nin akış şeması gösterilmektedir.



Şekil 3-2: ACO_R'nin akış şeması.

3.2 YAPAY ARI KOLONİSİ

Yapay Arı Kolonisi (*Artificial Bee Colony*, ABC) algoritması [39], Derviş Karaboğa tarafından geliştirilen evrimsel bir yaklaşımdır. Bu algoritma, kaliteli bir besin (nektar) kaynağı ararken bal arılarının yiyecek arama davranışından ilham almıştır. ABC algoritmasında yiyecek

konumlarının bir popülasyonu vardır ve yapay arılar bu yiyecek konumlarını zamanla değiştirir. Algoritma, en uygun çözümü bulmak için bal arıları adı verilen bir dizi hesaplama aracısını kullanır [40].

ABC algoritmasında, bir kolonideki arılar üç gruba ayrılır: işçi arılar, gözcü arılar ve izci arılar. Her yiyecek kaynağı için yalnızca bir işçi arı vardır. Yani işçi arı sayısı yiyecek kaynağı sayısına eşittir. Yiyecek (nektar) alanından çıkarılan bir işçi arı, yeni nektar kaynaklarını rastgele arayan bir izci arı olmaya zorlanır. İşçi arılar, kovandaki gözcü arılarla bilgi paylaşırlar, böylece gözcü arılar yiyecek aramak için bir besin kaynağı seçebilirler [41]. Arama uzayındaki her çözüm, bir nektar kaynağı konumunu temsil eden bir dizi optimizasyon parametresinden oluşur. Bir besin kaynağının kalitesine "uygunluk değeri" denir ve konumuyla ilişkilendirilir.

Algoritmada görevli arılar, yiyecek kaynaklarını araştırmaktan (uygunluk değerlerini kullanarak) ve gözcü arıları işe almak için bilgiyi paylaşmaktan sorumlu olacaktır. Görevli veya gözcü arıların sayısı popülasyondaki çözüm sayısına (n) eşittir. Her çözüm (besin kaynağı) $x_i(i = 1,2, \dots, n)$ D-boyutlu bir vektördür. Gözcü arılar bu bilgiye göre besin kaynağı seçimine karar vereceklerdir. Daha kaliteli bir besin kaynağının gözcü arılar tarafından seçilme olasılığı daha yüksek olacaktır. Bir arı sürüsünün arama ve en sonunda en iyi bilinen besin kaynağını seçme süreci, en uygun çözümü bulmak için kullanılan süreçtir. Bir gözcü arı, Eşitlik 3.4'teki gibi yiyecek kaynağıyla ilgili olasılık değerine bağlı olarak bir yiyecek kaynağı seçer.

$$p_i = fit_i \left(\sum_{q=1}^{n} fit_q \right)^{-1} \tag{3.4}$$

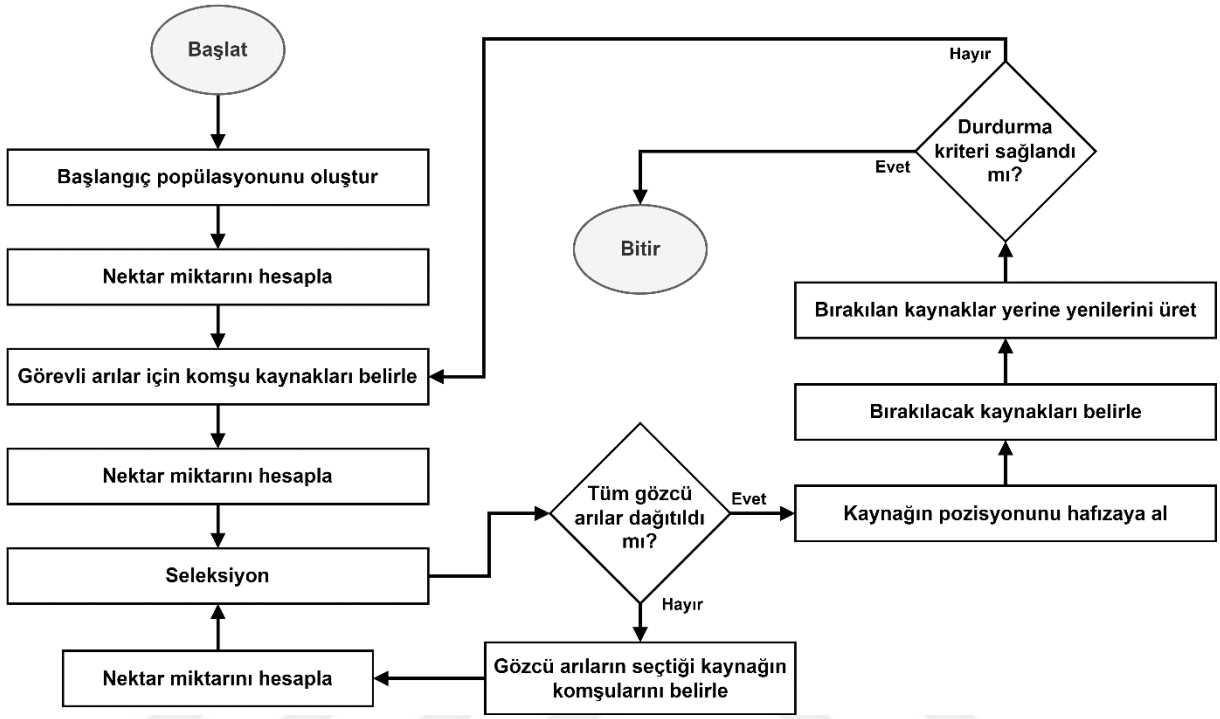
Burada fit_i, işçi arı tarafından değerlendirilen i-inci çözümün, i-inci konumundaki yiyecek kaynağının nektar miktarıyla orantılı uygunluk değeridir. SN, işçi arının (BN) sayısına eşit olan yiyecek kaynağı sayısıdır. Bu sayede işçi arılar izci arılarla bilgi alışverişinde bulunurlar. ABC, bırakılan kaynaklardan aday yiyecek konumu üretmek için Eşitlik 3.5 kullanılır.

$$x_{ij}^{*} = x_{ij} + \phi_{ij} \left(x_{ij} - x_{lj} \right) \tag{3.5}$$

Burada x_{ij}^{*}, önceki besin kaynağı x_{ij} ile rastgele seçilen besin kaynağını karşılaştırarak seçilen yeni uygun besin kaynağıdır, $l \in \{1,2, \dots, SN\}$ ve $j \in \{1,2, \dots, D\}$ rastgele seçilen

endekslerdir ve ϕ_{ij}, eski besin kaynağını bir sonraki yinelemede yeni besin kaynağı olacak şekilde ayarlamak için kullanılan $[-1, 1]$ arasındaki rastgele bir sayıdır.

Yapay Şekil 3.3'te Arı Kolonisi algoritmasının akış şeması [42] gösterilmektedir.



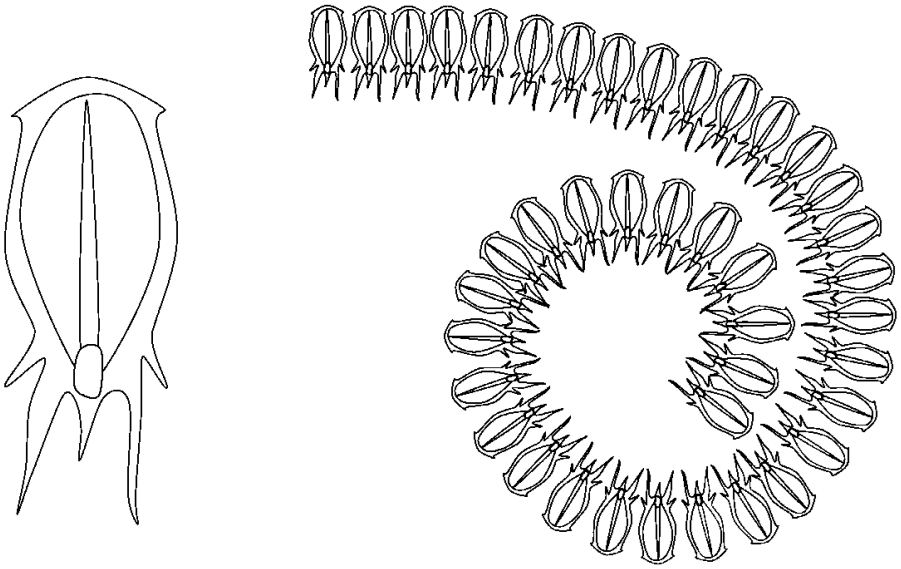
Şekil 3-3: Yapay Arı Kolonisi algoritmasının akış şeması.

3.3 SALP SÜRÜ ALGORİTMASI

Salp Sürü Algoritması (*Salp Swarm Algorithm,* SSA) [43], 2017 yılında Mirjalili ve ark. tarafından mühendislik tasarım problemlerinin çözümü için geliştirilen bir optimizasyon algoritmasıdır. Algoritmanın ilham kaynağı, salpların okyanuslarda gezinirken ve yiyecek ararken gerçekleştirmiş olduğu sürü davranışıdır. Salp, Salpidae familyasına ait olup şeffaf fıçı şeklinde gövdeye sahiptirler. Dokuları denizanalarına oldukça benzerdir. İleri yönlü hareket etmek için suyun vücuda pompalandığı denizanasına çok benzer şekilde hareket etmektedirler. Şekil 4(a)'da bir salpın temsili bir şekli gösterilmektedir.

Bu deniz canlısıyla ilgili biyolojik araştırmalar, yaşam ortamlarına erişimin ve laboratuvar ortamlarında muhafaza etmenin son derece zor olması nedeniyle henüz başlangıç aşamasındadır. Salplar, genellikle derin okyanuslarda "salp zinciri" olarak isimlendirilen bir sürü meydana getirir. Şekil 3.4'te bu salp zinciri gösterilmektedir. Salpların davranışının gerçek

sebebi henüz çok açık değildir, ancak bazı araştırmacılar bunun hızlı koordineli değişiklikler veya yiyecek arama stratejisi kullanarak daha iyi hareket edebilme becerisi kazanmak için yapıldığına inanmaktadır.



Şekil 3-4: Salp ve salp sürüsü veya zinciri.

Salp zincirlerinin matematiksel modelinde, lider ve takipçiler olmak üzere sürü (popülasyon) iki gruba ayrılır. Lider zincirin ön kısmındaki salptır, geri kalanlar ise takipçi salplar olarak belirlenir. Sürünün davranışı olarak, lider sürüyü yönlendirir ve takipçiler de birbirlerini izler (takip eder).

Diğer sürü tabanlı tekniklere benzer şekilde salpların konumu, n'nin belirli bir problemin değişken sayısı olduğu n boyutlu bir arama uzayında tanımlanır. Algoritmada, salpların konumları x olarak isimlendirilen 2 boyutlu bir matriste tutulur. Ayrıca, arama uzayında F adı verilen bir besin kaynağının salp sürüsünün hedefi olduğu varsayılmaktadır. Liderin konumunu güncellemek için Eşitlik 3.6 önerilmektedir:

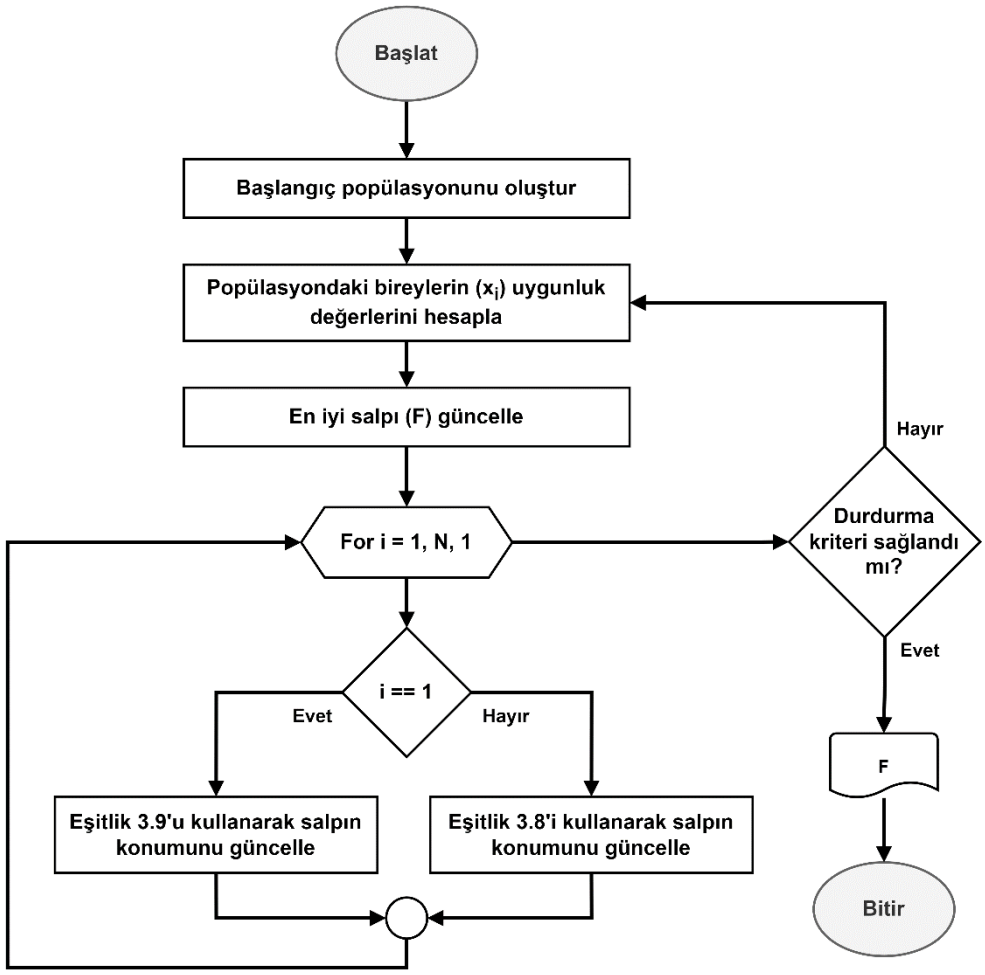
$$x_j^1 = \begin{cases} F_j + c_1\left(c_2\left(ub_j - lb_j\right) + lb_j\right), & c_3 \geq 0 \\ F_j - c_1\left(c_2\left(ub_j - lb_j\right) + lb_j\right), & c_3 < 0 \end{cases} \tag{3.6}$$

Burada x_j^1, j-inci boyuttaki ilk salpın (liderin) konumunu, F_j, besin kaynağının j-inci boyuttaki konumunu, lb_j ve ub_j sırasıyla j-inci boyutun alt ve üst sınır değerlerini, c_1, c_2 ve c_3

ise rastgele sayılardır. Bunlardan c_1 katsayısı, SSA'daki en önemli parametredir çünkü Eşitlik 3.7'deki gibi tanımlanan keşif ve sömürüyü dengeler:

$$c_1 = 2\exp\left(-\left(4t/T\right)^2\right) \tag{3.7}$$

Burada t geçerli yinelemedir ve T maksimum yineleme sayısıdır. c_2 ve c_3 parametreleri, $[0, 1]$ aralığında düzgün biçimde üretilen rastgele sayılardır. Aslında j-inci boyuttaki bir sonraki konumun adım boyutunun yanı sıra pozitif sonsuza mı yoksa negatif sonsuza mı doğru olacağını belirlerler. SSA'nın akış diyagramı Şekil 3.5'te gösterilmektedir.



Şekil 3-5: Salp Sürü Algoritmasının Akış Şeması.

Takipçilerin konumunu güncellemek için Eşitlik 3.8'de gösterildiği gibi Newton'un hareket yasası kullanılır.

$$x_j^i = \frac{1}{2}at^2 + v_0 t \tag{3.8}$$

Burada $i \geq 2$ olduğunda x_j^i, i-inci boyuttaki takipçi salp'ın konumunu gösterir, t zamandır, v_0 başlangıç hızıdır ve $a = v_{son}/v_0$ burada $v = (x - x_0)/t$. Optimizasyonda zaman kavramı yineleme (iterasyon) olarak ele alındığından, iterasyonlar arasındaki fark 1'e eşit olur ve $v_0 = 0$ dikkate alındığında bu denklem Eşitlik 3.9'daki gibi ifade edilebilir.

$$x_j^i = \frac{1}{2}\left(x_j^i - x_j^{i-1}\right) \tag{3.9}$$

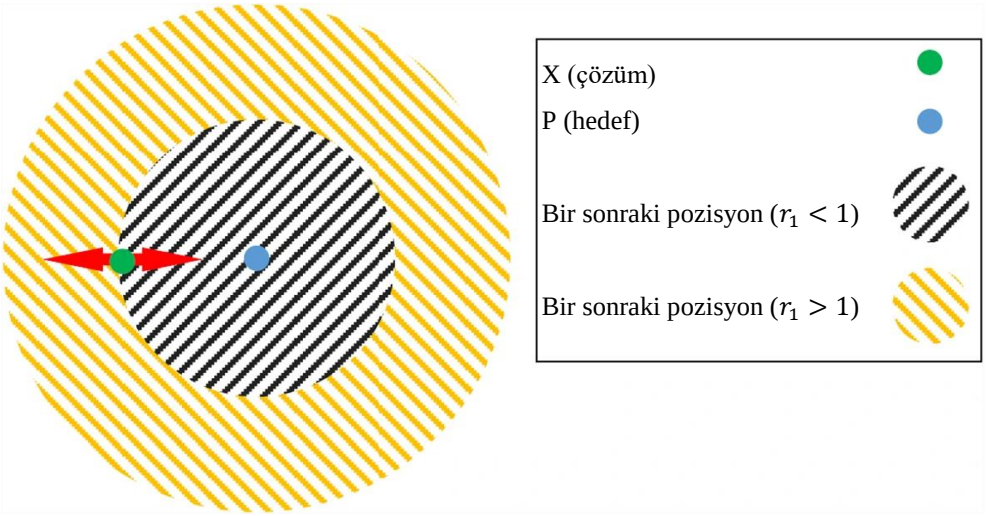
3.4 SİNÜS KOSİNÜS ALGORİTMASI

Sinüs Kosinüs Algoritması (*Sine Cosine Algorithm,* SCA), Mirjalili [44] tarafından çeşitli optimizasyon problemlerini çözmek için tanıtılan popülasyon tabanlı bir optimizasyon algoritmasıdır. SCA, rastgele başlangıç çözümleri üretir, sinüs ve kosinüs fonksiyonlarına dayanan matematiksel bir model kullanarak en iyi çözüme doğru ilerler. Arama uzayının araştırılması için çeşitli rastgele ve uyarlanabilir değişkenler bu algoritmayla birleştirilir. SCA'da iki ana arama stratejisiyle (küresel keşif ve yerel sömürü) etkili bir şekilde aramaya uygun zeki bir algoritma üretmek için SCA'da iki ana teknik (popülasyon arama stratejisi ve yerel arama stratejisi) sağlanmaktadır. Benzer metasezgisel algoritmalarla ilgili olarak SCA kolay, esnek ve basit bir şekilde uygulanır; bu nedenle çeşitli optimizasyon problemlerini çözmek için uygulanabilir. SCA'daki çözümler aşağıdaki denklemler kullanılarak güncellenir:

$$x_{ij}^{t+1} = x_{ij}^t + r_1 \cdot \sin\left(r_2\right) \cdot \left| r_3 P_{ij}^t - x_{ij}^t \right| \tag{3.10}$$

$$x_{ij}^{t+1} = x_{ij}^t + r_1 \cdot \cos\left(r_2\right) \cdot \left| r_3 P_{ij}^t - x_{ij}^t \right| \tag{3.11}$$

Sinüs ve kosinüs fonksiyonlarının Eşitlik 3.10 ve 3.11 üzerindeki etkileri, Şekil 3.6'da gösterilmektedir.



Şekil 3-6: Sinüs ve kosinüs fonksiyonlarının bir sonraki pozisyona etkileri [45].

Genel olarak yukarıda belirtilen eşitlikler, Eşitlik 3.12'de gösterildiği gibi kullanılmak üzere birleştirilir.

$$x_{ij}^{t+1} = \begin{cases} x_{ij}^{t} + r_1 \cdot \cos(r_2) \cdot \left| r_3 P_{ij}^{t} - x_{ij}^{t} \right|, & r_4 \geq 0.5 \\ x_{ij}^{t} + r_1 \cdot \sin(r_2) \cdot \left| r_3 P_{ij}^{t} - x_{ij}^{t} \right|, & r_4 < 0.5 \end{cases} \tag{3.12}$$

Burada x_{ij}^{t}, d'inci boyuttaki t yinelemesindeki mevcut i bireyini temsil etmektedir. P_{ij}^{t}, d'inci boyuttaki t yinelemesindeki en iyi bireyin konumunu gösterir. Son olarak, r_1, r_2, r_3 ve r_4 rastgele parametrelerdir. Bu parametreler, yerel optimumlara düşmekten kaçınmak, keşifsel ve sömürücü arama modellerini dengelemek için dâhil edilmiştir.

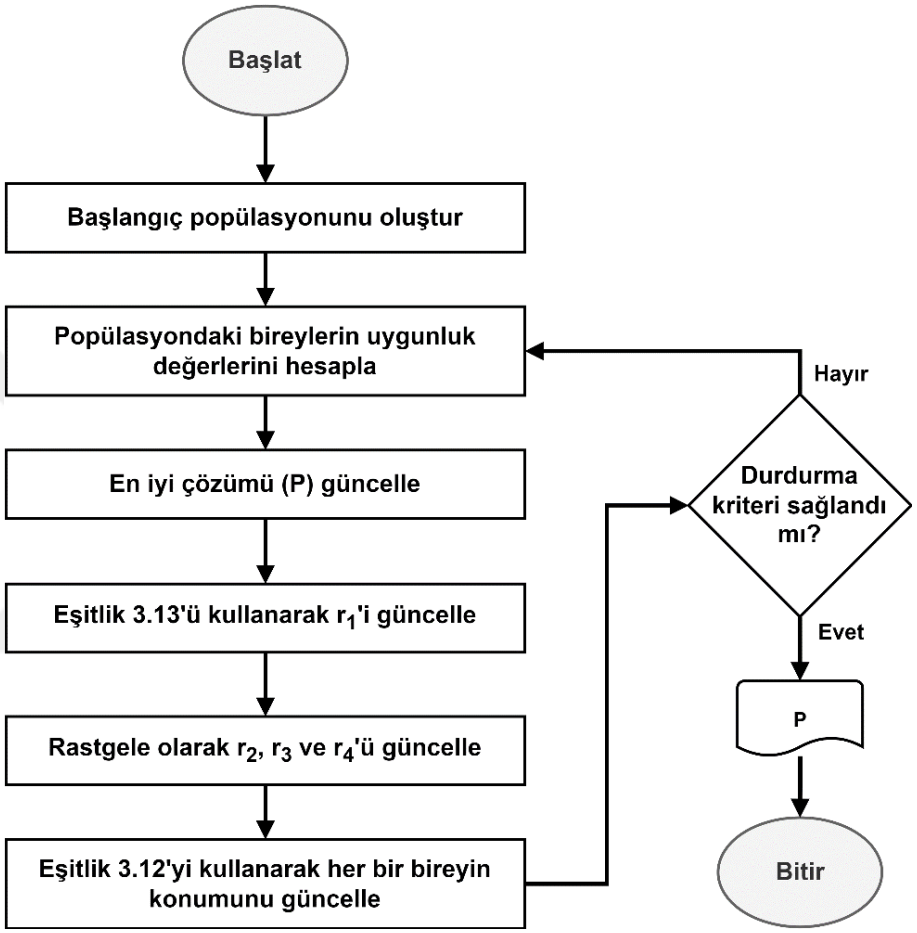
r_1 parametresi, bir çözümün konumunu en iyi çözüme doğru mu ($r_1 < 1$) yoksa dışarıya doğru mu ($r_1 > 1$) güncelleyeceğine karar verir. Keşifsel ve sömürücü arama modellerini dengelemek için r_1'in önceden belirlenmiş bir sabitten 0'a doğrusal olarak azalır. Aşağıdaki denklem kullanılarak güncellenir:

$$r_1 = a - t\left(a / T_{\max}\right) \tag{3.13}$$

Burada a bir sabiti, t geçerli yinelemeyi ve T_{max} ise izin verilen maksimum yinelemeyi temsil etmektedir. Eşitliklerdeki r_2 parametresi, bir çözümün hedefe doğru hareketinin ne kadar büyük olduğunu belirleyen $[0, 2\pi]$ aralığında ayarlanır. Diğer bir rastgele parametre olan r_3 ise hedefe $[0, 2]$ aralığında rastgele bir ağırlık değeri atar. Bu, diğer çözümlerin konum

güncellemesinin hedefinin etkisinin vurgulanmasına ($r_3 > 1$) veya vurgusunun kaldırılmasına ($r_3 < 1$) izin verir. Son olarak, r_4 rastgele parametresi $[0, 1]$ aralığındadır ve Eşitlik 3.12'deki sinüs kosinüsün trigonometrik fonksiyonları arasında seçim yapmak için bir anahtar görevi görür.

Sinüs Kosinüs Algoritmasının akış şeması Şekil 3.7'de gösterilmektedir.



Şekil 3-7: Sinüs Kosinüs Algoritmasının akış şeması.

4. GERÇEK DÜNYA PROBLEMLERİ

Metasezgisel algoritmaların en önemli başarılarından biri, iyi bilinen kısıtlı matematik ve mühendislik tasarım problemlerine daha iyi çözümler sağlamadaki etkinlikleridir. Bu zorluklar, metasezgisellerin ilham verici konsepte dayalı olarak geliştirilen güçlü matematiksel modellerden yararlanılmasıyla başarılabilir [46].

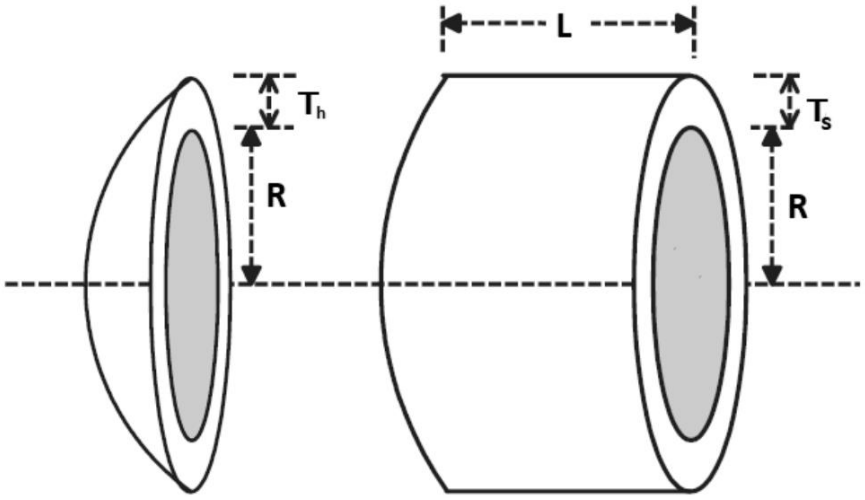
Gerçek dünyada karşılaşılan mühendislik tasarım problemleri genellikle birbiriyle çelişen birçok amacın varlığıyla karakterize edilir. Bu nedenle mühendislik tasarım problemini çok amaçlı bir optimizasyon problemi olarak düşünmek mantıklıdır.

Mühendislik tasarım problemleri çoğunlukla sayısal simülasyonlar, analitik hesaplamalar ve katalog seçimlerinin bir karışımından oluştuğundan, uygunluk/amaç fonksiyonunun türevlerini hesaplamanın kolay bir yolu yoktur. Sonuç olarak, bu tür problemler için gradyansız optimizasyon teknikleri daha uygun olacaktır [47].

Bu bölümde, tez çalışması kapsamında ele alınan metasezgisellerin performanslarını değerlendirmek amacıyla, IEEE Congress on Evolutionary Computation (CEC)'in arşivinden seçilen Makine Mühendisliği alanındaki sekiz farklı gerçek dünya problemi kullanılmıştır. Bunlar, Basınçlı Kap Tasarımı, Robot Kavrayıcı Problemi, Makaralı Rulman Tasarım Problemi, Kademeli Konik Kasnak Problemi, Germe Sıkıştırma Yayı Tasarımı, Üç Çubuklu Kafes Kiriş Tasarım Problemi, Hız Düşürücüsünün Ağırlık Minimizasyonu ve Kaynaklı Kiriş Tasarımı problemidir. Bu problemlerin tanıtımı aşağıda ayrıntılı olarak sunulmuştur.

4.1 BASINÇLI KAP TASARIMI

Bu tasarım problemi, basınçlı bir kap için toplam maliyetin en aza indirilmesini gerektirir. Problem, doğrusal olmayan bir uygunluk fonksiyonuna, üçü doğrusal (linear) ve biri doğrusal olmayan (non-linear) eşitsizlik kısıtlaması içermektedir. Buna ek olarak, ikisi ayrık ve ikisi sürekli olmak üzere dört adet tasarım değişkeni vardır. İlk iki ayrık değişken 0,0625'in katları olan değerleri elde edebilir [48]. Problemin şematiği Şekil 4.1'de gösterilmektedir.



Şekil 4-1: Basınçlı kap tasarımı problemi [49].

Basınçlı kap tasarımı probleminin uygunluk fonksiyonu Eşitlik 4.1'de verilmiştir.

$$f\left(\vec{x}\right)=1.7781z_2x_3^2+0.6224z_1x_3x_4+3.1661{z_1}^2x_4+19.84{z_1}^2x_3 \tag{4.1}$$

Problemdeki kısıtlar Eşitlik 4.2'de verilmiştir.

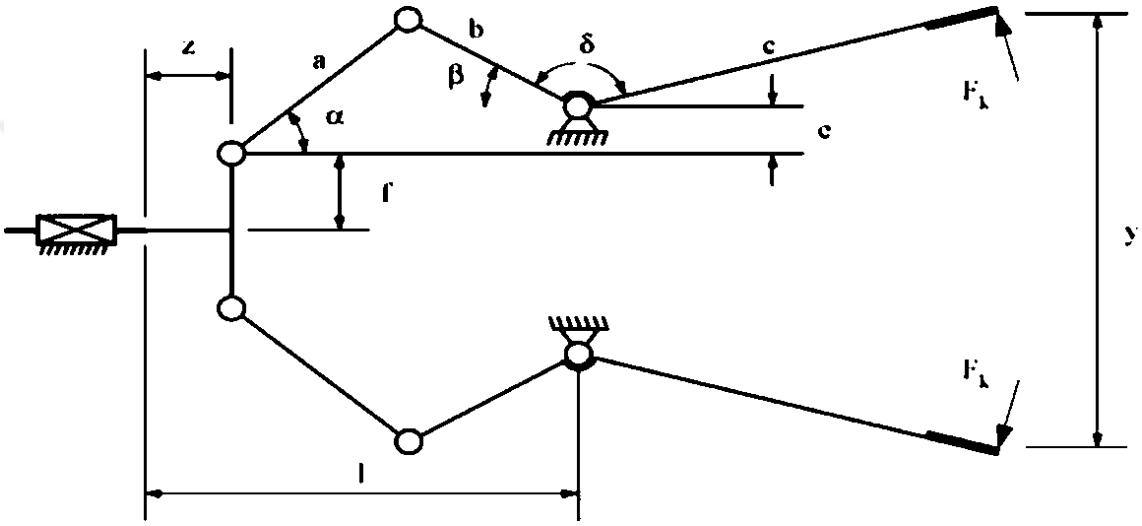
$$\begin{aligned} &g_1\left(\vec{x}\right)=0.000954x_3\le z_2,\\ &g_2\left(\vec{x}\right)=0.0193x_3\le z_1,\\ &g_3\left(\vec{x}\right)=x_4\le 240,\\ &g_4\left(\vec{x}\right)=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3\le -1296000. \end{aligned} \tag{4.2}$$

Eşitlik 4.3'te bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$\begin{aligned} &z_1=0.0625x_1,\\ &z_2=0.0625x_2\\ &10\le x_4,x_3\le 200,\\ &1\le x_1,x_2\le 99,\quad x_1,x_2\in\mathbb{Z}^+ \end{aligned} \tag{4.3}$$

4.2 ROBOT KAVRAYICI PROBLEMİ

Robot Kavrayıcı Problemi (RKP), yedi sürekli tasarım değişkeni ve geometrik boyutlar dikkate alınarak, bir kavrayıcı ucu yer değiştirme aralığı için bir kavrayıcı tarafından uygulanan maksimum ve minimum kuvvet arasındaki farkın en aza indirilmesini gerektirir. Tasarlanan robot kavrayıcıda optimal çözüm alanı, toplam arama alanının neredeyse %2,5'unu oluşturduğu altı farklı geometrik kısıtlama bulunmaktadır [48]. Robot kavrayıcı probleminin şematiği Şekil 4.2'de gösterilmektedir.



Şekil 4-2: Robot Tutucu Probleminin Şematiği [48].

Robot kavrayıcı probleminin uygunluk fonksiyonu Eşitlik 4.4'te verilmiştir.

$$f\left(\vec{x}\right) = -\min_{z} F_k\left(x,z\right) + \max_{z} F_k\left(x,z\right) \tag{4.4}$$

Problemdeki kısıtlar Eşitlik 4.5'te verilmiştir.

$$
\begin{aligned}
&g_1(\vec{x}) = -Y_{\min} + y(\vec{x}, Z_{\max}) \le 0,\\
&g_2(\vec{x}) = -y(x, Z_{\max}) \le 0,\\
&g_3(\vec{x}) = Y_{\max} - y(\vec{x}, 0) \le 0,\\
&g_4(\vec{x}) = y(\vec{x}, 0) - Y_G \le 0,\\
&g_5(\vec{x}) = l^2 + e^2 - (a+b)^2 \le 0,\\
&g_6(\vec{x}) = b^2 - (a-e)^2 - (l - Z_{\max})^2 \le 0,\\
&g_7(\vec{x}) = Z_{\max} - l \le 0
\end{aligned}
\tag{4.5}
$$

Bazı değişkenlerin alabileceği değerler ve sabit ifadeleri Eşitlik 4.6 ile 4.11 arasında verilmiştir.

$$\alpha = \cos^{-1}\left(\frac{a^2 + g^2 - b^2}{2ag}\right) + \phi, \;\; g = \sqrt{e^2 + (z-1)^2}, \tag{4.6}$$

$$\beta = \cos^{-1}\left(\frac{b^2 + g^2 - a^2}{2bg}\right) - \phi, \;\; \phi = \tan^{-1}\left(\frac{e}{l-z}\right), \tag{4.7}$$

$$y(x,z) = 2\left(f + e + c\sin(\beta + \delta)\right), \;\; F_k = \frac{Pb\sin(\alpha+\beta)}{2c\cos(\alpha)}, \quad Y_{\min} = 50, \tag{4.8}$$

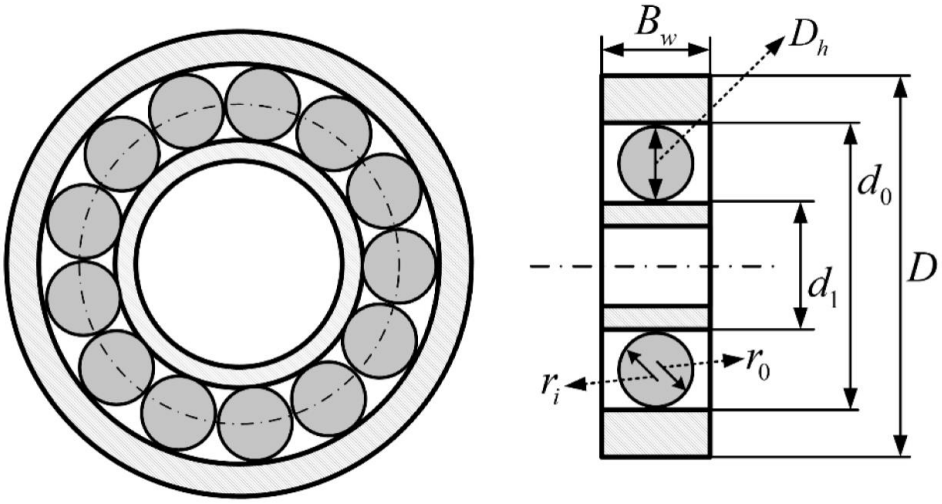
$$Y_{\max} = 100, \;\; Y_G = 150, \;\; Z_{\max} = 100, \;\; P = 100, \tag{4.9}$$

$$0 \le e \le 50, \;\; 100 \le c \le 200, \;\; 10 \le f, a, b \le 150, \tag{4.10}$$

$$1 \le \delta \le 3.14, \;\; 100 \le l \le 300. \tag{4.11}$$

4.3 MAKARALI RULMAN TASARIM PROBLEMİ

Makaralı Rulman Tasarım Problemi (MRTP)'nin temel amacı, yuvarlanan elemanlı rulmanın dinamik yük taşıma kapasitesini maksimuma çıkarmaktır. Bu optimum tasarımda, hatve çapı (D_m), bilya çapı (D_b), bilya sayısı (Z), iç ve dış yuvarlanma yolu eğrilik yarıçapı katsayısı (f_i ve f_o), $Kdmin$, $Kdmax$, δ, e ve ζ olmak üzere toplam 10 geometrik değişkenin dikkate alınması gerekir. MRTP'nin dokuz kısıtı vardır ve şematiği Şekil 4.3'te gösterilmektedir.



Şekil 4-3: Makaralı rulman tasarım probleminin şematik görünümü [50].

Makaralı rulman tasarım probleminin uygunluk fonksiyonu Eşitlik 4.12'de verilmiştir.

$$f(\vec{x}) = \begin{cases} f_c Z^{2/3} D_b^{1.8} & , \text{ if } D_b \le 25.4mm \\ 3.647 f_c Z^{2/3} D_b^{1.4} & , \text{ otherwise} \end{cases} \tag{4.12}$$

Probleme ait kısıt fonksiyonları Eşitlik 4.13'te verilmiştir.

$$\begin{aligned} g_1(\vec{x}) &= Z - \frac{\phi_0}{2\sin^{-1}(D_b / D_m)} - 1 \le 0, \\ g_2(\vec{x}) &= K_{D\min}(D-d) - 2D_b \le 0, \\ g_3(\vec{x}) &= 2D_b - K_{D\max}(D-d) \le 0, \\ g_4(\vec{x}) &= D_b - {}_w \le 0, \\ g_5(\vec{x}) &= 0.5(D+d) - D_m \le 0, \\ g_6(\vec{x}) &= D_m - (0.5+e)(D+d) \le 0, \\ g_7(\vec{x}) &= \varepsilon D_b - 0.5(D - D_m - D_b) \le 0, \\ g_8(\vec{x}) &= 0.515 - f_i \le 0, \\ g_9(\vec{x}) &= 0.515 - f_0 \le 0, \end{aligned} \tag{4.13}$$

Bazı değişkenlerin alabileceği değerler ve sabit ifadeleri Eşitlik 4.14 ile 4.17 arasında verilmiştir.

$$f_c = 37.91\left\{1+\left\{1.04\left(\frac{1-\frac{D_b\cos(\alpha)}{D_m}}{1+\frac{D_b\cos(\alpha)}{D_m}}\right)^{1.72}\left(\frac{f_i\left(2\frac{r_0}{D_{b\,0}}-1\right)}{f_0\left(2\frac{r_i}{D_{b\,i}}-1\right)}\right)^{0.41}\right\}^{10/3}\right\}^{-0.3}, \tag{4.14}$$

$$\phi_0 = 2\pi - 2\times\cos^{-1}\left(\frac{\left\{\frac{D-d}{2}-\frac{3T}{4}\right\}^2+\left\{\frac{D}{2}-\frac{T}{4}-D_b\right\}^2-\left\{\frac{d}{2}+\frac{T}{4}\right\}^2}{2\left\{\frac{D-d}{2}-3\frac{T}{4}\right\}\left\{\frac{D}{2}-\frac{T}{4}-D_b\right\}}\right)$$

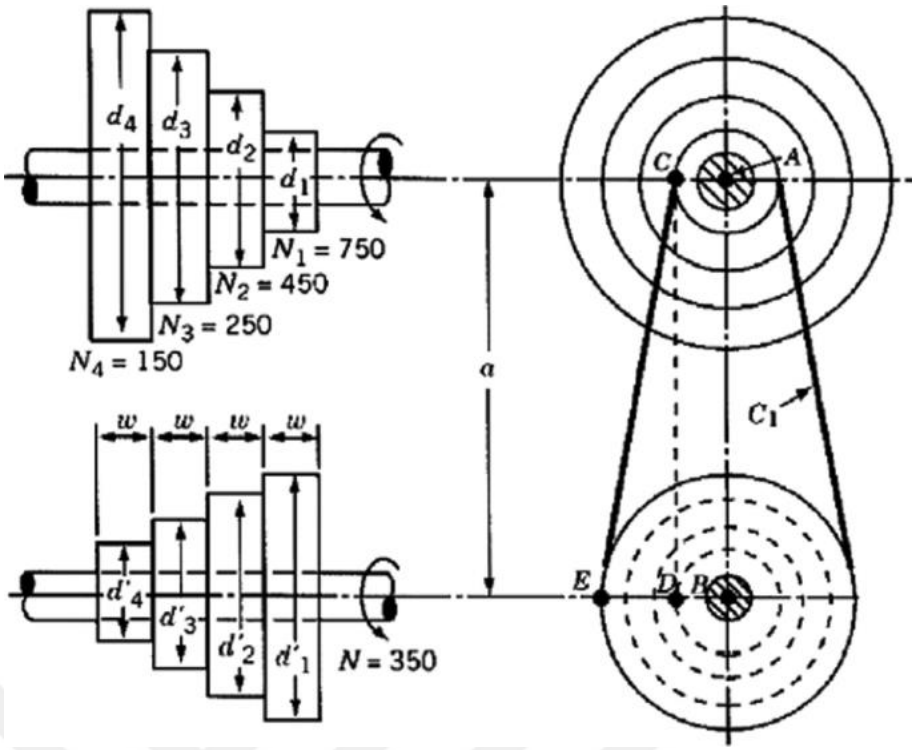
(4.15)

$$T = D-d-2D_b,\ D=160,\ d=90,\ B_w=30 \tag{4.16}$$

$$\begin{aligned}
&0.5(D+d)\le D_m\le 0.6(D+d),\\
&0.15(D-d)\le D_b\le 0.45(D-d),\\
&4\le Z\le 50,\\
&0.515\le f_i\le 0.6,\\
&0.4\le K_{D\min}\le 0.5,\\
&0.6\le K_{D\max}\le 0.7,\\
&0.3\le \varepsilon\le 0.4,\\
&0.02\le e\le 0.1,\\
&0.6\le \zeta\le 0.85,
\end{aligned} \tag{4.17}$$

4.4 KADEMELİ KONİK KASNAK PROBLEMİ

Kademeli Koni Kasnak Problemi (KKKP), kademeli bir koni oluşturan bir dizi makaradır. Millerin hız oranlarını değiştirmek için çift olarak kullanılır. Kasnaklar, üzerinde çalışan kayışlar veya halatlar aracılığıyla gücü bir şafttan diğerine önemli bir mesafede iletmek için kullanılır. Problemin temel amacı, beşincisi genişlik (w) olmak üzere her kademenin çapları için dört tasarım değişkeninden (d_1, d_2, d_3 ve d_4) oluşan beş tasarım değişkenini kullanarak minimum ağırlığa sahip dört adımlı konili bir kasnağın tasarımını bulmaktır. Şekil 4.4'te iki boyutlu olarak tasarlanan kademeli konik makarayı göstermektedir [51].



Şekil 4-4: Kademeli Konik Kasnak Probleminin Şematiği [51].

KKKP'nin uygunluk fonksiyonu Eşitlik 4.18'de verilmiştir.

$$f(\vec{x}) = \rho\omega\left[d_1^2\left(11+\left(\frac{N_1}{N_2}\right)^2\right)+d_2^2\left(1+\left(\frac{N_2}{N}\right)^2\right)+d_3^2\left(1+\left(\frac{N_3}{N}\right)^2\right)+d_4^2\left(1+\left(\frac{N_4}{N}\right)^2\right)\right] \tag{4.18}$$

Problemdeki kısıtlar Eşitlik 4.19 ve 4.20'de verilmiştir.

$$\begin{aligned} h_1(\vec{x}) &= C_1 - C_2 = 0, \\ h_2(\vec{x}) &= C_1 - C_3 = 0, \\ h_3(\vec{x}) &= C_1 - C_4 = 0, \end{aligned} \tag{4.19}$$

$$\begin{aligned} g_{i=1,2,3,4}(\vec{x}) &= -R_i \le 2, \\ g_{i=5,6,7,8}(\vec{x}) &= (0.75\times 745.6998) - P_i \le 0 \end{aligned} \tag{4.20}$$

Bazı değişkenlerin alabileceği değerler ve sabit ifadeleri Eşitlik 4.21 ile 4.24 arasında verilmiştir.

$$C_i = \frac{\pi d_i}{2}\left(1+\frac{N_i}{N}\right)+\frac{\left(\frac{N_i}{N}-1\right)^2}{4a}+2a,\ \ i=\{1,2,3,4\}, \tag{4.21}$$

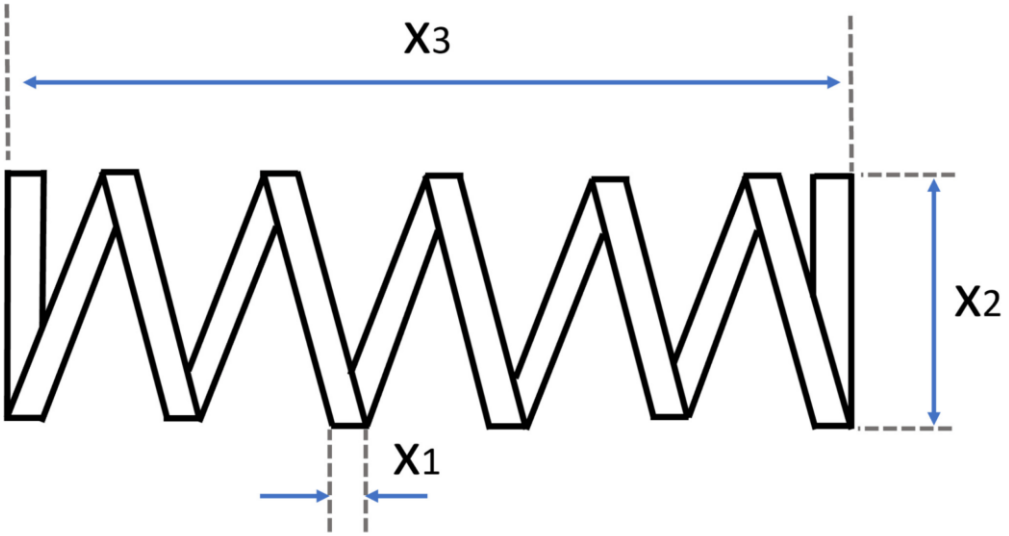
$$R_i = \exp\left(\mu\left\{\pi-2\sin^{-1}\left\{\left(\frac{N_i}{N}-1\right)\frac{d_i}{2a}\right\}\right\}\right),\ \ i=\{1,2,3,4\}, \tag{4.22}$$

$$P_i = st\omega\left(1-R_i\right)\frac{\pi d_i N_i}{60},\ \ i=\{1,2,3,4\}, \tag{4.23}$$

$$t=8 \text{ mm},\ s=1.75 \text{ MPa},\ \mu=0.35,\ \rho=7200\ \ kg/m^3,\ a=3 \text{ mm}. \tag{4.24}$$

4.5 GERME SIKIŞTIRMA YAYI TASARIMI

Germe Sıkıştırma Yayı Tasarımı (GSYT), sürekli (*continuous*) kısıtlı bir optimizasyon problemidir. Bu tasarım probleminin amacı, minimum sapma, dalgalanma frekansı ve kesme gerilimi sınırlamalarına bağlı kalarak sarmal yayın ağırlığını en aza indirmektir. Bu problemde, aktif bobinlerin sayısı (x_1), sarım çapı (x_2) ve tel çapı (x_3) olmak üzere üç tasarım değişkeni yer almaktadır [52]. Problemin şematiği Şekil 4.5'te gösterilmektedir.



Şekil 4-5: Germe Sıkıştırma Yayı Tasarımı [52]

GSYT'nin uygunluk fonksiyonu Eşitlik 4.25'te verilmiştir.

$$f\left(\vec{x}\right)=x_1^2\cdot x_2\cdot\left(x_3+2\right) \tag{4.25}$$

Problemdeki kısıtlar Eşitlik 4.26'da verilmiştir.

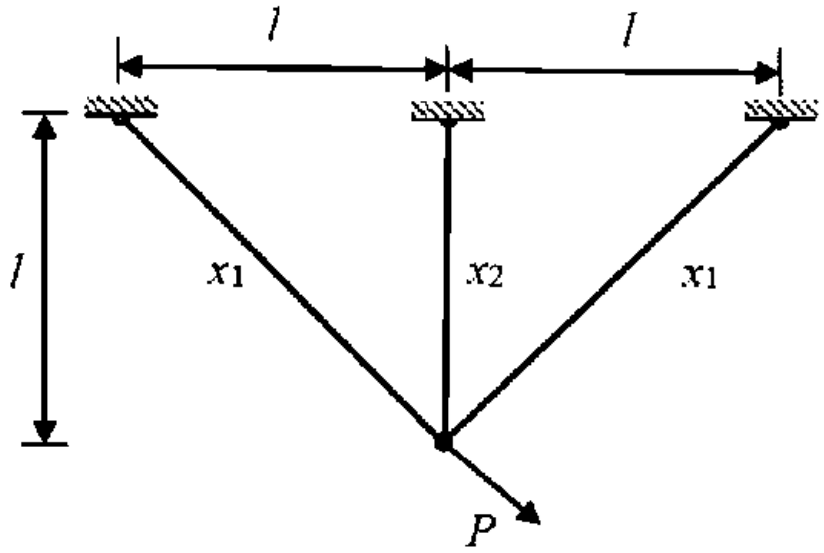
$$
\begin{aligned}
g_1(\vec{x}) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0, \\
g_2(\vec{x}) &= \frac{4x_2^2 - x_1 x_2}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \le 0, \\
g_3(\vec{x}) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0, \\
g_4(\vec{x}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0
\end{aligned}
\tag{4.26}
$$

Eşitlik 4.27'de bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$0.05 \le x_1 \le 2, \quad 0.25 \le x_2 \le 1.3, \quad 2 \le x_3 \le 15. \tag{4.27}$$

4.6 ÜÇ ÇUBUKLU KAFES KİRİŞ TASARIM PROBLEMİ

Üç Çubuklu Kafes Kiriş Tasarım Problemi (ÜÇKKTP), kısıtlı bir alana sahip inşaat mühendisliğinden alınmıştır. Bu problemin temel amacı, çubuk yapıların ağırlığını en aza indirmektir. Bu problemin kısıtlamaları, her çubuğun stres kısıtlamalarına dayanmaktadır. Bu minimizasyon probleminin iki durum/tasarım değişkeni ve doğrusal olmayan (non-linear) üç kısıtlama fonksiyonu vardır [53]. Üç çubuklu kafes kiriş tasarım problemi şematiği Şekil 4.6'da gösterilmektedir.



Şekil 4-6: Üç Çubuklu Kafes Tasarımı Probleminin Şematiği [53].

Eşitlik 4.28'de ÜÇKKTP'nin uygunluk fonksiyonu verilmiştir.

$$f\left(\vec{x}\right)=l\left(x_2+2\sqrt{2}x_1\right) \tag{4.28}$$

Problemdeki kısıtlar Eşitlik 4.29'da verilmiştir.

$$\begin{aligned} g_1\left(\vec{x}\right)&=\frac{x_2}{2x_2x_1+\sqrt{2}x_1^2}P-\sigma\le 0,\\ g_2\left(\vec{x}\right)&=\frac{x_2+\sqrt{2}x_1}{2x_2x_1+\sqrt{2}x_1^2}P-\sigma\le 0,\\ g_3\left(\vec{x}\right)&=\frac{1}{x_1+\sqrt{2}x_2}P-\sigma\le 0, \end{aligned} \tag{4.29}$$

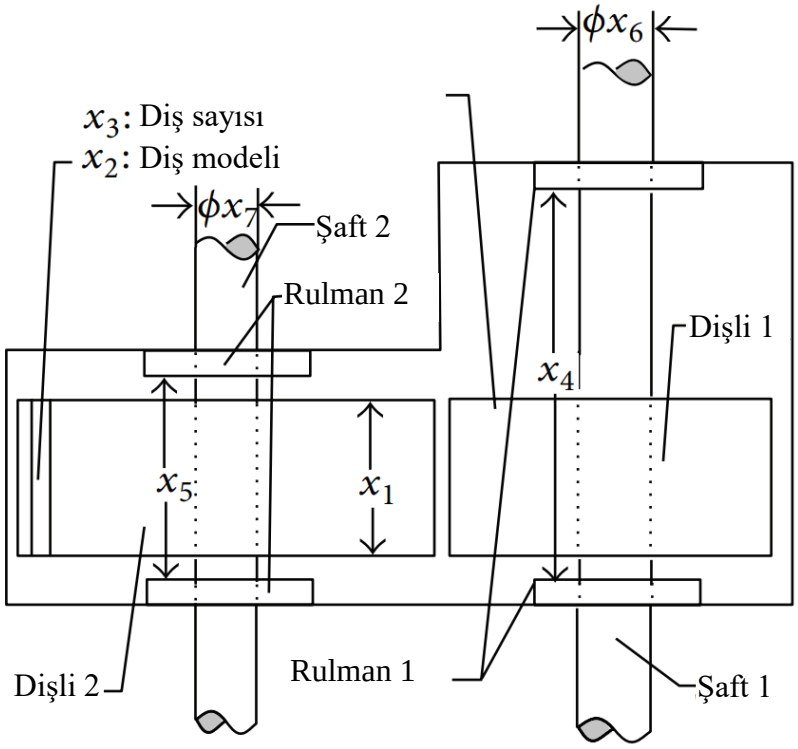
Eşitlik 4.30'da bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$0\le x_1,\ x_2\le 1,\ \ l=100,\ \ P=2,\ \ \sigma=2. \tag{4.30}$$

4.7 HIZ DÜŞÜRÜCÜSÜNÜN AĞIRLIK MİNİMİZASYONU

Hız düşürücü, mekanik sistemin dişli kutusunun bir parçasıdır ve birçok mühendislik uygulamasında kullanılmaktadır. Hız düşürücünün tasarımı oldukça zor bir problemdir çünkü

yedi tasarım değişkenini içerir. Şekil 4.7'de gösterildiği gibi hız düşürücünün tasarımı, alın genişliği (x_1), diş modülü (x_2), pinyondaki diş sayısı (x_3), yataklar arasındaki ilk milin uzunluğu (x_4) ile dikkate alınır. İkinci milin yataklar arasındaki uzunluğu (x_5), birinci milin çapı (x_6) ve ikinci milin çapını (x_7) temsil etmektedir [54].



Şekil 4-7: Hız Düşürücü Sistemin Şematiği. [54].

Eşitlik 4.31'de problemin uygunluk fonksiyonu verilmiştir.

$$f(\vec{x}) = 0.7854x_2^2x_1\left(14.9334x_3 - 43.0934 + 3.3333x_3^2\right) + 0.7854\left(x_5x_7^2 + x_{46}^2\right) - 1.508x_1\left(x_7^2 + x_6^2\right) + 7.477\left(x_7^3 + x_6^3\right) \quad (4.31)$$

Problemdeki kısıtlar Eşitlik 4.32'de verilmiştir.

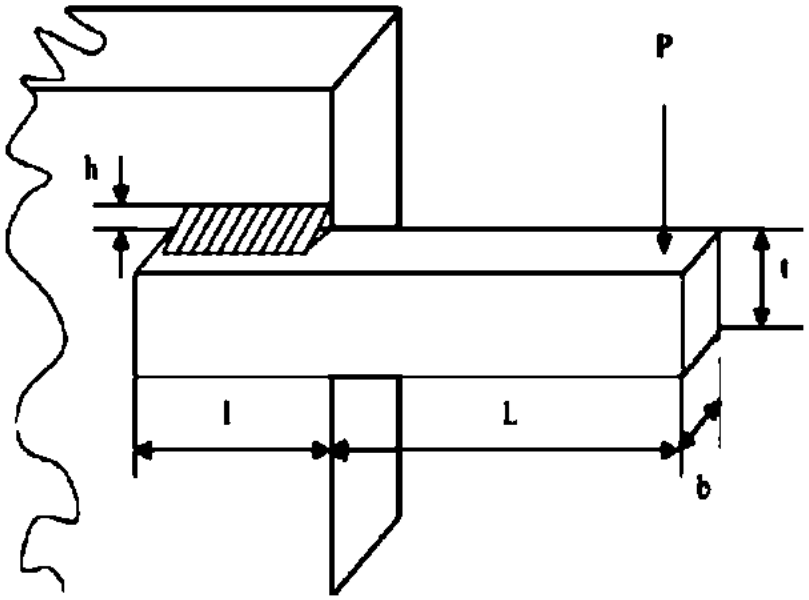
$$
\begin{aligned}
&g_1(\vec{x}) = -x_1 x_2^2 x_3 + 27 \le 0,\\
&g_2(\vec{x}) = -x_1 x_2^2 x_3^2 + 397.5 \le 0,\\
&g_3(\vec{x}) = -x_2 x_6^4 x_3 x_4^{-3} + 1.93 \le 0,\\
&g_4(\vec{x}) = -x_2 x_7^4 x_3 x_5^{-3} + 1.93 \le 0,\\
&g_5(\vec{x}) = 10 x_6^{-3} \sqrt{16.91\times10^6 + \left(745 x_4 x_2^{-1} x_3^{-1}\right)^2} - 1100 \le 0,\\
&g_6(\vec{x}) = 10_7^{-3} \sqrt{157.5\times10^6 + \left(745 x_5 x_2^{-1} x_3^{-1}\right)^2} - 850 \le 0,\\
&g_7(\vec{x}) = x_2 x_3 - 40 \le 0,\\
&g_8(\vec{x}) = -x_1 x_2^{-1} + 5 \le 0,\\
&g_9(\vec{x}) = x_1 x_2^{-1} - 12 \le 0,\\
&g_{10}(\vec{x}) = 1.5 x_6 - x_4 + 1.9 \le 0,\\
&g_{11}(\vec{x}) = 1.1 x_7 - x_5 + 1.9 \le 0,
\end{aligned}
\tag{4.32}
$$

Eşitlik 4.33'te bazı değişkenlerin sınır değerleri verilmiştir.

$$
\begin{aligned}
&2.6 \le x_1 \le 3.6,\\
&0.7 \le x_2 \le 0.8,\\
&17 \le x_3 \le 28,\\
&7.3 \le x_4, x_5 \le 8.3,\\
&2.9 \le x_6 \le 3.9,\\
&5 \le x_7 \le 5.5.
\end{aligned}
\tag{4.33}
$$

4.8 KAYNAKLI KİRİŞ TASARIMI

Kaynaklı Kiriş Tasarımı Problemi (KKTP), kısıtları kaynaktaki kesme gerilimi (τ), kirişteki eğilme gerilimi (θ), çubuk üzerindeki burkulma yükü (P_c), kirişin sapması (δ) olan kaynaklı bir kiriş [55] minimum maliyetle tasarlanmalıdır. KKTP'de kaynak kalınlığı ($h = x_1$), çubuğun ekli kısmının uzunluğu ($l = x_2$), çubuğun yüksekliği ($t = x_3$) ve çubuğun kalınlığı ($b = x_4$) olmak üzere dört tane tasarım değişkeni bulunmaktadır. Buna ilaveten, beş kısıt fonksiyonuna sahiptir ve şematiği Şekil 4.8'de gösterilmektedir.



Şekil 4-8: Kaynaklı Kiriş Tasarımı [56].

Eşitlik 4.34'te KKTP'nin uygunluk fonksiyonu verilmiştir.

$$f\left(\vec{x}\right)=0.04811x_3x_4\left(x_2+14\right)+1.1047x_1^2x_2 \tag{4.34}$$

Problemdeki kısıtlar Eşitlik 4.35'te verilmiştir.

$$\begin{aligned}
&g_1\left(\vec{x}\right)=x_1-x_4\le 0,\\
&g_2\left(\vec{x}\right)=\delta\left(\vec{x}\right)-\delta_{\max}\le 0,\\
&g_3\left(\vec{x}\right)=P\le P_c\left(\vec{x}\right)\\
&g_4\left(\vec{x}\right)=\tau_{\max}\ge \tau\left(\vec{x}\right),\\
&g_5\left(\vec{x}\right)=\sigma\left(\vec{x}\right)-\sigma_{\max}\le 0
\end{aligned} \tag{4.35}$$

Eşitlik 4.36'da bazı değişkenlerin alabileceği değerler ve sabit ifadeleri verilmiştir.

$$
\begin{aligned}
&\tau = \sqrt{\tau'^2 + \tau''^2 + 2\tau'\tau''\frac{x_2}{2R'}},\\
&\tau'' = \frac{RM}{j},\\
&\tau' = \frac{P}{\sqrt{2}x_2x_1},\\
&M = p\left(\frac{x_2}{2} + L\right),\\
&R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\\
&J = \left(\left(\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\sqrt{2}x_1x_2\right),\\
&\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},\\
&\delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4},\\
&P_c(\vec{x}) = \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
&L = 14in,\ P = 6000Ib,\ E = 30\times10^6\ psi,\ \sigma_{\max} = 30000\,psi,\\
&\tau_{\max} = 13600\,psi,\ G = 12\times10^6\ psi,\ \delta_{\max} = 0.25in.
\end{aligned}
\tag{4.36}
$$

Eşitlik 4.37’de bazı değişkenlerin sınır değerleri verilmiştir.

$$
\begin{aligned}
&0.1 \le x_3, x_2 \le 10,\\
&0.1 \le x_4 \le 2,\\
&0.125 \le x_1 \le 2.
\end{aligned}
\tag{4.37}
$$

5. DENEYSEL ÇALIŞMALAR

Optimum performans için bazı metasezgisel algorimalar, parametrelerinin önceden ayarlanmasını gerektirmektedir. Bu parametrelerin belirlenmesi, küresel optimalliğe ulaşmak için metasezgisel algoritmalarının arama uzayında hangi yöne gidileceğinin belirlenmesi açısından oldukça önemlidir. Bu nedenle, deneysel çalışmalarda kullandığımız metasezgisel yöntemlerin parametreleri, Tablo 5.1'de gösterildiği gibi geliştiriciler tarafından sıklıkla tercih edilen veya önerilen (varsayılan) değerlerden yararlanılmıştır.

Tablo 5-1: Metasezgisel algoritmaların parametreleri.

Metasezgiseller	**Belirlenen Parametreler**
ACO	Örnekleme sayısı 40 Yoğunlaştırma faktörü ya da seçilim baskısı değeri 0,5 Sapma-mesafe oranı 1
ABC	Hızlandırma katsayısınnın üst limiti 1 Eski besin kaynaklarını terk etme ya da deneme limiti $\lceil 0.6 \times D \times PS \rceil$ D: Problemin boyutu, PS: Popülasyon sayısı
SCA	Bu algoritmalar için önceden ayarlanması gereken herhangi bir parametre bulunmamaktadır.
SSA	

Deneysel çalışmalarda kullandığımız metasezgisellerin, Bölüm 4'te teknik ayrıntıları verilen gerçek dünya problemlerinin çeşitli yineleme ve popülasyon sayılarına göre yakınsama değerleri sınanmıştır. Yineleme sayıları sırasıyla 100, 500 ve 1000 olup, popülasyon sayısı ise 10 ve 50 olacak şekilde düzenlenmiştir. Buna ek olarak, her bir deney için metasezgisellerin çalışma süreleri de ölçülerek sonuçları mukayese edilmiştir.

5.1 PERFORMANS ÖLÇÜM KRİTERLERİ

Tez kapsamında ele alınan metasezgisellerin, gerçek dünya problemleri üzerindeki performanslarının ölçümü için küresel en iyi çözümün uygunluk değeri ve algoritmanın çalışma süresi temel alınmıştır. Buna göre, her bir deney bağımsız olarak 30 kez tekrar edildi ve elde edilen en iyi çözümlerin ortalama uygunluk değeri hesaplandı. Benzer biçimde, metasezgisel yöntemlerin çalışma sürelerinin de ortalaması alınmıştır. Optimizasyon algoritmaların çalışma

sürelerini analiz edildiği bilgisayar, Windows 10 işletim sistemi, AMD Ryzen5 2600 3.4 GHz mikroişlemci ve 32 GB sistem belleği özelliklerine sahiptir.

5.2 GERÇEK DÜNYA PROBLEMLERİNİN TESTLERİ

Bu çalışmada, literatürde iyi bilinen dört algoritmayı (ACO: Ant Colony Optimization, ABC: Artificial Bee Colony, SSA: Salp Swarm Algorithm ve SCA: Sine Cosine Algorithm) test etmek amacıyla gerçek dünya mühendislik problemleri kullanılmıştır. Bu problemler; Basınçlı Kap Tasarımı (BKT), Robot Kavrayıcı Problemi (RKP), Makaralı Rulman Tasarım Problemi (MRTP), Kademeli Konik Kasnak Problemi (KKKP), Germe Sıkıştırma Yayı Tasarımı (GSYT), Üç Çubuklu Kafes Kiriş Tasarım Problemi (ÜÇKKTP), Hız Düşürücüsünün Ağırlık Minimizasyonu (HDAM) ve Kaynaklı Kiriş Tasarımı (KKT) problemi olmak üzere sekiz tanedir. Tablo 5.2'de deneysel çalışmalarda kullanılan mühendislik problemleri ve bunların özellikleri verilmiştir.

Tablo 5-2: Mühendislik problemlerinin özellikleri.

No	Problem	Min/Max	Boyut	Kısıtlar	Alt Sınır	Üst Sınır
1	BKT	Min	4	4	0,51	200
2	RKP	Min	7	7	0	300
3	MRTP	Max	10	9	0,02	150
4	KKKP	Min	5	8	0	90
5	GSYT	Min	3	3	0,05	15
6	ÜÇKKTP	Min	2	3	0	1
7	HDAM	Min	7	11	0,7	28
8	KKT	Min	4	5	0,1	10

Yukarıdaki tabloda belirtilen özelliklerin tanımlamaları şu şekildedir:

- Boyut: Probleme ait tasarım değişkenlerinin sayısı (vektör uzunluğu),
- Kısıtlar: Problemin çözümlenmesi sırasında göz önünde bulundurulması gereken kısıt fonksiyonlarının sayısı,
- Alt Sınır: Tasarım değişkenlerinin alt sınır değerleri,
- Üst Sınır: Tasarım değişkenlerinin üst sınır değerlerini ifade etmektedir.

Deneysel çalışmalar altı farklı senaryoda gerçekleştirilmiştir. İlk iki deneyde yineleme sayısı 100, üçüncü ve dördüncü deneyde 500 ve son iki deneyde ise yineleme sayısı 1000 olarak belirlenmiştir. Ayrıca, tek sayılı (I, III ve V) deneylerde popülasyon sayısı 10 iken, çift sayılı deneylerde ise popülasyon sayısı 50 olarak sabit alınmıştır. Bu deneysel çalışmalardaki amaç, farklı yineleme ve popülasyon sayılarına bağlı olarak optimizasyon algoritmalarının performansını test etmek ve elde edilen sonuçları birbiriyle karşılaştırmaktır. Tablo 5.3'te deneylere ait özellikler verilmiştir.

Tablo 5-3: Deneylere ait yineleme ve popülasyon sayıları.

Deney No	Yineleme Sayısı	Popülasyon Sayısı
I	100	10
II	100	50
III	500	10
IV	500	50
V	1000	10
VI	1000	50

5.2.1 Deney I

Bu deneyde, yineleme sayısı 100, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi (*global best*) uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.4'te, Deney I'e göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-4: Deney I'e göre metasezgisellerin en iyi uygunluk değerleri.

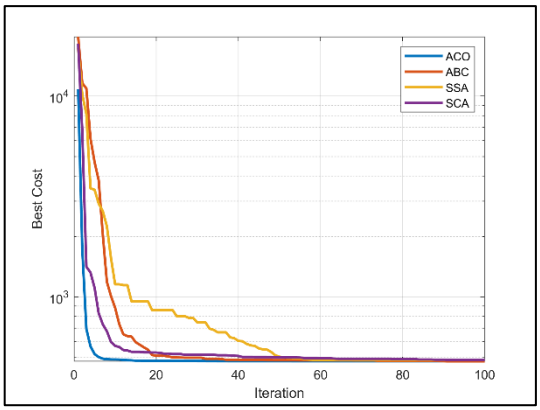
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	478,9057	**477,9938**	484,4800	485,8915
2	RKP	**4,3516**	6,9780	12,3018	36,5473
3	MRTP	**82044,9**	79372,3	53263,2	35388,3
4	KKKP	**8,2621**	8,8382	9,7220	12,3427
5	GSYT	**0,0126**	0,0129	0,0171	0,0139
6	ÜÇKKTP	**89,8504**	89,8542	**89,8504**	91,5346
7	HDAM	**2717,1390**	2722,1807	2797,8604	2913,4241
8	KKT	**1,7893**	2,0311	2,2108	2,0305

ACO_R algoritması 1 nolu problem haricinde tüm deneylerde en iyi skoru elde ederken, 2 nolu problemde rakiplerinden daha başarılı bir sonuç elde ettiği görülmektedir. ABC ve SSA yalnızca bir problemde daha başarılı olurken, SCA ise hiçbirinde en iyi skoru elde edememiştir. Algoritmaların çalışma süreleri, Tablo 5.5'te gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, 7 nolu problem dışında tüm deneylerde en hızlı çalışan algoritma olmuştur.

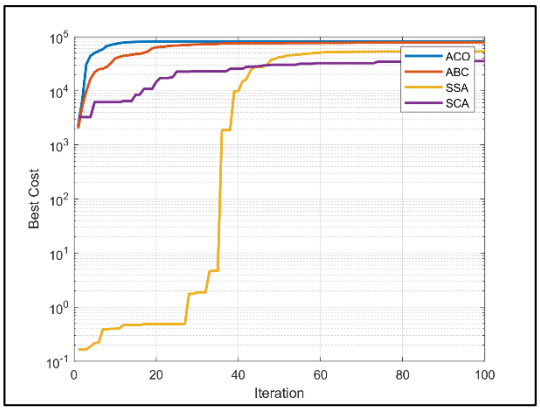
Tablo 5-5: Deney I'e göre metasezgisellerin çalışma süreleri (sn).

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	42,6015	21,7234	**9,1208**	9,4284
2	RKP	95,4876	71,2084	**29,6021**	51,8714
3	MRTP	73,5624	28,9248	**13,0382**	16,2019
4	KKKP	50,2567	24,5624	**10,1374**	11,8380
5	GSYT	37,7458	20,7402	**7,8194**	8,2876
6	ÜÇKKTP	29,6859	22,3271	**6,8670**	7,2907
7	HDAM	54,0416	27,2415	12,0812	**11,4556**
8	KKT	38,1975	23,3933	**7,7635**	8,5056

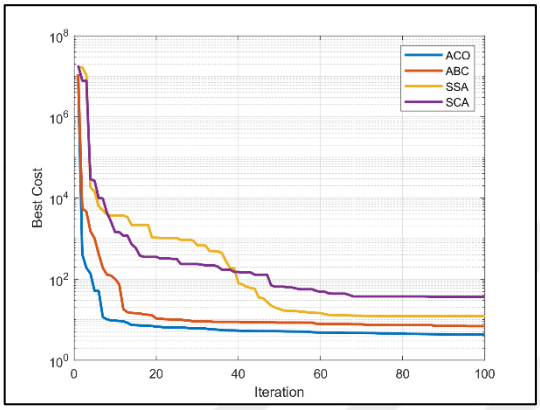
Deney I'e göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.1'de verilmiştir.



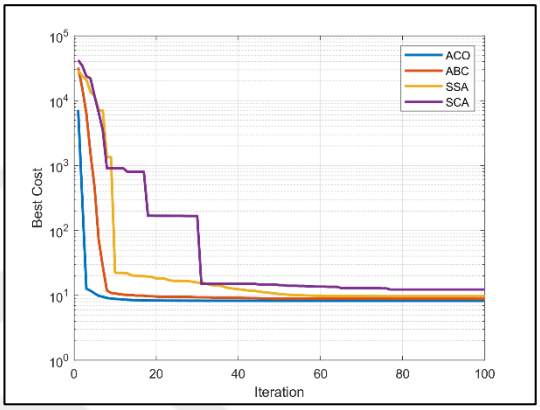
BKT



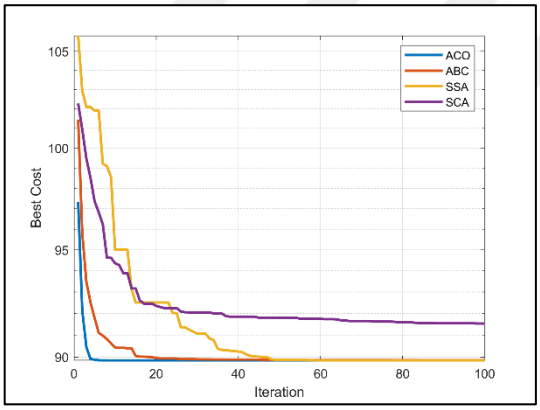
MRTP



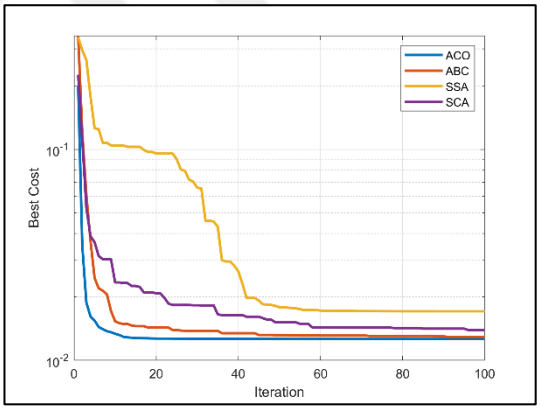
RKP



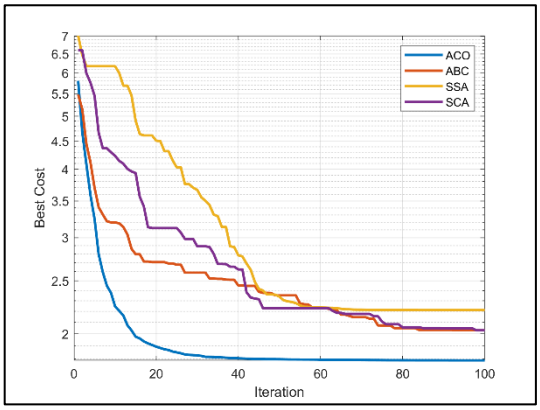
KKKP



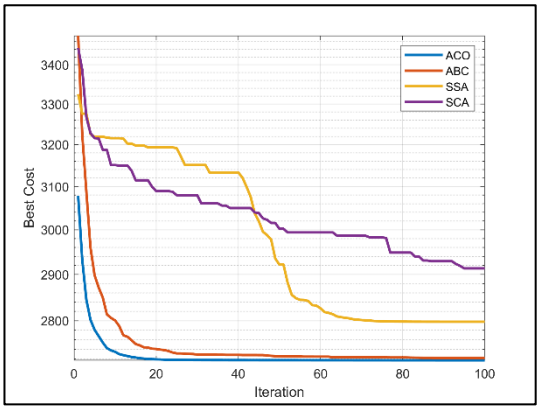
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-1: Deney I'e göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney I'e göre Şekil 5.1'de verilen algoritmaların yakınsama grafikleri incelendiğinde, BKT probleminde 100 yineleme süresinden önce metasezgisellerin hepsi optimum çözüme ulaştıkları görülmektedir. Genel olarak değerlendirilidiğinde, ACO_R algoritması tüm problemlerde en iyi yakınsama değerine sahiptir. Özellikle KKT probleminde diğer algoritmalara göre çok daha iyi sonuç elde etmiştir. SCA algoritması, RKP, ÜÇKKTP, MRTP, KKKP ve HDAM problemlerinde en kötü yakınsama değerine sahip olduğu görülmektedir.

5.2.2 Deney II

Bu deneyde, yineleme sayısı 100, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.6'da, Deney II'ye göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-6: Deney II'ye göre metasezgisellerin en iyi uygunluk değerleri.

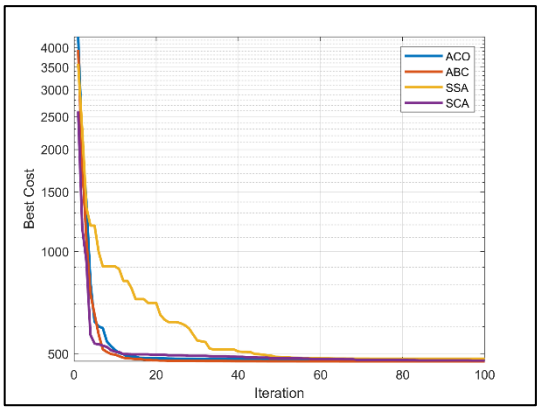
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	479,3181	**474,5254**	483,2615	477,3227
2	RKP	6,2606	**4,9287**	7,0604	6,8006
3	MRTP	**82101,8**	82082,5	69039,1	46290,8
4	KKKP	**8,1919**	8,2184	8,2714	10,3740
5	GSYT	**0,0124**	**0,0124**	0,0142	0,0125
6	ÜÇKKTP	**89,8504**	**89,8504**	**89,8504**	89,8537
7	HDAM	2716,0236	**2715,9995**	2746,6139	2887,9918
8	KKT	**1,6387**	1,6830	2,0620	1,7956

ACO_R ve ABC algoritması 5 deneyde en iyi skorla birinciliği paylaşan iki algoritma olmuştur. SSA yalnızca altı nolu problemde en iyi puanı elde ederken, SCA ise ilk deneyde olduğu gibi problemlerin hiçbirinde en iyi skora sahip değildir. Algoritmalarının çalışma süreleri, Tablo 5.7'de gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, 1 ve 2 nolu problemler dışında tüm deneylerde en hızlı çalışan algoritma olmuştur.

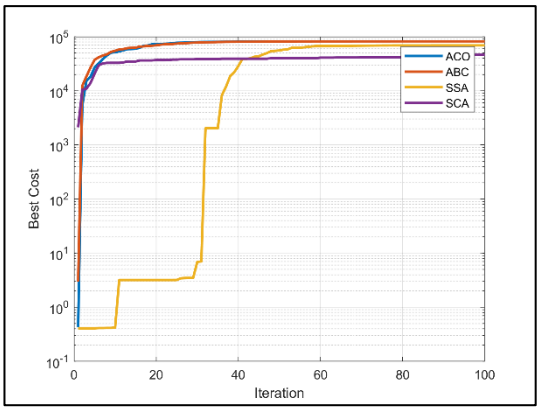
Tablo 5-7: Deney II'ye göre metasezgisellerin çalışma süreleri (sn).

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	340,5922	328,0823	139,5988	**139,1014**
2	RKP	**260,3329**	631,1179	332,4650	665,3854
3	MRTP	470,3396	476,9569	**198,3871**	230,4260
4	KKKP	379,6615	417,6725	**164,0837**	174,9866
5	GSYT	338,7804	347,6095	**125,5717**	136,1637
6	ÜÇKKTP	306,0639	362,9359	**123,1722**	125,6557
7	HDAM	399,7554	406,1834	**140,7905**	161,5556
8	KKT	364,1332	398,6565	**141,6681**	150,3467

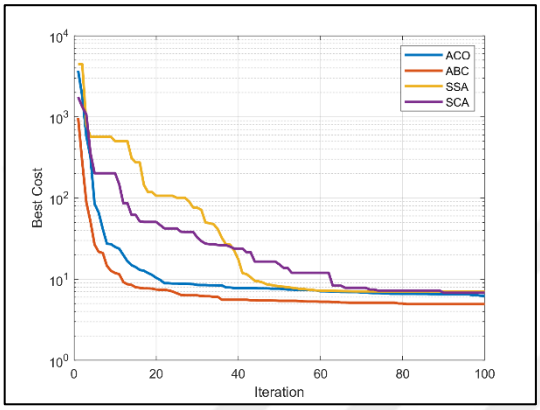
Deney II'ye göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.2'de verilmiştir.



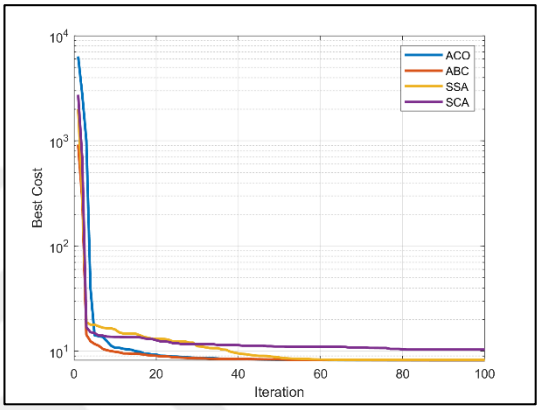
BKT



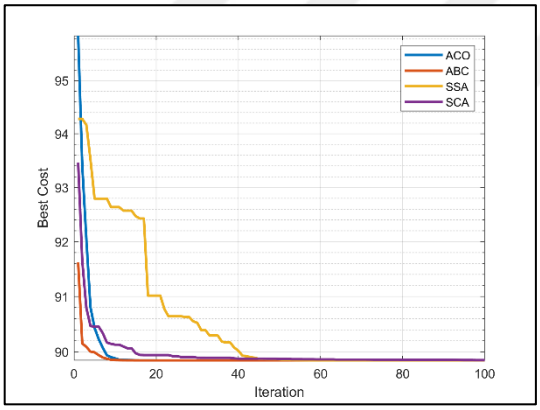
MRTP



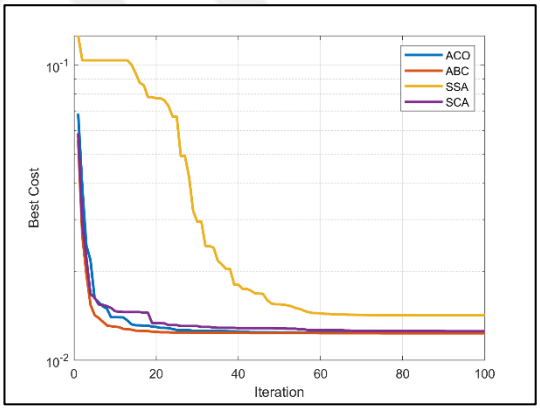
RKP



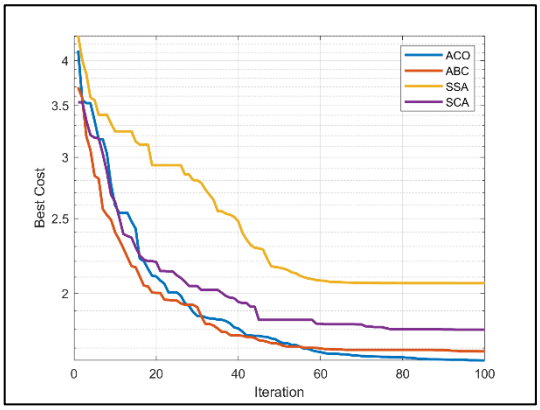
KKKP



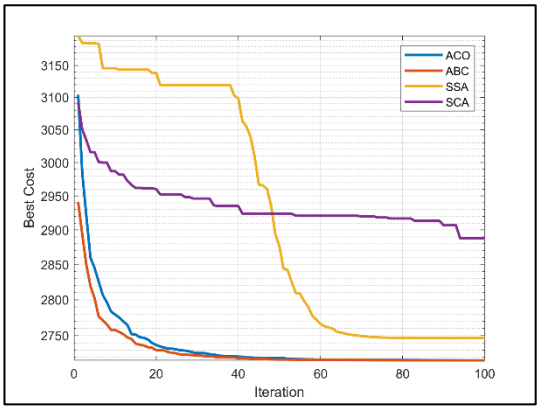
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-2: Deney II'ye göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney II'ye göre Şekil 5.2'de verilen algoritmaların yakınsama grafikleri incelendiğinde, popülasyon sayısının artmasıyla birlikte BKT ve ÜÇKKTP problemlerinde tüm metasezgiseller optimal sonuçları daha kısa sürede elde etmişlerdir. SSA algoritması, KKT ve GSYT problemleri için en kötü yakınsama değerine sahip olduğu görülmektedir. Bir maksimizasyon problemi olan MRTP'de ise en kötü yakınsama değeri SCA algoritmasına aittir.

5.2.3 Deney III

Bu deneyde, yineleme sayısı 500, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.8'de, Deney III'e göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-8: Deney III'e göre metasezgisellerin en iyi uygunluk değerleri.

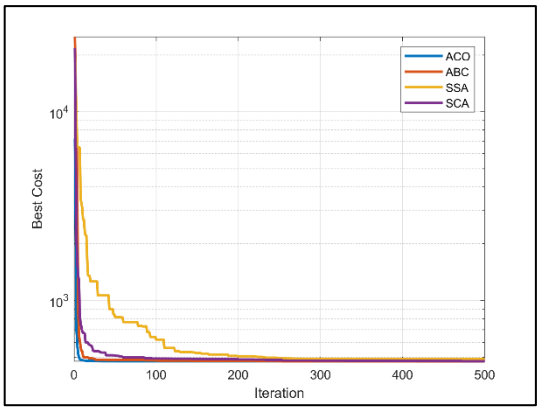
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	476,3155	**475,6138**	490,1005	476,2735
2	RKP	**4,0583**	6,0220	5,7649	5,5061
3	MRTP	**82104,8**	81375,1	63547,5	45863,8
4	KKKP	8,2709	8,7041	**8,2416**	10,3870
5	GSYT	**0,0124**	**0,0124**	0,0128	0,0126
6	ÜÇKKTP	**89,8504**	89,8520	**89,8504**	91,4325
7	HDAM	**2715,9872**	2718,0163	2758,7678	2875,2185
8	KKT	1,8284	1,8376	1,9119	**1,7836**

ACO_R algoritması 1, 4 ve 8 nolu problemler haricinde tüm deneylerde en iyi skoru elde etmiştir. ABC ve SSA, iki problemde ve SCA ise yalnızca bir problemde en iyi skoru elde etmiştir. Metasezgisellerin çalışma süreleri, Tablo 5.9'da gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, tüm deneylerde en hızlı çalışan algoritma olmuştur.

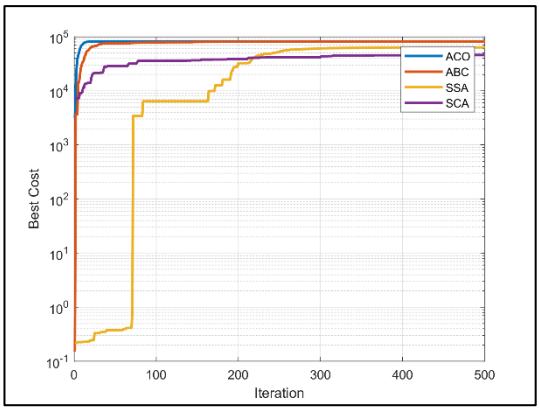
Tablo 5-9: Deney III'e göre metasezgisellerin çalışma süreleri (sn).

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	185,1463	96,4975	**37,8228**	41,7797
2	RKP	468,4439	332,9459	**137,3351**	287,3516
3	MRTP	358,6466	144,3775	**61,7456**	72,8864
4	KKKP	247,3981	121,2473	**48,8588**	54,8794
5	GSYT	169,6356	97,5699	**36,0501**	39,1379
6	ÜÇKKTP	142,3768	106,8492	**32,9257**	34,4966
7	HDAM	264,2581	134,8877	**47,8286**	55,9821
8	KKT	187,2139	114,2081	**37,6670**	42,5511

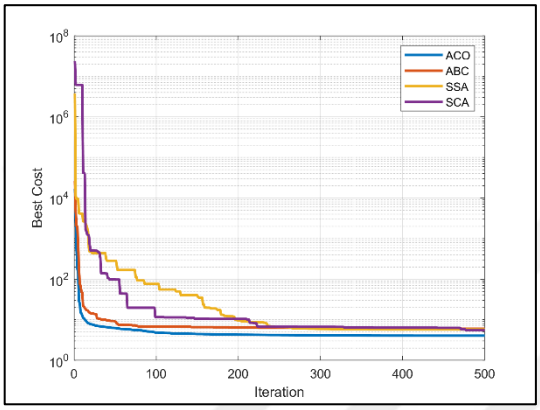
Deney III'e göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.3'te verilmiştir.



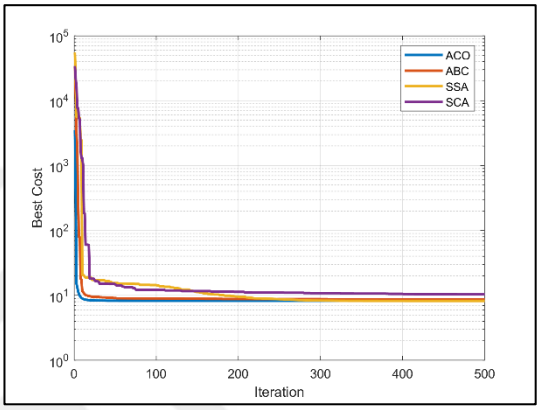
BKT



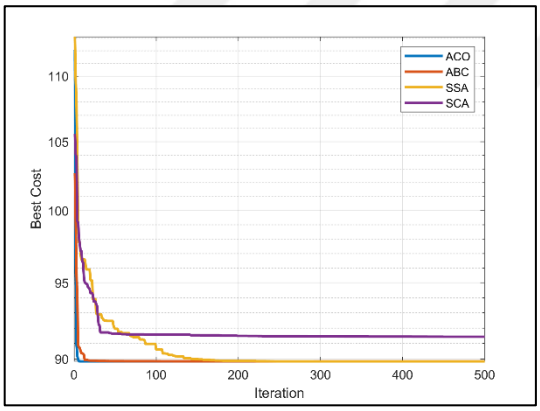
MRTP



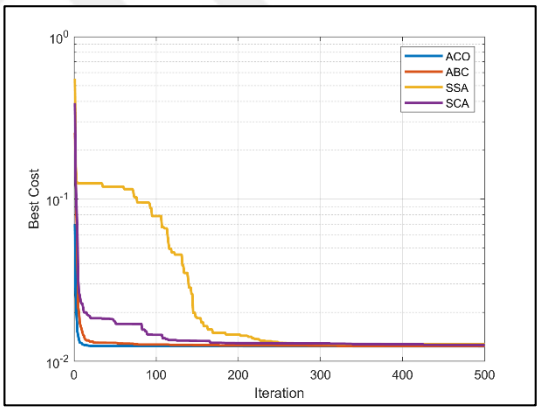
RKP



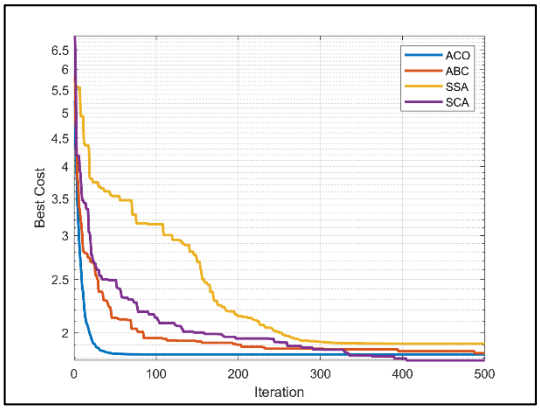
KKKP



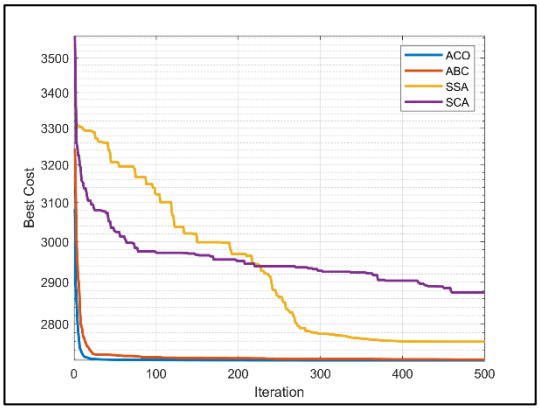
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-3: Deney III'e göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney III'e göre Şekil 5.3'te verilen algoritmaların yakınsama grafikleri incelendiğinde, BKT ve GSYT problemlerinin çözümünde tüm algoritmalar için 500 yineleme sayısı yeterli düzeyde olduğu görülmektedir. ÜÇKKTP probleminde, SCA haricindeki yöntemler 200 yineleme altında ideal çözüme ulaşırken, SCA yerel çözümün etkisinden kurtulamamıştır. HDAM probleminde ise ACO_R ve ABC algoritmaları kısa sürede yakınsayarak optimal çözüme ulaşırken, SSA ve SCA algoritmaları iyi bir performans sergileyememiştir.

5.2.4 Deney IV

Bu deneyde, yineleme sayısı 500, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.10'da, Deney IV'e göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-10: Deney IV'e göre metasezgisellerin en iyi uygunluk değerleri.

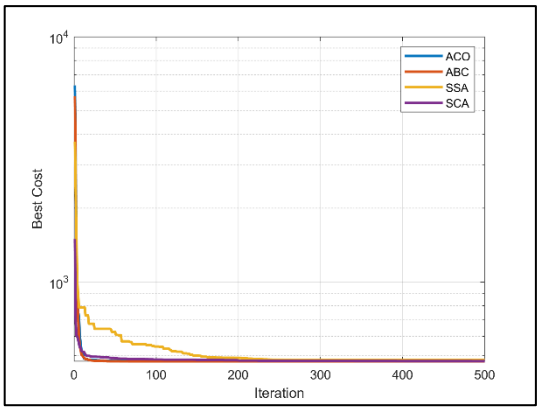
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	474,8162	**474,4043**	479,5544	475,2589
2	RKP	**3,6755**	4,1950	3,9235	4,3033
3	MRTP	**82125,8**	82125,4	72505,3	57620,9
4	KKKP	**8,1885**	**8,1885**	**8,1885**	9,7711
5	GSYT	**0,0123**	0,0124	0,0126	0,0124
6	ÜÇKKTP	**89,8504**	**89,8504**	**89,8504**	89,8529
7	HDAM	**2715,9872**	**2715,9872**	2737,2328	2792,5970
8	KKT	**1,6305**	1,6363	1,7253	1,7271

ACO_R algoritması 1 nolu problem haricinde tüm deneylerde en iyi skoru elde etmiştir. ABC ise dört deneyde en iyi skorla ikinci sırada ve SSA ise iki deneyde en iyi skorla üçüncü sırada yer almaktadır. SCA ise hiçbir problemde en iyi skoru elde edememiştir Algoritmalarının çalışma süreleri, Tablo 5.11'de gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, 2 nolu problemler dışında tüm deneylerde en hızlı çalışan algoritma olmuştur.

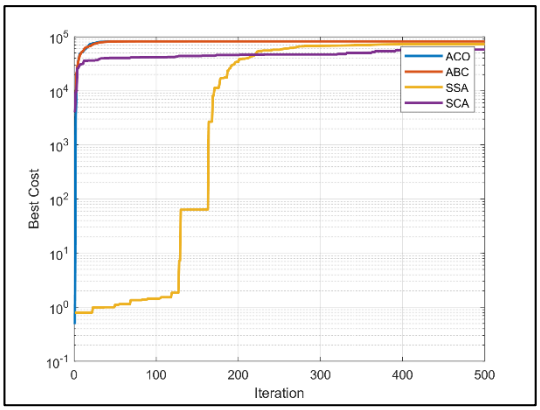
Tablo 5-11: Deney IV'e göre metasezgisellerin çalışma süreleri (sn).

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	583,9117	495,6912	**188,9991**	210,1969
2	RKP	**535,5466**	1456,3106	763,9661	1679,5669
3	MRTP	789,6234	792,5244	**320,5342**	377,8640
4	KKKP	661,0245	630,3324	**251,7666**	287,8837
5	GSYT	573,6928	504,4589	**186,1982**	200,7379
6	ÜÇKKTP	537,8684	560,0184	**168,3947**	175,5711
7	HDAM	677,5585	691,0457	**238,9354**	282,4484
8	KKT	587,6973	588,5523	**191,0540**	210,7186

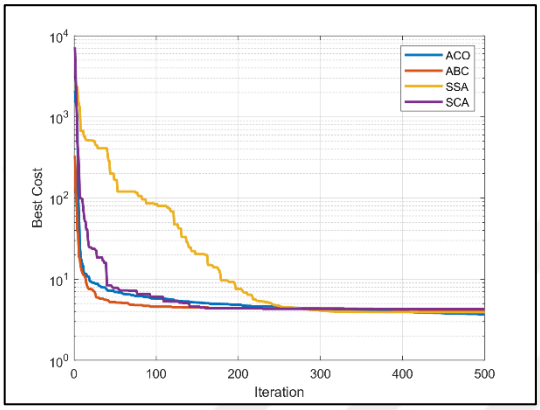
Deney IV'e göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.4'te verilmiştir.



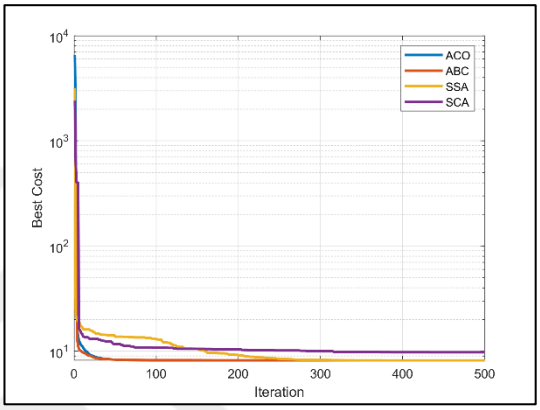
BKT



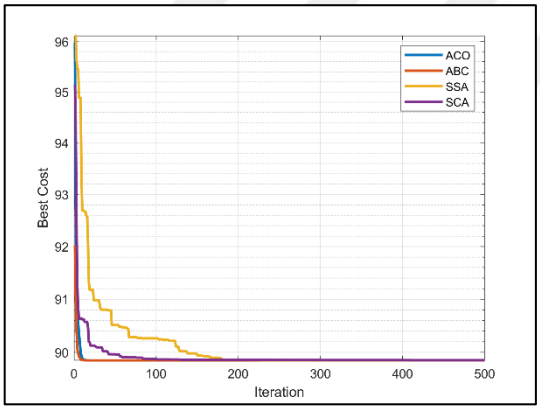
MRTP



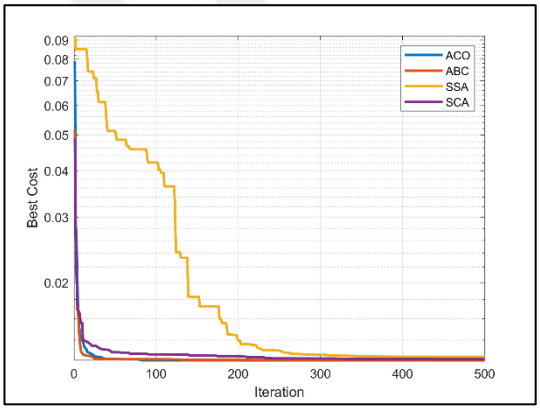
RKP



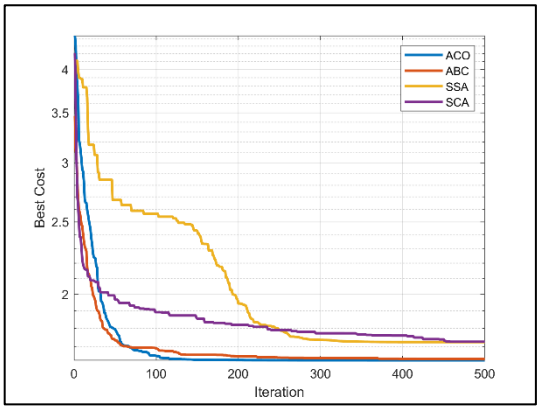
KKKP



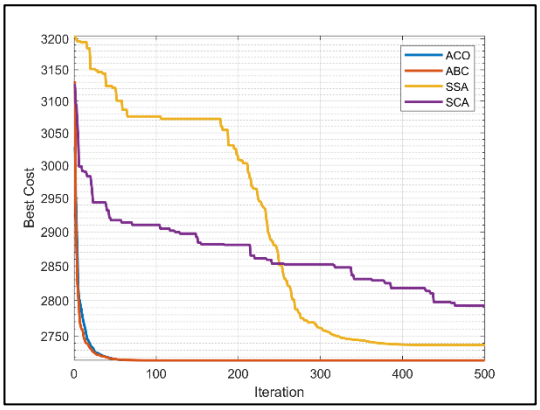
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-4: Deney IV'e göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney IV'e göre Şekil 5.4'te verilen algoritmaların yakınsama grafikleri incelendiğinde, popülasyon sayısının 50 olarak belirlenmesi tüm problemlerin çözümünde etkili olmuştur. Bu durum, BKT ve ÜÇKKTP problemleri için 200 civarında bir yinelemede optimum sonuçların elde edilmesini sağladığı görülmektedir. SCA algoritması, önceki deneylerde de olduğu gibi MRTP probleminde en kötü yakınsamaya sahiptir.

5.2.5 Deney V

Bu deneyde, yineleme sayısı 1000, popülasyon sayısı 10 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.12'de, Deney V'e göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-12: Deney V'e göre metasezgisellerin en iyi uygunluk değerleri.

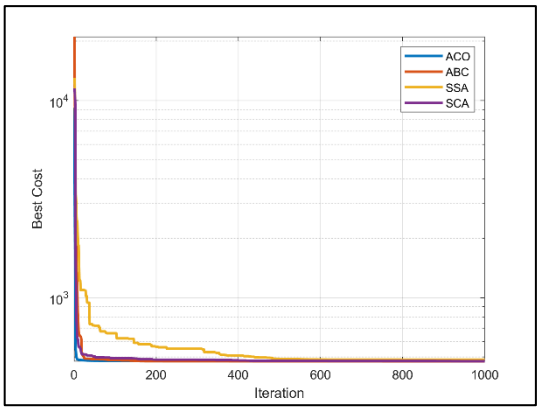
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	479,2746	**475,6983**	485,9391	476,3926
2	RKP	**3,5151**	5,1948	4,4472	4,7548
3	MRTP	81246,5	**81448,9**	74084,7	50245,7
4	KKKP	8,2382	8,6117	**8,1983**	15648,9397
5	GSYT	0,0125	**0,0124**	0,0130	0,0125
6	ÜÇKKTP	**89,8504**	89,8512	**89,8504**	94,5728
7	HDAM	**2715,9872**	2717,8021	2734,0179	2832,8562
8	KKT	**1,6970**	1,8126	1,7037	1,7253

ACO_R algoritması 1, 3, 4 ve 5 nolu problem haricinde tüm deneylerde en iyi skoru elde etmiştir. ABC, üç deneyde en iyi skorla ikinci sırada ve SSA ise iki deneyde en iyi skorla üçüncü sırada yer almaktadır. Optimizasyon algoritmalarının çalışma süreleri, Tablo 5.13'te gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, tüm deneylerde en hızlı çalışan algoritma olmuştur.

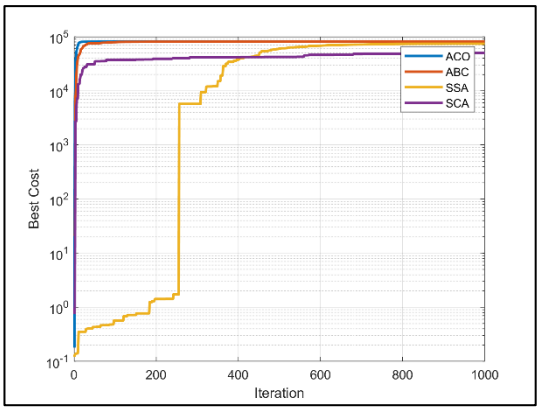
Tablo 5-13: Deney V'e göre metasezgisellerin çalışma süreleri (sn).

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	352,0897	181,5402	**71,0167**	80,6726
2	RKP	838,7533	615,6264	**259,0157**	574,0189
3	MRTP	682,1775	269,4687	**115,8177**	139,6405
4	KKKP	472,4440	228,0231	**92,5395**	104,7709
5	GSYT	327,2195	188,0169	**69,6981**	75,1521
6	ÜÇKKTP	273,9899	199,5363	**62,6357**	65,8694
7	HDAM	496,2187	245,1594	**88,3884**	105,0990
8	KKT	357,4127	217,2282	**70,8474**	79,4123

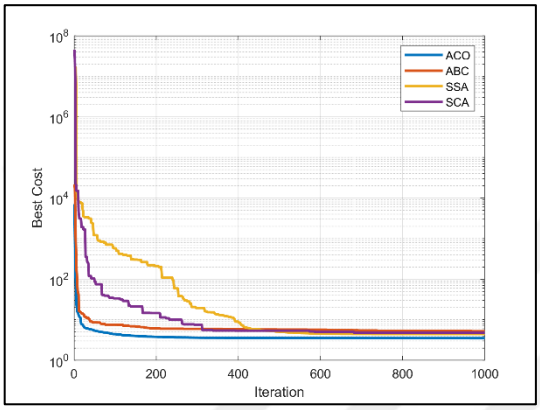
Deney V'e göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.5'te verilmiştir.



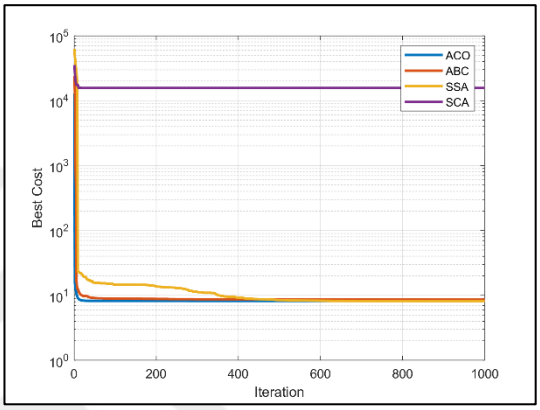
BKT



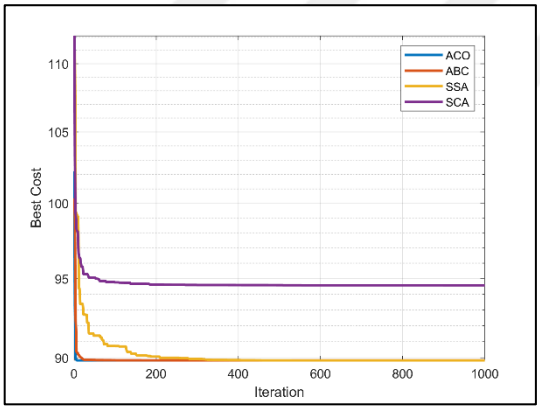
MRTP



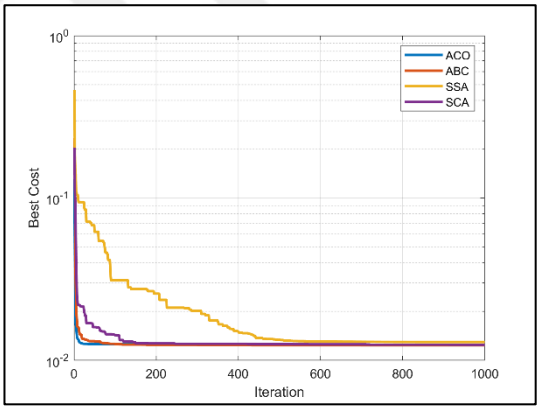
RKP



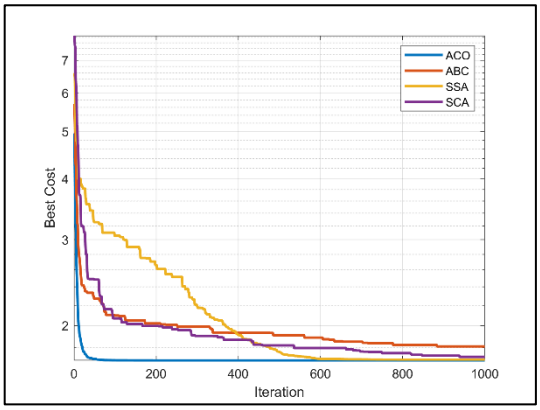
KKKP



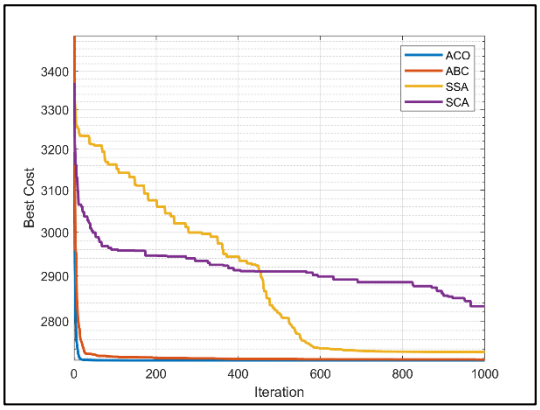
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-5: Deney V'e göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney V'e göre Şekil 5.5'te verilen algoritmaların yakınsama grafikleri incelendiğinde, SCA algoritmasının ÜÇKKTP ve KKKP problemlerinde yerel çözüm engeline takıldığı ve bunun sonucu olarak en kötü yakınsamayı gösteridiği göze çarpmaktadır. ACOR ve ABC algoritmaları, genel olarak tüm problemlerde en iyi yakınsamaya sahiptir. GSYT ve BKT problemlerinde tüm algoritmalar ideal çözüme ulaşktıkları görülmektedir.

5.2.6 Deney VI

Bu deneyde, yineleme sayısı 1000, popülasyon sayısı 50 olarak belirlenmiştir. Optimizasyon algoritmaları, her bir mühendislik problemi için 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi uygunluk değerlerinin ortalaması alınmıştır. Tablo 5.14'te, Deney VI'ya göre mühendislik problemlerinin küresel en iyi skorları verilmiştir.

Tablo 5-14: Deney VI'ya göre metasezgisellerin en iyi uygunluk değerleri.

No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	474,4144	**474,3841**	475,6687	475,0677
2	RKP	**3,0242**	4,3005	3,9888	4,2147
3	MRTP	**82125,8**	**82125,8**	75858,2	61754,5
4	KKKP	**8,1885**	**8,1885**	**8,1885**	9,3204
5	GSYT	**0,0123**	0,0124	0,0125	0,0124
6	ÜÇKKTP	**89,8504**	**89,8504**	**89,8504**	89,8510
7	HDAM	**2715,9872**	**2715,9872**	2734,6323	2792,6272
8	KKT	**1,6293**	1,6334	1,7144	1,7184

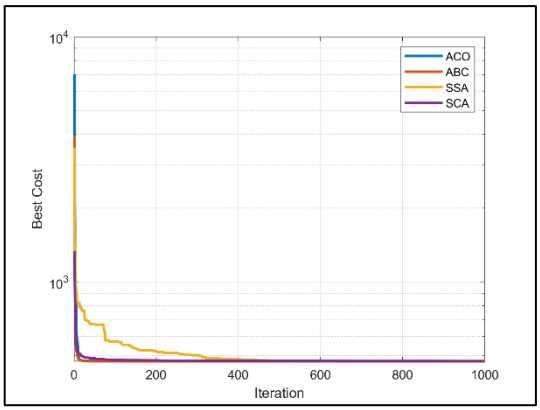
ACO_R algoritması 1 nolu problem haricinde tüm deneylerde en iyi skoru elde etmiştir. ABC ise beş deneyde en iyi skorla ikinci sırada ve SSA ise iki deneyde en iyi skorla üçüncü sırada yer almaktadır. Optimizasyon algoritmalarının çalışma süreleri, Tablo 5.15'te gösterildiği gibi her bir problem için saniye cinsinden ölçülmüştür. SSA, 2 nolu problem dışındaki tüm deneylerde en hızlı çalışan algoritma olmuştur.

Tablo 5-15: Deney VI'ya göre metasezgisellerin çalışma süreleri (sn).

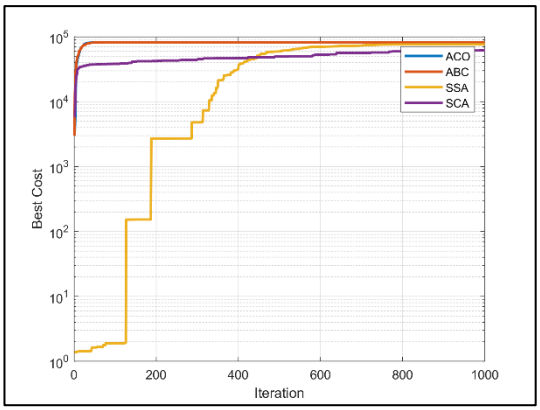
No	Problem	ACO_R	ABC	SSA	SCA
1	BKT	2441,3257	2415,4573	**894,9065**	990,4625
2	RKP	**1904,8256**	4673,6698	2636,8764	5689,5906
3	MRTP	3780,1957	4075,2605	**1606,6733**	1825,5777

4	KKKP	3010,9504	3246,5425	**1288,2404**	1388,2759
5	GSYT	2631,3144	2696,4627	**944,1845**	1044,4299
6	ÜÇKKTP	2518,8396	2902,1494	**914,8963**	974,2277
7	HDAM	3170,5489	3335,9925	**1159,1721**	1332,0030
8	KKT	2736,5262	3004,2976	**1015,2128**	1100,6511

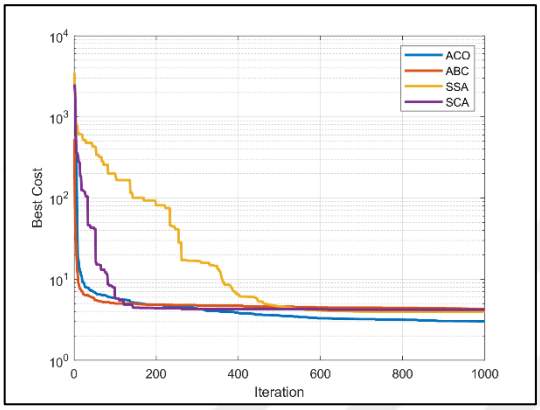
Deney VI'ya göre mühendislik problemlerinin yakınsama grafikleri Şekil 5.6'da verilmiştir.



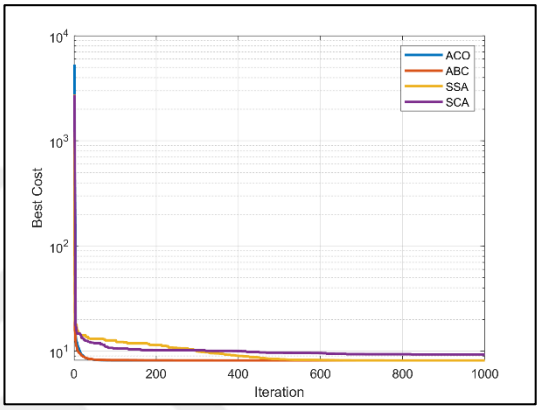
BKT



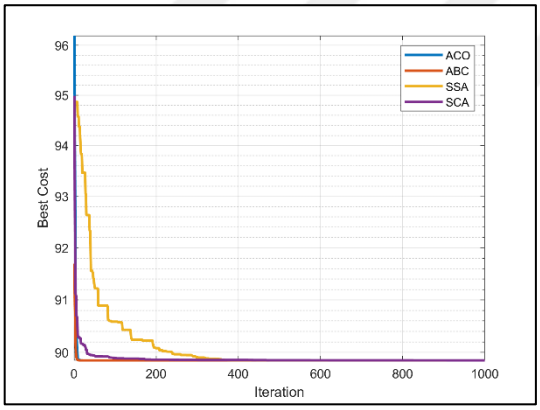
MRTP



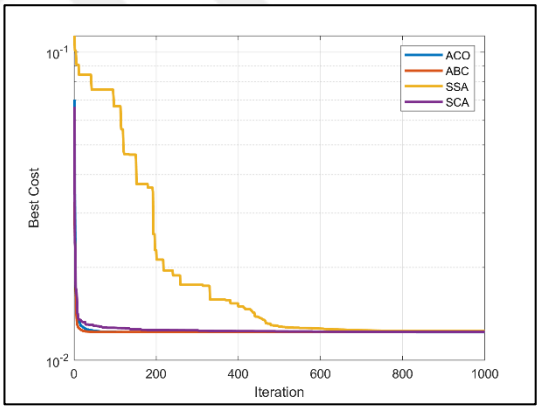
RKP



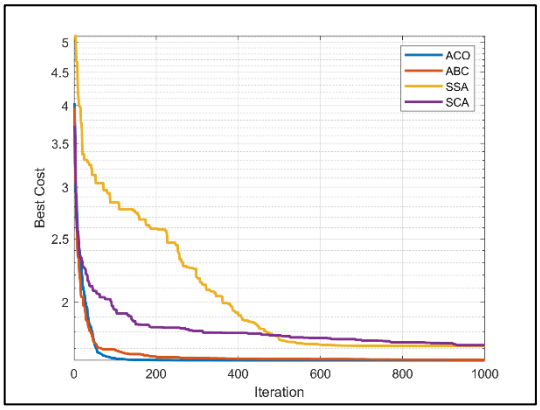
KKKP



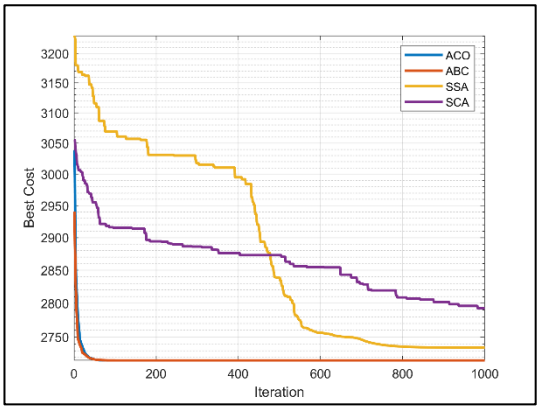
ÜÇKKTP



GSYT



KKT



HDAM

Şekil 5-6: Deney VI'ya göre optimizasyon algoritmalarının yakınsama grafikleri.

Deney VI'ya göre Şekil 5.6'da verilen algoritmaların yakınsama grafikleri incelendiğinde, BKT, ÜÇKKTP ve GSYT problemlerinde tüm algoritmaların optimal sonuca ulaştıkları görülmektedir. KKT ve HDAM probleminde, ABC ve ACO_R algoritmaları SSA ve SCA'ya göre çok daha hızlı yakınsamıştır. KKKP'de ise algoritmaların yakınsama hızları diğer problemlere göre daha yakın olduğu görülebilmektedir. SCA algoritması, MRTP probleminin çözümü için gerçekleştirilen deneylerin tümünde ek kötü yakınsamaya sahiptir.

6. TARTIŞMA VE SONUÇ

Bu tez çalışmasında, metasezgisel algoritmalarla gerçek dünya problemlerinin çözümü ve algoritmaların performans analizi gerçekleştirilmiştir. Elde edilen sonuçlara göre, makine mühendisliği alanındaki tasarım problemlerin çözümünde, hem uygunluk değeri hem de çalışma süresi açısından hangi yöntem daha uygun olduğu tespit edilmiştir.

Deneysel çalışmalarda kullanılan her bir problem için metasezgisel algoritmalar, 30 kez tekrarlı olarak çalıştırılmış ve elde edilen küresel en iyi skorların ortalaması hesaplanmıştır. Buna ilaveten, algoritmaların ortalama çalışma süreleri de göz önünde bulundurularak birbirleriyle karşılaştırılmıştır. Yineleme sayısı ve popülasyon sayısının değişen durumlarına bağlı olarak deneyler altı aşamada gerçekleştirilmiş ve aşağıdaki sonuçlar elde edilmiştir.

Deney I'de elde edilen sonuçlara göre; ACO_R algoritması, sekiz probleminin yedisinde en iyi skoru elde ederken, ABC ve SSA algoritmaları yalnızca bir problemde ve SCA ise problemlerin hiçbirinde en iyi skoru yakalayamamıştır. Çalışma süresine göre değerlendirildiğinde ise SSA algoritması yedi problemde, SCA ise bir problemin çözümünde en hızlı sonuç veren algoritma olmuştur. Deney II'de elde edilen sonuçlara göre; ABC ve ACO_R algoritmaları, problemlerinin beş tanesinde en iyi skoru elde ederken, SSA algoritması problemlerin yalnızca birinde en iyi skoru etmiştir. SCA ise ilk deneyde olduğu gibi hiçbirinde en iyi skora sahip olamamıştır. Algoritmaların çalışma süresine bakıldığında, SSA'nın en hızlı sonuç veren algoritma olduğu görülmektedir. Deney III'te elde edilen sonuçlara göre; ACO_R algoritması, problemlerin beşinde en iyi skoru elde ederken, ABC ve SSA problemlerin ikisinde ve SCA ise problemlerin yalnızca birinde en iyi skoru elde etmiştir. Yöntemlerin bu deneydeki çalışma süresi incelendiğinde, SSA'nın tüm deneylerde en hızlı çalışan algoritma olduğu görülmektedir. Deney IV'te elde edilen sonuçlara göre; ACO_R algoritması, problemlerin biri hariç tümünde en iyi skoru elde ederken, ABC problemlerin dördünde ve SSA ise problemlerin ikisinde en iyi skoru elde etmiştir. SSA algoritması, çalışma süresine göre en hızlı sonuç veren algoritma olmuştur. Deney V'te elde edilen sonuçlara göre; ACO_R algoritması, probleminin dördünde en iyi skoru elde ederken, ABC algoritması üç problemde ve SSA algoritması ise problemlerin sadece ikisinde en iyi skoru etmiştir. Çalışma süresine göre değerlendirildiğinde ise SSA'nın tüm deneylerde en hızlı sonuç veren algoritma olduğu görülmüştür. Altıncı ve son deneydeki sonuçlara göre; ACO_R algoritması, bir problem haricinde tümünde en iyi skoru elde etmiştir. ABC ise beş problemde en iyi skorla ikinci ve SSA ise iki problemin çözümünde en iyi

skorla üçüncü sırada yer almaktadır. Bu deneyde de en hızlı çalışan algoritmanın SSA olduğu görülmektedir.

Numerik optimizasyon problemleri, genellikle matematiksel fonksiyonların minimum veya maksimum değerlerini bulmayı amaçlayan problemlerdir. Bu tür problemler, genellikle matematiksel modellerle temsil edilir ve çeşitli optimizasyon algoritmaları kullanılarak çözülür. Nümerik optimizasyon problemleri, genellikle gerçek sayılarla ifade edilen değişkenlerin üzerinde çalışır. Nümerik olmayan optimizasyon problemleri ise genellikle ayrık, kategorik veya karma tip değişkenler içeren problemleri ifade eder. Örneğin, bir karar probleminde hangi ürünleri üreteceğinizi seçerken, bu ürünlerin sayısal değerlere sahip olmayan özelliklere (örneğin, renk, kategori) sahip olabileceği durumları düşünülebilir. Nümerik optimizasyon problemleri genellikle sürekli fonksiyonlarla çalıştığı için, çalışma süreleri genellikle daha hızlıdır. Bunun nedeni, sürekli fonksiyonların genellikle daha düzgün ve sürekli türevlere sahip olmaları ve bu tür özelliklerin optimizasyon algoritmalarının daha etkili çalışmasına izin vermesidir. Nümerik olmayan optimizasyon problemleri, genellikle karma tamsayılı programlama (Mixed-Integer Programming - MIP), genetik algoritmalar, simüle edilen tavlama gibi algoritmaları içerebilir. Bu tür problemlerin çözümü genellikle daha zorlu olabilir çünkü değişkenlerin sürekli olmayan doğası, bazı geleneksel nümerik optimizasyon yöntemlerini direkt olarak uygulamayı zorlaştırabilir. Her iki tür optimizasyon probleminin de çalışma süreleri, problem büyüklüğüne, karmaşıklığına ve kullanılan optimizasyon algoritmalarına bağlı olarak değişir. Problem türüne bağlı olarak, uygun bir optimizasyon yaklaşımını seçmek önemlidir. Bir problemin nümerik mi yoksa nümerik olmayan mı olduğunu belirlemek ve buna uygun bir çözüm stratejisi geliştirmek, genellikle problemi daha verimli bir şekilde çözmeye yardımcı olacaktır.

Yapılan deneysel çalışmalar genel olarak değerlendirildiğinde, ACO_R algoritmasının gerçek dünya mühendislik tasarımı problemlerinde rakiplerine göre daha iyi sonuçlar verdiği görülmektedir. Diğer optimizasyon algoritmaları başarı sırasına göre ABC, SSA ve SCA olarak söylenebilir. Algoritmaların çalışma süresine göre bir değelendirme yapıldığında, SSA'nın ezici bir üstünlükle en hızlı yöntem olduğu açıkça görülmektedir. Deneysel çalışmalarda, genel olarak ACO_R en yavaş çalışan algoritma olmasına rağmen, popülasyon sayısının 50 olduğu durumlardaki çalışma süresi ABC algoritması ile yakın değerlerde olduğu gözlemlenmiştir. SCA algoritması yapılan deneylerde en düşük sıralamaya sahiptir. Sonuç olarak, gerçek dünya problemlerinin çözümünde, ACO_R'nin daha iyi çözüm üreten ve ideal bir çalışma süresine sahip algoritma olduğunu söyleyebiliriz.

7. KAYNAKLAR

[1] Jiao, R., Commuri, S., Panchal, J., Milisavljevic-Syed, J., Allen, J. K., Mistree, F., & Schaefer, D. (2021). Design engineering in the age of industry 4.0. Journal of Mechanical Design, 143(7): 070801.

[2] Martins, J. R., & Ning, A. (2021). Engineering design optimization. Cambridge University Press.

[3] Carbas, S., Toktas, A., & Ustun, D. (Eds.). (2021). Nature-Inspired Metaheuristic Algorithms for Engineering Optimization Applications. Springer Tracts in Nature-Inspired Computing. doi: 10.1007/978-981-33-6773-9

[4] Abualigah, L., Elaziz, M. A., Khasawneh, A. M. et al. (2022). Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. Neural Comput & Applic 34: 4081–4110. doi: 10.1007/s00521-021-06747-4

[5] Çayıroğlu, İ., & Elen, A. (2012). A heuristic optimization approach for a real-world university timetabling problem. Advances in Computer Science and Engineering, 9(2): 103–131.

[6] Elen, A., & Çayıroğlu, İ. (2010). Solving of scheduling problem with heuristic optimization approach. Teknoloji, 13(3): 159–172.

[7] Elen, A. (2022). A Complete Solution to Exam Scheduling Problem: A Case Study. Journal of Scientific Reports-A, 2022(049): 12–34.

[8] Rather, S. A. & Bala, P. S. (2020). Swarm-based chaotic gravitational search algorithm for solving mechanical engineering design problems. World Journal of Engineering, 17(1): 97–114. doi: 10.1108/wje-09-2019-0254

[9] Braik, M., Sheta, A., & Al-Hiary, H. (2021). A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural computing and applications, 33: 2515-2547.

[10] Adekanmbi, O., & Green, P. (2015). Conceptual Comparison of Population Based Metaheuristics for Engineering Problems. The Scientific World Journal, 2015: 1–9.

[11] Farmer, J. D., Packard, N. H., & Perelson, A. S., (1986). The immune system, adaptation and machine learning, Physica D, 2:187–204.

[12] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, 4: 1942–1948.

[13] Storn, R., Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4): 341–359

[14] Simon, D. (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6): 702–713.

[15] Yang, X.S. (2008). Nature-Inspired Metaheuristic Algorithms. Luniver Press, Frome.

[16] Kaveh, A., Talatahari, S., (2010). A novel heuristic optimization method: charged system search, ActaMechanica, 213: 267–289.

[17] Yang, X. S. (2010). A New Metaheuristic Bat-Inspired Algorithm. In: González, J. R., Pelta, D. A., Cruz, C., Terrazas, G., Krasnogor, N. (eds) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, vol. 284. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-12538-6_6

[18] Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

[19] Meng X, Liu Y, Gao XZ, Zhang H. (2014) A new bio-inspired algorithm: Chicken swarm optimization. In: Advances in Swarm Intelligence. ICSI, Lecture Notes in Computer Science. Vol. 8794. Springer. pp. 86–94

[20] Bansal, J. C., Sharma, H., Jadon, S. S. & Clerc, M. (2014). "Spider monkey optimization algorithm for numerical optimization", Memetic Computing. 6(1):31–47.

[21] Saruhan, H. (2010). Yapısal Problemler Tasarımında Kuş Sürüsü Davranış Algoritması. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 16(2): 207–212.

[22] Balande, U., & Shrimankar, D. (2019). Srifa: stochastic ranking with improved-firefly-algorithm for constrained optimization engineering design problems. Mathematics, 7(3): 250. doi: 10.3390/math7030250

[23] Yıldırım, A. E., & Karcı, A. (2018). Application of three bar truss problem among engineering design optimization problems using artificial atom algorithm. In: 2018 international conference on artificial intelligence and data processing (IDAP), IEEE, 1–5.

[24] Chagwiza, G., Jones, B., Hove-Musekwa, S., & Mtisi, S. (2018) A new hybrid matheuristic optimization algorithm for solving design and network engineering problems. Int J Manage Sci Eng Manag 13(1): 11–19.

[25] Rahman, T. A. Z., Abdul Jalil, N. A., As'arry, A. & Raja Ahmad, R. K. (2021). "Performance evaluation of chaos-enhanced stochastic fractal search algorithm using constrained engineering design problems". 5th International Symposium on Applied Engineering and Sciences (SAES2017), Malaysia.

[26] Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. Future Generation Computer Systems, 101: 646–667.

[27] Kaleka, K. K., Kaur, A., & Kumar, V. (2020). A conceptual comparison of metaheuristic algorithms and applications to engineering design problems. International Journal of Intelligent Information and Database Systems, 13(2–4): 278–306.

[28] Zhi, L., Zhansheng, L., & Yigong, L. (2005). Dynamic simulation of distribution cam mechanism in internal combustion engine based on ant colony algorithm. Trans Chin Soc Agric Eng, 21(6): 64–67.

[29] Savsani, V., Rao, R. V., Vakharia, D. P. (2010). Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms. Mech Mach Theory 45: 531–541.

[30] Abderazek, H., Yildiz, A. R., & Mirjalili, S. (2020) Comparison of recent optimization algorithms for design optimization of a camfollower mechanism. Knowl-Based Syst 191: 105237.

[31] Salcedo-Sanz, S. (2016). Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. Phys. Rep. 655: 1–70.

[32] Wong, W., & Ming, C. I. (2019). A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications. 2019 7th International Conference on Smart Computing & Communications (ICSCC). doi: 10.1109/icscc.2019.8843624

[33] Meng, Z., Li, G., Wang, X., Sait, S. M., & Yıldız, A.R. (2021). A Comparative Study of Metaheuristic Algorithms for Reliability-Based Design Optimization Problems. Arch Computat Methods Eng, 28: 1853–1869.

[34] Abd Elaziz, M., Elsheikh, A. H., Oliva, D., Abualigah, L., Lu, S., & Ewees, A. A. (2021). Advanced Metaheuristic Techniques for Mechanical Design Problems: Review. Archives of Computational Methods in Engineering, 29: 695–716.

[35] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4): 28–39.

[36] Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. European Journal of Operational Research, 185(3): 1155–1173.

[37] Guntsch, M., & Middendorf, M., (2002). A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., & Raidl, G. (Eds.), Applications of

Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTim, 2279: 71-80. SpringerVerlag, Berlin/Germany.

[38] Kalami Heris, S. M., & Khaloozadeh, H. (2014). Ant Colony Estimator: An intelligent particle filter based on. Engineering Applications of Artificial Intelligence, 28: 78–85.

[39] Karaboga D. (2005). An idea based on honey bee swarm for numerical optimization technical report TR06, Erciyes University, Turkiye.

[40] De Rango, F., Palmieri, N., & Tropea, M. (2020). Multirobot coordination through bio-inspired strategies. Nature-Inspired Computation and Swarm Intelligence, 361–390. doi: 10.1016/b978-0-12-819714-1.00030-0

[41] Yang, X.-S. (2014). Other Algorithms and Hybrid Algorithms. Nature-Inspired Optimization Algorithms, 213–226. doi: 10.1016/b978-0-12-416743-8.00015-4

[42] http://www.nuhazginoglu.com/2018/05/15/yapay-ari-koloni-algoritmasi-artificial-bee-colony-abc-algorithm/ (Erişim Tarihi: 12.10.2023)

[43] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163–191. doi: 10.1016/j.advengsoft.2017.07.002

[44] Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Systems, 96, 120–133. doi: 10.1016/j.knosys.2015.12.022

[45] Gabis, A. B., Meraihi, Y., Mirjalili, S., & Ramdane-Cherif, A. (2021). A comprehensive survey of sine cosine algorithm: variants and applications. Artificial Intelligence Review, 54(7), 5469–5540. doi: 10.1007/s10462-021-10026-y

[46] Talatahari, S., & Azizi, M. (2020). Optimization of constrained mathematical and engineering design problems using chaos game optimization. Computers & Industrial Engineering, 145: 106560. doi: 10.1016/j.cie.2020.106560

[47] Andersson, J. (2000). A survey of multiobjective optimization in engineering design. Department of Mechanical Engineering, Linktjping University, Sweden.

[48] Savsani, P., & Savsani, V. (2016). Passing vehicle search (PVS): A novel metaheuristic algorithm. Applied Mathematical Modelling, 40(5-6), 3951–3978. doi: 10.1016/j.apm.2015.10.040

[49] Belkourchia, Y., Azrar, L., & Zeriab, E.-S. M. (2019). A Hybrid Optimization Algorithm for Solving Constrained Engineering Design Problems. 2019 5th International Conference on Optimization and Applications (ICOA). doi: 10.1109/icoa.2019.8727654

[50] Yao, J., Sha, Y., Chen, Y., & Zhao, X. (2022). A Novel Ensemble of Arithmetic Optimization Algorithm and Harris Hawks Optimization for Solving Industrial Engineering Optimization Problems. Machines, 10(8), 602. MDPI AG. doi: 10.3390/machines10080602

[51] Bahari, M. S., Harun, A., Zainal Abidin, Z., Hamidon, R., & Zakaria, S. (Eds.). (2021). Intelligent Manufacturing and Mechatronics. Lecture Notes in Mechanical Engineering. doi:10.1007/978-981-16-0866-7

[52] Alkan, B., & Kaniappan Chinnathai, M. (2021). Performance comparison of recent population-based metaheuristic optimisation algorithms in mechanical design problems of machinery components. Machines, 9(12): 341.

[53] Sheikhi Azqandi, M., Delavar, M. & Arjmand, M. An enhanced time evolutionary optimization for solving engineering design problems. Engineering with Computers 36, 763–781 (2020). doi: 10.1007/s00366-019-00729-w

[54] Lin, M.-H., Tsai, J.-F., Hu, N.-Z., & Chang, S.-C. (2013). Design Optimization of a Speed Reducer Using Deterministic Techniques. Mathematical Problems in Engineering, 2013, 1–7. doi: 10.1155/2013/419043

[55] Ragsdell, K.M. & Phillips, D.T. (1976) Optimal design of a class of welded structures using geometric programming, ASME Journal of Engineering for Industries 98(3):1021-1025, Series B.

[56] Sarkar, M., & Roy, T. K. (2017). Multi-Objective Welded Beam Design Optimization using T-Norm and T-Co-norm based Intuitionistic Fuzzy Optimization Technique. In Advances in Fuzzy Mathematics 12(3): 549-575. Research India Publications. doi: 10.37622/afm/12.3.2017.549-575

8. EKLER

EK 1. Karınca Kolonisi Optimizasyonunun Matlab ile Hazırlanmış Kaynak Kodu

```
% Ant Colony Optimization for Continuous Domains: MATLAB Kaynak Kodu
% Geliştirici: S. Mostapha Kalami Heris, sm.kalami@gmail.com
% https://www.mathworks.com/matlabcentral/fileexchange/52860

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet Function
boyut = 5; % Karar değişkenlerinin sayısı
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10; % Karar değişkenlerinin üst sınırı
%% ACO_R Parametereleri
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon sayısı
ornekSayisi = 40; % Örnek sayısı
q = 0.5; % Intensification Factor (Selection Pressure)
zeta = 1.0; % Deviation-Distance Ratio
%% Başlatma
% Boş bir karınca yapısı oluştur.
bosKarinca.Konum = [];
bosKarinca.Maliyet = [];
% Populasyon matrisini oluştur
suru = repmat(bosKarinca, popSayisi, 1);
% Başlangıç popülasyonunu oluştur
for i = 1 : popSayisi
   % Rastgele konumları belirle
   suru(i).Konum = unifrnd(altSinir, ustSinir, [1, boyut]);
   % Maliyeti hesapla
   suru(i).Maliyet = MaliyetFonk(suru(i).Konum);
end
% Popülasyonu sırala
[~, SortOrder] = sort([suru.Maliyet]);
suru = suru(SortOrder);
BestSol = suru(1); % En iyi çözümü güncelle
BestMaliyet = zeros(maxIterasyon, 1); % En iyi maliyet dizisi tanımla.
% Çözüm ağırlıkları
w = 1.0 / (sqrt(2.0 * pi) * q * popSayisi) * ...
   exp(-0.5*(((1:popSayisi)-1) / (q*popSayisi)).^2);
% Seçilim olasıklıkları
p = w / sum(w);
%% ACOR Ana Döngü
for t = 1 : maxIterasyon
   % Ortalama
```

```
    s = zeros(popSayisi, boyut);
    for l = 1 : popSayisi
        s(l, :) = suru(l).Konum;
    end
    % Standart sapmalar
    sigma = zeros(popSayisi, boyut);
    for l = 1 : popSayisi
        D = 0;
        for r = 1 : popSayisi
            D = D + abs(s(l, :) - s(r, :));
        end
        sigma(l, :) = zeta * D / (popSayisi - 1);
    end
    % Yeni bir popülasyon dizisi oluştur
    yeniSuru = repmat(bosKarinca, ornekSayisi, 1);
    for j = 1 : ornekSayisi
        % Başlangıç konumlarını belirle.
        yeniSuru(j).Konum = zeros(1, boyut);
        for i = 1 : boyut
            % Gauss Çekirdeği (Rulet Çarkı) Seçimi
            l = find(rand() <= cumsum(p), 1, 'first');
            % Rastgele Gauss Değişkeni Üret
            yeniSuru(j).Konum(i) = s(l, i) + sigma(l, i) * randn();
        end
        % Maliyet fonksiyonunu hesapla.
        yeniSuru(j).Maliyet = MaliyetFonk(yeniSuru(j).Konum);
    end
    % Ana Popülasyonu ve Yeni Popülasyonu Birleştir
    suru = [suru; yeniSuru]; %#ok
    % Popülasyonu sırala
    [~, SortOrder] = sort([suru.Maliyet]);
    suru = suru(SortOrder);
    % Ekstra üyeleri sil
    suru = suru(1:popSayisi);
    % En iyi çözümü güncelle
    BestSol = suru(1);
    % En iyi maliyeti sakla
    BestMaliyet(t) = BestSol.Maliyet;
    % İterasyon bilgisini göster
    disp(['Iterasyon ' num2str(t) ': En iyi maliyet = ' num2str(BestMaliyet(t))]);
end
%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('İterasyon'); ylabel('En iyi maliyet');
grid on;
```

EK 2. Yapay Arı Kolonisinin Matlab ile Hazırlanmış Kaynak Kodu

```
% Yapay Arı Kolonisi: MATLAB Kaynak Kodu
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% Geliştirici: S. Mostapha Kalami Heris, sm.kalami@gmail.com
% https://www.mathworks.com/matlabcentral/fileexchange/52966

clc; % Komut penceresini temizle
clear; % Değişkenleri temizle
close all; % Tüm figürleri kapat

%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
boyut = 5; % Karar değişkenlerinin sayısı
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10; % Karar değişkenlerinin üst sınırı

%% YAK Parametreleri
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon sayısı
gozcuSayisi = popSayisi; % Gözcü Arı sayısı
L = round(0.6 * boyut * popSayisi); % Vazgeçme Limiti
a = 1.0; % İvme Katsayısı Üst Sınırı

%% Başlatma
bosAri.Konum = [];% Boş arı yapısı
bosAri.Maliyet = [];;% Boş arı yapısı

% Başlangıç popülasyonu
suru = repmat(bosAri, popSayisi, 1);
% Başlangıç en iyi maliyet değeri
BestCozum.Maliyet = inf;

% Başlangıç popülasyonunu oluştur.
for i = 1 : popSayisi
   suru(i).Konum = unifrnd(altSinir, ustSinir, [1, boyut]);
   suru(i).Maliyet = MaliyetFonk(suru(i).Konum);
   if (suru(i).Maliyet < BestCozum.Maliyet)
      BestCozum = suru(i);
   end
end
C = zeros(popSayisi, 1); % Terk Etme Sayacı
BestMaliyet = zeros(maxIterasyon, 1); % En iyi maliyet dizisi
```

```
%% YAK Ana Döngü
for t = 1 : maxIterasyon
   %% İşçi Arılar
   for i = 1 : popSayisi
      % i'ye eşit olmayan k'yi rastgele seç
      K = [1:i-1, i+1:popSayisi];
      k = K(randi([1, numel(K)]));
      % İvme Katsayısını Tanımla.
      phi = a * unifrnd(-1.0 , 1.0, [1, boyut]);
      % Yeni arının konumunu belirle
      yeniAri.Konum = suru(i).Konum + ...
         phi .* (suru(i).Konum - suru(k).Konum);
      % Maliyeti hesapla
      yeniAri.Maliyet = MaliyetFonk(yeniAri.Konum);
      % Karşılaştır
      if (yeniAri.Maliyet < suru(i).Maliyet)
         suru(i) = yeniAri;
      else
         C(i) = C(i) + 1;
      end
   end

   % Uygunluk değerlerini ve seçilim olasılıklarını hesapla
   F = zeros(popSayisi, 1);
   ortMaliyet = mean([suru.Maliyet]);
   for i = 1 : popSayisi     % Maliyeti uygunluğa dönüştür
      F(i) = exp(-suru(i).Maliyet/ortMaliyet);
   end
   P = F / sum(F);

   %% Gözcü arılar
   for m = 1 : gozcuSayisi
      % Kaynak Alanı Seçimi (Rulet Çarkı)
      i = find(rand() <= cumsum(P), 1, 'first');
      % i'ye eşit olmayan k'yi rastgele seç
      K = [1:i-1, i+1:popSayisi];
      k = K(randi([1, numel(K)]));
      % İvme Katsayısını Tanımla.
      phi = a * unifrnd(-1.0, 1.0, [1, boyut]);
      % Yeni arının konumunu güncelle
      yeniAri.Konum = suru(i).Konum + ...
         phi .* (suru(i).Konum - suru(k).Konum);
      % Maliyetini hesapla
      yeniAri.Maliyet = MaliyetFonk(yeniAri.Konum);
      % Karşılaştır
      if (yeniAri.Maliyet < suru(i).Maliyet)
         suru(i) = yeniAri;
```

```
        else
            C(i) = C(i) + 1;
        end
    end

    %% İzci Arılar
    for i = 1 : popSayisi
        if (C(i) >= L)
            suru(i).Konum = unifrnd(altSinir, ustSinir, [1, boyut]);
            suru(i).Maliyet = MaliyetFonk(suru(i).Konum);
            C(i) = 0;
        end
    end

    %% En iyi çözümler
    for i = 1 : popSayisi   % Şimdiye Kadar Bulunan En İyi Çözümü Güncelle
        if (suru(i).Maliyet < BestCozum.Maliyet)
            BestCozum = suru(i);
        end
    end

    % En iyi maliyeti sakla.
    BestMaliyet(t) = BestCozum.Maliyet;
    % İterasyon bilgisini göster.
    disp(['İterasyon ' num2str(t) ': En iyi maliyet = ' num2str(BestMaliyet(t))]);
end

%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('İterasyon');
ylabel('En iyi maliyet');
grid on;
```

EK 3. Salp Sürü Algoritmasının Matlab ile Hazırlanmış Kaynak Kodu

```
% Salp Sürü Algoritması (SSA): MATLAB R2016a Kaynak Kodları (v 1.0)
% Geliştirici: Seyedali Mirjalili, ali.mirjalili@gmail.com
% https://www.mathworks.com/matlabcentral/fileexchange/63745

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
boyut = 5; % Karar değişkenlerinin sayısı
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10; % Karar değişkenlerinin üst sınırı
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon sayısı
if (size(ustSinir, 2) == 1)
   ustSinir = ones(1, boyut) * ustSinir;
   altSinir = ones(1, boyut) * altSinir;
end
% SalpKonum = zeros(popSayisi, boyut);
SalpMaliyet = zeros(popSayisi, 1);
% En iyi maliyet dizisi
BestMaliyet = zeros(maxIterasyon, 1);
% Başlangıç konumlarını belirle
sinirBoyutu = size(ustSinir, 2);
% Tüm değişkenlerin sınır değerleri aynı ise
if (sinirBoyutu == 1)
   r = rand(popSayisi, boyut);
   SalpKonum = r.*(ustSinir-altSinir)+altSinir;
End
% Her değişkenin farklı bir alt ve üst sınır değeri varsa
if (sinirBoyutu > 1)
   for i = 1 : boyut
      ubi = ustSinir(i);
      lbi = altSinir(i);
      r = rand(popSayisi, 1);
      SalpKonum(:, i) = r.*(ubi-lbi)+lbi;
   end
end
% Başlangıç popülasyonunun uygunluk değerlerini hesapla.
for i = 1 : size(SalpKonum, 1)
   SalpMaliyet(i, 1) = MaliyetFonk(SalpKonum(i, :));
End
% Maliyetleri küçükten büyüğe sırala.
[sortedSM, sortedIndx] = sort(SalpMaliyet);
```

```
besinKonum = SalpKonum(sortedIndx(1), :);
besinMaliyet = sortedSM(1);
% Ana Döngü
t = 1;
while (t < maxIterasyon + 1)
   t = t + 1;
   c1 = 2.0 * exp(-(4.0 * t / maxIterasyon)^2);    % Eq. (3.2) in the paper
   for i = 1 : size(SalpKonum, 1)
      if (i <= (popSayisi / 2))
         for j = 1 : 1 : boyut
            c2 = rand(); c3 = rand();
            % Eq. (3.1) in the paper
            k = c1 * ((ustSinir(j) - altSinir(j)) * c2 + altSinir(j));
            if (c3 < 0.5)
               SalpKonum(i, j) = besinKonum(j) + k;
            else
               SalpKonum(i ,j) = besinKonum(j) - k;
            end
         end
      elseif (i > (popSayisi / 2) &&  i < (popSayisi + 1))
         % Eq. (3.4) in the paper
         point1 = SalpKonum(i-1, :);
         point2 = SalpKonum(i, :);
         SalpKonum(i, :) = (point2 + point1) / 2;
      end
   end
   for i = 1 : size(SalpKonum, 1)
      Tp = SalpKonum(i, :) > ustSinir;
      Tm = SalpKonum(i, :) < altSinir;
      SalpKonum(i, :) = (SalpKonum(i, :).*(~(Tp+Tm))) + ustSinir.*Tp + altSinir.*Tm;
      SalpMaliyet(i, 1) = MaliyetFonk(SalpKonum(i, :));
      if (SalpMaliyet(i, 1) < besinMaliyet)
         besinKonum = SalpKonum(i, :);
         besinMaliyet = SalpMaliyet(i, 1);
      end
   end
   % En iyi maliyeti sakla.
   BestMaliyet(t) = besinMaliyet;
   % İterasyon bilgisini göster.
   disp(['İterasyon ' num2str(t) ': En iyi maliyet = ' num2str(BestMaliyet(t))]);
end
%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('İterasyon'); ylabel('En iyi maliyet');
grid on;
```

EK 4. Sinüs Kosinüs Algoritmasının Matlab ile Hazırlanmış Kaynak Kodu

```
% Sinüs Kosinüs Algoritması: MATLAB R2011b(7.13) Kaynak Kodu
% Geliştirici: Seyedali Mirjalili, ali.mirjalili@gmail.com
% https://www.mathworks.com/matlabcentral/fileexchange/54948

clc; clear; % Komut penceresini ve değişkenleri temizle
%% Problem Tanımı
MaliyetFonk = @(x) sum(x.^2); % Maliyet fonksiyonu
boyut = 5; % Karar değişkenlerinin sayısı
altSinir = -10; % Karar değişkenlerinin alt sınırı
ustSinir = 10; % Karar değişkenlerinin üst sınırı
maxIterasyon = 100; % Maksimum iterasyon sayısı
popSayisi = 25; % Populasyon sayısı
Maliyetler = zeros(popSayisi, 1);
BestMaliyet = zeros(maxIterasyon, 1);
bestKonum = zeros(1, boyut);
bestSkor = inf;
% Başlangıç konumlarını belirle
sinirBoyutu = size(ustSinir, 2);
% Tüm değişkenlerin sınır değerleri aynı ise
if (sinirBoyutu == 1)
   r = rand(popSayisi, boyut);
   Konumlar = r.*(ustSinir-altSinir)+altSinir;
End
% Her değişkenin farklı bir alt ve üst sınır değeri varsa
if (sinirBoyutu > 1)
   for i = 1 : boyut
      ubi = ustSinir(i);       lbi = altSinir(i);
      r = rand(popSayisi, 1);
      Konumlar(:, i) = r.*(ubi-lbi)+lbi;
   end
end
% Başlangıç popülasyonunun uygunluk değerini hesapla ve en iyisini bul.
for i = 1 : popSayisi
   Maliyetler(i) = MaliyetFonk(Konumlar(i, :));
   if (i == 1)
      bestKonum = Konumlar(i, :);
      bestSkor = Maliyetler(i);
   elseif (Maliyetler(i) < bestSkor)
      bestKonum = Konumlar(i, :);
      bestSkor = Maliyetler(i);
   end
end
```

```
% Ana Döngü
t = 1;
while (t <= maxIterasyon)
  t = t + 1;   % İterasyon sayacını bir artır.
  a = 2.0;   % Makaledeki Eşitlik (3.4)
  r1 = a - t * (a/maxIterasyon);   % r1 doğrusal olarak a'dan sıfıra azalır.
  % Hedefe göre çözümlerin konumunu güncelle.
  for i = 1 : popSayisi % i-inci çözüm
    for j = 1 : boyut % j-inci boyut
      % r2, r3 ve r4'ü makaledeki Eşitlik (3.3)'e göre güncelle.
      r2 = 2.0 * pi * rand();
      r3 = 2.0 * rand();
      r4 = rand();
      if (r4 < 0.5) % Makaledeki Eşitlik (3.3)
        % Makaledeki Eşitlik (3.1)
        Konumlar(i, j) = Konumlar(i, j) + ...
          (r1 * sin(r2) * abs(r3 * bestKonum(j) - Konumlar(i, j)));
      else % Makaledeki Eşitlik (3.2)
        Konumlar(i, j) = Konumlar(i, j) + ...
          (r1 * cos(r2) * abs(r3 * bestKonum(j) - Konumlar(i, j)));
      end
    end
  end
  for i = 1 : popSayisi
    % Arama uzayında sınır değer aşımlarını kontrol et ve güncelle.
    ust = Konumlar(i, :) > ustSinir;
    alt = Konumlar(i, :) < altSinir;
    Konumlar(i, :) = (Konumlar(i, :).*(~(ust+alt))) + ustSinir.*ust+altSinir.*alt;
    % Maliyet değerini hesapla.
    Maliyetler(i) = MaliyetFonk(Konumlar(i, :));
    % Daha iyi bir skor varsa en iyi olanı güncelle.
    if (Maliyetler(i) < bestSkor)
      bestKonum = Konumlar(i,:);
      bestSkor = Maliyetler(i);
    end
  end
  % En iyi maliyeti sakla.
  BestMaliyet(t) = bestSkor;
  % İterasyon bilgisini göster.
  display(['İterasyon ', num2str(t), ': En iyi maliyet = ', num2str(bestSkor)]);
end
%% Sonuçlar
figure;
semilogy(BestMaliyet, 'LineWidth', 2);
xlabel('İterasyon'); ylabel('En iyi maliyet');
grid on;
```