TAŞIT DEBRİYAJ ELEMANLARININ OPTİMİZASYONU İÇİN YAPAY ZEKA ALGORİTMASI TABANLI BİR SİSTEMİN GELİŞTİRİLMESİ

Alper KARADUMAN

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TAŞIT DEBRİYAJ ELEMANLARININ OPTİMİZASYONU İÇİN YAPAY ZEKA ALGORİTMASI TABANLI BİR SİSTEMİN GELİŞTİRİLMESİ

Alper KARADUMAN
0000-0001-6723-5136

Prof. Dr. Ali Rıza YILDIZ
0000-0003-1790-6987
(Danışman)

DOKTORA TEZİ

OTOMOTİV MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022
Her Hakkı Saklıdır

ÖZET

Doktora Tezi

TAŞIT DEBRİYAJ ELEMANLARININ OPTİMİZASYONU İÇİN YAPAY ZEKA ALGORİTMASI TABANLI BİR SİSTEMİN GELİŞTİRİLMESİ

Alper KARADUMAN

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Otomotiv Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ali Rıza YILDIZ

Bu tez çalışmasında taşıt debriyaj elemanlarının dinamik davranışı ve fonksiyonelliği göz önünde bulundurularak taşıt diyaframı ve damperinin tasarım optimizasyonu gerçekleştirilmiştir. Algoritmalarda keşif ve elde edilen bilginin kullanılması arasındaki dengenin ayarlanamaması ve uygun parametre ayarlarının gerçekleştirilememesi yerel optimuma takılma ve erken yakınsama problemine neden olabilir. Bu yüzden algoritmalar içerisindeki seçim mekanizması ve algoritma parametrelerinin ayarlanması önemlidir. Algoritma içerisinde çeşitlilik, bütünü arama ve yerel arama mekanizmalarının bilgiyi kullanma mekanizması ile yüksek performansta çalışması gerekmektedir. Bu amaçla kır kurtlarının sürü içerisindeki sosyal durumları ve çevreye uyum davranışlarından ilham alan Kır Kurdu Optimizasyon (COA) algoritması tabanlı yeni bir algoritma taşıt debriyaj elemanları için geliştirilmiştir. Bu çalışmada kır kurdu optimizasyon algoritmasının çeşitlilik ve arama mekanizmasının iyileştirilmesiyle geliştirilmiş bir algoritma elde edilmiştir. Ortaya çıkan geliştirilmiş algoritmanın kısıtlı ve kısıtsız mühendislik problemleri ile test edilip taşıt debriyaj elemanlarına uygulanarak gerçekliği kanıtlanmıştır. Önerilen algoritmanın test sonuçları, yeni algoritmanın etkinliğini ölçmek için literatürdeki diğer algoritmalara karşı kontrol edilmiştir. Deneysel sonuçlar, yeni algoritmanın kısıtlı ve kısıtsız test fonksiyonları üzerinde önemli ölçüde daha iyi verimliliğe sahip olduğunu göstermektedir. Bu eniyileme çalışması ile taşıt debriyaj elemanları diyafram yayının başlangıçtaki modeline göre belirtilen asgari ayrılma yüküne göre daha dinamik koşullarda daha iyi performans gösteren yeni bir diyafram modeli ortaya konulmuştur. İkinci eniyileme çalışmasında taşıt debriyaj damper elemanının başlangıçtaki modeline göre daha iyi sönümleme karakteristiğine sahip damper modeli elde edilmiştir. Böylece geliştirilen kır kurdu algoritması literatürde ilk olup taşıt debriyaj elemanlarının eniyilenmesinde kullanılmıştır.

Anahtar Kelimeler: Doğadan esinlenen algoritmalar, kır kurdu algoritması, diyafram yay, debriyaj, sistem optimizasyonu, parametre optimizasyonu
2022, vi + 202 sayfa.

ABSTRACT

PhD Thesis

DEVELOPMENT OF A SYSTEM BASED ON ARTIFICIAL INTELLIGENCE ALGORITHM FOR THE OPTIMIZATION OF VEHICLE CLUTCH COMPONENTS

Alper KARADUMAN

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Automotive Engineering

Supervisor: Prof. Dr. Ali Rıza YILDIZ

In this thesis, the vehicle diaphragm and the damper design were optimized for the vehicle clutch elements' dynamic behavior and functionality. Failure to adjust the balance between explorer and exploitation in algorithms and the inability to perform appropriate parameter settings may cause the problem of local optimum and early convergence. Therefore, it is essential to adjust the selection mechanisms and algorithm parameters in algorithms. For this purpose, an algorithm based on the Coyote Optimization algorithm, inspired by the coyote's social status and adaptation behaviors in the packs, was developed for vehicle clutch elements. An improved algorithm was obtained by improving the diversity and search mechanism of the coyote optimization algorithm. The resulting improved algorithm was tested with constrained and unconstrained engineering problems, and its validity was proven by applying it to vehicle clutch elements. Test results of the suggested algorithm were checked against other algorithms in the literatüre to measure the efficiency of the novel algorithm. The experimental results show that the novel algorithm has significantly better efficiency on constrained and unconstrained test functions. This optimization study introduced a new diaphragm model that performs better in more dynamic conditions than the specified minimum disengagement load compared to the original model of the vehicle's clutch elements diaphragm spring. In the second optimization study, a damper model with better damping characteristics was obtained than the original model of the vehicle clutch damper element. Thus, the coyote optimization algorithm developed is the first in the literature and optimized vehicle clutch elements.

Key words: Nature-inspired algorithms, coyote optimization algorithm, diaphgram spring, clutch, system optimization, real parameter optimization
2022, vii + 202 pages.

ÖNSÖZ ve TEŞEKKÜR

Bu tezin hazırlanması sürecinde yol gösteren, bilgi ve tecrübesi ile desteğini esirgemeyen tez danışmanım Prof. Dr. Ali Rıza YILDIZ'a en içten teşekkürlerimi sunarım.

Tez komitemde yer alan değerli hocalarım Doç. Dr. Hüseyin LEKESİZ ve Prof. Dr. Rukiye ERTAN'a bu süreçte destekleri için teşekkürlerimi sunarım.

Tez çalışmasının bir kısmında konu olan ve Teydeb-1505 / 5160065 numaralı 'Yapay Zeka Yöntemleri ile Değişken Devir Altında Yüksek Performanslı ve Katılıklı Debriyaj Diyaframı Geliştirilmesi ve Prototip Üretimi' proje kapsamında her türlü desteği için Tübitak'a teşekkürlerimi sunarım.

Desteği ve anlayışı ile yoğun tez çalışmalarım süresince verdiği destek, gösterdiği sabır ve fedakarlık ile her zaman yanımda olan sevgili eşim Hatice Hilal KARADUMAN'a ve bu çalışma süresince beni destekleyen aileme sonsuz teşekkürlerimi sunarım.

Alper KARADUMAN
…/…/…….

İÇİNDEKİLER

Sayfa

ÖZET....vi
ABSTRACT....vii
ÖNSÖZ ve TEŞEKKÜR....viii
SİMGELER ve KISALTMALAR DİZİNİ....xi
ŞEKİLLER DİZİNİ....xiv
ÇİZELGELER DİZİNİ....xxi
1. GİRİŞ....1
2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI....9
2.1. Sürü Tabanlı Sezgisel Algoritmalar....9
2.2. Kır Kurdu Optimizasyon Algoritması (COA) ve İlgili Çalışmalar....11
2.3. Taşıt Debriyaj Elemanlarının Sezgisel Algoritmalar Kullanılarak Optimizasyonu ile İlgili Çalışmalar....18
2.4. Tez Çalışmasının Literatüre Katkısı ve Diğer Çalışmalardan Farkı....19
3. MATERYAL ve YÖNTEM....21
3.1. Taşıt Debriyaj Sistemi....22
3.1. Kır Kurdu Algoritması....26
3.2. Kır Kurdu Algoritmasında Önerilen İyileştirme Mekanizmaları....36
3.2.1. Kaotik haritalı mekanizmalar....36
3.2.1.1. Chebyshev harita....37
3.2.1.2. Çember harita....37
3.2.1.3. Gaus harita....38
3.2.1.4. Iteratif harita....38
3.2.1.5. Lojistik harita....39
3.2.1.6. Parçalı harita....39
3.2.1.7. Sinüs harita....40
3.2.1.8. Singer harita....40
3.2.1.9. Sinüsoidal harita....41
3.2.1.10. Çadır harita....41
3.2.2. Kaotik bölgesel arama mekanizması....42
3.2.3. Karşıt tabanlı öğrenme mekanizması (obl)....43
3.2.4. Uyumluluk uzaklık dengesi (fdb) seçim mekanizması....44
3.2.5. Levy uçuş mekanizması....47
3.2.6. Laplace crossover mekanizması....48
3.3. Geliştirilmiş Kır Kurdu Algoritması....49
3.4. Geliştirilen Yeni Kır Kurdu Algoritmasının Testleri....58
3.4.1. Kısıtlı mühendislik tasarım problemlerinde performans analizi....61
3.4.1.1. Kaynaklı kiriş tasarım problemi (mtp1)....63
3.4.1.2. Bası yayı tasarım problemi (mtp2)....65
3.4.1.3. Basınçlı kap problemi (mtp3)....66
3.4.1.4. Hız düşürücü problemi (mtp4)....67
3.4.1.5. Dişli güç iletim problemi (mtp5)....68
3.4.1.6. Üç çubuk kafes yapı tasarım problemi (mtp6)....69
3.4.1.7. Konsol kiriş tasarım problemi (mtp7)....70
3.4.1.8. Çoklu disk kavramalı fren tasarımının optimizasyon problemi (mtp8)....71
3.4.1.9. Himmelblau nonlinear problemi (mtp9)....73
3.4.1.10. Küresel rulman tasarım problemi (mtp10)....74

3.4.2. Diyafram yayının optimizasyonu 75
3.4.2.1. Diyafram yayının optimizasyonu malzeme ve yöntem 77
3.4.2.2. Diyafram yayının optimizasyonu için tasarım havuzu 81
3.4.2.3. Fem sonlu elemanlar yöntemi 84
3.4.2.4. Optimum boyutlarda ve yük-deplasman karakteristiklerinde diyafram üretimi 85
3.4.2.5. Optimum diyafram için serbest yatak özellikleri testi 86
3.4.3. Taşıt debriyaj damper elemanının modellenmesi 86
3.4.3.1. Geliştirilmiş kır kurdu algoritması ile taşıt debriyaj damper elemanı sistem optimizasyonu 88
4. BULGULAR ve TARTIŞMA 92
4.1. Diyafram Parametre Etki Analizi Sonuçları 147
4.2. Diyafram Optimum Tasarım ve Prototip Üretimi 148
4.3. Diyafram Serbest Rulman Yük Taşıyan Yer Değiştirme Testi Sonuçları 150
4.4. Debriyaj Sisteminin Bir Boyutlu Modelinin Optimizasyon Sonuçları 152
5. SONUÇ 156
KAYNAKLAR 158
EKLER 176
EK 1 176
EK 2 198
EK 3 198
EK 4 199
EK 5 199
EK 6 200
ÖZGEÇMİŞ 201

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
a	Pozisyon Parametresi
b	Ölçek Parametresi
c	Kır Kurdu Sayısı
D	Parametre Sayısı
E	Young Modülü
g,p	Uyum Değeri
i	İterasyon Sayısı
j	Kontrol Değişkeni
L	Levy Dağılımı
m	Problem Boyutu
mk	Eşitlik Kısıtlarının Sayısı
N	Sürtünme Yüzey Sayısı
n	Popülasyon Sayısı
nk	Eşitsizlik Kısıtlarının Sayısı
O	Sıralanmış Sosyal Durum
P	Baskı Kuvveti
p	Sürü Sayısı
R	Tork İletiminin Yapıldığı Ortalama Balata Çapı
R	Gerilme Oranı
s	Minimum Örnek Boyutu
T	İletilen Tork Miktarı
t	Anlık Zaman
u	Rastgele Üretilen Sayı
w	Uygunluk Değeri Üzerindeki Etki Katsayısı
X	Çözüm Vektörü
γ	Poisson Oranı
μ	Minimum Adım Boyutu
r_R	Spearman Korelasyon Katsayısı

Kısaltmalar	Açıklama
ABC	Yapay Arı Kolonisi Optimizasyon Algoritması
ACO	Karınca Sürüsü Optimizasyon Algoritması
ADEL	Değiştirilmiş Diferansiyel Evrim Algoritması
ALM	Genişletilmiş Lagrange Tabanlı Optimizasyon Algoritması
APM	Uyarlanabilir Ceza Metotlu Genetik Algoritma
APSO	Uyarlanabilir Parçacıklı Sürü Algoritması
BA	Yarasa Algoritması
BFO	Bakteriyel Yiyecek Arama Optimizasyon Algoritması
BSA	Geri İzleme Arama Optimizasyonu Algoritması
CAEP	Diferansiyel Evrimli Kültürel Algoritma
ChOA	Şempanze Optimizasyon Algoritması
COA	Kır kurdu optimizasyon algoritması
CPA	Etçil Bitki Algoritması

CPSO	Evrimsel Parçacık Sürü Algoritması
CSA	Guguk Kuşu Optimizasyon Algoritması
CSGA	Hibrit Guguk Kuşu Genetik Optimizasyon Algoritması
DA	Yusufçuk Optimizasyon Algoritması
dBA	Yönlü Yarasa Algoritması
DE	Diferansiyel Gelişim Algoritması
DEDS	Dinamik Stokastik Seçimli Diferansiyel Evrim Algoritması
DELC	Seviye Karşılaştırmalı Diferansiyel Evrim Algoritması
Dp	Öklid Mesafesi
EA	Basit Evrimsel Algoritma
E-FA	Geliştirilmiş Ateş Böceği Algoritması
ES	Evrimsel Stratejiler
FA	Ateş Böceği Algoritması
FDB	Uyumluluk Uzaklık Dengesi
FE	Sonlu Elemanlar
FEM	Sonlu Elemanlar Metodu
FOA	Meyve Sineği Optimizasyon Algoritması
FSO	Uçan Sincap Optimizasyon Algoritması
GA	Genetik Algoritma
GA2	Kendinden Uyarlamalı Genetik Algoritma
GA3	Hakimiyete Dayalı Turnuva Seçimli Genetik Algoritma
GA-AIS	Hibrit Genetik Algoritma
GCOA	Geliştirilmiş Kır Kurdu Optimizasyon Algoritması
GRG	Genelleştirilmiş İndirgenmiş Gradyan Yöntemi
GWO	Gri Kurt Optimizasyon Algoritması
HHO	Harris Şahinleri Optimizasyon Algoritması
HS	Harmoni Arama Algoritması
IABC	Geliştirilmiş Yapay Arı Kolonisi Optimizasyon Algoritması
IMDDE	Yeni Diferansiyel Evrim Algoritması
ISA	İç Araştırma Algoritması
Lb	Alt Sınır
lb_j	Alt Sınır
LC	Laplace Crossover
LCA	Lig Şampiyonluk Algoritması
MBA	Mayın Patlaması Optimizasyon Algoritması
MFO	Güve Alevi Optimizasyon Algoritması
MPSO	Değiştirilmiş Parçacıklı Sürü Algoritması
MVO	Çoklu Evren Algoritması
Nc	Kır Kurdu Sayısı
NDE	Yeni Diferansiyel Evrim Algoritması
NM-PSO	Evrimsel Parçacık Sürü Algoritması
$normD_{P[i]}$	Normalize Uzaklık
$normF_{[i]}$	Normalize Uyumluluk
Np	Sürü Sayısı
NSGA-II	Hızlı Elit Genetik Algoritma
OBL	Karşıt Tabanlı Öğrenme Mekanizması
OBSSA	Karşıt Tabanlı Salp Sürüsü Optimizasyon Algoritması
PSGA	Genetik Algoritma Destekli Parçacıklı Sürü Optimizasyon Algoritması

PSO	Parçacıklı Sürü Optimizasyon
PSO-DE	Hibrit Diferansiyel Evrim Parçacıklı Sürü Algoritması
PVS	Geçen Araç Arama Algoritması
r_i	Rastgele [0,1] Arası Üretilen Sayı
QPSO	Gauss Kuantum Davranışlı Parçacık Sürü Optimizasyonu
SA	Tavlama Benzetim Algoritması
SCA	Sinüs Kosinüs Algoritması
SEM	Sonlu Elemanlar Modeli
$S_{P[i]}$	Uygunluk Uzaklık Dengesi Puanı
s_1	Bireysel Uygunluk Uzaklık Dengesi Puanı
SR	Stokastik Sıralama Algoritması
SSA	Salp Sürüsü Optimizasyon Algoritması
STA	İki Aşamalı Durum Geçiş Algoritması
TCA	T Cell Algoritması
TLBO	Öğretme Öğrenmeye Dayalı Optimizasyon Algoritması
Ub	Üst Sınır
ub_j	Üst Sınır
UFA	Yükseltilmiş Ateş Böceği Algoritması
x_g (k)	Mevcut En İyi Kır Kurdudur
x_g^l (k)	Yeni En İyi Kır Kurdudur
WCA	Su Döngüsü Algoritması
WOA	Balina Optimizasyon Algoritması
z^j (k)	Kaotik Haritalar İle Rastgele [0,1] Arası Üretilen Sayı

ŞEKİLLER DİZİNİ

		Sayfa
Şekil 2.1.	Doğadan esinlenmiş algoritmalar…….……..	9
Şekil 3.1.	Temsili debriyaj sistemi……..……	22
Şekil 3.2.	Temsili debriyaj sistemi montajı…….……	23
Şekil 3.3.	Debriyaj baskı komplesi……..…	23
Şekil 3.4.	Diyafram yük eğrileri a)Rulman yükü b) Diyafram yükü……..	24
Şekil 3.5.	Debriyaj sistemi a) Kavrama durumu b) Ayırma durumu……..	24
Şekil 3.6.	Temsili damper komplesi karakteristiği……..…….....	25
Şekil 3.7.	Temsili debriyaj sisteminde titreşimin sönümlenmesi…….……...	25
Şekil 3.8.	Kır kurdu model şeması…….……....	29
Şekil 3.9.	Kır kurdu algoritması akış şeması…….……..	34
Şekil 3.10.	Kır kurdu algoritması akış şeması…….……..	35
Şekil 3.11.	Chepyshev harita grafiği…….……	37
Şekil 3.12.	Çember harita grafiği…….…….....	37
Şekil 3.13.	Gaus harita grafiği……..…….....	38
Şekil 3.14.	Iteratif harita grafiği……..……..	38
Şekil 3.15.	Lojistik harita grafiği……..…….	39
Şekil 3.16.	Parçalı harita grafiği……..……..	39
Şekil 3.17.	Sinüs harita grafiği……..……	40
Şekil 3.18.	Singer harita grafiği……..……...	40
Şekil 3.19.	Sinüsoidal harita grafiği……..	41
Şekil 3.20.	Çadır harita grafiği……..…	41
Şekil 3.21.	Karşıt tabanlı öğrenme mekanizması…….….	43
Şekil 3.22.	Kalın bir nokta ile işaretlenmiş başlangıç noktasından başlayarak elli ardışık adımdan oluşan Lévy uçuşları……..	47
Şekil 3.23.	Geliştirilmiş Kır kurdu algoritması akış şeması……...	57
Şekil 3.24.	Kaynaklı kiriş tasarımı……..……..	63
Şekil 3.25.	Bası yayı tasarım problemi……..……	65
Şekil 3.26.	Basınçlı kap tasarım problemi……..……...	66
Şekil 3.27.	Hız düşürücü tasarım problemi…….……..	67
Şekil 3.28.	Dişli güç iletim tasarım problemi…….....	68
Şekil 3.29.	Üç çubuk kafes yapı tasarım problemi…….....	70
Şekil 3.30.	Konsol kiriş tasarım problemi…….……	70
Şekil 3.31.	Çoklu disk kavramalı fren (mdcb) tasarımı problemi……..	72
Şekil 3.32.	Küresel rulman tasarım problemi……..…..	74
Şekil 3.33.	Baskı plakası ve diyafram elemanının şematik gösterimi A) Baskı plakası montaj komplesi B) Diyafram yay tasarım parametreleri ……..	78
Şekil 3.34.	Diyafram fea modeli ve malzemesi şematik gösterimi A) Simetrik diyafram modeli için ağ yapısı, sınır ve yükleme koşulları B) Farklı test sıcaklık değerlerine göre malzeme gerilim – gerinim davranışı …….	84
Şekil 3.35.	En iyilenmiş diyaframın üretim adımları……..…..	85
Şekil 3.36.	Debriyaj sisteminin 1 boyutlu sistem modellenmesi……...	88

Şekil 3.37. Marş motoru karakteristiği……..…… 89
Şekil 3.38. Zamana bağlı motor hız eğrisi……..….... 90
Şekil 3.39. Zamana bağlı motor tork eğrisi……...…. 92
Şekil 3.40. Amesim entegreli yapay zeka tabanlı sistem akış şeması…… 91
Şekil 4.1. Önerilen kır kurdu algoritmasının sözde kodu (GCOA6)…… 94
Şekil 4.2. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F1 test fonksiyonu için karşılaştırması …… 95
Şekil 4.3. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F2 test fonksiyonu için karşılaştırması …… 95
Şekil 4.4. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F3 test fonksiyonu için karşılaştırması……..……...…… 96
Şekil 4.5. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F4 test fonksiyonu için karşılaştırması …….……...… 96
Şekil 4.6. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F5 test fonksiyonu için karşılaştırması……...…….. 97
Şekil 4.7. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F6 test fonksiyonu için karşılaştırması……. 97
Şekil 4.8. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F5 test fonksiyonu için karşılaştırması.……...……. 98
Şekil 4.9. COA ve GCOA ’no:6’ için F1 test fonksiyonu şematiği A) F1 test fonksiyonu parametre uzayı B) F1 test fonksiyonu için yakınsama eğrisi……..…… 100
Şekil 4.10. COA ve GCOA ’no:6’ için F2 test fonksiyonu şematiği A) F2 test fonksiyonu parametre uzayı B) F2 test fonksiyonu için yakınsama eğrisi……...……..……. 100
Şekil 4.11. COA ve GCOA ’no:6’ için F3 test fonksiyonu şematiği A) F3 test fonksiyonu parametre uzayı B) F3 test fonksiyonu için yakınsama eğrisi……...……..……. 101
Şekil 4.12. COA ve GCOA ’no:6’ için F4 test fonksiyonu şematiği A) F4 test fonksiyonu parametre uzayı B) F4 test fonksiyonu için yakınsama eğrisi……...……..……. 101
Şekil 4.13. COA ve GCOA ’no:6’ için F5 test fonksiyonu şematiği A) F5 test fonksiyonu parametre uzayı B) F5 test fonksiyonu için yakınsama eğrisi……..……... 101
Şekil 4.14. COA ve GCOA ’no:6’ için F6 test fonksiyonu şematiği A) F6 test fonksiyonu parametre uzayı B) F6 test fonksiyonu için yakınsama eğrisi……..……... 102
Şekil 4.15. COA ve GCOA ’no:6’ için F7 test fonksiyonu şematiği A) F7 test fonksiyonu parametre uzayı B) F7 test fonksiyonu için yakınsama eğrisi……..……..….. 102
Şekil 4.16. F1 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi ……...…….. 103
Şekil 4.17. F2 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..… 103
Şekil 4.18. F3 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi…..……..….. 104
Şekil 4.19. F4 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……...……... 104

Şekil 4.20. F5 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi ……..….... 105
Şekil 4.21. F6 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi ……..….... 105
Şekil 4.22. F7 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi ……. 106
Şekil 4.23. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F1 fonksiyonu için karşılaştırma B) F2 fonksiyonu için karşılaştırma……..…….. 106
Şekil 4.24. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F3 fonksiyonu için karşılaştırma B) F4 fonksiyonu için karşılaştırma…… 106
Şekil 4.25. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F5 fonksiyonu için karşılaştırma B) F6 fonksiyonu için karşılaştırma……... 107
Şekil 4.26. Rakip algoritmalarla karşılaştırılmalı tek modlu F7 kıyaslama fonksiyonunun değer dağılım kutu grafiği……. 107
Şekil 4.27. F8 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 109
Şekil 4.28. F9 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 110
Şekil 4.29. F10 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 110
Şekil 4.30. F11 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..… 111
Şekil 4.31. F12 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..……..……. 111
Şekil 4.32. Şekil 4.32. F13 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi...……..……. 112
Şekil 4.33. COA ve GCOA ’no:6’ için F8 test fonksiyonu şematiği
A) F8 test fonksiyonu parametre uzayı B) F8 test fonksiyonu için yakınsama eğrisi……..……..……. 112
Şekil 4.34. COA ve GCOA ’no:6’ için F9 test fonksiyonu şematiği
A) F9 test fonksiyonu parametre uzayı B) F9 test fonksiyonu için yakınsama eğrisi ……..... 112
Şekil 4.35. COA ve GCOA ’no:6’ için F10 test fonksiyonu şematiği
A) F10 test fonksiyonu parametre uzayı B) F10 test fonksiyonu için yakınsama eğrisi……...… 113
Şekil 4.36. COA ve GCOA ’no:6’ için F11 test fonksiyonu şematiği
A) F11 test fonksiyonu parametre uzayı B) F11 test fonksiyonu için yakınsama eğrisi……...…… 113
Şekil 4.37. COA ve GCOA ’no:6’ için F12 test fonksiyonu şematiği
A) F12 test fonksiyonu parametre uzayı B) F12 test fonksiyonu için yakınsama eğrisi……...…… 113

Şekil 4.38. COA ve GCOA 'no:6' için F13 test fonksiyonu şematiği
A) F13 test fonksiyonu parametre uzayı B) F13 test fonksiyonu için yakınsama eğrisi……...… 114
Şekil 4.39. F8 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 114
Şekil 4.40. F9 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 114
Şekil 4.41. F10 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..……..…….. 115
Şekil 4.42. F11 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..……... 115
Şekil 4.43. F12 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi..…… 116
Şekil 4.44. F13 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 116
Şekil 4.45. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F8 fonksiyonu için karşılaştırma B) F9 fonksiyonu için karşılaştırma……..…….. 117
Şekil 4.46. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F10 fonksiyonu için karşılaştırma B) F11 fonksiyonu için karşılaştırma…….……... 117
Şekil 4.47. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F12 fonksiyonu için karşılaştırma B) F13 fonksiyonu için karşılaştırma…… 117
Şekil 4.48. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F14 test fonksiyonu için karşılaştırması ……..... 120
Şekil 4.49. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F15 test fonksiyonu için karşılaştırması …… 120
Şekil 4.50. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F16 test fonksiyonu için karşılaştırması…….…… 121
Şekil 4.51. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F17 test fonksiyonu için karşılaştırması……. 121
Şekil 4.52. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F18 test fonksiyonu için karşılaştırması……..……... 122
Şekil 4.53. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F19 test fonksiyonu için karşılaştırması……. 122
Şekil 4.54. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F20 test fonksiyonu için karşılaştırması…….…… 123
Şekil 4.55. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F21 test fonksiyonu için karşılaştırması……. 123
Şekil 4.56. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F22 test fonksiyonu için karşılaştırması……. 124
Şekil 4.57. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F23 test fonksiyonu için karşılaştırması…… 124

Şekil 4.58. COA ve GCOA ’no:6’ için F14 test fonksiyonu şematiği A) F14 test fonksiyonu parametre uzayı B) F14 test fonksiyonu için yakınsama eğrisi……...…… 125
Şekil 4.59. COA ve GCOA ’no:6’ için F15 test fonksiyonu şematiği A) F15 test fonksiyonu parametre uzayı B) F15 test fonksiyonu için yakınsama eğrisi……...…… 125
Şekil 4.60. COA ve GCOA ’no:6’ için F16 test fonksiyonu şematiği A) F16 test fonksiyonu parametre uzayı B) F16 test fonksiyonu için yakınsama eğrisi……...…… 125
Şekil 4.61. COA ve GCOA ’no:6’ için F17 test fonksiyonu şematiği A) F17 test fonksiyonu parametre uzayı B) F17 test fonksiyonu için yakınsama eğrisi……...… 126
Şekil 4.62. COA ve GCOA ’no:6’ için F18 test fonksiyonu şematiği A) F18 test fonksiyonu parametre uzayı B) F18 test fonksiyonu için yakınsama eğrisi……... 126
Şekil 4.63. COA ve GCOA ’no:6’ için F19 test fonksiyonu şematiği A) F19 test fonksiyonu parametre uzayı B) F19 test fonksiyonu için yakınsama eğrisi……...… 126
Şekil 4.64. COA ve GCOA ’no:6’ için F20 test fonksiyonu şematiği A) F20 test fonksiyonu parametre uzayı B) F20 test fonksiyonu için yakınsama eğrisi……... 127
Şekil 4.65. COA ve GCOA ’no:6’ için F21 test fonksiyonu şematiği A) F21 test fonksiyonu parametre uzayı B) F21 test fonksiyonu için yakınsama eğrisi……...… 127
Şekil 4.66. COA ve GCOA ’no:6’ için F22 test fonksiyonu şematiği A) F22 test fonksiyonu parametre uzayı B) F22 test fonksiyonu için yakınsama eğrisi……...…… 127
Şekil 4.67. COA ve GCOA ’no:6’ için F22 test fonksiyonu şematiği A) F22 test fonksiyonu parametre uzayı B) F22 test fonksiyonu için yakınsama eğrisi……...…… 128
Şekil 4.68. F14 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 128
Şekil 4.69. F15 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 128
Şekil 4.70. F16 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 129
Şekil 4.71. F17 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 129
Şekil 4.72. F18 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 130
Şekil 4.73. F19 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 130
Şekil 4.74. F20 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 131
Şekil 4.75. F21 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 131
Şekil 4.76. F22 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi……..…… 132

Şekil 4.77. F23 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğris.…….....…… 132
Şekil 4.78. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F14 fonksiyonu için karşılaştırma B) F15 fonksiyonu için karşılaştırma……..…….. 133
Şekil 4.79. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F16 fonksiyonu için karşılaştırma B) F17 fonksiyonu için karşılaştırma……..…….. 133
Şekil 4.80. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F18 fonksiyonu için karşılaştırma B) F19 fonksiyonu için karşılaştırma…… 133
Şekil 4.81. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F20 fonksiyonu için karşılaştırma B) F21 fonksiyonu için karşılaştırma…… 134
Şekil 4.82. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği
A) F22 fonksiyonu için karşılaştırma B) F23 fonksiyonu için karşılaştırma…… 134
Şekil 4.83. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi
A) Mtp1 kaynaklı kiriş tasarım problemi B) Mtp2 bası yayı tasarım problemi……..…….. 143
Şekil 4.84. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi
A) Mtp3 basınçlı kap tasarım problemi B) Mtp4 hız düşürücü tasarım problemi…….……. 144
Şekil 4.85. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi
A) Mtp5 dişli güç iletim tasarım problemi B) Mtp6 üç çubuk kafes yapı problemi…….…… 144
Şekil 4.86. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi
A) Mtp7 konsol kiriş tasarım problemi B) Mtp8 çoklu disk kavramalı fren problemi……..…… 145
Şekil 4.87. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi
A) Mtp9 himmelblau nonlinear problemi B) Mtp10 küresel rulman tasarım problemi…….…… 145
Şekil 4.88. Geliştirilmiş algoritma en iyi değer dağılımı
A) Mtp1 kaynaklı kiriş tasarım problemi B) Mtp2 bası yayı tasarım problemi…….…… 146
Şekil 4.89. Geliştirilmiş algoritma en iyi değer dağılımı
A) Mtp3 basınçlı kap tasarım problemi B) Mtp4 hız düşürücü tasarım problemi…….…… 146

Şekil 4.90. Geliştirilmiş algoritma en iyi değer dağılımı
A) Mtp5 dişli güç iletim tasarım problemi B) Mtp6 üç çubuk kafes yapı problemi……. 146
Şekil 4.91. Geliştirilmiş algoritma en iyi değer dağılımı
A) Mtp7 konsol kiriş tasarım problemi B) Mtp8 Çoklu disk kavramalı fren problemi…….. 147
Şekil 4.92. Geliştirilmiş algoritma en iyi değer dağılımı
A) Mtp9 himmelblau nonlinear problemi B) Mtp10 küresel rulman tasarım problemi……. 147
Şekil 4.93. Ansys optiSlang yazılımına göre parametre etki analizi sonuçları…….. 148
Şekil 4.94. Prototip sonuçları
A) Eniyilenmiş tasarımın prototip resmi B) Baskı yük-yerdeğiştirme karakteristikleri……... 149
Şekil 4.95. Referans (noktalı çizgi) ve eniyilenmiş (düz çizgi) diyaframın ayırma rulman yükünün rulman yerdeğiştirmesine göre statik ve dinamik (6500 rpm) değişimi……... 150
Şekil 4.96. Optimize edilmiş damper tork & açı eğrisi……... 153
Şekil 4.97. Hedeflenen sönümleme eğrisi ile optimizasyon sonrası çıkan eğri arasındaki uyum…….. 153
Şekil 4.98. Optimize edilmiş damper ile sönümlenen tork…… 154
Şekil 4.99. Motor hızı ile damper hızlanması karakteristiği…….. 154
Şekil 4.100. Vites kutusu hızı ile damper hızlanması karakteristiği…… 155

ÇİZELGELER DİZİNİ

			Sayfa
Çizelge	1.1.	Mevcut ve hibrit algoritmaların araştırılması……..	7
Çizelge	1.1.	Mevcut ve hibrit algoritmaların araştırılması (devam)……..	8
Çizelge	2.1.	Hibrit ve standart Coa varyantları ile ilgili literatür araştırması……..	16
Çizelge	2.1.	Hibrit ve standart Coa varyantları ile ilgili literatür araştırması (devam)……..	17
Çizelge	2.1.	Hibrit ve standart Coa varyantları ile ilgili literatür araştırması (devam)……..	18
Çizelge	3.1.	Kır kurdu algoritmasının literatürdeki sınıflandırması……	31
Çizelge	3.2.	Kaotik haritalar……	43
Çizelge	3.3.	Kır kurdu optimizasyon algoritması tabanlı oluşturulan algoritmalar…….	58
Çizelge	3.4.	Algoritma performans testleri için parametre ayarı……	59
Çizelge	3.5.	Çalışmalarda kullanılan kısıtsız tek modlu kıyaslama fonksiyonları……..	60
Çizelge	3.6.	Sabit çok modlu test fonksiyonu……..	60
Çizelge	3.7.	Sabit boyutlu test fonksiyonu……..	61
Çizelge	3.8.	Mühendislik tasarım problem verileri…….	63
Çizelge	3.9.	Diyafram için tasarım değişkenleri ve limitleri……..	79
Çizelge	3.10.	LHS'ye göre belirlenen tasarım noktaları……	82
Çizelge	3.10.	LHS'ye göre belirlenen tasarım noktaları (devam)…….	83
Çizelge	3.11.	Amesim model parametreleri……..	89
Çizelge	4.1.	Tek modlu test fonksiyonları……..	98
Çizelge	4.1.	Tek modlu test fonksiyonları (devam)……	99
Çizelge	4.2.	Tek modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (No: 6)……	99
Çizelge	4.3.	Tek modlu kıyaslama fonksiyon sonuçları (No: 6)……..	99
Çizelge	4.4.	Tek modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması……..	99
Çizelge	4.4.	Tek modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması (devam)……	100
Çizelge	4.5.	Sabit çok modlu test fonksiyonları……..	107
Çizelge	4.5.	Sabit çok modlu test fonksiyonları (devam)……..	108
Çizelge	4.6.	Sabit çok-modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (GCOA6)……..	108
Çizelge	4.7.	Sabit çok-modlu benchmark test fonksiyonları için sonuçlar (GCOA6)……	108
Çizelge	4.8.	Sabit çok-modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması (GCOA6)…….	109
Çizelge	4.9.	Sabit modlu test fonksiyonları…….	118
Çizelge	4.10.	Sabit modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (GCOA6)……	119
Çizelge	4.11.	Sabit-mod benchmark test fonksiyonları için sonuçlar (GCOA6)……	119

Çizelge 4.12. Sabit-mod benchmark test fonksiyonları için karşılaştırmalı sonuçlar (GCOA6)…….. 119
Çizelge 4.13. Geliştirilmiş algoritmanın (GCOA) mühendislik tasarım problemi…….. 134
Çizelge 4.14. MTP1-MTP10 zaman karşılaştırması……. 134
Çizelge 4.14. MTP1-MTP10 zaman karşılaştırması (devam).……. 135
Çizelge 4.15. Kaynaklı kiriş tasarım problemi için önerilen algoritmanın literatürdeki mevcut algoritma ile istatiksel kıyaslanması….. 135
Çizelge 4.15.1. Kaynaklı kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 135
Çizelge 4.16. Bası yay tasarım problemi için önerilen algoritmanın literatürdeki mevcut algoritma ile istatiksel kıyaslanması….. 136
Çizelge 4.16.1. Bası yay tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 136
Çizelge 4.17. Basınçlı kap tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması……... 137
Çizelge 4.17.1. Basınçlı kap tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 137
Çizelge 4.18. Hız düşürücü tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması……... 138
Çizelge 4.18.1. Hız düşürücü tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 138
Çizelge 4.19. Dişli güç iletim tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması……... 138
Çizelge 4.19.1. Dişli güç iletim tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 139
Çizelge 4.20. Üç çubuk kafes yapı tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması….. 140
Çizelge 4.20.1. Üç çubuk kafes yapı tasarım problemi (Şekil 3.29) için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…… 139
Çizelge 4.20.1. Üç çubuk kafes yapı tasarım problemi (Şekil 3.29) için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması (devam).…… 140
Çizelge 4.21. Konsol kiriş tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması……... 140
Çizelge 4.21.1. Konsol kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 140
Çizelge 4.21.1. Konsol kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması (devam)…….. 141
Çizelge 4.22. Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi için önerilen algoritmanın mevcut algoritma ile kıyaslanması……... 141
Çizelge 4.22.1. Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 141
Çizelge 4.23. Himmelblau nonlinear problemi için önerilen algoritmanın mevcut algoritma ile kıyaslanması…….. 142

Çizelge 4.23.1. Himmelblau nonlinear problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 142
Çizelge 4.24. Küresel rulman tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması……... 142
Çizelge 4.24.1. Küresel rulman tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması…….. 143
Çizelge 4.25. Optimum tasarım değişkenleri ve elde edilen amaç fonksiyon değerleri…….. 149
Çizelge 4.26. Optimum tasarım değişkenleri ve elde edilen kavrama katılık değerleri…….. 151
Çizelge 4.27. Geliştirilmiş kır kurdu algoritması ile eniyilenmiş model parametreleri……... 152

1. GİRİŞ

Optimizasyon, belirli koşullar için bir problemin parametrelerinin (işçilik, süreç, boyut, ağırlık gibi) kısıtlamalarını ihlal etmeden, alternatifler arasından en iyi parametreleri seçerek belirli bir amaca ulaşan bir süreci tanıtır. Optimizasyon süreci, problemin temel özelliklerini ve problemin amacını içeren modelleme ile başlar. Modelleme, sistemin sürecini, amacını ve diğer sistemlerle etkileşimini yansıtan matematiksel ifadelerden oluşur. Ardından optimizasyon işlemi, optimize edilecek sistemi tanımlayan tasarım değişkenleri olan problem parametrelerinin karar süreci ile devam eder. Başka bir deyişle, kısıtlar altındaki bir optimizasyon probleminin düzinelerce çözümünün sonuçları arasından mümkün olan veya kabul edilebilir en iyi sonucu bulma işlemine en iyileme (optimizasyon) denir. En iyileme problemi aşağıdaki gibi ifade edilebilir:

Amaç: En az/çok f(x)
Kısıtlar:

$$g_i(x) \leq 0; i = 1.,2.,\dots,p \qquad \text{(eşitsizlik kısıtı)} \tag{1.1}$$

$$h_j(x) = 0; j = 1.,2.,\dots,q \qquad \text{(eşitlik kısıtı)} \tag{1.2}$$

$$\vec{x} = (x_1.,x_2.,\dots,x_D), \qquad \text{(Tasarım değişkenleri)} \tag{1.3}$$

Optimizasyon problemlerinde araştırma uzayı içindeki sonsuz sayıda çözüm adayı arasındaki en iyi çözüm en iyileme algoritmaları ile bulunur. Konusu olan meta-sezgisel algoritmaların geliştirilmesi optimizasyon sürecinin veya algoritmanın kapsamı için kritik öneme sahiptir. Meta-sezgisel algoritmalar doğada bulunan biyolojik evrimi taklit eden evrimsel temelli, fiziksel yasaları taklit eden fizik temelli ve gruplar halinde yaşayan bir canlının özelliklerini taklit eden sürü temelli olarak algoritmalar geliştirilmiştir. Birçok yinelemede problem çözümleri geliştiren algoritmaların en iyi çözüm adayını seçmesi ve en iyi çözüm için çözüm adaylarının birleştirilmesi ilkesine dayanan evrimsel algoritmalar, seçme, birleştirme ve mutasyon ilkelerine göre çalışır. Genetik algoritmalar, diferansiyel geliştirme algoritmaları ve evrim stratejileri evrimsel algoritmalardandır. Fiziksel yasalardan ve teorilerden veya fiziksel olaylardan esinlenen büyük patlama algoritması (BBCA), yerçekimi arama algoritması (GSA) ve benzetilmiş tavlama algoritmaları (SA) gibi fiziksel algoritmalar vardır. Karınca kolonisi optimizasyon

algoritması, çakal optimizasyon algoritması, yapay arı kolonisi algoritması gibi sürülerin davranışlarından ilham alan algoritmalar, sürü tabanlı algoritmalara örnektir. Örneğin Karınca koloni optimizasyonu (ACO), doğada yaşayan karıncaları ve karınca yiyecek arama yöntemlerini taklit eder. Guguk Kuşu Arama Algoritması (CSA), guguk kuşunun parazit üreme sürecini yansıtır. Yapay Arı Kolonisi (ABC) algoritması, arı sürüsünün davranışlarından ilham alır. Ek olarak, öğretme-öğrenme algoritmaları, ideoloji algoritmaları ve insan kentleşme algoritmaları gibi insanların davranışlarından ilham alan algoritmalar, insan davranışına dayalı olarak oluşturulur. Literatürdeki meta-sezgisel algoritmalar Tablo 1'de listelenmiştir. Optimizasyon problemleri, tasarım havuzunda minimum değeri veren bölgede bir tasarım bulmayı amaçlar. Amaç fonksiyonu, tasarım havuzunda yalnızca bir noktada veya birden fazla noktada global minimuma sahip olabilir. İşlev değeri, yerel minimum olarak adlandırılan belirli bir tasarım değişkeni etrafında minimum bir değere sahip olabilir. Ancak, birden fazla optimum değer varsa, çözüm havuzunun birden fazla yerel minimumu vardır. Problem çok sayıda denklem içerdiğinden amaç fonksiyonunun çok sayıda tasarım değişkeni ve kısıt fonksiyonu içerdiği durumlarda, en iyi çözüme ulaşmak için ilk tasarıma bir veya daha fazla değer verilerek en iyi koşullar elde edilene kadar iterasyon tekrar yapılır. Meta-sezgisel algoritmaların çözüm arayışı sürecinde algoritma içindeki çözüm bulucu birimlerin mümkün olduğu kadar çeşitli olması arama uzayındaki bilinmeyen bölgeleri keşfetmeyi mümkün kılar. Bununla birlikte çözüm bulucu birimlerden gelen bilgiden faydalanılarak çözüm arayışının sürdürülmesi meta-sezgisel algoritmalarda çok önemlidir. Ancak bu süreçte, özellikle birçok yerel en iyi noktaya sahip karmaşık problemlerle karşılaşıldığında, optimizasyon süreci yerel optimizasyona takılıp gerçek optimizasyona yakınsamayabilir. Bu nedenle, meta-sezgisel algoritmalarda, çözüm bulma birimlerinden gelen bilgileri kullanarak çözüm aramaya devam etmek çok önemlidir. Ancak meta-sezgisel algoritmaların başarı oranını arttıran en önemli faktör keşfetme sürecinin ve bilgiden yararlanmanın algoritma içinde dengede olmasıdır. Bu denge sayesinde bölgesel en iyi noktalarda takılmayıp küresel en iyi nokta bulanabilir. Keşif ve elde edilen bilginin kullanılması arasında dengenin sağlanması için algoritmalar üzerinde çalışmalar sürekli bir ihtiyaçtır. Bu nedenle metasezgisel algoritmalar, sağlamlık ve performans açısından uygun çözümler aramak için keşif ve kullanım arasında bir denge sağlamalıdır. Keşif, algoritmanın optimum çözümün dahil edilebileceği uygun bölgeleri keşfetme yeteneği

olarak ifade edilebilir. Sömürü, algoritmanın bu bölgeler üzerinde bölgesel bir keşif aramak için çözümü geliştirme yeteneğidir. Bu tür problemlerin üstesinden gelebilmek için, sömürü ve keşfi dengeleyebilen hibrit yapılar oluşturularak yeni optimizasyon yöntemlerinin geliştirilmesi zorunlu hale gelmiştir. Optimizasyon algoritmalarında yerel arama yeteneği ve global arama yeteneği kaçınılmazdır. Tasarım bölgesini keşfetmeye yönelik küresel arayışın ilk yinelemelerinde popülasyonun çeşitliliği mümkün olduğunca yüksektir. Daha sonra, en iyi çözümün elde edilebileceği yerel arama alanlarında iterasyonlar ilerledikçe popülasyon çeşitliliği azaltılmalıdır. Ek olarak, bir algoritmanın performansı genellikle yerel arama kabiliyetine ve küresel yakınsama kabiliyetine bağlıdır. Yerel arama, en iyi çözüme sonsuz derecede yaklaşmayı amaçlar. Bununla birlikte, küresel yakınsama yeteneği, optimal küresel çözümün yaklaşık konumunu bulmayı hedefler. Bu tezde Pierezan ve Coelho (2018) tarafından sunulan sürü zekası evrimsel buluşsal yöntemlerinden sınıflandırılan *canis latrans* kır kurdu türlerinin ölüm, beslenme ve üreme davranışlarından esinlenen popülasyon tabanlı kır kurtlarının sosyal organizasyonu ve çevreye uyumlarından ilham alınarak oluşturulan kır kurdu optimizasyon algoritmasının keşif ve elde edilen bilgi dengesinin geliştirilmesi sağlanarak yeni bir algoritma literatüre katılmıştır. Kır kurdu popülasyonları, küçük sürüleri, belirli bir sosyal yapıya sahip olmaları ve bölgesel olmaları, çevresel etkileri ve kır kurdu popülasyonu üzerindeki etkileri nedeniyle diğer popülasyonlardan farklıdır. Optimizasyon modeli, avlanma yerine sosyal yapıya ve kır kurtları arasında deneyim alışverişine bağlı uygunluk fonksiyonunu en aza indirir. Kır kurtları topluluğunda bireye temsil edilen çözümün uygunluğuna karşılık gelen sosyal durumun kalitesi ve uygun çözüm olarak ifade edilen sosyal durum, algoritmanın iki ana çözüm parametresidir. Ayrıca, her bir sürüdeki sürü sayısı ve kır kurdu sayısı olmak üzere sadece iki kontrol parametresine sahiptir ve böyle bir özellik, algoritmayı uygulamada çok basit hale getirir. Kır kurdu algoritması popülasyon üyelerinin etkileşiminden etkilenerek algoritmanın global arama kabiliyetini arttırdığı ve interaktif bir kültürel eğilime sahip olduğu için tercih edilmektedir. COA algoritması, sürüdeki çözüm arama sürecini sürü dışında çözüm arama stratejisiyle değiştirerek bu algoritmanın çalışma hızını artırmakta ve kır kurtları arasında bilgi paylaşarak popülasyonun çeşitliliğini artırmaktadır. Böylece, COA algoritması iyi izleme özellikleri, keşif ve kullanım aşamaları arasında bir denge sağlar. Ancak mevcut çakal algoritmasındaki keşif ve kullanım süreçlerindeki erken yakınsama

sorunu nedeniyle her nesilde elde edilen en iyi çözümler yeterince iyi değildir. Literatürdeki çalışmalar incelendiğinde COA, özellikle hibrit kıyaslama fonksiyonları ve birçok değişkeni içeren durumlara göre daha düşük ortalama performans elde etmiştir. COA algoritması global arama yeteneği açısından zayıftır ve çeşitlilik sağlanarak global arama yapıldığında yakınsama hızında yavaşlık sorunu ortaya çıkmaktadır. Başka bir deyişle, arama devam ederken kır kurdu optimizasyon algoritmasının (COA) çeşitli dezavantajları vardır. Bunlar, azaltılmış popülasyon çeşitliliği, yavaş yakınsama hızı, yerel optimuma kolay düşme ve kısıtlı optimizasyon problemini çözerken uygulanabilir bir çözüm elde edememe gibi durumları içerir. Bu nedenle, standart COA algoritmasının arama performansı güçlüdür, ancak geliştirme performansı zayıftır. COA algoritması, optimizasyon sürecinde keşif ve geliştirmeyi dengelemek için farklı yapılara, sorunlara ve mekanizmalara katkıda bulunsa da algoritmanın iyileştirilmesi gerekiyor. COA algoritması, sürü kültürel trendi aracılığıyla sürülerdeki kır kurtlarının ve kır kurtlarının sosyal konumlarının güncellenmesine rehberlik eder. Aramada kır kurdunun yol gösterici rolü etkili olduğu için sürülerdeki kır kurtları kolayca yönlendirilir. Kır kurtları tarafından en iyi yerel konum, algoritmanın erken yakınsamasına neden olur. COA algoritmasında, her yinelemede, alt popülasyondaki kır kurtları, alfa kır kurdu ve sürünün kültürel eğiliminin rehberliğinde her zaman tutarlı bir başarı sağlar. Yineleme ilerledikçe popülasyon benzerliği artar ve algoritmanın aranabilirliği zayıflar. Birden fazla yerel çözümdeki uç noktalar, alt popülasyondaki kır kurdu çözümlerinin en iyi yerel konumda olma olasılığını önemli ölçüde artırır. Alt popülasyondaki kır kurtları, yüksek derecede bilgi paylaşımına sahip değildir ve karmaşık optimizasyon problemlerini çözerken küresel arama yeteneği zayıftır. Arama verimliliğini ve yakınsama hızını artırmak, yerel optimumu korumak ve sonraki yinelemelerde popülasyon çeşitliliğini sağlamak için algoritma geliştirme stratejilerine ihtiyaç vardır. Önerilen algoritmanın arama davranışı karmaşık problemleri çözmek ve bilgiyi kullanma davranışı arama uzayı içindeki çözümün ne kadar iyi olduğunu kıyaslamak için geliştirilmiştir. Geliştirilen algoritmanın performansı, mevcut kır kurdu algoritmasındaki çözüm nesillerinde yapılan modifikasyonlar sayesinde geleneksel kır kurdu optimizasyon algoritmasından (COA) üstündür. İlk olarak, nüfus çeşitliliğini artırmak için kaotik sayı üretme stratejisi algoritma içerisinde tanıtılır. Ardından, en iyi kır kurtları arasındaki mesafe kontrol edilir. Ayrıca, mesafe etkisi, algoritma performansını artırmak için sürünün genel davranışına yansıtılır.

Böylece, popülasyon oluşumu için alt ve üst sınırlar arasında kaotik bir düzende bir değişken sağlanarak sürünün başlangıç konum çeşitliliğindeki zayıflık ortadan kaldırılır. Popülasyon oluşumu, popülasyon güncellemesi, yeni doğan bireylerin doğumu, levy uçuş dağılımı ilkesine göre rastgele parametreli hale getirilmiş, sürüler arası geçişler ve bölgesel en iyi kır kurdu arayışı kaotik bir düzende düzenlenmiştir. Bu sayede algoritmanın bölgesel en iyi noktalarda takılıp kalması, erken yakınsama sorununun önüne geçilmesi, sürüdeki çeşitliliğin artması, algoritma arama uzayının arttırılması ve bilginin verimli kullanılması sağlanmaktadır. Bu süreçte Laplace çaprazlama stratejisine göre algoritmaya ikinci bir doğum ve ölüm süreci eklenerek çözüm çeşitliliği ve algoritma performansı artırılmıştır. Kır kurtlarının benzerlik derecesi, kültürel eğilimin desteğiyle optimal çözüm alanını bulmak için COA algoritmasının arama sürecine dahil edilmektedir. Çözüm arayışı ilerledikçe, aday çözümler birbirine yaklaşır. Popülasyon çeşitliliği azalır ve tüm çözümler dar bir alana sıkıştırılır. Popülasyon çeşitliliği güncellenemez. Algoritma aramasının verimliliği düşer ve en iyi çözüm arayışı durur. Bu durumda çözüm bireyi etrafında yeni çözüm üretme mekanizmaları popülasyon çeşitliliğini artırabilir. Bu nedenle bireylerin, popülasyonun ve bireylerin sürüler arası geçişlerinin rastgele oluşturulmuş sayılar yerine kaotik haritalar ile güncellenmesinin sağlanması algoritmanın çözüm çeşitliliğini artırmıştır. Böylece algoritmanın keşif kabiliyeti artırılır ve bölgesel en iyi çözüm noktalarının takılması engellenir. COA algoritması, uygunluk-mesafe dengesi teoremi ile sürülerin sosyal eğilimlerini değerlendirerek ve Levy uçuş teoremi ile yeni bireylerin oluşturulmasını sağlayarak kaşif dengesini sağlamıştır. Ayrıca, Laplace çaprazlama teorisi ile yeni bireylerin doğum süreci kontrol edildiğinde keşif yeteneği ve yakınsama hızı performansı artmaktadır.

Algoritmanın başarısını artırmak için kaos tabanlı yöntemler, Levy uçuş teoremi tabanlı yöntemler, uygunluk-mesafe dengesi (FDB) tabanlı yöntemler ve Laplace çaprazlama tabanlı yöntemler uygulanmıştır. Bu şekilde, algoritmanın küresel keşif ve bilgi kullanımını dengeleme yeteneği etkin bir şekilde geliştirildi. Ek olarak, önerilen algoritmanın arama davranışı, karmaşık sorunları çözmek için geliştirilmiştir. Ayrıca, çözümün keşif alanında ne kadar iyi olduğunu karşılaştırmak için bilgi kullanımı geliştirilmiştir. Ayrıca yeni bireylerin arayış rolleri, yeni bireylerin seçimi, bireylerin sürüler arası göçü, yeni bireylerin doğumu, mevcut bireylerin popülasyona dahil edilmesi ve çözümleri belirli mesafelerde tutma süreçleri geliştirilmektedir. Önerilen algoritmanın

başarısını mevcut ve diğer algoritmalarla karşılaştırmak için algoritma test süreci iki ayrı test aşamasında gerçekleştirilmiştir. Birinci algoritmanın test aşamasında literatürde iyi bilinen yirmi üç kısıtsız problem test için kullanılmış ve mevcut algoritma ile karşılaştırılmıştır. İkinci algoritmanın test aşamasında, algoritmayı test etmek için literatürde iyi bilinen on adet kısıtlı gerçek mühendislik problemi kullanılmış ve literatürdeki diğer algoritmalarla karşılaştırılmıştır. Tüm testlerden elde edilen sonuç, önerilen algoritmanın ilk keşif işlemini gerçekleştirdiğini ve daha sonra bilgi yaklaşımını özellikle karmaşık problemlerde çok iyi kullandığını ve en iyi performansı, sağlamlığı ve en iyi sonucu verdiğini göstermektedir. Çalışma, literatürdeki diğer algoritmalara göre daha düşük standart sapma ve daha iyi en iyi değerlere sahip daha sağlam bir algoritma elde etmiştir. Dolayısıyla tüm testlerden elde edilen sonuç, önerilen algoritmanın ilk keşif işlemini gerçekleştirdiğini ve ardından bilgi yaklaşımını özellikle karmaşık problemlerde çok iyi kullandığını ve en iyi performansı, sağlamlığı ve en iyi sonucu verdiğini göstermektedir. Çalışma, literatürdeki diğer algoritmalara göre daha düşük standart sapma ve daha iyi en iyi değerlere sahip daha sağlam bir algoritma elde etmiştir. Önerilen algoritmanın performansını mevcut ve diğer algoritmalar ile kıyaslamak için algoritma test süreci üç ayrı test fazında gerçekleştirilmiştir. İlk algoritma test faz sürecinde literatürde iyi bilinen yirmi üç adet kısıtsız problem test için kullanılmıştır, mevcut algoritma ile karşılaştırılmıştır. İkinci algoritma test faz sürecinde on adet literatürde iyi bilinen kısıtlı gerçek mühendislik problemi algoritmanın testi için kullanılmıştır ve literatürdeki diğer algoritmalar ile karşılaştırılmıştır. Bununla birlikte taşıt debriyaj elemanlarının eniyilemesi son derece önemlidir, eğer uygun parametrelere sahip olmayan debriyaj elemanları ile devam edilirse sürüş konforu bozuk ve debriyaj sistemi ömrü az olur hatta eniyilenmemiş debriyaj elemanları motorun durmasına sebep olabilir. Bu yüzden en son algoritma test fazında önerilen algoritma taşıt güç aktarma sistemi içerisindeki motor hızının torkunun iletimini kesmek ve titreşimleri sönümlemek olan debriyaj sisteminin üzerindeki birkaç elemanın eniyilenmesi süreci ile test edilmiştir. Bütün testlerden elde edilen sonuç, önerilen algoritmanın özellikle karmaşık problemlerde önce keşfet sonra bilgiyi kullan yaklaşımını çok iyi bir şekilde gerçekleştirip en iyi performansı, sağlamlığı ve en iyi sonucu verdiği gözlemlenmiştir.

Çizelge 1.1. Mevcut ve hibrit algoritmaların araştırılması

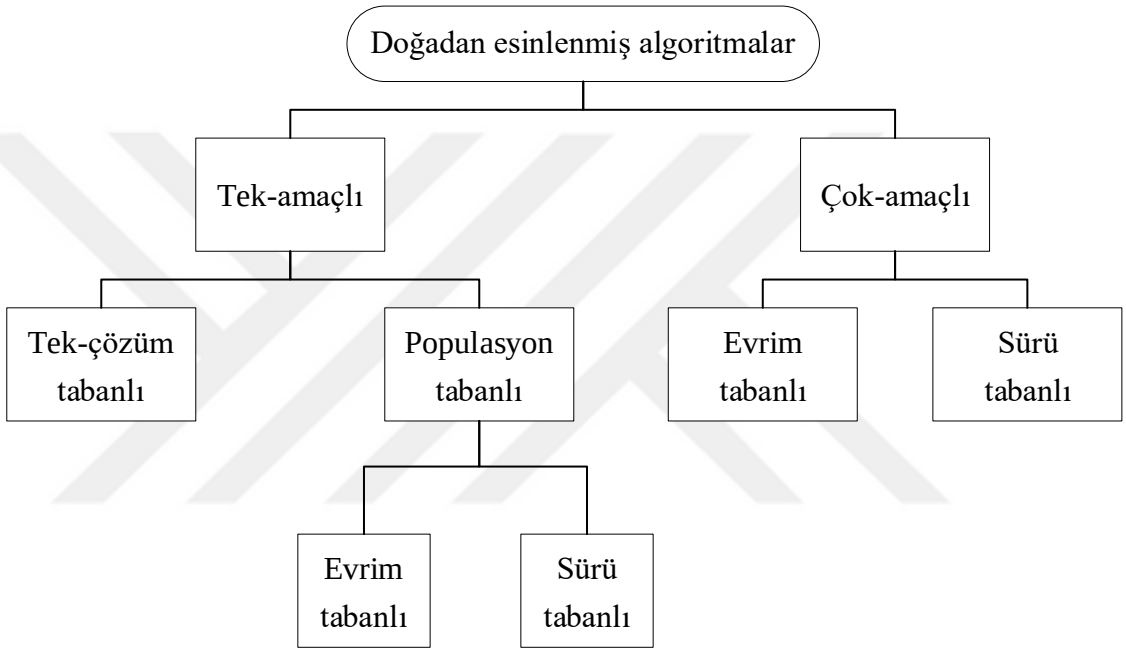
Algoritma	Araştırmacılar	Yıl	Test Sayısı	Problem Tipi
Gelişmiş Karşıt tabanlı Salp Sürü Algoritması	Hussien	2022	36	Standart test ve mühendislik tasarım problemleri
Bal Porsuğu Algoritması	Hashim ve ark.	2022	57	Standart test ve mühendislik tasarım problemleri
Yapay sinek kuşu algoritması	Zhao ve ark.	2022	91	Standart test ve mühendislik tasarım problemleri
İş birliği arama algoritması	Feng ve ark.	2021	86	Standart test ve mühendislik tasarım problemleri
Kaotik Yerel Arama Tabanlı Diferansiyel Evrim Algoritmaları	Gao ve ark.	2021	57	Standart test ve mühendislik tasarım problemleri
Etçil bir bitki algoritması	Ong ve ark.	2021	49	Standart test ve mühendislik tasarım problemleri
Açlık oyunu arama algoritması	Yang ve ark.	2021	23	Standart test ve mühendislik tasarım problemleri
Gelişmiş yiyecek konumlarına sahip yapay arı kolonisi	Sharma ve Abraham	2020	7	Mühendislik tasarım problemleri
Şempanze optimizasyon algoritması	Khishe ve Mosavi	2020	40	Standart test ve mühendislik tasarım problemleri
Yükseltilmiş ateş böceği algoritması	Feng ve ark.	2019	33	Standart test ve mühendislik tasarım problemleri
Uçan Sincap En iyileyici	Azizyan ve ark.	2019	18	Standart test ve mühendislik tasarım problemleri
Harris hawk optimizasyonu	Heidari ve ark.	2019	35	Standart test ve mühendislik tasarım problemleri
İyileştirilmiş meyve sineği optimizasyon algoritması	Brajević ve Ignjatović	2019	7	Standart test ve mühendislik tasarım problemleri
Kaotik salp sürüsü algoritması	Sayed ve ark.	2018	14	Standart test problemleri
Yeni yönlü yarasa algoritması	Chakri ve ark.	2017	20	Standart test problemleri
Astrofizikten ilham alan bir gri kurt algoritması	Kumar ve ark.	2017	20	Standart test ve mühendislik tasarım problemleri
Salp sürüsü algoritması	Mirjalili ve ark.	2017	26	Standart test ve mühendislik tasarım problemleri
Geliştirilmiş hızlandırılmış PSO algoritması	Guedria	2016	6	Mühendislik tasarım problemleri
Balina optimizasyon algoritması	Mirjalili ve Levis	2016	35	Standart test ve mühendislik tasarım problemleri
Sinüs Kosinüs Algoritması	Mirjalili	2016	20	Standart test ve mühendislik tasarım problemleri
Çoklu Ayet Optimize Edici	Mirjalili ve ark.	2016	24	Standart test ve mühendislik tasarım problemleri
Geçen araç arama	Savsani ve ark.	2016	13	Mühendislik tasarım problemleri
Gelişmiş Ateşböceği Algoritması	Brajevic	2015	4	Mühendislik tasarım problemleri
Güve-Alev Optimizasyon Algoritması	Mirjalili	2015	26	Standart test ve mühendislik tasarım problemleri
Gelişmiş parçacık sürüsü destekli genetik algoritma	Dhadwal ve ark.	2014	27	Standart test ve mühendislik tasarım problemleri
İç arama algoritması	Gandomi	2014	17	Standart test ve mühendislik tasarım problemleri
Gri kurt optimize edici	Mirjalili ve ark.	2014	33	Standart test ve mühendislik tasarım problemleri
Mayın patlatma algoritması	Sadollah ve ark.	2013	16	Standart test ve mühendislik tasarım problemleri
Su Döngüsü algoritması	Eskandar ve ark.	2012	7	Mühendislik tasarım problemleri
Guguklu arama algoritması	Gandomi ve ark.	2012	13	Mühendislik tasarım problemleri
Yarasa algoritması	Yang ve Gandomi	2012	8	Mühendislik tasarım problemleri
Lig şampiyonluk algoritması	Kashan	2011	59	Standart test problemleri
Öğretme-öğrenme tabanlı optimizasyon	Rao ve ark.	2011	15	Standart test ve mühendislik tasarım problemleri

Çizelge 1.1. Mevcut ve hibrit algoritmaların araştırılması (devam)

Algoritma	Araştırmacılar	Yıl	Test Sayısı	Problem Tipi
Değiştirilmiş T hücre algoritması	Aragon ve ark.	2010	27	Standart test ve mühendislik tasarım problemleri
Diferansiyel evrim ile hibritleşen parçacık sürüsü optimizasyonu	Liu ve ark.	2010	16	Standart test ve mühendislik tasarım problemleri
Büyük patlama – büyük kriz	Erol ve Eksin	2006	6	Standart test problemleri
Diferansiyel evrim stratejisi	Kim ve ark.	2005	5	Standart test ve mühendislik tasarım problemleri
Harmony arama teorisi ve pratiği	Lee ve Geem	2005	17	Standart test ve mühendislik tasarım problemleri

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

Geleneksel optimizasyon algoritmaları karmaşık problemlerin çözümünde yetersiz kalmaya başladığı için doğadaki fiziksel, kimyasal veya hayvan grupların davranışlarından esinlenmiş algoritmalar ortaya çıkmıştır. Söz konusu doğadan esinlenmiş algoritmalarda tek-amaçlı ve çok amaçlı algoritmalar olarak ikiye ayrılır. Aşağıdaki gibi sezgisel algoritmaların sınıflandırılması tasvir edilebilir.



Şekil 2.1. Doğadan esinlenmiş algoritmalar.

Bu tez çalışması tek amaç fonksiyonlu, sürü tabanlı doğadan esinlenmiş yeni bir algoritma üzerinedir.

2.1. Sürü Tabanlı Sezgisel Algoritmalar

Sürü tabanlı algoritmalar gruplar halinde yaşayan canlıların sürü olarak gerçekleştirdiği çevreye uyumdan ilham alır. Karıncalar yuvaları ile besin kaynağı arasında ilerlerken geçtiği yolda zamanla buharlaşan feromon adı verilen bir madde bırakır ve diğer karıncalar feromon miktarının yüksek olduğu yolu takip etme eğilimindedir, bu nedenle daha kısa yollarda zamanla en yüksek miktarda feromon birikir ve uzun süre kalır. Bu

süreç en kısa yolun koloni tarafından seçilme olasılığını arttırır. Bu doğal olayın sonucu olarak, Dorigo, Birattari ve Stutzle (2006) çalışmaları ile karınca sürüsü optimizasyon algoritması (ACO) karıncaların birbirleri arasındaki iletişim kabiliyetleri ve yuvaları ile besin kaynakları arasındaki en kısa yolu keşfetme yeteneklerinden ilham alınarak oluşturulmuştur.

Balık veya kuş sürüsündeki toplu davranış karakteristiğine göre her bireyin zamansal en iyi konumunun hızı değiştirilerek her bireyin yeni konumunun belirlenme esası ile sürünün küresel en iyi konumunu tespit etme prensibine göre Kennedy ve Eberhart (1995) parçacıklı sürü en iyileme (PSO) algoritmasını oluşturmuştur. Bu algoritma basit bir yapıya sahip olması ve tek bir arama mekanizması olması sebebiyle avantajlı olmasına rağmen küresel bir arama mekanizmasına sahip olmadığı için avantajlı değildir.

Karaboğa ve Baştürk (2007) işçi, gözcü ve kâşif arıların yiyecek arama davranışından ilham alarak sürü zekasına dayalı, basit, esnek ve kontrol parametresi az sayıda olan yapay arı kolonisi algoritmasını oluşturmuştur (ABC). Algoritmaya göre her bir çözüm besin kaynağı olarak nitelendirilir. İşçi arılar mevcut çözümleri ve uyumluluğunu değerlendirir. Gözcü arılar olasılık değerine göre yeni bölgeleri seçer. Kâşif arı üretilir ve yeni çözümler üretilir.

Yang ve Deb (2009) Guguk kuşlarının çoğalma davranışı olan başka kuşların yuvalarındaki yumurta ile kendi yumurtalarını değiştirmeyle yumurtalarının çıkma olasılığını arttırma davranışından ilham alarak guguk kuşu arama (CS) algoritmasını oluşturmuştur. Yuva arama Levy uçuşu ile algoritma içerinde modellenmiş olup her yumurta bir çözümdür.

Yang (2010) yarasaların sonar frekans ve genliği ile avların yerini veya engellerin yerini tespit etmesinden ilham alarak yarasa algoritmasını (BA) oluşturmuştur. Frekans ve genlik gıda kaynağına olan mesafenin fonksiyonu olarak değişir ve frekans yüksek olması durumunda yarasalar tarafından yerel çözüm arama yapılmasına zorlanılır. Diğer yandan eğer frekans düşükse arama uzayında genel çözüm araması yapılır.

Mirjalili (2016) yusufçuk kuşlarının avlanmaları sırasında oluşturduğu küçük sürülerin ve göç etme sırasında oluşturdukları büyük sürülerden ilham alarak ayırma, hizalama, uyum, çekicilik ve dikkat dağıtma faktörlerine dayanan yusufçuk algoritmasını (DA) oluşturmuştur. Mirjalili, Mirjalili ve Lewis (2014) küçük sürüler halinde ve dört farklı sosyal gruplar halinde (Alfa, beta, gama ve omega) yaşayan gri kurtların sosyal liderlik hiyerarşisinden ve avlanma stratejilerinden ilham alarak gri kurt eniyileme (GWO) algoritmasını oluşturmuştur. Gri kurtlardaki avlanma sırasındaki ava yaklaşma ve daha iyi av bulma davranışı en uygun (alfa) kurtlar ile yönlendirilmesi algoritma içerisine tanımlanmıştır. Mirjalili ve Lewis (2016) kambur balinaların yüzeye yakın sınırlı bir alanda daireler çizerek kabarcıklar yaratarak avı sararak en iyi avı çevreleyecek şekilde balinanın sürekli konumunu güncellemesi, avına saldırması ve yeni av arama davranışından ilham alarak kambur balina en iyileme (WOA) algoritmasını oluşturmuştur. Algoritmada balinaların her bir konumu çözümdür. Sezgisel algoritmalar ilham kaynakları bakımından doğadan esinlendikleri için benzerlik gösterirler ancak çözüm metotları bakımından farklıdırlar. Bazı birey bazlı sezgisel algoritmalarda bir aday çözüm ile başlanır bu algoritmaların maliyeti ve iterasyon sayısı azdır ancak bölgesel en iyi çözüme takılma olasılıkları oldukça yüksektir. Bu yüzden birden fazla çözüm ile başlanan popülasyon bazlı algoritmalar daha çok tercih edilmiştir.

2.2. Kır Kurdu Optimizasyon Algoritması (COA) ve İlgili Çalışmalar

Kır kurdu popülasyonları, küçük sürülere sahip olmaları, belirli bir sosyal yapıya sahip olmaları ve bölgesel olmaları nedeniyle diğer popülasyonlardan farklıdır. Her sürü, belirli bir bölge için aynı miktarda yiyeceğe sahiptir. Sürüdeki kır kurtları doğar, sosyal konum için savaşır, nüfus içinde dağılır, ürer ve ölür. Ayrıca, kır kurdu sürüsünde alfa, beta ve geçici bireyler vardır. Alfa kır kurtları liderdir ve alfa kır kurtları öldüğünde veya sürülerini değiştirdiğinde, beta kır kurtları alfa kır kurtlarının yerini alır ve lider olurlar. Bununla birlikte, mevcut bir pozisyon olması durumunda, kır kurtları geçici olarak sürü üyesi olurlar. Sürüde doğan yavrular zaman geçtikçe ölür veya yetişkin olurlar. Kır kurdu sayısı arttıkça, kır kurtlarının sürüden atılma olasılığı da artar; her kır kurdu cinsiyet, yaş, statü ve sürü ile karakterize edildi. Rastgele sayılar, sıfır ile bir arasındaki olasılık fonksiyonları tarafından belirlenir. Öte yandan, algoritma rastgele sayıyı seçtiğinde,

olasılık değerinden daha az olasılık fonksiyonu ile işlem gerçekleştirilir. Ayrıca, Yetişkin kır kurtlarının ölüm oranı, kır kurdu yaşının ikinci dereceden bir işlevi olarak kabul edilir. Ayrıca, kır kurdu yavrularının yüksek ölüm oranı, sürü içinde mevcut kaynakların sayısından kaynaklanabileceğinden, sabit bir ölüm oranı kullanılır. Ayrıca sürüye dahil edilen kır kurtlarının ölüm oranı da birey başına düşen yiyecek daha düşük olacağından yüksektir. Bu sürece dayanarak, Priezan ve Coelho (2018), kır kurtlarının sosyal yapısından ve birbirleri arasındaki deneyim alışverişinden esinlenerek, keşif ve bilgi kullanımını dengeleyen bir kır kurdu optimizasyon algoritması oluşturdular. Başka bir deyişle, Kır kurdu optimizasyon algoritması (COA), *Canis Latrans* türlerinin evrimsel olarak sürü zekasından ilham alan sürü tabanlı bir algoritmadır. Kır kurdu optimizasyon algoritması, belirtildiği gibi, sosyal hiyerarşi ve av yerine sosyal konfigürasyona, kır kurtları arasındaki deneyim alışverişine odaklanır. COA algoritması keşif ve sömürüyü dengelese de optimum çözümü bulmak için yetersizdir. COA algoritması ile ilgili son çalışmalar Tablo 2'de sunulmaktadır. Literatür çalışmalarından farklı türde problemleri çözmek için COA'nın çeşitli meta-sezgisel varyantlarının geliştirildiği fark edilmiştir. Mühendislik, tıp, güç akışı, güç dağıtımı, Pv modülleri ve görüntü işleme gibi gerçek problemler, çeşitli araştırmacılar tarafından COA algoritmalarının varyasyonları ile analiz edilmiştir. Keşif ve bilginin kullanımı arasındaki dengesizlik nedeniyle erken yakınsama ve yerel eniyileme, çoğu çalışmada ortak problemler olarak belirlendi. Dolayısıyla, herhangi bir algoritmanın başarısı, algoritmanın keşif ve bilgi kullanımı arasındaki dengesine bağlıdır. Bu nedenle, belirli bir amaç fonksiyonunu optimize etmek ve çözüm verimliliğini artırmak için COA algoritmasına farklı stratejiler etkin bir şekilde dahil edilmiştir. Li, Sun, Xue, Lie, Huang ve Mansour (2021) çok seviyeli eşik optimizasyon seçimi için yerel optimuma düşmekten kaçınmak için standart COA algoritmasına diferansiyel evrim stratejisini uyguladı. Başka bir deyişle, diferansiyel ölçeklendirme faktörü COA'ya uyarlanmıştır. COA'ya diferansiyel ölçekleme faktörünün eklenmesi, deneysel sonuçların görüntü segmentasyon kalitesi için en iyi optimum çözümü gösterdiği sonucuna varmıştır. Jin, Sun, Lei, Guo ve Zhu (2022) gri kurt optimizasyon algoritmasını ve yüzeye monte sabit mıknatıslı senkron motorları sürmek için oluşturulan COA'yı birleştirir. HWOA algoritması hızlı yakınsama sağlar ve yerel optimumlardan kaçınır. Duman, Kahraman, Guvenç ve Aras (2021) arama performansını, küresel keşif ve standart COA'nın yerel kullanımını dengelemek için Fitness-mesafe

dengesi (FDB) ve Levy uçuş stratejilerini uygulamıştır. Levy-Roulette-FDB-COA (LRFDB-COA) algoritması, doksan benchmark test fonksiyonu ve mühendislik problemleri ile test edilmiştir. Bununla birlikte, sonuçlar LRFDBCOA'nın pratik bir algoritma olduğunu göstermektedir. Tong, Zhu, Pierezan, Xu ve Coelho (2021) Kaotik Çakal Optimizasyon Algoritmasını (CCOA) önerdi. Ek olarak, düşük yakınsama hızı ve zayıf yerel optimum olan COA'nın zayıf yönleri nedeniyle kaotik haritaları içeren CCOA oluşturulmuştur. Çoğu araştırma, keşif ve sömürüyü dengeleyerek çözmek için erken yakınsama sorununa odaklanmıştır. Önerilen CCOA algoritması, iyi bilinen on kıyaslama işleviyle test edilmiş ve onaylanmıştır. Sonuç olarak, değiştirilmiş COA algoritma standart COA'dan daha verimlidir. Diab, Sultan, Do, Kamel ve Mossa (2020) PV modüllerinin ve güneş pillerinin değişkenlerini tanımlamak için COA algoritmasını kullandı. Sonuçlar, Coa algoritmasının çeşitli literatür algoritmalarıyla karşılaştırıldığında rekabetçi ve etkili olduğunu ortaya koydu. Abaza, El-Sehiemy, Mahmoud, Lehtonen ve Darwish (2021) tarafından proton değişim membranlı yakıt hücrelerinin doğru modellenmesini sağlamak için COA algoritması kullanıldı. Elde edilen sonuçlara göre yakıt pili parametreleri etkin ve güvenilir bulunmuştur. Birbirine bağlı güç sistemlerinin stabilizasyon kontrolü, Guesmi, Alshammari, Almalaq, Alateeg ve Alqunun (2021) tarafından COA algoritması ile parametre optimizasyonu ile yapılmıştır. Sonuçlar, kontrolörün güç sistemi salınımlarının sönümleme özelliklerinin performansını arttırdığını göstermektedir. Nguyen, Pham, Kien ve Van Dai (2020) güneş fotovoltaik dağıtım birimlerinin konumunu ve boyutunu optimize etmek için gelişmiş bir COA algoritması kullandı. Geliştirilen algoritmanın sonuçları, genetik algoritma (GA), ayçiçeği optimizasyonu (SFO), parçacık sürüsü optimizasyon algoritması (PSO), biyocoğrafya tabanlı Optimizasyon (BBO), salp sürüsü algoritması (SSA) ve standart COA algoritması gibi algoritmalarla karşılaştırılmıştır. Geliştirilmiş COA algoritması etkin bir şekilde daha iyi kararlılık ve kaliteli çözümler göstermiştir. Geliştirilmiş bir COA algoritması, Yuan, Wang, Wang ve Yıldızbaşı (2020) tarafından yakıt hücresi modelinin en iyi parametrelerini tanımlamak için kullanılmıştır. Algoritma sonuçları ampirik sonuçlarla karşılaştırıldığında geliştirilen COA algoritmasının geleneksel COA, SOA, ES algoritmalarına göre iyi uyum sağladığı ve üstün özellikler gösterdiği görülmektedir.

Wu, Li ve Wan (2021) tarafından bir hibrit algoritma oluşturuldu. Arama verimliliğini ve yakınsamayı iyileştirmek için geliştirilen kır kurdu algoritmasını kullandı. Böylece, yüksek yineleme süresine ve öğrenme verimliliğine sahip bir çekirdek artımlı aşırı öğrenme makinesi olarak yerel optimum düşüşü önlemek için geliştirilmiş böcek sürüsü kır kurdu algoritması hibrit algoritması elde edildi. Kien, Hien ve Nguyen (2021) güç iletim ağlarının kontrol parametrelerini belirlemek için kır kurdu algoritması içindeki çözüm nesillerinin yenilenmesi açısından kır kurdu algoritmasını geliştirmiştir. Önerilen yöntem ayçiçeği optimizasyon algoritması (SFOA), salp sürüsü algoritması (SSA) ve su döngüsü algoritması (WCA) olmak üzere üç algoritma ile karşılaştırılmış ve geliştirilmiş kır kurdu algoritmasının daha etkili olduğu bulunmuştur. Duman, Kahraman, Güvenc ve Aras (2021) yenilenebilir enerjiden elde edilen enerjinin en uygun güç akışını sağlamak için kültürel eğilimi uygunluk-mesafe-denge (FDB) ve yeni bireylerin doğuşunu Lévy-flight (LF) teoremi ile destekleyerek COA algoritmasını geliştirmiştir. Geliştirilen algoritma hesaplama karmaşıklığına sahip olmasına rağmen diğer algoritmalara göre rekabetçi davranış göstermektedir. Pham, Nguyen ve Dinh (2021) radyal dağıtım sistemlerinde fotovoltaik dağıtılmış jeneratörlerin optimal konumunu ve en iyi kapasitesini bulmak için Geliştirilmiş Kır Kurdu Optimizasyon Algoritmasını (GCOA) kullandı. SSA, COA, GA/PSO, SA, BFOA, HSA, IWO, QOTLBO ve AGA algoritmaları ile karşılaştırmalı sonuçlar, GCOA'nın optimizasyon çalışmasında güçlü bir yöntem olduğunu göstermektedir. Kamel, Amin, Selim ve Ahmed (2019) elektrik şebekesinde elektriğin en uygun dağıtımını işletme kısıtları altında asgari güç kaybını sağlayıp voltaj profilini kır kurdu algoritması ile iyileştirmişlerdir. Shi, Zhou ve Zhang (2020) fotovoltaik güç istasyonlarının güvenli ve istikrarlı çalışması kısıtına bağlı olarak işletme ve bakım maliyetlerini fotovoltaik güç istasyonu sayısı ve dağılımını eniyileyerek kır kurdu algoritması ile iyileştirmiştir. Mohamed, Morrow, Best, Bailie, Cupples ve Pollock (2020) batarya enerji depolama sistemlerinin en uygun konumunu, boyutunu, boşaltma ve şarj programlarını eniyileyerek enerji hattındaki stresleri azaltarak performansı arttırmayı amaçlamışlardır. Bu süreçte parçacık sürüsü algoritması, ateş böceği algoritması, yeni yarasa algoritması, krill sürüsü algoritması ve kır kurdu algoritması arasında karşılaştırma yaparak kır kurdu algoritmasının yeterli olmadığını belirtmiştir. Chang ve Nguyen (2020) güneş enerji sistemlerinin dağıtımında sistem kaybını azaltma, voltaj profili iyileştirme ve farklı yük seviyeleri altında regülatör kademe kontrollerini

dikkate alarak enerji dağıtım sistemini kır kurdu optimizasyon algoritması ile iyileştirmiştir. Kaymaz, Duman ve Güvenç (2021) rüzgar enerjisinin ekonomik dağıtım probleminin çözümünde kır kurdu algoritmasını (COA) kullanmıştır. Li, Sun, Xue, Li, Huang ve Mansour (2021) kır kurdu algoritmasını görüntü işleme ve tanı koyması üzerine kullanmıştır. Babu, Narrisetty ve Saikia (2019) güneş enerjisi sisteminin yük frekans kontrolünü kır kurdu algoritması ile yapmıştır. Boursianis, Papadopoulou, Pierezan, Mariani, Coelho, Sarigiannidis, Koulouridis ve Goudos (2021) anten geometrisinin optimizasyonunda kır kurdu algoritmasını kullanmıştır. Abdelghafar, Goda, Darwish ve Hassanien (2019) pillerin kalan kullanım ömrünü tahmin etmek için destek vektör regresyonu ile kır kurdu optimizasyonunu kullanmıştır. Jin, Sun, Lei, Guo ve Zhu (2022) sabit mıknatıslı senkron motorlardaki sürücülerinin tork dalgalanmasını azaltmak ve akı izleme doğruluğunu iyileştirmek için kır kurdu algoritması ile gri kurt algoritmasını birleştirerek tork kontrolünü sağlamıştır. Yang, Hu, Yang, Wang, Shi, Yu ve Huang (2020) tren işletiminde toplam tren gecikme süresini en aza indirmek için kır kurdu optimizasyon algoritmasını (COA) kullanmıştır. Standart Coa'nın genelde iyi sonuçları olmasına rağmen, Sayed, Khoriba ve Haggag (2020) deri hastalığının teşhisi ve tedavi planlaması için standart Coa'ya dayalı çok sürülü kır kurdu optimizasyon algoritmasını geliştirmiştir. Standart Coa bölgesel alanlarda sıkışıp kaldığı için çok sürülü kır kurdu optimizasyon algoritması oluşturuldu. Deneysel sonuçlar, geliştirilmiş COA algoritmasının keşif kabiliyetini önemli ölçüde geliştirerek diğer optimizasyon algoritmalarına (SSA, COA, ASO, ALO, QABC ve PSO) kıyasla kararlı ve etkili özellikler göstermiştir. Son zamanlarda, tasarım, güç dağıtım ağları ve görüntü işleme gibi çeşitli problemlerin optimizasyonunda araştırmacılar tarafından farklı COA varyasyonları kullanılmıştır. Optimizasyon problemlerinde değiştirilmiş bir COA algoritması kullanılarak farklı sorunlar optimize edilmektedir. Bu değiştirilmiş COA algoritmalarındaki tipik kusurlar, zayıf arama performansı, erken yakınsama ve çeşitlilik nedeniyle yerel en iyi noktalarda takılıp kalmadır. Araştırmacılar, bu kusurları iyileştirmek için kaotik stratejiler gibi farklı mekanizmalar kullandılar. Bu nedenle, bu çalışma, yeni, geliştirilmiş bir COA algoritması ile bu kusurları ortadan kaldırmayı amaçlamaktadır. Kır kurdu algoritmasının eksik noktalarını iyileştirmek için daha hızlı yakınsama, çeşitlilik, yerel ve küresel arama yeteneklerine sahip gelişmiş bir COA (GCOA) algoritması oluşturulmuştur.

Çizelge 2.1. Hibrit ve standart Coa varyantları ile ilgili literatür araştırması

Araştırmacı	Yıl	Algoritma	Önerilen çalışmaya göre ana bulgular
El Ela ve ark.	2021	COA	Dağıtılmış üretim (DG) entegrasyon performansını artırmak için bulanık tabanlı çok amaçlı (FBMO) formülasyonu kullanılır. Ayrıca, kurulu arıza akım sınırlayıcıları (FCL'ler) boyutlarını ekonomik hale getirerek arıza akımlarını azaltarak güç kayıplarını en aza indirmeyi amaçlar. Ek olarak, DG'ler ve FCL'ler arasındaki en iyi kombinasyonu bulmak için COA kullanılır. Sonuç olarak COA, PSO algoritmasına göre en iyi sonuçları ve ekonomik çözümleri vermektedir. COA, standart sapma ve ortalama süre kapsamında sağlamdır.
Li ve ark.	2021	ICOA	Çok seviyeli eşik optimizasyon seçimi için yerel optimuma düşmeyi önlemek için standart COA algoritmasına diferansiyel evrim stratejisi uygulanır. Ayrıca, Geliştirilmiş COA (ICOA), Bulanık Modifiye Hızlı Yapay Arı Kolonisi (FMQABCA), Gri Kurt Optimize Edici (GWO), Toplama Algoritması ve Bulanık Değiştirilmiş Ayrık Gri Kurt İyileştirici (FMDGWOA) algoritmalarıyla karşılaştırıldığında iyi sonuçlar gösterir.
Jin ve ark.	2021	HWOA	Gri kurt optimizasyon algoritması ile COA'yı birleştiren Hibrit Kurt Optimizasyon Algoritması (HWOA), yüzeye monte sabit mıknatıslı senkron motorları (SPMSM'ler) kontrol etmek için oluşturulmuştur. Ek olarak, HWOA hızlı yakınsama sağlar ve yerel optimumlardan kaçınır.
Huang ve Zhuang	2021	ICOA	Rüzgar enerjisi sistemlerinin istenen güç referansı izleme yeteneğini geliştirmek için Levy Flight ve Sinusoidal Map ile entegre COA ile hataya dayalı aktif bozucu reddetme kontrolüne (ADRC) dayalı bir adım kontrol stratejisi gerçekleştirilir.
Sultan ve ark.	2021	COA	Proton değişim membranlı yakıt hücresinin (PEMFC) bilinmeyen parametreleri, çakal optimizasyon algoritması (COA) kullanılarak tanımlanır. Sonuçlar, PEMFC parametrelerinin tanımlanmasının ve çıkarılmasının mühendislik problemleri için kolay ve makul olduğunu göstermektedir.
Duman ve ark.	2021	LRFDBCOA	Uygunluk-mesafe dengesi (FDB) ve Levy uçuş stratejileri, standart COA'nın dengeleyici arama performansını, standart COA'nın küresel keşfini ve standart COA'nın yerel kullanımını geliştirir. Geliştirilmiş COA (LRFDBCOA) algoritması, 90 kıyaslama testi işlevi ve mühendislik problemleri ile test edildi. Kıyaslama fonksiyonlarının sonuçları, LRFDBCOA'nın etkili bir algoritma olduğunu göstermektedir.
Tong ve ark.	2021	CCOA	Kaotik haritalar da dahil olmak üzere Kaotik Çakal Optimizasyon Algoritması (CCOA), düşük yakınsama hızı ve zayıf yerel optimum olan COA'nın zayıf yönlerini iyileştirmek için oluşturulur. Önerilen algoritma, iyi bilinen on kıyaslama işleviyle test edilmiş ve onaylanmıştır. Son olarak, CCOA algoritması verimlidir.
Vineeth ve ark.	2021	COA	COA algoritması, hastaların yaşamlarına daha iyi karar vermek için biyomedikal görüntülerin gürültüden arındırılması için iyi sonuçlar sağlar. Bununla birlikte, Coa, Yapay Arı Kolonisi gibi diğer algoritmalar kadar hızlı yakınsamaz.
Abou ve ark.	2021	COA	COA, yük frekansı kontrolünü (LFC) çözmek ve en etkili yenilenebilir enerji kaynaklarına sahip olmak için kontrolörlerin parametrelerini optimize etmede kullanılır. COA algoritması aracılığıyla filtre-PI (PDn-PI) denetleyicili kademeli orantılı-türev, diğer denetleyicilere kıyasla üstünlüğe sahiptir.
Nguyen ve ark.	2021	ICOA	Geliştirilmiş COA (ICOA), değiştirilmiş güç akışıyla toplam güç kayıplarını en aza indirir. ICOA, grubun eğilim çözümü yerine, popülasyonun şimdiye kadarki en iyi çözümü ile ilerler. En iyi çözüm yerel optimum çözüm ise, en iyi çözüm mevcut çözüm grubu etrafında güncellenir. ICOA, COA, PSO ve SFO'dan daha fazla çözüm kalitesine ve daha hızlı yakınsama oranına sahiptir.
Wu ve ark.	2021	HCOBSO	Hibrit Çakal Böceği Sürü Optimizasyonu (HCOBSO) yöntemi, çekirdek artımlı aşırı öğrenme makinelerinin yineleme sürelerini ve öğrenme verimliliğini optimize etmek için önerilmiştir. COA algoritmasını geliştirmek için Gauss küresel en iyi büyüyen operatör kullanılır. Ayrıca, COA'nın küresel arama yeteneğini arttırır. Dinamik mutasyon ve çadır eşleme ters öğrenme stratejileri, yerel bir optimuma düşmekten kaçınmak için böcek sürüsü optimizasyon algoritmasını (BSOA) iyileştirir.
Moschos ve Parisses	2021	COA	COA, bir senkron jeneratörün terminal voltajını belirli bir aralıkta tutmak için bir güç sisteminde Otomatik Voltaj Regülatörü (AVR) kontrolörünün optimize edilmesinde kullanılır. Ayrıca, önerilen COA'lı kontrolör kararlı özellikler sağlar.
Rezk ve ark.	2021	COA	Gölge yüzleri altında fotovoltaik diziden çıkarılan maksimum küresel gücü maksimize etmek için COA algoritması kullanılır. COA ile elde edilen sonuçlar Toplam Çapraz Bağlanmış (TCT), Çiçek Tozlaşma Algoritması (FPA), Su Do Ku, Kelebek Optimizasyon Algoritması (BOA) ve Deniz Yırtıcıları Algoritması (MPA) ile karşılaştırılır. Sonuç olarak, ICOA, gölgeli diziyi en iyi şekilde yeniden yapılandırmada algoritmanın üstünlüğünü doğruladı.

Çizelge 2.1. Hibrit ve standart Coa varyantları ile ilgili literatür araştırması (devam)

Araştırmacı	Yıl	Algoritma	Önerilen çalışmaya göre ana bulgular
Janamala ve Reddy	2021	COA	Kır Kurdu Optimizasyon Algoritması (COA), artan kayıpları ve voltaj sapmasını azaltmak için Fotovoltaik (PV) sisteminin boyutlandırılmasını ve optimum Hatlar Arası – Fotovoltaik (I-PV) parametrelerinin tanımlanmasını optimize eder. Coa'nın optimizasyon sonucu, GWO ve PSO'ya kıyasla daha iyi performans gösteriyor.
Sun ve ark.	2021	FO-COA	İş, tersinmezlik ve ekserji gibi maliyet fonksiyonlarının iyileştirilmesine bağlı olarak güvenilir ve doğru sonuçlar elde etmek için Kesirli Sıralı Çakal Optimizasyon Algoritması (FO-COA) kullanılmaktadır. Ayrıca, Sonuçlar ampirik verilerle doğrulanmıştır. Karşılaştırma sonuçları, orijinal COA ve Genetik Algoritmanın (GA) iyi sonuçlara sahip olduğunu göstermektedir.
Pierezan ve ark.	2021	CCOA	Kaotik Çakal Optimizasyon Algoritması (COA), dağılım ve ilişki olasılıklarını tanımlama açısından yapıları optimize etmek için kullanılır. Sonuçlar, orijinal COA'ya kıyasla çok daha sağlamdı.
Abaza ve ark.	2021	ECOA	Yenilenebilir enerji de dahil olmak üzere reaktif güç dağıtım sorunlarını optimize etmek için gelişmiş çakal optimizasyon algoritması (ECOA) kullanılır. Simülasyon sonuçları, ECOA'nın güç kayıplarında önemli bir azalma sağladığını göstermektedir.
Diab ve ark.	2020	COA	COA algoritması ile güneş pilleri ve PV modüllerinin bilinmeyen parametreleri tanımlanır. Sonuçlar, maksimum güç elde etmek için COA algoritmasının üstünlüğünü ve güvenilirliğini doğruladı.
Chang ve Cong	2020	COA	Gerilim profili iyileştirmesi, regülatör kademe kontrolleri ve kayıp azaltmayı içeren dağıtım sistemi, bir COA algoritması ile optimize edilmiştir. Sonuç, Gray Wolf Optimizer (GWO), Parçacık Sürü Optimizasyonu (PSO), Genetik Algoritma (GA), Karışık Tamsayılı Doğrusal Olmayan Programlama (MINLP) ve Biyocoğrafya Tabanlı Optimizasyon (BBO) ile karşılaştırıldığında çok verimlidir.7
Abdelwanis ve ark.	2019	COA	COA algoritması, tek ve üç fazlı güç transformatörlerinin parametrelerini tanımlamak için kullanılır. COA'dan elde edilen tahmini parametreler, parçacık sürüsü ve Jaya optimizasyon algoritmalarından elde edilen parametrelerden daha iyidir.
Babu ve ark.	2020	COA	Çanak Stirling güneş enerjisi sisteminin yük frekans kontrolü ve termal sistem, en iyi kazanç parametrelerini elde etmek için bir COA algoritması ile optimize edilmiştir. Optimizasyon süreci, orantılı-integral-türev denetleyicisini geliştirir.
Ribeiro ve ark.	2018	Hybrid COA	Tamamlayıcı Topluluk Ampirik Mod Ayrıştırma(CEEMD), Aşırı Öğrenme Makinesi (ELM), Gradyan Artırma Makinesi (GBM), Gauss Süreci (GP) ve Uygunluk Vektör Makineleri (RVM) ile oluşturulan hibrit COA algoritması, ticari ve konut elektriğini tahmin etmek için kullanıldı. enerjinin fiyatı. Sonuçlar, en iyi sonuçların COA-CEEMD hibrit algoritması ile sağlandığını göstermektedir.
Mohamed ve ark.	2020	COFL	Bulanık mantık (FL) sistemi ile birleştirilen kır kurdu optimizasyon algoritması (COA), kümeleme sürecini güçlendirmek ve dengelemek, kablosuz ağ ömrünü artırmak ve enerji tüketimini azaltmak için entegre edilmiştir. Önerilen COFL algoritması, geleneksel protokoller olan Parçacık Sürü Optimizasyonu (PSO), Kır Kurdu Optimizasyonu (COA), Kararlı Seçim Protokolü (SEP), Gri Kurt Optimizasyonu (GWO) ve Düşük Enerji Uyarlamalı Kümeleme Hiyerarşi Protokolü (LEACH) ile karşılaştırılır. COFL, diğer algoritmalara göre rekabetçi bir algoritmadır.
Yang ve ark.	2020	COA	COA, toplam tren gecikme süresini azaltmak için bir tren operasyonunun ayarını optimize eder. COA algoritması, ağırlık uyarlamalı parçacık sürü optimizasyonundan (IPSO) daha iyidir.
Boursianis ve ark.	2020	COA	Anten tasarımı COA optimizasyon algoritması ile gerçekleştirilmiştir. Tasarlanan anten, performans belirteçleri için kabul edilebilir sonuçlar vermektedir.
Shi ve ark.	2020	ICOA	Tek boyutlu büyüme stratejisi de dahil olmak üzere geliştirilmiş COA (ICOA) algoritması, standart COA'nın erken yakınsamaya sahip olması nedeniyle düşük maliyetli Fotovoltaik akıllı uç terminalleri (PVIET) sağlamayı amaçlar. ICOA, standart COA'dan daha yüksek doğruluk ve kararlılıkla PVIET miktarını ve konumunu optimize eder.
Kaymaz ve ark.	2020	LCOA	Rüzgar enerjisi kaynaklarının optimum güç akışı Levy Kır kurdu Optimizasyon Algoritması (LCOA) ile sağlandı. Yerel optimum noktalarda sıkışmayı önlemek için Levy uçuş stratejisi COA ile birleştirildi. Emisyonlar, yakıt maliyeti, aktif güç kaybı, voltaj kararlılığı ve voltaj profili dahil olmak üzere optimum güç akışını bulmak için on sekiz vaka LCOA ile değerlendirildi.
Souza ve ark.	2020	BCOA	COA (BCOA) ikili versiyonu, sınıflandırma için en iyi tipik alt kümeyi seçmek için kullanıldı. BCOA, seçilen özellikleri, hesaplama maliyetini ve sınıflandırma doğruluğunu optimize etmeyi amaçlar. Sonuçlar, hedef veri kümeleri için kabul edilebilir kesinlik gösterir.

Çizelge 2.1. Hibrit ve standart Coa varyantları ile ilgili literatür araştırması (devam)

Araştırmacı	Yıl	Algoritma	Önerilen çalışmaya göre ana bulgular
Yuan ve ark.	2020	DCOA	PEM yakıt hücrelerinin (PEMFC'ler) en iyi parametre tahminini tanımlamak için Kır kurdu optimizasyon algoritmasının (DCOA) geliştirilmiş bir versiyonu oluşturulur ve simülasyon ve ampirik veriler arasında minimum toplam hata sağlar. Sonuçlar, orijinal COA, Martı Optimizasyon Algoritması ve *(N + λ)* - ES algoritması ile karşılaştırıldığında bir dizi deneysel çalışmanın ardından rekabetçidir.
Fathy ve ark.	2019	COA	COA algoritması, hibrit bir elektrik güç sisteminde hidrojen tüketimini azalttı. COA algoritması Çoklu Ayet Optimizasyonu (MVO), Genetik Algoritma (GA), Çekirge Optimizasyon Algoritması (GOA), Ayçiçeği Optimizasyonu (SFO), Parçacık Sürü Optimize Edici (PSO), Harici Enerji Maksimizasyon Stratejisi (EEMS), Salp Sürü Algoritması (SSA) ve Gri kurt optimizasyonu (GWO).
Güvenç ve Kaymaz	2019	COA	COA algoritması ile ekonomik güç dağıtımı sağlandı. COA algoritması, parçacık sürüsü optimize ediciden (PSO) ve genetik algoritmadan (GA) daha iyi sonuç verir.
Mostafa ve Ibrahim	2019	COA	PV sistemi, küresel maksimum PowerPoint'i izlemek için kısmi gölgelemenin etkisine bağlı olarak COA ile optimize edilmiştir. Coa, izleme ve kararlı durum açısından Parçacık Sürü Optimizasyonu (PSO), Gelişmiş Gri Kurt Optimizasyonu (E-GWO), Karınca Aslanı Optimizasyonu (ALO) ve Yusufçuk Algoritmasından (DA) daha yüksek verimlilik sağlar.
Abdelghafar ve ark.	2019	COA-SVR	Pillerin kalan kullanım ömrünün tahmini, COA-SVR algoritması olarak adlandırılan kır kurdu optimizasyon algoritması (COA) birleşik destek vektör regresyon (SVR) ile gerçekleştirilmiştir. Deneysel sonuçlar, temel SVR algoritmasından ve alaka vektör makinesinden (RVM) daha iyi kararlılık ve verimlilik gösterdi.
Pierezan ve ark.	2019	MOCOA	Çok amaçlı COA(MOCOA) algoritması, Test Elektromanyetik Analiz Yöntemini 25 kıyaslama testi problemiyle çözmek için önerildi. Benchmark test fonksiyonlarının sonuçları MOCOA'nın etkili bir algoritma olduğunu göstermektedir.
Babu ve ark.	2019	COA	Otomatik enerji üretim kontrolü, termal sistem, güneş enerjisi santrali ve jeotermal arasında bir COA algoritması ile optimize edilmiştir. COA algoritması, ateş böceği ve guguk kuşu arama algoritmalarına kıyasla daha iyi sonuçlar verdi.
Pham ve ark.	2019	ECOA	Radyal dağıtım ağlarındaki dağıtılmış jeneratörlerin (DG'ler) kapasitesi ve konumu, voltaj kararlılığını artırmak, maliyetleri azaltmak ve güç kaybını azaltmak için gelişmiş bir çakal optimizasyon algoritması (ECOA) ile optimize edildi. ECOA, ilk çözüm üretimini geliştirmeye ve hesaplama süresini azaltmaya odaklanır. ECOA'nın etkinliğini doğrulamak için ECOA algoritması, COA, salp sürüsü algoritması (SSA), Ayçiçeği optimizasyonu (SOA) algoritmaları gibi diğer algoritmalarla karşılaştırılmıştır. ECOA, lokasyonları etkin ve hızlı bir şekilde belirler.
Qais ve ark.	2019	COA	COA, üç diyotlu fotovoltaik (PV) modelin dokuz bilinmeyen parametresini tanımlayarak deneysel ve hesaplanan sonuçlar arasındaki farkı en aza indirmek için kullanılır. Sonuç olarak, COA, PV modülünün belirli modeline sağlam bir çözüm sunar.
Chin ve Salam	2019	COA	COA'ya dayalı optimize ediciler, PV sistemlerinin arızasını ve güç ve akım hatalarından kaynaklanan güneş pili bozulmasını tespit etmek için PV hücre parametrelerini çıkarır. Coa tabanlı optimize edicilerin sonuçları dikkate değer sonuçlar gösterdi.
Pierezan ve ark.	2019	CCOA	Kültürel Çakal Optimizasyon Algoritması (CCOA), ağır hizmet tipi bir gaz türbininin çalıştırılması için yakıt tüketiminin en aza indirilmesi sorununa uygulanır. Sonuçlar, CCOA'nın diğer metasezgisel algoritmalara kıyasla bir dizi deney ve kıyaslama işlevinden sonra mükemmel bir algoritma olduğunu göstermektedir.

2.3. Taşıt Debriyaj Elemanlarının Sezgisel Algoritmalar Kullanılarak Optimizasyonu ile İlgili Çalışmalar

Taşıt debriyaj elemanlarının sezgisel algoritmalar kullanılarak optimizasyonu ile çalışmalar literatürde kısıtlı sayıdadır. Xiliang, Changqing, Fuwu, Hongming, Biao ve Yangfeng (2019) hibrit araçlarda kullanılan çift debriyaj iletim sistemini dişli oranı ve vites değiştirme eşik değeri parametrelerini değiştirerek asgari yakıt tüketimini amaçlamıştır. Bu optimizasyon çalışmasında geliştirilmiş parçacık sürüsü optimizasyon

algoritması kullanılmıştır. Zhang ve Zhu (2008) taşıt debriyaj elemanlarından diyafram yayının baskı yükü karakteristiğine ait matematiksel denklemi üzerinden yola çıkarak baskı yükünü parçacık sürüsü optimizasyon algoritmasına göre en iyi sonuçları bulmuştur. Genç ve Kaya (2020) taşıt debriyaj sistemlerinden olan damper sistemi içerisindeki metalik yaylar yerine kauçuk yay kullanarak titreşimleri sönümledi. Kauçuk yayın optimizasyonunu tavlama benzetim algoritması ile sağlamışlardır. Karaduman ve ark. debriyaj diyafram yayının asgari ağırlık ve azami ömür amaç fonksiyon ve istenen kavrama yükü ve asgari gerilim kısıtı için en uygun on değişkene sahip diyafram geometrisine diferansiyel gelişim algoritması ile sağlamıştır. Kartal, Kaya, Çakmak, Karaduman ve Karpat (2015) debriyaj elemanlarından olan metalik disk elemanının kavrama sırasında konfor etkisi sağlayan yük karakteristiğini diferansiyel gelişim algoritması ile eniyilemiştir. Bununla birlikte, No Free Lunch teoremi her optimizasyon probleminde tek bir algoritmanın iyi performans göstermeyeceğini belirtir bu yüzden bu araştırma alanında taşıt debriyaj elemanları için yeni bir algoritma önerilmiştir.

2.4. Tez Çalışmasının Literatüre Katkısı ve Diğer Çalışmalardan Farkı

Taşıt debriyaj elemanları yüksek ömür, yüksek dayanım, yüksek konfor, yüksek performansa ve mümkün mertebe düşük maliyete sahip olmalıdırlar. Bu nedenle en uygun şekilde taşıt debriyaj elemanlarını tasarlamak için optimizasyon algoritmalarına ihtiyaç duyulmuştur. Bu tez çalışmasında sosyal ve biyolojik tabanlı kır kurdu sürülerinden esinlenen kır kurdu algoritmasının performansını geliştirecek değişiklikler yapıp yeni ve rekabetçi on beş farklı algoritma oluşturulup literatüre kazandırılmıştır, en iyisi önerilen optimizasyon algoritması olarak ortaya konulmuştur. Mevcut kır kurdu algoritması yapısal problemleri olan özellikle değişken sayısı arttıkça başlangıçta lokal arama ihtimalinin yüksek oluşundan kaynaklı lokal minimuma takılma sorununa sahip bir algoritmadır. Yeni COA, kır kurtlarının belirli aralıklarla seçilmesi sonucunda sürünün genel davranışına yeni popülasyon oluşumunu ekleyerek, çakalların adaptasyonunun sürünün genel davranışı üzerindeki etkisini belirlemiştir. Arttırılmış keşif ve arttırılmış sömürü arasında denge sağlanmıştır. Yeni COA, popülasyon oluşumunu, popülasyon güncellemesini, populasyon içindeki kır kurtlarının seçimini uygunluk uzaklık dengesi stratejisine göre tasarım alanının keşfini gerçekleştirip, levy uçuş dağılımı ilkesine göre

parametrelenmiş yeni doğan bireylerin doğumunu, sürüler arasındaki geçişleri ve ikinci bir doğum prosesi ile kaotik bir düzende bölgesel en iyi çakal arayışını düzenlemiştir. Bu sayede algoritmanın bölgesel en iyi noktalarda takılıp kalmaması, erken yakınsama sorununun önüne geçilmesi, sürüdeki çeşitliliğin artırılması, algoritma arama uzayının iyileştirilmesi ve bilginin verimli kullanılması sağlanmaktadır. Sürü sayısı, her sürüdeki kır kurdu sayısı, maksimum yineleme sayısı ve deney sayısı algoritmanın kontrol parametreleridir. Bu süreçte algoritmaya Laplace çaprazlama stratejisine göre ikinci bir doğum ölüm süreci ve yerel arama yeteneği kaotik stratejisi eklenerek çözüm çeşitliliği ve algoritma performansı artırılmıştır. Bu şekilde tasarım uzayının gelecek vaat eden alanlarında bulunan çözümlere yoğunlaşılmasından sorumludur. Önerilen algoritma mevcut kır kurdu algoritmasına göre yerel arama mekanizması, küresel arama mekanizması, küresel yakınsama hızı ve performans bakımından daha üstündür. Önerilen algoritmanın üstünlüğü literatürde iyi bilinen yirmi üç adet kısıtlı ve on adet kısıtlı mühendislik problemiyle test edilmiştir. COA'nın parametreleri, COA'nın orijinal özelliklerini korumak için değiştirilmemiştir. Önerilen algoritma, etkinliği ölçmek için istatistiksel testlerle doğrulandı. Sonuç olarak, algoritmanın diğer algoritmalarla karşılaştırmalı analizi, yeni algoritmanın çözüm doğruluğunda önemli ölçüde üstün olduğunu ortaya koydu. Geliştirilmiş algoritma taşıt debriyaj elemanlarına literatürde ilk kez uygulanmıştır. Bununla birlikte iyi dengelenmiş yerel ve küresel arama mekanizmasının yanında önce keşfet sonra bilgiden yararlan prensibine göre tasarlanmıştır.

3. MATERYAL ve YÖNTEM

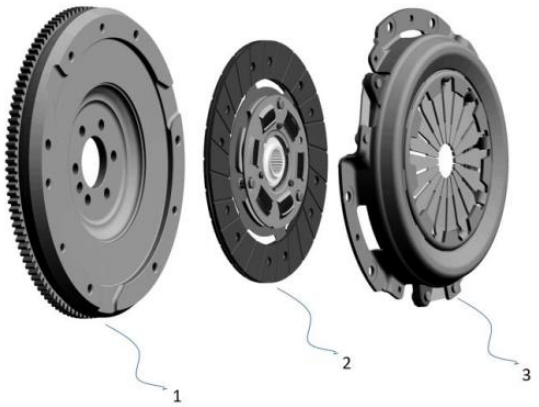
Mühendislik tasarımı süresince tasarlanan makine ya da parçalarının fonksiyonluğunu beklenen ölçüde sağlaması beklenir. Tasarım sırasında fonksiyonelliğinin önemli olmasının yanında maliyetinin de düşük olması günümüz rekabetçi pazarında son derece önemlidir. Mühendislik tasarımı fonksiyonellik ve maliyet dengesi göz önünde bulundurularak mühendislik tasarım parametreleri eniyilenmiştir. Tasarlanan makinelerin maliyetleri mühendislik tasarım süresince arge çalışmalarında belirlenebilmektedir. Arge çalışmaları kapsamında bilgisayar destekli paket programlar aracılığıyla çok kısa zaman dilimi içerisinde simülasyon ve eniyileme yapılarak hem zaman kazancı hem iş gücü kaybı azaltılarak mühendislik tasarımı yapılabilir. Mühendislik tasarımından beklenen gereklilikler deneysel veriler ile uyumluluk göstermesi halinde arge çalışmalarında büyük ölçüde verimlilik sağlar. Taşıt debriyaj elemanlarının tasarım aşamasında kavrama kesilmesi ve motordan gelen titreşimlerin sönümlenerek tork ve hızın vites kutusuna aktarılması önem arz etmektedir. Kavramanın istenildiğinde debriyaj elemanı ile kesilmemesinin sebeplerinden biri yüksek hız altında debriyaj diyafram yayının yeterli ayırma kuvvetini sağlayamamasındandır. Taşıt güç aktarma sistemlerinde kırılmaların ve gürültü problemlerinin çoğu motordan iletilen tork ve hızdaki salınımların istenilen ölçüde sönümlenememesinden kaynaklanmaktadır. Bu tür sistemlerde uygun olmayan tasarımlar vites kutusu dişlilerinde hasara yol açabilir. Bu tez çalışmasında taşıt debriyaj elemanlarından diyafram yayının üç boyutlu modeli oluşturularak Ansys APDL programında dinamik yük analizleri gerçekleştirilmiştir. Ek olarak taşıt debriyaj elemanlarından damper elemanının sürücünün anahtarı çevirmesi ile birlikte motorun çalışmasıyla oluşan tork ve hızın emniyetli bir şekilde sönümlenerek vites kutusu giriş miline aktarımı Amesim programında analiz edilerek sistem performansı değerlendirilmiştir. Başlangıçtaki debriyaj elemanlarına ait analiz verileri göz önüne alınarak matematiksel model oluşturup eniyileme yapılmıştır. Bu tez çalışması literatürde ilk olan geliştirilmiş yeni bir optimizasyon algoritmasını içermektedir. Literatürde algoritma geliştirme çalışmalarında var olan çeşitli arama, sonuçlardaki bilgiyi kullanma ve çeşitlilik mekanizmaları teoremlerinden faydalanarak kır kurdu algoritmasının (COA) performansı iyileştirilip melez geliştirilmiş bir algoritma oluşturulmuştur. Oluşturulan algoritma literatürde bulunan kısıtsız problemler ve çeşitli kısıtlı mühendislik problemleri

ile test edilip taşıt debriyaj elemanlarından iki yapıya uygulanarak gerçekliğinin kanıtlanabilmesi amaçlanmıştır.

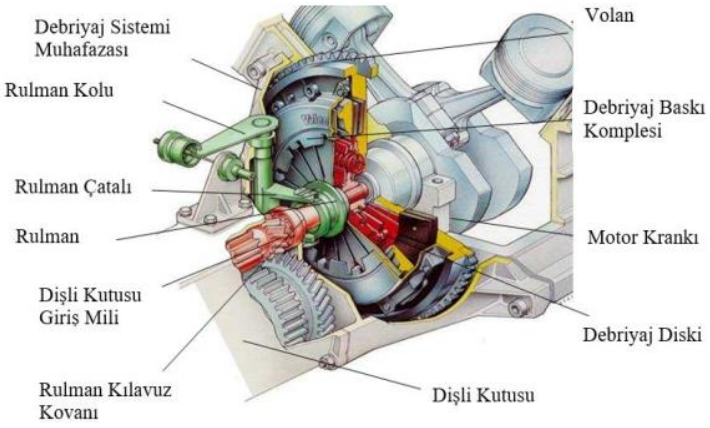
3.1. Taşıt Debriyaj Sistemi

Debriyaj sistemleri tork iletiminin ortam çeşitliliğine göre sürtünme yüzeylerinin etkisi ile olan kuru kavramalı ve hidrolik kuvvetlerin etkisiyle olan ıslak kavramalı olarak ikiye ayrılır. Bu tez çalışmasında kuru kavramalı sistemler ele alınmıştır.

Debriyaj motordan gelen tork ve hızı vites kutusu aracılığıyla tekerleklere iletirken motordan kaynaklanan titreşimleri de sönümlemekten sorumludur. Debriyaj sistemi (bkz. Şekil 3.1) motor krankı volan (1) elemanına ve debriyaj baskı komplesi (3) kapak elemanına bağlanır. Volan ile baskı komplesi arasında bulunan damper elemanını (2) baskı komplesi diyafram yayı ve plakası aracılığıyla sıkıştırarak tork ve hız akışını sağlar. Debriyaj pedalına sürücü tarafından kurs verilmediği sürece damper komplesi güç iletimini titreşimleri sönümleyerek iletmeye devam eder. Damper komplesine montajı yapılan dişli kutusu giriş mili aracılığıyla dişli kutusuna güç aktarımı yapılır. Debriyaj sistemi araç güç aktarma sisteminde motor torkunu ve hızını vites kutusu giriş miline iletimi sağlar. Vites geçişleri sırasında motor ile vites kutusu arasında iletimi keserek tork ve hız iletimini keser. Aracın çalıştırılması sırasında konforu sağlar, titreşim ve motor düzensizliklerini sönümler. Hızlı ve kolay vites değişimi sağlarken motordan gelen aşırı dalgalanma durumunda transmisyon sistemi için koruyucu olur.

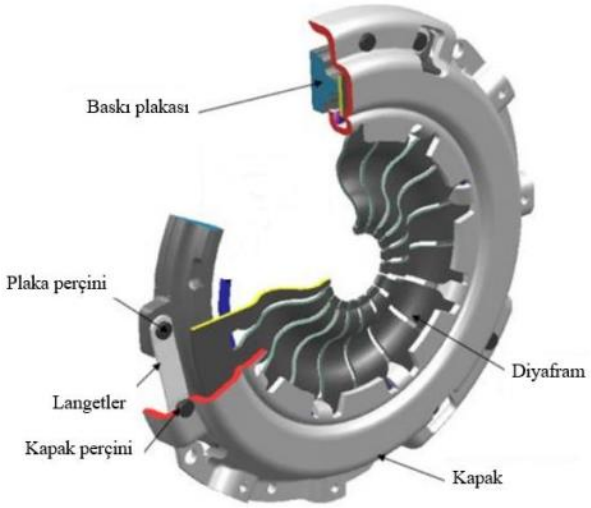


Şekil 3.1. Temsili debriyaj sistemi.

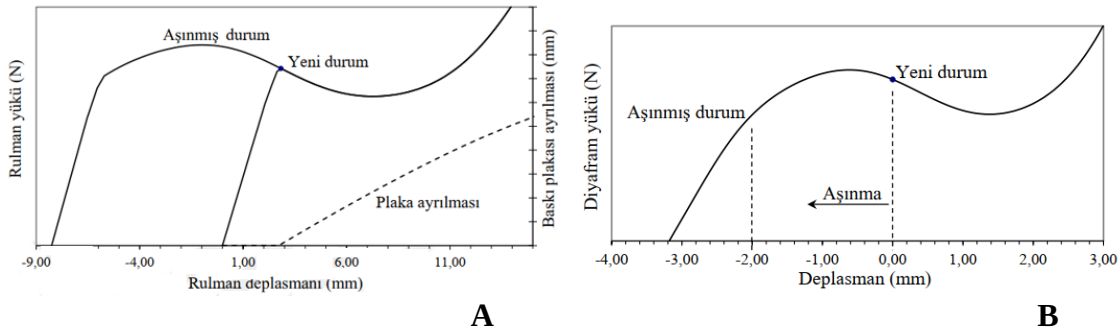


Şekil 3.2. Temsili debriyaj sistemi montajı.

Debriyaj sistemleri yüksek dinamik kuvvetler, statik kuvvetler ve termal kuvvetler altında belirli geometrik ve fonksiyonel kısıtlar ile istenen performans kriterlerini sağlamalıdır. Bu kısıtlar altında istenen performans kriterinin sağlanması için debriyaj tasarım aşamasında hesaplama ve optimizasyon faaliyetleri ile en uygun değerler elde edilmelidir. Debriyaj baskı komplesi volana baskı komplesi üzerinde yer alan kapak elemanında bulunan delikler ile birbirine bağlanır ve motor hızıyla birlikte dönmektedir. Baskı komplesinde bulunan diyafram yayı vasıtasıyla motor torkunun iletimi ve kesilmesi gerçekleştirilir. Diyafram yükü kavrama yükünü ve ayırmayı sağlayan rulman yükünü oluşturur (bkz. Şekil 3.4).



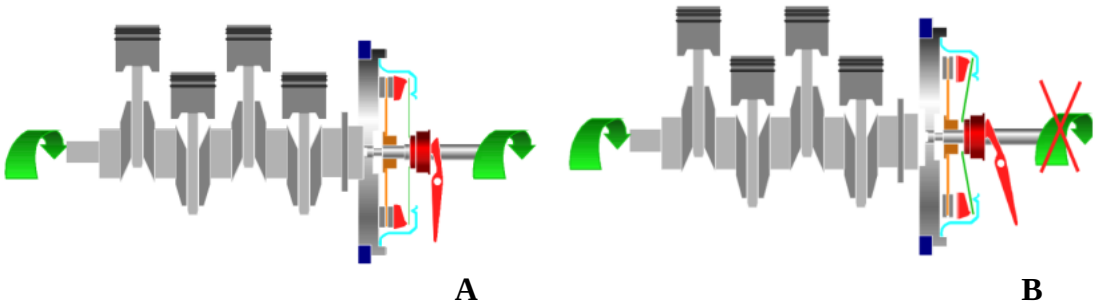
Şekil 3.3. Debriyaj baskı komplesi.



Şekil 3.4. Diyafram yük eğrileri.
A) Rulman yükü **B)** Diyafram yükü

Taşıt güç aktarma sistemlerinde oluşan kuvvetler motorun çalıştırılmasıyla oluşan tork ve hızın tekerleklere iletimi sırasında oluşur. Bu kuvvetler yüksek ve düşük motor devri altında aracın ivmelenmesi, vites değişikleri, rölanti çalışma sırasında ve motor çalıştırma sırasında oluşan sürekli kuvvetlerdir. Debriyaj sistemlerini en çok etkileyen kuvvetler yüksek motor devri altında motor titreşimlerinin yüksek olduğu kuvvetlerdir.

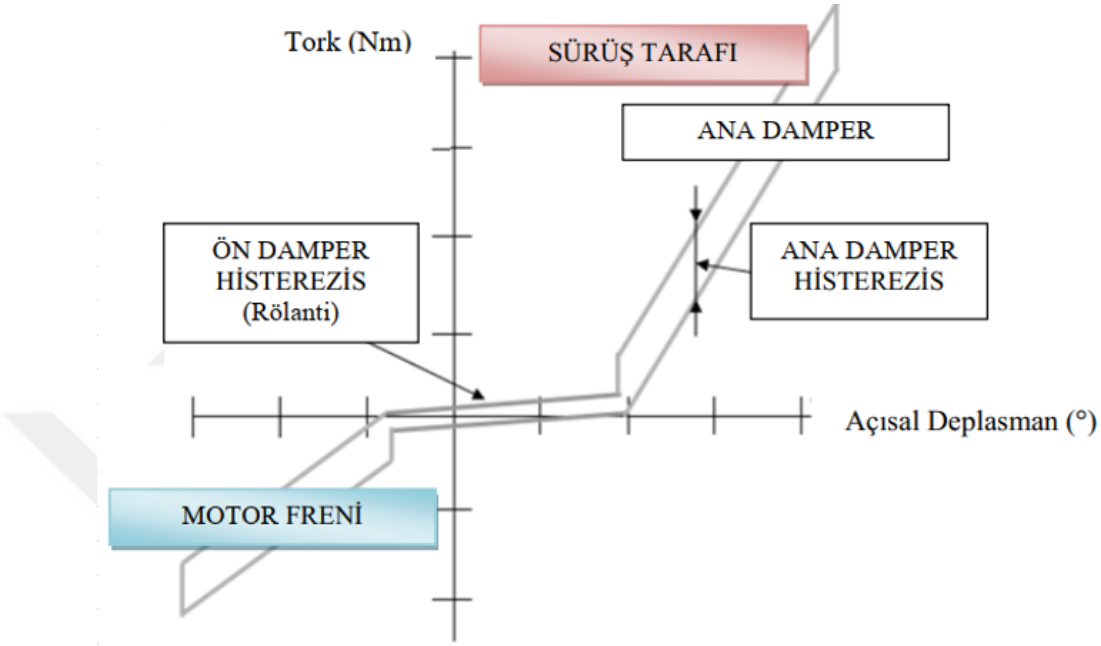
Debriyaj sisteminin güç iletimi yaptığı sürücü debriyaj pedalına basmadığı faz kavrama olarak adlandırılır. (bkz. Şekil 3.5 A) Debriyaj sisteminin güç iletimini kestiği sürücü debriyaj pedalına bastığı faz ayrılma olarak adlandırılır (bkz. Şekil 3.5 B).



Şekil 3.5. Debriyaj sistemi.
A) Kavrama durumu **B)** Ayırma durumu

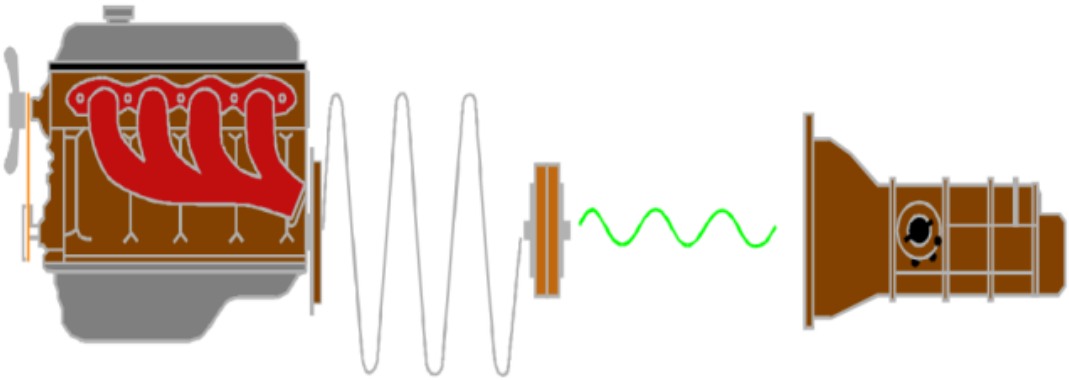
Debriyaj sisteminde ayırma ve kavrama durumu dışında bir de debriyaj pedalına tam basılmadığı durum olan ayrılmanın tam sağlanmadığı debriyaj elemanları arasında sürtünmenin olduğu kısmi kavrama fazı mevcuttur. Damper komplesi tork iletiminin ve

vites değişikliklerinin yapıldığı sırada üzerinde bulunan yaylar ve sürtünme elemanları ile birlikte tork sönümlemesini gerçekleştirir.



Şekil 3.6. Temsili damper komplesi karakteristiği.

Motordan elde edilen torku dişli kutusuna aktarırken motordaki titreşimler damper komplesinde sönümlenerek dişli kutusuna aktarılır. Damper komplesindeki yaylar motordan aktarılan torku karşılayabilecek şekilde tasarlanmalıdır ve damper komplesindeki iç sürtünme torku yaylara aktarılan torku sönümleyebilecek şekilde olmalıdır.



Şekil 3.7. Temsili debriyaj sisteminde titreşimin sönümlenmesi.

3.1. Kır Kurdu Algoritması

Optimizasyon bir problemin parametrelerinin (iş gücü, süreç, boyut, ağırlık gibi) kısıtlarını belirli koşullar için ihlal etmeden en iyi parametreleri alternatifler içerisinden seçerek belirli amacı sağlayan süreç olarak tanımlanmaktadır. Optimizasyon süreci problemin temel özelliklerini ve problemin amacını içeren modelleme ile başlar. Modelleme sistemin sürecini ve amacını yansıtan sistemin diğer sistemler ile etkileşimini belirten matematiksel ifadelerden oluşur. Eniyilenicek sistemi tanımlayan tasarım değişkenleri olan problem parametrelerinin karar süreci ile eniyileme süreci devam eder. Tasarım değişkenlerinin belirlenmesinden sonra problemin yapısına göre amaç ve kısıt fonksiyonlar belirlenir. Kısıt fonksiyon eşitsizlik ve eşitlik fonksiyonları olabileceği gibi kısıt fonksiyonun içermemesi gereken tasarım değişkeni olarak da modellenebilir. Anlamlı ve en iyi tasarım elde edebilmek için bir ya da birden fazla tasarım değişkeninin kısıt fonksiyonu etkilemesi şarttır. Kısıt fonksiyonu eşitsizlik içerdiğinde tasarım uzayında daha fazla değişkeni içerir, ancak eşitlik içerdiğinde tasarım uzayında sadece belirli değişkeni içerir. Bu nedenle eşitsizlik kısıtı eniyileme problemleri için daha basittir. Ancak bazı eniyileme problemlerinde herhangi bir kısıt fonksiyon bulunmamaktadır. Bir optimizasyon problemi aşağıdaki gibi matematiksel olarak ifade edilebilir. Arama uzayı tasarım kısıtları içerisinde tüm çözümleri içeren çözüm havuzudur. Çözüm havuzundaki en iyi çözümde amaç fonksiyonu sağlayan en iyi değerdir.

Amaç fonksiyonu:

$$f(x) = f\,(x_1, x_2, \dots, x_n) \tag{3.1}$$

x tasarım değişkenlerini ve n ise tasarım parametresinin sırasını ifade eder.
Kısıt fonksiyonları:

$$h_j(x) = h_j(x_1, x_2, \dots, x_n) = 0; \qquad j=1\dots p \tag{3.2}$$

Eşitlik kısıtlayıcılarının sayısı p ile ifade edilir.

$$g_i(x) = g_i(x_1, x_2, \dots, x_n) \leq 0; \qquad j=1\dots m \qquad (3.3)$$

Eşitlik kısıtlayıcılarının sayısı m ile ifade edilir.

Standart bir tasarım eniyileme probleminde eşitlik kısıtlarının sayısı (p) tasarım değişkenlerinin sayısına (n) eşit veya küçük olmalıdır. Aksi durumda (p>n) optimizasyon probleminin fazla denkleme sahip olduğunu gösterir ve en iyi çözümün olanaklı hale gelmesi için gereksiz denklemler silinir. Tasarım değişkenlerinin sayısı (n) eşitlik kısıtlarının sayısına (m) eşit olması durumunda eşitlik en iyi sonucu gösterdiği için eniyileme çalışması gereksizdir. Optimizasyon problemlerinde amaç tasarım havuzunda minimum değeri veren bölgede bir tasarım bulmaktır. Amaç fonksiyonu tasarım havuzunda sadece bir noktada global minimum sahip olabilir ya da birden fazla noktada global minimum sahip olabilir. Fonksiyon değeri belirli bir tasarım değişkeninin etrafında minimum değere sahip olabilir ve yerel minimum olarak adlandırılır. Eğer birden fazla minimum değer mevcutsa çözüm havuzu birden fazla yerel minimuma sahiptir. Amaç fonksiyonu çok sayıda tasarım değişkeni ve kısıt fonksiyon içerdiği durumlarda problem çok sayıda denklem içerdiği için en iyi çözüme ulaşmak için ilk tasarıma bir veya birden fazla değer verilerek en iyi şartlar sağlanıncaya kadar süreç tekrarlanır. Ancak bu süreç içerisinde özellikle çok sayıda yerel en iyi noktaya sahip karmaşık problemlerle karşılaşıldığında optimizasyon süreci yerel en iyiye takılması ve gerçek en iyiye yakınsanmaması ile karşılaşılabilir. Bu tip problemlerin üstesinden gelmek için melez yapılar oluşturarak yeni optimizasyon yöntemlerin geliştirilmesi zorunluluk olmuştur. Bu tez kapsamında literatürdeki algoritmalara göre daha düşük standart sapmaya sahip, daha iyi en iyi değerler elde edilen daha sağlam bir algoritma elde edilmiştir. Meta modelleme literatürde bulunan latin hiperküp, taguchi ortogonal dizileri gibi örnekleme metotları ile tasarım değişkenlerinin sınırları içerisinde simülasyon, analiz veya testlerle elde edilen fonksiyon değerlerine bağlı olarak sistemi temsil eden yaklaşık bir model elde etmektir. Bu çalışmada az sayıda deney ile sistemi temsil etme kabiliyetine sahip amaç fonksiyonu matematiksel olarak tanımlayan latin hiperküp örnekleme (LHÖ) yöntemi kullanılmıştır. Optimizasyon sürecinde latin hiperküp örnekleme yöntemi ile elde edilen denklem kullanılarak amaç fonksiyon değerini test veya analiz yapmadan elde etmeye olanak

sağlar. Meta-modelleme genellikle test veya analiz yapmanın çok pahalı olduğu ve çok zaman aldığı durumlarda tercih edilir. Latin hiperküp örnekleme yöntemi tasarım değişkenlerinin dağılımlarının olasılık değerlerine ve olasılık değerlerinin yoğunluk fonksiyonuna göre örneklem elde etmede kullanılan rastgele ve tabakalı bir prosedürdür. Tasarım değişkenleri olasılık yoğunluk fonksiyonuna göre olasılık dağılımı gerçekleştirilip örneklem sağlanır. Tasarım değişkeninin dağılımına göre hangi olasılıkla gerçekleşeceği aşağıdaki denklem ile ifade edilir.

$$\varphi_{\mu\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.4}$$

Olasılık yoğunluk fonksiyonu standart normal dağılım için aşağıdaki gibidir.

$$\varphi(x) = \varphi_{0,1}(x)\ = \frac{1}{\sqrt{2\pi}}\ e^{-\frac{(x)^2}{2}} \tag{3.5}$$

x: normal dağılıma sahip olan değişken;
μ: beklenen değer;
σ: varyans;

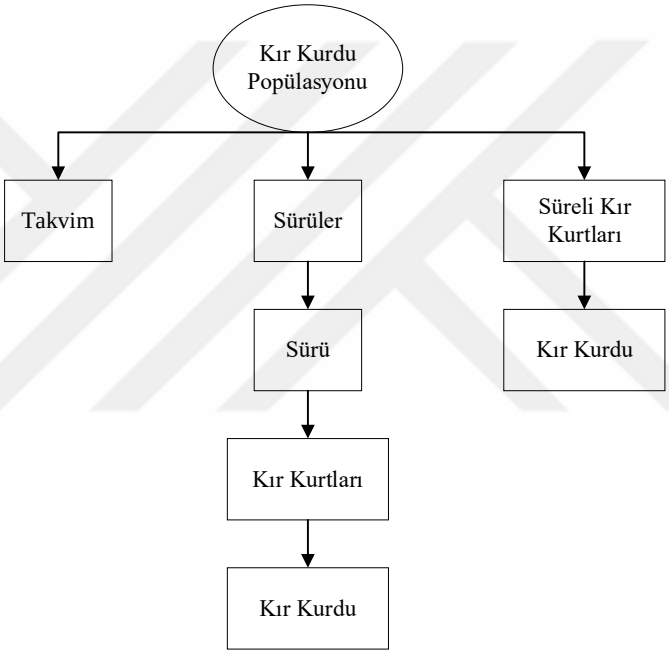
Tasarım değişkenlerinin belirli aralıkta değerini bulmayı sağlayan birikimli dağılım fonksiyonun denklemi aşağıdaki gibi ifade edilir.

$$\varphi_{\mu\sigma^2}(x) = \int_{-\infty}^{x} \varphi_{\mu\sigma^2}(u)du = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) du = \varphi\left(\frac{(x-\mu)}{\sigma}\right) \tag{3.6}$$

Latin hiperküp örnekleme metodunda dağılıma ait olasılık yoğunluk fonksiyonu örneklem sayısı kadar eşit alanlara bölünür ve rastgele bu parçalardan biri yalnız bir defa seçilir. Seçilen parça içerisinden rastgele bir nokta seçilir. Bu sayı birikimli dağılım fonksiyonunun tersine konularak tasarım noktası üretilir.

$$x = (\varphi_{\mu\sigma^2}(\textit{seçilen sayı}))^{-1} \tag{3.7}$$

Kır kurdu popülasyonları küçük sürülere sahip olmaları, belirli bir sosyal yapıya sahip olmaları ve bölgesel olmaları sebebiyle diğer popülasyonlardan farklıdır. Kır kurtları sosyal yapıya sahip olmaları ve bölgesel olmalarının yanında çevresel faktörler ile bütünleşik olduğunda popülasyon üzerinde büyük bir etkisi olduğu görülmüştür. Sonrasında, kır kurdu popülasyonunun bireysel ve sürü tabanlı modellemesi üzerine araştırmacılar çalışmaya başlamıştır. Bu modele göre popülasyon sürülere ve bölgesel olmayan kır kurtlarına bölünmüştür. Kır kurtlarının ölüm, beslenme ve üreme gibi biyolojik işlevlerinin bireysel kır kurtları ile tanımlanmıştır. Model baskınlık ve bölgesellik gibi davranışsal özellikleri net bir şekilde içerir. Model aşağıdaki gibidir.



Şekil 3.8. Kır kurdu model şeması.

Model kır kurtları, sürüler, kır kurdu etkinlikleri yılın zamanına bağlı olduğundan takvim ve sürülerden oluşan popülasyon modelinden oluşur. Her sürü belirli bir bölge için aynı miktarda yiyeceğe sahiptir. Kır kurtları doğar, sürü içinde sosyal konum için koşturur, popülasyon içinde dağılır, ürer ve ölür. Kır kurtları sürüsü içerisinde alfa, beta ve geçici bireyler bulunur. Alfa kır kurtları liderdir ve alfa kurtları öldüğünde veya sürü değiştirdiğinde beta kır kurtları alfa kır kurtlarının yerine geçer ve lider olurlar. Mevcut bir pozisyon olması durumunda geçici olarak sürüde olan kır kurtları sürünün üyesi olur. Sürü içerisinde doğan yavrular zaman geçtikçe ya ölür ya da yetişkin olarak büyür. Sürü içerisinde kır kurdu sayısı arttıkça sürüden kır kurtlarının kovulma olasılığı artar. Her kır

kurdu modelde cinsiyet, yaş, statü ve sürüsü ile karakterize edildi. Modelde olasılık fonksiyonları sıfır ile bir arasında belirlendi ve rastgele sayı seçildiğinde eğer olasılık değerinden küçükse olasılık fonksiyonu gerçekleştirildi. Yetişkin kır kurtlarının ölüm oranı çakal yaşının ikinci dereceden bir fonksiyonu olarak kabul edildi. Yavru kır kurtlarının yüksek ölüm oranı sürü içinde mevcut kaynakların miktarından kaynaklanabileceği için sabit bir ölüm oranı kullanıldı. Sürü içerisine dahil olan kır kurtlarının ölüm oranı da birey başına düşen yiyecek oranı az olacağından ölüm oranı yüksektir. Bu sürece dayanarak, Priezan ve Coelho (2018) kır kurtlarının sosyal yapısından ve birbirleri arasındaki deneyim alışverişinden ilham alarak keşif ve bilgi kullanma arasında denge sağlayan kır kurdu optimizasyon algoritmasını oluşturmuşlardır. Kır kurdu optimizasyon algoritması (COA) sürü zekâsı ve evrimsel olarak *Canis Latrans* türlerinden esinlenen popülasyon tabanlı bir algoritmadır. Kır kurtları algoritması belirtildiği gibi sosyal hiyerarşiye ve ava odaklanmadan sosyal yapıya, kır kurtları arasındaki deneyim alışverişine odaklanır. Matematiksel denklemler, çakalların davranışlarına dayalı olarak gerçekleştirilir. Her kır kurdu sürüsü, belirli bir bölge için aynı miktarda yiyeceğe sahiptir. Çakallar doğar, sürüde sosyal bir konum için rekabet eder, nüfus içinde dağılır, ürer ve ölür. Popülasyon, bu yapıdaki paketlere rastgele atanan eşit sayıda çakal (Nc) içeren paketlerden (Np) oluşur. Bu nedenle, popülasyon (n), Np ve Nc'yi çarpmaya eşdeğerdir. Bundan sonra popülasyon, arama uzayı içindeki rastgele başlangıç sosyal koşullarına göre güncellenir. Temsil edilen çözüm, t'inci an için pth paketinin cth çakalına bağlı olarak sosyal koşul soc olarak değerlendirilebilir. Popülasyon arama uzayı içerisinde sosyal koşullarına göre güncellenir. Her kır kurdu problemin bir çözümü ve sosyal durum amaç fonksiyonun sonucudur. Tüm parametreler, aynı zamanda tüm tasarım değişkenleri olan D ile temsil edilir. Algoritma problemin değişkenleri olan x kır kurtlarının sosyal koşullarına göre tasarlanmıştır (Denklem 3.8).

Çizelge 3.1. Kır kurdu algoritmasının literatürdeki sınıflandırması

Algoritma	Sınıf		Çözüm Tipi	İlham Kaynağı	Parametre
SA	Tek çözümlü	Sürü Tabanlı	Gerçek değerli	Termal denge	Başlangıç sıcaklığı, soğutma faktörü
GA	Popülasyon tabanlı	Evrim Tabanlı	İkili kodlu	En uygun adayın hayatta kalması	Çaprazlama ve Mutasyon oranları
DE			Gerçek değerli		Çaprazlama ve ölçekleme faktörü
PSO		Sürü tabanlı		Sosyal ve bilinçli deneyimler	c1,c2 ve w
ABC				İzci, işveren ve gözcü arılar	Sınırlar
Mevcut çalışma				Kır kurtlarının sosyal uyumu	Sürüden ayrılma ve dağılım olasılıkları

$$\mathrm{soc}_{c,\,i}^{p,t} = \vec{x} = (x_1, x_2, \ldots, x_D) \tag{3.8}$$

p: sürü sayısı

c: kır kurdu sayısı

t: anlık zaman

i: iterasyon sayısı

D: parametre sayısı

Arama uzayındaki olası her bir çözüme karşılık her bir kır kurdunun sosyal koşulunun karşılık geldiği başlangıç kır kurdu topluluğu oluşturulur. Kır kurtları arama uzayına rastgele dağıtılır. Çakalların sosyal koşullarına bağlı olan küresel nüfus rastgele olarak [0,1] aralığında oluşturulmuş gerçek bir rastgele sayıya (r_j) göre j. karar değişkeninin alt sınırlar (lb_j) ve üst sınırları (ub_j) aralığında yazılabilir. Topluluğun denklemi 3.9'a göre belirtilir.

$$\mathrm{soc}_{c,\,i}^{p,t} = lb_j + r_j * (ub_j - lb_j) \tag{3.9}$$

j: Kontrol değişkenidir.

r_j: Rastgele sıfır ile bir arası üretilen sayı

ub_j: Üst sınır

lb_j: Alt sınır

Kır kurdu algoritmasının mevcut sosyal koşullara ve çevreye uyumu denklem 3.10'a göre belirtilir.

$$\mathrm{fit}_c^{p,t} = \mathrm{f}\left(\mathrm{soc}_c^{p,t}\right) \tag{3.10}$$

Kır kurtları bazen sürülerini bırakıp yalnız kalırlar ya da Pe olasılığında bir sürüye katılırlar. Pe değeri 1'den büyük değerler alabileceği için sürü başına kır kurdu sayısı on dört ($N_c \leq 200$) ile limitlidir.

$$\mathrm{P_e} = 0.005 * \mathrm{N_c^2} \tag{3.11}$$

Her sürüde en iyi uyum sağlayan kır kurtları alfa olarak seçilir ve her biri yerel en iyi çözümdür. Sürüde iki adet alfa olmasına rağmen algoritmada bir adet alfa göz önüne alınır. Kır kurtları sürülerini sürdürmek için her sürünün kültürel eğilimine göre sosyal koşullarını paylaşmak için örgütlenirler. Alfa etkisine ve kültürel etkiye göre kır kurdu sosyal durumu güncellenir.

O: Her kır kurdunun sıralanmış sosyal durumunu temsil eder.

$$\mathrm{alfa}^{p,t} = \{\, \mathrm{soc}_c^{p,t} \,|\arg_{c=\{1,2,\ldots,N_c\}} \mathrm{minf}\left(\mathrm{soc}_c^{p,t}\right)\} \tag{3.12}$$

$$\mathrm{kültürel\ eğilim}_j^{p,t} = \begin{cases} O_{\frac{(N_c+1)}{2},j}^{p,t} & , \quad \mathrm{Nc\ çift} \\ \dfrac{O_{\frac{(N_c)}{2},j}^{p,t} + O_{\frac{(N_c)}{2}+1,j}^{p,t}}{2} & \mathrm{yoksa} \end{cases} \tag{3.13}$$

Kır kurtlarının kültürel eğilimi denklem 3.13'e göre hesaplanır.

$$\mathrm{s_a} = \mathrm{alpha} - \mathrm{soc}_{c1}^{p,t} \quad \text{(alfa etkisi)} \tag{3.14}$$

$$\mathrm{s_c} = \mathrm{kültürel\ eğilim} - \mathrm{soc}_{c2}^{p,t} \quad \text{(sürü etkisi)} \tag{3.15}$$

$$\mathrm{yeni_soc}_c^{p,t} = \mathrm{soc}_c^{p,t} + \mathrm{e1} * \mathrm{s_a} + \mathrm{e2} * \mathrm{s_c} \tag{3.16}$$

Başlangıç popülasyon temel alınarak rastgele iki birey etrafında yerel en iyi çözümlere ve sürünün ortalama eğilimine dayalı yeni bir topluluk oluşturulur. $e1\ ve\ e2$ rastgele üretilmiş sayılardır. Yeni sosyal duruma göre kır kurtlarının uyumu değerlendirilir.

$$\text{Yeni_fit}_c^{p,t} = f\left(\text{yeni_soc}_c^{p,t}\right) \tag{3.17}$$

Yeni uyum değerleri eski uyum değerleri ile karşılaştırılıp en iyi değerler değerlendirilir. Bu kısımda en iyi sosyal koşula sahip kır kurtlarının seçimi gerçekleştirilir.

$$soc_c^{p,t+1} = \begin{cases} \text{yeni_soc}_c^{\,p,t+1} & \text{yeni_J}_c^{\,p,t+1} < J_c^{\,p,t+1} \\ soc_c^{p,t+1} & \text{yoksa} \end{cases} \tag{3.18}$$

Yeni kır kurtlarının doğuşu, iki ebeveynin sosyal koşullarının ve çevresel etkilerin bir kombinasyonu nedeniyle sürüleri etkiler. Her sürü içerisinde doğmuş yeni bir kır kurdu sosyal popülasyona ve çevresel etkilere göre sürüye entegre edilir. Bundan sonra, sosyal durum, paketin alfa etkisi ve sosyal eğiliminden kaynaklı olarak güncellenir. Her doğan birey bir çözümdür ve her çözümün uyumu değerlendirilir.

$$pup_j^{p,t} = \begin{cases} soc_{r1,j}^{p,t} & rnd_j < P_s \quad j = j_1 \\ soc_{r2,j}^{p,t} & rnd_j \geq P_s + P_a \quad \text{veya } j = j_2 \\ R_j & \text{yoksa} \end{cases} \tag{3.19}$$

Dağılım ve birliktelik olasılıkları kültürel çeşitliliğe katkı sağlar. Dağılım olasılığı denklem 3.19'da görülmektedir. İlişkilendirme olasılığı denklem 3.20'de görülmektedir.

$$P_s = \frac{1}{D} \tag{3.20}$$

$$P_a = \frac{(1-Ps)}{2} \tag{3.21}$$

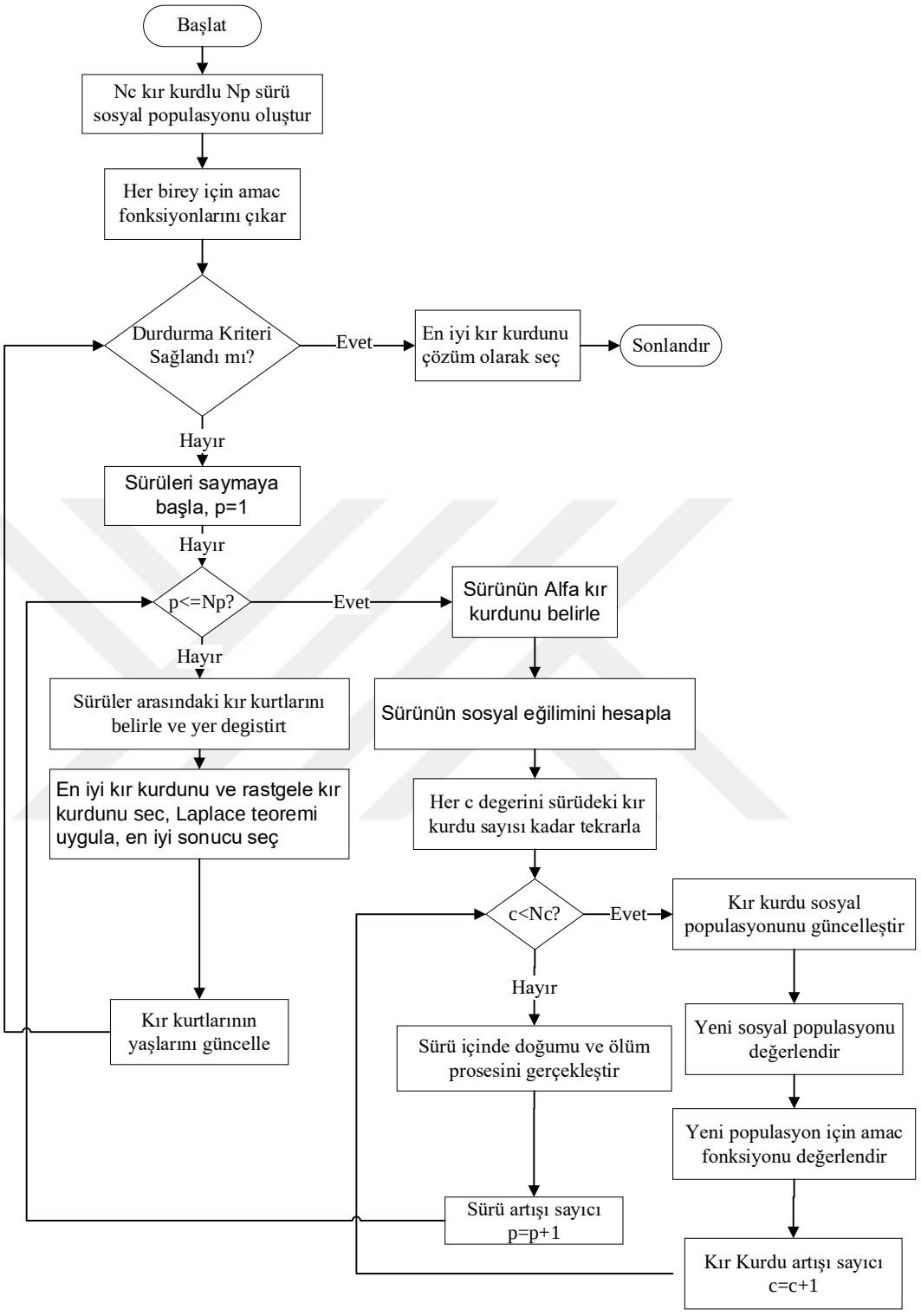
Çevreye yeni doğan kır kurtlarından daha kötü uyum sağlamış bireyler ölür. Kendini çevreye en iyi uyarlayan kır kurdu çözüm olarak kullanılır. Denklem 3.19'da elde edilen popülasyondaki yeni çözümlerin eski çözümler ile değiştirilmesi gerekip gerekmediği karar verilir.

$$soc_c^{\text{en kötü}} = \begin{cases} soc_c^{\text{en kötü}} & \text{yeni_J}_c^{\,\text{en kötü}} < J_c^{\,\text{yeni}} \\ soc_c^{\text{yeni}} & \text{yoksa} \end{cases} \tag{3.22}$$

Çözümleri çeşitlendirmek ve erken yakınsama ile yerel optimal bölgelere takılmayı önlemek için kır kurdu algoritması, Np grupları arasında kır kurdu yer değiştirmesi sağlar. Rastgele seçilen iki grup, rastgele seçilmiş iki çözüm sunacak ve konumları sonra değiş tokuş edilir. Eğer üretilen rastgele sayı denklem 3.11'den küçükse işlem gerçekleştirilir. Sonrasında algoritma akışı bittiğinde tüm sürülerdeki en iyi çözümler karşılaştırılır ve en iyi çözüm bulunur. Algoritmanın sözde kodu Şekil 3.9'da ve akış şeması Şekil 3.10'da belirtilmiştir.

1: Kontrol parametrelerini belirle
2: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur (Denk. 3.9).
3: Her birey için amaç fonksiyonlarını çıkar (Denk. 3.10).
4: **while** durdurma kriteri sağlanana kadar döngüye devam et (iterasyon sayısı)
5: **for** her p değerini sürü sayısı kadar tekrarla
6: Sürünün alfa kurdunu belirle (Denk. 3.12).
7: Sürünün sosyal eğilimini hesapla (Denk. 3.13).
8: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
9: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle (Denk. 3.16).
10: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir (Denk. 3.17).
11: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz (Denk. 3.18)
12: **end for**
13: Sürü içinde doğum ve ölüm (Denk. 3.19).
13.1 : kriter hesapla yaşam ve ölüm.
13.2 : **if** yaşam = 1 then
13.3 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
13.4 : **elseif** yaşam > 1 then
13.5 : Yeni doğan birey yaşar ve en yaşlı birey ölür
13.6 : **else**
13.7 : Yeni doğan birey ölür
13.8 : **end if**
14: **end for**
15: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir (Denk. 3.20).
16: Kır kurtlarının yaşını güncelle.
17: **end while**
18: En iyi sonucu veren kır kurdu seçilir.

Şekil 3.9. Kır kurdu algoritması sözde kodu.



Şekil 3.10. Kır kurdu algoritması akış şeması.

3.2. Kır Kurdu Algoritmasında Önerilen İyileştirme Mekanizmaları

Araştırmacılar algoritmaların performansını ve başarısını arttırmak için literatürde bulunan matematiksel teoremlerle ve başka algoritmalardaki yeteneklerle birleştirerek mevcut algoritmaları geliştirme ve iyileştirme yoluna gitmişlerdir. Araştırmacılar bu süreçte mevcut algoritmaların hızlı yakınsama, çeşitlilik, lokal arama ve global arama yeteneklerini iyileştirmek istemişlerdir. Oluşturulan yeni algoritmada beklenen en iyi sonuçların en doğru elde edilmesi olarak düşünülür.

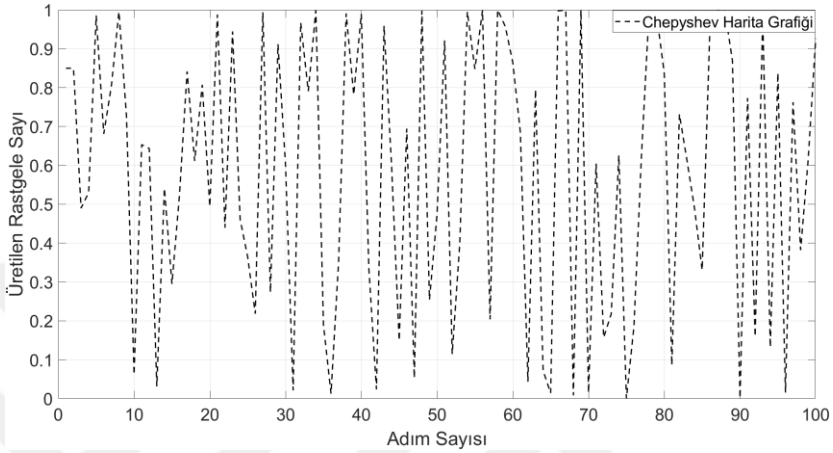
Bu tez çalışmasında, kır kurdu algoritmasındaki performans kayıplarını azaltmak amacıyla yeni bir kır kurdu algoritması önerilmiştir. Bunun için on beş farklı algoritma oluşturulmuş olup kendi içinde on farklı kaotik harita tipi için her bir algoritma ayrı ayrı değerlendirilmiştir. Yeni geliştirilmiş kır kurdu algoritması literatürde ilk kez taşıt debriyaj yapılarının optimizasyon problemine uygulanmıştır. Taşıt debriyaj yapılarının optimizasyonunda dinamik yük ve sönümleme karakteristiklerinin eniyilenmesi yapılmıştır.

3.2.1. Kaotik haritalı mekanizmalar

Kaos, rastgele olma durumudur ve başlangıç koşullarından bağımsız olarak formu sınırlama eğiliminde olan stokastik sistemlerin bir özelliğidir ve doğrusal olmayan yapıları içeren bir yapıdır. Rastgele olarak kaotik haritalar ile yayılımlı olarak üretilen sayıların tekrarlı kullanımı olmadan daha az zamanda ve daha düşük maliyet olması sebebiyle kaos tabanlı algoritmalar daha iyi sonuçlar gösterir. Kaotik mekanizma, kaotik haritaları COA'ya entegre ederek kullanıldı. Bu ekleme, düşük yakınsama hızı ve zayıf yerel optimum sorunları önleyerek COA'nın performansını iyileştirmeye yardımcı oldu. Sıfır bir aralığında farklı dağılımlar veren on farklı kaotik harita, bunların COA üzerindeki etkilerini değerlendirmektedir. Kaotik haritalar Çizelge 3.2'de temsil edilmektedir. Kaotik diziler oluşturan kaotik haritalar, keşfi geliştirmek için paket başlatma, sömürüyü geliştirmek için kır kurdu seçimini ve göçünü manipüle etmek için kullanıldı. Her harita kaotik kır kurdu optimizasyon algoritmasını (CCOA) test eder.

3.2.1.1. Chebyshev harita

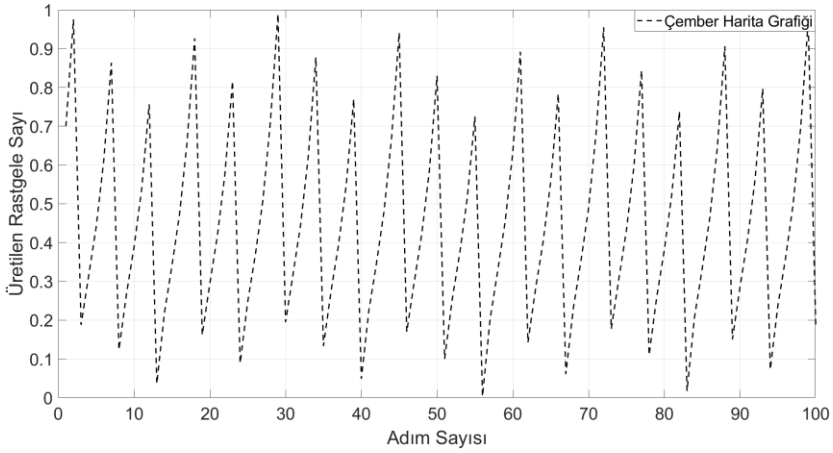
Chepyshev haritası ortogonal polinomlar dizisidir. Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi ve yüz iterasyon için harita grafiği Şekil 3.11'de gösterilmiştir.



Şekil 3.11. Chepyshev harita grafiği.

3.2.1.2. Çember harita

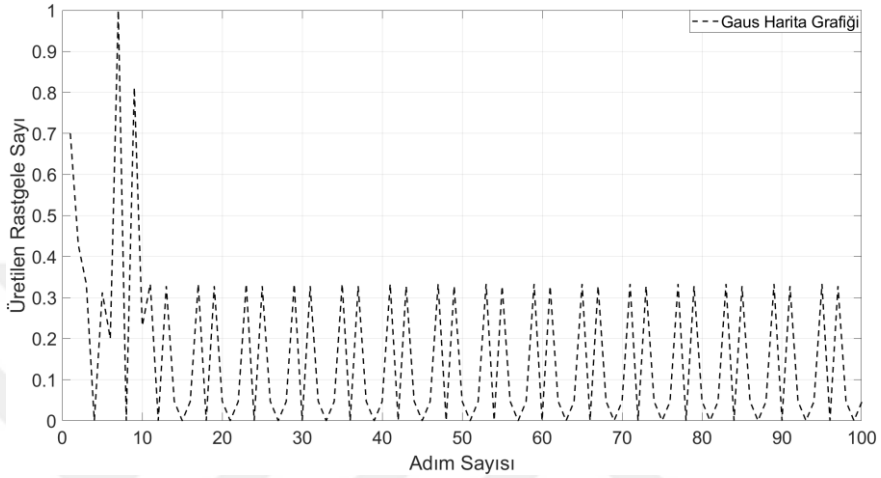
Çember haritalar ile sayı üretimi optimizasyon işlemlerinde sıklıkla kullanılır. Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Kontrol parametreleri a değeri sıfır nokta beş, b değeri sıfır nokta iki başlangıç noktası sıfır nokta yedi ve yüz tekrarlama için harita grafiği Şekil 3.12'de gösterilmiştir.



Şekil 3.12. Çember harita grafiği.

3.2.1.3. Gaus harita

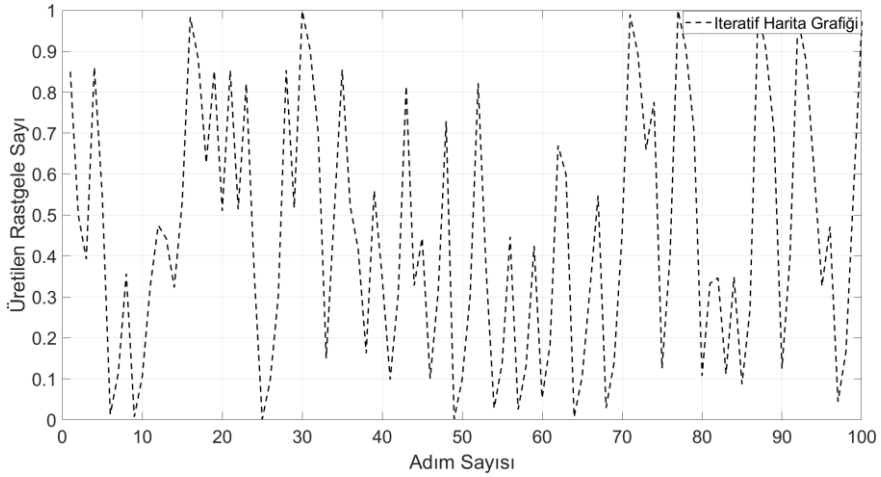
Gaus haritalar ile sıfır bir aralığında rastgele sayı üretimi optimizasyon işlemlerinde sıklıkla kullanılır. Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi ve yüz tekrarlama için harita grafiği Şekil 3.13'te gösterilmiştir.



Şekil 3.13. Gaus harita grafiği.

3.2.1.4. Iteratif harita

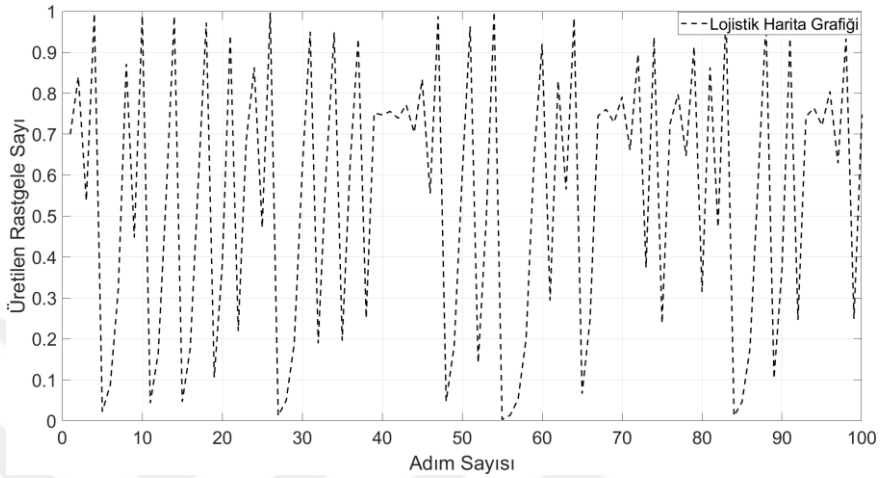
Iteratif haritalar ile sıfır bir aralığında rastgele sayı üretimi sıklıkla kullanılır. Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi ve yüz tekrarlama için harita grafiği Şekil 3.14'te gösterilmiştir.



Şekil 3.14. Iteratif harita grafiği.

3.2.1.5. Lojistik harita

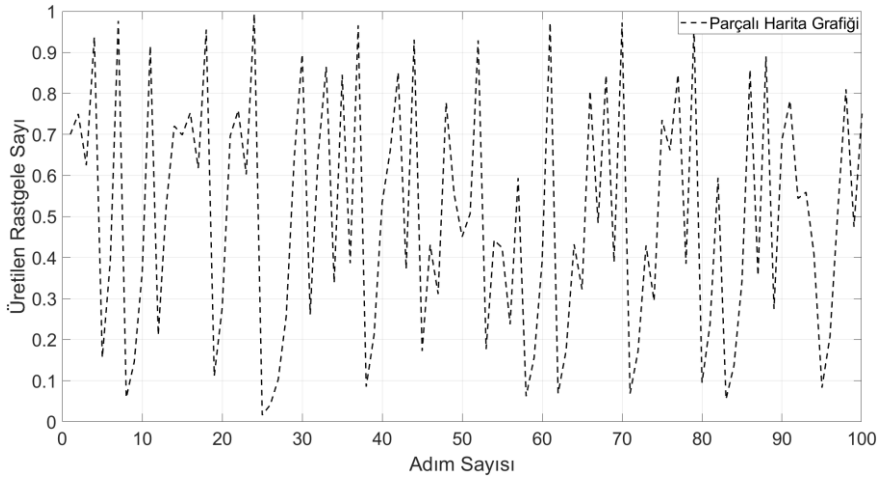
Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi, a değeri dört ve yüz tekrarlama için harita grafiği Şekil 3.15'te gösterilmiştir.



Şekil 3.15. Lojistik harita grafiği.

3.2.1.6. Parçalı harita

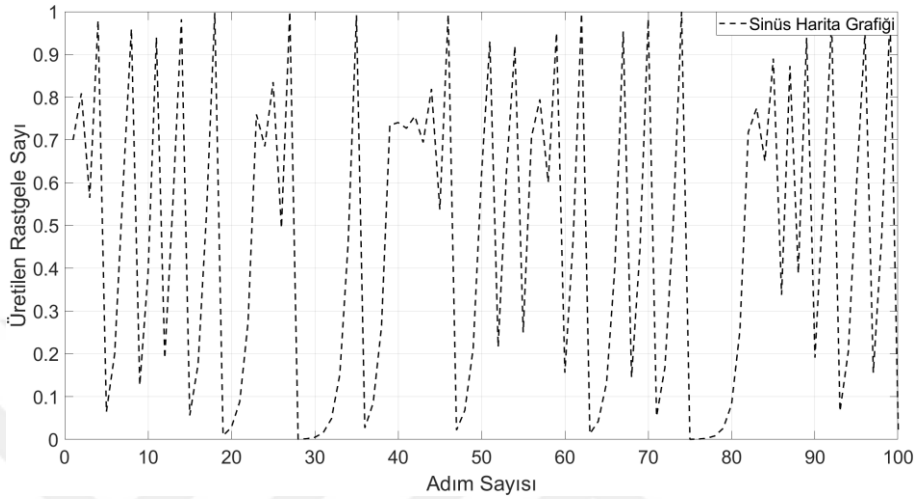
Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi, P değeri sıfır nokta dört ve yüz tekrarlama için harita grafiği Şekil 3.16'da gösterilmiştir.



Şekil 3.16. Parçalı harita grafiği.

3.2.1.7. Sinüs harita

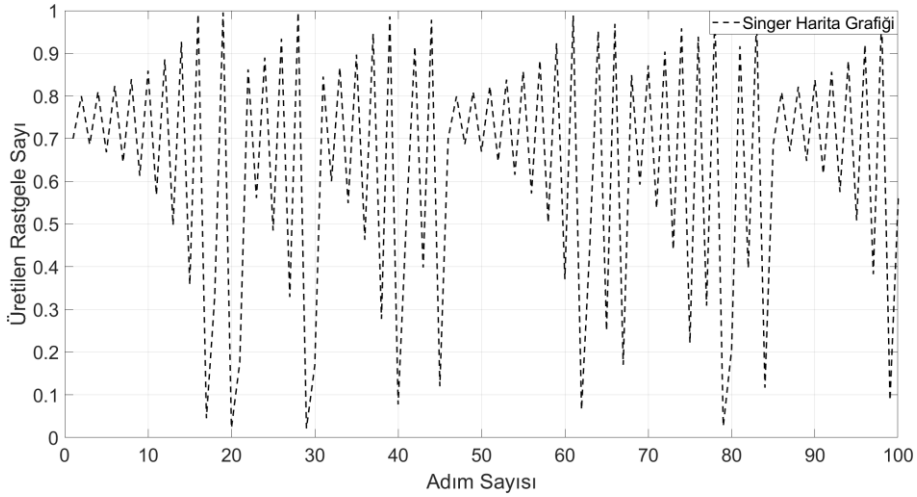
Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç sıfır nokta yedi, a değeri dört ve yüz tekrarlama için harita grafiği Şekil 3.17'de gösterilmiştir.



Şekil 3.17. Sinüs harita grafiği.

3.2.1.8. Singer harita

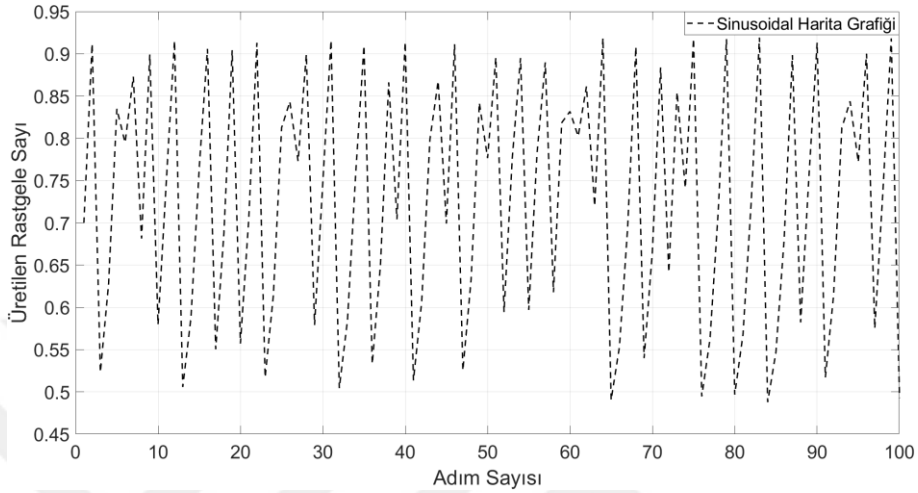
Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi, başlangıç değeri bir nokta sıfır yedi ve yüz tekrarlama için harita grafiği Şekil 3.18'de gösterilmiştir.



Şekil 3.18. Singer harita grafiği.

3.2.1.9. Sinüsoidal harita

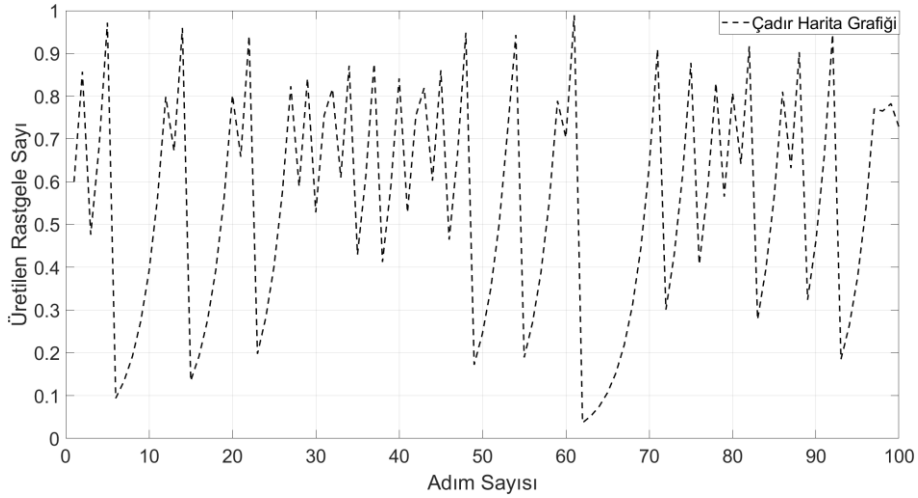
Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi, a değeri iki nokta üç ve yüz tekrarlama için harita grafiği Şekil 3.19'da gösterilmiştir.



Şekil 3.19. Sinüsoidal harita grafiği.

3.2.1.10. Çadır harita

Çizelge 3.2'de matematiksel gösterimi belirtilmiştir. Başlangıç noktası sıfır nokta yedi ve yüz tekrarlama tekrarlama için harita grafiği Şekil 3.20'de gösterilmiştir.



Şekil 3.20. Çadır harita grafiği.

3.2.2. Kaotik bölgesel arama mekanizması

Kaotik yerel arama, amaç fonksiyonlarının çözümlerinin kalite ve yakınsama hızı bakımından daha iyi hale getirilmesini sağlayan bir stratejidir. Kaotik yerel arama ile algoritmanın erken yakınsamasından ve arama hassasiyetinden kaynaklı zayıflığın giderilmesi için kullanılır. Kaotik yerel arama stratejisi ile algoritmanın arama davranışını arttırır ve yerel optimuma takılmayı önler. Böylece kaosun kendini tekrar etmeyen doğasından faydalanılarak popülasyon çeşitlendirilir ve daha geniş bir arama uzayı keşfedilir. Bu nedenle kaotik yerel arama keşif ve bilgiyi kullanmayı dengeler. Kaotik haritalar, kaotik yerel aramada kaotik bir durum oluşturmak için kullanılır. Mevcut küresel en iyi çözüme yakın daha iyi bir çözüm olup olmadığını anlamak için bir arama ve süreç karşılaştırması vardır. Kaotik yerel arama matematiksel olarak aşağıdaki gibidir.

$$x_{g'}(k) = x_g(k) + r(Ub - Lb)\left(z^j(k) - 0.5\right) \tag{3.23}$$

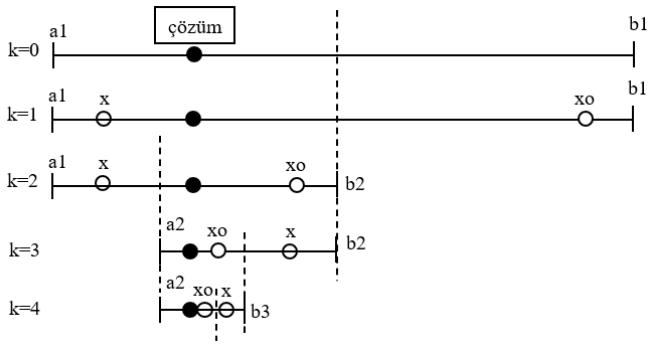
- $x_g(k)$ Mevcut en iyi kır kurdudur.
- $x_{g'}(k)$ en iyi kır kurdu olarak yeni bir kır kurdudur.
- $z^j(k)$ Kaotik haritalar tarafından üretilen değişkendir, j kaotik haritayı temsil eder ve k yineleme sayısıdır.
- Sıfır nokta beş ayarlanmış bir parametredir.
- $r(Ub - Lb)$ Çözünürlüktür, uzunluğu değişkenlerin üst sınırı ile alt sınırı arasında olan kaotik yerel aramanın arama uzunluğudur. Uzun arama uzunluğuna sahip kaotik yerel arama, gereksiz amaç fonksiyonu değerlendirmesine neden olabilir. Bu nedenle, en iyi çözümü keşfetmek ve en iyi çözüm kalitesini sağlamak için arama uzunluğunun küçük olarak ayarlandı.
- CLS yöntemi, literatürde algoritmaların performansını artırmanın yaygın bir yoludur. Bu tezde, Coa'nın performansını iyileştirmek için Cls yöntemi kullanılmaktadır.

Çizelge 3.2. Kaotik haritalar

No	İsim	Kaotik Harita	Aralık
1	Chebyshev	$X_{i+1}=\cos(i\cos^{-1}(x_i))$	(-1,1)
2	Circle	$X_{i+1}=\text{mod}(x_i+b-(\frac{a}{2\pi})\sin(2\pi x_k),1)$, a=0.5 ve b=0.2	(0,1)
3	Gauss/mouse	$x_{i+1}=\begin{cases} 1 & x_i=0 \\ \frac{1}{mod(x_i,1)} & otherwise \end{cases}$	(0,1)
4	Iterative	$x_{i+1}=\sin(\frac{a\pi}{x_i})$, a=0.7	(-1,1)
5	Logistic	$x_{i+1}=ax_i(1-x_i)$, a=4	(0,1)
6	Piecewise	$x_{i+1}=\begin{cases} \frac{x_i}{P} & 0\le x_i<P \\ \frac{x_i-P}{0.5-P} & P\le x_i<0.5 \\ \frac{1-P-x_i}{0.5-P} & 0.5\le x_i<1-P \\ \frac{1-x_i}{P} & 1-P\le x_i<1 \end{cases}, P=0.4$	(0,1)
7	Sine	$x_{i+1}=\frac{a}{4}\sin(\pi x_i)$, a=4	(0,1)
8	Singer	$x_{i+1}=$ $\mu(7.86x_i-23.31x_i^2+28.75x_i^3-13.302875x_i^4)$, μ=1.07	(0,1)
9	Sinusoidal	$x_{i+1}=ax_i^2\sin(\pi x_i)$, a=2.3	(0,1)
10	Tent	$x_{i+1}=\begin{cases} \frac{x_i}{0.7} & x_i<0.7 \\ \frac{10}{3}(1-x_i) & x_i\ge 0.7 \end{cases}$	(0,1)

3.2.3. Karşıt tabanlı öğrenme mekanizması (obl)

Karşıt tabanlı öğrenme mekanizması, mevcut çözümün tersi yönünde en uygun çözümü arama eğilimindedir. Diğer bir deyişle, uygulanabilir çözümü hesaplarken zıt çözümü hesaplar ve değerlendirir, iki çözümden en iyisini sağlayan birey en iyi olarak seçilir. Optimal çözüm, mevcut çözümün tam tersi ise, OBL yöntemi daha hızlı yakınsar. Aksi takdirde, yakınsama süresi daha fazla zaman alacaktır. Bu nedenle, OBL yöntemi bir sonraki yinelemeyi atlamadan önce her bir aday çözümün ters yönünü alt adımlarda değerlendirir. $x(k)$ 'in zıt sayısı, aralıkta, üst sınırda ve alt sınırda [Ub, Lb] $x_0(k)$ olarak ifade edilir. Karşıt tabanlı öğrenme mekanizmasının uygulanmasının sebebi erken yakınsama problemini önlemektir.



Şekil 3.21. Karşıt tabanlı öğrenme mekanizması.

$$x_0(k) = Ub + Lb - x(k) \quad (3.24)$$

Kır kurdu algoritması üzerinde karşıt tabanlı öğrenme sonucu elde edilen aday çözümlerin konumları Spearman korelasyon katsayısı ile ölçülür. Spearman korelasyon katsayısı değeri 1'den küçük ise aday çözüm uygun değildir.

$$r_R = 1 - \frac{6 * \sum_i^n (x_0(k) - x(k))^2}{(x_0(k) - x(k)) * ((x_0(k) - x(k))^2 - 1)} \quad (3.25)$$

3.2.4. Uyumluluk uzaklık dengesi (fdb) seçim mekanizması

Popülasyon tabanlı meta-sezgisel arama süreçlerinde çözüm adayları popülasyon içerisinden seçilir ve çözüm arama süreci yürütülür. Bu süreçte en iyi çözüm adaylarının keşfedilmesi için çözümlerin yerel en iyi çözümlerden kurtarılması gerekmektedir, bunun için çözüm havuzunda referans çözümlerden daha başarılı çözümlerin keşfedilip çeşitliliğin sağlanması gerekmektedir. Çözüm adaylarının seçimi rastgele olarak çeşitliliğe katkı sağlamak için yapılabilir. Rastgele çözüm adayları seçiminde çözümün uygunluk değeri dikkate alınarak en iyi çözüm adayları doğrultusunda çözüm araması gerçekleştirilir. Çözümlerin seçimi rulet çarkı ve turnuva yöntemleri ile de sağlanabilir. Bu yöntemde çözümlerin uygunluk değerlerine göre çözümlerin seçilme olasılığı hesaplanır. Rulet çarkı yönteminde seçim işlemi rastgele gerçekleşirken turnuva yönteminde seçim işlemi turnuvaya katılacak adayların rastgele seçilmesi ve seçilen adayların arasından en uygun değere sahip olanın çözüm adayı olarak seçilmesi ile gerçekleştirilir. Bu çözüm adaylarının seçiminde beklenen arama sürecinin yönünün etkili bir şekilde belirlenmesi ve arama operatörlerinin işinin kolaylaştırılmasıdır. Çözüm adaylarının rastgele bir seçimi söz konusu olursa erken yakınsama olabilir ve yetersiz çeşitlilik sağlandığı için çözüm adaylarının konumları birbirlerine çok yakın hale gelip yerel en iyi çözüm noktası problemi ile karşılaşılabilir. Bu durumu engelleyebilecek tek mekanizma algoritmanın çeşitlilik operatörüdür. Bu yüzden algoritmaların arama performansını iyileştirmek için uygunluk uzaklık dengesi (FDB) yöntemi geliştirilmiştir. FDB yöntemi ile her adayın seçim süreci için aday çözümün başarısını gösteren uygunluk değerine ve çözüm adayı ile en iyi çözüm arasındaki mesafeyi gösteren uzaklık değerine

bağlı puan hesaplanır. Bu yöntem ile algoritmaların arama performansı FDB yöntemi ile iyileşir ve yerel en iyi noktalara takılmayı önler. Fdb yöntemi temel olarak aday çözümün başarısını tanımlayan uygunluk değerlerinin ve her bir aday için en iyi çözüm adayından çeşitliliği tanımlayan uzaklık değerlerinin hesaplanmasına dayanır. Uzun mesafe değeri, adayların birbirine benzemediğini gösterir. FDB yöntemi, algoritmaların aranabilirliğini artırarak yerel optimal tuzakları önlemek için yüksek bir uygunluk değerine ve uzun mesafe değerine sahip olmayı amaçlar. Aynı anda arama uzayında birbirine benzeyen adayların seçilmesini engeller. Bu sayede popülasyon içindeki aday çözümlerin en iyi çözüme yakın ve uygunluk değeri yüksek aday çözümler belirlenir. Sonrasında en iyi aday çözüm için yeni üyelerin seçilmesi sağlanır.

Öklid mesafesi (Dp), problem boyutlarına (m), popülasyon sayısına (n) ve iterasyon sayısına (i) göre aşağıdaki gibi verilen denklemlerle hesaplanır.

$$D_{P_i} = \sqrt{(p_{i[1]} - p_{en\ iyi[1]})^2 + \cdots (p_{i[m]} - p_{en\ iyi\ [m]})^2} \tag{3.26}$$

$$_{i=1}^{n}\forall P_i, \tag{3.27}$$

$$D_P = \begin{bmatrix} d_1 \\ \cdot \\ \cdot \\ \cdot \\ d_n \end{bmatrix}_{nx1} \tag{3.28}$$

d_1: her bireyin mesafesini temsil eder

Normalize uyumluluk= norm$F_{[i]}$;

Normalize uzaklık = norm$D_{P_{[i]}}$;

[0,1] aralığındaki uygunluk değeri üzerindeki etki katsayısı = w ;

FDB puanı = $S_{P_{[i]}}$;

Bireysel FDB puanı = s_1

$$S_{P_{[i]}} = \text{w} * \text{norm}F_{[i]} + (1 - w) * \text{norm}D_{P_{[i]}} \tag{3.29}$$

$$S_P = \begin{bmatrix} s_1 \\ \cdot \\ \cdot \\ \cdot \\ s_n \end{bmatrix}_{nx1} \tag{3.30}$$

FDB, yerel çözümlerde sıkışıp kalmanın önlenmesini sağlarken ve düşük yakınsama arama sürecini sağlarken en iyi adayı belirlemekte etkili bir yöntemdir.

Uygunluk uzaklık dengesi seçim metodu sözde kodu

1: hedef uygunluk değerini sağlayan veri serisini seç

2: **if** minimum uygunluk değerini maksimum uygunluk değeri ile aynı olup olmadığını veya toplam uyumluluk değerinin sonsuza gidip gitmediğini veya minimum uyumluluk değerini sağlayan parametrelerin toplamı sonsuzdan küçük olup olmadığını kontrol et

3: Popülasyondan rastgele bir veri seç

4: **else**

5: **for** i = 1: popülasyon sayısı

6: **for** j = 1: boyut

7: En iyi değerin popülasyondaki her bir değere göre farklarını topla (3.26)

8: **end**

9: Her bir değerin en iyiye uzaklığını her popülasyon üyesi için hesapla (3.28)

10: **end**

11: Minimum uyum değerini hesapla,

12: Maximum uyum değeri ve Minimum uyum değeri arasındaki farkı hesapla

13: Minimum uzaklık değerini hesapla,

14: Maximum uzaklık değeri ve Minimum uzaklık değeri arasındaki farkı hesapla

15: **for** i = 1 : popülasyon sayısı

16: uyumluluk değerlerini normalize et [0,1]

17: uzaklık değerlerini normalize et [0,1]

18: uzaklık ve uyumluluk değerlerini topla, rastgele değerle çarp (3.29)

19: uyum puan vektörünü oluştur (3.30)

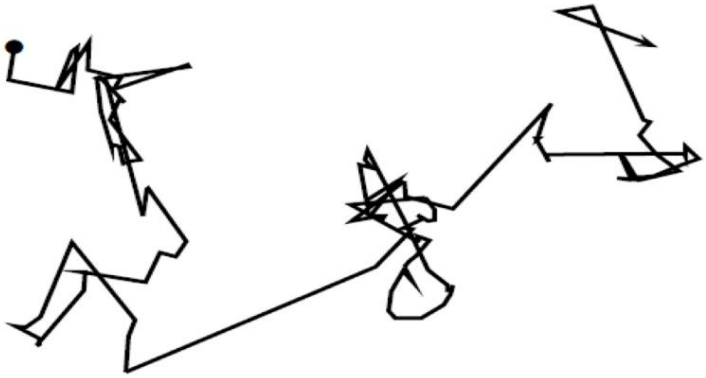
20: **end**

21: Arama amacına göre veriyi popülasyondan seç.

22:**end**

3.2.5. Levy uçuş mekanizması

Doğadan esinlenen birçok fenomen, Şekil 3.22'de yiyecek kaynaklarına yönelik avlanma ve hayatta kalma için tipik rastgele karakteristiği göstermiştir. Bu nedenle, arama uzayında optimum çözüm adaylarını bulmak için optimizasyon algoritmalarında Levy uçuş karakteristiği tercih edilmiş ve kullanılmıştır.



Şekil 3.22. Kalın bir nokta ile işaretlenmiş başlangıç noktasından başlayarak elli ardışık adımdan oluşan Lévy uçuşları.

Lévy uçuşları aşağıdaki gibi formüle edilmiştir.

Minimum adım boyutu = μ;

Minimum örnek boyutu = s;

Ölçek parametresi olarak Levy uçuş kontrol parametresi = μ;

Levy dağılımı = L;

Levy dağılımı, algoritmanın keşif yeteneklerini güçlendirmek için aday çözümlerin arama alanı içinde etkin bir şekilde konumlandırılmasını sağlar.

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left(-\frac{\gamma}{2(s-\mu)}\right) \frac{1}{(s-\mu)^{3}/_{2}}, & 0 < \mu < s < \infty \\ 0 & yoksa \end{cases} \quad (3.31)$$

Levy Uçuş sözde kodu

1: **Girdi:** Mevcut pozisyon, Pozisyon yönü, alt Limit, Üst Limit

2: **Çıktı:** Yeni pozisyon

3: Adım uzunluğunu hesapla (S)

4: Fark faktörü hesapla $DF = 0.01 \times S \times$ [Mevcut pozisyon − Pozisyon yönü]

5: Yeni pozisyonu hesapla

Yeni pozisyon = Mevcut pozisyon+$S \times rand$ ( ) ×[$size$(Mevcut pozisyon)]

6: Limit kontrolü yap

7: **end**

3.2.6. Laplace crossover mekanizması

Laplace çaprazlama teorisi temel olarak Laplace dağılım teorisine dayalı iki yeni çözüm üretir. Laplace dağılımının yoğunluk fonksiyonu:

$$f(x) = \frac{1}{2b}\exp\left(-\frac{|x-a|}{b}\right), -\infty < x < \infty \tag{3.32}$$

Laplace dağılımı:

$$F(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{|x-a|}{b}\right), & x \leq a \\ 1 - \frac{1}{2}\exp\left(\frac{-|x-a|}{b}\right), & x > a \end{cases} \tag{3.33}$$

Laplace dağılımından yararlanılarak rastgele β sayısı üretilir.

$$\beta = \begin{cases} a - b * \log(u), & u \leq 1/2 \\ a + b * \log(u), & u > 1/2 \end{cases} \tag{3.34}$$

a: Pozisyon parametresi

b: Ölçek parametresi b > 0

u: Rastgele üretilen sayı

β: Laplace dağılımı tarafından rastgele oluşturulur, u sıfır ile bir arasında rastgele bir sayıdır, a konum parametresidir ve b bir ölçek parametresidir.

$$\text{Çözüm}_{yeni1} = \text{Çözüm}_{eski1} + \beta * (\text{Çözüm}_{eski1} - \text{Çözüm}_{eski2}) \quad (3.35)$$

$$\text{Çözüm}_{yeni2} = \text{Çözüm}_{eski2} + \beta * (\text{Çözüm}_{eski1} - \text{Çözüm}_{eski2}) \quad (3.36)$$

Yeni çözümler eski çözümlerin konumuna göre simetrik olarak yerleştirilmiştir. Daha küçük β değerleri için yeni çözümler eski çözümlere yakındır, daha büyük β değerleri için yeni çözümler eski çözümlerde uzaktadır. Laplace Crossover (LC) mekanizması, yüksek bir keşif oranı ile daha iyi bir çeşitlilik popülasyonu sağlar.

3.3. Geliştirilmiş Kır Kurdu Algoritması

Kır kurdu algoritması ilk olarak J. Pierezan ve L. Coelho (2018) tarafından iki bin on sekiz yılında sunulmuştur. Bu algoritmanın temeli kır kurdu sürüsünün doğumunu, büyümesini, ölümünü ve göçünü tasvir eder. Kır kurdu algoritmasının performansını iyileştirmek için iyileştirme stratejilerinin avantajlı yönlerini birleştirerek etkili ve verimli bir geliştirilmiş kır kurdu algoritması oluşturulmuştur. Geliştirilen algoritma, tasarım alanının artan keşif kapasitesi ve artan sömürü kapasitesi arasında iyi bir denge sağlamayı amaçlamaktadır. Temel fikir, geliştirilmiş algoritmanın ilk yineleme süresini artırarak tasarım alanı içindeki uygun bölgeleri tespit etmek için çaba göstermesidir. Yeni kır kurdu grubu, her sürünün en iyi kır kurtları tarafından üretilir. Yinelemeler geçtikçe, geliştirilmiş algoritma her yerel alanda en iyi tasarımı arar. Yeni kır kurdu grubu, tüm sürü arasında en iyi kır kurdu tarafından çoğalır. Ayrıca, mevcut arama grubundan uzakta yeni tasarımlar oluşturmak için FDB, Laplace crossover ve kaotik yerel arama operatörleri kullanılır. Geliştirilen algoritma içerisinde probleme yönelik arama uzayı oluşturulur, algoritma parametreleri ayarlanır. Başlangıç popülasyonu bireyler için belirlenen alt ve üst limitler dikkate alınarak oluşturulur. Başlangıç popülasyon oluşturulduktan sonra popülasyondaki bireylerin uygunluk değeri değerlendirilir. Daha sonra kır kurdu sürüsünün sosyal yapısına bağlı olarak arama adımları başlar. Daha iyi

bireylerin bir sonraki nesle aktarımı için seçim aşaması başlatılır. Seçim aşamasında yüksek popülasyon çeşitliliğini korurken yakınsama verimliliğinin arttırılması ve diğer algoritmalara göre üstün performans göstermesi amaçlanır. Geliştirilmiş kır kurdu popülasyonu içerisinde her bir kır kurdunun uyumu ve en iyi kır kurdu ile arasındaki mesafeye göre popülasyonun eğilimi belirlenerek sürünün kır kurtlarının genel davranışları üzerindeki etkisi hesaplandı. Bu şekilde bir yöntem ile sadece temel fonksiyonların değil, yanıltıcı ve düzensiz problemlerinden de üstesinden gelinmiştir. İyi çözümlere öncelik vermek yeniden üretim sürecinde bilginin kullanımıdır. Ancak iyi çözümler tercih edildiğinde ortalama ve zayıf çözümler yeniden üretim sürecinde gözden çıkarılır. Bu nedenle popülasyon çeşitliliği azalır en iyi çözüm yerel en iyi noktada takılır ve erken yakınsama gerçekleşir. Birçok algoritma en uygun çözüme yakın değer bulduğunda arama uzayını keşfetmeyi durdurur ve bu durum algoritmanın yerel bir en iyi noktaya takılıp kalmasına ve küresel değeri kaçırmasına neden olabilir. Bu yüzden nüfusun bir sonraki popülasyonu mevcut popülasyonun en uygun bireyleri ve sürünün sosyal eğilimine göre kaotik olarak oluşturulur. Bu süreçte popülasyon çeşitliliği canlı tutularak en uygun çözüme yakınsama sağlandı, keşif ve bilginin kullanımı arasındaki dengenin ayarlanmasına yardımcı olunarak algoritma performansını arttırıldı. Yeni güncellenen popülasyonda her popülasyon üyesi değerlendirilerek uygunluk değerine göre popülasyon üyesi olup olmamasına karar verilir. Bu süreç sonucunda yeni nesil popülasyon kalitesi bir önceki popülasyona göre daha kaliteli duruma gelir. Daha iyi popülasyona sahip olma sürecinde iyi çözümlerin kopyalarının yerleşmesi olasılığı kaotik işlem ile engellenmiştir. Daha sonra yeni popülasyon içerisinde seçim operatörü ile iki kır kurdu kullanarak yavru bireyler Levy uçuş teoremine göre kaotik olarak üretilir. Algoritma çalıştırıldıktan sonra yavru bireylerin doğum parametresi Levy uçuş parametresine göre ayarlanır. Bu sayede algoritma davranışının her problemde en iyi değeri yakalaması sağlanabilir. Çünkü parametre ayarlanmadığı durumda algoritma bir problemde en iyi değeri bulabilirken başka bir problemde en iyi değeri yakalamayabilir. Bu süreçte yeni bireyler çeşitliliği arttırarak çözümün yerel en iyide sıkışmasını engeller ve algoritmanın keşif yeteneğini daha etkin kılar. Bu yüzden daha iyi bir küresel arama sağlanması için Levy uçuş arama stratejisinden yararlanılmıştır. Kır kurtlarının popülasyonda sürdürdüğü toplam süre göz önüne alındığında popülasyona katılan yeni birey uyum değerleri de göz önüne alınarak en yaşlı birey ile yer değiştirir. Ancak bu

yöntem popülasyonda en iyi bireylerin elenmesine sebep olabilir. Algoritma içerisinde arama işlemi süresince algoritma parametrelerini Levy uçuş arama stratejisine göre ayarlamak bu yüzden çok önemlidir. Kır kurtları sürüler arasında kaotik olarak sıfır ile bir aralığında geçiş olasılığı ile yer değiştirerek sürüler arasında çeşitliliği arttırır ve en iyi çözümün bölgesel bir alanda sıkışması engellenir. Algoritma içerisinde geniş çözüm araması yapılarak tüm çözümler değerlendirilir ve başlangıç en iyi çözüm belirlenir. Daha sonra kır kurdu sürüsünün sosyal yapısına bağlı olarak en iyi uyum sağlayan birey ile sürü içerisinden seçilen rastgele bir birey seçilerek Laplace çaprazlama stratejisine göre yavru bireyler üretilir. Yeni yavru bireyler uyum değerlerine göre popülasyondaki en kötü uyum sağlayan bireylerden daha iyi uyuma sahipse en kötü uyum sağlayan bireyler popülasyon içerisinde yok olur. Kır kurdu popülasyonunun her güncellemesinde popülasyon üyesi değerlendirilir ve uygunluk değerine göre popülasyon üyesi olup olmamasına karar verilir. Popülasyondaki ilk doğum/ölüm ve son doğum/ölüm seviyeleri gözlemlendiğinde tüm gruplardaki tüm çözümler ilk doğum/ölüm sırasında güncellenirken, ikinci doğum/ölüm sırasında her grup yeni bir çözüm üretir. Bu nedenle ikinci doğum/ölüm sürecinin çözüm üretme sürecine katkısı yüksektir. Bu sürece kadar algoritma içerisinde geniş çözüm araması yapılıp sonuçların değerlendirilmesi ile gelinen optimizasyon işleminin son aşamasında gelecek vaat eden çözüm bulunur. İyi bir başlangıç noktası ile dar bir arama uzayında bölgesel arama yaparak en uygun çözüme ulaşılır. Ardından algoritmada belirtilen bu süreçler sonlandırma kriteri sağlanana kadar döngüsel olarak tekrarlanır ve en uygun çözüm bulunur. Testlerde rastsallığın etkisini azaltmak için farklı başlangıç noktası içeren üç farklı çalıştırma uygulanmıştır. Aday çözümlerin oluşturulması, aday çözümlerin değerlendirilmesi ve en uygun aday çözümün tekrarlı güncellenmesi temel yoldur. En uygun çözümleri bulmak için algoritma içerisinde algoritma geliştirme stratejilerinden yararlanılarak performansı arttırılmış yeni bir algoritma oluşturulmuştur. Bu sayede geliştirilmiş kır kurdu algoritmasında akıllı stratejik algoritmalar kullanarak ilk üretilen çözümden tekrarlı yeni çözümler üretilmesiyle algoritma performansı arttırıldı. Farklı türde algoritma stratejileri ile algoritma yeni nesillerden elde edilen bilginin değerlendirmesinde daha etkin hale getirildi. Geliştirilmiş kır kurdu algoritması son gelinen noktada erken yakınsama problemini seçim baskısı problemini ve bölgesel alanda en iyi çözüm noktasında takılıp kalma olasılığını azaltmıştır. Geliştirilmiş kır kurdu algoritması arama uzayının tamamının araştırılarak

farklı aday çözümleri üretilmesi ve mevcut arama konumunda en iyi aday çözümün bulunduğu bölgeye odaklanılıp oradaki bilginin kullanılması arasındaki dengeyi iyi kurmaktadır.

Geliştirilmiş Kır Kurdu Optimizasyon Algoritmasının performansı algoritma başlangıç koşulları, problem çeşidi, problem karmaşıklığına bağlı olarak değişir. Algoritma parametrelerinin uygun değerlere getirilmesi oldukça önemlidir ve literatürde bu konuda çalışmalar yapılmaktadır. Bu tez çalışmasında literatürden farklı bir çalışmayla çeşitli matematiksel mekanizmalar kullanılarak yeni nesil hibrit geliştirilmiş kır kurdu algoritması elde edilmiştir. Tezin bu bölümünde önerilen yöntemlere ait detaylı bilgiler alt başlıklar halinde sunulmuştur.

Kır kurtlarının davranışlarının matematiksel modellemesinde her kır kurdu sürüsü belirli bir bölgede aynı miktarda yiyeceğe sahiptir. Kır kurtları doğar, sürüde sosyal konum için rekabet eder, populasyon içinde dağılır, ürer ve ölür. Kır kurdu populasyonu sürülerden oluşur ve denklem 3.8'e göre kır kurdu sosyal durumu değerlendirilir. Çakalların sosyal koşullarına bağlı olan küresel nüfusu rastgele kaotik olarak sıfır ile bir aralığında oluşturulmuş gerçek bir rastgele sayıya (chr_j) göre j. karar değişkeninin alt sınırlar (lb_j) ve üst sınırları (ub_j) aralığında yazılabilir.

$$soc_{c,i}^{p,t} = lb_j + chr_j * (ub_j - lb_j) \tag{3.37}$$

Kır kurdu algoritmasının mevcut sosyal koşullara ve çevreye uyumu denklem 3.10'a göre belirtilir. Çevreye en iyi uyum sağlayan kır kurtları her sürü için alfa olarak seçilir (Denklem 3.12).

Kır kurtları sürülerini sürdürmek için her sürünün kültürel eğilimine göre sosyal koşullarını paylaşmak üzere organize olurlar. Her adayın kültürel eğilimi yerel en iyi çözümün uzaklığı ve uyumuna göre değerlendirilir (Denklem 3.38, 3.39, 3.40). FDB yöntemine göre seçilen kır kurtları, önerilen algoritmanın sağlam küresel arama yeteneğini gerçekleştirmek için her yinelemede seçilen sürünün sosyal eğilimini hesaplar. Buradaki fikir, arama alanının homojen olarak kültürel eğilimlerini araştırarak her

sürüden seçilen kır kurtlarını sürece dahil etmektir. Her kır kurdunun mesafesi d ile belirtilir (Denklem 3.41).

$$\mathrm{fit}_{\mathrm{c}}^{\mathrm{p,t}} = \begin{bmatrix} \mathrm{f}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathrm{f}_{\mathrm{n}} \end{bmatrix}_{\mathrm{nx1}} \tag{3.38}$$

$$\mathrm{D}_{\mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{i}}} = \sqrt{\left(\mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{i[1]}} - \mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{alpha[1]}}\right)^2 + \left(\mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{i[D]}} - \mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{alpha\,[D]}}\right)^2} \tag{3.39}$$

$$_{\mathrm{i}=\ 1}^{\mathrm{n}}\forall \mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{\mathrm{i}}, \tag{3.40}$$

$$\mathrm{D}_{\mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}} = \begin{bmatrix} \mathrm{d}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathrm{d}_{\mathrm{n}} \end{bmatrix}_{\mathrm{nx1}} \tag{3.41}$$

w = [0,1] arasındaki uygunluk değeri üzerindeki etki katsayısı.

$S_{P_{[\dot{\mathrm{I}}]}}$ = Uygunluk mesafe denge puanıdır.

$$\mathrm{S}_{\mathrm{P}_{[\dot{\mathrm{I}}]}} = \mathrm{w} * \mathrm{normfit}_{\mathrm{c}}^{\mathrm{p,t}}{}_{[\dot{\mathrm{I}}]} + (1 - \mathrm{w}) * \mathrm{normD}_{\mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}}{}_{[\dot{\mathrm{I}}]}} \tag{3.42}$$

Kültürel eğilim uygunluk mesafe denge puanına göre hesaplanır.

$$\text{Kültürel eğilim}_{\mathrm{j}}^{\mathrm{p,t}} = \begin{bmatrix} \mathrm{s}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathrm{s}_{\mathrm{n}} \end{bmatrix}_{\mathrm{nx1}} \tag{3.43}$$

Alfa etkisi ve sürü etkisi denklem 3.14 ve 3.15'e göre heesaplanır.

Yeni kır kurtlarının doğuşu iki bireyin sosyal koşullarının ve çevresel etkilerin bir kombinasyonu nedeniyle sürüleri etkiler. Sosyal populasyon alfa etkisi ve kültürel eğiliminden kaynaklı kaotik olarak güncellenir (Denklem 3.44). $ch1$ ve $ch2$ sıfır ile bir aralığında kaotik olarak üretilmiş sayıdır.

$$\mathrm{yeni_soc}_{\mathrm{c}}^{\mathrm{p,t}} = \mathrm{soc}_{\mathrm{c}}^{\mathrm{p,t}} + \mathrm{ch1} * s_a + ch2 * s_c \tag{3.44}$$

Yeni sosyal duruma göre kır kurtlarının uyumu değerlendirilir (Denklem 3.17).

Yeni toplumsal durum eskisi ile karşılaştırılır ve en iyi uygunluk değerine sahip kır kurtları seçilir (Denklem 3.18). Yeni doğan kır kurtları rastgele seçilmiş iki kır kurdunun sosyal durumlarını, levy uçuş stratejisini ve çevresel etkiyi bütünleştirir. Bu işlem algoritmanın tasarım alanını keşfetmesine izin verir. Ebeveynler tarafından üretilen herhangi bir yeni doğan kır kurdu, tasarım alanındaki Levy uçuşuyla herhangi bir noktaya gitmesine izin verir. Ne kadar iyi kır kurdu yaratılırsa, yeni doğan kır kurtları o kadar iyi olur.

$$\mathrm{rnd_j} = \mathrm{L}(s,\gamma,\mu) = \mathrm{L(i)}$$

$$\mathrm{L(i)} = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left(-\frac{\gamma}{2(s-\mu)}\right) \frac{1}{(s-\mu)^{3}/_{2}}, 0 < \mu < s < \infty \\ 0 \qquad \text{yoksa} \end{cases} \qquad \text{v (3.45)}$$

Kültürel çeşitlilik dağılım olasılığı ve ilişki olasılığı ile değerlendirilir (Denklem 17 ve Denklem 18).

Yeni doğan kır kurtlarının modellenmesi çevresel etkiler de dahil olmak üzere yeni çözümleri temsil eder.

$$\mathrm{pup_j^{p,t}} = \begin{cases} \mathrm{soc_{r1,j}^{p,t}} & \mathrm{rnd_j} < \mathrm{P_s} \quad \mathrm{j} = \mathrm{j_1} \\ \mathrm{soc_{r2,j}^{p,t}} & \mathrm{rnd_j} \geq \mathrm{P_s} + \mathrm{P_a} \quad \text{or } \mathrm{j} = \mathrm{j_2} \\ \mathrm{R_j} & \text{yoksa} \end{cases} \qquad (3.46)$$

Uyum sağlayan kır kurtları populasyon içinde yaşarken uyumu kötü olanlar ölürler. Bununla birlikte, popülasyon büyüklüğünü sabit tutmak için çakalın ölümü ve doğumu arasında bir denge vardır (Denklem 3.22).

Kaotik olarak seçilen Çakallar, çeşitlilik sağlamak ve yerel optimal bölgelere düşmeyi önlemek için rastgele seçilen paketler arasında pozisyon değiştirmiştir (Denklem 3.11, Denklem 3.16).

Kır kurtları doğum ve ölüm sürecini Laplace stratejisiyle gerçekleştirir. En iyi kır kurtları ve rastgele seçilen kır kurtları iki yeni kır kurdu oluşturur. Yeni iki kır kurtları, popülasyondaki en kötü kır kurtları karşılaştırılır. En iyi uyum sağlayan kır kurtları mevcut popülasyondaki en kötü kır kurtları değiştirilir (Denklem 3.50 ve Denklem 3.51).

$$f(x) = \frac{1}{2b}\exp\left(-\frac{|\vec{x}-a|}{b}\right), -\infty < \vec{x} < \infty \tag{3.47}$$

$$F(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{|\vec{x}-a|}{b}\right), & \vec{x} \leq a \\ 1 - \frac{1}{2}\exp\left(\frac{-|\vec{x}-a|}{b}\right), & \vec{x} > a \end{cases} \tag{3.48}$$

$$\beta = \begin{cases} a - b * \log(u), & u \leq 1/2 \\ a + b * \log(u), & u > 1/2 \end{cases} \tag{3.49}$$

$$soc_c^{yeni1} = soc_c^{eski1} + \beta*(soc_c^{eski1} - soc_c^{eski2}) \tag{3.50}$$

$$soc_c^{yeni2} = soc_c^{eski2} + \beta*(soc_c^{eski1} - soc_c^{eski2}) \tag{3.51}$$

En iyi uyarlanmış kır kurdu, soruna geçici bir çözüm olarak seçilir. Kır kurtlarının aranabilirliğini ve kalitesini artırmak için geçici en iyi çözümler etrafında yeni değişkenler yaratılarak yeni çözümler sağlanır. En iyi çözüm etrafında düzensiz olarak oluşturulmuş bir arama yarıçapı (R) oluşturulmuştur. Her yinelemede kaotik bir değişken Z(i) üretilir. Arama uzayındaki yeni kır kurtlarının uyarlanmasının, küresel en iyi uyarlanmış kır kurdundan daha iyi olduğunu varsayalım. Yeni uyarlanan kır kurdu değiştirilir ve sonlandırılana kadar devam eder (Denklem 3.52). Son olarak, çözüm olarak sosyal olarak en iyi uyarlanmış kır kurdu seçilir ve tüm kır kurtlarının yaşı 1 artırıldığında yinelemeye ulaşılır (Denklem 3.53).

$$soc_{c,i}^{yeni2} = soc_{c,i}^{en\ iyi} + r(Ub - Lb)(z(i) - 0.5) \tag{3.52}$$

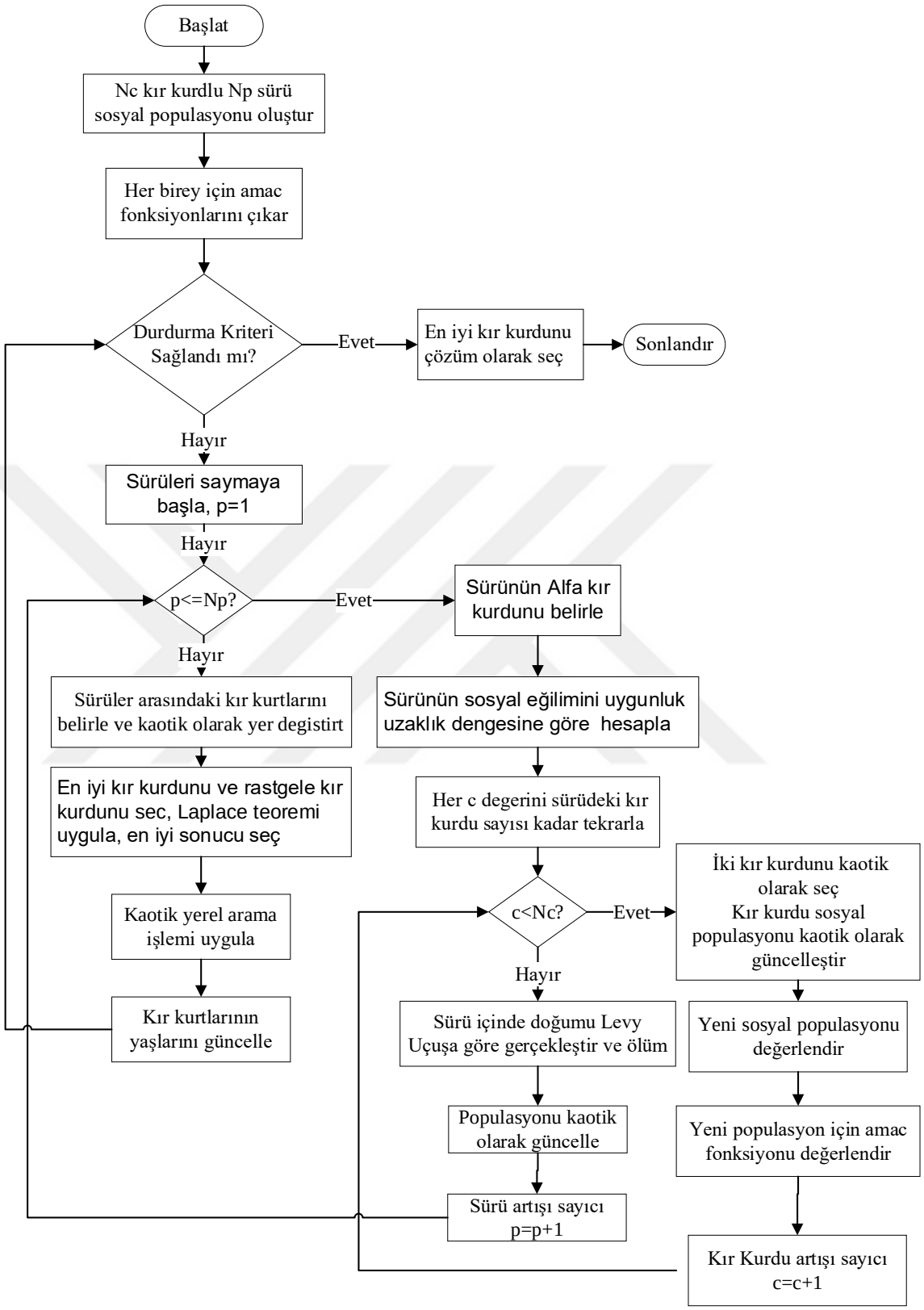
$$yaş_c^{p,t} = yaş_c^{p,t} + 1 \tag{3.53}$$

Geliştirilmiş algoritmanın sözde kodu

Durma kriterini ayarla = M_{itr}
Sürü sayısını = Np ve sürü başına kır kurdu sayısını = Nc ayarla
Deney sayısını = Nexp ve problem boyutunu = D ayarla
Kır kurdu sosyal populasyonunu başlat (Denklem 3.37)
Tüm kır kurtlarının uyumunu değerlendir (Denklem 3.10)
En iyi kır kurtlarını ayarla $x_{\text{en iyi}}$
while (Np*Nc < M_{itr})
  for i = 1 to Np
    Kır kurtlarının uyumuna göre her sürüdeki en iyi kır kurdunu (alfa) seç (Denklem 3.12)
    Uygunluk uzaklık dengesine (FDB) göre sosyal eğilimi hesapla (Denklem 3.43)

    $\text{cult}^{\text{p,t}} = \text{soc}^{\text{p,t}}_{\text{FDB}}$

    for c = 1: Nc
      İki kır kurdunu kaotik olarak seç
      Kır kurtlarının sosyal durumunu kaoitk olarak güncelle (Denklem 3.44)
      Her kır kurdunun uyumunu hesapla (Denklem 3.17)
      Uyumluluğu değerlendir (Denklem 3.18)
    end for
      Yeni kır kurtlarının doğumunu levy uçuşuna göre güncelle (Denklem 3.46)
      Populasyonu kaotik olarak güncelle
  end for
    Sürüler arasında kır kurtlarının pozisyonunu kaotik olarak değiştir.
Uyumluluğu değerlendir
En iyi kır kurdunu güncelle $x_{\text{en iyi}}$
En iyi kır kurdunu x_{eniyit} ve rastgele kır kurdunu $x_{\text{kır kurdu}}$ sec ve Laplace teoremi
uygula (Denklem 3.50, Denklem 3.51)
Eğer daha iyi bir çözüm varsa en iyi kır kurdunu $x_{\text{en iyi}}$ güncelle
Kaotik bölgesel arama işlemini uygula (Denklem 3.52)
Kır kurtlarının yaşını güncelle
end while
En iyi uyum sağlayan kır kurdunu seç



Şekil 3.23. Geliştirilmiş Kır kurdu algoritması akış şeması.

Çizelge 3.3. Kır kurdu optimizasyon algoritması tabanlı oluşturulan algoritmalar

Geliştirilmiş Algoritma No (GCOA)	Algoritma Kısaltması	Algoritma Açıklaması
1	CCOA	Kaotik harita kullanılarak kaotik harita tabanlı yeni bir kır kurdu optimizasyon algoritması
2	CLCCOA	Kaotik tabanlı, kaotik bölgesel arama tabanlı yeni bir kır kurdu optimizasyon algoritması
3	CLCOA	Kaotik bölgesel arama tabanlı yeni bir kır kurdu optimizasyon algoritması
4	COA	Standart Kır Kurdu Optimizasyon Yöntemi
5	FDBLFCOA	Uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
6	LCLFDBLFCCOA	Kaotik tabanlı, laplace dağılımlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
7	LCOA	Laplace dağılımı tabanlı yeni bir kır kurdu optimizasyon algoritması
8	LFDBLFCCOA	Laplace dağılımlı uygunluk uzaklık dengeli, levy flight, kaotik yeni bir kır kurdu optimizasyon algoritması
9	LFDBLFCOA	Laplace dağılımlı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
10	LOCCOA	Laplace dağılımlı, karşıt tabanlı, kaotik yeni bir kır kurdu optimizasyon algoritması
11	LOCLFDBLFCCOA	Kaotik tabanlı, Laplace dağılımlı, karşıt tabanlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
12	LOCOA	Laplace dağılımlı, karşıt tabanlı yeni bir kır kurdu optimizasyon algoritması
13	LOFDBLFCCOA	Laplace dağılımlı karşıt tabanlı uygunluk uzaklık dengeli, levy flight, kaotik yeni bir kır kurdu optimizasyon algoritması
14	LOFDBLFCOA	Laplace dağılımlı, karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
15	LOLFCCOA	Laplace dağılımlı, karşıt tabanlı, levy uçuş tabanlı kaotik yeni bir kır kurdu optimizasyon algoritması
16	OCCLCOA	Karşıt tabanlı, kaotik bölgesel arama tabanlı yeni bir kır kurdu optimizasyon algoritması
17	OCCOA	Karşıt tabanlı kaotik yeni bir kır kurdu optimizasyon algoritması
18	OCLFDBLFCCOA	Kaotik tabanlı, karşıt tabanlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
19	OCLLFCCOA	Kaotik tabanlı, karşıt tabanlı, kaotik bölgesel aramalı, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
20	OCOA	Karşıt tabanlı yeni bir kır kurdu optimizasyon algoritması
21	OFDBLFCCOA	Karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuş tabanlı Kaotik yeni bir kır kurdu optimizasyon algoritması
22	OFDBLFCOA	Karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuş tabanlı yeni bir kır kurdu optimizasyon algoritması
23	OLFCCOA	Karşıt tabanlı, levy uçuş tabanlı kaotik yeni bir kır kurdu optimizasyon algoritması

3.4. Geliştirilen Yeni Kır Kurdu Algoritmasının Testleri

Yirmi üç kıyaslama testi işlevi, geliştirilmiş COA algoritmasının performansını ortalama, en kötü, en iyi, medyan ve standart sapma kapsamında değerlendirir. Kıyaslama fonksiyonlarının testi MATLAB 2021b yazılımında yapılmıştır. Önerilen algoritmayı karşılaştırmak ve analiz etmek için otuz üç deney ve beş yüz durdurma kriteri alınmıştır. Benzer desteklerde, sonuçlar istatistiksel Wilcoxon toplam testi dikkate alınarak değerlendirilir. Algoritma kısıtsız problemler ve kısıtlı mühendislik problemleri olmak üzere iki farklı şekilde test edilmiştir. Optimizasyon algoritmalarını test etmek ve performanslarını ölçmek için literatürde bulunan tek modlu (tek bir en iyi sonuca sahip) ve çok modlu (bölgesel en iyi sonuçlara sahip ancak tek bir en iyi sonuca sahip)

matematiksel fonksiyonlar kullanılır. Bu fonksiyonların zorlukları parametrelerine bağlıdır. Oluşturulan bu fonksiyonlar çoğu mühendislik problemlerine benzer bu yüzden sıklıkla algoritmaların yeteneklerini test etmede yaygın bir şekilde kullanılır. Önerilen yöntemlerin performans ölçümleri otuz üç farklı fonksiyon için geliştirilmiş COA algoritmasının değerlendirilmesinde kullanıldı. Yirmi üç klasik kısıtsız fonksiyon F1-F23 literatürden alınmıştır. Standart kıyaslama işlevleri, parametre alanı içinde objektif uygunluğu bulmak için tanımlanmış optimal değer (fmin), boyut(n) ve arama aralığından (lb, ub) oluşur. Önerilen algoritmanın parametre ayarı Çizelge 3.4'te gösterilmektedir. Bu yirmi üç klasik problem değişen değişkenli tek modlu fonksiyonlar, değişken ve sabit değişkenli çok modlu fonksiyonlar olarak sınıflandırılır. F1'den F7'ye tek modlu kıyaslama testi fonksiyonları Çizelge 3.5 'te gösterilmektedir. Şekil 4.9 - Şekil 4.15, COA ve GCOA için tek modlu kıyaslama fonksiyonlarının yakınsama eğrisini göstermektedir. Benzer şekilde, çok modlu (MM) (F8 ila F13) ve sabit boyutlu (FD) (F14 ila F23) kıyaslama işlevleri Çizelge 3.6 ve Çizelge 3.7 'de gösterilmektedir. Şekil 4.33- Şekil 4.38 ve Şekil 4.58- Şekil 4.67, çok modlu ve sabit boyutlu kıyaslama fonksiyonları için yakınsama eğrisini göstermektedir. Arama yeteneği COA ve GCOA arasında kolayca karşılaştırılabilir. GCOA, COA'ya karşı üstün özellik gösterir. Tek modlu işlevler, çok modlu işlevler keşif yeteneğini değerlendirerek erken yakınsamayı önleyen yerel optimum noktaları test eder, sömürü yeteneğini değerlendirmek için algoritmanın yakınsama oranını tek bir küresel optimum ile test eder. Yirmi üç klasik kıyaslama fonksiyonunun detaylı bilgileri Çizelge 4.1'dedir. Yeni önerilen algoritmaların testi on adet literatürde bulunan mühendislik problemine ve iki adet sanayi problemine uygulanmış olup doğrulanmıştır.

Çizelge 3.4. Algoritma performans testleri için parametre ayarı

Parametre Ayarı	Önerilen Algoritma
Sürü Sayısı	20
Kır Kurdu Sayısı	5
İterasyon sayısı	500
Deney Sayısı	30

Çizelge 3.5. Çalışmalarda kullanılan kısıtsız tek modlu kıyaslama fonksiyonları

No	Fonksiyonel Denklem	n	Arama aralığı	f_{min}
F1	$\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
F2	$\sum_{i=1}^{n}\|x_i\| + \prod_{i=!}^{n}\|x_i\|$	30	[-10,10]	0
F3	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100,100]	0
F4	$\text{maks}_i\{\|x_i\|, 1 \le i \le n\}$	30	[-100,100]	0
F5	$\sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[-30,30]	0
F6	$\sum_{i=1}^{n}([x_i + 0.5])^2$	30	[-100,100]	0
F7	$\sum_{i=1}^{n} i x_i^4 + \text{random}[0,1]$	30	[-1.28,1.28]	0

Çizelge 3.6. Sabit çok modlu test fonksiyonu

No	Fonksiyonel Denklem	n	Arama aralığı	f_{min}
F8	$\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\|x_i\|}\right)$	30	[-500,500]	-418,98295
F9	$\sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
F10	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[-32,32]	0
F11	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[-600,600]	0
F12	$\frac{\pi}{n}\{\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + ..$ $+..\sum_{i=1}^{n} u(x_i, 10,100,4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
F13	$0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i, 5,100,4)$	30	[-50,50]	0

Çizelge 3.7. Sabit boyutlu test fonksiyonu

No	Fonksiyonel Denklem	n	Arama aralığı	f_{min}
F14	$\left(\frac{1}{500}+\sum_{i=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[-65.536,65.536]	1
F15	$\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2b_ix_3+x_4}\right]^2$	4	[-5,5]	0,00030
F16	$4x_i^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5,5]	-1,0316
F17	$\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1\frac{1}{8\pi}\right)\cos(x_1)+10$	2	Alt : [-5,5]	0,398
F18	$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $x[30+(2x_1-3x_2)^2x(18-32x_1+12x_1^2+48x_2-36x_1x_227x_1^2)]$	2	[-2,2]	3
F19	$-\sum_{i=1}^{4}c_i\exp\left(-\sum_{i=1}^{3}aij(x_j-p_{ij})^2\right)$	3	[1,3]	-3,32
F20	$-\sum_{i=1}^{4}c_i\exp\left(-\sum_{i=1}^{6}aij(x_j-p_{ij})^2\right)$	6	[0,1]	-3,32
F21	$-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10,1532
F22	$-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10,4028
F23	$-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	4	[0,10]	-10,5363

3.4.1. Kısıtlı mühendislik tasarım problemlerinde performans analizi

Günlük problemlerin çoğu eşitlik / eşitsizlik kısıtlamaları içerdiğinden önerilen yeni algoritmanın gerçek problemler üzerindeki performansını ölçmek için literatürde sıklıkla kullanılan mühendislik problemleri algoritmanın performans test aracı olarak kullanıldı. Kısıtlı problemlerin çözümünde literatürde farklı yaklaşımlar bulunmaktadır ancak ceza metodu uygulama kolaylığı ve basitliği sebebiyle genel olarak tercih edilmektedir. Ceza metodu ceza fonksiyonu aracılığıyla eşitlik ve eşitsizlik kısıtlarını problem uyum fonksiyonuna indirger ve kısıtlı arama uzayını kısıtsız arama uzayı ile eşleştirir (Denklem 3.54).

$$\min\prod(X)=f(X)+\sum_{i=1}^{n}u_i{g_i}^2(X)+\sum_{j=1}^{m}v_j{p_j}^2(X) \tag{3.54}$$

n: eşitsizlik kısıtlarının sayısı
m: eşitlik kısıtlarının sayısı
X: Çözüm vektörü
g, p: Uyum değeri

u büyük sıfır, v büyük sıfır uygunluk fonksiyonuna kısıtlamaların dahil edilmesi için kullanılan büyüklüklerdir. Bu büyüklüklerin büyük bir değer olması arama uzayını artırdığı için uygun çözümün bulunmasını engelleyebilir, küçük bir değer olması problem uyumunu sağlamadığı için kısıtların ihlaline yol açabilir. Bu yüzden bu büyüklükler iyi bir şekilde ayarlanmalıdır. Ceza fonksiyonun uyum fonksiyonuna eşit olduğu çözüm uygun çözümdür.

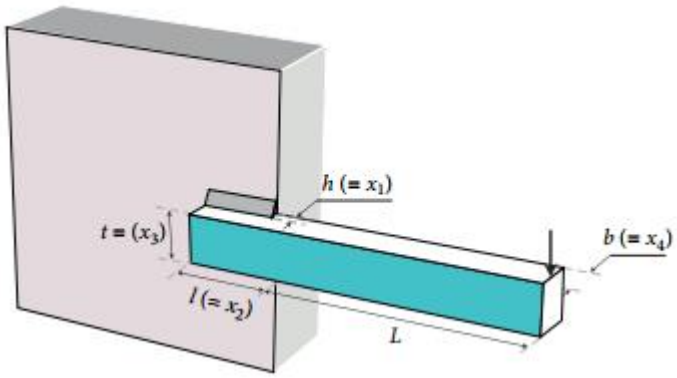
Kısıtlı problemler altındaki davranışı görmek için ceza metodu kullanılarak literatürden kaynaklı kiriş tasarım problemi, yay tasarım problemi, basınçlı kap problemi, hız düşürücü problemi, dişli güç iletim problemi, üç çubuk kafes yapı tasarım problemi, konsol kiriş tasarım problemi, çoklu disk kavramalı fren (mdcb) tasarımının optimizasyon problemi, himmelblau nonlinear problemi ve küresel rulman tasarım problemi olmak üzere on adet farklı probleme önerilen algoritma uygulanmıştır. Belirtilen mühendislik tasarım problemleri (EDP), EDP olarak kısaltılır. Çizelge 3.8'de on gerçek dünya tasarım problemi gösterilmiştir. En iyi, ortalama, en kötü ve standart sapma değerlerinin karşılaştırması Çizelge 4.13'te gösterilmiştir. Önerilen algoritma, performansı otuz deney ve beş yüz durdurma kriteri ile test eder. Ayrıca önerilen algoritma, mevcut optimizasyon yöntemlerinin sonuçları ile en kötü, ortalama, en iyi ve standart sapma değerleri kapsamında karşılaştırılmıştır. Şekil 4.83-Şekil 4.87, algoritmanın eniyiliğini doğrulamak için on mühendislik problemi için otuz deneysel sonuca bağlı mevcut kır kurdu algoritmasına göre yakınsama eğrisini temsil etmektedir.

Çizelge 3.8. Mühendislik tasarım problem verileri

Mühendislik Tasarım Problemi (MTP)	Problem	Değişkenler	Kısıtlar
MTP1	Kaynaklı Kiriş Tasarımı	4	7
MTP2	Bası Yayı Tasarımı	3	4
MTP3	Basınçlı Kap Tasarımı	4	4
MTP4	Hız Düşürücü Tasarımı	7	11
MTP5	Dişli Güç İletim Tasarımı	4	1
MTP6	Üç Çubuk Kafes Yapı Tasarımı	2	3
MTP7	Konsol Kiriş Tasarımı	5	1
MTP8	Çoklu Disk Kavramalı Fren Tasarımı	5	8
MTP9	Himmelblau's Nonlinear Problemi	5	3
MTP10	Küresel Rulman Tasarımı	10	9

3.4.1.1. Kaynaklı kiriş tasarım problemi (mtp1)

Kaynaklı kiriş tasarım probleminin ana odak noktası (bkz. Şekil 3.24) üretim maliyetlerini azaltmak iken, aynı zamanda minimum üretim maliyeti de hedeflenmiştir. Kiriş yüksekliği (x_3), kaynak genişliği (x_1), kiriş genişliği (x_4) ve kaynak uzaklığı (x_2) olmak üzere yedi kısıt fonksiyonuna tabi dört tasarım değişkeni vardır. Değişken ve kısıtlamalar aşağıdaki matematiksel denklemler cinsinden formüle edilir. Kaynaklı kiriş tasarımı için diğer algoritmalarla karşılaştırmalı sonuçlar Çizelge 4.15.1'de verilmiştir. Önerilen algoritma, analiz sonuçlarına kıyasla diğer algoritmalardan üstündür.



Şekil 3.24. Kaynaklı kiriş tasarımı.

Parametreler, $X = [x_1, x_2, x_3, x_4]$ = [kaynak genişliği, kaynak uzaklığı, kiriş yüksekliği, kiriş genişliği]

Değişken aralığı;

$X = (x_1, x_2, x_3, x_4)^T$;

$0.1 \leq x_1; x_4 \leq 2; \quad 0.1 \leq x_2 \leq 10;$

$0.3 \leq x_3 \leq 10;$

$\Delta = 6\,000\ lb;\ L = 14\ inch;\ E = 30\ x\ 10^6\ psi;\ G = 12\ x\ 10^6$ psi;

$\alpha_{max} = 13\,600\ psi;\ \beta_{max} = 30000\ psi;\ \vartheta_{max} = 0.25\ inch;$

Kısıtlar;

$$h_1(X) = \alpha(X) - \alpha_{max} \leq 0\ ;\ \text{Kesme gerilmesi (}\alpha\text{)}\ ; \tag{3.55}$$

$$h_2(X) = \beta(X) - \beta_{max} \leq 0\ ;\ \text{Eğilme gerilmesi (}\beta\text{)}; \tag{3.56}$$

$$h_3(X) = \vartheta(X) - \vartheta_{max} \leq 0\ ;\ \text{Kiriş üzerindeki eğim (}\vartheta\text{)}; \tag{3.57}$$

$$h_4(X) = x_1 - x_4 \leq 0; \tag{3.58}$$

$$h_5(X) = \Delta - \Delta_c\,(X) \leq 0\ ;\ \text{Burkulma yükü (}\Delta_c\text{)}; \tag{3.59}$$

$$h_6(X) = 0{,}125 - x_1 \leq 0; \tag{3.60}$$

$$h_7(X) = 0{,}10471x_1^2 + 0{,}04811x_3x_4(14 + x_2)\text{-}5 \leq 0\ ; \tag{3.61}$$

$$\alpha(Z) = \sqrt{\alpha'^2 + \frac{2\alpha'\alpha''x_2}{2R} + \alpha''^2}\ ; \tag{3.62}$$

$$\alpha' = \frac{\Delta}{\sqrt{2}x_1x_2}\ ; \tag{3.63}$$

$$\alpha'' = \frac{MR}{J}\ ; \tag{3.64}$$

$$M = \Delta\left(14 + \frac{x_2}{2}\right); \tag{3.65}$$

$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}; \tag{3.66}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{{x_2}^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\} \tag{3.67}$$

$$\beta(Z) = \frac{6\Delta L}{x_4\,x_3^2}\ ; \tag{3.68}$$

$$\vartheta(Z) = \frac{6\Delta L^3}{Ex_4\,x_3^2}\ ; \tag{3.69}$$

Amaç fonksiyon: $f(Z) = 1{,}10471x_2x_1^2 + 0{,}04811x_3x_4(14 + x_2)$; (3.70)

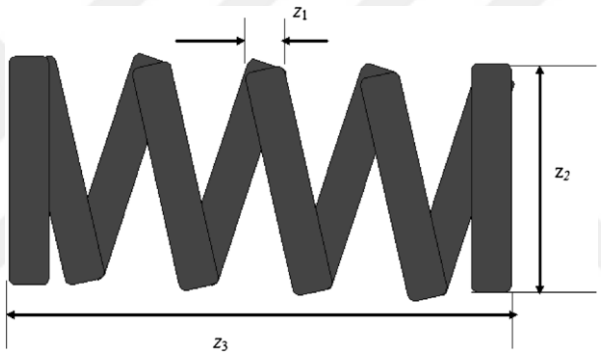
$$\Delta_c = \frac{4{,}013\ x\ E}{L^2}\sqrt{\frac{{x_3}^2{x_4}^6}{36}}\,x\left[1 - \left(\frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\right]; \tag{3.71}$$

$$\Delta = 6\,000 \text{ lb}; L = 14 \text{ inch}; E = 30 \times 10^6 \text{ psi}; G = 12 \times 10^6 \text{ psi}$$

$$\alpha_{max} = 13\,600 \text{ psi}; \beta_{max} = 30\,000 \text{ psi}; \vartheta_{max} = 0{,}25 \text{ inch}$$

3.4.1.2. Bası yayı tasarım problemi (mtp2)

Bası yayı tasarım probleminin ana odak noktası (bkz. Şekil 3.25), bası yayının ağırlığını minimize etmektir. Dört kısıt fonksiyonuna bağlı tel çapı (z_1), yay çapı (z_2), ve yay uzunluğu (z_3) olarak üç tasarım değişkeni vardır: Sıkışma, frekans, kayma gerilimi ve dış çap sınırları vardır. Matematiksel olarak formüle edilmiş bası yayı tasarım problemi aşağıdaki denklemlerde temsil edilmektedir. Önerilen yöntem, Çizelge 4.16.1'deki diğer algoritmalardan daha iyi performans özelliklerine sahiptir.



Şekil 3.25. Yay tasarım problemi.

Amaç fonksiyon: Asgari $f(Z) = (z_3 + 2)\, z_2 z_1^2$ (3.72)

Kısıtlar:

$$h_1(Z) = 1 - \frac{z_2^3 z_3}{71\,785\, z_1^4} \leq 0; \quad (3.73)$$

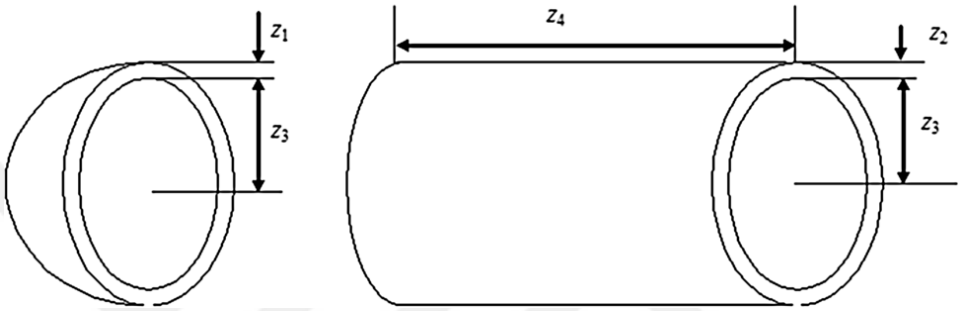
$$h_2(Z) = \frac{4z_2^2 - z_1 z_2}{12\,566(z_2 z_1^3 - z_1^4)} + \frac{1}{5\,108 z_1^2} - 1 \leq 0; \quad (3.74)$$

$$h_3(Z) = 1 - \frac{140{,}45 z_1}{z_2^2 z_3} \leq 0; \quad (3.75)$$

$$h_4(Z) = \frac{z_2 + z_1}{1{,}5} - 1 \leq 0; \quad (3.76)$$

3.4.1.3. Basınçlı kap problemi (mtp3)

Basınçlı kap tasarım problemi (bkz. Şekil 3.26), malzeme, şekillendirme ve kaynak maliyeti dahil olmak üzere bir basınçlı kabın üretim maliyetini en aza indirmektir. Kapak kalınlığı (z_1), gövde sacı kalınlığı (z_2), yarıçap (z_3), ve silindirin uzunluğu (z_4) olmak üzere dört kısıtlama fonksiyonuna (yer değiştirme, frekans, kesme gerilimi ve dış çap limitleri) tabi tutulmuş dört tasarım değişkeni vardır. Matematiksel formül aşağıdadır. Çizelge 4.17.1, diğer algoritmalara kıyasla önerilen COA'nın sonucunu gösterir.



Şekil 3.26. Basınçlı kap tasarım problemi.

Amaç fonksiyon; asgari f(Z)

$$f(Z) = 0{,}6224z_1z_3z_4 + 1{,}7781z_2z_3^2 + 3{,}1661{z_1}^2z_4 + 19{,}84{z_1}^2z_3; \quad (3.77)$$

Kısıtlar;

$$h_1(Z) = -z_1 + 0{,}0193 * z_3 \leq 0; \quad (3.78)$$

$$h_2(Z) = -z_2 + 0{,}00954 * z_3 \leq 0; \quad (3.79)$$

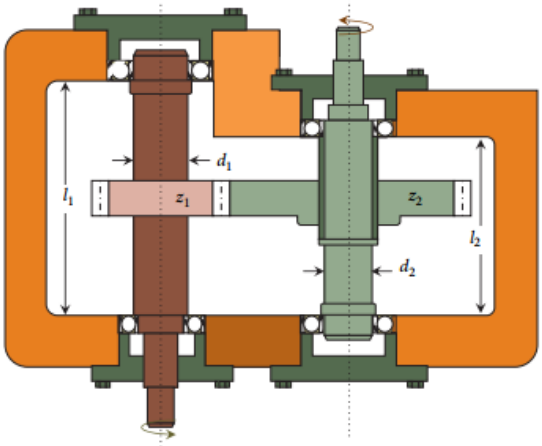
$$h_3(Z) = -\pi z_3^2 z_4^2 - \frac{4}{3}\pi z_3^3 + 1\,296\,000 \leq 0; \quad (3.80)$$

$$h_4(Z) = z_4 - 240 \leq 0; \quad (3.81)$$

$0 \leq z_1 \leq 99;$ $0 \leq z_2 \leq 99;$ $10 \leq z_3; z_4 \leq 200;$

3.4.1.4. Hız düşürücü problemi (mtp4)

Dişli kutusu tasarımlarında kullanılan hız düşürücü tasarım probleminin ana odak noktası (bkz. Şekil 3.27) dişli dişleri üzerindeki gerilmeler, dişli yüzeyi, miller ve millerin enine uzamasına bağlı hız düşürücünün ağırlığını en aza indirmektir. Yüzey kaydırma (z_1), diş modülü (z_2), diş sayısı (z_3), birinci mil uzunluğu (z_4), ikinci mil uzunluğu (z_5), birinci mil çapı (z_6), ikinci mil çapı (z_7) gibi dört kısıt fonksiyonuna bağlı yedi tasarım değişkeni vardır. Matematiksel formül aşağıdadır. Önerilen GCOA, Çizelge 4.18.1'deki diğer optimizasyon algoritmalarına kıyasla rekabetçi ve etkilidir.



Şekil 3.27. Hız düşürücü tasarım problemi.

Amaç fonksiyon; asgari f(Z);

$$f(Z) = 0{,}7854z_1z_2^2(3{,}3333z_3^2 + 14{,}9334z_3 - 43{,}0934) - 1{,}508z_1(z_6^2 + z_7^2) + 7{,}4777(z_6^3 + z_7^3) + 0{,}7854(z_4z_6^2 + z_5z_7^2) \quad (3.82)$$

Kısıtlar;

$$h_1(Z) = \frac{27}{z_1z_2^2z_3} - 1 \leq 0\,; \quad (3.83)$$

$$h_2(Z) = \frac{397{,}5}{z_1z_2^2z_3^2} - 1 \leq 0\,; \quad (3.84)$$

$$h_3(Z) = \frac{1{,}93z_4^3}{z_2z_3z_6^4} - 1 \leq 0; \quad (3.85)$$

$$h_4(Z) = \frac{1{,}93z_5^3}{z_2 z_3 z_7^4} - 1 \leq 0; \quad (3.86)$$

$$h_5(Z) = \frac{\sqrt{\left(\frac{745\, z_4}{z_2 z_3}\right)^2 + 16{,}9 \times 10^6}}{110 z_6^3} - 1 \leq 0; \quad (3.87)$$

$$h_6(Z) = \frac{\sqrt{\left(\frac{745\, z_5}{z_2 z_3}\right)^2 + 157{,}5 \times 10^6}}{85\, z_7^3} - 1 \leq 0; \quad (3.88)$$

$$h_7(Z) = \frac{z_2 z_3}{40} - 1 \leq 0; \quad (3.89)$$

$$h_8(Z) = \frac{5z_2}{z_1} - 1 \leq 0; \quad (3.90)$$

$$h_9(Z) = \frac{z_1}{12 z_2} - 1 \leq 0; \quad (3.91)$$

$$h_{10}(Z) = \frac{1{,}5z_6 + 1{,}9}{z_4} - 1 \leq 0; \quad (3.92)$$

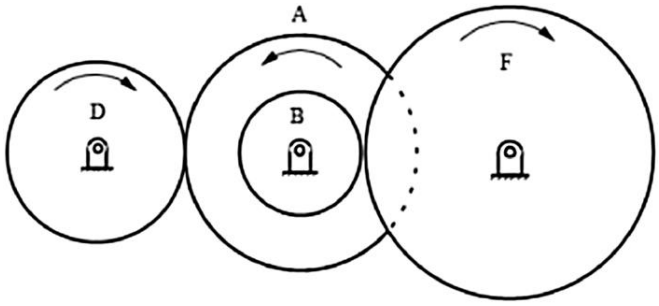
$$h_{11}(Z) = \frac{1{,}1z_7 + 1{,}9}{z_5} - 1 \leq 0; \quad (3.93)$$

$2{,}6 \leq z_1 \leq 3{,}6; \quad 0{,}7 \leq z_2 \leq 0{,}8; \quad z_3 \in \{17,18,19,\ldots,28\};$

$7{,}3 \leq z_4; \quad z_5 \leq 8{,}3; \quad 2{,}9 \leq z_6 \leq 3{,}9; \quad 5 \leq z_7 \leq 5{,}5;$

3.4.1.5. Dişli güç iletim problemi (mtp5)

Dişli takımı tasarım probleminin amacı (bkz. Şekil 3.28) optimum dişli oranı ile asgari maliyeti bulmaktır. Her dişli (z_1), (z_2), (z_3), (z_4) için diş sayısı ile ilgili dört tasarım değişkeni vardır. Matematiksel formül aşağıdadır. Çizelge 4.19.1'deki karşılaştırma sonuçları, önerilen GCOA'nın diğer algoritmalardan daha iyi sonuçlara sahip olduğunu ortaya koymaktadır.



Şekil 3.28. Dişli güç iletim tasarım problemi.

Amaç fonksiyon; asgari f(Z);

$$f(Z) = \left(\frac{1}{6{,}931} - \frac{z_2 z_3}{z_1 z_4}\right);^2 \quad (3.94)$$

Kısıt fonksiyonu:

$$12{,}0 \leq z_{1.,2.,3.,4} \leq 60{,}0; \quad (3.95)$$

3.4.1.6. Üç çubuk kafes yapı tasarım problemi (mtp6)

Üç çubuk kafes yapı tasarım probleminin ana odak noktası (bkz. Şekil 3.29) kafes kirişin ağırlığını azaltmaktır. Tepki kuvvetleri (P_1) ve (P_2) olmak üzere iki tasarım değişkeni, güçlendirme genişliği (z_1) ve güçlendirme aralığı (z_2) olmak üzere iki parametre ile eğilme, sehim ve gerilme olmak üzere üç kısıt fonksiyonuna tabi tutulur. Matematiksel denklemler aşağıdaki gibi formüle edilir. Sonucun analizi, önerilen yöntemin Çizelge 4.20.1'deki maliyet minimizasyonu hedefini geliştirdiğini ortaya koymaktadır.

Amaç fonksiyon; asgari f(Z);

$$f(Z) = \left(2\sqrt{2z_1} + z_2\right) * L; \quad (3.96)$$

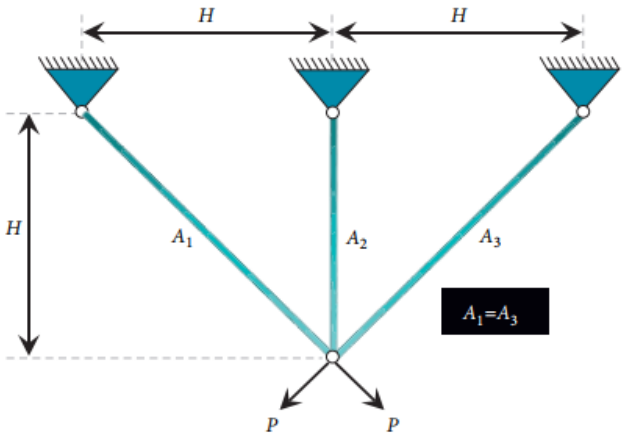
Kısıt fonksiyonu:

$$h_1(Z) = \frac{\sqrt{2}z_1 + z_2}{\sqrt{2}{z_1}^2 + 2z_1 z_2} P - \sigma \leq 0; \quad (3.97)$$

$$h_2(Z) = \frac{z_2}{\sqrt{2}{z_1}^2 + 2z_1 z_2} P - \sigma \leq 0; \quad (3.98)$$

$$h_3(Z) = \frac{1}{z_1 + \sqrt{2}z_2} P - \sigma \leq 0; \quad (3.99)$$

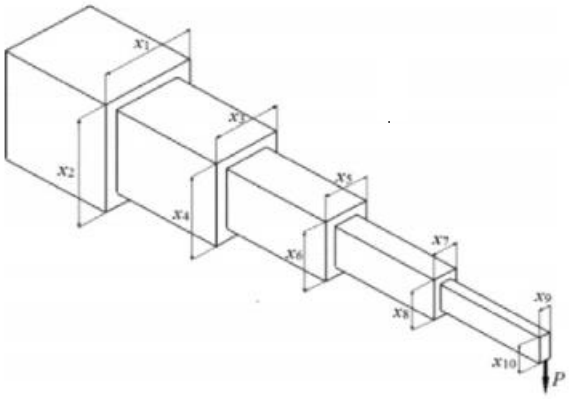
$0 \leq z_{1;}\, z_2 \leq 2$; L=100 cm, P = 2 KN / cm^3 ve σ = 2 KN / cm^3



Şekil 3.29. Üç çubuk kafes yapı tasarım problemi.

3.4.1.7. Konsol kiriş tasarım problemi (mtp7)

Konsol kiriş tasarım problemi (bkz. Şekil 3.30) her dikdörtgen profilin yüksekliğine ve genişliğine bağlı olarak hacmi azaltmayı amaçlamaktadır. $x_1 \ldots x_{10}$ olmak üzere on tasarım değişkeni vardır. Matematiksel formül aşağıdadır. Önerilen GCOA, Çizelge 4.21.1'deki diğer algoritmalarla karşılaştırıldığında rekabetçidir.



Şekil 3.30. Konsol kiriş tasarım problemi.

Amaç fonksiyon; asgari f(X);

$$f(X) = l(x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10}); \quad (3.100)$$

Kısıtlar;

$$h_1(X) = \frac{6\,Pl}{x_9 x_{10}^2} - \alpha_k \leq 0; \quad (3.101)$$

$$h_2(X) = \frac{6\,P(2\,l)}{x_7 x_8^2} - \alpha_k \leq 0; \quad (3.102)$$

$$h_3(X) = \frac{6\,P(3\,l)}{x_5 x_6^2} - \alpha_k \leq 0; \quad (3.103)$$

$$h_4(X) = \frac{6\,P(4\,l)}{x_3 x_4^2} - \alpha_k \leq 0; \quad (3.104)$$

$$h_5(X) = \frac{6\,P(5\,l)}{x_1 x_2^2} - \alpha_k \leq 0; \quad (3.105)$$

$$h_6(X) = \frac{Pl^3}{E}\left(\frac{244}{x_1 x_2^3} + \frac{148}{x_3 x_4^3} + \frac{76}{x_5 x_6^3} + \frac{28}{x_7 x_8^3} + \frac{4}{x_9 x_{10}^3}\right) - \vartheta_{max} \leq 0; \quad (3.106)$$

$$h_7(X) = \left(\frac{x_2}{x_1}\right) - 20 \leq 0; \quad (3.107)$$

$$h_8(X) = \left(\frac{x_4}{x_3}\right) - 20 \leq 0; \quad (3.108)$$

$$h_9(X) = \left(\frac{x_6}{x_5}\right) - 20 \leq 0; \quad (3.109)$$

$$h_{10}(X) = \left(\frac{x_8}{x_7}\right) - 20 \leq 0; \quad (3.110)$$

$$h_{11}(X) = \left(\frac{x_{10}}{x_9}\right) - 20 \leq 0; \quad (3.111)$$

$1 \leq x_1 \leq 5$; $30 \leq x_2 \leq 65$; $2{,}4 \leq x_3$; $x_5 \leq 3{,}1$;

$45 \leq x_4$; $x_6 \leq 60$; $1 \leq x_7$; $x_9 \leq 5$; $30 \leq x_8$; $x_{10} \leq 65$;

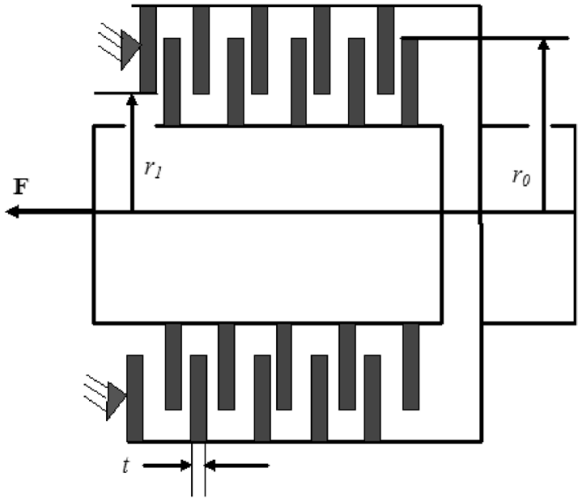
Her kiriş boyu l=100cm, Toplam kiriş boyu L = 500 cm,

P = 50 000 N ve α_k = 14 000 N / cm^2

$\vartheta_{max} = 2{,}7$; E =2x10^7 ;

3.4.1.8. Çoklu disk kavramalı fren tasarımının optimizasyon problemi (mtp8)

Çoklu Disk Kavramalı Fren Tasarımı (bkz. Şekil 3.31), disk kalınlığı (t), harekete geçiren kuvvet (F), iç & dış yarıçap (r_i) & (r_o), sürtünme yüzeyi (Z) olmak üzere beş ayrı değişkene ve sekiz kısıtlama fonksiyonuna bağlı asgari çoklu disk kavramalı fren kütlesini sağlayan en uygun parametreleri bulmayı amaçlar. Önerilen GCOA, Çizelge 4.22.1'deki diğer algoritmalara kıyasla optimal çözümler üzerinde dikkate değer bir etkiye sahiptir.



Şekil 3.31. Çoklu Disk Kavramalı Fren (MDCB) tasarımı problemi.

Δr = 20 mm ; $I_z = 55$ kgmm^2 ; P_{max} = 1 Mpa; F_{max} = 1 000 N; T_{max} =15 s;

μ = 0,5; s = 1,5; M_s = 40 Nm; M_f = 3 Nm; n=250 rpm; v_{srmax} = 10 m/s; l_{max} = 30mm;

$r_i = 60;\ 61; 62;\ 63 \ldots; 80;$

$r_o = 90;\ 91;\ 92;\ 93 \ldots; 110;$

$t = 1;\ 1{,}5; 2{,}0;\ 2{,}5 \ldots; 3;$

$F = 600;\ 610;\ 620;\ 630 \ldots; 1\,000;$

$Z = 2\ ;\ 3;\ 4;\ 5 \ldots; 9;$

Amaç fonksiyon: asgari f(min);

$$f(\min) = \pi({r_0}^2 - {r_i}^2) * t * (Z+1)\rho; \tag{3.112}$$

Kısıtlar;

$$h_1 = r_0 - r_i - \Delta r \geq 0; \tag{3.113}$$

$$h_2 = l_{max} - (Z+1)(t+\delta) \geq 0 \tag{3.114}$$

$$h_3 = P_{max} - P_{rz} \geq 0; \tag{3.115}$$

$$h_4 = P_{max}\vartheta_{sr\,max} - P_{rz}\vartheta_{sr} \geq 0 \tag{3.116}$$

$$h_5 = \vartheta_{sr\,max} - \vartheta_{sr} \geq 0; \tag{3.117}$$

$$h_6 = T_{max} - T \geq 0; \tag{3.118}$$

$$h_7 = M_h - sM_s \geq 0; \tag{3.119}$$

$$h_8 = T \geq 0; \quad (3.120)$$

$$M_h = \frac{2}{3}\mu F Z \frac{r_o^3 - r_i^2}{r_o^2 - r_i^3}; \quad (3.121)$$

$$P_{rz} = \frac{F}{\pi (r_o^2 - r_i^2)}; \quad (3.122)$$

$$v_{rz} = \frac{2\pi n (r_o^3 - r_i^3)}{90(r_o^2 - r_i^3)}; \quad (3.123)$$

$$T = \frac{I_z \pi n}{30(M_h - M_f)}; \quad (3.124)$$

3.4.1.9. Himmelblau nonlinear problemi (mtp9)

Himmelblau'nun doğrusal olmayan problemi optimizasyon algoritmalarının performansını ölçmek için kullanılır. Matematiksel olarak aşağıda olan bir amaç fonksiyonunu en aza indirmek için beş tasarım parametresi içerir. Önerilen GCOA algoritması, Çizelge 4.23.1'deki diğer algoritmalardan daha iyi sonuçlar göstermektedir.

$0 \leq h_1(X) \leq 92;$
$90 \leq h_2(X) \leq 110;$
$20 \leq h_3(X) \leq 25;$
$78 \leq x_1 \leq 102;\ 33 \leq x_2 \leq 45;$
$27 \leq x_3;\ x_4;\ x_5 \leq 45;$

Amaç fonksiyon: asgari f(X);

$$f(X) = 5{,}3578547x_3^2 + 0{,}8356891x_1x_5 + 37{,}293239x_1 - 40\,792{,}141; \quad (3.125)$$

Kısıtlar;

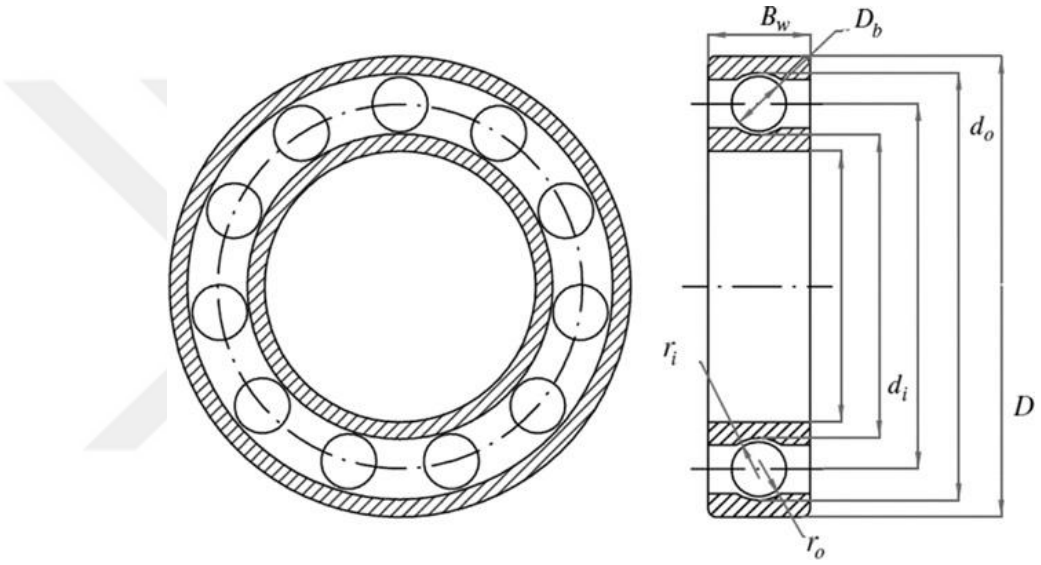
$$h_1(X) = 85{,}334407 + 0{,}0056858x_2x_5 + 0{,}0006262x_1x_4 - 0{,}0022053x_3x_5; \quad (3.126)$$

$$h_2(X) = 80{,}51249 + 0{,}0071317x_2x_5 + 0{,}0029955x_1x_2 - 0{,}0021813x_3^3; \quad (3.127)$$

$$h_3(X) = 9{,}300961 + 0{,}0047026x_3x_5 + 0{,}0012547x_1x_3 - 0{,}0019085x_3x_4; \quad (3.128)$$

3.4.1.10. Küresel rulman tasarım problemi (mtp10)

Rolling Element rulman tasarımı (bkz. Şekil 3.32), dokuz kısıtlama koşulu göz önünde bulundurularak maksimum dinamik yük taşıma kapasitesini bulmak için on değişken içerir. Bunlar hatve çapı (D_m), bilye çapı (D_b), bilye sayısı (Z), iç kanal eğrilik katsayıları (f_i), dış kanal eğrilik katsayıları (f_o), K_{Dmin}, K_{Dmax}, ε, e, ve ζ. Problem aşağıdaki denklemlerle formüle edilmiştir. Önerilen GCOA, Çizelge 4.24.1'deki diğer algoritmalardan üstündür.



Şekil 3.32. Küresel rulman tasarım problemi.

D=160, d=90, $B_w = 30$, $r_i = r_0$ = 11,033

$0{,}5(D+d) \leq D_m \leq 0{,}6(D+d)$;

$0{,}15(D-d) \leq D_b \leq 0{,}45(D-d)$;

$\gamma = \frac{D_b}{D_m}$, $f_i = \frac{r_i}{D_b}$, $f_0 = \frac{r_0}{D_b}$, $T=D-d-2D_b$

$4 \leq Z \leq 50$; $0{,}515 \leq f_i$ ve $f_o \leq 0{,}6$;

$0{,}3 \leq \varepsilon \leq 0{,}4$,

$0{,}4 \leq K_{Dmin} \leq 0{,}5$, $0{,}6 \leq K_{Dmax} \leq 0{,}7$;

$0{,}02 \leq e \leq 0{,}1$; $0{,}6 \leq \zeta \leq 0{,}85$;

$$\text{Max } C_d = f_c Z^{2/3} D_b{}^{1,8} \quad \text{if} \quad D \leq 25{,}4 \text{ mm}; \tag{9}$$

$$C_d = 3{,}647 f_c Z^{2/3} D_b{}^{1,4} \quad \text{if} \quad D > 25{,}4 \text{ mm}; \tag{3.130}$$

Kısıtlar;

$$h_1 = \frac{\varphi_0}{2\sin^{-1}\left({}^{D_b}/_{D_m}\right)} - Z + 1 \geq 0; \tag{3.131}$$

$$h_2 = 2D_b - K_{Dmin}(D - d) \geq 0; \tag{3.132}$$

$$h_3 = K_{Dmax}(D - d) - 2D_b \geq 0; \tag{3.133}$$

$$h_4 = \zeta B_w - D_b \leq 0; \tag{3.134}$$

$$h_5 = D_m - 0{,}5(D + d) \geq 0; \tag{3.135}$$

$$h_6 = (0{,}5 + e)(D + d) - D_m \geq 0; \tag{3.136}$$

$$h_7 = 0{,}5(D - D_m - D_b) - \varepsilon D_b \geq 0; \tag{3.137}$$

$$h_8 = f_i \geq 0{,}515; \tag{3.138}$$

$$h_9 = f_0 \geq 0{,}515; \tag{3.139}$$

Amaç fonksiyon; asgari f(min); (3.140)

$$f(\min) = 37{,}91\left[1 + \left\{1{,}04\left(\frac{1-\gamma}{1+\gamma}\right)^{1,72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0,41}\right\}^{10/3}\right]^{-0,3} \text{ x } \left[\frac{\gamma^{0,3}(1-\gamma)^{1,39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0,41}$$

Montaj açısı; (3.141)

$$\varphi_0 = 2\pi - 2 \text{ x } \cos^{-1}\left(\frac{\left[\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\}^2 + \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}^2 - \left\{\frac{D}{2} - \frac{T}{4}\right\}^2\right]}{2\left\{\frac{(D-d)}{2} - 3\left(\frac{T}{4}\right)\right\}\left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}}\right)$$

3.4.2. Diyafram yayının optimizasyonu

Diyafram yayının rulman yük karakteristiği kavrama ayrılma kalitesini belirlediği için diyafram yay kavrama sisteminin kritik bir parçasıdır. Statik koşullara kıyasla dinamik koşullar altında diyafram rulman yükü önemli ölçüde değişir. Yüksek motor devirlerinde debriyaj pedalı kursu arttıkça diyafram rulman yükünde önemli bir düşüş görülebilir. Bu

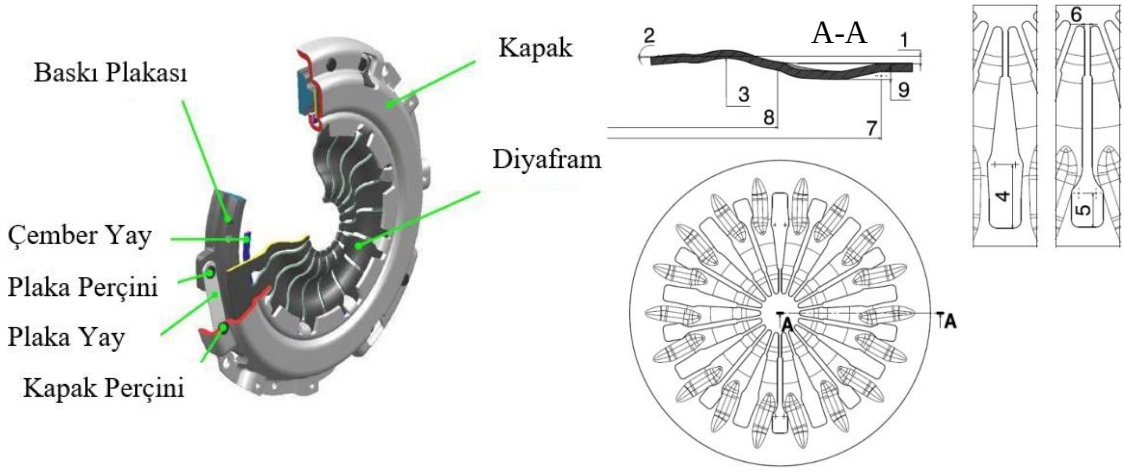
durum debriyaj performansı ve sürüş konforu açısından istenmeyen durumdur. Bu problem teknik literatürde yenidir. Bu çalışmada, C segment bir binek otomobilin diyafram yayı geliştirilmiş kır kurdu algoritması kullanılarak daha iyi hale getirildi. Eniyilenmiş diyafram yayı doğrulama için üretildi. Hali hazırda mevcut olan diyafram yayınınkilerle optimize edilmiş diyaframın yük taşıma özellikleri karşılaştırıldığında, yüksek hızlı dönüşlerde meydana gelen rulman yükü kaybı optimize edilmiş diyafram için yüzde elli ikiden yüzde yirmi seviyelerine azaltılmıştır. Parametre etkisi analizi ile diyafram rulman yükünü belirleyen parametrelerin parmak profili üzerinde olduğu belirlendi. Optimizasyon süreci ve parametre etki analizi kullanarak yüksek motor devirleri altında yüksek performansa sahip ayrılma davranışına sahip diyafram geometrisi elde edildi.

Debriyaj sistemi motordan torku şanzıman giriş miline iletir. Isıl, mekanik güvenilirlik ve yüksek devir altında dayanım emniyeti göz önünde bulundurularak tork iletiminin sağlanmasını ve kesilmesini debriyaj sistemi sağlar. Diyafram yayı, motor torkunu şanzıman giriş miline iletmek için gerekli eksenel yükü oluşturduğundan, tork kapasitesini belirleyen debriyaj sisteminin ana bileşenidir. Aktarılan tork, debriyaj ayırma rulmanının krank milinden giriş miline hareketiyle kademeli olarak artar ve değişen tork değerlerinde diyafram yayı baskı yükünü belirli bir limitte tutmalıdır. Diyafram ayrıca krank milinden ayırma rulmanına giden eksenel titreşimleri de kontrol eder ve krank milinden giriş miline motor dalgalanmasının sönümlenmesini sağlar. Bu nedenle, uygun diyafram tasarımı NVH performansını doğrudan ve dolaylı olarak etkiler (Erjavec ve Thompson, 2014). Tipik bir serbest bırakma rulman yükü ve serbest bırakma rulman hareketi grafiği için maksimum ve minimum noktalar vardır. Serbest bırakma rulman deplasmanı arttıkça yüksek hızlı dönüşlerde maksimum ve minimum yükler arasındaki fark artar, sürücü için rahatsızlık verebilir ve debriyaj performansını düşürebilir. Bu nedenle maksimum ve minimum noktaların azaltılması ve muhtemelen eşitlenmesi debriyaj performansını geliştirmeyi amaçlamaktadır. Araç güç performansı üzerindeki hayati önemine ve bazı tasarım değişiklikleriyle önemli gelişme potansiyeline rağmen literatürde optimizasyon teknikleri kullanılarak yüksek performansta diyafram tasarımı konusunda sınırlı sayıda çalışma bulunmaktadır. (Kaya, 2006; Shuxin, Lingyu, ve Xingwang, 2015; Wook-Hee, Choon-Yeol, Young, Jae-Do, Yong-Tak ve Seung-

Wan, 2000; Abhijit ve ark. 2016; He, Obubo, Fujii, ve Doman 2003). Bununla birlikte maksimum ve minimum diyafram kavrama yük farkını ortadan kaldırmaya odaklanan literatürde neredeyse tek çalışma vardır (Shangguan, Liu, Rakheja ve Hou, 2019). Optimizasyon algoritmaları, günümüz endüstrisinde yüksek verim elde etmek için en iyi tasarım parametrelerini belirlemede büyük önem taşımaktadır. Algoritmalar temel olarak evrimsel algoritmalar, fiziksel algoritmalar, sürü tabanlı algoritmalar olarak üç ana kategoriye ayrılabilir. Genetik algoritmalar (Darwin, 1859; Holland, 1975), genetik programlama (Koza, 1992), diferansiyel evrim (Storn ve Price, 1995), evrim stratejisi (Beyer ve Schwefel, 2002) ve biyo-ilhamlı algoritmalar (Simon 2008) gibi evrimsel algoritmalar, küresel çözümler içinde en iyi çözümü aramak ve bulmak için doğal evrimsel ilkelere dayanır. Big-bang big crunch (Erol ve Eksin, 2006), merkezi kuvvet (Formato, 2009) gibi fiziksel süreçler ve karınca kolonisi (Dorigo, Birattari ve Stutzle, 2006) ve yarasa (Shiqin, 2009) gibi sürü davranışından esinlenmiş fiziksel ve sürü temelli algoritmalar geliştirilmektedir. Bu çalışmada, yüksek dönme hızlarında (altı bin beş yüz rpm'e kadar) diyaframın optimizasyon problemini çözmek için geliştirilmiş kır kurdu algoritmasını sistemin reseptör düzenleme özelliği ile birleştirerek bir optimizasyon yaklaşımı geliştirilmiştir. Geliştirilmiş kır kurdu algoritması prosedürü hem arama süresinde hem de optimum parametreleri elde etmede daha fazla verim sağlar. Amaç fonksiyonları dinamik koşullar altında minimum diyafram rulman yükü kaybı, diyafram parmak rijitliği ve diyafram kavrama rijitliği olarak belirlenmiştir. Bu yeni yaklaşımla elde edilen optimum tasarımın prototipleri ayırma rulman yükü ayırma rulman yer değiştirme davranışı eniyileme çalışması için üretildi ve test edildi. Optimum diyafram tasarımının test sonuçları ve sonlu eleman simülasyon tahminleri karşılaştırıldı ve tasarımda önemli bir iyileştirme sağlandığı hem sayısal hem de deneysel olarak doğrulandı.

3.4.2.1. Diyafram yayının optimizasyonu malzeme ve yöntem

Bir C-Segment binek otomobili için debriyaj sisteminde kullanılan mevcut bir diyafram referans model olarak kabul edildi. Baskı plakası montaj komplesi ve diyafram tasarım değişkenleri sırasıyla Şekil 3.33 A, B'de gösterilmiştir.



A **B**

Şekil 3.33. Baskı plakası ve diyafram elemanının şematik gösterimi.
A) Baskı plakası montaj komplesi **B)** Diyafram yay tasarım parametreleri

Baskı plakası montaj komplesinin ana amacı, kavrama durumunda debriyaj diski ile volan arasında tork aktarımına izin vermek için yeterli baskı kuvvetini korumaktır. Ayrıca baskı montaj komplesi debriyaj diskinden uzaklaşarak volan ile bağlantıyı keser, bu sürece ayrılma denir. Diyafram yayı bazen kavrama yayı olarak adlandırılır ve baskı plakası kapağı ile baskı plakası arasında bulunur ve ayırma rulmanının eksenel hareketi ile baskı plakasının daha kontrollü ileri geri hareketini sağlar. Diyafram yayı kaldıraç işlevi görmesi için birkaç parmağa ayrılmıştır. Parmakların tasarlanan yük-yer değiştirme özelliklerine göre tasarlanan diyafram, uygun kavrama ve ayrılma için en kritik kısımdır. Parmakların şekli debriyaj performansını güçlü bir şekilde etkiler ve profili tanımlayan parametreler ayrıntılı olarak Şekil 3.33 B'de gösterilmektedir. Buradaki her parametre optimizasyon için bir değişken olacaktır ve bu değişkenler tasarım limitleri ile aşağıdaki Çizelge 3.9'da tanımlanmıştır. Bu çalışmada optimizasyon işlemi için geliştirilmiş kır kurdu algoritması kullanılmıştır.

Çizelge 3.9. Diyafram için tasarım değişkenleri ve limitleri

Değişken Sembolü	Değişken Adı	Alt Limit	Üst Limit
x1	Parmak yüksekliği (mm)	1,81	4,10
x2	İç açı (°)	4,10	14
x3	İç yarıçap (mm)	3,20	35,90
x4	Büyük delik derinliği (mm)	13,15	25,67
x5	Küçük delik derinliği (mm)	11,50	25,69
x6	Parmak ucu genişliği (mm)	0,81	1,79
x7	Kabartma sonu yarıçapı (mm)	170,44	199,69
x8	Kabartma başlangıcı yarıçapı (mm)	88,09	108,00
x9	Kabartma yüksekliği (mm)	2,20	4,25

Bir optimizasyon problemi en genel haliyle şu şekilde tanımlanabilir:

Amaç fonksiyon: En küçük f(x) değeri

Kısıt fonksiyon:

$g_k(x) \leq 0 \qquad x_j(a) \leq x_j \leq x_j(ü)\ (j= 1,..,n).$

$x_j(a)$ ve $x_j(ü)$ tasarım değişkenlerinin üst ve alt limitleridir. Tasarım parametreleri:

$x=\{x_1, x_2, x_3, ..., x_n)$

Geliştirilmiş kır kurdu algoritması kısaca aşağıdaki sekiz adımda açıklanabilir.

Adım 1: Başlangıç popülasyonu oluşturmak;

$$soc_{c,\,i}^{p,t} = lb_j + r_j * (ub_j - lb_j) \tag{3.142}$$

Adım 2: Çevreye uyumu değerlendirmek;

$$fit_c^{p,t} = f(soc_c^{p,t}) \tag{3.143}$$

Adım 3: Seçim ve sürü eğilimini değerlendirmek;

$$alfa^{p,t} = \{ soc_c^{p,t} \mid arg_{c=\{1,2,\ldots,N_c\}} minf(soc_c^{p,t}) \} \tag{3.144}$$

$$\text{kültürel eğilim}_j^{p,t} = \begin{cases} O_{\frac{(N_c+1)}{2},j}^{p,t} & , \quad Nc \text{ çift} \\ \dfrac{O_{\frac{(N_c)}{2},j}^{p,t} + O_{\frac{(N_c)}{2}+1,j}^{p,t}}{2} & \text{yoksa} \end{cases} \tag{3.145}$$

$$s_a = \text{alpha} - soc_c^{p,t}(\text{Alfa etkisi}) \tag{3.146}$$

$$s_c = \text{kültürel eğilim} - soc_c^{p,t}(\text{sürü etkisi}) \tag{3.147}$$

Adım 4: Yeni popülasyonu oluşturmak;

$$yeni_soc_c^{p,t} = soc_c^{p,t} + e1 * s_a + e2 * s_c \tag{3.148}$$

Adım 5: Yeni popülasyonun uyumunu değerlendirmek;

$$Yenifit_c^{p,t} = f\,(yeni_soc_c^{p,t}) \tag{3.149}$$

Adım 6: Yeni popülasyonu eski popülasyona göre değerlendirmek;

$$soc_c^{p,t+1} = \begin{cases} new_soc_c{}^{p,t+1} & new_J_c{}^{p,t+1} < J_c{}^{p,t+1} \\ soc_c^{p,t+1} & \text{yada} \end{cases} \tag{3.150}$$

Adım 7: Popülasyon çeşitliliğini arttırmak;

$$pup_j^{p,t} = \begin{cases} soc_{r1,j}^{p,t} & rnd_j < P_s \quad j = j_1 \\ soc_{r2,j}^{p,t} & rnd_j \geq P_s + P_a \quad \text{OR} j = j_2 \\ R_j & \text{yoksa} \end{cases} \tag{3.151}$$

Adım 8: İterasyonu durdur

$f_{max} - f_{min} \leq \varepsilon$, $\varepsilon \sim 10^{-6}$, $G = G_{max}$

Bu problemde amaç dinamik koşullar altında (yüksek dönüş hızları) ayırma rulman kuvveti kaybını en aza indirerek diyafram parmak katılığını ve diyafram dış çap bölgesi katılığını en üst düzeye çıkarmaktır. Parmak katılığı diyaframın serbest rulman tarafının katılığı iken diyafram dış çap bölgesi katılığı diyaframın baskı komplesi kapağı ile temas alanının katılığıdır. Daha yüksek katılık değeri, baskı plakası montaj komplesi için kavrama ve ayırma yük eğrilerindeki farkın sonucu olan sürtünmenin ortadan kaldırılmasına yardımcı olur. Bununla birlikte yüksek yük kayıpları tork iletiminin gerçekleşmemesine veya vites değişikliğinin olmamasına neden olabilir.

3.4.2.2. Diyafram yayının optimizasyonu için tasarım havuzu

Hesaplamanın işlem süresi sonlu elemanlar analizi ve optimizasyon çalışmalarında büyük önem taşımaktadır. Bu nedenle hesaplama süresini ve optimizasyon sürecini azaltmak için matematiksel modelin oluşturulması şarttır. Değişkenler ve sonuçlar arasındaki ilişkiyi ifade eden toplanan tasarım havuzuna bağlı deney tasarımından matematiksel model oluşturulur. Matematiksel model amaç fonksiyonlara ve kısıt fonksiyonlarına bağlı olarak en iyi tasarım parametrelerinin bulunmasını sağlar. Uygulanan eksenel yer değiştirmenin sonucu olarak üst ve alt sınır aralığında dinamik serbest bırakma rulman yükü kaybına tepki olarak dokuz tasarım değişkeni vardır. Örneklem boyutunu küçültmek için latin hiperküp örnekleme (LHS) yöntemi kullanılmış ve Çizelge 3.10'da verildiği gibi 100 tasarım noktası belirlenmiştir.

Çizelge 3.10. LHS'ye göre belirlenen tasarım noktaları

Tasarım noktaları	x1	x2	x3	x4	x5	x6	x7	x8	x9	Y1 (N)	Y2 (N/mm)	Y3 (N/mm)
1	2,4	5	21,5	21,5	11,5	1	186	108	2,7	1368,4896	1252,8	6532,6
2	2,80	6,50	14,72	19,51	18,91	1,64	178,24	103,30	2,74	1600,371	1130,1	6596,7
3	2,40	6,38	29,25	21,69	13,06	1,73	193,08	104,97	2,61	1589,8698	1209,8	6323,8
4	2,84	11,43	25,57	23,74	22,50	1,48	194,72	95,71	2,21	1583,0568	1502,2	6569,9
5	2,40	9,42	30,49	25,05	24,82	0,86	177,00	101,36	2,43	1581,156	1068,6	6765,1
6	2,15	6,66	30,13	23,38	13,18	1,77	183,96	95,22	2,71	1580,9742	1188,2	6461,6
7	3,82	5,10	18,09	22,70	16,39	1,07	189,33	108,00	2,55	1593,6246	1203	6428,5
8	3,81	6,23	20,55	25,18	15,28	1,10	185,93	107,94	2,71	1602,6786	2005,2	6444,2
9	3,22	9,23	5,61	16,07	19,73	1,29	194,26	103,50	2,73	1610,2476	1234	6274,4
10	2,64	8,35	5,81	19,03	17,14	1,75	177,50	98,06	2,64	1582,0686	1126,4	6605,5
.11	2,48	10,01	17,49	21,49	24,56	1,47	191,89	99,78	2,27	1579,374	1058,1	6543,9
12	3,30	7,01	6,71	23,12	17,20	1,11	181,25	103,01	2,69	1614,3336	1203,5	6542,6
13	2,40	6,04	26,77	22,83	15,91	0,82	199,69	105,70	2,52	1580,1066	1287,4	6330,8
14	2,74	9,07	6,38	22,39	21,90	1,70	195,59	97,17	2,51	1608,1902	1071,4	6358,8
15	2,91	7,52	27,04	21,38	15,74	1,70	194,57	102,02	2,73	1557,6282	1256,5	6270,6
16	3,09	10,89	10,85	25,39	19,48	1,46	199,34	98,00	2,35	1632,9816	1114,1	6498,1
17	2,40	8,11	25,37	22,13	14,42	1,16	177,00	95,27	2,57	1601,6166	1221,9	6692,9
18	2,52	8,46	24,27	15,94	21,47	1,00	182,03	105,24	2,28	1580,6286	1092	6709,6
19	2,68	5,54	15,55	15,25	12,95	0,93	184,63	96,51	2,54	1566,8892	1213,7	6536,2
20	2,39	10,17	34,09	18,01	16,49	1,54	186,38	94,47	2,52	1594,4976	1209,1	6500,3
21	3,55	11,24	3,70	19,10	11,99	0,97	179,37	105,22	2,28	1613,9286	1191,4	6792,3
22	2,44	10,56	28,44	24,53	25,29	1,23	179,27	93,64	2,58	1590,66	1097,8	6641,4
23	2,40	10,08	21,24	20,60	17,89	1,60	190,04	104,14	2,68	1576,944	1466,3	6346,9
24	3,50	5,13	10,66	15,10	13,64	1,30	192,47	99,97	2,83	1584,0558	1225,9	6229,6
25	2,07	7,34	9,30	17,38	23,68	1,49	194,19	104,42	2,75	1594,3392	1142,7	6248,6
26	2,19	9,79	19,84	25,28	20,19	1,21	176,50	105,64	2,45	1595,8224	1105,3	6760,7
27	2,40	9,93	27,63	18,51	12,19	0,85	189,52	103,00	2,42	1540,35	1255,3	6492
28	3,61	4,24	6,14	17,68	24,01	1,50	191,53	101,50	2,40	1574,8776	1072	6472,5
29	2,30	10,78	34,99	24,72	12,26	1,69	182,90	101,85	2,55	1585,7226	1187,9	6576,4
30	1,89	5,40	17,79	23,91	24,28	1,75	176,00	98,15	2,26	1582,6068	1117,1	6856,9
31	3,40	7,03	25,86	17,33	19,28	1,17	191,73	96,74	2,76	1598,4594	1295,3	6331,8
32	2,28	10,39	13,82	23,80	17,50	1,17	198,18	98,46	2,86	1577,2878	1339,8	6192,9
33	3,66	5,45	4,08	13,74	16,98	1,09	191,14	98,73	2,66	1605,3678	1171,1	6358,6
34	2,40	10,15	31,27	21,47	24,73	1,59	178,00	94,64	2,65	1603,845	1106,8	7691,7
35	2,93	9,03	4,36	18,70	22,92	1,62	176,00	99,38	2,33	1611,216	1046,3	6904,1
36	2,33	5,30	9,59	25,45	23,06	0,88	178,67	94,73	2,65	1584,3384	1136,8	6728,3
37	1,97	8,57	22,24	18,70	14,93	1,67	176,00	106,42	2,23	1578,1086	1020,6	6942,8
38	2,96	8,74	12,95	18,28	21,97	1,05	186,83	93,09	2,62	1632,9816	1114,1	6498,1
39	2,62	11,79	11,78	20,91	16,63	1,39	188,00	95,55	2,29	1577,0664	1174,2	6710
40	2,59	7,77	13,64	18,41	20,42	1,08	185,06	97,41	2,41	1587,0582	1128,5	6618,9
41	2,90	9,15	30,79	16,60	16,05	0,89	178,48	100,25	2,62	1574,442	1252,4	6740,2
42	3,24	5,96	14,13	23,41	19,08	1,52	178,00	93,96	2,70	1616,1462	1180	6680,9
43	1,88	8,82	5,16	15,52	23,51	1,32	192,19	94,96	2,22	1571,9436	1028,5	6623,4
44	3,42	5,56	7,21	21,78	19,40	1,40	178,00	100,67	2,83	1619,2296	1193,2	6618,7
45	2,85	11,66	12,53	25,59	24,40	1,31	195,19	97,74	2,29	1594,386	1104,4	6566,1
46	3,20	11,71	14,82	18,70	23,80	1,68	185,71	103,44	2,22	1600,3854	1048,2	6777,7
47	2,46	5,79	9,82	18,70	18,24	1,55	177,70	94,39	2,58	1598,9256	1088,4	6772,1
48	2,77	5,22	33,48	19,56	17,59	1,70	193,48	102,30	2,50	1576,575	1298,5	7387,1
49	2,40	8,01	34,57	15,70	17,77	1,27	197,27	105,01	2,39	1552,8852	1187,7	6478,7
50	3,05	7,82	31,85	16,42	14,95	1,11	179,72	100,05	2,79	1600,2306	1284,1	6618,5
51	3,12	4,31	26,68	15,06	11,52	0,94	184,17	105,41	2,86	1578,9222	1329,2	6466,9
52	3,48	7,43	4,88	24,35	25,19	1,58	182,39	96,44	2,25	1601,6292	1001,1	6858,2
53	2,75	11,55	12,02	18,79	22,25	1,08	184,78	104,00	2,21	1568,2266	1028,9	6793,7
54	3,39	6,82	5,32	15,47	18,77	1,30	199,18	98,81	2,82	1617,8814	1253,7	6223,1
55	2,40	8,54	35,90	24,23	11,76	1,24	195,81	101,07	2,35	1562,085	1208,5	7662,3
56	2,53	11,06	28,75	21,91	22,63	1,74	198,40	101,75	2,39	1571,8986	1078	6474,5
57	1,93	8,03	17,02	16,17	15,52	1,19	177,91	97,01	2,61	1595,385	1203,2	6801,9
58	2,92	7,28	16,78	13,25	12,42	0,95	174,99	107,25	2,72	1602,3708	1269,1	6704,5
59	2,65	8,64	31,03	19,70	13,43	1,75	188,39	99,56	2,59	1572,5826	1207,9	6498,5
60	3,02	4,97	10,24	17,79	24,66	0,81	193,20	96,62	2,75	1583,352	1188,9	6727,9
61	3,63	4,38	10,51	24,13	16,84	1,37	177,90	96,21	2,87	1580,9868	1218,2	6708,9
62	2,23	7,96	32,80	20,25	13,76	1,70	188,58	98,96	2,27	1588,4064	1177,3	6658,8

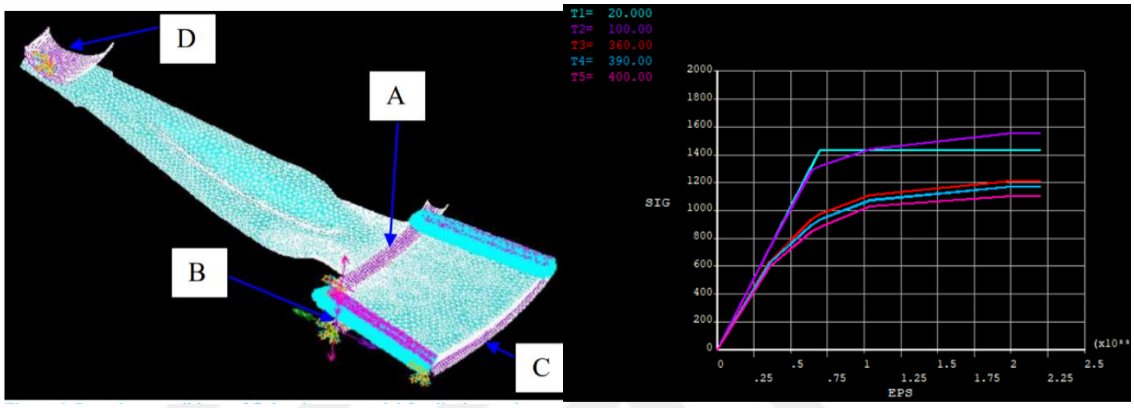
Çizelge 3.10. LHS'ye göre belirlenen tasarım noktaları (devam)

Tasarım noktaları	x1	x2	x3	x4	x5	x6	x7	x8	x9	Y1 (N)	Y2 (N/mm)	Y3 (N/mm)
63	2,30	8,92	9,09	15,37	22,38	1,04	193,69	104,32	2,32	1584,2862	1190,1	6658,9
64	1,83	10,73	12,12	22,22	18,15	1,35	198,62	107,39	2,60	1580,526	1190	6319,8
65	3,26	8,77	15,06	16,00	24,34	1,43	191,00	100,92	2,36	1595,0574	1035,9	6581,9
66	3,20	8,28	8,16	15,70	14,00	0,88	176,30	96,22	2,44	1587,8088	1186,4	7029,5
67	2,61	9,69	8,57	18,12	23,58	1,01	186,17	98,25	2,56	1573,7796	1115,3	6546,7
68	3,75	4,51	21,74	13,40	22,76	1,14	180,57	97,29	2,30	1574,6166	1063,4	6830,1
69	2,87	4,10	3,20	21,19	12,49	1,50	195,08	93,31	2,79	1600,3026	1197,7	6273,6
70	2,37	5,71	8,65	19,00	14,61	1,68	193,85	99,51	2,87	1593,7938	1177,7	6232
71	3,13	7,86	22,48	19,88	11,93	1,27	179,88	106,60	2,85	1601,262	1288,9	6607,7
72	2,24	4,92	27,83	20,51	25,69	0,81	192,32	95,99	2,20	1556,064	1039,5	6607,4
73	3,55	9,16	19,48	23,53	21,37	0,96	185,50	100,88	2,80	1608,3792	1233,3	6452,3
74	3,36	11,04	6,63	18,02	13,29	1,56	188,78	98,44	2,73	1602,8532	1303,2	7470,9
75	2,83	5,75	14,28	18,00	21,51	1,57	187,30	96,79	2,70	1599,0066	1152	6474,3
76	2,99	4,72	25,07	22,82	21,27	0,90	196,42	104,09	2,53	1553,1912	1148,3	6424,8
77	1,87	7,73	31,50	19,20	25,52	1,42	177,16	99,07	2,52	1586,6388	1066,3	5760,8
78	3,01	5,84	22,85	25,67	23,17	0,85	177,00	107,01	2,40	1601,0766	1136	6842,4
79	3,50	9,46	20,16	14,58	12,60	1,37	196,56	105,53	2,41	1188,3456	1250,9	6478,3
80	2,56	6,75	16,31	13,38	25,59	1,23	173,48	93,36	2,43	1607,562	1065,2	6873,9
81	3,70	9,60	26,20	24,05	20,04	1,59	190,58	105,93	2,76	1587,231	1192,6	6368,6
82	2,80	6,27	29,66	14,22	18,58	1,61	183,24	101,26	2,75	1579,9086	1206,4	6533,1
83	2,88	11,22	7,71	19,38	14,37	1,18	178,95	94,23	2,60	1619,6022	1317,2	6751,7
84	3,34	5,35	20,80	13,64	23,41	1,79	188,12	103,82	2,88	1589,5188	1190,8	6322,3
85	2,50	9,39	11,55	19,31	20,35	0,97	181,86	102,42	2,31	1592,928	1110,8	6837,8
86	3,78	4,65	29,13	24,76	15,84	0,98	180,09	107,12	2,55	1576,08	1192,3	6715,9
87	2,72	11,96	31,67	14,00	21,77	1,67	173,17	103,49	2,68	1620,4464	1140,3	6698,4
88	1,81	8,44	22,58	17,52	20,60	1,20	175,37	94,09	2,23	1580,7438	1059,4	6951,5
89	1,94	4,44	12,49	18,72	14,67	0,93	175,75	93,18	2,81	1591,0326	1254,6	7684,6
90	2,15	9,85	15,85	13,15	20,25	1,34	188,89	95,38	2,38	1551,3408	1153,1	6626,4
91	3,29	4,16	18,33	21,96	24,13	1,33	187,73	102,77	2,54	1592,4852	1124,2	6577
92	2,26	11,51	8,10	17,13	22,09	0,92	198,88	106,79	2,82	1599,6852	1287,5	6175,1
93	3,85	11,34	24,72	16,69	20,73	1,12	186,47	99,21	2,42	1599,0192	1248,6	6620,2
94	2,04	6,95	17,30	16,96	14,23	0,83	190,43	106,50	2,84	1566,468	1308,8	6312,1
95	2,32	6,70	30,38	15,85	12,76	1,02	181,53	107,64	2,34	1566,369	1186,7	6756,9
96	2,09	11,40	24,15	18,91	19,97	1,54	189,95	96,99	2,46	1566,5274	1148,3	6554,4
97	3,88	4,68	26,36	14,15	15,58	1,25	198,92	102,85	2,36	1571,0328	1158,5	6493,2
98	2,43	10,33	32,86	20,39	23,87	1,04	180,83	102,12	2,84	1589,4576	1207,7	6557,2
99	3,70	7,44	28,04	20,73	18,61	1,38	170,44	104,50	2,39	1582,6068	1106,2	6856,9
100	3,9	5	21,5	21,5	11,5	1	186	108	2,7	1587	1595	1480

Yukarıda açıklanan optimizasyon algoritması, Matlab ® kullanılarak gerçekleştirilmiştir. Bu optimizasyon sürecinde dokuz girdi, bir amaç fonksiyon ve iki kısıt fonksiyon olmak üzere üç adet çıktı parametresi bulunmaktadır. İşlenmesi gereken çok sayıda veri olması ve optimizasyon problemi olması nedeniyle, ön değerlendirmede ANSYS optiSlang kullanılarak parametre etki analizi yapılıp en iyi değerler için çözüm arama bölgesi daraltılmıştır.

3.4.2.3. Fem sonlu elemanlar yöntemi

Her tasarım noktası için amaç fonksiyonu, Sonlu Elemanlar Metodu (FEM) kullanılarak hesaplanır. FEM hesaplamaları için ANSYS APDL®'de diyaframın bir bölü on sekiz modeli oluşturulur. Bu simetrik model, tüm modele kıyasla hesaplama süresini önemli ölçüde azaltır. Mesh yapısı, sınır koşulları ve yükleme aşağıdaki Şekil 3.34'te gösterilmektedir.



A **B**

Şekil 3.34. Diyafram fea modeli ve malzemesi şematik gösterimi.
A) Simetrik diyafram modeli için ağ yapısı, sınır ve yükleme koşulları **B)** Farklı test sıcaklık değerlerine göre malzeme gerilim – gerinim davranışı

Diyafram yayı için malzeme young modülü iki yüz beş GPa ve poisson oranı sıfır nokta iki yüz yetmiş dokuz olan 51CrV4'tür. Çekme testinden elde edilen malzeme için gerilme-gerinim eğrisi yazılıma girildi. Görüldüğü gibi düşük sıcaklıklar için elastik bölgeyi takiben idealize edilmiş elastoplastik davranış göstermektedir.

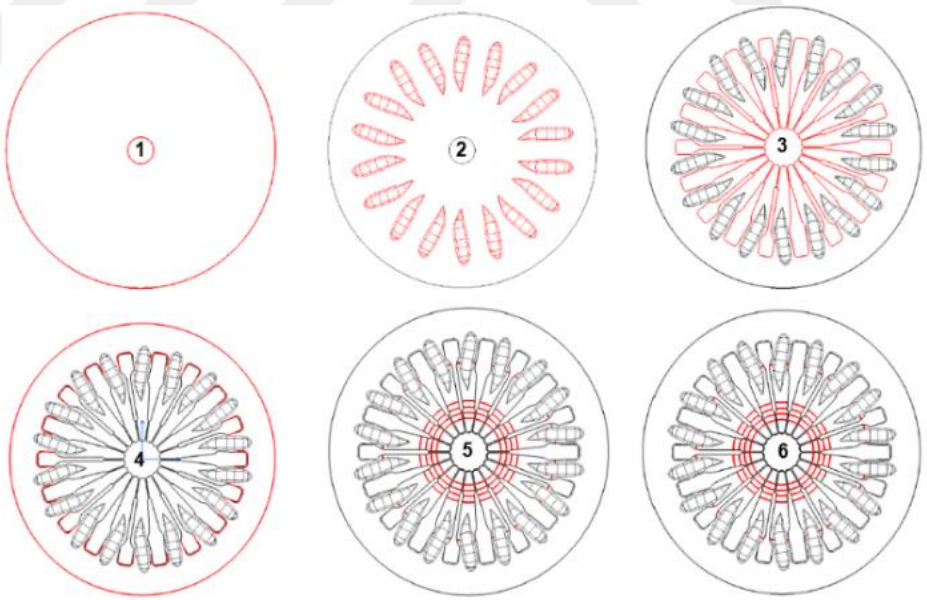
Sıfır nokta doksan beş mm genel ve Sıfır nokta beş mm yerel ağ boyutuna sahip dörtgen mesh (SOLID187) kullanılmış ve modelde toplam yüz iki bin dörtyüz dört düğüm ve altmış dokuz bin beşyüz seksen üç eleman oluşturulmuştur. Sınır koşulları aşağıdaki gibi tanımlanır:

Tüm yer değiştirmeler ve dönmeler, diyafram ve kapak arasındaki temas hattında sınırlandırılmıştır. Şekil 3.34'te A); diyafram ve dayanak noktası (B); diyafram ve kapak (C). Ayırma rulman yükü ekseni dışındaki tüm yer değiştirmeler, rulman ve parmak arasındaki temas hattı için sınırlandırılmıştır. Tüm dönüşler bu alan için

sınırlandırılmıştır. Şekil 3.34 A'da D). Yükleme için, altı bin beş yüz rpm'lik dönüş hızı, azami ayırma rulman yer değiştirmesinin üzerine bindirilir.

3.4.2.4. Optimum boyutlarda ve yük-deplasman karakteristiklerinde diyafram üretimi

Elde edilen en iyi parametrelere göre model oluşturuldu ve elde edilen optimum tasarım Şekil 3.35'te gösterildiği gibi altı adımda test için üretildi. Birinci adımda, telerezyon kullanılarak çevre ve kılavuz pimler için merkezde bir delik kesilir. İkinci adımda hidrolik pres kullanılarak kabartma geometrileri yapılır ve üçüncü adımda telerezyon ile aşağıdaki parmak yuvaları kesilir. Dördüncü adımda parmak yuvalarının kenarlarında ve çap çevresinde yarıçap oluşturulur. Ardından sırasıyla adım beş ve altıdaki işlemler sıcak şekillendirme ve stabilizasyon ile parmak profilleri oluşturulur. Telerezyon, hidrolik pres, sıcak şekillendirme ve stabilizasyon makineleri Bursa, Türkiye'de bulunan Valeo fabrikası tarafından sağlanmıştır. Yayın Yük-Deplasman davranışı da fabrikada hazır bulunan özel bir test cihazı kullanılarak test edilmiştir.



Şekil 3.35.: En iyilenmiş diyaframın üretim adımları.

3.4.2.5. Optimum diyafram için serbest yatak özellikleri testi

Optimum diyafram yayı bir baskı plakasına monte edilir ve ayrılma rulman yükü, ardışık olarak uygulanan bin, iki bin, üç bin, dört bin, beş bin ve altı bin beş yüz rpm dönüş hızları için ayrılma strokunun bir fonksiyonu olarak ölçülür. Rulman yükleri motor devrine göre değişir ve debriyajın ayrılma fonksiyonunu tanımlar. Test için Bursa, Türkiye'deki Valeo fabrikasında hazır bulunan test sistemi kullanılmıştır.

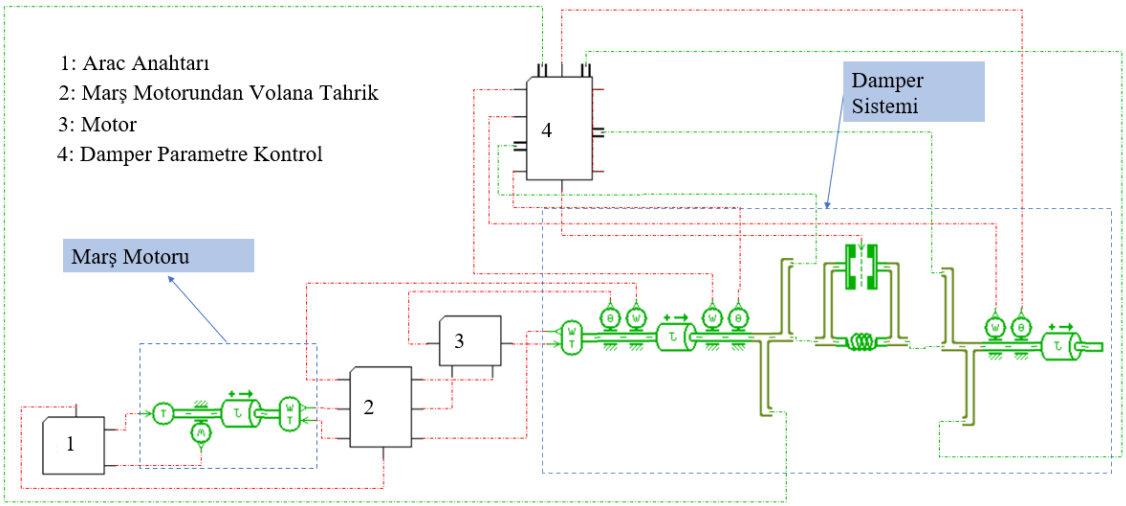
3.4.3. Taşıt debriyaj damper elemanının modellenmesi

Taşıt debriyajlarında motor krank mili ile baskı plakası arasına yerleştirilen burulma damperi motor torkunu iletir ve motordan gelen titreşimleri sönümler. Burada damper kütle, atalet özellikleri, damper yay katılık özellikleri, sürtünme torku özellikleri önemli parametreleridir. Bu çalışmada literatürde sıklıkla aktarma organlarının modellenmesinde kullanılan AMESim programı ile debriyaj damper modeli modellenmiştir. Debriyaj damper sisteminin görevi motorda oluşan hız ve tork değişimlerinden kaynaklı torsiyonel titreşimleri sönümlemektir. Motor titreşimleri periyodik olarak sinüs şeklinde zaman tabanlı ve frekans tabanlı olarak tasvir edilebilir. Eğer titreşimler debriyaj damper elemanı aracılığıyla sönümlenmez ise titreşimlerden kaynaklı şoklar ile gürültü problemleri ve mekanik problemler ortaya çıkabilir. Dizel araçlarda benzinli araçlara göre daha yüksek titreşimler olduğu için debriyaj damper komplesinin açısal yer değişimi kapasitesi daha yüksek olur. Sistem modellemesi yapılan debriyaj damper sistemi ile sistemin davranışının tasarım sürecinde kontrolü sağlanır. Sistem modelinin bileşenleri olan matematiksel modellere hazır kütüphaneler ile kolayca erişilebilir. Otomobil üreticileri debriyaj sistemi geliştiren ekibe motor hız bilgisi ve vites kutusu giriş mili hızını verir. Motor hız bilgisi ve vites kutusu hız bilgilerinden yararlanarak motor torku hesaplanır.

$$\text{Motor}_{\text{tork}} = (\text{J}_{\text{motor}} + \text{J}_{\text{volan}}) \cdot \text{ivmelenme}_{\text{motor}} + (\text{J}_{\text{Damper}} + \text{J}_{\text{Baskı Komplesi}}) \cdot \text{ivmelenme}_{\text{Damper}} \quad (3.152)$$

Eğer bu datalar elde edilmez ise motor silindir çapı, strok, krank mili yarıçapı, motor sürtünmesi ve silindirlerdeki basınç dalgalanmalarını veri halinde verir. Bu veriler ile

motor çalışması süresince oluşacak tork dalgalanmaları hesaplanıp damper tasarımı belirlenir. Damper tasarımında volan ile vites kutusu arasında sürtünme torku, damper yay katılığı ve iki tarafın birbiri ile arasındaki hız farkından kaynaklanan sönümleme torku ara çıktı olarak önemlidir. Sürtünme torku eğer çok fazla olursa motor debriyaj damperi harekete iletemez, eğer az olursa çok yüksek titreşimler vites kutusuna iletilir ve arıza durumu meydana gelebilir. Vites kutusunun ivmelenmesi sürtünme torkunun seviyesine göre değişir, az sürtünme torku yüksek vites kutusu ivmelenmesi sağlar. Bu yüzden debriyaj damperi hesaplanırken sürtünme torkunun hesabı çok önemlidir. Sürtünme torku yanında damper katılık değeri ve açısal yer değiştirme kapasitesi de hesaplanması gereken unsurlardır. Damper katılık değeri arttıkça vites kutusu giriş mili ivmelenmesi artar. Motor torku aktarılırken motor krank miline bağlanan volan atalet momenti ve krank mili katılığı giriş milinin ivmelenmesinde ters orantılı etki oluşturur. Motor titreşimlerini vites kutusu giriş miline sönümlenmiş şekilde iletmek için damper sürtünme torku, volan ataleti, krank mili katılığı ve damper katılığı son derece önemlidir. Bu çalışmada damper sürtünme torkunun optimizasyonu üzerine çalışılmıştır. Damper sisteminde sürtünme torku damperin açısal yer değiştirmesi boyunca oluşan toplam sürtünmedir. Sürtünme torku genel olarak sürtünen yüzey sayısına, sürtünme yüküne ortalama yük yarıçapına ve sürtünme katsayısına bağlı olarak değişir. Motor titreşimlerinin sönümlenmesi için sürtünme torkunun belirli bir değerde olup debriyaj ömrü boyunca korunmalıdır. Debriyaj sistemini modellerken sürücü aracın kontağını açtığı anda marş verilmesi, araç motorunun çalıştırılması ve çalışması süresince debriyaj sisteminin motor titreşimlerini sönümlemesi modellenmiştir. Sistem modellemesi ile motorun çalıştırılıp, damper ile sönümlemenin sağlanması ile damper sisteminin dayanıklılığı ve konforu eniyilenebilir.

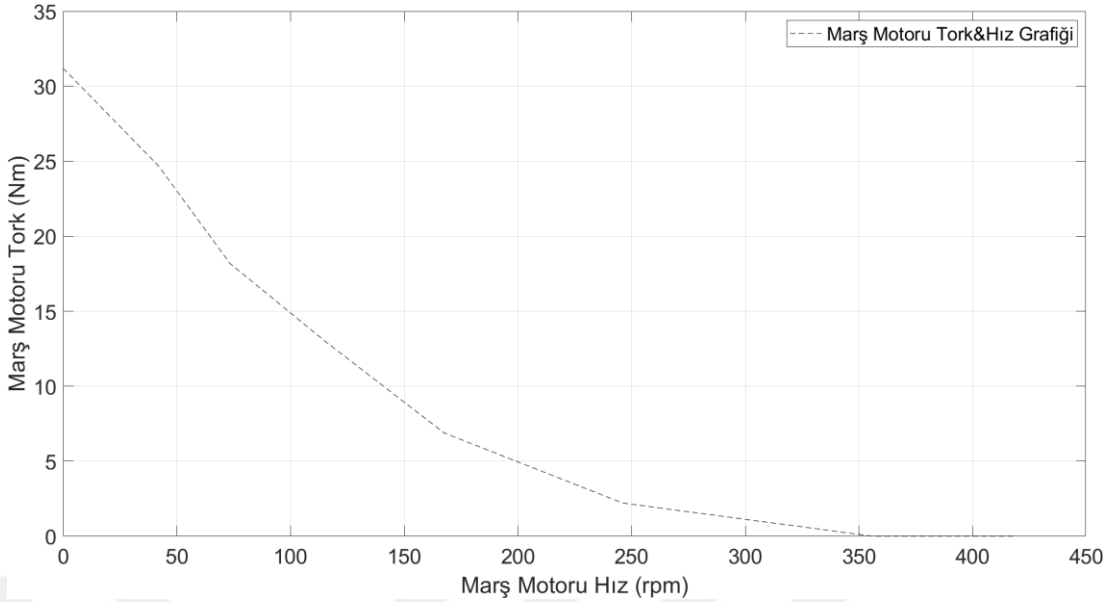


Şekil 3.36. Debriyaj sisteminin 1 boyutlu sistem modellenmesi.

Motor titreşimlerinin debriyaj tarafından sönümlenmesinin sisteme tanımlanmasında her bir alt ürünün atalet momenti değerlerinin damper yay rijitlik, sürtünme momenti gibi bilgiler önemlidir. Debriyaj damper modeli titreşimleri sönümleyen damper yay ve damper iç sürtünme momenti oluşturan parçalardan oluşur.

3.4.3.1. Geliştirilmiş kır kurdu algoritması ile taşıt debriyaj damper elemanı sistem optimizasyonu

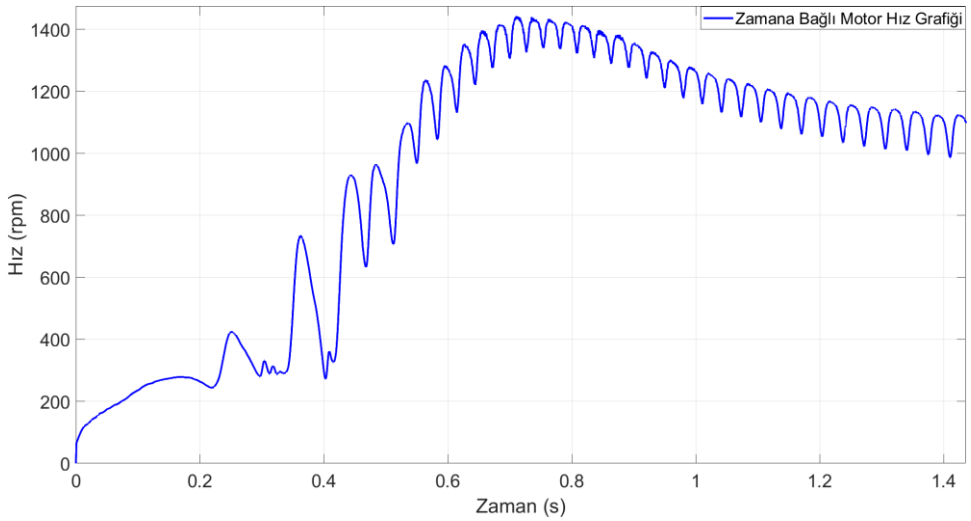
Optimizasyon algoritmalarında amaç fonksiyon deney tasarım havuzu oluşturarak değişkenlere bağlı matematiksel model elde edilebilir. Bu çalışmada motorun çalıştırılması sırasında motor titreşimlerini damper çalışma açısal limitleri içinde en iyi şekilde sönümleme görevini getiren sürtünme torku parametrelerinin eniyilenmesi amaçlanmıştır. Bunun için MATLAB program ara yüzünde algoritma AMESim modeli ile ilişkilendirilip, MATLAB algoritması ile Amesim girdileri sağlanıp eniyilenmiş damper sürtünme torku değerleri elde edilir. Araç çalıştırılması sırasındaki sönümleme durumu araştırıldığı için marş motoru dişli parametresi, volan dişli parametresi, marş motoru mili sertlik, marş motoru mili katılık ve marş motoru tork ve hız bilgileri bir boyutlu model içerisine dahil edilmiştir.



Şekil 3.37. Marş motoru karakteristiği.

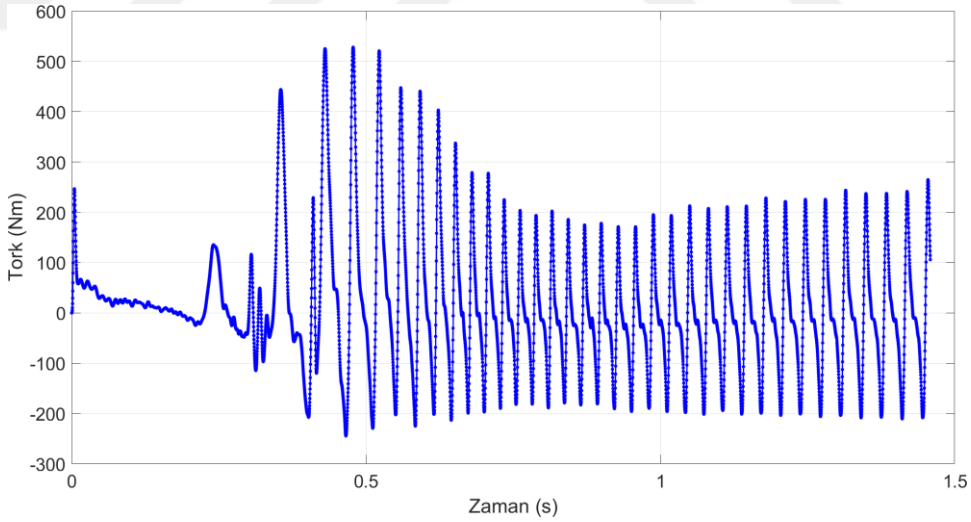
Çizelge 3.11. Amesim model parametreleri

Değişken Sembolü	**Değişken Adı**	**Değer**	**Optimizasyon Alt ve Üst Limiti**	
Im	Motor Atalet Momenti (kg.m^2)	0,03		
Iv	Volan Atalet Momenti (kg.m^2)	0,13		
Id	Damper Atalet Momenti (kg.m^2)	0,03		
Ib	Baskı Komplesi Atalet Momenti (kg.m^2)	0,055		
A1	Damper açısı 1. Kademe (°derece)	4		
A2	Damper açısı 2. Kademe (°derece)	25		
A3	Damper açısı 3. Kademe (°derece)	35		
K1	Damper yay katılığı 1. Kademe (Nm/°)	0,3		
K2	Damper yay katılığı 2. Kademe (Nm/°)	6		
K3	Damper yay katılığı 3. Kademe (Nm/°)	24		
Değişken Sembolü	**Değişken Adı**	**Değer**	**Alt Limit**	**Üst Limit**
H1	Damper Sürtünme Momenti 1. Kademe (Nm)	Optimizasyon	1	3
H2	Damper Sürtünme Momenti 2. Kademe (Nm)	Optimizasyon	80	160
H3	Damper Sürtünme Momenti 3. Kademe (Nm)	Optimizasyon	180	240



Şekil 3.38. Zamana bağlı motor hız eğrisi.

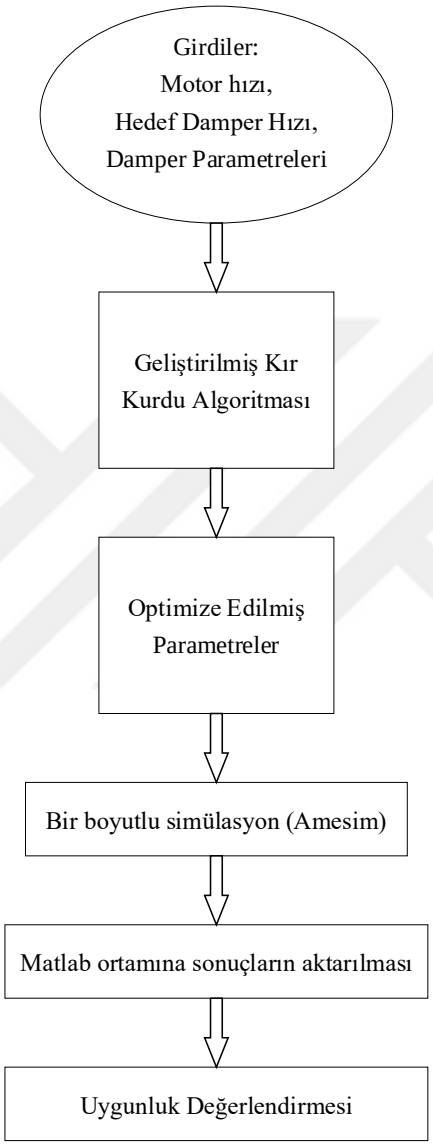
Bu çalışmada 1sn'de yüz otuz rpm'e ulaşan motor eğrisi belirlenip, bu motor hız eğrisinden motor tork ve titreşim değerleri hesaplanmıştır. Sonrasında bu motordan meydana gelen titreşimleri sönümlemek için gerekli damper parametreleri optimizasyon algoritması aracılığıyla elde edilmiştir.



Şekil 3.39. Zamana bağlı motor tork eğrisi.

Motordan elde edilen torkun ve hızın titreşimsiz sönümlenmiş olarak dişli kutusuna iletimi araç ömrü ve konforu açısından çok önemlidir. Bu yüzden debriyaj tasarım süreçlerinde damper karakteristiği çok önemlidir.

Geliştirilmiş kır kurdu algoritmasında optimizasyonu yapılan damper histerezis parametreleri Matlab ortamında bir boyutlu modelin çalıştırılmasıyla optimize edilmiş parametreler kullanılır ve bir boyutlu modelin sonuçları Matlab ortamında değerlendirilir ve sonuç elde edilir.



Şekil 3.40. Amesim entegreli yapay zeka tabanlı sistem akış şeması.

4. BULGULAR ve TARTIŞMA

Bu test çalışmasında önerilen her bir algoritma için aynı makine üzerinde üç kez çalıştırılmasından sonraki istatistiksel tanımlamaları içeren en kötü, ortalama, en iyi ve standart sapma uygunluk değerlerine göre performansı ölçülmüştür. Algoritmaların genel performansı için ortalama uygunluk değeri algoritmanın ne kadar benzer sonuclar üretebileceğini ölçen standart sapma değeri kadar önemlidir. Algoritmanın en iyi uygunluk değeri en iyi performansı temsil etmektedir.

Test fonksiyonlarının optimizasyonu, MATLAB yazılımı aracılığıyla farklı COA türleri için deneysel olarak gerçekleştirilmiştir. Bu deneyde, standart COA dahil olmak üzere aşağıdaki yirmi üç farklı yeni oluşturulan algoritma birbiri ile karşılaştırılmıştır. Önerilen algoritmalar otuz farklı çalıştırma sonucunda problemler üzerinde test edilmiştir. Kısıtlı problemlerde ceza metodu kullanılarak problemin uyum fonksiyonu değerlendirilmiştir.

Karşıt tabanlı öğrenme “O”yu temsil eder;
Kaotik yerel arama “CCL”yi, kaotik ”C”,yi temsil eder;(Iteratif kaotik harita)
Uygunluk mesafe dengesi “FDB”yi temsil eder;
Levy uçuşu “LF”yi temsil eder;
Laplacian “L”yi temsil eder;

COA'ya farklı kombinasyonlarla uygulanan bu algoritmaların elde edilen sonuçları Çizelge 4.1'de, Çizelge 4.5’te ve Çizelge 4.9’da verilmiştir. Algoritmanın optimal çözümünün her versiyonu çizelgede ortalama, en iyi, en kötü ve standart sapma hesaplamaları olarak listelenmiştir. Kıyaslama işlevleri, en iyi sonuçları ve en küçük standart sapma sonuçlarını değerlendirerek, algorima üçün (LCCLFDBLFCOA) yirmi üç işlevin tamamında optimal bir çözüm için en iyi kombinasyon olduğunu göstermektedir. Deneysel sonuçlara göre, GCOA(GCOA6), karakteristiğinin güçlü keşif kabiliyeti göstermesi ve yerel optimuma düşmeyi önlemesi nedeniyle standart COA'ya göre üstündür. Bu karşılaştırma, GCOA6’ün genel performansının, sürekli global arama yeteneği ile güçlü optimizasyon yeteneği nedeniyle en iyisi olduğunu ortaya koymaktadır.

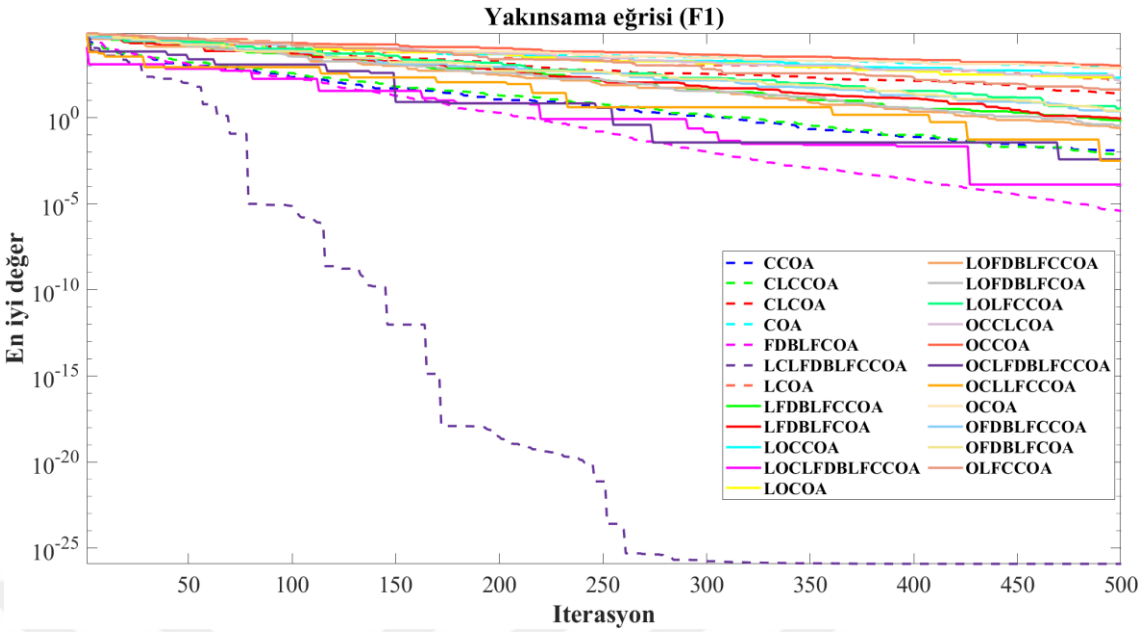
Bu çalışmada, kısıtsız optimizasyon problemlerini değerlendirmek için yirmi üç kıyaslama fonksiyonu kullanılmıştır. Öte yandan, deneysel çalışmalar önerilen algoritmayı doğrulamak için on kısıtlı mühendislik problemini de test etti. Birçok araştırmacı aynı kısıtlı ve kısıtsız problemler için çalışmıştır. Bu otuz üç kıyaslama işlevi temel olarak minimizasyon problemleri içindir. Birkaç model, problem boyutuyla katlanarak artan birçok yerel optimuma sahiptir. Global arama yeteneğinin yeterli olup olmadığı önerilen algoritmaya olanak sağlar. Araştırma sonucunda standart kır kurdu optimizasyon algoritmasından (COA) farklı olarak on beş türde algoritma oluşturuldu. Bu algoritmaların türetilmesi Muhalefete Dayalı Öğrenme, Kaotik, Kaotik Yerel Arama, Uygunluk Mesafe Dengesi, Levy Uçuş, Laplacian gibi farklı teoremlerle yapıldı. Ardından, tüm kombinasyonlar test edildi ve karşılaştırıldı. En iyi kombinasyon, en iyi ve önerilen algoritmayı ortaya çıkardı. Her sütun, standart bir kır kurdu optimizasyon algoritması ile farklı bir teorem kombinasyonu verir. En iyi, ortalama, en kötü ve standart sapma değerleri aşağıdaki çizelgelerde belirtilmiştir. Bu istatiksel bulgulara göre önerilen algoritmaların kısıtlı ve kısıtsız problemler üzerinde tutarlı sonuç verdiği görülmüştür.
Literatürdeki farklı algoritmaların mühendislik problemleri sonuclarına dayanarak önerilen algoritmalardan en iyisi ile karşılaştırma yapılmıştır, bu karşılaştırmaya göre önerilen algoritmanın en iyi sonucu verdiği görülmüştür. Geliştirilmiş algoritma GCOA6 debriyaj sistemlerinden diyafram yayının dinamik karakteristiğinin ve damper elemanının sürtünme torkunun eniyilenmesinde kullanılmıştır.

Durdurma kriterini ayarla = M_{itr}
Sürü sayısını ayarla = Np ve her sürüdeki kır kurdu sayısını ayarla = Nc
Deney sayısını ayarla = Nexp ve problem boyutunu ayarla = D
Kır Kurdu sosyal populasyonunu başlat (Denklem. 3.37)
Tüm Kır Kurtlarının uyumunu değerlendir (Denklem. 3.10)
En iyi Kır Kurdunu ayarla x_{best}
while (Np*Nc < M_{itr})
for i = 1 to Np
Kır kurtlarının uyumuna göre her sürü için alfa kır kurdunu seç (Denklem. 3.12)
Uygunluk mesafe dengesi (FDB) ile sosyal eğilimi hesapla (Denklem. 3.43)

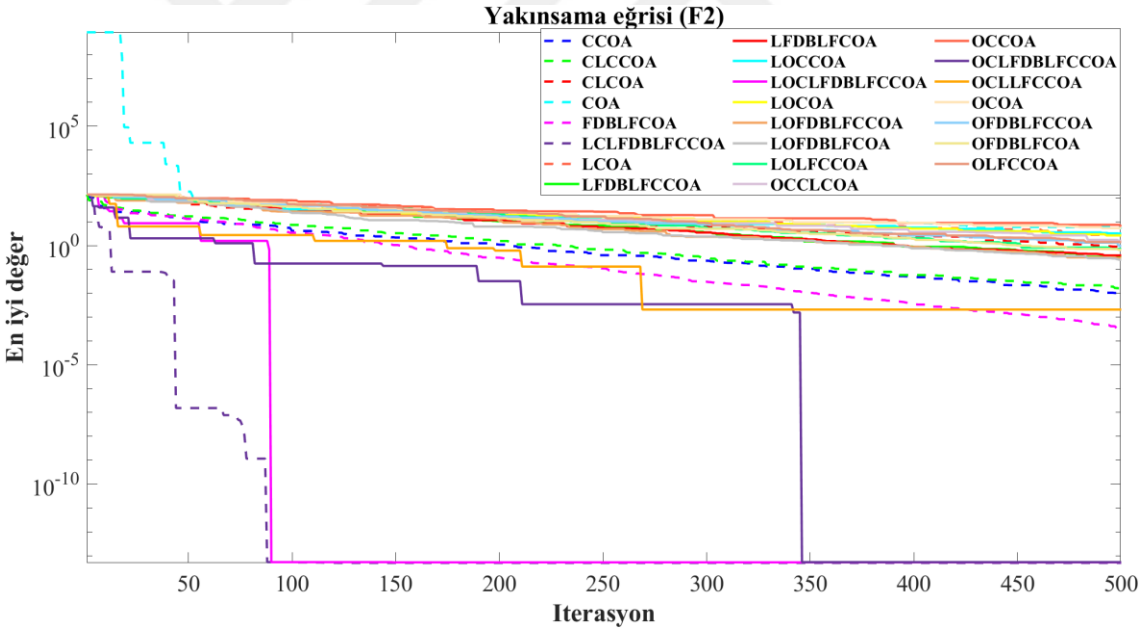
$$cult^{p,t} = soc^{p,t}_{FDB}$$

for c = 1: Nc
Kaotik olarak iki kır kurdu seç
Kaotik olarak kır kurtlarının sosyal durumunu güncelle (Denklem. 3.44)
Kır kurdu uyumunu güncelle (Denklem. 3.17)
Kır kurdu uyumlarını değerlendir (Denklem. 3.18)
end for
Levy uçuş dağılımına göre yeni kır kurtlarının doğumunu güncelle (Denklem. 3.46)
Kaotik olarak populasyonu güncelle
end for
Sürüler arasında kır kurtlarının pozisyonunu kaotik olarak güncelle
Uyumluluğu değerlendir
En iyi kır kurdunu güncelle x_{eniyi}
En iyi kır kurdunu x_{eniyi} ve rastgele kır kurdunu $x_{kır\ kurdu}$ seç ve laplas çaprazlaması uygula (Denklem. 3.50, 3.51)
Eğer daha iyi bir çözüm varsa en iyi kır kurdunu güncelle x_{eniyi}
Kaotik bölgesel arama işlemini uygula (Denklem. 3.52)
Kır kurtlarının yaşını güncelle
end while
En iyi uyumlu kır kurdunu seç

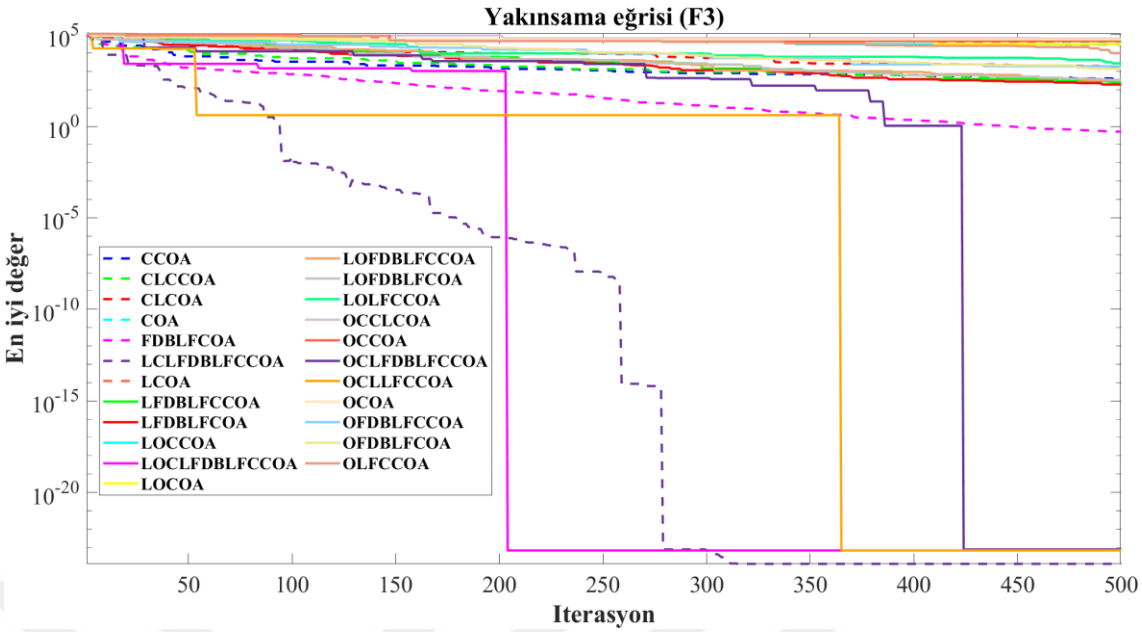
Şekil 4.1. Önerilen kır kurdu algoritmasının sözde kodu (GCOA6).



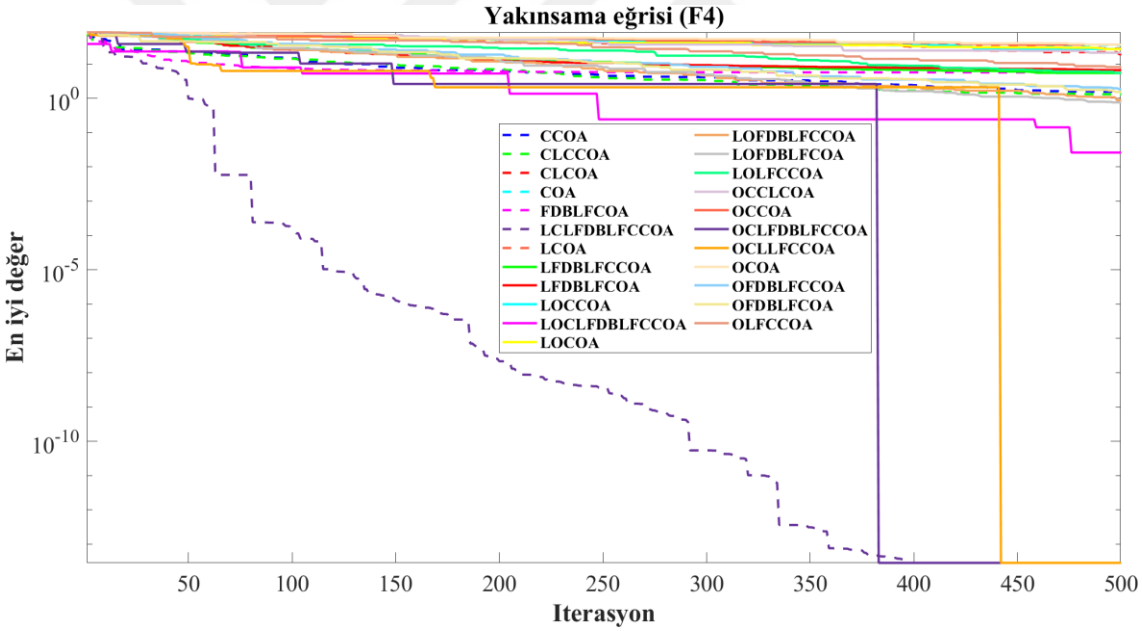
Şekil 4.2. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F1 test fonksiyonu için karşılaştırması.



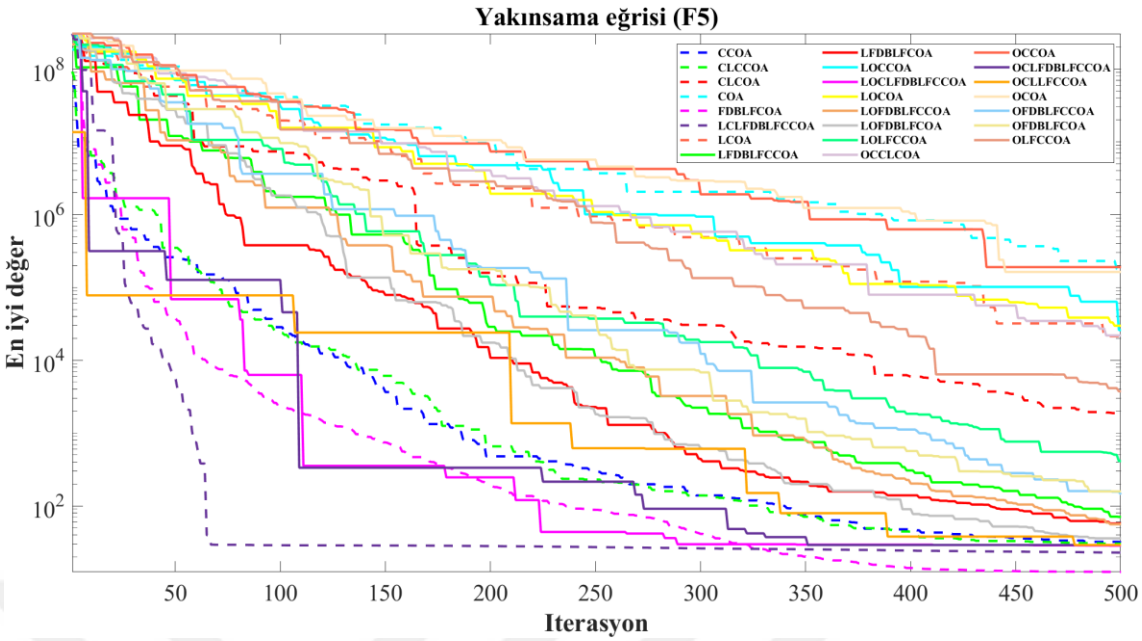
Şekil 4.3. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F2 test fonksiyonu için karşılaştırması.



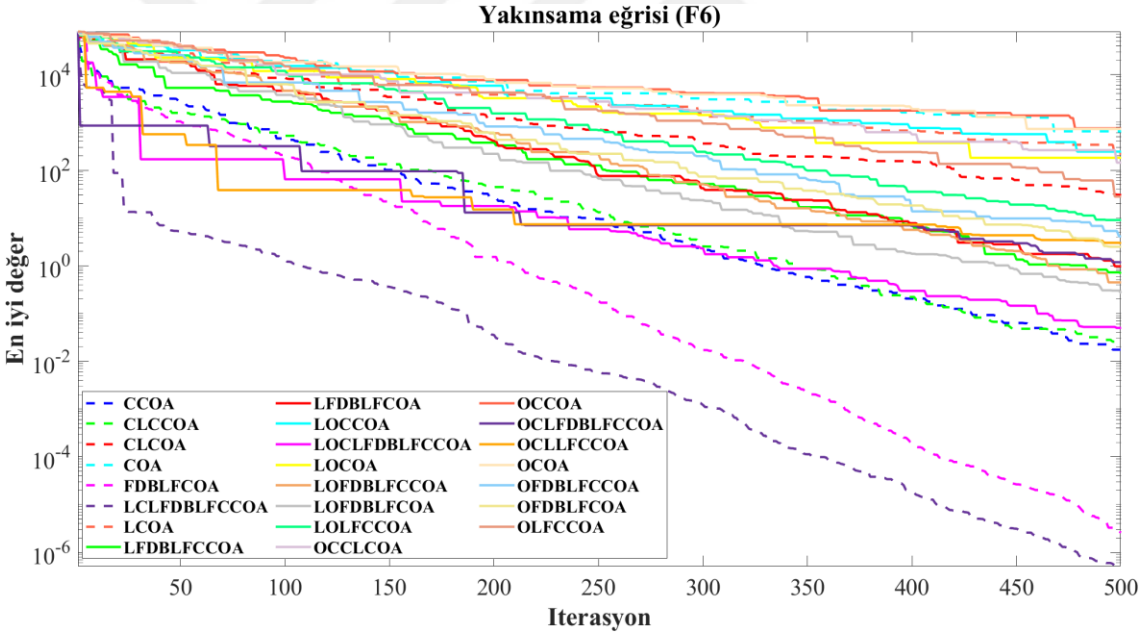
Şekil 4.4. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F3 test fonksiyonu için karşılaştırması.



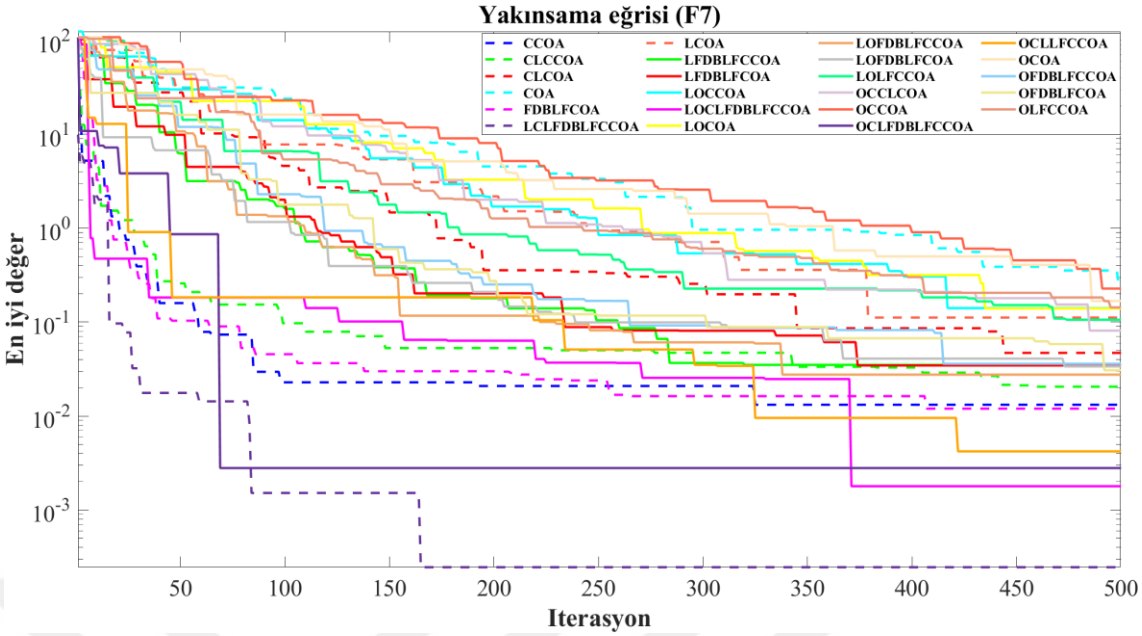
Şekil 4.5. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F4 test fonksiyonu için karşılaştırması.



Şekil 4.6. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F5 test fonksiyonu için karşılaştırması.



Şekil 4.7. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F6 test fonksiyonu için karşılaştırması.



Şekil 4.8. Yeni oluşturulan farklı kır kurdu algoritmalarının tek modlu F5 test fonksiyonu için karşılaştırması.

Çizelge 4.1. Tek modlu test fonksiyonları

No	Algoritmalar	Parametreler	Tek modlu test fonksiyonları						
			F1	F2	F3	F4	F5	F6	F7
1	CCOA	STD	0,108	0,068	1062,964	3,444	126,688	0,295	0,037
		Ortalama	0,087	0,059	983,021	3,342	109,452	0,235	0,035
2	CLCCOA	STD	0,102	0,055	1028,135	3,03	131,79	0,279	0,034
		Ortalama	0,08	0,05	968,581	2,897	114,272	0,221	0,033
3	CLCOA	STD	65,189	1,391	3344,282	28,666	6681,416	64,537	0,116
		Ortalama	59,36	1,367	3274,822	28,413	5782,659	61,705	0,11
4	COA	STD	990,919	6,546	51726,657	41,624	367770,999	1021,071	0,427
		Ortalama	980,85	6,513	51474,707	41,559	348622,199	1012,244	0,417
5	FDBLFCOA	STD	0	0,001	6,618	9,793	62,476	0	0,044
		Ortalama	0	0,001	4,475	9,474	51,141	0	0,038
6	LCLFDBLFCCOA	STD	0	0	0	2,922	24,74	0,001	0,003
		Ortalama	0	0	0	0,924	24,707	0	0,003
7	LCOA	STD	305,219	3,816	42901,285	34,739	57678,591	328,619	0,236
		Ortalama	300,287	3,793	42570,564	34,588	54308,206	321,961	0,23
8	LFDBLFCCOA	STD	2,801	0,802	492,845	11,132	265,819	5,945	0,079
		Ortalama	2,523	0,753	468,636	10,732	237,734	3,718	0,072
9	LFDBLFCOA	STD	2,394	0,802	475,392	12,692	299,782	2,732	0,086
		Ortalama	2,162	0,687	441,737	12,171	249,099	2,43	0,078
10	LOCCOA	STD	354,978	4,192	49531,089	34,095	72910,447	360,804	0,232
		Ortalama	347,589	4,152	48772,929	33,931	68440,071	356,008	0,227
11	LOCLFDBLFCCOA	STD	0,004	0,002	3,706	0,147	28,95	0,55	0,011
		Ortalama	0,003	0,001	2,156	0,121	28,949	0,472	0,01
12	LOCOA	STD	308,093	3,754	51361,403	34,833	68456,048	309,328	0,225
		Ortalama	299,811	3,719	50333,719	34,629	65059,22	303,638	0,218
13	LOFDBLFCCOA	STD	1,253	0,665	2923,734	2,07	216,031	1,773	0,059
		Ortalama	1,101	0,607	1864,482	1,876	176,062	1,53	0,056
14	LOFDBLFCOA	STD	1,085	0,917	1952,143	3,196	226,622	0,945	0,086
		Ortalama	0,914	0,645	1682,29	2,6	172,942	0,883	0,077
15	LOLFCCOA	STD	20,215	1,227	8444,972	12,091	3310,9	20,682	0,237
		Ortalama	17,859	1,171	7325,199	11,335	2433,544	18,681	0,225
16	OCCLCOA	STD	258,388	2,476	74481,949	33,965	40452,982	294,468	0,171
		Ortalama	254,537	2,43	74088,349	33,36	39413,63	288,054	0,165

Çizelge 4.1. Tek modlu test fonksiyonları (devam)

No	Algoritmalar	Parametreler	Tek modlu test fonksiyonları						
			F1	F2	F3	F4	F5	F6	F7
17	OCCOA	STD	1147,618	8,159	72968,224	42,714	422083,217	1185,745	0,435
		Ortalama	1134,265	8,11	72400,765	42,6	403113,599	1172,033	0,427
18	OCLFDBLFCCOA	STD	0,078	0,009	36,152	0,544	29,636	2,141	0,017
		Ortalama	0,054	0,005	14,832	0,423	29,628	2,017	0,016
19	OCLLFCCOA	STD	0,432	0,022	91,897	1,874	38,428	7,187	0,022
		Ortalama	0,242	0,017	50,713	1,252	36,329	7,111	0,019
20	OCOA	STD	1124,597	6,963	72048,386	43,678	398818,46	1144,904	0,475
		Ortalama	1102,116	6,907	71751,778	43,585	384410,764	1129,722	0,464
21	OFDBLFCCOA	STD	8,842	1,713	9388,369	3,036	826,806	8,881	0,082
		Ortalama	8,307	1,653	6617,746	2,98	546,058	8,556	0,077
22	OFDBLFCOA	STD	6,195	1,308	5114,546	3,251	506,269	7,489	0,087
		Ortalama	5,756	1,273	4214,767	3,097	379,394	6,764	0,082
23	OLFCCOA	STD	86,346	2,853	31520,893	13,975	16402,648	93,935	0,37
		Ortalama	82,295	2,734	27751,751	13,678	13505,51	86,723	0,334

Çizelge 4.2. Tek modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (No: 6)

Fonksiyon (F)	Ortalama Zaman	En İyi Zaman	En Kötü Zaman
F1	1,0389	0,9168	1,6255
F2	1,0810	0,9362	1,7885
F3	1,7954	1,5933	2,7137
F4	1,1617	0,9559	1,8378
F5	1,504	1,1873	2,0637
F6	1,504	1,2889	2,2942
F7	1,8421	1,5884	2,2948

Çizelge 4.3. Tek modlu kıyaslama fonksiyon sonuçları (No: 6)

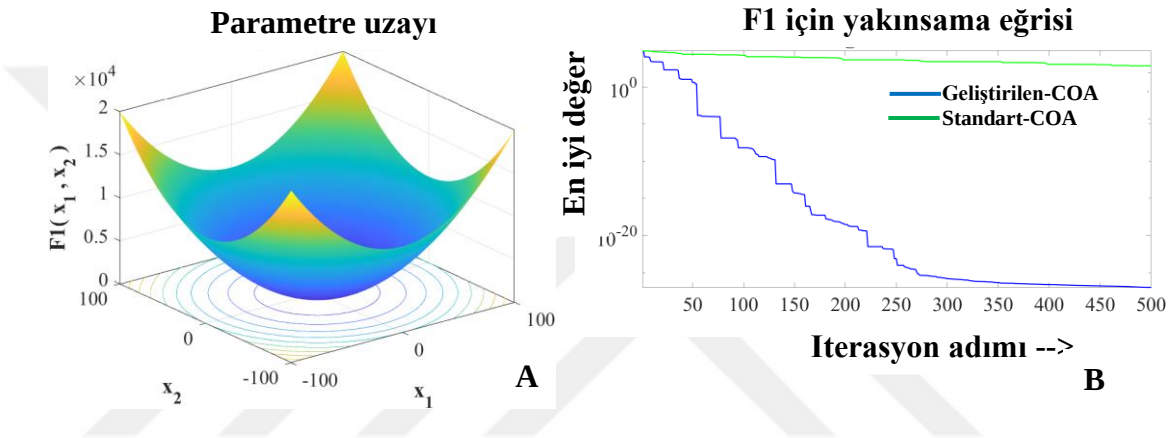
Fonksiyon (F)	Amaç fonksiyon uygunluğu					Wilcoxon rank Sum Test
	En İyi	Ortalama	En Kötü	STD Değer	Median Değer	P Değeri
F1	1,00e-27	2,01e-26	2,42e-26	5,59e-27	2,26e-26	1,2579e-11
F2	4,08e-14	5,23e-14	5,32e-14	2,5e-15	5,32e-14	3,9118e-12
F3	3,21e-27	2,34e-24	7,63e-24	3,33e-24	2,08e-25	1,4314e-11
F4	2,84e-14	0,3080	9,2415	1,6872	2,84e-14	6,3549e-12
F5	21,68	24,68	28,60	1,25	24,53	1,5099e-11
F6	1,108e-6	0,1179	3,5	0,6387	1,154e-5	1,5099e-11
F7	0,0004	0,0030	0,0088	0,0020	0,0025	1,5099e-11

Çizelge 4.4. Tek modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması

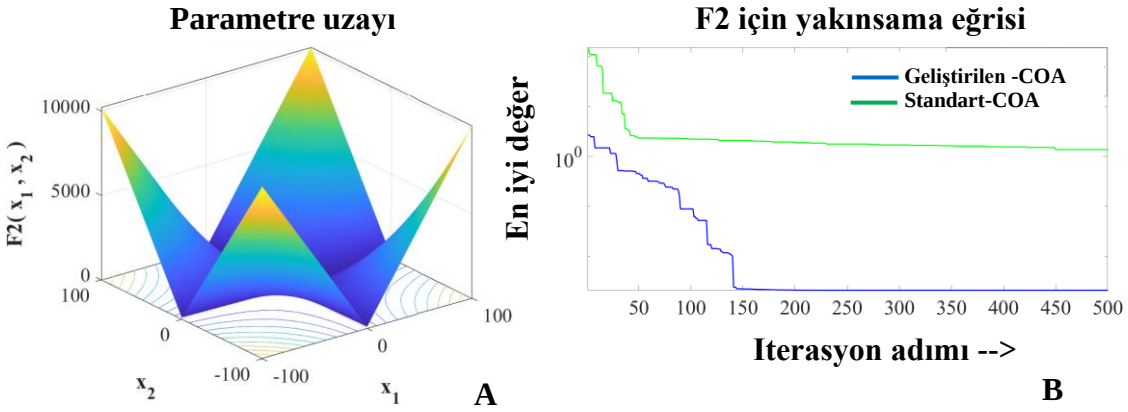
Algoritmalar	Parametreler	Tek modlu test fonksiyonları						
		F1	F2	F3	F4	F5	F6	F7
BA	STD	30085,7	229235	65790,6	64,57	71230824	33661	39,9
	Ortalama	29262,1	56755	62245,4	64,26	63563347	32799	36,74
BOA	STD	0	0	0	0	28,91	5,18	0,0011
	Ortalama	0	0	0	0	28,91	5,16	0,001
CS	STD	1251,6	66,01	11546,5	33,12	270488,4	1334,4	0,469
	Ortalama	1230,8	65,36	11405	33,00	259176,9	1305,9	0,449
DE	STD	0,0006	0,0029	30472,0	13,11	160,91	0,0007	0,0479
	Ortalama	0,0006	0,0029	30226,5	13,05	157,82	0,0006	0,0472
EHO	STD	0,00086	0,0134	0,00132	0,0103	28,77	2,832	3,08E-5
	Ortalama	0,00086	0,0134	0,00127	0,0103	28,77	2,823	2,14E-5
EBO with CMAR	STD	0	0	0	8,4E-5	2,478	0	0,0043
	Ortalama	0	0	0	6,8E-5	2,143	0	0,0036
COA	STD	989,5	6,75	52172,5	41,30	366910,2	1005,5	0,452
	Ortalama	972,4	6,72	51956,0	41,19	348563,5	992,51	0,442

Çizelge 4.4. Tek modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması (devam)

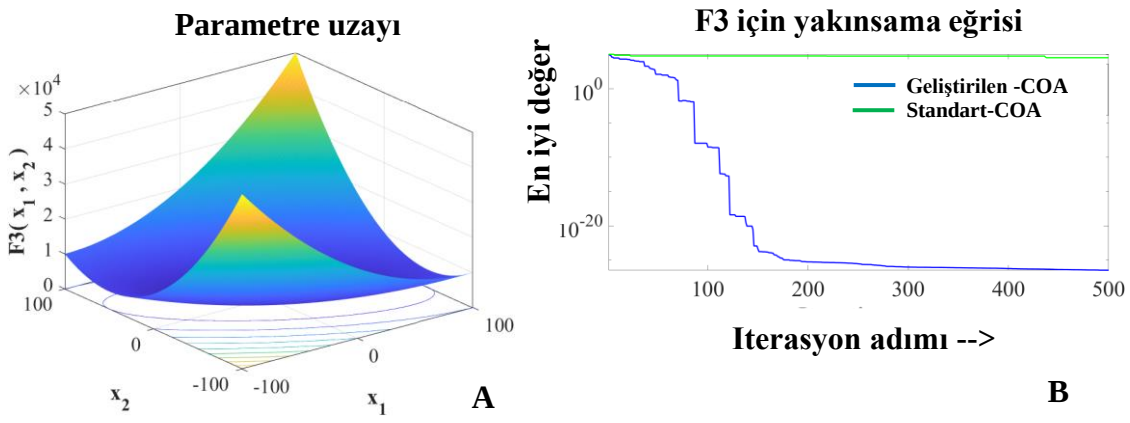
Algoritmalar	Parametreler	Tek modlu test fonksiyonları						
		F1	F2	F3	F4	F5	F6	F7
MBO	STD	29176	65,69	57697,6	37,80	1147E+5	30222	163,45
	Ortalama	18320	13,88	45885,8	28,93	62041667	19068	140,23
MFO	STD	2582	36,81	15208,3	40,99	36857,75	1807,6	3,66
	Ortalama	667,8	30,47	12599,6	39,73	15896,08	330,89	1,95
MS	STD	0	9,1E-8	0	4,7E-8	26,45	0,0459	3,12E-5
	Ortalama	0	6,7-8	0	3,3E-8	26,45	0,0430	2,49E-5
PSO	STD	1,86	46,86	25807,6	25,51	46577,32	1807,5	3,78
	Ortalama	0,98	42,90	24000,5	24,80	24844,42	330,75	1,88
WFS	STD	81,53	2,26	172,5	3,57	1576,76	49,68	0,017
	Ortalama	58,25	2,18	137,0	3,17	948,15	35,24	0,0132
Öneri GCOA6 'no 6'	STD	0	0	0	0	24,58	0,0457	0,0037
	Ortalama	0	0	0	0	24,54	0,0084	0,0029



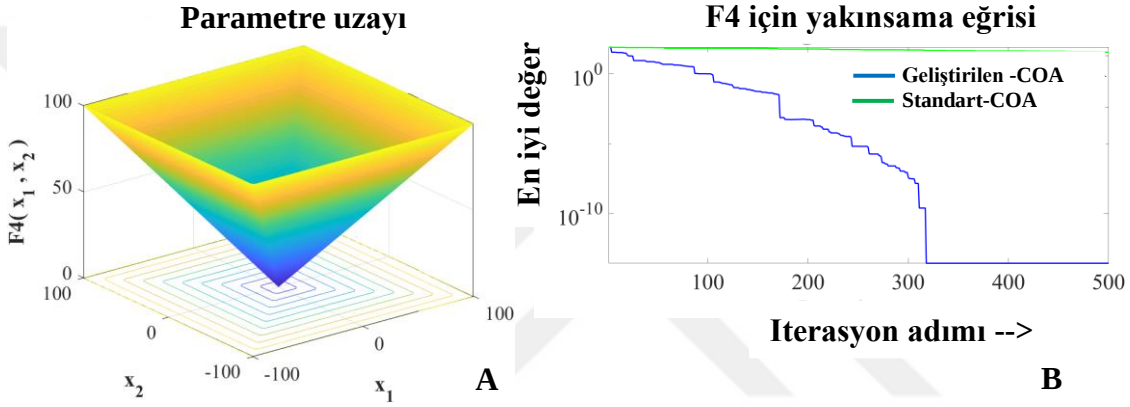
Şekil 4.9. COA ve GCOA 'no:6' için F1 test fonksiyonu şematiği.
A) F1 test fonksiyonu parametre uzayı **B)** F1 test fonksiyonu için yakınsama eğrisi



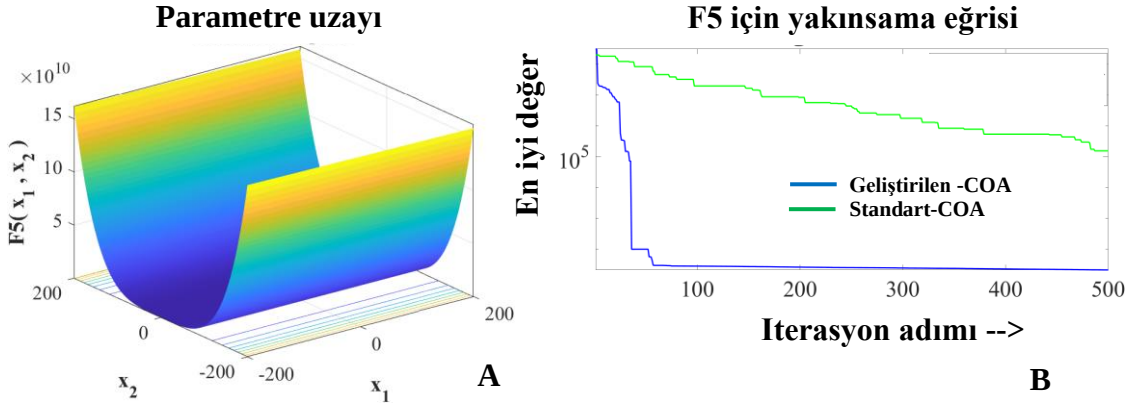
Şekil 4.10. COA ve GCOA 'no:6' için F2 test fonksiyonu şematiği.
A) F2 test fonksiyonu parametre uzayı **B)** F2 test fonksiyonu için yakınsama eğrisi



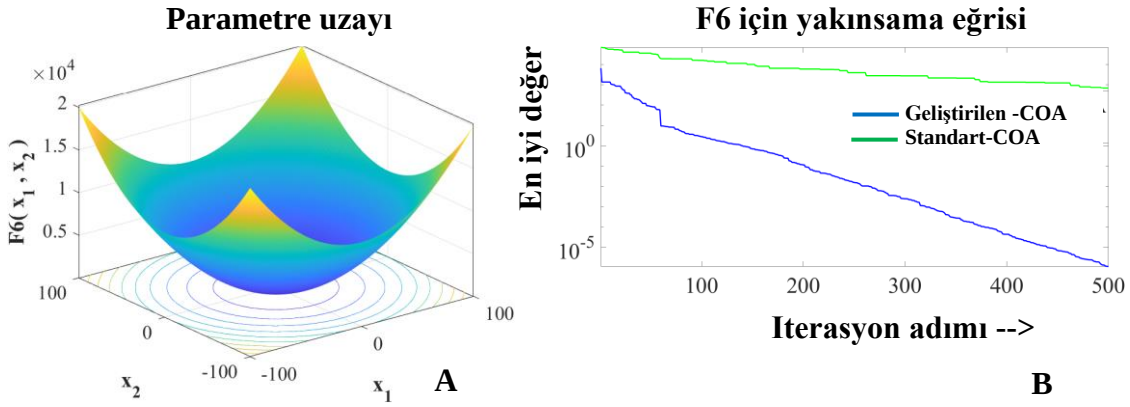
Şekil 4.11. COA ve GCOA 'no:6' için F3 test fonksiyonu şematiği.
A) F3 test fonksiyonu parametre uzayı **B)** F3 test fonksiyonu için yakınsama eğrisi



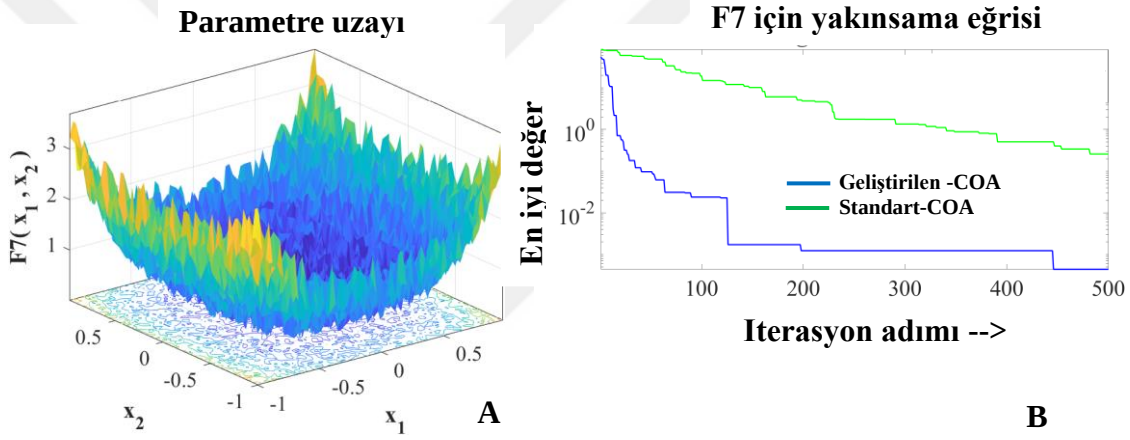
Şekil 4.12. COA ve GCOA 'no:6' için F4 test fonksiyonu şematiği.
A) F4 test fonksiyonu parametre uzayı **B)** F4 test fonksiyonu için yakınsama eğrisi



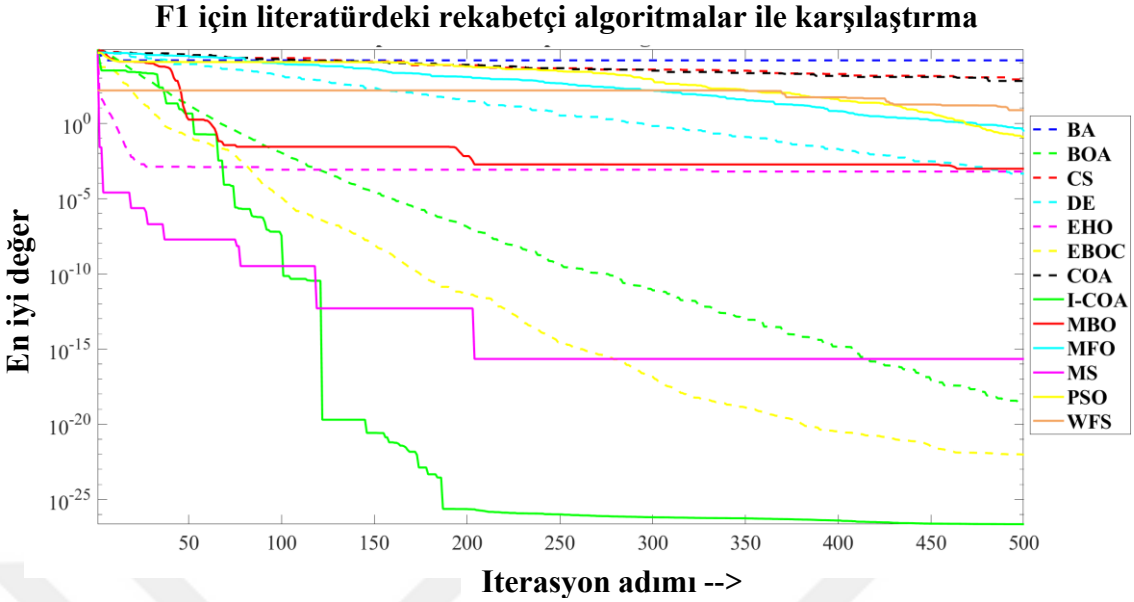
Şekil 4.13. COA ve GCOA 'no:6' için F5 test fonksiyonu şematiği.
A) F5 test fonksiyonu parametre uzayı **B)** F5 test fonksiyonu için yakınsama eğrisi



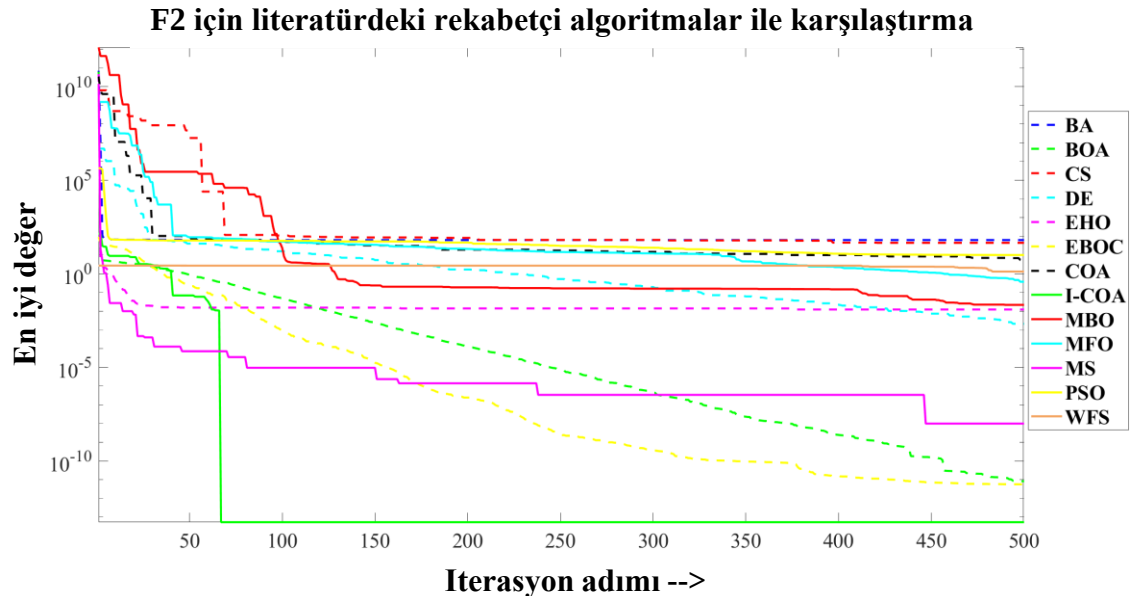
Şekil 4.14. COA ve GCOA 'no:6' için F6 test fonksiyonu şematiği.
A) F6 test fonksiyonu parametre uzayı **B)** F6 test fonksiyonu için yakınsama eğrisi



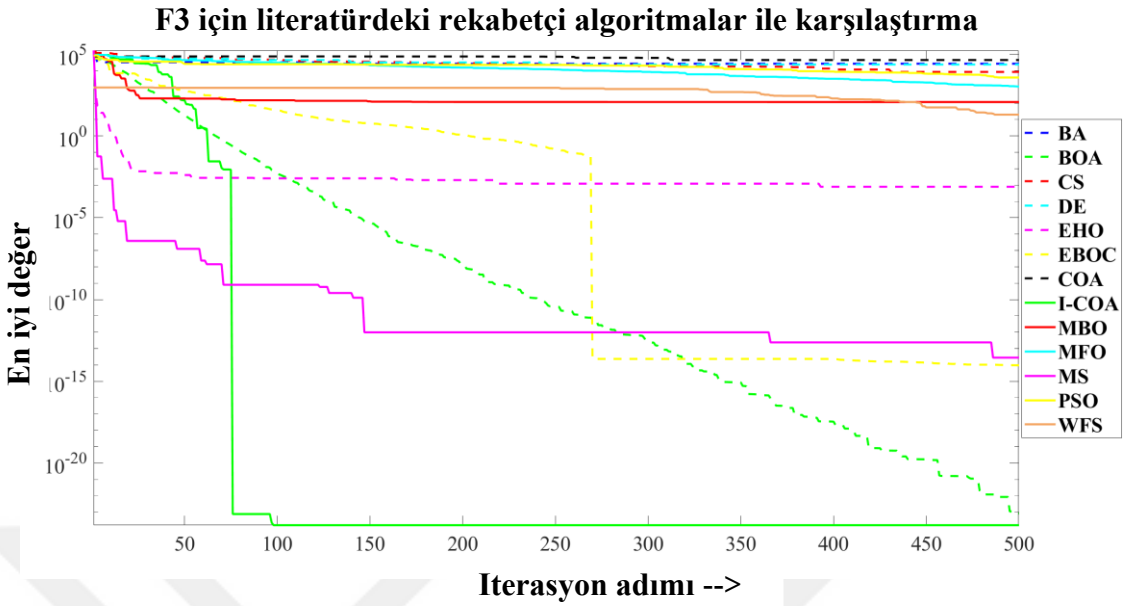
Şekil 4.15. COA ve GCOA 'no:6' için F7 test fonksiyonu şematiği.
A) F7 test fonksiyonu parametre uzayı **B)** F7 test fonksiyonu için yakınsama eğrisi



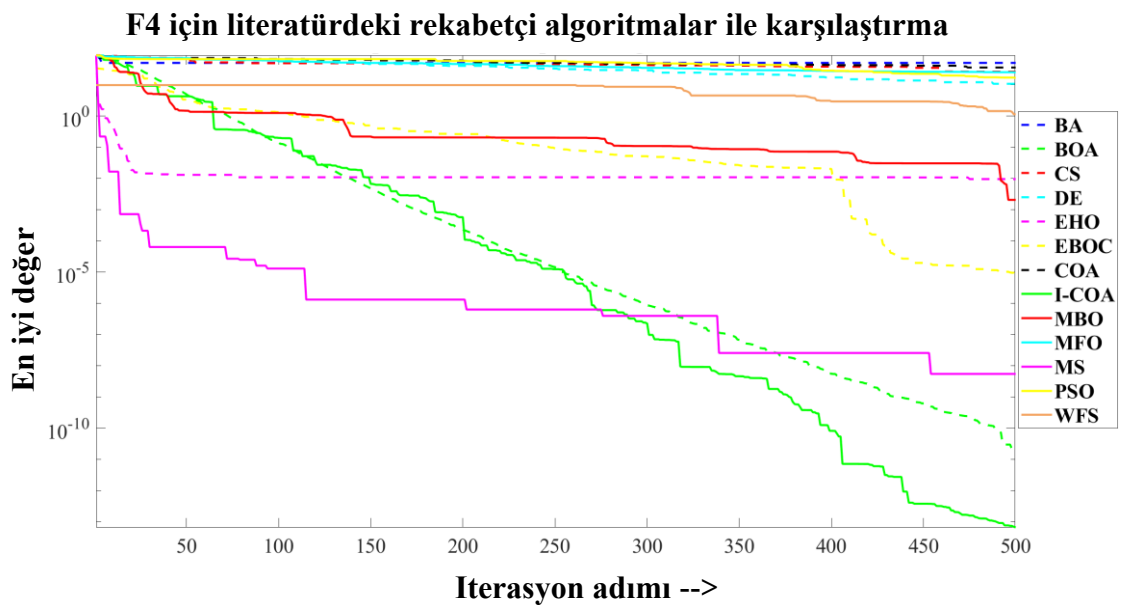
Şekil 4.16. F1 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



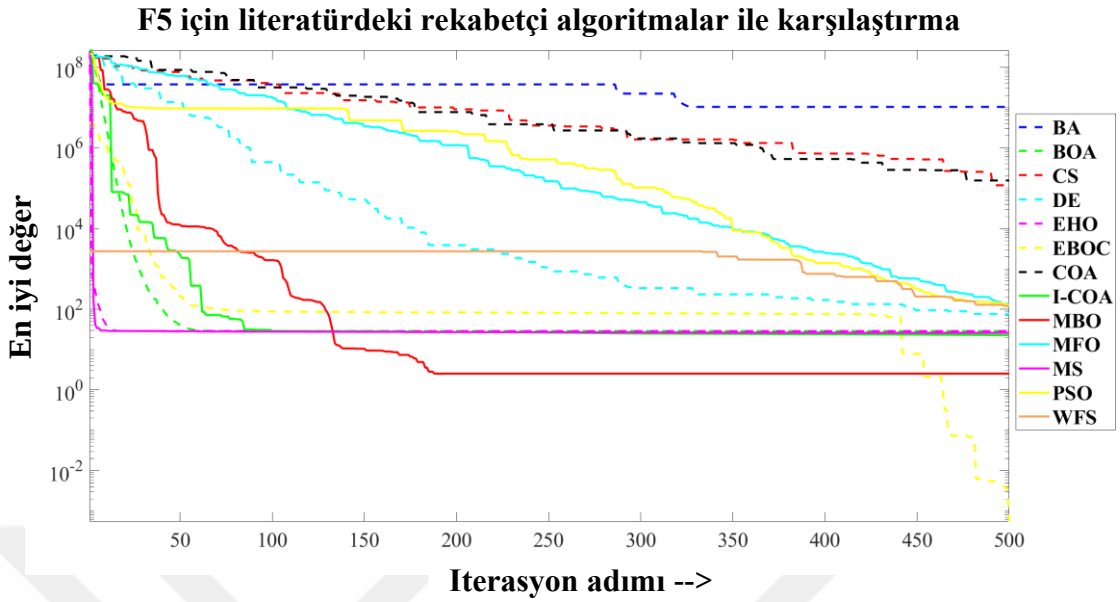
Şekil 4.17. F2 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



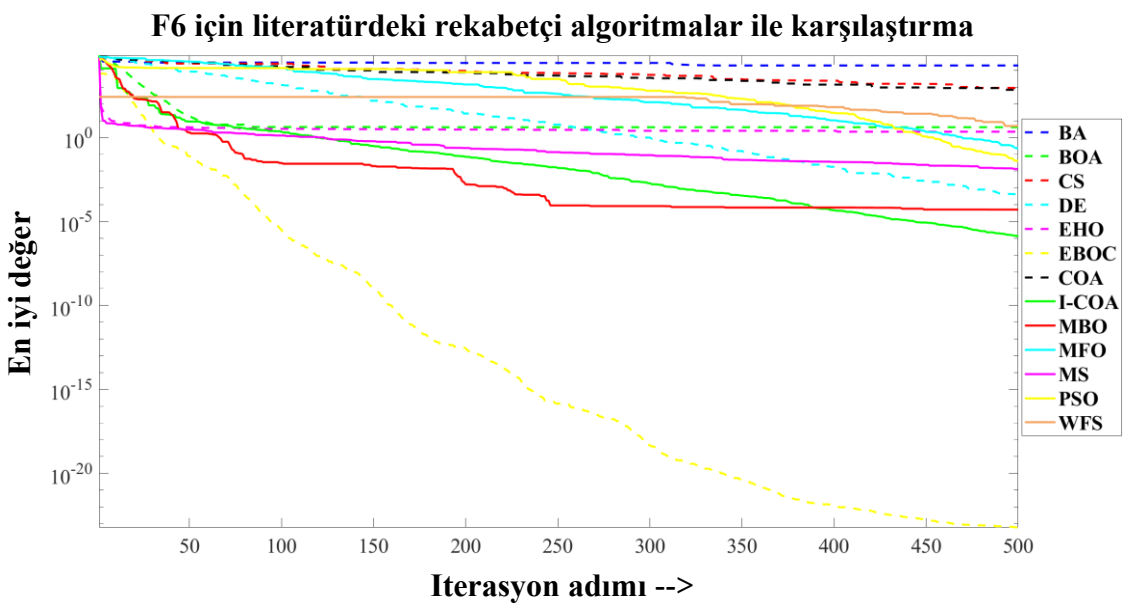
Şekil 4.18. F3 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



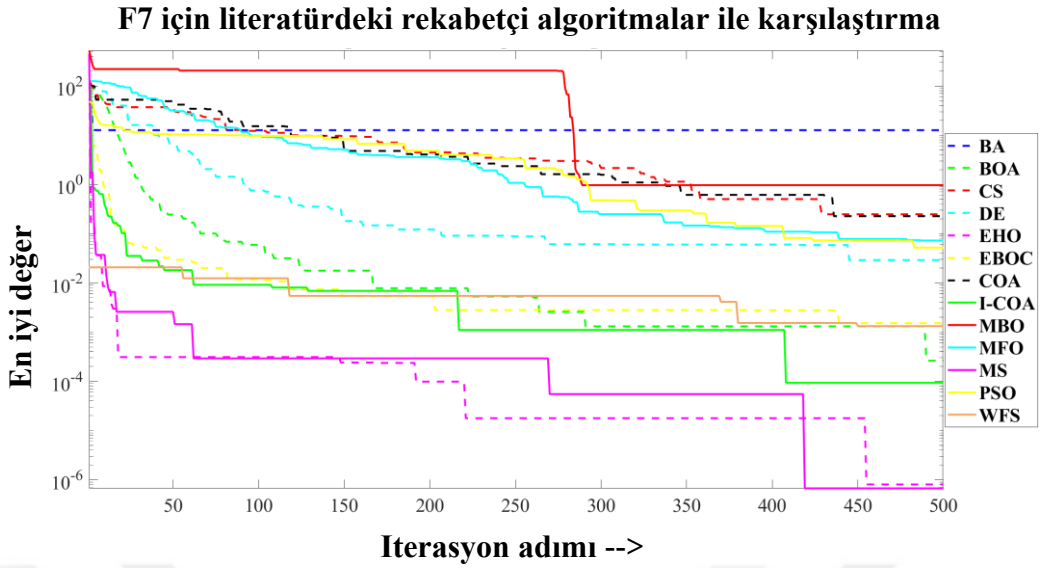
Şekil 4.19. F4 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



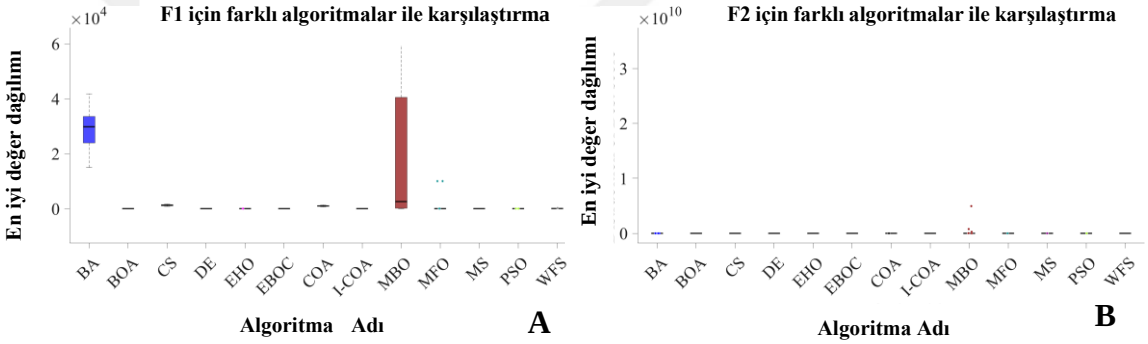
Şekil 4.20. F5 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



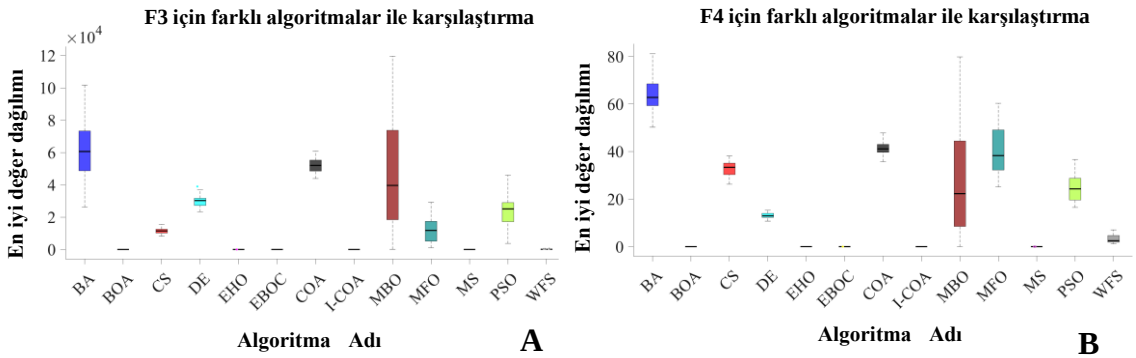
Şekil 4.21. F6 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



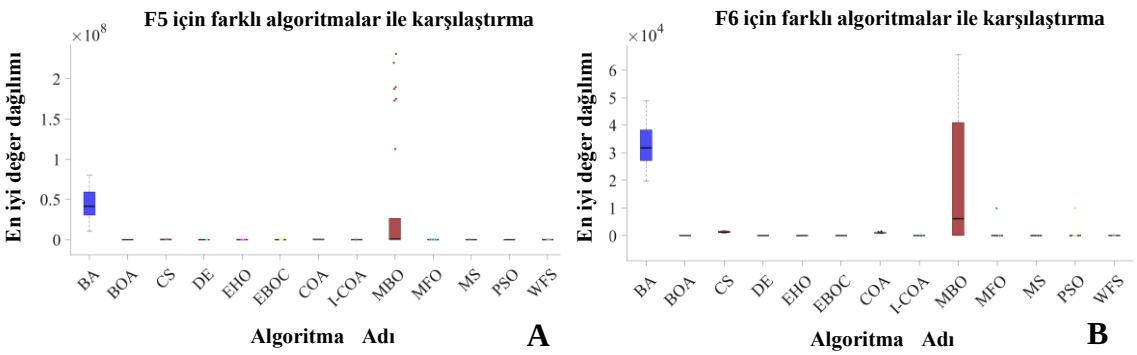
Şekil 4.22. F7 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



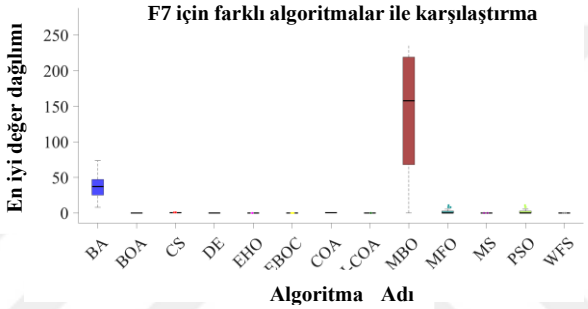
Şekil 4.23. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F1 fonksiyonu için karşılaştırma **B)** F2 fonksiyonu için karşılaştırma



Şekil 4.24. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F3 fonksiyonu için karşılaştırma **B)** F4 fonksiyonu için karşılaştırma



Şekil 4.25. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F5 fonksiyonu için karşılaştırma **B)** F6 fonksiyonu için karşılaştırma



Şekil 4.26. Rakip algoritmalarla karşılaştırılmalı tek modlu F7 kıyaslama fonksiyonunun değer dağılım kutu grafiği.

Çizelge 4.5. Sabit çok modlu test fonksiyonları

No	Algoritmalar	Parametreler	Sabit çok modlu test fonksiyonları						
			F8	F9	F10	F11	F12	F13	F14
1	CCOA	STD	11240,952	0,536	0,371	0,266	0,562	0,116	0,998
		Ortalama	-11237,203	0,452	0,247	0,199	0,393	0,101	0,998
2	CLCCOA	STD	11151,203	0,417	0,366	0,257	0,539	0,17	0,998
		Ortalama	-11146,181	0,326	0,227	0,211	0,332	0,134	0,998
3	CLCOA	STD	12070,998	27,305	11,403	1,565	4,356	37,585	0,998
		Ortalama	-12070,246	27,113	10,902	1,554	4,067	31,909	0,998
4	COA	STD	11994,654	34,405	18,209	10,2	9373,625	293564,52	0,998
		Ortalama	-11993,851	34,253	18,187	10,048	5431,438	253434,08	0,998
5	FDBLFCOA	STD	9699,106	19,194	2,317	0,019	0,352	3,476	0,998
		Ortalama	-9676,218	18,223	2,03	0,013	0,223	1,694	0,998
6	LCLFDBLFCCOA	STD	8959,542	0	0	0	0	0,621	0,998
		Ortalama	-8922,478	0	0	0	0	0,261	0,998
7	LCOA	STD	12079,004	30,957	6,797	3,539	36,101	15575,53	0,998
		Ortalama	-12078,39	30,613	6,742	3,498	33,642	11479,35	0,998
8	LFDBLFCCOA	STD	9613,468	51,75	1,108	0,958	1,471	6,575	0,998
		Ortalama	-9587,029	48,69	0,977	0,953	1,208	3,938	0,998
9	LFDBLFCOA	STD	10197,419	40,338	1,427	0,967	2,171	6,427	0,998
		Ortalama	-10176,06	39,487	1,328	0,963	1,579	4,482	0,998
10	LOCCOA	STD	8744,236	35,132	7,167	4,203	36,276	11963,66	0,998
		Ortalama	-8642,046	34,69	6,95	4,137	33,886	8727,931	0,998
11	LOCLFDBLFC COA	STD	8273,208	0	0,011	0,013	0,032	0,155	0,998
		Ortalama	-8175,329	0	0,007	0,007	0,017	0,14	0,998

Çizelge 4.5. Sabit çok modlu test fonksiyonları (devam)

No	Algoritmalar	Parametreler	Sabit çok modlu test fonksiyonları						
			F8	F9	F10	F11	F12	F13	F14
12	LOCOA	STD	8726,088	32,216	7,257	3,859	29,16	9795,09	0,998
		Ortalama	-8669,765	31,866	6,909	3,836	27,781	6102,33	0,998
13	LOFDBLFCCOA	STD	8041,082	36,591	0,855	0,8	0,16	0,476	0,998
		Ortalama	-7988,647	34,726	0,75	0,786	0,109	0,434	0,998
14	LOFDBLFCOA	STD	8747,541	39,509	1,064	0,645	0,203	0,367	0,998
		Ortalama	-8619,026	37,672	0,923	0,626	0,131	0,328	0,998
15	LOLFCCOA	STD	8672,065	50,612	1,91	1,122	2,013	79,225	0,998
		Ortalama	-8589,19	49,066	1,828	1,12	1,679	36,553	0,998
16	OCCLCOA	STD	8353,981	0,304	4,675	3,408	15,326	961,26	0,998
		Ortalama	-8237,541	0,114	4,651	3,379	15,053	603,395	0,998
17	OCCOA	STD	8502,473	42,417	12,561	11,414	24799,077	403081,43	0,998
		Ortalama	-8450,715	41,799	12,511	11,325	16597,77	375852,61	0,998
18	OCLFDBLFCCOA	STD	8184,442	0	0,03	0,056	0,179	0,753	0,998
		Ortalama	-8131,786	0	0,021	0,036	0,142	0,712	0,998
19	OCLLFCCOA	STD	8162,22	0	0,074	0,191	0,384	2,325	0,998
		Ortalama	-8090,318	0	0,049	0,142	0,355	2,288	0,998
20	OCOA	STD	8504,745	37,787	18,082	11,238	7295,983	384285,64	0,998
		Ortalama	-8437,5	37,287	18,042	11,119	5190,722	340633,10	0,998
21	OFDBLFCCOA	STD	8274,471	50,019	4,945	1,055	0,447	1,97	0,998
		Ortalama	-8206,432	47,527	4,122	1,055	0,39	1,839	0,998
22	OFDBLFCOA	STD	8764,603	45,373	15,299	1,043	0,428	1,848	0,998
		Ortalama	-8641,721	43,208	13,716	1,043	0,338	1,736	0,998
23	OLFCCOA	STD	8319,627	59,78	9,908	1,717	5,978	123,93	0,998
		Ortalama	-8266,365	57,994	8,835	1,684	4,958	44,679	0,998

Çizelge 4.6. Sabit çok-modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (GCOA6)

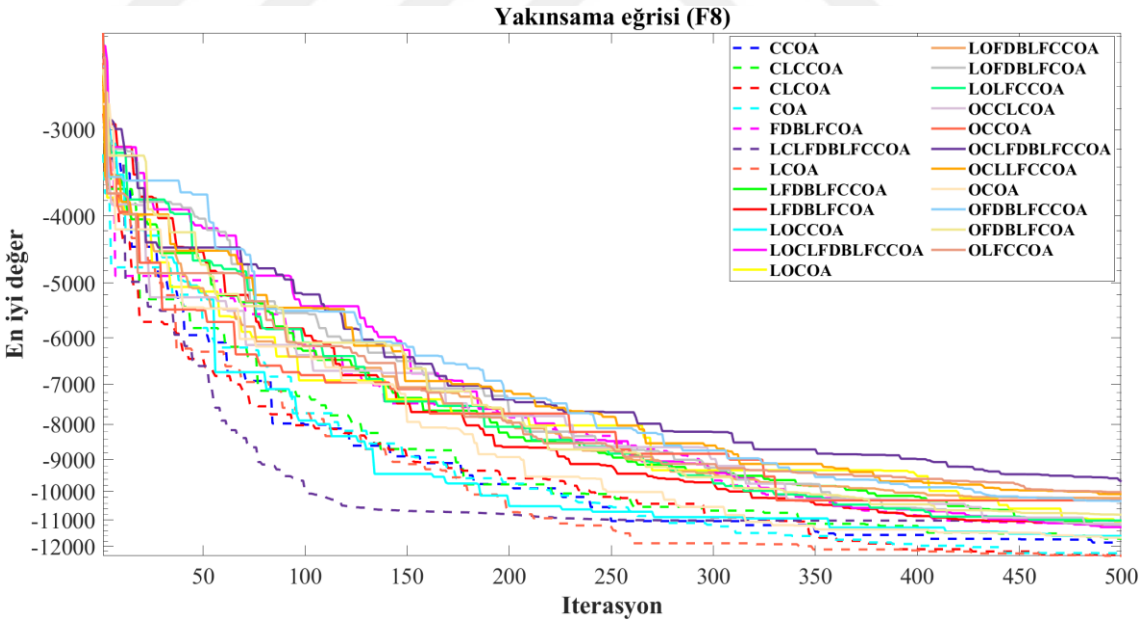
Fonksiyon (F)	Ortalama Zaman	En İyi Zaman	En Kötü Zaman
F8	1,6803	1,3862	2,1958
F9	1,2737	1,0594	1,8790
F10	1,2875	1,1164	1,8286
F11	1,4051	1,1439	1,9386
F12	2,5919	2,2687	3,2294
F13	2,5807	2,2846	2,9752

Çizelge 4.7. Sabit çok-modlu benchmark test fonksiyonları için sonuçlar (GCOA6)

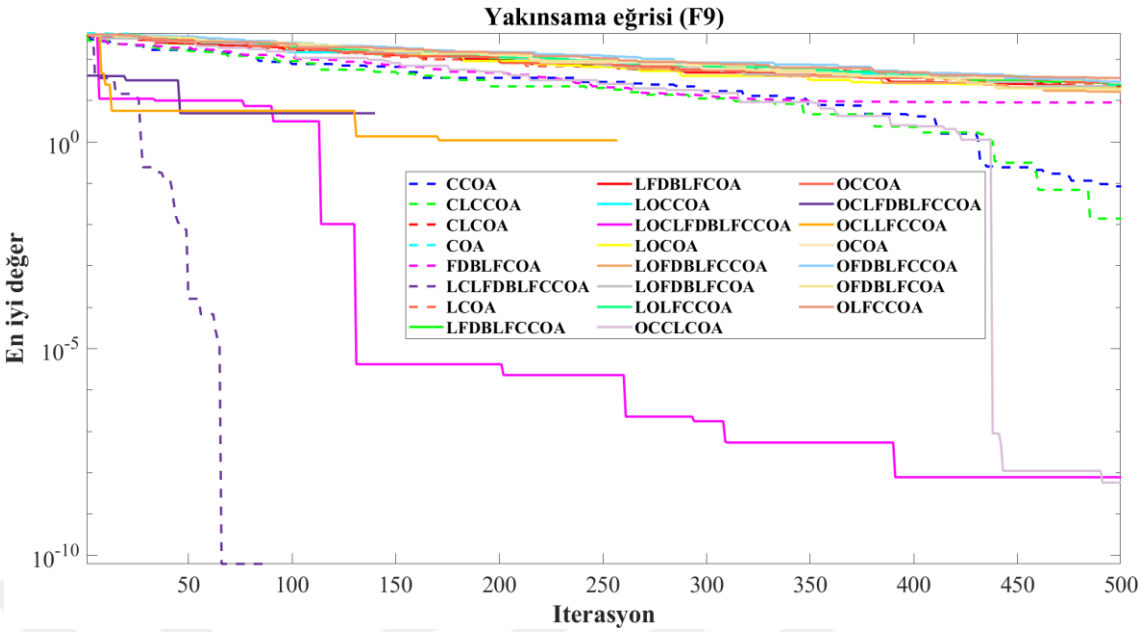
Fonksiyon (F)	Amaç fonksiyon uygunluğu					Wilcoxon rank Sum Test
	En İyi	Ortalama	En Kötü	STD Değer	Median Değer	P Değeri
F8	-10654,5	-8818,6	-7305,4	948,18	-8776,3	1,509e-11
F9	0	0	0	0	0	6,058e-13
F10	2,5e-14	2,8e-14	2,9e-14	1,08e-15	2,9e-14	9,310e-13
F11	0	0	0	0	0	6,058e-13
F12	6,0e-09	6,9e-08	3,8e-07	7,65e-08	3,6e-08	1,509e-11
F13	2,19e-08	0,190	1,82	0,4242	0,0109	1,509e-11

Çizelge 4.8. Sabit çok-modlu Benchmark test fonksiyonları için sonuçların karşılaştırılması (GCOA6)

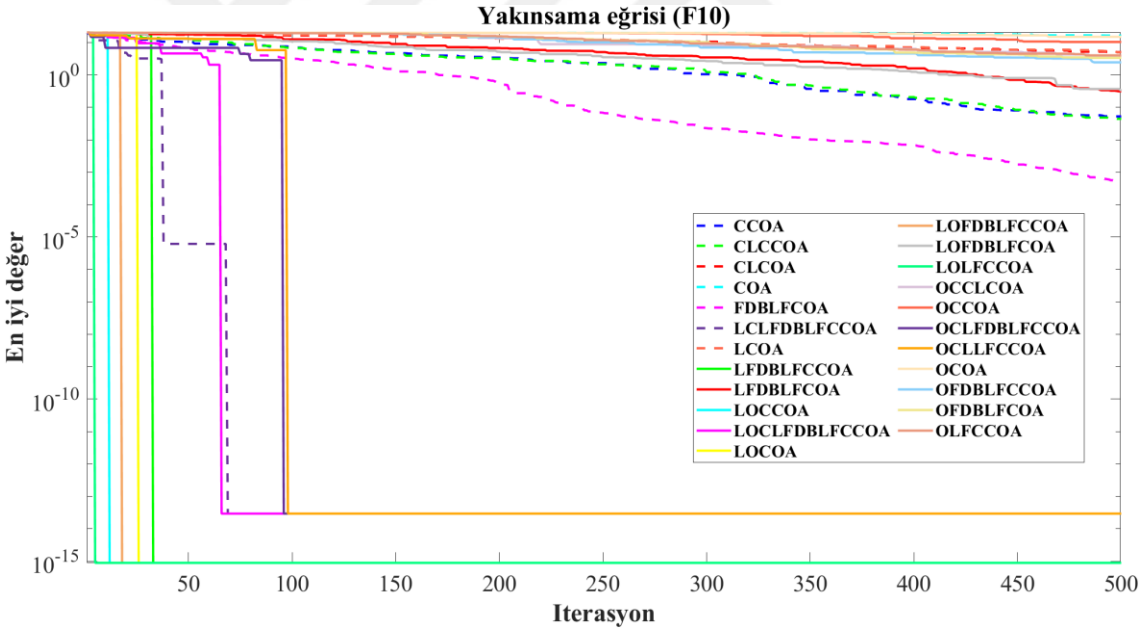
Algoritmalar	Parametreler	Çok Modlu test fonksiyonları					
		F8	F9	F10	F11	F12	F13
BA	STD	7511,34	343,4	19,59	279,36	7,32e7	1,65e8
	Ortalama	-6563,5	342,6	19,58	270,72	5,91e7	1,39e8
BOA	STD	3886,51	3,07e-05	0	0	0,502	2,672
	Ortalama	-3870,14	5,6e-06	0	0	0,493	2,653
CS	STD	7821,69	147,9	16,09	12,12	241,9	1,13e5
	Ortalama	-7819,75	147,5	16,06	11,89	113,88	9,2e4
DE	STD	9179,11	83,87	0,0071	0,0089	8,191e-05	0,00043
	Ortalama	-9175,73	83,64	0,0070	0,0065	7,88e-05	0,00042
EHO	STD	3952,66	0,00058	0,0072	0,0007	0,774	2,9199
	Ortalama	-3937,33	0,00058	0,00726	0,0007	0,771	2,9141
EBO with CMAR	STD	12569,4	0	0	0	0,026	0,002
	Ortalama	-12569,4	0	0	0	0,0069	0,00036
COA	STD	12013,5	33,67	18,566	10,57	8870,85	3,33e5
	Ortalama	-12012,5	33,36	18,555	10,45	5230,83	2,86e5
MBO	STD	8756,60	217,6	15,588	269,9	1,98e8	5,46e8
	Ortalama	-8301,75	172,6	13,230	182,08	1,01e8	3,38e8
MFO	STD	8756,30	127,7	13,009	23,36	4,67e7	4,8334
	Ortalama	-8714,88	122,7	9,835	6,73	8,53e6	3,9236
MS	STD	6488,22	0	5,59e-8	0	0,0054	0,4242
	Ortalama	-6446,91	0	3,76e-8	0	0,0050	0,4021
PSO	STD	8530,01	146,1	10,7	16,62	5,217	12,072
	Ortalama	-8499,35	142,4	6,63	3,60	4,354	8,9533
WFS	STD	11994,1	32,39	3,41	1,59	1,45	3,6749
	Ortalama	-11092,04	21,87	3,28	1,48	1,133	2,2495
Geliştirilmiş COA GCOA6 'no 6'	STD	9189,46	0	0	0	2,80e-07	0,4365
	Ortalama	-9154,77	0	0	0	1,54e-07	0,1351



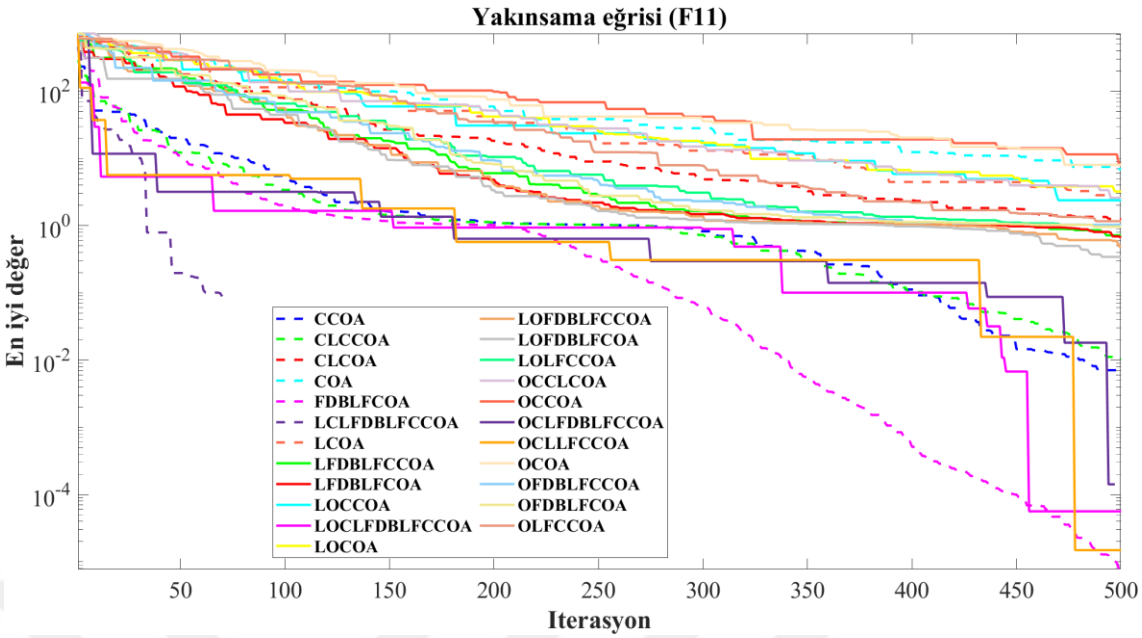
Şekil 4.27. F8 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



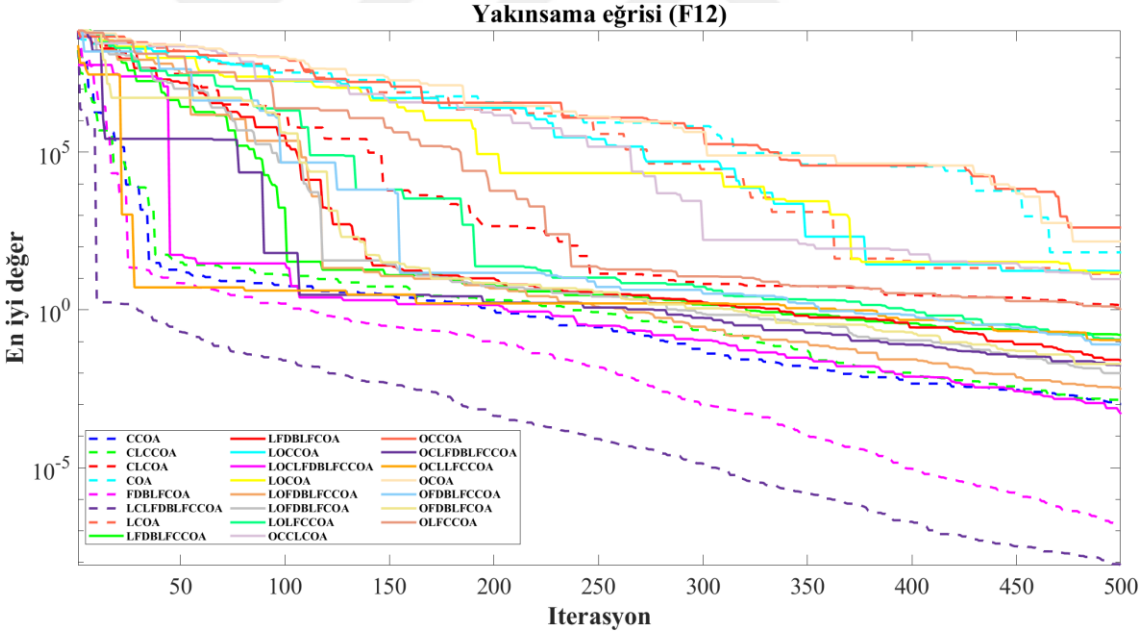
Şekil 4.28. F9 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



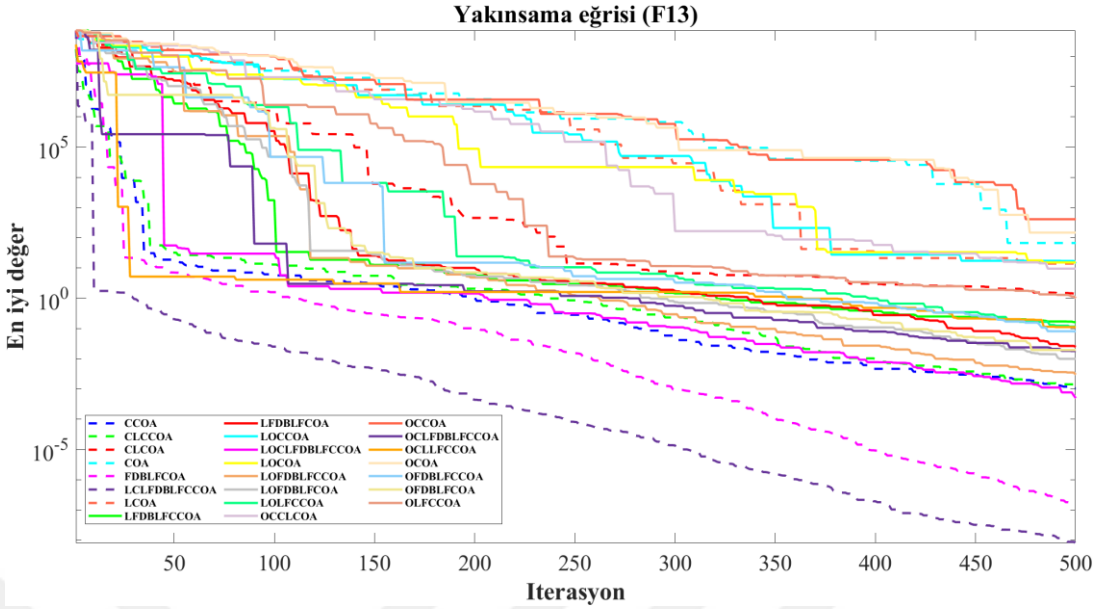
Şekil 4.29. F10 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



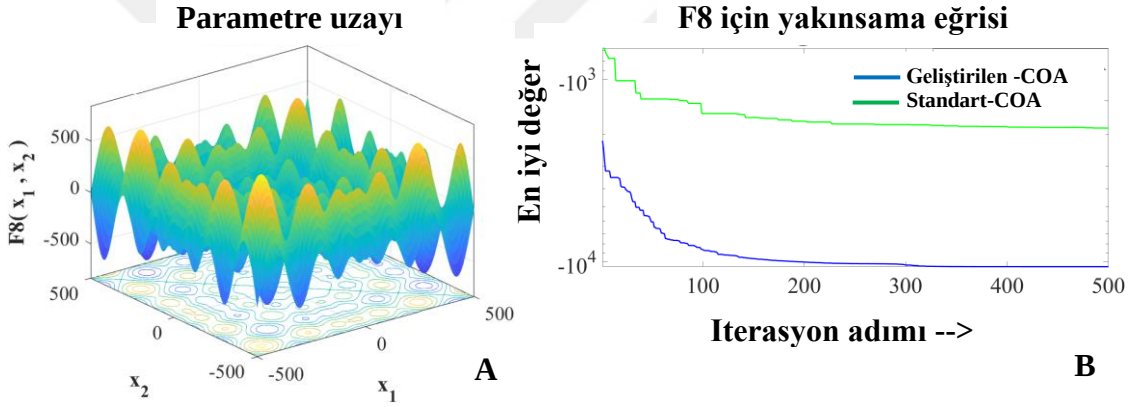
Şekil 4.30. F11 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



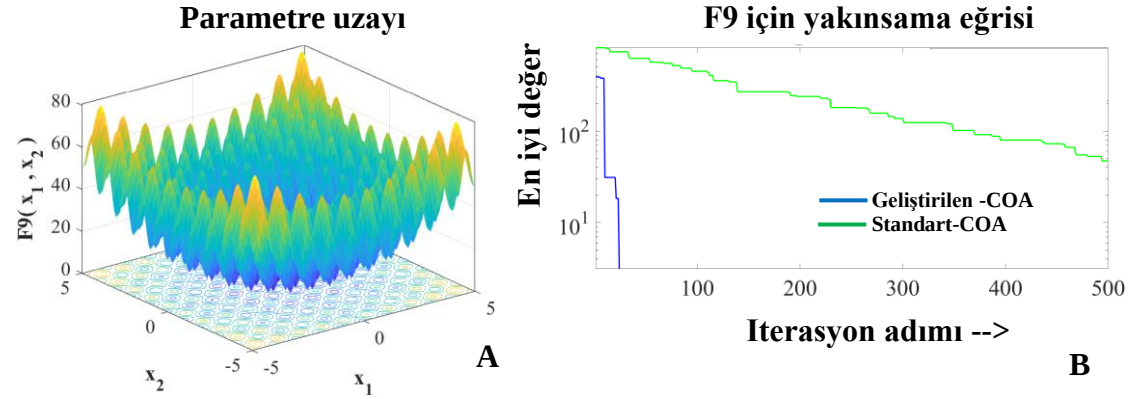
Şekil 4.31. F12 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



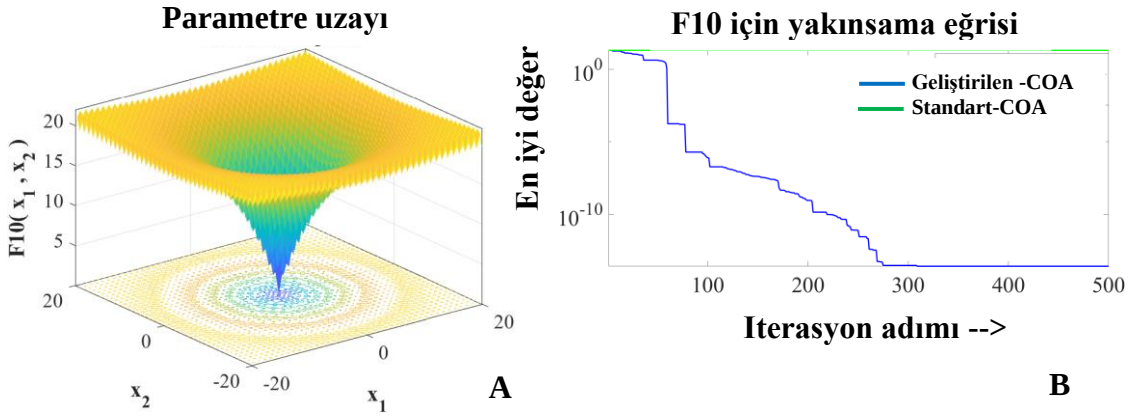
Şekil 4.32. F13 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



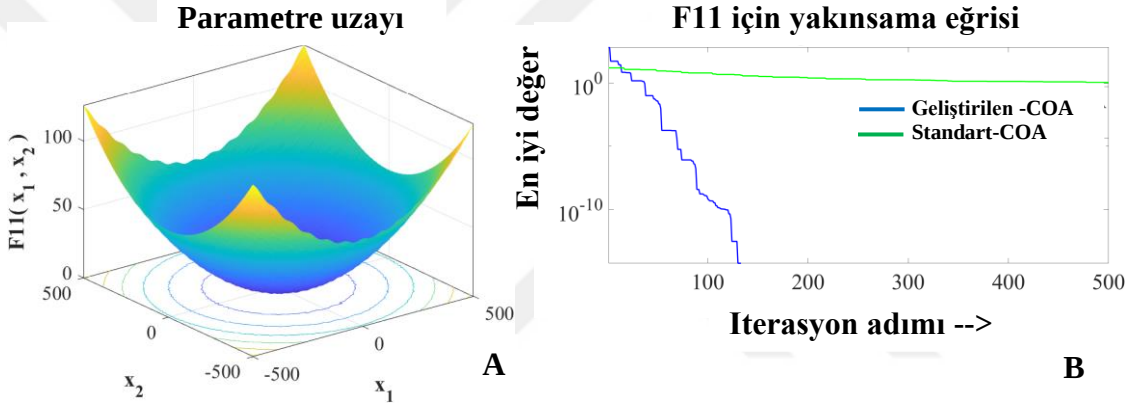
Şekil 4.33. COA ve GCOA 'no:6' için F8 test fonksiyonu şematiği.
A) F8 test fonksiyonu parametre uzayı **B)** F8 test fonksiyonu için yakınsama eğrisi



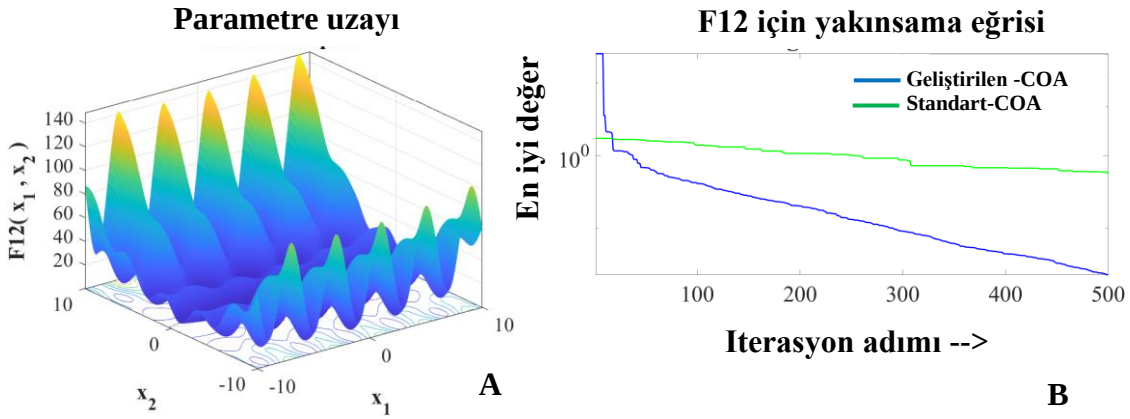
Şekil 4.34. COA ve GCOA 'no:6' için F9 test fonksiyonu şematiği.
A) F9 test fonksiyonu parametre uzayı **B)** F9 test fonksiyonu için yakınsama eğrisi



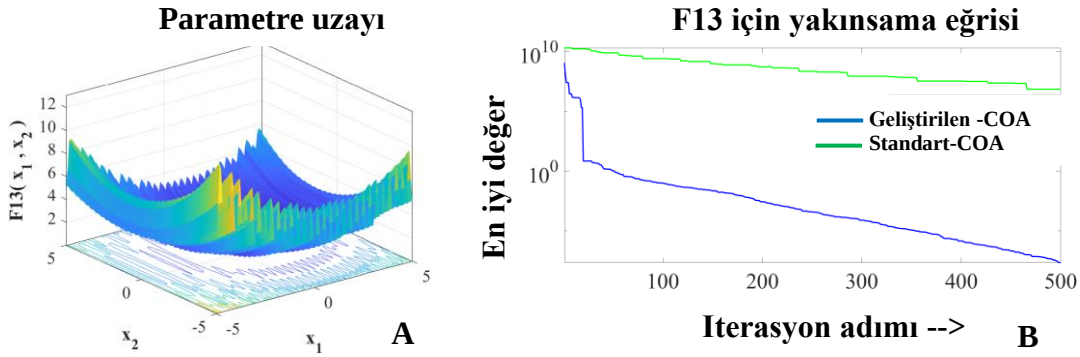
Şekil 4.35. COA ve GCOA 'no:6' için F10 test fonksiyonu şematiği.
A) F10 test fonksiyonu parametre uzayı **B)** F10 test fonksiyonu için yakınsama eğrisi



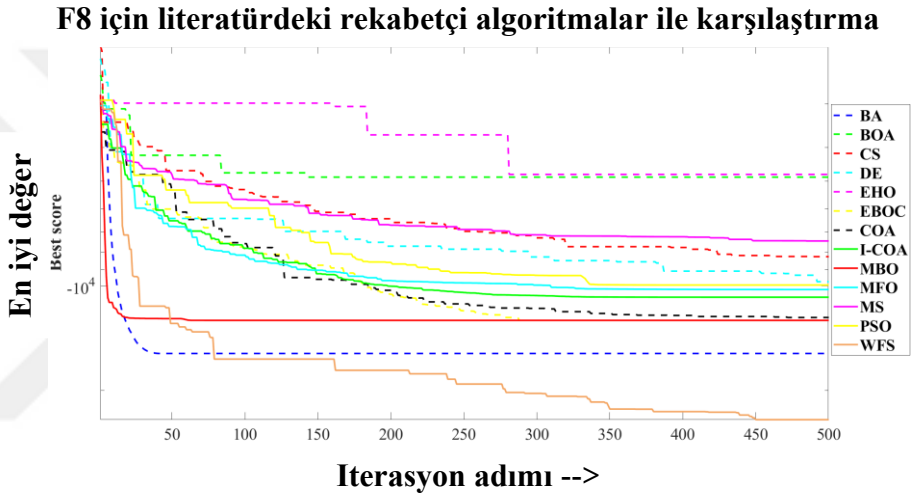
Şekil 4.36. COA ve GCOA 'no:6' için F11 test fonksiyonu şematiği.
A) F11 test fonksiyonu parametre uzayı **B)** F11 test fonksiyonu için yakınsama eğrisi



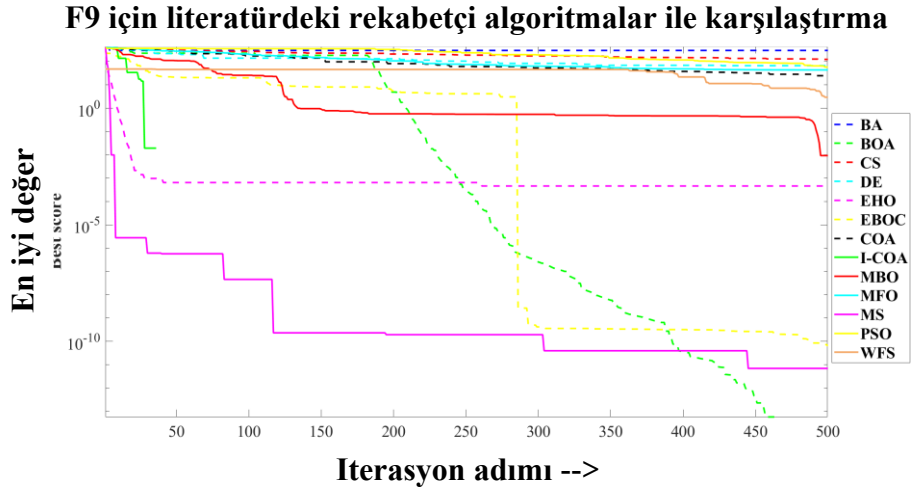
Şekil 4.37. COA ve GCOA 'no:6' için F12 test fonksiyonu şematiği.
A) F12 test fonksiyonu parametre uzayı **B)** F12 test fonksiyonu için yakınsama eğrisi



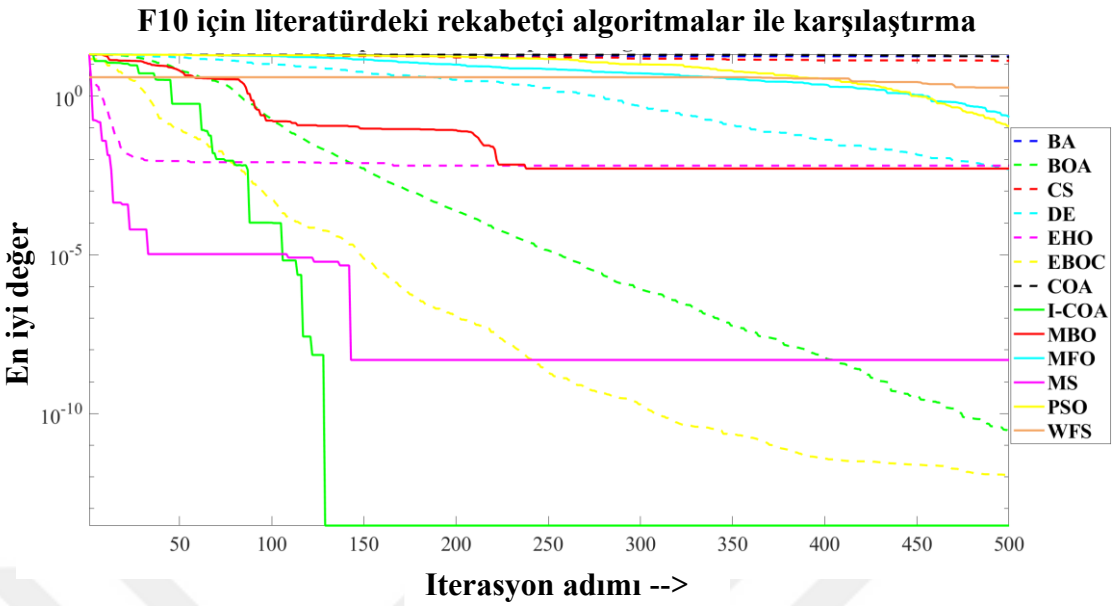
Şekil 4.38. COA ve GCOA 'no:6' için F13 test fonksiyonu şematiği.
A) F13 test fonksiyonu parametre uzayı **B)** F13 test fonksiyonu için yakınsama eğrisi



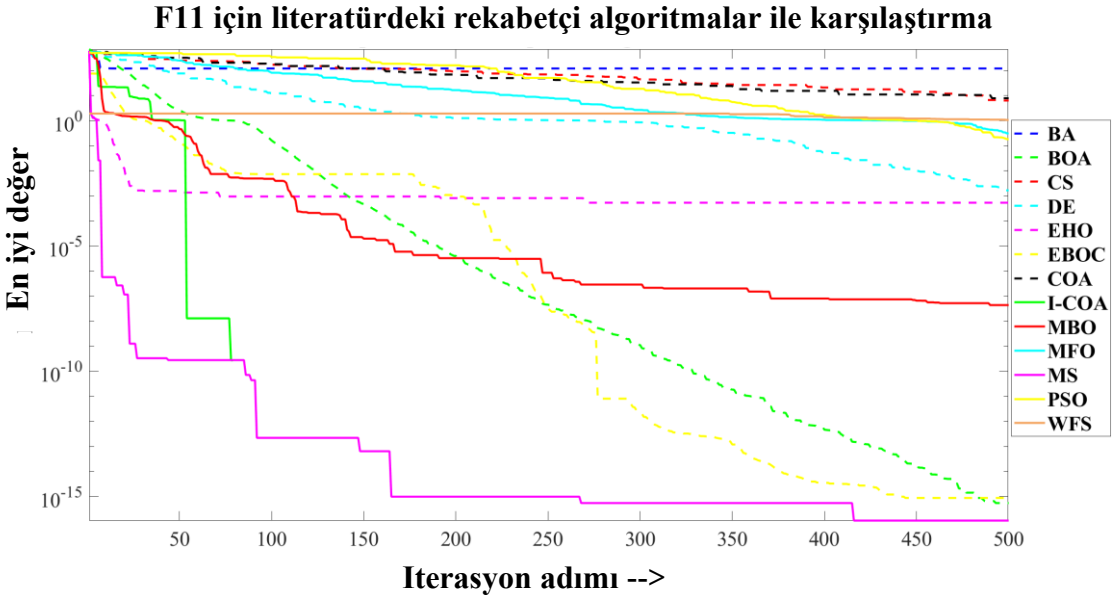
Şekil 4.39. F8 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



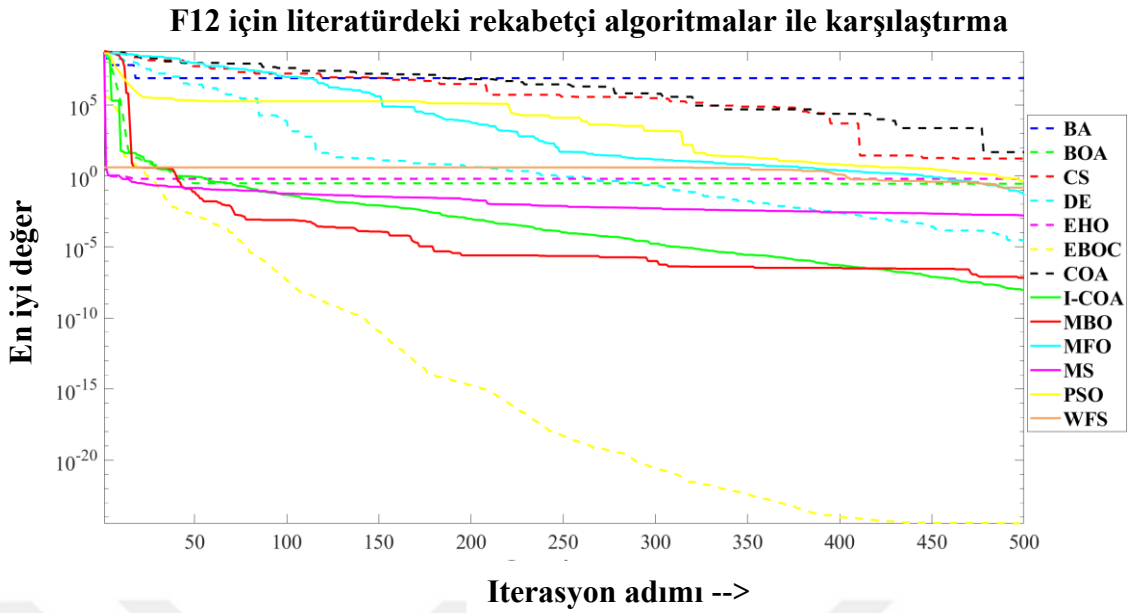
Şekil 4.40. F9 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



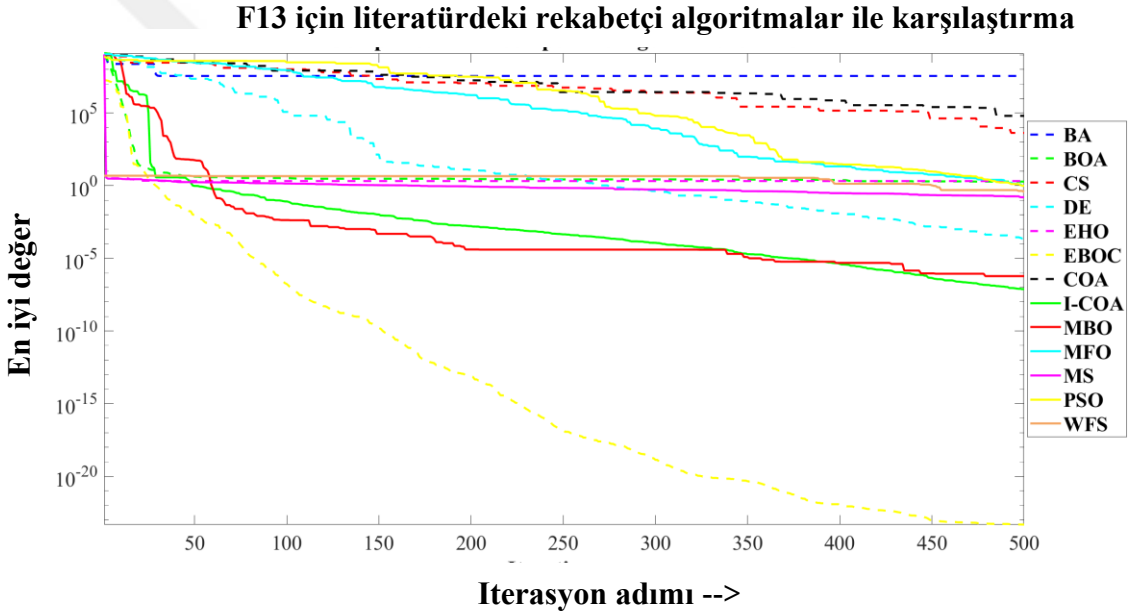
Şekil 4.41. F10 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



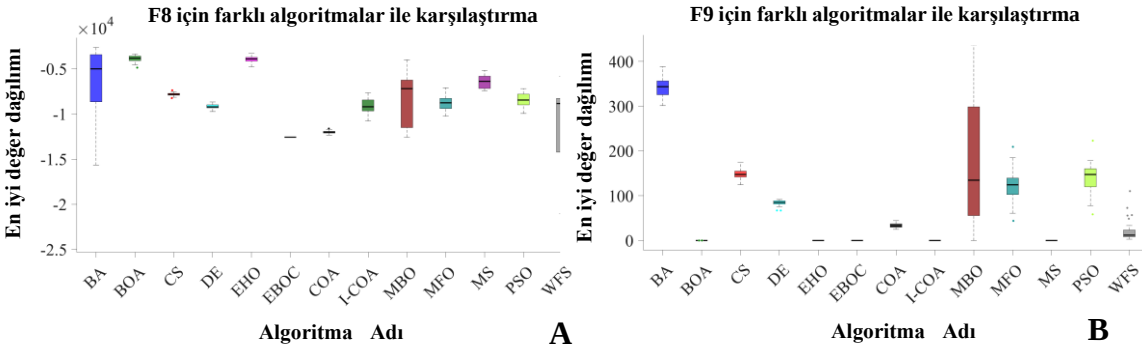
Şekil 4.42. F11 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



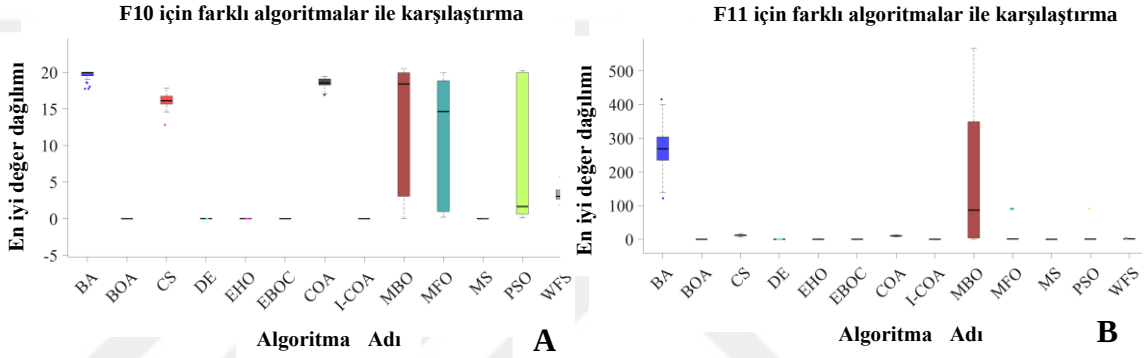
Şekil 4.43. F12 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



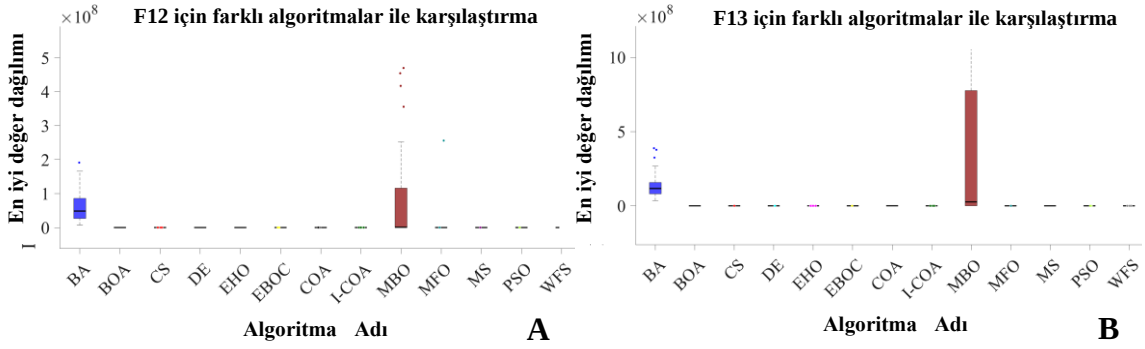
Şekil 4.44. F13 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



Şekil 4.45. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F8 fonksiyonu için karşılaştırma **B)** F9 fonksiyonu için karşılaştırma



Şekil 4.46. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F10 fonksiyonu için karşılaştırma **B)** F11 fonksiyonu için karşılaştırma



Şekil 4.47. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F12 fonksiyonu için karşılaştırma **B)** F13 fonksiyonu için karşılaştırma

Çizelge 4.9. Sabit modlu test fonksiyonları

No	Algoritmalar	Parametreler	Sabit modlu test fonksiyonları									
			F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
1	CCOA	STD	0,998	0	1,032	0,398	3	0,3	3,263	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,3	-3,26	-10,15	-10,40	-10,53
2	CLCCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,258	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,3	-3,25	-10,15	-10,40	-10,53
3	CLCOA	STD	0,998	0	1,032	0,398	3	0,3	3,322	10,153	10,403	10,53
		Ortalama	0,998	0	-1,032	0,398	3	-0,3	-3,32	-10,15	-10,40	-10,53
4	COA	STD	0,998	0,001	1,032	0,398	3	0,3	3,234	9,902	10,394	10,517
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,23	-9,857	-10,39	-10,51
5	FDBLFCOA	STD	0,998	0	1,032	0,398	3	0,3	3,318	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,3	-3,31	-10,15	-10,40	-10,53
6	LCLFDBLF CCOA	STD	0,998	0	1,032	0,398	3	0,3	3,239	10,153	10,403	10,402
		Ortalama	0,998	0	-1,032	0,398	3	-0,3	-3,23	-10,15	-10,40	-10,35
7	LCOA	STD	0,998	0,001	1,032	0,398	3	0,312	3,23	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,312	-3,23	-10,15	-10,40	-10,53
8	LFDBLFCCOA	STD	0,998	0	1,032	0,398	3	0,319	3,211	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,319	-3,21	-10,15	-10,40	-10,53
9	LFDBLFCOA	STD	0,998	0	1,032	0,398	3	0,308	3,225	10,153	10,403	10,536
		Ortalama	0,998	0	-1,032	0,398	3	-0,307	-3,22	-10,15	-10,40	-10,53
10	LOCCOA	STD	0,998	0,001	1,032	0,398	3	0,316	3,219	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,315	-3,21	-10,15	-10,40	-10,53
11	LOCLFDBLF CCOA	STD	0,998	0,001	1,032	0,398	3	0,314	3,225	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,314	-3,22	-10,15	-10,40	-10,53
12	LOCOA	STD	0,998	0,001	1,032	0,398	3	0,31	3,227	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,31	-3,22	-10,15	-10,40	-10,53
13	LOFDBLF CCOA	STD	0,998	0,001	1,032	0,398	3	0,312	3,222	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,312	-3,22	-10,15	-10,40	-10,53
14	LOFDBLF COA	STD	0,998	0,001	1,032	0,398	3	0,314	3,213	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,313	-3,21	-10,15	-10,40	-10,53
15	LOLFCCOA	STD	0,998	0,001	1,032	0,398	3	0,315	3,211	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,314	-3,21	-10,15	-10,40	-10,53
16	OCCLCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,207	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,20	-10,15	-10,40	-10,53
17	OCCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,212	10,138	10,402	10,533
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,21	-10,13	-10,40	-10,53
18	OCLFDBLF CCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,209	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,20	-10,15	-10,40	-10,53
19	OCLLFC COA	STD	0,998	0,001	1,032	0,398	3	0,3	3,211	10,15	10,40	10,53
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,21	-10,15	-10,40	-10,53
20	OCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,225	10,121	10,253	10,331
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,22	-10,12	-10,24	-10,32
21	OFDBLFC COA	STD	0,998	0,001	1,032	0,398	3	0,3	3,212	10,153	10,403	10,536
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,21	-10,15	-10,40	-10,53
22	OFDBLF COA	STD	0,998	0,001	1,032	0,398	3	0,3	3,215	10,15	10,40	10,53
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,21	-10,15	-10,40	-10,53
23	OLFCCOA	STD	0,998	0,001	1,032	0,398	3	0,3	3,205	10,15	10,40	10,53
		Ortalama	0,998	0,001	-1,032	0,398	3	-0,3	-3,20	-10,15	-10,40	-10,53

Çizelge 4.10. Sabit modlu kıyaslama fonksiyonları için simülasyon zamanı (s) (GCOA6)

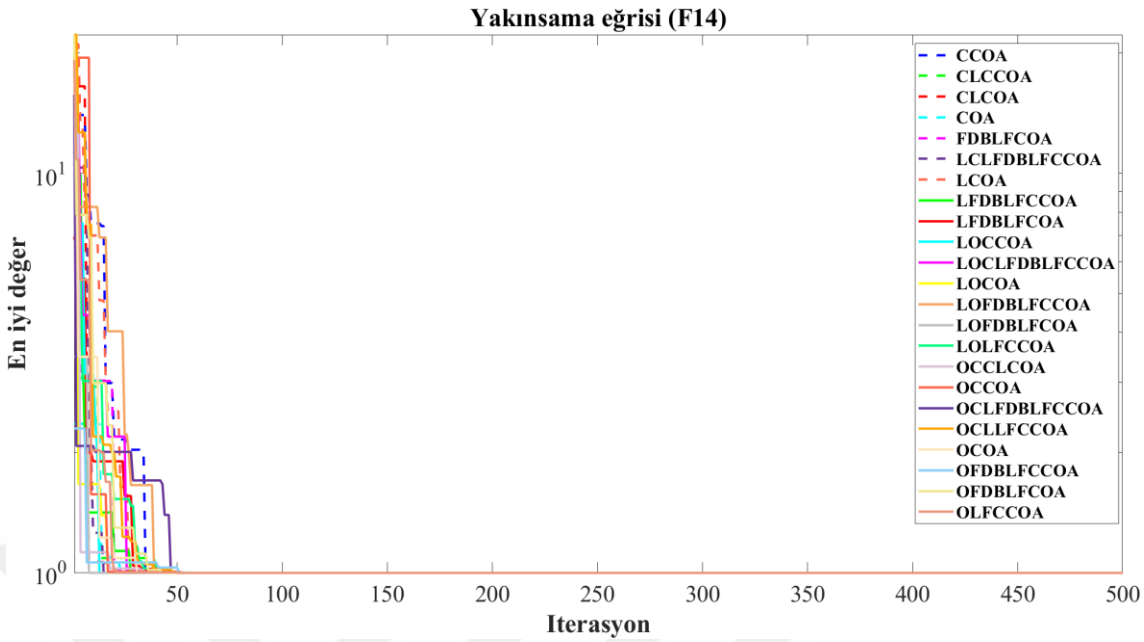
Fonksiyon (F)	Ortalama Zaman	En İyi Zaman	En Kötü Zaman
F14	3,3051	2,8546	3,7443
F15	1,2527	1,0423	1,7193
F16	1,1452	1,0109	1,6382
F17	1,2676	0,9795	1,6507
F18	1,1298	0,9746	1,8314
F19	1,1659	1,0349	1,6798
F20	1,2563	1,1057	1,7391
F21	1,3227	1,1261	2,0264
F22	1,3577	1,1749	1,9436
F23	1,5303	1,3893	2,5718

Çizelge 4.11. Sabit-mod benchmark test fonksiyonları için sonuçlar (GCOA6)

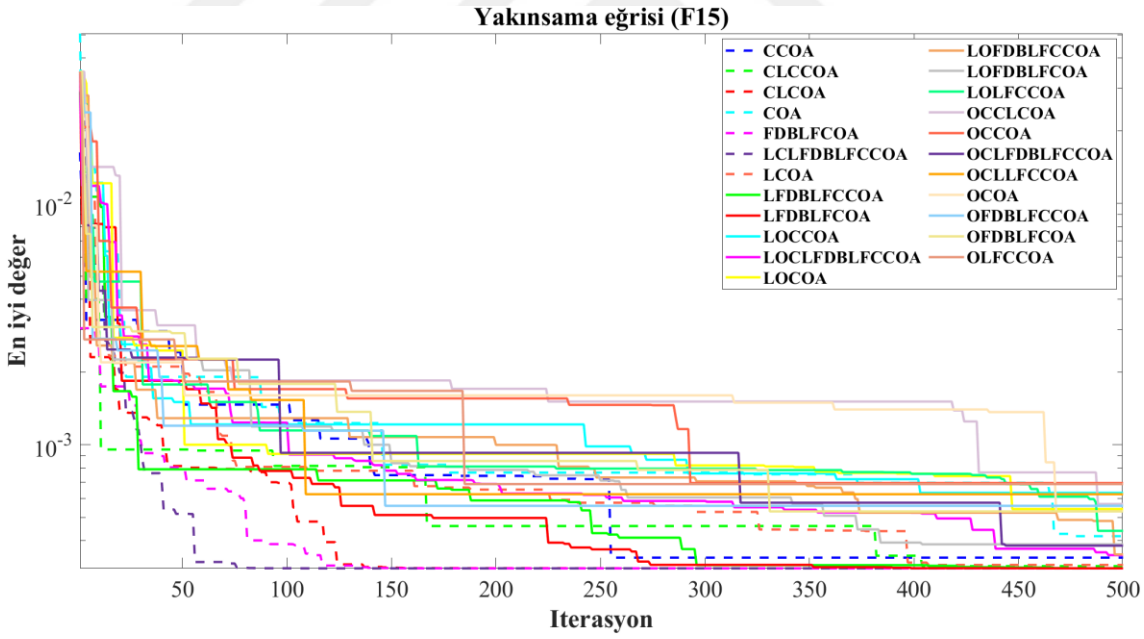
Fonksiyon (F)	Amaç fonksiyon uygunluğu					Wilcoxon rank Sum Test
	En İyi	Ortalama	En Kötü	STD Değer	Median Değer	P Değeri
F14	0,998	0,998	0,998	0	0,998	1
F15	0,0003	0,0003	0,0003	1,79 e-19	0,0003	1,466e-11
F16	-1,031	-1,031	-1,031	6,7 e-16	-1,0316	6,527e-04
F17	0,3978	0,3978	0,3978	0	0,3978	8,437e-15
F18	2,9999	2,9999	2,9999	1,853 e-15	2,9999	0,1856
F19	-3,862	-3,862	-3,862	2,71e-15	-3,8627	8,437e-15
F20	-3,321	-3,226	-3,203	0,0483	-3,2031	0,1181
F21	-10,15	-10,15	-10,15	7,06e-15	-10,1531	1,572e-12
F22	-10,40	-10,40	-10,40	8,07e-16	-10,4029	3,159e-12
F23	-10,53	-10,53	-10,53	9,32e-16	-10,5364	4,433e-12

Çizelge 4.12. Sabit-mod benchmark test fonksiyonları için karşılaştırmalı sonuçlar (GCOA6)

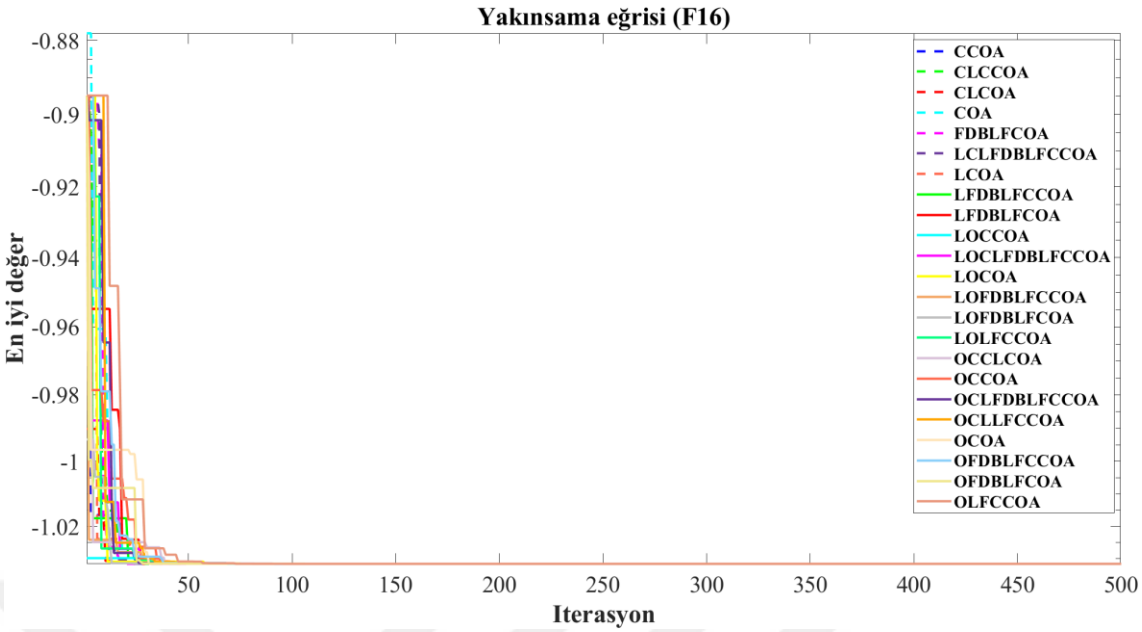
Algoritmalar	Parametreler	Sabit-Mod test fonksiyonları									
		F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
BA	STD	10,7	0,05	0,86	0,65	11,7	1,61	2,35	1,61	2,29	2,7
	Mean	8,42	0,03	-0,83	0,59	9,32	-1,43	-2,31	-1,41	-1,89	-2,2
BOA	STD	1,15	0	1,03	0,39	3,01	0,3	3,15	7,96	7,28	7,5
	Mean	1,11	0	-1,03	0,39	3,01	-0,3	-3,15	-7,85	-7,16	-7,3
CS	STD	0,99	0	1,03	0,39	3,00	0,3	3,32	10,15	10,4	10,5
	Mean	0,99	0	-1,03	0,39	3,00	-0,3	-3,32	-10,15	-10,4	-10,5
DE	STD	0,99	0	1,03	0,39	2,99	0,3	3,32	10,15	10,4	10,5
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,32	-10,15	-10,4	-10,5
EHO	STD	1,80	0	1,02	0,40	3,28	0,3	3,15	4,88	5,13	5,2
	Mean	1,59	0	-1,02	0,40	3,28	-0,3	-3,15	-4,87	-5,08	-5,1
EBO with CMAR	STD	0,99	0	1,03	0,39	2,99	0,3	3,32	10,15	10,4	10,5
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,32	-10,15	-10,4	-10,5
COA	STD	0,99	0	1,03	0,39	2,99	0,3	3,21	10,00	10,4	10,5
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,21	-10,00	-10,4	-10,5
MBO	STD	1,09	0	1,03	0,39	3,01	0,3	3,22	7,05	6,55	7,2
	Mean	1,06	0	-1,03	0,39	3,01	-0,3	-3,22	-6,37	-5,72	-6,2
MFO	STD	0,99	0	1,03	0,39	2,99	0,3	3,23	9,01	9,69	10,1
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,23	-8,63	-9,44	-10,1
MS	STD	6,28	0	1,03	0,39	3,00	0,3	3,26	5,05	5,08	5,1
	Mean	5,18	0	-1,03	0,39	3,00	-0,3	-3,26	-5,05	-5,08	-5,1
PSO	STD	0,99	0	1,03	0,39	2,99	0,3	3,25	9,04	9,69	9,5
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,25	-8,72	-9,44	-9,2
WFS	STD	1,12	0	1,03	0,39	3,00	0,29	3,25	9,14	8,84	9,6
	Mean	1,06	0	-1,03	0,39	3,00	-0,29	-3,25	-8,79	-8,31	-9,2
Geliştirilmiş COA GCOA6	STD	0,99	0	1,03	0,39	2,99	0,3	3,23	10,15	10,27	10,54
	Mean	0,99	0	-1,03	0,39	2,99	-0,3	-3,23	-10,15	-10,22	-10,54



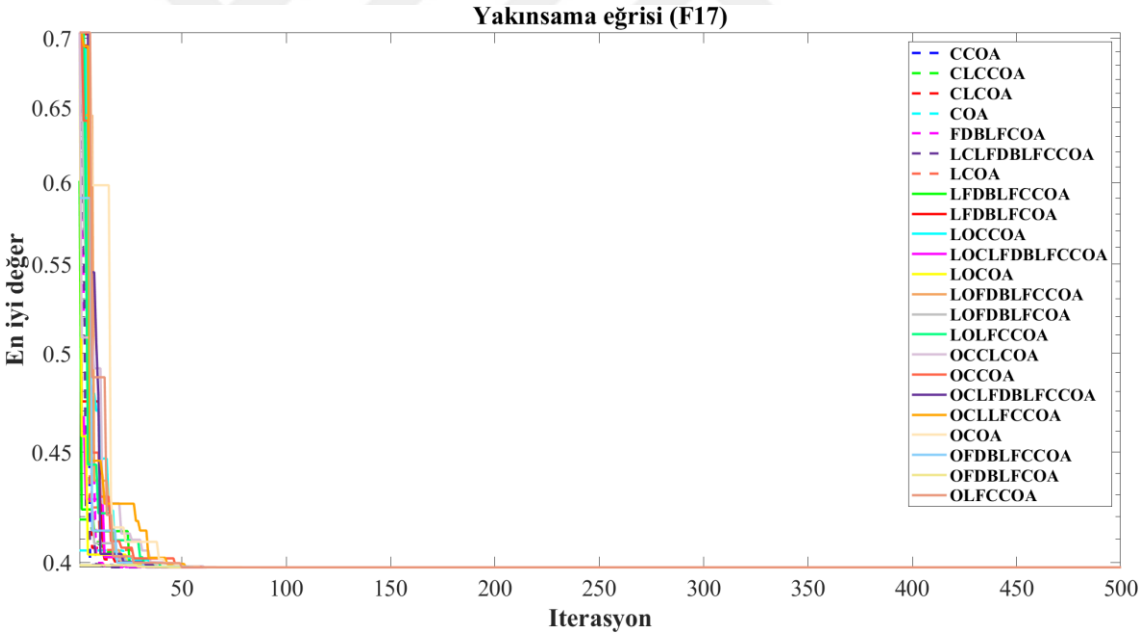
Şekil 4.48. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F14 test fonksiyonu için karşılaştırması.



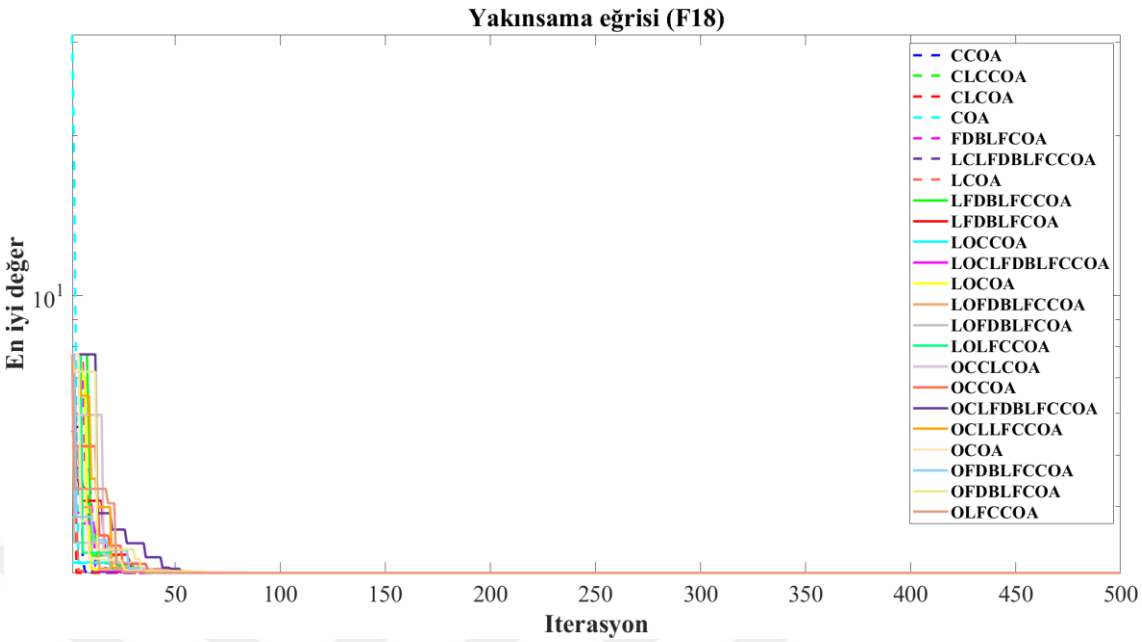
Şekil 4.49. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F15 test fonksiyonu için karşılaştırması.



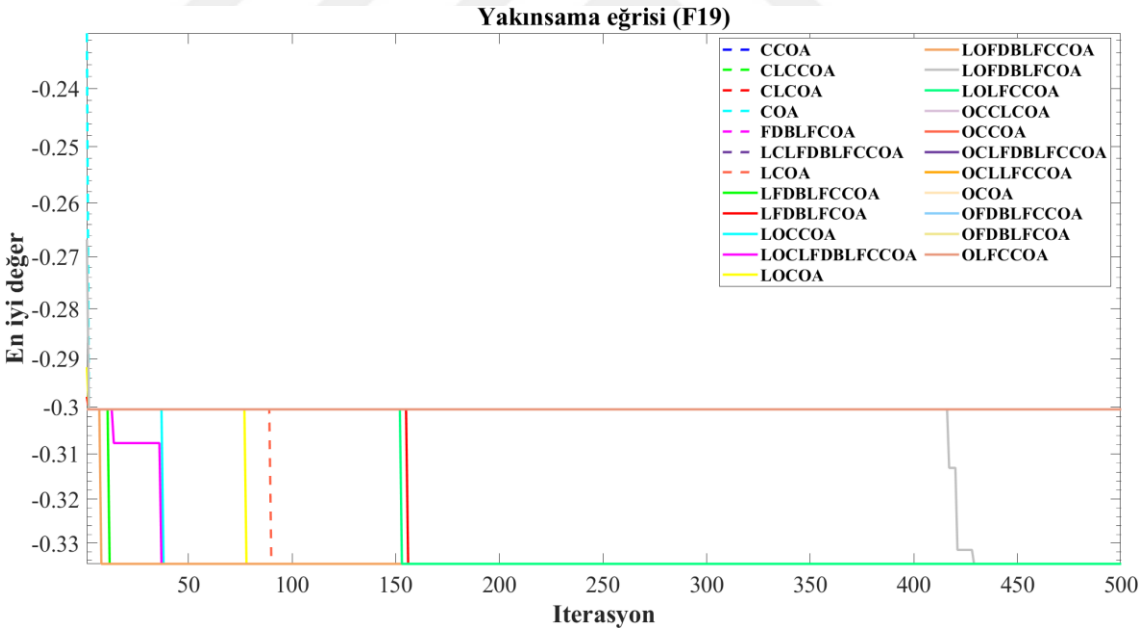
Şekil 4.50. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F16 test fonksiyonu için karşılaştırması.



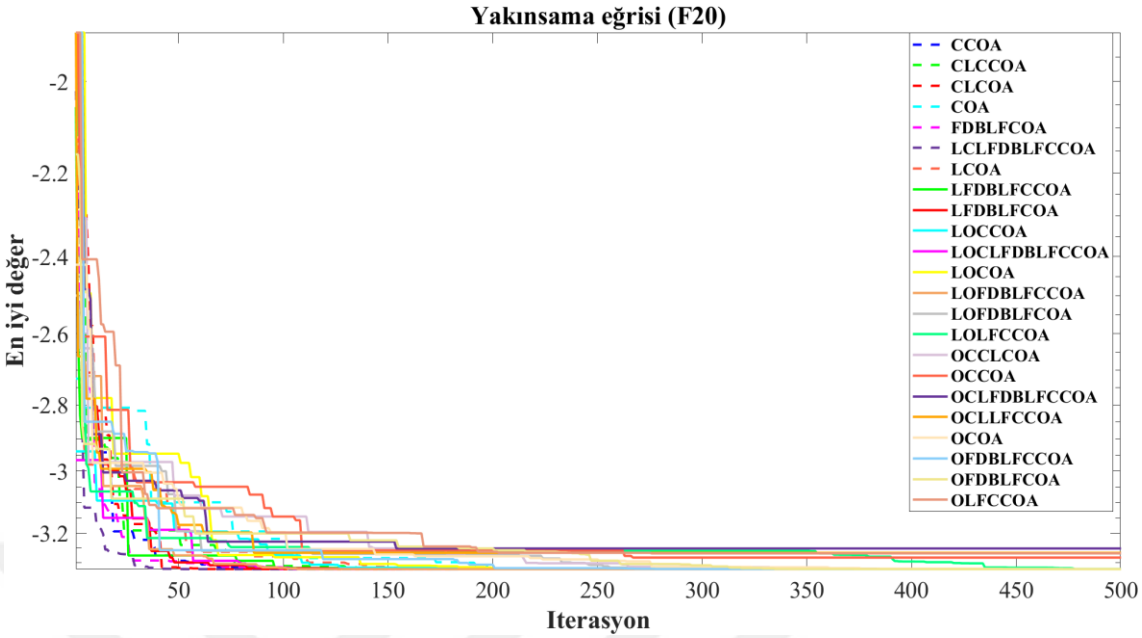
Şekil 4.51. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F17 test fonksiyonu için karşılaştırması.



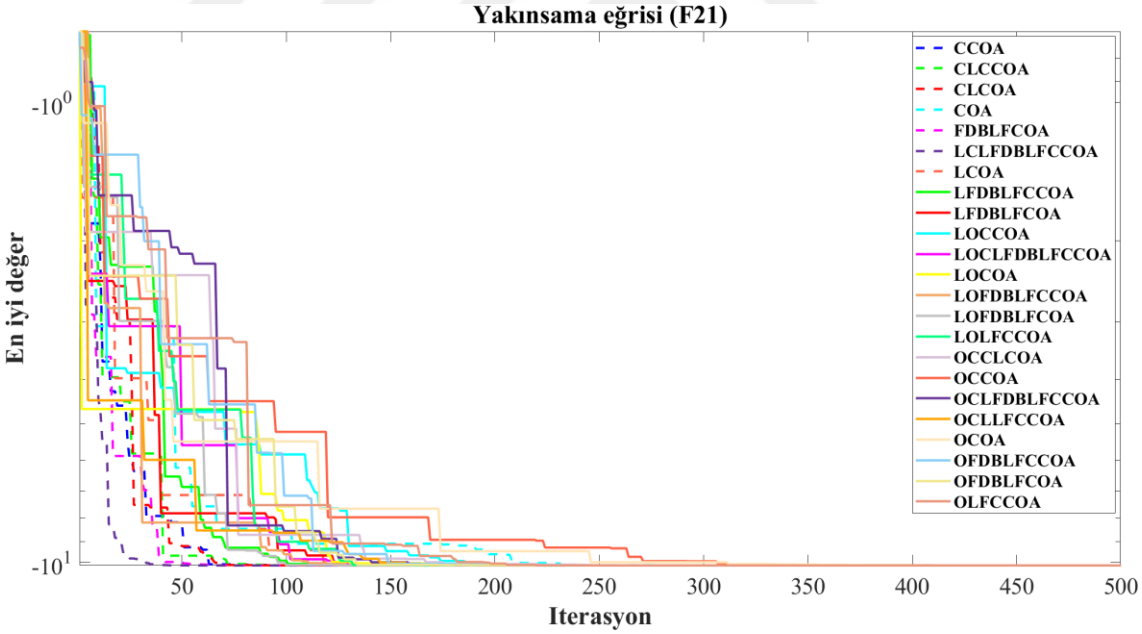
Şekil 4.52. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F18 test fonksiyonu için karşılaştırması.



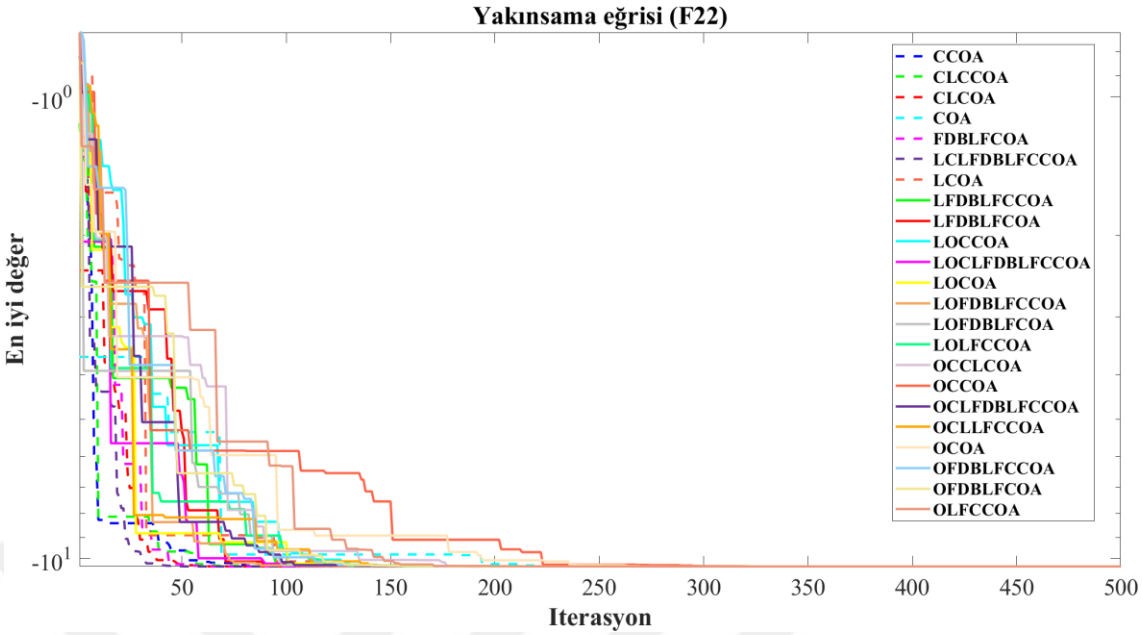
Şekil 4.53. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F19 test fonksiyonu için karşılaştırması.



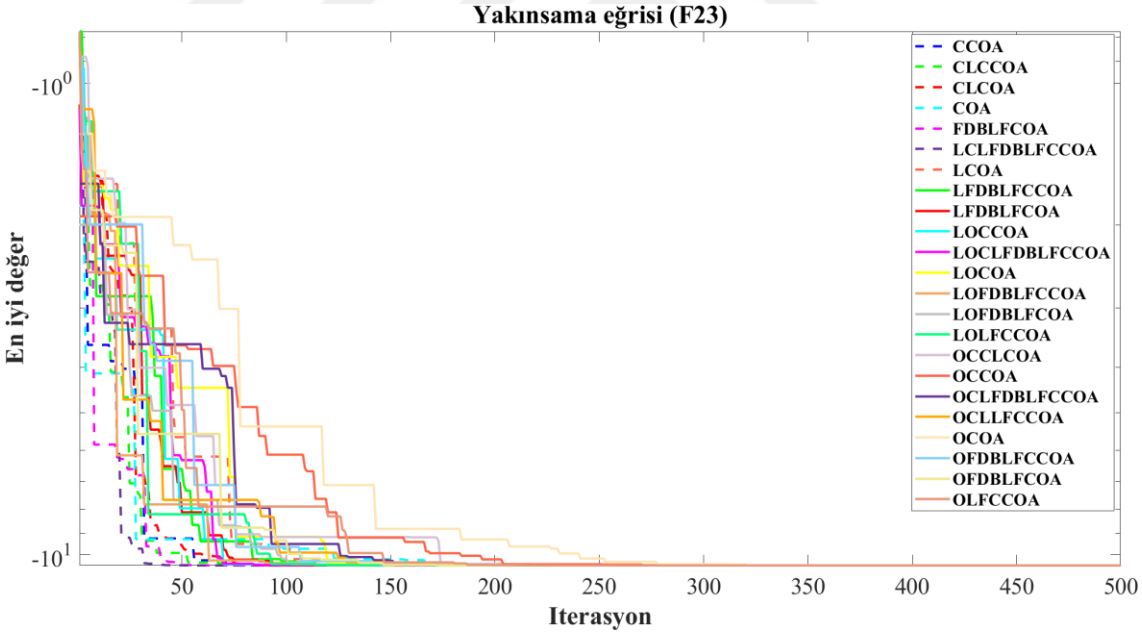
Şekil 4.54. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F20 test fonksiyonu için karşılaştırması.



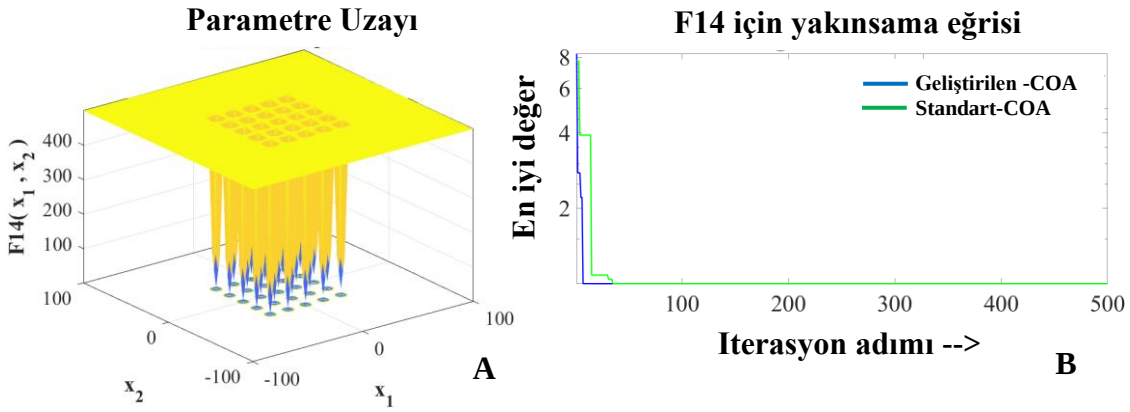
Şekil 4.55. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F21 test fonksiyonu için karşılaştırması.



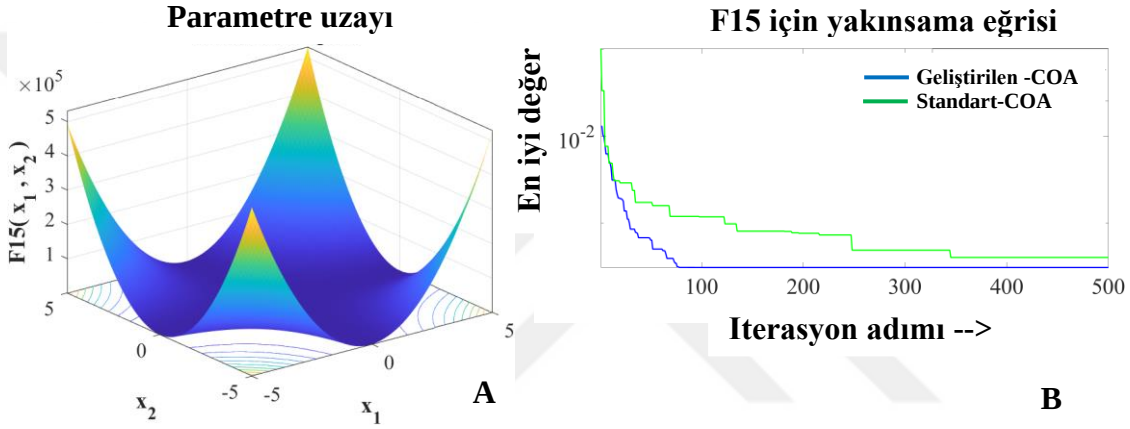
Şekil 4.56. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F22 test fonksiyonu için karşılaştırması.



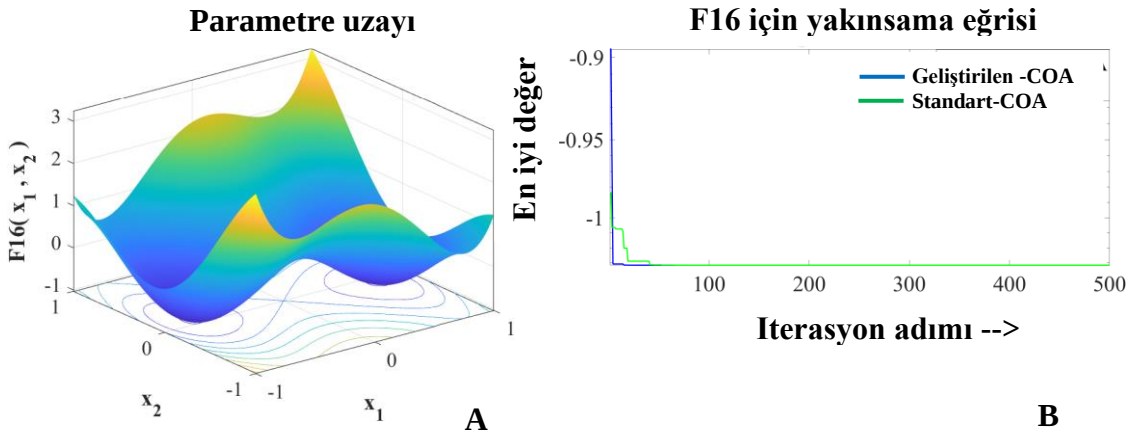
Şekil 4.57. Yeni oluşturulan farklı kır kurdu algoritmalarının sabit modlu F23 test fonksiyonu için karşılaştırması.



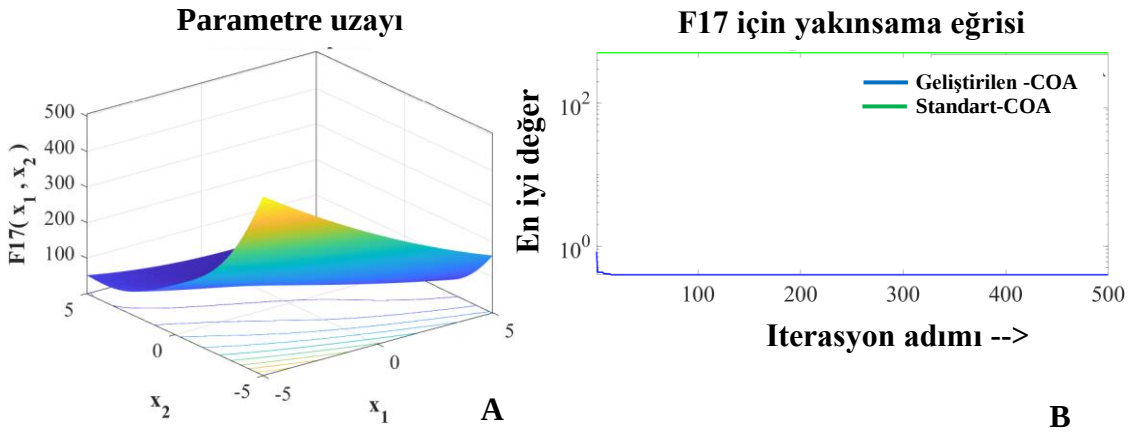
Şekil 4.58. COA ve GCOA 'no:6' için F14 test fonksiyonu şematiği.
A) F14 test fonksiyonu parametre uzayı **B)** F14 test fonksiyonu için yakınsama eğrisi



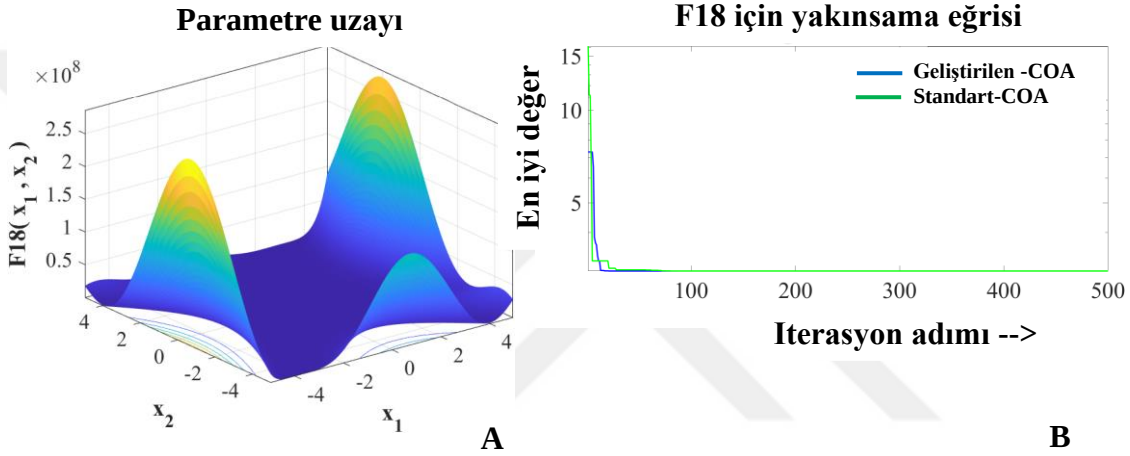
Şekil 4.59. COA ve GCOA 'no:6' için F15 test fonksiyonu şematiği.
A) F15 test fonksiyonu parametre uzayı **B)** F15 test fonksiyonu için yakınsama eğrisi



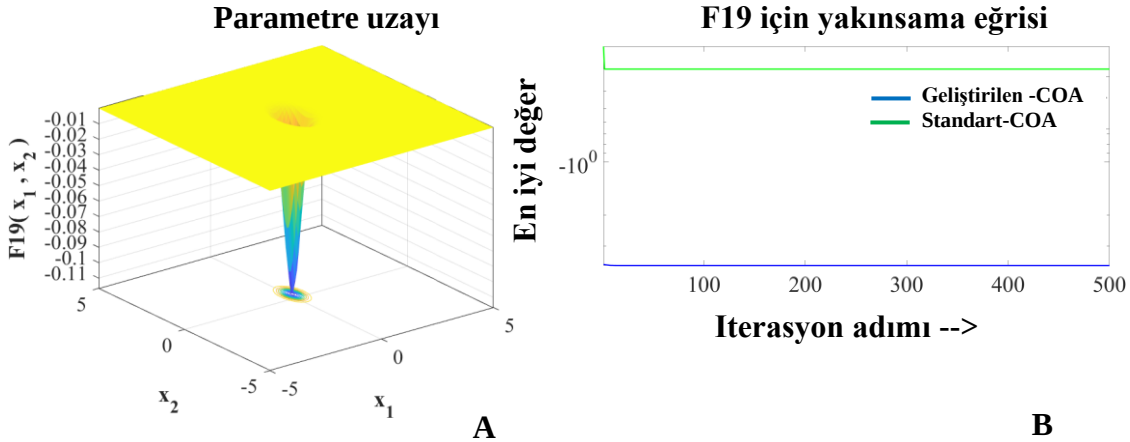
Şekil 4.60. COA ve GCOA 'no:6' için F16 test fonksiyonu şematiği.
A) F16 test fonksiyonu parametre uzayı **B)** F16 test fonksiyonu için yakınsama eğrisi



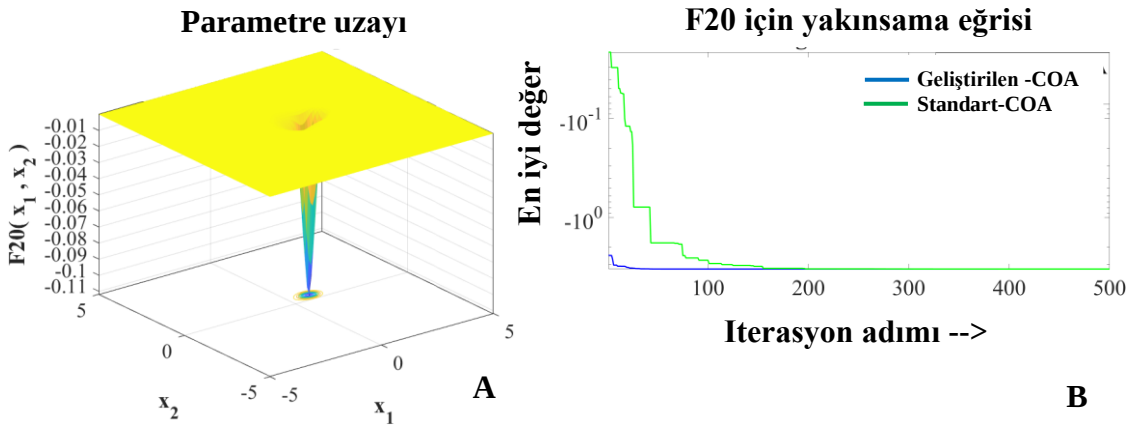
Şekil 4.61. COA ve GCOA 'no:6' için F17 test fonksiyonu şematiği.
A) F17 test fonksiyonu parametre uzayı **B)** F17 test fonksiyonu için yakınsama eğrisi



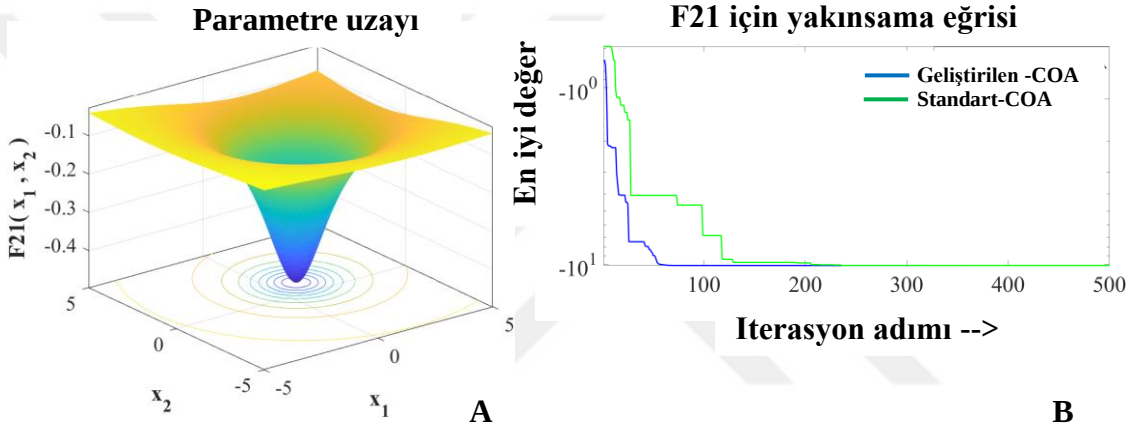
Şekil 4.62. COA ve GCOA 'no:6' için F18 test fonksiyonu şematiği.
A) F18 test fonksiyonu parametre uzayı **B)** F18 test fonksiyonu için yakınsama eğrisi



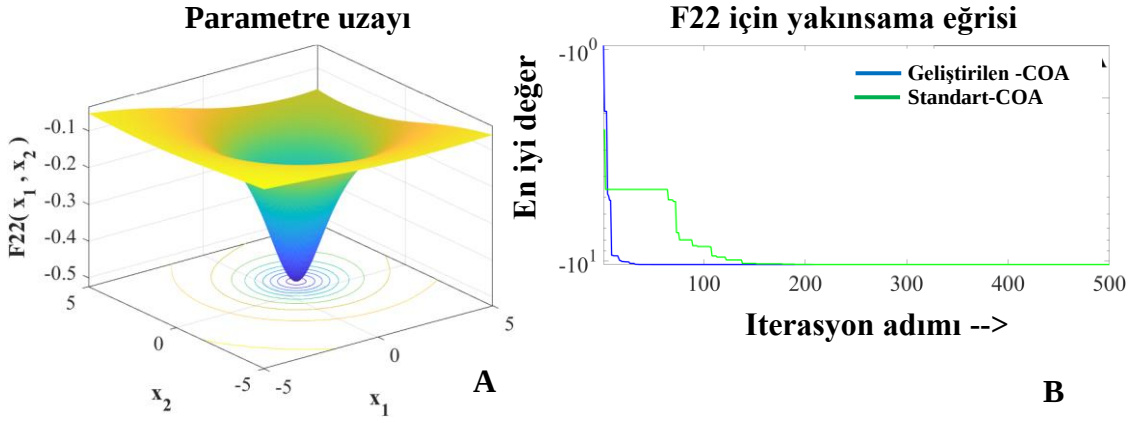
Şekil 4.63. COA ve GCOA 'no:6' için F19 test fonksiyonu şematiği.
A) F19 test fonksiyonu parametre uzayı **B)** F19 test fonksiyonu için yakınsama eğrisi



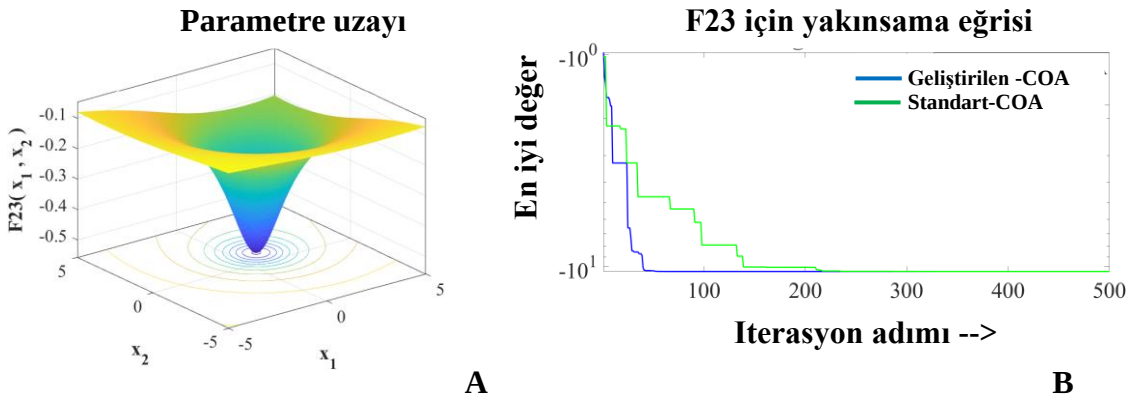
Şekil 4.64. COA ve GCOA 'no:6' için F20 test fonksiyonu şematiği.
A) F20 test fonksiyonu parametre uzayı **B)** F20 test fonksiyonu için yakınsama eğrisi



Şekil 4.65. COA ve GCOA 'no:6' için F21 test fonksiyonu şematiği.
A) F21 test fonksiyonu parametre uzayı **B)** F21 test fonksiyonu için yakınsama eğrisi

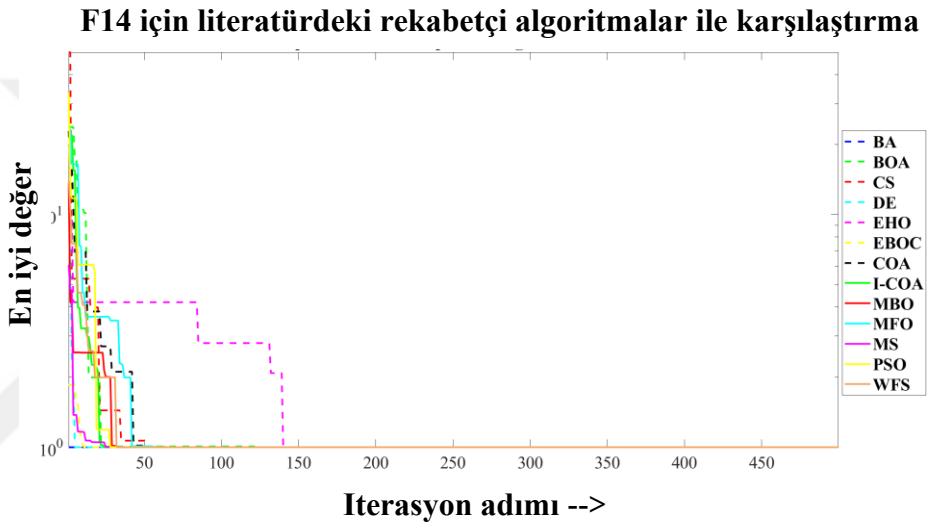


Şekil 4.66. COA ve GCOA 'no:6' için F22 test fonksiyonu şematiği.
A) F22 test fonksiyonu parametre uzayı **B)** F22 test fonksiyonu için yakınsama eğrisi

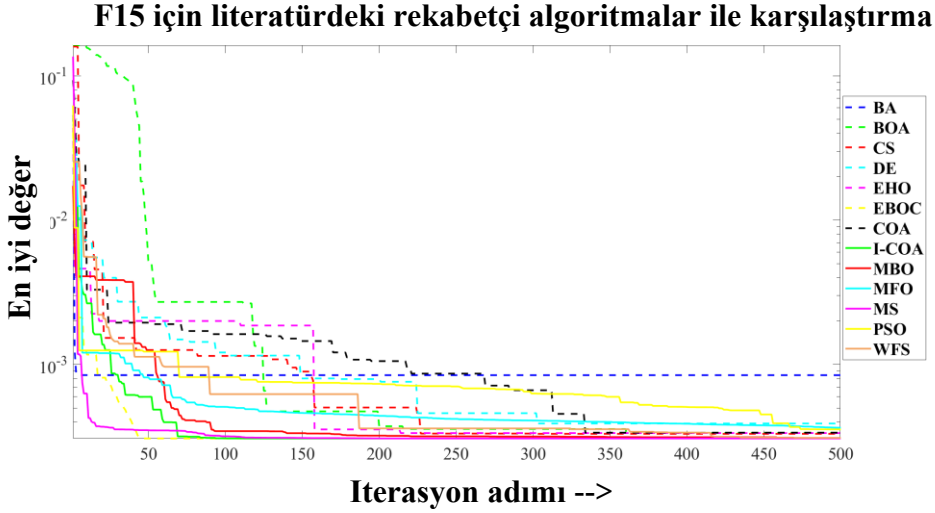


A **B**

Şekil 4.67. COA ve GCOA 'no:6' için F22 test fonksiyonu şematiği.
A) F22 test fonksiyonu parametre uzayı **B)** F22 test fonksiyonu için yakınsama eğrisi



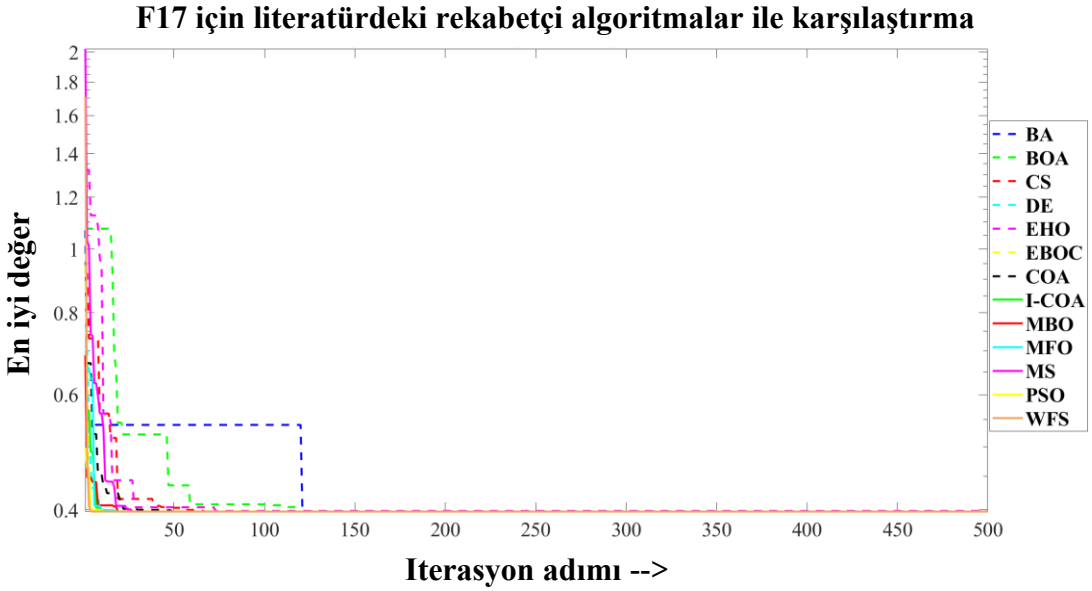
Şekil 4.68. F14 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



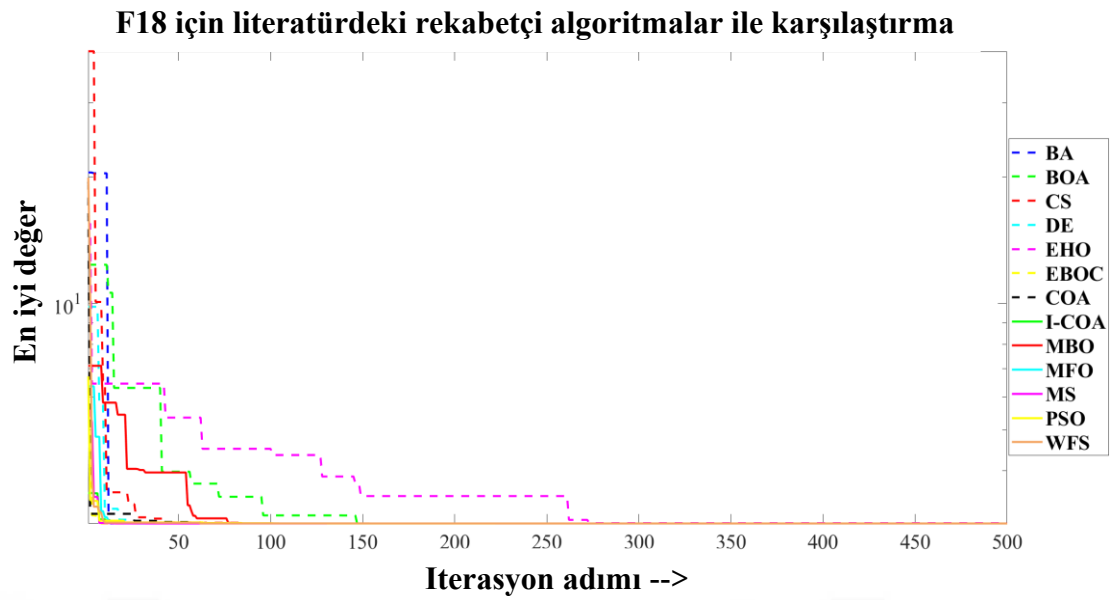
Şekil 4.69. F15 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



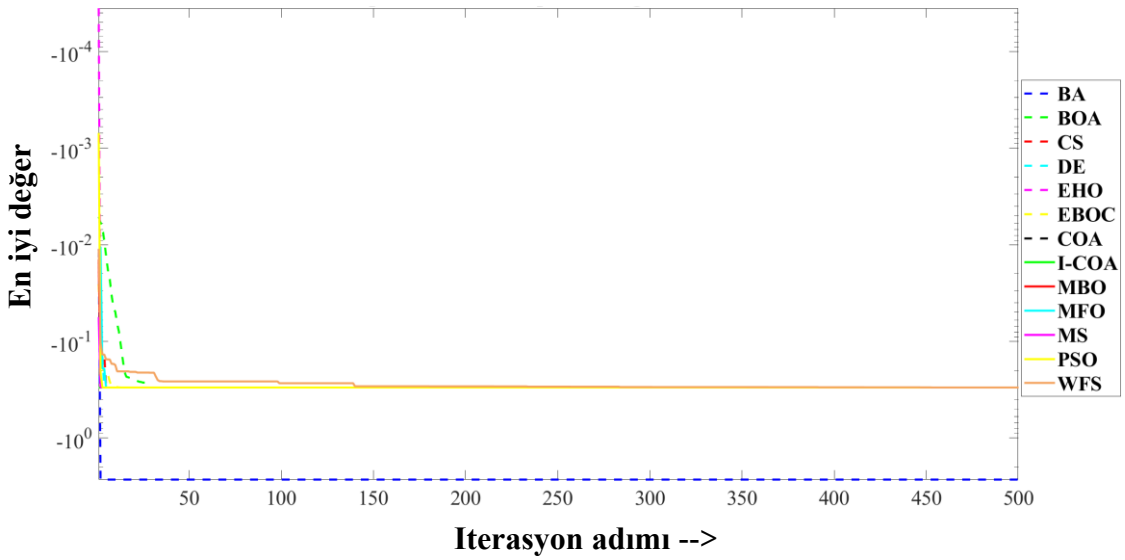
Şekil 4.70. F16 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



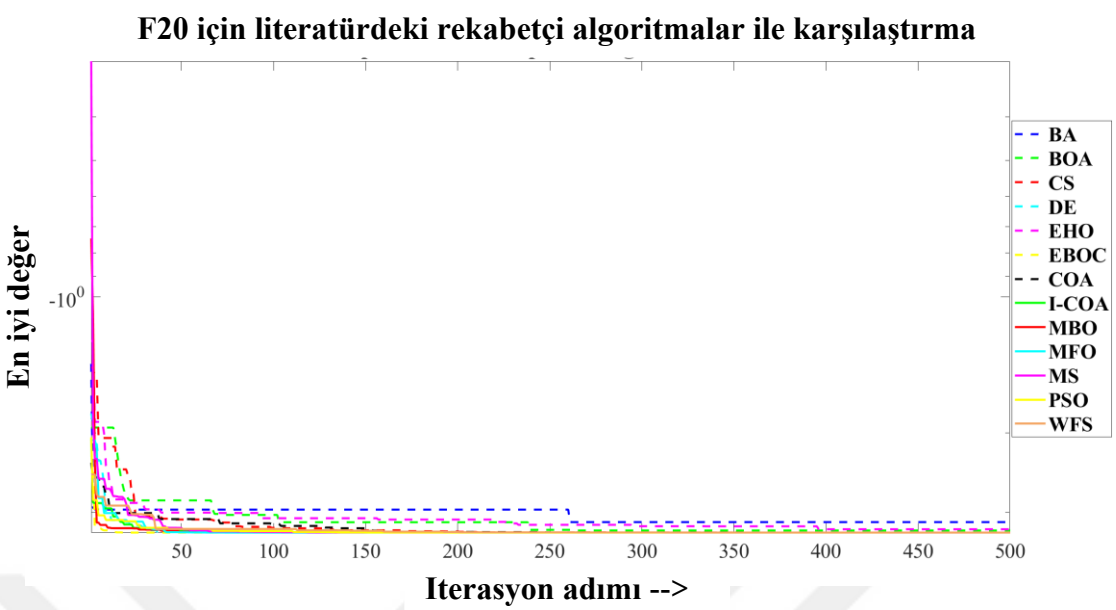
Şekil 4.71. F17 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



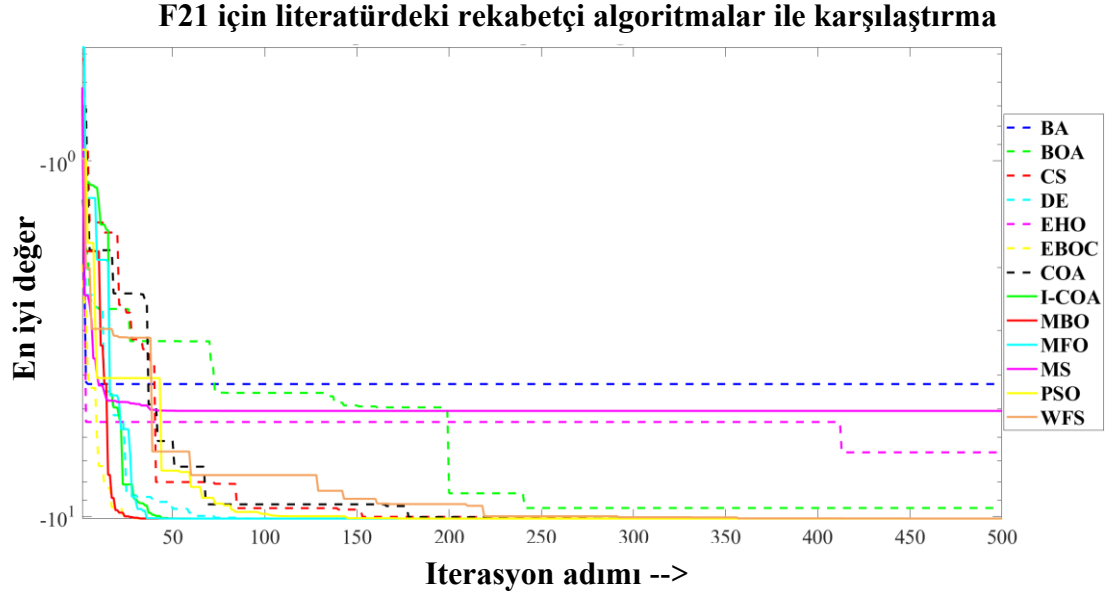
Şekil 4.72. F18 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



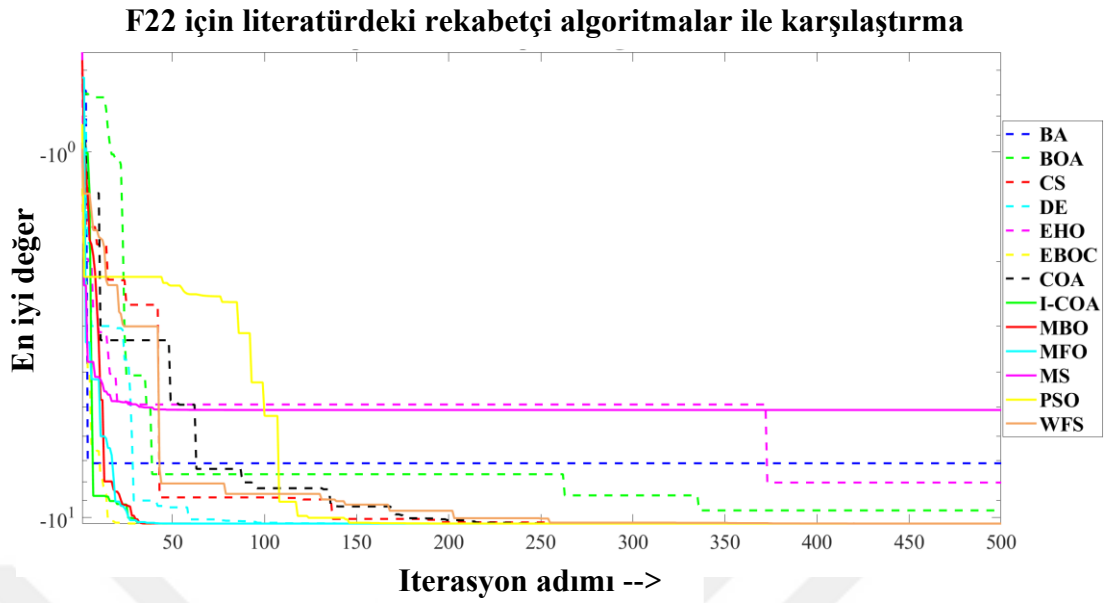
Şekil 4.73. F19 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



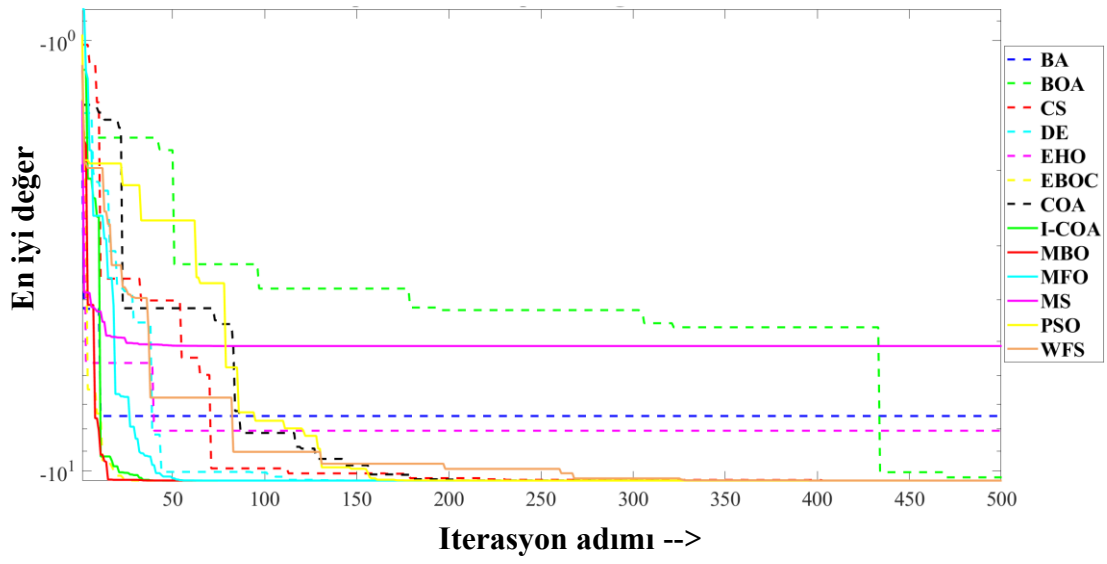
Şekil 4.74. F20 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



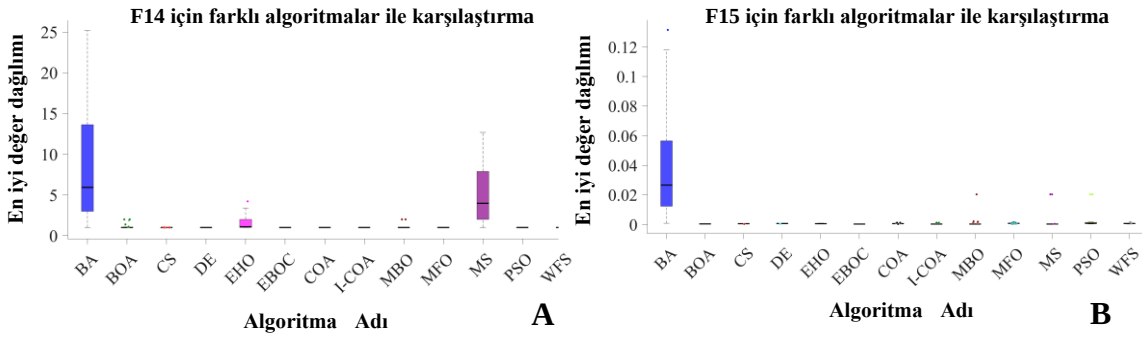
Şekil 4.75. F21 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



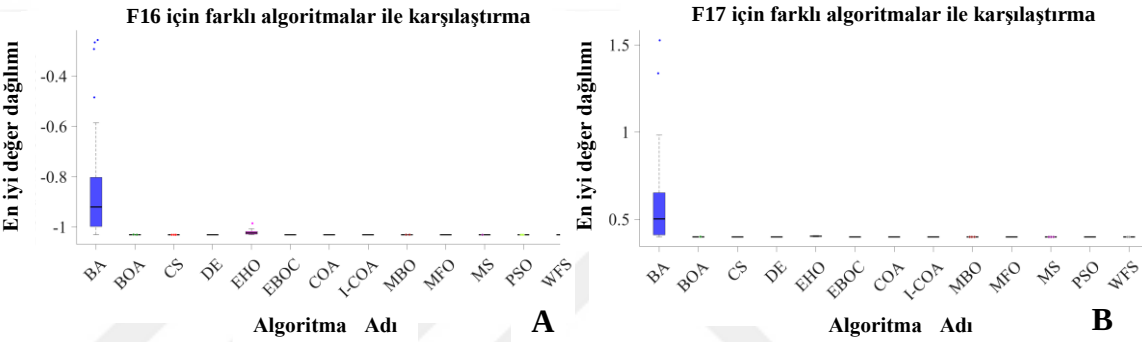
Şekil 4.76. F22 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



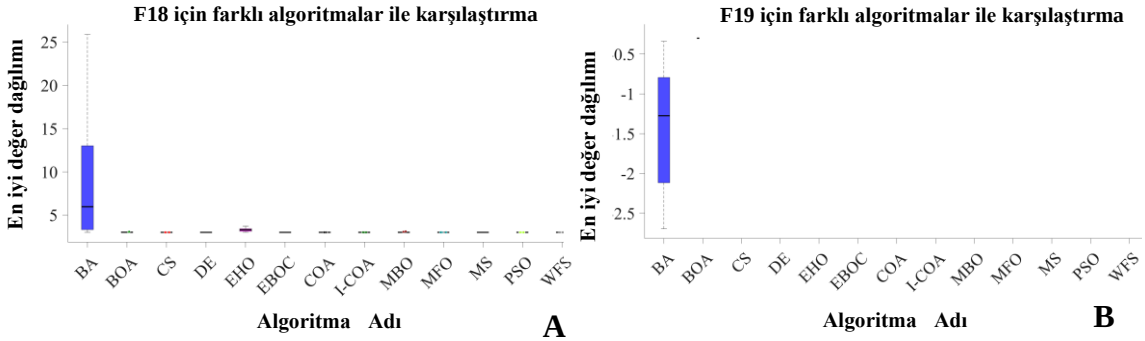
Şekil 4.77. F23 test fonksiyonu için literatürdeki algoritmalar ile karşılaştırmalı yakınsama eğrisi.



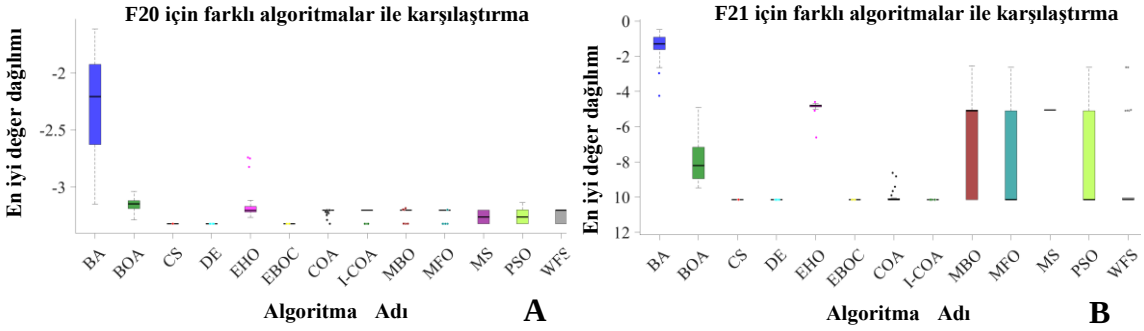
Şekil 4.78. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F14 fonksiyonu için karşılaştırma **B)** F15 fonksiyonu için karşılaştırma



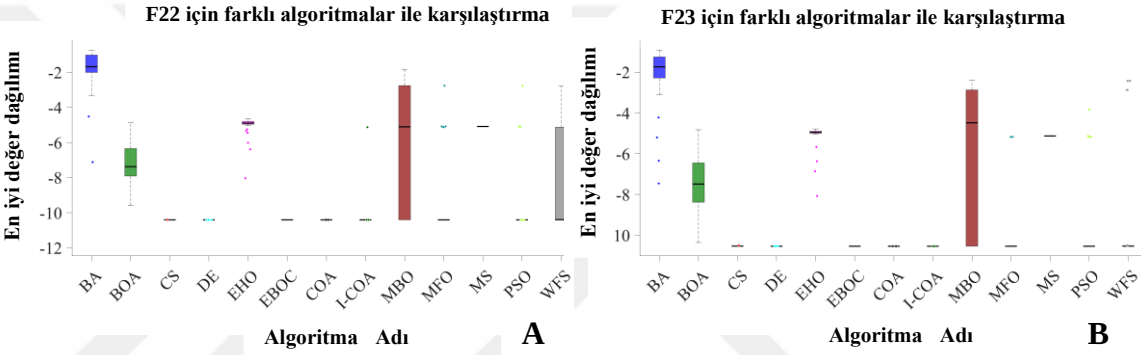
Şekil 4.79. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F16 fonksiyonu için karşılaştırma **B)** F17 fonksiyonu için karşılaştırma



Şekil 4.80. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F18 fonksiyonu için karşılaştırma **B)** F19 fonksiyonu için karşılaştırma



Şekil 4.81. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F20 fonksiyonu için karşılaştırma **B)** F21 fonksiyonu için karşılaştırma



Şekil 4.82. Rakip algoritmalarla karşılaştırılmalı tek modlu kıyaslama fonksiyonunun değer dağılım kutu grafiği.
A) F22 fonksiyonu için karşılaştırma **B)** F23 fonksiyonu için karşılaştırma

Çizelge 4.13. Geliştirilmiş algoritmanın (GCOA) mühendislik tasarım problemi

Mühendislik Tasarım Problemi (MTP)	En iyi	Ortalama	En kötü	STD Değer	Median Değer	P Değer
MTP1	1,6952	1,6952	1,6952	4,51e-16	1,6952	6,0585e-13
MTP2	0,01266	0,01266	0,01266	7,65e-08	0,01266	1,5099e-11
MTP3	5885,33	5885,33	5885,33	1,6888e-13	5885,33	8,6009e-13
MTP4	2994,47	2994,47	2994,47	1,3875e-12	2994,47	6,0585e-13
MTP5	0	2,0e-11	3,35e-10	7,14e-11	0	0,00015215
MTP6	186,602	186,602	186,602	8,67e-14	186,602	6,0585e-13
MTP7	61914,79	61914,80	61914,83	0,013821	61914,80	1,5099e-11
MTP8	0	0	0	0	0	6,0585e-13
MTP9	-40792,1	-40661,9	-36886,26	713,112	-40792,1	1,3545e-14
MTP10	-85539,1	-85539,1	-85539,1	4,67e-05	-85539,1	9,5586e-12

Çizelge 4.14. MTP1-MTP10 zaman karşılaştırması

Mühendislik Tasarım Problemi (MTP)	En iyi Zaman	Ortalama Zaman	En kötü Zaman
MTP1	1,4751	2,2040	3,7001
MTP2	1,2337	1,3597	2,0369
MTP3	1,2763	1,4822	2,1092
MTP4	1,5243	1,7522	2,5701

Çizelge 4.14. MTP1-MTP10 zaman karşılaştırması (devam)

Mühendislik Tasarım Problemi (MTP)	En iyi Zaman	Ortalama Zaman	En kötü Zaman
MTP5	1,2603	1,6552	2,4069
MTP6	1,2902	1,4508	2,0559
MTP7	1,7753	2,0337	2,6060
MTP8	2,0978	3,0667	7,0973
MTP9	1,7154	2,7356	5,2424
MTP10	1,4202	2,7582	5,5999

Çizelge 4.15. Kaynaklı kiriş tasarım problemi için önerilen algoritmanın literatürdeki mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Kaynaklı Kiriş Tasarım Problemi	En kötü	1,69524	1,6977
	Ortalama	1,69524	1,6965
	En iyi	1,69524	1,6953
	Standart Sapma	1,323E-11	0,0012

Geliştirilmiş COA algoritmasının kaynaklı kiriş tasarım problemi (bkz. Şekil 3.24) için sonuçları, aşağıdaki çizelgede literatürden birkaç algoritmalarla karşılaştırmalı olarak birlikte gösterilmiştir. Literatürdeki algoritmalar arasında yeni algoritma daha iyi sonuç vermiştir.

Çizelge 4.15.1. Kaynaklı kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	Metot	x1(h)	x2(l)	x3(t)	x4(b)	En uygun Sonuç
Akay and Karaboga	ABC	0,2057	3,4704	9,0366	0,2057	1,7249
Aragon ve ark.	TCA	0,2444	6,2186	8,2915	0,2444	2,3811
Bernardino ve ark.	GA-AIS	0,2444	6,2183	8,2915	0,2444	2,3812
Datta and Figueira	PSO	0,1875	1,7821	8,2500	0,2500	1,9553
Gandomi	ISA	0,2443	6,2199	8,2915	0,2443	2,3812
Han ve ark.	STA	0,2053	3,2603	9,0366	0,2057	1,6956
Kanagaraj ve ark.	CSGA	0,2443	6,2175	8,2915	0,2444	2,3809
Montes and Ocana	BFO	0,2057	3,4711	9,0367	0,2057	2,3868
Wang ve ark.	BSA	0,2057	3,4704	9,0366	0,2057	1,7249
Zhang ve ark.	DE	0,2444	6,2175	8,2915	0,2444	2,3810
Coello	GA2	0,2088	3,4205	8,9975	0,2100	1,748309
Coello ve ark.	CAEP	0,2057	3,4705	9,0366	0,2057	1,724852
Renato ve ark.	CPSO	0,202369	3,544214	9,048210	0,205723	1,728024
Mevcut çalışma	GCOA6	0,2057	3,2531	9,0366	0,2057	1,69524

Çizelge 4.16. Bası yay tasarım problemi için önerilen algoritmanın literatürdeki mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Yay Tasarım Problemi	En kötü	0,0126657	0,01268
	Ortalama	0,0126653	0,01268
	En iyi	0,0126659	0,01267
	Standart Sapma	2,078E-07	0,00001

Bası yay tasarım problemi (bkz. Şekil 3.25) için geliştirilmiş algoritmanın (GCOA6) sonuçları literatürden birkaç algoritma ile karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir. Literatürdeki algoritmalar arasında yeni algoritma aynı düzeyde sonuç vermiştir.

Çizelge 4.16.1. Bası yay tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	**Metot**	**x1(d)**	**x2(D)**	**x3(N)**	**En uygun Sonuç**
Akay ve Karaboga	ABC	0,051749	0,358179	11,203763	0,012665
Aragon ve ark.	TCA	0,051622	0,355105	11,384534	0,012665
Askarzadeh	CSA	0,051689	0,356716	11,289011	0,012665
Bernardino ve ark.	GA-AIS	0,051660	0,356032	11,32955	0,012666
Dos Santos Coelho	Q-PSO	0,051515	0,352529	11,538862	0,012665
Du ve ark.	FOA	0,05206590	0,36570924	10,78621813	0,012676
Gandomi.	ISA	-	-	-	0,012665
Han ve ark.	STA	0,0516800	0,3565001	11,3018335	0,012665
Mohammed	NDE	0,051689058	0,35671768	11,2889687	0,012665
Wang ve ark.	BSA	0,051743	0,358017	11,213187	0,012665
Zhang ve ark.	DE	0,05169	0,35672	11,289	0,012665
Zhang ve ark.	DEDS	0,051689	0,356717	11,288965	0,01267
Zahara ve ark.	NM–PSO	0,051620	0,355498	11,333272	0,01263
Wang ve ark.	DELC	0,051689	0,356717	11,288965	0,01267
Mevcut çalışma	GCOA6	0,051538	0,353096	11,505461	0,01266

Çizelge 4.17. Basınçlı kap tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Basınçlı Kap Tasarım Problemi	En kötü	5885,332	5894,247
	Ortalama	5885,332	5888,452
	En iyi	5885,332	5885,339
	Standart Sapma	0	5,023

Basınçlı kap tasarım problemi (bkz. Şekil 3.26) için geliştirilmiş algoritmanın (GCOA6) algoritmasının sonuçları literatürden birkaç algoritma ile karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir. Literatürdeki algoritmalar arasında yeni algoritma fark edilebilir seviyede iyi sonuçlar vermiştir.

Çizelge 4.17.1. Basınçlı kap tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	**Metot**	**Z1**	**Z2**	**Z3**	**Z4**	**En uygun Sonuç**
Akay and Karaboga	ABC	0,8125	0,4375	42,098446	176,63659	6059,7147
Aragon ve ark.	TCA	0,8125	0,4375	42,098429	190,78769	6390,554
Askarzadeh	CSA	0,8125	0,4375	42,098445	176,63659	6059,7144
Bernardino ve ark.	GA-AIS	0,8125	0,4375	42,0973	176,6509	6059,8546
Dos Santos Coelho	Q-PSO	0,8125	0,4375	42,0984	176,6372	6059,7208
Du ve ark.	FOA	0,7804	0,3849	40,3888	199,1172	5894,5981
Gandomi ve ark.	ISA	0,8125	0,4375	42,09845	176,6366	6059,714
Han ve ark.	STA	0,7785	0,3848	40,3389	199,7753	5886,45436
Mohammed	NDE	0,8125	0,4375	42,0984	176,63659	6059,7143
Montes and Ocana	BFO	0,8125	0,4375	42,096394	176,68323	6060,460
Wang ve ark.	BSA	0,8125	0,4375	42,098497	176,63596	6059,7082
Brajevic ve ark.	UFA	0,8125	0,4375	42,098445	176,63659	6059,714
Sadollah ve ark.	MBA	0,7802	0,3856	40,4292	198,4964	5889,321
Coello ve ark.	GA3	0,8125	0,4375	42,0974	176,6540	6059,9463
Mevcut çalışma	GCOA6	0,7781	0,3846	40,3196	199,99	5885,33

Çizelge 4.18. Hız düşürücü tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Hız Düşürücü Tasarım Problemi	En kötü	2994,47	2994,48
	Ortalama	2994,47	2994,48
	En iyi	2994,47	2994,49
	Standart Sapma	0	0,005

Hız düşürücü tasarım problemi (bkz. Şekil 3.27) için geliştirilmiş algoritmanın (GCOA6) algoritmasının sonuçları literatürden birkaç algoritma ile karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir. Literatürdeki algoritmalar arasında yeni algoritma fark edilebilir seviyede iyi sonuçlar vermiştir.

Çizelge 4.18.1. Hız düşürücü tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	**Metot**	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**En uygun Sonuç**
Akay and Karaboga	ABC	3,49	0,7	17	7,3	7,8	3,35	5,2878	2997,0584
Bernardino ve ark.	GA-AIS	3,5	0,7	17	7,3	7,8	3,35	5,2866	2996,3483
Kanagaraj ve ark..	CSGA	3,5	0,7	17	7,6	7,81	3,35	5,2687	2996,3482
Liu ve ark.	PSO-DE	3,5	0,7	17	7,3	7,8	3,35	5,2866	2996,3481
Mohammed	NDE	3,5	0,7	17	7,3	7,71	3,35	5,2866	2994,4710
Montes ve ark.	EA	3,5	0,7	17	7,3	7,96	3,36	5,3089	2996,3566
Montes ve ark..	DE	3,5	0,7	17	7,3	7,8	3,35	5,2866	2996,3566
Rao and Vakharia	TLBO	-	-	-	-	-	-	-	2996,3481
Wang ve ark.	BSA	7,5	0,7	17	7,3	7,71	3,35	5,2866	2994,47
Brajevic ve ark.	UFA	3,5	0,7	17	7,3	7,71	3,35	5,2866	2994,47
Zhang ve ark.	DEDS	3,5	0,7	17	7,3	7,71	3,35	5,2866	2994,47
Kashan ve ark.	LCA	3,5	0,7	17	7,3	7,8	3,35	5,2866	2996,34
Mevcut çalışma	GCOA6	3,49	0,7	17	7,3	7,71	3,35	5,2866	2994,47

Çizelge 4.19. Dişli güç iletim tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Dişli Güç İletim Tasarım Problemi	En kötü	3,08E-33	1,733E-31
	Ortalama	3,08E-33	5,7E-32
	En iyi	3,08E-33	0,00E+00
	Standart Sapma	0	1,000E-31

Dişli güç iletim tasarım problemi (bkz. Şekil 3.28) için geliştirilmiş algoritmanın (GCOA6) algoritmasının sonuçları literatürden birkaç algoritma ile karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir.

Çizelge 4.19.1. Dişli güç iletim tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	Metot	x1(d)	x2(m)	x3(z)	x4(l1)	En uygun Sonuç
Akay and Karaboga	ABC	49	16	19	43	2,700857E−12
Mirjali ve ark.	GWO	57	31	13	49	9,9398E−11
Kennedy ve ark.	PSO	34	13	20	53	2,3078E−11
Mirjali	SCA	51	15	26	53	2,3078E−11
Kannan ve ark.	ALM	33	13	15	41	2,4070E−08
Wu ve ark.	GA	33	14	17	50	1,3620E−09
Mirjali ve ark.	SSA	57	13	31	49	9,9399E−11
Mirjali	MFO	51	16	23	50	1,1834E−09
Mirjali ve ark.	WOA	55	14	17	30	1,3616E−09
G.I. Sayed ve ark.	Chaotic SSA	46	26	12	47	9,9216E−10
Sadollah ve ark.	MBA	43	16	19	49	2,7008E–12
Brajevic ve ark.	UFA	49	16	19	43	2,7008E–12
Gandomi ve ark.	CS	43	16	19	49	2,70E–12
Mevcut çalışma	GCOA6	32	12	17,09	44,46	3,08E-33

Çizelge 4.20. Üç çubuk kafes yapı tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Üç çubuk kafes yapı tasarım problemi	En iyi	186,6025	186,6025

Çizelge 4.20.1. Üç çubuk kafes yapı tasarım problemi (bkz. Şekil 3.29) için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Arastırmacı	Metot	x1	x2	En uygun Sonuç
Askarzadeh	CSA	0,788675128	0,4082483080	263,895843
Liu ve ark.	PSO-DE	0,7886751347	0,4082482900	263,895843
Mohammed	NDE	0,7886753196	0,4082477671	263,895843
Wang ve ark.	BSA	0,788675	0,408248	263,895843

Çizelge 4.20.1. Üç çubuk kafes yapı tasarım problemi (Şekil 3.29) için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması (devam)

Arastırmacı	Metot	x1	x2	En uygun Sonuç
Zhang ve ark.	DE	0,7886751359	0,4082482868	263,895843
Mirjali	SCA	0,78394	0,42219	263,9506
Gupta ve ark.	IGWO	0,8087	0,3592	264,678
Sadollah ve ark.	MBA	0,788565	0,4085597	263,895852
Hussien	OBSSA	0,78866	0,40819	263,895812
Gandomi ve ark.	CS	0,78867	0,40902	263,97156
Mevcut Çalışma	GCOA6	0,78863	0,28874	186,6025

Üç çubuk kafes yapı tasarım problemi için gelişmiş COA algoritmasının sonuçları literatürden birkaç algoritmalarla karşılaştırmalı olarak yukarıdaki tabloda gösterilmiştir. Literatürdeki algoritmalar arasında yeni algoritma fark edilebilir seviyede iyi sonuçlar vermiştir.

Çizelge 4.21. Konsol kiriş tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Konsol Kiriş Tasarım Problemi	En kötü	62020,00	65393,04
	Ortalama	62020,00	64411,77
	En iyi	62020,00	63884,80
	Standart Sapma	0,00000	850,57590

Konsol kiriş tasarım problemi (bkz. Şekil 3.30) için geliştirilmiş algoritmanın (GCOA6) algoritmasının sonuçları literatürden algoritmalarla aşağıdaki tabloda karşılaştırılmalı olarak gösterilmiştir.

Çizelge 4.21.1. Konsol kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	Metot	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	En Uygun Sonuç
Chakri ve ark.	dBA	3	61	2,8	56,4	2,5	50,6	2,2	44,1	1,75	35	63113,61
Dhadwal ve ark.	PSO	3	60	3,1	55	2,6	50	2,3	45,6	1,75	35	64578,374
Bernardino ve ark.	GA	3	60	3,1	55	2,6	50	2,3	45,6	1,75	35,1	64599,65
Patel ve ark.	ADEA	3	61	2,8	56,3	2,5	50,5	2,2	44,1	1,75	35	63109,163

Çizelge 4.21.1. Konsol kiriş tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması (devam)

Araştırmacı	Metot	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	En Uygun Sonuç
Gandomi ve ark.	FA	3	60	3,1	55	2,6	50	2,2	44,1	1,75	35	63893,525
Brajevic ve ark.	E-FA	3	60	3,1	55	2,6	50	2,3	45,6	1,75	35	64578,194
Lemonge ve ark.	APM	3	60	3,1	55	2,6	50	2,3	45,6	1,79	34,6	64698,56
Runarsson ve ark.	SR	3	60	3,1	55	2,6	50	2,3	45,6	1,75	35,1	64599,65
Karaboğa ve ark.	ABC	3	60	3,1	55	2,6	50	2,3	45,6	1,76	25	64599,67
Sharma ve ark.	I-ABC	3	60	3,1	55	2,6	50	2,3	45,5	2,07	35,1	64599,65
Mevcut çalışma	GCOA6	3	60	2,8	55,6	2,5	50,5	2,2	44,1	1,75	35	62020,00

Çizelge 4.22. Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi için önerilen algoritmanın mevcut algoritma ile kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi	En kötü	0,2494	0,38965
	Ortalama	0,2494	0,38965
	En iyi	0,24940	0,38965
	Standart Sapma	0	0,00000

Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi (bkz. Şekil 3.31) için geliştirilmiş algoritmanın (GCOA6) sonuçları literatürde çalışılan en son algoritmalarla aşağıdaki tabloda gösterilmiştir.

Çizelge 4.22.1. Çoklu Disk Kavramalı Fren (MDCB) tasarımının optimizasyon problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	Metot	x1	x2	x3	x4	x5	En uygun Sonuç
Deb ve ark.	NSGA-II	70	90	1,5	1000	3	0,4704
Rao ve ark.	TLBO	70	90	1,5	1000	3	0,3136
Eskander ve ark.	WCA	70	90	1,5	1000	3	0,3136
Yang	APSO	70	90	1,5	1000	3	0,3371
Sharma ve ark.	ABC	70	90	1,5	1000	3	0,3176
Sharma ve ark.	I-ABC	70	90	1,5	1000	3	0,3137
Mirjalili ve ark.	SSA	77,1459	97,2218	1	628,1937	3,3809	0,3758
Azizyan ve ark.	FSO	70	90	1	870	3	0,3136
Mirjalili	SCA	68,8526	90	1	1000	2,6774	0,3027
Khishe ve ark.	Chimp	69,8782	90	1	1000	2,6537	0,288
Geetha ve ark.	MPSO	70	90	1	1000	2,3128	0,2598
Mevcut çalışma	GCOA6	70	90	1	990	2,18	0,2494

Çizelge 4.23. Himmelblau nonlinear problemi için önerilen algoritmanın mevcut algoritma ile kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Himmelblau nonlinear problem	En kötü	-32217,431	-32217,431
	Ortalama	-32217,431	-32217,431
	En iyi	-32217,431	-32217,431
	Standart Sapma	0	0

Himmelblau nonlinear problemi (MTP9) için geliştirilmiş algoritmanın (GCOA6) sonuçları literatürden birkaç algoritmayla karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir.

Çizelge 4.23.1. Himmelblau nonlinear problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

Araştırmacı	**Metot**	**x1**	**x2**	**x3**	**x4**	**x5**	**En uygun Sonuç**
Himmelblau	GRG	78	33,44	31,07	44,18	35,22	−30373,949
Gen ve ark.	GA	81,49	34,09	31,24	42,2	34,37	−30183,576
Runarsson ve ark.	ES	78	33	29,99526	45	36,7758	−30665,539
He ve ark.	IPSO	78	33	29,99526	45	36,7758	−30665,539
Lee ve ark.	HS	78	33	29,995	45	36,776	−30665,5
Yang ve ark.	BA	78	33	29,99552	45	36,77521	−30665,492
Manoj ve ark.	PSGA	78	33	29,99552	45	36,7758	−30665,539
Coello	GA2	78,049	33,007	27,081	45	44,94	−31020,859
Gandomi ve ark.	CS	78	33	29,99616	45	36,776	−30665,233
Garg	ABC	78	33	29,995	45	36,775	−30665,566
Mevcut çalışma	GCOA6	78	34,38	27	42,59	27	-32217,431

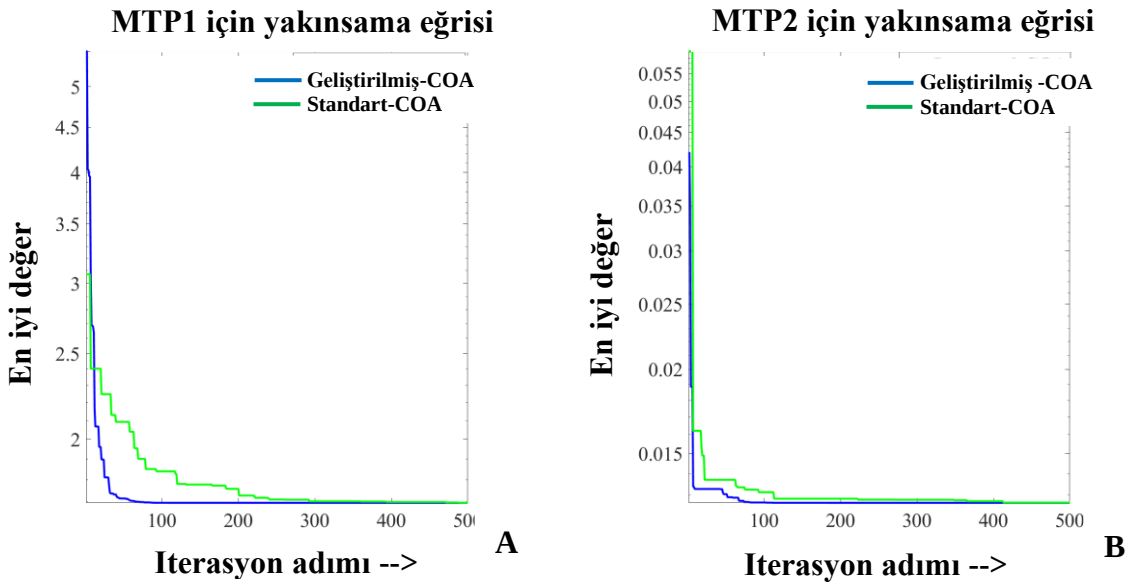
Çizelge 4.24. Küresel rulman tasarım problemi için önerilen algoritmanın mevcut algoritma ile istatiksel kıyaslanması

Geliştirilen en iyi algoritma ile standart algoritma karşılaştırması			
Fonksiyon	**Metrik**	**GCOA6**	**COA**
Küresel rulman tasarım problemi	En kötü	87510,91	85220,68
	Ortalama	87510,91	85421,42
	En iyi	87510,91	85537,48
	Standart Sapma	0,00000	174,55410

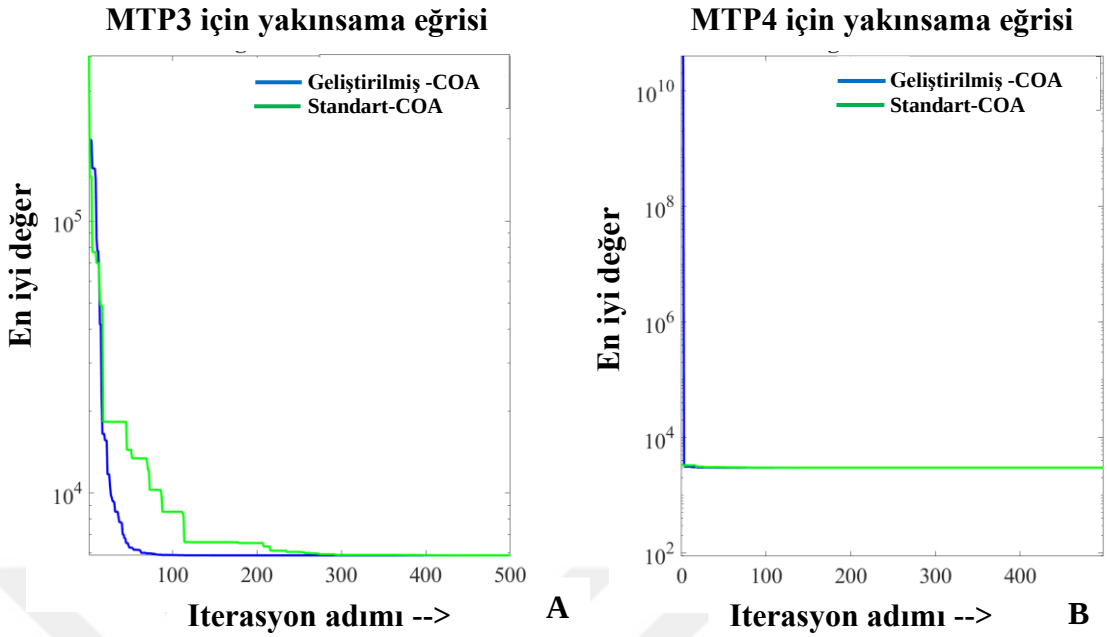
Küresel rulman tasarım problemi (bkz. Şekil 3.32) için geliştirilmiş algoritmanın (GCOA6) sonuçları literatürden birkaç algoritma ile karşılaştırmalı olarak aşağıdaki tabloda gösterilmiştir.

Çizelge 4.24.1. Küresel rulman tasarım problemi için önerilen algoritmanın literatürdeki algoritmalar ile kıyaslanması

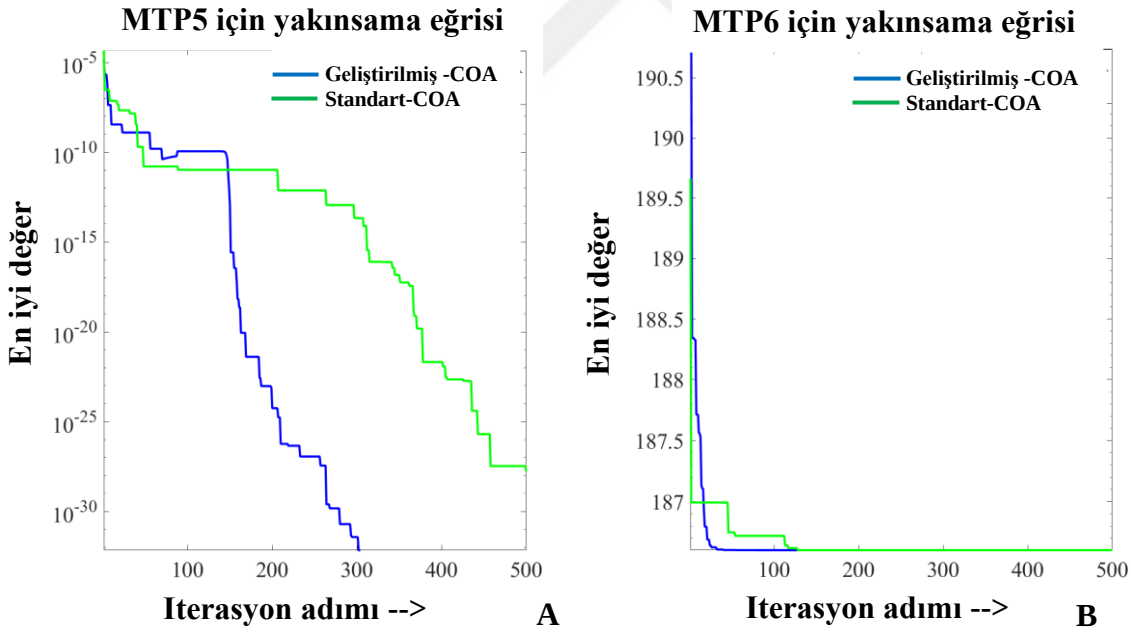
Araştırmacı	Metot	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	En uygun Sonuç
Gubta ve ark.	GA4	125,7	21,42	11	0,515	0,515	0,41	0,65	0,3	0,02	0,75	81 843,30
Rao ve ark.	TLBO	125,7	21,42	11	0,515	0,515	0,42	0,63	0,3	0,07	0,8	81 859,74
Eskander ve ark.	WCA	125,7	21,42	11	0,515	0,515	0,40	0,66	0,3	0,04	0,6	85 538,48
Sadollah ve ark.	MBA	125,7	21,42	11	0,515	0,515	0,49	0,62	0,3	0,09	0,64	85 535,96
Mohamed	NDE	125,7	21,42	11	0,515	0,515	0,46	0,61	0,3	0,04	0,65	85 549,23
Heidari ve ark.	HHO	125	21	11,1	0,515	0,515	0,4	0,6	0,3	0,05	0,6	83 011,88
Savsani ve ark.	PVS	125,7	21,42	11	0,515	0,515	0,4	0,68	0,3	0,08	0,7	81 859,74
Gong ve ark.	IMDDE	125,7	21,42	11,4	0,515	0,515	0,44	0,63	0,3	0,08	0,67	81 859,73
Mirjali	MVO	125,6	21,42	11	0,515	0,515	0,5	0,68	0,3	0,05	0,6	81 765,8
Mirjali	SCA	125	21,14	11	0,521	0,521	0,5	0,7	0,3	0,03	0,63	68 945,2
Ong ve ark.	CPA	125,7	21,42	11	0,515	0,515	0,47	0,61	0,3	0,09	0,68	81 849,21
Mevcut çalışma	GCOA6	125	21,87	10,77	0,515	0,515	0,415	0,68	0,3	0,045	0,60	87 510,91



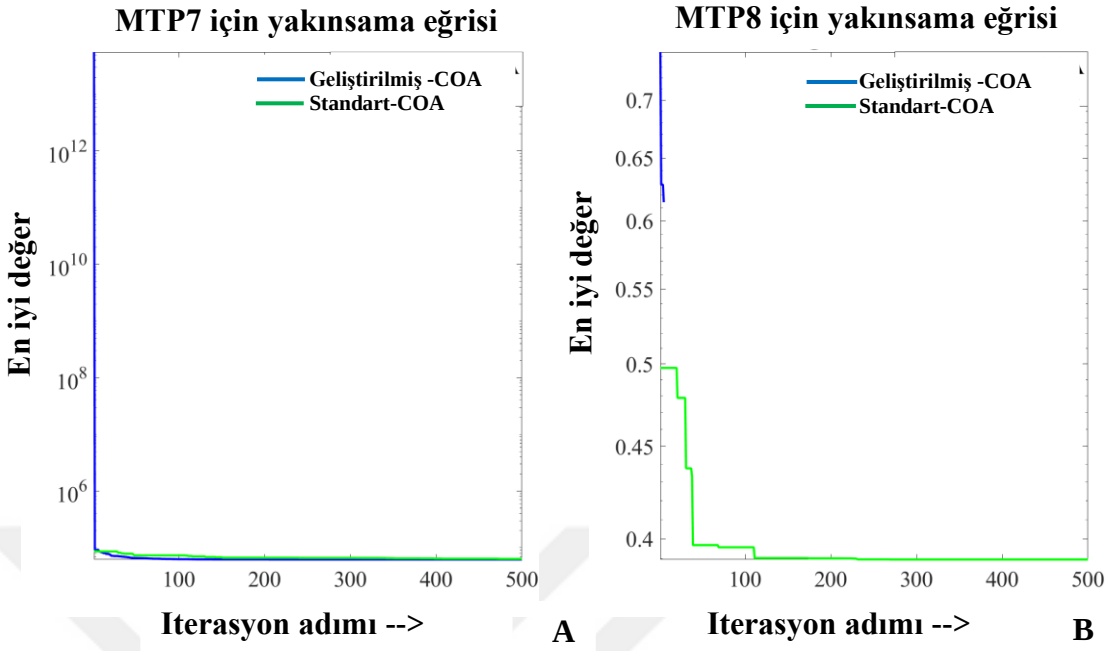
Şekil 4.83. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi. **A)** Mtp1 kaynaklı kiriş tasarım problemi **B)** Mtp2 bası yayı tasarım problemi



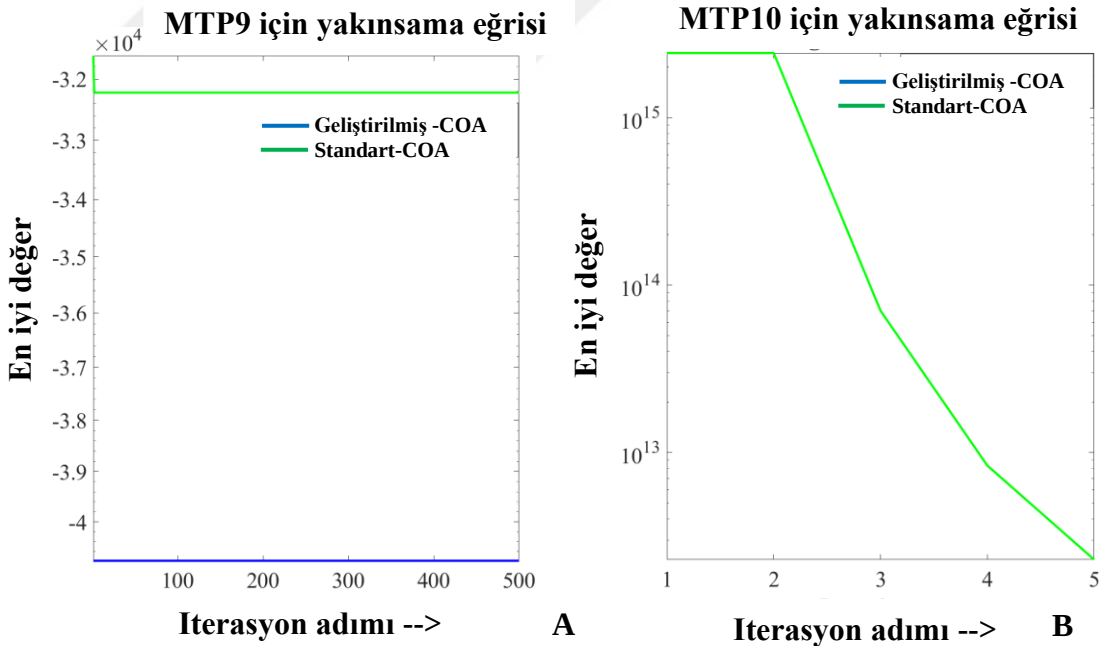
Şekil 4.84. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi. **A)** Mtp3 basınçlı kap tasarım problemi **B)** Mtp4 hız düşürücü tasarım problemi



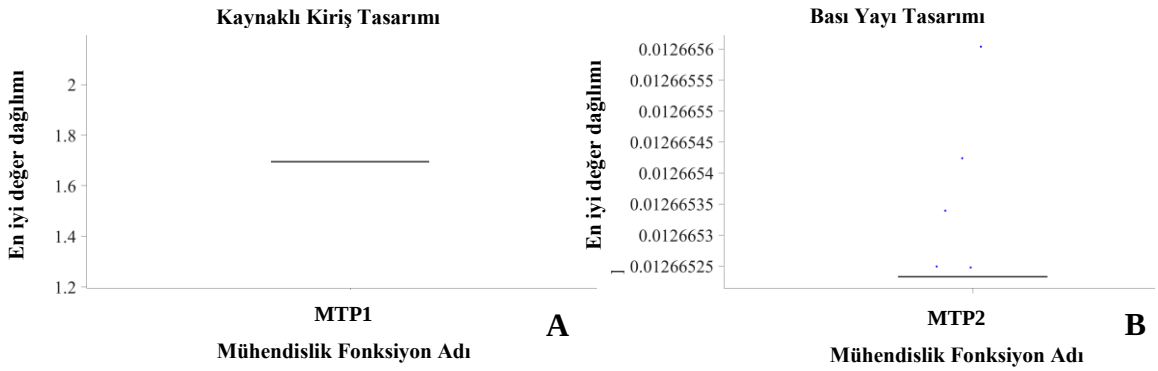
Şekil 4.85. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi. **A)** Mtp5 dişli güç iletim tasarım problemi **B)** Mtp6 üç çubuk kafes yapı problemi



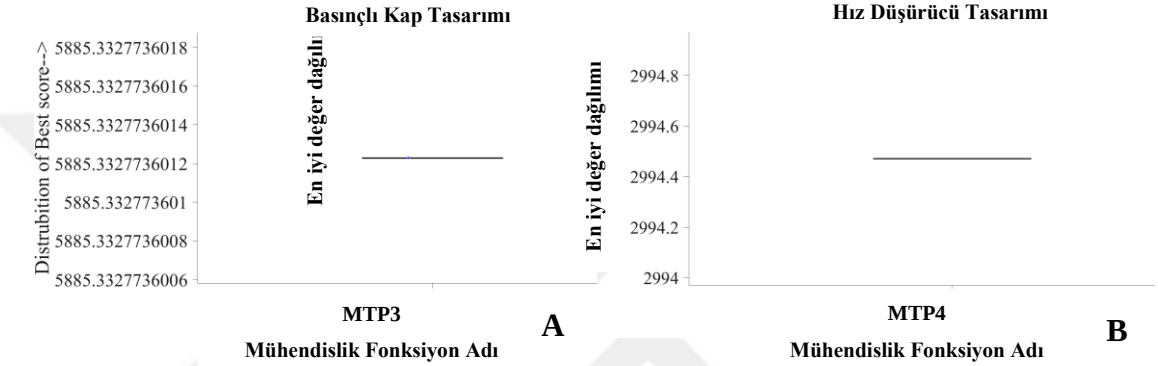
Şekil 4.86. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi. **A)** Mtp7 konsol kiriş tasarım problemi **B)** Mtp8 Çoklu disk kavramalı fren problemi



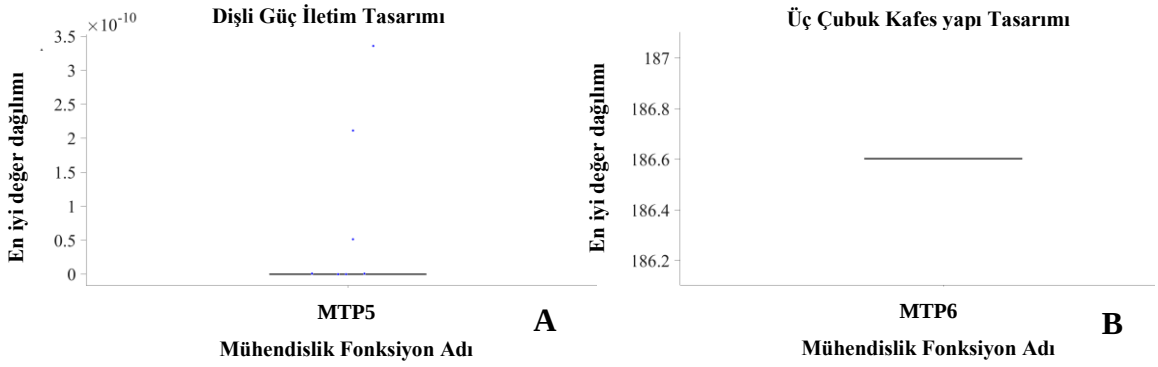
Şekil 4.87. Geliştirilmiş ve standart algoritmanın karşılaştırmalı yakınsama eğrisi. **A)** Mtp9 Himmelblau nonlinear problemi **B)** Mtp10 Küresel rulman tasarım problemi



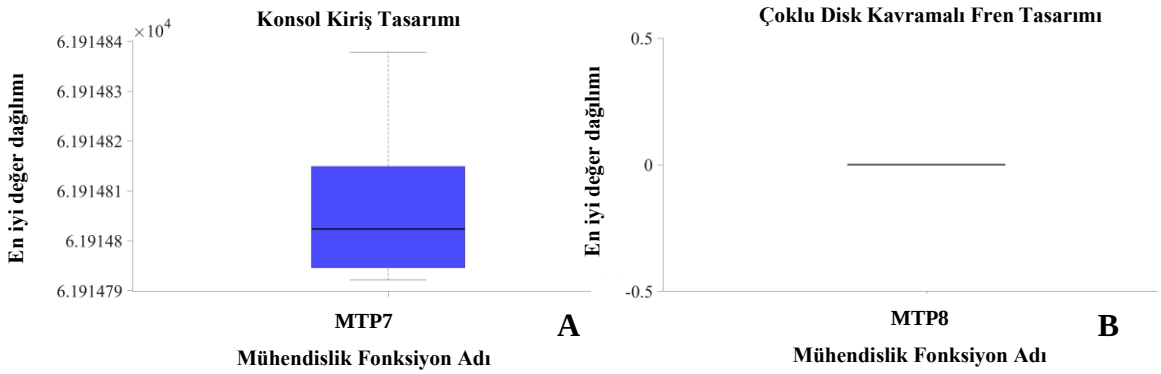
Şekil 4.88. Geliştirilmiş algoritma en iyi değer dağılımı.
A) Mtp1 kaynaklı kiriş tasarım problemi **B)** Mtp2 bası yayı tasarım problemi



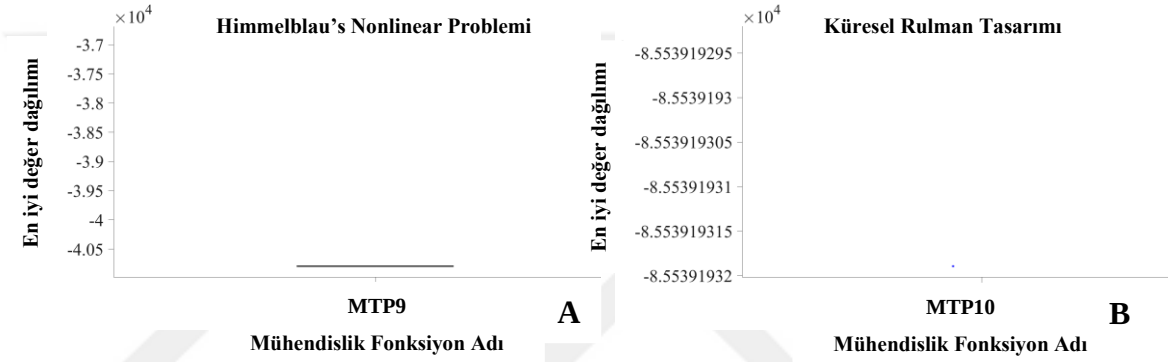
Şekil 4.89. Geliştirilmiş algoritma en iyi değer dağılımı.
A) Mtp3 basınçlı kap tasarım problemi **B)** Mtp4 hız düşürücü tasarım problemi



Şekil 4.90. Geliştirilmiş algoritma en iyi değer dağılımı.
A) Mtp5 dişli güç iletim tasarım problemi **B)** Mtp6 üç çubuk kafes yapı problemi



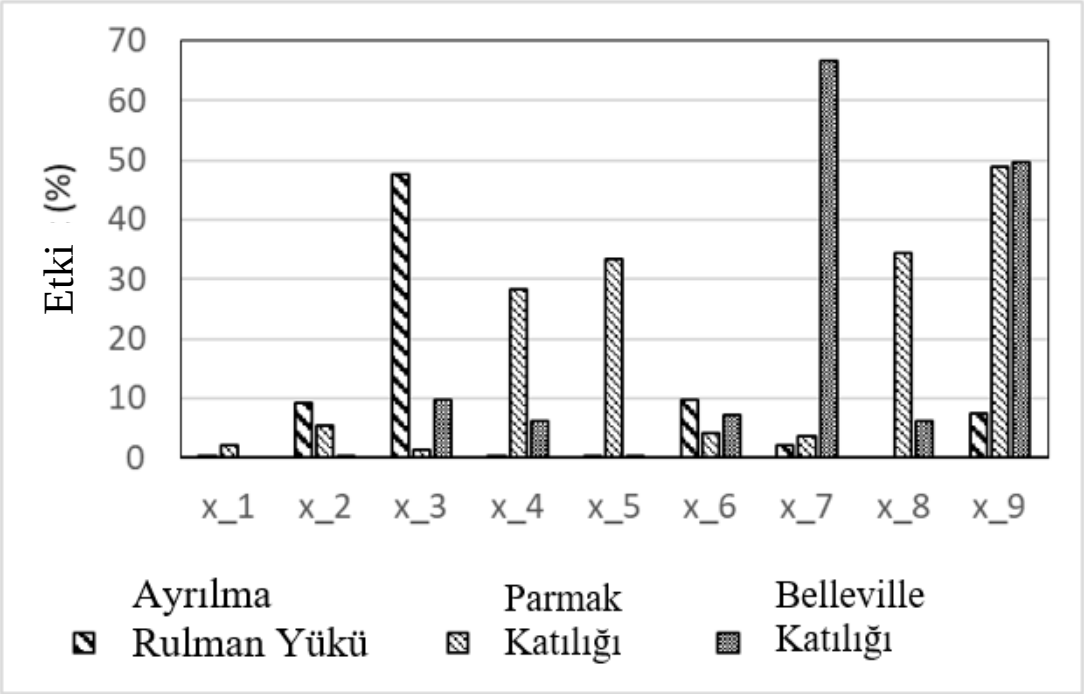
Şekil 4.91. Geliştirilmiş algoritma en iyi değer dağılımı.
A) Mtp7 konsol kiriş tasarım problemi **B)** Mtp8 Çoklu disk kavramalı fren problemi



Şekil 4.92. Geliştirilmiş algoritma en iyi değer dağılımı.
A) Mtp9 Himmelblau nonlinear problemi **B)** Mtp10 Küresel rulman tasarım problemi

4.1. Diyafram Parametre Etki Analizi Sonuçları

Bu optimizasyon çalışmasında temel olarak üç amaç (minimum ayrılma rulman yükü kaybı, maksimum parmak katılığı ve baskı yükü katılığı) vardır ve bu nedenle uygun optimizasyon yönünü araştırmak için bir parametre etki analizi şiddetle gereklidir. Daha önce açıklandığı gibi, Ansys ® optiSlang parametre etki analizi yapıldı ve dokuz tasarım parametresinin üç hedef üzerindeki etkisi aşağıdaki şekilde Şekil 4.93'te gösterilmektedir.



Şekil 4.93. Ansys optiSlang yazılımına göre parametre etki analizi sonuçları.

Şekil 4.93'te görüldüğü gibi, x3 (iç yarıçap, bkz. Şekil 3.33 B) ayırma rulman yükü üzerinde en büyük etkiye sahipken, x2, x6 ve x9'un etkisi çok az ve diğer parametrelerin ayırma rulman yükü üzerinde ihmal edilebilir bir etkisi vardır. Kavrama katılığı için x7 (kabartma sonu) ve x9 (kabartma derinliği) önemli bir etkiye sahiptir. X4 (büyük delik yüksekliği), x5 (küçük delik yüksekliği), x8 (kabartma başlangıcı) ve x9 (kabartma derinliği) diyafram parmak katılığında önemli bir etkiye sahiptir. Diyafram parmak katılığı için x4, x5, x8 ve x9, parametreleri arasında x9 en yüksek etkiye sahiptir. Debriyaj performansının bütünlüğünü amaç fonksiyonu temsil eder ve ayırma rulman yükünün korunması kavrama/devreden çıkarma işlevselliği için en önemli faktördür. Bu nedenle optimum tasarımlardan en az kayıplı ayırma rulman yükü seçildi. Prototip üretiminde x3 hassasiyetine özen gösterildi.

4.2. Diyafram Optimum Tasarım ve Prototip Üretimi

Geliştirilmiş kır kurdu algoritması x1, x2, x3, x5, x6 ve x9 için tasarım limitlerinde en iyi değerlere ulaşırken, x4, x7 ve x8 için tasarım limitleri arasında en uygun değerleri verir.

FEM ve testleri temel alan nihai tasarım değişkenleri ve amaç fonksiyon değerleri aşağıdaki Çizelge 4.25'te gösterilmektedir.

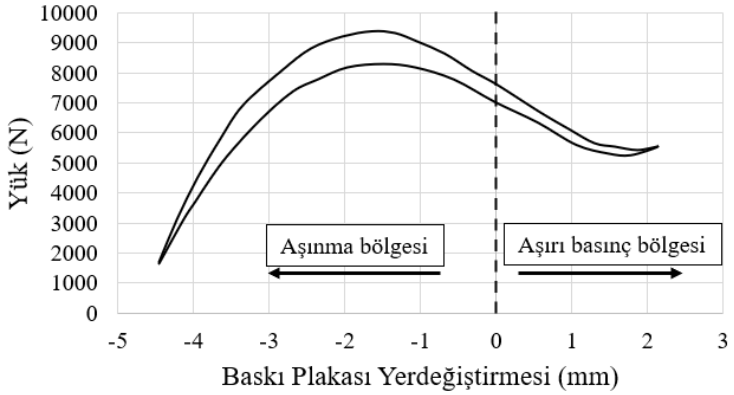
Çizelge 4.25. Optimum tasarım değişkenleri ve elde edilen amaç fonksiyon değerleri

Değişkenler		**Eniyilenmiş Parametreler**
Parmak yüksekliği	X1 (mm)	4,1
İç açı	X2 (°)	14
İkinci iç yarıçap	X3 (mm)	3,2
Büyük delik yüksekliği	X4 (mm)	21,5
Küçük delik yüksekliği	X5 (mm)	11,5
Parmak ucu genişliği	X6 (mm)	0,81
Kabartma sonu yarıçapı	X7 (mm)	173
Kabartma başlangıcı yarıçapı	X8 (mm)	88,09
Kabartma derinliği	X9 (mm)	4,25
Cevaplar		**FEM Sonuçları**
Ayırma Rulman Yükü@6500Rpm	Y1 (N)	1612
Parmak Katılığı	Y2 (N/mm)	1624
Kavrama Katılığı	Y3 (N/mm)	6942

Eniyilenmiş parametrelere dayalı prototipler üretildi. Ancak şekil vermeden kaynaklı yırtılma problemlerinden dolayı kabartma derinliği üç nokta iki mm ile sınırlıdır. Şekil 4.94 A'da bir prototip örneği verilmiştir. Diyafram yayının kavrama yük-yer değiştirme özelliği Şekil 4.94 B'de verilmiştir. Baz modele göre optimizasyon sayesinde kavrama tepe gidiş yükü ile dönüş yükü arasında yaklaşık yüzde yirmi iki fark vardır.



A

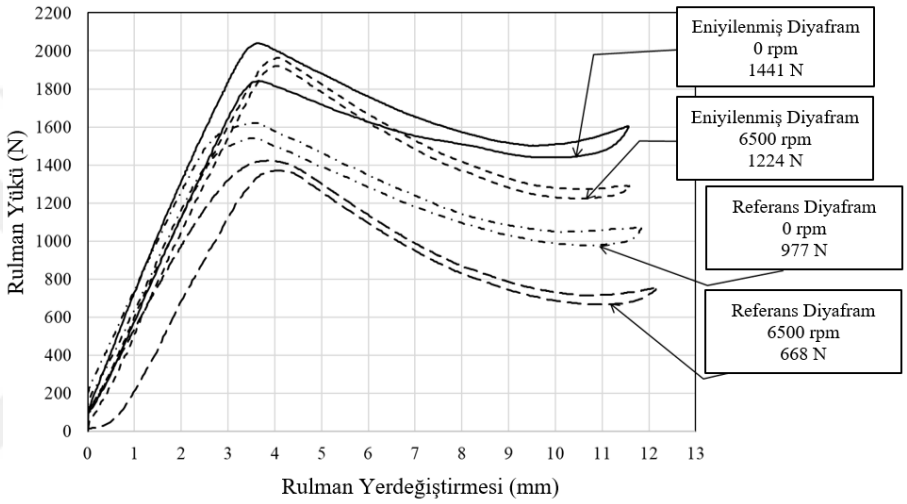


B

Şekil 4.94. Prototip sonuçları.
A) Eniyilenmiş tasarımın prototip resmi **B)** Baskı yük-yerdeğiştirme karakteristikleri

4.3. Diyafram Serbest Rulman Yük Taşıyan Yer Değiştirme Testi Sonuçları

Şekil 4.94 A'da gösterilen optimum boyutlara sahip prototip hem statik hem de dinamik (altı bin beş yüz rpm) koşullar için ayrılma rulman yükü özellikleri açısından test edilmiştir. Bu testin bir baskı plakası düzeneğindeki serbest bırakma rulman davranışını ölçtüğünü hatırlatarak, sadece parmak davranışını içeren Şekil 4.94 B'den farklı bir eğri özelliği gösterir. Mevcut ürün ile optimize edilmiş ürün arasında ayırma rulman yükü için bir karşılaştırma Şekil 4.95'te verilmiştir.



Şekil 4.95. Referans (noktalı çizgi) ve eniyilenmiş (düz çizgi) diyaframın ayırmarulman yükünün rulman yerdeğiştirmesine göre statik ve dinamik (6500 rpm) değişimi.

Rulman yükü maksimum noktaya kadar lineer olarak artar ve bu maksimum noktadan sonra lineer olmayan şekilde bir düşüş olur. Bu kayıp mümkün olduğunca dinamik durum ile statik durum arasında asgari olmalıdır ve bu çalışmada amaçlanan amaçlardan biridir. Şekilde görüldüğü gibi, optimizasyon ile iki şekilde önemli iyileştirmeler sağlanmaktadır. Birinde maksimum ve plato değerleri önemli ölçüde artırılır (dinamik durum için yüzde seksen üç) ve rulman yükündeki düşüş yüzde elli iki'den yüzde yirmi'ye (dinamik durum için) düşürülür. Bu iyileştirmeler, çok daha iyi ayırma işlevi ve diyafram aracılığıyla daha yüksek tork aktarımı anlamına gelir ve bu, optimizasyon yoluyla elde edilir. Azami ve asgari serbest bırakma rulman yükü arasındaki daha düşük fark, daha konforlu bir sürüş sağlayan daha iyi debriyaj anlamına gelir. Aksi takdirde, düşük ve yüksek hızlarda sürüş için farklı seviyelerde pedal kuvveti gerekir ve bu kuvvet sağlanamayacağı için kavrama

ayrılması gerçekleşmez. Kavrama katılığı grafiğin lineer kısmından ölçülebilir ve görüldüğü gibi dinamik durumda katılık statik duruma göre azalır ve bu hem optimize edilmiş hem de mevcut diyafram için geçerlidir. Gerçek testlere dayalı olarak hesaplanan parmak ve kavrama katılık değerlerinin karşılaştırması aşağıdaki Çizelge 4.26'da verilmektedir. Görüldüğü gibi optimize edilmiş ürün mevcut ürüne göre statik durum için yüzde on beş, dinamik durum için yüzde on üç artış sağlanmıştır.

Çizelge 4.26. Optimum tasarım değişkenleri ve elde edilen kavrama katılık değerleri

Kavrama Katılık Değeri	Statik Durum	6500 rpm devir altında
Mevcut Ürün	525 N/mm	454 N/mm
En iyilenmiş Ürün	604 N/mm	516 N/mm

Gerçek testlere dayalı olarak optimize edilmiş modelle yapılan iyileştirmeler somut olarak kanıtlanmıştır; ancak teorik ve gerçek sonuçlar arasındaki olası sapma kaynakları gelecekteki araştırma alanlarından olacaktır.

DOE Çizelge 3.10'da listelenen tüm tasarımlar FEM ve eniyileme algoritması kullanılarak çalıştırılır. FEM'den elde edilen amaç fonksiyon değerleri elde edilir. Bu nedenle güvenilir bir FE modeli çok önemlidir. Burada test edilen nihai tasarım, FE simülasyonu ile de elde edilmiş ve test ile FE sonuçları arasında minimum rulman yükü değerlerinde yaklaşık yüzde on iki'lik fark gözlemlenmiştir. Bu fark makul kabul edilir çünkü FEM mesh ile ilgili bazı yaklaşımlara öncülük eder ve test koşulları FE simülasyonunda tam olarak temsil edilemez. Örneğin, baskı plakası kapağı ve dayanak halkaları bir miktar esnekliğe sahiptir, ancak diyaframın bu parçalarla temasları, bu parçalar rijitmiş gibi kabul edilir. Bu parçaların esnek olarak modellenmesi, her deney için çözüm süresini önemli ölçüde artırabilir ve eniyileme sürecini hesaplama açısından çok maliyetli hale getirebilir. Her şeyden önce, optiSlang'ın parametre etkisi analizi, diyaframın fiziği hakkında fikir verir. Rulman yer değiştirmesine göre serbest rulman yükü kaybı ikinci iç yarıçaptan (bkz. Şekil 3.33 B) önemli ölçüde etkilenir; burada yarıçap küçüldükçe rulman kuvveti kaybı da küçülür. Bu rulman ayırma kuvveti kabartma geometrisinden ziyade diyafram parmak geometrisinden güçlü bir şekilde etkilendiğini gösterir. Öte yandan, kabartma uzunluğu ve derinliğinden etkilenen artan parmak katılığı ve kavrama katılığı daha yüksek kavrama yüküne yol açar. Bunun sonucu olarak daha iyi tork aktarımı ve daha iyi kavrama davranışı sağlanır. Bu nedenle, optimize edilmiş model

mevcut modelle aynı hacmi kaplamasana rağmen dinamik koşullarda daha yüksek ayırma rulman kuvvetine sahiptir. Statik durumdaki rulman kuvveti azami motor devirlerini içeren dinamik durumlarda neredeyse yarı yarıya azalır. Yüksek hızlı dönüşlerde merkezkaç etkileri diyaframı uygulanan kuvvet yönünde iter ve bu da rulman kuvvetinde daha fazla kayıpla sonuçlanır. Parmak katılığı arttıkça merkezkaç kuvveti ayırma rulman yükünü daha az etkiler. Bu dinamik durum için çok daha az rulman yükü kaybı ile sonuçlanır. Yüksek hızlı dönüşler için daha iyi kavrama özellikleri sağlanır. Bu durum yeni tasarımı yüksek hızlarda konforlu ve emniyetli sürüşe önem verilen spor otomobil modelleri için daha iyi bir aday haline getirir. Parametre etki analizi ile birlikte, parmak profili (ikinci iç yarıçap) rulman yük kaybı için en etkili parametre, kabartma profili (uzunluk ve derinlik) diyafram parmak ve kavrama katılığı için en önemli parametre olarak belirlenmiştir. Bu çalışma, gerçek prototip testleri ile bunu benzersiz bir şekilde doğrularken, daha iyi ve sağlam bir ürün tasarımı için optimizasyon yaklaşımının etkinliğini göstermektedir.

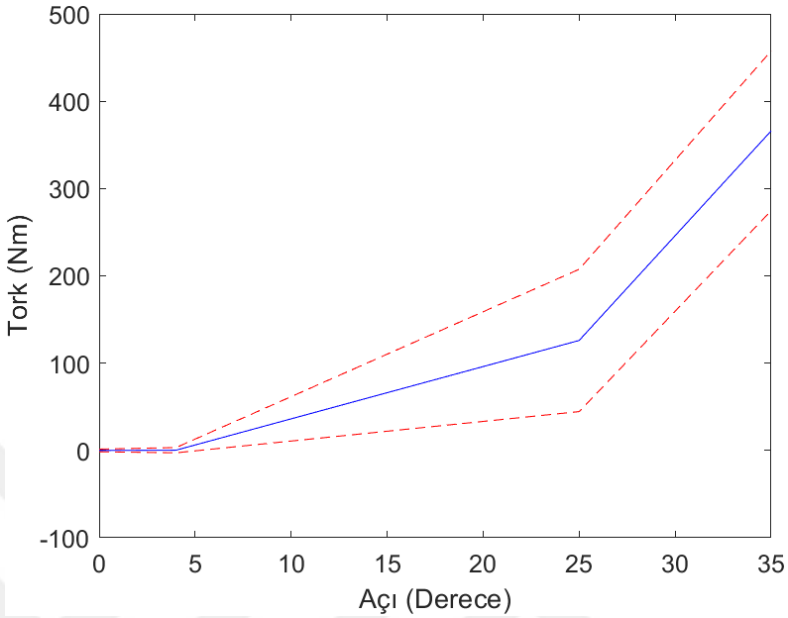
4.4. Debriyaj Sisteminin Bir Boyutlu Modelinin Optimizasyon Sonuçları

Matlab ara yüzünden geliştirilmiş kır kurdu algoritmasının çalıştırılıp bir boyutlu modelde simülasyonu gerçekleştirildiğinde eniyilenmiş sürtünme torku değerleri aşağıdaki gibidir.

Çizelge 4.27. Geliştirilmiş kır kurdu algoritması ile eniyilenmiş model parametreleri

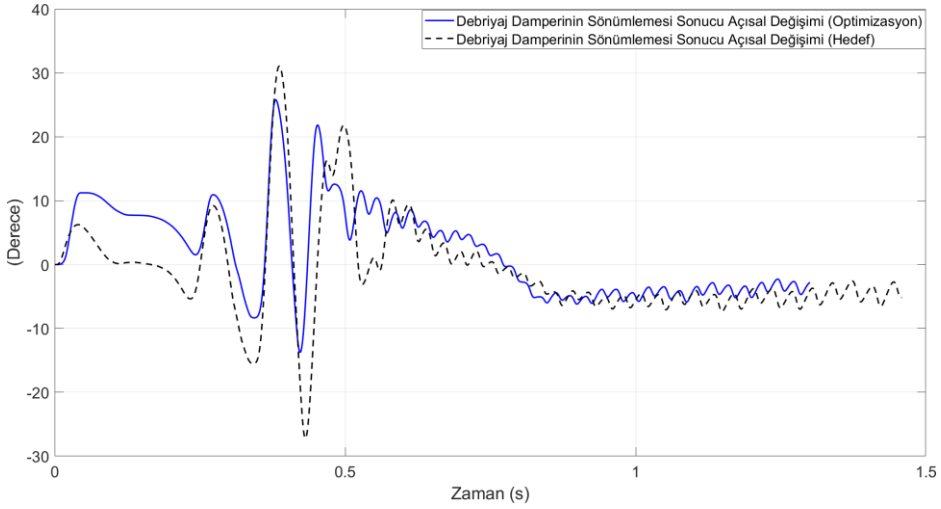
Değişken Sembolü	**Değişken Adı**	**Değer**	**Optimizasyon Alt ve Üst Limiti**	
Im	Motor Atalet Momenti (kg.m^2)	0,03		
Iv	Volan Atalet Momenti (kg.m^2)	0,13		
Id	Damper Atalet Momenti (kg.m^2)	0,03		
Ib	Baskı Komplesi Atalet Momenti (kg.m^2)	0,055		
A1	Damper açısı 1. Kademe (°derece)	4		
A2	Damper açısı 2. Kademe (°derece)	25		
A3	Damper açısı 3. Kademe (°derece)	35		
K1	Damper yay katılığı 1. Kademe (Nm/°)	0,3		
K2	Damper yay katılığı 2. Kademe (Nm/°)	6		
K3	Damper yay katılığı 3. Kademe (Nm/°)	24		
Değişken Sembolü	**Değişken Adı**	**Değer**	**Alt Limit**	**Üst Limit**
H1	Damper Sürtünme Momenti 1. Kademe (Nm)	3	1	3
H2	Damper Sürtünme Momenti 2. Kademe (Nm)	160	80	160
H3	Damper Sürtünme Momenti 3. Kademe (Nm)	180	180	240

Motor titreşimlerini sönümlemek için gereken optimize edilmiş damper eğrisi aşağıdaki gibidir.



Şekil 4.96. Optimize edilmiş damper tork & açı eğrisi.

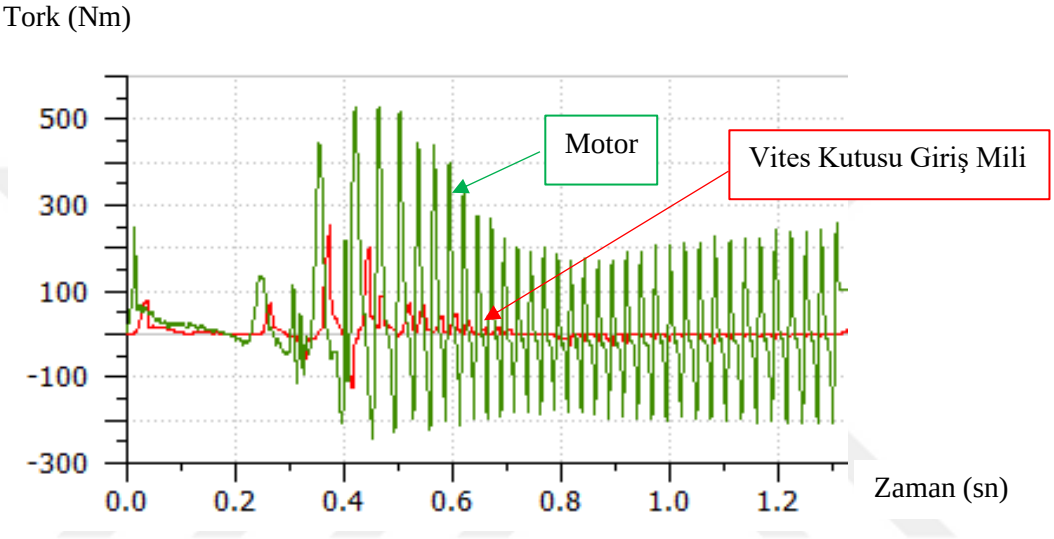
Debriyaj sisteminin optimizasyonu sonucunda damper sisteminin eksi on dört derece ve artı yirmi altı derece arasında çalışarak pozitif bölgede iki yüz otuz dört Nm ve negatif bölgede yüz dört Nm torka kadar sönümlediği sonucu çıkmıştır. Motor torku ile sönümleme sonrası tork arasındaki karakteristik aşağıdaki gibidir.



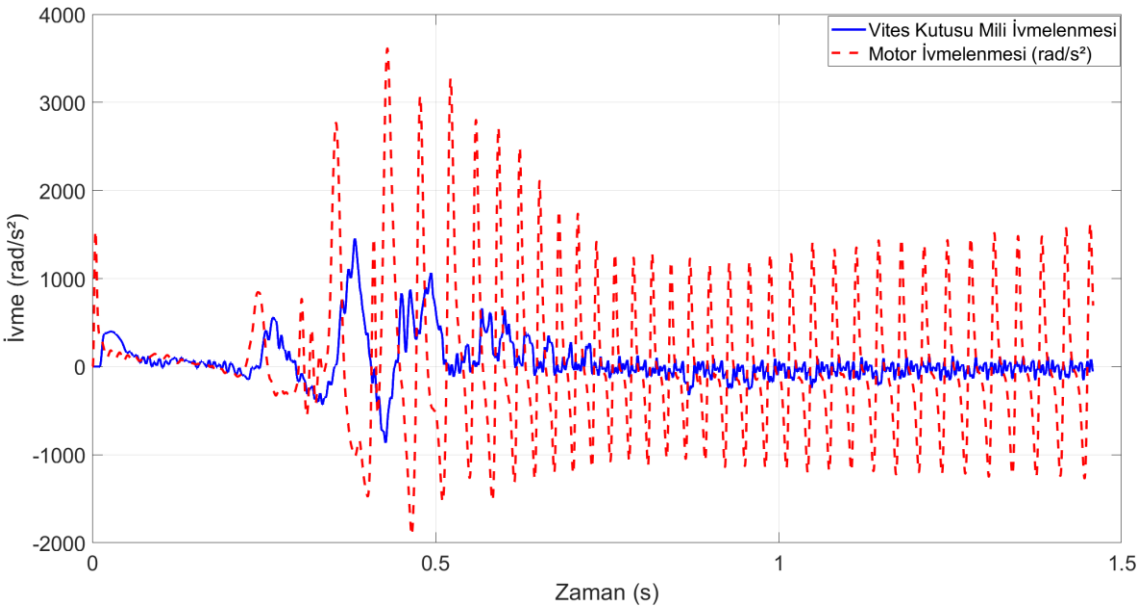
Şekil 4.97. Hedeflenen sönümleme eğrisi ile optimizasyon sonrası çıkan eğri arasındaki uyum.

$$\text{Uyum} = \frac{(\text{Simülasyon} - \text{Hedef})^2}{\text{Hedef}} \tag{4.91}$$

Motordan oluşan titreşimleri debriyaj damper elemanında sönümleme gerçekleştirirken damper yaklaşık sıfır nokta bir sn'lik kısımda yüksek açısal yer değiştirmeler göstererek titreşimleri sönümleyerek kararlı hale gelmiştir. Bu durum başarılı bir çalışmayı temsil etmektedir.

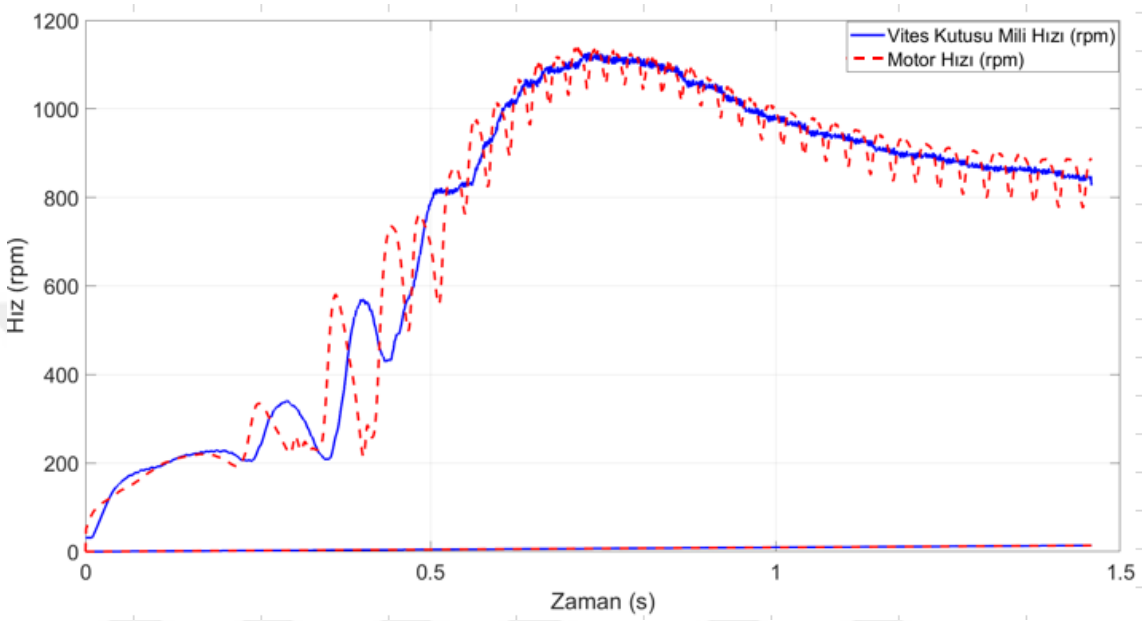


Şekil 4.98. Optimize edilmiş damper ile sönümlenen tork.



Şekil 4.99. Motor hızı ile damper hızlanması karakteristiği.

Sürücü anahtarı çevirmesiyle marş motoru volanı tahrik edip motorun çalışması ile beraber debriyaj damper sistemi motordaki hızı vites kutusu giriş miline titreşimsiz bir şekilde hızı ve torku transfer eder. Aşağıda vites kutusu giriş mili hızının sönümlenmiş şekilde transferi görülmektedir.



Şekil 4.100. Vites kutusu hızı ile damper hızlanması karakteristiği.

Burada motordan dalgalı şekilde çıkan hız karakteristiğinin damper sonrasında dalgalanmanın kaybolduğu net bir şekilde görülmektedir. Bu durum motordan çıkan hızın daha düzgün bir şekilde vites kutusuna aktarıldığını belirtmektedir. Bu çalışma sonucunda debriyaj damper sisteminin sürtünme torku etkisi ile motordan vites kutusuna titreşimlerin sönümlenmesi optimizasyon algoritması kullanarak belirlendi ve bir boyutlu simulasyon ile entegre edilerek doğrulandı.

5. SONUÇ

Bu çalışmada kır kurdu algoritmasının performansını iyileştirmek temel amaçtır. Algoritma performansını arttırmak için mesafe kontrollü her kır kurdunun uyumunun en iyi kır kurdunun uyumuna göre sürünün genel davranışına etkisi yeni popülasyon oluşumuna eklenmiştir. Popülasyon oluşumu için alt ve üst sınırla arasında kaotik düzende değişken sağlanarak sürünün ilk konum çeşitliliğindeki zayıflık giderilmiştir. Popülasyon oluşumu, popülasyon güncellenmesi, levy uçuş dağılımı prensibine göre rassal şekilde parametrelendirilmiş yeni doğan bireylerin doğuşu, sürüler arasında geçişler, bölgesel en iyi kır kurdunun aranmasında kaotik düzende organize edilmiştir. Bu sayede algoritmanın bölgesel en iyi noktalarda takılması engellenerek erken yakınsama problemi engellenip sürü içerisinde çeşitlilik arttırılarak algoritma arama uzayı arttırılıp bilginin verimli kullanılması sağlanmıştır. Bu süreçte algoritmaya laplace çaprazlama stratejisine göre ikinci bir doğum ölüm prosesi eklenerek çözüm çeşitliliği arttırılıp algoritma performansı arttırılmıştır. Çalışmanın performansı arttırmak için laplace çaprazlama, uygunluk mesafe dengesi, levy uçuş, kaotik haritalar ve kaotik yerel arama stratejisi olmak üzere beş farklı stratejiden yararlanılmıştır. Deneysel sonuçlara göre, GCOA, karakteristiğinin güçlü keşif kabiliyeti göstermesi ve yerel optimizasyondan kaçınması nedeniyle standart COA'dan daha üstündür. Bu karşılaştırma, sürekli küresel arama kabiliyeti ve güçlü optimizasyon kabiliyeti nedeniyle GCOA'nın genel performansının en iyisi olduğunu ortaya koymaktadır. Bu stratejiler algoritma oluşumunda kullanılarak farklı bileşimdeki algoritmalar kısıtsız optimizasyon problemleri için karşılaştırılmıştır. Ayrıca algoritma literatürde sıklıkla kullanılan yirmi üç kısıtsız test fonksiyonu, literatürde sıklıkla kullanılan on mühendislik problemi ve debriyaj taşıt elemanlarından diyafram yayının en iyileme problemi ile test edilmiştir. Sonuçlar standart kır kurdu algoritmasına göre analiz edildiğinde, geliştirilen algoritmanın çözüm kalitesi ve sağlamlığı açısından standart sapmalara dayalı istatistiksel verilere göre daha üstün olduğu görülmektedir. Problem boyutu arttıkça arama alanı arttıkça algoritma performansı azalmakta, ancak geliştirilmiş algoritmanın performansı standart algoritmaya göre üstünlüğünü korumaktadır. Elde edilen sonuçlara göre geliştirilen algoritmanın standart algoritmalardan daha iyi performans gösterdiği ve çalışmanın amacına ulaştığı görülmektedir. Bu tez çalışması ile literatüre etkin yeni bir

tekniğe sahip algoritma kazandırılmıştır. Çeşitliliği yüksek stratejiler ile tanıtılan geliştirmeler literatür için uygun olduğu gösterilmiştir. Taşıt debriyaj elemanlarından diyafram yayının dinamik koşullardaki en iyi ayrılma rulman yükü için diyafram parmak katılığı, kavrama katılığına ve geometrik limitlere bağlı olarak en iyilenmiştir. Diyafram yayının eniyilenmesi yüksek motor devirleri altında diyafram yayının eksenel yer değiştirmesi sonucunda oluşan diyafram ayrılma rulman yükü için gerçekleştirilmiştir. En uygun tasarımın üretilip test edilmesi sonrasında üretimde hali hazırda olan ürüne göre yüzde seksen iki iyileşme görülmüştür. Yapılan parametre analizi sonucunda diyafram ayrılma yükü için en etkili parametrenin parmak profili (ikinci iç yarıçap) olduğu görülmüştür. Gerçek prototip testleri ile tasarım doğrulanırken, daha iyi ve sağlam ürün tasarımları için eniyileme yaklaşımının etkinliğini göstermektedir. Diyafram yayının yüksek devir altındaki karakteristiğinin eniyilenmesinin yanında taşıtların çalıştırılması sırasında motordan gelen titreşimlerin sönümlenmesi amacıyla debriyaj damper elemanının sürtünme torku karakteristikleri en iyilenmiştir. Yapılan bu tez çalışması ile geliştirilmiş kır kurdu algoritmasının test problemleri ile doğrulanması ve taşıt debriyaj elemanlarının eniyilenmesi çalışmalarına uygulanması ile mevcut elemanların eniyilenerek yeniden tasarımları yapılmıştır. Önerilen bu yaklaşım ile dinamik şartlar altında yük davranışı ve aracı çalıştırma esnasında sönümleme karakteristiği göz önünde bulundurularak debriyaj elemanlarının optimizasyonunda kullanılmıştır. Geliştirilen algoritma literatürde ilk defa ürün geliştirme çalışmalarında kullanılmıştır. Bu çalışmada tasarım havuzu latin hiperküp örnekleme metodu ile oluşturulmuştur. Optimizasyon aşamasından sonra değerlerin bir boyutlu simülasyon ile doğrulanması yapılmıştır. Sonuçlara göre geliştirilmiş algoritma standart algoritmalardan daha iyi performans gösterdiği ve çalışmanın amacına ulaştığı açıktır.

KAYNAKLAR

Abaza, A., El-Sehiemy, R. A., Mahmoud, K., Lehtonen, M. & Darwish, M. M. F. (2021). *Optimal Estimation of Proton Exchange Membrane Fuel Cells Parameter Based on Coyote Optimization Algorithm.* Applied Sciences 11(5): 2052-2052.

Abaza, A., Fawzy, A., El-Sehiemy, R. A., Alghamdi, A. S. & Kamel, S. (2021). *Sensitive reactive power dispatch solution accomplished with renewable energy allocation using an enhanced coyote optimization algorithm.* Ain Shams Engineering Journal 12(2): 1723-1739.

Abdelghafar S., Goda E., Darwish A. ve Hassanien A. E. (2019) *Satellite Lithium-ion Battery Remaining Useful Life Estimation by Coyote Optimization Algorithm*. 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS). 2019. pp. 124-129. doi: 10.1109/ICICIS46948.2019.9014752.

Abdelwanis, M. I., Abaza, A., El-Sehiemy, R. A., Ibrahim, M. N. & Rezk, H. (2020). *Parameter Estimation of Electric Power Transformers Using Coyote Optimization Algorithm With Experimental Verification.* IEEE Access 8: 50036-50044.

Abou El-Ela, A. A., El-Sehiemy, R. A., Shaheen, A. M. & Diab, A. E.-G. (2021). *Enhanced coyote optimizer-based cascaded load frequency controllers in multi-area power systems with renewable.* Neural Computing and Applications 33(14): 8459-8477.

Akay, B. & Karaboga, D. (2012). *Artificial bee colony algorithm for large-scale problems and engineering design optimization.* Journal of Intelligent Manufacturing 23(4): 1001-1014.

Aragon, V., Susana, E.C., ve Coello, C.A. (2010) *A modified version of a t cell algorithm for constrained optimization problems.* International Journal for Numerical Methods in Engineering, 84(3):351–378, doi:10.1002/nme.2904.

Arora, J. S. (1967). *Introduction to optimum design*. 1989. McGraw-Mill Book Company.

Arora, S. & Singh, S. (2019). *Butterfly optimization algorithm: a novel approach for global optimization.* Soft Computing 23(3): 715-734.

Askarzadeh, A. (2016). *A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm.* Computers & Structures 169: 1-12.

Asma Chakri, H. R. & Yang, X.-S. (2018). *Nature-Inspired Algorithms and Applied Optimization.* Cham: Springer International Publishing.

Azizyan, G., Miarnaeimi, F., Rashki, M. ve Shabakhty, N. (2019) *Flying Squirrel Optimizer (FSO): A Novel SI-Based Optimization Algorithm for Engineering Problems*, Journal of Optimization Volume 11, Issue 2, 2019, 177-205 Research Paper.

Babu, N. R., Narrisetty, V. & Saikia, L. C. (2019). *Maiden Application of Coyote Optimizer Algorithm with TIDN Controller in AGC of a Multi-Area Multi-Source System.* In 2019 IEEE 16th India Council International Conference (INDICON), 1-4: IEEE.

Babu, N. R. ve Saikia, L. C. (2020) *Load Frequency Control of a Multi-area System Incorporating Dish-Stirling Solar Thermal System and Coyote Optimized PI minus DF Controller.* 2020 IEEE International Conference on Power Electronics. Smart Grid and Renewable Energy (PESGRE2020). pp. 1-6.doi: 10.1109/PESGRE45664.2020.9070654.

Barbosa, H.J.C., Lemonge, A.C.C., (2003) *A new adaptive penalty scheme for genetic algorithms,* Information Sciences, volume 156 issue 3-4 pp. 215-251 ISSN 0020-0255, https://doi.org/10.1016/S0020-0255(03)00177-4.

Bekoff, M. (1977) *Canis latrans.* Mammalian Species. vol. 1. no. 79. pp. 1–9.

Ben Guedria, N. (2016). *Improved accelerated PSO algorithm for mechanical engineering optimization problems.* Applied Soft Computing 40: 455-467.

Bernardino, H. S., Barbosa, H. J. C., Lemonge, A. C. C., ve Fonseca, L. G. (2008) *A new hybrid ais-ga for constrained optimization problems in mechanical engineering.* In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pages 1455–1462. ISSN 1089-778X. doi:10.1109/CEC.2008.4630985.

Beyer, H. G., & Schwefel, H. P. (2002). *Evolution strategies–a comprehensive introduction.* Natural computing, 1(1), 3-52. Doi: 10.1023/A:1015059928466

Boursianis, A. D., Papadopoulou, M. S., Pierezan, J., Mariani, V. C., Coelho, L. S., Sarigiannidis, P., Koulouridis, S., Goudos S. K. (2021). *Multiband Patch Antenna Design Using Nature-Inspired Optimization Method.* in IEEE Open Journal of Antennas and Propagation. vol. 2. pp. 151-162. 2021. doi: 10.1109/OJAP. 3048495.

Brajevic, I., Ignjatovic J. (2015). *An Enhanced Fırefly Algorıthm For Mıxed Varıable Structural Optımızatıon Problems*, Ser. Math. Inform., volume 30 no 4 pp. 401-417.

Brajević I. Ignjatović J. (2019). *An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems.* J Intell Manuf. https://doi.org/10.1007/s10845-018-1419-6.

Coello, CAC. (2000). *Use of a self-adaptive penalty approach for engineering optimization problems.* Comput Ind 2000;41: 113–27.

Caponetto, R., Fortuna L., Fazzino, S. & Xibilia. M. G. (2003). *Chaotic sequences to improve the performance of evolutionary algorithms*. IEEE transactions on evolutionary computation. 7(3). 289-304.

Chakri, A., Khelif, R., Benouaret, M., & Yang, X. (2017). *New directional bat algorithm for continuous optimization problems,* vol. 69, pp. 159-175.

Chang, G. W. ve Chinh, N. C. (2020). *Coyote Optimization Algorithm-Based Approach for Strategic Planning of Photovoltaic Distributed Generation.* In IEEE Access. vol. 8. pp. 36180-36190. 2020. doi: 10.1109/ACCESS.2020.2975107.

Chen, S., Zhang, L., & Cheng, X. (2015). *Genetic Algorithm Optimal Design on Diaphragm Spring by MATLAB.* In 2015 2nd International Conference on Electrical, Computer Engineering and Electronics (pp. 186-190). Atlantis Press. Doi: 10.2991/icecee-15.2015.42.

Chin, V. J. & Salam, Z. (2019). *Coyote optimization algorithm for the parameter extraction of photovoltaic cells.* Solar Energy 194: 656-670.

Coelho, L.D.S. (2010). *Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems.* Expert Syst Appl 37: 1676-1683.

Coello, C.A.C. (2000). *Constraint-Handling Using an Evolutionary Multiobjective Optimization Technique.* Civil Engineering and Environmental Systems 17: 319-346.

Coello, C.A.C. (2000). *Use of a self-adaptive penalty approach for engineering optimization problems.* Computers in Industry 2000(41): 113-127.

Coello, C.A.C. (2000). *An updated survey of GA-based multi-objective optimiza-tion techniques.* ACM Comput-ing Surveys Volume 32 Issue 2 June 2000 pp 109–143 https://doi.org/ 10.1145 /358923. 358929.

Coello, CA. C., Mezura, M. E. (2002). *Constraint-handling in genetic algorithms through the use of dominance-based tournament selection.* Adv Eng Inf 2002;16:193–203.

Coello, C., Becerra, RL., (2004). *Efficient evolutionary optimization through the use of a cultural algorithm.* Eng Optim 2004; 36:219–36.

Conner, M. M., Ebinger, M. R., ve Knowlton, F. F. (2008). *Evaluating coyote management strategies using a spatially explicit. individual-based. socially structured population model.* Ecological Modelling. vol. 219. no. 1-2. pp. 234–247.

Covic, N. & Lacevic, B. (2020). *Wingsuit Flying Search: A Novel Global Optimization Algorithm.* IEEE Access 8: 53883-53900.

Çakmak, T. (2018). *Taşitlarda kullanilan debriyaj sistemlerinin termo-mekanik özelliklerinin iyileştirilmesi,* Doktora Tezi.

Datta, D. F. J. R. (2011). *A real-integer-discrete-coded particle swarm optimization for design problems.* Applied Soft Computing 11(4): 3625-3633.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life.* H. Milford: Oxford University Press.

Deb, K. (2000). *An efficient constraint handling method for genetic algorithms*. Comput. Methods Appl. Mech. Eng. 186 311–338.

Deb, K., Pratap, A., Agarwal, S. ve Meyarivan, T. (2002). *A fast and elitist multiobjective genetic algorithm: NSGA-II,* in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, doi: 10.1109/4235.996017.

Deb, K. & Srinivasan, A. (2006). *Innovization: innovative design principles through optimization.* In Proceedings of the 8th annual conference on Genetic and evolutionary computation- GECCO '06, pp 1629-1629 New York, New York, USA: ACM Press.

Dhadwal, M.K., Jung, S.N. & Kim, C.J. (2014). *Advanced particle swarm assisted genetic algorithm for constrained optimization problems.* Comput Optim Appl 58, 781–806, https://doi.org/10.1007/s10589-014-9637-0.

Diab, A. A. Z., Sultan, H. M., Do T. D., Kamel, O. M. ve Mossa, M. A. (2020). *Coyote Optimization Algorithm for Parameters Estimation of Various Models of Solar Cells and PV Modules.* In IEEE Access. vol. 8. pp. 111102-111140. doi: 10.1109/ACCESS.2020.3000770.

Doan, A. T., Duong, M. Q. D. & Mussetta, M. (2021). *Optimally Placing Photovoltaic Systems in Distribution Networks Considering the Influence of Harmonics on Power Losses.* International Journal on Electrical Engineering and Informatics 13(2): 252-270.

Dorigo. M., Birattari. M., ve Stutzle. T. (2006). *Ant colony optimization*. IEEE Computational Intelligence Magazine. 1(4):28–39.

Du, T.-S., Ke, X.-T., Liao, J.-G., ve Shen, Y.-J. (2018). *Dslc-foa: Improved fruit fly optimization algorithm for application to structural engineering design optimization problems.* Applied Mathematical Modelling, 55:314 – 339, ISSN 0307-904X. doi:https://doi.org/10.1016/j.apm.2017.08.013.

Duman, S., Kahraman, H. T., Guvenc, U. & Aras, S. (2021). *Development of a Lévy flight and FDB-based coyote optimization algorithm for global optimization and real-world ACOPF problems.* Soft Computing 25(8): 6577-6617.

El-Ela, A. A. A., El-Sehiemy, R. A., Shaheen, A. M. & Ellien, A. R. (2021). *Optimal Allocation of Distributed Generation Units Correlated With Fault Current Limiter Sites in Distribution Systems.* IEEE Systems Journal 15(2): 2148-2155.

Erjavec, J., & Thompson, R. (2014). *"Clutches." Chapter 36 in Automotive technology: A Systems Approach.* Cengage Learning.

Erol, O. K., & Eksin, I. (2006). *A new optimization method: big bang–big crunch.* Advances in Engineering Software, 37(2), 106-111. Doi: 10.1016/j.advengsoft.2005.04.005.

Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M. (2012). *Water Cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems.* Comput Struct 110–111:151-166.

Fathy, A., Al-Dhaifallah, M. & Rezk, H. (2019). *Recent Coyote Algorithm-Based Energy Management Strategy for Enhancing Fuel Economy of Hybrid FC/Battery/SC System.* IEEE Access 7: 179409-179419.

Feng, Z., Niu, W., Liu, S. (2021). *Cooperation search algorithm: A novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems.* Applied Soft Computing 98 https://doi.org/10.1016/j.asoc.2020. 106734

Formato, R. A. (2009). *Central force optimization: a new deterministic gradient-like optimization metaheuristic.* Opsearch, 46(1), 25-51. Doi: 10.1007/s12597-009-0003-4.

Sayed, G.I., Khoriba, G. & Haggag, M.H. (2018). *A novel chaotic salp swarm algorithm for global optimization and feature selection.* Appl Intell 48, 3462–3481 https://doi.org/10.1007/s10489-018-1158-6

Gandomi, A.H. ve Yang, X.-S. (2011). *Benchmark Problems in Structural Optimization.* pages 259–281. Springer Berlin Heidelberg. Berlin. Heidelberg. ISBN 978-3-642-20859-1.

Gandomi, A. H., Yang, X.-S., Alavi, A.H., (2011). *Mixed variable structural optimization using Firefly Algorithm*, Computers & Structures, volume 89 issue 23-24 pp. 2325-2336 ISSN 0045-7949, https://doi.org/10.1016/j.compstruc.2011.08.002

Gandomi, AH. Yang, XS. Alavi, AH (2013). *Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems.* Eng Comput. 29:17–35.

Gandomi, A. H. (2014). *Interior search algorithm (isa): A novel approach for global optimization.* ISA Transactions, 53(4):1168 – 1183, ISSN 0019-0578. doi: https://doi.org/10.1016/j.isatra.2014.03.018

Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J. ve Zhou M. (2021). *Chaotic Local Search-Based Differential Evolution Algorithms for Optimization.* in *IEEE Transactions on Systems. Man. and Cybernetics: Systems*. vol. 51. no. 6. pp. 3954-3967. doi: 10.1109/TSMC.2019.2956121

Garg, H. (2014). *Solving structural engineering design optimization problems using an artificial bee colony algorithm.* Journal of Industrial & Management Optimization 10(3): 777-794.

Geetha, T. ve Sathya, M., (2012). *Modified Particle Swarm Optimization (MPSO) algorithm for Web Service Selection (WSS) problem,* 2012 International Conference on Data Science & Engineering (ICDSE), pp. 113-116, doi: 10.1109/ICDSE.2012.6281954

Ghasemian, H., Ghasemian, F. & Vahdat-Nejad, H. (2020). *Human urbanization algorithm: A novel metaheuristic approach.* Mathematics and Computers in Simulation 178: 1-15.

Genç, M.O. ve Kaya, N. (2020). *Vibration Damping Optimization using Simulated Annealing Algorithm for Vehicle Powertrain System*. Engineering. Technology & Applied Science Research Vol. 10. No. 1. 2020. 5164-5167.

Gese, E. M., Ruff, R. L., ve Crabtree, R. L. (1996). *Foraging ecology of coyotes (Canis latrans): the influence of extrinsic factors and a dominance hierarchy.* Canadian Journal of Zoology. vol. 74. no. 5. pp. 769–783.

Golinski, J. (1970). *Optimal synthesis problems solved by means of nonlinear programming and random methods.* Journal of Mechanisms. 5(3):287 – 309. ISSN 0022-2569. doi:https://doi.org/10.1016/0022-2569(70)90064-9

Guedria, NB. (2016). *Improved accelerated PSO algorithm for mechanical engineering optimization problems.* Appl Soft Comput 40:455–467.

Guesmi, T., Alshammari, B. M., Almalaq, Y., Alateeq, A. & Alqunun, K. (2021). *New Coordinated Tuning of SVC and PSSs in Multimachine Power System Using Coyote Optimization Algorithm.* Sustainability 13(6): 3131-3131.

Gupta, S. & Deep, K. (2020). *Enhanced leadership-inspired grey wolf optimizer for global optimization problems.* Engineering with Computers 36(4): 1777-1800.

Gupta, S., Tiwari, R., Shivashankar, BN. (2007). *Multi-objective design optimization of rolling bearings using genetic algorithm.* Mech Mach Theory 2007;42:1418–43.

Güvenç, U. ve Kaymaz, E. (2019). *Economic Dispatch Integrated Wind Power Using Coyote Optimization Algorithm.* 2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG). pp. 179-183. doi: 10.1109/SGCF.2019.8782354

Hamdani, H., Radi, B. & El Hami, A. (2019). *Optimization of solder joints in embedded mechatronic systems via Kriging-assisted CMA-ES algorithm.* International Journal for Simulation and Multidisciplinary Design Optimization 10: A3-A3.

Hasan, K. (2019). *Parkinson's Disease Recognition using Gauss Map based Chaotic Particle Swarm-Neural Network*. International Conference on Engineering and Telecommunication (EnT), pp. 1-4, doi: 10.1109/EnT47717.2019.9030560

Han, J., Yang, C., Zhou, X., ve Gui, W. (2018). *A two-stage state transition algorithm for constrained engineering optimization problems.* International Journal of Control,

Automation and Systems, 16(2):522–534, ISSN 2005-4092. doi:10.1007/s12555-016-0338-6

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. & Al-Atabany, W. (2022). *Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems.* Mathematics and Computers in Simulation 192: 84-110

He, H., Obubo, K., Fujii, T., & Doman, Y. (2003). *Optimize the load-deflection curve of diaphragm springs for automobile clutches using residual stress.* WIT Transactions on The Built Environment, 67. Doi: 10.2495/OP030131

He, S., Prempain, E., Wu, Q.H. (2004). *An improved particle swarm optimizer for mechanical design optimization problems.* Eng. Optim. 36 585–605.

Heidari, AA., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H. (2019). *Harris hawks optimization: algorithm and applications.* Future Gener Comput Syst 97:849–872.

Himmelblau, D.M. (1972). *Applied Nonlinear Programming.* McGraw-Hill Companies.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press. Ann Arbor.

Huan, T. T., Kulkarni A. J., Kanesan, J., Huang, C. J. & Abraham, A. (2017). *Ideology algorithm: a socio-inspired optimization methodology.* Neural Computing and Applications 28(S1): 845-876.

Huang, C. & Zhuang J. (2021). *Error-based Active Disturbance Rejection Control for Pitch Control of Wind Turbine by Improved Coyote Optimization Algorithm.* IEEE Transactions on Energy Conversion: 1-1.

Husseinzadeh Kashan, A. (2011). *An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA).* Computer-Aided Design 43(12): 1769-1792.

Hussien, A. G. (2022). *An enhanced opposition based Salp Swarm Algorithm for global optimization and engineering problems.* Journal of Ambient Intelligence and Humanized Computing 13(1): 129-150.

Ibrahim, R. A., Oliva, D., Ewees, A. A. & Lu, S. (2017). *Feature Selection Based on Improved Runner-Root Algorithm Using Chaotic Singer Map and Opposition-Based Learning.* 156-166 Cham: Springer International Publishing.

Janamala, V. & Sreenivasulu Reddy, D. (2021). *Coyote optimization algorithm for optimal allocation of interline –Photovoltaic battery storage system in islanded electrical distribution network considering EV load penetration.* Journal of Energy Storage 41: 102981-102981.

Jin, Z., Sun, X., Lei, G., Guo, Y. & Zhu, J. (2022). *Sliding Mode Direct Torque Control of SPMSMs Based on a Hybrid Wolf Optimization Algorithm.* IEEE Transactions on Industrial Electronics 69(5): 4534-4544.

Jingqiao, Z. & Sanderson, A. C. (2009). *JADE: Adaptive Differential Evolution With Optional External Archive.* IEEE Transactions on Evolutionary Computation 13(5): 945-958.

Kahraman, H.T., Aras, S. ve Gedikli, E. (2020). *Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms.* Knowledge-Based Systems. Volume 190. 105169. ISSN 0950-7051. https://doi.org/10.1016/j.knosys.2019.105169

Kamel, S., Amin, A., Selim, A. ve Ahmed, M. H. (2019). *Application of coyote optimizer for Optimal DG Placement in Radial Distribution Systems.* 2019 International Conference on Computer. Control. Electrical. and Electronics Engineering (ICCCEEE). 2019. pp. 1-6. doi: 10.1109/ICCCEEE46830.2019.9070817

Kanagaraj, G., Ponnambalam, S. G., Jawahar, N. & Nilakantan, J. M. (2014). *An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization.* Engineering Optimization 46(10): 1331-1351.

Kanagaraj, G., Ponnambalam, S. G. & Lim, W. C. E. (2014). *Application of a hybridized cuckoo search-genetic algorithm to path optimization for PCB holes drilling process.* In 2014 IEEE International Conference on Automation Science and Engineering (CASE), pp 373-378: IEEE.

Kannan, B. K. & Kramer, S. N. (1994). *An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design.* Journal of Mechanical Design 116(2): 405-411.

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

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization. Teknik rapor. Teknik rapor-tr06.* Erciyes üniversitesi. Mühendislik Fakültesi. Bilgisayar Mühendisliği Bölümü.

Karaduman, A. ve Yıldız, A.R. (2020). *Experimental and numerical fatique-based design optimisation of clutch diaphragm spring in the automotive industry.* Internatinal Journal of Vehicle Design. vol.80. pp.330-345.

Karaduman, A., Yıldız. A.R., ve Lekesiz, H. (2019). *Release Bearing Characteristic of Diaphragm Spring under Dynamical Condition.* Society of Automotive Engineering International (SAE).

Kartal, S., Kaya, N., Çakmak, T., Karaduman, A., ve Karpat, F. (2015). *Shape Optimization Of Clutch Cushion Disc Using Differential Evolution Method*. Journal Of Applıed Mechanıcs-Transactıons Of The Asme.

Karaduman, A. ve Yıldız, A.R. (2019). *Load and Stress Optimization of Elastic Washer Using NLPQL and Evolutionary Algorithm for Automotive Clutch and Investigation of Parameter Impacts.* International Conference on Artificial Intelligence and Applied Mathematics in Engineering.

Karaduman, A. ve Yıldız, A.R. (2018). *Optimal Design Of Finger Slots Of Clutch Diaphgram Spring.* 8. Otomotiv Teknolojileri Kongresi Otekon 2018.

Kashan, A.H. (2011). *An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA).* Comput Des 43:1769 – 1792.

Kaya, N. (2006). O*ptimal design of an automotive diaphragm spring with high fatigue resistance.* International journal of vehicle design, 40(1-3), 126-143. Doi: 10.1504/IJVD.2006.008457

Kaya, N. (2014). *Shape Optimization of Rubber Bushing Using Differential Evolution Algorithm.* Hindawi Publishing Corporation Scientific World Journal. Volume 2014. Article ID 379196.

Kaymaz, E., Duman, S. & Guvenc, U. (2021). *Optimal power flow solution with stochastic wind power using the Lévy coyote optimization algorithm.* Neural Computing and Applications 33(12): 6775-6804.

Kazarlis, S. A., Bakirtzis, A. G. & Petridis, V. (1996). *A genetic algorithm solution to the unit commitment problem.* IEEE Transactions on Power Systems 11(1): 83-92.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95 - International Conference on Neural Networks, pp 1942-1948: IEEE.

Khishe, M, Mosavi, M.R. (2020). *Chimp optimization algorithm.* Expert Syst Appl 149:113338

Kien, L. C., Hien, C. T. & Nguyen, T. T. (2021). *Optimal Reactive Power Generation for Transmission Power Systems Considering Discrete Values of Capacitors and Tap Changers.* Applied Sciences 11(12): 5378-5378.

Kim, H-KKH-K. Chong, J-KCJ-K. Park, K-YPK-Y. Lowther, DA. (2007). *Differential evolution strategy for constrained global optimization and application to practical engineering problems.* IEEE Trans Magn 43:1565–1568.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science 220(4598): 671-680.

Krohling, R. A. & dos Santos Coelho, L. (2006). *Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems.* IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 36(6): 1407-1416.

Koza, J.R. (1992). *Genetic programming II, automatic discovery of reusable subprograms.* MIT Press. Cambridge

Kumar, A., Misra, R. K. & Singh, D. (2017). *Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase.* In 2017 IEEE Congress on Evolutionary Computation (CEC), pp 1835-1842: IEEE.

Kumar, V., Kumar, D., (2017). *An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems,* Adv. Eng. Softw. 112 231–254.

Leandro, S.C. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Systems with Applications, 37(2):1676 – 1683, 2010. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2009.06.044

Lee, K.S., Geem, Z.W. (2005). *A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice.* Comput. Methods Appl. Mech. Eng. 194, 3902–3933.

Lemonge, B., (2004). *An adaptive penalty scheme for genetic algorithms in structural optimization,* Int. J. Numer. Meth. Engng 59 703–736 (10.1002/nme.899).

Li, L., Sun, L., Xue Y., Li S., Huang X. ve Mansour R. F. (2021). *Fuzzy Multilevel Image Thresholding Based on Improved Coyote Optimization Algorithm.* In *IEEE Access*. vol. 9. pp. 33595-33607. 2021. doi: 10.1109/ACCESS.2021.3060749

Liu, H., Cai, Z., ve Wang, Y. (2010). *Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization.* Applied Soft Computing, 10(2):629 – 640. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2009.08.031

Mezura-Montes, E., Coello, CA. C., ve Landa-Becerra, R. (2003). *Engineering optimization using simple evolutionary algorithm.* In Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, pages 149– 156. IEEE.

Mezura Montes, E., Velázquez Reyes, J. & Coello Coello, C. A. (2006). *Modified differential evolution for constrained optimization.* IEEE International Conference on Evolutionary Computation: pp 25-32.

Mezura-montes, E., Coello Coello, C. A. & Velázquez-reyes, J. (2006). *Increasing Successful Offspring and Diversity in Differential Evolution for Engineering Design.* In Proc. Adaptive Computing in Design and Manufcature (ACDM 2006), pp 131-139.

Mezura-Montes, E. & Hernández-Ocana, B. (2008). *Bacterial foraging for engineering design problems: preliminary results.* In In Memorias del 4o Congreso Nacional de Computacion Evolutiva (COMCEV 2008).

Mirjalili, S. (2016). *Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems.* Neural Comput & Applic 27, 1053–1073 (2016). https://doi.org/10.1007/s00521-015-1920-1.

Mirjalili, S., Mirjalili, S. M. ve Lewis, A. (2014). *Grey wolf optimizer*. Advances in Engineering Software. 69:46 – 61.

Mirjalili, S., (2015). *Moth-Flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm, Knowledge-Based Systems,* vol. 89, pp. 228-249, 2015.

Mirjalili, S. ve Lewis, A. (2016). *The whale optimization algorithm*. Advances in Engineering Software. 95:51 – 67.

Mirjalili, S. (2016). *SCA: A Sine Cosine Algorithm for solving optimization problems*. Knowledge-Based Systems, 96:120 – 133. ISSN 0950-7051
https://doi.org/10.1016/j.knosys.2015.12.022

Mirjalili, S., Mirjalili, S.M., Hatamlou, A., (2016). *Multi-Verse Optimizer: a nature inspired algorithm for global optimization,* Neural Comput. Appl. 27 495–513

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.

Mitsuo Gen, R. C. (1997). *Genetic Algorithms and Engineering Design.* John Wiley & Sons, Inc., New York.

Mohamed, A., Saber, W., Elnahry, I. & Hassanien, A. E. (2020). *Coyote Optimization Based on a Fuzzy Logic Algorithm for Energy-Efficiency in Wireless Sensor Networks.* IEEE Access 8: 185816-185829.

Mohamed, A. W. (2018). A novel differential evolution algorithm for solving constrained engineering optimization problems. Journal of Intelligent Manufacturing, 29(3):659–692, ISSN 1572-8145. doi:10.1007/s10845-017-1294-6

Mohamed, A. A. R., Morrow, D. J., Best R. J., Bailie I., Cupples A. ve Pollock J. (2020). *Battery Energy Storage Systems Allocation Considering Distribution Network Congestion.* 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe). pp. 1015-1019. doi: 10.1109/ISGT-Europe47291.2020.9248982

Moschos, I. & Parisses, C. (2022). *A novel optimal PIλDND2N2 controller using coyote optimization algorithm for an AVR system.* Engineering Science and Technology, an International Journal 26: 100991-100991.

Mostafa, H. H. & Ibrahim, A. M. (2019). *Performance Investigation for Tracking GMPP of Photovoltaic System Under Partial Shading Condition Using Coyote Algorithm.* In 2019 21st International Middle East Power Systems Conference (MEPCON), pp 34-40: IEEE.

Nam, W. H., Lee, C. Y., Chai, Y. S., Kwon, J. D., Bae, Y. T., & Woo, S. W. (2000). *A study on fatigue and durability characteristics of clutch diaphragm spring according to tempering condition* (No. 2000-05-0133). SAE Technical Paper.

Nguyen, T. T., Pham, T. D., Kien, L. C. & Van Dai, L. (2020). *Improved Coyote Optimization Algorithm for Optimally Installing Solar Photovoltaic Distribution Generation Units in Radial Distribution Power Systems.* Complexity 2020: 1-34.

Ong, K. M., Ong, P. & Sia, C. K. (2021). *A carnivorous plant algorithm for solving global optimization problems.* Applied Soft Computing 98: 106833-106833.

Patel, J. L., Rana, P. B. & Lalwani, D. I. (2020). *Optimization of five stage cantilever beam design and three stage heat exchanger design using amended differential evolution algorithm.* Materials Today: Proceedings 26: 1977-1981.

Pham, T. D., Nguyen, T. T. & Dinh, B. H. (2021). *Find optimal capacity and location of distributed generation units in radial distribution networks by using enhanced coyote optimization algorithm.* Neural Computing and Applications 33(9): 4343-4371.

Pierezan, J. & Dos Santos Coelho, L. (2018). *Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems.* In 2018 IEEE Congress on Evolutionary Computation (CEC), pp 1-8: IEEE.

Pierezan, J., dos Santos Coelho, L., Cocco Mariani, V., Hochsteiner de Vasconcelos Segundo, E. & Prayogo, D. (2021). *Chaotic coyote algorithm applied to truss optimization problems.* Computers & Structures 242: 106353-106353.

Pierezan, J., dos Santos Coelho, L., Mariani, V. C. & Lebensztajn, L. (2019). *Multiobjective Coyote Algorithm Applied to Electromagnetic Optimization.* In 2019 22nd International Conference on the Computation of Electromagnetic Fields (COMPUMAG), pp 1-4: IEEE.

Pierezan, J., Maidl, G., Massashi Yamao, E., dos Santos Coelho, L. & Cocco Mariani, V. (2019). *Cultural coyote optimization algorithm applied to a heavy duty gas turbine operation.* Energy Conversion and Management 199: 111932-111932.

Pitt, W. C., Box, P. W., ve Knowlton F. F. (2003). *An individual-based model of canid populations: Modelling territoriality and social structure.* Ecological Modelling. vol. 166. no. 1-2. pp. 109–121.

Poessel S. A., Gese E. M., ve Young J. K. (2014). *Influence of habitat structure and food on patch choice of captive coyotes.* Applied Animal Behaviour Science. vol. 157. pp. 127–136.

Qais, M. H., Hasanien, H. M., Alghuwainem, S. & Nouh, A. S. (2019). *Coyote optimization algorithm for parameters extraction of three-diode photovoltaic models of photovoltaic modules.* Energy 187: 116001-116001.

Rao, S. S. (1996). *Engineering Optimization: Theory and Practice*. 3rd Edition. Wiley-Interscience.

Rao, R.V., Savsani, V.J., ve Vakharia, D.P. (2011). *Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems.* Computer-Aided Design, 43(3):303 – 315, 2011. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2010.12.015

Rehab A., I., Oliva, D., Ewees, A.A., & Lu, S. (2017). *Feature Selection Based on Improved Runner-Root Algorithm Using Chaotic Singer Map and Opposition-Based Learning. ICONIP*.

Rashedi, E., Nezamabadi-pour, H. & Saryazdi, S. (2009). *GSA: A Gravitational Search Algorithm.* Information Sciences 179(13): 2232-2248.

Renato, A. K. ve Leandro, S. C. (2006). *Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems*, in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 36, no. 6, pp. 1407-1416, doi: 10.1109/TSMCB.2006.873185

Rezk, H., Fathy, A. & Aly, M. (2021). *A robust photovoltaic array reconfiguration strategy based on coyote optimization algorithm for enhancing the extracted power under partial shadow condition.* Energy Reports 7: 109-124.

Ribeiro, M. H. D. M., da Silva, R. G., Canton, C., Fraccanabbia, N., Mariani, V. C. & Coelho, L.d.S. (2020). *Electricity energy price forecasting based on hybrid multi-stage heterogeneous ensemble: Brazilian commercial and residential cases.* In 2020 International Joint Conference on Neural Networks (IJCNN), pp 1-8: IEEE.

Runarsson, T. & Yao, X. (2003). *Constrained Evolutionary Optimization. In Evolutionary Optimization*, Vol. 48, pp 87-113 Boston: Kluwer Academic Publishers.

Runarsson, TP, Yao, X. (2000). *Stochastic ranking for constrained evolutionary optimization.* IEEE Trans Evol Comput 4(3):284–294.

Runarsson, T., Yao, X. (2003). *Constrained Evolutionary Optimization. in Evolutionary Optimization.* Interna-tional Series in Operations Research & Management Science. vol 48. Springer.

Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M. (2013). *Mine blast algorithm: a new population-based algorithm for solving constrained engineering optimization problems*. Appl Soft Comput J 13:2592–2612. https://doi.org/10.1016/j.asoc.2012.11.026

Salcedo-Sanz, S.(2016). *Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures.* Physics Reports.vol. 655. pp. 1–70.

Sandgren, E. (1990). *Nonlinear integer and discrete programming in mechanical design optimization.* Journal of Mechanical Design. 112(2):223. doi:10.1115/1. 2912596.

Savsani, P., Savsani, V., (2016). *Passing vehicle search (pvs): A novel metaheuristic algorithm,* Applied Mathematical Modelling 40(5-6): 3951-3978.

Sayed, G. I., Khoriba, G. & Haggag, M. H. (2018). *A novel chaotic salp swarm algorithm for global optimization and feature selection.* Applied Intelligence 48(10): 3462-3481.

Sayed, G. I., Khoriba, G. & Haggag, M. H. (2020). *The novel multi-swarm coyote optimization algorithm for automatic skin lesion segmentation.* Evolutionary Intelligence.

Shangguan, W. B., Liu, X. L., Rakheja, S., & Hou, Q. (2019). *Effective utilizing axial nonlinear characteristics of diaphragm spring and waveform plate to enhance breakaway performances of a clutch.* Mechanical Systems and Signal Processing, 125, 123-141. Doi: 10.1016/j.ymssp.2018.05.060

Sharma, TK., Abraham, A (2020). *Artificial bee colony with enhanced food locations for solving mechanical engineering design problems.* J Ambient Intell Human Comput 11. 267–290. https://doi.org/10.1007//s12652-019-01265-7

Shi, S., Zhou, S. ve Zhang, L. (2020). *Application of Improved Coyote optimization Algorithm in Optimal Configuration of Photovoltaic Intelligent Edge Terminal*. 2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2). pp. 3998-4003. Doi: 10.1109/EI250167.2020.9346923

Shiqin, Y., Jianjun, J., & Guangxing, Y. (2009). *A dolphin partner optimization.* In 2009 WRI global congress on intelligent systems (Vol. 1, pp. 124-128). IEEE. Doi: 10.1109/GCIS.2009.464

Shuxin, C., Lingyu, Z., Xingwang, C., (2015). *Genetic Algorithm Optimal Design on Diaphragm Spring by Matlab*. Second International Conference on Electrical. Computer Engineering and Electronics China. Doi: 10.2991/icecee-15.2015.42.

Simon, D. (2008). *Biogeography-based optimization.* IEEE transactions on evolutionary computation, 12(6), 702-713. Doi: 10.1109/TEVC.2008.919004.

Storn, R., & Price, K. (1997). *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces.* Journal of global optimization, 11(4), 341-359. Doi: 10.1023/A:1008202821328

Stoyanov, B. (2014). *Pseudo-random Bit Generation Algorithm Based on Chebyshev Polynomial and Tinkerbell Map*. Applied Mathematical Sciences. Vol. 8. 2014. no. 125. 6205 – 6210.

Sultan, H. M., Menesy, A. S., Kamel, S. & Jurado, F. (2021). *Developing the coyote optimization algorithm for extracting parameters of proton-exchange membrane fuel cell models.* Electrical Engineering 103(1): 563-577.

Sun, L., Han, X.-F., Xu, Y.-P. & Razmjooy, N. (2021). *Exergy analysis of a fuel cell power system and optimizing it with Fractional-order Coyote Optimization Algorithm.* Energy Reports 7: 7424-7433.

Sun, W., Li. Y., Huang. J. (2011). *Nonlinear Characteristics Study and Parameter Optimization of DMF-RS.* SAE International Kournal of Passenger Cars-Mechanical Systems. Vol:4. Issue:2. pp. 1050-1057.

Şencan, İ. ve Doğan, G. (2017). *Bilimsel yayınlarda kaynak gösterme. tablo ve şekil oluşturma rehberi APA 6 kuralları.* Ankara: Türk Kütüphaneciler Derneği.

Thom de Souza, R. C., de Macedo, C. A., dos Santos Coelho, L., Pierezan, J. & Mariani, V. C. (2020). *Binary coyote optimization algorithm for feature selection.* Pattern Recognition 107: 107470-107470.

Tizhoosh, H. R. (2005). *Opposition-Based Learning: A New Scheme for Machine Intelligence.* International Conference on Computational Intelligence for Modelling. Control and Automation and International Conference on Intelligent Agents. Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). pp. 695-701. doi: 10.1109/CIMCA.2005.1631345

Tong, H., Zhu, Y., Pierezan, J., Xu, Y. & Coelho, L. d. S. (2021). *Chaotic Coyote Optimization Algorithm.* Journal of Ambient Intelligence and Humanized Computing.

Vineeth, P., M, V. B. & Suresh, S. (2021). *Performance evaluation and analysis of population-based metaheuristics for denoising of biomedical images.* Research on Biomedical Engineering 37(2): 111-133.

Wagdy, Ali. (2018). *A novel differential evolution algorithm for solving constrained engineering optimization problems.* Journal of Intelligent Manufacturing. 29. 10.1007/s10845-017-1294-6.

Wang, G.-G. (2018). *Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems.* Memetic Computing 10(2): 151-164.

Wang, G.-G., Deb, S. & Coelho, L. d. S. (2015). *Elephant Herding Optimization.* In 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), pp 1-5: IEEE.

Wang, G.-G., Deb, S. & Cui, Z. (2019). *Monarch butterfly optimization.* Neural Computing and Applications 31(7): 1995-2014.

Wang, H., Hu, Z., Sun, Y., Su, Q. & Xia, X. (2019). *A novel modified BSA inspired by species evolution rule and simulated annealing principle for constrained engineering optimization problems.* Neural Computing and Applications 31(8): 4157-4184.

Wang, L. & Li, L.-p. (2010). *An effective differential evolution with level comparison for constrained engineering design.* Structural and Multidisciplinary Optimization 41(6): 947-963.

Weise, T. (2008). *Global Optimization Algorithms – Theory and Application*. Thomas Weise. 2008-1-4 edition.

Wenyin Gong, Z. C., Dingwen Liang, (2014). *Engineering optimization by means of an improved constrained differential evolution.* Computer Methods in Applied Mechanics and Engineering 268: 884-904.

Wolpert, D. H. ve Macready, W. G. (1997). *No free lunch theorems for Optimization.* IEEE Transactions on Evolutionary Computation. vol. 1. no. 1. pp. 67–82. April 1997.

Wook-Hee, N., Choon-Yeol, L., Young, C., Jae-Do, K., Yong-Tak, B., Seung-Wan, W., (2000). *A Study on Fatique and Durability Characteristics of Clutch Diaphragm Spring According to Tempering Condition*. Seoul 2000 FISITA World Automotive Congress, Seoul, Korea.

Wu, D., Li, T. & Wan, Q. (2021). *A hybrid deep kernel incremental extreme learning machine based on improved coyote and beetle swarm optimization methods.* Complex & Intelligent Systems 7(6): 3015-3032.

Wu, S.-J. & Chow, P.-T. (1995). *Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems Via Meta-Genetic Parameter Optimization.* Engineering Optimization 24(2): 137-159.

Xiliang, C., Changqing, D., Fuwu, Y., Hongming, X., Biao, H., Yanfeng, S. (2019). *Parameter Optimization of Dual Clutch Transmission for an Axle-split Hybrid Electric Vehicle*. International Federation of Automatic Control "IFAC" PapersOnLine 52-5 (2019) 423-430.

Xin-She, Y. (2008). *Nature-Inspired Metaheuristic Algorithms.* Luniver Press.

Yang X. ve Deb S. (2009). *Cuckoo search via lévy flights*. In 2009 World Congress on Nature Biologically Inspired Computing (NaBIC). pages 210–214.

Yang, XS. (2010). *Engineering optimization: an introduction with metaheuristic applications.* Wiley, Hoboken, 2010.

Yang, XS. (2010). *A New Metaheuristic Bat-Inspired Algorithm*. pages 65–74. Springer Berlin Heidelberg. Berlin. Heidelberg.

Yang, X.S., Gandomi, A.H. (2012). *Bat algorithm: a novel approach for global engineering optimization.* Eng. Comput. 29(5), 464–483.

Yang, X.-S. (2013). O*ptimization and metaheuristic algorithms in engineering. In Xin-She Yang. Amir Hossein Gandomi. Siamak Talatahari. and Amir Hossein Alavi. editors.* Metaheuristics in Water. Geotechnical and Transport Engineering. pages 1 – 23. Elsevier. Oxford. 2013.

Yang, X., Hu, H., Yang, S., Wang, W., Shi, Z., Yu, H. & Huang, Y. (2020). *Train Operation Adjustment Method of Cross-line Train in Urban Rail Transit Based on Coyote Optimization Algorithm.* In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp 1-6: IEEE.

Yang, Y., Chen, H., Heidari, A. A. & Gandomi, A. H. (2021). *Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts.* Expert Systems with Applications 177: 114864-114864.

Yong-hai, W. (2009). *Multi-Objective Optimization Design of Vehicle Clutch Diaphragm Spring.* 2009 Second International Conference on Intelligent Computation Technology and Automation. 2009. pp. 194-197. doi: 10.1109/ICICTA.2009.513.

Yuan, Z., Wang, W., Wang, H. & Yildizbasi, A. (2020). *Developed Coyote Optimization Algorithm and its application to optimal parameters estimation of PEMFC model.* Energy Reports 6: 1106-1117.

Zahara, E. Kao, YT. (2009) *Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems.* Expert Syst Appl 36:3880–6.

Zhang, J. ve Sanderson, A. C. (2009). *Jade: Adaptive differential evolution with optional external archive*. IEEE Transactions on Evolutionary Computation. 13(5):945–958.

Zhang, J., Jia, D. ve Jiao, Y. (2009). *Chaotic Local Search Based Differential Evolution*. 2009 Fifth International Conference on Natural Computation. pp. 168-171. doi: 10.1109/ICNC.2009.256

Zhang, M., Luo, W., Wang, X. (2008). *Differential evolution with dynamic stochastic selection for constrained optimization.* Inform Sci 178:3043–74.

Zhang, W. ve Zhu, G. (2008). *Research and Application of PSO Algorithm for the Diaphragm Spring Optimization.* 2008 Fourth International Conference on Natural Computation pp. 549-553. doi: 10.1109/ICNC.2008.363

Zhao, W., Wang, L. & Mirjalili, S. (2022). *Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications.* Computer Methods in Applied Mechanics and Engineering 388: 114194-114194.

EKLER

EK 1 Kır Kurdu Optimizasyon Algoritmasına Dayalı Algoritmaların Sözde Kodları

GCOA1 “Kaotik tabanlı kır kurdu optimizasyon algoritması sözde kodu”
1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 then
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 then
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle
27: Amaç fonksiyonu değerlendir
28: En iyi kır kurdunu güncelle
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

GCOA2 “Kaotik tabanlı, kaotik bölgesel arama tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle
27: Amaç fonksiyonu değerlendir
28: En iyi kır kurdunu güncelle
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: Kaotik bölgesel arama işlemi uygula
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA3 “Kaotik bölgesel arama tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA5 “Uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
15.3 : **if** yaşam = 1 **then**
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 **then**
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
23: **end while**
24: En iyi sonucu veren kır kurdu seçilir.

GCOA6 “Kaotik tabanlı, laplace dağılımlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
16.1 : Kriter hesapla yaşam ve ölüm.
16.2 : Levy uçuş teoremine göre kaotik yeni bireyler oluştur (Algoritma Levy)
16.3 : **if** yaşam = 1 **then**
16.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
16.5 : **elseif** yaşam > 1 **then**
16.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
16.7 : **else**
16.8 : Yeni doğan birey ölür
16.9 : **end if**
16.10: Populasyonu kaotik olarak güncelle
17: **end for**
18: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
19: Kır kurtlarının yaşını güncelle.
20: Amaç fonksiyonu değerlendir.
21: En iyi kır kurdunu güncelle.
22: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
23: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
24: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
25: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
26: **end while**
27: En iyi sonucu veren kır kurdu seçilir.

GCOA 7 “Laplace dağılımlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi,

çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Yeni bireyler oluştur
15.3 : **if** yaşam = 1 **then**
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 **then**
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
23: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
24: **end while**
25: En iyi sonucu veren kır kurdu seçilir.

GCOA8 “Kaotik tabanlı, laplace dağılımlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi,

çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
15: Sürü içinde doğum ve ölüm
16.1 : Kriter hesapla yaşam ve ölüm.
16.2 : Levy uçuş teoremine göre kaotik yeni bireyler oluştur (Algoritma Levy)
16.3 : **if** yaşam = 1 **then**
16.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
16.5 : **elseif** yaşam > 1 **then**
16.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
16.7 : **else**
16.8 : Yeni doğan birey ölür
16.9 : **end if**
16.10: Populasyonu kaotik olarak güncelle
17: **end for**
18: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
19: Kır kurtlarının yaşını güncelle.
20: Amaç fonksiyonu değerlendir.
21: En iyi kır kurdunu güncelle.
22: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
23: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
24: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
25: **end while**
26: En iyi sonucu veren kır kurdu seçilir.

GCOA9 “Laplace dağılımlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).

2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
15.3 : **if** yaşam = 1 **then**
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 **then**
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
23: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
24: **end while**
25: En iyi sonucu veren kır kurdu seçilir.

GCOA10 “Laplace dağılımlı, karşıt tabanlı kaotik kır kurdu optimizasyon algoritması sözde kodu”
1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle

9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: for her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
31: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA11 “Kaotik tabanlı, Laplace dağılımlı, karşıt tabanlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuş tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).

2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy ucus teoremine göre kaotik yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
31: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
32: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
33: **end while**
34: En iyi sonucu veren kır kurdu seçilir.

GCOA12 “Laplace dağılımlı, karşıt tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)

31: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA13 “Laplace dağılımlı, karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kaotik kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: for her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
15.3 : **if** yaşam = 1 **then**
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 **then**
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
16.10: Populasyonu kaotik olarak güncelle
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında kaotik yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
23: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
24: **end while**
25: En iyi sonucu veren kır kurdu seçilir.

GCOA14 “Laplace dağılımlı, karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
15.3 : **if** yaşam = 1 **then**
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 **then**
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
23: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
24: **end while**
25: En iyi sonucu veren kır kurdu seçilir.

GCOA15 “Laplace dağılımlı, karşıt tabanlı, levy uçuşu tabanlı kaotik kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et

3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
11: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
12: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
13: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
14: **end for**
15: Sürü içinde doğum ve ölüm
15.1 : Kriter hesapla yaşam ve ölüm.
15.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
15.3 : **if** yaşam = 1 then
15.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
15.5 : **elseif** yaşam > 1 then
15.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
15.7 : **else**
15.8 : Yeni doğan birey ölür
15.9 : **end if**
15.10: Populasyonu kaotik olarak güncelle
16: **end for**
17: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
18: Kır kurtlarının yaşını güncelle.
19: Amaç fonksiyonu değerlendir.
20: En iyi kır kurdunu güncelle.
21: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
22: Laplace çaprazlama teoremini uygula (Algoritma Laplace çaprazlama)
23: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
24: **end while**
25: En iyi sonucu veren kır kurdu seçilir.

GCOA16 “Karşıt tabanlı, kaotik bölgesel arama tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle

9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA17 “Karşıt tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar

5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle (Denk. 3.13).
9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

GCOA18 “Kaotik tabanlı, karşıt tabanlı, kaotik bölgesel aramalı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”;

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi,

çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurdlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy uçuş teoremine göre kaotik yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA19 “Kaotik tabanlı, karşıt tabanlı, kaotik bölgesel aramalı, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu”

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc <Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle (Denk. 3.13).
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: İki kır kurdunu kaotik olarak sec, Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik olarak güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy uçuş teoremine göre kaotik yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
23.10: Populasyonu kaotik olarak güncelle
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik olarak yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.

29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: Kaotik bölgesel arama işlemi uygula (Algoritma kaotik bölgesel arama)
32: **end while**
33: En iyi sonucu veren kır kurdu seçilir.

GCOA20 "Karşıt tabanlı kır kurdu optimizasyon algoritması sözde kodu"

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Yeni bireyler oluştur
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
26: Kır kurtlarının yaşını güncelle.

27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

GCOA21 "Karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu"

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**

24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

GCOA22 "Karşıt tabanlı, uygunluk uzaklık dengeli, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu"

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini uygunluk uzaklık dengesine göre hesapla (Algoritma FDB)
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: Sosyal popülasyonu alfa ve sosyal eğilime göre güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm (Denk.3.20).
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**
23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür

23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

GCOA23 "Karşıt tabanlı, levy uçuşu tabanlı kır kurdu optimizasyon algoritması sözde kodu"

1: Kontrol parametrelerini belirle (Iterasyon durdurma kriteri, sürü sayısı, her sürüdeki kır kurdu sayısı, kaotik yerel arama stratejisi çeşidi, deney sayısı, kaotik harita tipi, çözünürlük).
2: Parametre değişkenlerinin alt ve üst sınırlarını kontrol et
3: Arama uzayı içerisinde Nc kır kurtlu Np sürü sosyal popülasyonu oluştur
4: Her birey için amaç fonksiyonlarını çıkar
5: En iyi kır kurtlarını belirle.
6: **while** (Np*Nc < Durdurma Kriteri)
7: **for** her p değerini sürü sayısı kadar tekrarla
8: Sürünün alfa kurdunu belirle
9: Sürünün sosyal eğilimini hesapla
10: **for** her x değerini sürü sayısı kadar devam ettir
11: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir
12: üst sınırı kır kurdu sosyal durumuna göre daralt
13: alt sınırı kır kurdu sosyal durumuna göre daralt
14: tüm sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
15: **end**
16: **end**
17: Sürünün aday çözümlerini ters yönde ara ve faydalı ise aday çözüm olarak al. (Algoritma OBL)
18: **for** her c değerini sürüdeki kır kurdu sayısı kadar tekrarla
19: Sosyal popülasyonu alfa ve sosyal eğilime göre kaotik güncelle
20: Yeni sosyal popülasyonun amaç fonksiyonunu değerlendir
21: Eğer yeni sonuçlar eski sonuçlardan iyiyse eski sonuçların parametrelerin üzerine yenilerini yaz
22: **end for**
23: Sürü içinde doğum ve ölüm
23.1 : Kriter hesapla yaşam ve ölüm.
23.2 : Levy uçuş teoremine göre yeni bireyler oluştur (Algoritma Levy)
23.3 : **if** yaşam = 1 **then**

23.4 : Yeni doğan birey yaşar ve yaşlı bireyler ölür
23.5 : **elseif** yaşam > 1 **then**
23.6 : Yeni doğan birey yaşar ve en yaşlı birey ölür
23.7 : **else**
23.8 : Yeni doğan birey ölür
23.9 : **end if**
24: **end for**
25: Her iterasyon iki kır kurdu sürüleri arasında kaotik yer değiştirir
26: Kır kurtlarının yaşını güncelle.
27: Amaç fonksiyonu değerlendir.
28: En iyi kır kurdunu güncelle.
29: En iyi kır kurdunu ve rastgele bir kır kurdunu seç
30: En iyi kır kurdunu daha iyi bir sonuç varsa güncelle
31: **end while**
32: En iyi sonucu veren kır kurdu seçilir.

EK 2 FDB Algoritmasının Sözde Kodu

1: minimum uygunluk değerini sağlayan veri serisini seç
2: **if** minimum uygunluk değerini maksimum uygunluk değeri ile aynı olup olmadığını veya toplam uyumluluk değerinin sonsuza gidip gitmediğini veya min. uyumluluk değerini sağlayan parametrelerin toplamı sonsuzdan küçük olup olmadığını kontrol et
3: Popülasyondan rastgele bir veri seç
4: **else**
5: **for** i = 1: popülasyon sayısı
6: **for** j = 1: boyut
7: En iyi değerin popülasyondaki her bir değere göre farklarını topla (3.27)
8: **end**
9: Her bir değerin en iyiye uzaklığını her popülasyon üyesi için hesapla (3.29)
10: **end**
11: Minimum uyum değerini hesapla,
12: Maximum uyum değeri ve Minimum uyum değeri arasındaki farkı hesapla
13: Minimum uzaklık değerini hesapla,
14: Maximum uzaklık değeri ve Minimum uzaklık değeri arasındaki farkı hesapla
15: **for** i = 1 : popülasyon sayısı
16: uyumluluk değerlerini normalize et [0,1]
17: uzaklık değerlerini normalize et [0,1]
18: uzaklık ve uyumluluk değerlerini topla
19: **end**
20: uzaklık ve uyumluluk değerlerini topla, rastgele değerle çarp ve r değerine ata (3.30)
21: En iyi veriyi popülasyondan seç.
22:**end**

EK 3 OBL Algoritmasının Sözde Kodu

1: **for** her x değerini sürü sayısı kadar devam ettir
2: **for** her y değerini her sürüdeki kır kurdu sayısı kadar devam ettir

```
3:            üst sınırı kır kurdu sosyal durumuna göre daralt
4:            alt sınırı kır kurdu sosyal durumuna göre daralt
5         Sürüdeki alfa kır kurtlarına göre sigmoid fonksiyonu oluştur ve eşik değeri belirle
6:          end
7:   end
8:              for her i değerini sonuç sayısı kadar tekrar ettir
9:                 for her j değerini parametre sayısı kadar tekrar ettir
10:                   fark hesapla; Her sürünün alfa değerini aynı sürüdeki arama kurtlarının
                      sosyal durumundan çıkar
11:                      if fark eşik değerinden küçükse daha maksimum değerler olarak
                         değeri tanımla
12:                          g_no=g_no+1
13:                      end if
14:                          eğer değilse en küçük değerler olarak tanımla
15:                          l_no=l_no+1
16:                   end for
17:                   Spearman sıralama korelasyon katsayısını oluşan farka göre hesapla
18:                if eğer spearman sıralama korelasyon katsayısı sıfırdan küçük veya eşitse
19:                   if eğer maksimum değerler minimum değerlerden küçükse
20:                      for her j değeri minimum değer boyutu kadar
21:                         yeni pozisyonları belirle
22:                      end
23:                   else
24:                      for her j değeri maksimum değer boyutu kadar
25:                           yeni pozisyonları belirle
26:                      end
27:                   end
28:                end
36:             end
```

EK 4 Kaotik Bölgesel Arama Algoritmasının Sözde Kodu

```
1:            Rastgele sayılar üret
2:          for j = 1:Kaotik Harita Sayısı;
3:             Mevcut en iyinin yakınındaki adayları belirle
4:             Yakınındaki komşu adayların limitleri aşıp aşmadığını kontrol et
5:          end
6:             Komşu adayları değerlendir,
7:              Yeni en iyi adayı belirle
8:            if Yeni en iyi aday mevcut en iyiden daha iyiyse
9:               en iyi kır kurdunu komşu adayla değiştir, değerlendir
10:           end
```

EK 5 Laplace Crossover Algoritmasının Sözde Kodu

```
1:  Girdi: En iyi parametre, Rastgele parametre , alt Limit, Üst Limit, Boyut
2:  Çıktı: En iyi parametre
3:  Laplace dağılımından yararlanarak rastgele sayı üret (S)
```

$$\beta = \begin{cases} a - b * \log(u), & u \leq 1/2 \\ a + b * \log(u), & u > 1/2 \end{cases}$$

Çözüm_{yeni1} = Çözüm_{eski1} +β*(Çözüm_{eski1} − Çözüm_{eski2})

Çözüm_{yeni2} = Çözüm_{eski2} +β*(Çözüm_{eski1} − Çözüm_{eski2})

4: Yeni çözüm üret

5:**end**

EK 6 Levy Uçuş Algoritmasının Sözde Kodu

1: **Girdi:** Mevcut pozisyon, Pozisyon yönü, alt Limit, Üst Limit

2: **Çıktı:** Yeni pozisyon

3: Adım uzunluğunu hesapla (S)

4: Fark faktörü hesapla DF = 0.01 × S × [Mevcut pozisyon − Pozisyon yönü]

5: Yeni pozisyonu hesapla

Yeni pozisyon = Mevcut pozisyon + S ×$rand$( )×[$size$(Mevcut pozisyon)]

6:Limit kontrolü yap

7:**end**