T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ÇOKLU MOBİL ROBOT SİSTEMLERİ İÇİN METASEZGİSEL VE DERİN ÖĞRENME TABANLI YOL PLANLAMA OPTİMİZASYONU

Hazırlayan
Mustafa Yusuf YILDIRIM

Danışman
Doç. Dr. Rüştü AKAY

Doktora Tezi

Temmuz 2025
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ÇOKLU MOBİL ROBOT SİSTEMLERİ İÇİN METASEZGİSEL VE DERİN ÖĞRENME TABANLI YOL PLANLAMA OPTİMİZASYONU

(Doktora Tezi)

Hazırlayan
Mustafa Yusuf YILDIRIM

Danışman
Doç. Dr. Rüştü AKAY

Bu çalışma, Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FDK-2021-11472 kodlu proje ile desteklenmiştir.

Temmuz 2025
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

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

Mustafa Yusuf Yıldırım

“Çoklu Mobil Robot Sistemleri için Metasezgisel ve Derin Öğrenme Tabanlı Yol Planlama Optimizasyonu” adlı Doktora tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi’ ne uygun olarak hazırlanmıştır.

Hazırlayan	**Danışman**
Mustafa Yusuf YILDIRIM	Doç. Dr. Rüştü AKAY

Mekatronik Mühendisliği ABD Başkanı

Prof. Dr. Şahin YILDIRIM

Doç. Dr. Rüştü AKAY danışmanlığında **Mustafa Yusuf YILDIRIM** tarafından hazırlanan **"Çoklu Mobil Robot Sistemleri için Metasezgisel ve Derin Öğrenme Tabanlı Yol Planlama Optimizasyonu"** adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Mekatronik Mühendisliği** Anabilim Dalında **Doktora** tezi olarak kabul edilmiştir.

… / … / 2025

JÜRİ:

Danışman	: Doç. Dr. Rüştü AKAY	.....
Üye	: Prof. Dr. Şahin YILDIRIM	.....
Üye	: Prof. Dr. Ahmet Turan ÖZDEMİR	.....
Üye	: Doç. Dr. Ahmet Şakir DOKUZ	.....
Üye	: Dr. Öğr. Üyesi M. Kürşat YALÇIN	.....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun ……...… tarih ve ……..……sayılı kararı ile onaylanmıştır.

……. /……../ ……

Prof. Dr. Munise Didem DEMİRBAŞ

Enstitü Müdürü

TEŞEKKÜR

Bana çalışmalarım süresince her türlü yardımı ve fedakârlığı sağlayan saygıdeğer danışman hocam Doç. Dr. Rüştü AKAY'a, optimizasyon konusunu bizlere sevdiren saygıdeğer hocam Prof. Dr. Derviş KARABOĞA'ya ve tez izleme komitemde yer alan hocalarım Prof. Dr. Şahin YILDIRIM, Prof. Dr. Ahmet Turan ÖZDEMİR'e, ayrıca tez savunma sınavı jüri hocalarımdan Doç. Dr. Ahmet Şakir DOKUZ ile Dr. Öğr. Üyesi M. Kürşat YALÇIN'a saygı ve teşekkürlerimi sunarım.

Destek ve anlayışını esirgemeyen sevgili eşim Yasemin YILDIRIM'a, sağladıkları motivasyondan dolayı oğlum Yavuz Alp ve kızım Yağmur Ela'ya, hayatım boyunca verdikleri tüm desteklerden dolayı aileme teşekkür ederim.

Erciyes Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi'ne FDK-2021-11472 kodlu proje desteğinden dolayı teşekkürlerimi sunarım.

Mustafa Yusuf YILDIRIM

Temmuz 2025, KAYSERİ

ÇOKLU MOBİL ROBOT SİSTEMLERİ İÇİN METASEZGİSEL VE DERİN ÖĞRENME TABANLI YOL PLANLAMA OPTİMİZASYONU

Mustafa Yusuf YILDIRIM

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Doktora Tezi, Temmuz 2025
Danışman: Doç. Dr. Rüştü AKAY

ÖZET

Bu tez kapsamında mobil robotların yol planlama süreçlerini iyileştirmek amacıyla dört farklı simülasyon çalışması ve bir gerçek zamanlı çalışma gerçekleştirilmiştir. Küresel yol planlama bağlamında tek bir mobil robotun ızgara tabanlı ortamlarda yol planlaması için iyileştirilmiş bir yapay arı koloni algoritması önerilmiştir. Bu algoritma yol üzerindeki gereksiz köşeleri ortadan kaldırarak daha verimli rotalar oluşturmayı hedefleyen bir strateji ile desteklenmiştir. Aynı zamanda yine küresel çapta yol planlama probleminin karmaşıklığını azaltmak için engellerin kümelenmesine dayalı hibrit bir model önerilmiştir. Bu model metasezgisel algoritmalar ile kümeleme yöntemlerini bir araya getirilmiştir. Yerel yol planlama bağlamında çoklu mobil robot sistemlerinin yol planlaması için bir sinüs kosinüs algoritma varyantı önerilmiştir. Bir robotun dinamik ortamlardaki yol planlama başarısını artırmak için algoritmaya diferansiyel tabanlı güncelleme stratejileri ve adaptif öğrenme mekanizmaları entegre edilmiştir. Ayrıca tek ve çok robotlu sistemlerin küresel yol planlaması için genişletilmiş evrişim ve sıkma-uyarma bloğu gibi katmanların entegre edildiği ResNet tabanlı bir derin öğrenme modeli önerilmiştir. Bu modelin tam evrişimli ağa kıyasla daha optimal yollar ürettiği gözlemlenmiştir. Bu simülasyonlara ek olarak bir mobil robotun bir ortamda planlanan bir yolu istenen şekilde takip etmesi için bir gerçek zamanlı çalışma gerçekleştirilmiştir. Bu çalışmada bir takip kontrolörü tasarlanmış, bu kontrolör kullanılarak robotun planlanan yolda kaydadeğer bir hata olmadan hareket ederek hedef noktaya vardığı gözlemlenmiştir.

Anahtar Kelimeler: Mobil Robot, Yol Planlama, Metasezgisel Algoritmalar, Optimizasyon, Derin Öğrenme

METAHEURISTIC AND DEEP LEARNING BASED PATH PLANNING OPTIMIZATION FOR MULTIPLE MOBILE ROBOT SYSTEMS

Mustafa Yusuf YILDIRIM

Erciyes University, Graduate School of Natural and Applied Sciences
PhD Thesis, July 2025
Supervisor: Assoc. Prof. Dr. Rüştü AKAY

ABSTRACT

In this thesis, four different simulation studies and one experimental study were conducted to improve the path planning processes of mobile robots. In the context of global path planning, an improved artificial bee colony algorithm was proposed for the path planning of a single mobile robot in grid-based environments. This algorithm was supported by a strategy aimed at eliminating unnecessary corners along the path to generate more efficient routes. Moreover, to reduce the complexity of the global path planning problem, a hybrid model based on obstacle clustering was developed by integrating metaheuristic algorithms with clustering techniques. For local path planning, a variant of sine cosine algorithm was proposed for multi-robot systems. To improve the performance of a robot in dynamic environments, differential-based update strategies and adaptive learning mechanisms were incorporated into the algorithm. Furthermore, a ResNet-based deep learning model was introduced for both single and multi-robot global path planning by integrating extended convolution and squeeze-excitation blocks. This model was observed to produce more optimal paths compared to a fully convolutional network. In addition to these simulations, an experimental study was carried out in which a mobile robot was tasked with tracking a planned path in a given environment. In this study, a tracking controller was designed, and it was observed that the robot was able to track the planned path and reach the target point with no significant error.

Keywords: Mobile Robot, Path Planning, Metaheuristic Algorithms, Optimization, Deep Learning

İÇİNDEKİLER

ÇOKLU MOBİL ROBOT SİSTEMLERİ İÇİN METASEZGİSEL VE DERİN ÖĞRENME TABANLI YOL PLANLAMA OPTİMİZASYONU

BİLİMSEL ETİĞE UYGUNLUK ii
YÖNERGEYE UYGUNLUK iii
KABUL VE ONAY iii
TEŞEKKÜR v
ÖZET vi
ABSTRACT vii
İÇİNDEKİLER viii
KISALTMALAR xiii
SİMGELER xv
TABLOLAR LİSTESİ xviii
ŞEKİLLER LİSTESİ xx
GİRİŞ 1

1. BÖLÜM
GENEL BİLGİLER

1.1. Giriş 3
1.2. Literatür Taraması 4
1.2.1. Klasik Algoritma Tabanlı Çalışmalar 5
1.2.2. Metasezgisel Algoritma Tabanlı Çalışmalar 6
1.2.3. Makine Öğrenmesi Tabanlı Çalışmalar 8
1.2.4. Hibrit Algoritma Tabanlı Çalışmalar 10
1.3. Tezin Amacı 12

2. BÖLÜM
YOL PLANLAMA PROBLEMİ

2.1. Giriş 14
2.2. Küresel Yol Planlama 15
2.2.1. Ortam Tanımı 17
2.2.1.1. Graf Tabanlı Ortam 17
2.2.1.2. Izgara Ortamı 18
2.2.1.3. Sürekli Uzay Ortamı 19
2.2.2.Yol Planlama Süreci 20
2.3. Yerel Yol Planlama 21

3. BÖLÜM
YOL PLANLAMA YÖNTEMLERİ

3.1. Giriş 24
3.2. Klasik Algoritmalar 25
3.2.1. Dinamik Programlama 25
3.2.2. Dijkstra Algoritması 27
3.2.3. Genişlik Öncelikli Arama 28
3.2.4. A* Algoritması 29
3.2.5. Hızla Keşfeden Rastgele Ağaç 31
3.3. Metasezgisel Algoritmalar 32
3.3.1. Evrimsel Algoritmalar 33
3.3.1.1. Genetik Algoritma 33
3.3.1.2. Diferansiyel Gelişim 35
3.3.2. Sürü Zekâsı ve İnsan İlhamlı Algoritmalar 36
3.3.2.1. Parçacık Sürü Optimizasyonu 36
3.3.2.2. Yapay Arı Koloni Algoritması 37
3.3.2.3. Öğretme-Öğrenme Tabanlı Optimizasyon 39
3.3.3. Matematik İlhamlı Algoritmalar 41
3.3.3.1. Stokastik Fraktal Arama 41
3.3.3.2. Sinüs Kosinüs Algoritması 43
3.3.3.3. Aritmetik Optimizasyon Algoritması 44

3.4. Makine Öğrenmesi 45
3.4.1. Evrişimli Sinir Ağları 47
3.4.1.1. LeNet-5 48
3.4.1.2. AlexNet 48
3.4.1.3. VGG16 49
3.4.1.4. GoogleNet 50
3.4.1.5. Tam Evrişimli Ağ 51
3.4.1.6. ResNet 51
3.4.2. Takviyeli Öğrenme 52
3.4.2.1. Aktör-Kritik Algoritması 53
3.4.2.2. Q-Öğrenme 54
3.4.2.3. Derin Deterministik Politika Gradyanı 55

4. BÖLÜM
BULGULAR

4.1. Giriş 58
4.2. İyileştirilmiş Yapay Arı Kolonisi Algoritmasını Kullanan Etkili Bir Izgara Tabanlı Yol Planlama Yaklaşımı 59
4.2.1. Problem Tanımı 60
4.2.2. Önerilen Yöntem 62
4.2.2.1. Doğrusallaştırma Stratejisi 62
4.2.2.2. Yerel Arama Stratejisi 64
4.2.2.3. IABC Algoritmasının Karmaşıklığı 67
4.2.3. Bulgular 69
4.2.3.1. IABC Algoritmasının ABC ile Karşılaştırılması 69
4.2.3.2. IABC Algoritmasının Ablasyon Analizi 72
4.2.3.3. IABC Algoritmasının Ölçeklenebilirlik Analizi 73
4.2.3.4. IABC Algoritmasının İyi Bilinen ve Yeni Geliştirilmiş Algoritmalarla Karşılaştırılması 75
4.2.3.5. IABC Algoritmasının Güncel Çalışmalardaki Sonuçlarla Karşılaştırılması 76
4.3. Çok Engelli Ortamlarda Hızlı Yol Planlama 85
4.3.1. Problem Tanımı 85

4.3.2. Önerilen Yöntem 87
4.3.2.1. K-Ortalamalar Kümeleme Algoritması 88
4.3.2.2. Engel Kümeleme Algoritması 89
4.3.3. Bulgular 91
4.3.3.1. Önerilen Modelin Detaylı Analizi 91
4.3.3.2. Önerilen Modelin Farklı Metasezgisel ve Kümeleme Algoritmalarıyla Gerçekleştirimi 99
4.4. Çoklu Robot Yol Planlaması için Çoklu Strateji ve Öz Uyarlamalı Diferansiyel Sinüs-Kosinüs Algoritması 100
4.4.1. Problem Tanımı 101
4.4.1.1. Yol Sapma Hatası 103
4.4.1.2. Kalan Hedef Uzaklığı 104
4.4.2. Önerilen Yöntem 104
4.4.3. Bulgular 108
4.4.3.1. Önerilen sdSCA'nın CEC2015 Test Fonksiyonlarında Gerçekleştirimi 108
4.4.3.2. Önerilen sdSCA'nın CEC2020 Gerçek Dünya Optimizasyon Problemlerinde Gerçekleştirimi 111
4.4.3.3. Önerilen sdSCA'nın Çoklu Robot Yerel Yol Planlama Probleminde Gerçekleştirimi 114
4.5. Izgara Haritalarda ResNet Tabanlı Çoklu-Robot Yol Planlaması 130
4.5.1. Problem Tanımı 131
4.5.2. Önerilen Yöntem 132
4.5.2.1. Veri Seti 132
4.5.2.2. Önerilen IResNet Mimarisi 133
4.5.2.3. Değerlendirme Metrikleri 134
4.5.3. Bulgular 135
4.5.3.1. Tekli Robot Senaryosu 136
4.5.3.2. Çoklu Robot Senaryosu 139
4.6. Bir Mobil Robotun Gerçek Zamanlı Yol Planlaması ve Takip Kontrolü 142
4.6.1. Mobil Robot Tasarımı 142
4.6.2. Mobil Robotun Ters Kinematiği 144
4.6.3. Platform ve Engeller 144

4.6.4. Görüntü İşleme ve Gerçek Zamanlı Yol Planlama 147
4.6.5. Gerçek Zamanlı Yol Takip Kontrolü 151

5. BÖLÜM
SONUÇ VE ÖNERİLER

5.1. Sonuçlar 161
5.2. Gelecekteki Çalışmalar 164

KAYNAKÇA 165
ÖZGEÇMİŞ 176

KISALTMALAR

ABC : Artificial bee colony (yapay arı koloni)

AβHC : Adaptive β-hill climbing (Adaptif β-tepe tırmanışı)

AC : Actor-critic (aktör-kritik)

ACO : Ant colony optimization (karınca koloni optimizasyonu)

AOA : Arithmetic optimization algorithm (aritmetik optimizasyon algoritması)

BFS : Breadth first search (genişlik öncelikli arama)

BWO : Beluga whale algorithm (beyaz balina algoritması)

BWOA : Black widow spider algorithm (karadul örümceği algoritması)

CEC : Congress on evolutionary computation (evrimsel hesaplama kongresi)

CMA-ES : Covariance matrix adaptation evolution strategy (kovaryans matrisi adaptasyon evrim stratejisi)

CNN : Convolutional neural network (evrişimli sinir ağı)

COLSHADE : L-SHADE'nin bir varyantı (L-SHADE: successful history-based adaptive DE with linear population size reduction (doğrusal popülasyon boyutu azaltmalı başarılı geçmişe dayalı adaptif DE)

CR : Crossover rate (çaprazlama oranı)

DDPG : Deep deterministic policy gradient (derin deterministik politika gradyanı)

DE : Differential evolution (diferansiyel gelişim)

DP : Dynamic programming (dinamik programlama)

DSAF : Doğrusallaştırma stratejisi tabanlı amaç fonksiyonu

EABC : Exponential rank ABC (üstel sıralı ABC)

EHO : Elk herd optimizer (geyik sürüsü optimizasyonu)

EKA : Engel kümeleme algoritması

EnMODE : Enhanced multi-operator DE (geliştirilmiş çoklu operatörlü DE)

FCN : Fully convolutional networks (tam evrişimli ağ)

GA : Genetic algorithm (genetik algoritma)

GWO : Grey wolf optimizer (gri kurt optimizasyonu)

HC : Hierarchical clustering (hiyerarşik kümeleme)

HHO : Harris hawk optimization (Harris şahini optimizasyonu)

IABC : Improved ABC (iyileştirilmiş ABC)

IResNet : Improved ResNet (iyileştirilmiş ResNet)

KMC : K-means clustering (K-ortalamalar kümeleme)
LAPO : Lightning attachment procedure optimization (yıldırım tutunma prosedürü optimizasyonu)
MOA : Math optimizer accelerated (matematik optimizasyon hızlandırıcı)
MOP : Math optimizer probability (matematik optimizasyon olasılığı)
MR : Mutation rate (mutasyon oranı)
MSE : Mean square error (ortalama karesel hata)
$OKHU$: Ortalama kalan hedef uzaklığı
$OSYH$: Ortalama yol sapma hatası
PSO : Particle swarm optimization (parçacık sürü optimizasyonu)
PgAFWA : Best firework updating guided adaptive fireworks algorithm (en iyi havaifişek güncelleme destekli adaptif havaifişek algoritması)
QL : Q-learning (Q-öğrenme)
RAM : Random access memory (rastgele erişim belleği)
ReLU : Rectified linear unit (doğrultulmuş doğrusal birim)
ResNet : Residual network (artık ağ)
RRT : Rapidly-exploring random tree (hızla keşfeden rastgele ağaç)
SASS : Self-adaptive spherical search (kendi kendine adaptif küresel arama)
SCA : Sine cosine algorithm (sinüs kosinüs algoritması)
sCMAgES : CMA-ES'nin bir varyantı
sdSCA : Self-adaptive differential SCA (kendi kendine adaptif diferansiyel SCA)
SFS : Stochastic fractal search (stokastik fraktal arama)
SOA : Seagull optimization algorithm (martı optimizasyon algoritması)
SPSA : Salp swarm algorithm (salp sürü algoritması)
STOA : Sooty tern optimization algorithm (isli sumru optimizasyon algoritması)
SSA : Sparrow search algorithm (serçe arama algoritması)
TLBO : Teaching-learning-based optimization (öğrenme-öğretme tabanlı optimizasyon)
WOA : Whale optimization algorithm (balina optimizasyon algoritması)
WSO : White shark optimizer (beyaz köpekbalığı optimizasyonu)
YAS : Yerel arama startejisi
YOLO : You only look once (sadece bir kez bak)

SİMGELER

$\mathbb{R}, \mathbb{N}$	: Reel sayı kümesi, doğal sayı kümesi
$\mathbb{E}, E, O, G$	: Ortam kümeleri ailesi, ortam kümesi, engel kümesi, graf kenar kümesi
$P \ni p$	: Nokta kümesi (veya düğüm / hücre kümesi)
p_s, p_t	: Robotun başlangıç ve hedef konumu
n_p	: Graf veya ızgara ortamında toplam düğüm / hücre sayısı
m, n, h	: Ortamın boyutları (genişlik, yükseklik, derinlik)
p^o, p^{os}, p^{od}	: Tüm engellerin konumu, statik engel konumu, dinamik engel konumu
n_o, n_{os}, n_{od}	: Toplam engel sayısı, statik engel sayısı, dinamik engel sayısı
Y, n_y	: Planlanan yol, planlanan yolu oluşturan nokta / düğüm / hücre sayısı
v, φ	: Robotun doğrusal hızı ve iki boyuttaki yönelimi
ϕ	: Robotun üç boyuttaki dikey yönelimi
f	: Maliyet
$\|\cdot\|$	: İki nokta / düğüm / hücre arasındaki Öklidyen mesafesi
p^e	: DP ve Dijkstra'da ebeveyn düğüm / hücre
n_f	: A*'da uygun komşu düğüm / hücre sayısı
f^g, f^h	: A* algoritmasında gerçek maliyet, sezgisel maliyet
p_r, ε	: RRT'de ortamdan rastgele seçilen nokta, adım büyüklüğü
T, t	: Maksimum iterasyon sayısı, mevcut iterasyon
$X \ni x$	: Popülasyon
x_l, x_h	: Arama sınırları
r, Φ	: Rastgele üretilen sayı
S, D	: Popülasyon boyutu, problem boyutu
δ, ρ	: Seçilme olasılığı, birikimli olasılık
e, l	: GA'da çaprazlama kesme noktaları
$\mathcal{F}$	: Amaç fonksiyonu veya genel fonksiyon sembolü
F	: DE'de ölçekleme faktörü
ϑ, w	: PSO'da parçacıkların hızı, eylemsizlik ağırlığı,
c_1, c_2	: PSO'da bilişsel ve sosyal katsayılar
$\hat{x}, \hat{f}$	: Popülasyondaki en iyi çözüm, en iyi çözümün maliyeti
fit	: ABC'de uygunluk değeri

$\tilde{x}$: TLBO'da popülasyondaki tüm çözümlerin ortalaması

T_F : TLBO'da öğretme faktörü

$n_{dif}, walk$: SFS'de difüzyon sayısı, yürüme oranı

$\mathcal{N}, \sigma$: SFS'de normal dağılım, standart sapma

a : SCA'da r_1 sayısı için ayarlanan parametre

MOA_l, MOA_h : AOA'da MOA parametresinin sınır değerleri

$\alpha^{(AOA)}$: AOA'da kullanım doğruluğunu tanımlayan parametre

μ : AOA'da arama sürecini ayarlayan parametre

s_t, a_t : Takviyeli öğrenmede mevcut durum, mevcut aksiyon

π, r_t : Takviyeli öğrenmede politika, ödül

α, γ : Takviyeli öğrenmede öğrenme oranı, indirim faktörü

A_t, V_θ : AC'de avantaj fonksiyonu, değer fonksiyonu

θ_a, θ_c : AC'de aktörün parametreleri, kritiğin paramatreleri,

μ, Q, μ', Q' : DDPG'de aktör ağı, kritik ağ, hedef aktör ağı, hedef kritik ağ

N_t : DDPG'de rastgele gürültü

n_m, τ : DDPG'de minibatch boyutu, yumuşak güncellenme oranını

θ^Q, θ^μ : DDPG'de aktör ve kritik ağlarının parametreleri

$\theta^{Q'}, \theta^{\mu'}$: DDPG'de hedef aktör ve hedef kritik ağlarının parametreleri

n_V, υ : Engel ihlal sayısı, vektör

$\mathrm{O}(\cdot), t^r$: Büyük O notasyonu, çalışma süresi

$\mathcal{E}$: Dönüş enerji tüketimi

f_l, f_h : WSO'da dalgalı hareketin frekans sınır değerleri

τ, a_0, a_1, a_2 : WSO'da ivme katsayısı, kullanum-keşif dengesini kontrol eden sabitler

B_r : EHO'da boğa oranı

$\mu^{(CMA-E\ )}$: CMA-ES'de ebeveyn sayısı

c_σ, d_σ : CMA-ES'de adım boyutu için öğrenme oranı, sönümleme parametresi

c_c, c_μ : CMA-ES'de kovaryans güncelleme parametreleri

L : Uzunluk

β, V : Engel ihlal faktörü, engel ihlal fonkiyonu

r^o, r^r : Engellerin ve robotların yarıçapı

L_g : Güvenlik mesafesi

$\mathbb{Q} \ni q$: KMC'de küme merkezleri

n_v, n_q : KMC'de veri sayısı, küme sayısı

λ : Kümeleme oranı

p_c, p_n : Robotun mevcut ve yeni konumu

n_r : Robot sayısı

v^{od}, φ^{od} : Dinamik engellerin iki boyuttaki doğrusal hızı ve yönelimi

I, H : Robotun ideal mesafesi, robotun hedefe olan uzaklığı

R : Algoritma koşma sayısı

n_{st}, c : Robotların adım sayısı, stratejilerin seçim sayacı

f^* : Bilinen en iyi fonksiyon değeri

g, h : CEC2020'de eşitsizlik ve eşitlik kısıtlarının sayıları

b : WOA'da logaritmik spiralin şeklini tanımlayan parametre

$\beta^{(HHO)}$: HHO'da Levy uçuşunda kullanılan sabit parametre

M_{truth}, M_{pred} : Gerçek yol matrisi, ağ tarafından tahmin edilen yol matrisi

η_{sc}, η_{op} : Başarı oranı, yol optimalitesi

n_{sc}, n_{op} : Başarılı planlanan yol sayısı, optimal planlanan yol sayısı

n_t : Test verisi sayısı

TABLOLAR LİSTESİ

Tablo 4.1. Bu tez kapsamındaki çalışmalarının özellikleri 59
Tablo 4.2. Ortamların engel oranı, özellikleri ve tasarım kriterleri 70
Tablo 4.3. IABC ve ABC'nin aynı boyuttaki ortamlarda performans karşılaştırması... 70
Tablo 4.4. ABC, IABC-1, IABC-2 ve önerilen IABC algoritmalarının Ortam 2 ve 4 için yol uzunluğu karşılaştırması 72
Tablo 4.5. IABC ve ABC'nin 30 x 30 ve 40 x 40 boyutlardaki ortamlarda performans karşılaştırması 74
Tablo 4.6. Tüm algoritmaların kontrol parametreleri 75
Tablo 4.7. Ortam 2 için IABC'nin iyi bilinen ve yeni geliştirilmiş algoritmalarla performans karşılaştırılması 76
Tablo 4.8. IABC ile ABC ve [81]'deki algoritmaların yol uzunluğu karşılaştırması 79
Tablo 4.9. IABC ile ABC ve [82]'deki algoritmaların yol uzunluğu karşılaştırması 81
Tablo 4.10. IABC ile ABC ve [83]'teki algoritmaların yol uzunluğu karşılaştırması... 82
Tablo 4.11. IABC ile ABC, [84] ve [85]'teki algoritmaların yol uzunluğu karşılaştırması 84
Tablo 4.12. Her iki ortam için elde edilen ortalama yol uzunlukları ve bu uzunlukların kümelemesiz çalışmaya göre artış oranları 93
Tablo 4.13. Her iki ortam için elde edilen ortalama çalışma süreleri ve bu sürelerin kümelemesiz çalışmaya göre azalma oranları 97
Tablo 4.14. Her iki ortam için kümeleme oranlarına göre kazanç oranları 98
Tablo 4.15. TLBO, ABC, DE ve GA'nın kontrol parametreleri..... 99
Tablo 4.16. Her iki ortam için PSO, TLBO, ABC, DE ve GA ile elde edilen ortalama yol uzunlukları ve ortalama çalışma süreleri..... 99
Tablo 4.17. CEC2015 test fonksiyonları..... 108
Tablo 4.18. CEC2015 test fonksiyonları için sdSCA'nın SCA, EABC, AβHC ve PgAFWA ile karşılaştırması 109
Tablo 4.19. CEC2020 gerçek dünya optimizasyon problemleri 111
Tablo 4.20. CEC2020 gerçek dünya optimizasyon problemleri için sdSCA'nın SASS, COLSHADE, EnMODE ve sCMAgES ile karşılaştırması..... 112
Tablo 4.21. Algoritmaların kontrol parametreleri..... 117

Tablo 4.22. SCA ve sdSCA'nın farklı robot sayıları için $OYSH$ karşılaştırması.....117
Tablo 4.23. SCA ve sdSCA'nın farklı robot sayıları için $OKHU$ karşılaştırması.....117
Tablo 4.24. SCA ve sdSCA'nın farklı robot sayıları için toplam uygunluk değeri karşılaştırması 118
Tablo 4.25. SCA ve sdSCA'nın farklı robot sayıları için çalışma süresi karşılaştırması 118
Tablo 4.26. Ortam 1'de her robot ve toplam için ortalama adım sayısı karşılaştırması 120
Tablo 4.27. Ortam 1'de her robot ve toplam için ortalama yol uzunluğu karşılaştırması 121
Tablo 4.28. Ortam 1'de $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları 121
Tablo 4.29. Ortam 2'de her robot ve toplam için ortalama adım sayısı karşılaştırması 124
Tablo 4.30. Ortam 2'de her robot ve toplam için ortalama yol uzunluğu karşılaştırması 125
Tablo 4.31. Ortam 2'de $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları 125
Tablo 4.32. Ortam 3'te her robot ve toplam için ortalama adım sayısı karşılaştırması 127
Tablo 4.33. Ortam 3'te her robot ve toplam için ortalama yol uzunluğu karşılaştırması 128
Tablo 4.34. Ortam 3'te $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları.....129
Tablo 4.35. Tekli robot test verilerinde 10 x 10, 15 x 15 ve 20 x 20 boyutlu ortam verisi ile eğitilen IResNet modelinin farklı test ortamlarında FCN ile karşılaştırması.....136
Tablo 4.36. Çoklu robot test verilerinde 15 x 15 ortam boyutu ile eğitilen IResNet modelinin 1, 2 ve 3 yol bulma durumlarında FCN ile karşılaştırması.....140

ŞEKİLLER LİSTESİ

Şekil 1.1. Literatürdeki yol planlama çalışmalarının genel sınıflandırılması 4
Şekil 2.1. Yol planlama probleminin kapsamı 14
Şekil 2.2. Küresel yol planlamanın blok diyagramı 16
Şekil 2.3. Örnek graf tabanlı ortamlar 18
Şekil 2.4. Örnek ızgara ortamlar 19
Şekil 2.5. Örnek sürekli uzay ortamları 20
Şekil 2.6. Yerel yol planlamanın blok diyagramı 21
Şekil 2.7. Yerel yol planlamada noktasal bir robotun bir sonraki konumu 22
Şekil 3.1. Yol planlama yöntemlerinin sınıflandırılması 25
Şekil 3.2. Makine öğrenmesi ile derin öğrenme kapsamları 46
Şekil 3.3. Tipik bir evrişimli sinir ağı yapısı 47
Şekil 3.4. Temel bir artık blok 52
Şekil 3.5. Tipik bir takviyeli öğrenme yapısı 53
Şekil 4.1. Hücre sayısı ve toplam dönüş açısı kavramları 61
Şekil 4.2. Doğrusallaştırma stratejisinin prensibi 63
Şekil 4.3. Yerel arama stratejisinin prensibi 65
Şekil 4.4. Önerilen IABC algoritmasının akış diyagramı 67
Şekil 4.5. Aynı boyutlarda dört farklı ızgara tipi ortam 69
Şekil 4.6. ABC ve IABC algoritmalarının Ortam 1-4 için planladığı yollar 70
Şekil 4.7. Ortam 1-4 için ortalama yakınsama grafikleri 71
Şekil 4.8. Farklı boyutlarda iki farklı ızgara tipi ortam 73
Şekil 4.9. ABC ve IABC algoritmalarının Ortam 5 ve 6 için planladığı yollar 74
Şekil 4.10. [81], [82], [83], [84] ve [85]'te tasarlanan ortamlar 77
Şekil 4.11. FWOA ve IABC algoritmalarının Ortam 7 için planladığı yollar 78
Şekil 4.12. FWOA ve IABC algoritmalarının Ortam 8 için planladığı yollar 78
Şekil 4.13. IACO-IABC ve IABC algoritmalarının Ortam 9 için planladığı yollar 80
Şekil 4.14. IACO-IABC ve IABC algoritmalarının Ortam 10 için planladığı yollar 80
Şekil 4.15. ISSA ve IABC algoritmalarının Ortam 11 için planladığı yollar 82
Şekil 4.16. ISSA ve IABC algoritmalarının Ortam 12 için planladığı yollar 82
Şekil 4.17. IHMACO ve IABC algoritmalarının Ortam 13 için planladığı yollar 84

Şekil 4.18. IA* ve IABC algoritmalarının Ortam 14 için planladığı yollar 84
Şekil 4.19. Engelden kaçınma kontrolünün temsili çizimi 87
Şekil 4.20. Önerilen modelin akış diyagramı 88
Şekil 4.21. Birden fazla engelin kümelenmesi 90
Şekil 4.22. Önerilen modelin test edildiği ortamlar 92
Şekil 4.23. Her kümeleme oranı için elde edilen ortalama yol uzunlukları 93
Şekil 4.24. Örnek koşma için Ortam 15'te modelin kümelemesiz ve kümelemeli olarak planladığı yollar 94
Şekil 4.25. Örnek koşma için Ortam 15'teki yakınsama eğrileri 95
Şekil 4.26. Örnek koşma için Ortam 16'da modelin kümelemesiz ve kümelemeli olarak planladığı yollar 96
Şekil 4.27. Örnek koşma için Ortam 16'daki yakınsama eğrileri 96
Şekil 4.28. Her kümeleme oranı için elde edilen ortalama çalışma süreleri 97
Şekil 4.29. $\mathcal{F}_1$, $\mathcal{F}_4$, $\mathcal{F}_7$ ve $\mathcal{F}_{11}$ fonksiyonları için SCA ve sdSCA'nın ortalama yakınsama eğrileri 110
Şekil 4.30. $\mathcal{F}_1$, $\mathcal{F}_4$, $\mathcal{F}_7$ ve $\mathcal{F}_{11}$ fonksiyonları için SCA ve sdSCA'nın kutu grafikleri 110
Şekil 4.31. Çoklu robotların yerel yol planlaması için tasarlanan ortamlar 115
Şekil 4.32. SCA ve sdSCA için toplam robot sayısına göre ortalama kalan hedef uzaklığı ve ortalama çalışma süresi 118
Şekil 4.33. Ortam 17'de algoritmalar tarafından elde edilen örnek yollar 120
Şekil 4.34. Ortam 17'de her robot için ortalama adım sayısının çubuk grafiği 121
Şekil 4.35. Ortam 17 için her algoritmanın adım sayısına göre yakınsama eğrileri 122
Şekil 4.36. Ortam 18'de algoritmalar tarafından elde edilen örnek yollar 123
Şekil 4.37. Ortam 18'de her robot için ortalama adım sayısının çubuk grafiği 124
Şekil 4.38. Ortam 18 için her algoritmanın adım sayısına göre yakınsama eğrileri 126
Şekil 4.39. Ortam 19'da algoritmalar tarafından elde edilen örnek yollar 127
Şekil 4.40. Ortam 19'da her robot için ortalama adım sayısının çubuk grafiği 128
Şekil 4.41. Ortam 19 için her algoritmanın adım sayısına göre yakınsama eğrileri 129
Şekil 4.42. Giriş ve çıkış verilerinin bir temsili 132
Şekil 4.43. Önerilen IResNet mimarisi 134
Şekil 4.44. Tekli robot test verilerinde 10 x 10, 15 x 15 ve 20 x 20 boyutlu ortam verisi ile eğitilen IResNet modelinin FCN ile karşılaştırma grafiği 137

Şekil 4.45. Tekli robot test verilerinde 15 x 15 boyutlu ortam verisi ile eğitilen IResNet modelinin her boyuttaki ortamda planladığı optimal ve yarı-optimal yollardan örnekler 138
Şekil 4.46. Çoklu robot test verilerinde 15 x 15 ortam boyutu ile eğitilen IResNet modelinin 1, 2 ve 3 yol bulma durumlarında FCN ile karşılaştırma grafikleri..... 140
Şekil 4.47. Çoklu robot test verilerinde 15 x 15 boyutlu ortam verisi ile eğitilen IResNet modelinin 2 ve 3 yol bulma durumlarında planladığı optimal ve yarı-optimal yollardan örnekler 141
Şekil 4.48. Mobil robotun genel görünümü ve donanımı 143
Şekil 4.49. Mobil robotun elektronik bağlantı şeması 143
Şekil 4.50. Platform 145
Şekil 4.51. Genius Widecam F100 Webcam 145
Şekil 4.52. Ortamın üstten kamera görüntüsü..... 146
Şekil 4.53. Engeller..... 146
Şekil 4.54. Tasarlanan ortamlar 147
Şekil 4.55. Simülasyon çalışmasında yol planlama sonucu..... 148
Şekil 4.56. Gerçek zamanlı çalışmada robotun ve engellerin algılanması..... 150
Şekil 4.57. Gerçek zamanlı çalışmada yol planlama sonucu 151
Şekil 4.58. Gerçek zamanlı çalışmada referans yol üzerinde oluşturulan ara hedef noktaları..... 152
Şekil 4.59. Robotun global eksen takımına göre yönelimi ve ara hedef noktaya olan yönelimi 153
Şekil 4.60. Yol takip kontrolünün blok diyagramı..... 154
Şekil 4.61. Ortam 20 için robotun hareket aşamaları..... 156
Şekil 4.62. Ortam 20 için robotun hareket aşamalarının kamera görüntüleri 157
Şekil 4.63. Ortam 21 için robotun hareket aşamaları..... 158
Şekil 4.64. Ortam 21 için robotun hareket aşamalarının kamera görüntüleri 159
Şekil 4.65. Planlanan ve gerçek yolların karşılaştırması..... 160
Şekil 4.66. RMS hataları 160

GİRİŞ

Mobil robotlar, çevrelerini algılayarak otonom bir şekilde hareket etme kabiliyetine sahip sistemlerdir ve bu özellikleriyle endüstriden tarıma, lojistikten arama-kurtarma çalışmalarına kadar geniş bir yelpazede kullanılmaktadır. Bu robotların otonom hareket kabiliyeti, haritalama, lokalizasyon ve yol planlama gibi temel becerilere dayanır. Haritalama, robotun çevresini tanımasını ve bu çevreyi modelleyerek haritalar oluşturmasını sağlarken, lokalizasyon, robotun bu harita üzerindeki konumunu hassas bir şekilde belirlemesini sağlar. Ancak, haritalama ve lokalizasyon, robotun hedefe ulaşabilmesi için tek başına yeterli değildir; robotun hedefe ulaşması için yol planlama süreci kritik bir rol oynar.

Yol planlama, robotun çevresel engelleri dikkate alarak, güvenli, verimli ve enerji tasarruflu bir yol oluşturmasını hedefler. Bu süreçte, robotun çevresel durumu, hareket hızı, enerji tüketimi ve güvenlik gibi faktörler göz önünde bulundurulur. Yol planlama yöntemleri genel olarak üç ana gruba ayrılır: Klasik arama algoritmaları, metasezgisel algoritmalar ve makine öğrenme tabanlı yaklaşımlar. Klasik arama algoritmaları, genellikle sabit haritalar üzerinde çalışır ve belirli kurallar çerçevesinde sistematik çözümler sunar. Bu yöntemler, iyi tanımlanmış ve basit çevresel koşullarda oldukça etkili olabilir. Öte yandan, metasezgisel algoritmalar, daha karmaşık ve dinamik çevreler için geliştirilmiştir. Bu algoritmalar, geniş çözüm uzaylarında esnek arama yetenekleri sayesinde optimal ya da optimale yakın yollar belirler ve robotun değişen çevre koşullarına uyum sağlamasını kolaylaştırır. Makine öğrenme tabanlı yaklaşımlar ise robotun çevreden gelen geri bildirimlere göre karar verme ve öğrenme becerisini geliştirir. Bu yöntemler, özellikle dinamik engellerin bulunduğu karmaşık ortamlar için etkili çözümler sunar.

Bu tezin amacı, mobil robotların yol planlaması problemlerine yönelik metasezgisel algoritmaların etkinliğini artırmak ve bu alandaki mevcut yöntemlere yenilikçi katkılar sağlamaktır. Çalışma kapsamında hem tekli mobil robot hem de çoklu mobil robot

sistemleri için optimize edilmiş yolların planlanması hedeflenmiş, bu süreçte metasezgisel algoritmaların keşif ve sömürü dengesini iyileştiren yeni yaklaşımlar geliştirilmiştir. Geliştirilen yöntemlerin performansı, literatürdeki standart algoritmalarla karşılaştırılarak değerlendirilmiştir. Ayrıca çoklu robot sistemlerinin yol planlaması için bir derin öğrenme modeli de tanıtılmış ve farklı bir derin öğrenme modeli ile karşılaştırılarak performansı test edilmiştir. Son olarak bir mobil robotun bir ortamda planlanan bir yolu istenen şekilde takip etmesi için bir takip kontrolörü tasarlanmış ve robotun gerçek zamanlı takip kontrolü için bir gerçek zamanlı çalışma gerçekleştirilmiştir.

Tezin organizasyonu şu şekildedir: Birinci bölümde genel bilgiler verilmiş ve yol planlama problemine yönelik literatürdeki güncel çalışmalar sunulmuştur. İkinci bölümde yol planlama problemi ayrıntılı olarak bahsedilmiştir. Model yaklaşımı, ortam modellemesi ve yol planlama kapsamları açıklanmıştır. Üçüncü bölümde yol planlama algoritmaları ayrıntılı olarak bahsedilmiştir. Klasik, metasezgisel ve makine öğrenmesi algoritmaları açıklanmıştır. Dördüncü bölümde tez kapsamında gerçekleştirilen dört simülasyon çalışması ve bir gerçek zamanlı çalışma ayrıntılı olarak sunulmuştur. Son olarak beşinci bölümde ise sonuç ve gelecekteki çalışmalardan bahsedilmiştir.

1. BÖLÜM

GENEL BİLGİLER

1.1. Giriş

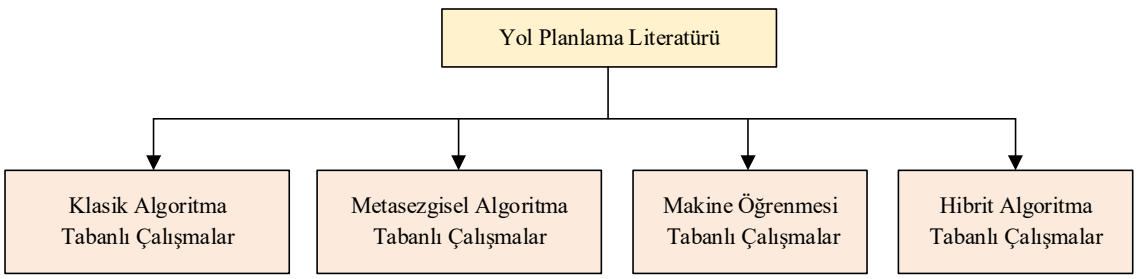
Mobil robotlar, çevrelerini algılayabilme ve bu çevreyle etkileşim kurabilme kapasitesine sahip, otonom sistemler olarak tanımlanabilir. Bu robotlar, genellikle gelişmiş algılayıcılar, işlemciler ve kontrol sistemleri ile donatılır, böylece çevrelerindeki değişikliklere hızla uyum sağlayabilirler. Otonom robotlar, endüstriyel üretimden lojistik çözümlerine, tarım uygulamalarından keşif görevlerine kadar geniş bir yelpazede kullanılır. Modern robotlar, insan müdahalesini azaltarak verimliliği artırmak için tasarlanmıştır. Örneğin, fabrikalarda malzeme taşımak, depolarda envanter yönetimini sağlamak, zorlu arazi koşullarında tarım faaliyetlerini gerçekleştirmek ya da doğal afet bölgelerinde arama-kurtarma operasyonlarına katılmak gibi görevler üstlenebilirler. Ayrıca, otonom araçlar, akıllı robotik sistemler ve insansız hava araçları gibi teknolojilerin temel yapı taşlarını oluştururlar ve gelecekteki yenilikçi uygulamalar için temel oluştururlar [1].

Bir mobil robotun otonom hareket edebilmesi için haritalama, lokalizasyon ve yol planlama gibi temel becerilere sahip olması gerekir. Haritalama, robotun çevresini tanıyıp bu çevreyi haritalar aracılığıyla modellemesine olanak tanırken, lokalizasyon, robotun harita üzerinde kendi konumunu belirlemesini sağlar. Ancak yalnızca haritalama ve lokalizasyon, robotların hedeflerine ulaşmasını sağlamak için yeterli değildir. Yol planlama, robotun belirlenen hedefe güvenli, verimli ve etkili bir şekilde ulaşmasını sağlamak adına kritik bir rol oynar. Bu süreçte, robotun çevresel faktörleri ve engelleri dikkate alarak en uygun yolu seçmesi gerekir. Yol planlamanın etkin bir şekilde yapılabilmesi için, robotun çevresine dair derin bir anlayışa ve gelişmiş hesaplama yeteneklerine sahip olması gerekir [2].

Yol planlama, mobil robotların belirlenen bir hedefe, engellerden kaçınarak ve en uygun yolu seçerek ulaşmasını sağlayan kritik bir süreçtir. Bu süreçte, robotun çevresel durumu, enerji verimliliği, hareket hızı ve güvenlik gibi faktörler dikkate alınır. Yol planlama yöntemleri genel olarak üç ana kategoriye ayrılır: klasik algoritmalar, metasezgisel algoritmalar ve makine öğrenme tabanlı yöntemler. Klasik algoritmalar, genellikle robotun sabit bir çevrede belirli bir harita üzerinde çalıştığı durumlarda kullanılır ve sistematik arama stratejileri ile çözüm üretir. Bu yöntemler, basit ve iyi tanımlı durumlar için oldukça etkili olabilir. Metasezgisel algoritmalar ise daha karmaşık ve sürekli alanlardaki problemler için geliştirilmiştir. Bu yöntemler, geniş çözüm uzaylarında esnek arama yetenekleri sayesinde optimal veya optimale yakın çözümler sunarak, robotun çevresel koşullara daha iyi uyum sağlamasına olanak tanır. Makine öğrenme tabanlı yöntemler ise robotların çevreden aldığı geri bildirimlere göre karar verme becerisini geliştirir. Özellikle, robotun öğrenme ve adaptasyon yeteneklerini destekleyen bu yöntemler, statik ve dinamik engellerin bulunduğu karmaşık ortamlarda başarılı sonuçlar sağlar. Çevresel algılama ve karar verme süreçlerini destekleyen bu yaklaşımlar, robotların hedefe daha verimli ve güvenli bir şekilde ulaşmasını mümkün kılar.

1.2. Literatür Taraması

Bu tez çalışması için mobil robotların yol planlama problemine yönelik literatür taraması yapılmış ve farklı yaklaşımlar üzerine gerçekleştirilen çalışmalar incelenmiştir. Literatürde yol planlama süreci çözmek için Şekil 1.1'de gösterildiği gibi klasik algoritmalar, metasezgisel algoritmalar, makine öğrenmesi ve bu yöntemlerin hibrit yapıları gibi çeşitli yöntemler kullanılmıştır. Bu başlıklar altında, ilgili çalışmalara dair temel bulgular ve yapılan katkılar tartışılmıştır.



Şekil 1.1. Literatürdeki yol planlama çalışmalarının genel sınıflandırılması

1.2.1. Klasik Algoritma Tabanlı Çalışmalar

Klasik algoritma tabanlı çalışmalar genellikle belirli haritalar üzerinde yapılan ve sistematik arama stratejileri kullanan araştırmaları kapsamaktadır. Güncel çalışmalardan bazıları şu şekilde özetlenebilir: Zhang ve Zhao, karmaşık ortamlarda mobil robotlar için çevre bilgisinin amaç fonksiyonuna eklendiği iyileştirilmiş bir A* algoritması önermiştir. Bu algoritma, çocuk düğüm seçimi için optimal kurallar ve yol düzleştirme için çift yönlü optimizasyon stratejisi kullanarak arama alanını küçültür ve yolun düzgünlüğünü artırır [3]. Liu ve arkadaşları, engebeli arazi koşullarında mobil robotlar için yer erişilebilirliği ve yer engebeliği modellerine dayalı olarak iyileştirilmiş bir A* algoritması önermişlerdir. Bu algoritma, yükselti maliyet fonksiyonunu entegre ederek ve orijinal mesafe maliyet fonksiyonu ile birleştirerek daha optimize edilmiş yol planlaması sağlamaktadır [4]. Li ve arkadaşları, A* algoritmasını iyileştirerek mobil robotlar için daha verimli ve düzgün yol planlaması sağlamak amacıyla bir çift yönlü alternatif arama stratejisi ve üssel zayıflama ile ağırlıklandırılmış sezgisel fonksiyon önermiştir. Ayrıca, yol üzerindeki gereksiz düğümleri azaltmak için filtreleme fonksiyonu ekleyerek dönüş açılarını küçültmüş ve Bézier eğrileri kullanarak düzgün yol planlaması sağlamışlardır [5]. Wang ve arkadaşları, verimlilik, sağlamlık ve düğüm gezinti sayısını iyileştirmek amacıyla bir iyileştirilmiş A* algoritması önermişlerdir. Bu geliştirmeler arasında minimum yığın kullanılarak düğüm depolama yapısının optimize edilmesi, adaptif ağırlık ve dönüş cezası eklenerek sezgisel fonksiyonun iyileştirilmesi, 8-komşu stratejisinin 16-komşuya yükseltilmesi ve çift yönlü arama mekanizmasının uygulanması yer almaktadır [6]. Guo ve arkadaşları, hızla keşfeden rastgele ağaç yıldız (rapidly-exploring random tree star, RRT*) algoritmasının yavaş birleşme, yüksek zaman maliyeti ve zayıf çevresel adaptasyon gibi kısıtlamalarını ele alarak, mobil robot yol planlama alanındaki uygulamaların iyileştirilmesi için yeni bir iki yönlü RRT* algoritması önermişlerdir. Bu algoritma, adaptif hedef sapma örnekleme ve değişken adım boyutu gibi yenilikçi mekanizmalar ile hem planlama hızını artırmakta hem de çevresel uyumu iyileştirmektedir [7]. Zhong ve arkadaşları, Halton dizisi tabanlı yeni bir RRT algoritması önermişlerdir. Bu yöntem, düzensiz örnekleme sorunlarını çözmekte ve bellek yetersizliği problemlerini aşmak için aday örnekleme havuzu stratejisini kullanmaktadır. Ayrıca, yol optimizasyonu ve düzeltilmesi için çok seviyeli planlama yaklaşımı ve kübik B-spline yöntemi ile genel planlama kalitesini artırmaktadır [8]. Han ve arkadaşları, yeni bir teta* (theta*) algoritması önermişlerdir ve bu algoritma üç boyutlu haritalarla

birleştirilmiş yol yumuşatma yöntemine dayanmaktadır. Ayrıca bu algoritma, engel bilgilerini kullanarak toplam maliyet fonksiyonunu optimize etmiş ve minimum snap polinomu ile yolun stabilitesini artırmıştır [9]. Huang ve arkadaşları, dar geçitlerdeki yol planlama başarısını artırmak için hibrit uniform örnekleme ve Gaussian örnekleme içeren iyileştirilmiş bir olasılıksal yol haritası (probabilistic roadmap) yöntemi önermişler. Bu yöntem, dar geçitlerde örnekleme yoğunluğunu artırarak ve geniş alanlarda örnekleme fazlalığını azaltarak yol planlamasının etkinliğini ve verimliliğini artırmaktadır [10]. Alshammrei ve arkadaşları, robotun önceden belirlenmiş yolu takip ederken karşılaşabileceği engelleri tespit edip grafiği dinamik olarak güncelleyerek hedef noktasına ulaşmasını sağlamak için iyileştirilmiş bir Dijkstra algoritması önermişlerdir. Ayrıca, elde edilen optimal yol, hareket kontrolüne dönüştürülmüş ve çizgi izleme algılayıcıları kullanılarak pratik olarak uygulanmıştır [11]. Li ve arkadaşları, belirsiz dinamiklere sahip mobil robotlar için iyileştirilmiş bir yapay potansiyel alan (artificial potantial field) yöntemine dayalı yeni bir yol planlama ve kontrol stratejisi önermişlerdir. Bu strateji, temel yapay potansiyel alanın yerel minimuma düşme eğilimini aşmak için çekici potansiyel alan rotasyon yöntemi ve hedefe uzak mesafelerde aşırı çekici kuvvet nedeniyle meydana gelen çarpışmaları önlemek için yeni bir çekici potansiyel alan sınıfı tanımlamaktadır [12].

1.2.2. Metasezgisel Algoritma Tabanlı Çalışmalar

Metasezgisel algoritma tabanlı çalışmalar genellikle karmaşık ve sürekli alanlarda çözüm arayarak daha esnek bir yol planlama yaklaşımı sunmuş ve birçok çalışmada uygulanmıştır. Güncel çalışmalardan bazıları şu şekilde özetlenebilir: Cai ve arkadaşları, ateşböceği algoritmasına (firefly algorithm) dayanan iyileştirilmiş bir karınca koloni optimizasyon algoritması (ant colony optimization, ACO), sezgisel fonksiyon tabanlı iyileştirilmiş bir ACO ve yeni bir yapay potansiyel alan yöntemine dayanan iyileştirilmiş bir ACO olmak üzere üç farklı yöntem önermişlerdir. Bu yöntemler, iki boyutlu ve üç boyutlu ortamlarda mobil robot yol planlama performansını iyileştirerek, yol planlama süresini kısaltmakta ve ortalama yol uzunluğunu azaltmaktadır [13]. Huo ve arkadaşları, Ackermann mobil robotlar için iyileştirilmiş bir ACO ve B-eğri fonksiyonuna dayanan yeni bir yöntem önermişlerdir. Bu yöntem, yol uzunluğu ve yol yumuşatma kısıtlamalarını içeren çok amaçlı bir optimizasyon problemi ile robotun dönüş açısı kısıtlamasını dikkate alarak daha hızlı bir yakınsama oranı sağlar ve kinematik

kısıtlamalara uyumlu yollar üretir [14]. Ab Wahab ve arkadaşları, genetik algoritma (genetic algorithm, GA) için lineer sıralama ve boşluk tabanlı olasılıksal yol haritası tekniklerini içeren iyileştirilmiş bir yöntem önermişlerdir. Bu yöntem, yeni bir nüfus başlatma yöntemi ve çeşitli genetik operatörlerin kombinasyonunu kullanarak yol planlamasının kalitesini artırmakta ve optimum yolun daha hızlı bulunmasını sağlamaktadır [15]. Duan ve arkadaşları, iyileştirilmiş bir egemen olmayan sıralamalı GA-II (non-dominated sorting GA-II) önermişlerdir. Bu algoritma, yol uzunluğu, yol güvenliği ve yol yumuşaklığı gibi çoklu hedefleri aynı anda optimize etmekte, başlangıç popülasyonunun çeşitliliğini artırmak için hibrit bir başlatma stratejisi kullanmakta ve sorunlara özgü evrimsel operatörler geliştirmektedir. Ayrıca, adaptif değişken komşuluk arama stratejisi ve hibrit küresel arama stratejisi kullanarak algoritmanın keşif yeteneğini artırmaktadır [16]. Tao ve Kim, iki alt popülasyona ve rastgele pertürbasyon stratejisine sahip bir parçacık sürü optimizasyon algoritması (particle swarm optimization, PSO) algoritması önermişlerdir. Bu yöntem, parçacıkların kalitesini ve rastgele seçilen bir parçacığın optimal çözümünü dikkate alarak global arama yeteneklerini artırmakta ve yerel arama yeteneklerini lineer bilişsel katsayı ayarlama stratejisi ile güçlendirmektedir. Ayrıca, belirli bir eşik değeri aşıldığında rastgele pertürbasyon eklenerek çeşitlilik artırılmakta ve yerel optimumdan kaçış yeteneği güçlendirilmektedir [17]. Lin ve arkadaşları, çok stratejili sentezlemeye dayalı iyileştirilmiş bir yapay arı koloni (artificial bee colony, ABC) algoritması önermişlerdir. Bu algoritma, insansız hava araçları için daha üstün uçuş yolları sağlamak amacıyla hibrit mekanizma kullanarak popülasyonu başlatmakta ve kesişen rastgele evrim mekanizması ile uçuş yolu kalitesini en üst düzeye çıkarmaktadır [18]. Li ve arkadaşları, kendiliğinden uyarlanan popülasyon boyutuna sahip iyileştirilmiş bir ateşböceği algoritmasına dayanan optimal bir yol planlama yöntemi önermişlerdir. Bu yöntem, çarpışma derecesine göre popülasyon boyutunu belirlemek için iki doğrusal olmayan fonksiyon kullanmakta ve geçersiz çözümleri ortadan kaldırarak yol planlamasının çözüm kararlılığını, yakınsama hızını ve çalışma süresini iyileştirmektedir [19]. Zhang ve Hao, geleneksel mobil robotların karmaşık yol planlama sorunlarını çözmek için iyileştirilmiş bir serçe arama algoritmasına (sparrow search algorithm, SSA) dayalı bir 2D-3D yol planlama yöntemi önermişlerdir. Bu yöntem, ACO tabanlı stratejileri ve bir yerel arama mekanizmasını entegre ederek algoritmanın yol arama yeteneğini artırmaktadır [20]. Tang ve arkadaşları, iyileştirilmiş bir Harris şahini optimizasyon algoritması (harris hawk optimization, HHO)

önermişlerdir. Bu yöntem, çift adaptif ağırlık stratejisi, boyut öğrenme tabanlı avlanma stratejisi ve gübre böceği optimizasyon algoritmasına (dung beetle optimizer) dayalı pozisyon güncelleme stratejisi ile keşif ve sömürü arasında denge kurarak, optimizasyon yeteneği, yakınsama hızını ve kararlılığı önemli ölçüde artırmaktadır [21]. Gao ve arkadaşları, örümcek yaban arısı optimizasyon algoritması (spider-wasp optimizer) tabanlı iyileştirilmiş bir yol planlama yöntemi önermişlerdir. Bu yöntem, algoritmanın küresel optimizasyon performansını artırmak için öğrenme stratejisi, algoritmanın arama kabiliyetini artırmak için çift medyan nokta yönlendirme stratejisi ve yerel optimal yollardan kaçış yeteneğini artırmak için daha iyi bir yönlendirme stratejisi içermektedir [22]. Shi ve arkadaşları, dinamik durumlarda hareketli engellerden kaçınmak için iyileştirilmiş bir simüle edilmiş tavlama algoritması (simulated annealing) algoritması önermişlerdir. Bu algoritma, hesaplama yükünü azaltmak amacıyla başlangıç yol seçimi yöntemi ve silme işlemi kullanmakta ve hem statik hem de dinamik ortamlarda optimal çözümler sağlamaktadır [23]. Yıldırım ve Akay, kullanıcı tanımlı iki boyutlu ortamlarda statik engellerle birlikte mobil robotlar için PSO, ABC ve GA algoritmaları içeren optimal yol planlama yazılımı geliştirmeyi ve bu yazılımı kullanarak algoritmaların performansını analiz etmeyi amaçlamışlardır. Bu yazılım, çalışma alanında farklı şekil ve boyutlarda engeller oluşturarak ve seçilen optimizasyon algoritmasını kullanarak robot için en kısa yolu bulmak üzere tasarlanmıştır [24].

1.2.3. Makine Öğrenmesi Tabanlı Çalışmalar

Makine öğrenmesi tabanlı çalışmalar genellikle robotların çevreden aldıkları geri bildirimlerle karar verme yeteneklerini geliştirmeye odaklanmaktadır. Güncel çalışmalardan bazıları şu şekilde özetlenebilir: Galarza-Falfan ve arkadaşları, derin öğrenmeye dayalı yapay görme sistemlerinin otonom mobil robotlara entegrasyonunu kapsamlı bir şekilde analiz ederek bir yaklaşım önermişlerdir. Gerçek zamanlı nesne tespiti için ResNet18 (residual network) ve YOLOv3 (you only look once) modellerini karşılaştırarak, robotların dinamik ortamlardaki uyarlanabilirliğini ve verimliliğini artırmak için bir yöntem sunmuşlardır [25]. Han, lojistik robotlarının yol planlama ve engel kaçınma yeteneklerini geliştirmek için üç boyutlu bir evrişimli sinir ağı (convolutional neural network, CNN), uzun kısa süreli bellek ve dijkstra algoritmasının yeni bir hibritini önermişlerdir. Nesne tanıma, uzaysal-zamansal modelleme ve optimize edilmiş karar vermeyi birleştirerek, dinamik lojistik ortamlarında robotların otonomisini

ve güvenilirliğini artırmak için bir yaklaşım sunmuşlardır [26]. Farkh ve arkadaşları, bir hizmet robotuna yönelik hedef izleme ve yörünge tahmini için CNN'leri otomatik direksiyon ve hız kontrolü için bir oransal-integral-türevsel denetleyicisiyle birleştiren bir kontrol sistemi önermişlerdir. Sistem, görüntü girişlerine dayalı gerçek zamanlı çıkarımlar için bir Raspberry Pi kullanarak otonom çizgi takibi ve yörünge kontrolüne yönelik bir yaklaşım tanıtmaktadır [27]. Li ve arkadaşları, düşük başarı oranı ve yavaş eğitim hızını gidermek için iyileştirilmiş bir derin deterministik politika gradyan algoritması (deep deterministic policy gradient, DDPG) önermişlerdir. Bu yöntem, öğrenme verimliliğini artırmak için öncelikli deneyim tekrarı ve transfer öğrenimi ile dinamik gecikme güncelleme stratejisi ve Ornstein-Uhlenbeck gürültüsü eklemekte, böylece yol planlamada başarı oranını ve eğitim hızını artırmaktadır [28]. Deshpande ve arkadaşları, belirsizlikleri ele alarak mobil robotların statik ve dinamik engeller karşısında güçlü davranışlar sergilemesi için iyileştirilmiş bir Markov karar süreci (Markov decision process) modelini kullanarak bir yaklaşım önermişlerdir. Bu yaklaşım, iki mobil robot üzerinde yapılan simülasyonlarla test edilerek, gözlem olasılığı yayılımının artırılmasıyla sistemin çarpışma oranını azaltarak dayanıklılığı artırdığını göstermiştir [29]. Deshpande ve arkadaşları, DDPG algoritmasına diferansiyel oyun (differential game) stratejisinin entegre edildiği yeni bir yöntem önermişlerdir. Bu yöntem, başarılı bölümlerin sayısını artırarak ve başarısız bölümlerin sayısını azaltarak daha hızlı öğrenmeyi sağlamaktadır [30]. Zhou ve arkadaşları, yerel yol planlaması için yeni bir eylem seçme politikası, yeni bir ödül fonksiyonu ve kök ortalama kare yayılımı (root mean square propagation) yöntemini içeren iyileştirilmiş bir Q-öğrenme (Q-learning, QL) algoritması önermişlerdir. Bu yöntem, öğrenme hızını artırarak ve yol planlama verimliliğini artırarak mevcut algoritmalara göre tüm metriklerde iyileştirmeler sağlamaktadır [31]. Zhang ve arkadaşları, otonom mobil robotların kinematiklerini dikkate alarak dinamik ortamlarda performans ve yanıt verebilirlik gereksinimlerini karşılamak amacıyla çok ajanlı politika öğrenimi (multi-agent policy learning) tabanlı bir yöntem önermişlerdir. Bu yöntem, merkezi öğrenme ve dağıtılmış yürütme tabanlı bir yol planlama çerçevesi sunmakta olup, geleneksel sinir ağlarını kullanarak kinematiği göz önünde bulundurarak politikanın öğrenilmesini sağlamaktadır. Ayrıca, hata deneyimlerini düzelterek öğrenme süreçlerini hızlandıran geliştirilmiş bir yakın politika optimizasyon algoritması geliştirilmiştir [32]. Yan ve arkadaşları, yörünge açısı, doğrusal hız ve güvenlik derecesini değerlendirme indeksleri olarak kullanıldığı ve çok amaçlı performans indeksini ödül fonksiyonuna

entegre edildiği iyileştirilmiş bir DDPG algoritması önermişlerdir. Bu yöntem ayrıca deneyim örneklerini optimize etmek için bağışıklık optimizasyon algoritmasını (immune optimization algorithm) kullanarak düşük öğrenme ve eğitim verimliliği sorunlarını çözmektedir [33]. Deshpande ve arkadaşları, özel bir yol planlama için kısmen gözlemlenebilir Markov karar süreci (partially observable Markov decision process) matrislerinin boyutlarını ve seyrekliğini kontrol edebilen yeni bir algoritma önermişlerdir. Bu algoritma, duruma ait bileşenlerin ayrıştırılmasının inceliği ve gözlem olasılık dağılımının yayılımı ayarlanarak zaman karmaşıklığı ve bu Markov karar sürecinin çözümünün dayanıklılığı arasında bir denge kurmayı sağlamaktadır [34]. Low ve arkadaşları, üç güncelleme ile iyileştirilmiş bir QL algoritması önermişlerdir. Bu değişiklikler, hedefe doğru yönlendirme için bir mesafe metriği eklenmesi, QL fonksiyonunun daha etkili bir şekilde çıkmazları aşacak şekilde düzeltilmesi ve çıkmazları atlamak için sanal hedef kavramının tanıtılmasıdır [35]. Das ve Mishra, mobil robotun yönlü hareketini sağa ve sola dönüş olarak ayırmak için stokastik gradyan inişine (stochastic gradient descent) dayalı ve doğrusal regresyonun entegre edildiği yeni bir yaklaşım önermişlerdir. Ayrıca, geliştirilen algoritma yol planlama ve navigasyon amaçları için kullanılmıştır [36]. Chen ve arkadaşları, karmaşık kimyasal tesislerdeki altı ayaklı robotların yol planlaması için PSO ve çift derin Q ağı (double deep Q network) algoritmalarına dayalı bir yöntem önermişlerdir. Bu yöntem, rastgele seçim stratejisi yerine PSO algoritmasını kullanarak verileri toplar ve bu verilerle geliştirilen modeli eğitir [37].

1.2.4. Hibrit Algoritma Tabanlı Çalışmalar

Hibrit algoritma tabanlı çalışmalar ise birden fazla yaklaşımın güçlü yönlerini birleştirerek çözüm üretmeyi amaçlayan yeni yöntemler önermektedir. Güncel çalışmalardan bazıları şu şekilde özetlenebilir: Hu ve arkadaşları, bulanık mantık (fuzzy logic), A*, QL ve yapay potansiyel alan yaklaşımlarını içeren bir yöntem önermişlerdir. Bu yöntem, bulanık mantık ile A* algoritmasını, kuantum hesaplama ve çok aşamalı eğitim yöntemlerini birleştirerek A* algoritmasını iyileştirmekte ve QL algoritmasının yakınsama hızını artırmaktadır. Hareketli engellerin bulunduğu ortamlarda, yapay potansiyel alan alt hedef noktalarını planlamak için A* yol noktalarını kullanmaktadır. Yerel minimumlara düşen mobil robotlar için ise kuantum çok aşamalı QL yöntemi devreye girmekte ve yerel minimumlar ile alt hedef noktası arasında bir yol

planlamaktadır [38]. Wang ve arkadaşları, iyileştirilmiş bir A* algoritması, bulanık mantık ve dinamik pencere yaklaşımını (dynamic window approach) birleştiren bir yol planlama yöntemi önermişlerdir. Bu yöntem, çevresel engel oranını dikkate alarak A* algoritmasını iyileştirmekte, arama komşuluğunu optimize ederek düğüm arama verimliliğini artırmakta ve bulanık mantık içeren yerel yol planlama stratejisi ile engellerden güvenli mesafede durarak engel kaçınma kararlılığını artırmaktadır [39]. Tao ve Kim, yumuşak aktör-kritik algoritması (soft actor-critic), karo kodlama ve dinamik pencere yaklaşımını birleştirerek yol planlama problemine yönelik hibrit bir yöntem önermişlerdir. Bu yöntem keşif ve sömürü dengesini sağlamak için otomatik entropi ayar mekanizmasını kullanmakta, iyileştirilmiş özellik temsili için karo kodlamayı entegre etmekte ve hedef başlığı, engel mesafesi ve hız gibi parametrelerle eylem alanını tanımlamak için dinamik pencere yaklaşımını kullanmaktadır [40]. Zhang ve arkadaşları, nükleer enerji santrallerinin rutin denetimlerindeki mobil robotların yol planlaması için iki seviyeli çok amaçlı bir programlama çerçevesi önermişlerdir. Bunun için, iyileştirilmiş bir ACO, GA ve iyileştirilmiş bir A* algoritmasını entegre edilmesiyle yeni bir iki seviyeli hibrit algoritma geliştirilmiştir. Üst seviyede, GA tabanlı düzensiz başlangıç feromon dağılımı, adaptif sezgisel fonksiyon ve feromon güncellemesi için elit strateji ile ACO kullanılarak denetim hedeflerinin optimal geçiş sırası belirlenmiştir. Alt seviyede ise, yol uzunluğu, risk derecesi ve enerji tüketimi gibi birden fazla kısıtlamayı dikkate alarak çift yönlü yolları planlamak için iyileştirilmiş bir A* algoritması kullanılmıştır [41]. Wei ve arkadaşları, bilinmeyen ortamlarda derin pekiştirmeli öğrenme algoritmalarının eğitim süresinin uzun olması ve kararsızlık gibi sorunlarını çözmek için DDPG algoritmasında iyileştirmeler yapmışlardır. Deney havuzu farklı deney havuzlarına bölünmüş, deneyler çeşitli oranlarda toplanarak robotun engellerden kaçınma yeteneği artırılmış ve kılavuz ödül fonksiyonu ile algoritmanın yakınsama hızı iyileştirilmiştir [42]. Gao ve arkadaşları, D*lite algoritması ve dinamik pencere yaklaşımı arasında kombinasyon sağlayan, çift katmanlı bir harita ve uygulanabilir alan stratejisi kullanan bir yeni bir stratejisi önermişlerdir. Bu strateji, D*lite algoritmasının verimliliğini artırmakta ve dinamik pencere yaklaşımınının yerel planlama yeteneklerini tam anlamıyla kullanabilmesini sağlamaktadır [43]. Li ve arkadaşları, iyileştirilmiş bir ACO ve ABC'den oluşan hibrit bir yaklaşım önermişlerdir. Bu yöntem yön bilgisine sahip iyileştirilmiş bir sezgisel mekanizma, yeni komşuluk arama mekanizması ve yol optimizasyon mekanizması içermektedir. Yöntem, yol dönüş sayısını ve yol uzunluğunu

azaltarak algoritmanın yakınsama hızını artırmakta ve yüksek kaliteli yol planlama sonuçları elde etmektedir [44]. Yu ve arkadaşları, su akışı potansiyel alan yöntemi (water flow potential field) ve böcek anten arama algoritmasının (beetle antennae search) kombinasyonuna dayalı bir mobil robot yol planlama yöntemi önermişlerdir. Bu yaklaşım, böcek genetik operatörü kullanarak küresel yolu bölmelere ayırmakta, yerel yol planlamasını engellerin özelliklerine göre sınıflandırmakta ve yapay potansiyel alan ile aramayı yönlendirmektedir [45]. Zhang ve arkadaşları, mobil robotlar için iyileştirilmiş bir ACO, A* ve PSO algoritmasına dayalı hibrit bir yöntem önermişlerdir. Bu yöntem, küresel arama yeteneği ve yakınsama hızını dengelemek amacıyla A* algoritması, iyileştirilmiş feromon ve adaptif sezgisel fonksiyon kullanmaktadır. Ayrıca PSO'yu kullanarak en iyi ACO kontrol parametrelerini ve çok amaçlı ağırlık katsayılarını elde etmektedir [46]. Tian ve arkadaşları, balina optimizasyon algoritması (whale optimization algorithm, WOA) ve ateşböceği algoritmasına dayalı çok popülasyonlu ve tersine öğrenme temelli hibrit bir algoritma önermişlerdir. Bu yöntem, karmaşık mobil robot çalışma ortamlarında optimal yolu hızla bulabilmekte ve keşif ile sömürü arasında denge sağlamaktadır [47].

1.3. Tezin Amacı

Bu tezin amacı, mobil robotların yol planlaması problemlerine yönelik metasezgisel algoritmaların etkinliğini artırmak ve bu alandaki mevcut yöntemlere yenilikçi katkılar sağlamaktır. Günümüzde mobil robotların çeşitli endüstriyel, askeri ve günlük yaşam uygulamalarında artan kullanımı, güvenilir ve verimli yol planlaması algoritmalarına duyulan ihtiyacı artırmaktadır. Özellikle karmaşık ve dinamik ortamlarda, robotların en kısa, en güvenli ve en enerji-verimli yolları planlayarak hedeflerine ulaşması büyük bir önem taşımaktadır.

Bu doğrultuda, mobil robotların otonom hareket kabiliyetlerini geliştirmek adına, hem tekli hem de çoklu mobil robot sistemleri için optimize edilmiş yolların planlanması hedeflenmiştir. Tekli mobil robot sistemlerinde, bir robotun belirlenen başlangıç ve hedef noktaları arasındaki en uygun yolu planlaması amaçlanırken, çoklu mobil robot sistemlerinde ise birden fazla robotun çarpışma önleyici stratejilerle birlikte koordineli bir şekilde hareket etmesi sağlanarak daha karmaşık senaryolara çözüm üretilmiştir.

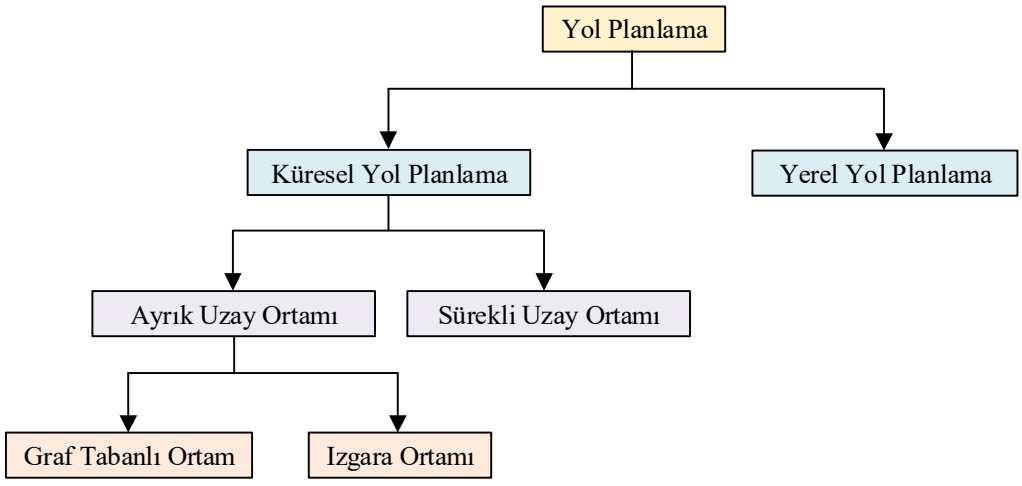
Bu tezde, metasezgisel algoritmaların keşif ve sömürü dengesini iyileştiren yeni yaklaşımlar geliştirilerek, algoritmaların doğruluk, hız ve hesaplama verimliliği açısından performanslarını artırmaya yönelik çeşitli stratejiler uygulanmıştır. Keşif sürecinin daha etkili hale getirilmesi, robotların bilinmeyen veya değişken ortamlarda daha başarılı navigasyon yapmalarını sağlarken, sömürü sürecinin optimize edilmesi ise algoritmaların küresel en iyi çözüme daha hızlı ve hassas bir şekilde ulaşmasına katkı sağlamıştır. Bu çerçevede, farklı metasezgisel algoritmaların geliştirilmesi ve hibrit yöntemlerin tasarlanması gibi çeşitli teknikler araştırılmış ve uygulanmıştır. Ayrıca makine öğrenmesi teknikleri kapsamında bir derin öğrenme modeli de geliştirilmiş ve çoklu robot sistemlerinin yol planlama problemi için uygulanmıştır. Bu sayede eğitim sürecine biraz zaman harcansa da, model eğitildikten sonra her boyutta çok kısa bir sürede sonuç alınabilmiş ve sistemin verimliliği önemli ölçüde artırılmıştır.

2. BÖLÜM

YOL PLANLAMA PROBLEMİ

2.1. Giriş

Mobil robotların yol planlama problemi, bir robotun belirli bir başlangıç noktasından hedef bir noktaya en uygun şekilde ulaşmasını sağlayan bir süreçtir [48]. Bu süreç, robotun belirlenen yolda ilerlerken çevresel engellerden kaçınmasını, enerji tüketimini minimum seviyede tutmasını ve en kısa veya en hızlı yolu bulmasını gerektirir. Yol planlama, robotun otonom şekilde hareket edebilmesi ve belirlenen görevleri yerine getirebilmesi için hayati bir öneme sahiptir. Özellikle sanayi, tarım ve lojistik gibi alanlarda, mobil robotların etkili bir şekilde kullanılabilmesi için yol planlama algoritmalarının doğru ve hızlı sonuçlar üretmesi gerekir. Şekil 2.1 yol planlama probleminin kapsamını göstermektedir.



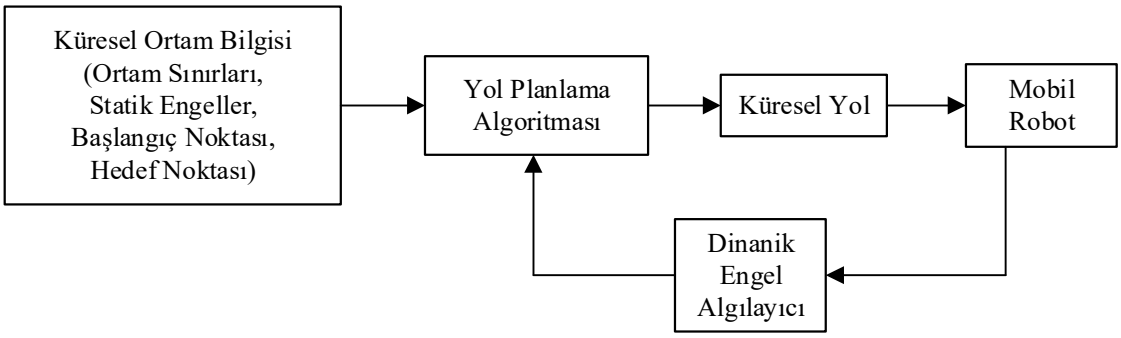
Şekil 2.1. Yol planlama probleminin kapsamı

Bu problem, robotun faaliyet göstereceği alanın uygun şekilde modellenmesiyle başlar. Hareket edilebilecek alan, genellikle ızgara ortamı (ayrık uzay ortamı) veya sürekli uzay ortamı olarak ifade edilir. Izgara ortamlar, hareket alanını hücrelere bölerek robotun hangi hücrelerden geçebileceğini belirler. Sürekli uzay ortamları ise robotun herhangi bir engelle karşılaşmadan hareket edebileceği sürekli alanları ifade eder. Çevresel koşulların doğru bir şekilde modellenmesi, yol planlama algoritmalarının başarılı bir şekilde çalışabilmesi için kritik bir adımdır.

Mobil robotların yol planlama problemleri, çözüm yöntemlerine bağlı olarak iki ana kategoriye ayrılır: küresel yol planlama ve yerel yol planlama. Küresel yol planlama, robotun tüm çevreyi bildiği durumlarda en uygun yolu belirlemeye odaklanır. Bu tür planlama, genellikle önceden bilinen haritalar üzerinde gerçekleştirilir ve uzun vadeli stratejiler gerektirir. Örneğin, bir depoda belirli raflar arasında ürün taşımak için robotun tüm depo haritasına sahip olması ve en kısa yolu planlaması küresel yol planlama kapsamında değerlendirilir. Yerel yol planlama ise robotun yalnızca yakın çevresine dayalı olarak hareket etmesini sağlar. Robot, çevresindeki engelleri algılayarak anlık kararlar alır ve yolunu buna göre günceller. Bu tür planlama, genellikle dinamik ve önceden bilinmeyen ortamlarda, robotun çevresine adapte olması gereken durumlarda tercih edilir. Örneğin, bir tarım arazisinde önceden belirlenmemiş engeller arasında dolaşması gereken bir robot, yerel yol planlama yöntemlerini kullanır. Hem küresel hem de yerel planlama yöntemleri, robotların güvenli ve verimli bir şekilde hareket etmesini sağlamak için birbirini tamamlayıcı bir şekilde kullanılabilir.

2.2. Küresel Yol Planlama

Küresel yol planlama, robotun çevresel bilgilerini başlangıçtan itibaren tam olarak bildiği varsayımına dayanır [49]. Robotun bu bilgiyi kullanarak, hedefe ulaşmasını sağlayan en etkili yolu hesaplaması beklenir. Genellikle, küresel yol planlama otonom araçlar (insansız hava aracı, tekerlekli mobil robot gibi) ve robot kolların çeşitli uygulamalarında kullanılır. Planlamanın temel amacı, robotun hedefine hızlı, güvenli ve enerji açısından verimli bir şekilde ulaşmasını sağlayan küresel çapta bir yol planlamaktır. Küresel yol planlamanın blok diyagramı Şekil 2.2'de gösterilmektedir.



Şekil 2.2. Küresel yol planlamanın blok diyagramı

Robotun hareket ettiği genel ortam iki veya üç boyutlu bir harita şeklinde tanımlanabilir. Bu harita robotun hareket edebileceği geçilebilir alan ile robotun çarpışmaktan kaçınması gereken engelleri içerir. Ayrıca robotun başlangıç ve hedef noktaları da geçilebilir alanlar içinde önceden tanımlanması gerekir. Bu tanımlamalar Eşitlik (2.1)'de gösterilmektedir.

$$\mathbb{E} = \{E \mid \exists O \subseteq E, \{p_s, p_t\} \subseteq E \backslash O\} \tag{2.1}$$

Burada, E genel ortamı kümesini, $E \backslash O$ robotun hareket edebileceği geçilebilir alanı, O engel kümesini, p_s ve p_t robotun sırasıyla başlangıç ve hedef konumlarını temsil eder. Geçilebilir alan, robotun güvenle hareket edebileceği bölgeyi temsil ederken, engeller bu alanların dışındaki tehlikeli veya yasaklı bölgeleri temsil eder. Robotun yolu, yalnızca geçilebilir alan içinde oluşturulabilir ve engellerle çakışmaması gereklidir. Bu tür bir ortam modeli, robotun çevresel bilgileri anlaması ve bu bilgiler doğrultusunda hareket planı oluşturması için temel bir çerçeve sunar.

Planlama sürecinde robotun yalnızca geçilebilir alanda hareket etmesi yeterli değildir; aynı zamanda bu hareketin belirli performans kriterlerini de sağlaması gerekir. Bu kriterler arasında yolun uzunluğu, yolun güvenliği, engellerden kaçınma başarısı ve enerji verimliliği gibi faktörler yer alır. Küresel yol planlama, bu kriterlerin bir kombinasyonunu optimize etmeyi hedefler. Örneğin, bir otonom aracın bir şehirde yol alırken hem en kısa yolu takip etmesi hem de trafik kurallarına ve güvenlik standartlarına uyması beklenir. Benzer şekilde, bir robotun enerji tüketimini minimize ederken engellerden kaçınması ve hedefine en hızlı şekilde ulaşması ideal bir çözüm olarak değerlendirilir. Bu çerçevede küresel yol planlama, genelde bir optimizasyon problemine dönüşür ve çözümü etkili bir matematiksel modelleme ile mümkün olur.

2.2.1. Ortam Tanımı

Küresel yol planlamada ilk adım olarak ortamın doğru bir şekilde tanımlanması gerekmektedir. Bu tanımlama robotun hareket edeceği alanın yapısını belirler ve yol planlamanın temellerini atar. Ortamlar genellikle üç ana kategoriye ayrılır: graf tabanlı ortamlar, ızgara ortamlar ve sürekli uzay ortamları. Graf tabanlı ortamlar, ayrık bir yapıya sahip olup ağ yapıları şeklinde modellenir. Her bir düğüm belirli bir konumu temsil eder ve bu düğümler arasındaki bağlantılar robotun hareket edebileceği yolları gösterir. Izgara ortamları belirli hücrelerden oluşan ve her hücrenin belirli bir durum aldığı ayrık bir yapıyı ifade eder. Sürekli uzay ortamları ise robotun her türlü konumunun ve hareketinin kesintisiz olarak tanımlandığı daha esnek bir model sunar. Bu ortamlar yol planlama algoritmalarının uygulama biçimini ve karmaşıklığını doğrudan etkiler.

2.2.1.1. Graf Tabanlı Ortam

Graf tabanlı ortamlar küresel yol planlamanda genellikle ayrık bir yapıyı modellemek için kullanılır. Bu tür ortamlar ortamı bir dizi düğüm ve bu düğümler arasındaki kenar ile temsil eder. Her bir düğüm robotun belirli bir konumunu veya durumunu, kenarlar ise bu düğümler arasındaki geçiş yollarını ifade eder. Graf tabanlı modelleme genellikle ortamdaki engellerin konumları hakkında bilgi vermez, engel durumu kenarlarda ele alınır. Bir graf tabanlı ortam iki ve üç boyutlu için Eşitlik (2.2)'deki gibi tanımlanır.

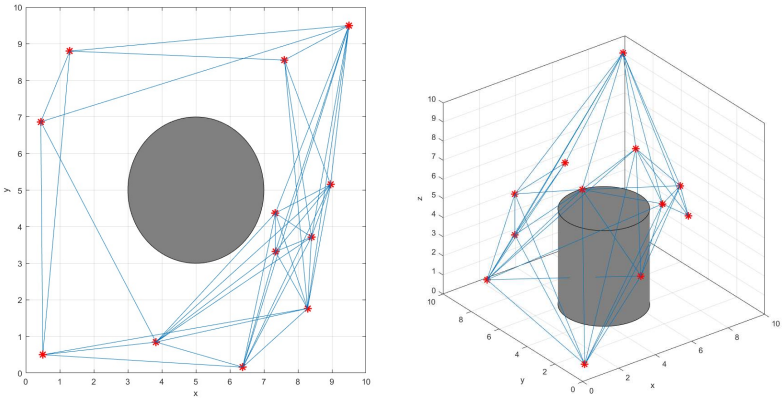
$$E = \{(P, G) \mid P \subset \mathbb{R}^D, G \subseteq P \times P, D \in \{2,3\}\} \tag{2.2}$$

Burada, P düğüm kümesini, G kenar (bağlantı) kümesini ve D ortam boyutunu temsil eder. Düğüm kümesi iki ve üç boyutlu için Eşitlik (2.3)'teki gibi tanımlanır.

$$P = \left\{ p_i \in \mathbb{R}^D \;\middle|\; p_i \notin O \cup \{p_s, p_t\}, i \in \{1,2,\dots,n_p\}, n_p \in \mathbb{N}^+, D \in \{2,3\} \right\} \tag{2.3}$$

Burada, p_i i'inci düğümü ve n_p graftaki toplam düğüm sayısını temsil eder. Kenar kümesi iki ve üç boyutlu için Eşitlik (2.4)'teki gibi tanımlanır. Şekil 2.3 örnek graf tabanlı ortamları göstermektedir.

$$G = \left\{ \{p_i, p_j\} \;\middle|\; \begin{array}{l} p_i, p_j \in P \subset \mathbb{R}^D,\ \{p_i, p_j\} \cap O = \emptyset, \\ i, j \in \{1,2,\dots,n_p\},\ i \neq j,\ D \in \{2,3\} \end{array} \right\} \tag{2.4}$$



(a) **(b)**

Şekil 2.3. Örnek graf tabanlı ortamlar: **(a)** İki boyutlu, **(b)** Üç boyutlu (Gri daire ve gri silindir engelleri, kırmızı yıldızlar düğümleri, mavi çizgiler ise kenarları temsil eder.)

2.2.1.2. Izgara Ortamı

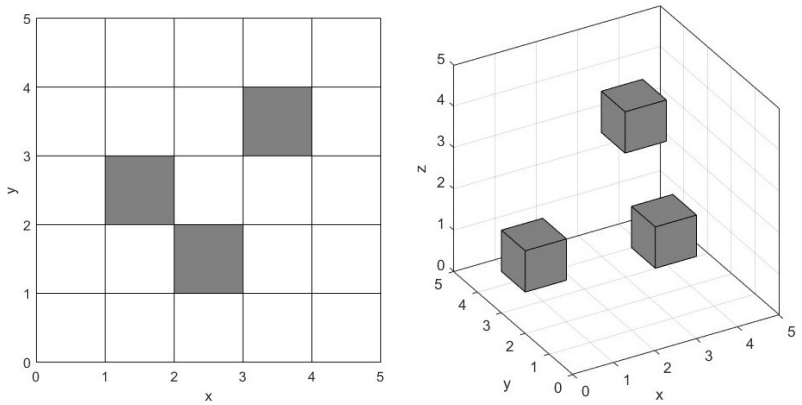
Izgara ortamı, bir robotun hareket edebileceği alanın düzenli bir şekilde bölümlere ayrıldığı ve bu bölümlerin birer hücre olarak tanımlandığı bir çevresel modeldir. Bu yapı, çevrenin dijital olarak temsil edilmesi ve robotun hareket planlamasının kolaylaştırılması amacıyla kullanılır. Izgara ortamı genellikle kare, dikdörtgen veya altıgen gibi düzenli geometrik şekillerden oluşan hücrelerle modellenir. Matematiksel olarak bu ortamlar iki ve üç boyutlu için Eşitlik (2.5) kullanılarak tanımlanır, ızgara hücrelerinden oluşan m x n veya m x n x h boyutunda ızgara matrisleri olarak düşünülebilir.

$$E = \left\{ \begin{matrix} p_{ij} \\ p_{ijk} \end{matrix} \middle| \begin{matrix} i \in \{1,2,\dots,m\}, j \in \{1,2,\dots,n\}, & eğer\ D = 2\ ise \\ i \in \{1,2,\dots,m\}, j \in \{1,2,\dots,n\}, k \in \{1,2,\dots,h\}, & eğer\ D = 3\ ise \end{matrix} \right\} \quad (2.5)$$

Burada, $m, n, h \in \mathbb{N}^+$, p_{ij} ve p_{ijk} ızgara matrisindeki hücreyi; m, n ve h matrisin boyutlarını yani sırasıyla satır, sütun ve katman sayısını temsil eder. Her hücre (p) Eşitlik (2.6)'da tanımlandığı gibi bir durum değişkeni ile işaretlenir: "0" değeri hücrenin geçilebilir alan olduğunu belirtirken, "1" değeri hücrenin bir engel olduğunu ifade eder.

$$p = \left\{ \begin{matrix} 0, & eğer\ p \in E \backslash O\ ise \\ 1, & eğer\ p \in O\ ise \end{matrix} \right\} \quad (2.6)$$

Şekil 2.4 örnek ızgara ortamlarını göstermektedir. Bu ortamların durum değişkenlerini içeren ızgara matrisleri de Eşitlik (2.7) ve (2.8)'de gösterilmektedir.



Şekil 2.4. Örnek ızgara ortamlar: **(a)** İki boyutlu, **(b)** Üç boyutlu (Gri kare ve gri küpler engelleri temsil eder.)

$$E_{2D} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

$$E_{3D} = \left[\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right] \tag{2.8}$$

2.2.1.3. Sürekli Uzay Ortamı

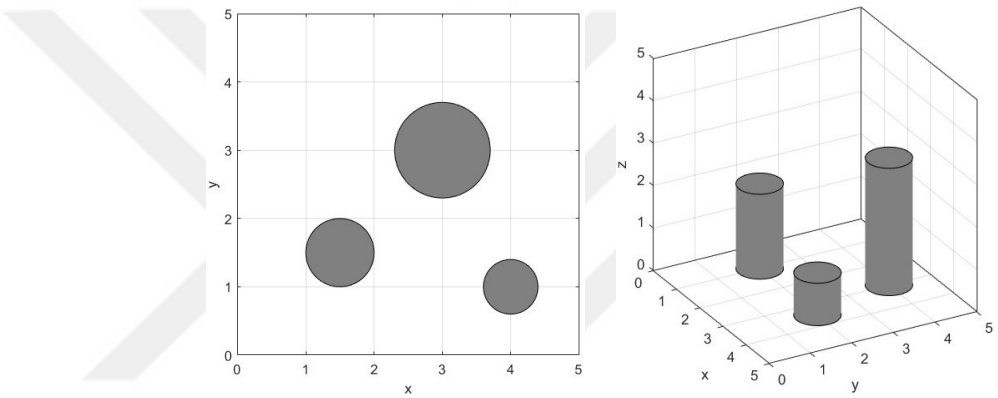
Sürekli uzay ortamı ızgara ortamından farklı olarak, robotun hareket edebileceği ortamın sürekli bir alan olarak temsil edilmesiyle tanımlanır. Bu tür bir ortamda, robotun ve engellerin konumları herhangi bir gerçek sayısal koordinatta ifade edilebilir ve yol düz çizgilerden eğrilere kadar sürekli bir fonksiyon olarak modellenir. Sürekli uzayda, robotun geçilebilir alan içinde engellere çarpmadan en uygun yolu bulması hedeflenir. Bu yaklaşım, yüksek hassasiyet gerektiren ve doğal yolların kritik olduğu durumlarda avantaj sağlar. Sürekli uzay ortamlarının esnekliği, robotun hareketlerini kesintisiz şekilde planlamayı mümkün kılarak, hem enerji verimliliği hem de güvenlik açısından daha optimize çözümler sunar. Matematiksel olarak bu ortamlar iki ve üç boyutlu için Eşitlik (2.9)'da gösterildiği gibi noktalar ve sürekli fonksiyonlarla tanımlanır. Noktalar belirli bir mesafeyle ayrılmamışlardır.

$$E = \{p \in \mathbb{R}^D \mid D \in \{2,3\},\ \exists \mathcal{F}: [0\ 1] \to \mathbb{R}^D,\ \mathcal{F} \in C([0\ 1], \mathbb{R}^D)\} \tag{2.9}$$

Burada, p herhangi bir noktayı temsil eder. Engeller iki ve üç boyutlu için Eşitlik (2.10)'daki gibi ifade edilebilir.

$$O = \{p^o \in \mathbb{R}^D \mid i = \{1,2,\dots,n_o\}, D \in \{2,3\}\} \tag{2.10}$$

Burada, p_i^o i'inci engelin konumunu, n_o engel sayısını temsil eder. Engeller farklı geometrik şekillerde tasarlanabilir. Örneğin, Şekil 2.5 dairesel ve silindir şeklindeki engellere sahip örnek sürekli uzay ortamlarını göstermektedir.



(a) **(b)**

Şekil 2.5. Örnek sürekli uzay ortamları: **(a)** İki boyutlu, **(b)** Üç boyutlu (Gri daire ve gri silindirler engelleri temsil eder.)

2.2.2. Yol Planlama Süreci

Tanımlanan bu ortamlarda çeşitli algoritma ve yöntemlerle başlangıç hücresinden hedef hücreye doğru bir yol planlanır. Bu planlama sonunda graf tabanlı ve ızgara ortamlarda robotun yolu Eşitlik (2.11)'de tanımlandığı gibi belli düğümlerin bir dizisi olarak tanımlanır.

$$Y = \left[p_s, p_1, p_2, \dots, p_i, \dots, p_{n_y}, p_t\right] \tag{2.11}$$

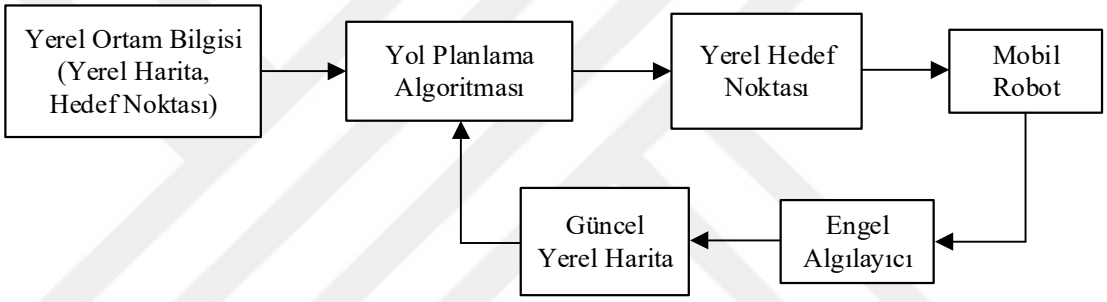
Burada, Y yol dizisini, p_i yolu oluşturan düğümleri veya hücreleri (başlangıç-hedef hariç) ve n_y yolu oluşturan bu düğüm veya hücrelerin sayısını temsil eder. Sürekli uzay ortamlarında robotun yolu iki ve üç boyutlu için Eşitlik (2.12)'de gösterildiği gibi geçilebilir alan içinde genelde noktalardan oluşan sürekli bir dizi olarak tanımlanır.

$$Y = \left\{ p_i \in \mathbb{R}^D \;\middle|\; p_s \leq p_i \leq p_t,\ [p_s, p_t] \rightarrow E \backslash O,\ i \in \{1,2,\dots,n_y\},\ D \in \{2,3\} \right\} \quad (2.12)$$

Burada p_i yolun i'inci noktasını, n_y yolu oluşturan noktaların sayısını temsil eder.

2.3. Yerel Yol Planlama

Yerel yol planlama, robotun çevresel bilgileri tam olarak bilmediği ve hedef noktasına adım adım ulaştığı varsayımına dayanır [50]. Genellikle dinamik engellerin olduğu ortamlarda kullanılan bu planlamada robot sınırlı ve yerel bir alanda bir sonraki noktaya hareket etmek için çalışır. Yerel yol planlamanın blok diyagramı Şekil 2.6'da gösterilmektedir.



Şekil 2.6. Yerel yol planlamanın blok diyagramı

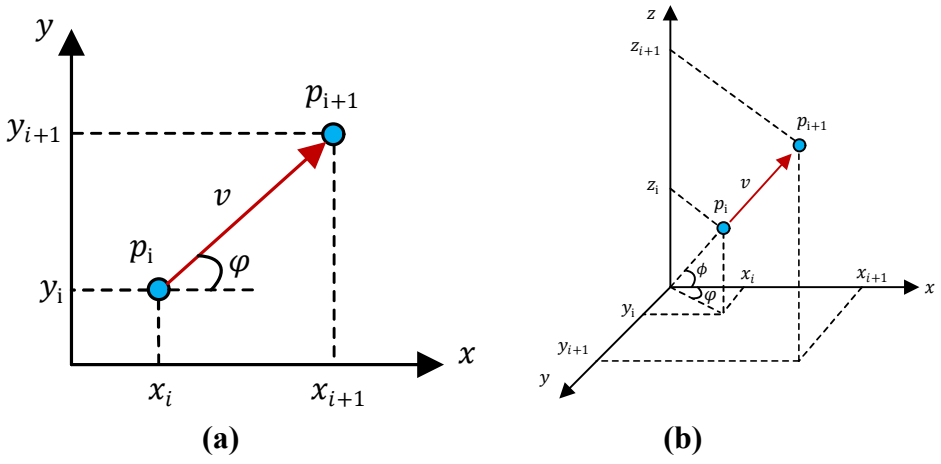
Bu süreçte robot, kinematik modellere ve algılayıcılarından aldığı verilere dayanarak hareket eder. Robotun hareket kabiliyeti, kinematik modeli ile tanımlanır. Kinematik model, robotun nasıl hareket ettiğini ve nasıl yönlendiğini, robotun hızını, yönelimini ve pozisyonunu birleştirerek tanımlar. Robotun pozisyonu, genellikle x ve y koordinatları (iki boyutlu uzayda) ya da x, y ve z koordinatları (üç boyutlu uzayda) ile belirtilir. Her iki durumda da robotun bir sonraki konumunun nasıl hesaplanacağı, robotun hızına ve mevcut yönelimine bağlıdır.

Yerel yol planlama, robotun anlık çevresine odaklanarak engellerden kaçınmasını ve hedefe doğru ilerlemesini sağlar. Bu yöntem, dinamik ortamlarda robotun değişen koşullara hızlı bir şekilde uyum sağlamasına olanak tanır. Çevresel bilgilerin sürekli güncellenmesi, robotun mevcut duruma göre en güvenli ve uygun rotayı seçmesini mümkün kılar. Ayrıca, yerel yol planlama, daha düşük hesaplama gücü gerektirdiğinden, gerçek zamanlı uygulamalarda tercih edilir. Robotun yalnızca yakın çevresindeki verileri

kullanması, büyük haritaların veya önceden belirlenmiş rotaların oluşturulmasını gerektirmez, bu da yönteminin esnekliğini artırır.

Yerel yol planlama, yalnızca robotun yakın çevresine odaklandığından, küresel bir perspektif sağlayamaz. Bu durum, robotun hedefe ulaşırken optimal bir yol seçememesine veya döngüsel hareketlerle sıkışıp kalmasına neden olabilir. Örneğin, robot bir engelin etrafında dönerek yanlışlıkla aynı noktaya geri dönebilir. Ayrıca, yerel planlama, robotun uzun vadeli stratejik bir rota belirlemesini zorlaştırabilir. Dinamik ortamlarda sürekli güncelleme gerekliliği, ani değişikliklerde planlamada gecikmelere veya yanlış kararlar alınmasına yol açabilir. Bu yöntem, karmaşık veya engellerle dolu ortamlarda daha az verimli olabilir ve küresel yol planlama stratejileriyle birlikte kullanılması gerekebilir.

Matematiksel olarak robotun bir sonraki konumunu hesaplamak için kullanılan kinematik model robotun mevcut konumu, yönelimi ve hızını dikkate alır. Noktasal modellenen bir robot için bir sonraki noktanın hesaplanması Şekil 2.7'de gösterilmektedir. Bu hesap iki ve üç boyutlu için sırasıyla Eşitlik (2.13) ve (2.14)'te gösterilmektedir.



Şekil 2.7. Yerel yol planlamada noktasal bir robotun bir sonraki konumu: **(a)** İki boyutlu **(b)** Üç boyutlu (Mavi daire noktasal robotu temsil eder.)

$$p_{i+1} = p_i + v \begin{bmatrix} cos\varphi \\ sin\varphi \end{bmatrix} \Delta t \tag{2.13}$$

$$p_{i+1} = p_i + v \begin{bmatrix} cos\phi . cos\varphi \\ cos\phi . sin\varphi \\ sin\phi \end{bmatrix} \Delta t \tag{2.14}$$

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \quad p_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} \tag{2.15}$$

Burada p_i robotun mevcut konumu, p_{i+1} robotun bir sonraki konumu, v robotun doğrusal hızını, φ robotun iki boyuttaki yönelimini, ϕ robotun üç boyuttaki dikey yönelimini, Δt ise zaman adımını temsil eder. Robotun bir sonraki noktası hesaplandıktan sonra o noktaya yönelir ve planlama hedef noktaya ulaşana kadar bu şekilde devam eder.

3. BÖLÜM

YOL PLANLAMA YÖNTEMLERİ

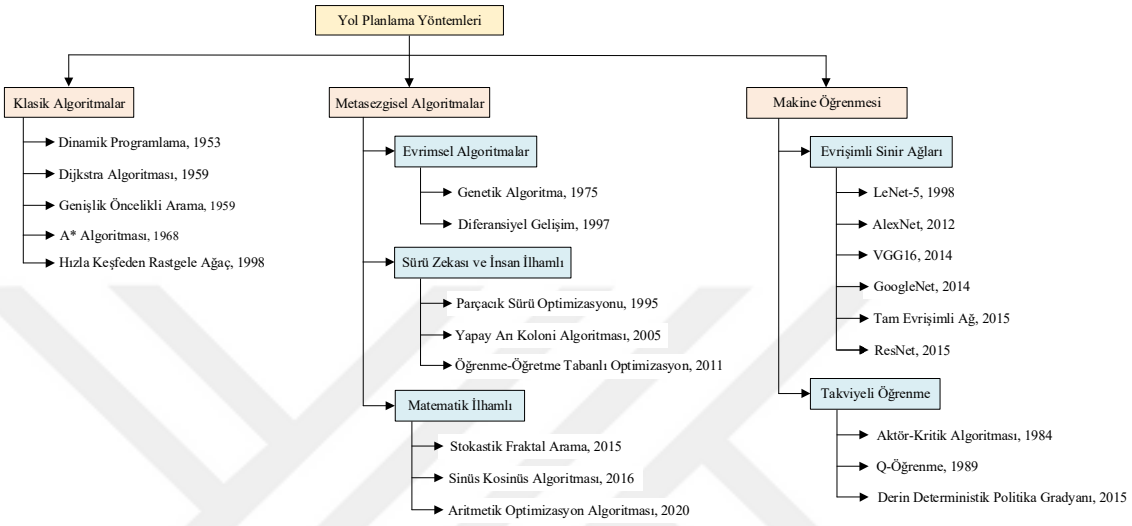
3.1. Giriş

Yol planlama yöntemleri, mobil robotların belirli bir başlangıç noktasından hedef bir noktaya en kısa, en güvenli veya en verimli şekilde ulaşmalarını sağlayan matematiksel ve hesaplamalı algoritmalardır. Bu algoritmalar, robotun hedefe ulaşırken çeşitli çevresel ve operasyonal faktörleri dikkate alarak optimal bir yol bulmasını sağlar.

Bu yöntemlerin temel amacı, robotun hareket ederken çevresindeki engellerden kaçınmasını ve aynı zamanda belirli kısıtlamalara uymasını sağlamaktır. Engeller, statik veya dinamik olabilir ve robotun yol boyunca bu engelleri tespit edip güvenli bir şekilde manevra yapmasını gerektirir. Ayrıca, robotun fiziksel özellikleri, hareket kabiliyeti, enerji tüketimi ve çevredeki dinamik unsurlar gibi faktörler de bu süreçte önemli rol oynar. Örneğin, bir robotun maksimum hız sınırı, batarya kapasitesi veya belirli yüzeylerdeki hareket kabiliyeti gibi özellikler, yol planlama algoritmalarının dikkate alması gereken kısıtlamalardır.

Bu bölüm yol planlama yöntemlerini üç farklı grupta açıklamaktadır: Klasik algoritmalar, metasezgisel algoritmalar ve makine öğrenmesi. Klasik algoritmalar olarak arama algoritmaları dâhilinde dinamik programlama, dijkstra algoritması, genişlik öncelikli arama, A* algoritması ve örnekleme algoritması dâhilinde hızla keşfeden rastgele ağaçtan bahsedilmektedir. Metasezgisel algoritmalar olarak evrimsel algoritma dâhilinde genetik algoritma, diferansiyel gelişim, sürü zekası ve insan ilhamlı dâhilinde parçacık sürü optimizasyonu, yapay arı koloni algoritması, öğrenme-öğretme tabanlı optimizasyon, matematik ilhamlı dâhilinde stokastik fraktal arama, sinüs kosinüs algoritması ve aritmetik optimizasyon algoritmasından bahsedilmektedir. Makine öğrenmesi yöntemleri

olarak evrişimli sinir ağları dâhilinde LeNet, AlexNet, VGG16, GoogleNet, tam evrişimli ağ, ResNet ve takviyeli öğrenme dâhilinde aktör-kritik, Q-öğrenme, derin deterministik politika gradyanı algoritmalarından bahsedilmektedir. Şekil 3.1.'de yol planlama yöntemlerinin sınıflandırılması gösterilmektedir.



Şekil 3.1. Yol planlama yöntemlerinin sınıflandırılması

3.2. Klasik Algoritmalar

Klasik algoritmalar, bir probleme çözüm bulmak amacıyla belirli bir yöntem ve stratejiyle çözüm uzayını keşfeden algoritmalardır. Bu algoritmalar genellikle çözümün doğruluğunu garantilemek için sistematik bir yaklaşım kullanır ve tüm olasılıkları belirli bir sırayla değerlendirir. Bu tür algoritmaların çoğu deterministik bir yapıya sahiptir, yani algoritmanın her çalıştırılmasında aynı başlangıç koşulları ve belirli kurallar doğrultusunda aynı sonuca ulaşılır. Her adımda kullanılan yöntemler sabit olup, herhangi bir rastlantısal unsur barındırmaz. Bu özellikleriyle klasik arama algoritmaları, sonuçların tutarlı ve tekrarlanabilir olmasını sağlar, fakat bazı durumlarda daha verimli olmayan sonuçlar da doğurabilir, çünkü arama alanını tamamen tarama gereksinimi doğurur.

3.2.1. Dinamik Programlama

Dinamik programlama (DP), 1953 yılında Bellman tarafından geliştirilen geniş kapsamlı bir klasik arama algoritmasıdır [51]. Graf ortamlarda yol planlama problemi için DP'nin amacı başlangıç düğümü ile hedef düğümü arasındaki en kısa yolu bulmaktır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: Başlangıç düğümünün ulaşım

maliyeti sıfır, diğer düğümlerin ulaşım maliyeti sonsuz olarak ayarlanır. Başlangıç düğümünden itibaren her düğümün maliyeti özyineli olarak hesaplanır. DP bir graftaki herhangi bir p_i düğümünün ulaşım maliyetini Eşitlik (3.1) kullanılarak hesaplar.

$$f_{p_i} = f_{p_j} + \|p_i - p_j\|, \;\; p_j \in P_{p_i} \tag{3.1}$$

Burada, f_{p_i} p_i düğümünün ulaşım maliyetini, P_{p_i} p_i düğümünün giriş komşu kümesini, p_j giriş komşu kümesindeki herhangi bir komşu düğümü, f_{p_j} başlangıç düğümünden p_j düğümüne kadar olan ulaşım maliyetini, $\|\,.\,\|$ iki düğüm arasındaki maliyeti (mesafeyi) temsil eder. Giriş komşu kümesinden hangi komşu düğümün maliyeti daha düşükse o komşu düğüm ebeveyn düğüm olarak Eşitlik (3.2)'de tanımlandığı gibi atanır.

$$p_i^e = \arg\min_{p_i \in P} \left(f_{p_i}\right) \tag{3.2}$$

Burada, p_i^e p_i düğümüne atanan ebeveyni temsil eder. Bir düğümün ulaşım maliyeti hesaplandıktan sonra bu maliyet diğer düğümlerin maliyetinin hesaplanmasında kullanılır. Bu işlem hedef düğüme ulaşana kadar devam eder. Hedefe ulaşıldıktan sonra geriye izleme aşaması devreye girer. Bu aşamada hedef düğümden başlayarak ve ebeveyn düğümleri geriye doğru izlenerek başlangıç düğümüne kadar en kısa yol Eşitlik (3.3) ve tersi (yolun kendisini elde etmek için) Eşitlik (3.4)'te tanımlandığı gibi planlanır.

$$Y^r = \left\{ \left\{ p_t, p_{n_y}, \dots, p_i, \dots, p_1, p_s \right\} \mid p_{i+1} = p_i^e, i > 0, n_y \in \mathbb{N}^+ \right\} \tag{3.3}$$

$$Y = \{\mathcal{F}(Y^r) \mid \mathcal{F}: Y^r \to Y\} \tag{3.4}$$

Burada, Y yol vektörünü, Y^r yol vektörünün tersini, p_i yolu oluşturan düğümleri, p_s ve p_t başlangıç ve hedef düğümlerini, n_y başlangıç-hedef hariç yolu oluşturan düğüm sayısını temsil eder. Geriye izleme aşaması, algoritmanın sadece maliyet hesaplaması değil, aynı zamanda yol bilgisini de sağlayan önemli bir bileşenidir. Bu adım olmadan, hedef düğüme ulaşmanın maliyeti bilinse bile bu yolun hangi düğümlerden geçtiği bilinemez. Bu yüzden yol planlama probleminde DP için kritik öneme sahiptir. DP algoritmasının temel adımları Algoritma 3.1 ile gösterilmektedir.

Algoritma 3.1: DP algoritmasının temel adımları

```
Tüm düğümlerin maliyetinin sonsuz atanması
Başlangıç düğümünün maliyetinin sıfır atanması
while (tüm düğümlerin maliyeti hesaplanana kadar)
    Her düğümün komşularının tespit edilmesi
    Eşitlik (3.1) ile her düğümün maliyetinin hesaplanması
    Eşitlik (3.2) ile her düğümün ebeveyninin atanması
end while
Geriye izleme aşaması
Yolun ve maliyetin raporlanması
```

3.2.2. Dijkstra Algoritması

Dijkstra algoritması, 1959 yıllarda Dijkstra tarafından geliştirilen bir klasik arama algoritmasıdır [52]. Bu algoritma bir grafın başlangıç düğümünden diğer tüm düğümlere olan en kısa yolları bulmak için kullanılır. Bu algoritmanın temel çalışma prensibi, graf üzerindeki düğümleri ardışık olarak ziyaret ederek her adımda en kısa yolu seçmek ve bu seçimi ilerleyen adımlarda optimize etmektir. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: Başlangıç düğümünün ulaşım maliyeti sıfır, diğer düğümlerin ulaşım maliyeti sonsuz olarak ayarlanır. Başlangıçta yalnızca başlangıç düğümü içeren bir optimum yol vektörü Eşitlik (3.5)'te tanımlandığı gibi oluşturulur.

$$Y = \{p_s\} \tag{3.5}$$

Bu küme algoritmanın ilerleyen adımlarında güncellenir ve en kısa yola dâhil edilen düğümleri içerir. Algoritma bir graftaki herhangi bir p_i düğümünün ulaşım maliyetini Eşitlik (3.6) kullanılarak hesaplar.

$$f_{p_i} = f_{p_j} + \|p_i - p_j\|, \quad p_j \in P_{p_i}, \quad P_{p_i} \subset P \backslash Y \tag{3.6}$$

Burada, f_{p_i} p_i düğümünün ulaşım maliyetini, P_{p_i} p_i düğümünün tüm komşu kümesini, p_j komşu kümesindeki herhangi bir komşu düğümü, f_{p_j} başlangıç düğümünden p_j düğümüne kadar olan ulaşım maliyetini, P düğüm kümesini temsil eder. Bu eşitlikte komşu kümesinin ziyaret edilmeyen yani optimum yol vektörüne eklenmeyen düğümlerin kümesi olduğu görülmektedir. Mevcut düğümden komşu düğümlere olan maliyetler hesaplanır ve en kısa mesafeli düğüm Eşitlik (3.7)'de tanımlandığı gibi seçilir.

$$p_i^e = \arg\min_{p_i \in P} \left(f_{p_i}\right) \tag{3.7}$$

Burada, p_i^e komşular içinde en düşük maliyete sahip düğümü temsil eder. Bu düğüm ziyaret edilmiş olarak işaretlenir ve Eşitlik (3.8)'de tanımlandığı gibi Y kümesine eklenir.

$$Y = Y \cup \{p_i^e\} \tag{3.8}$$

Bu adımlar hedef düğüme ulaşıncaya veya tüm düğümler ziyaret edilinceye kadar devam eder. Sonunda en kısa yol Eşitlik (3.9)'da tanımlandığı gibi planlanır.

$$Y = \left\{ \left\{ p_s, p_1, p_2 \dots, p_i, \dots, p_{n_y}, p_t \right\} \mid i > 0, n_y \in \mathbb{N}^+ \right\} \tag{3.9}$$

Burada, Y yol vektörünü, p_i yolu oluşturan düğümleri, p_s ve p_t başlangıç ve hedef düğümleri, n_y başlangıç-hedef hariç yolu oluşturan düğüm sayısını temsil eder. Dijkstra algoritmasının temel adımları Algoritma 3.2 ile gösterilmektedir.

Algoritma 3.2: Dijkstra algoritmasının temel adımları

```
Tüm düğümlerin maliyetinin sonsuz atanması
Başlangıç düğümünün maliyetinin sıfır atanması
Optimum yol vektörüne başlangıç düğümünün eklenmesi
while (hedef düğüme ulaşana kadar)
    Her düğümün komşularının tespit edilmesi
    Eşitlik (3.6) ile her düğümün maliyetinin güncellenmesi
    Eşitlik (3.7) ile en küçük maliyete sahip komşunun seçilmesi
    Seçilen komşu düğümün optimum yol vektörüne eklenmesi
end while
Yolun ve maliyetin raporlanması
```

3.2.3. Genişlik Öncelikli Arama

Genişlik öncelikli arama (breadth first search, BFS), 1959 yıllarda Moore tarafından geliştirilen bir klasik arama algoritmasıdır [53]. Bu algoritma, graf üzerindeki düğümleri sırasıyla ve eşit mesafede keşfederek, en kısa yolun bulunmasını sağlar. BFS, özellikle ağırlıksız ya da eşit ağırlıklı kenarlarla tanımlanmış graf yapılarında etkili bir şekilde çalışır ve en kısa yolu belirlerken, her bir düğümü ziyaret etme sırasına göre ilerler. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: Başlangıç düğümünün ulaşım maliyeti sıfır, diğer düğümlerin ulaşım maliyeti sonsuz olarak ayarlanır. Bir kuyruk dizisi oluşturulur ve ilk olarak başlangıç düğüm bu diziye eklenir. Başlangıç düğümünden başlamak üzere kuyruğun başındaki düğüm p_i çıkarılır ve p_i düğümümün tüm komşuları

p_j incelenir. Eğer p_j ziyaret edilmemişse mesafesi Eşitlik (3.10)'da tanımlandığı gibi güncellenir.

$$f_{p_j} = f_{p_i} + 1, \quad p_j \in P_{p_i} \tag{3.10}$$

Bu komşular doğrudan kuyruk dizisine eklenir, bu yüzden bu algoritma "en maliyetsiz komşu seçimi" şeklinde bir işlem yapmaz. Bunun yerine, bir düğüm ilk kez keşfedildiğinde, bu düğümün mesafesi otomatik olarak en kısa mesafe olarak kabul edilir. Çünkü her düğüme ulaşma maliyeti eşit ve 1'dir. Kuyruk dizisine eklenen p_i düğümlerinden ilki ziyaret edilmiş olarak işaretlenir ve bu düğümün tüm komşuları üzerinde aynı işlemler gerçekleştirilir. Bu süreç kuyruk dizisi boş olana kadar devam eder. BFS algoritmasının temel adımları Algoritma 3.3 ile gösterilmektedir.

Algoritma 3.3: BFS algoritmasının temel adımları

```
Tüm düğümlerin maliyetinin sonsuz, başlangıç düğümününkinin sıfır atanması
Bir kuyruk dizisinin oluşturulması
Başlangıç düğümünün bu kuyruk dizisine eklenmesi
while (kuyruk dizisine boş olana kadar)
    Kuyruk dizisinin başındaki düğümün çıkarılması
    Çıkarılan düğümün tüm komşularının tespit edilmesi
    Eşitlik (3.10) ile komşuların ulaşım maliyetinin güncellenmesi
    Komşuların kuyruk dizisine eklenmesi
end while
Yolun ve maliyetin raporlanması
```

3.2.4. A* Algoritması

A* algoritması 1968 yılında Hart, Nilsson ve Raphael tarafından geliştirilen ve genellikle en kısa yol problemlerini çözmek için kullanılan sezgisel bir arama algoritmasıdır [54]. Video oyunları ve robotik gibi alanlarda sıkça kullanılmaktadır. Bu algoritma, bir başlangıç düğümünden (veya hücreden) hedef düğüme, ızgara haritası veya graf üzerinde en kısa yolu bulmayı amaçlar. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak başlangıç düğümü mevcut düğüm olarak belirlenir ve bir ana döngü başlar. Bu döngü, başlangıç düğümünden hedef düğüme en kısa yolu bulmak için yinelemeli olarak çalışır. Döngünün her yinelemesinde, mevcut düğümden erişilebilecek uygun komşular, Eşitlik (3.11)'de verilen tanıma göre belirlenir. Bu uygun komşuların belirlenmesi, kenarların herhangi bir engeli ihlal etmemesi kriterine dayanmaktadır.

$$P_{p_i} = \{p_j \mid p_j \in P - \{p_i\},\ \{(p_i, p_j)\} \in E \backslash O,\ j \in \{1,2,\dots,n_f\}\} \tag{3.11}$$

Burada, p_i mevcut düğümü, p_j mevcut düğümün uygun komşu düğümlerini ve n_f ise uygun komşu düğümlerin sayısını temsil eder. Her bir uygun komşu için iki alt maliyet hesaplanır: Gerçek maliyet ve sezgisel maliyet. Gerçek maliyet, mevcut düğüm ile j. uygun komşusu arasındaki toplam maliyeti ifade ederken, sezgisel maliyet, j. uygun komşu ile hedef düğüm arasındaki maliyeti ifade eder. Bu maliyetler sırasıyla Eşitlik (3.12) ve (3.13) kullanılarak hesaplanır.

$$f_{ij}^{g} = f_0 + \|p_i - p_j\| \tag{3.12}$$

$$f_{ij}^{h} = \|p_j - p_t\| \tag{3.13}$$

Burada, f_{ij}^{g} ve f_{ij}^{h} p_i düğümü ile p_j komşusu arasındaki sırasıyla gerçek ve sezgisel maliyetleri, f_0 başlangıç düğümünden p_i düğümüne kadar olan geçmiş maliyeti ve p_t ise hedef düğümü temsil eder. Eşitlik (3.14)'te gösterildiği gibi, toplam maliyet bu iki alt maliyetin toplamıdır.

$$f_{ij} = f_{ij}^{g} + f_{ij}^{h} \tag{3.14}$$

Burada, f_{ij} p_i düğümü ile p_j komşusu arasındaki toplam maliyettir. Her uygun düğümün toplam maliyeti karşılaştırılarak, en düşük maliyete sahip düğüm seçilir ve mevcut düğüm olarak atanır. Bu döngü, hedef düğüme ulaşılana kadar devam eder. Döngünün sonunda elde edilen yol çıktı olarak döndürülür. A* algoritmasının temel adımları Algoritma 3.4 ile gösterilmektedir.

Algoritma 3.4: A* algoritmasının temel adımları

```
Bir yol dizisinin oluşturulması
while (hedef düğüme ulaşıncaya kadar)
    Eşitlik (3.11) ile mevcut düğümün uygun komşularının tespit edilmesi
    Eşitlik (3.12) ile her komşu düğümün gerçek maliyetinin hesaplanması
    Eşitlik (3.13) ile her komşu düğümün sezgisel maliyetinin hesaplanması
    Eşitlik (3.14) ile her komşu düğümün toplam maliyetinin hesaplanması
    En düşük maliyete sahip düğümün yol dizisine eklenmesi
end while
Yolun ve maliyetin raporlanması
```

3.2.5. Hızla Keşfeden Rastgele Ağaç

Hızla keşfeden rastgele ağaç algoritması (rapidly-exploring random tree, RRT) 1998 yılında LaValle tarafından geliştirilen bir klasik arama algoritmasıdır [55]. Bu algoritma özellikle sürekli uzay ortamlarında yol planlama problemlerine çözüm bulmak amacıyla geliştirilmiştir. Algoritma, özellikle yüksek boyutlu uzaylarda yol planlama problemleri için uygundur. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: Algoritma, başlangıçta bir ağaç dizisi Y oluşturur. Bu ağaç ilk olarak sadece başlangıç noktasını içerir. Sürekli uzay ortamında rastgele bir nokta ($p_r \in E \backslash O$) seçilir ve ağaçtan bu noktaya en yakın nokta (p') belirlenir. Bunun için p_r noktası ile ağaçtaki tüm noktalar (p_i) arasında mesafe hesaplanır ve mesafesi minimum olan nokta en yakın nokta olarak Eşitlik (3.15)'te tanımlandığı gibi belirlenir.

$$p' = \arg \min_{p_i \in Y} (\|p_i - p_r\|), \quad i \in \{1,2, \dots, n_y^c\} \tag{3.15}$$

Burada n_y^c o anki ağaçta bulunan nokta sayısıdır. Ardından, p_r yönünde p' noktasından belirli bir adım büyüklüğünde (ε) ilerlenerek yeni bir nokta (p'') Eşitlik (3.16)'da tanımlandığı gibi oluşturulur.

$$p'' = p' + \varepsilon \frac{p_r - p'}{\|p_r - p'\|} \tag{3.16}$$

Eğer p'' engel ihlali yapmıyorsa ($p'' \in E \backslash O$ ise) bu nokta Y ağacına Eşitlik (3.17)'de tanımlandığı gibi eklenir.

$$Y = Y \cup \{p''\} \tag{3.17}$$

Bu süreç maksimum iterasyon sayısına (T) erişene kadar veya hedef noktaya ulaşana kadar yinelemeli olarak devam eder. Hedef noktaya ulaşma kontrolü de eğer p_n'nin hedef noktaya belli bir mesafesinden daha yakın olup olmadığı şeklinde gerçekleşir. Bu durdurma kriterlerinden biri sağlandığında algoritma durur ve yolu raporlar. RRT algoritmasının temel adımları Algoritma 3.5 ile gösterilmektedir.

Algoritma 3.5: RRT algoritmasının temel adımları

Kontrol parametrelerinin ayarlanması (T, ε)
Bir ağacın oluşturulması
Başlangıç noktasının ağaç dizisine eklenmesi
while (hedefe ulaşıncaya veya maksimum iterasyon sayısına erişinceye kadar)
    Ortamda rastgele bir noktanın seçilmesi
    Eşitlik (3.15) ile seçilen noktaya en yakın ağaç noktasının tespit edilmesi
    Eşitlik (3.16) ile yeni noktanın hesaplanması
    Eğer yeni nokta engel ihlali yapmıyorsa bu noktanın ağaç dizisine eklenmesi
end while
Yolun ve maliyetin raporlanması

3.3. Metasezgisel Algoritmalar

Metasezgisel algoritmalar, genellikle karmaşık ve büyük ölçekli optimizasyon problemleri için geliştirilmiş, sistematik olmayan ve çoğu zaman belirli bir çözüm alanına rastgele bir keşif yaparak daha iyi çözümler bulmaya çalışan algoritmalardır. Bu algoritmalar, genellikle başlangıçta rastgele bir çözüm seti ile başlarlar ve daha sonra çözüm alanında ilerleyerek iyileştirmeler yapar. Temelde, optimal çözümü bulmaya yönelik doğrudan bir strateji izlemek yerine, arama alanında geniş bir keşif yaparak potansiyel çözümleri keşfeder ve daha sonra en uygun çözümle yakın sonuçlara ulaşmaya çalışır. Klasik arama yöntemlerinin sınırlamalarını aşmak için kullanılan bu algoritmalar, özellikle yerel optimumlarda sıkışıp kalma riskini en aza indirmeye yönelik tasarlanır.

Metasezgisel algoritmaların en büyük avantajlarından biri, esneklikleri ve çok farklı optimizasyon problemlerine uygulanabilir olmalarıdır. Bu sayede, belirli bir problem için özel olarak geliştirilmiş algoritmalar yerine, tek bir meta-sezgisel algoritma ile çok çeşitli problemlere çözüm üretilebilmektedir. Ayrıca, bu algoritmaların çok sayıda parametreyi kontrol edebilme yetenekleri, problemlerin dinamik doğasına adapte olmalarını sağlar. Bununla birlikte, meta-sezgisel algoritmaların dezavantajları arasında, çözüm kalitesinin her zaman garanti edilmemesi önemli bir sorundur. Çoğu zaman algoritma, global optimumu bulmak yerine yerel optimumda sıkışabilir. Ayrıca, bu tür algoritmaların performansını belirleyen parametrelerin ayarlanması oldukça zordur ve farklı problem koşullarında başarılı olabilmesi için parametrelerin doğru şekilde optimize edilmesi gerekir. Son olarak, hesaplama maliyetlerinin yüksek olabilmesi, özellikle büyük ve karmaşık problemlerde uygulama süresini artırabilir ve çözüm bulmayı daha zor hale getirebilir.

3.3.1. Evrimsel Algoritmalar

3.3.1.1. Genetik Algoritma

Genetik algoritma (genetic algorithm, GA) 1960 yılında Holland tarafından geliştirilen bir metasezgisel algoritmadır [56]. Doğal seçilim ve evrimsel süreçlerinden ilham alınarak geliştirilmiştir. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak bir çözüm kümesi (popülasyon) Eşitlik (3.18) kullanılarak rastgele üretilir.

$$X = x_l + r(x_h - x_l), \quad r \sim U(0,1)^{S \times D} \tag{3.18}$$

Burada, X popülasyonu, $[x_l, x_h]$ popülasyonun arama sınırlarını, r [0 1] aralığında ve $S \times D$ boyutunda sürekli düzgün dağılımda üretilen rastgele sayıları, S popülasyon boyutunu (çözüm kümesindeki çözüm sayısını) ve D ise problemin boyutunu temsil eder. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Ardından popülasyondaki her çözüm seçim, çaprazlama ve mutasyon adı verilen üç işleme tabi tutulur. Seçim işlemi, mevcut çözümlerden yeni çözümler üretmek için ebeveynlerin belirlenmesi işlemidir. Bu aşamanın temel amacı, daha yüksek uygunluk değerine sahip bireylerin seçilme olasılığını artırarak çözümlerin iyileştirilmesini sağlamaktır. Seçim aşamasında genelde rulet tekerleği veya turnuva seçimi stratejileri kullanılır. Rulet tekerleği stratejisinde her çözümün seçilme olasılığı Eşitlik (3.19) kullanılarak hesaplanır.

$$\delta_i = \frac{f_i}{\sum_{j=1}^{S} f_j} \tag{3.19}$$

Burada, δ_i i'inci çözümün seçilme olasılığı, f_i i'inci çözümün maliyeti, eşitliğin payda kısmı ise popülasyondaki tüm çözümlerin maliyetlerinin toplamını temsil eder. Sonra, Eşitlik (3.20) kullanılarak çözümlerin birikimli olasılıkları hesaplanır.

$$\rho_k = \sum_{i=1}^{k} \delta_i \tag{3.20}$$

Burada, $k \in \{1,2,\dots,S\}$ ve ρ_k k'ıncı çözüme kadar olan toplam olasılığı ifade eder. Rastgele bir sayı üretilir ve bu sayı $[\rho_{k-1}, \rho_k]$ aralığında ise (veya ρ_k'dan küçükse) k'ıncı çözüm (x_k) ebeveyn olarak seçilir. Turnuva seçimi stratejisinde popülasyondan S_t sayıda

rastgele çözüm alınır. Bu çözümlerden minimum maliyete sahip çözüm Eşitlik (3.21)'de tanımlandığı gibi seçilir.

$$\{x_{k_1}, x_{k_2}\} = \arg\min_{i \in S_t} f(x_i) \tag{3.21}$$

Burada, $S_t \geq 2$ ve x_k ise ebeveyn olarakseçilen çözümü temsil eder. Bu seçim stratejileriyle iki ebeveyn çözüm seçilir. Sonra çaprazlama oranı (CR) kontrol parametresiyle çaprazlama işlemi başlar. Rastgele bir sayı üretilir ve bu sayı CR'den küçükse çaprazlama işlemi gerçekleştirilir. Bu işlem iki ebeveyn çözüm üzerinde belirli noktalarda keserek çaprazlama yapılır ve iki çocuk çözüm üretilir. Bu kesim tek veya iki noktalı olabilir. Tek noktalı çaprazlama Eşitlik (3.22)-(3.23)'te tanımlanmıştır.

$$x_{h_1} = \{x_{k_1}^j \mid j \in [1, e]\} \cup \{x_{k_2}^j \mid j \in [e+1, D]\} \tag{3.22}$$

$$x_{h_2} = \{x_{k_2}^j \mid j \in [1, e]\} \cup \{x_{k_1}^j \mid j \in [e+1, D]\} \tag{3.23}$$

Burada, x_{h_1} ve x_{h_2} iki çocuk çözümü, x_{k_1} ve x_{k_2} iki ebeveyn çözümü ve e kesme noktasını temsil eder. Çift noktalı çaprazlama Eşitlik (3.24)-(3.25)'te tanımlanmıştır.

$$x_{h_1} = \{x_{k_1}^j \mid j \in [1, e]\} \cup \{x_{k_2}^j \mid j \in [e+1, l]\} \cup \{x_{k_1}^j \mid j \in [l+1, D]\} \tag{3.24}$$

$$x_{h_2} = \{x_{k_2}^j \mid j \in [1, e]\} \cup \{x_{k_1}^j \mid j \in [e+1, l]\} \cup \{x_{k_2}^j \mid j \in [l+1, D]\} \tag{3.25}$$

Burada, e gibi l de bir kesme noktasıdır. Bu şekilde iki yeni çözüm üretilmiş olur. Sonra mutasyon oranı (MR) kontrol parametresiyle mutasyon işlemi başlar. Rastgele bir sayı üretilir ve bu sayı MR'den küçükse mutasyon işlemi gerçekleştirilir. Bu işlemde yeni çocuk çözümlerin herhangi bir boyutu Eşitlik (3.26)'da tanımlandığı gibi değişime uğrar.

$$\left(x_h^j\right)' = \begin{cases} x_h^j + \mathcal{F}_h^j, & Eğer\ r \leq MR\ ise \\ x_h^j, & aksi\ taktirde \end{cases} \tag{3.26}$$

Burada, $h = \{h_1, h_2\}$, $r \sim U(0,1)$ rassal bir sayıyı, x_h^j yeni çocuk çözümlerin j'inci değerini, $\left(x_h^j\right)'$ yeni çocuk çözümlerin j'inci değerinin güncellenmiş hâlini ve $\mathcal{F}_h^j\ x_h^j$ üzerinde yapılan mutasyon fonksiyonunu temsil eder. Bu fonksiyon, mevcut değeri değiştiren herhangi bir matematiksel işlemi temsil edebilir (örneğin, bir bit tersine

çevrilmesi veya mevcut değerin rastgele bir değerle değiştirilmesi). Bu işlemlerden sonra güncel çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değeri hesaplanır. Güncel çözüm mevcut en iyi çözümden daha iyi ise en iyi çözüm olarak atanır. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. GA algoritmasının temel adımları Algoritma 3.6 ile gösterilmektedir.

Algoritma 3.6: GA algoritmasının temel adımları

```
Kontrol parametrelerinin ayarlanması (T, S, CR, MR)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.21) ile iki çözümün seçilmesi
    Eşitlik (3.22)-(3.23) veya (3.24)-(3.25) ile iki yeni çözümün üretilmesi
    Eşitlik (3.26) ile bir çözümün mutasyona tabi tutulması
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması
```

3.3.1.2. Diferansiyel Gelişim

Diferansiyel gelişim (differential evolution, DE) 1997 yılında Storn ve Price tarafından geliştirilen bir metasezgisel algoritmadır [57]. Evrimsel süreçlerden ve fark tabanlı yöntemlerden ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Bu çözümler GA'ya benzer şekilde sırasıyla mutasyon, çaprazlama ve seçim adı verilen üç işleme tabi tutulur. Mutasyon işleminde her çözüm için, diğer iki çözüm arasındaki farkı içeren bir fark vektörü oluşturulur. Bu fark, popülasyonun çeşitliliğini sağlamak için kullanılır. Fark vektörü (DE/rand/1) Eşitlik (3.27) kullanılarak hesaplanır.

$$x_u = x_{r_1} + F\left(x_{r_2} - x_{r_3}\right) \tag{3.27}$$

Burada, $r_1, r_2, r_3 \neq i$, x_u i'inci çözümün fark vektörünü, $x_{r_1}, x_{r_2}, x_{r_3}$ poülasyondan rastgele seçilen çözümleri, F ise [0 1] aralığında değer alan ölçekleme faktörünü temsil eder. Çaprazlama işleminde fark vektörü kullanılarak yeni aday çözüm önerisi yapılır. Bu işlemde rastgele bir sayı üretilir ve bu sayı çaprazlama oranından küçükse mevcut çözüm

fark vektörü ile değiştirilir. Üretilen rastgele sayı çaprazlama oranından büyükse mevcut çözüm değişmez. Bu işlem Eşitlik (3.28)'de tanımlanmıştır.

$$x'_u = \begin{cases} x_u, & Eğer\ r \leq CR\ ise \\ x_i, & aksi\ taktirde \end{cases} \tag{3.28}$$

Burada, x_i i'inci çözümü, x'_u i'inci çözüm için yeni aday çözümü, $r \sim U(0,1)$ rassal bir sayıyı ve CR çaprazlama oranını temsil eder. Seçim işleminde üretilen aday çözüm ve mevcut çözümün maliyetleri karşılaştırılır. Eğer aday çözümün maliyeti mevcut çözümünkinden daha iyi ise mevcut çözüm yeni aday çözümle değiştirilir. Bu işlem Eşitlik (3.29)'da tanımlanmıştır.

$$x'_i = \begin{cases} x'_u, & Eğer\ f(x'_u) \leq f(x_i)\ ise \\ x_i, & aksi\ taktirde \end{cases} \tag{3.29}$$

Burada, x'_i i'inci çözümün güncellenmiş hâlini temsil eder. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. DE algoritmasının temel adımları Algoritma 3.7 ile gösterilmektedir.

Algoritma 3.7: DE algoritmanın temel adımları

```
Kontrol parametrelerinin ayarlanması (T, S, F, ς)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.27) ile fark vektörünün üretilmesi (mutasyon)
    Eşitlik (3.28) ile yeni aday çözümün üretilmesi (çaprazlama)
    Eşitlik (3.29) ile mevcut çözümün güncellenmesi (seçim)
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması
```

3.3.2. Sürü Zekâsı ve İnsan İlhamlı Algoritmalar

3.3.2.1. Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu (particle swarm optimization, PSO) 1995 yılında Kennedy ve Eberhart tarafından geliştirilen bir metasezgisel algoritmadır [58]. Kuş ve balık sürülerinin beslenme davranışlarından ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu

popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Ayrıca popülasyondaki her çözümün hızları da mevcuttur ve başlangıçta rastgele veya sıfır olarak üretilir. Yinelemeli süreç başladığında bu hızlar Eşitlik (3.30) kullanılarak güncellenir.

$$\vartheta_i' = w\vartheta_i + rc_1(\hat{x}_i - x_i) + rc_2(\hat{x} - x_i) \tag{3.30}$$

Burada, $r \sim U(0,1)^{1\times D}$ rassal bir sayıyı, ϑ_i i'inci çözümün hızını, ϑ_i' i'inci çözümün hızının güncellenmiş hâlini, w eylemsizlik ağırlığını, c_1 ve c_2 bilişsel ve sosyal katsayıları, $\hat{x}_i$ i'inci çözümün o ana kadarki en iyi değerini, $\hat{x}$ popülasyondaki en iyi çözümü, x_i ise i'inci çözümü temsil eder. Bu hızlar güncellendikten sonra çözümler Eşitlik (3.31) kullanılarak güncellenir.

$$x_i' = x_i + v_i \tag{3.31}$$

Burada, x_i' i'inci çözümün güncellenmiş hâlini temsil eder. Bu işlemlerden sonra güncel çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değeri hesaplanır. Bu çözüm mevcut en iyi çözümden daha iyi ise en iyi çözüm olarak atanır. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. PSO algoritmasının temel adımları Algoritma 3.8 ile gösterilmektedir.

Algoritma 3.8: PSO algoritmanın temel adımları

```
Kontrol parametrelerinin ayarlanması (T, S, w, c1, c2)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.30) ile çözümlerin hızlarının güncellenmesi
    Eşitlik (3.31) ile çözümlerin güncellenmesi
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması
```

3.3.2.2. Yapay Arı Koloni Algoritması

Yapay arı koloni algoritması (artificial bee colony, ABC) 2005 yılında Karaboğa tarafından geliştirilen bir metasezgisel algoritmadır [59]. Yiyecek arayışında olan bal arılarının işbirliğinden ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde

açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Bu başlangıç popülasyonu işçi arı, gözcü arı ve kâşif arı aşamalarına yönlendirilir. İşçi arı aşamasında bir çözümün rastgele bir parametresi seçilir ve bu parametre Eşitlik (3.32) kullanılarak güncellenir.

$$\left(x_i^j\right)' = x_i^j + \Phi_i^j\left(x_i^j - x_{r_1}^j\right) \tag{3.32}$$

Burada, $r_1 \neq i$, x_i^j i'inci çözümün j'inci parametresini, $\left(x_i^j\right)'$ i'inci çözümün j'inci parametresinin güncellenmiş hâlini, $x_{r_1}^j$ popülasyondan rastgele seçilen bir çözümün j'inci parametresini ve $\Phi_i^j \sim U(0,1)$ ise rassal bir sayıyı temsil eder. Bu işlemlerden sonra güncel çözüm amaç fonksiyonunda değerlendirilir ve maliyeti hesaplanır. Bu maliyet kullanılarak farklı bir uygunluk değeri hesaplanır. Bu hesaplama Eşitlik (3.33)'te gösterilmektedir.

$$fit_i = \begin{cases} \dfrac{1}{1+f_i}, & Eğer\ f_i \geq 0\ ise \\ 1 + |f_i|, & aksi\ taktirde \end{cases} \tag{3.33}$$

Burada, f_i i'inci çözümün maliyetini, fit_i ise i'inci çözümün uygunluk değerini temsil eder. Gözcü arı aşamasına geçmeden önce uygunluk değerleri kullanılarak her çözümün seçilme olasılığı Eşitlik (3.34)'te gösterildiği gibi hesaplanır.

$$\delta_i = \frac{fit_i}{\sum_{j=1}^{S} fit_j} \tag{3.34}$$

Burada, δ_i i'inci çözümün seçilme olasılığını temsil eder. Gözcü arı aşamasında, tercih edilen çözümler daha iyi uygunluk değerlerine sahip olanlara yönlendirilir, bu da daha yüksek uygunluk değerlerine sahip olanların seçilme olasılığını artırır. Her çözüm için rastgele bir sayı üretilir ve bu sayı ilgili çözümün seçilme olasılığından küçükse, ilgili çözüm Eşitlik (3.32) kullanılarak güncellenir. Güncellenen çözüm amaç fonksiyonunda değerlendirilir, maliyeti elde edilir ve uygunluk değeri Eşitlik (3.33) kullanılarak hesaplanır. Kâşif arısı aşamasında, belirli bir limit değerinde güncellenmeyen çözümler yerine Eşitlik (3.18) kullanılarak yeni çözümler üretilir. Yinelemeli süreç maksimum

iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. ABC algoritmasının temel adımları Algoritma 3.9 ile gösterilmektedir.

Algoritma 3.9: ABC algoritmanın temel adımları

```
Kontrol parametrelerinin ayarlanması (T, S, limit)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.32) ile çözümlerin güncellenmesi
    Eşitlik (3.33) ile çözümlerin uygunluk değerlerinin hesaplanması
    Eşitlik (3.34) ile her çözümün seçilme olasılıklarının hesaplanması
    Eşitlik (3.32) ile seçilen çözümlerin güncellemesi
    Eşitlik (3.18) ile limiti aşan çözümlerin yerine rastgele çözümlerin üretilmesi
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması
```

3.3.2.3. Öğretme-Öğrenme Tabanlı Optimizasyon

Öğretme-öğrenme tabanlı optimizasyon (teaching-learning-based optimization, TLBO) 2011 yılında Rao ve arkadaşları tarafından geliştirilen bir metasezgisel algoritmadır [60]. Bir sınıf ortamında gerçekleşen öğretmen-öğrenci etkileşimi ve öğrencilerin birbirlerinden öğrenmesi süreçlerinden ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Öncelikle popülasyondaki tüm çözümlerin ortalaması Eşitlik (3.35) kullanılarak hesaplanır.

$$\tilde{x} = \frac{1}{S}\sum_{i=1}^{S} x_i \tag{3.35}$$

Burada, x_i i'inci çözümü temsil eder. Ardından, popülasyondaki en iyi çözüm öğretmen olarak seçilir. Bu çözüm, öğretmen olarak görev yapacak ve diğer çözümleri yönlendirecektir. Algoritma bu hesaplamalardan sonra öğretmen ve öğrenci olmak üzere iki aşamayı takip eder. Öğretmen aşamasında her bir çözüm öğretmene ve popülasyonun ortalamasına doğru hareket eder. Bu hareketin ne kadar olacağı, rastgele belirlenen

öğretme faktörü parametresi ile belirlenir. Bu değer çözümün öğretmene ne kadar yaklaşacağını kontrol eder. Buna göre çözümler Eşitlik (3.36) kullanılarak güncellenir.

$$x_i' = x_i + r(\hat{x} - T_F\tilde{x}) \tag{3.36}$$

Burada $r \sim U(0,1)^{1\times D}$ rassal bir sayıyı, x_i i'inci çözümü, x_i' i'inci çözümün güncellenmiş hâlini, , $\hat{x}$ popülasyondaki en iyi çözümü, $T_F \sim \{1,2\}$ ise öğretme faktörünü temsil eder. Öğrenci aşamasında ise çözümler birbirlerinden öğrenir ve Eşitlik (3.3) kullanılarak güncellenir.

$$x_i'' = \begin{cases} x_i' + r\left(x_i' - x_{r_1}\right), & Eğer\ f(x_i') < f\left(x_{r_1}\right)\ ise \\ x_i' + r\left(x_{r_1} - x_i'\right), & aksi\ taktirde \end{cases} \tag{3.37}$$

Burada, $r_1 \neq i$, $r \sim U(0,1)^{1\times D}$ rassal bir sayıyı, x_i'' i'inci çözümün son güncellenmiş hâlini, x_{r_1} popülasyondan rastgele seçilen bir çözümü temsil eder. Popülasyon güncellendikten sonra her çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. TLBO algoritmasının temel adımları Algoritma 3.10 ile gösterilmektedir.

Algoritma 3.10: TLBO algoritmanın temel adımları

```
Kontrol parametrelerinin ayarlanması (T, S)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.35) ile çözümlerin ortalamasının hesaplanması
    En iyi çözümün (öğretmenin) seçilmesi
    Eşitlik (3.36) ile çözümlerin güncellenmesi
    Eşitlik (3.37) ile çözümlerin güncellenmesi
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması
```

3.3.3. Matematik İlhamlı Algoritmalar

3.3.3.1. Stokastik Fraktal Arama

Stokastik fraktal arama (stochastic fractal search, SFS) 2015 yılında Salimi tarafından geliştirilen bir metasezgisel algoritmadır [61]. Fraktalların rastlantısal difüzyon davranışları ve doğadaki rastlantısal büyüme süreçlerinden ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Popülasyondaki her bir çözüm öncelikle bir difüzyon sürecinden geçer. Bu süreçte iki farklı Gauss yürüyüşünden biri ile yeni aday çözümler üretilir. Bu aday çözümlerin sayısı maksimum difüzyon sayısı (n_{dif}) adı verilen bir kontrol parametresi ile belirlenir. Rastgele bir sayı üretilir ve bu sayının yürüme oranı ($walk$) adı verilen bir kontrol parametresiyle olan ilişkisine göre bu iki yürüyüşten biri kullanılır. Bu difüzyon güncellemesi Eşitlik (3.38)'de gösterilmektedir.

$$x_{i_k} = \begin{cases} \mathcal{N}(|\hat{x}|, \sigma) + (r\hat{x} - rx_i), & Eğer\ r < walk\ ise \\ \mathcal{N}(|x_i|, \sigma), & aksi\ taktirde \end{cases} \tag{3.38}$$

$$\sigma = \left| \frac{\log(t)}{t} (x_i - \hat{x}) \right| \tag{3.39}$$

Burada, $k \in \{0,1, \dots n_{dif}\}$, $r \sim U(0,1)$ rassal bir sayıyı, x_{i_k} i'inci çözüm için üretilen k'ıncı yeni aday çözümü, x_i i'inci çözümü, $\hat{x}$ popülasyondaki en iyi çözümü, $\mathcal{N}$ Gauss dağılımını (normal dağılım) ve t ise mevcut iterasyonu temsil eder. Mevcut çözüm ile yeni aday çözümlerin uygunluk değerleri hesaplanır ve bunların arasındaki en iyi çözüm difüzyon süreci fonksiyonunun çıktısı olarak verilir. Ardından popülasyon iki güncelleme sürecinden geçer. Birinci güncelleme sürecinde ilk olarak çözümler Eşitlik (3.40)'e göre sıralanır.

$$\delta_i = \frac{rank(x_i)}{S} \tag{3.40}$$

Burada, δ_i i'inci çözümün seçilme olasılığı ve $rank(x_i)$ ise i'inci çözümün uygunluk değerine göre sıralamadaki konumunu temsil eder. Her bir çözümün her bir boyutu için

bir rassal sayı belirlenir. Bu sayı i'inci çözümün seçilme olasılığından büyükse Eşitlik (3.41) kullanılarak çözüm güncellenir. Bu sayı i'inci çözümün seçilme olasılığından küçükse mevcut çözüm değişmez.

$$\left(x_i^j\right)' = x_{r_1}^j - r\left(x_{r_2}^j - x_i^j\right) \tag{3.41}$$

Burada, $r_1, r_2 \neq i$, $r \sim U(0,1)$ rassal bir sayıyı, x_i^j i'inci çözümün j'inci parametresini, $\left(x_i^j\right)'$ i'inci çözümün j'inci parametresinin birinci güncelleme sürecinde güncellenmiş hâlini, $x_{r_1}^j$ ve $x_{r_2}^j$ popülasyondan seçilen rastgele çözümleri temsil eder. İkinci güncelleme sürecinde ise birinci güncelleme aşamasında elde edilen tüm çözümler yine Eşitlik (3.40)'a göre sıralanır. Her bir çözüm için tekrar bir rastgele sayı belirlenir. Bu sayı i'inci çözümün olasılıksal değerinden büyükse Eşitlik (3.42) kullanılarak çözüm güncellenir. Bu sayı i'inci çözümün olasılıksal değerinden küçükse mevcut çözüm değişmez.

$$x_i'' = \begin{cases} x_i' - r\left(x_{r_1} - \hat{x}\right), & Eğer\ r \leq 0.5\ ise \\ x_i' + r\left(x_{r_1} - x_{r_2}\right), & aksi\ taktirde \end{cases} \tag{3.42}$$

Burada, $r_1, r_2 \neq i$, $r \sim U(0,1)$ rassal bir sayıyı, x_i'' i'inci çözümün ikinci güncelleme sürecinde güncellenmiş hâlini, x_i' i'inci çözümün birinci güncelleme sürecinden sonraki hâlini, x_{r_1} ve x_{r_2} birinci güncelleme sürecinden sonra popülasyondan seçilen rastgele çözümleri temsil eder. Popülasyon güncellendikten sonra her çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. SFS algoritmasının temel adımları Algoritma 3.11 ile gösterilmektedir.

Algoritma 3.11: SFS algoritmanın temel adımları

Kontrol parametrelerinin ayarlanması ($T, S, n_{dif}, walk$)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.38) ile çözümlerin difüzyon sürecine tabi tutulması
    Eşitlik (3.40) ile her çözümün olasılıksal değerinin hesaplanması
    Sırasıyla Eşitlik (3.41) ve (3.42) ile çözümlerin güncellenmesi
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması

3.3.3.2. Sinüs Kosinüs Algoritması

Sinüs kosinüs algoritması (sine cosine algorithm, SCA) 2016 yılında Mirjalili tarafından geliştirilen bir metasezgisel algoritmadır [62]. Matematikteki sinüs ve kosinüs trigonometrik fonksiyonlarından ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yineleme süreci başladığında, r_1 parametresi Eşitlik (3.43) kullanılarak hesaplanır.

$$r_1 = a - t\frac{a}{T} \tag{3.43}$$

Burada, a sabit bir kontrol parametresini ve t mevcut iterasyonu temsil eder. Ardından çözümler Eşitlik (3.44) kullanılarak güncellenir.

$$x_i' = \begin{cases} x_i + r_1 sin(r_2)|r_3\hat{x} - x_i|, & Eğer\ r_4 < 0.5\ ise \\ x_i + r_1 cos(r_2)|r_3\hat{x} - x_i|, & aksi\ taktirde \end{cases} \tag{3.44}$$

Burada, $r_2 \sim U(0,2\pi)$, $r_3 \sim U(0,2)$, $r_4 \sim U(0,1)$ rassal sayıları, x_i i'inci çözümü, x_i' i'inci çözümün güncellenmiş hâlini, $\hat{x}$ popülasyondaki en iyi çözümü temsi eder Popülasyon güncellendikten sonra her çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yinelemeli süreç maksimum iterasyon sayısına (T) erişinceye kadar devam eder ve en iyi çözüm raporlanır. SCA algoritmasının temel adımları Algoritma 3.12 ile gösterilmektedir.

Algoritma 3.12: SCA algoritmanın temel adımları

Kontrol parametrelerinin ayarlanması (M, S, a)
Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
Popülasyonun amaç fonksiyonunda değerlendirilmesi
En iyi çözümün tespit edilmesi
while (maksimum iterasyon sayısına erişinceye kadar)
    Eşitlik (3.43) ile r_1'in güncellenmesi
    Eşitlik (3.44) kullanılarak çözümlerin güncellenmesi
    Popülasyonun amaç fonksiyonunda değerlendirilmesi
    En iyi çözümün tespit edilmesi
end while
En iyi çözümün ve maliyetin raporlanması

3.3.3.3. Aritmetik Optimizasyon Algoritması

Aritmetik optimizasyon algoritması (arithmetic optimization algorithm, AOA) 2021 yılında Abualigah ve arkadaşları tarafından geliştirilen bir metasezgisel algoritmadır [63]. Matematiksel hesaplamaların temelini oluşturan aritmetik işlemlerin dağılım özelliklerinden ilham alınmıştır. Algoritmanın çalışma prensibi şu şekilde açıklanabilir: İlk olarak popülasyon Eşitlik (3.18) kullanılarak rastgele üretilir. Bu popülasyon üretildikten sonra tüm çözümler amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yineleme süreci başladığında ise matematik optimizasyon hızlandırıcı (math optimizer accelerated, MOA) ve matematik optimizasyon olasılığı (math optimizer probability, MOP) olmak üzere iki fonksiyon sırasıyla Eşitlik (3.45) ve (3.46) kullanılarak hesaplanır.

$$MOA = MOA_l + t\left(\frac{MOA_h - MOA_l}{T}\right) \tag{3.45}$$

$$MOP = 1 - \frac{t^{\frac{1}{\alpha}}}{T^{\frac{1}{\alpha}}} \tag{3.46}$$

Burada, MOA_l ve MOA_h değerleri MOA'nın sınır değerlerini, t mevcut iterasyonu, T maksimum iterasyon sayısını ve α kullanım doğruluğunu tanımlayan bir kontrol parametresini temsil eder. Her bir çözüm için bir rastgele sayı $r_1 \sim U(0,1)$ üretilir ve bu sayı mevcut iterasyondaki MOA'dan büyükse çözümler Eşitlik (3.47) kullanılarak güncellenir. Eğer bu sayı MOA'dan küçükse o zaman da çözümler Eşitlik (3.48) kullanılarak güncellenir.

$$\left(x_i^j\right)' = \begin{cases} \hat{x}^j \div (MOP + \epsilon)\left(\left(x_h^j - x_l^j\right)\mu + x_l^j\right), & Eğer\ r_2 < 0.5\ ise \\ \hat{x}^j \times MOP\left(\left(x_h^j - x_l^j\right)\mu + x_l^j\right), & aksi\ taktirde \end{cases} \quad (3.47)$$

$$\left(x_i^j\right)' = \begin{cases} \hat{x}^j - MOP\left(\left(x_h^j - x_l^j\right)\mu + x_l^j\right), & Eğer\ r_3 < 0.5\ ise \\ \hat{x}^j + MOP\left(\left(x_h^j - x_l^j\right)\mu + x_l^j\right), & aksi\ taktirde \end{cases} \quad (3.48)$$

Burada, $r_2, r_3 \sim U(0,1)$ rassal bir sayıyı, $\left(x_i^j\right)'$ i'inci çözümün j'inci parametresinin güncellenmiş hâlini, $\hat{x}^j$ popülasyondaki en iyi çözümün j'inci parametresini, ϵ küçük bir sayıyı, x_l^j ve x_h^j j'inci parametrenin sınır değerlerini, μ arama sürecini ayarlayan bir kontrol parametresini temsil eder. Popülasyon güncellendikten sonra her çözüm amaç fonksiyonunda değerlendirilir ve uygunluk değerleri hesaplanır. Yinelemeli süreç maksimum iterasyon sayısına erişinceye kadar devam eder ve en iyi çözüm raporlanır. AOA algoritmasının temel adımları Algoritma 3.13 ile gösterilmektedir.

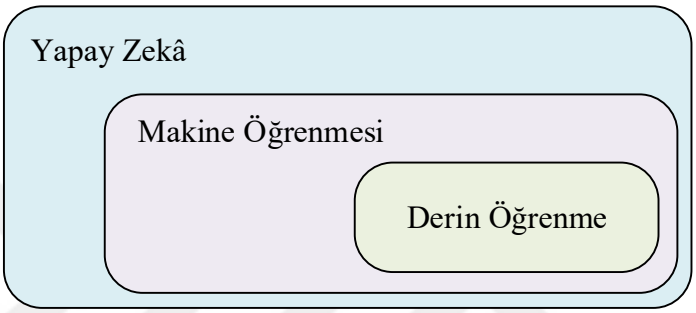
Algoritma 3.13: AOA algoritmanın temel adımları

1: Kontrol parametrelerinin ayarlanması ($T, S, MOA_l, MOA_h, \alpha, \mu$)
2: Eşitlik (3.18) ile popülasyonun rastgele üretilmesi
3: Popülasyonun amaç fonksiyonunda değerlendirilmesi
4: En iyi çözümün tespit edilmesi
5: **while** (maksimum iterasyon sayısına erişinceye kadar)
6: Eşitlik (3.45) ve (3.46) ile MOA ve MOP değerlerinin güncellenmesi
7: r_1 sayısının üretilmesi
8: r_1 ve MOA'ya göre Eşitlik (3.47) veya (3.48) ile çözümlerin güncellenmesi
10: Popülasyonun amaç fonksiyonunda değerlendirilmesi
11: En iyi çözümün tespit edilmesi
12: **end while**
13: En iyi çözümün ve maliyetin raporlanması

3.4. Makine Öğrenmesi

Makine öğrenmesi, bilgisayarların verilerden desenler öğrenerek tahminler yapmasını sağlayan bir yapay zekâ dalıdır [64]. Bu süreçte, algoritmalar belirli kurallar çerçevesinde eğitilir ve zamanla doğruluklarını artırır. Makine öğrenmesi yöntemleri genellikle ikiye ayrılır: gözetimli ve gözetimsiz öğrenme. Gözetimli öğrenme etiketli verilerle modelin eğitildiği bir yaklaşımken; gözetimsiz öğrenme ise verilerdeki gizli desenleri keşfetmeye odaklanır. Örneğin, bir makine öğrenmesi modeli, el yazısı rakamları tanımak için önce insan tarafından belirlenen kenar, eğim gibi özellikleri kullanarak öğrenme sürecini

tamamlar. Ancak, daha karmaşık problemlerde geleneksel makine öğrenmesi yaklaşımları, özellik mühendisliği ve büyük veri ile başa çıkmada yetersiz kalabilir. Bu noktada derin öğrenme devreye girer. Derin öğrenme, insan beyninin çalışma şeklini taklit eden derin sinir ağlarını kullanarak daha karmaşık veri yapılarını öğrenen bir makine öğrenmesi alt dalıdır. Bu yöntem, birçok katmandan oluşan sinir ağları sayesinde büyük ve karmaşık veri kümeleriyle etkili bir şekilde çalışabilir. Makine öğrenmesi ile derin öğrenme kapsamları Şekil 3.2'de gösterilmektedir.



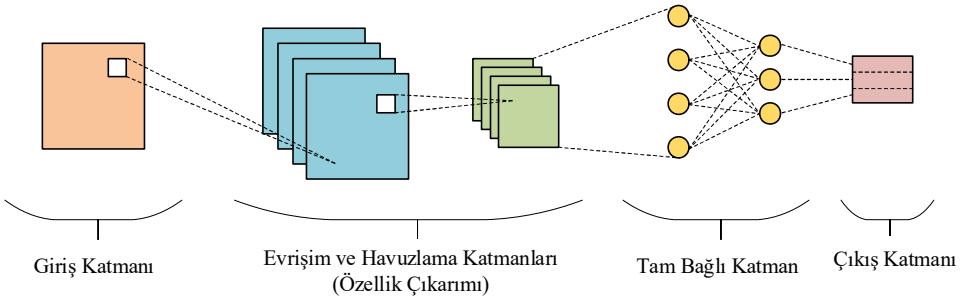
Şekil 3.2. Makine öğrenmesi ile derin öğrenme kapsamları

Geleneksel makine öğrenmesinin aksine, derin öğrenme algoritmaları özellikleri manuel olarak belirlemek yerine, veriden otomatik olarak öğrenir ve kendiliğinden en uygun temsilleri keşfeder. Bu sayede, özellikle büyük ölçekli veri setleriyle çalışırken insan müdahalesine duyulan ihtiyaç azalır ve modeller daha doğru sonuçlar üretebilir. Örneğin, bir görüntü tanıma görevinde geleneksel yöntemlerde öncelikle görüntüden kenar, doku veya renk gibi belirli özelliklerin insan tarafından çıkarılması gerekirken, derin öğrenme modelleri ham piksel verisini doğrudan alarak, katmanlar boyunca giderek soyutlaşan özellikler öğrenir ve nihai sınıflandırmayı gerçekleştirir. İlk katmanlar temel kenar ve dokuları öğrenirken, daha derin katmanlar nesneleri, yüzleri veya daha karmaşık yapıları tanıyabilir. Benzer şekilde, doğal dil işleme alanında, derin öğrenme algoritmaları ham metin verisini analiz ederek kelimeler arasındaki anlam ilişkilerini otomatik olarak öğrenir ve çeviri, metin özetleme veya duygu analizi gibi görevleri başarıyla yerine getirebilir. Derin öğrenmenin bu yetenekleri, büyük veri çağında daha başarılı sonuçlar elde edilmesini sağlar ve sağlık, otomotiv, finans, güvenlik gibi birçok alanda devrim niteliğinde uygulamalara kapı aralar. Örneğin, otonom araçlar, çevresindeki nesneleri gerçek zamanlı olarak algılamak ve güvenli sürüş kararları vermek için derin öğrenmeden yararlanır. Aynı şekilde, tıbbi görüntüleme sistemleri, hastalıkları teşhis etmek için

röntgen ve MR görüntülerini analiz edebilir. Derin öğrenmenin sağladığı bu otomatik ve ölçeklenebilir çözümler, birçok endüstride insan müdahalesini azaltarak hız, doğruluk ve verimliliği artırmaktadır. Mobil robotların yol planlamasında makine öğrenmesi yöntemleri arasında evrişimli sinir ağları ve takviyeli öğrenme giderek daha fazla tercih edilmektedir [62]. Evrişimli sinir ağları, çevresel algılamada etkili özellik çıkarımı sağlarken, takviyeli öğrenme ise robotun dinamik ortamlarda en uygun yolu keşfetmesine ve çevresel değişikliklere uyum sağlamasına yardımcı olmaktadır.

3.4.1. Evrişimli Sinir Ağları

Evrişimli sinir ağı (convolutional neural network, CNN), özellikle görüntü ve video analizinde kullanılan, derin öğrenme tabanlı bir yapay sinir ağı türüdür. İnsan beyninin görme korteksinden esinlenerek geliştirilen CNN'ler karmaşık görsel desenleri ve özellikleri otomatik olarak öğrenme yeteneğine sahiptir. Bu sayede görüntü sınıflandırma, nesne tanıma, yüz tanıma gibi birçok alanda üstün performans gösterirler. CNN'lerin temelinde evrişim (convolution) ve havuzlama (pooling) gibi özel katmanlar bulunur. Bu katmanlar, görüntüdeki önemli özellikleri vurgulayarak, ağın daha etkili öğrenmesini sağlar. Şekil 3.3'te tipik bir CNN yapısı gösterilmektedir.



Şekil 3.3. Tipik bir evrişimli sinir ağı yapısı

Giriş katmanında ağın aldığı ham veri olan görüntü işlenmeye başlar. Görüntü, piksel değerlerini taşıyan bir matris olarak giriş katmanına gelir. Bu katmanda, ağın işleyeceği temel veriler belirlenir ve bu veri sonraki katmanlar için temel oluşturur. Özellik çıkarımı CNN'nin temelini oluşturan evrişim ve havuzlama işlemlerinin yapıldığı yerdir. Bu katman görüntüdeki kenarlar, köşeler, dokular gibi temel özellikleri çıkarır. Evrişim işlemi, görüntü üzerinde küçük filtreler gezdirerek özellik haritaları oluştururken, havuzlama işlemi bu haritaların boyutunu küçülterek daha soyut temsiller elde eder. Tam

bağlı katman (fully connected layer), özellik çıkarımından elde edilen soyut özellikleri kullanarak, görüntünün ne olduğunu tahmin etmeye çalışır. Bu katman klasik bir yapay sinir ağı gibi çalışır ve her bir nöronun bir önceki katmandaki tüm nöronlara bağlı olduğu bir yapıya sahiptir. Çıkış katmanı ağın tahmin sonucunu verir. Genellikle bir olasılık dağılımı şeklinde sonuçlar üretir. Örneğin, bir görüntü sınıflandırma probleminde, bu katman her bir sınıf için bir olasılık değeri vererek görüntünün hangi sınıfa ait olduğunu tahmin eder. CNN'ler 1980'lerden itibaren gelişmeye başlamış ve zaman içinde birçok farklı mimari ortaya çıkmıştır. Bu bölümde CNN'lerin gelişimine önemli katkılarda bulunmuş öne çıkan bazı modeller ele alınacaktır.

3.4.1.1. LeNet-5

LeNet-5, 1998 yılında LeCun ve arkadaşları tarafından geliştirilen bir derin öğrenme modelidir [66]. Bu model derin öğrenme alanındaki ilk başarılı CNN örneği olarak, görüntü tanıma görevlerinde kullanılan ilk derin ağ yapılarından birisidir. Modelin mimarisi şu şekilde özetlenebilir: İlk katman, 5 x 5 boyutunda 6 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve tanh aktivasyon fonksiyonu kullanılır. Ayrıca 2 x 2 boyutunda ve 2 adım kaydırma ile ortalama havuzlama (average pooling) işlemi yapılır. Sonraki katman, 5 x 5 boyutunda 16 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve tanh aktivasyon fonksiyonu kullanılır. Ayırca yine 2 x 2 boyutunda ve 2 adım kaydırma ile ortalama havuzlama işlemi yapılır. Sonraki katman, 5 x 5 boyutunda 120 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve tanh aktivasyon fonksiyonu kullanılır. Sonraki katman, tam bağlı katmandır ve tanh aktivasyon fonksiyonunu kullanır. Çıkış katmanına bağlanan son katman ise softmax aktivasyon fonksiyonunu kullanan bir tam bağlı katmandır.

3.4.1.2. AlexNet

AlexNet, 2012 yılında Krizhevsky ve arkadaşları tarafından tarafından geliştirilen bir derin öğrenme modelidir [67]. Bu model, grafik işlemci hızlandırmasını kullanarak büyük veri setleri üzerinde eğitim yapmış ve doğrultulmuş doğrusal birim (rectified linear unit, ReLU) aktivasyon fonksiyonunu etkin bir şekilde kullanarak eğitim sürecini hızlandırmıştır. Modelin mimarisi şu şekilde özetlenebilir: İlk katman, 11 x 11 boyutunda 96 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 4 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 3 x 3 boyutunda ve 2 adım kaydırma ile

maksimum havuzlama işlemi yapılır. Sonraki katman, 5 x 5 boyutunda 256 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 3 x 3 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Sonraki iki katman, 3 x 3 boyutunda 384 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Sonraki katman, 3 x 3 boyutunda 256 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Sonraki üç katmanlar tam bağlı katmanlardır, bunların ilk ikisinde ReLU, üçüncüsünde ise softmax aktivasyon fonksiyonu kullanılır. Ayrıca her tam bağlı katmana %50 oranla kullanılan bir dropout katmanı bağlıdır. Son tam bağlı katman ise çıkış katmanına bağlanır.

3.4.1.3. VGG16

VGG16, 2014 yılında Simonyan ve Zisserman tarafından geliştirilen bir derin öğrenme modelidir [68]. Bu modelde katman sayısının derinliği ve 3 x 3 boyutunda evrişim katmanlarının üst üste kullanılmasıyla çok derin bir ağ yapısı oluşturulmuş ve bu sayede ağın derinliği artırılarak yüksek doğruluk elde edilmiştir. Modelin mimarisi şu şekilde özetlenebilir: İlk iki katman, 3 x 3 boyutunda 64 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 2 x 2 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Sonraki iki katman, 3 x 3 boyutunda 128 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 2 x 2 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Sonraki üç katman, 3 x 3 boyutunda 256 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 2 x 2 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Sonraki üç katman, 3 x 3 boyutunda 512 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 2 x 2 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Sonraki üç katman, yine 3 x 3 boyutunda 512 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca yine 2 x 2 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapar. Sonraki iki katman tam bağlı katmanlarıdır ve ReLU aktivasyon fonksiyonu kullanılır. Çıkış katmanına bağlanan son katman ise softmax aktivasyon fonksiyonunu kullanan bir tam bağlı katmandır.

3.4.1.4. GoogleNet

GoogleNet (Inception v1), 2014 yılında Szegedy ve arkadaşları tarafından geliştirilen bir derin öğrenme modelidir [69]. Bu model başlangıç (inception) tanıtarak, aynı katmanda farklı boyutlardaki filtreleri ve havuzlama işlemlerini birleştirmiştir. Ayrıca yardımcı sınıflandırıcı (auxiliary classifier) adı verilen bir başka blok da tanıtılmıştır. Modelin mimarisi şu şekilde özetlenebilir: İlk katman, 7 x 7 boyutunda 64 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 2 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca 3 x 3 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi ve ardından yerel tepki normalizasyonu (local response normalization) yapılır. Sonraki katman, 1 x 1 boyutunda 64 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Sonraki katman, 3 x 3 boyutunda 192 filtre kullanarak evrişim işlemi yapar. Adım uzunluğu 1 olarak ayarlanır ve ReLU aktivasyon fonksiyonu kullanılır. Ayrıca yerel tepki normalizasyonu, ardından 3 x 3 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Daha sonra 9 adet başlangıç bloğu uygulanır ve birleştirilir. Ayrıca 4. ve 7. başlangıç bloklarıyla paralel olarak yardımcı sınıflandırıcı blokları uygulanır. Son başlangıç bloğunun ardından 7 x 7 boyutunda global ortalama havuzlama işlemi (global average pooling) ve %40 oranında dropout katmanı uygulanır. Çıkış katmanına bağlanan son katman ise bir tam bağlı katmandır ve softmax aktivasyon fonksiyonu kullanılır.

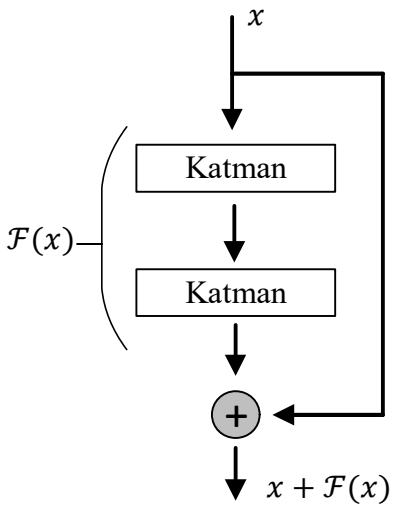
Başlangıç bloğu şu şekilde özetlenebilir: Bloğun kendi yerel girişi 1 x 1 boyutunda 3 adet evrişim ve 3 x 3 boyutunda 1 adet maksimum havuzlama katmanlarına aynı anda gönderilir. 3 adet evrişim katmanından biri doğrudan bloğun yerel çıkışına yönlendirilir. Bir diğeri 3 x 3 boyutunda başka bir evrişim katmanına bağlanır ve bu katmanın çıkışı yerel çıkış noktasına yönlendirilir. Bir diğeri 5 x 5 boyutunda başka bir evrişim katmanına bağlanır ve bu katmanın çıkışı yerel çıkış noktasına yönlendirilir. Maksimum havuzlama katmanı ise 1 x 1 boyutunda başka bir evrişim katmanına bağlanır ve bu katmanın çıkışı yerel çıkış noktasına yönlendirilir. Son olarak yerel çıkışa gönderilen her bir bağlantı birleştirilir. Yardımcı sınıflandırıcı bloğu ise şu şekilde özetlenebilir: Bloğun kendi yerel girişine ilk olarak global ortalama havuzlama işlemi uygulanır. Ardından 1 x 1 boyutunda başka bir evrişim katmanına bağlanır. Sonraki iki katman tam bağlı katmanlarıdır ve ReLU aktivasyon fonksiyonu kullanılır. Bloğun kendi yerel çıkış katmanında ise softmax aktivasyon fonksiyonu kullanılır.

3.4.1.5. Tam Evrişimli Ağ

Tam evrişimli ağ (fully convolutional networks, FCN), 2015 yılında Long ve arkadaşları tarafından geliştirilen bir derin öğrenme modelidir [70]. Bu model tamamen evrişim katmanlar kullanarak görüntü segmentasyonu gibi görevlerde piksel düzeyinde tahminler yapar ve tam bağlı katmanları ortadan kaldırarak modelin daha verimli hale getirir. Modelin mimarisi şu şekilde özetlenebilir: FCN'nin temel yapı taşları evrişim, havuzlama ve yukarı örnekleme (upsampling) katmanlarıdır. Evrişim katmanları, belirli bir boyutta filtreler kullanarak girişten özellik çıkarır ve her konvolüsyon işlemi için belirlenen adım uzunluğuna göre çıktıyı oluşturur. Bu katmanlarda genellikle ReLU aktivasyon fonksiyonu kullanılır. FCN, temel CNN'lerden farklı olarak, tam bağlantılı katmanlar içermez; bunun yerine 1 x 1 evrişim katmanları kullanarak sınıflandırma veya regresyon işlemlerini doğrudan uzamsal boyutta gerçekleştirir. Modelin en önemli bileşenlerinden biri yukarı örnekleme katmanlarıdır, çünkü bunlar havuzlama katmanları tarafından küçültülen uzamsal boyutları tekrar büyüterek çıktıyı orijinal giriş boyutuna yaklaştırır. Bu süreç, özellikle görüntü segmentasyonu gibi piksel tabanlı tahminler gerektiren görevlerde kritik öneme sahiptir. FCN, klasik sınıflandırma ağlarının tam bağlı katmanlarını kaldırarak herhangi bir boyuttaki girdilerle çalışabilir ve piksellerin konum bilgilerini koruyarak çıktı üretebilir. Son olarak, FCN'nin çıkış katmanı genellikle softmax aktivasyon fonksiyonuyla tamamlanır ve her pikselin belirli bir sınıfa ait olma olasılığını hesaplar.

3.4.1.6. ResNet

ResNet, 2015 yılında He ve arkadaşları tarafından geliştirilen bir derin öğrenme modelidir [71]. Bu model artık (residual) bağlantılar sayesinde derin ağların eğitimini kolaylaştırarak kaybolan gradyan problemini etkili bir şekilde çözmüştür. Modelin mimarisi şu şekilde özetlenebilir: İlk katman, 7 x 7 boyutunda 64 filtreli bir evrişim katmanıdır; bu katman 2 adım boyutu kullanarak girişi işler. Bu katmanda ReLU aktivasyon fonksiyonu kullanılır ve toplu normalizasyon gerçekleştirilir. Ayrıca 3 x 3 boyutunda ve 2 adım kaydırma ile maksimum havuzlama işlemi yapılır. Ağın geri kalanı, residual bloklar olarak organize edilmiştir. Tipik bir artık blok Şekil 3.4'te gösterilmektedir.

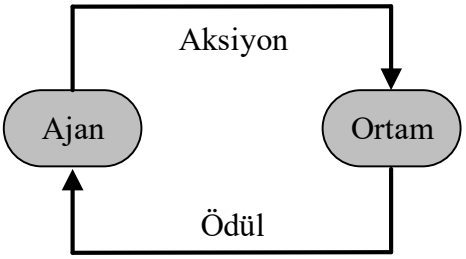


Şekil 3.4. Temel bir artık blok

Her artık blok, iki adet 3 x 3 evrişim katmanı, toplu normalizasyon ve ReLU aktivasyonunu içerir. Daha derin katmanlara geçerken, her yeni artık bloğun ilk evrişim katmanı filtre sayısını iki katına çıkarır ve boyut küçültmek için adım boyutu 2 olarak belirlenir. Boyut değişen bloklarda kısa yol bağlantısı için 1 x 1 evrişim katmanı kullanılır. Bunların ardından, global ortalama havuzlama katmanı ile tüm uzaysal özellikler sıkıştırılarak sabit boyutlu bir vektör elde edilir. Bunu, 1000 çıkışlı bir tam bağlı katman ve ardından softmax katmanı takip eder, böylece sınıflandırma yapılır.

3.4.2. Takviyeli Öğrenme

Takviyeli öğrenme (reinforcement learning), bir ajan (agent) ile ortam (environment) arasındaki etkileşimi temel alan bir makine öğrenme yöntemidir. Ajan, ortama etkileşimde bulunarak belirli hedeflere ulaşmayı amaçlar. Ortamdan aldığı ödülleri kullanarak kendi politikasını (hangi duruma hangi aksiyonun uygun olduğunu belirleyen strateji) optimize eder. Ajanın amacı, zaman içinde aldığı ödülleri maksimize etmek için en iyi aksiyonları seçmektir. Bu süreç bir durum (state), bir aksiyon (action) ve alınan bir ödül (reward) döngüsüyle gerçekleşir. Şekil 3.5'te tipik bir takviyeli öğrenme yapısı gösterilmektedir.



Şekil 3.5. Tipik bir takviyeli öğrenme yapısı

Takviyeli öğrenmede ajan, keşif ve kullanım arasında bir denge kurmak zorundadır. Keşif ajanının yeni aksiyonları denemesi ve ortamın dinamiklerini öğrenmesi anlamına gelirken, kullanım ise ajanın şu ana kadar öğrendiği en iyi aksiyonları seçmesidir. Bu iki strateji arasında bir denge kurmak önemlidir; aksi halde ajan yalnızca bildiği aksiyonları seçerek yeni fırsatlar kaçırabilir veya gereksiz keşiflerde bulunarak verimsiz olabilir. Ayrıca, RL transfer öğrenmeyi de içerir, yani ajan bir görevde öğrendiği bilgileri başka bir göreve aktarabilir. Örneğin, bir ajan bir ortamda başarılı bir şekilde öğrenmeyi tamamladıktan sonra, başka bir benzer ortamda ajanın daha hızlı öğrenebilmesi sağlanabilir. Transfer öğrenme, takviyeli öğrenmenin hesaplama yükünü azaltabilir ve daha genel stratejiler geliştirilmesine yardımcı olabilir. Bu bölümde RL'deki öne çıkan bazı modeller ele alınacaktır.

3.4.2.1. Aktör-Kritik Algoritması

Aktör-kritik algoritması (actor-critic, AC), ilk olarak 1984 yılında Sutton tarafından geliştirilen bir RL modelidir [72]. Bu model politika tabanlı (aktör) ve değer tabanlı (kritik) bileşenleri birleştirerek, hem politika öğrenmesini hem de değer fonksiyonu tahminini paralel bir şekilde optimize eder. Algoritma, iki ana bileşenden oluşur: Aktör ve kritik. Aktör, mevcut politika kullanılarak aksiyonlar seçerken, kritik, bu aksiyonların değerini tahmin eder ve aktöre geri bildirim sağlar. Algoritmanın çalışma prensibi, bir ajanın bir ortamda sürekli olarak etkileşime girip ödül toplaması sürecine dayanır. Ajan, çevreden aldığı durum bilgisiyle, belirli bir politika (π) kullanarak bir aksiyon seçer. Kritik, seçilen aksiyonun değerini tahmin eder ve bu tahmin bir avantaj fonksiyonu (A_t) ile ilişkilendirilir. Aktör bu avantaj fonksiyonunu kullanarak politikasını günceller, böylece daha yüksek ödül getiren eylemleri tercih eder. İlk olarak aksiyon seçilirerek ona

yönelik ödül ve bir sonraki durum gözlemlenir. Ardından avantaj fonksiyonu Eşitlik (3.49) kullanılarak güncellenir.

$$A_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \tag{3.49}$$

Burada, r_t mevcut ödülü, γ indirim faktörünü, $V_\theta(s_{t+1})$ kritik tarafından tahmin edilen bir sonraki durumun değerini ve $V_\theta(s_t)$ ise değer fonksiyonunu temsil eder. Ardından aktör ve kritik Eşitlik (3.50) ve (3.51) kullanılarak güncellenir.

$$\theta_a^{t+1} = \theta_a^t + \alpha_a \nabla_\theta \log \pi_\theta(s_t, a_t)\, A_t \tag{3.50}$$

$$\theta_c^{t+1} = \theta_c^t + \alpha_c \big(r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)\big)^2 \tag{3.51}$$

Burada, θ_a aktörün parametrelerini, α_a aktörün öğrenme oranını, $\pi_\theta(s_t, a_t)$ mevcut durum s_t ve mevcut aksiyon a_t için ilgili politikayı, θ_c kritiğin parametrelerini ve a_c kritiğin öğrenme oranını temsil eder. Bu süreç durduma kriteri sağlanana kadar devam eder. AC'nin temel adımları Algoritma 3.14 ile gösterilmektedir.

Algoritma 3.14: AC'nin temel adımları

```
Hiperparametrelerin ayarlanması (α, β, γ)
Aktör politikasının ve kritik değer fonksiyonunun değerlerinin başlatılması
while (maksimum epoch sayısına erişinceye kadar)
    s durumunun başlatılması
    while (maksimum adım sayısına erişinceye kadar)
        Aktör politikasına göre bir aksiyonun seçilmesi
        Seçilen aksiyonu gerçekleştirilmesi
        Ödül ve bir sonraki durumun gözlemlenmesi
        Eşitlik (3.49) ile avantaj fonksiyonunun güncellenmesi
        Eşitlik (3.50) ile aktörün güncellenmesi
        Eşitlik (3.51) ile kritiğin güncellenmesi
    end while
end while
Optimal politikanın raporlanması
```

3.4.2.2. Q-Öğrenme

Q-öğrenme (Q-learning, QL), 1989 yılında Watkins tarafından geliştirilen bir RL modelidir [73]. QL çevrenin dinamiklerini bilmeden, sadece gözlemler ve ödüllerle karar almayı öğrenir. QL temel olarak Q değerleri adı verilen bir yapı kullanır. Bu değerler bir durum ve aksiyon çiftinin değerini temsil eder. Bu değer, o durumda belirtilen aksiyonun alınmasının gelecekteki ödülleri ne kadar artıracağına dair bir tahmindir. Ajan bu Q

değerlerini zamanla güncelleyerek en iyi politikasını öğrenir. Modelin çalışma prensibi şu şekilde özetlenebilir: Başlangıçta tüm Q değerleri sıfır olarak ayarlanır veya küçük rastgele değerlerle başlatılabilir. Ajan herhangi bir durumdayken bir aksiyon seçer. Bu seçim genellikle epsilon açgözlü stratejisi (ε-greedy) ile yapılır, yani ajan rastgele bir aksiyon seçme olasılığına sahip olup, çoğunlukla en yüksek Q değerine sahip aksiyonu tercih eder. Seçilen aksiyon uygulanır, yeni bir durum gözlemlenir ve bu yeni durumda alınan ödül elde edilir. Ajan, Q değerini Eşitlik (3.52)'de gösterilen QL güncelleme kuralına göre günceller:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right) \tag{3.52}$$

Burada, $Q(s_t, a_t)$ mevcut durum ve aksiyon için mevcut Q değerini, α öğrenme oranını, r_t alınan ödülü, γ indirim faktörünü, s_{t+1} bir sonraki durumu ve a' ise yeni durumda yapılabilecek tüm olası aksiyonları temsil eder. Ajan, bu işlemi çevreyi keşfederek ve ödülleri öğrenerek devam eder. Sonunda, Q değerleri stabil hale gelir ve ajan en yüksek Q değerine sahip aksiyonları seçerek optimal politikayı öğrenmiş olur. QL'nin temel adımları Algoritma 3.15 ile gösterilmektedir.

Algoritma 3.15: QL'nin temel adımları

```
Hiperparametrelerin ayarlanması (α, γ)
Tüm durum ve aksiyonlar için Q değerlerinin başlatılması
while (maksimum epoch sayısına erişinceye kadar)
    s durumunun başlatılması
    while (maksimum adım sayısına erişinceye kadar)
        Epsilon açgözlü stratejisi ile bir aksiyonun seçilmesi
        Seçilen aksiyonun gerçekleştirilmesi
        Ödül ve bir sonraki durumun gözlemlenmesi
        Eşitlik (3.52) ile Q değerinin güncellenmesi
        Mevcut durumun bir sonraki durum olarak güncellenmesi
    end while
end while
Optimal politikanın raporlanması
```

3.4.2.3. Derin Deterministik Politika Gradyanı

Derin Deterministik Politika Gradyanı (deep deterministic policy gradient, DDPG), 2015 yılında Lillicrap ve arkadaşları tarafından geliştirilen bir RL modelidir [74]. Bu model sürekli eylem alanlarında politika gradyanı yöntemlerini derin öğrenme ile birleştirir.

Algoritma, aktör ve kritik ağları olmak üzere iki ana bileşene sahiptir. Aktör ağı verilen bir durum için en iyi aksiyonu üretirken, kritik ağ verilen bir durum ve aksiyon çiftinin ne kadar iyi olduğunu değerlendirir. DDPG, öğrenme sürecini daha stabil hale getirmek için deney tekrarı (experience replay) ve hedef ağlar (target networks) gibi ek teknikler kullanır. Modelin çalışma prensibi şu şekilde özetlenebilir: Algoritmanın ilk adımında, kritik ağı ve aktör ağı rastgele başlatılır. Bunlar ajan tarafından kullanılan temel öğrenme modelleridir. Ayrıca, bu ağların birer hedef kopyaları oluşturulur. Hedef ağlar, öğrenme sürecindeki dalgalanmaları azaltarak algoritmanın daha stabil çalışmasını sağlar. Ardından, deney tekrar havuzu oluşturulur. Bu havuz, ajanın çevrede deneyimlediği durum-geçişlerini saklar ve rastgele örnekleme yaparak öğrenmeyi daha etkili hale getirir. DDPG, her zaman adımında aksiyon keşfi için rastgele bir gürültü başlatır. Ajan, çevreden ilk durumu alır ve süreç başlar. Her zaman adımında, ajan Eşitlik (3.53)'te gösterildiği gibi mevcut duruma göre bir aksiyon seçer. Bu aksiyon, aktör ağı tarafından belirlenir ancak keşfi artırmak için bir miktar rastgele gürültü eklenir.

$$a_t = \mu(s_t|\theta^\mu) + N_t \tag{3.53}$$

Burada, a_t t anındaki aksiyonu, $\mu(s_t|\theta^\mu)$ aktör ağını, N_t ise t anındaki keşfi artırmak için kullanılan rastgele gürültüyü temsil eder. Bu gürültü genellikle zamanla korelasyonlu Ornstein-Uhlenbeck süreci olarak modellenir. Seçilen aksiyon çevrede uygulanır ve ajan ödülü (r_t) ve bir sonraki durumu (s_{t+1}) gözlemler. Bu geçiş deney tekrar havuzuna (s_t, a_t, r_t, s_{t+1}) olarak kaydedilir. Deney tekrar havuzundan rastgele bir minibatch (n_m adet geçiş) örneklenir. Deney tekrar havuzundan örneklenen her minibatch için, hedef Q değeri Bellman denklemine göre Eşitlik (3.54) kullanılarak hesaplanır.

$$y_i = r_{t_i} + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \tag{3.54}$$

Burada, r_{t_i} i'inci minibatch örneğinde elde edilen ödülü, γ indirim faktörünü, μ' ve Q' ise hedef ağları temsil eder. Kritik ağ, bu hedef Q değerine yaklaşmak amacıyla Eşitlik (3.55)'te gösterildiği gibi güncellenir.

$$\theta^Q = \theta^Q - \alpha_Q \nabla_{\theta^Q} L(\theta^Q) \tag{3.55}$$

$$L(\theta^Q) = \frac{1}{n_m}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2 \tag{3.56}$$

Burada, $Q(s_i, a_i|\theta^Q)$ kritik ağı temsil eder. Aktör ağı Eşitlik (3.56)'da gösterildiği gibi deterministik politika gradyan yöntemiyle güncellenir:

$$\nabla_\theta J \approx \frac{1}{n_m}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \tag{3.57}$$

Ardından, hedef ağlar yumuşak güncellenme ile Eşitlik (3.57) ve (3.58)'de gösterildiği gibi güncellenir.

$$\theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{3.58}$$

$$\theta^{\mu'} = \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{3.59}$$

Burada, τ yumuşak güncellenme oranını temsil eder. Bu süreç, belirlenen zaman adımı veya bölüm sayısı tamamlanana kadar tekrar edilir. DDPG'nin temel adımları Algoritma 3.16 ile gösterilmektedir.

Algoritma 3.16: DDPG'nin temel adımları

```
Hiperparametrelerin ayarlanması (τ, γ, n_m)
Aktör ve kritik ağlarının başlatılması
Hedef ağların başlatılması
Deney tekrar havuzunun oluşturulması
while (maksimum epoch sayısına erişinceye kadar)
    s durumunun başlatılması
    while (maksimum adım sayısına erişinceye kadar)
        Eşitlik (3.53) ile aksiyonun seçilmesi
        Seçilen aksiyonun gerçekleştirilmesi
        Ödül ve bir sonraki durumun gözlemlenmesi
        Bu deneyimin deney havuzuna eklenmesi
        Havuzdan rastgele bir minibatch'in örneklenmesi
        Eşitlik (3.54) ile hedef Q değeri hesaplanması
        Eşitlik (3.55) ile kritik ağın güncellenmesi
        Eşitlik (3.56) ile aktör ağının güncellenmesi
        Eşitlik (3.57) ve (3.58) ile hedef ağların güncellenmesi
    end while
end while
Optimal politikanın raporlanması
```

4. BÖLÜM

BULGULAR

4.1. Giriş

Bu tez çalışması kapsamında mobil robotların yol planlama sorunlarına yönelik çözümler geliştirmek amacıyla dört farklı simülasyon çalışması ve bir gerçek zamanlı çalışma gerçekleştirilmiştir. Bu çalışmalar farklı yol tipleri, engel yapıları ve ortam özellikleri dikkate alınarak yol planlama yöntemlerini iyileştirerek yeni yaklaşımlar geliştirmeye odaklanmıştır. Bu tez çalışması kapsamında gerçekleştirilen dört simülasyon çalışmasında iki boyutlu yol planlama problemine yönelik farklı metasezgisel ve makine öğrenmesi tabanlı yaklaşımlar önerilmiş ve performansları test edilmiştir. Birinci çalışmada tek bir mobil robotun ızgara ortamında küresel yol planlaması için iyileştirilmiş bir yapay arı koloni algoritması önerilmiştir. İkinci çalışmada tek bir mobil robotun sürekli uzay ortamında küresel yol planlaması için metasezgisel ve kümeleme algoritmalarının hibrit bir modeli önerilmiştir. Üçüncü çalışmada çok sayıda mobil robotun sürekli uzay ortamında yerel yol planlaması için iyileştirilmiş bir sinüs kosinüs algoritması önerilmiştir. Dördüncü çalışmada ise tek ve çok robotlu sistemlerinin ızgara ortamında küresel yol planlaması için ResNet tabanlı bir model önerilmiştir. Gerçek zamanlı çalışmada ise simülasyon çalışmalarını desteklemek amacıyla mobil robotun sürekli uzay ortamında küresel yol planlaması ve yol takip kontrolü gerçekleştirilmiştir. Bu çalışmaların özellikleri Tablo 4.1'de özetlenmiştir. Bu tez kapsamda ele alınan çalışmalar mobil robotların yol planlama süreçlerinde etkinliği artırmayı ve olası zorluklara karşı çözüm üretmeyi hedeflemektedir.

Tablo 4.1. Bu tez kapsamındaki çalışmalarının özellikleri

#	Çalışma Tipi	Programlama Dili	Yöntem Sınıfı	Robot Sayısı	Planlama Kapsamı	Ortam Tipi
4.2	Simülasyon	MATLAB	Metasezgisel	Tek	Küresel	Izgara
4.3	Simülasyon	MATLAB	Hibrit	Tek	Küresel	Sürekli Uzay
4.4	Simülasyon	MATLAB	Metasezgisel	Çok	Yerel	Sürekli Uzay
4.5	Simülasyon	Python	Makine Öğrenmesi	Tek & Çok	Küresel	Izgara
4.6	Gerçek Zamanlı	Python	Metasezgisel / P Kontrol	Tek	Küresel	Sürekli Uzay

4.2. İyileştirilmiş Yapay Arı Kolonisi Algoritmasını Kullanan Etkili Bir Izgara Tabanlı Yol Planlama Yaklaşımı

Yol planlama problemleri robotik ve otonom sistemler için kritik öneme sahiptir. Literatürdeki çeşitli algoritmalar bu problemleri çözmek için etkili yöntemler sağlamış olsa da, daha hızlı, daha doğru ve enerji açısından verimli çözümler geliştirmeye hala ihtiyaç vardır. Bunlar arasında ABC algoritması, yol planlama problemleri için sıklıkla kullanılan sağlam ve güçlü bir metasezgisel optimizasyon algoritması olarak öne çıkmaktadır [75]. Ancak No Free Lunch teoremi doğrultusunda hiçbir optimizasyon algoritmasının tüm problem tiplerinde optimum performans göstermesi beklenemez. Bu makalenin odak noktası yol planlama problemini çözme yeteneğini artırmak için ABC algoritmasını iyileştirmektir. Bu nedenle bu çalışmada ızgara tabanlı yol planlama problemleri için iyileştirilmiş bir ABC algoritması (IABC) önerilmiştir. Bu çalışmanın katkıları aşağıdaki gibi özetlenebilir:

- Yol planlama problemi hesaplama açısından zor bir optimizasyon problemidir. ABC algoritmasının kısıtları vardır ve ızgara tabanlı yol planlamasını ele alırken tatmin edici sonuçlar üretemez.

- IABC, kullanım-keşif dengesini iyileştirmek için tasarlanmış iki mekanizmanın entegre edilmesiyle geliştirilmiştir.

- Bu mekanizmalardan biri olan doğrusallaştırma stratejisi, dönüş sayısını azaltmak ve yol uzunluğunu kısaltmak için önerilmiştir. Bu strateji, gereksiz köşeleri ortadan kaldırarak robotun yolunu basitleştirmeyi, daha verimli hareket etmeyi ve enerji tasarrufu sağlamayı amaçlamaktadır.

- Bu mekanizmalardan bir diğeri olan yerel arama stratejisi, ABC algoritmasının yakınsama hızını artırmak ve küresel optimum çözümü bulma yeteneğini geliştirmek için önerilmiştir.

- Simülasyon deneylerinde IABC, temel ABC ve literatürdeki diğer algoritmalarla karşılaştırıldı ve sadece yol uzunluğunda değil, aynı zamanda yolu oluşturan hücre sayısı ve toplam dönüş açısında da diğer algoritmalardan üstün gelmiştir.

4.2.1. Problem Tanımı

Izgara tabanlı yol planlama problemi, bir mobil robotun engellere çarpmadan hedef noktaya ulaşmak için en az sayıda ızgara hücresinden geçmesi gereken bir minimizasyon problemi olarak tanımlanabilir. Izgara ortamları için 0 ve 1'den oluşan bir matris oluşturulur. Bu matriste 0, engelsiz bir hücreyi ve 1, bir engel hücresini temsil eder. Aday çözümler, Eşitlik (4.1)'deki gibi iki boyutlu bu ızgara ortamında bir dizi ızgara hücresinden oluşan bir yolu temsil eder.

$$Y = \left\{ p_1, p_2, \dots, p_i, \dots, p_{n_y} \mid n_y \in \mathbb{N}^+ \right\} \tag{4.1}$$

Burada, Y planlanan yolu, p_i yolun i'inci hücresini ve n_y ise yolu oluşturan hücre sayısı temsil eder. Bu çalışmada her yolun değerlendirildiği amaç fonksiyonu Eşitlik (4.2)'deki gibi modellenmiştir.

$$\arg\min_{Y} \mathcal{F} = \begin{cases} f_1, & eğer\ n_V = 0\ ise \\ f_2, & aksi\ taktirde \end{cases} \tag{4.2}$$

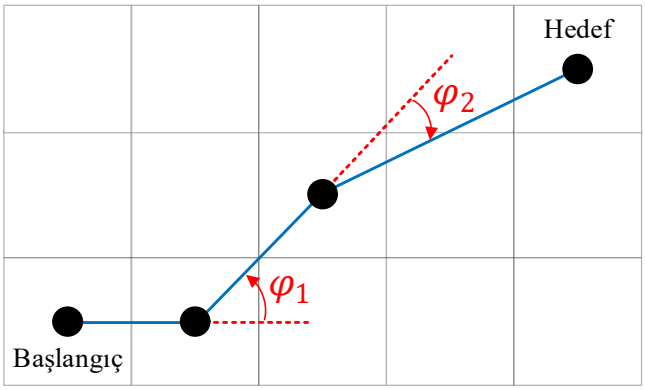
Burada, $\mathcal{F}$ amaç fonksiyonunu ve n_V engel ihlal sayısını temsil eder. Engel ihlal sayısı bir çözüm tarafından temsil edilen yolun kaç engeli ihlal ettiğini gösterir. Bu fonksiyon bir yolu iki farklı şekilde değerlendirir. Fonksiyon, engel ihlali yoksa yalnızca yolun uzunluğunu (f_1), engel ihlali varsa bir ceza puanı (f_2) üretir. f_1 yolun uzunluğunu hesaplar ve Eşitlik (4.3)'de gösterilir. f_2 ise n_V'ye bağlı olarak bir ceza puanı üretir ve Eşitlik (4.4)'te gösterilir.

$$f_1 = \sum_{i=1}^{n_y - 1} \| p_{i+1} - p_i \| \tag{4.3}$$

$$f_2 = m \cdot n \cdot n_V \tag{4.4}$$

Burada, m ve n sırasıyla ortam matrisinin satır ve sütun sayılarını (ızgara ortamının boyutunu) temsil eder. Önerilen algoritmanın kapsamlı değerlendirmesi için dört farklı metrik dikkate alınmıştır: Yol uzunluğu, yolu oluşturan hücre sayısı, toplam dönüş açısı ve dönüş enerji tüketimi. Bu metrikler, algoritmanın doğruluğunu ve verimliliğini değerlendirmede kritik öneme sahiptir, çünkü algoritmanın performansını objektif olarak ölçmek ve potansiyel iyileştirmeleri belirlemek için gereklidir. Yol uzunluğu ve hücre sayısı yukarıda bahsedilmektedir. Toplam dönüş açısı (φ_t) robotun yolda döndüğü açıların toplamıdır. Dönüş açısı ardaşık üç hücreden (p_i, p_{i+1}, p_{i+2}) oluşan iki düzlemsel çizgi arasındaki açı olarak düşünülebilir. Bu çizgiler hücreler arasındaki yol parçalarıdır. Toplam dönüş açısı bu çizgiler arasındaki dönüş açıların toplamıdır ve Eşitlik (4.5)'te gösterildiği gibi hesaplanmıştır. Hücre sayısı ve toplam dönüş açısı kavramları Şekil 4.1'de gösterilmektedir.

$$\varphi_t = \sum_{i=1}^{n_y-2} \varphi_i = \sum_{i=1}^{n_y-2} \cos^{-1}\left(\frac{(p_{i+1} - p_i) \cdot (p_{i+2} - p_{i+1})}{\|p_{i+1} - p_i\| \cdot \|p_{i+2} - p_{i+1}\|}\right) \tag{4.5}$$



Şekil 4.1. Hücre sayısı ve toplam dönüş açısı kavramları (Mavi çizgi planlanan bir yolu, siyah noktalar hücreleri ve φ_1 ve φ_2 ise dönüş açılarını temsil eder.)

Robot hareket yönünü değiştirdiği durumlarda düz ilerlemeye kıyasla daha fazla enerji harcamaktadır. Bu nedenle dönüş anlarındaki enerji tüketiminin de dikkate alınması gerekmektedir. Bu çalışmada dönüş enerji tüketimi ($\mathcal{E}_t$) robotun belirli bir yol üzerindeki hareketi sırasında yönünü değiştirdiği noktalarda harcadığı enerjilerin toplamı olarak tanımlanmıştır. Bu metrikte dönüş açısı 0° ise (robot düz gidiyorsa) dönüş enerji tüketimi sıfır kabul edilirken; 45°, 90°, 135° ve 180° gibi dönüşlerde tüketilen enerji daha

yüksektir. Ayrıca bu çalışmadaki dönüş enerji tüketimi sadece belirli dönüş açılarına değil, tüm yön değişimlerinin etkisine duyarlıdır. Her bir dönüş açısı 45°'lik referans değere oranlanarak normalize edilmiş ve toplam tüketim bu oranların toplamı üzerinden belirlenmiştir. Dönüş enerji tüketimi Eşitlik (4.6)'daki gibi hesaplanmıştır.

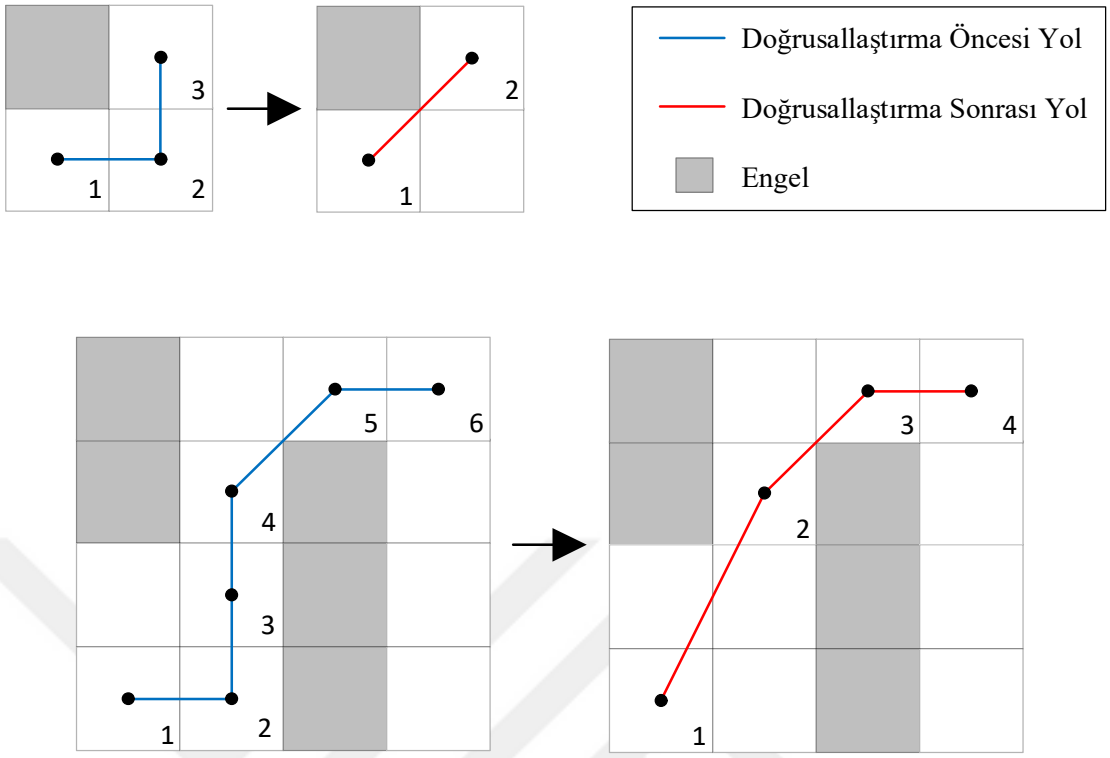
$$\mathcal{E}_t = \sum_{i=1}^{n_y-2} \mathcal{E}_i = \sum_{i=1}^{n_y-2} \frac{\varphi_i}{45°} \tag{4.6}$$

4.2.2. Önerilen Yöntem

Önerilen IABC için iki mekanizma eklenmiştir: doğrusallaştırma stratejisi ve yerel arama stratejisi. Doğrusallaştırma stratejisi, ızgara ortamında oluşturulan yolun gereksiz köşelerini ortadan kaldırmaya odaklanan bir kullanım tabanlı iyileştirmedir. Yerel arama stratejisi, en iyi çözümün maliyetini daha da optimize etmeyi amaçlayan bir keşif tabanlı mekanizmadır. Böylece bu iyileştirmeler algoritmanın kullanım-keşif dengesini bozmadan gerçekleştirilmiştir.

4.2.2.1. Doğrusallaştırma Stratejisi

Izgara tabanlı yol planlamasında, temel ABC optimum yolları planlarken engel içermeyen gereksiz ızgara hücrelerini kullanabilir. Doğrusallaştırma stratejisini kullanarak bu gereksiz ızgara hücreleri göz ardı edilir ve yol daha doğrusal hale getirilir. Böylece daha uygun maliyetli yollar elde edilebilir. $n_V = 0$ durumu için, f_1 alt fonksiyonunu hesaplamadan hemen önce, yolu doğrusallaştırmak için bu strateji kullanılır. Bundan sonra algoritma doğrusallaştırılmış yolun uzunluğunu hesaba katarak çalışmaya devam eder. Bu strateji, yolun ızgara hücrelerinin üçlü gruplarına odaklanarak ilerler. İlk noktayı ve üçüncü noktayı temel alır ve ikinci noktada bir engel olup olmadığını kontrol eder. Engel yoksa ikinci nokta gereksiz olarak tanımlanır ve yol dizisinden çıkarılır. Böylece daha doğrusal bir yol parçası elde edilir. İkinci noktada bir engel varsa hiçbir işlem yapılmaz ve bir sonraki üçlü gruba odaklanılır. Bu döngü, yolun tüm hücreleri üçlü gruplar halinde incelenene kadar devam eder. Doğrusallaştırma stratejisinin prensibini Şekil 4.2'de, doğrusallaştırma stratejisi tabanlı amaç fonksiyonunun (DSAF) sözde kodu ise Algoritma 4.1'de gösterilmektedir.



Şekil 4.2. Doğrusallaştırma stratejisinin prensibi (Yukarıdaki ortamda mavi yolun ikinci hücreyi ve aşağıdaki ortamda mavi yolun ikinci ve üçüncü hücreleri gereksiz olarak tanımlanabilir. İki yatay veya dikey ızgara hücresinin merkezleri arasındaki uzaklığın 1 br olduğu düşünüldüğünde, aşağıdaki ortam için mavi yolun uzunluğu 5.41 br, kırmızı yolun uzunluğu ise 4.65 br'dir. Bu sonuç yol uzunluğuna göre %14.05'lik bir iyileştirme olduğunu göstermektedir.)

Algoritma 4.1: DSAF'nin sözde kodu

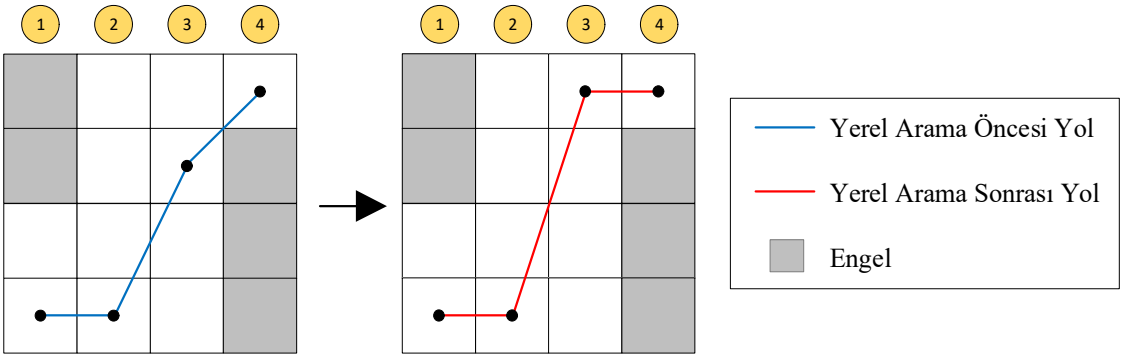
Girdi: x, E // Aday çözüm, ortam matrisi
Çıktı: f // Maliyet

$n_V \leftarrow 0,\ f \leftarrow 0, [m, n] \leftarrow size(E)$
$Y \leftarrow \mathcal{F}(x)$ // Aday çözümün yola dönüştürülmesi
$L_1 \leftarrow f_1(Y)$
for $i = 1 : L_1$
    $p_i \leftarrow Y(i)$
    if $E(p_i) == 1$
        $n_V \leftarrow n_V + 1$
    end if
end for
if $n_V == 0$
    $Y' \leftarrow Y$
    $L_2 \leftarrow f_1(Y')$
    $i \leftarrow 1$
    while $i \neq L_2 - 2$
        $p_i \leftarrow Y'(i)$
        $p_{i+2} \leftarrow Y'(i+2)$
        if $E(p_i : p_{i+2}) == 0$
            $Y'(i+1) = \emptyset$
            $i \leftarrow i - 1$
        end if
        $i \leftarrow i + 1$
        $L_2 \leftarrow f_1(Y')$
    end while
    $f \leftarrow \sum_{i=1}^{n_y - 1} \|p_{i+1} - p_i\|,\ Y' : \{p_1, \ldots, p_{n_y}\} \subset E$
else
    $f \leftarrow m \cdot n \cdot n_V$
end if

4.2.2.2. Yerel Arama Stratejisi

ABC bir çözümün komşularına ve problemin tek bir parametresine odaklanarak güncelleme gerçekleştirir. Yinelemelerde üretilen en iyi çözüm her zaman en iyi olmayabilir ve yerel minimuma düşebilir. Bu nedenle IABC'de her yinelemenin sonunda üretilen en iyi çözüm, kâşif arısı aşamasına benzer bir yerel arama stratejisine tabi tutulur. Bu stratejide, en iyi çözümün her parametresi için rastgele yeni bir değer üretilir. Bu yeni parametreli çözümün maliyeti mevcut çözümünkine göre daha düşükse yeni parametreli çözüm tercih edilir. Bu arama çözümün tüm boyutu boyunca devam eder. Bu strateji ile en iyi çözümden daha iyi maliyete sahip çözümler üretilebilir ve böylece algoritmanın arama performansı iyileştirilebilir. Yerel arama stratejisinin (YAS) prensibi Şekil 4.3'te,

bu strateji ile önerilen IABC algoritmasının sözde kodları sırasıyla Algoritma 4.2 ve 4.3'te, önerilen IABC algoritmasının akış diyagramı ise Şekil 4.4'te gösterilmektedir.



Şekil 4.3. Yerel arama stratejisinin prensibi (En iyi çözümü temsil eden bu yolun üçüncü hücreyi güncellenmiştir. İki yatay veya dikey ızgara hücresinin merkezleri arasındaki uzaklığın 1 br olduğu düşünüldüğünde mavi yolun uzunluğu 4.65 br, kırmızı yolun uzunluğu ise 4.16 br'dir. Bu sonuç yol uzunluğuna göre %10.54'lük bir iyileştirme olduğunu göstermektedir.)

Algoritma 4.2: YAS'ın sözde kodu

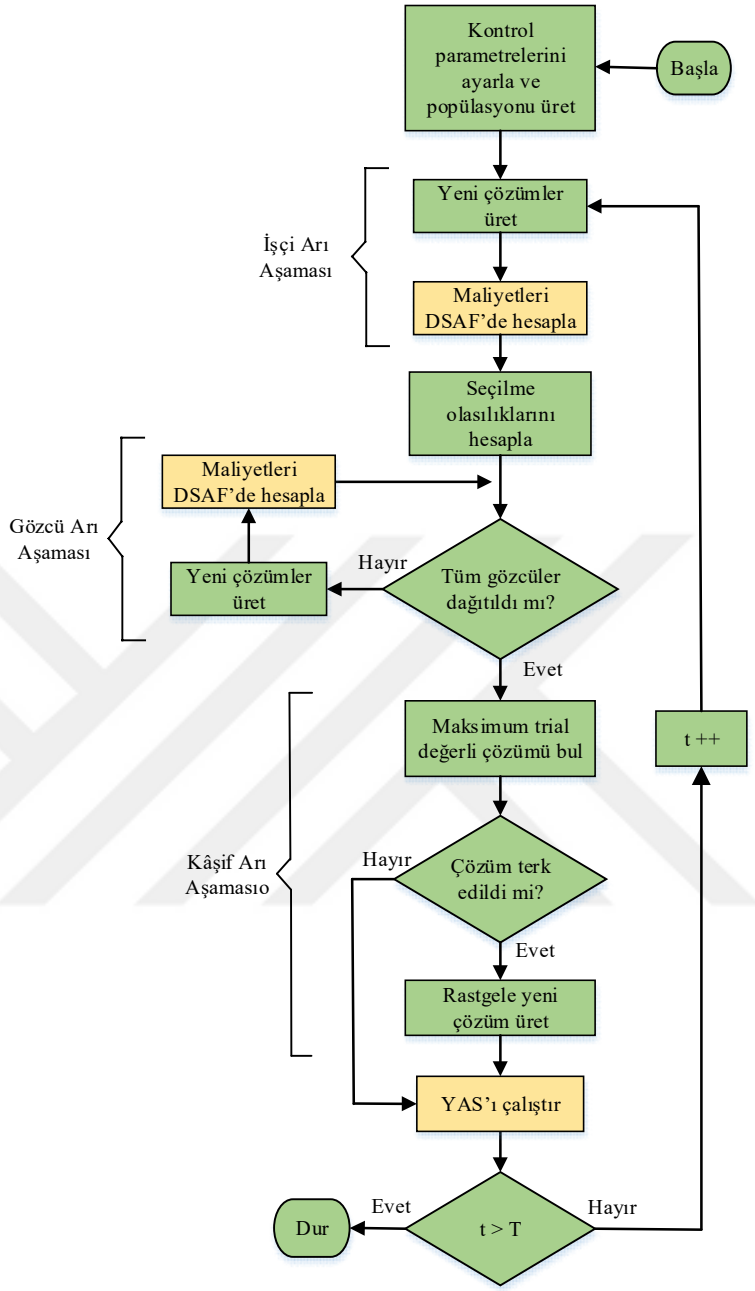
Girdi: $\hat{x}, \hat{f}, E, x_l, x_h$ // Eski en iyi çözüm, eski en iyi çözümün maliyeti, ortam matrisi, arama sınırları

Çıktı: $\hat{x}, \hat{f}$ // Yeni en iyi çözüm, yeni en iyi çözümün maliyeti

$m \leftarrow size(\hat{x})$
for $i = 1{:}m$
  $x' \leftarrow \hat{x}$
  $x' \leftarrow x_l + r(x_h - x_l), \;\; r \sim U(0,1)$
  $f' \leftarrow \text{DSAF}(x', E)$ // Algoritma 4.1
  if $f' < \hat{f}$
    $\hat{x} \leftarrow x'$
    $\hat{f} \leftarrow f'$
  end if
end for

Algoritma 4.3: Önerilen IABC algoritmasının sözde kodu

$E, D, x_{min}, x_{max} \leftarrow$ Ortam matrisi, problem boyutu, arama sınırları
$T, S, limit \leftarrow$ Kontrol parametreleri
$X \leftarrow x_l + r(x_h - x_l),\ \ r \sim U(0,1)^{S \times D}$
$f_i \leftarrow \text{DSAF}(x_i, E),\ \ i \in \{1,2,\dots,S\},\ \ x_i \in X$ // Algoritma 4.1
$fit_i \leftarrow \mathcal{F}(f_i)$ // Uygunluk değerleri - Eşitlik (3.33)
$(\hat{x}, \hat{f}) \leftarrow \arg\max_{x \in X} fit(x)$
for $t = 1:T$
    for $i = 1:S$
        $(x_i^j)' \leftarrow x_i^j + \Phi_i^j (x_i^j - x_{r_1}^j),\ \ x_{r_1} \sim X \setminus \{x_i\},\ \ j \sim \{1,2,\dots,D\}$
        $f_i' \leftarrow \text{DSAF}\left((x_i^j)', E\right)$ // Algoritma 4.1
        $fit_i' \leftarrow \mathcal{F}(f_i')$ // Uygunluk değeri - Eşitlik (3.33)
        if $fit_i' > fit_i$
            $x_i^j \leftarrow (x_i^j)'$
            $trial_{x_i} \leftarrow 0$
        else
            $trial_{x_i} \leftarrow trial_{x_i} + 1$
        end if
    end for
    $\delta \leftarrow \mathcal{F}(fit)$ // Seçilme olasılıkları - Eşitlik (3.34)
    for $i = 1:S$
        if $r \sim U(0,1) < \delta_i$
            $(x_i^j)' \leftarrow x_i^j + \Phi_i^j (x_i^j - x_{r_1}^j),\ \ x_{r_1} \sim X \setminus \{x_i\},\ \ j \sim \{1,2,\dots,D\}$
            $f_i' \leftarrow \text{DSAF}\left((x_i^j)', E\right)$ // Algoritma 4.1
            $fit_i' \leftarrow \mathcal{F}(f_i')$ // Uygunluk değeri - Eşitlik (3.33)
            if $fit_i' > fit_i$
                $x_i^j \leftarrow (x_i^j)'$
                $trial_{x_i} \leftarrow 0$
            else
                $trial_{x_i} \leftarrow trial_{x_i} + 1$
            end if
        end if
    end for
    $x_w \leftarrow \arg\max_{x \in X} trial(x)$
    if $trial_{x_w} > limit$
        $x_w \leftarrow x_l + r(x_h - x_l),\ \ r \sim U(0,1)$
        $f_{x_w} \leftarrow \text{DSAF}(x_w, E)$ // Algoritma 4.1
        $fit_{x_w} \leftarrow \mathcal{F}(f_{x_w})$ // Uygunluk değeri - Eşitlik (3.33)
    end if
    $(\hat{x}, \hat{f}) \leftarrow \arg\max_{x \in X} fit(x)$
    $(\hat{x}, \hat{f}) \leftarrow \text{YAS}(\hat{x}, \hat{f}, E, x_l, x_h)$ // Algoritma 4.2
end for
$\hat{x}, \hat{f}$



Şekil 4.4. Önerilen IABC algoritmasının akış diyagramı

4.2.2.3. IABC Algoritmasının Karmaşıklığı

Bu çalışmadaki amaç fonksiyonunda herhangi bir çözümün değerlendirme zamanı t_f^r olarak kabul edilirse, karmaşıklığı $O(t_f^r)$ olarak ifade edilir. Temel ABC'de, önce başlangıç popülasyonu amaç fonksiyonunda değerlendirilir. Popülasyon büyüklüğü S ile gösterilirse, ABC'nin başlangıç karmaşıklığı Eşitlik (4.7)'de gösterilmektedir.

$$O\left(S \cdot t_f^r\right) = O(S \cdot t^r) \tag{4.7}$$

Daha sonra işçi arı, gözcü arı ve kâşif arı aşamaları maksimum iterasyon sayısı (T) için çalıştırılır. Amaç fonksiyonu işçi ve gözcü arı aşamalarında S kez, bir çözümün sayacı limit değerine ulaşırsa kâşif arı aşamasında bir kez çağrılır. ABC'nin iteratif karmaşıklığı ve toplam karmaşıklığı sırasıyla Eşitlik (4.8) ve (4.9)'da gösterilmektedir [76].

$$\begin{aligned} &O\left(T \cdot \left(S \cdot t_f^r + S \cdot t_f^r + t_f^r\right)\right) \\ &= O\left(T \cdot \left(2 \cdot S \cdot t_f^r\right)\right) \\ &= O(T \cdot S \cdot t^r) \end{aligned} \tag{4.8}$$

$$O^{(ABC)} = O(S \cdot t^r) + O(T \cdot S \cdot t^r) = O(T \cdot S \cdot t^r) \tag{4.9}$$

Önerilen IABC'de ilk olarak, başlangıç popülasyonu amaç fonksiyonunda değerlendirilir. Ancak, amaç fonksiyonunu çağırmadan önce bir çözüme karşılık gelen yol doğrusallaştırma stratejisine tabi tutulur. Bu doğrusallaştırılmış yol (güncellenmiş çözüm) amaç fonksiyonunda değerlendirilir. Bu stratejinin çalışma süresi t_i^r olarak gösterilirse, IABC'nin başlangıç karmaşıklığı Eşitlik (4.10)'da gösterilmektedir.

$$O\left(S \cdot \left(t_i^r + t_f^r\right)\right) = O(S \cdot t^r) \tag{4.10}$$

Daha sonra işçi arı, gözlemci arı ve kâşif arısı aşamaları maksimum yineleme sayısı (T) boyunca çalıştırılır. Tıpkı ABC'de olduğu gibi, amaç fonksiyonu işçi ve gözlemci arı aşamalarında S kez ve çözümün sayacı limit değerine ulaşırsa kâşif arısı aşamasında bir kez çağrılır. Ancak, her amaç fonksiyonu çağrısından önce doğrusallaştırma stratejisi uygulanır. Amaç fonksiyonu yerel arama stratejisinde de çağrılır. Bu strateji yalnızca en iyi çözüm üzerinde çalıştığı için amaç fonksiyonu yalnızca bir kez çağrılır. Yerel arama stratejisinin çalışma zamanı t_o^r olarak gösterilirse, IABC'nin iteratif karmaşıklığı ve toplam karmaşıklığı Eşitlik (4.11) ve (4.12)'de gösterilmektedir.

$$\begin{aligned} &O\left(T \cdot \left(S \cdot \left(t_i^r + t_f^r\right) + S \cdot \left(t_i^r + t_f^r\right) + \left(t_i^r + t_f^r\right) + \left(t_o^r + t_f^r\right)\right)\right) \\ &= O\left(T \cdot \left(S \cdot t^r + S \cdot t^r + t^r + t^r\right)\right) \\ &= O\left(T \cdot \left(2 \cdot S \cdot t^r\right)\right) \\ &= O(T \cdot S \cdot t^r) \end{aligned} \tag{4.11}$$

$$O^{(IABC)} = O(S \cdot t^r) + O(T \cdot S \cdot t^r) = O(T \cdot S \cdot t^r) \quad (4.12)$$

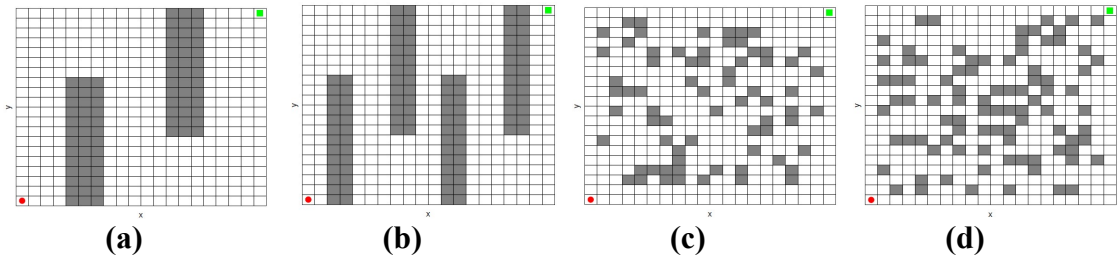
Bu denklemler ABC ve IABC algoritmalarının aynı hesaplama karmaşıklığına sahip olduğunu göstermektedir.

4.2.3. Bulgular

Bu çalışmadaki tüm simülasyonlar MATLAB programlama dilinde kodlanmış ve 16 GB RAM ve 2.6 GHz işlemciye sahip bir bilgisayarda çalıştırılmıştır. Simülasyon deneylerinde öncelikle önerilen IABC'nin performansı gerçekleştirilmiştir. IABC'nin performans analizi için öncelikle aynı boyuttaki ortamlarda temel ABC ile detaylı bir şekilde karşılaştırılmıştır. İkinci olarak, IABC'nin iyileştirme bileşenlerinin katkılarını test etmek için bir ablasyon analizi gerçekleştirilmiştir. Üçüncü olarak IABC'nin performansı farklı boyutlardaki ortamlarda test edilmiş ve algoritmanın ölçeklenebilirliği araştırılmıştır. Son olarak IABC iyi bilinen ve yeni geliştirilmiş algoritmalarla ve güncel çalışmalardan elde edilen sonuçlarla kapsamlı bir şekilde karşılaştırılmıştır.

4.2.3.1. IABC Algoritmasının ABC ile Karşılaştırılması

Bu simülasyonda önerilen IABC temel ABC ile karşılaştırılmıştır. İlk olarak, Şekil 4.5'te gösterildiği gibi farklı zorluk seviyelerine sahip dört adet 20 x 20 br^2 boyutunda ızgara tipi ortam tasarlanmıştır. İki yatay veya dikey ızgara hücresinin merkezleri arasındaki uzaklık 1 br'dir. Bu ortamlarda başlangıç noktası sol alt köşe (1, 1) ve hedef noktası sağ üst köşedir (20, 20). Ortamların engel oranı, özellikleri ve tasarım kriterleri Tablo 4.2'de gösterilmektedir.

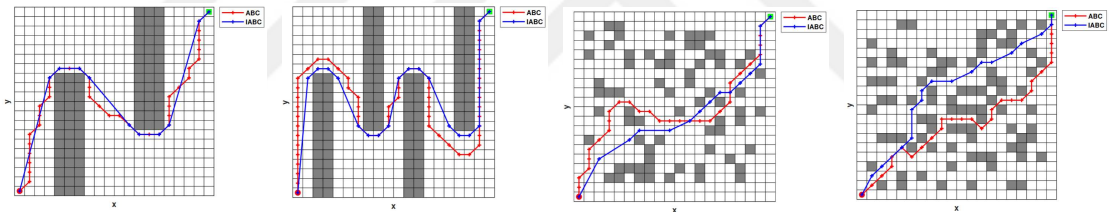


Şekil 4.5. Aynı boyutlarda dört farklı ızgara tipi ortam: **(a)** Ortam 1, **(b)** Ortam 2, **(c)** Ortam 3, **(d)** Ortam 4 (Gri kareler engelleri, kırmızı daireler ve yeşil kareler sırasıyla robotun başlangıç ve hedef noktalarını temsil eder.)

Tablo 4.2. Ortamların engel oranı, özellikleri ve tasarım kriterleri

Ortam	Engel Oranı	Özellik	Tasarım Kriteri
1	0.19	Düzenli engellerle basit bir ortam	Alternatif çözümlerin daha az olduğu kısıtlı ortamlarda IABC algoritmasının performansını değerlendirmek.
2	0.26	Düzenli engellerle karmaşık bir ortam	
3	0.17	Rastgele dağıtılmış engellerle basit bir ortam	Düzensiz ve kaotik ortamlarda IABC algoritmasının performansını değerlendirmek.
4	0.22	Rastgele dağıtılmış engellerle karmaşık bir ortam	

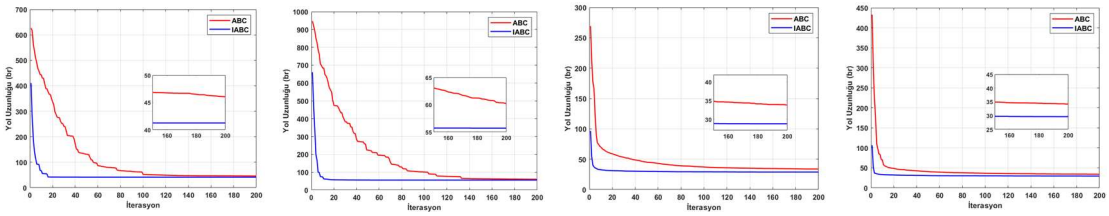
Her iki algoritma için de maksimum iterasyon sayısı, popülasyon boyutu ve limit değeri sırasıyla 200, 50 ve 100'dür [77]. Algoritmalar 30 koşma ile çalıştırılmıştır. Simülasyon sonucunda aynı boyuttaki ortamlarda her iki algoritma tarafından planlanan yollar, IABC'nin ABC ile performans karşılaştırması ve ortalama yakınsama grafikleri sırasıyla Şekil 4.6, Tablo 4.3 ve Şekil 4.7'de gösterilmektedir.



Şekil 4.6. ABC ve IABC algoritmalarının Ortam 1-4 için planladığı yollar: **(a)** Ortam 1, **(b)** Ortam 2, **(c)** Ortam 3, **(d)** Ortam 4

Tablo 4.3. IABC ve ABC'nin aynı boyuttaki ortamlarda performans karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Ortam	Algoritma	Yol Uzunluğu (br)			Hücre Sayısı			Toplam Dönüş Açısı (rad)			Dönüş Enerji Tüketimi		
		OP	OR	SS	OP	OR	SS	OP	OR	SS	OP	OR	SS
1	ABC	43.21	46.06	1.80	38	41.33	1.88	11.78	15.39	2.26	14	19.53	3.03
	IABC	41.24	41.24	0	11	11	0	4.94	4.94	0	2.36	2.36	0
2	ABC	57.21	60.25	2.43	53	55.96	4.86	14.92	18.09	2.94	13	19.93	5.31
	IABC	55.63	55.71	0.32	18	18.93	0.25	11.35	11.36	0.04	2.54	2.66	0.39
3	ABC	30.79	33.84	1.65	24	29.03	2.25	7.06	13.32	2.84	9	17.80	4.93
	IABC	27.46	28.79	0.98	8	10.40	1.63	1.95	4.20	1.47	2	3.95	1.05
4	ABC	31.21	34.17	1.18	26	29.10	1.49	10.21	15.47	2.45	10	16.56	4.17
	IABC	28.50	29.61	0.47	12	15.03	1.27	5.53	7.38	1.23	2.59	4.28	1.64



Şekil 4.7. Ortam 1-4 için ortalama yakınsama grafikleri: **(a)** Ortam 1, **(b)** Ortam 2, **(c)** Ortam 3, **(d)** Ortam 4

Şekil 4.6'da gösterilen yollar, IABC'nin ABC'den daha kısa ve daha düzgün yollar ürettiğini göstermektedir. Doğrusallaştırma stratejisi nedeniyle IABC tarafından üretilen yollardaki hücre sayısı ABC'den daha azdır. ABC, yolun geçtiği her hücreyi hesaba katmıştır, bu da yolun maliyetini artırır. Tablo 4.3, IABC'nin yol uzunlukları açısından her ortamda ABC'den üstün olduğunu göstermektedir. IABC, ABC'ye kıyasla ortalama yol uzunlukları açısından Ortam 1 için %10.45, Ortam 2 için %7.53, Ortam 3 için %14.9 ve Ortam 4 için %13.35'lik bir iyileştirme sağlamıştır. Standart sapmalar, IABC'nin ABC'den daha kararlı sonuçlar ürettiğini göstermektedir. Ayrıca, hücre sayısına göre önerilen IABC ABC'den daha etkili görünmektedir. Ortalama hücre sayısı açısından IABC, ABC'ye kıyasla Ortam 1 için %73.38, Ortam 2 için %66.17, Ortam 3 için %61.17 ve Ortam 4 için %48.34 oranında iyileştirme sağlamıştır. Bu sonuçlar, önerilen algoritmanın diğer yöntemlere göre daha az hücre kullanarak aynı optimizasyon hedeflerine ulaşabileceğini göstermektedir. Toplam dönüş açısına göre, IABC her ortam için ABC'den daha düşük açılı ve daha düzgün yollar planlamıştır. Böylece diğer metriklerde olduğu gibi bu metrikte de etkili bir yöntem olduğu görülmektedir. Robotların dönüş hareketlerinde de enerji tükettiği düşünüldüğünde, önerilen algoritmanın robotun enerjisini optimum şekilde yönettiği söylenebilir. Ortalama toplam dönüş açısı açısından, IABC, ABC algoritmasına kıyasla Ortam 1 için %67.89, Ortam 2 için %37.18, Ortam 3 için %68.44 ve Ortam 4 için %52.26 oranında bir iyileştirme sağlamıştır. Şekil 4.7'de gösterilen yakınsama grafikleri, IABC'nin ABC'den daha hızlı yakınsadığını göstermektedir. Ayrıca, her iki algoritma da düzensiz engel ortamlarında düzenli engel ortamlarına göre daha hızlı yakınsamıştır. Bu sonuçlar, IABC'nin tüm değerlendirme ölçütleri açısından temel ABC'den üstün olduğunu kanıtlamaktadır.

4.2.3.2. IABC Algoritmasının Ablasyon Analizi

Ablasyon analizi, yapay zekâda bir algoritmanın farklı bileşenlerinin performansı üzerindeki etkisini incelemek için kullanılan bir yöntemdir. Modelin belirli bir özelliğini veya bileşenini kaldırarak performans değişikliği analiz edilir ve bu da kritik bileşenlerin tanımlanmasına olanak tanır. Önerilen IABC'nin doğrusallaştırma ve yerel arama olmak üzere iki iyileştirme bileşeni olduğundan, bu iki bileşenin ayrı ayrı etkilerini incelemek ve hangisinin daha kritik bir iyileştirme olduğunu belirlemek önemlidir. Bu etkiyi incelemek için, IABC-1 ve IABC-2 olmak üzere iki algoritma dikkate alınmıştır. IABC-1 sadece doğrusallaştırma bileşenini içerirken, IABC-2 sadece yerel arama bileşenini içerir. IABC-1 ve IABC-2, Alt Başlık 4.2.3.1'de bahsedilen Ortam 2 ve 4 için test edilmiştir. Adil bir karşılaştırma için, IABC-1 ve IABC-2'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 200 ve 30 olarak ayarlanmıştır. Algoritmaların limit değerleri de 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. ABC, IABC-1, IABC-2 ve IABC'nin Ortam 2 ve 4 için yol uzunluğu karşılaştırması Tablo 4.4'te gösterilmektedir.

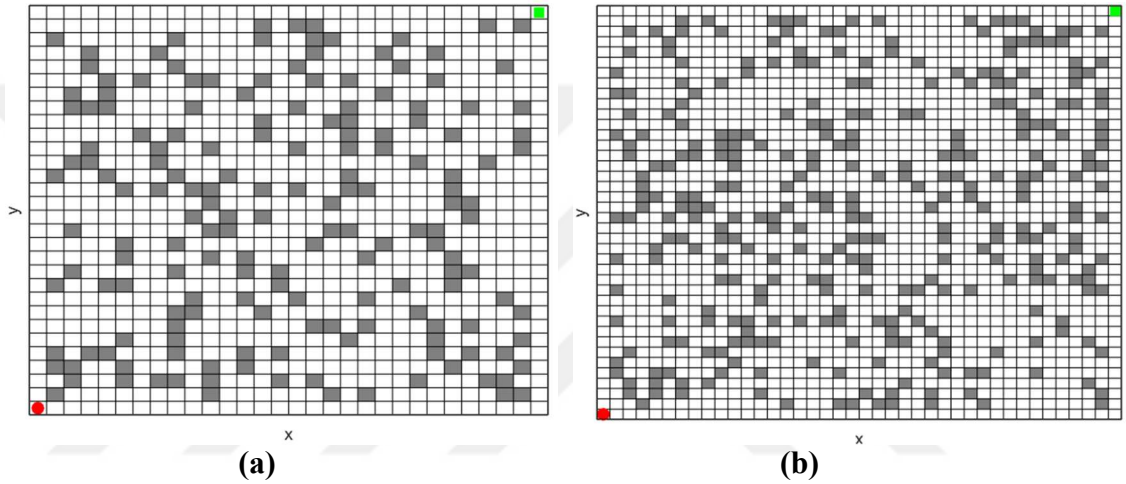
Tablo 4.4. ABC, IABC-1, IABC-2 ve önerilen IABC algoritmalarının Ortam 2 ve 4 için yol uzunluğu (br) karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Algoritma	**Ortam 2**			**Ortam 4**		
	OP	**OR**	**SS**	**OP**	**OR**	**SS**
ABC	57.21	60.25	2.43	31.21	34.17	1.18
IABC-1	55.63	56.36	1.32	28.83	30.60	0.83
IABC-2	57.21	58.79	1.45	29.21	33.06	1.28
IABC	55.63	55.71	0.32	28.50	29.61	0.47

Tablo 4.4'te IABC-1'in IABC-2'den daha fazla iyileştirme sağladığı görülmektedir. Ortam 2'deki ortalamalar açısından, IABC-1 ABC'ye kıyasla %6.45'lik bir iyileştirme sağlarken, IABC-2 %2.43'lük bir iyileştirme sağlamıştır. Ortam 4 için ortalamalar açısından, IABC-1 ABC'ye kıyasla %10.53'lük bir iyileştirme sağlarken, IABC-2 %3.26'lık bir iyileştirme sağlamıştır. Ayrıca, Ortam 2 için ortalamalar açısından IABC, IABC-1'e kıyasla %1.17 ve IABC-2'ye kıyasla %5.21'lik bir iyileştirme göstermiştir. Ortam 4 için ortalamalar açısından IABC, IABC-1'e kıyasla %3.24 ve IABC-2'ye kıyasla %10.44'lük bir iyileştirme göstermiştir. Bu sonuçlara göre, doğrusallaştırma bileşeni yerel arama bileşeninden daha kritiktir.

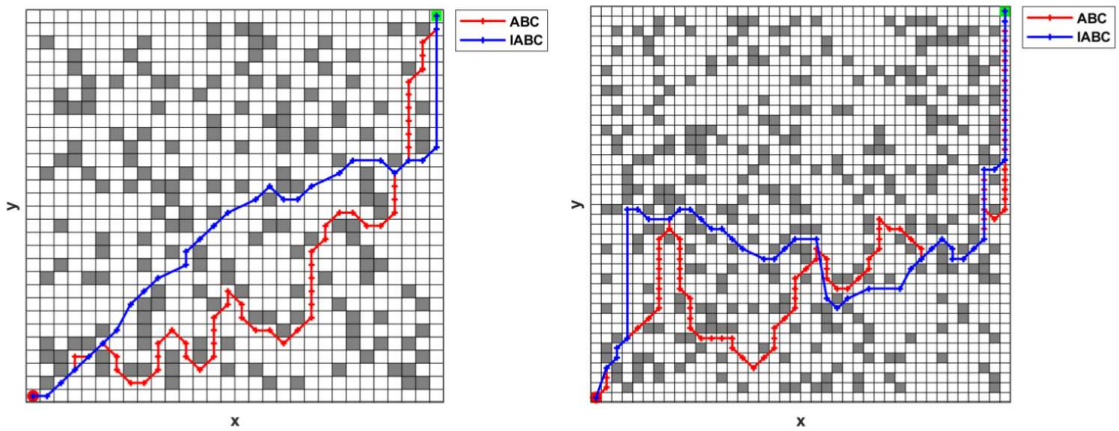
4.2.3.3. IABC Algoritmasının Ölçeklenebilirlik Analizi

Önerilen algoritmanın farklı boyutlardaki veri kümelerinde benzer performans gösterip göstermediğini belirlemek de önemlidir. Bu nedenle, 30 x 30 br^2 ve 40 x 40 br^2 boyutlarında Ortam 5 ve Ortam 6 olarak adlandırılan iki ek ortam daha tasarlanmıştır. Bu ortamlarda başlangıç noktaları sol alt köşe (1, 1); hedef noktaları sağ üst köşelerdir, Ortam 5 için (30, 30) ve Ortam 6 için (40, 40). Ortam 5 ve 6'nın engel oranları Alt Başlık 4.2.3.1'de bahsedilen Ortam 4 ile aynıdır. Bu ortamlar Şekil 4.8'de gösterilmektedir.



Şekil 4.8. Farklı boyutlarda iki farklı ızgara tipi ortam: **(a)** Ortam 5, **(b)** Ortam 6 (Gri kareler engelleri, kırmızı daireler ve yeşil kareler sırasıyla robotun başlangıç ve hedef noktalarını temsil eder.)

Adil bir karşılaştırma için ABC ve IABC'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 200 ve 50 olarak ayarlanmıştır. Algoritmaların limit değerleri 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. Ortam 5 ve 6 için ABC ve IABC tarafından planlanan yollar Şekil 4.9'da gösterilmektedir. IABC ve ABC'nin farklı boyutlardaki ortamlarda performans karşılaştırması Tablo 4.5'te gösterilmektedir.



(a) **(b)**

Şekil 4.9. ABC ve IABC algoritmalarının Ortam 5 ve 6 için planladığı yollar: **(a)** Ortam 5, **(b)** Ortam 6

Tablo 4.5. IABC ve ABC'nin 30 x 30 br^2 ve 40 x 40 br^2 boyutlardaki ortamlarda performans karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Ortam	Algoritma	Yol Uzunluğu (br)			Hücre Sayısı			Toplam Dönüş Açısı (rad)			Dönüş Enerji Tüketimi		
		OP	OR	SS	OP	OR	SS	OP	OR	SS	OP	OR	SS
5	ABC	56.52	64.54	3.75	48	56.40	3.72	21.99	29.68	4.15	22	35.06	5.69
	IABC	45.74	52.10	3.15	17	22.16	2.56	10.35	15.73	2.65	1.18	6.79	2.92
6	ABC	90.42	114.5	13.39	79	102.4	12.96	32.98	47.83	8.50	37	60.53	15.77
	IABC	68.67	91.25	11.77	23	32.76	4.89	15.38	26.66	5.86	6.40	15.36	5.09

Şekil 4.9, IABC'nin 30 x 30 br^{2}ve 40 x 40 br^2 ortamlarda 20 x 20 br^2 ortamlarda olduğu gibi ABC'den daha kısa yollar ürettiğini göstermektedir. Tablo 4.5 dikkate alındığında, IABC tüm değerlendirme metriklerinde ABC'den yine daha iyi performans göstermiştir. Ortalama yol uzunlukları açısından IABC, Ortam 5 için %19.26 ve Ortam 6 için %20.31'lik bir iyileştirme göstermiştir. Ortalama hücre sayısı açısından, IABC sırasıyla %60.71 ve %68.01'lik bir iyileştirme göstermiştir. Ortalama toplam dönüş açısı açısından, IABC sırasıyla %47.02 ve %44.24'lük bir iyileştirme göstermiştir. Her metriğin standart sapmalarına göre, IABC'nin ABC'den daha kararlı bir algoritma olduğu söylenebilir. Bu sonuçlar, önerilen IABC'nin farklı boyutlardaki veri kümelerinde benzer performans gösterdiğini ve ızgara tabanlı yol planlama probleminde kullanışlı bir algoritma olduğunu kanıtlamaktadır.

4.2.3.4. IABC Algoritmasının İyi Bilinen ve Yeni Geliştirilmiş Algoritmalarla Karşılaştırılması

Bu simülasyonda, önerilen IABC'nin yol planlama performansı DE, PSO, GA ve kovaryans matrisi adaptasyon evrim stratejisi (covariance matrix adaptation evolution strategy, CMA-ES) [78] gibi iyi bilinen algoritmalarla karşılaştırılmıştır. Ayrıca literatürdeki beyaz köpekbalığı optimizasyonu (white shark optimizer, WSO) [79] ve geyik sürüsü optimizasyonu (elk herd optimizer, EHO) [80] gibi yeni geliştirilmiş algoritmalarla da karşılaştırılmıştır. Bu karşılaştırmalar için Alt Başlık 4.2.3.1'de belirtilen Ortam 2 kullanılmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. Tüm algoritmaların kontrol parametreleri Tablo 4.6'da, IABC'nin iyi bilinen ve yeni geliştirilmiş algoritmalarla performans karşılaştırılması Tablo 4.7'de gösterilmektedir.

Tablo 4.6. Tüm algoritmaların kontrol parametreleri (T maksimum iterasyon sayısı, N ve λ popülasyon boyutu, F ölçekleme faktörü, CR çaprazlama oranı, c_1 ve c_2 kişisel ve sosyal katsayılar, MR mutasyon oranı, f_l ve f_h dalgalı hareketin sınır frekansları, τ ivme katsayısı, a_0, a_1 ve a_2 pozitif sabitler, B_r boğa oranı, μ ebeveyn sayısı, c_σ ve d_σ adım boyutu kontrolü için öğrenme oranı ve sönümleme parametresi, c_c, ve c_μ kovaryans güncelleme parametreleridir. CMA-ES'nin parametreleri bu çalışmadaki yol planlama problemi için problem boyutu 18 ve arama alanı sınırları [1, 20] ile ayarlanmıştır.)

Algoritma	**Parametre**
ABC	$T = 200, S = 50, limit = 100$
DE	$T = 200, S = 50, F = [0.2, 0.8], CR = 0.2$
PSO	$T = 200, S = 50, c_1 = 2, c_2 = 2$
GA	$T = 200, S = 50, CR = 0.9, MR = 0.1$
CMA-ES	$T = 200, S = 50, \mu = 25, c_\sigma = 0.43, d_\sigma = 1.43, c_c = 0.2, c_\mu = 0.0581$
WSO	$T = 200, S = 50, f_l = 0.07, f_h = 0.75, \tau = 4.12, a_0 = 6.25,$ $a_1 = 100, a_2 = 0.0005$
EHO	$T = 200, S = 50, B_r = 0.2$
IABC	$T = 200, S = 50, limit = 100$

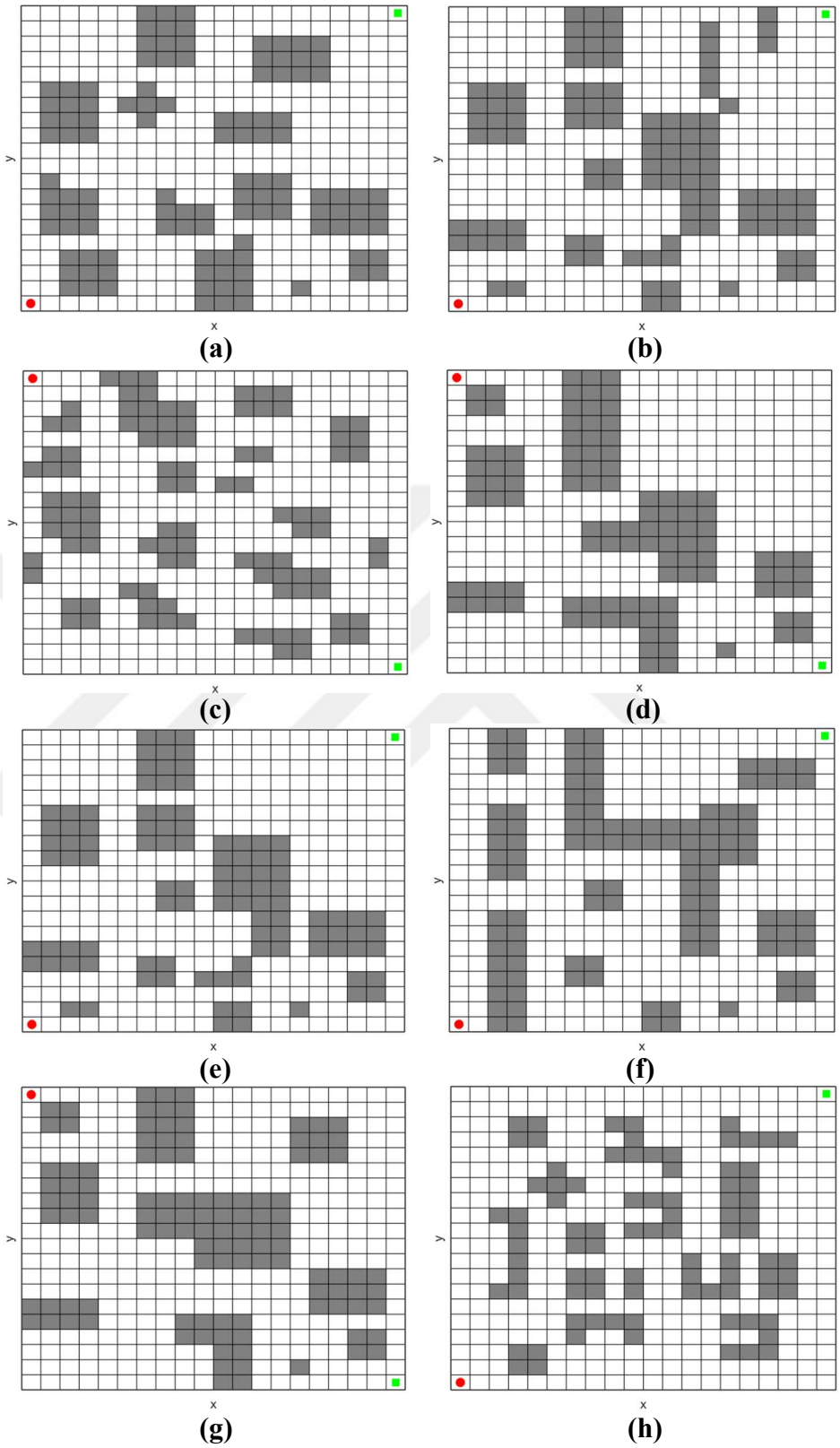
Tablo 4.7. Ortam 2 için IABC'nin iyi bilinen ve yeni geliştirilmiş algoritmalarla performans karşılaştırılması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Algoritma	Yol Uzunluğu (br)			Hücre Sayısı			Toplam Dönüş Açısı (rad)			Dönüş Enerji Tüketimi		
	OP	OR	SS	OP	OR	SS	OP	OR	SS	OP	OR	SS
ABC	57.21	60.25	2.43	53	55.96	4.86	14.92	18.09	2.94	13	19.93	5.31
DE	57.21	57.21	0.00	52	52.00	0.00	14.92	18.22	1.20	13	16.13	1.69
PSO	75.21	78.49	1.84	70	73.06	2.39	16.49	26.65	3.05	20	33.53	4.88
GA	80.04	99.57	14.56	64	84.70	16.32	24.34	35.18	4.17	25	50.53	10.54
CMA-ES	93.45	106.4	3.76	94	103.7	4.14	16.49	24.19	6.39	23	35.40	7.95
WSO	57.80	65.78	9.54	54	62.83	11.54	16.49	24.24	5.24	15	30.83	10.97
EHO	57.21	60.77	4.49	52	55.76	3.22	14.92	19.16	2.40	11	20	4.54
IABC	55.63	55.71	0.32	18	18.93	0.25	11.35	11.36	0.04	2.54	2.66	0.39

Tablo 4.7'ye göre, IABC'nin tüm değerlendirme ölçütlerinde iyi bilinen ve yeni geliştirilmiş algoritmalardan daha iyi performans gösterdiği görülmektedir. Ortalama yol uzunluğu açısından IABC, DE'ye kıyasla %2.62, PSO'ya kıyasla %29.02, GA'ya kıyasla %44.05, CMA-ES'ye kıyasla %47.67, WSO'ya kıyasla %15.31 ve EHO'ya kıyasla %8.33 oranında bir iyileştirme göstermiştir. Ortalama hücre sayısı açısından, IABC aynı sırayla %63.59, %74.09, %77.65, %81.74, %69.87 ve %66.05 oranında bir iyileştirme göstermiştir. Ortalama toplam dönüş açısı açısından IABC, sırasıyla %37.64, %57.36, %67.71, %53.03, %53.13 ve %40.71 oranında iyileştirme göstermiştir. Standart sapmalardan da görülebileceği gibi IABC diğer algoritmalardan daha kararlıdır. Yol uzunluğu ve hücre sayısının standart sapmaları açısından IABC, DE ile rekabet hâlinde olmuştur; ancak DE'nin yerel minimumda takılıp kaldığı ve IABC'nin en iyi ve ortalama değerler açısından DE'den üstün olduğu açıktır. Bu sonuçlar önerilen IABC'nin iyi bilinen ve yeni geliştirilmiş algoritmalardan daha iyi performans gösterdiğini kanıtlamaktadır.

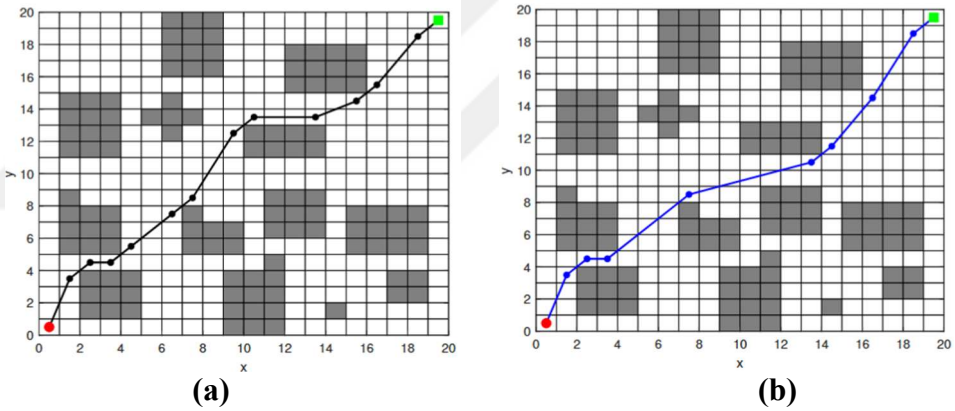
4.2.3.5. IABC Algoritmasının Güncel Çalışmalardaki Sonuçlarla Karşılaştırılması

Bu simülasyon, önerilen IABC'nin yol planlama performansını literatürdeki son çalışmalarda önerilen yöntemlerin sonuçlarıyla karşılaştırmayı amaçlamaktadır. Bu nedenle IABC'nin yol planlama performansı 20 x 20 br^2 boyutlu ızgara ortamlarda ABC ve [81], [82], [83], [84] ve [85]'te önerilen algoritmaların performanslarıyla karşılaştırılmıştır. Bu referanslarda tasarlanan ortamlar Şekil 4.10'da gösterilmektedir.

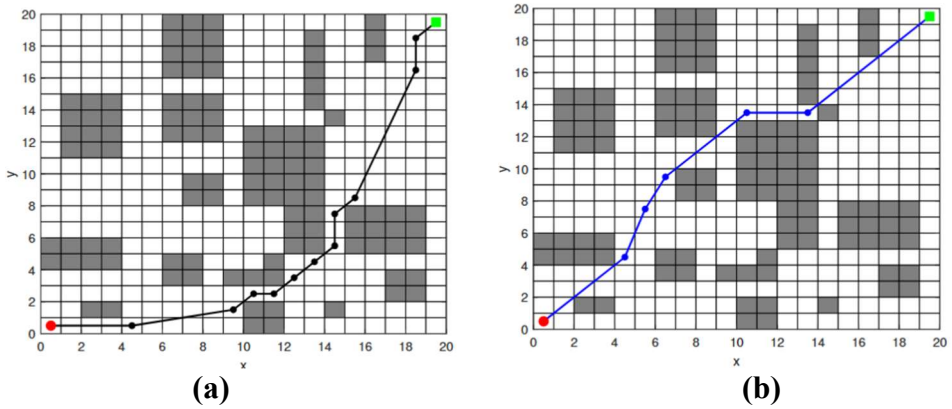


Şekil 4.10. [81], [82], [83], [84] ve [85]'te tasarlanan ortamlar: **(a)** Ortam 7 [81], **(b)** Ortam 8 [81], **(c)** Ortam 9 [82], **(d)** Ortam 10 [82], **(e)** Ortam 11 [83], **(f)** Ortam 12 [83], **(g)** Ortam 13 [84], **(h)** Ortam 14 [85]

[81]'deki algoritmalar WOA, PSO, gri kurt optimizasyonu (grey wolf optimizer, GWO), isli sumru optimizasyon algoritması (sooty tern optimization algorithm, STOA), salp sürü algoritması (salp swarm algorithm, SPSA), martı optimizasyon algoritması (seagull optimization algorithm, SOA) ve [81]'de önerilen algoritma olan hibrit ateş böceği-balina optimizasyon algoritmasıdır (hybrid firefly-whale optimization algorithm, FWOA). Adil bir karşılaştırma için, ABC ve IABC'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 500 ve 60 olarak ayarlanmıştır. ABC ve IABC'nin limit değerleri 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. [81]'de tasarlanan ortamlarda (Ortam 7 ve 8) FWOA ve bu çalışmada önerilen IABC tarafından planlanan yollar Şekil 4.11 ve 4.12'de gösterilmektedir. Bu ortamlarda başlangıç noktası sol alt köşe (1, 1) ve hedef noktası sağ üst köşedir (20, 20). Algoritmaların yol uzunluğu karşılaştırması Tablo 4.8'de gösterilmektedir.



Şekil 4.11. FWOA ve IABC algoritmalarının Ortam 7 için planladığı yollar: **(a)** FWOA, **(b)** IABC



Şekil 4.12. FWOA ve IABC algoritmalarının Ortam 8 için planladığı yollar: **(a)** FWOA, **(b)** IABC

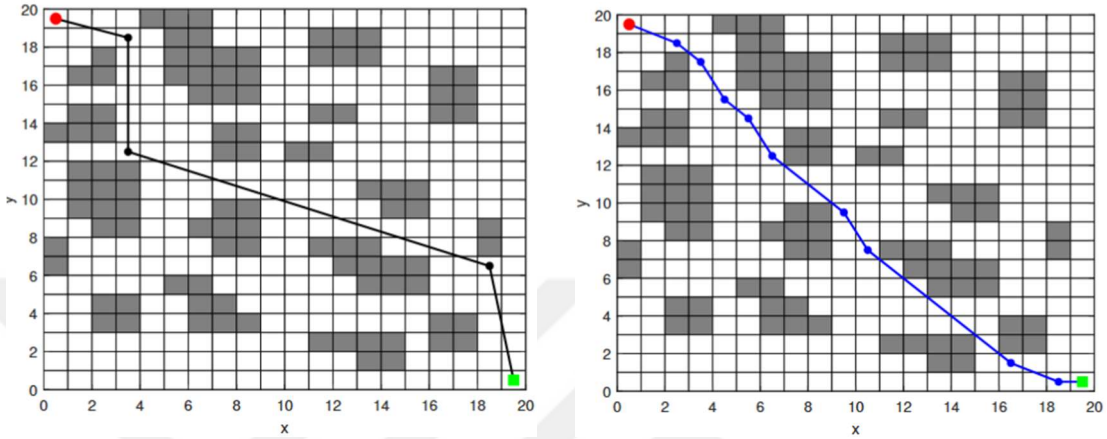
Tablo 4.8. IABC ile ABC ve [81]'deki algoritmaların yol uzunluğu (br) karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Algoritma	Ortam 7			Ortam 8		
	OP	OR	SS	OP	OR	SS
ABC	29.79	31.07	0.64	30.38	31.83	0.93
WOA	28.70	45.48	67.09	29.80	45.71	66.95
PSO	28.46	31.67	4.60	29.91	44.59	67.15
GWO	28.82	30.55	2.40	29.21	31.05	0.92
STOA	29.40	29.45	0.13	30.87	31.14	0.17
SSA	28.82	29.85	0.36	29.11	31.33	0.54
SOA	28.82	29.45	0.22	30.54	31.16	0.21
FWOA	28.46	29.37	0.56	28.42	30.55	1.36
IABC	28.46	28.52	0.18	28.19	28.39	0.34

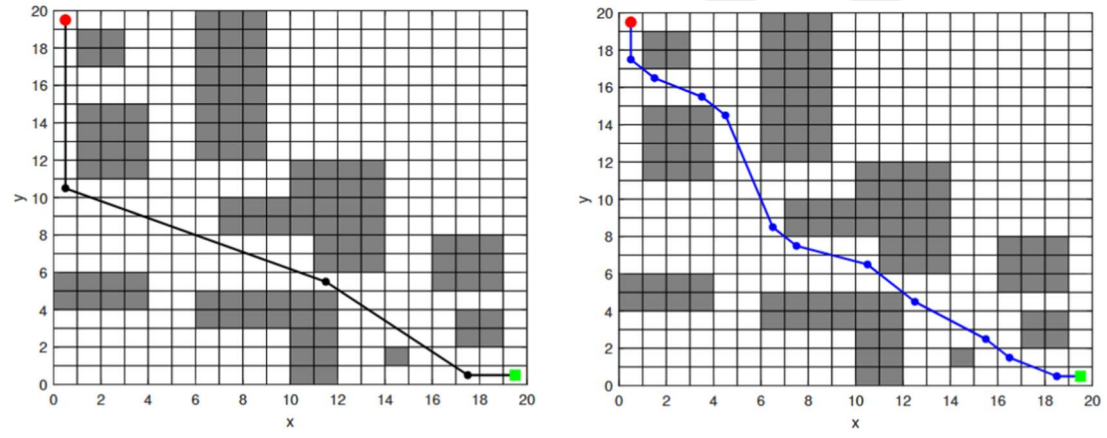
Ortam 7 için, IABC ile planlanan yolların en iyi uzunlukları ABC'ye kıyasla %4.48, WOA'ya kıyasla %0.82, STOA'ya kıyasla %3.20 ve GWO, SSA ve SOA'ya kıyasla %1.26 oranında kısaltılmıştır. IABC, FWOA ve PSO ile aynı sonucu üretmiştir. Tablo 4.9'daki sıraya göre ortalama uzunluklarda sırasıyla %8.22, %37.28, %9.95, %6.66, %3.15, %4.45, %3.17 ve %2.89 oranında iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi, IABC diğer algoritmalardan çoğunlukla daha kararlı olmasına rağmen, STOA'dan biraz daha kararsızdır. Ancak bu kararsızlık, STOA'dan daha iyi olan en iyi-ortalama bandındadır. En kararsız algoritma WOA'dır. Ortam 8 için IABC ile planlanan yolların en iyi uzunlukları %7.20, %5.39, %5.73, %3.50, %8.67, %3.16, %7.67 ve %0.81 oranında kısaltılmıştır. Ortalama uzunluklarda %10.80, %37.89, %36.33, %8.57, %8.84, %9.39, %8.89 ve %7.08 oranında bir iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi IABC'nin STOA, SOA ile yaklaşık olarak aynı kararlılık seviyesinde olduğu söylenebilir. En kararsız algoritmaların PSO ve WOA olduğu görülmektedir.

[82]'deki algoritmalar ACO, ABC, WOA, yıldırım tutunma prosedürü optimizasyonu (lightning attachment procedure optimization, LAPO), SSA, GWO, beyaz balina algoritması (beluga whale algorithm, BWO), karadul örümceği algoritması (black widow spider algorithm, BWOA) ve [82]'de önerilen algoritma olan iyileştirilmiş bir ACO ve iyileştirilmiş bir ABC'nin hibritidir (IACO-IABC). Adil bir karşılaştırma için, ABC ve IABC'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 200 ve 40 olarak ayarlanmıştır. ABC ve IABC'nin limit değerleri 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. [82]'de tasarlanan ortamlarda (Ortam 9 ve 10)

IACO-IABC ve bu çalışmada önerilen IABC tarafından planlanan yollar Şekil 4.13 ve 4.14'te gösterilmektedir. Bu ortamlarda başlangıç noktası sol üst köşe (1, 20) ve hedef noktası sağ alt köşedir (20, 1). Algoritmaların yol uzunluğu karşılaştırması Tablo 4.9'da gösterilmektedir.



Şekil 4.13. IACO-IABC ve IABC algoritmalarının Ortam 9 için planladığı yollar: **(a)** IACO-IABC, **(b)** IABC



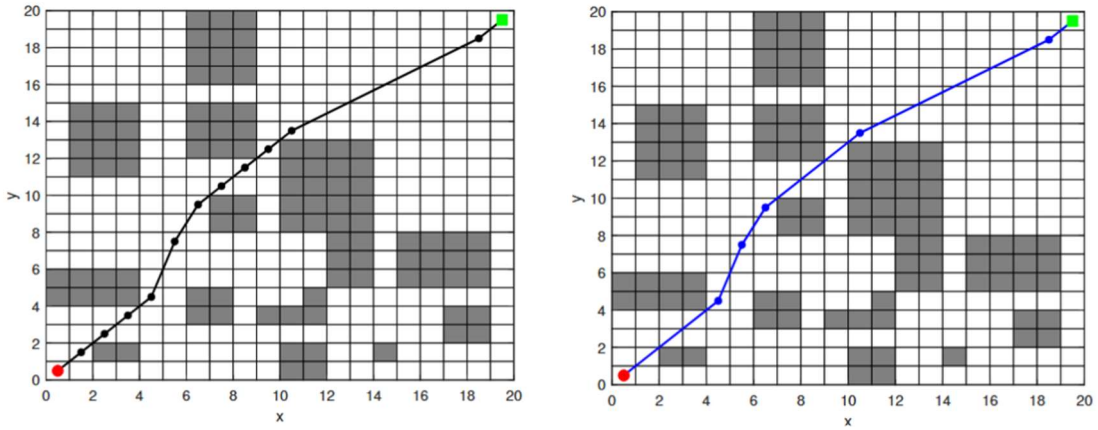
Şekil 4.14. IACO-IABC ve IABC algoritmalarının Ortam 10 için planladığı yollar: **(a)** IACO-IABC, **(b)** IABC

Tablo 4.9. IABC ile ABC ve [82]'deki algoritmaların yol uzunluğu (br) karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

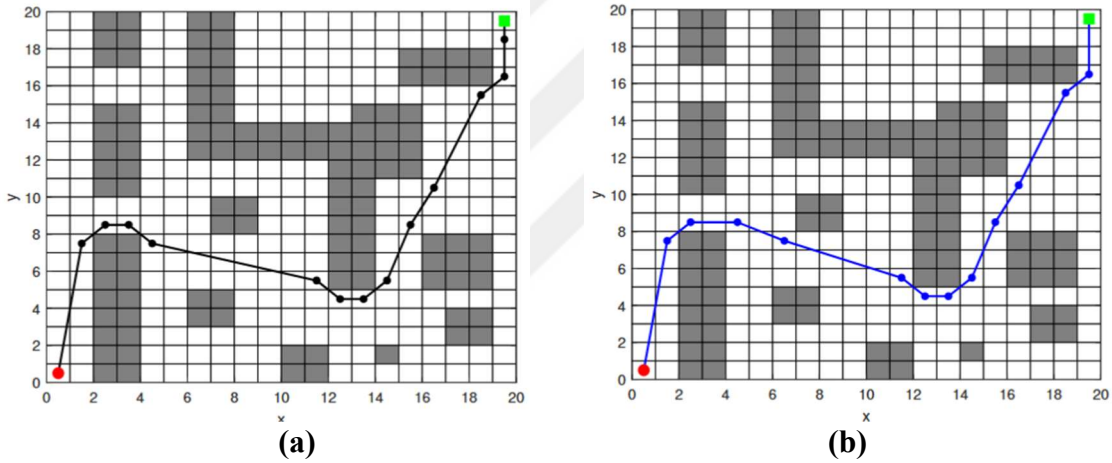
Algoritma	Ortam 9			Ortam 10		
	OP	OR	SS	OP	OR	SS
ABC	29.77	30.11	0.27	31.20	31.57	0.30
ACO	28.62	31.62	2.44	30.38	32.09	1.39
WOA	30.00	30.93	0.76	32.78	33.15	0.30
LAPO	30.02	30.52	0.40	32.77	33.25	0.39
SSA	30.36	31.17	0.66	30.84	30.88	0.03
GWO	30.11	30.56	0.37	31.84	31.89	0.04
BWO	32.23	32.52	0.23	33.37	33.46	0.06
BWOA	30.23	30.61	0.31	31.10	31.28	0.15
IACO-IABC	31.40	31.40	0.00	30.89	30.89	0.00
IABC	27.73	28.52	0.45	29.04	29.33	0.22

Ortam 9 için IABC ile planlanan yolların en iyi uzunlukları Tablo 4.10'daki sıraya göre sırasıyla %6.85, %3.12, %7.56, %7.64, %8.65, %7.88, %13.95, %8.27 ve %11.68 oranında kısaltılmıştır. Ortalama uzunluklarda sırasıyla %5.29, %9.8, %7.79, %6.56, %8.49, %6.68, %12.3, %6.83 ve %9.16 oranında iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi IABC bazı yöntemlere göre biraz daha kararsızdır. Ancak bu kararsızlık bu yöntemlerden daha iyi olan en iyi-ortalama bandındadır. En kararsız algoritma ACO'dur. Ortam 10 için IABC ile planlanan yolların en iyi uzunlukları sırasıyla %6.91, %4.39, %11.38, %11.35, %5.82, %8.77, %12.98, %6.6 ve %5.97 oranında kısaltılmıştır. Ortalama uzunluklarda sırasıyla %7.08, %8.57, %11.52, %11.79, %5.02, %8.01, %12.32, %6.23 ve %5.04 oranında iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi IABC'nin diğer algoritmalarla yaklaşık olarak aynı kararlılık seviyesinde olduğu söylenebilir. En kararsız algoritmanın yine ACO olduğu görülmektedir.

[83]'teki algoritmalar ACO, ACO ve GA'nın bir hibriti (ACO+GA), SSA ve [83]'te önerilen algoritma olan iyileştirilmiş SSA (improved SSA, ISSA) algoritmasıdır. Adil bir karşılaştırma için, ABC ve IABC'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 200 ve 50 olarak ayarlanmıştır. ABC ve IABC'nin limit değerleri 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. [83]'te tasarlanan ortamlar (Ortam 11 ve 12) ile ISSA ve bu çalışmada önerilen IABC tarafından planlanan yollar Şekil 4.15 ve 4.16'da gösterilmektedir. Bu ortamlarda başlangıç noktası sol alt köşe (1, 1) ve hedef noktası sağ üst köşedir (20, 20). Algoritmaların yol uzunluğu karşılaştırması Tablo 4.10'da gösterilmektedir.



Şekil 4.15. ISSA ve IABC algoritmalarının Ortam 11 için planladığı yollar: **(a)** ISSA, **(b)** IABC



Şekil 4.16. ISSA ve IABC algoritmalarının Ortam 12 için planladığı yollar: **(a)** ISSA, **(b)** IABC

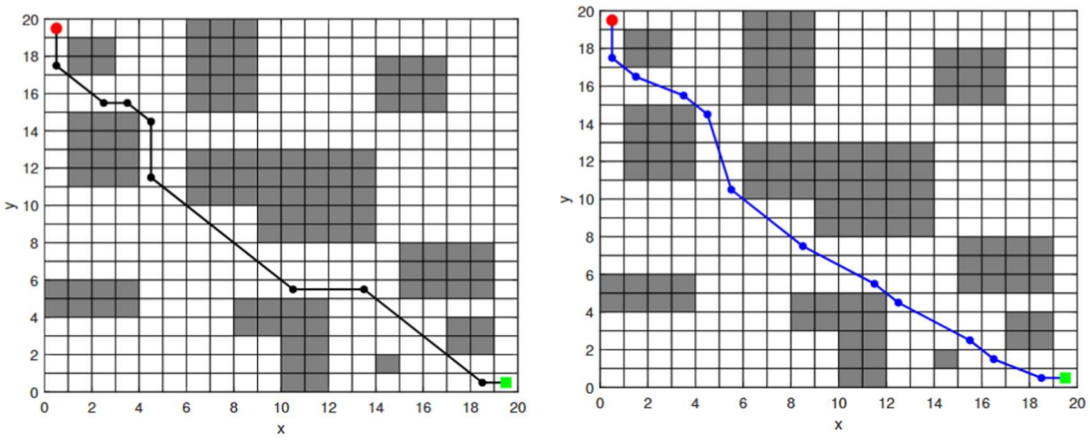
Tablo 4.10. IABC ile ABC ve [83]'teki algoritmaların yol uzunluğu (br) karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Algoritma	Ortam 11			Ortam 12		
	OP	OR	SS	OP	OR	SS
ABC	29.21	33.21	1.95	38.97	40.91	1.38
ACO	29.21	31.81	2.02	39.79	45.45	3.06
ACO+GA	28.62	30.70	1.43	38.97	40.78	1.13
SSA	29.21	32.60	1.24	38.97	42.31	3.25
ISSA	27.56	27.89	0.61	37.13	37.26	0.12
IABC	27.56	27.64	0.15	37.13	37.18	0.06

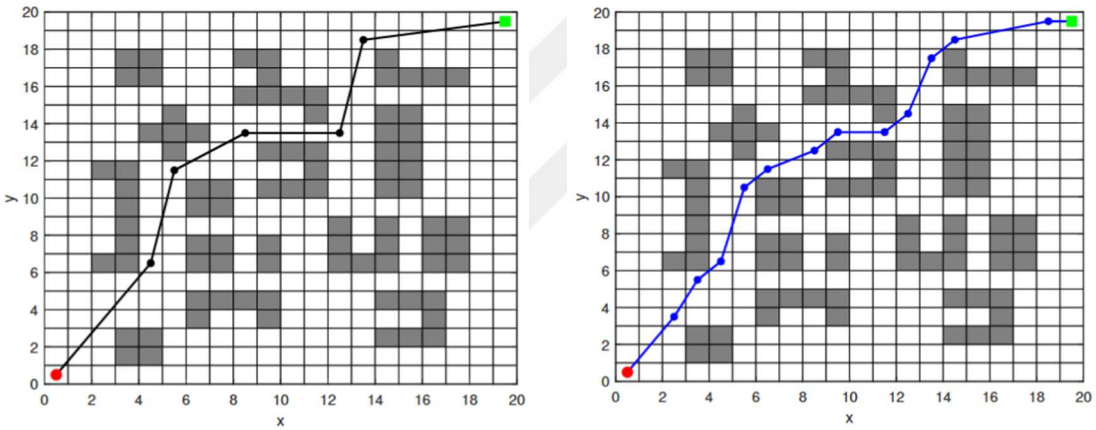
Ortam 11 için, IABC ile planlanan yolların en iyi uzunlukları ABC, ACO ve SSA ile karşılaştırıldığında %5.65 ve ACO+GA ile karşılaştırıldığında %3.73 oranında kısaltılmıştır. IABC, ISSA ile aynı sonucu üretmiştir. Ortalama uzunlukta ABC ile

karşılaştırıldığında %16.76, ACO ile karşılaştırıldığında %13.11, ACO+GA ile karşılaştırıldığında %9.97, SSA ile karşılaştırıldığında %15.2 ve ISSA ile karşılaştırıldığında %0.89 oranında bir iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi, en kararlı algoritma IABC iken, en kararsız algoritma ACO'dur. Ortam 12 için, IABC ile planlanan yolların en iyi uzunlukları ACO ile karşılaştırıldığında %6.7 ve ABC, ACO+GA ve SSA ile karşılaştırıldığında %4.72 oranında kısaltılmıştır. IABC, yine ISSA ile aynı sonucu üretmiştir. ABC ile karşılaştırıldığında ortalama uzunlukta %9.10, ACO ile karşılaştırıldığında %18.19, ACO+GA ile karşılaştırıldığında %8.82 ve SSA ile karşılaştırıldığında %12.12 ve ISSA ile karşılaştırıldığında %0.21 iyileşme gözlemlenmiştir. IABC, ISSA ile yaklaşık olarak aynı kararlılık düzeyine sahiptir, diğer algoritmalar daha kararsızdır. Genel olarak, önerilen IABC diğer algoritmalardan daha iyi performans göstermiştir. ISSA ile aralarında ufak bir fark olmasına rağmen, yine de ondan üstün olduğu söylenebilir. Bu sonuçlar, IABC'nin yol uzunluğu açısından diğer metasezgisellerden üstün olduğunu kanıtlamaktadır.

[84] ve [85]'teki algoritmalar dijkstra, A*, [84]'te önerilen iyileştirilmiş sezgisel mekanizmalı ACO (improved heuristic mechanism ACO, IHMACO) ve [85]'te önerilen iyileştirilmiş A* (improved A*, IA*) algoritmasıdır. Adil bir karşılaştırma için, ABC ve IABC'nin maksimum iterasyon sayısı ve popülasyon boyutu sırasıyla 200 ve 30 olarak ayarlanmıştır. ABC ve IABC'nin limit değerleri 100 olarak ayarlanmış ve algoritmalar bağımsız olarak 30 kez çalıştırılmıştır. [84] ve [85]'te tasarlanan ortamlarda (Ortam 13 ve 14) IHMACO, IA* ve bu çalışmada önerilen IABC tarafından planlanan yollar Şekil 4.17 ve 4.18'de gösterilmektedir. Ortam 13'te başlangıç noktası sol üst köşe (1, 20) ve hedef noktası sağ alt köşedir (20, 1). Ortam 14'te başlangıç noktası sol alt köşe (1, 1) ve hedef noktası sağ üst köşedir (20, 20). Algoritmaların yol uzunluğu karşılaştırması Tablo 4.11'de gösterilmektedir.



Şekil 4.17. IHMACO ve IABC algoritmalarının Ortam 13 için planladığı yollar: **(a)** IHMACO, **(b)** IABC



Şekil 4.18. IA* ve IABC algoritmalarının Ortam 14 için planladığı yollar: **(a)** IA*, **(b)** IABC

Tablo 4.11. IABC ile ABC, [84] ve [85]'teki algoritmaların yol uzunluğu (br) karşılaştırması (OP: Optimum, OR: Ortalama, SS: Standart Sapma)

Algoritma	**Ortam 13 [84]**			**Ortam 14 [85]**		
	OP	**OR**	**SS**	**OP**	**OR**	**SS**
ABC	30.97	33.58	2.10	30.97	35.82	3.23
Dijkstra	29.79	29.79	0.00	32.72	32.72	0.00
A*	29.79	29.79	0.00	32.09	32.09	0.00
IHMACO	29.79	29.79	0.00	-	-	-
IA*	-	-	-	31.01	31.01	0.00
IABC	28.70	29.01	0.68	29.55	30.09	0.47

Ortam 13 için, IABC ile planlanan yolların en iyi uzunlukları ABC'ye kıyasla %7.33, dijkstra, A* ve IHMACO'ya kıyasla %3.66 oranında kısaltılmıştır. Ortalama uzunlukta ABC'ye kıyasla %7.33, dijkstra, A* ve IHMACO'ya kıyasla %2.62 oranında bir iyileşme

gözlemlenmiştir. Ortam 14 için, IABC ile planlanan yolların en iyi uzunlukları ABC, dijkstra, A* ve IA*'a göre sırasıyla %4.59, %9.69, %7.92 ve %4.71 oranında kısaltılmıştır. Ortalama uzunluklarda sırasıyla %16.01, %8.03, %6.23 ve %2.97 oranında bir iyileşme gözlemlenmiştir. Standart sapmalardan da görülebileceği gibi, dijkstra ve A* gibi klasik algoritmalar rastgelelik ve keşif mekanizmalarını içermediğinden, tüm çalışmalarda aynı sonucu üretmeleri ve standart sapmanın sıfır olması kaçınılmazdır. Önerilen IABC rastgelelik içerdiğinden, her çalışma farklı sonuçlar üretir ve bu standart sapmaya yansır. Ancak, en iyi ve ortalama sonuçlar IABC'nin klasik arama algoritmalarından daha verimli olduğunu göstermektedir. Sonuç olarak, önerilen algoritma ızgara tabanlı yol planlama problemi için etkili bir yöntem olabilir.

4.3. Çok Engelli Ortamlarda Hızlı Yol Planlama

Literatürde birçok yol planlama algoritması geliştirilmesine rağmen, bu algoritmalar karmaşık ve çok engelli ortamlarda uygulandığında uzun çalışma sürelerine ihtiyaç duymaktadır. Bunun için bu algoritmaların çalışma hızlarının artırılması gerekmektedir. Ancak ihtiyaç duyulan hızlanma genellikle algoritma tarafında çözülmeye çalışılmaktadır. Daha hızlı algoritmalar geliştirilmekte veya mevcut algoritmalar iyileştirilerek karmaşıklıkları azaltılmaktadır. Bu çalışmada ise hızlanmanın problem tarafında çözülmesine odaklanılmış ve problemin basitleştirilmesi üzerinde durulmuştur. Bunun için engellerin kümelenmesiyle ortam karmaşıklığının azaltılması ve bu sayede yol planlama algoritmalarının çalışma hızlarının artırılması amaçlanmıştır. Bu amaçlar doğrultusunda metasezgisel ve kümeleme algoritmalarının bir arada kullanıldığı hibrit bir model önerilmiştir. Öncelikle PSO ve k-ortalamalar kümeleme algoritmaları ile önerilen modelin detaylı analizi gerçekleştirilmiştir. Ardından, önerilen modelin etkinliği TLBO, ABC, DE, GA ve hiyerarşik kümeleme algoritmalarının kullanılması ile karşılaştırmalı olarak değerlendirilmiştir.

4.3.1. Problem Tanımı

Bu çalışmadaki yol planlama kübik eğri interpolasyonu (cubic spline interpolation) ile gerçekleştirilmiştir. Kübik eğri interpolasyonunun uygulanabilmesi için belli noktalara ihtiyaç duyulur. PSO algoritmasındaki çözümlere karşılık gelen bu noktalar parametre noktaları olarak, noktaların sayısı da parametre sayısı (D) olarak ifade edilir. Öncelikle, amaç fonksiyonuna girdi olarak gelen bu parametre noktalarının x ve y konumları,

planlanacak yolun başlangıç ve bitiş noktaları ile birlikte farklı satır vektörlerine aktarılır. Bu vektörler n_y adet arama noktasına sahip kübik eğri interpolasyonu ile ayrı ayrı interpole edilir. Bu şekilde planlanan yolun x ve y nokta dizileri elde edilmiş olur, burada x dizisindeki her bir nokta p_{ix}, y dizisindeki her bir nokta p_{iy} olarak ifade edilir. Daha sonra, parametre noktalarının uygunluk değerleri Eşitlik (4.13)'te gösterilen amaç fonksiyonu ile hesaplanır. Bu denklem iki kısımdan oluşmaktadır. Birinci kısım robotun gideceği yolun uzunluğunu hesaplarken, ikinci kısım da robot ile engeller arasındaki uygulanabilir mesafeyi (engelden kaçınma kontrolü) hesaplar [86].

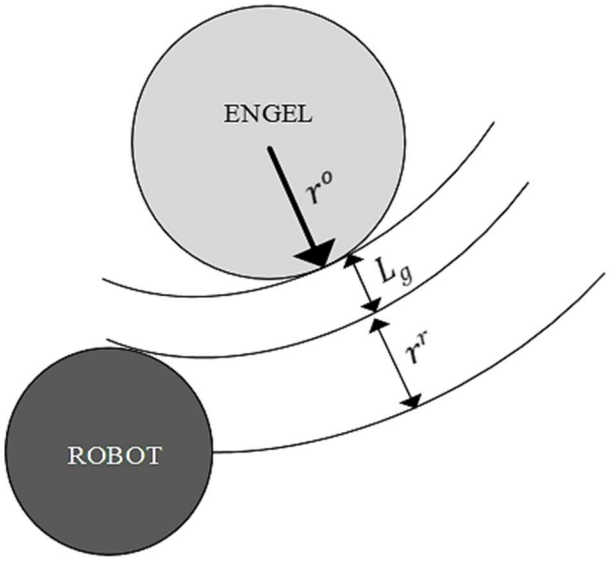
$$\arg\min_{p_i \in Y} \mathcal{F} = L(1 + \beta V) \tag{4.13}$$

Burada, $i \in \{1,2,\dots,n_y\}$, Y planlanan yolu, p_i bu yolun i'inci noktasını, L yol uzunluk fonksiyonunu, β engel ihlal faktörünü, V engelden kaçınma fonksiyonunu temsil eder. Yol uzunluğu Eşitlik (4.14)'te ve engelden kaçınma fonksiyonu Eşitlik (4.15)'te gösterilmektedir.

$$L = \sum_{i=1}^{n_y - 1} \|p_{i+1} - p_i\| \tag{4.14}$$

$$V = \sum_{i=1}^{n_y} \sum_{k=1}^{n_o} \begin{cases} \left(1 - \dfrac{\|p_i - p_k^o\|}{r_k^o}\right), & eğer \left(1 - \dfrac{\|p_i - p_k^o\|}{r_k^o}\right) > 0 \; ise \\ 0, & aksi \; halde \end{cases} \tag{4.15}$$

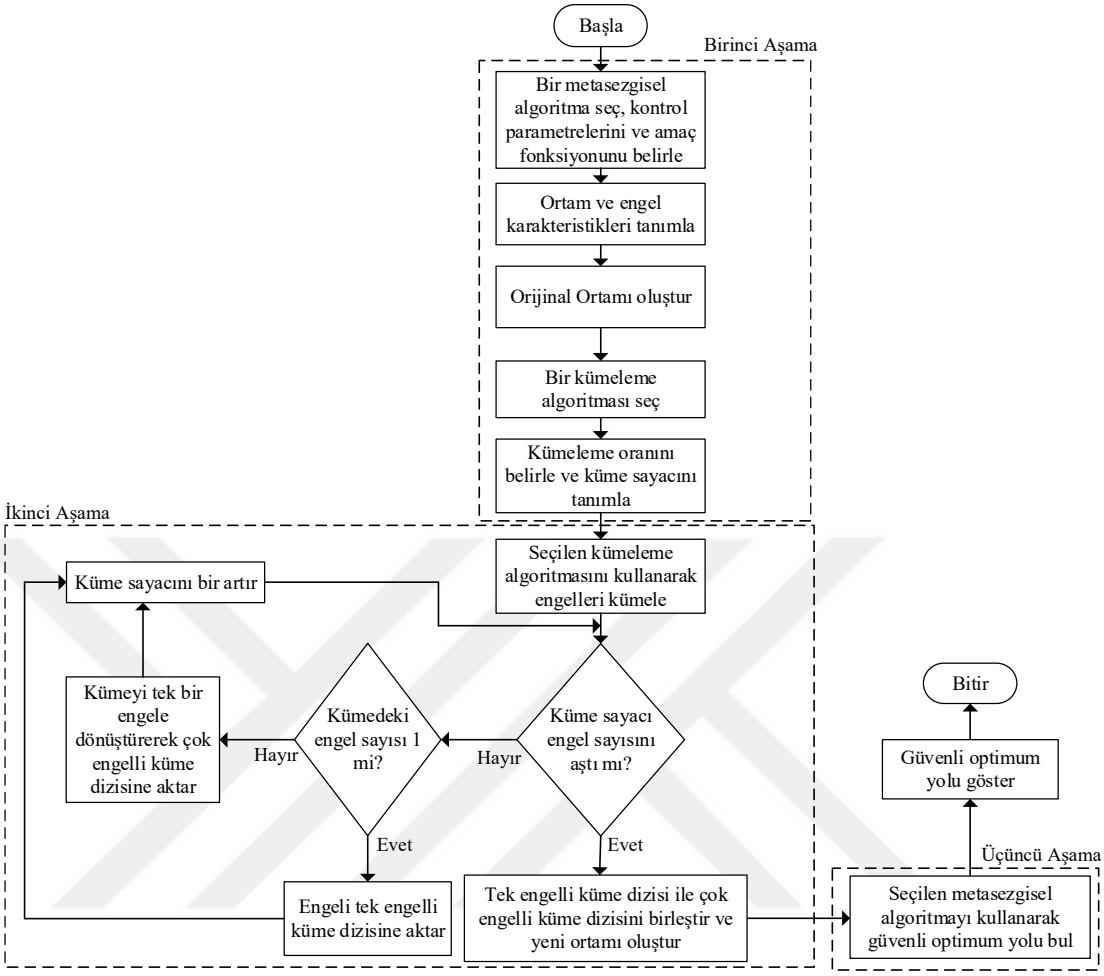
Burada, n_o engel sayısını, p_i yolun i'inci noktasını, p_k^o k'ıncı engelin merkez noktasını ve r_k^o ise k'ıncı engelin yarıçapını temsil eder. Bu çalışmada mobil robot ve engeller dairesel olarak tasarlanmıştır. Engelden kaçınma kontrolünde sadece engelin yarıçapı değil (r^o), aynı zamanda yarıçapı boyutu (r^r) ve güvenlik mesafesi (L_g) de hesaba katılmıştır. Güvenlik mesafesi robot ile engel arasındaki boş alanı ifade eder. Engelden kaçınma kontrolünün temsili çizimi Şekil 4.19'da gösterilmektedir.



Şekil 4.19. Engelden kaçınma kontrolünün temsili çizimi

4.3.2. Önerilen Yöntem

Bu çalışmada yol planlama algoritmalarının çalışma süreleri üzerinde durulmuş ve metasezgisel ile kümeleme algoritmalarının bir arada kullanıldığı hibrit bir model önerilmiştir. Bu model üç aşamadan oluşur: İlk aşamada, kullanılan metasezgisel algoritmanın kontrol parametreleri ile kümeleme oranı belirlenir, ortam ve engel karakteristikleri tanımlanarak orijinal ortam oluşturulur. İkinci aşama olarak, bir kümeleme yöntemine dayanarak geliştirilen bir engel kümeleme algoritması çalıştırılır. Bu algoritma ile orijinal ortamdaki engeller belli sayılarda kümelenir. Kümelenen engeller tek bir engel olarak tanımlanır. Böylece ortam karmaşıklığı azaltılır ve yol planlama için yeni bir ortam oluşturulur. Üçüncü aşamada ise söz konusu metasezgisel algoritmanın kullanıldığı bir yol planlama simülasyonu ile bu yeni ortamda optimum yol planlanır. Önerilen modelin akış diyagramı Şekil 4.20'de gösterilmektedir.



Şekil 4.20. Önerilen modelin akış diyagramı

4.3.2.1. K-Ortalamalar Kümeleme Algoritması

K-ortalamalar kümeleme algoritması (K-means clustering, KMC) verilerin belli bir kritere göre otomatik olarak gruplandırılması için yaygın olarak kullanılan ve iteratif olarak çalışan denetimsiz bir öğrenme tekniğidir. Bu kümeleme yöntemindeki temel prensip, küme sayısının ve küme merkezlerinin belirlenerek birbirine benzerlik gösteren verilerin aynı kümelere yerleştirilmesidir. Küme merkezlerinin birbirine olabildiğince uzak konumlarda belirlenmesi gerekir. Küme sayısının belirlenmesi bu yöntemin kilit noktasıdır. Algoritma öncelikle küme sayısınca rastgele merkezler oluşturur ve her bir verinin küme merkezleri arasındaki mesafe Eşitlik (4.16) kullanılarak hesaplanır ve bu mesafe aracılığıyla veriler en yakın kümeye atanır.

$$L_i = \left\| p_i - q_j \right\| \tag{4.16}$$

Burada, $i \in \{1,2,\dots,n_v\}$, $j \in \{1,2,\dots,n_q\}$, p_i i'inci veriyi, q_j j'inci küme merkezini, L_i ise i'inci veri ile j'inci küme merkezi arasındaki Öklidyen mesafeyi, n_v veri sayısını ve n_q ise küme sayısını temsil eder. Öklidyen mesafesi yerine farklı mesafe denklemleri de kullanılabilir. Veriler en yakın kümelere atandıktan sonra Eşitlik (4.17) kullanılarak her kümenin ortalaması alınır ve böylece yeni küme merkezleri oluşturulur.

$$q_j = \frac{1}{|q_j|} \sum_{p \in q_j} p \tag{4.17}$$

Burada, $|q_j|$ j'inci kümedeki veri sayısını ve p ise j'inci kümedeki verileri temsil eder. Bu yeni küme merkezleri ve veriler arasındaki mesafe tekrar hesaplanır. Bu iteratif süreç tüm verilerin en yakın merkezlere atanmasıyla (durdurma kriteri) son bulur. KMC algoritmasının sözde kodu Algoritma 4.4'te gösterilmektedir [87-90].

Algoritma 4.4: KMC algoritmasının sözde kodu

Girdi: p, n_q, T, x_l, x_h // Veri, küme sayısı, maksimum iterasyon sayısı, arama sınırları
Çıktı: $\mathbb{Q}$ // Küme merkezleri

$n_v \leftarrow size(p)$
$\{q_1, q_2, \dots, q_k\} \subset \mathbb{Q} \leftarrow x_l + r(x_h - x_l),\ \ r \sim U(0,1)$
for $t = 1:T$
  for $i = 1:n_v$
    for $j = 1:n_q$
      $L_i \leftarrow \|p_i - q_j\|$
    end for
    $q_i^m \leftarrow \arg\min_{q \in \mathbb{Q}} L_i(q)$
    $q_i^m \leftarrow q_i^m \cup \{p_i\}$
  end for
  for $j = 1:n_q$
    $q_j \leftarrow \frac{1}{|q_j|} \sum_{(p \in q_j)} p$
  end for
end for

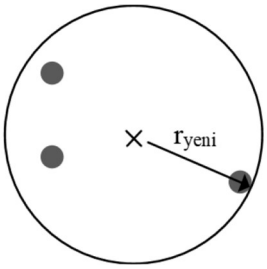
4.3.2.2. Engel Kümeleme Algoritması

Bu çalışmada çeşitli kümeleme algoritmalarının kullanılabildiği bir engel kümeleme algoritması önerilmiştir. Engeller dairesel oldukları için gruplandırma da dairesel olarak tasarlanmıştır. Detaylı analizde kümeleme işlemi KMC algoritması ile gerçekleştirilmiştir. Bu yöntem için küme sayısı gerekli olsa da, bu çalışma yol planlama algoritmalarının çalışma hızlarını farklı küme sayıları ile analiz eder. Bu sebeple küme

sayısını değiştiren bir oran oluşturulmuştur. Çalışmada bu oran, % biriminde kümeleme oranı (λ) olarak ifade edilir ve engellerin yüzde kaçının kümeleneceğini temsil eder. Önerilen modelde birbirine yakın engellerin kümelenmesi esas alınır. Her engel kümelenmediği için tek engelli kümeler oluşabilir. Kümeleme oranı arttıkça kümelerin içindeki engel sayısı her zaman artar, ancak küme sayısı belli bir seviyeye kadar artış gösterir. Bu seviyeden daha yüksek kümeleme oranlarında tek engelli küme kalmaz ve kümelenecek engel sayısı arttığı için daha büyük kümeler oluşur. Bu durumda küme sayısı azalır ve kümelerde çakışma durumu meydana gelir. Ancak algoritma bu kümeleri yeni statik engeller olarak belirlediği için yol planlama konusunda herhangi bir zorluğa neden olmaz. Bu sayede engel sayısı (n_o) ve kümeleme oranı kullanılarak KMC algoritması için gerekli olan küme sayısı hesaplanır. Küme sayısı Eşitlik (4.18) kullanılarak hesaplanır.

$$n_q = n_o \times \lambda \tag{4.18}$$

Birden fazla engelin kümelenmesinde Şekil 4.21'de olduğu gibi merkeze en uzak engelin tamamını kapsayacak şekilde $r^{o\prime}$ yarıçaplı bir daire oluşturulur [91].



Şekil 4.21. Birden fazla engelin kümelenmesi

Önerilen bu algoritma, bu daireleri ve tek engelli kümeleri yeni bir engel dizisine aktarır ve yeni engel dizisini esas alarak çalışır. Bu şekilde engel sayısı azaltılarak ortam karmaşıklığının basitleştirilmesi sağlanır. Bu çalışmadaki KMC tabanlı engel kümeleme algoritmasının sözde kodu Algoritma 4.5'te gösterilmektedir. Bu algoritmada r^o engellerin yarıçapını, O eski engel dizisini ve O' ise yeni engel dizisini temsil eder.

Algoritma 4.5: KMC tabanlı engel kümeleme algoritmasının sözde kodu

Girdi: λ, O, T, x_l, x_h // Kümeleme oranı, eski engel dizisi, maksimum iterasyon sayısı, arama sınırları

Çıktı: O' // Yeni engel dizisi

$n_o \leftarrow size(O)$
$n_q \leftarrow n_o \times \lambda$
$q_i \in \mathbb{Q} \leftarrow \text{KMC}\left(O, n_q, T, x_l, x_h\right), \quad i \in \{1,2,\dots,n_q\}$ // Algoritma 4.4
for $i = 1{:}\,n_q$
    $n_{oi} \leftarrow size(q_i)$
    if $n_{oi} == 1$
        $O_{tek} \leftarrow \{p_o, r^o\}$
    else
        $O_{çok} \leftarrow \{q_i\}$
        for $j = 1{:}\,n_{oi}$
            $L_i(j) \leftarrow \|q_i - p_j^o\|$
        end for
        $L_{max}, r_{max}^o \leftarrow \arg\max_i(L_i)$
        $r^{o'} \leftarrow L_{max} + r_{max}^o$
        $O_{çok} \leftarrow O_{çok} \cup \{r^{o'}\}$
    end if
end for
$O' \leftarrow \{O_{tek}, O_{çok}\}$

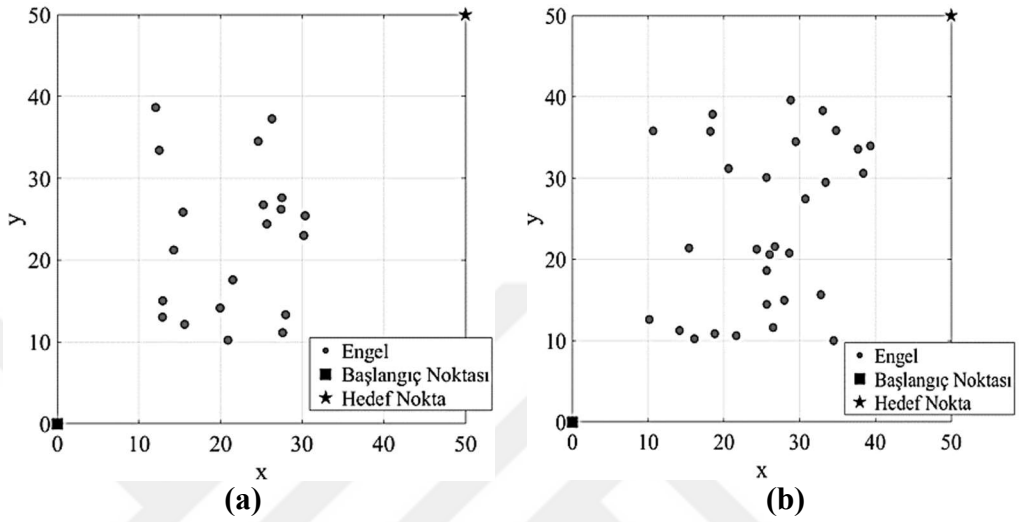
4.3.3. Bulgular

Önerilen model, MATLAB 2019 programlama dilinde kodlanmış ve Windows 10 işletim sistemi, INTEL CORE i7 işlemcisi, 16 GB RAM'e sahip bir bilgisayarda çalıştırılmıştır. Öncelikle önerilen modelin detaylı analizi gerçekleştirilmiştir. Bu analiz sonucunda yol planlama algoritmalarının optimum çalışma hızlarına tekabül eden optimum kümeleme oranları elde edilmiştir. Ardından, detaylı analizi desteklemek amacıyla model üzerinde farklı metasezgisel ve farklı kümeleme algoritmalarının performansları değerlendirilmiştir.

4.3.3.1. Önerilen Modelin Detaylı Analizi

Önerilen model engellerin rastgele konumlarda üretildiği ortamlarda test edilmiştir. Bunun için 20 ve 30 engelli olmak üzere 50 x 50 br^2 boyutlu sürekli uzay formatında iki farklı ortam oluşturulmuştur. Ancak aynı alan içerisinde engel sayısı arttıkça modelin problemi çözmesi zorlaşmaktadır. Bu çalışmada daha fazla engelli ortamlar oluşturulsa da, 30 engelin üzerindeki ortamlar için orta ve yüksek seviyelerdeki kümeleme oranlarında alan darlığından dolayı model problemi çözememektedir. Bunun için alanın

genişletilmesi gerekir, ancak bu çalışmada önerilen model sabit alan üzerinde test edilmiştir. 20 engelin altındaki ortamlar ise değerlendirme için anlamlı değildir. Bu çalışmada tasarlanan bu iki ortam Şekil 4.22'de gösterilmektedir. Bu ortamlarda robotun başlangıç konumu sol alt köşe (0, 0) ve hedef konumu ise sağ üst köşedir (50, 50).



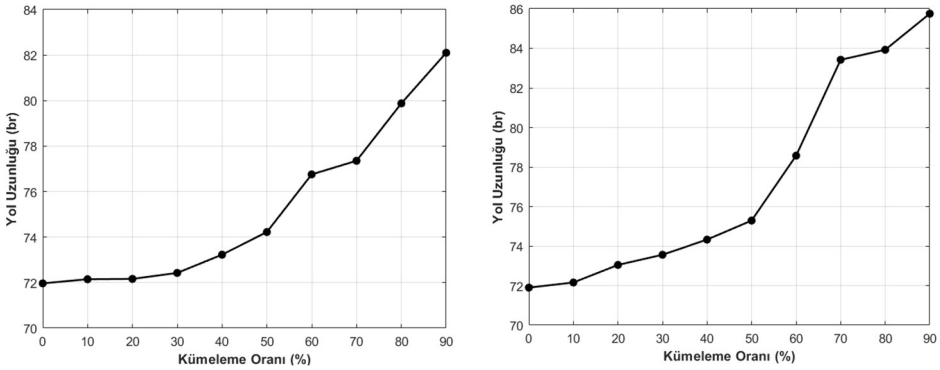
Şekil 4.22. Önerilen modelin test edildiği ortamlar: **(a)** Ortam 15 (20 engelli ortam), **(b)** Ortam 16 (30 engelli ortam)

Bu ortamlarda engelsiz en kısa mesafe 70.71 br'dir. Amaç fonksiyonundaki engel ihlali faktörü (β) ve kübik eğri interpolasyonu için arama noktası sayısı (q) 100 olarak, arama alanının sınır değerleri x ve y için [0 50] br olarak, engel konumlarının sınır değerleri x ve y için [10 40] br olarak belirlenmiştir. Engellerin yarıçapı 0.5 br, robotun yarıçapı 0.3 br ve güvenlik mesafesi 0.2 br olarak ayarlanmıştır. Detaylı analiz için metasezgisel algoritmalardan PSO ve kümeleme algoritmalarından KMC algoritması tercih edilmiştir. PSO algoritması maksimum iterasyon sayısı 50 ve popülasyon boyutu 20 için çalıştırılmıştır. Kişisel ve sosyal deneyim katsayıları [2, 2], eylemsizlik ağırlığının başlangıç değeri 1, eylemsizlik ağırlığının sınır değerleri [0.1 0.9] olarak belirlenmiştir. Problemdeki parametre sayısı (D) 2, 3, 4, 5, 6 ve 8 için ayrı ayrı test edilmiş ve en uygun değerinin 2 olduğu gözlemlenmiştir. Bu sebeple bu çalışmadaki tüm simülasyonlarda parametre sayısı 2 olarak ayarlanmıştır. Detaylı analiz için önerilen model kümelemesiz ve kümelemeli olarak 30 koşma ile çalıştırılmıştır. Kümelemeli çalışmada 9 farklı kümeleme oranı kullanılmıştır. Ayrıca model çalışma süresi bakımından, engel kümeleme algoritmasının (EKA) çalışma süresinin dâhil olduğu ve olmadığı durumlar için ayrı ayrı değerlendirme gerçekleştirilmiştir. Her iki ortam için elde edilen ortalama

yol uzunlukları ve bu uzunlukların kümelemesiz çalışmaya göre artış oranları Tablo 4.12'de gösterilmektedir. Şekil 4.23, Tablo 4.12'deki yol uzunluğu değerlerini grafiksel olarak göstermektedir. Şekil 4.23'te kümeleme oranlarındaki 0 değerleri kümelemesiz çalışmayı temsil eder.

Tablo 4.12. Her iki ortam için elde edilen ortalama yol uzunlukları ve bu uzunlukların kümelemesiz çalışmaya göre artış oranları (Bu bulgular 30 koşmanın ortalamasıdır.)

Çalışma	**Kümeleme Oranı (%)**	**Ortam 15**		**Ortam 16**	
		Yol Uzunluğu (br)	**Artış Oranı (%)**	**Yol Uzunluğu (br)**	**Artış Oranı (%)**
Kümelemesiz	-	71.96	-	71.89	-
Kümelemeli	10	72.15	0.25	72.15	0.36
	20	72.16	0.27	73.04	1.59
	30	72.42	0.63	73.56	2.31
	40	73.22	1.75	74.32	3.37
	50	74.22	3.13	75.28	4.71
	60	76.75	6.65	78.57	9.28
	70	77.35	7.48	83.42	16.02
	80	79.87	10.99	83.92	16.72
	90	82.10	14.08	85.75	19.26

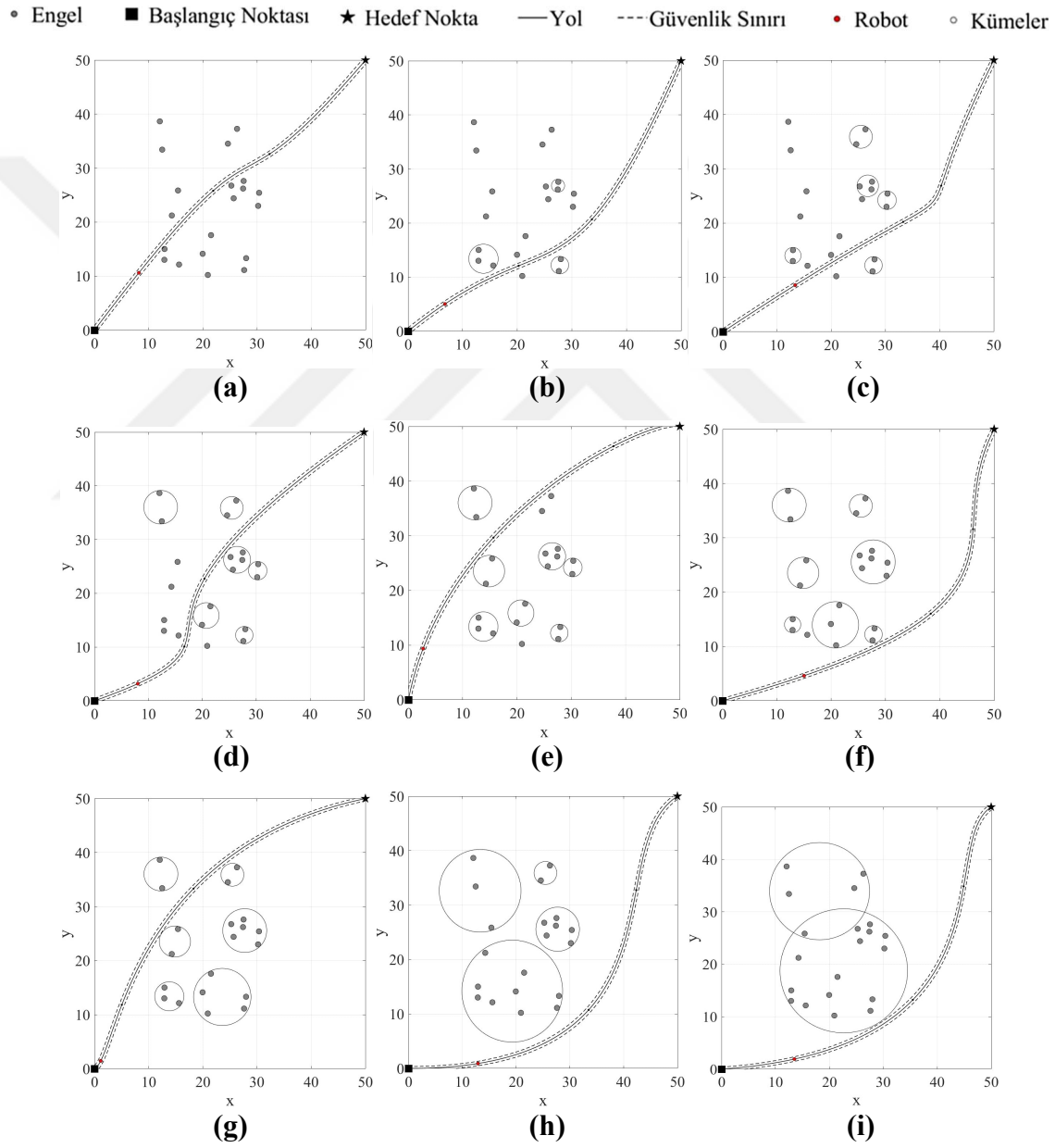


(a) **(b)**

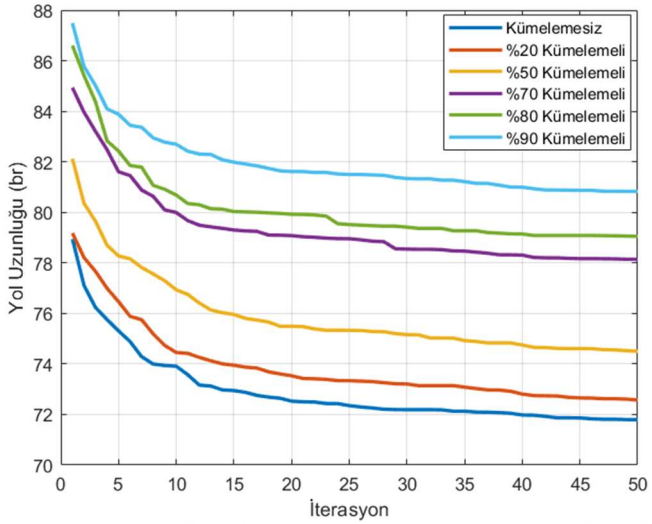
Şekil 4.23. Her kümeleme oranı için elde edilen ortalama yol uzunlukları: **(a)** Ortam 15 **(b)** Ortam 16

Tablo 4.12 ve Şekil 4.23 genel olarak incelendiğinde, kümeleme oranı arttıkça elde edilen yol uzunluklarında artış görülmektedir. Bunun nedeni, kümeleme oranının artmasıyla yeni ortamdaki engellerin boyutlarının artması ve bundan dolayı yolların daha kavisli olmasıdır. Ancak Tablo 4.12'deki artış oranları göz önüne alındığında, yüksek kümeleme oranları dışında telafi veya ihmal edilebilir bir artış mevcuttur. Ortam 15 için yüksek

kümeleme oranları %80 ve %90 olarak kabul edilirse, diğer 7 kümeleme oranında artış miktarı %10'u geçmemektedir. Ortam 16 için yüksek kümeleme oranları %70, %80 ve %90 olarak kabul edilirse, diğer 6 kümeleme oranında artış miktarı yine %10'u geçmemektedir. Çalışmadan örnek bir koşma için Ortam 15'te modelin kümelemesiz ve kümelemeli olarak planladığı yollar Şekil 4.24'te, bu örnek koşma için yakınsama eğrileri Şekil 4.25'te gösterilmektedir.

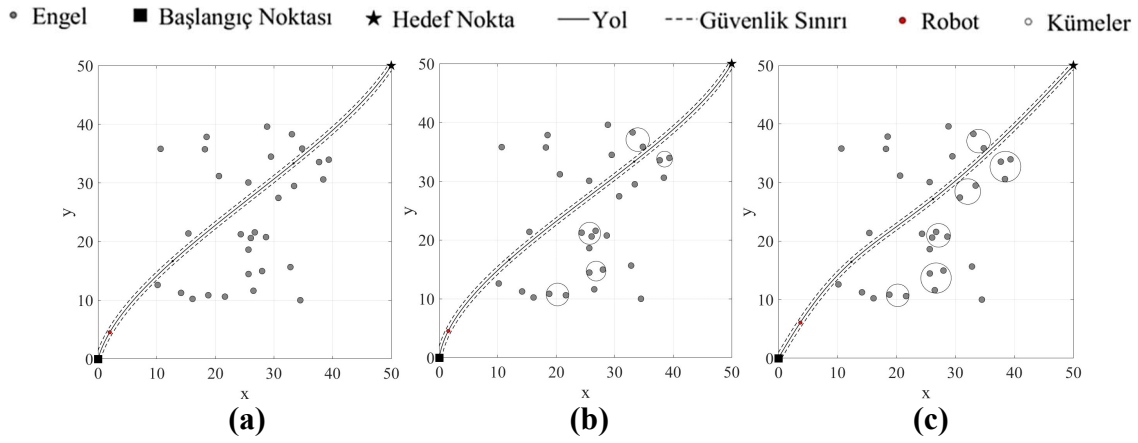


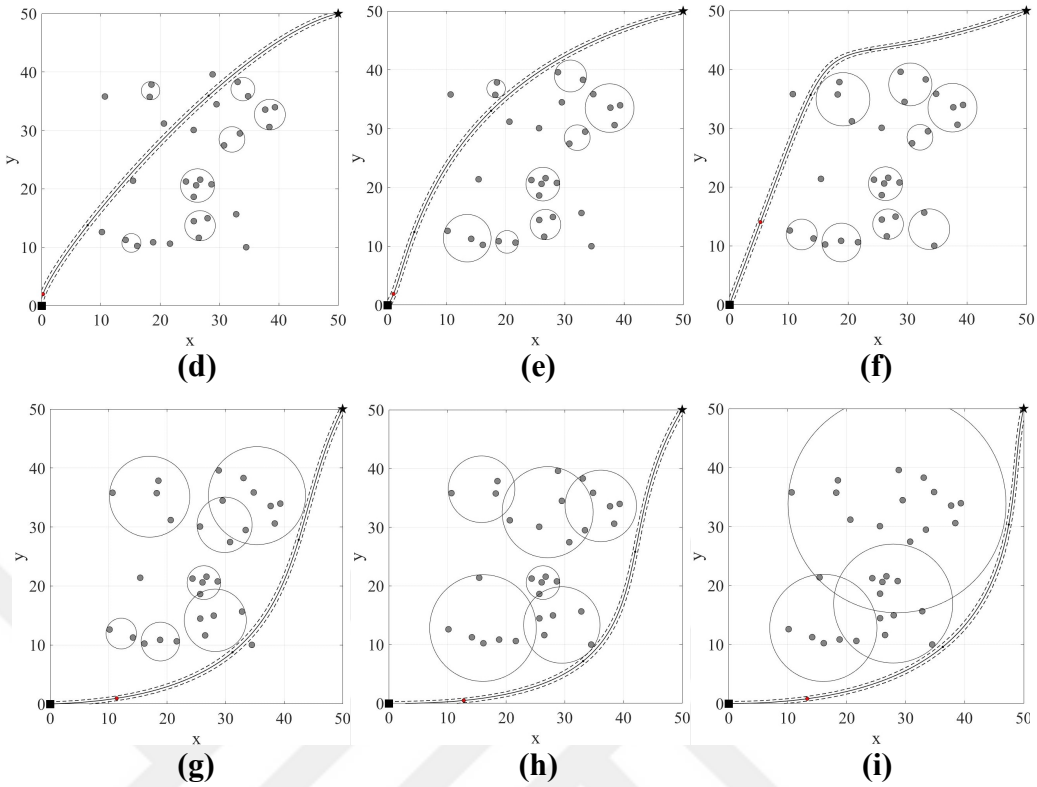
Şekil 4.24. Örnek koşma için Ortam 15'te modelin kümelemesiz ve kümelemeli olarak planladığı yollar: **(a)** Kümelemesiz **(b)** %20 kümelemeli **(c)** %30 kümelemeli **(d)** %40 kümelemeli **(e)** %50 kümelemeli **(f)** %60 kümelemeli **(g)** %70 kümelemeli **(h)** %80 kümelemeli **(i)** %90 kümelemeli



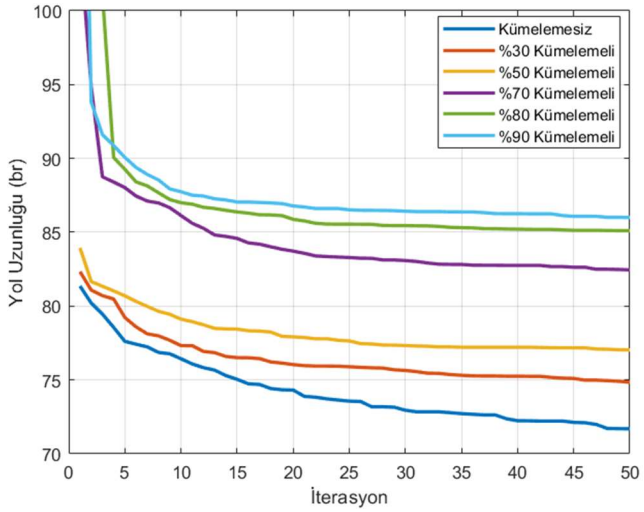
Şekil 4.25. Örnek koşma için Ortam 15'teki yakınsama eğrileri

Şekil 4.24 genel olarak incelendiğinde, kümeleme oranı arttıkça yolların daha kavisli bir şekilde elde edildiği görülmektedir. Ayrıca Şekil 4.25'te görüldüğü gibi kümeleme oranı arttıkça elde edilen yol uzunluklarının optimum değerlerinde artış görülmektedir. Ancak bu artışlar yüzdesel olarak düşük seviyelerdedir. Bazı kümeleme oranlarındaki yakınsama eğrilerinde çakışmalar tespit edildiği için bunların bir kısmı gösterilmemiştir. Örnek bir koşma için Ortam 16'da modelin kümelemesiz ve kümelemeli olarak planladığı yollar Şekil 4.26'da, bu örnek koşma için yakınsama eğrileri Şekil 4.27'de gösterilmektedir.





Şekil 4.26. Örnek koşma için Ortam 16'da modelin kümelemesiz ve kümelemeli olarak planladığı yollar: **(a)** Kümelemesiz **(b)** %20 kümelemeli **(c)** %30 kümelemeli **(d)** %40 kümelemeli **(e)** %50 kümelemeli **(f)** %60 kümelemeli **(g)** %70 kümelemeli **(h)** %80 kümelemeli **(i)** %90 kümelemeli



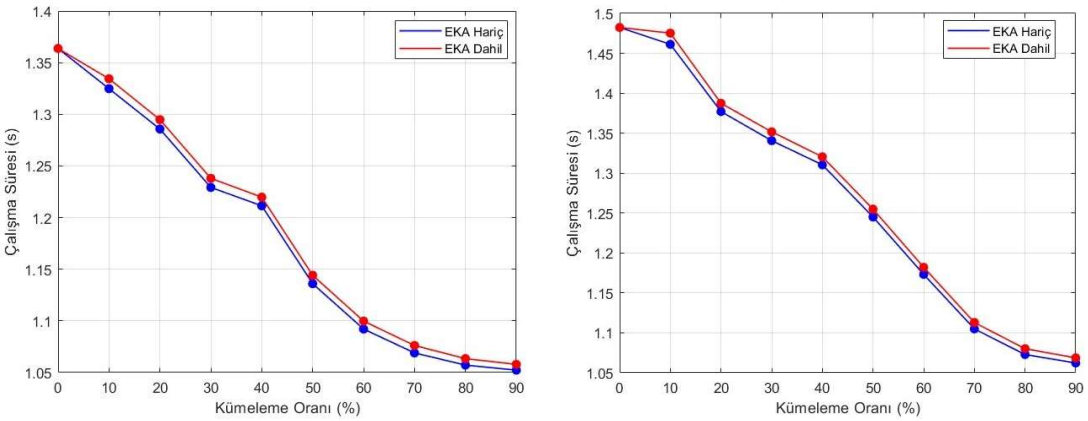
Şekil 4.27. Örnek koşma için Ortam 16'daki yakınsama eğrileri

Şekil 4.26 genel olarak incelendiğinde, kümeleme oranı arttıkça yolların Ortam 15'e göre çok daha kavisli bir şekilde elde edildiği görülmektedir. Ayrıca Şekil 4.27'de görüldüğü gibi kümeleme oranı arttıkça elde edilen yol uzunluklarının optimum değerlerinde yine

artış görülmektedir ve bu artış Ortam 15'e göre daha fazladır. Ancak bu artışlar da yüzdesel olarak düşük seviyelerdedir. Her iki ortam için elde edilen ortalama çalışma süreleri ve bu sürelerin kümelemesiz çalışmaya göre azalma oranları Tablo 4.13'te gösterilmektedir. Şekil 4.28, Tablo 4.13'teki çalışma süresi değerlerini grafiksel olarak göstermektedir. Şekil 4.28'de kümeleme oranlarındaki 0 değerleri kümelemesiz çalışmayı temsil etmektedir.

Tablo 4.13. Her iki ortam için elde edilen ortalama çalışma süreleri ve bu sürelerin kümelemesiz çalışmaya göre azalma oranları (Bu bulgular 30 koşmanın ortalamasıdır. KO: Kümeleme Oranı, ÇS: Çalışma Süresi, AO: Azalma Oranı)

Çalışma	KO (%)	Ortam 15				Ortam 16			
		EKA Hariç		EKA Dâhil		EKA Hariç		EKA Dâhil	
		ÇS (s)	AO (%)	ÇS (s)	AO (%)	ÇS (s)	AO (%)	ÇS (s)	AO (%)
Kümelemesiz	-	1,36	-	1,36	-	1,48	-	1,48	-
Kümelemeli	10	1,32	2,85	1,33	2,14	1,46	1,42	1,47	0,47
	20	1,28	5,17	1,29	5,05	1,37	7,10	1,38	6,40
	30	1,22	9,86	1,23	9,21	1,34	9,55	1,35	8,81
	40	1,21	11,16	1,21	10,55	1,31	11,60	1,32	10,92
	50	1,13	16,69	1,14	16,10	1,24	16,00	1,25	15,34
	60	1,09	19,91	1,10	19,35	1,17	20,85	1,18	20,25
	70	1,06	21,61	1,07	21,08	1,10	25,45	1,11	24,91
	80	1,05	22,47	1,06	22,01	1,07	27,62	1,08	27,13
	90	1,05	22,83	1,05	22,42	1,06	28,33	1,06	27,90



Şekil 4.28. Her kümeleme oranı için elde edilen ortalama çalışma süreleri: **(a)** Ortam 15 **(b)** Ortam 16

Tablo 4.13 ve Şekil 4.28 genel olarak incelendiğinde, kümeleme oranı arttıkça elde edilen çalışma sürelerinde azalma görülmektedir. Bunun nedeni, kümeleme oranının artmasıyla engel sayısının ve ortam karmaşıklığının azalmasıdır. Ayrıca Tablo 4.13'teki azalma

oranları göz önüne alındığında kayda değer azalma oranları elde edilmiştir ve bu oranlarda kararlılık mevcuttur. Ortam 15'teki azalma oranları Ortam 16'daki oranlara göre daha yüksektir, çünkü ortam karmaşıklığı daha düşük seviyededir. Kümelemeli çalışmalarda EKA'nın dâhil edilmesi çalışma süresini sadece milisaniyelik düzeyde artmasına sebep olduğu için azalma oranları yine kayda değer seviyede kalmıştır.

Çalışmadaki değerlendirme, yol uzunlukları ile çalışma süreleri arasındaki ilişki üzerinedir. Bu ilişki çalışma sürelerindeki azalma ile yol uzunluklarındaki artış arasındaki farktır, kazanç olarak da düşünülebilir. Bu kazanç tanımında yol uzunluğu ve çalışma süresi eşit ağırlığa sahiptir ve kazanç oranları bu şekilde hesaplanmıştır. Her iki ortam için kümeleme oranlarına göre kazanç oranları Tablo 4.14'te gösterilmektedir.

Tablo 4.14. Her iki ortam için kümeleme oranlarına göre kazanç oranları

Kümeleme Oranı (%)	**Ortam 15**		**Ortam 16**	
	EKA Hariç	**EKA Dâhil**	**EKA Hariç**	**EKA Dâhil**
10	2,60	1,89	1,06	0,11
20	4,90	4,78	5,51	4,81
30	9,23	8,58	7,24	6,50
40	9,41	8,80	8,23	7,55
50	13,56	12,97	11,29	10,63
60	13,26	12,70	11,57	10,97
70	14,13	13,60	9,43	8,89
80	11,48	11,02	10,90	10,41
90	8,75	8,34	9,07	8,64

Tablo 4.14 genel olarak incelendiğinde, kümeleme oranı arttıkça kazanç oranlarında önce bir artış ve daha sonra bir azalma görülmektedir. Orta seviyelerdeki kümeleme oranlarında maksimum kazanç elde edilirken, daha düşük veya daha yüksek kümeleme oranlarında bu kazanç oranları düşmektedir. Ortam 15'te %70 kümeleme oranında maksimum kazanç değeri elde edilirken, Ortam 16 için bu oran %60'tır. EKA'nın dâhil edilmesi çalışma süresini etkilediği gibi kazanç oranlarını da sadece minimal düzeyde etkilemiştir. Ayrıca model farklı ortamlarda test edilmesine rağmen sonuç değişmemiş ve optimum kümeleme oranının her zaman orta seviyelerde olduğu tespit edilmiştir. Engelleri kümelemek yol uzunluğu açısından küçük bir dezavantaj gibi görünse de, çalışma süresi açısından yol uzunluğundaki zararı telafi edebilecek ve buna ek olarak hız konusunda kazanç sağlayacak düzeyde bir avantaj sağlamaktadır.

4.3.3.2. Önerilen Modelin Farklı Metasezgisel ve Kümeleme Algoritmalarıyla Gerçekleştirimi

Önerilen modelin detaylı analizinde optimum kümeleme oranları elde edilmişti. Modelin etkinliğini göstermek ve analizi desteklemek amacıyla, bu kümeleme oranları ile farklı metasezgisel ve kümeleme algoritmalarının performansları Alt Başlık 4.3.3.1'de bahsedilen her iki ortamda da karşılaştırmalı olarak değerlendirilmiştir. Engellerin kümelenmesi için KMC algoritmasına ek olarak hiyerarşik kümeleme (hierarchical clustering, HC) algoritması kullanılmıştır. Metasezgisel algoritmalardan da PSO'ya ek olarak TLBO, ABC, DE ve GA ele alınmıştır. Amaç fonksiyonu ve problem parametreleri Alt Başlık 4.3.3.1'deki simülasyonla aynıdır. TLBO, ABC, DE ve GA'nın kontrol parametreleri ise Tablo 4.15'te gösterilmektedir.

Tablo 4.15. TLBO, ABC, DE ve GA'nın kontrol parametreleri (T maksimum iterasyon sayısı, S popülasyon boyutu, D problem boyutu, F ölçekleme faktörü, CR çaprazlama oranı, MR mutasyon oranı)

Algoritma	Parametre
TLBO	$T = 200, S = 50$
ABC	$T = 200, S = 50, limit = 0.5 \times D \times S$
DE	$T = 200, S = 50, F = [0.2, 0.8], CR = 0.2$
GA	$T = 200, S = 50, CR = 0.98, MR = 0.1$

Önerilen modelde bu algoritmalar her iki kümeleme algoritması için ayrı ayrı 30 koşma ile çalıştırılmıştır. Her iki ortam için optimum kümeleme oranlarında bu algoritmalar tarafından elde edilen ortalama yol uzunlukları ve EKA'nın dâhil olduğu ortalama çalışma süreleri Tablo 4.16'da gösterilmektedir.

Tablo 4.16. Her iki ortam için PSO, TLBO, ABC, DE ve GA ile elde edilen ortalama yol uzunlukları ve ortalama çalışma süreleri (Bu bulgular 30 koşmanın ortalamasıdır. YU: Yol Uzunluğu, EDÇS: EKA Dâhil Çalışma Süresi)

Metasezgisel Algoritma	Ortam 15 (%70 Kümeleme Oranı)				Ortam 16 (%60 Kümeleme Oranı)			
	KMC		HC		KMC		HC	
	YU (br)	EDÇS (s)	YU (br)	EDÇS (s)	YU (br)	EDÇS (s)	YU (br)	EDÇS (s)
PSO	77,35	1,07	77,04	1,12	78,57	1,18	75,45	1,15
TLBO	75,49	1,60	75,22	1,59	74,95	1,64	74,25	1,61
ABC	75,24	1,61	75,20	1,60	74,53	1,72	74,08	1,65
DE	87,37	1,17	87,35	1,15	90,59	1,30	85,76	1,28
GA	79,50	1,08	80,11	1,07	81,93	1,14	79,51	1,08

Tablo 4.16 kümeleme algoritmaları açısından incelendiğinde, her iki ortam için HC'nin KMC'ye göre hem yol uzunluğu hem de çalışma süresi açısında avantajlı olduğu söylenebilir. Çalışmada ayrıca, HC ve KMC'nin engellerin kümelenmesi için harcadığı süreler sırasıyla yaklaşık 6 ve 20 ms olarak tespit edilmiştir. Bu değerler de bu avantajı desteklemektedir. Tablo 4.16 metasezgisel algoritmalar için yol uzunluğu açısından incelendiğinde, her iki ortam için en iyi performansı TLBO ve ABC algoritmaları göstermiştir. DE, PSO ve GA daha verimsiz çalışırken, bunlar arasında en kötü performans DE ile alınmıştır. Çalışma süresi açısından incelendiğinde ise Ortam 15'teki KMC hariç diğer senaryolarda en hızlı algoritma GA olmuştur.

4.4. Çoklu Robot Yol Planlaması için Çoklu Strateji ve Öz Uyarlamalı Diferansiyel Sinüs-Kosinüs Algoritması

Bedava Öğle Yemeği Yok (No Free Lunch) teoremine göre, bir algoritma tüm optimizasyon problemlerini çözmeyi garanti edemez. Bu nedenle, farklı optimizasyon tekniklerinin geliştirilmesi gerekir. Mobil robotların yol planlaması optimizasyonda hesaplama açısından zor bir problemdir ve bu problemde metasezgisel algoritmalar popülerdir [92]. Literatürde çoklu robotların yol planlama problemi için çeşitli metasezgisel algoritmalar önerilmiş olmasına rağmen, bunların çoğu hala bir seferde bir değişkeni güncellemeye dayanır. Bu, algoritmaların performansını ve yakınsamasını sınırlar. Ayrıca, araştırmacılar bir probleme yönelik uygun bir güncelleme stratejisi geliştirmek için birçok girişimde bulunmak zorundadır. Bu da yüksek hesaplama maliyetleri ve uzun çalışma sürelerine neden olur. Bu dezavantajlar nedeniyle, tek bir problem yerine daha geniş bir problem kümesi için optimizasyon algoritmaları geliştirmek önemlidir. Bu algoritmalardan biri olan SCA, karmaşık optimizasyon problemlerinde (özellikle çoklu robot sisteminin yol planlama probleminde) tatmin edici sonuçlar üretemez. Bu çalışmada, strateji havuzu ve kendi kendine adaptif bir mekanizmaya sahip olan sdSCA adında yeni bir algoritma önerilmiştir. Bu çalışmanın temel katkıları şu şekilde özetlenebilir:

- Mobil robotların yol planlaması optimizasyonda hesaplama açısından zor bir problemdir ve SCA çoklu robotların iki boyutlu yol planlama probleminde tatmin edici sonuçlar üretemez.

- Daha geniş bir problem kümesi için SCA'yı geliştirmek amacıyla tek bir güncelleme stratejisi yerine bir strateji havuzu düşünülmüştür. Bu havuz orijinal güncelleme stratejisine ek olarak yeni diferansiyel stratejiler eklenerek oluşturulmuştur.

- Strateji havuzunun kendi kendine adaptif olduğu, yani en tatmin edici sonucu üreten stratejinin daha sık kullanıldığı yeni bir algoritma tanıtılmıştır. Böylece, temel SCA'nın tek bir stratejiye bağımlılığı ortadan kaldırılmış ve daha geniş bir problem kümesi için daha kararlı bir algoritma önerilmiştir.

- Bu çalışma, özellikle dinamik engellerin olduğu ortamlarda insansız kara araçlarından oluşan çoklu robot sistemlerinin yerel yol planlama probleminde çoklu stratejili kendi kendine adaptif optimizasyon algoritmalarının kullanıldığı ilk uygulamalardan biridir.

4.4.1. Problem Tanımı

Çoklu robotların yerel yol planlaması, karmaşık ortamlarda statik-dinamik engellere ve diğer robotlara çarpmadan bir sonraki noktaya en az maliyetle hareket etmeyi ve böylece adım adım hedef noktaya ulaşmayı amaçlayan bir minimizasyon problemidir [93]. Bu çalışmada, robotların dairesel ve homojen olduğu çoklu mobil robot sisteminlerinin yerel yol planlaması amaçlanmıştır. Bu simülasyonda bir robotun bir sonraki konumu Eşitlik (4.19) kullanılarak hesaplanır.

$$p_i^n = p_i^c + v_i \begin{bmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{bmatrix} \Delta t \tag{4.19}$$

Burada, p_i^n i'inci robotun bir sonraki konumunu, p_i^c i'inci robotun mevcut konumunu, v_i ve φ_i sırasıyla i'inci robotun hızı ve yönelimini, Δt zaman adımını temsil eder. Hız, adım başına kat edilen mesafe olarak tanımlanır. Robotların bir sonraki konumlarını belirlemek için her adımda bir optimizasyon gerçekleştirilir. Bu optimizasyon tüm robotların hızına ve yönelimlerine dayanır. Öncelikle, hız $[v_l, v_h]$ aralığında ve yönelim $[\varphi_l, \varphi_h]$ aralığında robotların etrafında yerel bir arama uzayı oluşturulur. Ardından, bu arama uzayındaki en uygun bir sonraki konumun hızı ve yönelimi belirlenir. Robotlar hedef noktalarına ulaşana kadar bu uygulanabilir konumlar üzerinde hareket eder. Tüm robotların hızı ve yönelimi bir bütün olarak bir optimizasyon sürecinden geçer. Bu

nedenle, problem boyutu (D) sabit değildir, robot sayısıyla (n_r) orantılıdır. Optimizasyon robotların bu iki kinematik karakteristiğine dayandığından, $D = n_r \times 2$ şeklinde hesaplanır. Bu konsept için tasarlanan genel amaç fonksiyonunun parametrik formu Eşitlik (4.20)'deki gibi temsil edilir.

$$\arg\min_{v,\varphi} \mathcal{F} = \left\{ f(v_i, \varphi_i) \;\middle|\; \begin{array}{c} i \in \{1,2,\dots,n_r\} \\ v \in [v_l, v_h] \\ \varphi \in [\varphi_l, \varphi_h] \end{array} \right\} \tag{4.20}$$

Genel amaç fonksiyonu aşağıdaki dört maliyeti değerlendirir:

- f_1: Yol uzunluğu maliyeti
- f_2: Statik engellerden kaçınma maliyeti
- f_3: Dinamik engellerden kaçınma maliyeti
- f_4: Diğer robotlardan kaçınma maliyeti

Birinci maliyet (f_1) Eşitlik (4.21)'de gösterildiği gibi tanımlanmıştır.

$$f_1 = \sum_{i=1}^{n_r} (\|p_i^n - p_i^c\| + \|p_i^n - p_{ti}\|) \tag{4.21}$$

Burada, p_{ti} i'inci robotun hedef konumunu temsil eder. İkinci maliyet (f_2) Eşitlik (4.22)'de gösterildiği gibi tanımlanmıştır.

$$f_2 = \sum_{i=1}^{n_r} \sum_{j=1}^{n_{os}} \begin{cases} f; & eğer \ \|p_i^n - p_j^{os}\| \le L_g \ ise \\ 0; & aksi \ taktirde \end{cases} \tag{4.22}$$

Burada $f \in \mathbb{R}^+$ maliyeti (büyük bir sayıyı), L_g önceden belirlenmiş bir güvenlik mesafesini, n_{os} statik engellerin sayısını ve p_j^{os} ise j'inci statik engelin konumunu temsil eder. Üçüncü maliyet (f_3) Eşitlik (4.23)'te gösterildiği gibi tanımlanmıştır.

$$f_3 = \sum_{i=1}^{n_r} \sum_{j=1}^{n_{od}} \begin{cases} f; & eğer \ \|p_i^n - p_j^{od}\| \le L_g \ ise \\ 0; & aksi \ taktirde \end{cases} \tag{4.23}$$

Burada n_{od} dinamik engellerin sayısını ve p_j^{od} ise j'inci dinamik engelin konumunu temsil eder. Dördüncü maliyet (f_4) Eşitlik (4.24)'te gösterildiği gibi tanımlanmıştır.

$$f_4 = \sum_{i=1}^{n_r} \sum_{j=1}^{n_r-1} \begin{cases} f; & eğer \ \left\| p_i^n - p_j^r \right\| \leq L_g \ ise \\ 0; & aksi \ taktirde \end{cases} \tag{4.24}$$

Burada p_j^r ise j'inci diğer robotun konumunu temsil eder. Genel amaç fonksiyonu Eşitlik (4.25)'te tanımlandığı gibi bu maliyetlerin toplamıdır.

$$\mathcal{F} = f_1 + f_2 + f_3 + f_4 \tag{4.25}$$

Ayrıca dinamik engeller her adımda Eşitlik (4.26)'da tanımlandığı gibi hareket eder.

$$p_i^{od} = p_i^{od} + v_i^{od} \begin{bmatrix} \cos(\varphi_i^{od}) \\ \sin(\varphi_i^{od}) \end{bmatrix} \Delta t \tag{4.26}$$

Burada, p_i^{od} i'inci dinamik engelin anlık konumunu, v_i^{od} ve φ_i^{od} i'inci dinamik engelin sırasıyla sabit hızını ve hedef konumuna göre sabit yönelimini temsil eder. Bu çalışmada önerilen algoritmanın performansı adım sayısı, yol uzunluğu, yol sapma hatası (YSH), kalan hedef uzaklığı (KHU), toplam uygunluk değeri ve çalışma süresi açısından değerlendirilmiştir.

4.4.1.1. Yol Sapma Hatası

İterasyon başlamadan önce, yol planlama algoritması her robot için ideal yolu belirler. Bu yol başlangıç ve hedef noktaları arasında düz bir çizgidir. Yol sapma hatası (YSH), bu ideal yolun mesafesi ile algoritma tarafından planlanan yolun mesafesi arasındaki farktır ve algoritmanın k'ıncı koşusunda YSH Eşitlik (4.27)'deki gibi tanımlanmıştır.

$$YSH_k = \sum_{i=1}^{n_r} \left(L_i^k - I_i \right) \tag{4.27}$$

Burada, L_i^k algoritmanın k'ıncı koşusunda i'inci robotun katettiği mesafesini ve I_i i'inci robotun ideal yol uzunluğunu temsil eder. Ortalama yol sapma hatası ($OYSH$) ise Eşitlik (4.28) kullanılarak hesaplanır. Bu eşitlikte R algoritmanın koşma sayısını temsil eder.

$$OYSH = \frac{\sum_{k=1}^{R} YSH_k}{R} \tag{4.28}$$

4.4.1.2. Kalan Hedef Uzaklığı

Yol planlama algoritması maksimum adıma sahip robotun hedef noktasına ulaşmasına kadar işler. Bu süreçte her robotun her adımda hedefine olan uzaklığı saklanır. Robotların hedef noktalarına olan uzaklıkları Öklid formu açısından Eşitlik (4.29) kullanılarak hesaplanır.

$$H_i^{j,k} = \left\| p_i^{j,k} - p_{ti} \right\| \tag{4.29}$$

Burada $H_i^{j,k}$ algoritmanın k'ıncı koşusunda i'inci robotun j'inci adımında hedef noktasına olan uzaklığını, $p_i^{j,k}$ algoritmanın k'ıncı koşusunda i'inci robotun j'inci adımında optimizasyon sonunda elde edilen yeni konumunu ve p_{ti} i'inci robotun hedef konumunu temsil eder. Algoritmanın k'ıncı koşusunda kalan hedef uzaklığı (KHU) Eşitlik (4.30) kullanılarak hesaplanır.

$$KHU_k = \sum_{j=1}^{n_{st}} \sum_{i=1}^{n_r} H_i^{j,k} \tag{4.30}$$

Burada, n_{st} adım sayısı maksimum olan robotun adım sayısını temsil eder. Ortalama kalan hedef uzaklığı ($OKHU$) ise Eşitlik (4.31) kullanılarak hesaplanır. Bu eşitlikte R algoritmanın koşma sayısını temsil eder.

$$OKHU = \frac{\sum_{k=1}^{R} KHU_k}{R} \tag{4.31}$$

4.4.2. Önerilen Yöntem

Popülasyon tabanlı metasezgisel algoritmalar, rastgele bir başlangıç popülasyonu üretir. Bu popülasyon bir amaç fonksiyonu ile değerlendirildikten sonra iteratif bir güncelleme süreci başlar. Çoğu algoritmada bu popülasyondaki aday çözümler genellikle tek bir güncelleme stratejisi ile güncellenir ve iteratif süreç sona erdiğinde en iyi çözüm raporlanır. Ancak, tek bir güncelleme stratejisi kullanmak algoritmaların arama performansını kısıtlar. SCA da Eşitlik (3.44)'te verilen tek bir güncelleme stratejisine

sahip popülasyon tabanlı bir algoritmadır. SCA'nın bu kısıtlamasını ortadan kaldırmak için SCA'nın orijinal güncelleme stratejisine ilave olarak birkaç diferansiyel tabanlı güncelleme stratejisi eklenmiş ve sdSCA adında yeni bir algoritma önerilmiştir. Bu algoritmada popülasyondaki her çözüm bu stratejilerden birini seçerek kendini günceller. Ancak bu seçim rastgele değildir, rulet tekerleğine dayalı bir olasılık hesabı yoluyla gerçekleştirilir. Algoritma daha tatmin edici sonuçlar üreten stratejiyi adaptif bir şekilde öğrenir. Algoritma bu stratejiyi daha sık seçer ve böylece temel SCA'nın arama performansı iyileşir. İlave edilen güncelleme stratejileri Eşitlik (4.32), (4.33) ve (4.34)'te tanımlanmıştır [94-96]. $r_1, r_2, r_3 \neq i$ olmak üzere,

$$x_i' = \begin{cases} x_{r_1} + F\left(x_{r_2} - x_{r_3}\right); & eğer\ r < CR\ ise \\ x_i; & aksi\ taktirde \end{cases} \tag{4.32}$$

$$x_i' = \begin{cases} x_{r_1} + F\left(\hat{x} - x_i + x_{r_1} - x_{r_2}\right); & eğer\ r < CR\ ise \\ x_i; & aksi\ taktirde \end{cases} \tag{4.33}$$

$$x_i' = x_i + r\left(x_{r_1} - x_i\right) + F\left(x_{r_2} - x_{r_3}\right); \tag{4.34}$$

Burada, $r{\sim}U(0,1)$ rassal bir sayısı, x_i i'inci çözümü, x_i' i'inci çözümün güncellenmiş hâlini, $\hat{x}$ popülasyondaki en iyi çözümü, x_{r_1}, x_{r_2} ve x_{r_3} popülasyondan rastgele seçilen üç çözümü, F ölçekleme faktörünü ve CR çaprazlama oranını temsil eder. Önerilen algoritmanın çalışma prensibi aşağıdaki gibi açıklanabilir: Dört güncelleme stratejisi olan Eşitlik (3.44), (4.32), (4.33) ve (4.34) ile bir strateji havuzu oluşturulmuştur. Başlangıçta, her bir stratejinin seçilme olasılığı eşit ve %25'tir. Başlangıç popülasyonu Eşitlik (3.18) kullanılarak rastgele oluşturulur. Her çözüm rulet tekerleği tekniğini kullanarak kendi stratejisini seçer. Çözümler bir amaç fonksiyonunda değerlendirildikten sonra iteratif süreç başlar. Bu süreçte çözümler kendi stratejilerini kullanarak kendilerini günceller, ayrıca bu stratejilerin seçim sayaçları da güncellenir. Yeni popülasyon amaç fonksiyonunda değerlendirilir ve her stratejinin seçilme olasılığı Eşitlik (4.35) kullanılarak hesaplanır.

$$\delta_k = \frac{c_k}{\sum_{i=1}^{4} c_i} \tag{4.35}$$

Burada, $k \in \{1, 2, 3, 4\}$, δ_k k'ıncı stratejinin seçilme olasılığını ve c stratejilerin seçim sayacını temsil eder. Bu olasılıkları göz önünde bulundurarak her çözüm rulet tekerleği

tekniğini kullanarak kendi stratejisini günceller ve iteratif süreç durdurma kriteri sağanana kadar devam eder. En iyi çözüm, iteratif süreç sona erdiğinde raporlanır. Minimizasyon problemleri için sdSCA tabanlı çoklu robot yerel yol planlama algoritması ve önerilen sdSCA'nın sözde kodları sırasıyla Algoritma 4.6 ve 4.7'de gösterilmektedir.

Algoritma 4.6. sdSCA tabanlı çoklu robot yerel yol planlama algoritmasının sözde kodu

$p_s, p_t, p^{os}, p^{od} \leftarrow$ Robotların başlangıç ve hedef konumları, statik ve dinamik engellerin konumu
$v^{od}, \varphi^{od} \leftarrow$ Dinamik engellerin sabit hızı ve sabit yönelimi
for $i = 1{:}\, n_r$
    $p_i^c \leftarrow p_{s_i}$
    $L_i \leftarrow \left\| p_i^c - p_{t_i} \right\|$
    $n_{st_i} \leftarrow 0$
end for
$k \leftarrow \arg\max_{n_r}(L)$
while $p_k^c \neq p_{t_k}$
    $(v_i, \varphi_i) \leftarrow sdSCA(p_i^c, p^{os}, p^{od}), \quad i \in \{1,2,\dots,n_r\}$ // Algoritma 4.6
    for $i = 1{:}\, n_r$
        if $p_i^c \neq p_{t_i}$
            $p_i^n \leftarrow p_i^c + v_i \begin{bmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{bmatrix} \Delta t$
            $n_{st_i} \leftarrow n_{st_i} + 1$
        end if
    end for
    for $i = 1{:}\, n_{od}$
        $p_i^{od} \leftarrow p_i^{od} + v_i^{od} \begin{bmatrix} \cos(\varphi_i^{od}) \\ \sin(\varphi_i^{od}) \end{bmatrix} \Delta t$
    end for
    $p_i^c \leftarrow p_i^n, \quad i \in \{1,2,\dots,n_r\}$
    for $i = 1{:}\, n_r$
        $L_i \leftarrow \left\| p_i^c - p_{t_i} \right\|$
    end for
    $k \leftarrow \arg\max_{n_r}(L)$
end while

Algoritma 4.7: sdSCA'nın sözde kodu

Girdi: p^c, p^{os}, p^{od} // Robotun mevcut konumu, statik ve dinamik engellerin konumu
Çıktı: (v, φ) // Robotun hızı ve yönelimi

$D, x_l, x_h \leftarrow$ Problem boyutu, arama sınırları
$T, S, a, F, CR \leftarrow$ Kontrol parametreleri
$\hat{f} \leftarrow \infty,\ \delta_k \leftarrow 0.25,\ c_k \leftarrow 0,\quad k \in \{1,2,3,4\}$
$X \leftarrow x_l + r(x_h - x_l),\ r \sim U(0,1)^{S \times D}$
for $i = 1:S$
  $f_{x_i} \leftarrow \mathcal{F}(x_i, p^c, p^{os}, p^{od})$ // Amaç fonksiyonu - Eşitlik 4.25
  if $f_{x_i} < \hat{f}$
    $\hat{x} \leftarrow x_i,\ \hat{f} \leftarrow f_{x_i}$
  end if
  $xs_i \leftarrow \mathcal{F}(\delta)$ // Rulet tekerleği
end for
for $t = 1:T$
  for $i = 1:S$
    switch xs_i
      case 1
        $r_1 \leftarrow a - t(a/T)$
        $r_2, r_3, r_4 \sim U(0,2\pi), U(0,2), U(0,1)$
        if $r_4 < 0.5$
          $x_i' \leftarrow x_i + r_1 sin(r_2) |r_3 \hat{x} - x_i|$
        else
          $x_i' \leftarrow x_i + r_1 cos(r_2) |r_3 \hat{x} - x_i|$
        end if
      case 2
        if $r \sim U(0,1) < CR$
          $x_i' \leftarrow x_{r_1} + F(x_{r_2} - x_{r_3})$
        end if
      case 3
        if $r \sim U(0,1) < CR$
          $x_i' \leftarrow x_{r_1} + F(\hat{x} - x_i + x_{r_1} - x_{r_2})$
        end if
      case 4
        $x_i' \leftarrow x_i + r(x_{r_1} - x_i) + F(x_{r_2} - x_{r_3})$
    end switch
    $f_i' \leftarrow \mathcal{F}(x_i', p^c, p^{os}, p^{od})$ // Amaç fonksiyonu - Eşitlik 4.25
    if $f_i' < f_{x_i}$
      $x_i \leftarrow x_i',\ f_{x_i} \leftarrow f_i'$
      $c(xs_i) \leftarrow c(xs_i) + 1$
      if $f_{x_i} < \hat{f}$
        $\hat{x} \leftarrow x_i,\ \hat{f} \leftarrow f_{x_i}$
      end if
    end if
  end for
  $\delta_k \leftarrow c_k \,/ \sum_{i=1}^{4} c_i,\ k \in \{1,2,3,4\}$
  for $i = 1:S$
    $xs_i \leftarrow \mathcal{F}(\delta)$ // Rulet tekerleği
  end for
end for
$(v, \varphi) \leftarrow \hat{x}$

4.4.3. Bulgular

Simülasyonlar için MATLAB programlama dili ve 16 GB RAM'e sahip bir bilgisayar kullanılmıştır. Bu simülasyonlar aşağıdaki gibi özetlenebilir: İlk olarak, önerilen algoritmanın etkinliğini değerlendirmek için CEC2015 test fonksiyonları kullanılmıştır [97]. İkinci olarak, bu değerlendirmeyi desteklemek için algoritma CEC2020 gerçek dünya optimizasyon problemlerine uygulanmıştır [98]. Son olarak, çoklu robot yerel yol planlama problemine uygulanmış ve bazı güncel metasezgisel algoritmalarla karşılaştırılmıştır.

4.4.3.1. Önerilen sdSCA'nın CEC2015 Test Fonksiyonlarında Gerçekleştirimi

sdSCA'nın etkinliğini kanıtlamak için kullanılan CEC2015 test fonksiyonları Tablo 4.17'de gösterilmektedir.

Tablo 4.17. CEC2015 test fonksiyonları (f^* bilinen en iyi fonksiyon değerini temsil eder.)

Grup	**#**	**Fonksiyonun Adı**	f^*
Tek Modlu Fonksiyonlar	$\mathcal{F}_1$	Döndürülmüş Çok Koşullu Eliptik Fonksiyon	100
	$\mathcal{F}_2$	Döndürülmüş Puro Fonksiyonu	200
Basit Çok Modlu Fonksiyonlar	$\mathcal{F}_3$	Kaydırılmış ve Döndürülmüş Ackley Fonksiyonu	300
	$\mathcal{F}_4$	Kaydırılmış ve Döndürülmüş Rastrigin Fonksiyonu	400
	$\mathcal{F}_5$	Kaydırılmış ve Döndürülmüş Schwefel Fonksiyonu	500
Hibrit Fonksiyonlar	$\mathcal{F}_6$	Hibrit Fonksiyon 1 ($D = 3$)	600
	$\mathcal{F}_7$	Hibrit Fonksiyon 1 ($D = 4$)	700
	$\mathcal{F}_8$	Hibrit Fonksiyon 1 ($D = 5$)	800
Kompozisyon Fonksiyonları	$\mathcal{F}_9$	Kompozisyon Fonksiyonu 1 ($D = 3$)	900
	$\mathcal{F}_{10}$	Kompozisyon Fonksiyonu 2 ($D = 3$)	1000
	$\mathcal{F}_{11}$	Kompozisyon Fonksiyonu 3 ($D = 5$)	1100
	$\mathcal{F}_{12}$	Kompozisyon Fonksiyonu 4 ($D = 5$)	1200
	$\mathcal{F}_{13}$	Kompozisyon Fonksiyonu 5 ($D = 5$)	1300
	$\mathcal{F}_{14}$	Kompozisyon Fonksiyonu 6 ($D = 7$)	1400
	$\mathcal{F}_{15}$	Kompozisyon Fonksiyonu 7 ($D = 10$)	1500

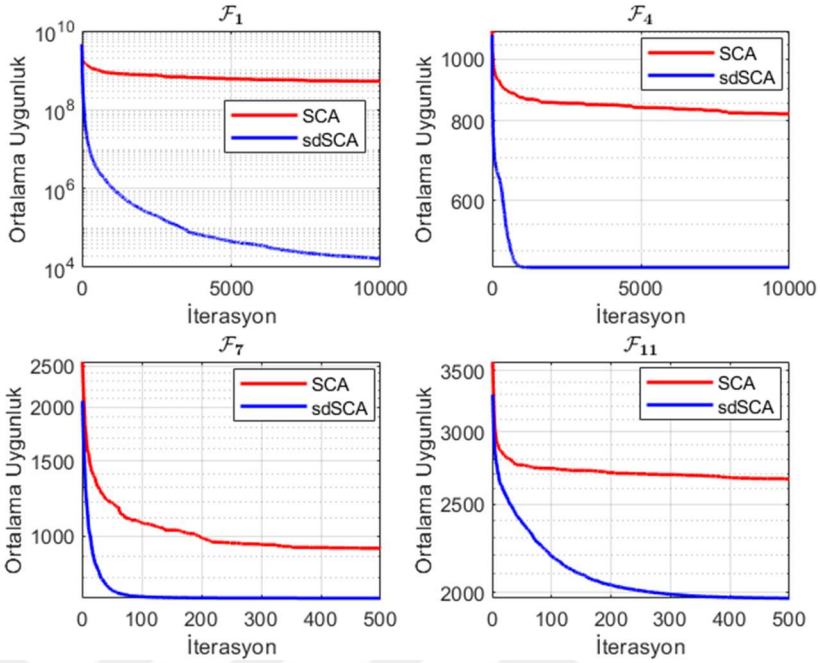
Tüm test fonksiyonları 30 boyutlu ve [-100 100] aralığı için 30 kez çalıştırılmıştır. SCA ve sdSCA için maksimum uygunluk değerlendirme sayısı, popülasyon boyutu ve α sırasıyla 10000 x D, 30 ve 2 olarak; sdSCA için F ve CR parametreleri sırasıyla 0.8 ve

0.95 olarak ayarlanmıştır. CEC2015 test fonksiyonları için sdSCA'nın SCA, EABC [99], AβHC [100] ve PgAFWA [101] ile karşılaştırması Tablo 4.18'de gösterilmektedir.

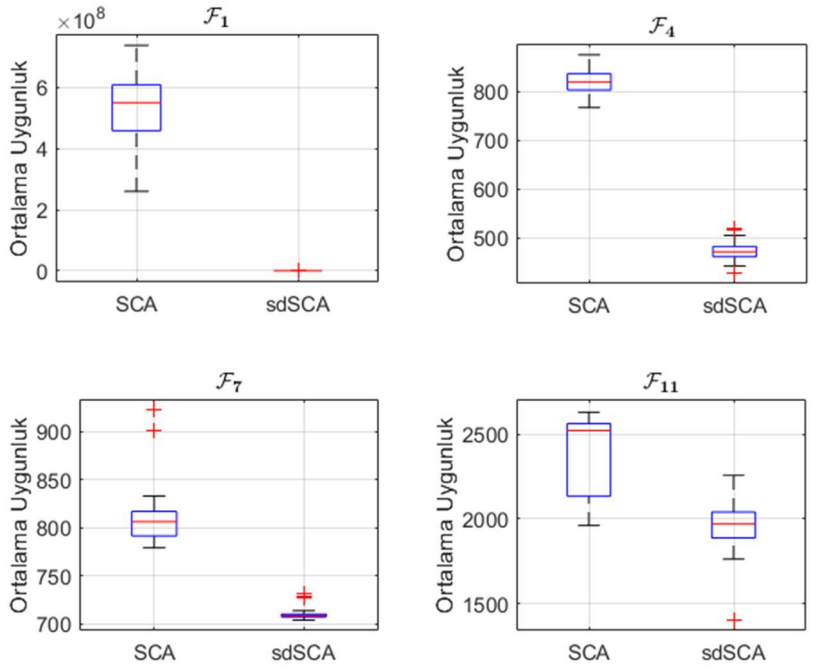
Tablo 4.18. CEC2015 test fonksiyonları için sdSCA'nın SCA, EABC, AβHC ve PgAFWA ile karşılaştırması (Bu bulgular 30 koşmanın ortalamasıdır.)

Fonksiyon	SCA	EABC	AβHC	PgAFWA	sdSCA
$\mathcal{F}_1$	5.29e+08	6.91e+05	1.86e+07	1.70e+06	1.65e+04
$\mathcal{F}_2$	5.02e+10	5.45e+00	1.93e+08	3.98e+05	9.44e-01
$\mathcal{F}_3$	2.09e+01	2.00e+01	2.00e+01	3.20e+02	2.08e+01
$\mathcal{F}_4$	4.19e+02	3.39e+01	1.19e+02	5.09e+02	7.14e+01
$\mathcal{F}_5$	7.19e+03	1.29e+03	3.18e+03	3.69e+03	3.48e+03
$\mathcal{F}_6$	1.38e+07	2.14e+05	4.82e+06	1.99e+05	2.29e+03
$\mathcal{F}_7$	1.11e+02	4.20e+00	2.85e+01	7.15e+02	1.00e+01
$\mathcal{F}_8$	2.90e+06	1.12e+05	1.16e+06	1.12e+05	6.82e+02
$\mathcal{F}_9$	2.61e+02	1.04e+02	1.76e+02	1.12e+03	1.20e+02
$\mathcal{F}_{10}$	1.14e+07	1.10e+05	2.80e+06	1.73e+05	1.28e+03
$\mathcal{F}_{11}$	1.29e+03	2.82e+02	8.49e+02	1.68e+03	8.69e+02
$\mathcal{F}_{12}$	1.83e+02	1.05e+02	1.23e+02	1.31e+03	1.50e+02
$\mathcal{F}_{13}$	2.17e-01	8.33e+01	4.43e-02	1.42e+03	6.87e-02
$\mathcal{F}_{14}$	4.87e+04	2.45e+04	3.37e+04	2.22e+04	3.44e+04
$\mathcal{F}_{15}$	1.31e+04	1.00e+02	1.08e+02	1.60e+03	1.00e+02

Tablo 4.18, önerilen algoritmanın tüm test fonksiyonlarında temel SCA'dan çok daha iyi performans gösterdiğini göstermektedir. Bu performans özellikle tek modlu fonksiyonlarla birlikte $\mathcal{F}_6$, $\mathcal{F}_8$, $\mathcal{F}_{10}$ ve $\mathcal{F}_{15}$'te görülmektedir. Diğer fonksiyonlarda iyileştirme etkisi nispeten daha az olmuştur. Buna rağmen literatürdeki diğer algoritmalarla karşılaştırıldığında oldukça verimli ve rekabetçi olduğu söylenebilir. Çalışma süresine açısından her iki algoritma için de en az süre alan fonksiyon $\mathcal{F}_2$'dir, bu fonksiyonda SCA'nın çalışma süresi 5.13 saniye iken, sdSCA'nın çalışma süresi 5.83 saniyedir. Yine her iki algoritma için de en fazla süre alan fonksiyon $\mathcal{F}_{15}$'tir, bu fonksiyonda SCA'nın çalışma süresi 43.24 saniye iken, sdSCA'nın çalışma süresi 40.88 saniyedir. Önerilen algoritmanın çalışma süresi açısından SCA ile yaklaşık aynı olduğu, hatta $\mathcal{F}_{15}$ fonksiyonu için biraz daha iyi olduğu söylenebilir. Ortalama yakınsama eğrileri ve kutu grafikleri için her gruptan için bir fonksiyon seçilmiştir. $\mathcal{F}_1$, $\mathcal{F}_4$, $\mathcal{F}_7$ ve $\mathcal{F}_{11}$ fonksiyonları için SCA ve sdSCA'nın ortalama yakınsama eğrileri ve kutu grafikleri sırasıyla Şekil 4.29 ve 4.30'da gösterilmektedir.



Şekil 4.29. $\mathcal{F}_1$, $\mathcal{F}_4$, $\mathcal{F}_7$ ve $\mathcal{F}_{11}$ fonksiyonları için SCA ve sdSCA'nın ortalama yakınsama eğrileri



Şekil 4.30. $\mathcal{F}_1$, $\mathcal{F}_4$, $\mathcal{F}_7$ ve $\mathcal{F}_{11}$ fonksiyonları için SCA ve sdSCA'nın kutu grafikleri

Ortalama yakınsama eğrileri göz önüne alındığında, sdSCA'nın geliştirilmesiyle temel SCA'nın optimizasyon yeteneğinin ve yakınsama hızının önemli ölçüde iyileştirildiği görülmektedir. Kutu grafikleri göz önüne alındığında, sdSCA'nın çoğu fonksiyonda düşük varyans ve medyan açısından daha performanslı olduğu ve temel algoritmadan

daha kararlı olduğu görülmektedir. Her iki algoritmanın maksimum ve minimum verileri de bu sonucu desteklemektedir.

4.4.3.2. Önerilen sdSCA'nın CEC2020 Gerçek Dünya Optimizasyon Problemlerinde Gerçekleştirimi

Önerilen algoritmanın CEC2015 test fonkisyonlarındaki etkinliğini desteklemek için algoritma CEC2020 gerçek dünya optimizasyon problemlerine uygulanmıştır. Bu problemlerden 20 tanesi kullanılmıştır ve bunlar Tablo 4.19'da gösterilmektedir.

Tablo 4.19. CEC2020 gerçek dünya optimizasyon problemleri (D problemin boyutu, g eşitsizlik kısıtlarının sayısı, h eşitlik kısıtlarının sayısı ve f^* bilinen en iyi fonksiyon değerini temsil eder.)

Grup	**#**	**Problem**	D	g	h	f^*
Endüstriyel Kimyasal Süreçler	$\mathcal{F}_1$	Alkilasyon Ünitesinin Optimum İşlemi	7	14	0	-4.52e+03
	$\mathcal{F}_2$	Reaktör Ağ Tasarımı	6	1	4	-3.88e-01
Süreç Sentezi ve Tasarım Problemleri	$\mathcal{F}_3$	Süreç Sentezi Problemi 1	2	2	0	2.00e+00
	$\mathcal{F}_4$	Süreç Sentezi ve Tasarım Problemi	3	1	1	2.55e+00
	$\mathcal{F}_5$	Süreç Akış Şeması Problemi	3	3	0	1.07e+00
	$\mathcal{F}_6$	Süreç Sentezi Problemi 2	7	9	0	2.92e+00
	$\mathcal{F}_7$	Süreç Tasarım Problemi	5	3	0	2.68e+04
	$\mathcal{F}_8$	Çok Ürünlü Yarı Mamul Tesisi	10	10	0	5.36e+04
Makine Mühendisliği Problemleri	$\mathcal{F}_9$	Hız Azaltıcı Ağırlık Minimizasyonu	7	11	0	2.99e+03
	$\mathcal{F}_{10}$	Gerilim/Kompressiyon Yay Tasarımı (V1)	3	3	0	1.26e-02
	$\mathcal{F}_{11}$	Kaynaklı Kiriş Tasarımı	4	5	0	1.67e+00
	$\mathcal{F}_{12}$	Üç Çubuklu Gerilme Problemi	2	3	0	2.63e+02
	$\mathcal{F}_{13}$	Adım Koni Kasnağı Problemi	5	8	3	1.60e+01
	$\mathcal{F}_{14}$	Robot Uçişlevci Problemi	7	7	0	2.52e+00
	$\mathcal{F}_{15}$	Hidrostatik Yataklama Tasarımı	4	7	0	1.61e+03
	$\mathcal{F}_{16}$	10 Çubuklu Gerilme Tasarımı	10	3	0	5.24e+02
	$\mathcal{F}_{17}$	Gaz İletim Kompresör Tasarımı	4	1	0	2.96e+06
	$\mathcal{F}_{18}$	Gerilim/Kompressiyon Yay Tasarımı (V2)	3	8	0	2.61e+00
	$\mathcal{F}_{19}$	Himmelblau Fonksiyonu	5	6	0	-3.06e+04
	$\mathcal{F}_{20}$	Topoloji Optimizasyonu	30	30	0	2.63e+00

Tüm problemler 30 kez çalıştırılmıştır. Maksimum uygunluk değerlendirme sayısı $\mathcal{F}_1$-$\mathcal{F}_{19}$ için 1e+5 ve $\mathcal{F}_{20}$ için 2e+5 olarak, popülasyon boyutu 40 olarak ayarlanmıştır. SCA ve sdSCA için diğer parametreler Alt Başlık 4.4.3.1'deki ayarlarla aynıdır. Önerilen algoritma, CEC2020 yarışmasında en üst sıralarda yer alan algoritmalarla (SASS [102], COLSHADE [103], EnMODE [104] ve sCMAgES [105]) karşılaştırılmıştır. Bu dört algoritmanın parametreleri literatürde sunulan ayarlarla aynıdır [106]. Kısıt işleme tekniği olarak öz uyarlamalı ceza stratejisi kullanılmıştır [107]. Bu karşılaştırma Tablo 4.20'de gösterilmektedir.

Tablo 4.20. CEC2020 gerçek dünya optimizasyon problemleri için sdSCA'nın SASS, COLSHADE, EnMODE ve sCMAgES ile karşılaştırması (OP: Optimum, EK: En Kötü, OR: Ortalama, SS: Standart Sapma)

Problem		**SASS**	**COLSHADE**	**EnMODE**	**sCMAgES**	**sdSCA**
$\mathcal{F}_1$	OP	-1.42e+02	-4.52e+03	-4.52e+03	-4.52e+03	-4.52e+03
	EK	-1.42e+02	-4.36e+03	-4.52e+03	-4.15e+03	-4.52e+03
	OR	-1.42e+02	-3.71e+03	-4.52e+03	-3.32e+03	-4.52e+03
	SS	1.07e-05	3.33e+02	1.62e-12	3.82e+02	1.78e-12
$\mathcal{F}_2$	OP	-3.88e-01	-3.88e-01	-3.88e-01	-3.88e-01	-3.87e-01
	EK	-3.88e-01	-3.80e-01	-3.74e-01	-3.87e-01	1.94e+01
	OR	-3.88e-01	-3.58e-01	-3.69e-01	-3.74e-01	5.15e+00
	SS	1.27e-06	1.15e-02	3.45e-03	2.98e-03	8.74e+00
$\mathcal{F}_3$	OP	2.00e+00	2.00e+00	2.00e+00	2.00e+00	2.00e+00
	EK	2.00e+00	2.00e+00	2.00e+00	2.00e+00	2.00e+00
	OR	2.00e+00	2.00e+00	2.00e+00	2.00e+00	2.00e+00
	SS	6.12e-15	2.60e-16	2.28e-16	1.84e-16	1.88e-16
$\mathcal{F}_4$	OP	2.55e+00	2.55e+00	2.55e+00	2.55e+00	2.56e+00
	EK	2.55e+00	2.55e+00	2.55e+00	2.55e+00	3.55e+00
	OR	2.55e+00	2.55e+00	2.55e+00	2.55e+00	3.00e+00
	SS	2.74e-10	9.11e-16	9.11e-16	8.01e-09	3.75e-01
$\mathcal{F}_5$	OP	1.07e+00	1.07e+00	1.07e+00	1.07e+00	1.07e+00
	EK	1.07e+00	1.08e+00	1.17e+00	1.07e+00	1.25e+00
	OR	1.07e+00	1.25e+00	2.25e+00	1.07e+00	1.09e+00
	SS	1.18e-13	3.87e-02	8.85e-02	2.40e-14	4.73e-02
$\mathcal{F}_6$	OP	2.92e+00	2.92e+00	2.92e+00	2.92e+00	2.92e+00
	EK	2.92e+00	2.93e+00	2.92e+00	2.93e+00	4.20e+00
	OR	2.92e+00	3.08e+00	2.92e+00	2.95e+00	3.01e+00
	SS	8.63e-10	3.52e-02	9.11e-16	1.15e-02	2.34e-01

Tablo 4.20. Devam

Problem		SASS	COLSHADE	EnMODE	sCMAgES	sdSCA
$\mathcal{F}_7$	OP	2.68e+04	2.68e+04	2.68e+04	2.68e+04	2.68e+04
	EK	2.68e+04	2.68e+04	2.68e+04	2.68e+04	2.68e+04
	OR	2.68e+04	2.68e+04	2.68e+04	2.68e+04	2.68e+04
	SS	4.07e-10	7.46e-12	1.12e-11	1.12e-11	1.48e-11
$\mathcal{F}_8$	OP	5.85e+04	5.85e+04	5.85e+04	5.36e+04	5.36e+04
	EK	5.87e+04	5.85e+04	5.85e+04	5.49e+04	5.91e+04
	OR	6.18e+04	5.85e+04	5.85e+04	5.92e+04	5.46e+04
	SS	9.33e+02	2.83e-11	7.83e-09	1.67e+03	2.07e+03
$\mathcal{F}_9$	OP	2.99e+03	2.99e+03	2.99e+03	2.99e+03	2.99e+03
	EK	2.99e+03	2.99e+03	2.99e+03	2.99e+03	2.99e+03
	OR	2.99e+03	2.99e+03	2.99e+03	2.99e+03	2.99e+03
	SS	5.91e-09	9.33e-13	1.40e-12	7.66e-12	0.00e+00
$\mathcal{F}_{10}$	OP	1.26e-02	1.26e-02	1.26e-02	1.26e-02	1.26e-02
	EK	1.26e-02	1.26e-02	1.27e-02	1.26e-02	1.26e-02
	OR	1.26e-02	1.26e-02	1.27e-02	1.26e-02	1.26e-02
	SS	8.13e-10	5.11e-09	1.97e-05	2.53e-06	6.70e-18
$\mathcal{F}_{11}$	OP	1.67e+00	1.67e+00	1.67e+00	1.67e+00	1.67e+00
	EK	1.67e+00	1.67e+00	1.67e+00	1.67e+00	1.67e+00
	OR	1.67e+00	1.67e+00	1.67e+00	1.67e+00	1.67e+00
	SS	7.72e-13	6.83e-16	6.97e-16	1.57e-13	2.22e-16
$\mathcal{F}_{12}$	OP	2.63e+02	2.63e+02	2.63e+02	2.63e+02	2.63e+02
	EK	2.63e+02	2.63e+02	2.63e+02	2.63e+02	2.63e+02
	OR	2.63e+02	2.63e+02	2.63e+02	2.63e+02	2.63e+02
	SS	1.55e-12	0.00e+00	0.00e+00	6.66e-13	0.00e+00
$\mathcal{F}_{13}$	OP	1.60e+01	1.60e+01	1.60e+01	1.60e+01	1.60e+01
	EK	1.60e+01	1.60e+01	1.60e+01	1.62e+01	1.60e+01
	OR	1.60e+01	1.60e+01	1.60e+01	1.67e+01	1.60e+01
	SS	3.47e-09	4.89e-15	3.04e-14	1.84e-01	7.22e-15
$\mathcal{F}_{14}$	OP	2.54e+00	2.54e+00	2.54e+00	2.61e+00	2.54e+00
	EK	2.54e+00	2.54e+00	2.54e+00	2.88e+00	2.54e+00
	OR	2.54e+00	2.54e+00	2.54e+00	3.34e+00	2.54e+00
	SS	5.27e-09	5.11e-14	4.75e-12	1.90e-01	2.01e-14
$\mathcal{F}_{15}$	OP	1.61e+03	1.61e+03	1.61e+03	2.13e+03	1.61e+03
	EK	1.61e+03	1.62e+03	1.61e+03	3.01e+03	1.61e+03
	OR	1.63e+03	1.66e+03	1.61e+03	3.62e+03	1.61e+03
	SS	4.15e+00	1.47e+01	1.00e-10	3.83e+02	4.22e-13

Tablo 4.20. Devam

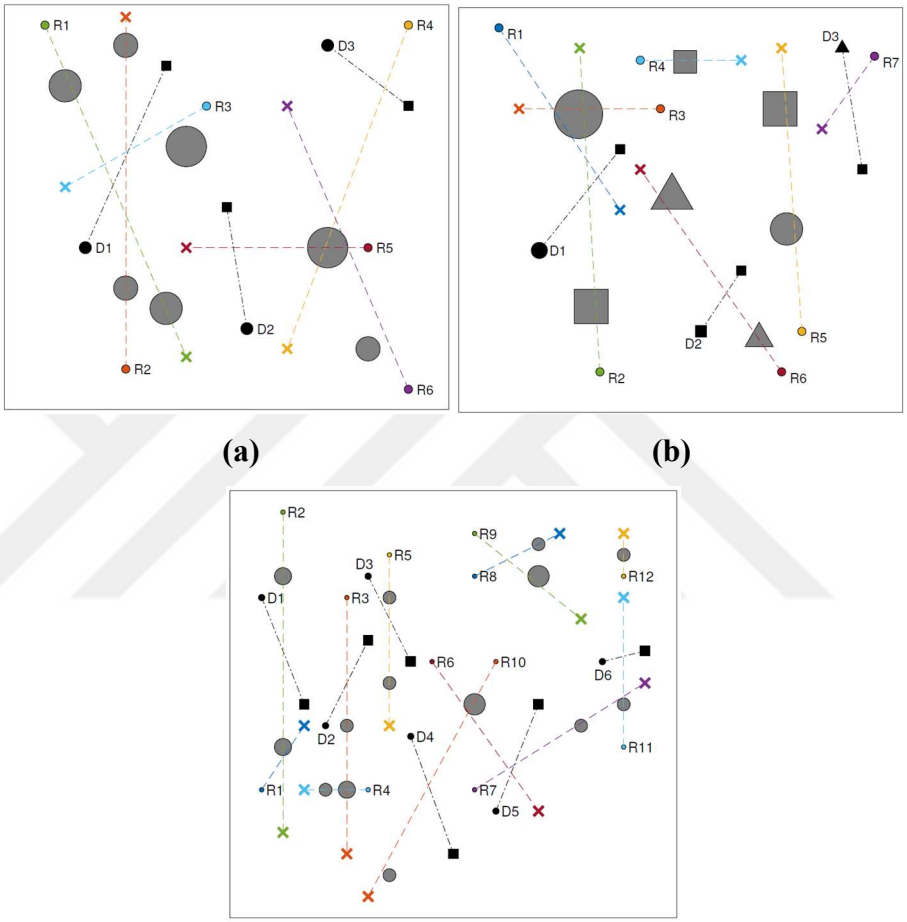
Problem		SASS	COLSHADE	EnMODE	sCMAgES	sdSCA
$\mathcal{F}_{16}$	OP	5.24e+02	5.24e+02	5.24e+02	5.24e+02	5.24e+02
	EK	5.24e+02	5.24e+02	5.24e+02	5.24e+02	5.30e+02
	OR	5.24e+02	5.24e+02	5.24e+02	5.24e+02	5.24e+02
	SS	9.84e-03	1.05e-09	4.32e-09	1.13e-01	1.55e+00
$\mathcal{F}_{17}$	OP	2.96e+06	2.96e+06	2.96e+06	2.96e+06	2.96e+06
	EK	2.96e+06	2.96e+06	2.96e+06	2.96e+06	2.96e+06
	OR	2.96e+06	2.96e+06	2.96e+06	2.96e+06	2.96e+06
	SS	1.74e-09	9.56e-10	9.56e-10	1.08e+01	1.42e-09
$\mathcal{F}_{18}$	OP	2.65e+00	2.65e+00	2.65e+00	2.97e+00	2.65e+00
	EK	2.65e+00	2.65e+00	2.70e+00	4.28e+00	2.65e+00
	OR	2.65e+00	2.65e+00	3.63e+00	6.35e+00	2.65e+00
	SS	2.28e-11	4.56e-16	2.18e-01	9.08e-01	4.51e-16
$\mathcal{F}_{19}$	OP	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04
	EK	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04
	OR	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04	-3.06e+04
	SS	2.17e-09	8.35e-13	3.73e-12	2.01e-10	1.11e-11
$\mathcal{F}_{20}$	OP	2.63e+00	2.63e+00	2.63e+00	2.63e+00	2.63e+00
	EK	2.63e+00	2.63e+00	2.63e+00	2.63e+00	3.05e+00
	OR	2.63e+00	2.63e+00	2.63e+00	2.63e+00	2.68e+00
	SS	1.87e-09	1.28e-15	7.28e-16	1.14e-15	9.01e-02

Tablo 4.20 sdSCA'nın diğer dört algoritma gibi çoğunlukla bilinen en iyi fonksiyon değerine ulaştığını göstermektedir. Önerilen algoritma karşılaştırılan algoritmalarla rekabet halindedir. Algoritma, diğer algoritmalardan daha kötü sonuçlar veren problemlerde bile bilinen en iyi fonksiyon değerine yakın sonuçlar üretmiştir.

4.4.3.3. Önerilen sdSCA'nın Çoklu Robot Yerel Yol Planlama Probleminde Gerçekleştirimi

Bu simülasyonda önerilen algoritma çoklu robotların yerel yol planlaması probleminde uygulanmıştır. Robotlar dairesel, homojen ve aynı boyutta tasarlanmıştır. Dinamik engeller de homojendir ve sabit bir hızla iki belirli nokta arasında doğrusal olarak hareket edecek şekilde düşünülmüştür. Bu çalışma için hem statik hem de dinamik engelleri içeren üç ortam oluşturulmuştur. Ortam 17, 100 x 100 br boyutunda tasarlanmış ve 6 robot kullanılmıştır. Bu ortamda 7 statik engel ve 3 dinamik engel vardır. Ortam 18, 100

x 100 br boyutunda tasarlanmış ve 7 robot kullanılmıştır. Bu ortamda 7 statik engel ve 3 dinamik engel vardır. Ortam 19, 200 x 200 br boyutunda tasarlanmış ve 12 robot kullanılmıştır. Bu ortamda 14 statik engel ve 6 dinamik engel vardır. Statik ve dinamik engeller Ortam 18'de farklı şekillerdeyken, diğer ortamlarda çeşitli yarıçaplarda daireseldir. Bu ortamlar Şekil 4.31'de gösterilmektedir.



Şekil 4.31. Çoklu robotların yerel yol planlaması için tasarlanan ortamlar: **(a)** Ortam 17, **(b)** Ortam 18, **(c)** Ortam 19

Ortam 17'de robotlar farklı renk kodlarıyla R1-R6 olarak temsil edilmektedir. Bu renklere karşılık gelen noktalı çizgiler ve çarpı işaretleri sırasıyla robotların ideal yollarını ve hedef noktalarını göstermektedir. Statik engeller gri dairelerle, dinamik engeller ise D1-D3 siyah dairelerle temsil edilmektedir. Siyah kesik çizgiler ve siyah kare işaretler sırasıyla dinamik engellerin yollarını ve hedef noktalarını göstermektedir. Dinamik engeller düz bir çizgide 0.5 br/s (D1), 0.45 br/s (D2) ve 1.2 br/s (D3) sabit hızlarla hareket etmektedir. Dinamik engellerin bu hızları, robotları kısıtlayacak ve her ortam için problemi

karmaşıklaştıracak şekilde seçilmiştir. Ortam 18'de robotlar farklı renk kodlarıyla R1-R7 olarak temsil edilmektedir. Bu renklere karşılık gelen noktalı çizgiler ve çarpı işaretleri sırasıyla robotların ideal yollarını ve hedef noktalarını göstermektedir. Statik engeller gri dairelerle, dinamik engeller ise D1-D3 siyah dairelerle temsil edilmektedir. Statik engeller iki daire, iki üçgen ve üç kare biçimindeyken, dinamik engeller bir daire, bir üçgen ve bir kare şeklinde tasarlanmıştır. Siyah kesik çizgi ve siyah kare işaretler dinamik engellerin sırasıyla yollarını ve hedef noktalarını göstermektedir. Dinamik engeller 0.5 br/s (D1), 0.1 br/s (D2) ve 1.1 br/s (D3) sabit hızlarıylaa düz bir çizgide hareket eder. Ortam 19'da robotlar farklı renk kodlarıyla R1-R12 olarak temsil edilmektedir. Bu renklere karşılık gelen noktalı çizgiler ve çarpı işaretleri sırasıyla robotların ideal yollarını ve hedef noktalarını göstermektedir. Statik engeller gri dairelerle, dinamik engeller ise D1-D6 siyah dairelerle temsil edilmektedir. Siyah kesik çizgi ve siyah kare işaretler dinamik engellerin sırasıyla yollarını ve hedef noktalarını göstermektedir. Dinamik engeller 0.5 br/s (D1), 0.5 br/s (D2), 0.6 br/s (D3), 0.3 br/s (D4), 0.4 br/s (D5) ve 0.25 br/s (D6) sabit hızlarıyla düz bir çizgide hareket eder. Her ortam için, robotların yarıçapı 1 br iken dinamik engellerin yarıçapı 1.5 br olarak tanımlanmıştır. Optimize edilecek parametreler olan hız ve yönelim aralıkları sırasıyla [1 1.5] br/s ve [0 2π] radyandır. Amaç fonksiyonlarından F_2, F_3 ve F_4'teki ε değerleri 10^5, zaman adımı (Δt) da 1 s olarak ayarlanmıştır.

İlk olarak, sdSCA'nın çoklu robot sistemindeki toplam robot sayısı üzerindeki performansı Ortam 17 için incelenmiştir. Ardından, önerilen algoritma Ortam 17, 18 ve 19'da uygulanmış ve temel SCA ile SFS, AOA, WOA [108] ve HHO [109] algoritnaları ile karşılaştırılmıştır. Algoritmaların kontrol parametreleri ise Tablo 4.21'de gösterilmektedir.

Tablo 4.21. Algoritmaların kontrol parametreleri (T maksimum uygunluk değerlendirme sayısı, S popülasyon boyutu, a r_1 sayısı için ayarlanan parametre, n_{dif} difüzyon sayısı, $walk$ yürüme oranı, α kullanım doğruluğunu tanımlayan parametre, μ arama sürecini ayarlayan parametre, b logaritmik spiralin şeklini tanımlayan parametre, β Levy uçuşunda kullanılan sabit parametre, F ölçekleme faktörü, CR çaprazlama oranı)

Algoritma	**Parametre**
SCA	$T = 1000, S = 30, a = 2$
SFS	$T = 1000, S = 30, n_{dif} = 2, walk = 0.5$
AOA	$T = 1000, S = 30, \alpha = 5, \mu = 0.5$
WOA	$T = 1000, S = 30, b = 1$
HHO	$T = 1000, S = 30, \beta = 1.5$
sdSCA	$T = 1000, S = 30, a = 2, F = 0.8, CR = 0.95$

Robot Sayısının Performansa Etkisi: Çoklu robot sistemindeki toplam robot sayısı üzerinde sdSCA'nın performansını incelemek için, Ortam 17'de sdSCA farklı robot sayılarıyla (2, 3, 4 ve 5) 30 kez çalıştırılmış ve ortalama yol sapma hatası ($OYSH$), ortalama kalan hedef uzaklığı ($OKHU$), toplam uygunluk değeri ve çalışma süresi açısından temel SCA ile karşılaştırılmıştır. Bunlar sırasıyla Tablo 4.22, 4.23, 4.24 ve 4.25'te gösterilmektedir. SCA ve sdSCA için $OKHU$ ve çalışma süresi robot sayısına göre Şekil 4.32'de grafiksel olarak da gösterilmektedir.

Tablo 4.22. SCA ve sdSCA'nın farklı robot sayıları için $OYSH$ karşılaştırması

Robot Sayısı	***OYSH* (br)**		**İyileştirme Oranı (%)**
	SCA	**sdSCA**	
2	27.58	14.55	47.24
3	59.56	16.20	72.80
4	157.50	30.55	80.60
5	278.15	50.83	81.72

Tablo 4.23. SCA ve sdSCA'nın farklı robot sayıları için $OKHU$ karşılaştırması

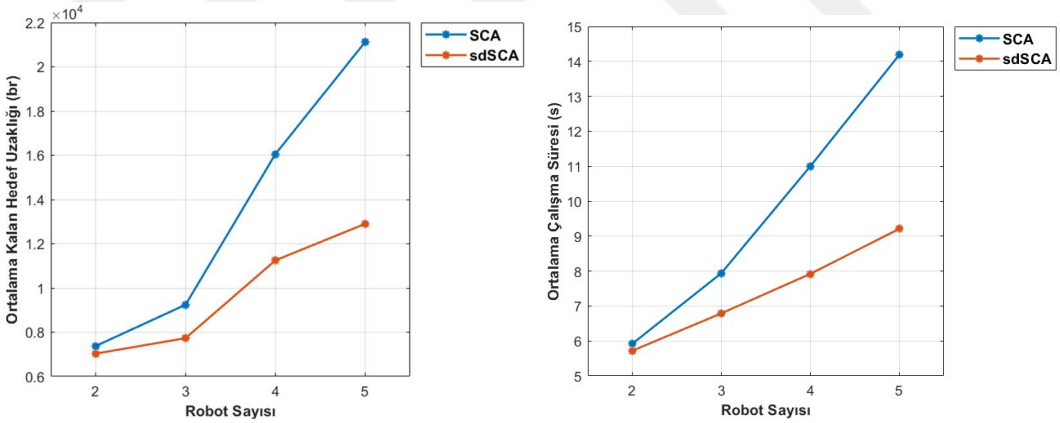
Robot Sayısı	***OKHU* (br)**		**İyileştirme Oranı (%)**
	SCA	**sdSCA**	
2	7377	7034	4.65
3	9243	7733	16.33
4	16052	11250	29.91
5	21124	12901	38.92

Tablo 4.24. SCA ve sdSCA'nın farklı robot sayıları için toplam uygunluk değeri karşılaştırması

Robot Sayısı	Toplam Uygunluk Değeri		İyileştirme Oranı (%)
	SCA	sdSCA	
2	7893	7550	4.34
3	9560	8033	15.97
4	16459	11631	29.33
5	21442	13401	37.50

Tablo 4.25. SCA ve sdSCA'nın farklı robot sayıları için çalışma süresi karşılaştırması

Robot Sayısı	Çalışma Süresi (s)		İyileştirme Oranı (%)
	SCA	sdSCA	
2	5.92	5.72	3.37
3	7.94	6.79	14.48
4	11.00	7.92	28.00
5	14.19	9.21	35.09



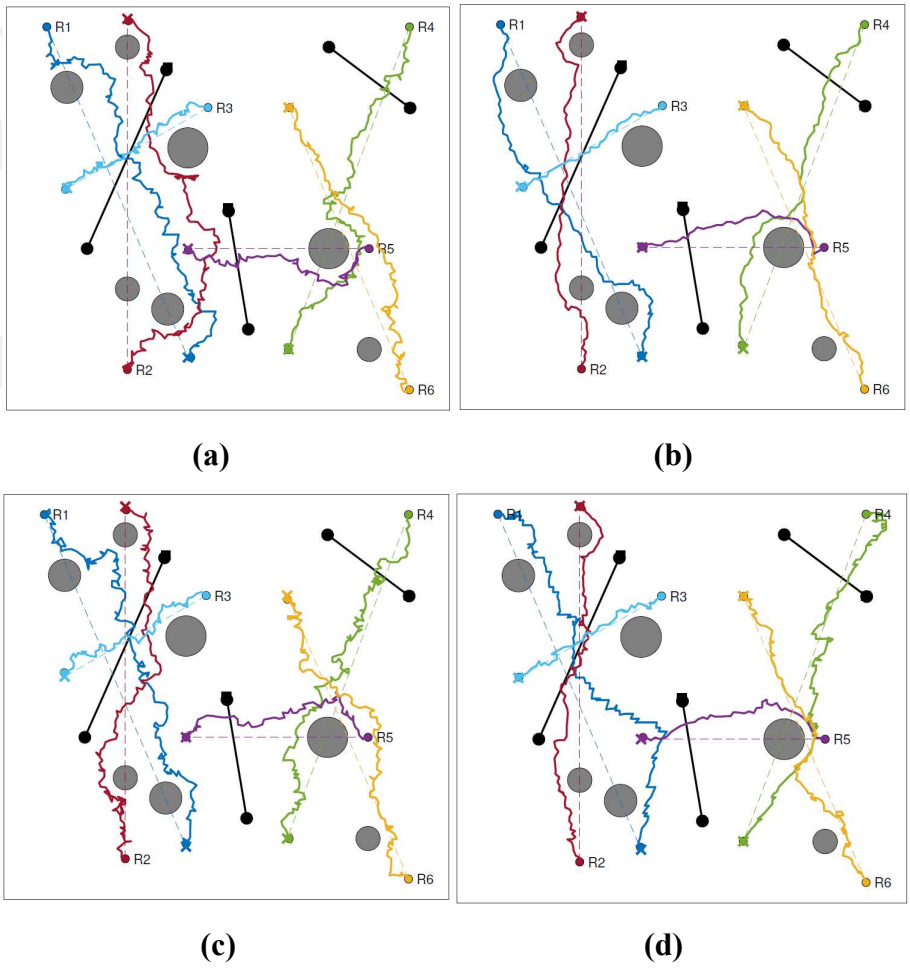
(a) **(b)**

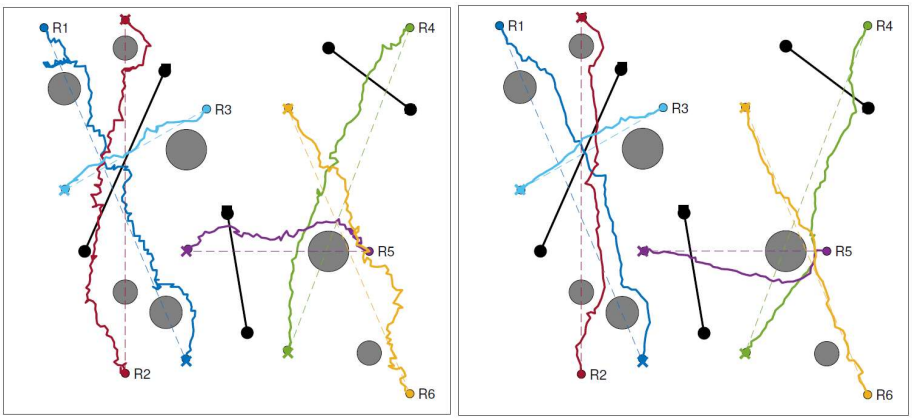
Şekil 4.32. SCA ve sdSCA için toplam robot sayısına göre ortalama kalan hedef uzaklığı ve ortalama çalışma süresi: **(a)** Ortalama kalan hedef uzaklığı, **(b)** Ortalama çalışma süresi

Çoklu robot sistemindeki toplam robot sayısı arttıkça problem de zorlaşmaktadır. Tablo 4.23, 4.24, 4.25 ve 4.26 dikkate alındığında, sdSCA hem performans ($OYSH$, $OKHU$, toplam uygunluk değeri) hem de ortalama çalışma süresi açısından temel SCA'ya göre oldukça önemli bir iyileştirme sağlamıştır. Dahası, robot sayısı artmasına rağmen iyileştirme oranı da artmıştır. En fazla iyileştirme $OYSH$'de görülmüş, iyileştirme oranı robot sayısı arttıkça %47.24'ten %81.72'ye yükselmiştir. Diğer kriterler nispeten daha az

olmasına rağmen, hepsinde iyileştirme oranının arttığı görülmektedir. Robot sayısı arttıkça iyileştirme oranları da $OKHU$ için %4.65'ten %38.92'ye, toplam uygunluk değeri için %4.34'ten %37.50'ye ve ortalama çalışma süresi için %3.37'den %35.09'a yükselmiştir. Bu sonuçlar sdSCA'nın yüksek verimliliğini göstermektedir.

Önerilen Algoritmanın Ortam 17'de Gerçekleştirimi: Bu ortamda 6 robot olduğundan, Alt Başlık 4.4.1'de belirtildiği gibi problem boyutu (D) 12 olur. Tüm algoritmalar 30 kez çalıştırılmıştır. Ortam 17'de algoritmalar tarafından elde edilen örnek yollar Şekil 4.33'te gösterilmektedir.





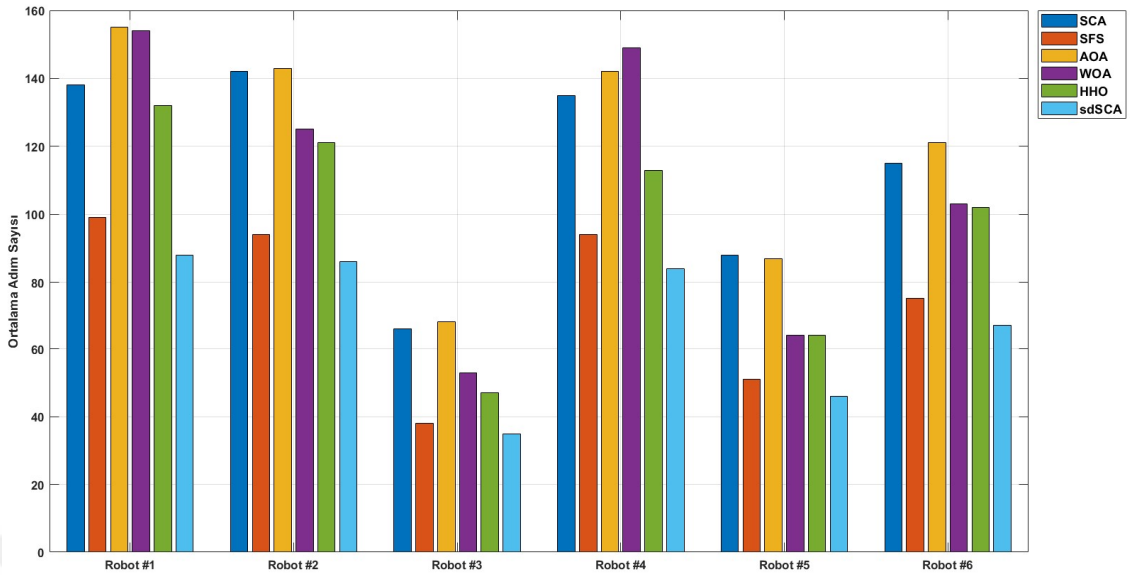
(e) **(f)**

Şekil 4.33. Ortam 17'de algoritmalar tarafından elde edilen örnek yollar: **(a)** SCA, **(b)** SFS, **(c)** AOA, **(d)** WOA, **(e)** HHO, **(f)** sdSCA

Örnek yollar göz önüne alındığında, önerilen algoritma tarafından planlanan yolun daha kısa ve düzgün olduğu görülmektedir. Diğer algoritmaların planladığı yollar daha kıvrımlıdır ve bu durum yolların daha uzun olmasına neden olur. Ortam 17'de ortalama adım sayısı karşılaştırması, ortalama adım sayısının çubuk grafiği ve ortalama yol uzunluğu karşılaştırması sırasıyla Tablo 4.26, Şekil 4.34 ve Tablo 4.27'de gösterilmektedir. Ayrıca, algoritmalar $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi açısından karşılaştırılmıştır ve bu bulgular Tablo 4.28'de gösterilmektedir.

Tablo 4.26. Ortam 17'de her robot ve toplam için ortalama adım sayısı karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	138	99	155	154	132	88
Robot #2	142	94	143	125	121	86
Robot #3	66	38	68	53	47	35
Robot #4	135	94	142	149	113	84
Robot #5	88	51	87	64	64	46
Robot #6	115	75	121	103	102	67
Toplam	684	451	716	648	579	406



Şekil 4.34. Ortam 17'de her robot için ortalama adım sayısının çubuk grafiği

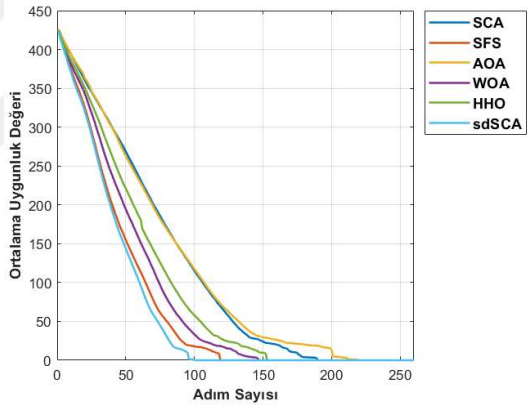
Tablo 4.27. Ortam 17'de her robot ve toplam için ortalama yol uzunluğu (br) karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	167.81	117.04	190.72	177.61	158.95	105.58
Robot #2	171.64	110.76	176.83	144.71	146.20	103.70
Robot #3	80.74	45.95	83.69	62.71	57.78	43.43
Robot #4	165.05	110.44	175.19	172.32	136.82	101.68
Robot #5	106.63	60.90	107.56	75.77	78.47	54.82
Robot #6	140.58	89.52	149.52	119.91	124.17	82.04
Toplam	832.48	534.63	883.54	753.04	702.41	491.27

Tablo 4.28. Ortam 17'de $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
$OYSH$ (br)	409.41	111.56	460.47	329.97	279.35	68.20
$OKHU$ (br)	27976	17544	29693	27995	23669	15923
Toplam Uygunluk Değeri	28501	18244	30009	28156	24850	16513
Ortalama Çalışma Süresi (s)	41.49	34.51	39.15	35.78	64.88	18.54

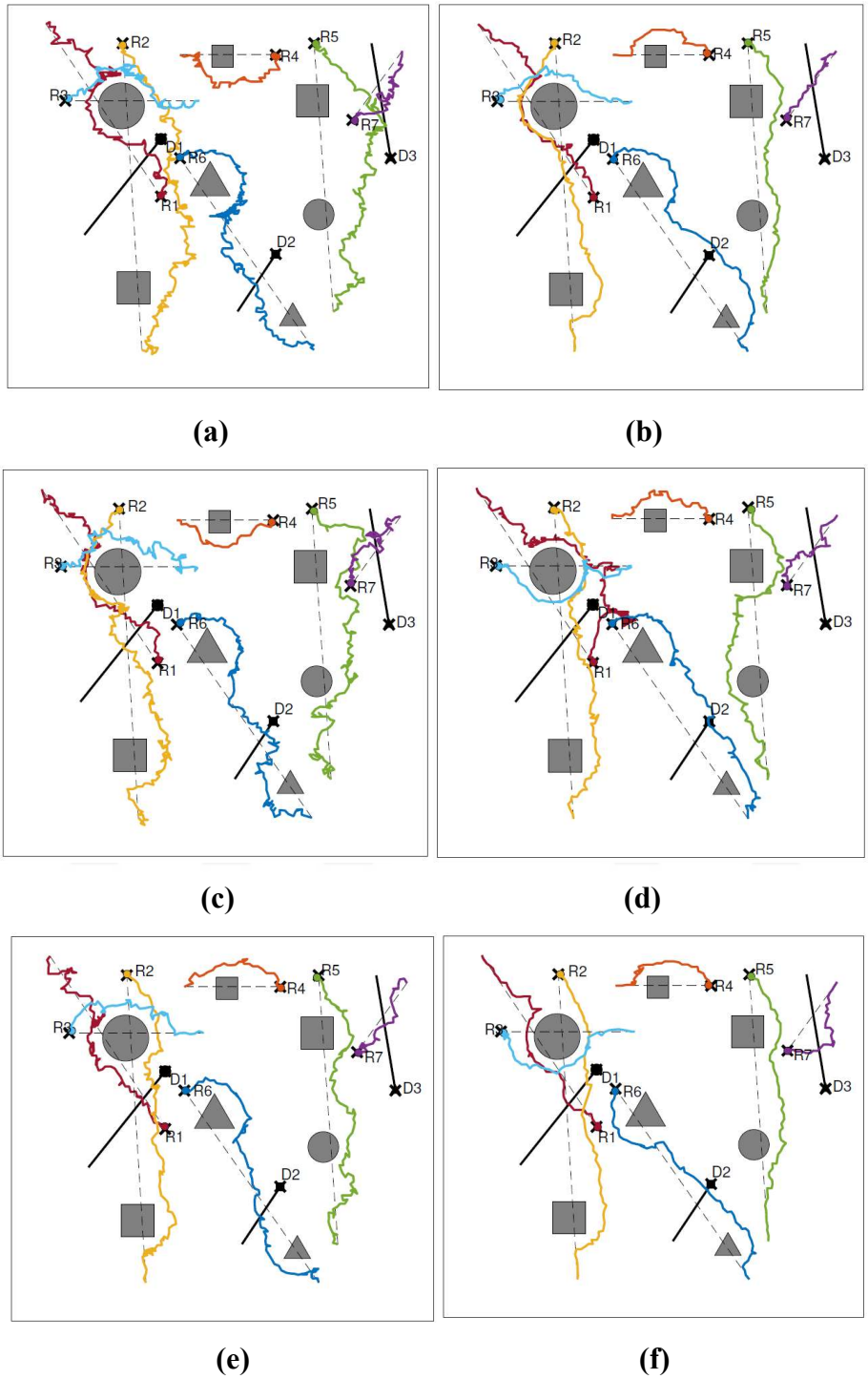
Önerilen algoritma, tüm robotların hedef noktalarına minimum adımda ulaşmasını sağlamıştır. Temel SCA'ya kıyasla toplam adım sayısını %40.63 oranında azaltmıştır. Ayrıca, yol uzunlukları değerlendirildiğinde, önerilen algoritmanın hem temel SCA'dan hem de diğer metasezgisel algoritmalardan daha iyi performans gösterdiği görülmektedir. Temel SCA'ya kıyasla ortalama yol uzunluğunu %40.98 oranında azaltmıştır. Önerilen algoritma, adım sayısı ve yol uzunluğu açısından tüm robotlar için başarılı olmuştur. Ayrıca, Tablo 4.28'deki karşılaştırma bulguları da bu sonucu desteklemektedir. Özellikle, önerilen algoritmanın yol planlama sürecini diğer algoritmalara kıyasla daha kısa sürede tamamlaması performansını kanıtlamaktadır. Sonuç olarak, önerilen algoritma her kriterde en iyi performansı göstermiştir. AOA en kötü performansı gösterirken, HHO yol planlama sürecini en uzun sürede tamamlayan algoritma olmuştur. Ortam 17 için her algoritmanın adım sayısına göre yakınsama eğrileri Şekil 4.35'te gösterilmektedir.



Şekil 4.35. Ortam 17 için her algoritmanın adım sayısına göre yakınsama eğrileri

Şekil 4.35'te önerilen algoritmanın Ortam 17'de diğer algoritmalardan daha hızlı yakınsadığı ve diğer bulguları desteklediği görülmektedir.

Önerilen Algoritmanın Ortam 18'de Gerçekleştirimi: Bu ortamda 7 robot olduğundan problem boyutu (D) 14 olur. Tüm algoritmalar 30 kez çalıştırılmıştır. Ortam 18'de algoritmalar tarafından elde edilen örnek yollar Şekil 4.36'da gösterilmektedir.



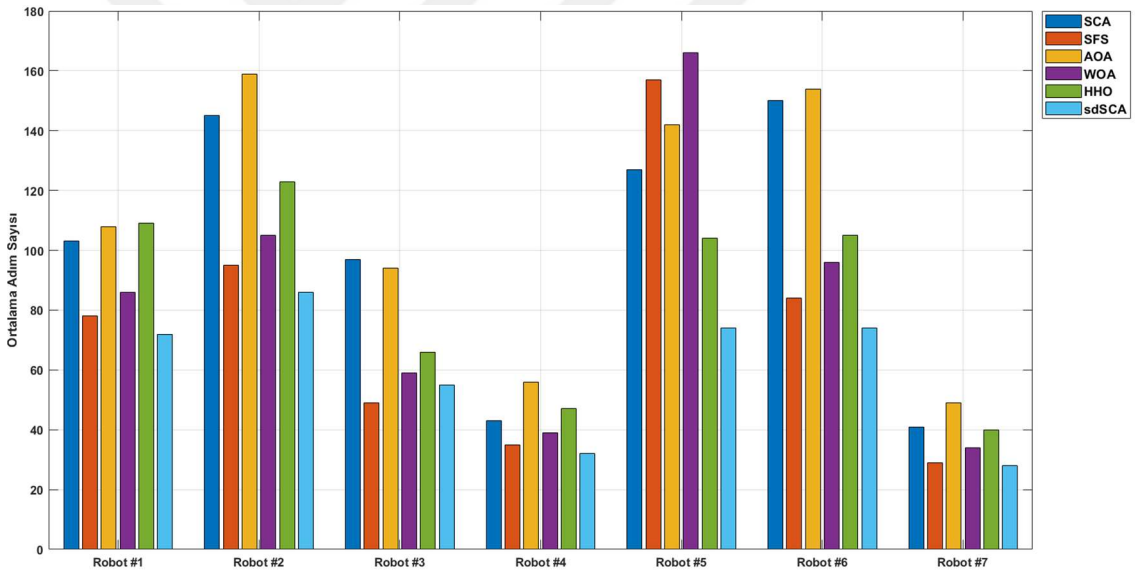
Şekil 4.36. Ortam 18'de algoritmalar tarafından elde edilen örnek yollar: **(a)** SCA, **(b)** SFS, **(c)** AOA, **(d)** WOA, **(e)** HHO, **(f)** sdSCA

Örnek yollar göz önüne alındığında, önerilen algoritma tarafından planlanan yolun Ortam 17'deki gibi daha kısa ve düzgün olduğu görülmektedir. Ortam 18'de ortalama adım sayısı karşılaştırması, ortalama adım sayısının çubuk grafiği ve ortalama yol uzunluğu karşılaştırması sırasıyla Tablo 4.29, Şekil 4.37 ve Tablo 4.30'da gösterilmektedir. Ayrıca,

algoritmalar $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi açısından karşılaştırılmıştır ve bu bulgular Tablo 4.31'de gösterilmektedir.

Tablo 4.29. Ortam 18'de her robot ve toplam için ortalama adım sayısı karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	103	78	108	86	109	72
Robot #2	145	95	159	105	123	86
Robot #3	97	49	94	59	66	55
Robot #4	43	35	56	39	47	32
Robot #5	127	157	142	166	104	74
Robot #6	150	84	154	96	105	74
Robot #7	41	29	49	34	40	28
Toplam	706	527	762	585	594	421



Şekil 4.37. Ortam 18'de her robot için ortalama adım sayısının çubuk grafiği

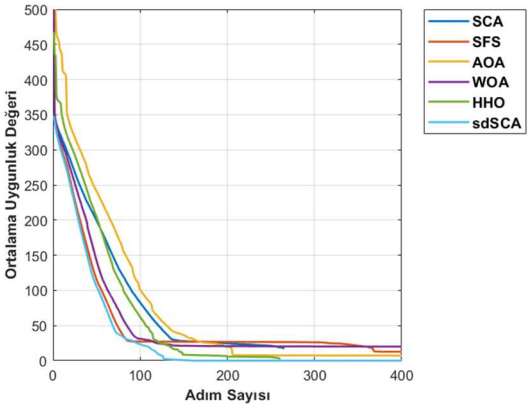
Tablo 4.30. Ortam 18'de her robot ve toplam için ortalama yol uzunluğu (br) karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	125.03	90.40	132.63	102.48	129.93	85.74
Robot #2	174.81	109.18	194.58	124.18	145.73	103.15
Robot #3	116.09	56.91	114.23	70.09	79.02	64.68
Robot #4	51.91	39.95	69.50	47.23	56.89	37.95
Robot #5	153.60	172.45	174.31	185.82	124.40	88.92
Robot #6	179.61	96.13	187.49	112.55	125.34	88.44
Robot #7	49.49	33.08	60.42	40.91	48.33	34.08
Toplam	850.56	598.12	933.19	683.29	709.66	503.00

Tablo 4.31. Ortam 18'de $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
$OYSH$ (br)	502.90	250.46	585.54	335.63	362.00	155.34
$OKHU$ (br)	23472	30872	26134	43786	18975	13927
Toplam Uygunluk Değeri	25389	41374	44233	60575	22973	13870
Ortalama Çalışma Süresi (s)	22.60	28.76	26.59	26.32	34.16	13.59

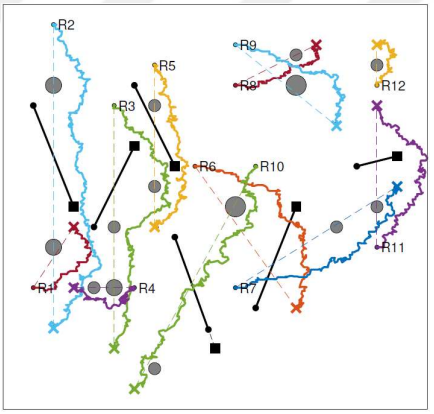
Ortam 18'de, adım sayıları ve yol uzunlukları değerlendirildiğinde, önerilen algoritmanın genel olarak diğer algoritmalardan daha iyi performans gösterdiği görülmektedir. Yol uzunluğu açısından 7 robottan 6'sı için diğer algoritmalardan daha iyi performans göstermiştir. Ancak, üçüncü robot için SFS ve sdSCA'nın yol uzunluğu açısından performansları arasında anlamlı bir fark olmadığı söylenebilir. Ayrıca, toplam dikkate alındığında, önerilen algoritmanın diğer tüm algoritmalardan üstün olduğu görülmektedir. Önerilen algoritma, temel SCA'ya kıyasla adım sayısını %40.36 ve yol uzunluğunu %40.86 oranında azaltmıştır. Tablo 4.31'de, önerilen algoritmanın diğer algoritmalara kıyasla tüm kriterlerde en iyi performansı gösterdiği görülmektedir. Ayrıca, Ortam 17'de olduğu gibi, Ortam 18'de de en kötü performansa sahip algoritmanın AOA, yol planlama sürecini en uzun sürede tamamlayan algoritmanın ise HHO olduğu görülmektedir. Ortam 18 için her algoritmanın adım sayısına göre yakınsama eğrileri Şekil 4.38'de gösterilmektedir.



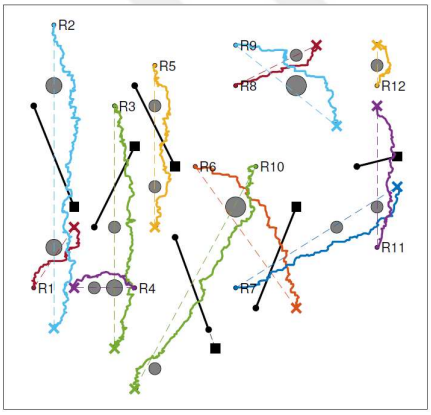
Şekil 4.38. Ortam 18 için her algoritmanın adım sayısına göre yakınsama eğrileri

Şekil 4.38'de önerilen algoritmanın Ortam 18'de diğer algoritmalardan daha hızlı yakınsadığı ve diğer bulguları desteklediği görülmektedir.

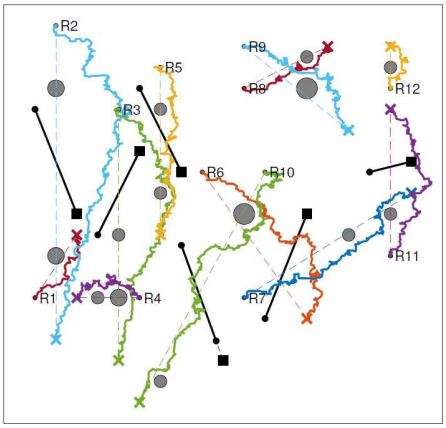
Önerilen Algoritmanın Ortam 19'da Gerçekleştirimi: Bu ortamda 12 robot olduğundan problem boyutu (D) 24 olur. Tüm algoritmalar 30 kez çalıştırılmıştır. Ortam 19'da algoritmalar tarafından elde edilen örnek yollar Şekil 4.39'da gösterilmektedir.



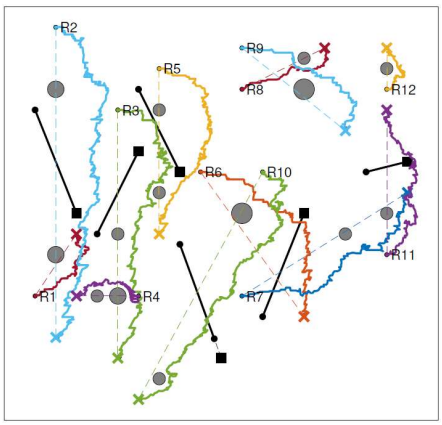
(a)



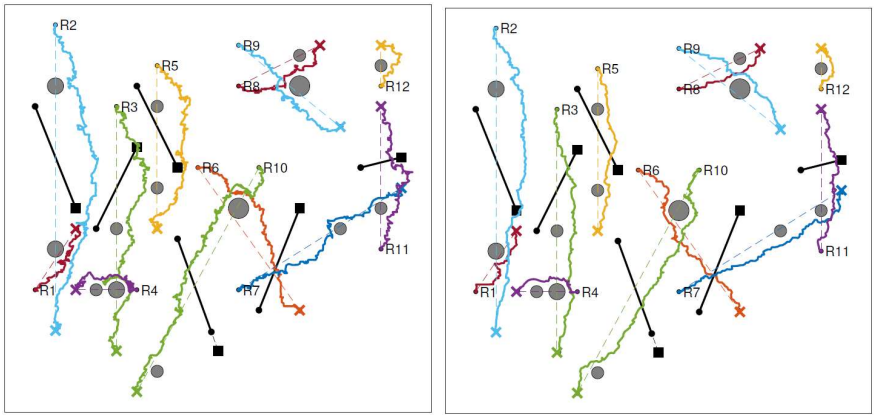
(b)



(c)



(d)



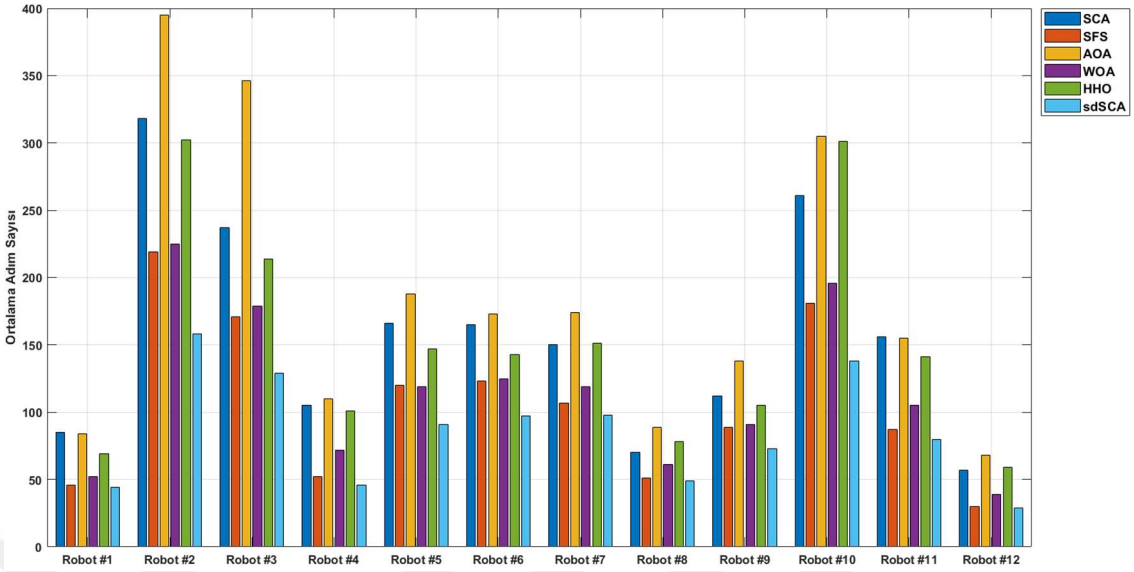
(e) **(f)**

Şekil 4.39. Ortam 19'da algoritmalar tarafından elde edilen örnek yollar: **(a)** SCA, **(b)** SFS, **(c)** AOA, **(d)** WOA, **(e)** HHO, **(f)** sdSCA

Örnek yollar göz önüne alındığında, önerilen algoritma tarafından planlanan yolun Ortam 17 ve 18'e benzer şekilde daha kısa ve düzgün olduğu görülmektedir. Ortam 19'da ortalama adım sayısı karşılaştırması, ortalama adım sayısının çubuk grafiği ve ortalama yol uzunluğu karşılaştırması sırasıyla Tablo 4.32, Şekil 4.40 ve Tablo 4.33'te gösterilmektedir. Ayrıca, algoritmalar $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi açısından karşılaştırılmıştır ve bu bulgular Tablo 4.34'te gösterilmektedir.

Tablo 4.32. Ortam 19'da her robot ve toplam için ortalama adım sayısı karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	78	47	85	65	73	44
Robot #2	294	220	312	386	268	160
Robot #3	241	174	262	274	215	128
Robot #4	101	53	114	98	100	46
Robot #5	176	118	187	174	146	93
Robot #6	165	126	177	165	149	98
Robot #7	148	107	178	155	152	98
Robot #8	71	52	83	64	77	48
Robot #9	115	88	138	111	110	74
Robot #10	250	183	291	299	248	138
Robot #11	154	88	159	145	143	79
Robot #12	59	32	59	49	56	31
Toplam	1852	1288	2045	1985	1737	1037



Şekil 4.40. Ortam 19'da her robot için ortalama adım sayısının çubuk grafiği

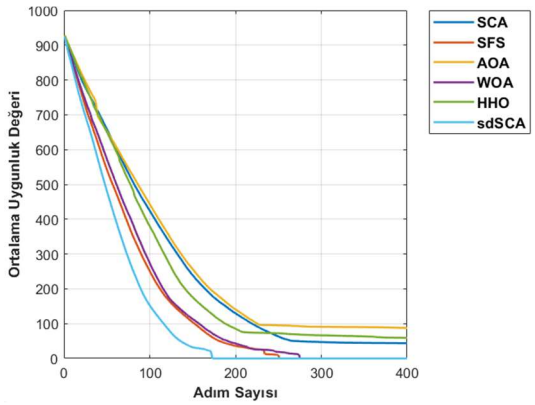
Tablo 4.33. Ortam 19'da her robot ve toplam için ortalama yol uzunluğu (br) karşılaştırması

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
Robot #1	95.51	54.54	104.60	72.83	86.13	53.16
Robot #2	357.87	248.67	381.84	429.68	313.50	194.78
Robot #3	294.43	199.42	321.58	308.14	253.31	156.00
Robot #4	121.73	59.80	138.61	110.62	118.54	54.74
Robot #5	214.93	135.88	228.76	199.33	174.56	112.80
Robot #6	201.54	145.61	217.61	189.09	178.21	119.83
Robot #7	182.70	123.54	219.65	177.38	181.87	121.14
Robot #8	86.94	60.47	102.39	71.65	91.71	58.56
Robot #9	140.71	102.38	169.60	126.65	131.38	89.65
Robot #10	304.99	207.99	355.06	336.41	290.37	166.75
Robot #11	188.05	101.81	194.75	165.77	170.85	95.73
Robot #12	71.39	36.80	72.15	54.05	66.11	37.10
Toplam	2260.8	1477.0	2506.7	2241.7	2056.6	1260.3

Tablo 4.34. Ortam 19'da $OYSH$, $OKHU$, toplam uygunluk değeri ve ortalama çalışma süresi karşılaştırmaları

	SCA	**SFS**	**AOA**	**WOA**	**HHO**	**sdSCA**
$OYSH$ (br)	1340.4	556.5	1586.2	1321.2	1136.1	339.8
$OKHU$ (br)	95334	84580	152250	107690	95317	65262
Toplam Uygunluk Değeri	97323	68086	115160	106860	90128	53926
Ortalama Çalışma Süresi (s)	118.38	94.75	128.32	159.01	211.65	63.99

Önerilen algoritma, Ortam 17 ve 18'de olduğu gibi Ortam 19'da de tüm robotların minimum adım sayısında hedef noktalarına ulaşmasını sağlamıştır. Yol uzunlukları değerlendirildiğinde, önerilen algoritmanın genel olarak diğer algoritmalardan daha iyi performans gösterdiği görülmektedir. Önerilen algoritma, adım sayısı açısından tüm robotlar için başarılı olsa da, yol uzunluğu açısından 12 robottan 11'i için diğer algoritmalardan daha iyi performans göstermiştir. Ancak, 12. robot için SFS ve sdSCA'nın performansları arasında anlamlı bir fark olmadığı söylenebilir. Ayrıca, toplam dikkate alındığında, önerilen algoritmanın diğer tüm algoritmalardan üstün olduğu görülmektedir. Önerilen algoritma, temel SCA'ya kıyasla adım sayısını %44.00 ve yol uzunluğunu %44.25 oranında azaltmıştır. Tablo 4.34'te, önerilen algoritmanın diğer algoritmalara kıyasla tüm kriterlerde en iyi performansı gösterdiği görülmektedir. Ayrıca, Ortam 17 ve 18'de olduğu gibi, Ortam 19'da de en kötü performansa sahip algoritmanın AOA, yol planlama sürecini en uzun sürede tamamlayan algoritmanın ise HHO olduğu görülmektedir. Ortam 19 için her algoritmanın adım sayısına göre yakınsama eğrileri Şekil 4.41'de gösterilmektedir.



Şekil 4.41. Ortam 19 için her algoritmanın adım sayısına göre yakınsama eğrileri

Şekil 4.41'de Ortam 19'da önerilen algoritmanın diğer algoritmalardan daha hızlı yakınsadığı ve diğer bulguları desteklediği görülmektedir. Sonuç olarak, bu çalışmada önerilen algoritmanın performansı farklı zorluk seviyelerine sahip üç farklı ortamda test edilmiş ve önerilen algoritmanın literatürdeki güncel metasezgisel algoritmalardan üstün olduğu kanıtlanmıştır.

4.5. Izgara Haritalarda ResNet Tabanlı Çoklu-Robot Yol Planlaması

Izgara tabanlı mobil robot yol planlama problemlerinde standart FCN mimarileri derinleştikçe kaybolan gradyan problemi yaşayabilir ve bu durum ağı derinleştirmenin etkinliğini sınırlayabilir. Bu problem, özellikle yol planlama gibi karmaşık görevlerde robotun çevresindeki engellerin doğru bir şekilde tanınması ve bu engellere karşı verimli bir yol haritası çıkarılması için kritik öneme sahiptir. Gradyan kaybolması, geri yayılım sırasında gradyanların hızla küçülmesi nedeniyle ağın daha derin katmanlarının etkili bir şekilde öğrenmesini engeller. Bu da, robotun çevresindeki karmaşık engelleri ve yol tıkanıklıklarını doğru bir şekilde öğrenmesini zorlaştırır. Ayrıca, standart FCN'ler, artık (residual) katmanlar içermediğinden, daha derin katmanların öğrenmeye katkısı sınırlıdır, bu da ağın genel öğrenme sürecini verimsiz hale getirir. Bu durum, robotun yol planlama görevindeki öğrenme kapasitesini kısıtlar ve modelin daha fazla katman eklemenin fayda sağlamadığı sınırlı çözünürlükteki haritalar üretmesine yol açar. Bunun yanı sıra, FCN'lerde alıcı alanı genellikle sınırlıdır. Bu da robotun çevresindeki geniş bağlamsal ilişkileri öğrenme yeteneğini engeller. Özellikle yol planlama gibi görevlerde, robotun çevresindeki engelleri ve uygun yolları daha geniş bir bağlamda anlaması gerekmektedir. Algılama alanının sınırlı olması modelin yalnızca yerel bilgiyle sınırlı kalmasına neden olur, bu da genelleme yeteneğini zayıflatır ve modelin aşırı öğrenme gibi problemlerle karşılaşmasına yol açar. Standart FCN'lerin bu sınırlamalarını aşmak amacıyla, bu çalışmada genişletilmiş evrişim (dilated convolution), sıkma-uyarma bloğu (squeeze-and-excitation) ve dropout içeren iyileştirilmiş bir ResNet (IResNet) mimarisi önerilmiştir. Önerilen modelin ana katkıları şunlardır:

- Genişletilmiş evrişim kullanılarak alıcı alan genişletilmiştir. Böylece hem yerel hem de küresel özelliklerin daha etkili şekilde öğrenilmesi ve gelişmiş bağlamsal bilgi edinilmesi sağlanmıştır.

- Sıkma-uyarma bloğu eklenerek kanal bazında dikkat mekanizması sağlanmıştır. Böylece modelin önemli görsel özelliklere odaklanması ve gereksiz bilgilerin baskılanarak daha verimli öğrenme sürecinin gerçekleştirilmesi hedeflenmiştir.

- Toplu normalizasyonun (batch norm) etkisini desteklemek amacıyla dropout kullanılmıştır. Böylece modelin daha sağlam genelleme yapması sağlanmış ve aşırı öğrenme riski azaltılmıştır.

4.5.1. Problem Tanımı

Bu çalışma, bir mobil robotun engellere çarpmadan hedefe ulaşmasını sağlayacak optimum yol planlamasını tahmin etmeyi amaçlamaktadır. Robotun başlangıç ve bitiş noktası verildiğinde robotun en az sayıda ızgara hücresinden geçerek hedefe ulaşması gereken yol, derin öğrenme kullanılarak tahmin edilmiştir. Çalışmada iki boyutlu bir ızgara haritası kullanılmıştır. Bu harita, ikili bir görüntü formatında temsil edilir. Her bir hücre boş (0) veya engelli (1) olabilir. Robotun yolu, 1'ler ile temsil edilen hücreler üzerinden geçerken, 0'lar engel olmayan boş hücrelerdir. Bu ortamlar, görüntü olarak modellenir, burada boş alanlar beyaz (0) ve engeller siyah (1) olarak kodlanır. Yol planlaması m x n boyutunda bir ikili ızgara görüntüsü M_{truth} (gerçek yol haritası) ile verilecektir. Model M_{pred} (tahmin edilen yol haritası) adlı bir çıkış üretir. Bu çıkış robotun geçmesi gereken hücrelerin 1 olarak işaretlendiği bir ikili matristir. Modelin amacı M_{pred} ile M_{truth} arasındaki farkı minimize etmektir. Buradaki fark Eşitlik (4.36)'da gösterilen MSE kayıp fonksiyonu ile hesaplanır.

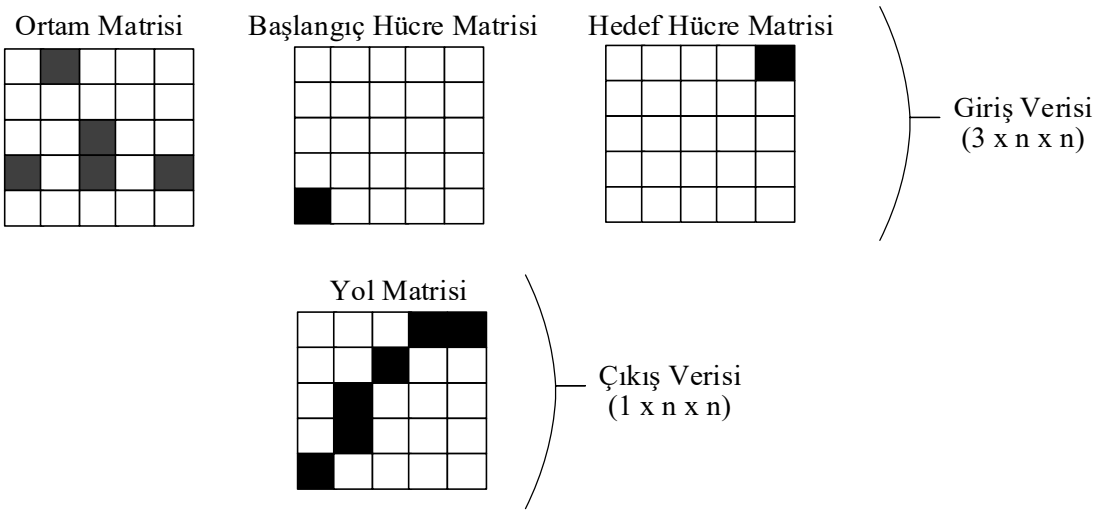
$$MSE = \frac{1}{n_p}\sum_{i=1}^{n_p}\left(M_{truth}(i) - M_{pred}(i)\right)^2 \tag{4.36}$$

Burada n_p harita üzerindeki toplam hücre sayısını, $M_{truth}(i)$ ve $M_{pred}(i)$ ise sırasıyla gerçek yol ve tahmin edilen yol haritasındaki i'inci hücrenin değerini temsil eder. Tahmin edilen yol haritası, doğru şekilde belirlenen 1'ler ile başlangıç noktasından hedefe giden en kısa yolu temsil eder. Modelin görevi optimum yolun doğru şekilde tahmin edilmesini sağlamaktır. Bu, her hücredeki tahminin, gerçek yol ile ne kadar yakın olduğunu ölçerek minimize edilir. MSE ağın her iki harita arasındaki farkı minimize etmeyi amaçlar.

4.5.2. Önerilen Yöntem

4.5.2.1. Veri Seti

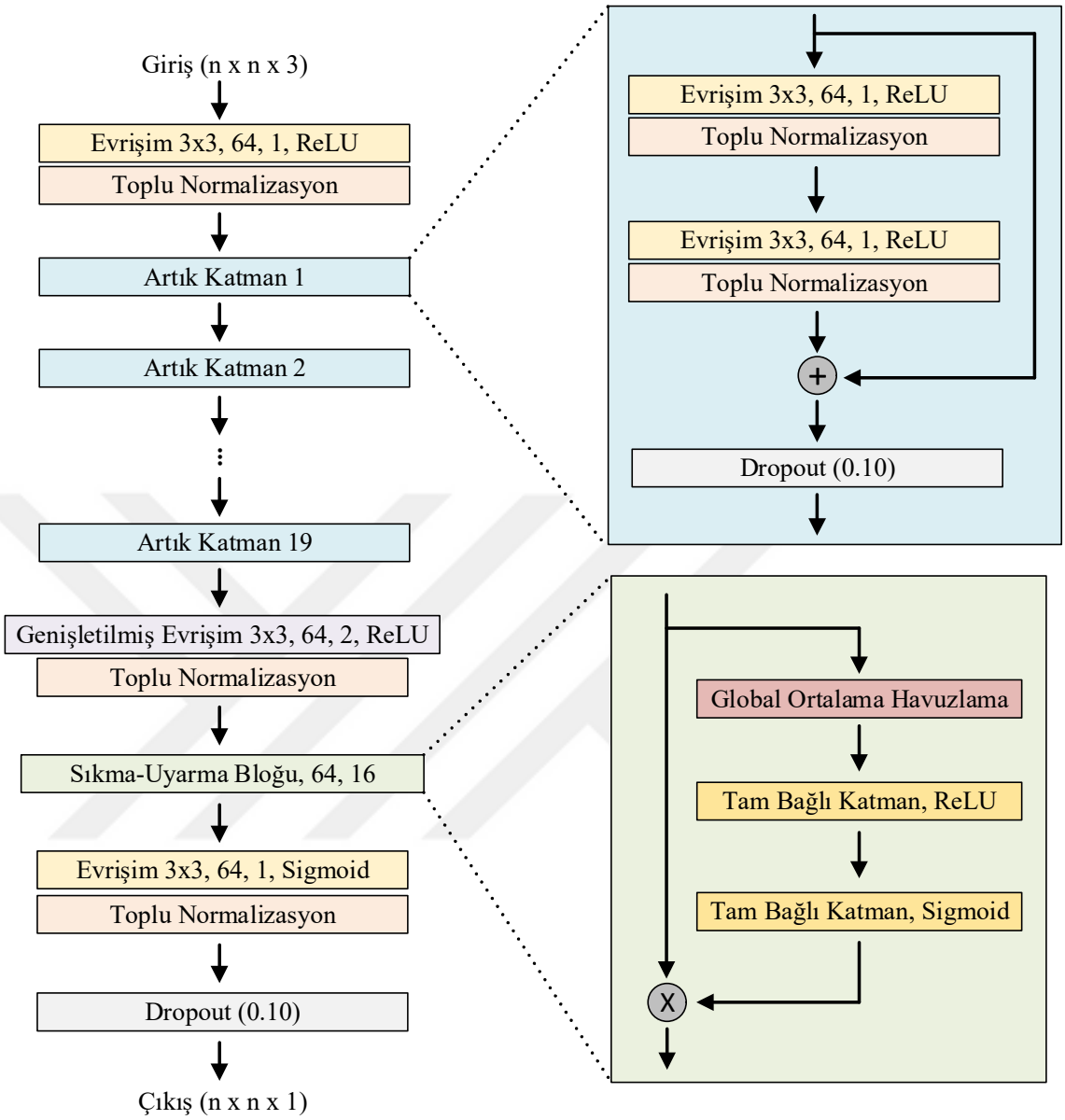
Bu çalışmada [110]'daki veri seti kullanılmıştır. Ağın girişine üç farklı veri sağlanmaktadır: ortam matrisi, başlangıç noktası matrisi ve hedef noktası matrisi. Ortam matrisi engel bilgilerini içerir ve n x n boyutundadır. Bu matriste engeller 1, boş alanlar 0 ile temsil edilmektedir. Engel dağılımı rastgele belirlenmiş ve her hücrenin engel olma olasılığı %60 olarak atanmıştır. Başlangıç ve hedef hücre matrisleri de aynı boyutta olup, sadece ilgili hücrede 1 değeri bulunurken diğer hücreler 0'dır. Modelin çıktısı ise tahmin edilen yolun 1'lerle gösterildiği n x n boyutunda tek bir matristir. Bu matristeki yollar optimum yol olarak ifade edilmiş ve A* algoritması ile 8 yönlü olarak planlanmıştır. Çalışmada toplam 30000 veri örneği kullanılmış ve üç farklı ortam boyutu ele alınmıştır: 10 x 10 br^2, 15 x 15 br^2 ve 20 x 20 br^2. Tekli robot senaryolarında başlangıç ve hedef hücreleri farklı ve rastgele olacak şekilde belirlenmiştir. Çoklu robot senaryoları ise yalnızca 15 x 15 br^2 boyutundaki ortamlarda test edilmiş ve bu senaryolarda tüm robotlar aynı hedef hücresine yönlendirilmiştir. İki robotlu durumda başlangıç hücreleri (1, 1) ve (1, 15), üç robotlu durumda ise (1, 1), (1, 15) ve (15, 1) olarak belirlenmiştir. Tüm senaryolarda hedef hücresi ortamın merkezinde bulunan (8, 8) hücresidir. Giriş ve çıkış verilerinin bir temsili Şekil 4.42'de gösterilmektedir.



Şekil 4.42. Giriş ve çıkış verilerinin bir temsili

4.5.2.2. Önerilen IResNet Mimarisi

Önerilen IResNet mimarisi, geleneksel artık ağ yapısını temel alarak, sıkma-uyarma ve genişletilmiş evrişim mekanizmalarının entegre edildiği 41 evrişim katmanlı bir derin öğrenme modelidir. Modelin giriş katmanı, adım uzunluğu 1 olan 3 x 3 boyutunda 64 filtre içeren bir evrişim katmanı ile başlamaktadır. Bu katman, ReLU aktivasyon fonksiyonu ile etkinleştirilmekte ve ardından bir toplu normalizasyon işlemi uygulanmaktadır. Modelin ana gövdesini her biri 64 filtre içeren 19 adet artık blok oluşturmaktadır. Her artık blok 3 x 3 boyutunda, adım uzunluğu 1 olan ve ReLU aktivasyon fonksiyonuna sahip iki evrişim katmanından meydana gelmektedir. Bu katmanlardan her birine toplu normalizasyon işlemi uygulanarak modelin istikrarı artırılmıştır. Artık bağlantı mekanizması ile giriş ve çıkış bilgileri toplanarak öğrenme sürecinin verimliliği yükseltilmiştir. Her artık blok sonunda aşırı öğrenmeyi önlemek amacıyla %10 oranında dropout uygulanmıştır. Artık blokların ardından, modelin daha geniş bir alandan özellik çıkarabilmesi için genişletilmiş evrişim tekniği kullanılmıştır. Burada genişleme oranı 2 olan 64 filtreli 3 x 3 boyutunda bir evrişim katmanı eklenmiştir. Bu işlem, daha büyük alansal ilişkileri öğrenerek modelin daha güçlü temsil gücüne sahip olmasını sağlamaktadır. Modelde dikkat mekanizmasını güçlendirmek amacıyla, sıkıştırma-uyarma bloğu kullanılmıştır. Kanal sayısı 64 ve sıkıştırma oranı 16 olarak belirlenmiştir. Dolayısıyla, 4 nöronlu-ReLU aktivasyonlu ve 64 nöronlu-sigmoid aktivasyonlu iki tam bağlantılı tasarlanmış ve giriş tensörüne uygulanmıştır. Çıkış aşamasında tek kanal içeren bir çıktı üretmek amacıyla, 1 filtreli ve adım uzunluğu 1 olan 3 x 3 boyutunda bir evrişim katmanı eklenmiştir. Bu katmanda sigmoid aktivasyon fonksiyonu ile çıktı değerleri 0 ile 1 arasında normalize edilmiştir. Son olarak toplu normalizasyon işlemi ve %10 oranında dropout uygulanarak model tamamlanmıştır. Önerilen IResNet mimarisi Şekil 4.43'te gösterilmektedir.



Şekil 4.43. Önerilen IResNet mimarisi

4.5.2.3. Değerlendirme Metrikleri

Önerilen modelin tahmin ettiği yollar, eğitim verisinde A* algoritması ile planlanan en kısa yollar ile karşılaştırılarak değerlendirilmiştir. Model performansını ölçmek amacıyla başarı oranı ve yol optimalitesi metrikleri kullanılmıştır. Başarı oranı (η_{sc}), test ortamlarında modelin başlangıç ve hedef hücreleri arasında geçerli bir yol planlayabildiği durumların yüzdesi olarak tanımlanmış Eşitlik (4.37) kullanılarak hesaplanmıştır.

$$\eta_{sc} = \left(\frac{n_{sc}}{n_t}\right) \cdot 100\% \tag{4.37}$$

Burada, n_{sc} model tarafından başarılı şekilde üretilen yolların sayısını, n_t ise toplam test ortamı sayısını temsil eder. Yol optimalitesi (η_{op}), model tarafından üretilen yolların, A* algoritması tarafından hesaplanan en kısa yollar ile aynı olup olmadığını belirlemek amacıyla kullanılmıştır. Modelin tahmin ettiği yolların uzunluğu ($L_{IResNet}$) A* algoritması ile elde edilen en kısa yolun uzunluğu (L_{A^*}) ile karşılaştırılmıştır. Eğer $L_{IResNet} = L_{A^*}$ koşulu sağlanıyorsa, yol optimal kabul edilmiştir. Yol optimalitesi Eşitlik (4.38) kullanılarak hesaplanmıştır.

$$\eta_{op} = \left(\frac{n_{op}}{n_t}\right) \cdot 100\% \tag{4.38}$$

Burada, n_{op} model tarafından planlanan optimal yolların sayısını temsil eder. Bu metrikler, modelin genel başarımını ve en kısa yolu üretebilme yetisini değerlendirmek için kullanılmıştır.

4.5.3. Bulgular

Bu çalışmada, önerilen IResNet modelinin performansı hem tekli hem de çoklu robot senaryolarında test edilmiş ve [110]'daki FCN ile elde edilen sonuçlarla kıyaslanmıştır. Tüm simülasyonlar Python programlama dilinde kodlanmış, Intel Core i7-1255U işlemci ve 32 GB RAM'e sahip bir bilgisayarda çalıştırılmıştır. Önerilen model, sadece tekli robot senaryosunda eğitilmiş, ancak eğitilen model hem tekli hem de çoklu robot senaryolarında test edilmiştir. Tüm eğitim parametreleri [110]'daki ayarlarla aynıdır. Modelin eğitiminde toplam 30000 veriden 26000'i eğitim, 2000'i ise doğrulama için ayrılmıştır. Modelin daha önce görmediği ortamlarda genel performansını değerlendirmek amacıyla, 2000 test verisi yalnızca eğitimde kullanılmayan ortamları içermektedir. Çoklu ortam senaryosundaki analiz için ise 1000 test verisi kullanılmıştır. Model 10 x 10 br^2, 15 x 15 br^2 ve 20 x 20 br^2 boyutlu ortamlar için ayrı ayrı eğitilmiştir. Eğitim sürecinde, model Adam optimizasyon algoritması kullanılarak MSE kaybı ile eğitilmiştir. Performans ölçütü olarak doğruluk metriği izlenmiştir. Modelin aşırı öğrenmesini önlemek amacıyla, erken durdurma mekanizması uygulanmıştır. Erken durdurma, doğrulama doğruluğunu izleyerek 10 epoch boyunca iyileşme gözlemlenmediğinde eğitimi sonlandıracak şekilde ayarlanmıştır. Ayrıca, modelin en iyi ağırlıklarını kaydetmek için ModelCheckpoint kullanılmış ve doğrulama doğruluğu en yüksek olan modelin ağırlıkları saklanmıştır. Eğitim verileri uygun bir formatta yeniden

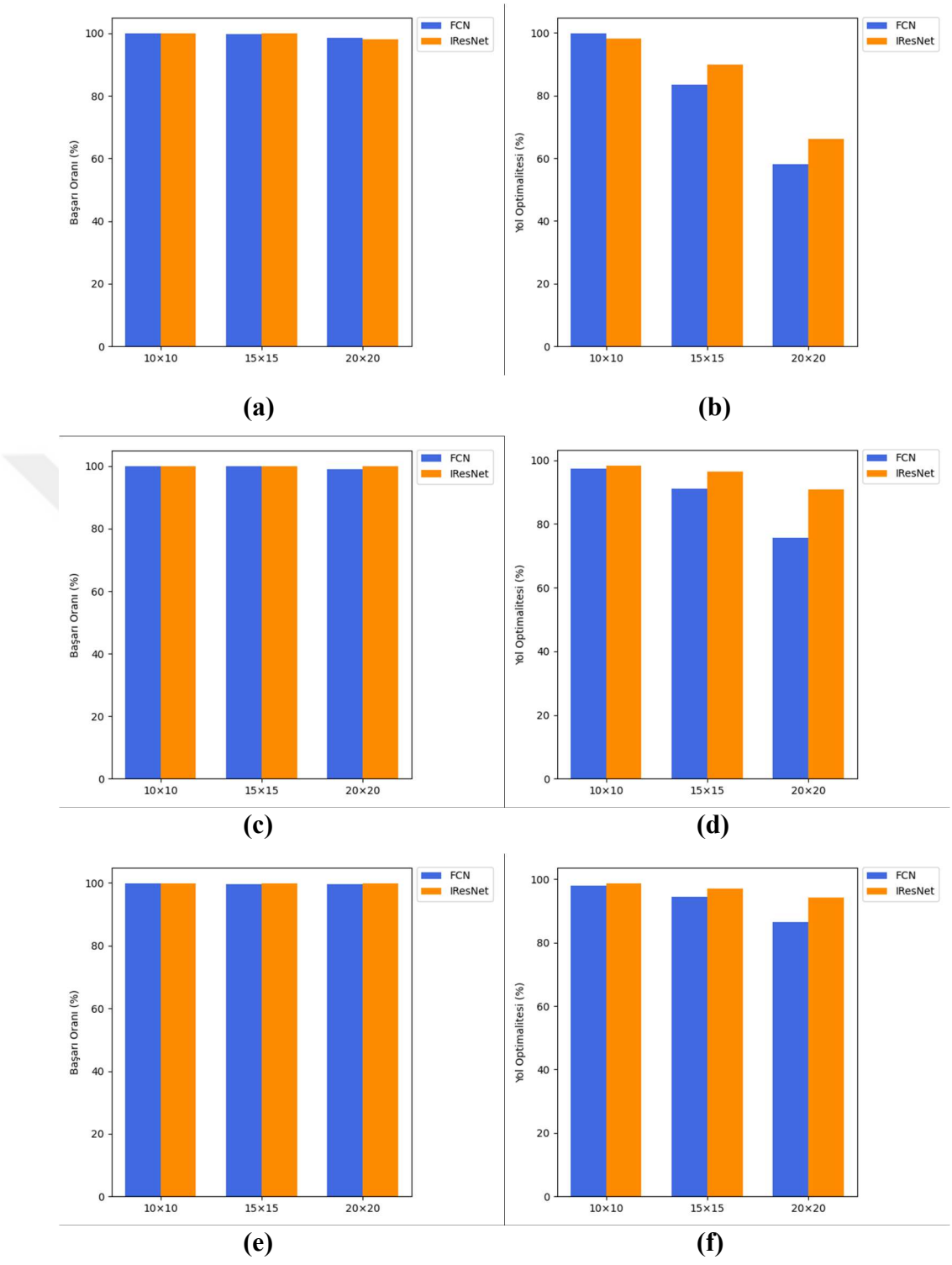
şekillendirilerek modele verilmiş, veriler 64'lük batch boyutunda işlenmiş ve epoch sayısı 100 olarak ayarlanmıştır. Eğitim süreci boyunca modelin performansı takip edilerek en iyi sonuçları veren ağırlıklar kaydedilmiş ve analiz için kullanılmıştır.

4.5.3.1. Tekli Robot Senaryosu

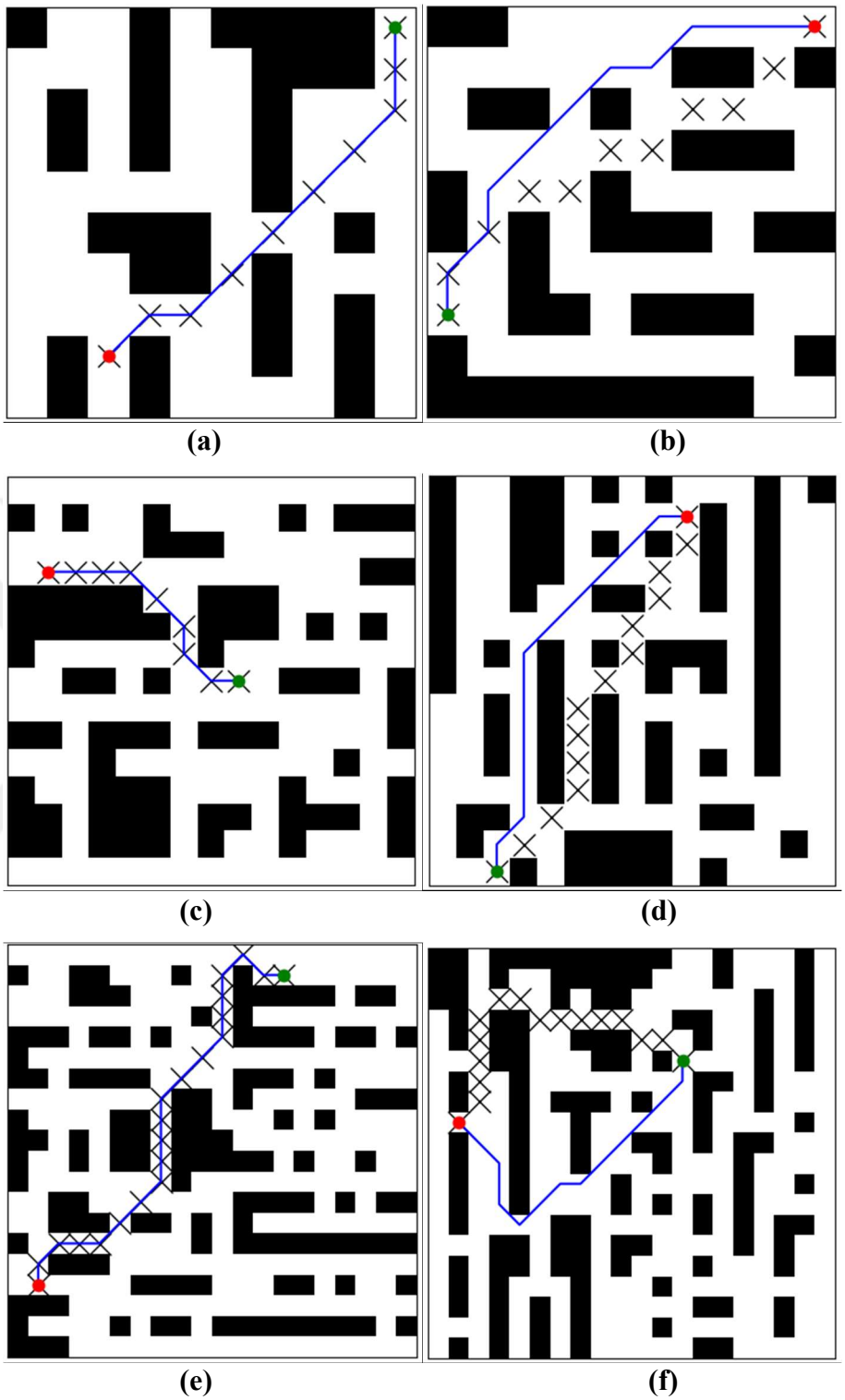
Bu çalışmada, önerilen modelin performansı farklı ortam boyutlarında test edilerek değerlendirilmiştir. İlk olarak, modelin 10 x 10 br^2 boyutundaki bir ızgara ortamında eğitim süreci gerçekleştirilmiştir. Bu eğitimden sonra, bu model hem 10 x 10 br^2 hem de 15 x 15 br^2 ile 20 x 20 br^2 boyutlarındaki ızgara ortamlarında test edilmiştir. Bunun yanı sıra, 15 x 15 br^2 ve 20 x 20 br^2 boyutlarında ayrı ayrı eğitilen modeller de benzer şekilde, her üç ortam boyutunda da test edilerek genel performansları karşılaştırılmıştır. Bu çoklu test ortamları, modelin genelleme yeteneğini ve farklı ortam boyutlarına adaptasyon kabiliyetini analiz etmek amacıyla kullanılmıştır. Tekli robot test verilerinde 10 x 10 br^2, 15 x 15 br^2 ve 20 x 20 br^2 boyutlu ortam verileri ile eğitilen IResNet modellerinin farklı test ortamlarında FCN ile karşılaştırmaları sırasıyla Tablo 4.35'te, bunların çubuk grafikleri ise sırasıyla Şekil 4.44'te gösterilmektedir. Ayrıca 15 x 15 boyutlu ortam verisi ile eğitilen IResNet modelinin her boyuttaki ortamda planladığı optimal ve yarı-optimal yollardan örnekler Şekil 4.45'te gösterilmektedir.

Tablo 4.35. Tekli robot test verilerinde 10 x 10 br^2, 15 x 15 br^2 ve 20 x 20 br^2 boyutlu ortam verisi ile eğitilen IResNet modelinin farklı test ortamlarında FCN ile karşılaştırması

	Model	**Başarı Oranı (%)**			**Yol Optimalitesi (%)**		
		10 x 10 Test	**15 x 15 Test**	**20 x 20 Test**	**10 x 10 Test**	**15 x 15 Test**	**20 x 20 Test**
10 x 10 Eğitim	FCN	100	99.75	98.45	99.85	83.51	58.10
	IResNet	100	99.85	98.10	98.05	89.85	66.25
15 x 15 Eğitim	FCN	99.90	99.95	98.90	97.25	91.00	75.53
	IResNet	100	100	99.95	98.30	96.45	90.70
20 x 20 Eğitim	FCN	99.90	99.70	99.60	98.05	94.38	86.55
	IResNet	99.90	99.75	99.85	98.70	97.00	94.25



Şekil 4.44. Tekli robot test verilerinde 10 x 10 br^2, 15 x 15 br^2 ve 20 x 20 br^2 boyutlu ortam verisi ile eğitilen IResNet modelinin FCN ile karşılaştırma grafiği: **(a)** 10 x 10 br^2 başarı oranı, **(b)** 10 x 10 br^2 yol optimalitesi, **(c)** 15 x 15 br^2 başarı oranı, **(d)** 15 x 15 br^2 yol optimalitesi, **(e)** 20 x 20 br^2 başarı oranı, **(f)** 20 x 20 br^2 yol optimalitesi



Şekil 4.45. Tekli robot test verilerinde 15 x 15 br^2 boyutlu ortam verisi ile eğitilen IResNet modelinin her boyuttaki ortamda planladığı optimal ve yarı-optimal yollardan örnekler: **(a)** 10 x 10 br^2 optimal, **(b)** 10 x 10 br^2 yarı-optimal, **(c)** 15 x 15 br^2 optimal, **(d)** 15 x 15 br^2 yarı-optimal, **(e)** 20 x 20 br^2 optimal, **(f)** 20 x 20 br^2 yarı-optimal (Siyah çarpılar gerçek yolu, mavi çizgiler önerilen IResNet modeli ile tahmin edilen yolları, yeşi ve kırmızı noktalar sırasıyla başlangıç ve hedef noktalarını temsil eder.)

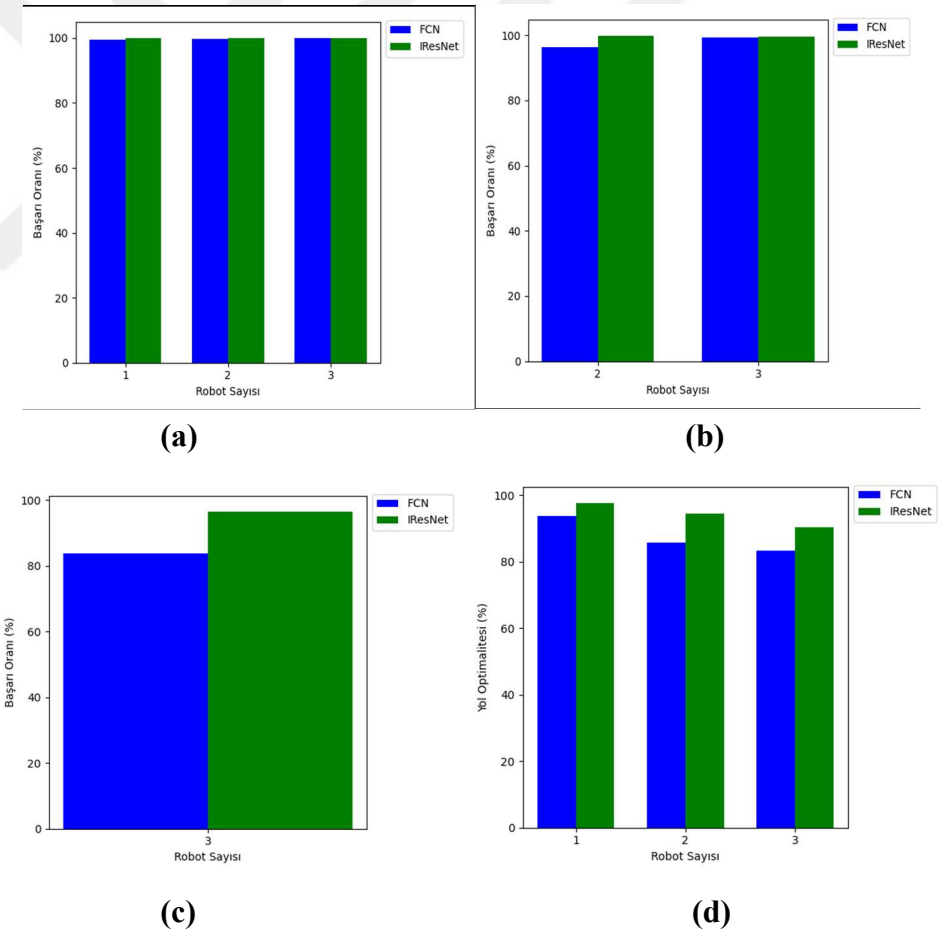
Başarı oranı açısından değerlendirildiğinde, önerilen IResNet modeli tüm ortam boyutlarında FCN ile rekabet edebilir sonuçlar üretmiş ve benzer bir başarı oranı elde etmiştir. Ancak, yol optimalitesi açısından yapılan değerlendirmelerde, IResNet modelinin genel olarak FCN'ye kıyasla daha üstün bir performans sergilediği gözlemlenmiştir. Küçük boyutlu ortamlarda eğitilen modeller arasında, 10 x 10 br^2 boyutlu ortam verisiyle eğitilen IResNet'in, aynı boyuttaki test verisinde FCN'ye kıyasla görece düşük bir optimaliteye sahip olduğu, ancak ortam boyutu büyüdüğünde FCN'yi geride bıraktığı tespit edilmiştir. Ortam boyutu büyüdükçe IResNet'in bu metrikteki başarısı belirgin şekilde artmış ve 15 x 15 br^2 ile 20 x 20 br^2 boyutlu ortam verileriyle eğitilen modeller tüm test ortamlarında FCN'ye kıyasla daha yüksek yol optimalitesi sunmuştur. Özellikle daha büyük test ortamlarında FCN'nin yol uzunluğu açısından ciddi performans kayıpları yaşadığı görülürken, IResNet modelinin optimal yollar üretme konusunda daha kararlı sonuçlar verdiği ortaya konmuştur.

4.5.3.2. Çoklu Robot Senaryosu

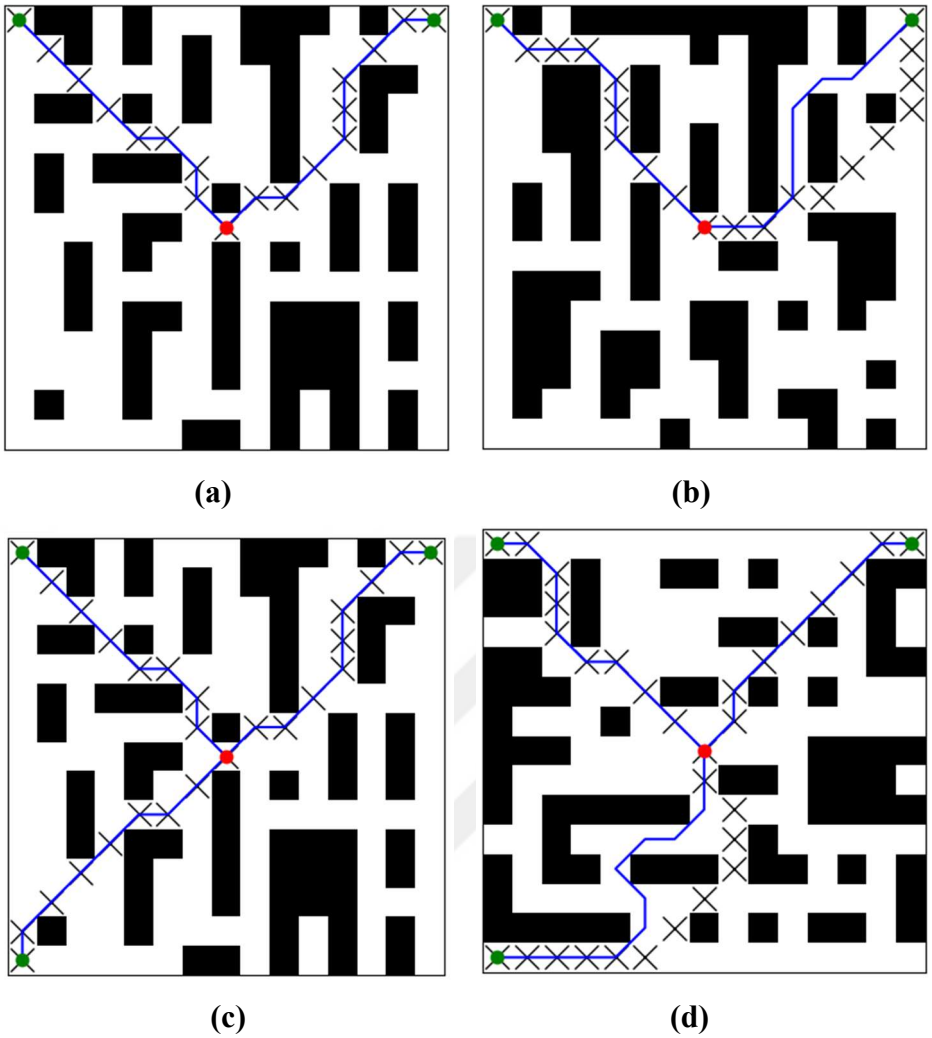
Bu senaryoda ek bir eğitim yapılmadan, tekli robot senaryosunda 15 x 15 br^2 ortam boyutu ile eğitilen model kullanılarak birden fazla robotun yol planlamasındaki performansı incelenmiştir. Buradaki amaç, farklı başlama noktalarından aynı hedef noktasına ulaşan birden fazla robot yolunun tahmin edilmesidir. Çoklu robot senaryosunda sadece 15 x 15 br^2 boyutundaki ortamda yer alan 1000 test verisi kullanılmıştır. Bu kapsamda, başarı oranı ve yol optimalitesi dikkate alınarak 1, 2 ve 3 yol tahmin etme durumu değerlendirilmiştir. Çoklu robot test verilerinde 15 x 15 br^2 ortam boyutu ile eğitilen IResNet modelinin 1, 2 ve 3 yol bulma durumlarında FCN ile karşılaştırması sırasıyla Tablo 4.36'da, bunların çubuk grafikleri ise sırasıyla Şekil 4.46'da gösterilmektedir. Ayrıca bu modelinin 2 ve 3 yol bulma durumlarında planladığı optimal ve yarı-optimal yollardan örnekler Şekil 4.47'de gösterilmektedir.

Tablo 4.36. Çoklu robot test verilerinde 15 x 15 br^2 ortam boyutu ile eğitilen IResNet modelinin 1, 2 ve 3 yol bulma durumlarında FCN ile karşılaştırması

Değerlendirme Metriği		**Model**	**Robot Sayısı**		
			1	**2**	**3**
Başarı Oranı (%)	1 Yol Bulma Durumu	FCN	99.50	99.80	100
		IResNet	100	100	100
	2 Yol Bulma Durumu	FCN	-	96.40	99.20
		IResNet	-	99.80	99.50
	3 Yol Bulma Durumu	FCN	-	-	83.90
		IResNet	-	-	96.50
Yol Optimalitesi (%)		FCN	93.77	85.88	83.33
		IResNet	97.70	94.46	90.42



Şekil 4.46. Çoklu robot test verilerinde 15 x 15 br^2 ortam boyutu ile eğitilen IResNet modelinin 1, 2 ve 3 yol bulma durumlarında FCN ile karşılaştırma grafikleri: **(a)** 1 yol bulma durumunda başarı oranı, **(b)** 2 yol bulma durumunda başarı oranı, **(c)** 3 yol bulma durumunda başarı oranı, **(d)** Yol optimalitesi



Şekil 4.47. Çoklu robot test verilerinde 15 x 15 br^2 boyutlu ortam verisi ile eğitilen IResNet modelinin 2 ve 3 yol bulma durumlarında planladığı optimal ve yarı-optimal yollardan örnekler: **(a)** 2 yol bulma durumunda optimal, **(b)** 2 yol bulma durumunda yarı-optimal, **(c)** 3 yol bulma durumunda optimal, **(d)** 3 yol bulma durumunda yarı-optimal (Siyah çarpılar gerçek yolu, mavi çizgiler önerilen IResNet modeli ile tahmin edilen yolları, yeşi ve kırmızı noktalar sırasıyla başlangıç ve hedef noktalarını temsil eder.)

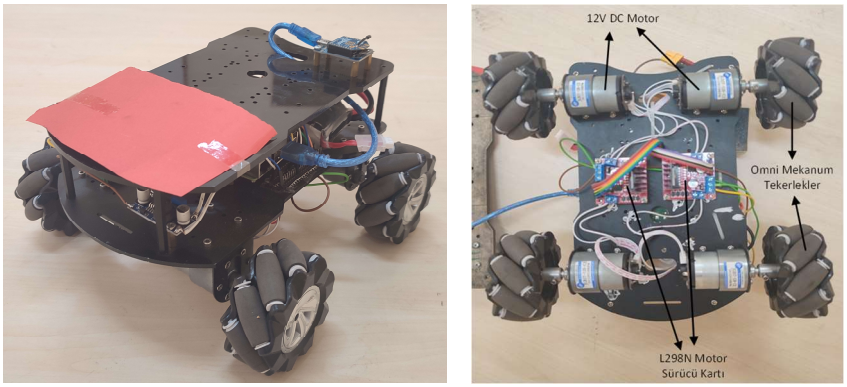
Başarı oranı açısından değerlendirildiğinde, önerilen IResNet modelinin tüm test senaryolarında FCN modelinden daha üstün performans sergilediği açıkça görülmektedir. Özellikle araştırılan yol sayısı arttıkça, bu performans farkı daha belirgin bir şekilde ortaya çıkmaktadır. İlk başta, her iki modelin başarı oranları arasında küçük farklar bulunsa da, yol sayısı arttıkça IResNet'in yüksek başarı oranı daha net bir şekilde fark edilmektedir. Bu durum, modelin daha karmaşık ve geniş ortamlar ile daha fazla yol bulma gereksinimlerini daha etkin şekilde karşılayabildiğini göstermektedir. Ayrıca, önerilen IResNet modeli, FCN'ye kıyasla daha yüksek bir optimalite oranına sahiptir. Bu

da, IResNet'in sadece daha fazla sayıda başarılı çıkış sağlamakla kalmadığını, aynı zamanda daha verimli yolları seçerek daha optimal sonuçlar sunduğunu ortaya koymaktadır. IResNet'in yüksek başarı oranı ile birlikte, düşük maliyetli ve daha kısa yollar üretme yeteneği, onun yol planlama alanında güçlü bir alternatif olarak öne çıkmasını sağlamaktadır. Sonuç olarak IResNet, büyük ölçekli ve karmaşık ortamlar için daha başarılı bir yol planlama yaklaşımı sunmaktadır. Hem başarı oranı hem de yol optimalitesi açısından FCN modelini geride bırakarak, özellikle robotik ve otonom sistemlerde yüksek doğruluk ve verimlilik gerektiren uygulamalarda tercih edilmesi gereken bir model olarak dikkat çekmektedir. Bu durum, IResNet'in çeşitli robotik senaryolarda daha verimli ve güvenilir çözümler sunduğunu ve dolayısıyla daha geniş çaplı kullanım alanları için uygun olduğunu göstermektedir.

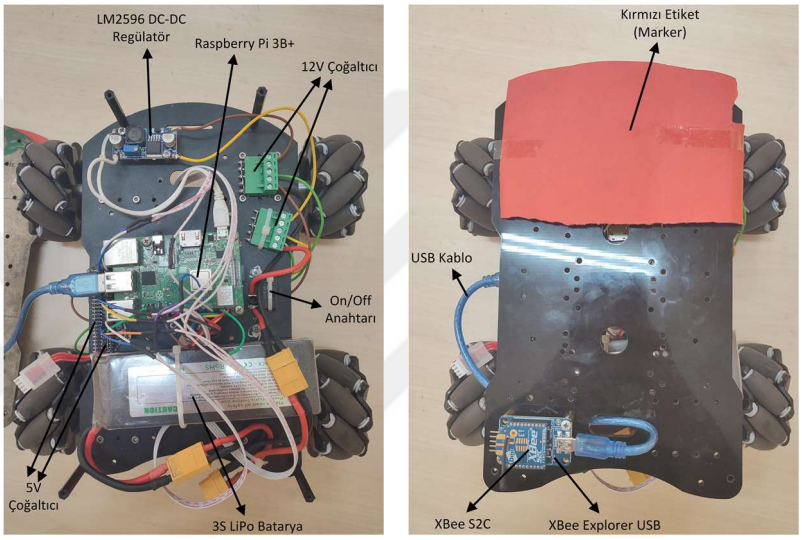
4.6. Bir Mobil Robotun Gerçek Zamanlı Yol Planlaması ve Takip Kontrolü

4.6.1. Mobil Robot Tasarımı

Bu çalışmada dört tekerlekli ve 4WD (four-wheel drive) sürüş sistemine sahip siyah bir mobil robot kullanılmıştır. Robotun şasesi iki katmandan oluşmaktadır: Alt katman alüminyum, üst katman ise akrilik plastik malzemeden imal edilmiştir. Tekerlekler omni mekanum tipinde seçilmiştir, bu sayede robot sağa-sola veya çapraz gibi çok yönlü hareket edebilmektedir. Robotun ana kontrol birimi olarak Raspberry Pi 3B+ kullanılmıştır. Tüm sistemin enerji ihtiyacı 3S LiPo batarya ile karşılanmıştır. Tekerlek motorları 12V DC ile çalıştığı için bu gerilim doğrudan motor sürücülerine yönlendirilmiştir. Motor sürücüsü olarak iki adet L298N kartı kullanılmıştır. Her sürücü kartı iki motor kontrol edebildiği için toplamda dört motor için iki sürücü yeterli olmuştur. Raspberry Pi kartı ise 5V DC ile çalıştığı için bataryadan gelen gerilim LM2596 ayarlanabilir DC-DC regülatör ile düşürülmüştür. Ayrıca, robot ile ana bilgisayar arasında kablosuz iletişimi sağlamak amacıyla XBee S2C haberleşme modülü kullanılmıştır. XBee'nin Raspberry Pi kartına kolayca bağlanabilmesi için XBee Explorer USB tercih edilmiştir. Mobil robotun genel görünümü ve donanımı Şekil 4.48'de, elektronik bağlantı şeması ise Şekil 4.49'da gösterilmektedir.

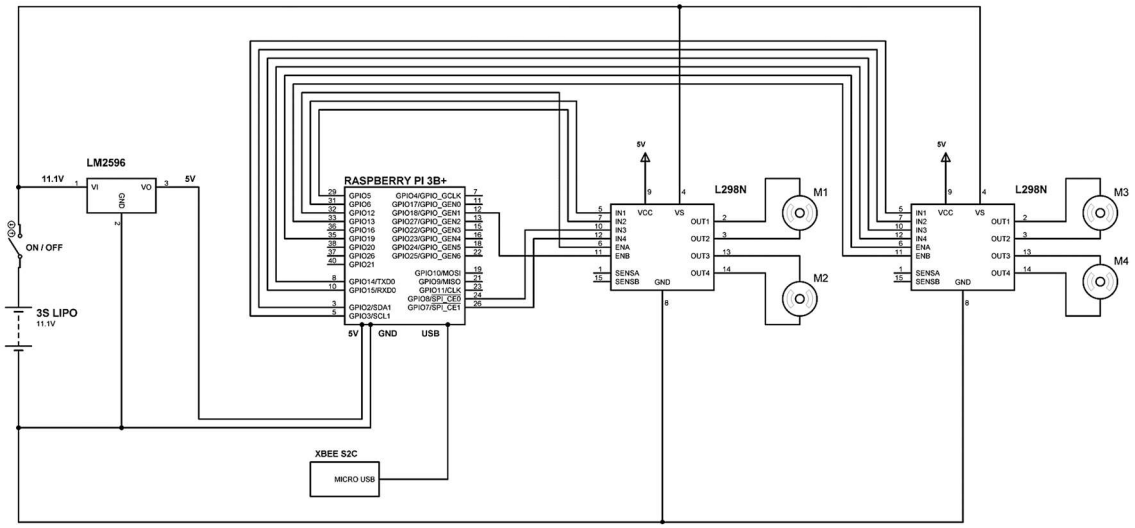


(a) **(b)**



(c) **(d)**

Şekil 4.48. Mobil robotun genel görünümü ve donanımı: **(a)** Genel görünüm, **(b)** Ara katmanın alt yüzeyi, **(c)** Ara katmanın üst yüzeyi, **(d)** Üst katman



Şekil 4.49. Mobil robotun elektronik bağlantı şeması

4.6.2. Mobil Robotun Ters Kinematiği

Bu çalışmada kullanılan robot çok yönlü tekerleklere sahip olduğundan her yöne hareket edebilir, yani robot holonomik olarak tanımlanabilir. Bu robotun hareketi merkezine yerleştirilen yerel bir koordinat sisteminde tanımlanır. Bu sistemde robotun X yönündeki doğrusal hızı v_x, Y yönündeki doğrusal hızı v_y ve Z ekseni etrafındaki açısal hızı ω_z olmak üzere üç hız bileşeni bulunur. Bu hızlar vektörel olarak Eşitlik (4.39)'daki gibi gösterilebilir.

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{4.39}$$

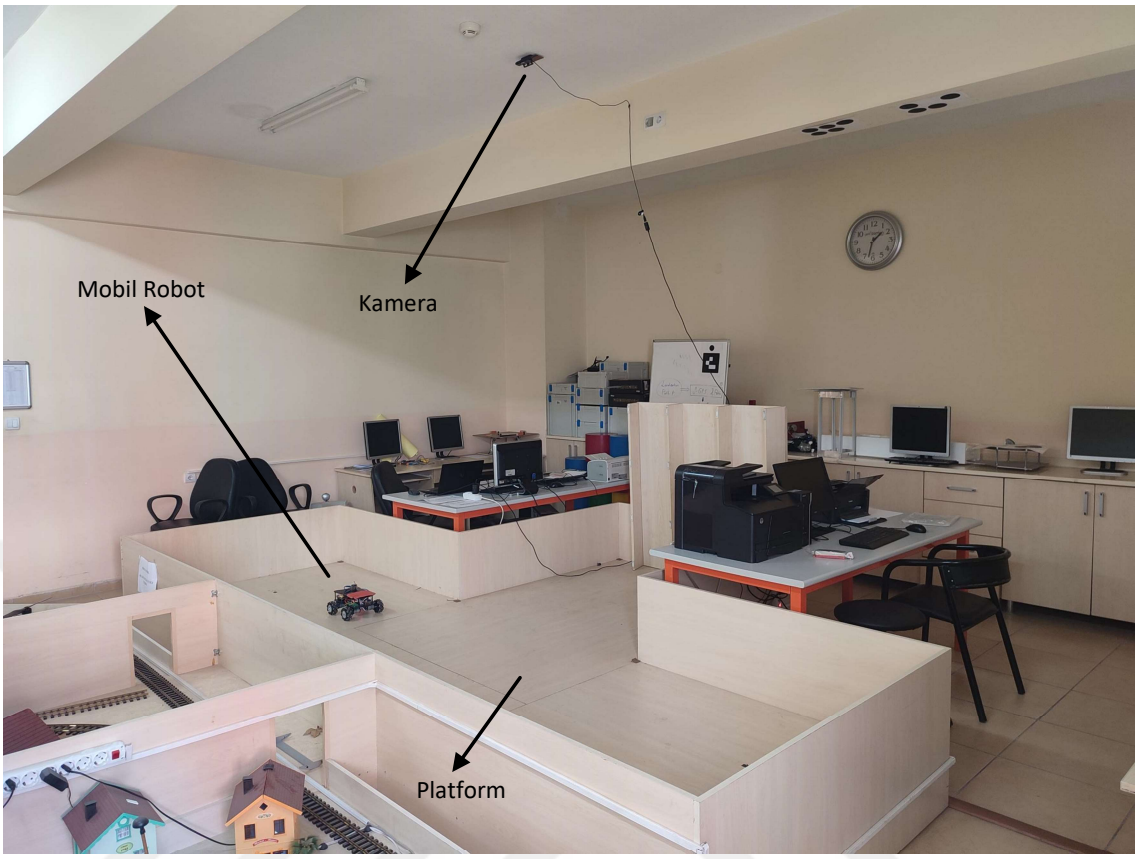
Burada, $\vec{v}$ robotun genel hız vektörünü temsil eder. Ters kinematik ise robotun bu üç hızından hareketle her bir tekerleğin açısal hızının (ω_1, ω_2, ω_3, ω_4) elde edilmesidir. 4WD sürüşlü ve omni mekanum tekerlekli bir mobil robot için bu tekerlek hızları Eşitlik (4.40) kullanılarak hesaplanır.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r_w} \begin{bmatrix} 1 & -1 & -(l_h + l_v) \\ 1 & 1 & (l_h + l_v) \\ 1 & -1 & (l_h + l_v) \\ 1 & 1 & -(l_h + l_v) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{4.40}$$

Burada, r_w tekerleklerin yarıçapını, l_h ve l_v robot gövde merkezinin tekerlek merkezine olan yatay ve dikey uzaklıklarını temsil eder. Hesaplanan tekerlek hızları robotun belirli bir referans yol üzerinde istenilen şekilde ilerlemesini sağlamak amacıyla geliştirilen kontrol algoritmalarında kullanılır.

4.6.3. Platform ve Engeller

Bu çalışmada, mobil robotun yol planlaması için bölümümüzün El-Cezeri Mobil Robotik Sistemler Laboratuvarındaki çalışma platformu kullanılmıştır. Platformun genel yapısı "T" harfi formundadır, ancak gerçek zamanlı çalışma sadece bu yapının üst kısmında yer alan dikdörtgen alan tercih edilmiştir. Bu bölüm robotun hem serbestçe hareket edebilmesini hem de sınırlar içinde kalmasını sağlayacak şekilde seçilmiştir. Bu çalışmada kullanılan platform Şekil 4.50'de gösterilmektedir.



Şekil 4.50. Platform

Laboratuvarın tavanına Genius Widecam F100 model bir webcam sabitlenmiş ve bu kamera ile platform tamamı yukarıdan izlenebilmiştir. Kullanılan kamera Şekil 4.51'de, platform üstten kamera görüntüsü ise Şekil 4.52'de gösterilmektedir.

Şekil 4.51. Genius Widecam F100 Webcam

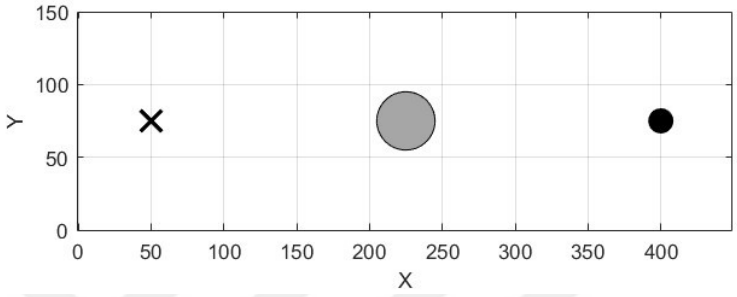
Şekil 4.52. Platform üstten kamera görüntüsü

Platform üzerine yerleştirilen engeller mavi renkli silindirik cisimler olarak seçilmiştir. Renk ve şekil tercihleri robotun üzerindeki kamera ile çalışan görsel algılama sisteminin bu engelleri daha kolay tespit edebilmesi göz önünde bulundurularak yapılmıştır. Mavi silindirik engeller Şekil 4.53'te gösterilmektedir.

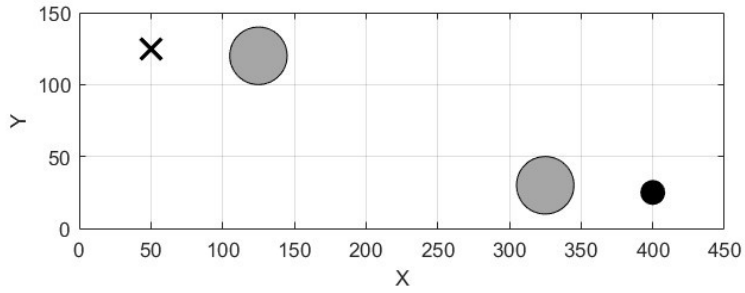
Şekil 4.53. Engeller

4.6.4. Görüntü İşleme ve Gerçek Zamanlı Yol Planlama

Gerçek zamanlı çalışmadan önce, MATLAB programlama dilinde çalışmada kullanılan iki farklı ortam tasarlanmış ve bu ortamlar için yol planlama simülasyonu gerçekleştirilmiştir. Şekil 4.54'te gösterilen bu ortamlar gerçek zamanlı çalışmada da kullanılmıştır.



(a)



(b)

Şekil 4.54. Tasarlanan ortamlar: **(a)** Ortam 20, **(b)** Ortam 21 (Siyah daireler robotu, çarpı işaretleri hedef noktasını, gri daireler ise engelleri temsil eder.)

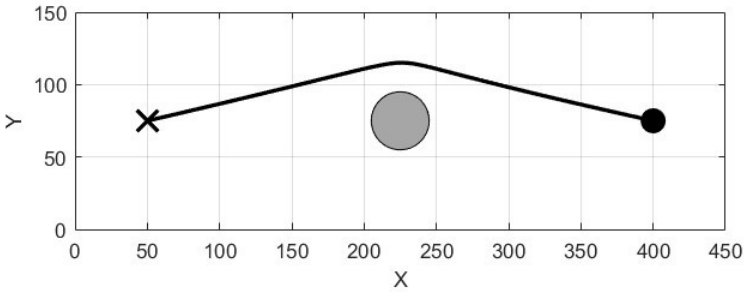
Bu simülasyonda robot ve hedef noktası arasında kübik spline interpolasyonu tabanlı optimum bir yol (P_r) planlanmıştır. Bu optimziasyon ABC algoritması kullanılarak gerçekleştirilmiş ve robotun gideceği en kısa ve en güvenli yol planlanmıştır. ABC algoritmasında kullanılan amaç fonksiyonu Eşitlik (4.41)'de gösterilmektedir.

$$\arg\min_{P_r} \mathcal{F} = \mathcal{F}_1 + \beta\mathcal{F}_2 \tag{4.41}$$

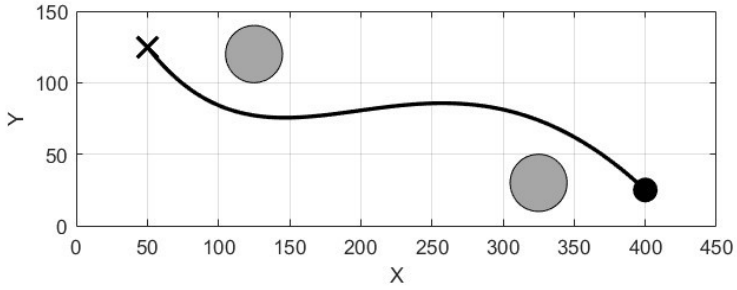
$$\mathcal{F}_1 = \sum_{i=1}^{n_y} \|p_{i+1} - p_i\| \tag{4.42}$$

$$\mathcal{F}_2 = \sum_{i=1}^{n_y} \begin{cases} \left(1 - \dfrac{\|p_i - p^o\|}{r^o + L_g}\right), & eğer \left(1 - \dfrac{\|p_i - p^o\|}{r^o + L_g}\right) > 0 \; ise \\ 0, & aksi \; halde \end{cases} \tag{4.43}$$

Burada β engel ihlal faktörünü, n_y yolu oluşturan nokta sayısını, p_i yolun i'inci noktasını, p^o engelin merkez noktasını, r^o engelin yarıçapını ve L_g ise güvenlik mesafesini temsil eder. ABC algoritması için maksimum iterasyon sayısı 200, popülasyon sayısı ve limit değeri 100 olarak ayarlanmıştır. Simülasyondaki yol planlama sonucu Şekil 4.55'te gösterilmektedir.



(a)



(b)

Şekil 4.55. Simülasyon çalışmasında yol planlama sonucu: **(a)** Ortam 20, **(b)** Ortam 21 (Siyah çizgi planlanan yolu temsil eder.)

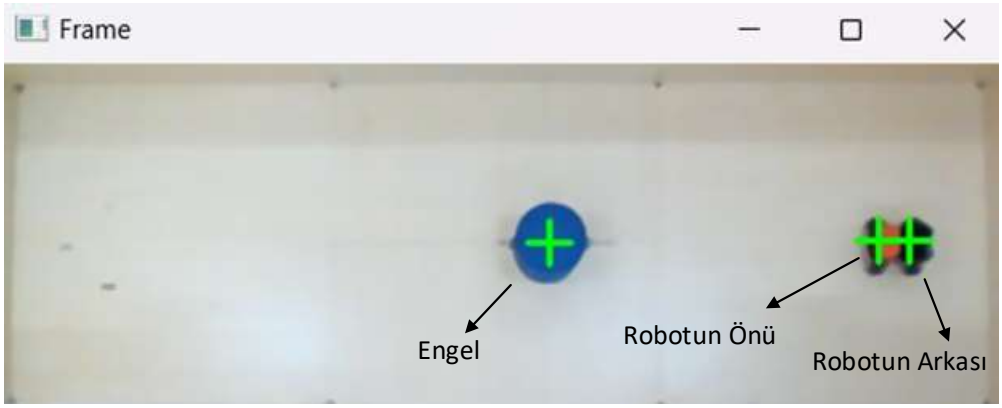
Gerçek zamanlı çalışma Python programlama dilinde kodlanmış ve görüntü işleme için OpenCV kütüphanesinden yararlanılmıştır. İlk olarak, "T" harfi biçimindeki platformun yalnızca üstteki dikdörtgen bölümü sınırlandırılmıştır. Global referans ekseni ise bu dikdörtgen ekranın sol üst köşesidir. İlk aşamada ana bilgisayarda robotun ve engellerin konumları belirlenmiştir. Bu amaçla renk tabanlı bir görüntü işleme algoritması kullanılmıştır. Algoritmada belirli renkleri tespit edebilmek için HSV renk uzayında eşikleme yöntemi uygulanmış ve bu işlem sonucunda ikili bir maske görüntüsü elde edilmiştir. Maske görüntüsünde yalnızca eşik değerini sağlayan pikseller 1 (beyaz), diğer tüm pikseller ise 0 (siyah) olarak atanmıştır. Elde edilen bu maske üzerinde kontur analizi

gerçekleştirilmiş ve böylece ilgili renk bölgelerinin merkez koordinatları hesaplanmıştır. Bu sayede robot ve engellerin konumları kamera görüntüsünden başarıyla tespit edilmiştir. Ancak robotun yönelim bilgisi de gerektiğinden, robotun tespiti sırasında yalnızca kendi rengi yeterli olmamış ve robot üzerinde farklı bir renk daha kullanılmıştır. Bu yüzden robotun ön kısmına kırmızı bir etiket (marker) yerleştirilmiş ve arka kısmı ise robotun kendi rengi olan siyah ile ayırt edilmiştir. Aynı görüntü işleme algoritması ile robotun üzerindeki kırmızı ve siyah renk bölgeleri ayrı ayrı algılanmış ve böylece robotun ön ve arka noktası belirlenmiştir. Ardından bu iki nokta kullanılarak robotun global eksen takımına göre konum ve yönelimi Eşitlik (4.44) ve (4.45)'teki gibi hesaplanmıştır.

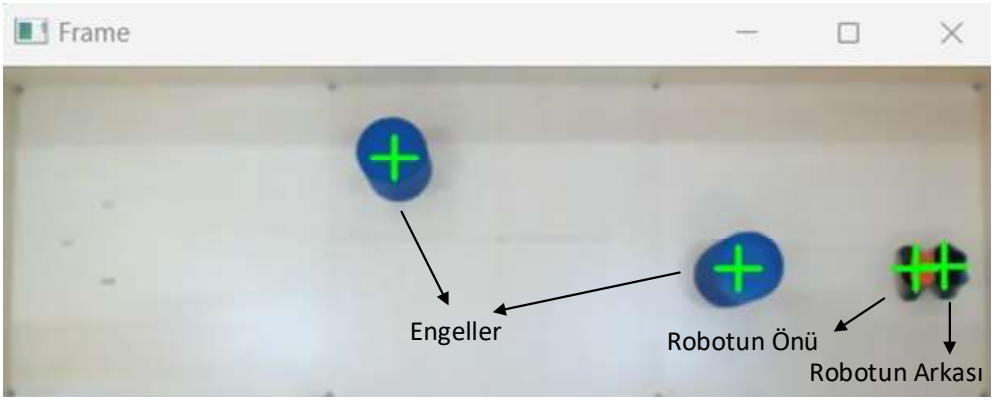
$$X_r = \frac{X_{\text{ön}} + X_{arka}}{2}, \qquad Y_r = \frac{Y_{\text{ön}} + Y_{arka}}{2} \tag{4.44}$$

$$\theta_r = \tan^{-1}\left(\frac{Y_{\text{ön}} - Y_{arka}}{X_{\text{ön}} - X_{arka}}\right) \tag{4.45}$$

Burada (X_r, Y_r) robotun merkez konumunu, $(X_{\text{ön}}, Y_{\text{ön}})$ robotun ön noktasının konumunu, (X_{arka}, Y_{arka}) robotun arka noktasının konumunu, θ_r ise robotun yönelimini temsil eder. Gerçek zamanlı çalışmada robotun ve engellerin algılanması Şekil 4.56'da, ana bilgisayardaki görüntü işleme algoritmasının temel adımları ise Algoritma 4.8'de gösterilmektedir.



(a)



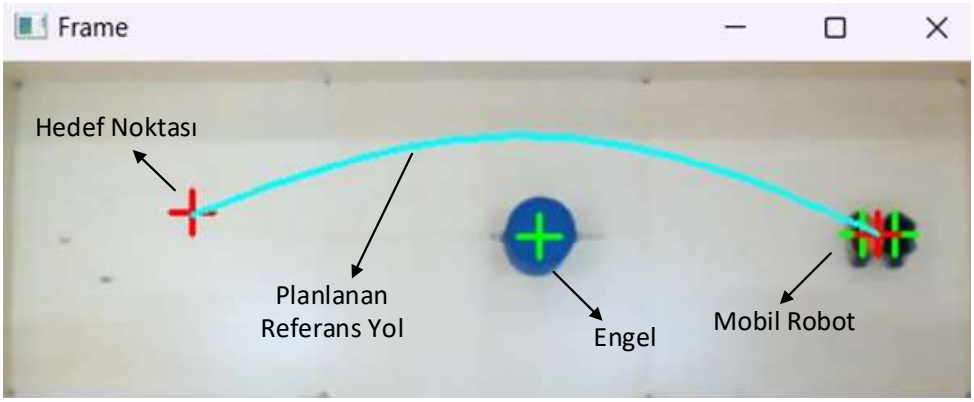
(b)

Şekil 4.56. Gerçek zamanlı çalışmada robotun ve engellerin algılanması: **(a)** Ortam 20, **(b)** Ortam 21

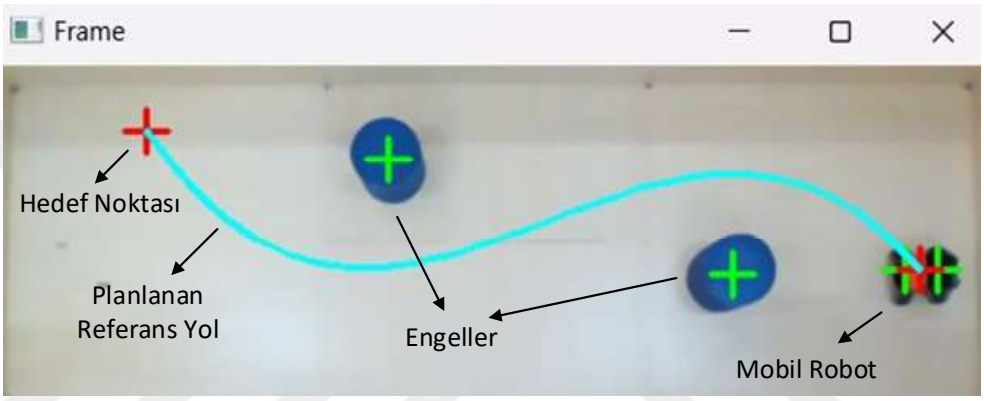
Algoritma 4.8: Ana bilgisayardaki görüntü işleme algoritmasının temel adımları

1: Kameranın başlatılması
2: Kameradan görüntü alınması
3: Gürültülerin medyan filtre ile azaltılması
4: Görüntünün HSV uzayına dönüştürülmesi
5: Mavi (engel), kırmızı (robot ön) ve siyah (robot arka) renkli bölgelerin tespit edilmesi
6: Kontur analizi ile bu renkli bölgelerin merkez noktalarının hesaplanması
7: Eşitlik (4.39) ve (4.40) kullanılarak robotun konum ve yöneliminin hesaplanması
8: İşlenmiş görüntünün ekranda gösterilmesi

Robotun ve engellerin tespit edilmesinden sonra robotun konumu ve daha önceden girdi olarak alınan hedef nokta arasında tekrar ABC algoritması kullanılarak yol planlanmıştır. Gerçek zamanlı çalışmada ABC algoritmasında yine Eşitlik (4.41)'deki amaç fonksiyonu kullanılmış, maksimum iterasyon sayısı, popülasyon sayısı ve limit değeri de simülasyon çalışmasında kullanılan ayarlarla aynıdır. Yol planlama sonucunda elde edilen bu koordinat dizisi (planlanan yol) takip kontrolünde referans yol olarak kullanılmıştır. Gerçek zamanlı çalışmada yol planlama sonucu Şekil 4.57'de, ana bilgisayardaki gerçek zamanlı yol planlama algoritmasının temel adımları ise Algoritma 4.9'da gösterilmektedir.



(a)



(b)

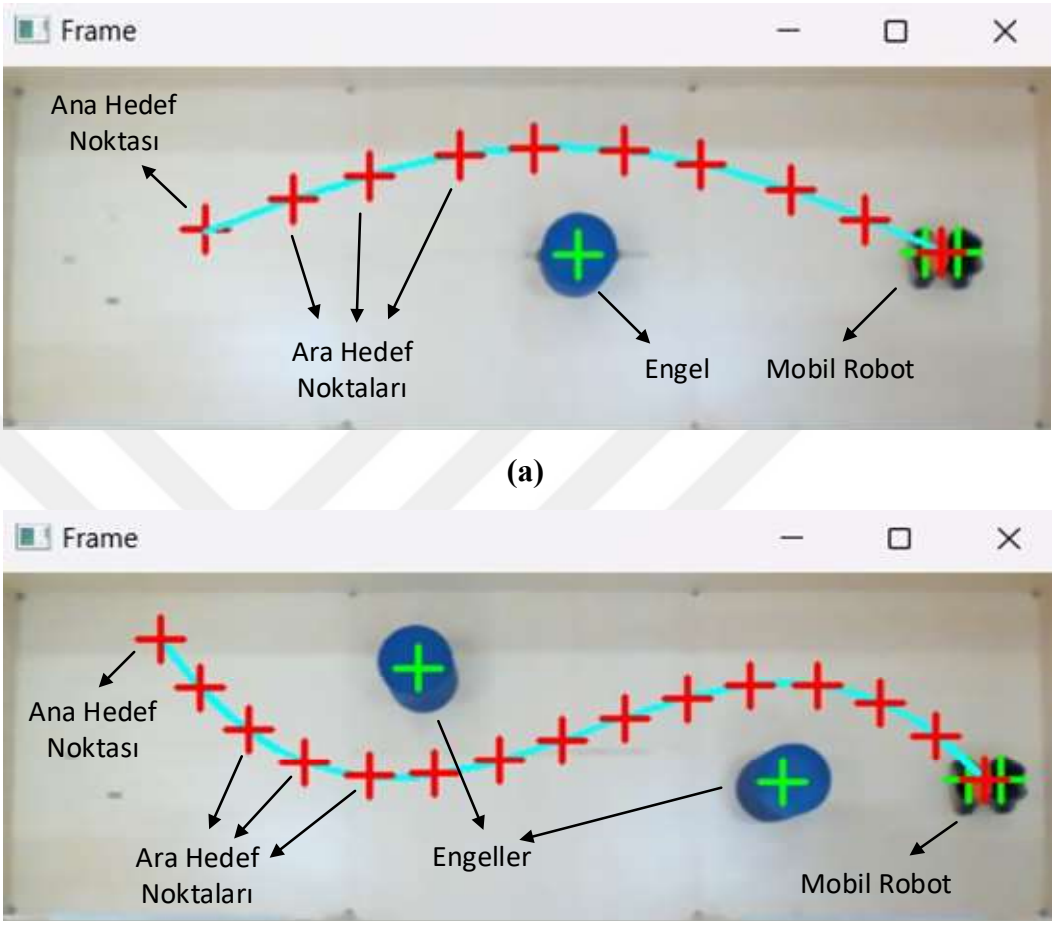
Şekil 4.57. Gerçek zamanlı çalışmada yol planlama sonucu: **(a)** Ortam 20, **(b)** Ortam 21

Algoritma 4.9: Ana bilgisayardaki gerçek zamanlı yol planlama algoritmasının temel adımları	
1:	Hedef noktanın kullanıcı tarafından girilmesi
2:	Robotun başlangıç noktasının o anki durduğu yer olarak atanması
3:	Görüntü işleme algoritmasından engel bilgilerinin (merkezi ve yarıçapı) alınması
4:	Eşitlik (4.41) kullanılarak amaç fonksiyonu tanımı
5:	ABC algoritması kullanılarak optimum yolun hesaplanması
6:	Planlanan referans yolun işlenmiş görüntüde gösterilmesi
7:	Planlanan referans yolun koordinat dizisi olarak kaydedilmesi

4.6.5. Gerçek Zamanlı Yol Takip Kontrolü

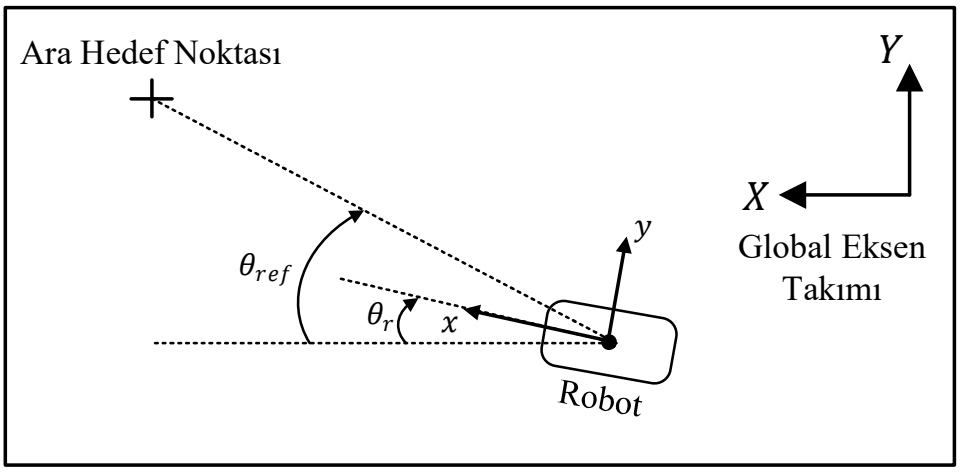
Robotun planlanan referans yol üzerinde en az hata ile hedef noktaya ulaşması için bir yol takip kontrol algoritması geliştirilmiştir. Bu süreç başlamadan önce robot ile ana bilgisayar arasında kablosuz haberleşmeyi sağlamak için XBee modülü başlatılmıştır. İlk olarak referans yol belirli sayıda ara hedef noktaya (yerel hedefler) bölünmüştür. Bu

noktalar robotun ana hedefe kademeli olarak yönelmesini sağlar. Gerçek zamanlı çalışmada referans yol için oluşturulan ara hedef noktaları Şekil 4.58'de gösterilmektedir.



Şekil 4.58. Gerçek zamanlı çalışmada referans yol üzerinde oluşturulan ara hedef noktaları: **(a)** Ortam 20, **(b)** Ortam 21

Ara hedef noktaları oluşturulduktan sonra görüntü işleme algoritması kullanılarak robotun anlık olarak global eksen takımına göre konum ve yönelim bilgisi elde edilmiştir. Ayrıca robotun konumuna en yakın ara hedef noktasına olan yönelim de elde edilir. Robotun global eksen takımına göre yönelimi ve ara hedef noktaya olan yönelimi Şekil 4.59'da gösterilmektedir.



Şekil 4.59. Robotun global eksen takımına göre yönelimi ve ara hedef noktaya olan yönelimi

Bu iki yönelim açısı arasındaki hata Eşitlik (4.46)'da tanımlandığı gibi hesaplanmıştır.

$$\theta_e = \theta_{ref} - \theta_r \tag{4.46}$$

Burada θ_e açısal hatayı ve θ_{ref} ise robotun hedef noktaya olan yönelimini temsil eder. Açısal hata XBee modülü aracılığıyla robot üzerinde çalışan Raspberry Pi kartındaki hareket algoritmasına girdi olarak verilmiştir. Robot üzerinde çalışan hareket algoritması açısal hatayı P kontrol organına verir. P kontrol organının girişi açısal hata iken çıkışı robot merkezinin Z ekseni etrafındaki açısal hızı (ω_z) olarak tanımlanmıştır. Bu kontrol organında açısal hata Eşitlik (4.47)'de gösterildiği gibi sabit bir katsayı (K_p) ile çarpılır.

$$\omega_z = K_p \theta_e \tag{4.47}$$

Bu sayede robot merkezinin açısal hızı hesaplanır ve bu hız ters kinematik model kullanarak tekerlek hızlarına çevrilir. Böylece hata ne kadar küçükse o kadar yumuşak, hata ne kadar büyükse o kadar keskin dönüş sağlanır. Ancak açısal hatanın çok küçük olduğu durumlarda hareket algoritması robotun ileri doğrultuda (X doğrultusunda) sabit bir lineer hızda (v) düz gitmesini sağlar ve yanal hareket yoktur. Bu sebeple robotun genel hız vektörü Eşitlik (4.48)'deki gibi tanımlanmıştır.

$$\vec{v} = \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} v \\ 0 \\ K_p \theta_e \end{bmatrix} \tag{4.48}$$

Bu durumda robotun her bir tekerleğin açısal hızı ters kinematik model kullanılarak Eşitlik (4.49)-(4.52)'deki gibi hesaplanmıştır.

$$\omega_1 = \frac{1}{r_w}\left(v - (l_h + l_v)K_p\theta_e\right) \tag{4.49}$$

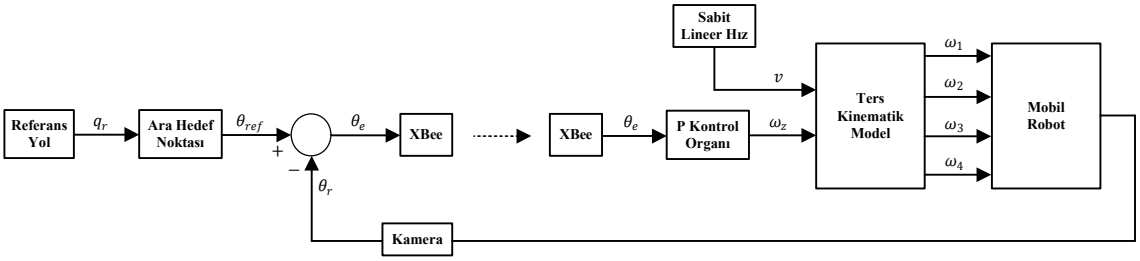
$$\omega_2 = \frac{1}{r_w}\left(v + (l_h + l_v)K_p\theta_e\right) \tag{4.50}$$

$$\omega_3 = \frac{1}{r_w}\left(v + (l_h + l_v)K_p\theta_e\right) \tag{4.51}$$

$$\omega_4 = \frac{1}{r_w}\left(v - (l_h + l_v)K_p\theta_e\right) \tag{4.52}$$

$$0 \leq \left|(l_h + l_v)K_p\theta_e\right| \leq v \tag{4.53}$$

Hesaplanan bu tekerlek hızları ile robot sürülmüş ve en yakın ara hedef noktasına ulaşana kadar bu kontrol döngüsü devam etmiştir. Robot en yakın ara hedef noktasına ulaştığında bir sonraki hedef noktası robotun yeni yerel hedefi olmuş ve aynı kontrol döngüsü tekrarlanmıştır. Bu süreç robot ana hedef noktaya ulaşana kadar devam etmiştir. Bu basit kontrol mantığı sayesinde robot ara hedef noktalarını sırayla takip ederek referans yol üzerinde güvenli ve kararlı bir şekilde ana hedefe yönlendirilebilmiştir. Yol takip kontrolünün blok diyagramı Şekil 4.60'ta, ana bilgisayardaki yol takip kontrol algoritmasının temel adımları Algoritma 4.10'da ve robot üzerinde çalışan hareket algoritmasının temel adımları ise Algoritma 4.11'de gösterilmektedir.



Şekil 4.60. Yol takip kontrolünün blok diyagramı

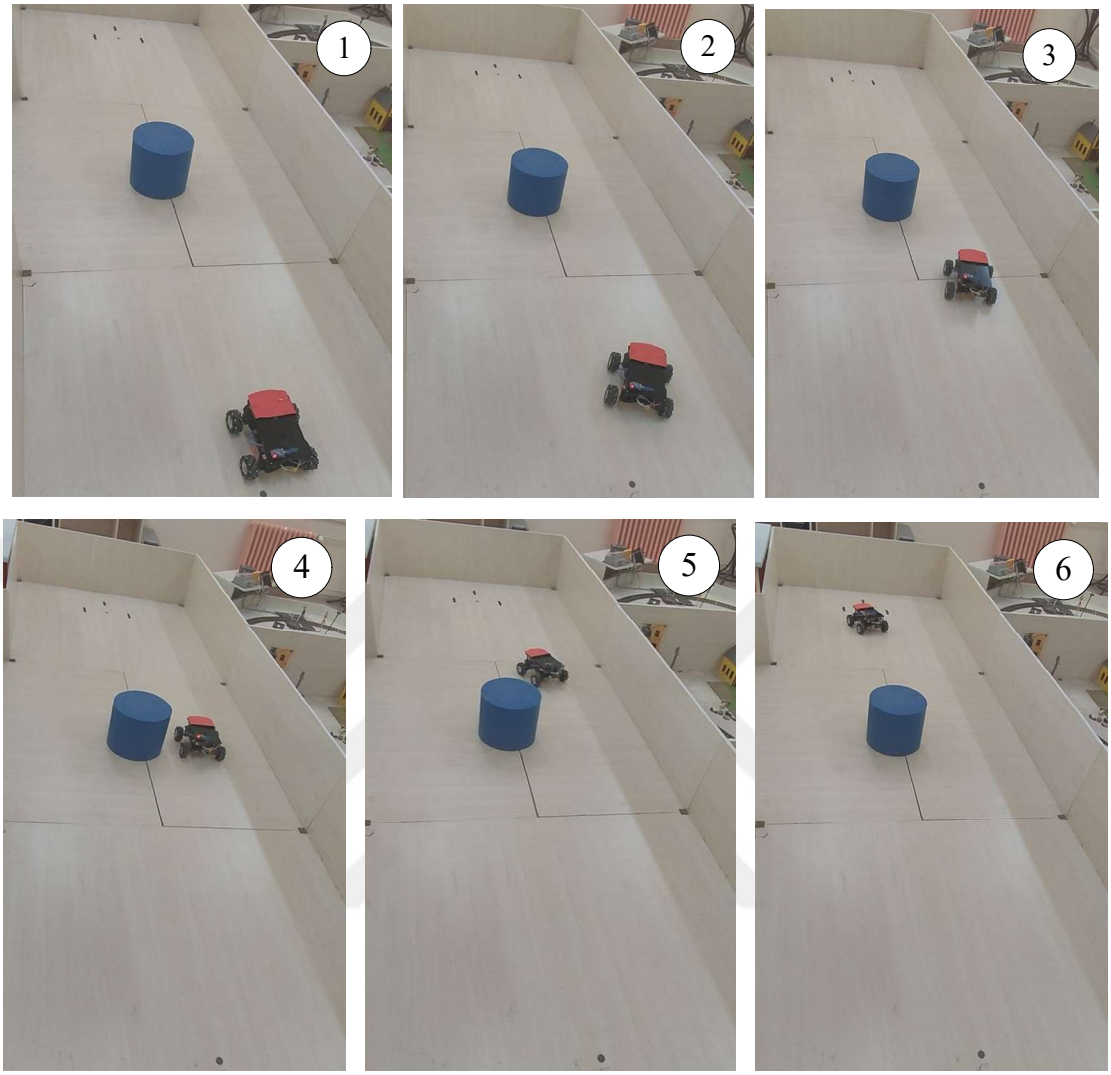
Algoritma 4.10: Ana bilgisayardaki yol takip kontrol algoritmasının temel adımları

```
XBee modülünün başlatılması
Ara hedef noktalarının belirlenmesi
while (Ana hedef noktasına ulaşılana kadar)
    for (Her ara hedef noktasında)
        while (Ara hedef noktasına ulaşılana kadar)
            Robotun konum ve yöneliminin tespit edilmesi
            Robotun ara hedef noktasına olan yöneliminin tespit edilmesi
            Eşitlik (4.46) kullanılarak açısal hatanın hesaplanması
            XBee modülü ile açısal hatanın robota gönderilmesi
        end while
    end for
end while
XBee modülünün kapatılması
```

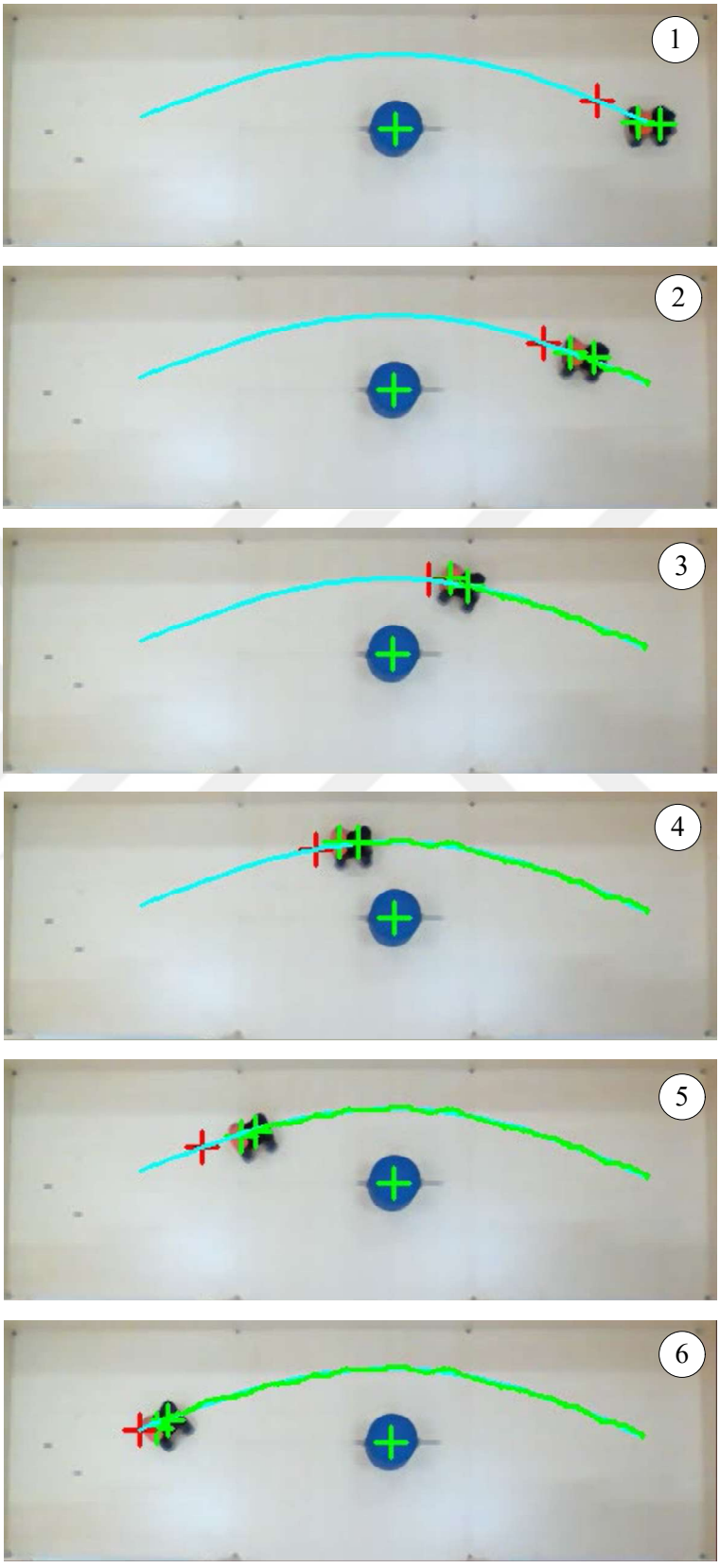
Algoritma 4.11: Robot üzerinde çalışan hareket algoritmasının temel adımları

```
GPIO pin ayarlarının yapılması
XBee modülünün başlatılması
while (XBee modülünden "Dur" emri gelene kadar)
    XBee modülünden açısal hatanın alınması
    Eşitlik (4.47) kullanılarak robot merkezinin açısal hızının hesaplanması
    Eşitlik (4.49)-(4.52) kullanılarak tekerlek hızlarının hesaplanması
    Robotun bu tekerlek hızlarında sürülmesi
end while
XBee modülünün kapatılması
```

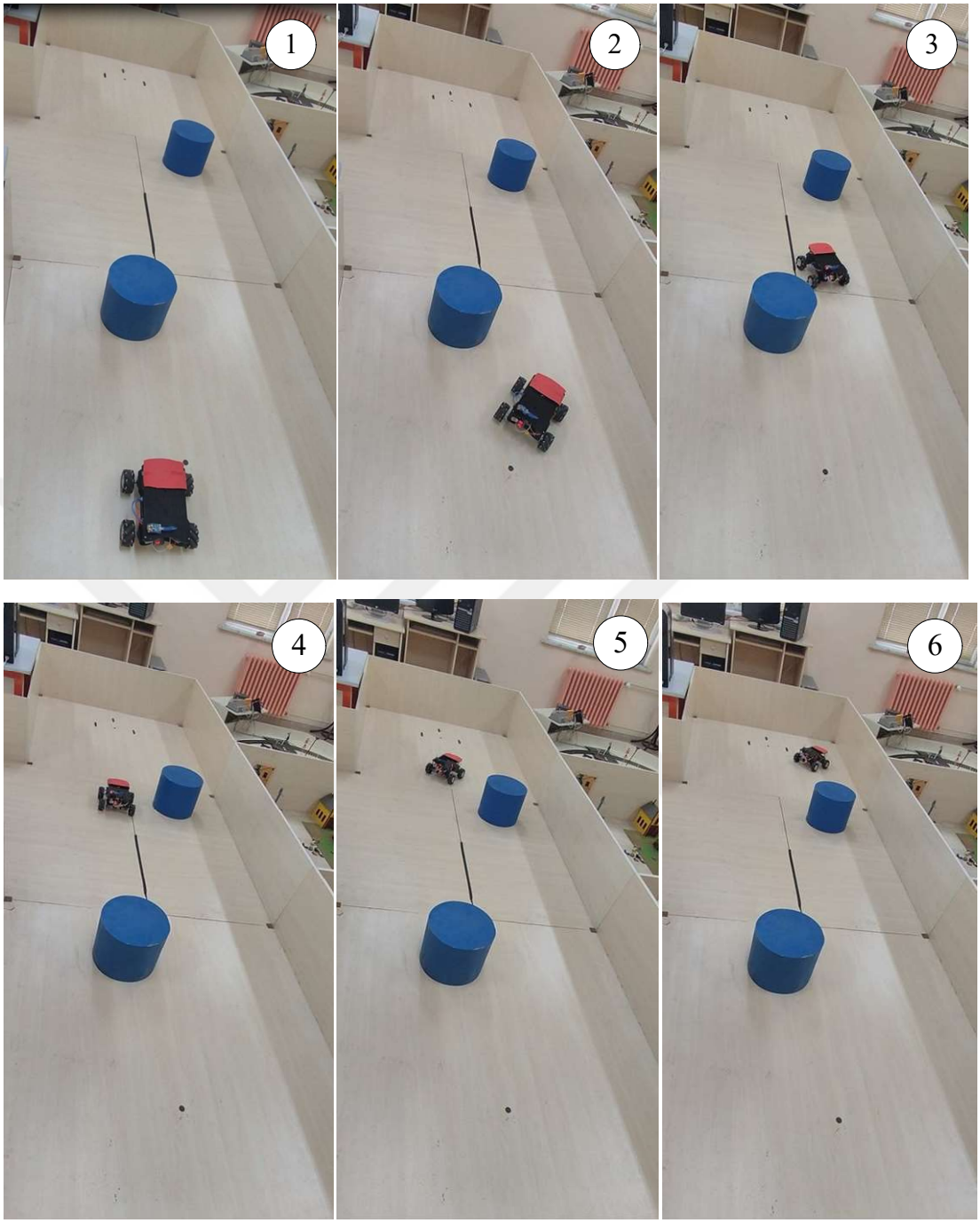
Tasarlanan ortamlar için gerçek zamanlı bir yol takip çalışması gerçekleştirilmiştir. Her iki ortam için robotun hareket aşamaları Şekil 4.61 ve 4.62'de, bu hareket aşamalarının kamera görüntüleri Şekil 4.63 ve 6.64'te, planlanan ile gerçek yolların karşılaştırması Şekil 4.65'te, RMS hataları ise Şekil 4.66'te gösterilmektedir.



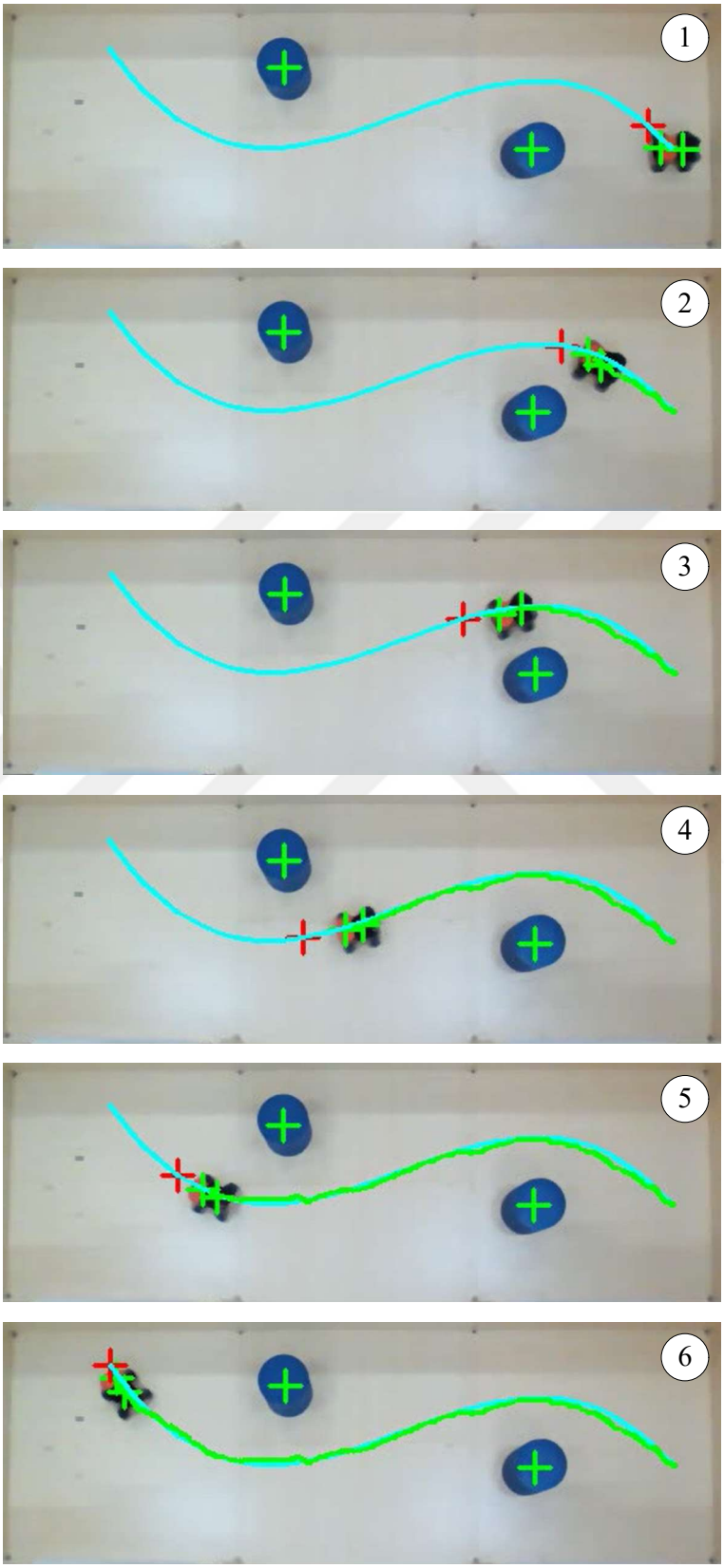
Şekil 4.61. Ortam 20 için robotun hareket aşamaları



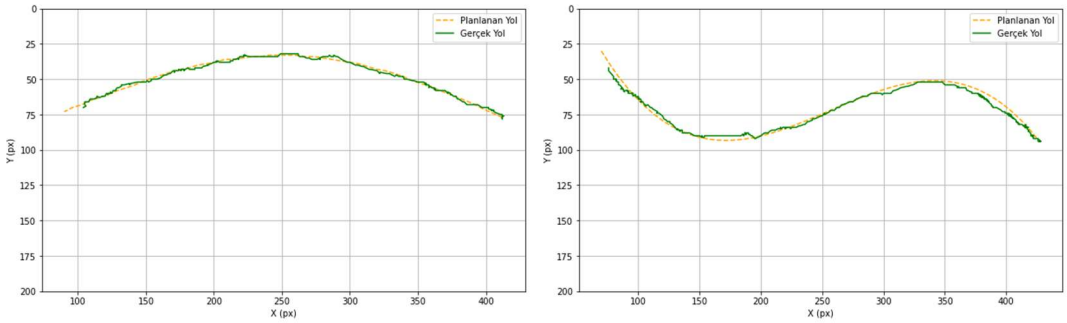
Şekil 4.62. Ortam 20 için robotun hareket aşamalarının kamera görüntüleri (Planlanan yol turkuaz çizgi, gerçek yol ise yeşil çizgi ile temsil edilmiştir.)



Şekil 4.63. Ortam 21 için robotun hareket aşamaları

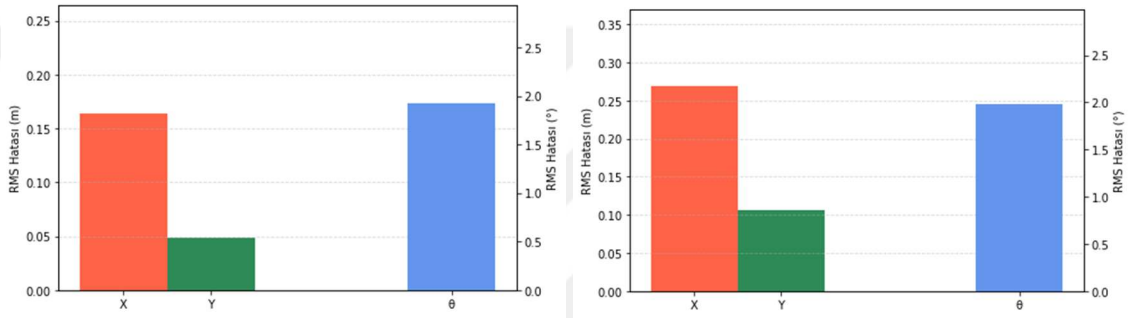


Şekil 4.64. Ortam 21 için robotun hareket aşamalarının kamera görüntüleri (Planlanan yol turkuaz çizgi, gerçek yol ise yeşil çizgi ile temsil edilmiştir.)



(a) **(b)**

Şekil 4.65. Planlanan ve gerçek yolların karşılaştırması: **(a)** Ortam 20, **(b)** Ortam 21



(a) **(b)**

Şekil 4.66. RMS hataları: **(a)** Ortam 20, **(b)** Ortam 21

Şekil 4.61-4.66 göz önüne alındığında, mobil robotun referans yolu başarılı bir şekilde takip ettiği görülmektedir. Bu takip kontrolü boyunca kayda değer bir sapma veya kararsızlık gözlemlenmemiştir. Ortam 20'de X, Y ve θ için RMS hataları sırasıyla 0.164 metre, 0.049 metre ve 1.93° olarak hesaplanmıştır. Ortam 21'de ise RMS hataları sırasıyla 0.269 metre, 0.106 metre ve 1.98° olarak hesaplanmıştır. Bu hatalar 4.5 x 1.5 metrelik platform ve 0.26 x 0.24 metrelik robot boyutlarıyla karşılaştırıldığında kabul edilebilir seviyelerdedir. Bu doğrultuda tasarlanan kontrol algoritmasının referans yol takibinde etkin performans sergilediği söylenebilir. Tasarlanan algoritma belirlenen ara hedef noktalar sayesinde hassas bir yönelim takip kontrolü sağlamıştır.

5. BÖLÜM

SONUÇ VE ÖNERİLER

5.1. Sonuçlar

Mobil robotlar, çevrelerini algılayarak otonom hareket edebilen sistemler olup, endüstri, tarım, lojistik ve arama-kurtarma gibi birçok alanda yaygın olarak kullanılmaktadır. Bu robotların etkin çalışabilmesi için haritalama, lokalizasyon ve yol planlama gibi temel becerilere sahip olması gerekir. Haritalama, robotun çevresini tanıyıp modellemesini sağlarken, lokalizasyon, bu harita üzerinde konumunu belirlemeye yardımcı olur. Ancak, robotun hedefe ulaşabilmesi için en kritik adım yol planlamadır. Yol planlama süreci, güvenli, verimli ve enerji tasarruflu rotalar oluşturmayı amaçlar ve çevresel engeller, hareket kısıtlamaları ve dinamik değişimler gibi faktörleri göz önünde bulundurur. Yol planlama yöntemleri genel olarak klasik arama algoritmaları, metasezgisel yaklaşımlar ve makine öğrenme tabanlı teknikler olarak üç gruba ayrılır. Klasik algoritmalar sabit haritalarda etkili çözümler sunarken, metasezgisel yöntemler daha geniş çözüm uzaylarını keşfederek karmaşık ve dinamik çevrelerde daha uygun yollar belirleyebilir. Makine öğrenme tabanlı yaklaşımlar ise çevresel geri bildirimleri değerlendirerek adaptif ve esnek çözümler sunar, özellikle dinamik engellerin bulunduğu ortamlarda robotun karar verme sürecini güçlendirir.

Bu tez çalışması kapsamında mobil robotların yol planlama sorunlarına yönelik çözümler geliştirmek amacıyla dört farklı simülasyon çalışması ve bir gerçek zamanlı çalışma gerçekleştirilmiştir. Birinci çalışmada tek bir mobil robotun ızgara ortamında küresel yol planlaması için bir IABC algoritması önerilmiştir. Bu algoritma için iki mekanizma düşünülmüştür: Doğrusallaştırma stratejisi ve yerel arama stratejisi. Doğrusallaştırma stratejisi, ızgara ortamında oluşturulan yolun gereksiz köşelerini ortadan kaldırmaya

odaklanan bir kullanım tabanlı iyileştirmedir. Yerel arama stratejisi, en iyi çözümün maliyetini daha da optimize etmeyi amaçlayan bir keşif tabanlı mekanizmadır. Böylece bu iyileştirmeler algoritmanın kullanım-keşif dengesini bozmadan gerçekleştirilmiştir. İkinci çalışmada yol planlama probleminin basitleştirilmesi üzerinde durulmuştur. Bunun için engellerin kümelenmesiyle ortam karmaşıklığının azaltılması ve bu sayede algoritmaların çalışma hızlarının artırılması amaçlanmıştır. Bu amaç doğrultusunda, tek bir mobil robotun sürekli uzay ortamında küresel yol planlaması için metasezgisel ve kümeleme algoritmalarının bir arada kullanıldığı hibrit bir model önerilmiştir. Üçüncü çalışmada çok sayıda mobil robotun sürekli uzay ortamında yerel yol planlaması için sdSCA algoritması önerilmiştir. Bu algoritmada temel SCA'nın orijinal güncelleme stratejisine ilave olarak birkaç diferansiyel tabanlı güncelleme stratejisi eklenmiştir. Bu algoritmada popülasyondaki her çözüm bu stratejilerden birini seçerek kendini günceller. Ancak bu seçim rastgele değildir, rulet tekerleğine dayalı bir olasılık hesabı yoluyla gerçekleştirilir. Algoritma daha tatmin edici sonuçlar üreten stratejiyi adaptif bir şekilde öğrenir. Dördüncü çalışmada ise tek ve çok robotlu sistemlerinin ızgara ortamında küresel yol planlaması için ResNet tabanlı bir model önerilmiştir. Bu model genişletilmiş evrişim (dilated convolution), sıkma-uyarma bloğu (squeeze-and-excitation) ve dropout katmanları içerir. Gerçek zamanlı çalışmada ise bir mobil robotun bir ortamda planlanan bir yolu istenen şekilde takip etmesi için bir takip kontrolörü tasarlanmış ve robotun gerçek zamanlı takip kontrolü gerçekleştirilmiştir.

Bu çalışmalar ile elde edilen neticeleri listeleyecek olursak:

- Önerilen IABC, aynı boyuttaki ortamlarda ABC ile karşılaştırıldığında ortalama yol uzunluğunda %7-14'lük bir iyileşme gözlemlenmiştir. Ayrıca, farklı boyutlardaki ortamlarda gerçekleştirilen ölçeklenebilirlik testleri ABC'ye kıyasla %19-20 aralığında iyileşmeler ortaya koymuştur. İyi bilinen ve yeni geliştirilmiş algoritmalarla yapılan karşılaştırmalar %2-78 aralığında iyileşmeler görülmüştür. Son olarak, güncel çalışmalardaki sonuçlara kıyasla %1-37 aralığında iyileşmeler elde edilmiştir.

- Önerilen hibrit modelin analizi sonucu, yol uzunluğu açısından küçük seviyelerde kayıplar gözlemlenmiştir. Ancak çalışma süreleri incelendiğinde, modelin bu kayıpları fazlasıyla telafi ettiği ve çalışma hızı bakımından kazanç elde edildiği

söylenebilir. Ayrıca farklı ortamlarda algoritmaların test edilmesi sonucu değiştirmemiştir. Dahası, önerilen modelde PSO'ya ek olarak TLBO, ABC, DE, GA algoritmaları ve KMC'ye ek olarak HC yöntemi de kullanılmıştır. Bu değerlendirme sonucunda yol uzunluğu açısından en iyi performansı TLBO ve ABC algoritmaları gösterirken en hızlı algoritma GA olmuştur. Ayrıca HC yönteminin KMC yöntemine göre daha verimli çalıştığı söylenebilir.

- Önerilen sdSCA, CEC2015 ve CEC2020 test problemlerinde uygulanmış ve başarılı sonuçlar elde etmiştir. CEC2015 problemlerinde belirgin bir iyileşme sağlanırken, CEC2020'de kazanan algoritmalarla rekabet edebilecek düzeyde performans göstermiştir. Algoritmanın etkinliği, statik ve dinamik engeller içeren üç farklı senaryoda çoklu robotların yol planlaması için yapılan simülasyonlarda da test edilmiştir. Ortam 17'de toplam adım sayısı %40.63, toplam yol uzunluğu %40.98 oranında azaltılmıştır. Ortam 18'de bu oranlar sırasıyla %40.36 ve %40.86 olarak gerçekleşirken, Ortam 19'da %44.00 ve %44.25'lik bir iyileşme sağlanmıştır.

- Önerilen IResNet modeli tek robotlu senaryoda, özellikle daha büyük ortamlarda, FCN ile yol planlama başarısında rekabet halinde olsa da IResNet'in tahmin ettiği yollar daha optimaldir. Çok robotlu senaryoda, IResNet FCN'yi hem yol planlama başarısında üstün gelmiş hem de daha optimal yollar planlamıştır. Ayrıca robot sayısı ve ortam karmaşıklığı arttıkça IResNet'in avantajı daha belirgin hale gelir.

- Gerçek zamanlı çalışmada tasarlanan kontrolör kullanılarak robotun planlanan yolda kayda değer bir hata olmadan hareket ederek hedef noktaya vardığı gözlemlenmiştir. Ortam 20-21'de RMS hataları X ekseni için 0.164-0.269 metre, Y ekseni için RMS hataları 0.049-0.106 metre, θ için RMS hataları 1.93°-1.98° olarak ölçülmüştür. Bu hatalar 4.5 x 1.5 metrelik platform ve 0.26 x 0.24 metrelik robot boyutlarıyla karşılaştırıldığında kabul edilebilir seviyelerdedir.

Bu tezde yapılan çalışmalarla, mobil robotların yol planlama süreçlerinde etkinliğin artırılması ve karşılaşılabilecek zorluklara yönelik çözümler geliştirilmesi amaçlanmıştır. Bu doğrultuda, farklı algoritmalar incelenmiş, çeşitli senaryolar üzerinde testler gerçekleştirilmiş ve elde edilen sonuçlar doğrultusunda yol planlama performansının

iyileştirilmesine yönelik öneriler sunulmuştur. Geliştirilen yöntemler, hem statik hem de dinamik engellerin bulunduğu ortamlarda robotların daha verimli ve güvenli bir şekilde hareket edebilmesine katkı sağlamayı hedeflemiştir. Böylece, mobil robotların farklı uygulama alanlarında daha etkin kullanılabilmesi için önemli kazanımlar elde edilmiştir.

5.2. Gelecekteki Çalışmalar

Bu çalışmada geliştirilen algoritmaların ve modellerin başarısı, mobil robotların yol planlamasında önemli iyileştirmeler sağladığını göstermektedir. Gelecekteki çalışmalarda, önerilen yöntemlerin farklı robot türleri ve daha karmaşık dinamik ortamlar üzerinde test edilmesi hedeflenebilir. Özellikle, gerçek dünyadaki belirsizlikleri ve sensör hatalarını daha iyi yönetebilmek için çevrimiçi öğrenme ve adaptif kontrol mekanizmaları entegre edilebilir. Ayrıca, daha büyük ölçekli çoklu robot sistemlerinde koordinasyon ve çakışma önleme stratejilerinin geliştirilmesi, robotların otonom hareket kabiliyetini artıracaktır.

Bunun yanı sıra, derin öğrenme tabanlı modellerin yol planlama süreçlerine daha etkin entegrasyonu önemli bir araştırma alanı olarak öne çıkmaktadır. Önerilen ResNet tabanlı modelin farklı ağ yapılarıyla genişletilerek daha karmaşık engel senaryolarına uyarlanması ve transfer öğrenme tekniklerinin kullanılması, algoritmaların genelleştirme yeteneğini artırabilir. Ayrıca, hibrit metasezgisel yöntemlerin gelişmiş optimizasyon teknikleriyle birleştirilmesi, yol planlama süreçlerinde daha hızlı ve verimli çözümler sunulmasını sağlayabilir. Son olarak gerçek zamanlı çalışmara yönelik, haberleşme yeteneğine sahip çoklu mobil robotlarla işbirlikçi yol planlaması ve takip kontrolü üzerine odaklanılabilir.

KAYNAKÇA

1. Loganathan, A., Ahmad**,** N. S., 2023. A systematic review on recent advances in autonomous mobile robot navigation. **Engineering Science and Technology, an International Journal, 40**: 101343.
2. Borkar, K. K., Aljrees, T., Pandey, S. K., Kumar, A., Singh, M. K., Sinha, A., Singh K. U., Sharma, V., 2023. Stability analysis and navigational techniques of wheeled mobile robot: A review. **Processes, 11** (12): 3302.
3. Zhang, Y., Zhao, Q., 2024. Complex environment based on improved A* algorithm research on path planning of inspection robots. **Processes, 12** (5): 855.
4. Liu, Z., Guo, S., Yu, F., Hao, J., Zhang, P., 2024. Improved A* algorithm for mobile robots under rough terrain based on ground trafficability model and ground ruggedness model. **Sensors, 24** (15): 4884.
5. Li, C., Huang, X., Ding, J., Song, K., Lu, S., 2022. Global path planning based on a bidirectional alternating search A* algorithm for mobile robots. **Computers & Industrial Engineering, 168**: 108123.
6. Wang, D., Liu, Q., Yang, J., & Huang, D., 2024. Research on path planning for intelligent mobile robots based on improved A* algorithm. **Symmetry, 16** (10): 1311.
7. Guo, S., Gong, J., Shen, H., Yuan, L., Wei, W., Long, Y., 2024. DBVSB-P-RRT*: A path planning algorithm for mobile robot with high environmental adaptability and ultra-high speed planning. **Expert Systems with Applications, 266**: 126123.
8. Zhong, H., Cong, M., Wang, M., Du, Y., Liu, D., 2024. HB-RRT: A path planning algorithm for mobile robots using Halton sequence-based rapidly-exploring random tree. **Engineering Applications of Artificial Intelligence, 133**: 108362.
9. Han, L., He, L., Sun, X., Li, Z., Zhang, Y., 2023. An enhanced adaptive 3D path planning algorithm for mobile robots with obstacle buffering and improved Theta* using minimum snap trajectory smoothing. **Journal of King Saud University-Computer and Information Sciences, 35** (10): 101844.
10. Huang, Y., Wang, H., Han, L., Xu, Y., 2024. Robot path planning in narrow passages based on improved PRM method. **Intelligent Service Robotics, 17**: 609-620.

11. Alshammrei, S., Boubaker, S., Kolsi, L., 2022. Improved dijkstra algorithm for mobile robot path planning and obstacle avoidance. **Comput. Mater. Contin, 72**: 5939-5954.
12. Li, Y., Song, H., Ji, Y., Zhang, L., 2024. Path planning method and control of mobile robot with uncertain dynamics based on improved artificial potential field and its application in health monitoring. **Mathematics, 12** (19): 2965.
13. Cai, Z., Liu, J., Xu, L., Wang, J., 2024. Cooperative path planning study of distributed multi-mobile robots based on optimised ACO algorithm. **Robotics and Autonomous Systems, 179**: 104748.
14. Huo, F., Zhu, S., Dong, H., Ren, W., 2024. A new approach to smooth path planning of Ackerman mobile robot based on improved ACO algorithm and B-spline curve. **Robotics and Autonomous Systems, 175**: 104655.
15. Ab Wahab, M. N., Nazir, A., Khalil, A., Ho, W. J., Akbar, M. F., Noor, M. H. M., Mohamed, A. S. A., 2024. Improved genetic algorithm for mobile robot path planning in static environments. **Expert Systems with Applications,** 249: 123762.
16. Duan, P., Yu, Z., Gao, K., Meng, L., Han, Y., Ye, F. 2024. Solving the multi-objective path planning problem for mobile robot using an improved NSGA-II algorithm. **Swarm and Evolutionary Computation, 87**: 101576.
17. Tao, B., Kim, J. H., 2024. Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy. **Journal of King Saud University-Computer and Information Sciences, 36** (2): 101974.
18. Lin, S., Li, F., Li, X., Jia, K., Zhang, X., 2022. Improved artificial bee colony algorithm based on multi-strategy synthesis for UAV path planning. **IEEE Access, 10**: 119269-119282.
19. Li, F., Fan, X., Hou, Z., 2020. A firefly algorithm with self-adaptive population size for global path planning of mobile robot. **IEEE Access, 8**: 168951-168964.
20. Zhang, M., Hao, P., 2024. 2D and 3D path planning for mobile robots based on improved SSA algorithm. **International Journal of Intelligent Robotics and Applications,** 1-13.
21. Tang, C., Li, W., Han, T., Yu, L., Cui, T., 2024. Multi-strategy improved harris hawk optimization algorithm and its application in path planning. **Biomimetics, 9** (9): 552.

22. Gao, Y., Li, Z., Wang, H., Hu, Y., Jiang, H., Jiang, X., Chen, D., 2024. An improved spider-wasp optimizer for obstacle avoidance path planning in mobile robots. **Mathematics, 12** (17): 2604.
23. Shi, K., Wu, Z., Jiang, B., Karimi, H. R., 2023. Dynamic path planning of mobile robot based on improved simulated annealing algorithm. **Journal of the Franklin Institute, 360** (6): 4378-4398.
24. Yıldırım, M. Y., Akay, R., 2021. A comparative study of optimization algorithms for global path planning of mobile robots. **Sakarya University Journal of Science, 25** (2): 417-428.
25. Galarza-Falfan, J., García-Guerrero, E. E., Aguirre-Castro, O. A., López-Bonilla, O. R., Tamayo-Pérez, U. J., Cárdenas-Valdez, J. R., Hernández-Mejía, C., Borrego-Dominguez, S., Inzunza-Gonzalez, E., 2024. Path planning for autonomous mobile robot using intelligent algorithms. **Technologies, 12** (6): 82.
26. Han, Z., 2023. Multimodal intelligent logistics robot combining 3D CNN, LSTM, and visual SLAM for path planning and control. **Frontiers in Neurorobotics, 17**: 1285673.
27. Farkh, R., Quasim, M. T., Al Jaloud, K., Alhuwaimel, S., Siddiqui, S. T., 2021. Computer vision-control-based CNN-PID for mobile robot. **Computers, Materials & Continua, 68** (1).
28. Li, P., Chen, D., Wang, Y., Zhang, L., Zhao, S., 2024. Path planning of mobile robot based on ımproved TD3 algorithm in dynamic environment. **Heliyon, 10** (11): e32167.
29. Deshpande, S. V., Harikrishnan, R., Walambe, R., 2024. POMDP-based probabilistic decision making for path planning in wheeled mobile robot. **Cognitive Robotics, 4**: 104-115.
30. Deshpande, S. V., Harikrishnan, R., Ibrahim, B. S. K. K., Ponnuru, M. D. S., 2024. Mobile robot path planning using deep deterministic policy gradient with differential gaming (DDPG-DG) exploration. **Cognitive Robotics, 4**: 156-173.
31. Zhou, Q., Lian, Y., Wu, J., Zhu, M., Wang, H., Cao, J., 2024. An optimized Q-Learning algorithm for mobile robot local path planning. **Knowledge-Based Systems, 286**: 111400.

32. Zhang, L., Cai, Z., Yan, Y., Yang, C., Hu, Y., 2024. Multi-agent policy learning-based path planning for autonomous mobile robots. **Engineering Applications of Artificial Intelligence, 129**: 107631.
33. Yan, C., Chen, G., Li, Y., Sun, F., Wu, Y., 2023. Immune deep reinforcement learning-based path planning for mobile robot in unknown environment. **Applied Soft Computing, 145**: 110601.
34. Deshpande, S. V., Harikrishnan, R., Sampe, J., Patwa, A., 2024. An algorithm to create model file for partially observable markov decision process for mobile robot path planning. **MethodsX, 12**: 102552.
35. Low, E. S., Ong, P., Low, C. Y., Omar, R., 2022. Modified Q-learning with distance metric and virtual target on path planning of mobile robot. **Expert Systems with Applications, 199**: 117191.
36. Das, S., Mishra, S. K., 2022. A machine learning approach for collision avoidance and path planning of mobile robot under dense and cluttered environments. **Computers and Electrical Engineering, 103**: 108376.
37. Chen, L., Wang, Q., Deng, C., Xie, B., Tuo, X., Jiang, G., 2024. Improved double deep Q-network algorithm applied to multi-dimensional environment path planning of hexapod robots. **Sensors, 24** (7): 2061.
38. Hu, L., Wei, C., Yin, L., 2025. Fuzzy A∗ quantum multi-stage Q-learning artificial potential field for path planning of mobile robots. **Engineering Applications of Artificial Intelligence, 141**: 109866.
39. Wang, Y., Fu, C., Huang, R., Tong, K., He, Y., Xu, L., 2024. Path planning for mobile robots in greenhouse orchards based on improved A* and fuzzy DWA algorithms. **Computers and Electronics in Agriculture, 227**: 109598.
40. Tao, B., Kim, J. H., 2024. Deep reinforcement learning-based local path planning in dynamic environments for mobile robot. **Journal of King Saud University-Computer and Information Sciences, 36** (10): 102254.
41. Zhang, D., Yin, Y., Xiong, G., Zou, S., 2024. A bi-level hybrid algorithm for solving multi-target inspection path planning problem of mobile robot in complex radioactive indoor environment. **Expert Systems with Applications, 266**: 126095.

42. Wei, L., Xu, Q., Hu, Z., 2024. Mobile robot path planning based on multi-experience pool deep deterministic policy gradient in unknown environment. **International Journal of Machine Learning and Cybernetics, 15**: 5823-5837.
43. Gao, Y., Han, Q., Feng, S., Wang, Z., Meng, T., Yang, J., 2024. Improvement and fusion of D* lite algorithm and dynamic window approach for path planning in complex environments. **Machines, 12** (8): 525.
44. Li, G., Liu, C., Wu, L., Xiao, W., 2023. A mixing algorithm of ACO and ABC for solving path planning of mobile robot. **Applied Soft Computing, 148**: 110868.
45. Yu, Z., Yuan, J., Li, Y., Yuan, C., Deng, S., 2023. A path planning algorithm for mobile robot based on water flow potential field method and beetle antennae search algorithm. **Computers and Electrical Engineering, 109**: 108730.
46. Zhang, D., Yin, Y. B., Luo, R., Zou, S. L., 2023. Hybrid IACO-A*-PSO optimization algorithm for solving multiobjective path planning problem of mobile robot in radioactive environment. **Progress in Nuclear Energy, 159**: 104651.
47. Tian, T., Liang, Z., Wei, Y., Luo, Q., Zhou, Y., 2024. Hybrid whale optimization with a firefly algorithm for function optimization and mobile robot path planning. **Biomimetics, 9** (1): 39.
48. Abdulsaheb, J. A., Kadhim, D. J., 2023. Classical and heuristic approaches for mobile robot path planning: A survey. **Robotics, 12** (4): 93.
49. Karur, K., Sharma, N., Dharmatti, C., Siegel, J. E., 2021. A survey of path planning algorithms for mobile robots. **Vehicles, 3** (3): 448-468.
50. Toufan, N., Niknafs, A., 2020. Robot path planning based on laser range finder and novel objective functions in grey wolf optimizer. **SN Applied Sciences, 2** (8): 1324.
51. Bellman, R., 1966. Dynamic programming. ***Science*, *153*** (3731): 34-37.
52. Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. **Numerische Mathematik, 1**: 269-271.
53. Moore, E.F., 1959. The shortest path through a maze, 285-292. *The International Symposium on the Theory of Switching*, June 29-July 1, 1959, Massachusetts, USA.
54. Hart, P. E., Nilsson, N. J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. **IEEE transactions on Systems Science and Cybernetics, 4** (2): 100-107.

55. LaValle, S., 1998. Rapidly-exploring random trees: A new tool for path planning. *Research Report,* 9811.
56. Holland, J. H., 1992. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor.
57. Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization, 11:** 341-359.
58. Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, 1942-1948. *International Conference on Neural Networks*, November 27-December 1, 1995, Perth, WA, Australia.
59. Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. **Journal of Global Optimization, 39**: 459-471.
60. Rao, R. V., Savsani, V. J., Vakharia, D. P., 2011. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. **Computer-Aided Design, 43** (3): 303-315.
61. Salimi, H., 2015. Stochastic fractal search: a powerful metaheuristic algorithm. **Knowledge-Based Systems, 75**: 1-18.
62. Mirjalili, S., 2016. SCA: A sine cosine algorithm for solving optimization problems. **Knowledge-Based Systems, 96**: 120-133.
63. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A. H., 2021. The arithmetic optimization algorithm. **Computer Methods in Applied Mechanics and Engineering, 376**: 113609.
64. Manakitsa, N., Maraslidis, G. S., Moysis, L., Fragulis, G. F., 2024. A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision. **Technologies, 12** (2): 15.
65. Zhou, C., Huang, B., Fränti, P., 2022. A review of motion planning algorithms for intelligent robots. **Journal of Intelligent Manufacturing, 33** (2): 387-424.
66. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. **Proceedings of the IEEE, 86** (11): 2278-2324.

67. Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. **Advances in Neural Information Processing Systems**, 25.

68. Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. **ArXiv Preprint ArXiv:** 1409.1556.

69. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, 1-9. *IEEE Conference on Computer Vision and Pattern Recognition*, 7-12 June, 2015, Boston, MA, USA.

70. Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation, 3431-3440. *IEEE Conference on Computer Vision and Pattern Recognition*, 7-12 June, 2015, Boston, MA, USA.

71. He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, 770-778. *IEEE Conference on Computer Vision and Pattern Recognition*, 27-30 June, 2016, Las Vegas, NV, USA.

72. Sutton, R. S., McAllester, D., Singh, S., Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. **Advances in neural information processing systems**, 12.

73. Watkins, C. J., Dayan, P., 1992. Q-learning. **Machine Learning, 8**: 279-292.

74. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. **ArXiv Preprint Arxiv**: 1509.02971.

75. Karaman, A., Pacal, I., Basturk, A., Akay, B., Nalbantoglu, U., Coskun, S., Sahin O., Karaboga, D., 2023. Robust real-time polyp detection system design based on YOLO algorithms by optimizing activation functions and hyper-parameters with artificial bee colony (ABC). **Expert Systems with Applications, 221**: 119741.

76. Ye, T., Wang, W., Wang, H., Cui, Z., Wang, Y., Zhao, J., Hu, M., 2022. Artificial bee colony algorithm with efficient search strategy based on random neighborhood structure. **Knowledge-Based Systems, 241**: 108306.

77. Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. **Applied Soft Computing, 8** (1): 687-697.

78. Hansen, N., Ostermeier, A., 2001. Completely derandomized self-adaptation in evolution strategies. **Evolutionary Computation, 9** (2): 159-195.

79. Braik, M., Hammouri, A., Atwan, J., Al-Betar, M. A., Awadallah, M. A., 2022. White shark optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. **Knowledge-Based Systems, 243**: 108457.
80. Al-Betar, M. A., Awadallah, M. A., Braik, M. S., Makhadmeh, S., Doush, I. A., 2024. Elk herd optimizer: a novel nature-inspired metaheuristic algorithm. **Artificial Intelligence Review, 57** (3): 48.
81. Tian, T., Liang, Z., Wei, Y., Luo, Q., Zhou, Y., 2024. Hybrid whale optimization with a firefly algorithm for function optimization and mobile robot path planning. **Biomimetics, 9** (1).
82. Li, G., Liu, C., Wu, L., Xiao, W., 2023. A mixing algorithm of ACO and ABC for solving path planning of mobile robot. **Applied Soft Computing, 148**: 110868.
83. Zhang, Z., He, R., Yang, K., 2022. A bioinspired path planning approach for mobile robots based on improved sparrow search algorithm. **Advances in Manufacturing, 10** (1): 114-130.
84. Liu, C., Wu, L., Xiao, W., Li, G., Xu, D., Guo, J., Li, W., 2023. An improved heuristic mechanism ant colony optimization algorithm for solving path planning. **Knowledge-Based Systems, 271**: 110540.
85. Li, Y., Yang, J., 2024. Path planning and obstacle avoidance technology for mobile robots using improved a-star algorithm, 121-124. *5th International Seminar on Artificial Intelligence, Networking and Information Technology,* March 29-31, 2024, Nanjing, China.
86. Chołodowicz, E., Figurowski, D., 2017. Mobile robot path planning with obstacle avoidance using particle swarm optimization. **Pomiary Automatyka Robotyka, 21** (3): 59-68.
87. Zhao, F., Hung, D. L., Wu, S., 2020. K-means clustering-driven detection of time-resolved vortex patterns and cyclic variations inside a direct injection engine. **Applied Thermal Engineering, 180**: 115810.
88. Aytaç, E., 2020. Unsupervised learning approach in defining the similarity of catchments: Hydrological response unit based k-means clustering, a demonstration on Western Black Sea Region of Turkey. **International soil and water conservation research, 8** (3): 321-331.
89. Rehman, T. U., Mahmud, M. S., Chang, Y. K., Jin, J., Shin, J., 2019. Current and future applications of statistical machine learning algorithms for agricultural

machine vision systems. **Computers and electronics in agriculture, 156**: 585-605.

90. Tang, Y., Zhou, R., Sun, G., Di, B., Xiong, R., 2020. A novel cooperative path planning for multirobot persistent coverage in complex environments. **IEEE Sensors Journal, 20** (8): 4485-4495.
91. Peng, Y., Qu, D., Zhong, Y., Xie, S., Luo, J., Gu, J., 2015. The obstacle detection and obstacle avoidance algorithm based on 2-d lidar, 1648-1653. *IEEE International Conference on Information and Automation*, August 8-10, 2015, Lijiang, Yunnan, China.
92. Lin, S., Liu, A., Wang, J., Kong, X., 2022. A review of path-planning approaches for multiple mobile robots. **Machines, 10** (9): 773.
93. Toufan, N., Niknafs, A., 2020. Robot path planning based on laser range finder and novel objective functions in grey wolf optimizer. **SN Applied Sciences, 2** (8): 1324.
94. Das, S., Suganthan, P. N., 2010. Differential evolution: A survey of the state-of-the-art. **IEEE Transactions on Evolutionary Computation, 15** (1): 4-31.
95. Zhang, J., Sanderson, A. C., 2009. JADE: adaptive differential evolution with optional external archive. **IEEE Transactions on Evolutionary Computation, 13** (5): 945-958.
96. Wang, Y., Cai, Z., Zhang, Q., 2011. Differential evolution with composite trial vector generation strategies and control parameters. **IEEE transactions on Evolutionary Computation, 15** (1): 55-66.
97. Liang, J. J., Qu, B. Y., Suganthan, P. N., Chen, Q., 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. **Technical Report 201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 29**: 625-640.
98. Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., Das, S., 2020. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. **Swarm and Evolutionary Computation, 56**: 100693.

99. Awadallah, M. A., Al-Betar, M. A., Bolaji, A. L. A., Alsukhni, E. M., Al-Zoubi, H., 2019. Natural selection methods for artificial bee colony with new versions of onlooker bee. **Soft Computing, 23** (15): 6455-6494.

100. Al-Betar, M. A., Aljarah, I., Awadallah, M. A., Faris, H., Mirjalili, S., 2019. Adaptive β-hill climbing for optimization. **Soft Computing, 23** (24): 13489-13512.

101. Zhao, H., Zhang, C., Ning, J., 2019. A best firework updating information guided adaptive fireworks algorithm. **Neural Computing and Applications, 31**: 79-99.

102. Kumar, A., Das, S., Zelinka, I., 2020. A self-adaptive spherical search algorithm for real-world constrained optimization problems, 13-14. *The 2020 Genetic and Evolutionary Computation Conference Companion*, July 26, 2020, Cancún, Mexico.

103. Gurrola-Ramos, J., Hernàndez-Aguirre, A., Dalmau-Cedeño, O., 2020. COLSHADE for real-world single-objective constrained optimization problems, 1-8. *2020 IEEE Congress on Evolutionary Computation (CEC)*, July 19-24, 2020, Glasgow, United Kingdom.

104. Sallam, K. M., Elsayed, S. M., Chakrabortty, R. K., Ryan, M. J., 2020. Multi-operator differential evolution algorithm for solving real-world constrained optimization problems, 1-8. *2020 IEEE Congress on Evolutionary Computation (CEC)*, July 19-24, 2020, Glasgow, United Kingdom.

105. Kumar, A., Das, S., Zelinka, I., 2020. A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems, 11-12. *The 2020 Genetic and Evolutionary Computation Conference Companion*, July 26, 2020, Cancún, Mexico.

106. Xu, J., Xu, L., 2021. Optimal stochastic process optimizer: A new metaheuristic algorithm with adaptive exploration-exploitation property. **IEEE Access, 9**: 108640-108664.

107. Tessema, B., Yen, G. G., 2006. A self adaptive penalty function based algorithm for constrained optimization, 246-253. *IEEE International Conference On Evolutionary Computation*, July 16-21, 2006, Vancouver, BC, Canada.

108. Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. **Advances in Engineering Software, 95**: 51-67.

109. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. **Future Generation Computer Systems, 97**: 849-872.

110. Kulvicius, T., Herzog, S., Lüddecke, T., Tamosiunaite, M., Wörgötter, F., 2020. One-shot multi-path planning for robotic applications using fully convolutional networks, 1460-1466. *IEEE International Conference on Robotics and Automation,* May 31-August 31, 2020, Paris, France.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Mustafa Yusuf YILDIRIM

Uyruğu: Türkiye (T.C)

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Yüksek Lisans	Karabük Üniversitesi, Mekatronik Mühendisliği Anabilim Dalı	2019
Lisans	Erciyes Üniversitesi, Mekatronik Mühendisliği Bölümü	2015
Lise	Bolu Atatürk Anadolu Lisesi	2010

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2019-Hâlen	Erciyes Üniversitesi, Fen Bilimleri Enstitüsü	Araştırma Görevlisi
2017-2019	Niğde Ömer Halisdemir Üniversitesi, Mekatronik Mühendisliği Bölümü	Araştırma Görevlisi
2016-2017	Aremas Özderya Asansör Tic. ve San. Ltd. Şti.	Mühendis

YABANCI DİL

İngilizce

YAYINLAR

SCI, SSCI, AHCI İndekslerine Giren Dergilerde Yayımlanan Makaleler

1. Yıldırım, M. Y., Akay. R. (2025). An efficient grid-based path planning approach using improved artificial bee colony algorithm. *Knowledge-Based System*, *318*, ss 113528.

2. Akay, R., Yıldırım, M. Y. (2025). SBA*: An efficient method for 3D path planning of unmanned vehicles. *Mathematics and Computers in Simulation*, *231*, ss 294-317.

3. Akay, R., Yıldırım, M. Y. (2023). Multi-strategy and self-adaptive differential sine-cosine algorithm for multi-robot path planning. *Expert Systems with Applications, 232*, ss 120849.

4. Suveren M., Akay R., Yıldırım M. Y., Kanaan M. (2022). Application of hybrid metaheuristic with Levenberg-Marquardt algorithm for 6-dimensional magnetic localization. *Evolving Systems, 13*(6), ss 849-867.

5. Yıldırım M. Y., Akay R. (2021). Mobil robotlar için çok engelli ortamlarda hızlı yol planlama. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 36*(3), ss 1552-1564.

Diğer Dergilerde Yayımlanan Makaleler

1. Savas S., Yıldırım M. Y., Akay R. (*kabul edildi*). The effect of path linearity on mobile robot path planning and tracking control. Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi.

2. Yıldırım M. Y., Akay R. (2024). Robot tutucu problemi için çok stratejili aritmetik optimizasyon algoritması. *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji, 12*, ss 108-116.

3. Yıldırım M. Y., Akay R. (2021). Mobil robotların yol planlamasında doğrusallığın incelenmesi. *Avrupa Bilim ve Teknoloji Dergisi*, *24*, ss 138-142.

4. Yıldırım M. Y., Akay R. (2021). A comparative study of optimization algorithms for global path planning of mobile robots. *Sakarya University Journal of Science, 25*(2), ss 417-428.

5. Yıldırım M. Y., Anutgan M. (2020). Development of an industrial robotic arm education kit based on object recognition and robot kinematics for engineers. *Journal of Selcuk-Technic, Special Issue,* ss 47-65.

Hakemli Kongre / Sempozyum Bildiri Kitaplarında Yer Alan Yayınlar

1. Yıldırım M. Y., Akay R. (2024). Welded beam design optimization using honey badger algorithm. *1st Bilsel International Aspendos Scientific Researches Congress,* Antalya, Türkiye, 24-25 Şubat 2024, ss 59-66.

2. Yıldırım M. Y., Akay R. (2021). Izgara bazlı yol planlama için matematik tabanlı metasezgisellerin karşılaştırılması. International Conference on Design, Research and Development, Kayseri, Türkiye, 15-18 Aralık 2021, ss 157.

3. Yıldırım M. Y., Akay R. (2020). DC motor hız kontrolü için PID denetleyici parametrelerinin PSO algoritması ile gerçek zamanlı optimizasyonu. *2nd International Eurasian Conference on Science, Engineering and Technology*, Gaziantep, Türkiye, 7-9 Ekim 2020, ss 543-549.

4. Yıldırım M. Y., Anutgan M. (2018). Stereo görme ve 3 eksenli robot kol kullanılarak nesne sınıflandırma. *Anadolu I. Uluslararası Multidisipliner Çalışmalar Kongresi*, Diyarbakır, Türkiye, 28-29 Aralık 2018, ss 412-418.

5. Yıldırım M. Y., Anutgan M. (2018). Ters Kinematik Analizin Yapay Sinir Ağları ile Simülasyonu. *Anadolu I. Uluslararası Multidisipliner Çalışmalar Kongresi*, Diyarbakır, Türkiye, 28-29 Aralık 2018, ss 407-411.