KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MEKATRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OTONOM MOBİL ROBOTLAR İÇİN METASEZGİSEL YÖNTEMLER KULLANILARAK YOL PLANLAMA ALGORİTMASININ GELİŞTİRİLMESİ

YUNUS TEZEL

KOCAELİ 2024

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MEKATRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OTONOM MOBİL ROBOTLAR İÇİN METASEZGİSEL YÖNTEMLER KULLANILARAK YOL PLANLAMA ALGORİTMASININ GELİŞTİRİLMESİ

YUNUS TEZEL

Dr. Öğr. Üyesi Suat KARAKAYA		
Danışman,	**Kocaeli Üniversitesi**	**.....**
Dr. Öğr. Üyesi Kaplan KAPLAN		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**
Prof. Dr. Veysel GAZİ		
Jüri Üyesi,	**Yıldız Teknik Üniversitesi**	**.....**

Tezin Savunulduğu Tarih: 27.06.2024

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/Projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

[X] Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

[ ] Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

.....
Yunus TEZEL

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanımı bana ait olacaktır.
Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.
Yükseköğretim kurulu tarafından yayınlanan ***"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

☒ Tezim/projem ile ilgili gizlilik kararı verilmemiştir.

.....
Yunus TEZEL

ÖNSÖZ VE TEŞEKKÜR

Tez çalışmamın her aşamasında gösterdiği ilgi, emek ve çalışmamın ilerlemesine olan katkıları için değerli danışmanım Dr. Suat KARAKAYA'ya çok teşekkür ederim. Ayrıca kendisini örnek aldığım, bölümümüzün emekli hocası ve eski danışmanım Prof. Dr. Hasan OCAK'a, lisans öğrenimim süresince bana olan katkılarından dolayı hocalarım Prof. Dr. H. Metin ERTUNÇ ve Doç. Dr. Selçuk KİZİR'e, lisans bitirme çalışması danışmanım Dr. Serkan ZEREN'e teşekkürlerimi sunarım.

Eğitim hayatım boyunca her zaman yanımda olan ve beni destekleyen anneme, babama, akrabalarıma ve hayatım boyunca aldığım her kararda beni destekleyeceğini bildiğim sevgili kardeşime gönülden teşekkür ederim. Her zaman bana zaman ayıran ve her konuda kendisine danışabildiğim dayım Dr. Mehmet Nurullah AKKURT'a, desteğini her daim hissettirdiği için teşekkür ederim. Bana olan desteklerini esirgemeyen arkadaşlarım Aleyna YILMAZ ve Barış KABLAN'a teşekkür ederim.

Yıldız Teknik Üniversitesi, Kontrol ve Otomasyon Bölümü'ne; başta dekanımız Prof. Dr. Şeref Naci ENGİN ve bölüm başkanımız Prof. Dr. Veysel GAZİ olmak üzere tüm hocalarıma ve araştırma görevlisi arkadaşlarıma destekleri ve sundukları mutlu çalışma ortamı için içtenlikle teşekkürlerimi sunarım.

Haziran-2024 Yunus TEZEL

İÇİNDEKİLER

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ i
YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI ii
ÖNSÖZ VE TEŞEKKÜR iii
İÇİNDEKİLER iv
ŞEKİLLER DİZİNİ vi
TABLOLAR DİZİNİ viii
SİMGELER VE KISALTMALAR DİZİNİ ix
ÖZET xi
ABSTRACT xii
1. GİRİŞ 1
2. GENEL BİLGİLER 5
2.1. Robotların Sınıflandırılması 10
2.1.1. Sabit Robotlar 12
2.1.1.1. Kartezyen (PPP) Düzenleşim 14
2.1.1.2. Silindirik (RPP) Düzenleşim 15
2.1.1.3. SCARA (RRP) Düzenleşim 15
2.1.1.4. Küresel (RRP) Düzenleşim 16
2.1.1.5. Dönel (RRR) Düzenleşim 16
2.1.2. Mobil Robotlar 17
2.2. Yol Planlama 19
2.2.1. Global Yol Planlama 19
2.2.2. Yerel Yol Planlama 20
2.3. Engel Çeşitleri 20
2.4. Harita Türleri 21
2.4.1. Izgara Tabanlı Haritalar 21
2.4.2. Nokta Tabanlı Haritalar 22
2.4.3. Ağ Tabanlı Haritalar 22
3. LİTERATÜR TARAMASI 25
4. YOL PLANLAMA ALGORİTMALARI 28
4.1. Geleneksel Algoritmalar 28
4.1.1. Hücrelere Ayrıştırma 28
4.1.2. Olasılıksal Yol Haritası 29
4.1.3. Yapay Potansiyel Alan 30
4.2. Sezgisel Algoritmalar 31
4.2.1. A* Algoritması 32
4.2.2. D* Algoritması 34
4.2.3. Dijkstra Algoritması 36
4.3. Metasezgisel Algoritmalar 38
4.3.1. Benzetimli Tavlama 40
4.3.2. Genetik Algoritma 46
4.3.3. Guguk Kuşu Arama 58
4.3.4. Parçacık Sürü Optimizasyonu 60
5. GLOBAL YOL PLANLAMA 65
5.1. Maliyet Fonksiyonu 65
5.2. Uygunluk Değeri 66
5.3. Engel Tespiti 68

5.4. Benzetimli Tavlama ile Yol Planlama 69
5.5. Genetik Algoritma ile Yol Planlama 74
5.6. Guguk Kuşu Arama ile Yol Planlama 76
6. ALGORİTMALARIN KARŞILAŞTIRILMASI 79
7. YEREL YOL PLANLAMA 83
7.1. Bresenham Çizgi Algoritması ile Engel Tespiti 83
7.2. Pencere Yöntemiyle Bağlantı Noktasını Belirleme 85
7.3. PSO ile Yerel Yol Planlama 87
8. SONUÇ VE ÖNERİLER 90
8.1. Elde Edilen Sonuçlar 90
8.2. Öneriler 94
KAYNAKLAR 95
KİŞİSEL YAYIN VE ESERLER 101
ÖZGEÇMİŞ 102

ŞEKİLLER DİZİNİ

Şekil 1.1. Yol Planlama Örneği 2
Şekil 1.2. Yol Planlama Algoritmaları 3
Şekil 2.1. Robotların Sınıflandırılması 10
Şekil 2.2. Endüstriyel Robot Kol 11
Şekil 2.3. Otonom Mobil Robot 11
Şekil 2.4. Uygulama Alanlarına Göre Robotların Sınıflandırılması 12
Şekil 2.5. Eğitim Robotu 13
Şekil 2.6. a) Kuka KR1000 Titan Seri Robotu b) Kuka KR Delta Paralel Robotu 13
Şekil 2.7. Seri Robot Kol Eklem Örnekleri: a) Prizmatik Eklem b) Döner Eklem 14
Şekil 2.8. Kartezyen Robot ve Çalışma Uzayı 15
Şekil 2.9. Silindirik Robot ve Çalışma Uzayı 15
Şekil 2.10. SCARA Robot ve Çalışma Uzayı 16
Şekil 2.11. Küresel Robot ve Çalışma Uzayı 16
Şekil 2.12. Dönel Robot ve Çalışma Uzayı 17
Şekil 2.13. Mobil Robotların Sınıflandırılması 17
Şekil 2.14. Mobil Robotlar: a) ERTUĞRUL Bomba İmha Robotu b) Baykar TB3 İHA c) Xiaomi Vacuum-Mop 2 Pro 18
Şekil 2.15. Yol Planlamada Engellerin Sınıflandırılması 20
Şekil 2.16. Izgara Tabanlı Harita Örneği 22
Şekil 2.17. Nokta Tabanlı Harita Örneği 23
Şekil 2.18. Ağ Tabanlı Harita Örneği 23
Şekil 4.1. Geleneksel Yol Planlama Algoritmaları 28
Şekil 4.2. Hücrelere Ayrıştırma Yöntemi 29
Şekil 4.3. Olasılıksal Yol Haritası ile Yol Planlama Aşamaları 29
Şekil 4.4. Olasılıksal Yol Haritası Algoritması Akış Diyagramı 30
Şekil 4.5. Yapay Potansiyel Alan Yöntemi ile Yol Planlama Örneği 30
Şekil 4.6. Sezgisel Yol Planlama Algoritmaları 31
Şekil 4.7. A* Algoritması Yol Bulma Problemi Örneği 33
Şekil 4.8. A* Algoritması Örneği Adım Adım Gösterimi 35
Şekil 4.9. D* Algoritması Örneği 36
Şekil 4.10. Dijkstra Algoritması Adımları 38
Şekil 4.11. Metasezgisel Algoritmaların Sınıflandırılması 40
Şekil 4.12. Tavlama İşlemi Adımları 41
Şekil 4.13. Tavlama ve Söndürme İşlemleri 42
Şekil 4.14. Tavlama İşlemi ile Benzetimli Tavlama Arasındaki Benzetim 42
Şekil 4.15. Komşuluk Hareketleri 43
Şekil 4.16. Benzetimli Tavlama Algoritması Akış Diyagramı 45
Şekil 4.17. Soğutma Çizelgesi Örnekleri 45
Şekil 4.18. Genetik Algoritma Akış Diyagramı 46
Şekil 4.19. Genetik Algoritma Çözüm Kümesinin İsimlendirilmesi 47
Şekil 4.20. Genetik Algoritma Kodlama Türleri 48
Şekil 4.21. Rulet Tekeri Yöntemi 49
Şekil 4.22. Turnuva Seçim Yöntemi 51
Şekil 4.23. Sıralama Tabanlı Seçim Örneği 52

Şekil 4.24. Kromozomların Çaprazlanması 53
Şekil 4.25. Tek Noktalı Çaprazlama 54
Şekil 4.26. Çok Noktalı Çaprazlama....54
Şekil 4.27. Reel Sayı Kodlama için BLX-α Yöntemi ile Çaprazlama....55
Şekil 4.28. Bit Düzeyinde Mutasyon İşlemi 56
Şekil 4.29. Reel Sayı Kodlamada Mutasyon İşlemi....57
Şekil 4.30. Elitizm İşlemi....57
Şekil 4.31. Guguk Kuşunun Yerleştirdiği Yumurtalar....58
Şekil 4.32. Brownian Yürüyüşü ve Lévy Uçuşu....59
Şekil 4.33. Parçacık Sürü Optimizasyonu Benzetimi 61
Şekil 4.34. Parçacık Sürü Optimizasyonu Bileşenleri 62
Şekil 4.35. PSO Algoritması Sözde kodu 63
Şekil 4.36. Parçacık Sürü Optimizasyonu Akış Diyagramı 63
Şekil 5.1. Mesafe Hesaplama Yöntemleri: (a) Öklid Mesafesi, (b) Manhattan Mesafesi, (c) Mincowski Mesafesi 66
Şekil 5.2. Engel Tespit Yöntemi....69
Şekil 5.3. Yol Planlaması için Rastgele Çözüm....70
Şekil 5.4. Yeni Çözüm Üretme Stratejisi 70
Şekil 5.5. Benzetimli Tavlama Algoritması ile Planlanan Yol 72
Şekil 5.6. Benzetimli Tavlama için Bulunan Çözümlerin Değerleri....72
Şekil 5.7. Benzetimli Tavlama için Bulunan En İyi Değer 73
Şekil 5.8. Benzetimli Tavlama için Her İterasyondaki Kabul Olasılığı....73
Şekil 5.9. Benzetimli Tavlamada Kullanılan Soğutma Çizelgesi 73
Şekil 5.10. Genetik Algoritma ile Planlanan Yol 75
Şekil 5.11. Genetik Algoritma için Çözümlerin Değerleri....75
Şekil 5.12. Genetik Algoritma için Bulunan En İyi Değer 76
Şekil 5.13. Guguk Kuşu Arama Algoritması ile Planlanan Yol 77
Şekil 5.14. Guguk Kuşu Arama Algoritması için Bulunan Çözümlerin Değerleri 78
Şekil 6.1. Izgara Tabanlı Haritalar: (a) Harita 1 (b) Harita 2 (c) Harita 3....79
Şekil 6.2. Algoritmaların Elde Ettiği Optimal Yollar: (a) Harita 1 (b) Harita 2 (c) Harita 3 80
Şekil 7.1. Yol Planlama Adımları....84
Şekil 7.2. Bresenham Algoritması ile Çizgi Çizme 85
Şekil 7.3. Pencere Yöntemi ile Bağlantı Noktası Belirleme Akış Diyagramı....86
Şekil 7.4. Pencere Yöntemi ile Bağlantı Noktası Belirleme 86
Şekil 7.5. PSO ile Parçacıkların Dağıtılması 87
Şekil 7.6. Öklid Mesafe Dönüşümü Örneği: (a) Örnek Harita, (b) EDT Sonucu 88
Şekil 8.1. Geliştirilen Yol Planla Algoritmasının Akış Diyagramı....92
Şekil 8.2. Geliştirilen Algoritma ile Planlanan Yol....93
Şekil 8.3. 64x64 Birim Harita için Öklid Mesafe Dönüşümü....93

TABLOLAR DİZİNİ

Tablo 4.1. A* Algoritması Yol Bulma Problemi Örneği için Sezgisel Değerler 33
Tablo 6.1. Simülasyon Sonucu Elde Edilen Sonuçlar (Ort: Ortalama, Std: Standart Sapma) 81
Tablo 6.2. Literatürdeki Çalışmada Elde Edilen Sonuçlar (Ort: Ortalama, Std: Standart Sapma) 82
Tablo 8.1. Yol Planlama Algoritmalarının Simülasyon Sonucları 90

SİMGELER VE KISALTMALAR DİZİNİ

C :Maliyet fonksiyonu
c_1 :Bilişsel katsayı
c_2 :Sosyal katsayı
d :İki nokta arası mesafe
d_{mh} :Manhattan mesafesi
d_{mc} :Mincowski mesafesi
F, f :Uygunluk değeri
k :Boltzmann sabiti
P :Parçacık pozisyonu
p_a :Guguk kuşu yumurtasının fark edilme olasılığı
T :Sıcaklık
V :Parçacık hızı
w :Eylemsizlik katsayısı
$\in$:Hata değeri
μ :Ortalama
σ :Standart sapma

Kısaltmalar

2D :Two Dimensional (İki Boyutlu)
ABC :Artifical Bee Colony (Yapay Arı Kolonisi)
ACO :Ant Colony Optimization (Karınca Koloni Optimizasyon)
AMR :Autonomous Mobile Robot (Otonom Mobil Robot)
AOA :Arithmetic Optimization Algorithm (Aritmetik Optimizasyon Algoritması)
APF :Artificial Potential Field (Yapay Potansiyel Alan)
BA :Bat Algorithm (Yarasa Algoritması)
BLX-α :Blended-α Crossover (Karışık-α Çaprazlama)
CPSO :Constricted Particle Swarm Optimization (Kısıtlanmış Parçacık Sürü Optimizasyonu)
CRO :Chemical Reaction Optimization (Kimyasal Reaksiyon Optimizasyonu)
CS :Cuckoo Search (Guguk Kuşu Arama)
DA :Dijkstra's Algorithm (Dijkstra Algoritması)
DE :Differential Evolution (Diferansiyel Evrim)
EDT :Euclidean Distance Transform (Öklid Mesafe Dönüşümü)
ENIAC :Electronic Numerical Integrator and Computer (Elektronik Sayısal Entegratör ve Bilgisayar)
FA :Firefly Algorithm (Ateşböceği Algoritması)
FPA :Flower Pollination Algorithm (Çiçek Tozlaşma Algoritması)
GA :Genetic Algorithm (Genetik Algoritma)
GWO :Grey Wolf Optimizer (Gri Kurt Optimizasyon)
HPCRO :Hybrid Particle Swarm and Chemical Reaction Optimization (Hibrit Parçacık Sürü ve Kimyasal Reaksiyon Optimizasyonu)
HS :Harmony Search (Harmoni Arama)
IFR :International Federation of Robotics (Uluslararası Robotik Federasyonu)

ISO	:International Organization for Standardization (Uluslararası Standardizasyon Örgütü)
ISR	:International Symposium on Industrial Robotics (Uluslararası Endüstriyel Robotik Sempozyumu)
ISS	:International Space Station (Uluslararası Uzay İstasyonu)
İHA	:İnsansız Hava Aracı
JIRA	:Japan Industrial Robot Association (Japon Endüstriyel Robot Derneği)
LIDAR	:Laser Detection and Ranging (Lazer Tabanlı Alan Tarayıcı)
NASA	:National Aeronautics and Space Administration (Ulusal Havacılık ve Uzay Dairesi)
PF	:Potential Field (Potansiyel Alan)
PRM	:Probabilistic Road Map (Olasılıksal Yol Haritası)
PSO	:Particle Swarm Optimization (Parçacık Sürü Optimizasyonu)
PUMA	:Programmable Universal Manipulator Arm (Programlanabilir Evrensel Manipülatör Kol)
RAS	:Robotics and Automation Society (Robotik ve Otomasyon Birliği)
RIA	:Robotic Industries Association (Robot Endüstrileri Derneği)
ROS	:Robot Operating System (Robot İşletim Sistemi)
RRT	:Rapidly-Exploring Random Tree (Hızlı Keşfeden Rastgele Ağaç)
SA	:Simulated Annealing (Benzetimli Tavlama)
SCA	:Sine Cosine Algorithm (Sinüs Kosinüs Algoritması)
SCARA	:Selective Compliance Assembly Robot Arm (Seçici Uyumlu Montaj Robot Kolu)
SFS	:Stochastic Fractal Search (Stokastik Fraktal Arama)
SI	:Système international d'unités (Uluslararası Birimler Sistemi)
SSRMS	:Space Station Remote Manipulator System (Uzay İstasyonu Uzaktan Manipülasyon Sistemi)
TS	:Tabu Search (Tabu Arama)
TÜBİTAK	:Türkiye Bilimsel ve Teknolojik Araştırma Kurumu

OTONOM MOBİL ROBOTLAR İÇİN METASEZGİSEL YÖNTEMLER KULLANILARAK YOL PLANLAMA ALGORİTMASININ GELİŞTİRİLMESİ

ÖZET

Mobil robotlar için yol planlama, belirli bir hedefe veya göreve ulaşmak amacıyla engellere çarpmadan en uygun rotayı belirleme sürecidir. Bu süreç mobil robotun bir başlangıç noktasından hedef noktaya en az maliyetle ulaşmasını amaçlayan bir optimizasyon problemidir. Yol planlama, metasezgisel algoritmalar kullanılarak gerçekleştirilebilir.

Metasezgisel algoritmalar popülasyon tabanlı ve tek çözüm tabanlı olmak üzere ikiye ayrılır. Popülasyon tabanlı algoritmalar kendi içinde sürü tabanlı, fizik tabanlı, evrimsel tabanlı ve insan tabanlı olmak üzere dört farklı alt kategoriye ayrılır. Bu tez kapsamında global yol planlama için üç farklı metasezgisel algoritma kullanılacaktır. Bu metasezgisel algoritmaların performanslarının karşılaştırılması için ızgara tabanlı farklı senaryolar içeren haritalar oluşturulmuştur. Bu haritalar robot çalışmaya başlamadan önce robot tarafından bilinen, haritalanmış (mapped) statik engeller içermektedir. Bu haritalar kullanılarak farklı yol planlama problemlerinde algoritmaların nasıl bir sonuç verdiği gözlemlenmiştir. Gözlemlerin sonucunda, bu algoritmaların performansları oluşturulan farklı haritalarda test edilmiş ve karşılaştırmaları yapılmıştır.

Metasezgisel algoritmalar kullanılarak global yol planlama yapıldıktan sonra ortama robot tarafından daha önce bilinmeyen haritalanmamış (unmapped) statik engeller eklenmiştir. Robot bu engellerle karşılaştığında geliştirilen yerel yol planlama algoritmasını çalıştırarak tekrar rotasına dönmektedir. Geliştirilen yerel yol planlama algoritması Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) temellidir. Robot haritalanmamış bir engelle karşılaştığında geliştirilen pencere yöntemi ile rotasına yeniden bağlanacağı noktayı belirler. Daha sonra PSO algoritması belirli bir bölge içerisine rastgele parçacıklar yerleştirir. Bu parçacıkların maliyet değeri her iterasyonda güncellenir. Robot, algoritma sonunda bulunan en iyi parçacığı kullanarak engelden kaçınır ve rotasına geri döner. Robot rotasında ilerler ve tekrar bir engel ile karşılaşırsa aynı adımları tekrarlayarak hedefine ulaşmaya çalışır.

Anahtar Kelimeler: Metasezgisel Algoritmalar, Optimizasyon, Otonom Mobil Robotlar, Yol Planlama.

DEVELOPMENT OF PATH PLANNING ALGORITHM USING METAHEURISTIC METHODS FOR AUTONOMOUS MOBILE ROBOTS

ABSTRACT

Path planning for mobile robots is the process of determining the optimal route to reach a specific goal or task without hitting any obstacles. This process is an optimization problem that aims to ensure that the mobile robot reaches the target point from a starting point at the least cost. Path planning can be performed using metaheuristic algorithms.
Metaheuristic algorithms are classified into two categories: population-based and single-solution-based. Population-based algorithms are divided into four different subcategories: swarm-based, physic-based, evolutionary-based and human-based. Within the scope of this thesis, three different metaheuristic algorithms will be used for global path planning. Various grid-based maps with specialized scenarios were created to compare the performance of these metaheuristic algorithms. These maps contain mapped static obstacles that are known before the robot starts operating. The results of algorithms in different path planning problems have been observed using these maps. As a result of the observations, the performances of these algorithms were tested and compared on different maps created.
After performing global path planning using metaheuristic algorithms, unmapped static obstacles that the robot did not know before starting work were added to the environment. When the robot encounters these obstacles, it returns to its route by running the developed local path planning algorithm. The developed local path planning algorithm is based on Particle Swarm Optimization (PSO). When the robot encounters an unmapped obstacle, it determines the point to reconnect to its route with the developed window method. The PSO algorithm then places random particles within a specific region. The cost value of these particles is updated at each iteration. The robot avoids the obstacle using the best particle found at the end of the algorithm and returns to its route. The robot proceeds along its route, and if it encounters another obstacle, it repeats the same steps to reach its destination.

Keywords: Metaheuristic Algorithms, Optimization, Autonomous Mobile Robots, Path Planning.

1. GİRİŞ

Endüstri 4.0 ile birlikte otonom mobil robotların endüstride kullanımı önem kazanmıştır. Otonom mobil robotlar yük taşıma amacıyla fabrikalarda, depolarda ve ofis ortamlarında tercih edilmeye başlanmıştır (Eyüpoğlu, 2022). Güvenlik ve çevresel faktörler göz önünde bulundurulduğunda mobil robotların tercih edilmesinin önemi artmaktadır. Son yıllarda mobil robotlar, önemli insansız görevleri yerine getirmek için askeri, endüstri ve güvenlik gibi farklı alanlarda başarıyla uygulanmıştır (Zexuan ve diğ., 2015).

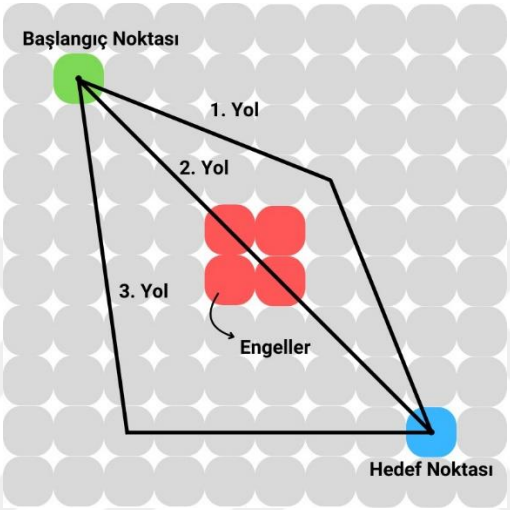
Otonom bir mobil robotun navigasyonu, başlangıç noktasından hedef noktasına engellere çarpmadan ulaşma süreci olarak tanımlanabilir. Bu süreç temel olarak dört bileşenden oluşur (Garip, 2018):

- Algılama: Mobil robot üzerindeki sensörler aracılığı ile çevre bilgisini toplar.
- Konumlandırma: Mobil robot çalışma uzayındaki konumu tespit eder.
- Yol planlama: Mobil robotun hedef konumuna ulaşması için engellere çarpmadan izleyeceği yolu belirler.
- Hareket kontrol: Mobil robot planlanan yolu gerçekleştirmek için hareketini düzenler.

Otonom mobil robotların navigasyon süreci için gerekli olan dört bileşenden biri de yol planlamadır. 1960'lı yılların ortalarından itibaren yol planlama birçok bilim insanının ilgisini çekmiştir. Mobil robotlarda iyi bir yol planlama yapılması sadece zamandan tasarruf kazandırmaz ayrıca mobil robotun kullanım ömrünü arttırır ve sermaye yatırımlarını azaltır (Zhang ve diğ, 2018). Mobil robotlar için yol planlama, belirli bir hedefe veya göreve ulaşmak amacıyla engellere çarpmadan en uygun rotayı belirleme sürecidir. Bu süreç mobil robotun bir başlangıç noktasından hedef noktaya en az maliyetle ulaşmasını amaçlayan bir optimizasyon problemidir. Yol planlama, karmaşık iç mekan ortamlarında mobil bir robotun otonom bir şekilde gezinebilmesi ve keşif yapabilmesi için çözülmesi gereken en temel problemlerden biridir (Contreras-Cruz ve diğ., 2015).

Yukarıda yapılan yol planlama tanımından anlaşılacağı üzere yol planlama problemi temel olarak hedefe ulaşmak için engellere çarpmadan en kısa mesafeye sahip yolu bulmayı amaçlar. Bu kriterlere ek olarak algoritmanın çalışma süresi, robotun hedefe ulaşma süresi, robotun harcadığı enerji gibi kriterler de eklenebilir. Şekil 1.1'de üç farklı alternatif yol içeren bir yol planlama örneği gösterilmiştir. 1. Yol'u inceleyecek olursak

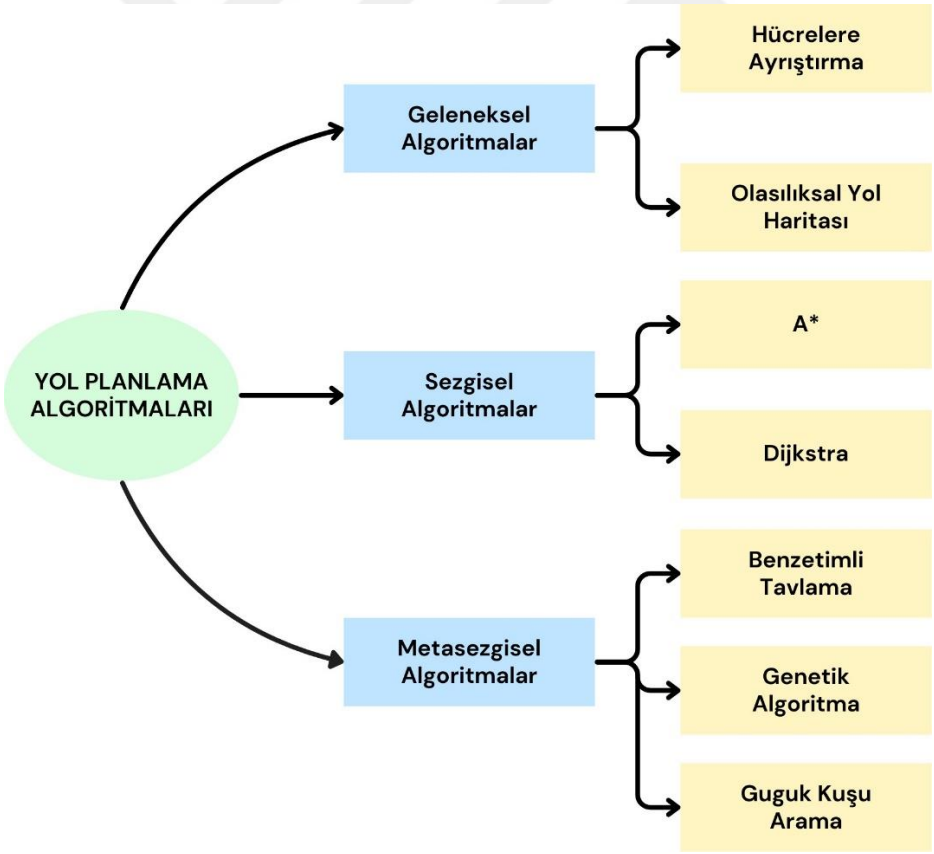
engele çarpma durumunun olmadığı ve yol uzunluğu açısından uygun bir yol olduğu görülmektedir. 2. Yol aralarındaki en kısa mesafeye sahip olan yol olmasına rağmen engele çarpma durumu olduğu için bu yolun uygun olmadığını söyleyebiliriz. 3. Yol'da ise engele çarpma durumu yoktur fakat diğer yollara kıyasla en uzun mesafeye sahiptir. 1.Yol'la kıyaslandığında engelle arasında daha fazla mesafe bulunduğu için daha emniyetlidir. Buradan anlaşılacağı üzere engele çarpma olmaması bize uygun bir yol verir fakat yol planlama probleminin amacı en uygun (optimal) yolu bulmaya çalışmaktır.



Şekil 1.1. Yol Planlama Örneği

Genel olarak yol planlama, çevreyle ilgili tüm bilgilerin erişilebilir olup olmamasına göre iki kategoriye ayrılabilir. Global yol planlamada, çevreye ilişkin tüm bilgiler robot çalışmaya başlamadan önce robot tarafından bilinir. Yerel yol planlamada ise çevreye ilişkin hemen hemen tüm bilgiler robot tarafından çalışmaya başlamadan önce bilinmemektedir (Hoang ve diğ., 2013). Yol planlama yapılırken, çevrede genellikle iki tür engel bulunmaktadır: Statik engeller, sabit konumda olan ve hareket etmeyen engellerdir. Statik engellere duvarlar, bariyerler, sütunlar, raflar, konveyörler, tezgahlar örnek olarak verilebilir. Dinamik engeller, hareket edebilen ve beklenmedik şekilde konumları değişebilen engellerdir. Dinamik engellere insanlar, ortamdaki diğer mobil robotlar, araçlar örnek olarak verilebilir. Bu tez çalışmasında ortamdaki engeller statik olarak kabul edilmiştir. Bu engeller, robot tarafından daha önceden bilinen haritalanmış (mapped) ve robotun hareketi esnasında sensörleri aracılığıyla algılayacağı haritalanmamış (unmapped) engellerdir.

Yol planlama algoritmalarını Şekil 1.2'deki gibi geleneksel algoritmalar, sezgisel algoritmalar ve metasezgisel algoritmalar olmak üzere üç sınıfa ayırabiliriz (Xu ve diğ., 2023). Son yıllarda mobil robot navigasyonunda kritik bir önem taşıyan yol planlama üzerine çalışmalar yapılmakta ve bu problemi çözmek için metasezgisel algoritmalar kullanılmaktadır. Metasezgisel algoritmalar, insan zekasından veya doğadan ilham alır ve geniş anlamda bir stokastik (değişken, rastlantısal) optimizasyon algoritması olarak sınıflandırılabilir. Bu tez çalışması kapsamında global yol planlama için üç farklı metasezgisel algoritma kullanılacaktır. Bu algoritmalar; tek çözüm tabanlı olan Benzetimli Tavlama (Simulated Annealing, SA), popülasyon ve evrimsel tabanlı algoritma olan Genetik Algoritma (Genetic Algorithm, GA), popülasyon ve sürü tabanlı olan Guguk Kuşu Arama (Cuckoo Search, CS) algoritmalarıdır. Yerel yol planlama için ise yine metasezgisel bir algoritma olan Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) algoritması kullanılacaktır.



Şekil 1.2. Yol Planlama Algoritmaları

Tez çalışmasının ikinci bölümünde robotlar hakkında genel bilgiler verilmiş ve mobil robotlar için yol planlamada kullanılan harita türleri açıklanmıştır.

Üçüncü bölümde literatür taraması yapılarak yol planlama üzerine yapılan çalışmalar aktarılmıştır.

Dördüncü bölümde tez kapsamında kullanılan yol planlama algoritmaları detaylı olarak açıklanmıştır.

Beşinci bölümde tez kapsamında kullanılan metasezgisel yol planlama algoritmaları aktarılmıştır.

Altıncı bölümde geliştirilen global yol planlama algoritmaları açıklanmış ve farklı harita türlerinde simüle edilerek karşılaştırılmıştır.

Yedinci bölümde tez kapsamında geliştirilen yerel yol planlama yöntemi detaylı olarak açıklanmıştır.

Sekizinci ve son bölümde, simülasyon sonucu elde edilen sonuçlar sunulmuş ve çalışmanın geliştirilmesine ilişkin önerilerde bulunulmuştur.

2. GENEL BİLGİLER

Bu bölümde robotlar ve yol planlama algoritmaları hakkında genel bilgiler verilecektir. Robotların sınıflandırılmasından bahsedildikten sonra yol planlama, global ve yerel yol planlama olmak üzere iki kategoride incelenecektir. Son olarak yol planlamada kullanılan engel çeşitleri ve harita türleri tanımlanacaktır.

Robot teknolojisi 20. yüzyılın ikinci yarısından itibaren gelişmeye başlamış ve günümüzde halen hızlı bir şekilde gelişimini sürdürmektedir. Sensörlerin, mikroişlemcilerin, kontrol sistemlerinin, eyleyicilerin gelişmesi ve Endüstri 4.0 ile birlikle endüstriyel otomasyon ve akıllı sistemlere duyulan ilgilinin artması robot teknolojisinin kullanımı artırmış ve gelişimine ivme kazandırmıştır. 1921 senesinde robot kelimesini ilk olarak kullanan kişi Çek yazar Karel Čapek'tır. Čapek "Rossum's Universal Robots" adlı bilimkurgu türündeki tiyatro oyununda bu kelimeye yer vermiştir. Robot kelimesi bazı Slav dillerinde köle işi, angarya ağır iş gibi anlamlara gelen robota kelimesinden türetilmiştir (Bayraktaroğlu, 2018). Bu tiyatro oyununda insansı robotlar olarak tanımladığımız robotlar başlangıçta insanoğluna hizmet ederken, oyunun sonunda insanoğluna isyan ederler ve insan ırkının yok olmasına sebep olurlar.

20. yüzyılda gelişimini sürdüren robotlar bilim kurgu edebiyatının ilgisini çekmiştir ve 1942 yılında Rus bilimkurgu yazarı Isaac Asimov "Runaround" adlı kısa öyküsünde robot yasalarından bahsetmiştir. Asimov'un önerdiği bu Üç Robot Kanun'u ileride insan robot etkileşimindeki etik kurallara ve çıkarılacak yasalara ışık tutacaktır. Asimov'un "Ben, Robot" adlı romanından alınan Üç Robot Kanun'u şu şekildedir (Asimov, 2022):

- Birinci Kanun: Robotlar, insanlara zarar veremez ya da eylemsiz kalarak onlara zarar gelmesine göz yumamaz.
- İkinci Kanun: Robotlar, Birinci Kanun'la çelişmediği sürece insanlar tarafından verilen emirlere itaat etmek zorundadır.
- Üçüncü Kanun: Robotlar, Birinci ya da İkinci Kanun'la çelişmediği sürece kendi varlıklarını korumak zorundadır.

Robot kelimesinin tanımlarına bakacak olursak Oxford İngilizce Sözlüğü'ne göre robotun tanımı: "Karmaşık bir dizi eylemi otomatik olarak gerçekleştirebilen, özellikle de

bilgisayar tarafından programlanabilen bir makinedir.” Bu tanımdaki eylemleri otomatik olarak gerçekleştirme işlemini birçok otomat adı verilen cihaz da yapabilmektedir buna ilave olarak eklenen karmaşık eylemleri gerçekleştirme işlemi açık bir tanım değildir ve görecelidir. Son kısımda bahsedilen “bilgisayar tarafından programlanabilen” tanımı birçok mekanik makineyi elese dahi günümüzde bilgisayar ile çalışan birçok cihaz bulunmaktadır. Bu tanımda eksik olan ve robotlar için vazgeçilmez olan unsur sensörlerdir. Bir robot sensörleri vasıtasıyla çevresini algılar ve buna göre aksiyon alır.

Amerika Robot Enstitüsü’ne (Robotics Institute of America) göre robot tanımı: Robot, çeşitli görevlerin yerine getirilmesi için malzemeyi, parçaları, aletleri veya özel cihazları değişken programlanmış hareketler yoluyla hareket ettirmek üzere tasarlanmış, yeniden programlanabilir, çok işlevli bir manipülatördür.

Bayraktaroğlu’na (2018) göre robot, mekanik hareket sistemi, algılayıcı sistemi ve bir otomatik kontrol sisteminden oluşan çok serbestlik dereceli ve çok işlevli cihazlardır. Bayraktaroğlu robot çalışmalarının, mekanik, elektrik-elektronik, otomatik kontrol, matematik ve bilgisayar bilimleri gibi temel alanların üzerine inşa edildiğini söylemektedir.

Bingül ve Küçük (2019) robotu birçok farklı görevi yapmak üzere programlanabilen bir makine olarak tanımlamaktadır ve sıradan bir makineden farklı olarak robotların birçok kez programlanabilen ve bünyelerinde mekanik sistemlerin ve eyleyicilerin yanında bir de kapalı döngü denetleyici içerdiğini söylemektedirler. Genel anlamda robotu, kapalı döngülü bilgisayar kontrollü bir endüstriyel manipülatör olarak tanımlamaktadırlar.

Uluslararası Standardizasyon Örgütü’nün (International Organization for Standardization, ISO) tanımladığı ve Uluslararası Robotik Federasyonu’nun (International Federation of Robotics, IFR) kabul ettiği robot tanımı ise: Bir robot, hareket, manipülasyon veya konumlandırma gerçekleştirmek için belirli bir özerkliğe sahip olan, programlanmış, çalıştırılan bir mekanizmadır. ISO’ya göre endüstriyel bir robot, endüstriyel ortamda otomasyon uygulamalarında kullanılmak üzere yerine sabitlenebilen veya mobil bir platforma sabitlenebilen, üç veya daha fazla eksende programlanabilen, otomatik olarak kontrol edilen, yeniden programlanabilen, çok amaçlı

bir manipülatördür. ISO mobil robotu kendi kontrolü altında seyahat edebilen bir robot olarak tanımlamıştır.

IFR'ın sitesinden edinilen bilgilere (URL-7) ve Bayraktaroğlu'na (2018) göre robotların tarihçesi:

- 1946 İlk modern bilgisayar ENIAC (Electronic Numerical Integrator and Computer), Pensilvanya Üniversitesi, J. Presper Eckert ve John Mauchly.
- 1949 İlk mekanik tele-manipülatör, Raymond Goertz.
- 1954 İlk elektrik tahrikli tele-manipülatör, Argonne National Laboratory.
- 1954 George C. Devol tarafından 1950'lerde icat edilen Unimate robotuna patent başvurusu yapıldı. Patent 1961 yılında kabul edilmiştir.
- 1956 İlk robot üretici firma Unimation Inc. (Universal Automation) George C. Devol ve Joseph F. Engelberger tarafından kuruldu.
- 1959 George Devol ve Joseph Engelberger tarafından ilk endüstriyel robotun geliştirilmesi: 2 ton ağırlığındaydı ve hidrolik aktüatörler kullanıyordu. Bir inçin 1/10,000 hassasiyetindeydi.
- 1961 PUMA (Programmable Universal Manipulator Arm), Unimation Inc.
- 1961 Unimation, General Motors (GM) fabrikasına ilk endüstriyel robotu kurdu: Dünyanın ilk endüstriyel robotu, Trenton, GM Ternstedt fabrikasındaki bir üretim hattında kullanılan, kapı ve pencere kolları, vites topuzları, aydınlatma armatürleri ve otomotiv iç mekanları için diğer donanımları üreten robottu.
- 1962 American Machine and Foundry'nin (AMF) ilk silindirik robotu Versatran: 6 adet Versatran robotu AMF tarafından ABD'nin Canton kentindeki Ford fabrikasına kuruldu. Versatran adını çok yönlü aktarım anlamına gelen "versatile transfer" kelimesinden aldı.
- 1963 Robot için ilk yapay görü sistemi
- 1967 Avrupa'nın ilk endüstriyel robotu: Avrupa'nın ilk endüstriyel robotu Unimate, İsveç'te kuruldu.
- 1968 Kawasaki Heavy Industries firmasının Unimation Inc. ile teknoloji transferi içeren anlaşması.
- 1968 Marvin Minsky tarafından ahtapota benzeyen Dokunaç Kolu isimli robotu geliştirildi.

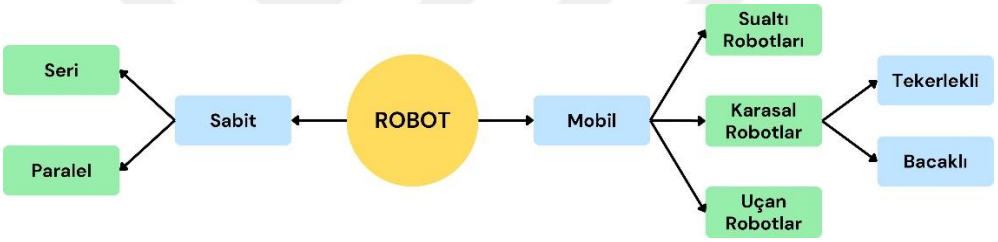
- 1969 GM, Lordstown montaj fabrikasına ilk nokta kaynak robotlarını kurdu: Geleneksel tesisler kaynak işlerinin %20 ile %40'ı otomatikken, Unimation robotları bu oranı %90'lara çıkardı.
- 1969 Trallfa Norveç'te ilk ticari boyama robotunu sundu.
- 1969 İlk elektrik tahrikli ve 6 serbestlik derecelik robot: Stanford Arm, Stanford Üniversitesi, Victor Scheinman.
- 1969 Hitachi (Japonya), plan çizimlerindeki nesneleri birleştiren dünyanın ilk görüntü tabanlı tam otomatik akıllı robotunu geliştirdi
- 1970 İlk Uluslararası Endüstriyel Robotik Sempozyumu (ISIR/ISR) Chicago, ABD.
- 1971 Daimler-Benz otomotiv şirketi, Sindelfingen'de hidrolik tahrikli robotlara sahip ilk üretim hattını kurdu: KUKA, Daimler-Benz şirketi için 1971'de Avrupa'nın ilk robotlu kaynak aktarma hattını inşa etti.
- 1971 Japon Endüstriyel Robot Derneği (JIRA/ sonrasında JARA) kuruldu: Bu ilk ulusal robot derneğiydi.
- 1972 Mobil robot rehberliğine yönelik robot görüşüne sahip Shakey Stanford Araştırma Enstitüsü'nde gösterildi: Mobil robot rehberliğine yönelik robot görüşü SRI'da gösterilmiştir. Robot Shakey, 1966 ile 1972 yılları arasında SRI Yapay Zeka Merkezi'nde geliştirildi. Robot görüşüyle çevresinde akıllıca gezinebilen ilk robottu ve günümüzün Otonom Mobil Robot'larının (AMR) temellerinin atılmasına yardımcı oldu. Aynı zamanda yapay zekayı kullanan ilk robottu.
- 1973 İlk robot programlama dili: Wave, Stanford Üniversitesi.
- 1973 İlk mikroişlemci kontrollü, elektrik tahrikli endüstriyel robot: Asea/ABB IRB-6.
- 1973 Scheinemann, Vicarm Inc., ABD'de Vicarm/Stanford kolunun üretimine başladı.
- 1973 Hitachi, beton kazık ve direk endüstrisi için otomatik cıvatalama robotunu geliştirdi.
- 1974 Robot Endüstrileri Derneği'nin (RIA) kuruluşu.
- 1974 İlk mini bilgisayar kontrollü endüstriyel robot pazara çıktı.
- 1975 Hitachi, ilk sensör tabanlı ark kaynağı robotu Mr. AROS'u geliştirdi.
- 1976 Robot kollar Viking 1 ve 2 uzay araçlarında kullanıldı.

- 1978 Japonya'nın Yamanashi Üniversitesi'nden Hiroshi Makino, SCARA (Selective Compliance Assembly Robot Arm) robotunu geliştirdi: Düşük maliyetli ve küçük parça montajına uygun bu tasarım elektronik imalatının gelişimine önemli ölçüde katkı sağlamıştır.
- 1980'lerde yüksek yük taşıma kapasitesine sahip ama çalışma uzayları seri robotlardan daha küçük olan paralel robotların geliştirilmesi.
- 1981 Richard Paul tarafından robotlar üzerine yazılan ilk mesleki kitap "Robot Manipulators: Mathematics, Programming and Control" yayınlandı.
- 1982 Robot araştırmalarını konu alan ilk bilimsel dergi: International Journal of Robotics Research.
- 1982 IBM, robot bilimi için bir programlama dili olan AML'yi geliştirdi.
- 1987 Uluslararası Robotik Federasyonu'nun (IFR) kuruluşu.
- 1988 IEEE Robotik ve Otomasyon Birliği'nin (RAS) kuruluşu.
- 1988 Hastanede ilk AMR kullanıldı.
- 1997 Sojourner mobil robotunun Mars gezegenine inişi, NASA JPL.
- 1997 İlk robotlar arası futbol turnuvası RoboCup-97 Japonya'da düzenlendi.
- 1999 Intuitive Surgical'ın ticarileştirilmiş Da Vinci cerrahi robotu: Bir doktor tarafından bir konsol aracılığıyla uzaktan kontrol edilen, cerrahi aletler ve kameralarla donatılmış dört kolu olan bir robottur.
- 2001 New York'ta (ABD) bulunan cerrahların, internet üzerinden kontrol ettikleri uzak robotlar aracılığıyla Strasbourg'daki (Fransa) bir hastayı ameliyatı.
- 2001 Uluslararası Uzay İstasyonu'nun (ISS) devam eden inşasında kullanılmak üzere Uzay İstasyonu Uzaktan Manipülasyon Sistemi'nin (SSRMS) Endeavour uzay aracı ile fırlatılması.
- 2004 Sprit ve Opportunity mobil robotlarının Mars gezegenine inişi, NASA JPL.
- 2005 İki elle senkronize manipülasyon için ilk çift kollu ticari robot: Yaskawa Motoman.
- 2006 Ağırlık / Yük kapasitesi oranı "1" olan 7 serbestlik dereceli KUKA robot kolu.
- 2008 Seri üretilen ilk insansı robot: NAO, Aldebaran
- 2009 Yaskawa Motoman, Japonya, 8 robota kadar senkronizasyon sağlayan kontrol sistemini tanıttı.

- 2009 ROS'un (Robot Operating System) yayınlanması.
- 2010'lar Esnek ve işbirlikçi robotların gelişimi.
- 2011 Robonaut (R2B), Uluslararası Uzay İstasyonuna fırlatıldı. Uzaya çıkan ilk insansı robot oldu.
- 2012 ISS'de ilk insansı robot: Robonaut
- 2012 Curiosity mobil robotun Mars gezegenine inişi: NASA JPL.
- 2016 Endüstri 4.0
- 2017 İlk kez çift kollu bir robotun müzik orkestrasını yönetmesi: ABB YuMi.

2.1. Robotların Sınıflandırılması

Robotlar temel olarak Şekil 2.1'deki gibi sabit ve mobil robotlar olmak üzere iki sınıfa ayrılabilir.



Şekil 2.1. Robotların Sınıflandırılması

Sabit robotlar kendi içinde seri ve paralel olmak üzere ikiye ayrılır. Sabit robotlar genellikle iyi bilinen ortam koşullarında çalışırlar. Sabit robot deyince genellikle akla fabrikalarda çalışan endüstriyel robot kollar gelir. Bu robotlar fabrikalarda kaynak veya lehim yapmak amacıyla, parça boyama amacıyla özellikle otomotiv sektöründe kullanılır. Ayrıca teknolojinin gelişmesi sensörlerin hassasiyetinin artmasına neden oldu. Çok hassas sensörlere sahip olan sabit robotik manipülatörler yüksek hassasiyet gerektiren cerrahi operasyonlarda kullanılmaya ve bu alanda geliştirilmeye başladı. Şekil 2.2'de sabit robot örneği verilmiştir. Bu robot endüstriyel amaçlı kullanılan bir boya robotudur.

Mobil robotları ise çalışma ortamlarına göre sualtı robotları, karasal robotlar ve uçan robotlar olmak üzere üçe ayırabiliriz. Mobil robotlar genellikle bilinmeyen veya az bilinen ortamlarda çalışırlar. Bu yüzden mobil robotlardan görev sırasında karşılaştıkları statik ve dinamik engelleri aşmaları beklenir.



Şekil 2.2. Endüstriyel Robot Kol (URL-1)

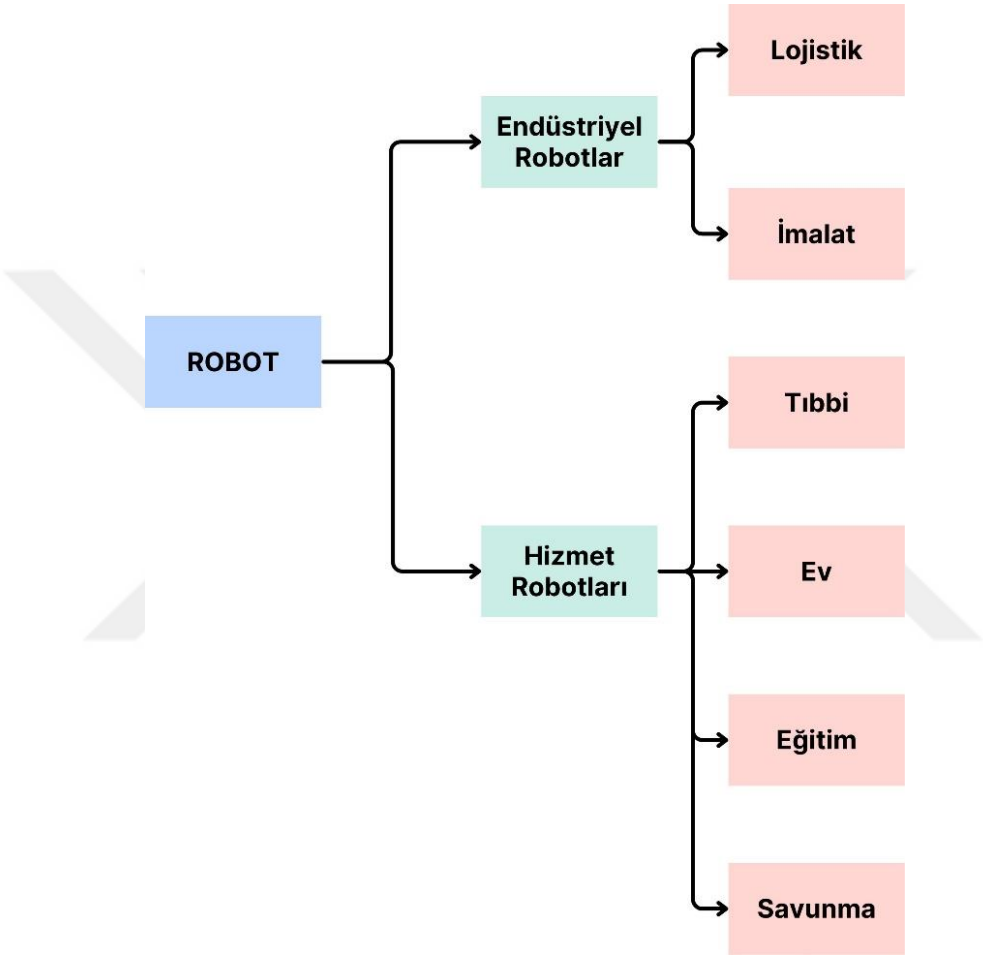
Bu engeller duvarlar, kolonlar, masa, dolap, tezgah gibi statik engeller olabileceği gibi insan, araç, kedi, köpek ve diğer mobil robotlar gibi dinamik engellerde olabilir. Mobil robotlara, fabrikalarda kullanılan yük taşıma robotları, robot süpürgeler ve otonom sürücüsüz araçlar örnek verilebilir. Şekil 2.3'te 750 kg'a kadar yük taşıyabilen bir mobil robot örneği görülmektedir.



Şekil 2.3. Otonom Mobil Robot (URL-2)

Robotlar gerçekleştirdikleri görevlere ve uygulama alanlarının amacına göre sınıflandırabilir (Ben-Ari ve Mondada, 2018). Bu sınıflandırmada robotlar endüstriyel ve hizmet robotları olmak üzere Şekil 2.4'teki gibi ikiye ayrılır.

Endüstriyel robotlar çalışma ortamlarının iyi bilinmesinden dolayı ilk robotlardı. Hizmet robotları insanların işlerini kolaylaştırmak ve görevlerinde onlara yardımcı olmaktadır. Ameliyat robotları tıp alanında, robot süpürgeler ev işlerinde, Şekil 2.5'te bir örneği gösterilen eğitim robotu basit blok diyagram kodlama yapısı sayesinde eğitimde, insansız hava araçları gibi robotlar savunma sektöründe insanlara hizmet etmektedir.



Şekil 2.4. Uygulama Alanlarına Göre Robotların Sınıflandırılması

2.1.1. Sabit Robotlar

Sabit robotlar temel olarak seri ve paralel robotlar olmak üzere iki gruba ayrılır. Şekil 2.6'da seri ve paralel robot örnekleri gösterilmektedir. Robotlar bir dizi eklem (joint) ve bu eklemlerin birbirleri ile bağlantısını sağlayan bağlardan (link) oluşur. Seri robotların eklemleri birbirlerine seri şekilde bağlandığı için hareketini sıralı ve ardışık bir biçimde gerçekleştirir.



Şekil 2.5. Eğitim Robotu (URL-3)

Örneğin insan kolu seri robota benzer bir yapıya sahiptir. Her eklem bir öncekini takip eder ve hareket sıralı bir biçimde sonuçlandırılır. Seri robotlar hassas bir şekilde hareket eder ve çok yönlü karmaşık hareketleri gerçekleştirebilir fakat paralel robotlara göre ağırlığına oranla daha az yük taşıma kapasitesine sahiptir. Endüstride kaynak, lehim, boyama, paketleme ve montaj gibi işlemleri gerçekleştirmek için sıklıkla kullanılırlar. Paralel robotlar, eklemleri birbirlerine paralel bir şekilde bağlanan kapalı zincirlerdir. Seri robotlara göre daha sağlam bir mekanik yapıya sahiptir ve yük taşıma ağırlığının mekanik yapısının toplam kütlesinin ağırlığına oranı büyüktür. Paralel robotların ileri yönlü kinematik denklemleri seri robotlara göre daha karmaşıktır.

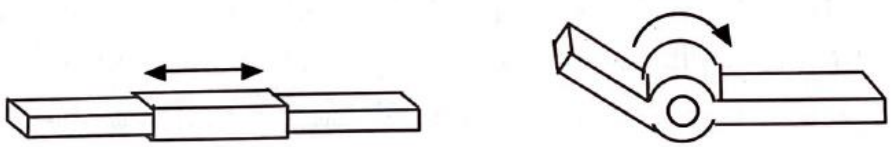


(a) (b)

Şekil 2.6. a) Kuka KR1000 Titan Seri Robotu (URL-8) b) Kuka KR Delta Paralel Robotu(URL -9)

Sabit robotlar, serbestlik derecelerine, kontrol yöntemlerine, eyleyicilerin yapısına, hassaslık ve kesinlik derecelerine ve eklem yapılarına göre çok çeşitli şekilde sınıflandırılabilir. Bunların içinde en yaygın olarak kullanılanı eklem yapılarına ve eklemlerin yerleşimine göre olan iki harfli kod sınıflandırılmasıdır (Bingül ve Küçük, 2019).

Bu tezde sabit robotlar eklem düzenleşimine göre sınıflandırılacaktır. Günümüzde genellikle 6 serbestlik dereceli robotlar tercih edilmektedir bu sınıflandırmada seri robotların ilk üç eklemi dikkate alınarak sınıflandırma yapılır. Robotların eklem yapılarını Şekil 2.7'de görüldüğü gibi döner eklem (revolute joint) ve prizmatik eklem (prismatic joint) olmak üzere ikiye ayırabiliriz.



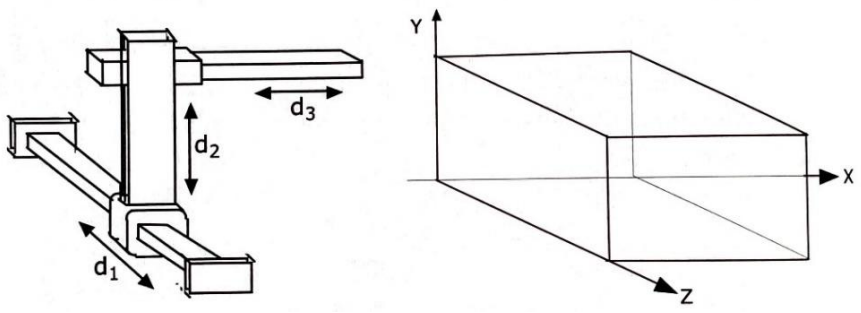
(a) (b)

Şekil 2.7. Seri Robot Kol Eklem Örnekleri: a) Prizmatik Eklem b) Döner Eklem (Bingül ve Küçük, 2019)

Döner eklem bir bağın bir eksen etrafında dönmesini sağlar. Bağ bir rotasyon ekseni boyunca dönme hareketini gerçekleştirir. Döner eklem bir robot kolda genellikle omuz, dirsek veya bilek bölgelerinde bulunur. Prizmatik eklem ise bağın düz bir hat boyunca ileri geri hareket etmesini sağlar. Döner eklem dönme hareketini gerçekleştirirken, prizmatik eklem öteleme hareketi yapar. Döner eklemler kısaca "R" ve prizmatik eklemler kısaca "P" harfi ile temsil edilecektir. Prizmatik eklemler, bir robot bağının belirli bir yönde düz çizgi boyunca hareket etmesini sağlar.

2.1.1.1. Kartezyen (PPP) Düzenleşim

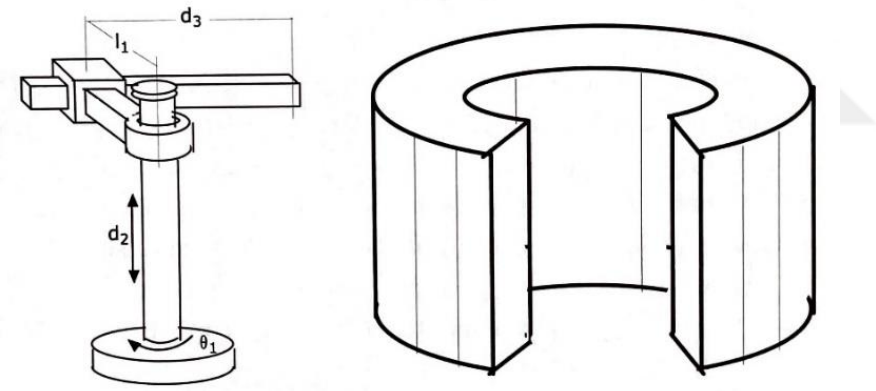
Seri robotun ilk üç eklemi prizmatik ekleme sahipse bu robota kartezyen robot denir. Bu robot endüstride yük taşıma ve montaj amacıyla kullanılır. Bu robotlar doğrusal hareket gerektiren uygulamalarda kullanılır. Kartezyen robotların kinematik denklemleri kolay olduğu için kontrolü kolaydır. Şekil 2.8'de kartezyen düzenleşime sahip robot ve çalışmayı uzayı verilmektedir.



Şekil 2.8. Kartezyen Robot ve Çalışma Uzayı (Bingül ve Küçük, 2019)

2.1.1.2. Silindirik (RPP) Düzenleşim

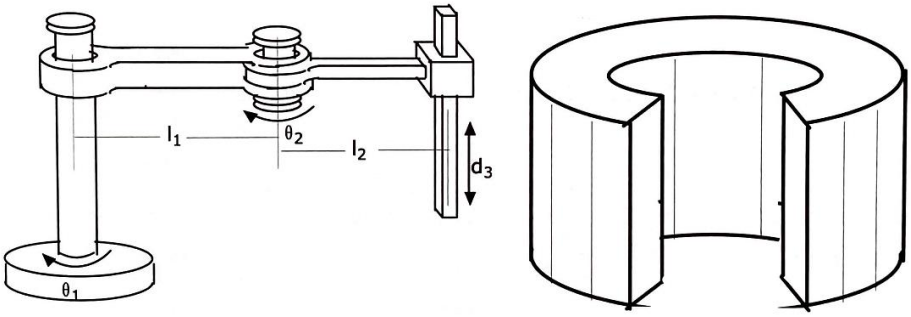
Seri robotun ilk eklemi döner, ikinci eklem birinci ekleme dik ve prizmatik, üçüncü eklem de ikinci ekleme dik ve prizmatikse bu düzenleşime sahip robota silindirik robot denir. Kinematik denklemleri kolaydır ve ilk eklem döner olduğu için uç işlevci hızlı bir şekilde hareket eder. Şekil 2.9'da silindirik düzenleşime sahip robot ve çalışması uzayı verilmektedir.



Şekil 2.9. Silindirik Robot ve Çalışma Uzayı (Bingül ve Küçük, 2019)

2.1.1.3. SCARA (RRP) Düzenleşim

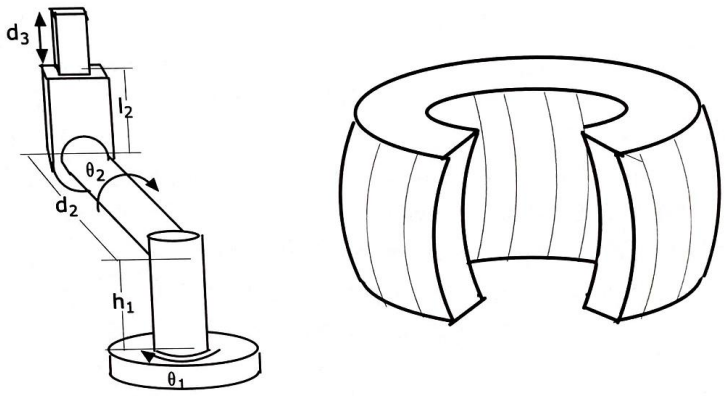
SCARA (Selective Compliance Assembly Robot Arm) robot adından da anlaşılacağı üzere endüstride montaj ve tut-bırak işlemleri için sıklıkla tercih edilmektedir. SCARA düzenleşimine sahip robotun ilk üç ekleminin tamamı birbirine tamamen paraleldir. SCARA robotlar, yatay düzlemde geniş hareket aralığına sahiptir ve yüksek hızlarda hassasiyetle hareket edebilirler. Şekil 2.10'da SCARA düzenleşimine sahip robot ve çalışma uzayı gösterilmektedir.



Şekil 2.10. SCARA Robot ve Çalışma Uzayı (Bingül ve Küçük, 2019)

2.1.1.4. Küresel (RRP) Düzenleşim

Küresel düzenleşime sahip robotun ilk iki eklemi döner ve son eklemi prizmatiktir. SCARA'dan farklı olarak eklemler birbirine paralel değildir. Bu düzenleşimde art arda gelen eklemler birbirlerine diktir. Büyük çalışma uzayına sahiptir fakat karmaşık kinematik denklemlere sahip olduğu için kontrolü zordur. Hareket kabiliyeti her noktada farklı olduğu için özellikle ana çerçeve civarında çok düşüktür. Bu yüzden eklem açılarındaki küçük bir değişim uç işlevcinin büyük yer değiştirmesine sebep olur. Şekil 2.11'de küresel düzenleşime sahip robot ve çalışma uzayı gösterilmektedir.

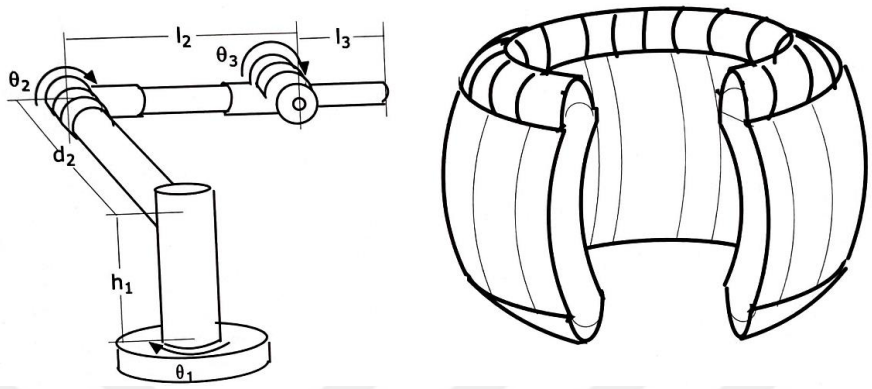


Şekil 2.11. Küresel Robot ve Çalışma Uzayı (Bingül ve Küçük, 2019)

2.1.1.5. Dönel (RRR) Düzenleşim

Seri robotun üç eklemi de döner eklem ise bu düzenleşime sahip robota dönel (articulated, anthropomorphic) robot denir. Bu robotlar büyük çalışma uzayına sahiptir, hızlı ve esnektirler, tüm eklemleri döner olduğu için hareket ettirilmesi kolaydır. Karmaşık kinematik yapıya sahip olduğu için kontrolü zordur ve çalışma uzayının her noktasına

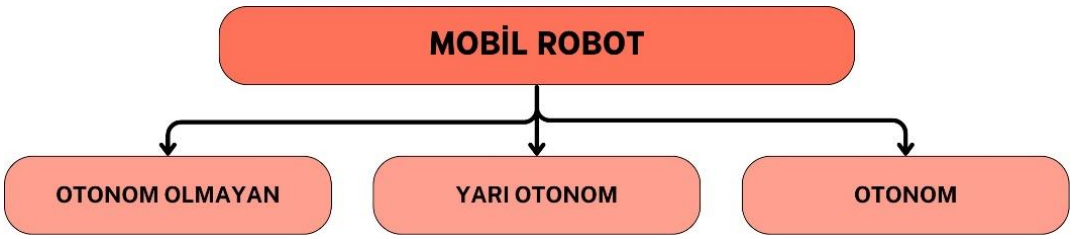
ulaşamaz. Her eklemin oluşturduğu küçük hatalar toplanarak büyük bir hataya sebep olduğu için doğruluğu düşüktür. Şekil 2.12'de dönel düzenleşime sahip robot ve çalışma uzayı gösterilmektedir.



Şekil 2.12. Dönel Robot ve Çalışma Uzayı (Bingül ve Küçük, 2019)

2.1.2. Mobil Robotlar

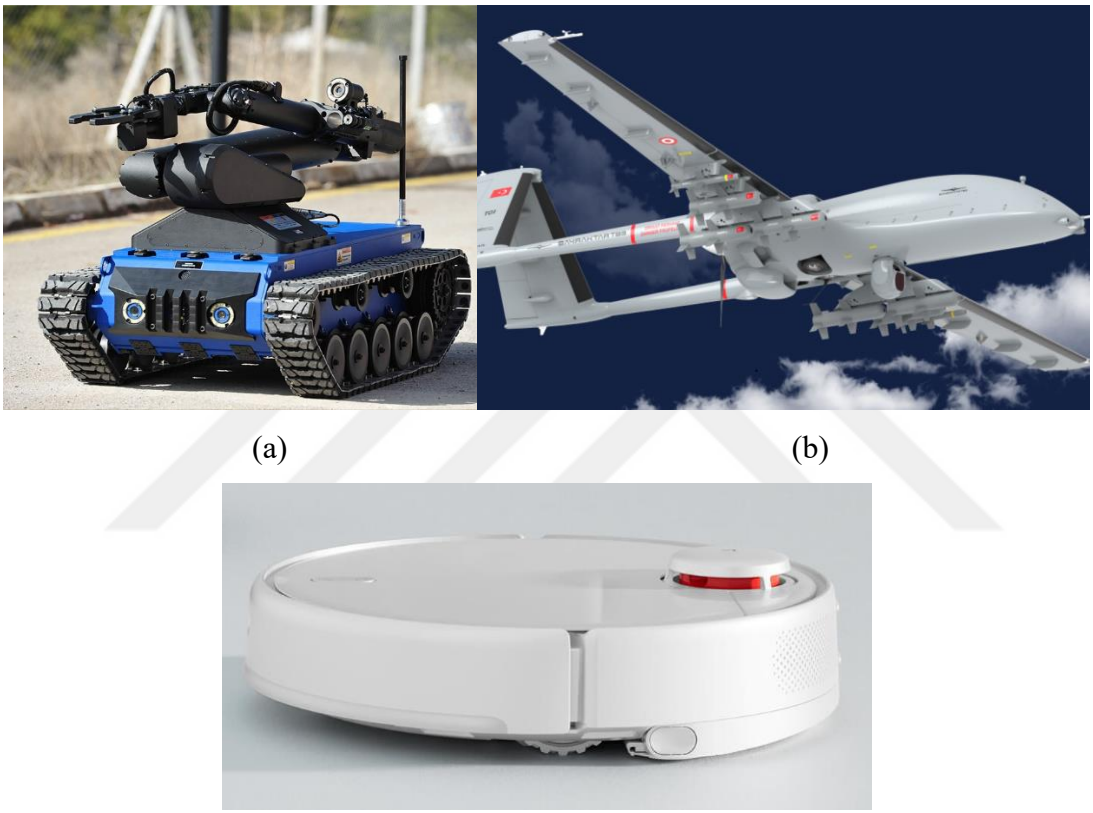
Mobil robotların en önemli özelliği, görevini gerçekleştirmek veya bir hedefe ulaşmak için çalışma ortamında rahatlıkla hareket etmesidir. Bu hareket bir operatör tarafından sağlanabileceği gibi robot üzerindeki sensörler vasıtasıyla navigasyonu gerçekleştirebilir. Mobil robotları otonomilerine göre otonom olmayan, yarı otonom ve otonom olmak üzere Şekil 2.13'teki gibi üç kategoriye ayırabiliriz.



Şekil 2.13. Mobil Robotların Sınıflandırılması

Otonom olmayan mobil robotlara Şekil 2.14a'da gösterilen bomba imha robotu örnek verilebilir. Bunun dışında hava fotoğrafçılığı, uzay ve tehlikeli bölgelerin keşfinde de otonom olmayan mobil robotlar kullanılır. Yarı otonom mobil robotlara örnek olarak insansız hava araçları (İHA) verilebilir. İHA'larda otopilot sistemi uçuşu dengelerken İHA operatörü uçuşun rotasını belirler. Şekil 2.14b'de bir İHA örneği gösterilmektedir.

Otonom mobil robotlar ise robot üzerindeki sensörler vasıtasıyla çevresini algılar. Navigasyon sistemi sayesinde konumunu tespit eder ve sahip olduğu yazılımlarla yolunu planlayıp bu yolu takip ederek görevini gerçekleştir. Bunu yaparken otonom mobil robot bir operatöre ihtiyaç duymaz. Şekil 2.14c'de gösterilen robot süpürge otonom bir mobil robottur çünkü sensörleriyle evdeki engelleri tespit edebilir, hedefine ulaşmak için yolunu planlayabilir ve bu yolu takip edebilir, sahip olduğu batarya sayesinde ortamda serbestçe dolaşabilir ve şarjı azaldığında şarj istasyonuna giderek bataryasını şarj edebilir.



(a) (b)

(c)

Şekil 2.14. Mobil Robotlar: a) ERTUĞRUL Bomba İmha Robotu (URL-4) b) Baykar TB3 İHA (URL-5) c) Xiaomi Vacuum-Mop 2 Pro (URL-6)

Mobil robotların kullanım alanları:

- Endüstriyel Otomasyon: Fabrikalarda üretim süreçlerini otomatikleştirmek ve verimliliği artırmak için mobil robotlar kullanılmaktadır. Malzeme taşıma, montaj işlemleri, paketleme, envanter yönetimi gibi birçok görevde mobil robotlar tercih edilmektedir.

- Hizmet Robotları: Restoranlarda, otellerde, havalimanlarında ve diğer hizmet sektörlerinde müşteri hizmeti veya temizlik gibi işlevler için kullanılabilirler.
- Araştırma ve Keşif: Uzay araştırmaları, derin deniz keşifleri, arama kurtarma operasyonları gibi alanlarda kullanılabilirler.
- Tarım: Tarım alanında otomatik hasat, otomatik sulama ve zirai ilaçlama gibi işlemlerde kullanılabilirler.
- Lojistik ve Dağıtım: Depo ve lojistik firmaları, depo içi taşıma işlemlerini hızlandırmak ve optimize etmek için mobil robotları kullanmaktadır. Bu robotlar, depo içinde malzemeleri taşıyabilir ve düzenleyebilirler.
- Savunma ve Güvenlik: Askeri amaçlar için mobil robotlar geliştirilmekte ve kullanılmaktadır. Bu robotlar, keşif, gözetleme, mayın temizleme gibi görevlerde kullanılmaktadır. Güvenlik operasyonlarında mobil robotlardan faydalanılmaktadır.
- Eğitim ve Araştırma: Üniversiteler ve araştırma kuruluşları, mobil robotlar üzerine araştırmalar yapmakta ve eğitim amaçlı olarak kullanmaktadır. Öğrencilerin robotik sistemlerle ilgili becerilerini geliştirmek için mobil robotlar sıkça kullanılmaktadır.

2.2. Yol Planlama

Yol planlama algoritmaları ortam bilgisine göre global yol planlama ve yerel yol planlama olmak üzere ikiye ayrılır.

2.2.1. Global Yol Planlama

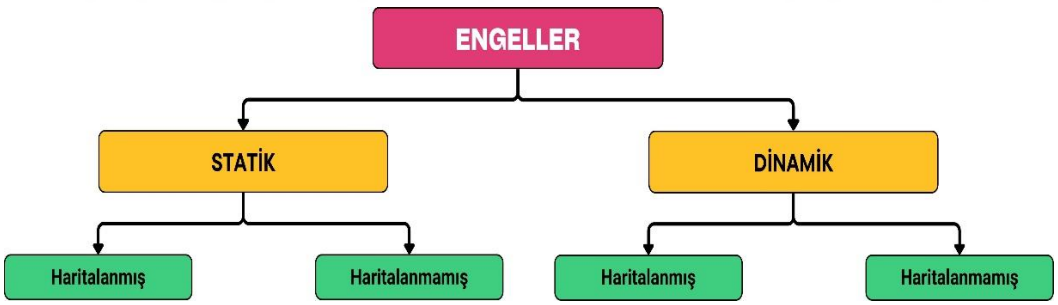
Global yol planlamada robotun ortam hakkındaki tüm bilgilere sahip olduğu varsayımı yapılır. Global yol planlama işlemi robot çalışmaya başlamadan önce yapılır. Global yol planlamada tüm harita bilgisine sahip olunduğu için tüm harita üzerinde inceleme yapılır. Robotun başlangıç konumundan hedef noktasına kadar bir yol belirlenir ve robot çalışmaya başlayınca bu yolu takip eder. Global yol planlama algoritmaları tüm harita ve engel bilgisine sahip olduğu için yerel yol planlama algoritmalarına göre daha uygun yollar bulabilir (Uzun, 2022). Tüm harita üzerinde işlem yapıldığı için ve robot hareket halinde olmadığı için süreç yerel yol planlamaya göre daha yavaştır.

2.2.2. Yerel Yol Planlama

Yerel yol planlamada robotun, harita ve harita üzerindeki engeller hakkında hiçbir bilgiye sahip olmadığı ya da kısmi bir bilgiye sahip olduğu varsayılır. Robot çevrimiçiyken yani hareket halindeyken çalışır. Mobil robotun üzerinde genellikle LIDAR (Laser Detection and Ranging) sensörü bulunur robot bu sensör vasıtasıyla engeli tespit eder ve yerel yol planlama ile engele çarpmadan yolunu planlar. LIDAR sensörünün yanı sıra robotun üzerine yerleştirilen kamera, ultrasonik mesafe sensörü ve diğer optik lazer algılayıcılar kullanılabilir. Hareket halindeki robotun engellerden kaçınma süresi sınırlı olduğundan, yerel yol planlama algoritmasının ani tepkiler vermesi global yol planlama algoritmasına kıyasla daha kritik öneme sahiptir. Global yol planlama algoritmalarına kıyasla daha hızlı çalışır ancak daha az ideal yollar bulabilir (Uzun, 2022).

2.3. Engel Çeşitleri

Yol planlamada engeller, statik engeller ve dinamik engeller olmak üzere ikiye ayrılır. Ayrıca bu engelleri kendi içlerinde haritalanmış ve haritalanmamış engeller olmak üzere Şekil 2.15'te gösterildiği gibi ikiye ayırabiliriz.



Şekil 2.15. Yol Planlamada Engellerin Sınıflandırılması

Otonom mobil robot tarafından engelin konum bilgisi biliniyorsa ve robot çalışmaya başlamadan önce bu engele karşı önem alıyorsa bu engel haritalanmış bir engeldir. Eğer robot engelin konumu hakkında bir bilgiye sahip değilse ve çalışma durumunda engeli tespit edecekse bu engel haritalanmamış engel olarak adlandırılır.

Statik engeller çalışma ortamında durağan engelleri temsil eder. Bu engeller hareket etmezler ve yer değiştirmezler. Çalışma ortamındaki duvarlar, kolonlar, sandalye, masa, tezgahlar vb. statik engele örnek olarak verilebilir.

Dinamik engeller ise çalışma ortamındaki hareketli engelleri temsil eder. Dinamik engeller hareket eder ve konumları sabit değildir. Çalışma ortamındaki araçlar, diğer mobil robotlar, arabalar, insanlar, hayvanlar vb. dinamik engellere örnektir.

Mobil robotun statik engeller için ayrı dinamik engeller için ayrı yol planlama stratejisi vardır. Çalışma ortamında oluşabilecek koşullar aşağıdaki gibidir:

1) Statik ve haritalanmış engel içeren ortam
2) Statik ve haritalanmamış engel içeren ortam
3) Dinamik ve haritalanmış engel içeren ortam
4) Dinamik ve haritalanmamış engel içeren ortam

Bu tez çalışmasında 1. ve 2. koşulları içeren çalışma ortamı üzerinde çalışılacaktır. Öncelikle, otonom mobil robotun çevrimdışı olduğu varsayımıyla, harita üzerinde statik ve haritalanmış engellerin bulunduğu durumlar için metasezgisel algoritmalar kullanılarak yol planlama gerçekleştirilecektir. Burada yapılan yol planlama işlemi global yol planlamadır. Bu durum için farklı haritalar üzerinde testler yapılıp algoritmaların performansları değerlendirilecektir. Daha sonra global yol planlama yapılan ortama haritalanmamış engeller eklenecektir. Robot hareket halindeyken bu engelleri tespit edecek ve bu engelleri aşarak hedefine ulaşması için yerel yol planlama yapılacaktır.

2.4. Harita Türleri

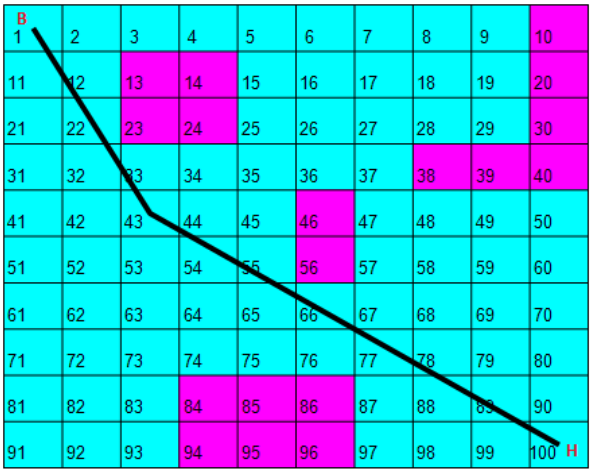
Literatürde mobil robotların yol planlaması için farklı metotlar içeren harita türleri kullanılmaktadır. Bu bölümde çalışmada kullanılan ızgara tabanlı haritadan ve literatürdeki diğer harita türlerinden bahsedilecektir.

2.4.1. Izgara Tabanlı Haritalar

Izgara tabanlı harita yönteminde, iki boyutlu (2D) ortamda x ve y eksenleri arasındaki aralık belirli bir boyutta bölünerek ızgara oluşturulur (Jung ve Kim, 2020). Izgara tabanlı haritalar, yol planlama probleminde sıklıkla kullanılan bir yaklaşımdır. Izgara tabanlı haritalar, genellikle iki boyutlu bir düzlemi ızgara formunda temsil eder. Çalışma ortamı ızgara olarak adlandırılan belirli boyutlarda hücrelere bölünerek haritalama yapılır. Her bir hücre kendisi hakkında bilgi içerir, bu bilgi yol planlama problemi için genellik o

hücrede engel olup olmaması durumudur. Engelli bölgeler çalışma ortamındaki dolu alanları (kolonlar, duvar, tezgahlar, diğer mobil robotlar, insanlar vb.) veya geçmeye imkan tanımayan bölgeleri temsil ederken, engelsiz bölgeler serbest geçiş alanlarını temsil eder. Şekil 2.16'da tezde kullanılan ızgara tabanlı harita örneği verilmiştir.

Şekil 2.16'da pembe renkli hücreler engelin var olduğu hücreleri temsil ederken, açık mavi renkli hücreler ise geçişin serbest olduğu hücreleri temsil eder. Ayrıca şekilde gösterilen "B" harfi ile işaretlenen hücre robotun başlangıç pozisyonunu, "H" harfi ile işaretlenen hücre ise robotun hedef noktasını temsil etmektedir. Yol planlama sonucu elde edilen optimal yol ise siyah bir çizgi ile gösterilmiştir.



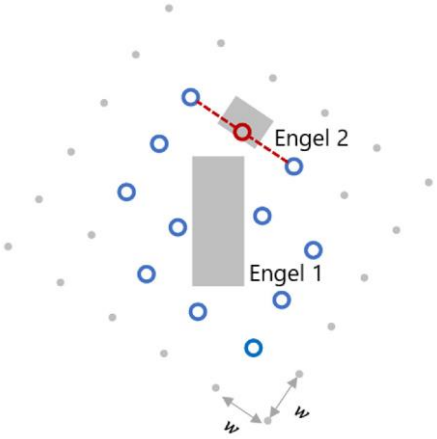
Şekil 2.16. Izgara Tabanlı Harita Örneği

2.4.2. Nokta Tabanlı Haritalar

Nokta tabanlı haritalar, çalışma ortamının belirli noktalarını ve bu noktalar arasındaki ilişkileri temsil eder. Genellikle belirli engellerin veya cisimlerin konumlarını tanımlamak için kullanılırlar. Bir mobil robot, algılayıcıları aracılığıyla çevredeki belirli noktaları tanır ve bu noktaları bir harita üzerinde işaretler. Bu noktalar, robotun konumunu belirlemek ve hareket etmek için referans noktaları olarak kullanılabilir. Şekil 2.17'de nokta tabanlı harita örneği gösterilmektedir.

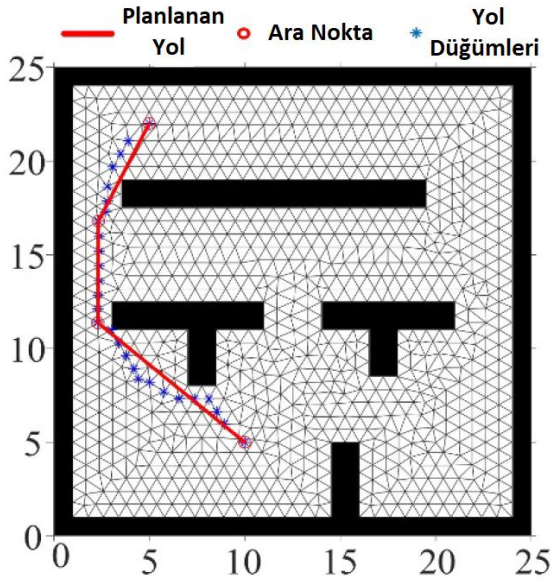
2.4.3. Ağ Tabanlı Haritalar

Çalışma ortamının hücre olarak adlandırılan ızgaralara ayrıldığı ızgara tabanlı haritalamadan farklı olarak, ağ tabanlı haritalama bir ağ oluşturarak ortamı temsil eder.



Şekil 2.17. Nokta Tabanlı Harita Örneği (Han ve Seo, 2016)

Bu teknik, karmaşık ve düzensiz ortamların değişen detay seviyeleriyle temsil edilmesi için özellikle kullanışlıdır. Ortam, birbirine bağlı üçgenler veya çokgenlerden oluşan bir ağ olarak temsil edilir. Ağ içindeki her üçgen veya çokgen çalışma ortamının bir bölümünü temsil eder. Ortamın detaylarını ve geometrik özelliklerini yakalayabilir. Dinamik ortamlarla başa çıkabilir çünkü ağ tabanlı haritalar ortamdaki değişikliklere göre güncellenebilir. Liu ve Jiang (2020) üçgensel ağ tabanlı harita yöntemi kullanarak yaptıkları çalışma sonucunda üçgensel ağ tabanlı haritanın uzun dar geçitlere sahip haritalarda ve karmaşık haritalarda (örneğin labirent haritası gibi) yapılan yol planlamasında iyi performans gösterdiği sonucuna ulaşmışlardır. Şekil 2.18'de ağ tabanlı harita örneği verilmiştir.



Şekil 2.18. Ağ Tabanlı Harita Örneği (Liu ve Jiang, 2020)

Şekli inceleyecek olursak Liu ve Jiang (2020) çalışmalarında, öncelikle çalışma ortamını üçgensel bir biçimde ağ tabanlı olarak haritalandırmışlardır. Ardından, yol planlama yaparak yol düğüm noktalarını belirlemişlerdir. Belirlenen düğüm noktalarından ara noktaları tespit ettikten sonra, bu ara noktaları kullanarak optimal planlanan yolu oluşturmuşlardır.

3. LİTERATÜR TARAMASI

Literatürde metasezgisel algoritmalarla yol planlanma üzerine birçok çalışma mevcuttur. Gul ve diğ. (2021) yol planlama için üç aşamadan oluşan bir yöntem önermişlerdir. Bu yöntem ilk aşamada Parçacık Sürü Optimizasyonu ve Gri Kurt Optimizasyon (Grey Wolf Optimizer, GWO) algoritmalarını hibrit bir şekilde kullanarak en kısa yol mesafesini hesaplar, ikinci aşamada PSO-GWO tarafından üretilen uygun olmayan noktalar Yerel Arama (Local Search) yöntemiyle entegre edilerek uygun noktalara dönüştürülür son aşamada ise engellerden kaçınma çalışması yapılır.

Mohanty ve Parhi (2016) statik engeller içeren, bilinmeyen veya kısmen bilinen ortamlarda otonom bir mobil robot için Guguk Kuşu Arama algoritmasını kullanarak bir metodoloji tanımlamışlardır. Önerilen algoritma simülasyon ortamında test edilmiş daha sonra doğrulamak için mobil robot üzerinde testler yapılmıştır.

Cong ve Ponnambalam (2009), mobil robot yol planlama problemini çözmek için Karınca Kolonisi Optimizasyon (Ant Colony Optimization, ACO) algoritmasını önermişlerdir. Önerilen ACO ile yol planlama algoritmasını literatürden elde ettikleri haritalar üzerinde test etmişlerdir. Söz konusu makale ile kendi çalışmalarını karşılaştırmışlardır.

Amar ve Jasim (2021), çeşitli statik ve dinamik engeller içeren bir ortamda otonom bir mobil robotun global yol planlama problemi için bir metodoloji geliştirmişlerdir. Parçacık Sürü Optimizasyonu ve Karınca Kolonisi Optimizasyon algoritmalarını kullanarak dinamik bir ortamda simülasyonlar gerçekleştirmişlerdir. Daha sonra PSO ve ACO algoritmalarının hibrit olarak kullanıldığı bir yöntem önermişlerdir. Algoritmaları karşılaştırdıklarında optimal yol uzunluğunu ve en kısa çalışma süresini PSO-ACO hibrit algoritmasının verdiğini elde etmişlerdir. Bununla birlikte PSO algoritması ACO algoritmasına kıyasla daha iyi performans göstermiştir.

Lin ve diğ. (2023) depoda çalışacak bir mobil robotun yol planlaması için Benzetimli Tavlama algoritmasından ilham alarak PSO-SA hibrit algoritmasını geliştirmişlerdir. Geliştirdikleri algoritmayı, Parçacık Sürü Optimizasyonu, Benzetimli Tavlama, Harmoni Arama (Harmony Search, HS), Ateşböceği Algoritması (Firefly Algorithm, FA), Yapay Arı Kolonisi (Artifical Bee Colony, ABC) ve Genetik Algoritma gibi metasezgisel algoritmalarla karşılaştırmışlardır. PSO-SA hibrit algoritmasının ortalama iterasyon

sayısı PSO, SA, HS, FA, ABC ve GA algoritmalarına göre sırası ile %3, %35, %40, %42, %36 ve %44 daha az olarak bulunmuştur. PSO-SA hibrit algoritmasının çalışması süresi her bir algoritma için yine aynı sıra ile %26, %64, %96, %100, %99 ve %99 daha az olarak belirlenmiştir. Sonuç olarak PSO-SA hibrit algoritmasının diğer algoritmalara kıyasla hızlı yakınsamaya, yüksek doğruluğa ve daha az çalışma süresine sahip olduğunu tespit etmişlerdir.

Wahab ve diğ. (2020) çalışmalarında Potansiyel Alan (PF), Dijkstra Algoritması (DA), Rastgele Ağaç Yöntemi (RRT), Olasılıksal Yol Haritası (PRM) gibi geleneksel yol planlama yaklaşımları ile GA, PSO, Diferansiyel Evrim (DE) ve CS gibi metasezgisel tabanlı yol planlama algoritmaları arasında bir karşılaştırma yapmak için farklı deney koşullarında bu algoritmaları test etmişlerdir. Algoritmaları test ederken hedefe ulaşma yeteneği, engele çarpma sayısı, çalışma süresi, pil tüketimi, katedilen mesafe, iterasyon sayısı gibi kriterleri göz önünde bulundurmuşlardır. Tüm deneyler ve kriterler değerlendirildiğinde en iyi sonucu PSO'nun bir versiyonu olan Kısıtlanmış Parçacık Sürü Optimizasyo'nu (CPSO) vermiştir.

Hussein ve diğ. (2024) yol planlama problemine metasezgisel tabanlı optimal çözüm sunmuşlardır. Çalışmaların Tabu Arama (TS), SA ve GA metasezgisel algoritmalarını kullanmışladır. Bu algoritmaları iyi bilinen statik bir ortamda test etmişlerdir. Test sonucunda SA'nın zaman açısından en iyi yaklaşım olduğunu, ardından GA'nın ve ardından TS'nin geldiğini gözlemlemişlerdir.

Pattnaik ve diğ. (2022) çalışmalarında farklı bilinmeyen statik ve dinamik ortamlarda çok amaçlı yerel yol bulma problemlerinin çözümü için yeni bir popülasyon tabanlı hibrit optimizasyon algoritması olan Hibrit Parçacık Sürü ve Kimyasal Reaksion Optimizasyonu (HPCRO) algoritması önerilmiştir. Mümkün olan en kısa ve en düzgün yolu bulmak için mobil robotun açısı, engelleri ve hedef konumu dikkate alınarak yeni bir amaç fonksiyonu formüle edilmiştir. Yol planlama problemini çözmek için simülasyonda dairesel, çokgen ve hem dairesel hem de çokgen engellerin bir arada bulunduğu farklı bilinmeyen ortam dikkate alınmıştır. Önerilen HPCRO yöntemi, simülasyon yoluyla elde edilen sonuçlarla birlikte Kimyasal Reaksiyon Optimizasyonu (CRO), PSO ve GA gibi diğer popülasyon tabanlı optimizasyon algoritmalarıyla karşılaştırılmıştır. Karşılaştırma sonucunda önerilen HPCRO algoritmasının

performansının ve verimliliğinin diğer algoritmalardan daha iyi olduğunu gözlemlemişlerdir. Bu bulguları niceliksel olarak açıklamak için önerilen HPCRO algoritmasının üstünlüğünü gösteren bazı basit istatistiksel analizler sunmuşlardır.

Ghosh ve diğ. (2017) otonom bir mobil robotun farklı şekil ve boyutlarda engeller içeren farklı varyasyonlara sahip bilinmeyen ortamlarda yol planlamasını gerçekleştirmek için doğadan ilham alınarak geliştirilen Çiçek Tozlaşma Algoritması (FPA) ve Yarasa Algoritması'nı (BA) kullanmışlardır. İki algoritmayı karşılaştırmak için MATLAB ortamında birçok simülasyon ve Arduino Mega 2560 mikrodenetleyicisine sahip bir mobil robot kullanarak testler yapmışlardır. Yol uzunluğu, engele çarpma durumu ve çalışma süresine göre algoritma değerlendirildiğinde FPA'nın verimliliğinin BA'dan daha iyi sonucuna varmışlardır. Ayrıca gerçek zamanlı yapılan testlerle bu sonucu doğrulamışlardır.

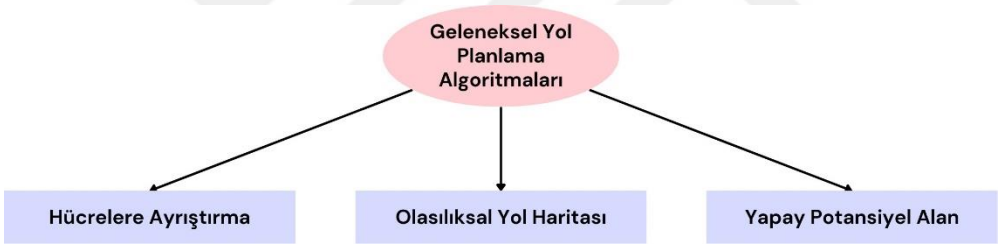
Akka ve Khaber (2018), ızgara tabanlı haritalarda mobil robotun yol planlaması için kullanılan ACO algoritmasına iyileştirmeler yapmışladır. Dört iyileştirme önermişlerdir: güvenli ve erişilebilir bir ızgara seçmek için geçiş kuralına yeni bir yaklaşım, görünürlük doğruluğunu ve arama verimliliğini artırmak için serbest adım uzunluğu ilkesini kullanan küresel sezgisel bilgisi, gelişmiş feromon güncelleme kuralı ve dinamik buharlaşma küresel arama yeteneğini ve yakınsama hızını artırma stratejisi. Simülasyon sonuçlarıyla, geliştirilmiş karınca kolonisi algoritmasının, kısa yol ve hızlı yakınsama açısından klasik karınca kolonisi algoritmasına göre önemli ölçüde üstün olduğunu kanıtlamışlardır.

4. YOL PLANLAMA ALGORİTMALARI

Yol planlama algoritmaları bir başlangıç noktasından hedef noktaya olan en uygun yolu belirlemek için kullanılan algoritmalardır. Yol planlama problemi temel olarak engele çarpmama ve en kısa yolu bulma kriterlerini içerir. Kullanıcı kendi problemine göre bu koşullara ek olarak başka koşullar ekleyebilir. Bu koşullar güvenlik faktörü, çalışma süresi, güç tüketimi gibi koşullar olabilir. Bölüm 1'de yol planlama algoritmaları; Geleneksel Algoritmalar, Sezgisel Algoritmalar ve Metasezgisel Algoritmalar olmak üzere üçe ayırılmıştı. Bu bölümde bu algoritmalar incelenecektir.

4.1. Geleneksel Algoritmalar

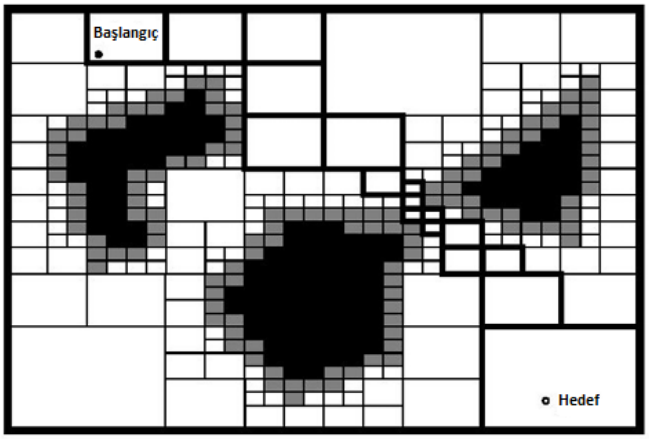
Bu bölümde Şekil 4.1'de gösterilen geleneksel yol planlama algoritmalarından Hücrelere Ayrıştırma yöntemi, Olasılıksal Yol Haritası ve Yapay Potansiyel Alan yönteminden bahsedilecektir.



Şekil 4.1. Geleneksel Yol Planlama Algoritmaları

4.1.1. Hücrelere Ayrıştırma

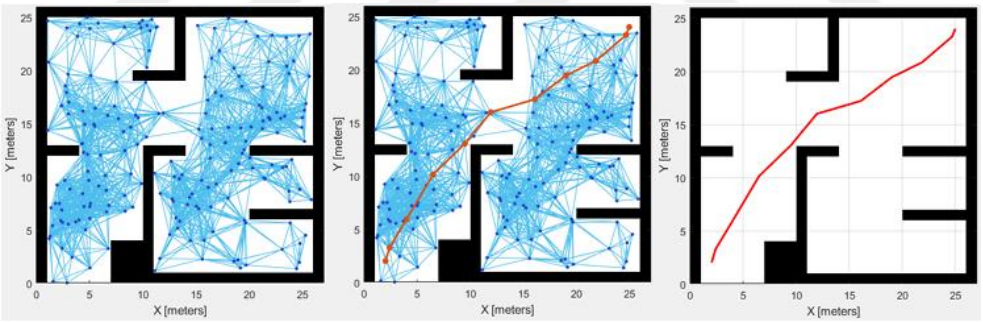
Hücrelere Ayrıştırma yöntemi mobil robotun engellerin olduğu bir harita üzerinde hareket etmesi için kullanılan bir yol planlama yöntemidir. Bu yöntem, haritayı farklı boyutlarda hücrelere böler ve her hücreyi boş, engelli veya başka bir durumda olup olmadığına bağlı olarak sınıflandırır. Ardından, bu hücreler arasındaki geçişler için uygun bir yol bulunur. Yol planlama işlemi iki adımdan oluşur: Algoritma önce robotun başlangıç hücresini ve hedef hücresini belirler daha sonra planlayıcı bu hücrelerinin bitişiğindeki uygun yolu arar. Şekil 4.2'de Hücrelere Ayrıştırma yöntemi örneği gösterilmektedir.



Şekil 4.2. Hücrelere Ayrıştırma Yöntemi (Aggarwal ve Kumar, 2020)

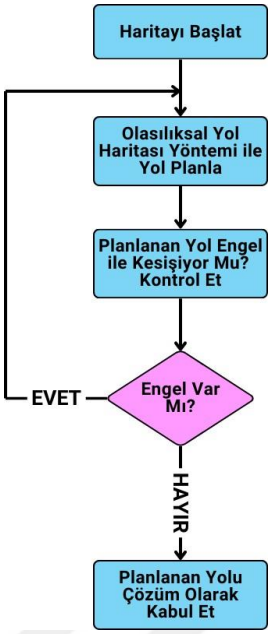
4.1.2. Olasılıksal Yol Haritası

Olasılıksal Yol Haritası (PRM), bir mobil robotun başlangıç noktasından hedef noktasına ulaşması için kullanılan bir yol planlama algoritmasıdır. PRM, hareket edilecek alanın önceden haritalandırılması ve bu haritanın üzerinde olası yolların belirlenmesi esasına dayanır. Şekil 4.3'te Olasılıksal Yol Haritası ile yol planlama aşamaları gösterilmektedir.



Şekil 4.3. Olasılıksal Yol Haritası ile Yol Planlama Aşamaları (Ouach ve diğ., 2023)

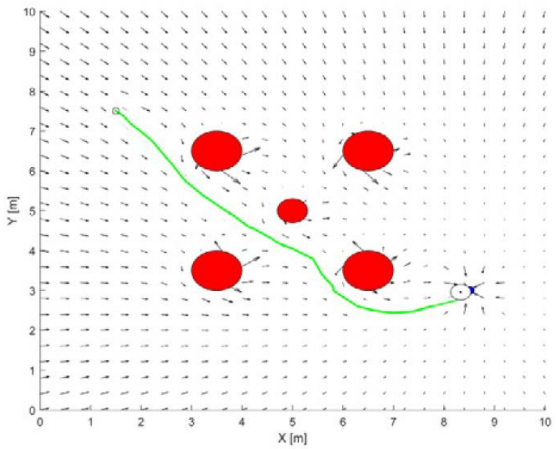
1996 yılında yeni bir yol planlama yöntemi olarak sunulan PRM algoritması öğrenme aşaması ve sorgu aşaması olmak üzere iki aşamadan oluşur (Atalı, 2018). Olasılıksal Yol Haritası yönteminde öğrenme aşamasında düğümler, iki düğüm çakışmayacak şekilde rastgele dağıtır. Daha sonra düğümler arasında yollar oluşturarak bu yolları graf olarak saklar. Bu graf, hareket edilecek alanın geometrisini ve engellerini temsil eder. Başlangıç ve hedef noktaları bu graf üzerinde bulunur ve bu noktalar arasındaki en iyi veya en uygun yolu bulmak için graf arama algoritmaları kullanılır. Şekil 4.4'te PRM algoritmasının akış diyagramı verilmektedir.



Şekil 4.4. Olasılıksal Yol Haritası Algoritması Akış Diyagramı

4.1.3. Yapay Potansiyel Alan

1985 yılında Khatib tarafından geliştirilmiştir. Yapay Potansiyel Alan (APF) yöntemi, çarpışma ve engellerden kaçınma işlemleri yol planlamanın bir parçası olduğu için mobil robotlar için yol planlamada en yaygın kullanılan tekniklerden biridir (Sabudin ve diğ., 2021). Potansiyel alana elektrik, manyetik ve yerçekimi alanları örnek verilebilir. Yapay potansiyel alan metodu otonom mobil robotun belirli bir alandaki hareketini düzenlemek için yapay potansiyel alanları kullanır. Şekil 4.5'te Yapay Potansiyel Alan yöntemi kullanılarak yapılan bir yol planlama örneği gösterilmektedir.

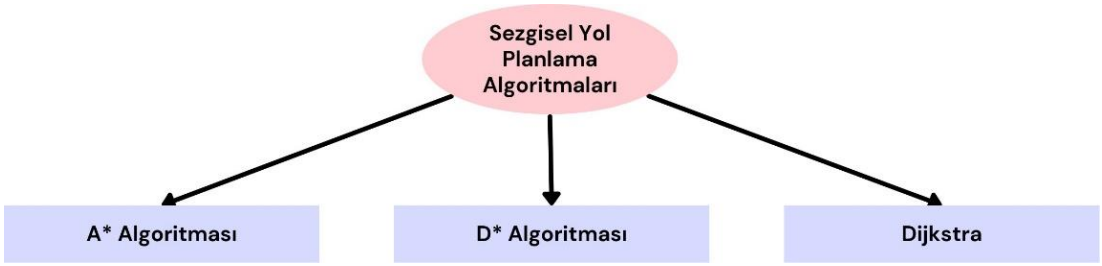


Şekil 4.5. Yapay Potansiyel Alan Yöntemi ile Yol Planlama Örneği (Orozco-Rosas ve diğ., 2019)

Mobil robot en yüksek potansiyelden en düşük potansiyele kadar simüle edilir. Burada hedef düğüm en düşük potansiyele sahipken, başlangıç düğümü maksimum potansiyele sahip olacaktır. Dolayısıyla mobil robotun en yüksek potansiyelden en düşük potansiyele doğru hareket ettiğini söyleyebiliriz (Siddiqui, 2018). Çekici alan ve itici alan olmak üzere iki tür yapay potansiyel alan oluşturulur. Ortamdaki engeller itici alanlar üretirken hedef noktası çekici alan özelliği sergiler. Ayrıcı robotun çalışma ortamı sınırlandırılmış ise bu sınır noktaları da itici kuvvet özelliği göstererek robotun sınırlardan uzak kalmasını sağlar. Potansiyel alan içindeki kuvvetler genellikle mobil robotun konumu ile hedef ve engel noktaları arasındaki mesafeye göre hesaplanır. Bu mesafe, ızgara tabanlı alanlar kullanılarak birim ölçülere dönüştürülebilir (Akarsu ve Çetin, 2022).

4.2. Sezgisel Algoritmalar

Sezgisel algoritmalar, deneyim, sezgi veya önceki bilgilere dayanarak bir sonraki adımı belirleyip çözüme ulaşmayı amaçlayan algoritma türleridir. Sezgisel algoritmalar en iyi sonucu bulmayı garanti etmezler fakat kısa bir süre içerisinde en iyi çözüme yakın, kabul edilebilir bir çözümü bulmayı garanti ederler. Genellikle en iyiye yakın olan çözüm yoluna hızlı ve kolay bir şekilde ulaşırlar. Bu bölümde Şekil 4.6'da gösterilen sezgisel yol planlama algoritmalarından; A* algoritması, D* algoritması ve Dijkstra algoritmasından bahsedilecektir.



Şekil 4.6. Sezgisel Yol Planlama Algoritmaları

Sezgisel algoritma, çözüm uzayında "akıllı" bir arama gerçekleştiren özel bir algoritma türüdür (Laubenbacher ve diğ., 2013). Olası her çözümü aramak zorunda kalmamak için rastgele bir çözüm noktasından başlar bulunan önceki çözümlerden aldığı girdilere dayanarak arama yöntemini günceller. Bu işlem daha iyi bir çözüm bulunamayana kadar veya belirlenen optimizasyon kriteri sağlanana kadar ilgili maliyet fonksiyonun değerini minimize etmek amacıyla değerleri iyileştirmek için çeşitli yöntemler kullanır. Algoritma

doğru bir şekilde oluşturulup çalıştırılırsa çözüm uzayının rastgele aranmasında büyük avantaj sağlar. Sezgisel algoritmalar çoğu modelin çözümü için kullanılabilir. Her zaman optimal çözümü vermese de diğer yöntemlerin başarısız olabileceği durumlarda her zaman kullanılabilecek bir "kaba kuvvet" tekniği sağlarlar (Laubenbacher ve diğ., 2013).

4.2.1. A* Algoritması

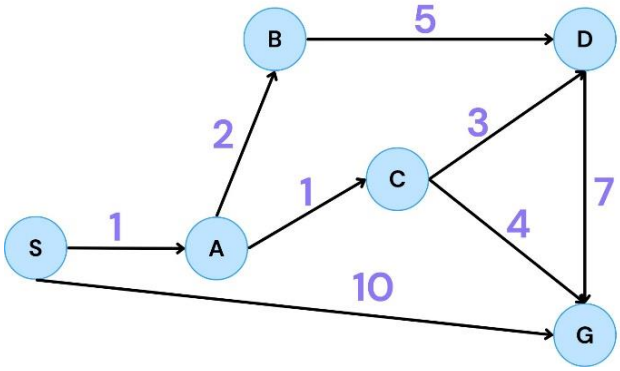
A* algoritması ilk kez 1968 yılında Peter E. Hart, Nils J. Nilsson ve Bertham Raphael tarafından yayınlanmıştır (Hart ve diğ., 1968). A* algoritması graf tabanlı bir algoritmadır ve iki nokta arasındaki en kısa yolu bulmak için kullanılır. A* algoritması Dijkstra algoritmasının geliştirilmesi ile ortaya çıkmıştır. İlk olarak A algoritması olarak adlandırılmış daha sonra kullanmış olduğu sezgisel yaklaşımdan dolayı A* olarak adlandırılmıştır (Tuncer, 2013). Sahip olduğu sezgisel işlev sayesinde diğer kör arama algoritmalarından üstündür.

A* algoritmasında ele alacağımız değerlendirme fonksiyonu $f(n)$,

$$f(n) = g(n) + h(n) \tag{4.1}$$

burada $g(n)$, başlangıç düğümünden mevcut düğüme (n. düğüme) kadar bulunan toplam minimum maliyet ve $h(n)$, mevcut düğümden hedef düğüme olan mesafenin tahmin edilen değeridir. $h(n)$ algoritmada kullanılan sezgisel değerdir. Yani $g(n)$, başlangıç düğümünden n. düğüme kadar olan en kısa mesafe değeriyken, $h(n)$, n. düğümden hedef düğüme olan en kısa mesafe değeridir. Bu değer hedef düğüm için sıfır olarak kabul edilir. Sonuç olarak $f(n)$ hesaplanan toplam yol uzunluğunun maliyetini verir. Yol planlamada amacımız $f(n)$ değerini minimize etmektir. A* algoritması temelde her yolun maliyetini göz önünde bulundurur ve aramayı hedefe doğru yönlendirir. Bunun için sezgisel işlevi kullanır ve arama sınırlarını yinelemeli olarak genişletir (Albayrak ve Sümen, 2023). Dijkstra algoritması mevcut düğümden bütün yönlere doğru ilerlerken, A* algoritması mevcut düğümden hedef düğüme doğru ilerleme gösterir.

Şekil 4.7'de bir yol bulma problemi ve düğümlerin birbirlerine olan uzaklıkları verilmiştir. Şekil üzerinde "S" noktası robotun başlangıç noktasını ve "G" noktası hedef noktayı temsil etmektedir. Başlangıç noktasından hedef noktasına olan en kısa mesafeyi A* algoritması kullanarak bulalım.



Şekil 4.7. A* Algoritması Yol Bulma Problemi Örneği

Maliyet fonksiyonu $f(n)$'yi hesaplayabilmemiz için her bir düğümün hedef noktaya olan sezgisel uzaklık değerine ihtiyacımız vardır. Tablo 4.1'de bu problem için sezgisel uzaklık değerleri verilmiştir.

Tablo 4.1. A* Algoritması Yol Bulma Problemi Örneği için Sezgisel Değerler

Düğüm Noktası	Sezgisel Uzaklık h(n)
S	7
A	3
B	4
C	2
D	5
G	0

Denklem 4.1 kullanılarak başlangıç noktasından ilk düğümlere olan maliyet fonksiyonunun değeri hesaplanır. 1. adım için iki farklı olasılık vardır:

S → A için $f(n) = 1 + 3 = 4$ (4.2)

S → G için $f(n) = 10 + 0 = 10$ (4.3)

Eşitlik 4.2 ve 4.3'ü karşılaştıracak olursak 4.2'nin maliyet değeri daha küçük olduğu için A düğümü seçilir. Burada seçilmeyen G düğümünün değerinin hafızada daha sonrası için saklanması gerektiği unutulmamalıdır.

2. adımda seçilen A düğümünden B ve C noktalarına olan maliyet fonksiyonunu hesaplanır:

S → A → B için $f(n) = 3 + 4 = 7$ (4.4)

S → A → C için $f(n) = 2 + 2 = 4$ (4.5)

Eşitlik 4.4 ve 4.5 karşılaştırıldığın maliyet değeri küçük olan C düğümü seçilir. Önceki adımda olduğu gibi seçilmeyen B düğümünün maliyeti hafızada daha sonrası için tutulur.

3. adımda seçilen C düğümünden D ve G noktalarına olan maliyet fonksiyonları hesaplanır:

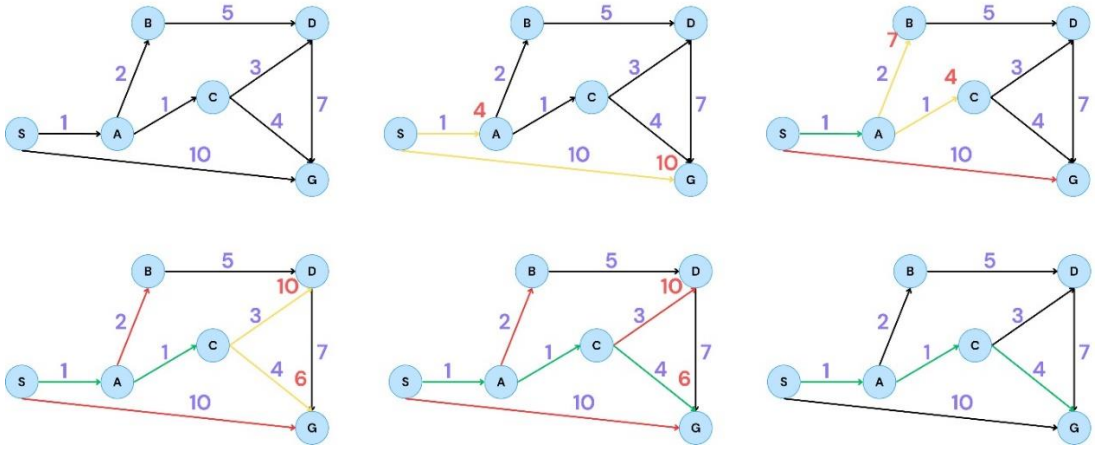
S → A → C → D için $f(n) = 5 + 5 = 10$ (4.6)

S → A → C → G için $f(n) = 6 + 0 = 6$ (4.7)

Eşitlik 4.6 ve 4.7 karşılaştırılırsa maliyet değeri küçük olan G düğümü seçilir. Hedef noktasına ulaşıldığı için bulunan 4.7 çözümü daha önce seçilmeyen hafızada tutulan maliyet değerleri ile karşılaştırılır. 4.7'de bulunan değer hafızada tutulan 4.3 ve 4.4 değerlerinden küçük olduğu için optimal çözüm olarak kabul edilir. Eğer hafızada tutulan değerlerden biri sonucun maliyet değerinden küçük olsaydı o düğümden başlanarak diğer alternatif yollarında hesaplanması gerekirdi. Şekil 4.8'de yukarıda yapılan A* algoritması ile en kısa yol hesaplama işlemleri adım adım gösterilmiştir.

4.2.2. D* Algoritması

D* algoritması, 1994 yılında Anthony Stentz tarafından Carnegie Mellon Üniversitesi'nde geliştirilmiştir (Stentz, 1994). D* algoritması graf tabanlı bir algoritmadır ve dinamik çevrelerde yol planlama yaparken mevcut harita üzerinde güncellemeler yapar. D* algoritması, A* algoritmasının dinamik bir versiyonudur ve adı buradan gelmektedir.



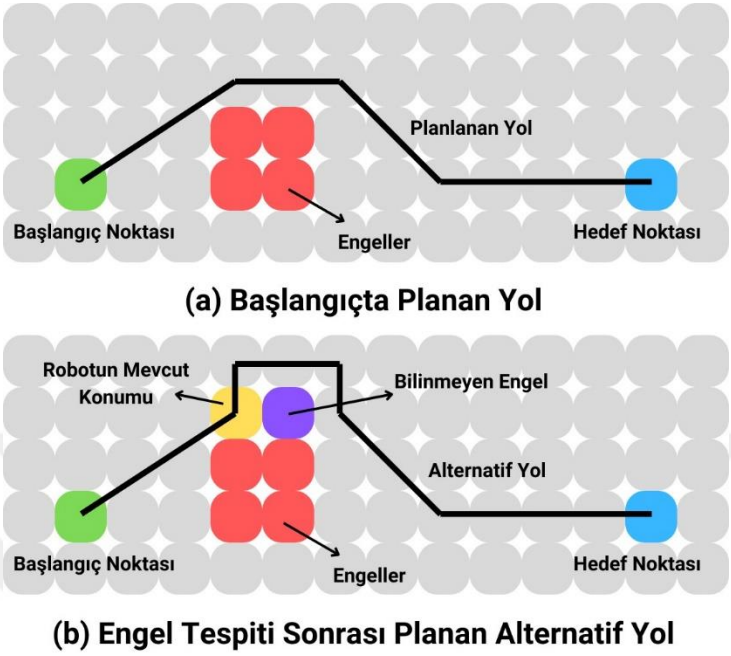
Şekil 4.8. A* Algoritması Örneği Adım Adım Gösterimi

D* algoritması, A* algoritmasına benzer şekilde çalışır fakat haritada değişiklikler olduğunda örneğin yeni engel eklendiğinde, mevcut engel yer değiştirdiğinde veya engel ortadan kalktığında A* algoritması bunlarla baş edemezken, D* algoritması engelleri tespit ettikten sonra hafızasındaki mevcut harita üzerinde güncelleme yaparak alternatif yollar arar. Özellikle çevre hakkında az bilgi bilindiği veya hiç bilgi sahibi olunmadığı durumlarda kullanılır.

Şekil 4.9'da D* algoritması ile yol planlama örneği verilmiştir. Şekil 4.9a'da mobil robot başlangıçta mevcut harita bilgisini kullanarak yol planlama işlemini gerçekleştirilmiştir. Şekil 4.9b'de ise mobil robot planlanan yolda hareket halindeyken haritalandırılmamış bilinmeyen bir engel ile karşılaşmıştır. Robot bu engeli tespit edip mevcut haritasını güncelledikten sonra bulunduğu mevcut konumundan hedef noktasına alternatif bir yol planlamıştır. D* algoritmasını, A* algoritmasından ayıran en önemli özellik hareket sırasında karşılaşılan robot tarafından bilinmeyen haritalanmamış engellere karşı çözüm üretebilmesidir.

D*-Lite algoritması, Sven Koenig ve Maxim Likhachev tarafından önerilen dinamik bir yol planlama algoritmasıdır (Koenig ve Likhachev, 2005). Başlangıçta planlanan yolun düğüm bilgilerini kullanarak yeniden yol planlama işlemi gerçekleştirir ve bu yol planlama verimliliğini artırabilme avantajına sahiptir dolayısıyla bilinmeyen ortamlarda robotların dinamik yol bulmasına uygulanabilir (Yu ve diğ., 2022). D*-Lite, D* algoritması gibi graf tabanlı arama bir algoritmadır ve özellikle dinamik çevrelerde

kullanılmak üzere tasarlanmıştır. D*-Lite, D* algoritmasının bir türevidir ve dinamik çevrelerde gerçek zamanlı yol planlama için daha etkili bir şekilde çalışır.



Şekil 4.9. D* Algoritması Örneği

D*-Lite algoritması, D* algoritmasının birçok özelliğini içerir ve tıpkı D* algoritması gibi temeli A* algoritmasına dayanır. Ancak D*-Lite, D* algoritması üzerinde bazı iyileştirmeler ve optimizasyonlar yapılarak dinamik çevrelerde daha etkili bir şekilde çalışacak şekilde tasarlanmıştır.

Focused D* algoritması ise D*-Lite algoritmasının bir varyasyonudur. Focused D* algoritması, D*-Lite'ın hızlı bir uygulamasıdır ve genellikle gerçek zamanlı sistemlerde kullanılır. Focused D* algoritması, D* ve D*-Lite algoritmasında olduğu gibi A* algoritmasına dayanır. Bu üç algoritmada dinamik çevre koşullarında yol planlama için gerçek zamanlı olarak kullanılabilir. Bu algoritmalar, çevredeki dinamik değişikliklere hızlı bir şekilde tepki vererek ve mevcut haritalarını sürekli olarak güncelleyerek yol planlama problemini gerçek zamanlı olarak optimize etmeyi hedefler.

4.2.3. Dijkstra Algoritması

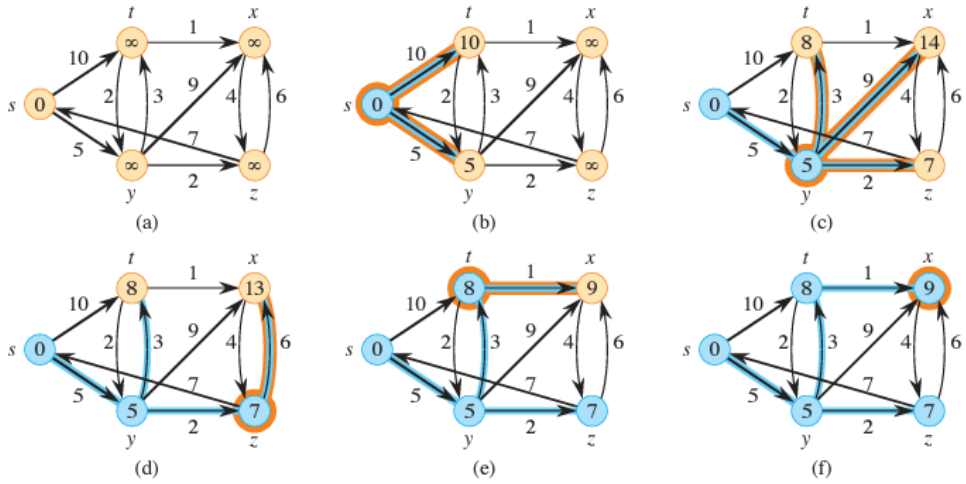
Dijkstra algoritması, Hollandalı bilgisayar bilimci Edsger Dijkstra tarafından 1956'da tasarlanmış ve 1959'da yayınlanmıştır (Deng ve diğ., 2012). Bir başlangıç noktasından

hedef noktasına olan en kısa yolu hesaplamak için kullanılan graf tabanlı bir algoritmadır. Dijkstra algoritması, tüm yollar arasında en düşük maliyetli yolu sırayla aradığından, başlangıç noktasından başlayarak arama bölgesi boyunca eş merkezli olarak genişler. Başlangıç ile hedef düğüm arasındaki tüm yolları teker teker deneyerek en uygun yolu bulmaya çalışır. Dolayısıyla bu yöntemin, zayıf arama verimliliği ve hedefe olan mesafe büyük olduğunda uzun arama süresi gibi dezavantajları vardır (Noto ve Sato, 2000). Dijkstra algoritması basit ve kolay olması nedeniyle robot yol planlaması için iyi bir yöntemdir ve ağ optimizasyonu, ulaşım, lojistik, elektronik ve diğer alanlarda yaygın olarak kullanılmaktadır örneğin GPS navigasyon sistemleri genellikle yol planlama için Dijkstra algoritması kullanılır (Zhang ve Zhao, 2014). Dijkstra algoritması en kısa yolu hesaplarken aşağıdaki adımları sırasıyla izler:

1. Başlangıç düğümüne 0 değerini atayın ve geriye kalan bütün düğümlerle arasında olan mesafeyi sonsuz (∞) olarak kabul edin ve sonsuz değeri atayın.
2. Başlangıç düğümünü geçici olarak işaretleyin.
3. Geçici olarak işaretlenen düğüme komşu olan tüm düğümleri gözden geçirin. Her komşu düğüm için başlangıç düğümünden o düğüme olan mesafeyi güncelleyin. Eğer başlangıç düğümünden komşu düğüme olan mesafe daha önceden hesaplanan mesafeden kısa ise yeni değer ile güncelleyin. İlk aşamada tüm düğümler sonsuz olarak kabul edildiği için ilk hesaplanan mesafe her zaman daha kısa olacaktır. Bu aşamada başlangıçtan komşu düğümlere olan mesafeler güncellenir.
4. Şimdiye kadar işaretlenmemiş ve en kısa mesafeye sahip olan düğümü seçin ve bu düğümü geçici olarak işaretleyin.
5. Seçilen düğümü kalıcı olarak işaretleyin ve 3. adımdaki gibi komşu düğümlerinin mesafeleri güncelleyin.
6. Tüm düğümler işaretlenene kadar 4. adıma geri dönün.

Dijkstra algoritması, bu adımları izleyerek tüm düğümlere olan en kısa yolları hesaplar ve başlangıç ile hedef düğüm arasındaki en kısa yolu belirler. Algoritma, her adımda en yakın düğümü seçerek ilerler ve bu nedenle açgözlü (greedy) algoritmalar sınıfına dahil edilir. Dijkstra algoritması adım adım Şekil 4.10'da gösterilmektedir.

Şekil 4.10a'da görüldüğü üzere ilk önce başlangıç düğümüne sıfır değeri, geriye kalan bütün düğümlere sonsuz değeri atanır. Şekil 4.10b'de ise başlangıç düğümünden komşu düğümlere olan mesafe hesaplanmış ve düğümlerin değerleri güncellenmiştir.



Şekil 4.10. Dijkstra Algoritması Adımları (Cormen ve diğ., 2022)

Şekil 4.10c'de görüldüğü gibi düğümler arasından değeri en küçük olan yani en kısa mesafeyi veren düğüm seçilmiş ve komşu düğümlerinin değerleri hesaplanmıştır. Eğer yeni hesaplanan değer mevcut değerden küçük ise değer güncellenmiştir. Geriye kalan şekillerden de görebileceğimiz gibi bu işlemler tüm düğümler ziyaret edilip değerleri güncellene kadar devam etmiştir.

4.3. Metasezgisel Algoritmalar

Metasezgisel algoritmalar, insan zekasından veya doğadan ilham alır. Bu tür algoritmalar kesin çözümü garanti etmezler sadece kesin çözüme yakın bir çözümü garanti edebilirler (Karaboğa 2020). Metasezgisel algoritmalar farklı şekillerde sınıflandırılır. Meraihi ve ark. metasezgisel algoritmaları Şekil 4.11'deki gibi sınıflandırmışlardır (Meraihi ve diğ. 2021). Bu tez kapsamında tek çözüm tabanlı algoritmalardan Benzetimli Tavlama (SA), popülasyon ve evrimsel tabanlı olan Genetik Algoritma (GA), popülasyon ve sürü tabanlı olan Guguk Kuşu Arama (CS) algoritması ve popülasyon ve sürü tabanlı olan Parçacık Sürü Optimizasyonu (PSO) algoritması incelenecektir.

Metasezgisel algoritmalar, optimize etmesi zor olan problemleri çözmek için kullanılan bir tür optimizasyon algoritmasıdır. Metasezgisel algoritmalar birçok farklı problem türünde başarılı bir şekilde kullanılabilir örneğin:

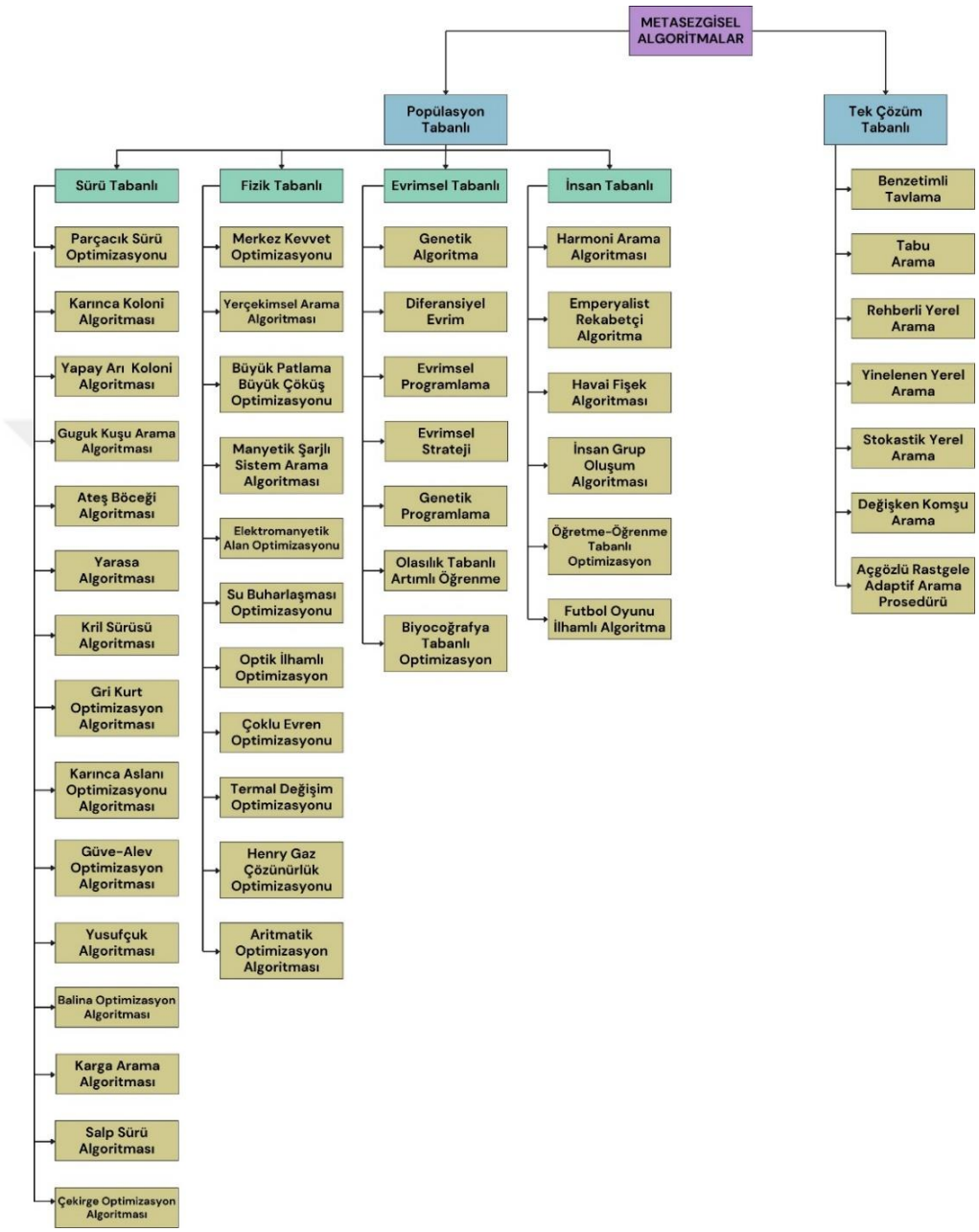
- Karmaşık matematiksel fonksiyonların optimizasyonu
- Mobil robot yol planlama problemi
- Mühendislik tasarım problemleri
- Bir üretim sürecinin optimize edilmesi
- Veri madenciliği
- Yapay zeka
- Bir portföyün risk-getiri dengesinin iyileştirilmesi
- Makine öğrenmesi
- Gezgin satıcı problemi
- Graf problemleri
- Karmaşık planlama ve zamanlama problemi

gibi alanlarda etkili bir şekilde kullanılabilir.

Metasezgisel algoritmalar, başlangıçta belirlenen amaç fonksiyonunu optimize etmek için birçok iterasyon üzerinde çalışır. Bu algoritmalar, genellikle rastgele bir veya daha fazla başlangıç noktasından başlayarak, problem alanında daha iyi çözümler bulmak için bir arayış sürecini simüle ederler. Bu süreçte, potansiyel çözümler üretilir, değerlendirilir ve ardından mevcut en iyi çözüm veya çözümlerle karşılaştırılarak süreç optimizasyon kriteri sağlanana kadar devam eder.

Bu tür algoritmaların avantajı, birçok farklı problemlere uygulanabilen ve problem alanındaki spesifik bilgiye çok az veya hiç ihtiyaç duymayan genel amaçlı optimizasyon teknikleri olmalarıdır. Metasezgisel algoritmalarla optimizasyon işleminde amaç fonksiyonunun doğru bir biçimde oluşturulması ve kullanılan metasezgisele göre parametrelerin belirlenmesi çok önemlidir. Bu parametreler tecrübe ile belirlenip kesin bir değeri olmadığı için metasezgisel algoritmanın sonucu programcının bilgisi ve tecrübesine bağlıdır. Metasezgisel algoritmaların dezavantajları arasında performanslarının problem ve veri setine bağlı olarak değişkenlik gösterebilmesi,

sonucun başlangıç nokta veya noktalarına bağlı olması ve bazı durumlarda optimize edilmesi gereken parametrelerin sayısının fazla olması sayılabilir.

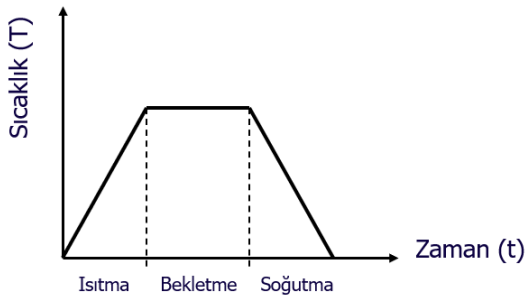


Şekil 4.11. Metasezgisel Algoritmaların Sınıflandırılması

4.3.1. Benzetimli Tavlama

Benzetimli Tavlama algoritması 1983 yılında Kirkpatrick ve arkadaşları tarafından optimizasyon problemlerinin çözümü için geliştirilmiştir (Kirkpatrick ve diğ., 1983).

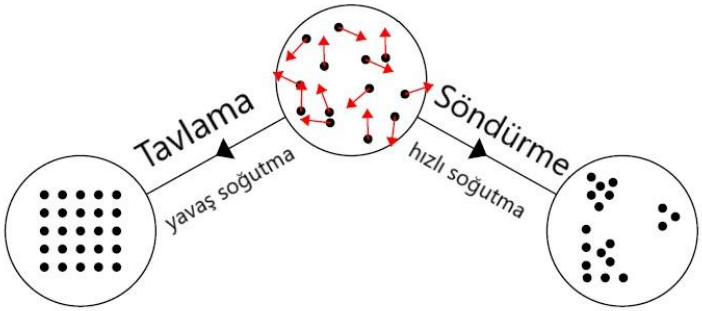
Benzetimli Tavlama algoritması karmaşık optimizasyon problemlerini çözmek için kullanılan bir metasezgisel algoritmadır. SA esnek bir optimizasyon yöntemidir ve büyük ölçekli entegre optimizasyon problemlerine uygundur (Venkateswaran ve diğ., 2022). SA bir fonksiyonun veya bir problemin minimum veya maksimum değerini bulmak amacıyla kullanılır ve amaç fonksiyonu buna göre oluşturulur. Bu algoritma, metallerin tavlama işleminden ilham alınarak optimizasyon problemlerinin çözümü için geliştirmiştir. Tavlama işleminde, bir metalin sıcaklığı belirli bir maksimum dereceye kadar artırılır, daha sonra yavaş yavaş soğutulur eğer soğutma işlemi uygun şekilde yapılırsa güçlü kristal yapı elde edilir. Tavlama işlemi iki adımda gerçekleştirilir: Metal istenilen sıcaklığa kadar ısıtılır, belli bir sıcaklıkta bir süre tutulduktan sonra yavaş yavaş soğutulur (Karaboğa, 2020). Bu işlem adımları Şekil 4.12'de gösterilmektedir.



Şekil 4.12. Tavlama İşlemi Adımları (Karaboğa, 2020)

Bir metal yeterince yüksek bir sıcaklığa ısıtıldığında, atomları büyük genliğe sahip olur düzensiz bir şekilde hareket etmeye başlar. Eğer metal aşamalı olarak soğutulursa, atomların hareketleri azalır ve minimum enerjiyle düzenli bir kristal yapıda sabit konumlar etrafında stabilize olma eğilimi gösterir (Chopard ve Tomassini, 2018). Bu durumda metalin sünekliği artacağı için metal kolay bir şekilde işlenir. Yapılan bu yavaş soğutma işlemine metalürjistler tarafından tavlama adı verilir ve bu işlem, metalin veya alaşımın hızlı bir şekilde soğutulmasını içeren söndürme işlemiyle karıştırılmamalıdır. Söndürme işleminde soğutma işlemi aniden uygulandığı için metal daha kırılgan olur. Tavlama ve söndürme işlemleri Şekil 4.13'te verilmiştir. Şekildeki atomlar yüksek sıcaklıkta yüksek enerjiye sahip oldukları için rastgele ve hızlı bir şekilde hareket eder. Yavaş soğutma işlemi yani tavlama uygulanırsa atomlar minimum enerjili düzenli duruma geçer. Eğer soğutma işlemi hızlı ve ani bir şekilde uygulanırsa atomlar düzenli ve düzensiz bir biçimde rastgele dağılacaktır.

Bu işlemleri metasezgisel açıdan şu şekilde yorumlayabiliriz; atomlar ilk durumda rastgele bir şekilde hareket eder ve enerjileri yüksektir yani SA algoritması rastgeleliklere izin verir ve sürekli yeni çözümler üretir. Fakat yavaş soğutma işlemi gerçekleştikçe sıcaklık azalır ve atomların enerjileri düşer ve git gide oldukları pozisyonu korumak isterler. Bu da algoritmada, yeni üretilen rastgele çözümlerden kötü olanlarının kabul edilme olasılığının düşürüldüğü kısma karşılık gelir.



Şekil 4.13. Tavlama ve Söndürme İşlemleri

Son olarak soğutma işlemi tamamlandığında atomlar pozisyonları koruyacaktır ve hareket etmeyerek donacaklardır. Bu kısım SA algoritmasında sıcaklığın sıfır olduğu yani yeni rastgele komşu çözümlerin kabul edilmediği ve nihayetinde algoritmanın tamamladığı anlamına gelir. SA optimizasyon algoritması ile tavlama işlemi arasındaki benzerlik (Xavier-de-Souza ve diğ., 2010) Şekil 4.14'teki gibidir.

Tavlama İşlemi		Benzetimli Tavlama Algoritması
Fiziksel Malzeme Durumu		Problem Çözümleri
Enerji Durumu	⇒	Maliyet Fonksiyonu
Sıcaklık		Kontrol Parametresi

Şekil 4.14. Tavlama İşlemi ile Benzetimli Tavlama Arasındaki Benzetim

SA algoritmasının çalışma adımları şu şekildedir:

1. Başlangıçta bir tane rastgele çözüm üret.
2. Mevcut çözümün maliyetini hesapla.
3. Komşuluk işlemlerini kullanarak bir sonraki aday çözümü rastgele bir şekilde üret.
4. Komşu çözümün maliyetini hesapla.

5. Var olan çözüm ile komşu değerin maliyetini karşılaştır. Eğer komşu çözüm daha iyiyse bunu yeni çözüm olarak kabul et. Komşu çözüm var olan çözümden kötü ise komşu çözümü belirli bir olasılığa bağlı olarak yeni çözüm olarak kabul et.
6. Kabul edilme olasılığını var olan çözüm ile komşu çözümün maliyetleri arasındaki farka ve sıcaklığa bağlı olarak hesapla. (Yüksek sıcaklıklarda kötü çözümün kabul edilme olasılığı yüksektir. Bu rastgele bir arama sağlar ve yerel minimum veya maksimuma takılmayı önleyebilir. Sıcaklık değeri azaldıkça kötü çözümlerin kabul edilme olasılığı azalır ve algoritma yerel minimum veya maksimuma yakınsar. Ayrıca var olan çözüm ile komşu çözümün maliyetleri arasındaki fark azaldıkça kabul edilme olasılığı artar.)
7. Belirli bir iterasyon sayısı sağlanana kadar veya optimizasyon kriteri sağlanana kadar 3. adıma geri dön.

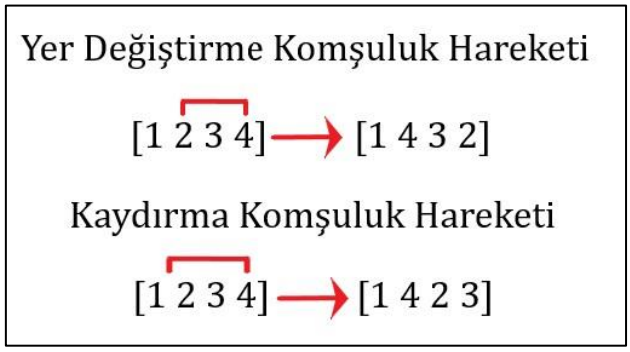
SA algoritması tek çözüm tabanlı bir algoritmadır ve algoritma rastgele üretilen tek çözüm üzerinden ilerler. Rastgele üretilen çözümler temel olarak üç şekilde aşağıdaki gibi gösterilebilir:

$$\text{İkili Gösterim} \rightarrow [1\ 0\ 1\ 1\ 0\ 1\ 0\ 1] \tag{4.8}$$

$$\text{Tam Sayı Gösterimi} \rightarrow [7\ 3\ 1\ 8\ 13\ 4\ 0\ 41] \tag{4.9}$$

$$\text{Reel Sayı Gösterimi} \rightarrow [4.7\ 1.2\ 2.4\ 4.8\ 0.7\ 4.4\ 0.1\ 9.1\ ] \tag{4.10}$$

SA algoritmasında var olan çözüm kullanılarak üretilen yeni çözüm komşu çözüm olarak adlandırılır. Komşu çözüm elde etmek için komşuluk hareketleri kullanılır. Şekil 4.15'te gösterilen yer değiştirme ve kaydırma komşuluk hareketleri kullanılabilir.



Şekil 4.15. Komşuluk Hareketleri

SA algoritması yerel minimuma veya maksimuma takılmamak için kötü çözümleri belirli bir olasılıkla kabul edilebilir. Bu olasılık var olan çözüm ile komşu çözümün maliyetleri arasındaki fark ve sıcaklığa bağlıdır. Olasılık koşulu aşağıdaki denklemlerde verilmiştir.

$$\Delta C = C_i - C_{i-1} \tag{4.11}$$

$$R < e^{\left(\frac{-\Delta C}{kT}\right)} \tag{4.12}$$

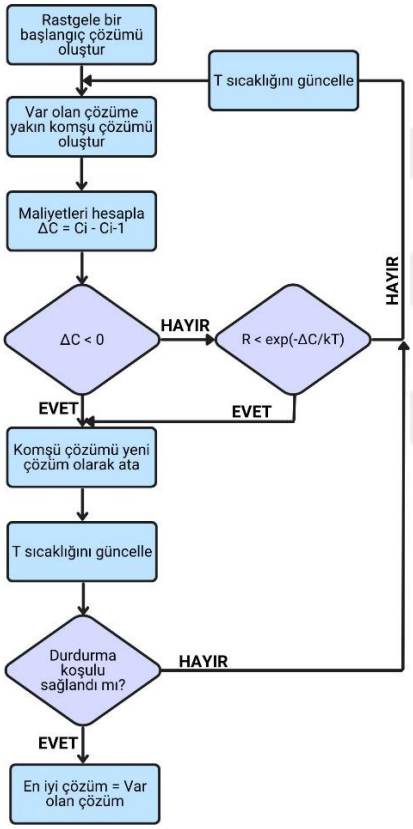
Burada, ΔC maliyet fonksiyonları arasındaki farkı, C_i komşu çözümün maliyetini, C_{i-1} var olan çözümün maliyetini, R sıfır ile bir arasında üretilen rastgele sayıyı, k sabiti ve T sıcaklığı temsil etmektedir. Denklem 4.12'deki k sabiti Boltzmann sabitini temsil etmektedir. TÜBİTAK Ulusal Metroloji Enstitüsü'nün sitesinde yer alan (URL-10) ve 16 Kasım 2018'de gerçekleşen 26. Uluslararası Ağırlıklar ve Ölçüler Genel Konferansı'nda alınan karara göre SI sistemindeki temel birimlerinin tanımlarında esas alınan yedi evrensel sabitler güncellenmiş ve Boltzmann sabitinin güncel değeri 1,380649 x 10^{-23} olarak belirlenmiştir. Denklemleri inceleyecek olursak $e^{\left(\frac{-\Delta C}{kT}\right)}$ ifadesi olasılık koşulunu ifade etmektedir eğer rastgele üretilen sayı bu değerden küçük ise komşu çözüm yeni çözüm olarak kabul edilir. Eksponansiyel ifadeye bakacak olursak maliyet fonksiyonları arasındaki fark azaldıkça ve sıcaklık değeri arttıkça komşu çözümün kabul olasılığı artmaktadır. Böylece algoritma yerel minimuma veya maksimuma takılmaz global minimum veya maksimum değerine yakınsar. SA algoritmasının işlem adımları Şekil 4.16'da gösterilmektedir.

Şekil 4.16'daki akış diyagramına bakacak olursak SA algoritması temel olarak iki stokastik süreçten oluşmaktadır. Birincisi maliyete ve sıcaklık değerine bağlı olarak yeni komşu çözümlerin üretildiği ve yeni çözümün kabul koşulunu içeren süreçtir. İkinci süreç ise sıcaklığın güncellendiği ve algoritmayı durdurma koşulu olan optimizasyon kriterinin sağlanıp sağlanmadığını kontrol eden süreçtir.

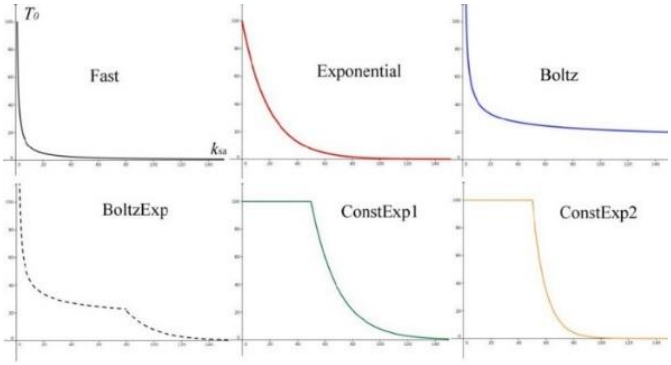
SA algoritmasının performansı sıcaklığa bağlıdır bunun için kullanılan soğutma çizelgesinin seçimi önemlidir. Şekil 4.17'de kullanılabilecek farkı soğutma çizelgesi örnekleri gösterilmektedir. Ayrıca soğutma işleminde kullanılan başlangıç sıcaklığı, son sıcaklık, soğutma katsayısı ve algoritmanın iterasyon sayısının seçimi iyi bir sonuç almak

için son derece önemlidir. SA algoritması çözüme arama uzayında rastgele bir noktadan başladığı için başlangıç noktasına bağlı bir algoritmadır.

SA algoritması maliyet minimizasyonunda, yol bulma problemlerinde, üretim süreçlerinin optimizasyonunda, lojistik problemlerde, rota planlamada, makine öğrenmesinde model parametrelerinin optimizasyonunda, elektronik devre tasarımında, envanter yönetiminde, ağ trafiği yönetiminde, veri iletimi optimizasyonunda ve finans sektöründe risk yönetimi ve tahmin modeli oluşturma gibi alanlarda kullanılabilir.



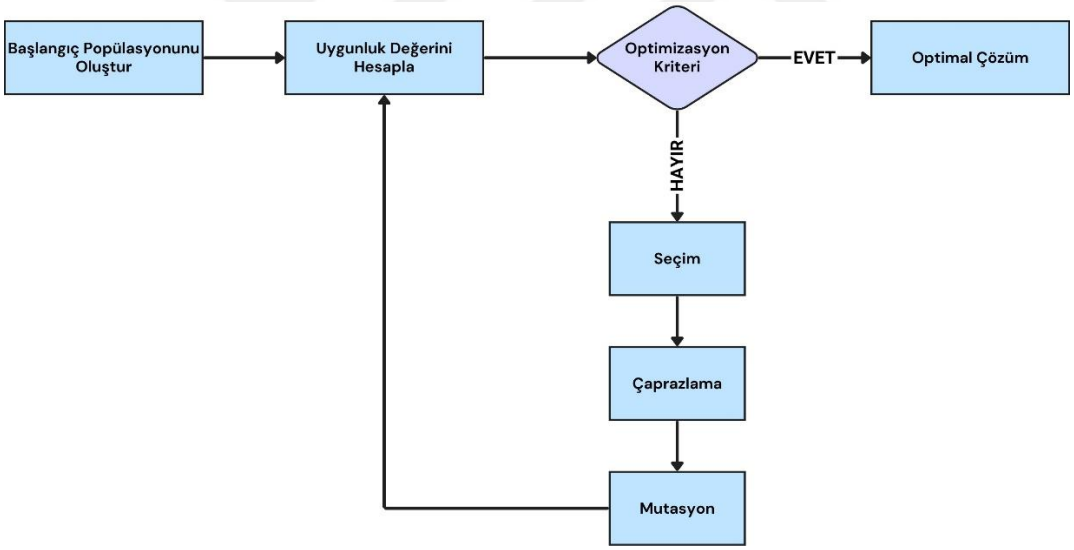
Şekil 4.16. Benzetimli Tavlama Algoritması Akış Diyagramı



Şekil 4.17. Soğutma Çizelgesi Örnekleri (Gonzales ve diğ., 2015)

4.3.2. Genetik Algoritma

Genetik algoritma 1974 yılında J. Holland ve arkadaşları tarafından geliştirilmiştir. Darwin'in evrim teorisini temel alır. Genetikte olduğu gibi en iyi olanın hayatta kalması veya güçlünün zayıfı yenmesi felsefesine dayanır (Bekdaş ve diğ., 2021). Evrim teorisinde yer alan; üreme, çaprazlama ve mutasyon genetik işlemlerini kullanır. GA temel prensip olarak en iyi olanın hayatta kalma şansının daha yüksek olmasına dayansa da seçim ve çaprazlama işlemleri ile rastgele çözümleri birleştirerek bir arama gerçekleştirir. Paralel bir arama yöntemi olduğu için çalışma uzayının tümünde arama yapabilir. Arama uzayının farklı noktalarında aynı anda arama yaptığı için yerel minimum veya maksimum değerlere takılmadan küresel çözüme ulaşma olasılığı yüksektir. Hesaba dayalı yöntemler matematiksel ve iteratif formüller kullanan yöntemlerken, GA nesilden nesile çözümün gelişerek optimum değerin bulunduğu bir yöntemdir. Genetik algoritmanın akış diyagramı Şekil 4.18'de gösterildiği gibidir.

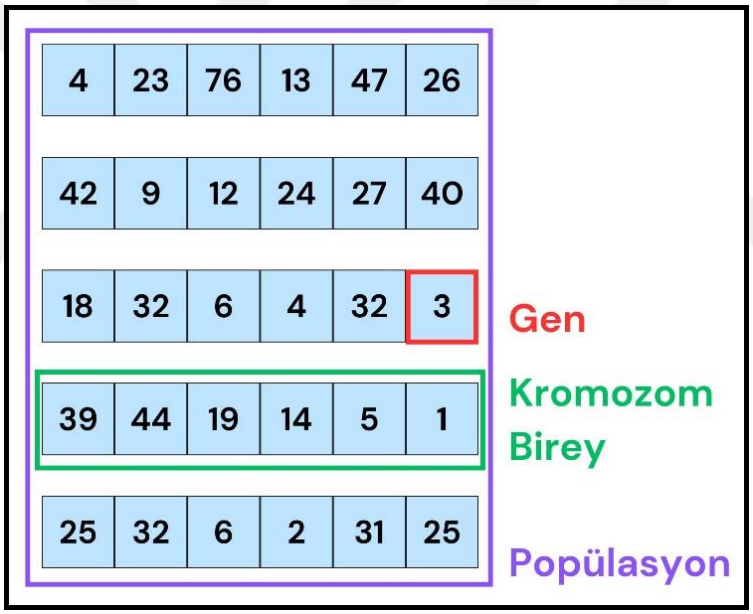


Şekil 4.18. Genetik Algoritma Akış Diyagramı

Şekil 4.18'deki diyagramı inceleyecek olursak GA'da ilk önce başlangıç popülasyonu oluşturulur. Oluşturulan başlangıç popülasyonun her bir bireyinin uygunluk değeri hesaplanır ve optimizasyon kriteri kontrol edilir. Eğer optimizasyon kriteri sağlanmıyorsa bireyler arasından rastgele seçim yapılır. Bu seçim işlemi yapılırken uygunluk değeri yüksek olan bireyin seçilme olasılığı daha yüksektir ve aynı birey birden fazla kez seçilebilir. Popülasyon sayısı kadar seçim işlemi yapılır ve bireyler arasında rastgele çiftler oluşturularak çaprazlama işlemi gerçekleştirilir. Çaprazlama sonucunda yeni

bireyler oluşur. Oluşan yeni bireylerin her bir geni küçük bir olasılıkla mutasyon işlemine uğrar. Mutasyon işleminin amacı çözümün tek bir nokta etrafında yığılmasını önleyerek çeşitliliği arttırmaktır. Daha sonra yeni oluşan bireylerin uygunluk değerleri hesaplanır ve optimizasyon kriteri kontrol edilerek işlemler tekrarlanır.

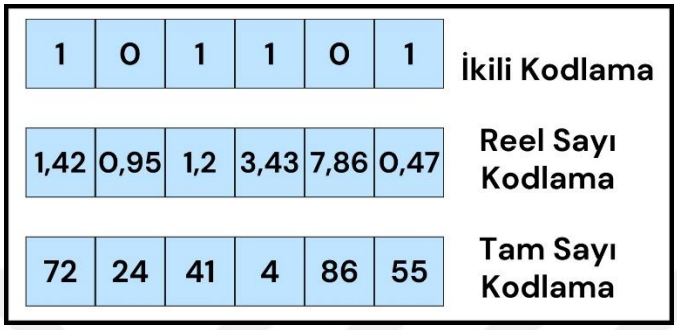
GA'da tüm çözümler popülasyon, popülasyon içerisindeki her bir tek çözüm kromozom veya birey, kromozomun yani tek bir çözümün her bir değişkeni ise gen olarak adlandırılır. Şekil 4.19'da genetik algoritmanın çözüm kümesi ve çözüm kümesinin isimlendirilmesi verilmiştir. GA'da başlangıç popülasyonun büyüklüğü çok önemlidir ve algoritma boyunca sabit kalır. Seçim işlemi popülasyon içeresindeki bireyleri veya kromozomları seçmeyi içerir. Mutasyon işlemi ise her bir kromozomun her genine küçük bir olasılık ile uygulanır.



Şekil 4.19. Genetik Algoritma Çözüm Kümesinin İsimlendirilmesi

GA'da başlangıç popülasyonu oluşturmadan önce yapılması gereken ilk iş kodlama türünün seçilmesidir. Kodlama işlemi Şekil 4.20'de de gösterildiği gibi ikili kodlama, reel sayı kodlama ve tam sayı kodlama olarak yapılabilir. Algoritma boyunca çaprazlama ve mutasyon işlemleri bu kodlama türü üzerinden yapılacağı için kodlama türüne algoritma oluşturulmadan karar verilmedir. Probleme uygun olarak başka kodlama türleri de kullanılabilir.

GA en iyi bireylerin bir sonraki nesile aktarılması prensibine dayanır. Bu da seçim işlemi ile gerçekleştirilir. Seçim işleminde popülasyon içerisinden seçilen bireyler ebeveyn olarak adlandırılır ve bu ebeveynler kullanılarak üretilen yeni çözümler çocuk olarak adlandırılır. İyi çocuklar yani yeni çözümler üretmek için seçim işleminin nasıl yapıldığı son derece önemlidir.



Şekil 4.20. Genetik Algoritma Kodlama Türleri

Seçim işleminin yapılabilmesi için ilk önce popülasyondaki tüm bireylerin maliyet fonksiyonu yardımıyla maliyeti hesaplanır. Genetik algoritmanın diğer aşamaları probleme bağlı değilken maliyet fonksiyonu probleme bağlıdır. Maliyet fonksiyonu probleme uygun olarak doğru bir biçimde oluşturulmalıdır. Maliyet fonksiyonu probleme bağlı tek kısım olduğu için genetik algoritmada maliyet fonksiyonu değiştirilerek aynı algoritma üzerinden farklı problemler çözülebilir. Maliyet fonksiyonu belirlendikten sonra maliyet değerleri uygunluk değerlerine dönüştürülür. Bunun nedeni farklı problem çözümlerinde oluşacak değer farklılıklarını ortadan kaldırmaktır. Popülasyondaki her bir bireyin uygunluk değeri orantısal olarak Denklem 4.13 kullanılarak belirlenebilir.

$$F_i = F_{max} - (C_i - C_{min})/(C_{max} - C_{min}) \quad (4.13)$$

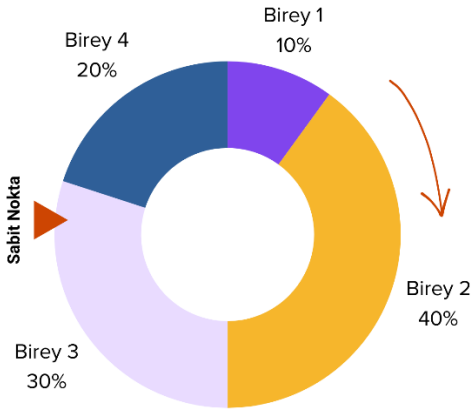
Burada F_i i. bireyin uygunluk değeri, F_{max} uygunluk değerinin maksimum değeridir örneğin uygunluk değerinin 0 ile 1 arasında bir değer olmasını istiyorsak F_{max} bir olarak seçilir. C_i ise i. bireyin maliyet fonksiyonunun değeri, C_{max} maliyet fonksiyonunun maksimum değeri, C_{min} maliyet fonksiyonunun minimum değeridir. Uygunluk değeri belirlendikten sonra uygunluk değeri yüksek olan bireylerin seçilme olasılığı daha yüksek olacak şekilde seçim işlemi yapılır.

Seçim işlemi için farklı yöntemler mevcuttur. Genetik algoritmada seçim işlemi için genellikle rulet tekeri yöntemi kullanılır. Bu yöntemde bireyler uygunluk değeri ile doğru orantılı olacak şekilde seçilir. Problem için en uygun bireyler büyük uygunluk değerine sahip olacağı için seçilme olasılıkları fazla olacaktır. Rulet tekeri daire şeklindedir ve bireyler uygunluk değerine göre daire diliminde yer alır. Rulet tekeri çevrilir ve rastgele bir şekilde bireyler seçilir. Bu yöntemde uygunluk değeri çok küçük olan bireylerin düşük olasılıkla da olsa seçilme ihtimali vardır. Rulet tekeri yönteminin işlem adımları aşağıdaki gibidir:

1. Her bireyin uygunluk değeri hesaplanır. Uygunluk değeri yüksek olan bireyler o problem için en uygun olan çözümlerdir.
2. Her bireyin seçilme olasılığı uygunluk değeri ile doğru orantılı olacak şekilde belirlenir. Daha uygun bireyler, rulet tekerinde daha büyük bir dilime sahip olduğu için seçilme olasılıkları daha yüksektir.
3. Rulet tekeri döndürülür ve rulet tekeri üzerindeki sabit noktaya denk gelen birey seçilir. Kodlama kısmında rastgele bir sayı üretilir ve bu sayının rulet tekerinde denk geldiği dilim seçilir.
4. Seçilen birey bir sonraki nesli oluşturmak için ebeveyn olarak çiftleşme havuzuna alınır.

Şekil 4.21'de bireylerin uygunluk değerine göre seçildiği rulet tekeri yöntemi gösterilmiştir.

Popülasyon	Uygunluk Değeri
Birey 1	0.1
Birey 2	0.4
Birey 3	0.3
Birey 4	0.2



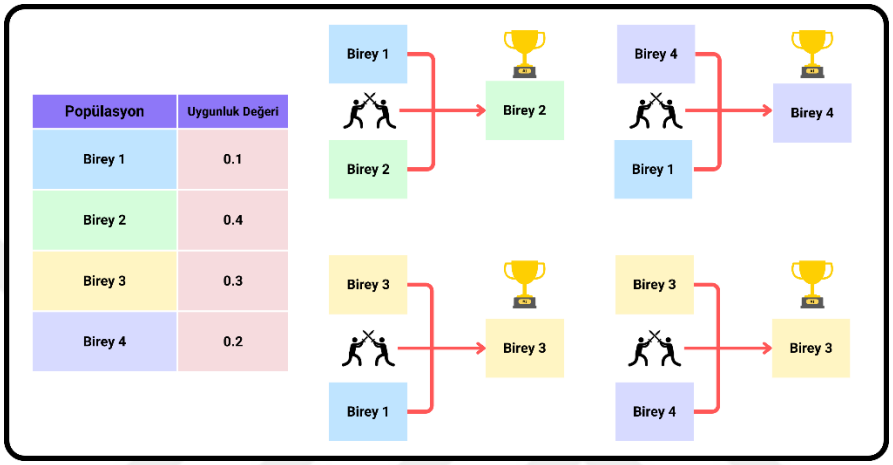
Şekil 4.21. Rulet Tekeri Yöntemi

Şekil 4.21'i inceleyecek olursak bireyler şekilde gösterildiği gibi uygunluk değerleri ile doğru orantılı olacak şekilde rulet tekerinde dilim büyüklüğüne sahiptir. Birey 2'nin uygunluk değeri en yüksek olduğu için rulet tekerinde en büyük dilime sahipken, en düşük uygunluk değerine sahip olan Birey 1 en küçük dilime sahiptir. Rulet tekeri şekilde gösterilen ok yönünde döndürülür ve durduğunda şekildeki sabit noktaya denk gelen dilimdeki birey ebeveyn olarak çiftleşme havuzuna aktarılır. Bu şekilde popülasyon büyüklüğü kadar seçim işlemi yapılır. Şekilde popülasyon büyüklüğü dört olduğu için bu işlem dört kez tekrarlanır. Burada unutulmaması gereken Birey 1'in en düşük olasılığa sahip olmasına rağmen seçilebilmesidir ayrıca bir birey birden fazla kez seçilebilir. Seçilen ebeveyn bireyler çiftleşme havuzunda toplanır ve seçim sırasına göre veya rastgele olacak şekilde çiftlere ayrılıp çaprazlama işlemi uygulanarak yeni çocuk bireyler elde edilir.

Seçim işlemi için kullanılan bir diğer yöntem ise turnuva seçim yöntemidir. Turnuva seçimi adından da anlaşılabileceği gibi turnuva prensibine dayanır. Bu yöntemde popülasyon içinden alt küme seçilir ve bu alt kümedeki bireyler arasından uygunluk değeri yüksek olan birey turnuvayı kazanarak ebeveyn birey olarak çiftleşme havuzuna alınır. Popülasyonun büyüklüğüne bağlı olarak alt küme sayısı en az iki olacak şekilde belirlenir. Alt küme sayısı turnuva büyüklüğü olarak adlandırılır. Turnuvaya alınan bireylerden her zaman uygunluk değeri en yüksek olan birey galip geleceği için popülasyon içindeki bireylerden uygunluk değeri en küçük olan hiçbir zaman bir sonraki nesile aktarılmaz. Uygunluk değeri en küçük olan birey haricindeki diğer bireylerin seçilme olasılığı her zaman vardır. Turnuva büyüklüğü rastgele olarak belirlenebilir fakat turnuva sayısı popülasyon büyüklüğünü korumak için her zaman popülasyon büyüklüğü kadar olmalıdır. Turnuva seçim yönteminin işlem adımları aşağıdaki gibidir:

1. Popülasyon büyüklüğü ile aynı sayıda turnuva sayısı belirlenir. Her bir turnuva için en az iki birey rastgele olarak seçilir. Aynı turnuva için bir birey iki kez seçilmemelidir.
2. Turnuvadaki bireylerin uygunluk değerine bakılır ve uygunluk değeri en uygun olan birey seçilir. Yani turnuvadan uygunluk değeri en yüksek olan birey galip olarak çıkar.
3. Seçilen birey, ebeveyn birey olarak çiftleşme havuzunda saklanır.
4. Bu işlemler popülasyon sayısı kadar turnuva gerçekleşene kadar devam eder.

Şekil 4.22'de turnuva seçim yöntemine bir örnek verilmiştir. Şekil 4.22'yi inceleyecek olursak dört popülasyon büyüklüğü için turnuva sayısı dört olarak belirlenmiştir daha sonra turnuvalar için popülasyon içerisinden ikişer birey rastgele olacak şekilde seçilmiştir. Aynı turnuvada bir birey iki kez seçilemeyeceği için birey turnuvada kendisi ile karşılaşmayacaktır. Şekilde görüldüğü gibi turnuvalardan uygunluk değeri büyük olan bireyler galip gelmiştir.



Şekil 4.22. Turnuva Seçim Yöntemi

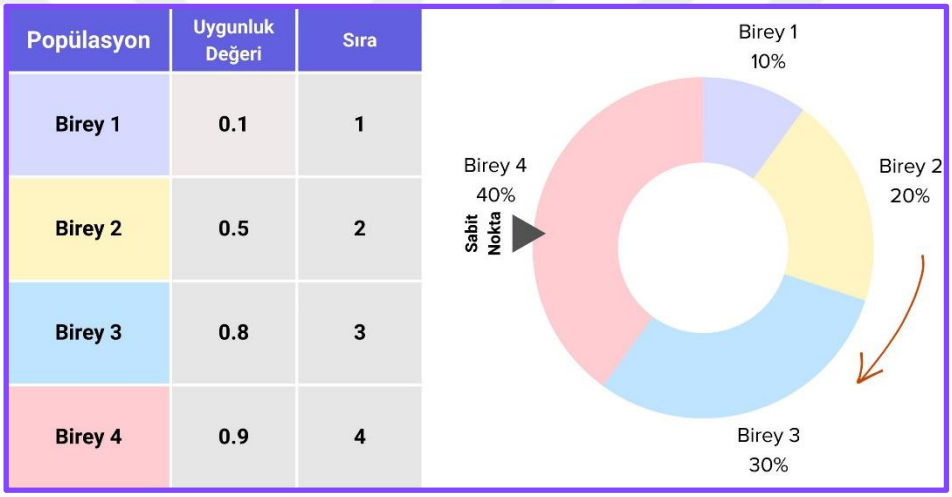
Galip gelen bireyler bir sonraki nesile ebeveyn birey olarak aktarılır. Burada dikkat edilmesi gereken nokta uygunluk değeri en küçük olan Birey 1'in hiçbir zaman turnuvadan galip gelemeyeceğidir. Şekilde Birey 1 üç kere seçilmesine rağmen hiçbir zaman galip gelememiştir. Turnuva seçim yönteminde uygunluk değeri en küçük olan bireyin hiçbir zaman sonraki nesile aktarılmayacağı unutulmamalıdır.

Seçim yöntemlerinden bir diğeri ise sıralama tabanlı seçim (rank-based selection) yöntemidir. Bu yöntemde popülasyondaki bireyler uygunluk değerlerine göre sıralanır ve her bir bireyin seçilme olasılığı bu sıra ile belirlenir. Uygunluk değeri en küçük bireyden başlanarak uygunluk değeri en büyük olan değere doğru sıralama yapılır. Daha iyi uygunluk değerine sahip olan bireyler, daha yüksek sıralamaya sahip olacağı için daha yüksek seçilme olasılığına sahip olurlar. Bu yöntemde de rulet tekerinde olduğu gibi düşük uygunluk değerine sahip bireylerin düşük olasılıkla da olsa seçilme şansı vardır. Sıralama tabanlı seçim yönteminin işlem adımları aşağıdaki gibidir:

1. Popülasyondaki her bir bireyin uygunluk değeri hesaplanır.

2. Bireyler uygunluk değerine göre sıralanır. En iyi uygunluk değerine sahip bireyler yüksek sıralamaya sahip olur.
3. Her bir bireyin seçilme olasılığı uygunluk değerine göre yapılan sıralamaya göre belirlenir. İyi uygunluk değerine sahip olan bireylerin seçilme olasılığı daha yüksek olur.
4. Rulet tekerinde olduğu gibi seçilme olasılıkları bir dairenin dilimleri şeklinde temsil edilebilir. Daire döndürülür ve sabit noktaya denk gelen birey ebeveyn birey olarak çiftleşme havuzuna aktarılır. Kodlama yaparken bu işlem rastgele sayı üretilerek yapılır.

Şekil 4.23'te sıralama tabanlı seçim yöntemine bir örnek gösterilmektedir.

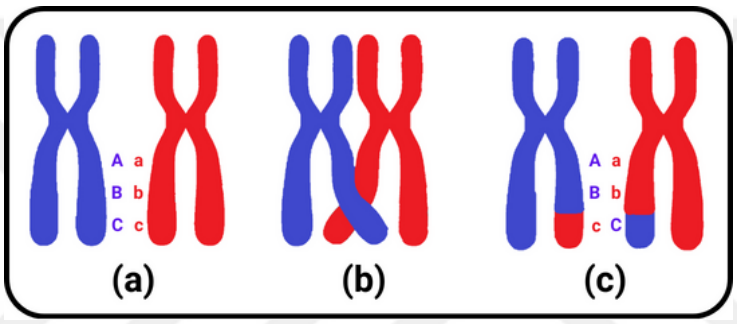


Şekil 4.23. Sıralama Tabanlı Seçim Örneği

Şekil 4.23'ü inceleyecek olursak ilk önce bireyler uygunluk değerlerine göre küçükten büyüğe doğru sıralanır. Daha sonra uygunluk değeri en küçük bireyden başlanarak sıra değeri atanır. Sıra değeri büyük olan yani uygunluk değeri büyük olan bireylerin seçilme olasılıkları şekilde de görüldüğü gibi daha yüksek olur. Bireyin seçilme olasılığını hesaplamak için bireyin sıra değeri, sıra değerleri toplamına bölünür. Daha sonra rulet yönteminde olduğu gibi daire ok yönünde döndürülür ve daire durduğunda sabit noktaya denk gelen birey seçilerek ebeveyn olarak çiftleşme havuzuna alınır. Rulet tekeri yönteminde olduğu gibi bir birey birden fazla kez seçilebilir ve uygunluk değeri düşük olan bireylerin de seçilme olasılığı vardır. Rulet tekeri yönteminden temel farkı ise rulet tekeri yönteminde uygunluk değeri birbirine yakın olan bireylerin seçilme olasılıkları da birbirine yakın olurken, sıralama tabanlı yöntem doğrudan uygunluk değerine bağlı

değildir sıralama daha ön plandadır. Örneğin Şekil 4.23'te Birey 3'ün değeri 0,8 yerine 0,89 olsaydı da bu durum seçilme olasılığını etkilemeyecekti çünkü bu değer Birey 3'ün sıra numarasını değiştirmeyecektir. Rulet tekeri yönteminde bir bireyin uygunluk değerinin değişmesi doğrudan tüm bireylerin seçilme olasılıklarını etkiler.

Çaprazlama (crossover) işleminde seçilen ebeveyn bireyler veya kromozom çiftlerinin özellikleri birleştirilerek yeni çocuk bireyler veya kromozomlar elde edilir. Bu işlem biyolojik evrimdeki homolog kromozomların bir araya gelerek çaprazlama işlemi ile belli genlerini değiştirerek yeni kromozomlar üretme işlemine benzer. Biyolojik evrimdeki çaprazlama işlemi Şekil 4.24'te verilmiştir.

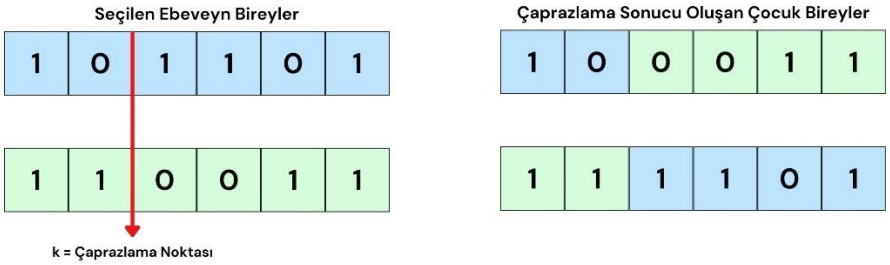


Şekil 4.24. Kromozomların Çaprazlanması

Şekil 4.24a'da homolog kromozomlar bir araya gelir, Şekil 4.24b'de homolog kromozomlar çaprazlanır ve Şekil 4.24c'de görüldüğü üzere genetiği farklı yeni kromozomlar oluşur. Bu biyolojik evrimde genetik çeşitliliği sağlarken GA'da var olan çözümlerden yeni çözümler elde edilmesini sağlar. Genetik algoritmada, çaprazlama işlemi ile ebeveyn bireyler belirli genlerini değiş tokuş eder ve yeni çocuk bireylerin oluşmasını sağlar. İkili kodlama ile çalışırken çaprazlama işlemi tek noktalı veya çok noktalı olarak gerçekleştirilebilir. Tek noktalı çaprazlamada çaprazlama noktası rastgele bir biçimde 1 ile G-1 arasında bir nokta olarak seçilir. Burada "G" bireyin gen sayısını temsil etmektedir. Çaprazlama noktası belirlendikten sonra Şekil 4.25'te olduğu gibi seçilen ebeveyn bireylerin karşılıklı genleri yer değiştirerek yeni çocuk bireyler elde edilir. Bu işlem biyolojik evrimdeki krossing over işlemine benzerdir bu yüzden çaprazlama noktası "k" ile temsil edilecektir.

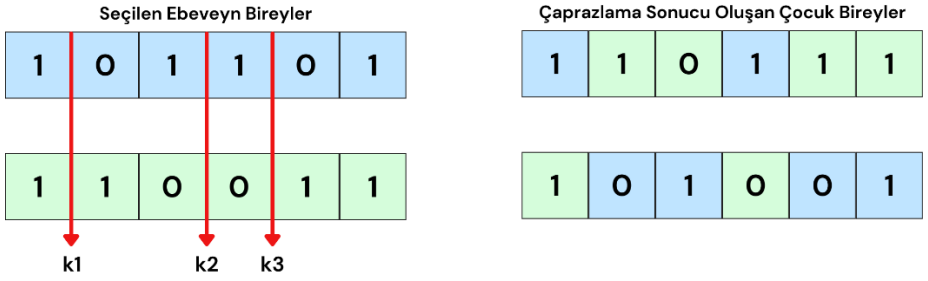
Şekil 4.25'te çaprazlama noktası (k=2) iki olarak belirlenmiştir. Daha sonra bireylerin çaprazlama noktasından önceki genleri korunarak çaprazlama noktasından sonraki

genleri yer değiştirmiştir. Bu işlem sonucunda yeni çocuk bireyler yani yeni çözümler elde edilmiştir.



Şekil 4.25. Tek Noktalı Çaprazlama

Çaprazlama işlemi tek noktalı olabileceği gibi çok noktalı da olabilir. Çok noktalı çaprazlama işleminde çaprazlama noktası sayısı kadar 1 ile G-1 arasında rastgele sayı üretilir. Genellikle bireylerin gen sayısı arttıkça çaprazlama noktası sayısınıda artırmamız gerekir. Şekil 4.26'da çok noktalı çaprazlama örneği verilmiştir.



Şekil 4.26. Çok Noktalı Çaprazlama

Şekil 4.26'da çok noktalı çaprazlama işlemi görülmektedir. Çaprazlama noktaları rastgele olarak 1,3 ve 4 olarak belirlenmiştir. 6 adet gen olduğu için burada çaprazlama noktası maksimum 5 olarak belirlenebilirdi. Çaprazlama noktası belirlendikten sonra ilk kısım sabit tutularak ikinci kısımdaki genler yer değiştirir. Daha sonra üçüncü kısım sabit tutulur ve dördüncü kısımdaki genler yer değiştirir. Çaprazlama noktası artırılırsa işlem bu şekilde devam eder.

Bu çaprazlama işlemleri ikili kodlama için uygulanabilir. Reel sayı ile kodlama yaparken farklı bir yöntem kullanmamız gerekir. Bunun için BLX-α (Blended-α Crossover) çaprazlama yöntemi kullanılır. Bu yöntem Eshelman ve Schaffer (1993) tarafından önerilmiştir. BLX-α ile çocuk bireyler şu şekilde üretilir (Takahashi ve Kita, 2001):

1) Popülasyondan x^1 ve x^2 olmak üzere rastgele iki birey seçilir.

2) Çocuk birey vektörü x^c'nin her bir elemanının x_i^c değeri, düzgün dağılama göre $[X_i^1, X_i^2]$ aralığından rastgele seçilir. Burada:

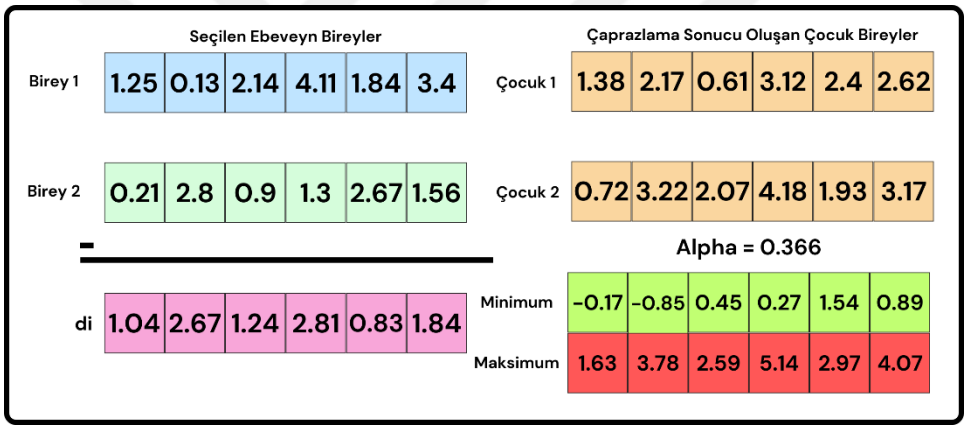
$$X_i^1 = \min(x_i^1, x_i^2) - \alpha d_i \quad (4.14)$$

$$X_i^2 = \max(x_i^1, x_i^2) + \alpha d_i \quad (4.15)$$

$$d_i = \left| x_i^1 - x_i^2 \right| \quad (4.16)$$

denklemlerdeki x_i^1 ve x_i^2 sırasıyla x^1 ve x^2'nin i. elemanlarıdır ve α pozitif bir parametredir.

Eshelman ve Schaffet çalışmalarında α değerini 0,5 olarak kullanmışlardır (Takahashi ve Kita, 2001). Takahashi ve Kita (2001) çalışmalarında α değerini 0,366 olarak almışlardır ve deneylerinde daha iyi sonuçlar elde etmişlerdir. Şekil 4.27'de BLX-α çaprazlama yöntemi kullanılarak yapılan bir reel sayı kodlama çaprazlama örneği verilmiştir.



Şekil 4.27. Reel Sayı Kodlama için BLX-α Yöntemi ile Çaprazlama

Şekil 4.27'de seçilen ebeveyn bireyler kullanılarak Denklem 4.16'daki formülle d_i değeri hesaplanmıştır. Daha sonra d_i değeri Denklem 4.14'de yerine konularak her bir gen için minimum değer, Denklem 4.15'te yerine koyularak ise her bir gen için maksimum değer hesaplanmıştır. Her bir gen için minimum ve maksimum değerler arasından rastgele bir sayı belirlenerek çocuk bireylerin genleri oluşturulmuştur. Örnekte α değeri Takahaski ve Kita'nın çalışmalarında kullandığı 0,366 olarak kabul edilmiştir.

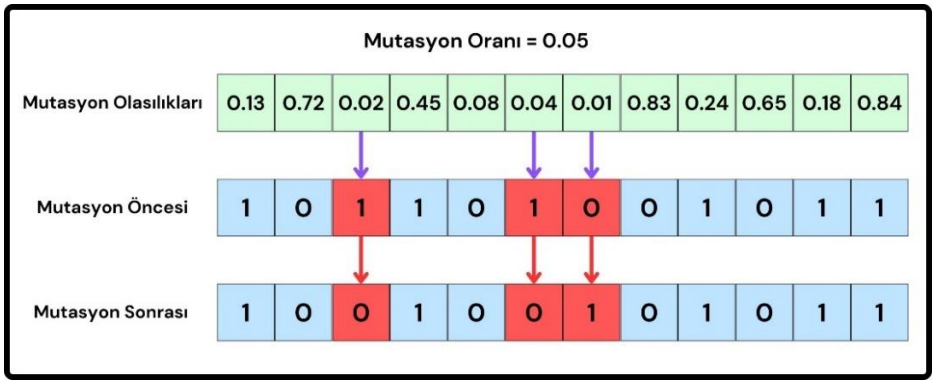
Genetik algoritmada mutasyon işlemi, bireyin genlerine belirli bir olasılık ile rastgele değişiklik uygulama işlemidir. Mutasyonda amaç bireylerin çeşitliliğini artırmak ve yerel minimum veya maksimuma takılmayı önlemektir. Mutasyon işlemi ile yeni çözümlerin keşfedilmesi sağlanır. Mutasyon işlemi popülasyon içerisindeki genetik çeşitliliği devam

ettirir ve algoritmanın bir alan içinde sıkışmasını önler. Evrim sürecinde zayıf bireyler eleneceği için bazı genler kaybolur ve bireylerin birbirlerine çok yaklaşmaları da genetik çeşitliliği azaltır. Mutasyon işlemi genellikle çaprazlamadan sonra her bir bireyin her bir geni için gerçekleştirir. Her bir gen düşük bir olasılık ile mutasyon işlemine uğrar.

Mutasyon işlemi genellikle şu adımlar ile gerçekleştirilir:

1. Mutasyon işlemine uğrayacak birey seçilir.
2. Bireyin her bir geni için rastgele bir sayı atanır eğer bu sayı mutasyon olasılığından küçükse o gen mutasyona uğrar.
3. Mutasyon işlemi genellikle, belirli bir gen veya genlerin yer değiştirilmesi, bit düzeyinde ters çevirme, gene rastgele bir sayı ekleme veya çıkarma gibi işlemleri içerir.
4. Yeni oluşturulan bireyler, mutasyona uğramış halleriyle birlikte popülasyona eklenir.

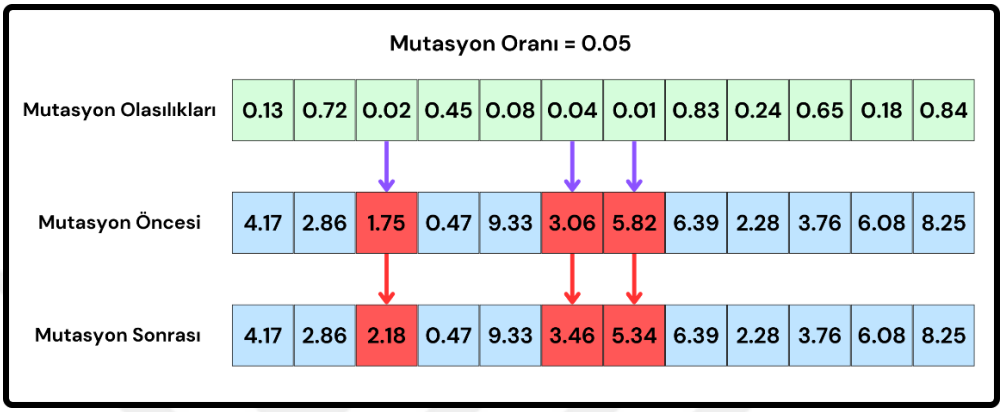
İki kodlama yapılırken genellikle bit düzeyinde tersleme işlemi yapılarak mutasyon işlemi gerçekleştirilir. Bireyin tüm genlerine [0,1] aralığında rastgele bir sayı atanır daha sonra bu sayı mutasyon oranından küçük ise bireyin o geninin değeri "1" ise "0" veya "0" ise "1" yapılır. Şekil 4.28'de bit düzeyinde yapılan mutasyon işlemi örneği gösterilmektedir.



Şekil 4.28. Bit Düzeyinde Mutasyon İşlemi

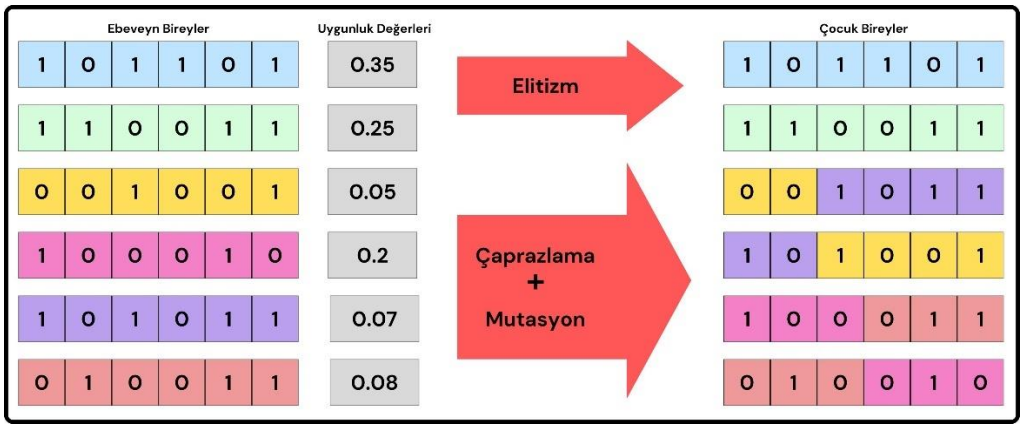
İki kodlamada binary sayının tersi alınarak mutasyon işlemi yapılabileceği gibi genlerin yerini değiştirme, geni kaydırma, belli bir gen parçasının tersini alma veya karıştırma gibi işlemler yapılabilir. Reel sayılarla kodlamada mutasyon işlemi yapılırken genellikle o gene rastgele bir sayı değeri eklenir veya çıkarılır. Burada da bu işlem her bireyin her bir geni için yapılır. Şekil 4.29'da reel sayı kodlaması için mutasyon işlemi örneği verilmiştir.

Seçim, çaprazlama ve mutasyon GA'nın temel adımlarıdır. Bunlara ek olarak elitizm işlemi de uygulanabilir. Elitizm işleminde, popülasyondaki uygunluk değeri en iyi olan belirli sayıdaki ebeveyn bireyler bir sonraki nesile doğrudan aktarılır. Bu ebeveyn bireyler çaprazlama ve mutasyon işlemlerine uğramadıkları için iyi özellikleri bozulmadan bir sonraki nesile aktarılmış olurlar.



Şekil 4.29. Reel Sayı Kodlamada Mutasyon İşlemi

Şekil 4.30'da elitizm işlemi gösterilmektedir. Her neslin en iyi bireyleri bir sonraki nesil için yeni başlangıç noktası olarak kullanılır. Bu işlem genellikle GA'nın çözüm kalitesini geliştirmek ve global optimum değere daha hızlı bir şekilde yakınsamasını sağlamak amacıyla yapılır. Elitizm işleminde ilk önce en iyi ebeveyn bireyler belirlenir. Daha sonra bu bireyler bir sonraki nesilde ki en kötü çocuk bireylerle yer değiştirir. Burada ki amaç popülasyonun sayısının korunmasıdır. GA'da elitizm işlemi aşırı kullanılırsa popülasyonun çeşitliliği azalacağı için yerel minimum veya maksimumlara takılabilir. Bu nedenle elitizm işlemi dengeli bir biçimde kullanılmalıdır.



Şekil 4.30. Elitizm İşlemi

4.3.3. Guguk Kuşu Arama

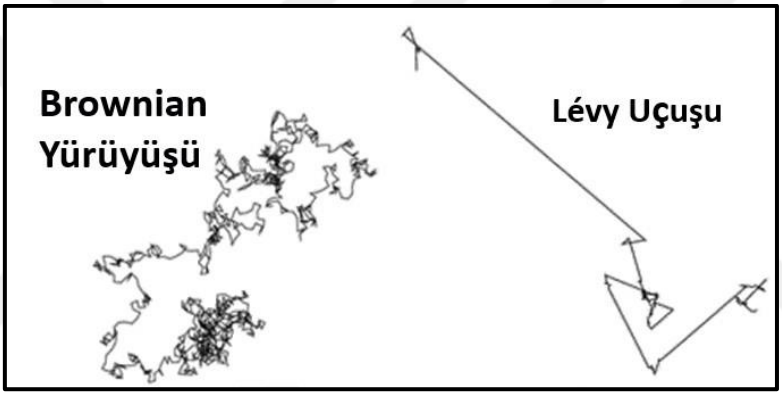
2009 yılında X. Yang ve S. Deb tarafından guguk kuşlarının üremelerinden ilham alınarak geliştirilmiştir (Yang ve Deb, 2009). Doğada birçok farklı türde kuş türü bulunmaktadır. Farklı tür dişi kuşlar farklı şekillerde, renkte ve boyutta yumurtalar bırakır. Birçok kuş türü yırtıcılardan korunmak ve yumurtalarını güvende tutmak için güvenli gördükleri yerlerde yuvalar inşa eder.

Guguk kuşlarının bazı türleri yumurtalarını konak (yabancı) kuşların yuvalarına bırakır. Dişi guguk kuşu yumurtalarını konak kuşun yumurtalarına şekil ve renk açısından benzeyecek şekilde üretir. Üremek için bu tür kurnazlığa başvuran kuşlara kuluçka paraziti denir. Bu tür kuşlar kendi yuvalarını yapmazlar, yumurtalarını başka bir türün yuvasına bırakırlar ve yavruların bakımını ev sahibine bırakırlar (Shebab ve diğ., 2017). Böylece guguk kuşu kendi yumurtalarının konak kuş tarafından fark edilmesini engellemeye çalışır. Şekil 4.31'de guguk kuşunun konak kuşun yuvasına yerleştirdiği benzer yumurtalar görülmektedir. Dişi guguk kuşu yumurtalarını bırakacağı yuvayı seçerken yumurtalarının, konak kuşun yumurtalarından önce çatlayacak şekilde zamanlamaya dikkat eder. Guguk kuşu yumurtası, konak kuşun yumurtasından önce çatlarsa konak kuş içgüdüsel olarak kendi yumurtalarını kör bir itişle yuvadan atar. Bunun sonucunda guguk kuşunun yavrularına sağlanan bakım ve yiyecek miktarı artar. Eğer konak kuş, yumurtaların kendisine ait olmadığını fark ederse, yabancı yumurtaları yuvadan atar ya da yuvasını terk edip başka bir yerde yeni yuva yapar. Bundan dolayı guguk kuşunun, ev sahibi yumurtaları taklit etmede daha yetenekli olması gerekirken, konak kuşun parazit yumurtayı belirleme becerilerini geliştirmesi gerekir. CS algoritması bu hayatta kalma mücadelesinden esinlenmiştir.



Şekil 4.31. Guguk Kuşunun Yerleştirdiği Yumurtalar (URL-11)

CS algoritmasında yeni guguk kuşları üretilirken Lévy uçuşundan faydalanılır çünkü birçok hayvan ve böceğin uçuş davranışları Lévy uçuşunun tipik özelliklerini sergiler. Lévy uçusu, tesadüfi olarak değişen birer adımın boyutunu ve yönünü karakterize eden bir süreçtir. Bu standart rastgele yürüyüşlerden örneğin Brownian hareketinden farklıdır. Şekil 4.32'ye bakacak olursak Lévy uçuşu, standart normal bir dağılımdan daha yavaş bir şekilde azalan ancak zaman zaman uzun mesafeler kat eden olağandışı büyük adımlar içerirken Brownian yürüşü, keskin ve ani dönüşler içermeyen kısa adımlar içeren bir süreçtir. Her ikisi de rastgele hareketi modelleyen iki farklı matematiksel yöntemdir. Brownian hareketinin adım uzunlukları genellikle normal (Gaussian) dağılıma uyarken, Lévy uçuşunun adım uzunlukları Lévy dağılımna uyar.



Şekil 4.32. Brownian Yürüyüşü ve Lévy Uçuşu (Amirsadri ve diğ., 2018)

Normal rastgele yürüyüşlerle karşılaştırıldığında, Lévy uçuşları geniş ölçekli arama alanlarını keşfetmede daha verimlidir. Bunun temel nedeni Lévy uçuş varyanslarının normal rastgele yürüyüşe göre çok daha hızlı artmasıdır. Lévy uçuşları, normal rastgele yürüyüşe kıyasla optimizasyon algoritmasının iterasyon sayısını yaklaşık 4 kat azaltabilir (Mareli ve Twala, 2018).

Lévy uçuşu ile yeni çözümlerin üretilmesi:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Lévy(\lambda) \tag{4.17}$$

$$Lévy \sim u = t^{-\lambda}, (1 < \lambda \leq 3) \tag{4.18}$$

Burada $x_i^{(t+1)}$ yeni çözümü, $x_i^{(t)}$ var olan çözümü, α adım büyüklüğünü (α>0) ve $Lévy(\lambda)$ Lévy uçusunu temsil etmektedir.

Guguk Kuşu Arama algoritmasını basit bir şekilde açıklamak için üç ideal kural kullanılır (Yang ve Deb, 2009):

1) Her guguk kuşu her seferde sadece bir yumurta bırakır ve yumurtayı rastgele seçilmiş yuvaya bırakır.
2) Yüksek kaliteli yumurtalara sahip en iyi yuvalar bir sonraki nesle aktarılır.
3) Mevcut ana yuva sayısı sabittir ve guguk kuşu tarafından bırakılan yumurta konak kuş tarafından $p_a \in [0,1]$ olasılığıyla fark edilir. Bu durumda:
   a. Konak kuş, guguk kuşunun yumurtalarını atabilir.
   b. Yuvasını terk edip yeni yuva yapabilir.

Basitlik açısından, bu son varsayım p_a oranında n yuvanın yeni yuvalarla (yeni rastgele çözümlerle) değiştirildiği şeklinde yaklaşık olarak hesaplanabilir.

CS algoritmasının çalışma adımları şu şekildedir:

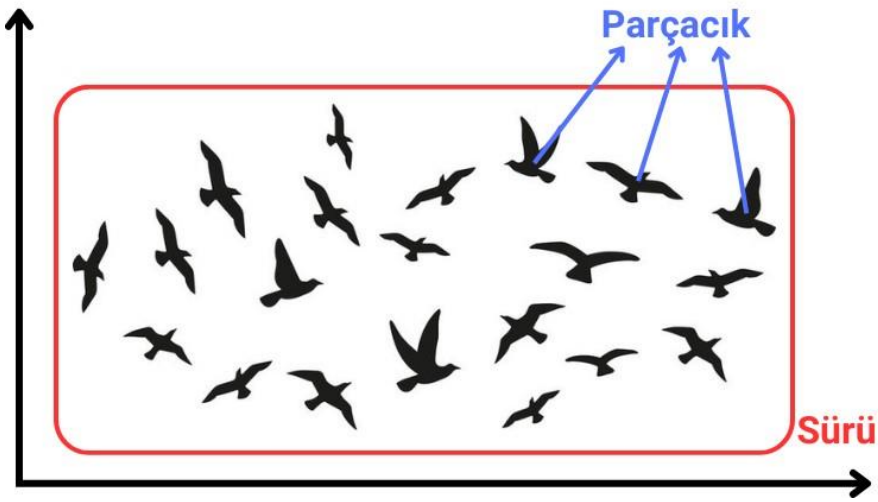
1. n adet yuva için başlangıç popülasyonu oluşturulur.
2. n adet yuva içinden en iyi çözüm hafızada tutulur.
3. Lévy uçuşu ile yeni guguk kuşları oluşturulur.
4. Oluşturulan guguk kuşları var olan guguk kuşlarından iyi ise yer değiştirilir.
5. Guguk kuşu yumurtalarının konak kuş tarafından fark edilme olasılığını (p_a) kontrol edilir.
6. Fark edilen yumurtalar yerine yeni yuvalar oluşturulur.
7. İterasyon sayısı veya optimizasyon kriteri sağlanana kadar 2. adıma geri dön.

4.3.4. Parçacık Sürü Optimizasyonu

Çok sayıda parçacığı (ajanı) basit kurallarla hareket ettirerek daha karmaşık olayları oluşturmak fikriyle bu yöntemin temelleri bilgisayar animasyonlarında doğa olaylarının taklit edilmesi sırasında görüntülerin birleştirilmesi amacıyla atılmıştır. 1983 yılında Lucasfilm'de çalışan Reeves, bulut veya patlama gibi bulanık bir objenin oluşturulmasını gerçekleştirmek için çok sayıda ajanın kullanıldığı parçacık sistemini uygulamıştır. Bir parçacık sistemi öncelikle stokastik olarak bir dizi hareketli nokta üretir, bu hareketli noktalar genellikle önceden tanımlanmış konumlara göre başlatılır. Yapılacak olan grafiksel simülasyonda bu noktaların renk, doku, sınırlı kullanım ömrü gibi ek özellikleri

olabilir. Bazı rastgele faktörlere göre hız vektörü yinelemeli olarak ayarlanır. Daha sonra her bir parçacığın bulunduğu mevcut konumundan hız vektörü alınarak hareketin gerçekçi görünmesi için gereken yerlerde kısıtlı bir açı ile ayarlanarak hareket ettirilir. Özel efektler ve gerçekçi etkileşimli grafik ortamlarının oluşturulmasında bu tür sistemler yaygın olarak kullanılmaktadır (Banks ve diğ., 2007). Daha sonra 1987 yılında Reynolds modellenen cisim açısından çok daha yüksek dereceli sürü algoritmasına temel olarak 1983 yılında Reeves'in tanımladığı parçacık sistemini kullanmıştır (Karaboğa, 2020). Reynolds bu modeline Boid (Bird-oid) adını vermiştir. Parçacık Sürü Optimizasyonu algoritmasının doğrudan kaynağı olan bu model kuşların davranışlarını simüle etmek için tasarlanmıştır (Wang ve diğ., 2018). PSO, Kennedy ve Eberhart tarafından 1995 yılında öne sürülmüş sürü tabanlı bir stokastik bir optimizasyon tekniğidir (Kennedy ve Eberhart, 1995).

Nüfusa dayalı optimizasyon algoritmaları nüfus içerisindeki iş birliği ve rekabet sayesinde çoğu zaman verimli ve etkili bir şekilde çok iyi çözümler bulabilir (Shi ve Eberhart, 1999). PSO algoritması kuşlar, balıklar, böcekler gibi sürü halinde hareket eden hayvanların sosyal davranışlarını simüle eder. Bu sürüler yiyecek bulmak için iş birliğine dayalı bir yol izler. Sürüdeki her birey, kendi öğrenme deneyimlerine ve sürüdeki diğer üyelerin öğrenme deneyimlerine göre kendi arama modelini güncellemeye devam eder. Sürü zekasına dayanan bir optimizasyon algoritmasıdır. Şekil 4.33'te PSO algoritmasının benzetim şekli verilmiştir. PSO'da her bir aday çözüme parçacık veya ajan, bütün parçacıkları içeren çözüm kümesine ise sürü veya popülasyon adı verilir.



Şekil 4.33. Parçacık Sürü Optimizasyonu Benzetimi

PSO algoritmasında her bir parçacığın istenilen çözüme yakınlığı maliyet fonksiyonu ile belirlenir. PSO'da her bir parçacık kendi deneyimi ve sürü içerisindeki en iyi parçacığın deneyimi ile kendini güncelleyerek en iyi değere ulaşmaya çalışır. Bunun için algoritma her iterasyonda parçacığın en iyi değerini ve sürünün en iyi değerini günceller. Her bir parçacığın kendini güncellemesi için hız aşağıdaki gibi hesaplanır:

$$V_i^{t+1} = wV_i^t + c_1 r_1 \left(P_{best,i}^t - P_i^t\right) + c_2 r_2 (G_{best}^t - P_i^t) \quad (4.19)$$

Eşitlik 4.19'da V_i^t i. parçacığın hızını, V_i^{t+1} i. parçacığın yeni hızını, P_i^t i. parçacığın pozisyonunu, $P_{best,i}^t$ i. parçacığın en iyi değerini, G_{best}^t o ana kadar ki en iyi değere sahip parçacığın değerini, w eylemsizlik katsayısını (inertia parameter), c_1 bilişsel katsayıyı (cognitive parameter), c_2 sosyal katsayıyı (social parameter), r_1 ve r_2 ise [0,1] aralığında ki rastgele sayıyı temsil etmektedir. Eylemsizlik katsayısı ile parçacık mevcut konumu korumak ister, bilişsel katsayı ile kendi deneyimlerinden yola çıkarak pozisyonunu günceller, sosyal katsayı ile sürünün en iyisinin deneyiminden yararlanarak pozisyonunu günceller. Çözümün bir noktaya yoğunlaşması (yerel minimuma veya maksimuma takılması) r_1 ve r_2 rastgele sayıları ile engellenir. Daha sonra Eşitlik 4.20'de verilen formül kullanılarak parçacığın pozisyonuna hız değeri eklenir ve yeni çözüm elde edilir.

$$P_i^{t+1} = P_i^t + V_i^{t+1} \quad (4.20)$$

Burada P_i^{t+1} i. parçacığın yeni değerini temsil etmektedir. Parçacığın hız değeri Şekil 4.34'te gösterildiği gibi eylemsizlik, bilişsel bileşen ve sosyal bileşen olmak üzere üç kısımdan oluşur.



Şekil 4.34. Parçacık Sürü Optimizasyonu Bileşenleri

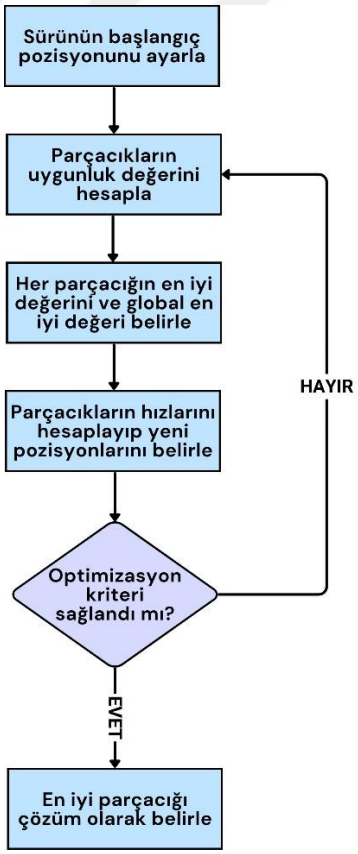
PSO her iterasyonda parçacığın en iyi değerini ve global en iyi değeri kullanarak hızını günceller. Belirlenen hız değeri parçacığın konumuna eklenerek parçacığın yeni konumu belirlenir. PSO algoritmasının sözde kodu Şekil 4.35'te verilmiştir.

PSO Algoritması Sözde Kodu

```
for each zaman adımı t do
    for each sürüdeki her parçacık i do
        denklem 4.19 ve 4.20'yi kullanarak P pozisyonunu güncelle
        parçacığın uygunluk değerini hesapla
        Pbest ve Gbest değerlerini güncelle
    end for
end for
```

Şekil 4.35. PSO Algoritması Sözde Kodu

Parçacığın hızı çok fazla artarsa çözüm problem alanının dışına çıkacağı için parçacık hızlarının alt ve üst limitlerini belirleyip sınırlamak gereklidir. Eberhart ve ark. her parçacığın hızını $[-V_{max}, V_{max}]$ aralığı ile sınırlanmasını ve bu V_{max} değerinin parçacığın maksimum konum değerinin 0,1 ile 1,0 katı arasında olarak belirlenmesini önermişlerdir (Banks ve diğ, 2007). Bu sınırlama yapılmazsa Denklem 4.19'daki r_1 ve r_2 rastgele ağırlık değerleri hızların artarak sonsuza, dolayısıyla parçacık konumlarının da sonsuza yaklaşmasına sebep olur.



Şekil 4.36. Parçacık Sürü Optimizasyonu Akış Diyagramı

PSO algoritmasının işlem adımları Şekil 4.36'daki akış diyagramında gösterildiği gibi optimizasyon kriteri sağlanana kadar devam eder. Sürünün başlangıç hızları sıfır olarak kabul edilebileceği gibi başlangıçta rastgele hız değerleri de atanabilir. Denklem 4.19'da bulunan bilişsel katsayı c_1 ve sosyal katsayı c_2 genellikle 2 olarak belirlenir. Optimizasyon durdurma kriteri olarak belirli bir iterasyon sayısı, algoritmanın çalışması süresi veya yeni üretilen çözüm ile eski çözümün maliyet değerleri arasındaki farka bakılabilir. Yeni çözüm değeri ile global en iyi değer arasındaki fark ihmal edilebilecek kadar az ise algoritma durdurulabilir. Optimizasyon kriteri sağlandıktan sonra parçacıklar içerisinden en iyi parçacığın değeri çözüm olarak kabul edilir.

5. GLOBAL YOL PLANLAMA

Global yol planlama robotun başlangıç noktasından hedef noktasına kadar olan genel rotayı belirleme sürecidir. Bu yaklaşımda genel yol planlama hedeflenir yani mobil robot tüm ortamı göz önünde bulundurarak yol planlama işlemini gerçekleştirir. Mobil robot harekete geçmeden önce ortam bilgisine sahiptir ve global yol planlama algoritmasını çalıştırarak yolunu planlar ve harekete geçer. Bu yol planlama sadece bilinen engellerin olduğu ortamlarda gerçekleştirilebilir. Ortamda bilinmeyen engeller mevcut ise buna ek olarak yerel yol planlama yapılması gerekir. Global yol planlamada ortamın tamımı ele alındığı için yerel yol planlamaya göre daha fazla hesap gücü ve bellek gerektirebilir.

Bu bölümde ilk önce maliyet fonksiyonu tanımlanacaktır daha sonra maliyet fonksiyonunun uygunluk değerine dönüştürülmesinden bahsedilip mobil robotun engel tespit metodu anlatılacaktır. Son olarak global yol planlama için kullanılan metasezgisel yöntemler incelenip simülasyon sonuçları değerlendirilecektir.

5.1. Maliyet Fonksiyonu

Yol planlama probleminin amacı robotun başlangıç pozisyonu ile hedef nokta arasındaki en uygun yolu bulmaktır. En uygun yol, yolculuk için en kısa, en az zaman alan ve en az enerji gerektiren yol olabilir (Tuncer ve Yıldırım, 2012). Yol planlama probleminin amacı genellikle engellere çarpmadan hedefe ulaşılabilecek en kısa yolu belirlemektir. Bunun için algoritma ile oluşturulan yollarının maliyetlerinin hesaplanması gerekir. Bu çalışmada kullanılan maliyet fonksiyonu aşağıdaki denklemde verilmiştir:

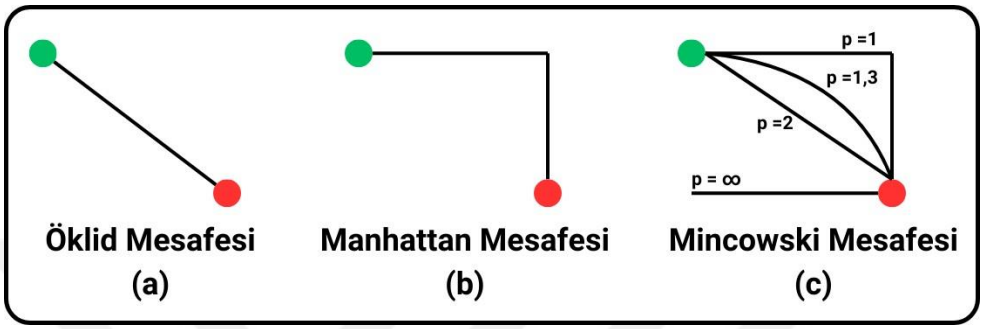
$$C = \begin{cases} \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + ceza, & engel\ varsa \\ \sum_{i=1}^{n-1} d(p_i, p_{i+1}), & aksi\ durumda \end{cases} \tag{5.1}$$

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{5.2}$$

Burada, C maliyet fonksiyonu, p_i oluşturulan çözüm noktalarından i. nokta, n problemin boyutu, d iki nokta arasındaki uzunluk, x_i ve y_i robotun mevcut konumunun yatay ve dikey pozisyonu, x_{i+1} ve y_{i+1} robotun bir sonraki konumunun yatay ve dikey pozisyonudur. Denklem 5.1 ve 5.2'de görüldüğü üzere maliyet fonksiyonu hesaplanırken

planlanan yolun uzunluğu hesaplanır. Engelle kesişen yol uygun bir çözüm olmayacağı için ceza puanı probleme uygun olarak büyük bir değer seçilmelidir.

Yukarıda yol uzunluğu hesaplanırken Öklid (Euclidean) mesafesi kullanılmıştır. Bunun dışında Şekil 5.1'de gösterilen Manhattan mesafesi ve Minkowski mesafesi de kullanılabilir.



Şekil 5.1. Mesafe Hesaplama Yöntemleri: (a) Öklid Mesafesi, (b) Manhattan Mesafesi, (c) Mincowski Mesafesi

Şekil 5.1b ve Şekil 5.1c'de verilen Manhattan mesafesi ve Mincowski mesafesi aşağıdaki denklemler ile hesaplanabilir:

$$d_{mh}(x,y) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| + |y_{i+1} - y_i| \quad (5.3)$$

$$d_{mc}(x,y) = \left(\sum_{i=1}^{n-1} (|x_{i+1} - x_i| + |y_{i+1} - y_i|)^p\right)^{\frac{1}{p}} \quad (5.4)$$

Burada, d_{mh} Manhattan mesafesini ve d_{mc} Mincowski mesafesini temsil etmektedir. Denklem 5.4'te verilen Mincowski mesafesi formülünde p kuvvet değerinin seçimi önemlidir. Şekil 5.1c'de görüldüğü üzere bu değer 1 olarak belirlenirse formül Manhattan mesafe formülüne, 2 olarak belirlenirse Öklid mesafe formülüne ve ∞ olarak belirlenirse Chebyshev mesafe formülüne dönmektedir.

5.2. Uygunluk Değeri

Maliyet fonksiyonu değerleri arasındaki farkı ortadan kaldırmak için, maliyet fonksiyonu değerleri uygunluk değerlerine dönüştürülür. Uygunluk değeri atama işleminde çözümün kendi maliyet fonksiyonu değeri ile diğer çözümlerin maliyet fonksiyonu değerleri kullanılır. Uygunluk değerine dönüştürme işlemi popülasyon tabanlı yani birden fazla

çözüm ile işlem yapılan algoritmalar için gereklidir, tek çözüm tabanlı algoritmalarda uygunluk değeri dönüşümü yapılması gerekmez.

Uygunluk değerine dönüştürme işlemi normalizasyon ile gerçekleştirilebilir. Normalizasyon işlemiyle bir verideki özelliklerin veya değişkenlerin değerleri benzer bir ölçeğe dönüştürülür. Normalizasyon bir ön işleme tekniğidir. Normalizasyonda amaç yol planlama yaparken tüm çözümlerin eşit katkı sağlamasını ve çözümdeki büyük değerlerin baskınlığı azaltmaktır. Normalizasyon işlemi yapılmazsa çözümdeki büyük değerler baskın olacağı için algoritma o noktaya takılacaktır. Bu çözüm alanını bir noktaya yoğunlaştırarak çözüm çeşitliliğini azaltacaktır.

Normalizasyon işlemi birçok farklı şekilde gerçekleştirilebilir. Genellikle min-max normalizasyon, Z-score normalizasyon ve logaritmik normalizasyon işlemleri maliyet fonksiyonu değerini uygunluk değerine çevirmek için kullanılır.

Min-max normalizasyonunda maliyet fonksiyonun minimum ve maksimum değerleri kullanılarak normalizasyon işlemi gerçekleştirilir. Min-max normalizasyon ile uygunluk değeri belirleme formülü Denklem 5.5'te verilmiştir.

$$f_i = \frac{C_i - C_{min}}{C_{max} - C_{min}} \tag{5.5}$$

Burada, f_i i. çözümün uygunluk değerini, C_i i. çözümün maliyet fonksiyon değerini, C_{min} maliyet fonksiyon değerlerinin minimumunu, C_{max} maliyet fonksiyonu değerlerinin maksimumunu temsil etmektedir. Denklem 5.5 maliyet fonksiyonu değerlerini [0,1] aralığında normalize etmektedir. Maliyet fonksiyonu değeri maksimum olan çözümün uygunluk değeri 1, maliyet fonksiyonu değeri minimum olan çözümün uygunluk değeri ise 0 olacaktır. Fakat yol planlama yaparken amacımız en kısa yolu hesaplamak yani minimizasyon problemi olduğu için denklem aşağıdaki gibi güncellenir:

$$f_i = f_{max} - (C_i - C_{min}) / (C_{max} - C_{min}) \tag{5.6}$$

Burada eklenen f_{max} değeri normalizasyon sonucunda uygunluk değerinin almasını istediğimiz maksimum değerdir yani $[0, f_{max}]$ arasında normalizasyon yapılır. Maliyet fonksiyon değeri en büyük olan değer 0, en düşük olan değer ise f_{max} olacak şekilde uygunluk değeri atanır.

Z-score normalizasyon işleminde hesaplama yaparken maliyet fonksiyonu değerlerinin ortalamasından ve standart sapma değerinden yararlanır. Z-score normalizasyonda maliyet fonksiyonunun değeri ortalamadan büyük ise uygunluk değeri pozitif, ortalamadan küçük ise uygunluk değeri negatif değer alır. Uygunluk değeri ile işlem yapılırken bu dikkate alınmalıdır. Z-score normalizasyonun formülü aşağıda verildiği gibidir:

$$f_i^z = \frac{C_i - \mu_c}{\sigma_c} \tag{5.7}$$

Burada, f_i^z i. çözümün Z-score normalizasyona göre uygunluk değerini, μ_c maliyet fonksiyon değerlerinin ortalamasını, σ_c maliyet fonksiyonu değerlerinin standart sapmasını temsil etmektedir.

Logaritmik normalizasyon da ise maliyet fonksiyonu değerlerinin on tabanında logaritması alınarak normalizasyon işlemi gerçekleştirir. Logaritmik normalizasyonun formülü aşağıda verildiği gibidir:

$$f_i^{log} = \log C_i \tag{5.8}$$

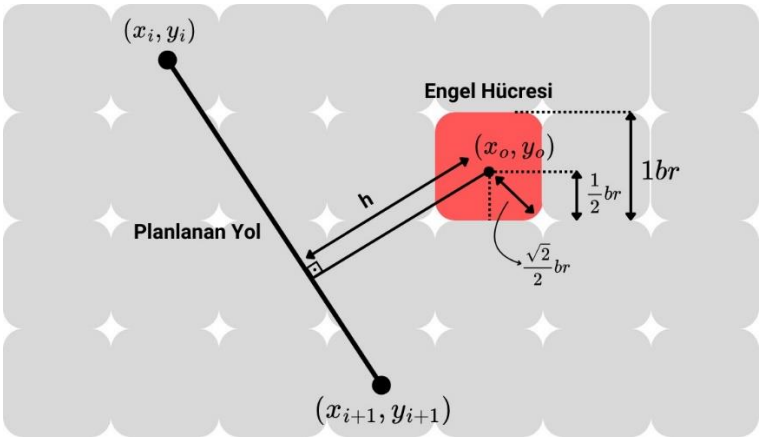
Burada, f_i^{log} i. çözümün logaritmik normalizasyona göre uygunluk değerini, C_i i. çözümün maliyet fonksiyonu değerini temsil etmektedir.

Bu tez çalışmasında Denklem 5.6'da verilen min-max normalizasyon yöntemi kullanılarak uygunluk değeri hesabı yapılmıştır.

5.3. Engel Tespiti

Maliyet fonksiyonu hesaplanırken tespit edilen her engel için ceza puanı eklenir. Bu yüzden engel tespitinin yapılması gerekir. Planlanan yolun her parçasının engelle kesişip kesişmediğinin kontrol edilmesi gerekir. Engel tespit yöntemi olarak Şekil 5.2'de gösterilen yöntemden yararlanılmıştır.

Şekil 5.2'de görüldüğü üzere planlanan yolun engele olan en kısa mesafesini hesaplamak için bir doğrunun bir noktaya olan en kısa mesafe formülünü kullanabiliriz.



Şekil 5.2. Engel Tespit Yöntemi

Bu formül Denklem 5.9'da verilmiştir.

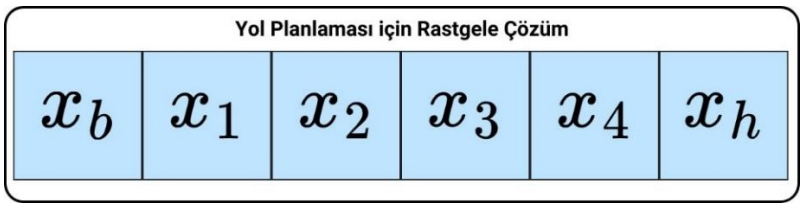
$$h = \frac{|(x_{i+1}-x_i)(y_i-y_o)-(x_i-x_o)(y_{i+1}-y_i)|}{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}} \tag{5.9}$$

Burada, x_i, y_i, x_{i+1} ve y_{i+1} planlanan yolun bir parçasının x ve y koordinatlarını, x_o ve y_o ise engelin bulunduğu hücrenin merkezinin x ve y koordinatlarını temsil etmektedir. Engel hücresinin bir kenarının uzunluğunu 1 birim olacak şekilde kabul edersek engel hücresinin merkez noktasının, engel çevresine olan en uzak mesafesinin $\sqrt{2}/2$ birim olduğu Şekil 5.2'de görülmektedir. Burada $\sqrt{2}/2$ birim değeri sınır değeridir. Eğer hesaplanan h değeri, $\sqrt{2}/2$ değerinden küçükse yol parçası engel ile kesişmektedir aksi durumda yol parçası engeli kesmemektedir. Burada tek koşul olarak $\sqrt{2}/2$ değeri kontrol edildiği için sadece bu değer artırılarak robotun engelden uzak hareket ettiği güvenli bir yol planlanabilir. Örneğin güvenli yol mesafesi engelden 1 birim uzak olacak şekilde seçilirse $\sqrt{2}/2$ değerine 1 eklenir. Bu durumda h mesafesinin $\frac{2+\sqrt{2}}{2}$ değerinden küçük olma koşulu değerlendirilir.

5.4. Benzetimli Tavlama ile Yol Planlama

Benzetimli Tavlama ile optimizasyon Bölüm 4.3.1'de anlatılmıştı. SA tek çözüm üzerinden ilerleyen bir metasezgisel yöntemdir. SA ile yol planlama yapılırken ilk adım olarak başlangıç çözümünü oluşturmak gerekir. Başlangıç çözümü problem boyutu kadar değişken içerir. Yol planlama probleminin boyutu, yol parçası sayısından bir fazladır.

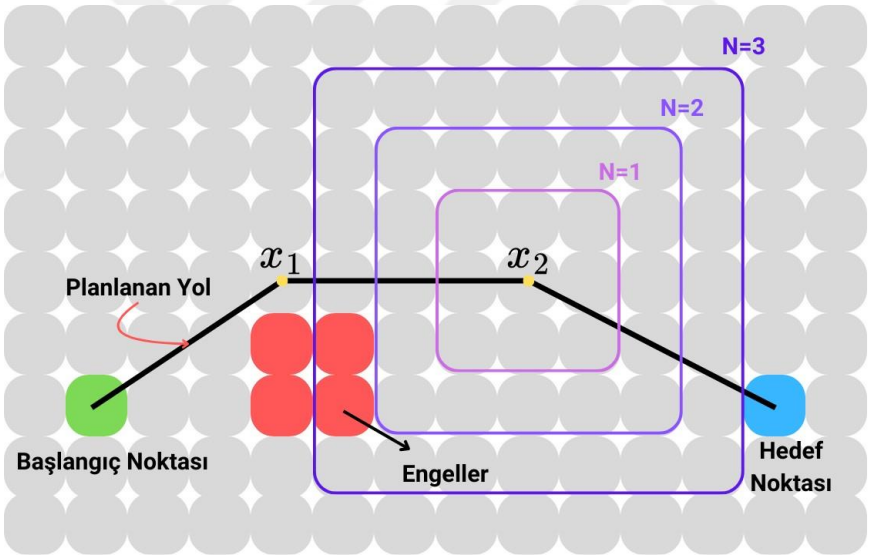
Başlangıç çözümü oluşturulurken Şekil 5.3'te gösterildiği gibi çözümün ilk değişkeni başlangıç noktası, son değişkeni ise hedef noktası olmalıdır.



Şekil 5.3. Yol Planlaması için Rastgele Çözüm

Şekil 5.3'te x_b robotun başlangıç noktasını, x_h robotun hedef noktasını temsil etmektedir. Bu iki değişkenin arasında kalan ara noktalar rastgele bir şekilde oluşturulmalıdır.

Komşu çözüm oluşturmak için komşuluk hareketleri, rastgele değer ekleme gibi farklı yöntemler mevcuttur. Yol planlama için bu yöntemler dışında Şekil 5.4'te gösterilen yeni bir yöntem geliştirilmiştir.



Şekil 5.4. Yeni Çözüm Üretme Stratejisi

Bu yöntemde yeni çözüm üretilirken başlangıç ve hedef noktası hariç her düğüm noktası için rastgele N sayısı belirlenir. Değişkenin düğüm noktasının başlangıç konumunu merkez alan N sayısı büyüklüğünde bir alan oluşturulur. Düğüm noktasının yeni değeri bu alan içinde yer alan rastgele bir nokta olarak belirlenir.

SA'da bu yöntem ile komşu çözümler üretilmiştir. Daha sonra var olan çözüm ile komşu çözümün maliyet fonksiyonu değerleri karşılaştırılır. Komşu çözüm daha iyi ise yeni

çözüm olarak kabul edilir. Eğer komşu çözüm var olan çözümden kötü ise belirli bir olasılık ile bu çözüm kabul edilir. Bu olasılık sıcaklığa ve maliyetler arasındaki fark bağlıdır. Bu yüzden SA'da sıcaklık çizelgesinin seçimi önemlidir. Birçok farklı soğutma çizelgesi olmasına karşın bu tez kapsamında geometrik ve logaritmik soğutma çizelgesine değinilecektir. Geometrik ve logaritmik soğutma çizelgesi denklemleri aşağıda verilmiştir:

$$T_t = T_b \alpha^t \tag{5.10}$$

$$T_t = \frac{c}{\log(1+t)} \tag{5.11}$$

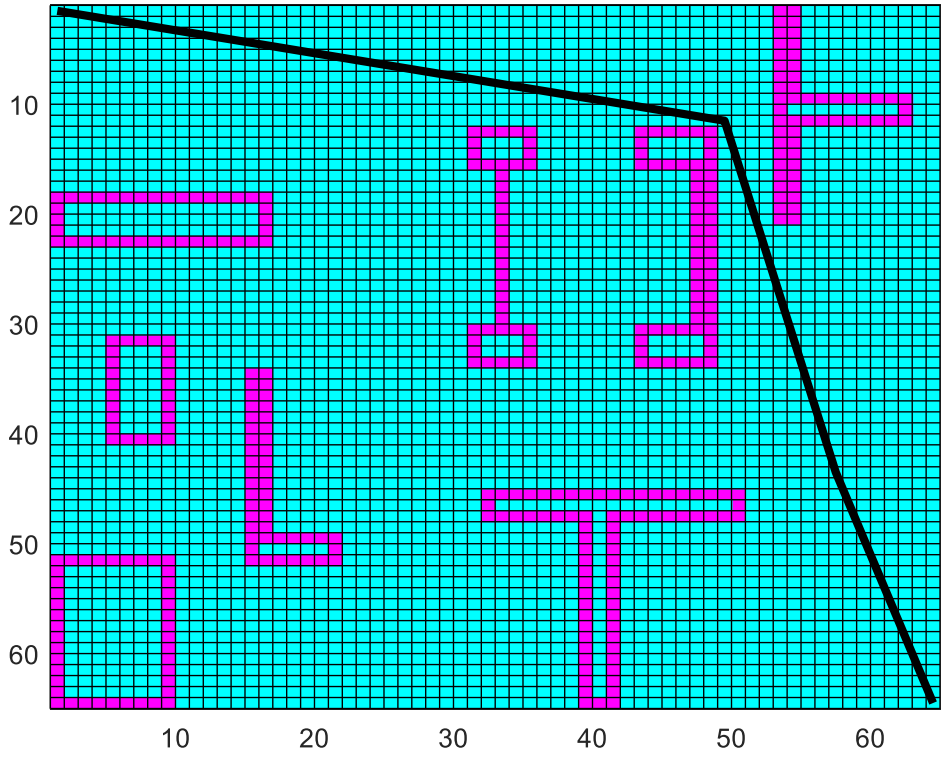
Denklem 5.10'da geometrik soğutma çizelgesinin formülü verilmiştir burada, T_b başlangıç sıcaklığını, α ise 0 ile 1 aralığındaki sabit bir sayıyı temsil etmektedir. Sıcaklığın yavaş bir şekilde azalması için α sabitinin 1'e yakın bir değer seçilmesi önemlidir.

Denklem 5.11'de logaritmik soğutma çizelgesinin formülü verilmiştir burada, c pozitif sabit bir sayıyı temsil etmektedir ve bu c sabiti t'den bağımsızdır.

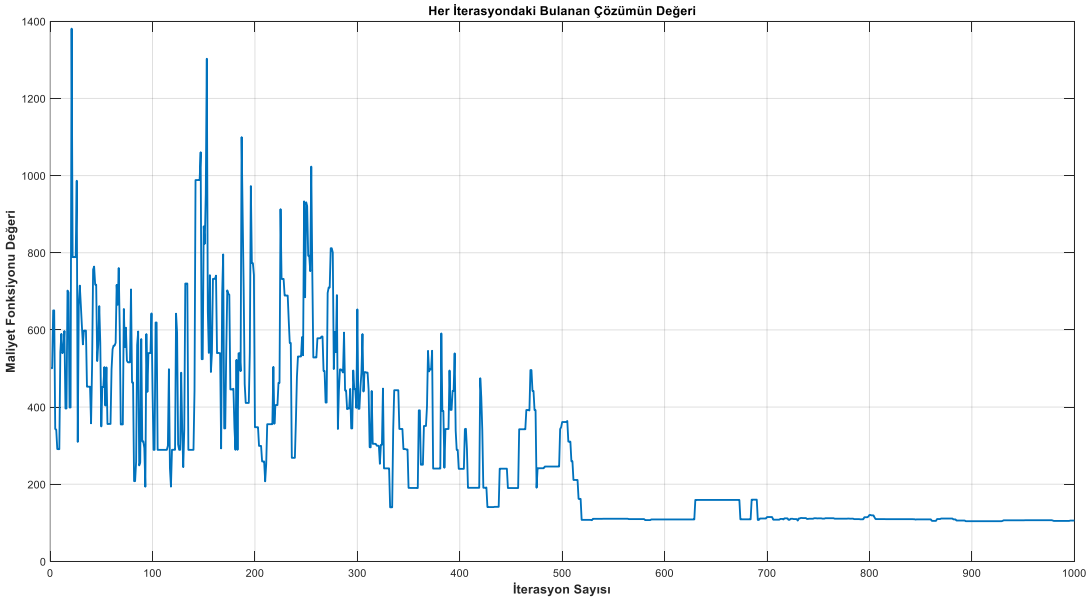
Geometrik soğutma çizelgesi, geometrik olarak azaldığı için logaritmik soğutma şemasından daha hızlıdır (Mahdi ve diğ., 2017). Bu tez kapsamında bu iki soğutma çizelgesi arasından geometrik soğutma çizelgesi tercih edilmiştir.

SA algoritması 64x64 birim haritada 1000 iterasyon değerinde çalıştırılarak aşağıdaki şekildeki sonuçlar elde edilmiştir. Başlangıç sıcaklığı 1000 olarak alınmış ve geometrik soğutma çizelgesindeki soğutma katsayısı α 0,994 olarak belirlenmiştir. Problem boyutu 4 ve ceza puanı 50 olarak alınmıştır. Harita boyutu 64x64 birim olduğu için her değişken 4096 farklı değer alabilmektir. Bu problem boyutu için olasılık sayısı yaklaşık 16.8 milyondur. Bu koşullar altında SA algoritmanın planladığı yol Şekil 5.5'te verilmiştir.

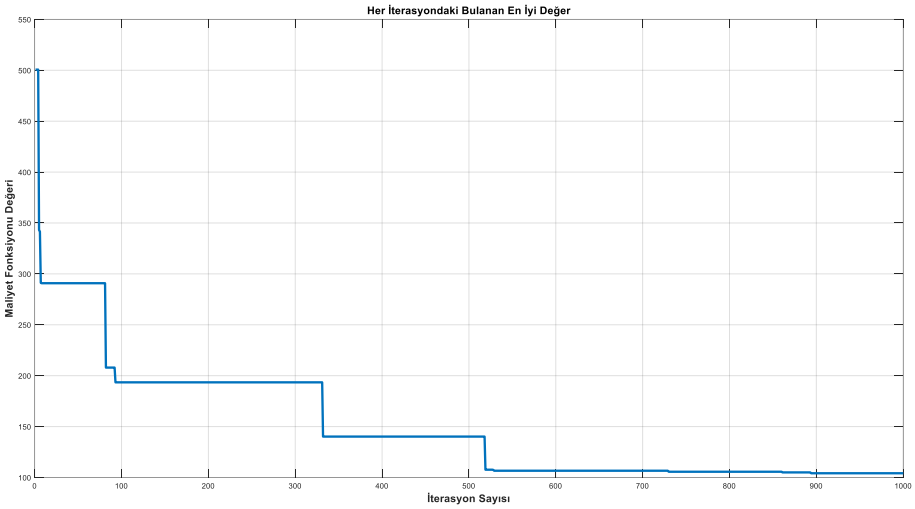
Şekil 5.6'yı inceleyecek olursak 500. iterasyona kadar maliyet fonksiyonu değerinde büyük değişiklik olduğu görülmektedir. 500. iterasyondan sonra ise maliyet fonksiyonu değerinde ufak değişiklikler görülmektedir bu Şekil 5.9'da gösterilen soğutma çizelgesi ile alakalıdır.



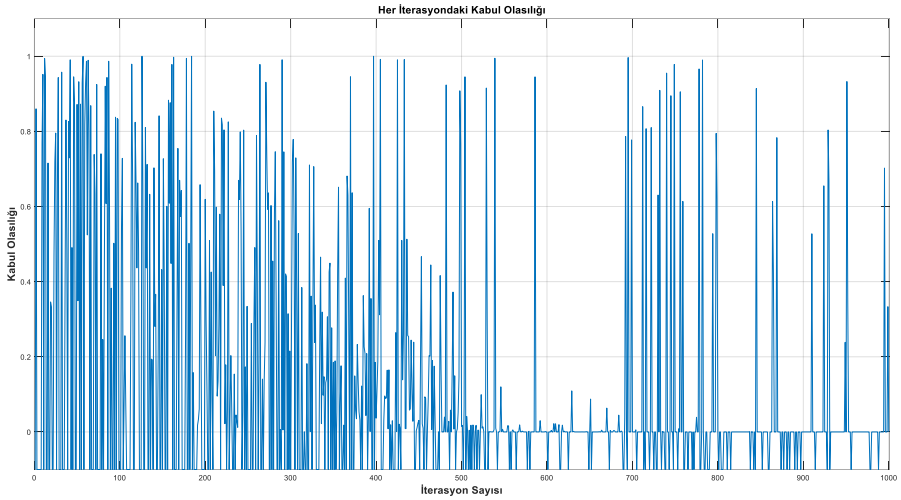
Şekil 5.5. Benzetimli Tavlama Algoritması ile Planlanan Yol



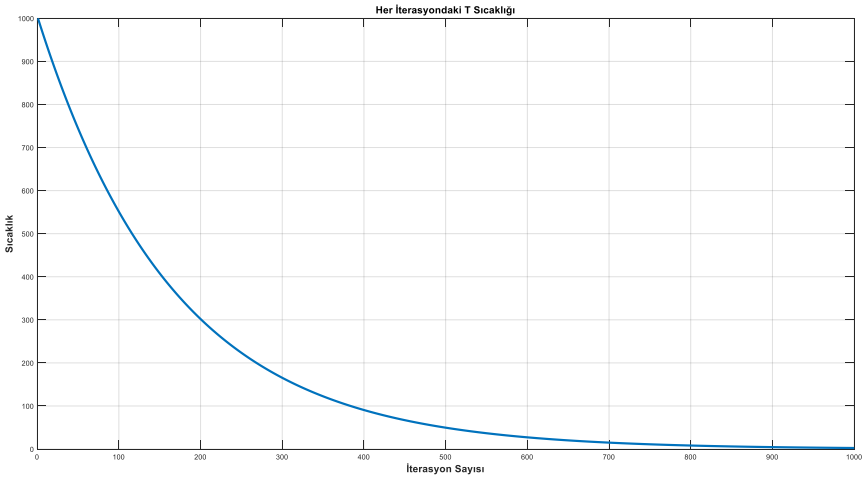
Şekil 5.6. Benzetimli Tavlama için Bulunan Çözümlerin Değerleri



Şekil 5.7. Benzetimli Tavlama için Bulunan En İyi Değer



Şekil 5.8. Benzetimli Tavlama için Her İterasyondaki Kabul Olasılığı



Şekil 5.9. Benzetimli Tavlamada Kullanılan Soğutma Çizelgesi

Sıcaklık değeri yüksekken kötü komşu çözümlerin kabul edilme olasılığı yüksek olduğu için bu şekilde bir sonuç elde edilmiştir. Şekil 5.8'de gösterilen her iterasyon için kabul olasılığı grafiğinden de bu sonuç çıkarılabilir. Şekil 5.8'de gösterilen negatif değerler kabul olasılığına bakılmaksızın yeni çözüm olarak kabul edilen daha iyi değerlerdir. Son olarak Şekil 5.7'yi inceleyecek olursak 500. iterasyondan sonra bulunan en iyi değer de büyük bir değişiklik görülmemektedir. SA algoritması yaklaşık 900. iterasyonda bulabildiği en iyi değeri bulmuş ve bu adımdan sonra yol planlamasında iyileştirme yapmamıştır.

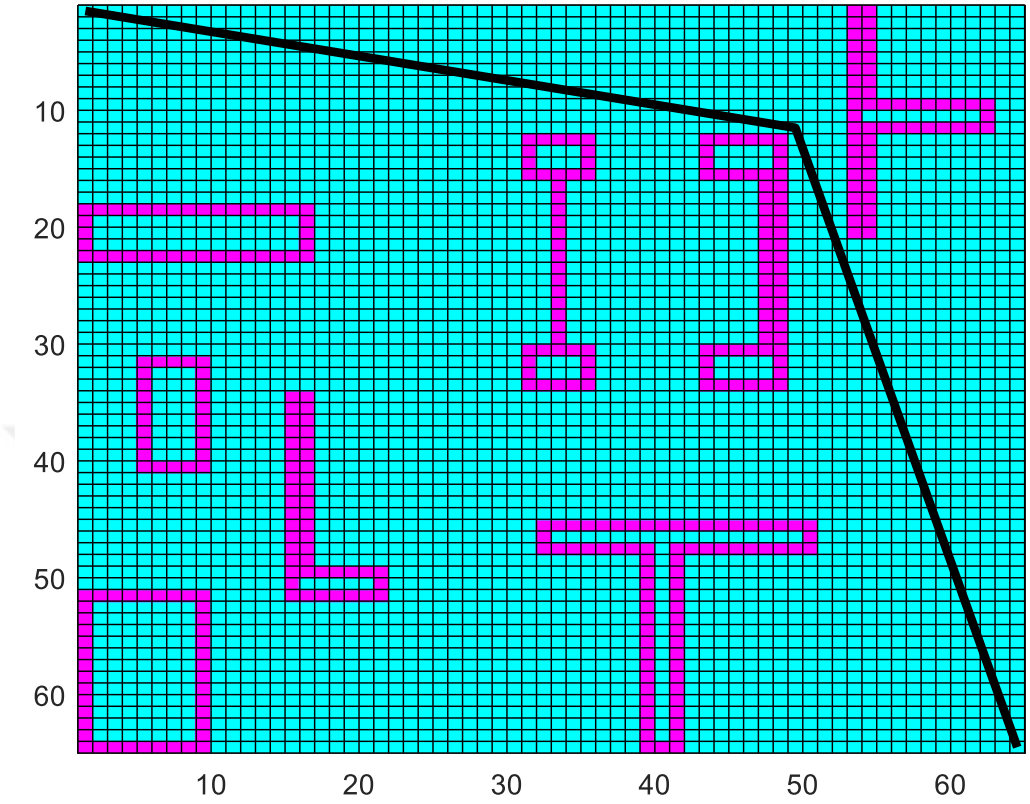
5.5. Genetik Algoritma ile Yol Planlama

Genetik Algoritma ile optimizasyon Bölüm 4.3.2'de anlatılmıştı. GA temel olarak seçim, çaprazlama ve mutasyon işlemlerinden oluşur. GA popülasyon tabanlı bir algoritmadır ve yol planlama yaparken rastgele bir şekilde başlangıç nüfusunun oluşturulması gerekir. Burada Şekil 5.3'te gösterilen robotun başlangıç ve hedef noktasının değerinin tutulduğu değişkenin sabit olduğu unutulmamalıdır. Bu noktalar mutasyon işlemine de uğramaz. Başlangıç popülasyonu oluşturulduktan sonra popülasyonun maliyet fonksiyonu hesaplanır. Seçim işlemi için maliyet fonksiyonu değerleri uygunluk değerlerine dönüştürülür. Bu tez kapmasında GA için anlatılan seçim işlemleri arasından rulet tekeri yöntemi kullanılmıştır. Popülasyonun uygunluk değeri referans alınarak rulet tekeri yöntemi ile seçim işlemi yapılmıştır. Çaprazlama için tek noktalı çaprazlama işlemi tercih edilmiştir. Mutasyon adımında ise popülasyondaki her bireyin her bir geninin mutasyona uğrayıp uğramayacağı kontrol edilir. Mutasyon işleminde de Şekil 5.4'te geliştirilen yöntem kullanılmıştır. Bu yöntem yol planlama problemi için diğer temel yöntemlerden daha uygundur. Çünkü değişkene değer ekleme veya çıkarma işlemi yapılırsa pozisyonu sadece yatay eksende değişecektir. Bu yöntem sayesinde mutasyona uğrayan noktanın hem yatay hem de dikey pozisyonu değişmektedir.

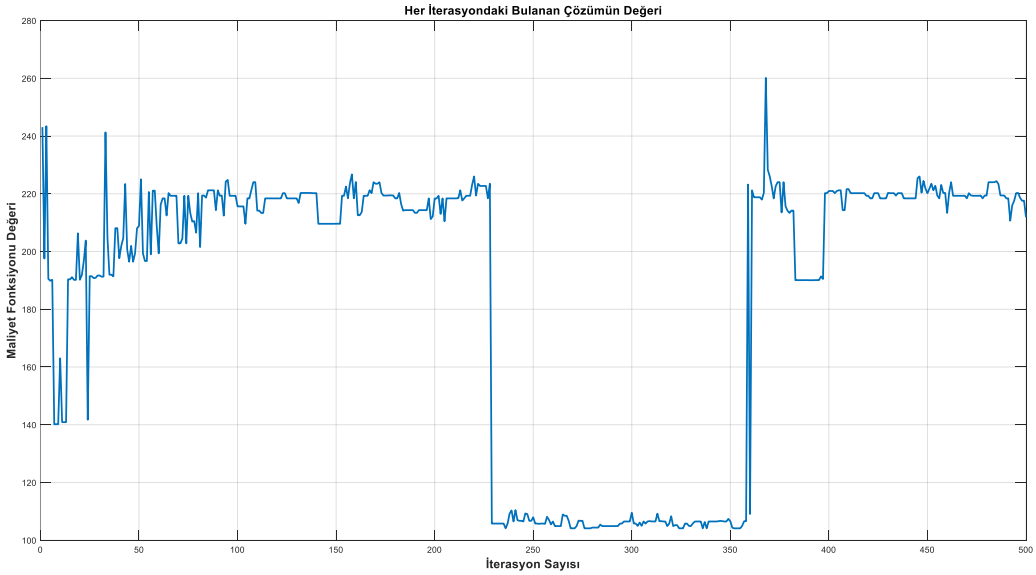
GA'da elitizm adımı opsiyoneldir ve bu tez kapsamında uygulanmamıştır. Bunun yerine her iterasyonda en iyi değer hafızada tutulmuştur. Bu şekilde hafızalı GA kullanılarak algoritma sonunda en iyi değerin kaybolması önlenir.

GA, 64x64 birim haritada 500 iterasyon değerinde çalıştırılarak aşağıdaki şekildeki sonuçlar elde edilmiştir. Popülasyon sayısı 100 ve mutasyon olasılığı 0,35 olarak

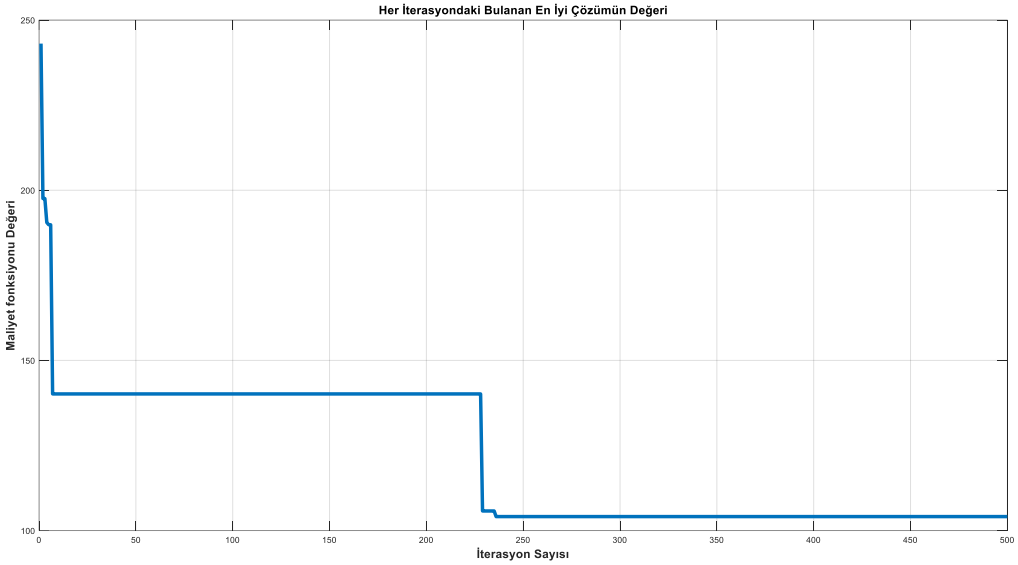
alınmıştır. Problem boyutu 4 ve ceza puanı 50 olarak alınmıştır. Bu koşullar altında GA ile planlanan yol Şekil 5.10'da verilmiştir.



Şekil 5.10. Genetik Algoritma ile Planlanan Yol



Şekil 5.11. Genetik Algoritma için Çözümlerin Değerleri



Şekil 5.12. Genetik Algoritma için Bulunan En İyi Değer

Şekil 5.11'i inceleyecek olursak grafik her iterasyon için popülasyon içindeki en iyi değeri göstermektedir ve yaklaşık 225. iterasyonda algoritma boyunca bulunan en iyi değeri bulmuştur. Grafikte maliyet fonksiyonu değerinin düzensiz olarak arttığı veya azaldığı görülmektedir. Özellikle mutasyon işlemi uygulandığında grafikte ani sıçramalar olabilir. Bu değişim her zaman pozitif yönde olmaz. Grafikten de görüldüğü üzere algoritma sonlandığında bulunan değer algoritma boyunca bulunan en iyi değer değildir. Bu yüzden elitizm kullanmak veya en iyi değeri bir değişkende tutmak algoritmanın performansı için son derece önemlidir. Şekil 5.12'de ise algoritma boyunca o iterasyona kadar olan en iyi maliyet fonksiyonu değerleri gösterilmektedir. Bu grafiğe bakıldığında algoritmanın yaklaşık 250. iterasyondan sonra arama yapmasına rağmen daha iyi bir sonuç bulamadığı görülmektedir.

5.6. Guguk Kuşu Arama ile Yol Planlama

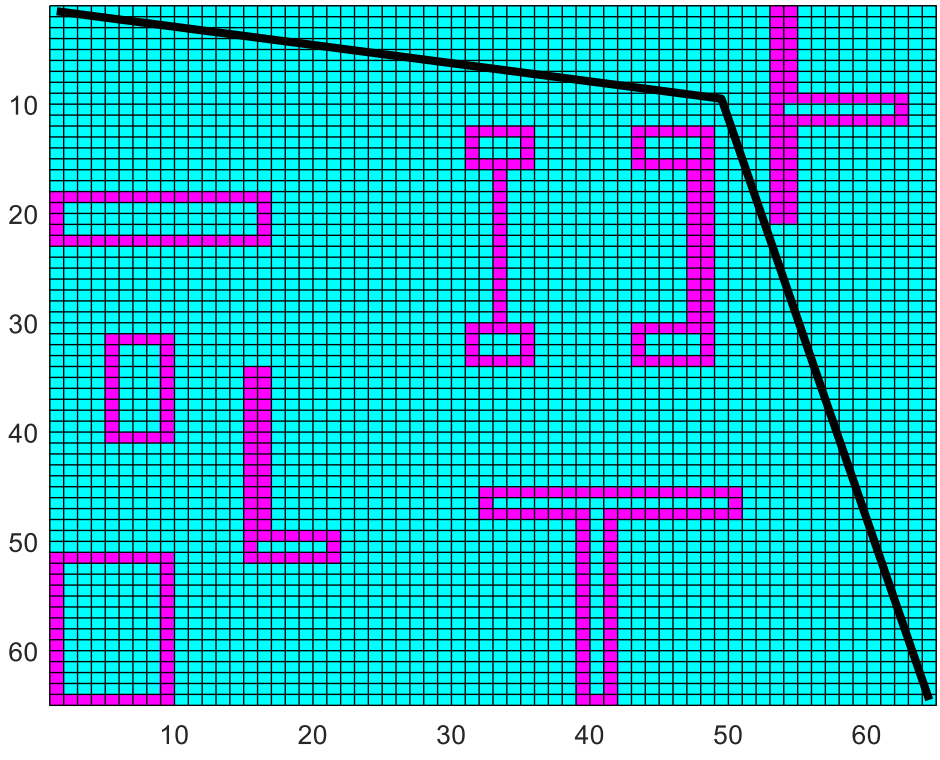
Guguk Kuşu Arama algoritması ile optimizasyon Bölüm 4.3.3'de anlatılmıştı. CS algoritması da GA gibi popülasyon tabanlı bir algoritma olduğu için başlangıç nüfusu rastgele olacak şekilde GA'ya benzer şekilde oluşturulur. CS algoritması açgözlü (greedy) bir algoritma olduğu için GA'daki gibi elitizm işlemine veya hafızalı bir sisteme ihtiyaç duymaz. CS algoritması guguk kuşlarının üreme stratejine dayanır. Guguk kuşları konak kuşun yuvasına, konak kuşun yumurtalarına benzer yumurtalar yerleştirir ve bu yumurtaların fark edilmeden o yuvada gelişmesini hedefler. Bu yüzden CS

algoritmasında fark edilme olasılığı olan p_a'nın seçimi son derece önemlidir. Fark edilen yuvalar için yeni yuvalar oluşturulduğu kabul edilmiştir.

CS algoritması ile yol planlama yaparken yeni guguk kuşları yani yeni çözümler Lévy uçuşu ile oluşturulmuştur. Lévy uçuşu kuş ve böceklerin uçuş davranışlarının tipik özelliklerine sahiptir. Lévy uçuşu ile oluşturulan rastgele adımlar ani yön değişimleri ve uzun mesafeler içerebilir. Fakat p_a olasılığı ile fark edilen yuvalar yerine yeni yuva oluşturulurken farklı bir yöntem kullanılmıştır. Bu yöntem Denklem 5.12'de verilmiştir.

$$\left(C_i^j\right)_{yeni} = C_i^j + r\left(C_{d1}^j - C_{d2}^j\right) \tag{5.12}$$

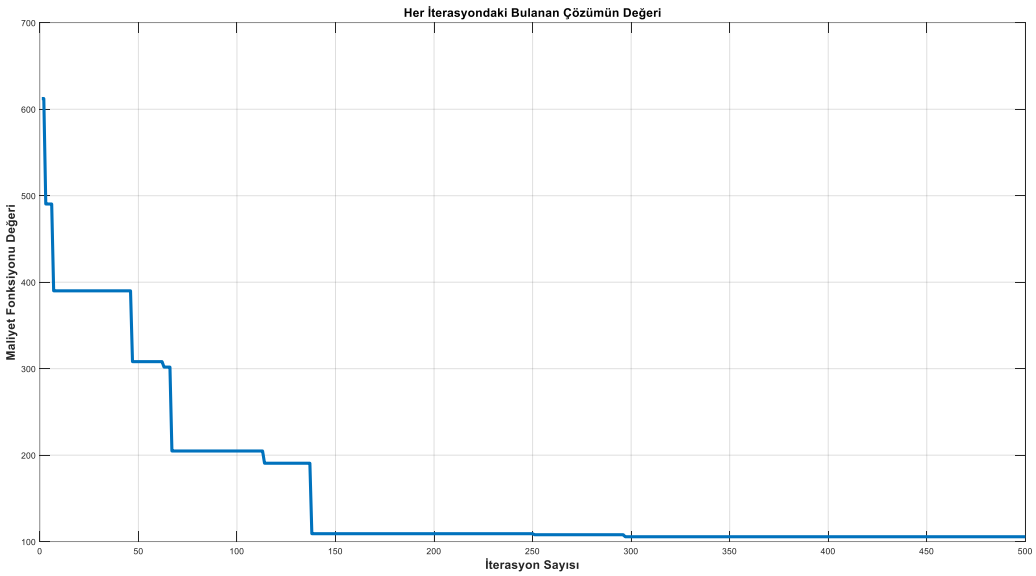
Burada, $\left(C_i^j\right)_{yeni}$ i. guguk kuşunun j. değişkeninin yeni değeri, C_i^j i. guguk kuşunun j. değişkeninin değeri, r rastgele 0 ile 1 aralığında bir sayı ve d_1 ile d_2 ise 1 ile popülasyon büyüklüğü aralığındaki rastgele bir tam sayıyı temsil etmektedir. Yani guguk kuşunun yumurtası fark edilirse yeni yuva, guguk kuşunun mevcut değeri ile popülasyon içerisinden rastgele olarak seçilen iki guguk kuşunun değerleri arasındaki fark kullanılarak oluşturulur.



Şekil 5.13. Guguk Kuşu Arama Algoritması ile Planlanan Yol

CS algoritması 64x64 birim haritada 500 iterasyon değerinde çalıştırılarak aşağıdaki şekildeki sonuçlar elde edilmiştir. Popülasyon sayısı 10, yumurtaların fark edilme olasılığı p_a 0,5, Lévy uçuşunda λ değeri 1,5 olarak alınmıştır. Problem boyutu 4 ve ceza puanı 50 olarak alınmıştır. Bu koşullar altında CS ile planlanan yol Şekil 5.13'te verilmiştir.

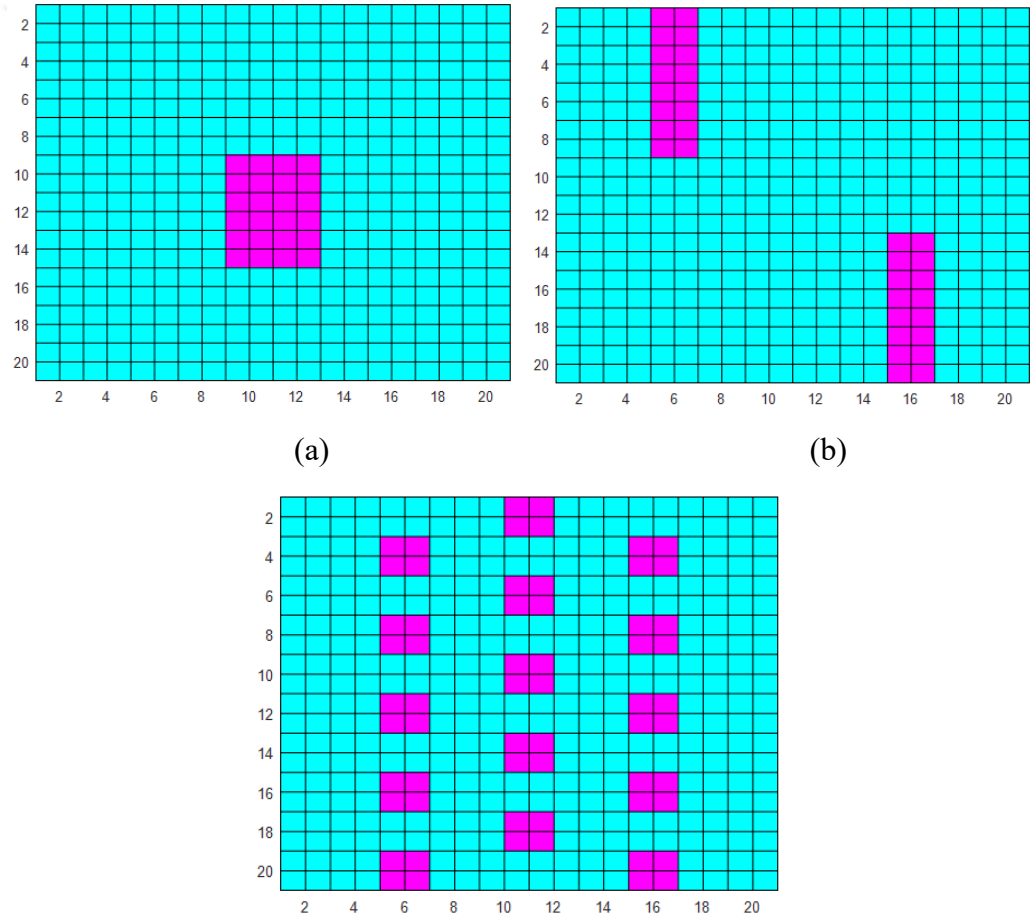
Şekil 5.14'te görüldüğü üzere CS algoritması açgözlü bir algoritma olduğu için bulunan değerler sürekli olarak iyiye doğru gitmiştir. Algoritma yaklaşık 150. iterasyonda kabul edilebilir bir değer bulmuştur. Grafik dikkatli bir şekilde incelendiğinde, algoritmanın 250. ve 300. iterasyonlarda küçük iyileştirmeler yaptığı gözlemlenmiştir.



Şekil 5.14. Guguk Kuşu Arama Algoritması için Bulunan Çözümlerin Değerleri

6. ALGORİTMALARIN KARŞILAŞTIRILMASI

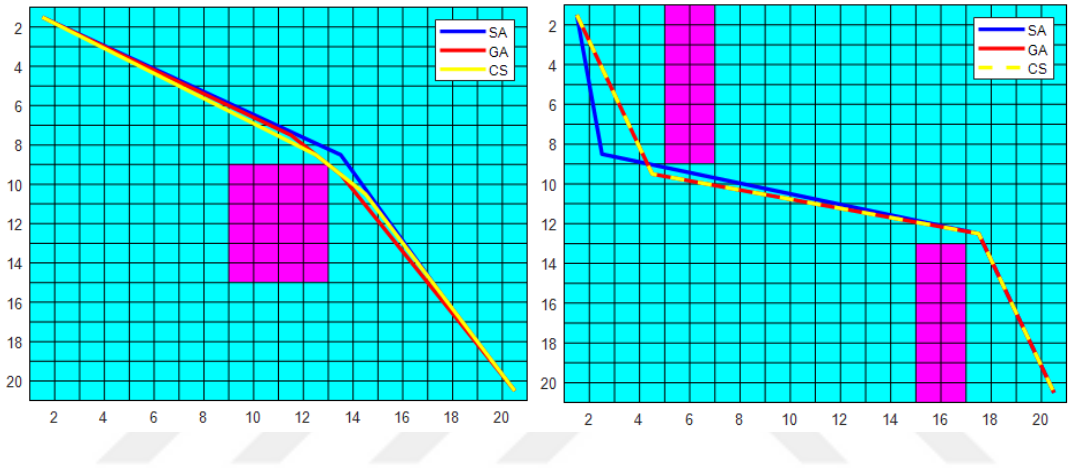
SA, GA ve CS yol planlama algoritmalarını karşılaştırmak için literatürde daha önce üzerinde çalışılmış üç farklı tipte harita seçilmiştir. Mobil robot bu haritalarda sol üst köşede başlamakta ve hedef noktası olan sağ alt noktaya en kısa yolu hesaplayarak ulaşmayı hedeflemektedir. Şekil 6.1'de görüldüğü üzere haritalar 20x20 birim boyutundadır ve farklı şekillerde engeller içermektedir. Böylelikle algoritmalar farklı harita türlerinde denenmiş, algoritmaların hangi harita türünde nasıl bir sonuç verdiği gözlemlenmiştir. Şekildeki pembe renkli hücreler engelleri, açık mavi renkli hücrelerse serbest geçiş bölgelerini temsil etmektedir.



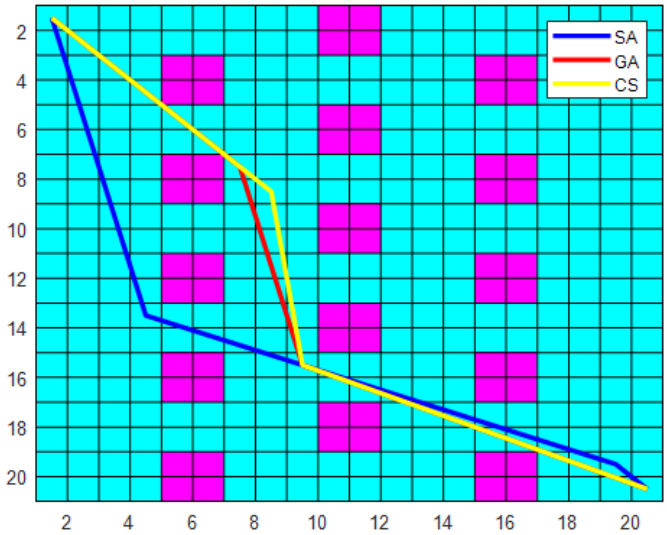
Şekil 6.1. Izgara Tabanlı Haritalar: (a) Harita 1 (b) Harita 2 (c) Harita 3

Metasezgisel yol planlama algoritmaları MATLAB 2023b ortamında kodlanmış ve yine aynı ortamda simüle edilmiştir. Kullanılan bilgisayar INTEL CORE i7 işlemciye, 16 GB RAM'e ve Windows 10 Home işletim sistemine sahiptir. Kullanılan tüm algoritmalarda

problem boyutu dört ve ceza puanı 10 olarak alınmıştır. Tüm algoritmalar 50 iterasyon değerinde 30 kere yeniden çalıştırılarak sonuçlar elde edilmiştir. GA ve CS algoritmalarının popülasyon sayısı 30 olarak alınmıştır. SA'da başlangıç sıcaklığı 1000, geometrik soğutma çizelgesindeki soğutma katsayısı α 0,84; GA'da çaprazlama olasılığı 0,95, mutasyon olasılığı 0,4; CS'de yumurtaların fark edilme olasılığı p_a 0,25, Lévy uçusunda ki λ değeri 1,5 olarak alınmıştır. Üç yol planlama algoritmasının 30 kere çalıştırılması sonucunda elde edilen optimal yollar Şekil 6.2'de gösterilmiş ve sonuçlar Tablo 6.1'de verilmiştir.



(a) (b)



(c)

Şekil 6.2. Algoritmaların Elde Ettiği Optimal Yollar: (a) Harita 1 (b) Harita 2 (c) Harita 3

Tablo 6.1. Simülasyon Sonucu Elde Edilen Sonuçlar (Ort: Ortalama, Std: Standart Sapma)

Yol Planlama Algoritmaları	Harita 1				Harita 2				Harita 3			
	Maliyet			Çalışma Süresi (sn)	Maliyet			Çalışma Süresi (sn)	Maliyet			Çalışma Süresi (sn)
	Ort.	Std.	En iyi		Ort.	Std.	En iyi		Ort.	Std.	En iyi	
SA	28,7720	0,9741	27,7857	0,0001	43,2319	8,8826	31,1392	0,0001	44,4287	11,3214	29,9390	0,0002
GA	27,7844	0,1026	27,5287	0,0072	31,5461	0,8241	30,4297	0,0076	29,8145	0,4819	28,8145	0,0084
CS	27,7365	0,0912	27,5287	0,0556	31,4482	0,8763	30,4297	0,0392	29,5977	0,3363	29,0536	0,1121

Şekil 6.2'yi incelenirse GA ve CS algoritmaları SA'ya göre daha uygun yollar planlamıştır. GA ve CS'nin Harita 1 ve Harita 2'de planladığı en iyi yolların uzunlukları yakınken Harita 3'te GA daha iyi bir sonuç vermiştir. Tablo 6.1'de görüldüğü üzere SA diğer algoritmalara oranla daha uzun yol planlamasına rağmen çalışma süresi açısından diğer algoritmalardan üstündür. GA ve CS planlanan yolun uzunluğu bakımından hemen hemen birbirlerine yakın sonuçlar vermişlerdir. GA ve CS'yi karşılaştıracak olursak GA'nın çalışması süresi CS'den daha iyi iken CS'nin standart sapma değerinin GA'dan daha iyi olduğu görülmektedir. Daha ayrıntılı bilgi edinmek için "Otonom Mobil Robotlar için Metasezgisel Yöntemler Kullanılarak Yol Planlama Algoritmasının Geliştirilmesi ve Performanslarının Karşılaştırılması" adlı bildiri incelenebilir (Tezel ve Karakaya, 2024).

Haritalar literatürdeki "Izgara Bazlı Yol Planlama için Matematik Tabanlı Metasezgisellerin Karşılaştırılması" adlı çalışmadan alınmıştır (Yıldırım ve Akay, 2022). Karşılaştırma yapılabilmesi için çalışmadaki aynı haritalar kullanılmış ve çalışmada olduğu gibi algoritmalar 50 iterasyon değerinde ve 30 kez çalıştırılmıştır. Literatürdeki çalışmada matematik tabanlı metasezgisel algoritmalar olan Stokastik Fraktal Arama (SFS), Aritmetik Optimizasyon Algoritması (AOA) ve Sinüs Kosinüs Algoritması (SCA) kullanılmıştır. Çalışma MATLAB 2020 ortamında kodlanmış ve Windows 10 işletim sistemine, INTEL CORE i7 işlemciye, 16 GB RAM'e sahip bilgisayar kullanılmıştır. Çalışmada elde edilen sonuçlar Tablo 6.2'de verilmiştir.

Tablo 6.1 ve Tablo 6.2'ye bakarak yapılan çalışma ile sonuçları karşılaştıracak olursak, GA ve CS algoritmaları planlanan yolun ortalama maliyeti, standart sapma, algoritmanın ortalama çalışması süresi bakımından AOA ve SCA'dan daha iyi sonuçlar vermiştir.

Tablo 6.2. Literatürdeki Çalışmada Elde Edilen Sonuçlar (Ort: Ortalama, Std: Standart Sapma)

Yol Planlama Algoritmaları	Harita 1				Harita 2				Harita 3			
	Maliyet			Çalışma Süresi (sn)	Maliyet			Çalışma Süresi (sn)	Maliyet			Çalışma Süresi (sn)
	Ort.	Std.	En iyi		Ort.	Std.	En iyi		Ort.	Std.	En iyi	
SFS	27,7855	0,0014	-	3,1590	31,4098	5,3684	-	3,4920	30,7431	1,44e-14	-	4,3800
AOA	28,8371	1,2027	-	2,8414	37,5400	14,5533	-	3,1085	112,8774	118,4936	-	4,0377
SCA	28,1195	0,5191	-	2,8452	33,1045	10,9593	-	3,1683	81,2305	102,1205	-	4,0402

SA planlanan yolun ortalama maliyeti açısından Harita 1'de AOA ve SCA'ya yakın, Harita 2'de daha kötü, Harita 3'te ise bu algoritmalardan daha iyi sonuç vermiştir. GA ve CS planlanan yolun ortalama maliyetinde Harita 1 ve Harita 2'de SFS'ye yakın bir sonuç verirken, Harita 3'te SFS'den daha iyi sonuç vermiştir. SFS, Harita 1 ve Harita 3'te standart sapma değeri olarak GA ve CS'den daha iyi sonuçlar vermiştir. Süre açısından baktığımızda ise SA, GA ve CS'nin bu algoritmalara büyük bir üstünlük kurduğunu söyleyebiliriz.

7. YEREL YOL PLANLAMA

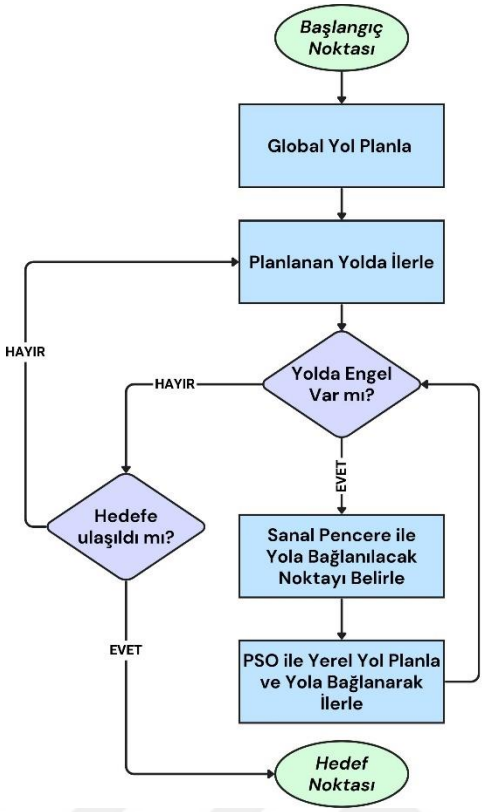
Global yol planlama yaparken, ortamda bulunan önceden bilinen haritalanmış engeller üzerinde çalışılır. Yerel yol planlamada ise ortamda haritalanmış engellere ek olarak bilinmeyen haritalanmamış engeller de mevcuttur. Gerçekçi bir uygulama olması açısından yol planlama yaparken ortamda haritalanmış ve haritalanmamış engellerin olduğunu varsaymamız gerekir. Yerel hareket planlama yöntemleri, ortamdaki sensör verilerini ve hedef konumunu kullanarak manevralarını değiştirmeyi amaçlar (Karakaya ve Ocak, 2020).

Geliştirilen yerel yol planlama yöntemi, global yol planlamada olduğu gibi metasezgisel temellidir. Bu yöntemde ortamda haritalanmış ve haritalanmamış statik engeller olduğu varsayılır. İlk önce Bölüm 5'te anlatılan global yol planlama yöntemiyle ana yol belirlenir. Burada SA, GA ve CS'den herhangi biri kullanılabilir. Ana yol belirlendikten sonra mobil robot planlanan ana yol üzerinde ilerlerken sensörleri aracılığıyla engel kontrolü yapar. Eğer ana yol üzerinde haritalanmamış bir engel tespit ederse yerel yol planlama yaparak engelden kaçınır ve tekrar planladığı yola bağlanarak hedefine doğru hareket eder.

Robot haritalanmamış engelle karşılaştığında etrafında sanal bir pencere oluşturarak ana yola tekrar hangi noktadan bağlanacağını belirler. Ana yola bağlanacağı optimal yolu bulmak için PSO algoritmasını kullanır. Mobil robotun haritalanmış ve haritalanmamış statik engeller içeren ortamda başlangıç konumundan hedef konumuna ulaşması için takip ettiği adımlar Şekil 7.1'deki akış diyagramında gösterilmektedir.

7.1. Bresenham Çizgi Algoritması ile Engel Tespiti

Mobil robot planlanan yolda ilerlerken haritalanmamış engelleri tespit etmek için üzerinde bulunan sensörlerden aldığı verileri kullanır. Fakat simülasyon ortamında yol planlama algoritmasını test ederken Bresenham Çizgi Algoritması kullanılacaktır. Bu algoritma iki boyutlu düzlemde iki nokta arasındaki çizgiyi belirlemek için özellikle bilgisayar grafikleri ve çizim programlarında kullanılır.



Şekil 7.1. Yol Planlama Adımları

Bu yaklaşım az işlemci gücü gerektirdiği için tercih edilmiştir. Algoritma iki nokta arasında çizilecek doğrunun eğimi ve yönüne bağlı olarak piksellerin konumlarını hesaplar. Her adımda çizgiye en yakın pikseli seçer ve bu şekilde doğrunun kaba bir temsilini oluşturur.

Bresenham Çizgi Algoritmasının çalışma adımları aşağıdaki gibidir:

1. Doğrunun başlangıç ve bitiş noktalarının koordinatlarını belirleyin: (x_1, y_1) ve (x_2, y_2).
2. x ve y koordinatları arasındaki farkları hesaplayın:
   a. $d_x = x_2 - x_1$
   b. $d_y = y_2 - y_1$
3. x_1 ile x_2 ve y_1 ile y_1 noktasının değerini karşılaştırarak yönü tespit et.
4. Hatayı hesapla: $\epsilon = d_x - d_y$
5. Hatayı kullanarak karar parametresini hesapla: $D = 2\epsilon$
6. Başlangıç noktasının pikselini çiz.
7. Eğer $\epsilon > -d_y$ ise hata değerini güncelle ve x'i belirlenen yönde arttır.

a. $\in = \in -d_y$

b. x_1'i belirlenen yönde arttır.

8. Eğer $\epsilon < -d_x$ ise hata değerini güncelle ve y'i belirlenen yönde arttır.
   a. $\in = \in +d_x$
   b. y_1'i belirlenen yönde arttır.
9. Adım 7'yi x_1 ile x_2 ve y_1 ile y_2 eşit oluncaya kadar tekrarla.
10. Bitiş noktasının pikselini çiz.

Şekil 7.2'de bu algoritma adımları kullanılarak başlangıç noktası $(2,3)$ ve bitiş noktası $(9,7)$ olan çizgi çizdirilmiştir. Pembe renkli hücreler çizgeye denk gelen algoritma ile hesaplanan pikselleri göstermektedir.



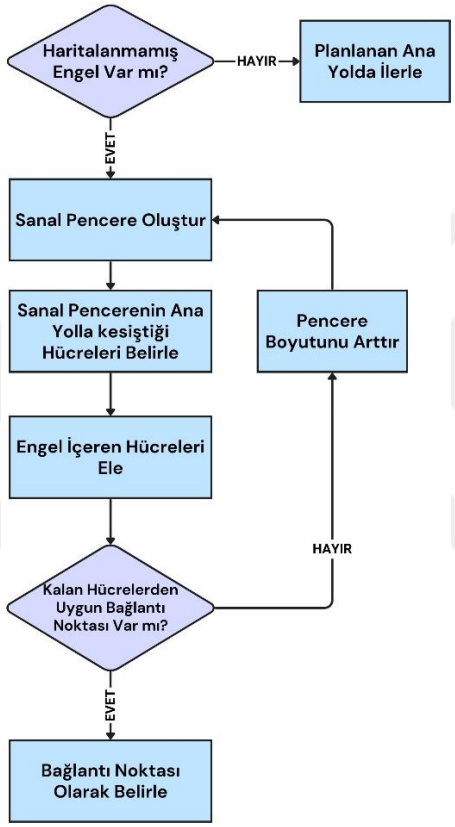
Şekil 7.2. Bresenham Algoritması ile Çizgi Çizme

Global yol bulma algoritması ile planlanan yolun tüm parçalarına Bresenham Çizgi Algoritması uygulanır. Bu sayede ana yolun geçtiği hücreler tespit edilir. Simülasyonda mobil robotun bu hücrelerden sırasıyla geçerek hareket ettiği varsayılır. Böylelikle robotun global olarak planladığı yolun üzerinde haritalanmamış engel varsa tespit edilir.

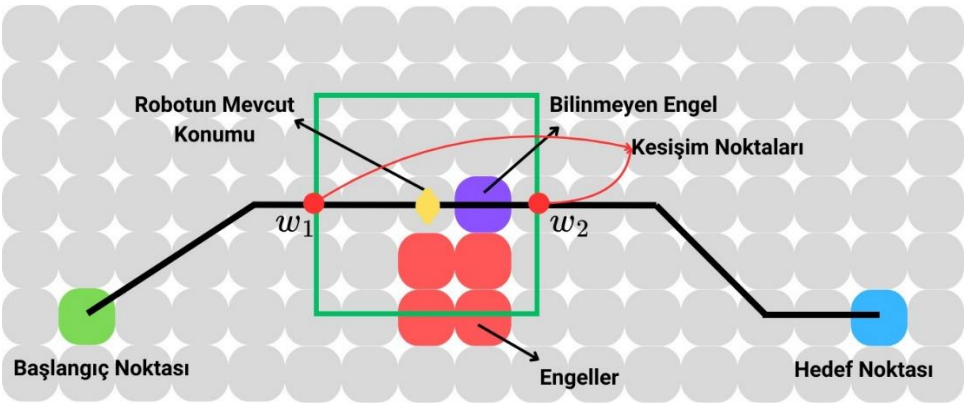
7.2. Pencere Yöntemiyle Bağlantı Noktasını Belirleme

Robot haritalanmamış bir engel ile karşılaştığında engelden kaçınarak tekrar planladığı ana yol üzerinde hareket eder. Bu hareketi yapabilmesi için robotun engelden kaçındıktan sonra ana yola döneceği bağlantı noktasını tespit etmesi gerekir. Bunun için robotun etrafına sanal bir pencere çizilir. Bu sanal pencerenin ana yolla kesiştiği hücreler bağlantı

noktası olarak kabul edilir. Kesişen hücrelerde önce engel kontrolü yapılır. Engel içeren hücreler elendikten sonra robotun ilerleme yönüne uygun olarak bağlantı noktası belirlenir. Eğer kesişen hücrelerin hiçbiri bir çözüm üretmiyorsa pencerenin boyutu artırılır. Pencere boyutu artırıldıktan sonra tekrar kesişen hücre kontrolü yapılarak devam edilir. Şekil 7.3'te pencere yöntemiyle bağlantı noktası belirleme işleminin akış diyagramı verilmiştir ve Şekil 7.4'te ise bu yöntem gösterilmektedir.



Şekil 7.3. Pencere Yöntemi ile Bağlantı Noktası Belirleme Akış Diyagramı

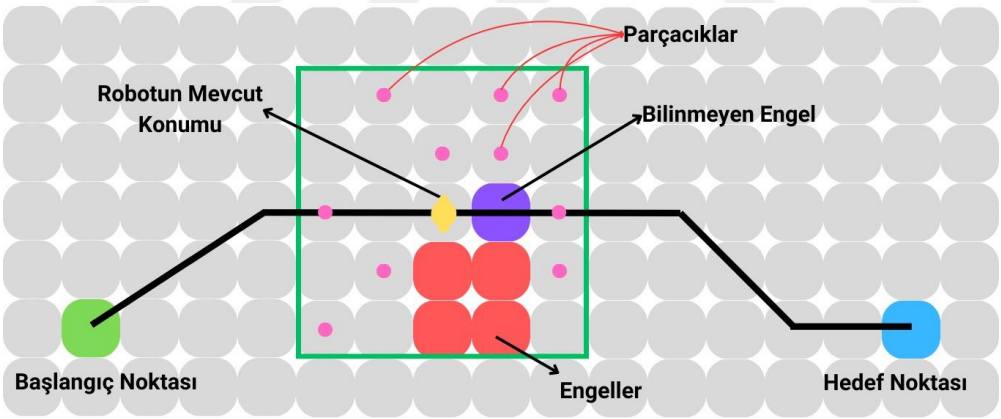


Şekil 7.4. Pencere Yöntemi ile Bağlantı Noktası Belirleme

Şekil 7.4'te görüldüğü üzere robotun mevcut konumunun bulunduğu hücreden 2 birim genişliğinde sanal bir pencere çizilmiştir. Sanal pencerenin planlanan yol ile kesiştiği noktalar w_1 ve w_2 noktalarıdır. Bu noktalar engel içeren hücreye denk düşmediği için ikisinin de değerlendirilmesi gerekir. Robotun hareket yönü dikkate alınarak bağlantı noktası w_2 noktası seçilir.

7.3. PSO ile Yerel Yol Planlama

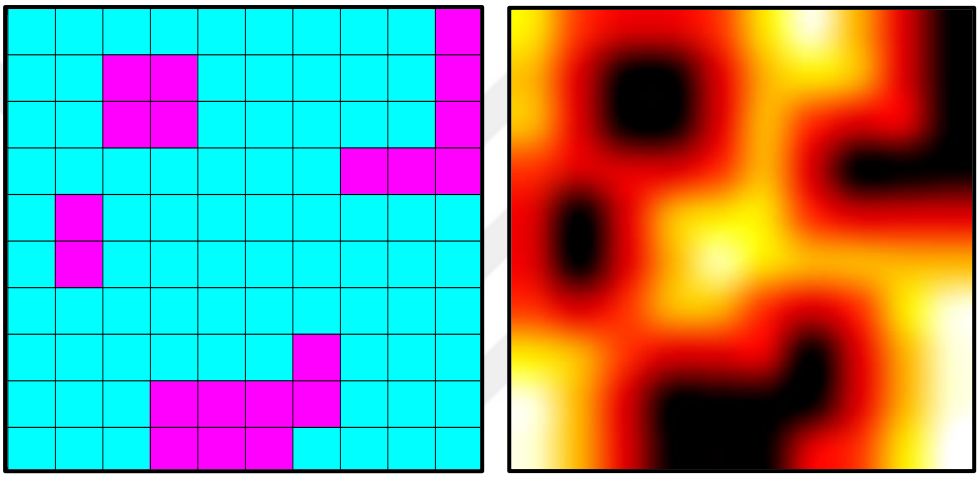
Parçacık Sürü Optimizasyonu Bölüm 4.3.4'te anlatılmıştı. PSO popülasyon tabanlı bir algoritmadır. Başlangıçta parçacıkların rastgele bir şekilde dağıtılması gerekir. Robotun mevcut konumu bellidir ve pencere yöntemi ile de robotun yola bağlanacağı bağlantı noktası belirlenir. Her iterasyonda bu noktalar sabit olacağı için parçacıklar sadece ara noktanın değerini tutar. Parçacıklar dağıtılırken yine sanal bir pencere çizilir ve parçacıklar bu sanal pencere içerisinde dağıtılır. Böylelikle çözümün arama uzayı daraltılır ve çözüm daha hızlı bulunur. Parçacıkların sanal bir pencere içerisinde dağıtılmasının temsili gösterimi Şekil 7.5'te verilmiştir.



Şekil 7.5. PSO ile Parçacıkların Dağıtılması

Başlangıç sürüsünün oluşturulması esnasında parçacıklar rastgele bir şekilde dağıtılırken engelin üzerine denk düşen parçacık varsa bu parçacık yerine yeni rastgele bir parçacık üretilir. Her parçacık bir ara noktadır ve her parçacık için robotun konumundan bağlantı noktasına olan maliyet fonksiyonu hesaplanır. Maliyet fonksiyonu Denklem 5.1 yardımıyla hesaplanır. Daha önce kullanılan maliyet fonksiyonu hesabına ek olarak Öklid Mesafe Dönüşümü (Euclidean Distance Transform, EDT) eklenir. Böylelikle, maliyet fonksiyonuna sezgiselliğe ek olarak, sabit iterasyon boyunca değişmeyen maliyet değeri

eklenmiş olunur. EDT'de engelden uzaklaştıkça hücrelerin aldığı değerler büyür. Yol planlama problemi minimizasyon problemi olduğu için EDT ile bulunan sonuçlar normalize edilerek engelden uzak olan hücrelerin daha küçük değer alması sağlanır. Şekil 7.6'da örnek bir haritanın EDT çıktısı gösterilmektedir. Şekil 7.6a'da örnek harita gösterilmektedir. Bu haritanın EDT çıktısı Şekil 7.6b'deki gibidir. Şekildeki ısı haritasında siyah renkler engelleri göstermektedir. Engele yakın olan ve maliyet değerleri yüksek olan noktalar kırmızı renkle, engelden uzak olan noktalar ise beyaz renkle gösterilmiştir. Diğer noktalar ise maliyet değerlerine göre ara geçiş renkleri ile gösterilmektedir.



(a) (b)

Şekil 7.6. Öklid Mesafe Dönüşümü Örneği: (a) Örnek Harita, (b) EDT Sonucu

Denklem 5.1'i güncellersek yeni maliyet fonksiyonu aşağıdaki gibi olur:

$$C = \begin{cases} \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + C_{EDT} + ceza, & engel\ varsa \\ \sum_{i=1}^{n-1} d(p_i, p_{i+1}) + C_{EDT}, & aksi\ durumda \end{cases} \quad (7.1)$$

Burada C_{EDT}, EDT'den gelen maliyet değeridir. PSO'da parçacıklar, kendi deneyimlerinden ve sürü içerisindeki en iyi parçacığın deneyiminden yola çıkarak kendi değerlerini günceller ve en iyi değere ulaşmayı hedefler. Parçacığın en iyi değeri ve sürünün en iyi değeri maliyet fonksiyonu ile belirlenir. Parçacıklar bu deneyimlerden yola çıkarak hız değerlerini her iterasyonda günceller. Böylelikle parçacıklar her iterasyonda sürüdeki en iyi parçacığa doğru yaklaşma eğilimi gösterir. PSO algoritması optimizasyon kriteri sağlanana kadar çalıştırır. Son durumda sürü içindeki en iyi parçacık çözüm olarak kabul edilir. Bulunan çözüm ara noktayı belirler. Robot mevcut konumundan önce ara

noktaya, ara noktadan da bağlantı noktasına doğru hareket ederek engelden kaçınır ve robot, önceden global olarak planladığı yolda devam eder. Yol üzerinde tekrar haritalanmamış engelle karşılaşması durumunda bu işlemler tekrarlanır.

8. SONUÇ VE ÖNERİLER

8.1. Elde Edilen Sonuçlar

Tez kapsamında SA, GA ve CS metasezgisel algoritmaları ile literatürdeki matematik tabanlı SFS, AOA ve SCA metasezgisel algoritmalarının global yol planlama performansları karşılaştırılmıştır. Tez kapsamında kullanılan SA, GA ve CS algoritmalarını daha kapsamlı incelemek için Bölüm 5'te verilen 64x64 birim haritada her bir yol planlama algoritması 250 iterasyon değerinde ve 1000 kez çalıştırılarak test edilmiş ve Tablo 8.1'deki sonuçlar elde edilmiştir. SA'da başlangıç sıcaklığı 1000, soğutma katsayısı 0,98; GA'da popülasyon sayısı 100, mutasyon olasılığı 0,35; CS'de popülasyon sayısı 10, yumurtaların fark edilme olasılığı p_a 0,5 olarak alınmıştır.

Tablo 8.1. Yol Planlama Algoritmalarının Simülasyon Sonucları

Yol Planlama Algoritmaları	Ortalama Maliyet	Standart Sapma	Bulunan En İyi Değer	Ortalama Süre
SA	371,8461	158,0484	104,1124	0.0020
GA	129,8891	37,7090	103,7672	0,4664
CS	113,9105	25,3271	103,8208	0,1931

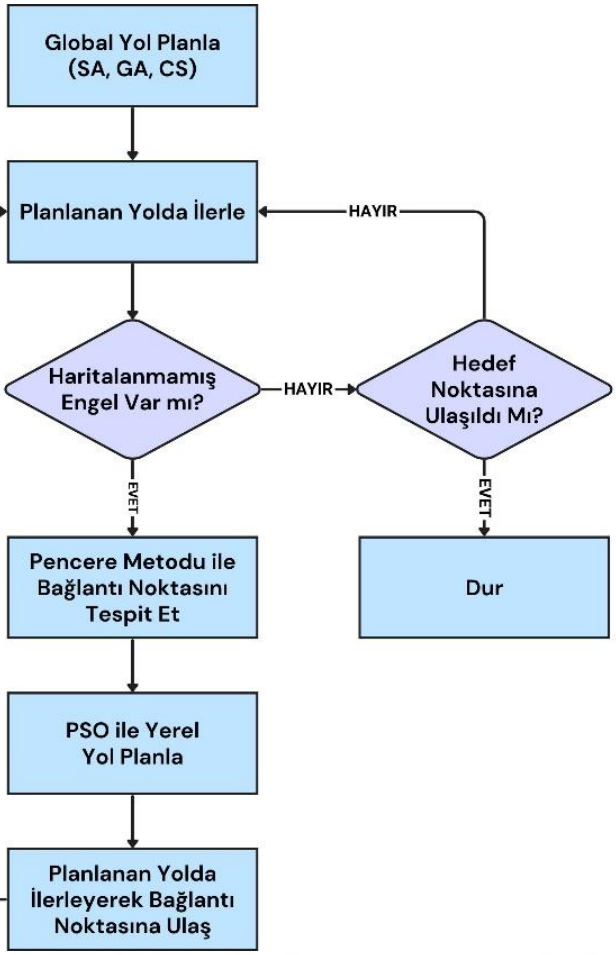
Tablo 8.1. incelendiğinde CS algoritması planlanan yolun ortalama maliyeti ve standart sapma değeri açısından en iyi sonucu vermiştir. Algoritmalar çalıştırılırken buldukları en iyi değerler hafızada tutulmuştur. Test sonucunda en uygun yolu GA hesaplamıştır. SA, tek çözüm tabanlı bir metasezgisel algoritma olduğu için Tablo 8.1'de de görüldüğü üzere diğer algoritmalardan daha hızlı çalışmaktadır. CS algoritması, GA'dan yaklaşık 2 kat daha hızlı çalışmaktadır.

Bu tez kapsamında algoritmaların global yol planlama performansları karşılaştırıldıktan sonra yeni bir yerel yol planlama yöntemi geliştirilmiştir. Geliştirilen algoritmanın simüle edileceği ortam haritalanmış ve haritalanmamış engeller içeren statik bir ortamdır. Mobil robot çalışmaya başlamadan önce hafızasındaki haritayı kullanarak global yol planlama yapar. Bu harita sadece haritalanmış engelleri içermektedir. Fakat robot harekete

başladıktan sonra hafızasındaki haritada bulunmayan engeller ile karşılaşabilir. Bu durumda o engelden kaçınıp tekrar rotasında devam etmesi gerekir.

Geliştirilen algoritmada ilk önce SA, GA veya CS metasezgisel algoritmalarından biri kullanılarak global yol planlama yapılır ve robotun gideceği ana yol belirlenir. Robot planladığı ana yolda ilerlerken üzerindeki sensörler vasıtasıyla engel kontrolü yapar. Eğer robot ana yol üzerinde bir engel ile karşılaşırsa yerel yol planlama algoritmasını çalıştırarak engelden kaçınır ve tekrar ana rotasına dönerek hedefine doğru ilerler. Yerel yol planlama işlemi için yine metasezgisel bir optimizasyon algoritması olan PSO kullanılmıştır. Robot, engel tespit ettiği anda durur ve rotasına geri dönmek için yerel yol planlama algoritmasını çalıştırır. Robot ilk önce geliştirilen pencere metodunu kullanarak ana yola tekrar bağlanacağı noktayı belirler. Bu noktayı belirledikten sonra PSO algoritması devreye girer ve parçacıklar rastgele bir şekilde dağıtılır. Her parçacığın maliyet değeri hesaplanır ve optimizasyon koşulu sağlanana kadar parçacıklar kendi değerlerini günceller. Optimizasyon koşulu sağlandıktan sonra en iyi parçacık seçilir. Robot bulunduğu konumdan, önce seçilen parçacığın olduğu noktaya doğru hareket eder daha sonra o noktadan bağlantı noktasına doğru harekete geçer. Robot bağlantı noktasına ulaştığında artık global olarak planladığı ana rotasına geri dönmüştür ve bu rotayı takip eder. İlerlerken engel kontrolü yapar ve yeniden bir engel ile karşılaşırsa aynı işlemleri tekrarlar. Robot hareketini hedef noktasına varıncaya kadar devam ettirir. Geliştirilen yol planlama algoritmanın akış diyagramı Şekil 8.1'de gösterilmektedir.

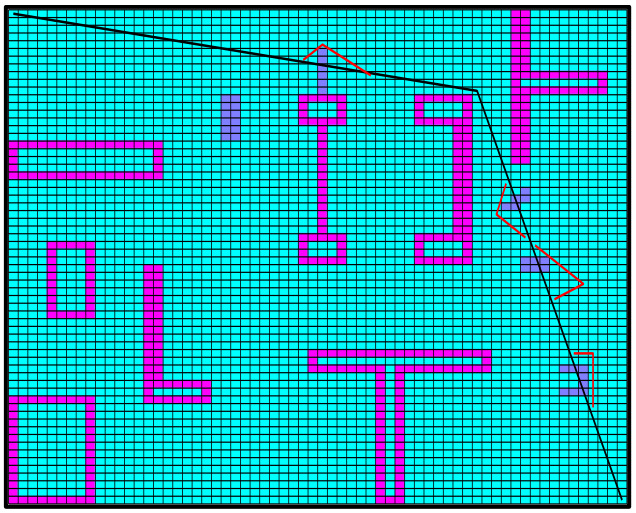
Algoritma MATLAB 2023b versiyonu kullanılarak kodlanmış ve bu programda simüle edilmiştir. Kullanılan bilgisayar INTEL CORE i7 işlemciye, 16 GB RAM'e ve Windows 10 Home işletim sistemine sahiptir. Yerel yol planlama için kullanılan PSO'nun parçacık sayısı 10 olarak belirlenmiş ve parçacıklar robotun konumundan en fazla 10 birim uzaklıkta olacak şekilde sınırlı tutulmuştur. PSO'da eylemsizlik katsayısı w 0,9, bilişsel katsayısı c_1 2 ve sosyal katsayı c_2 2 olarak alınmıştır. Parçacıkların hız sınırı $[-10,10]$ aralığında olacak şekilde sınırlanmıştır ve PSO algoritması 100 iterasyon değerinde çalıştırılmıştır. Robotun haritalanmamış engelleri 2 birim öteden algıladığı varsayılmıştır. Bağlantı noktasını tespit etmek için geliştirilen pencere yönteminde sanal pencerenin boyutu robotun mevcut konumundan 7 birim uzaklıkta olacak şekilde belirlenmiştir yani sanal pencerenin boyutu 15x15 birimdir.



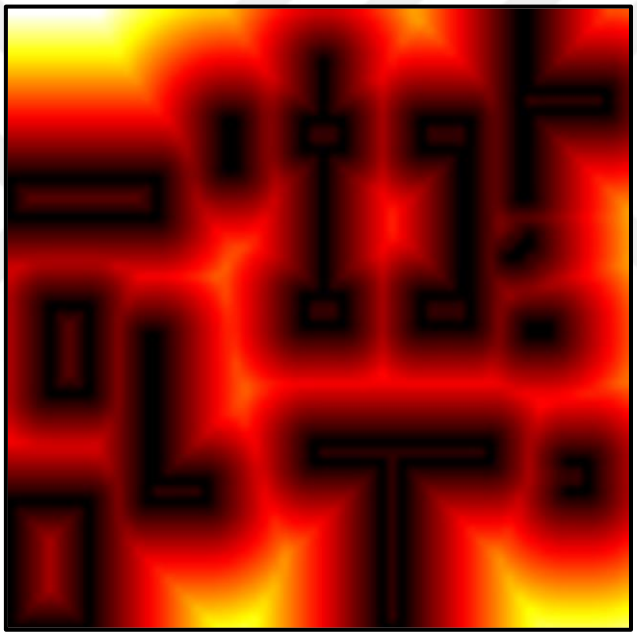
Şekil 8.1. Geliştirilen Yol Planla Algoritmasının Akış Diyagramı

Geliştirilen algoritma bu koşullar altında sorunsuz bir şekilde çalışarak hedefe ulaşmak için gereken yolu planlamıştır. Global yol planlama ile yol belirlendikten sonra algoritmanın haritalanmamış engelleri tespit etmesi ve alternatif yolları planlaması için harcanan süre yaklaşık olarak 0,7 saniyedir. Algoritma haritalanmış ve haritalanmamış statik engeller içeren 64x64 birim haritada çalıştırılmış ve Şekil 8.2'deki planlanan yol elde edilmiştir. Maliyet fonksiyonunda kullanılan Öklid Mesafe Dönüşümü'nün ısı grafiği Şekil 8.3'te gösterilmiştir.

Şekil 8.2'de siyah yol global planlanan yolu, kırmızı yollar ise robotun haritalanmamış engellerle karşılaştığında yerel yol planlama yöntemiyle planladığı yolu göstermektedir. Haritadaki açık mavi hücreler serbest geçiş bölgelerini, pembe renkli hücreler haritalanmış engelleri ve mor renkli hücreler ise haritalanmamış engelleri temsil etmektedir. Şekil incelendiğinde robotun haritalanmamış engellerden kaçınarak hedefe ulaşması için başarılı bir yol planladığı görülmektedir.



Şekil 8.2. Geliştirilen Algoritma ile Planlanan Yol



Şekil 8.3. 64x64 Birim Harita için Öklid Mesafe Dönüşümü

Şekil 8.3'te 64x64 birim boyutlu haritanın Öklid Mesafe Dönüşümü'nün ısı haritası görülmektedir. Şekilde siyah renkler engelleri temsil etmektedir. Şekildeki kırmızı renkler maliyeti yüksek olan hücreleri temsil etmektedir. Kırmızıdan beyaz renge doğru gittikçe maliyet azalmaktadır. Şekildeki ısı haritasında haritalanmamış engellerde görülmektedir çünkü geliştirilen algoritmada robot haritalanmamış bir engel ile karşılaştığında Öklid Mesafe Dönüşümü'nü günceller. Maliyet fonksiyonu; yolun uzunluğu, engelle kesişen yol sayısı kadar ceza puanı ve Öklid Mesafe Dönüşümü'nden elde edilen değerden oluşur. Yolun uzunluğu ve eklenen ceza puanı her iterasyonda rastgele değişmektedir. Bu sezgisel

rastgeleliği kırmak için algoritma boyunca sabit olan sadece yeni bir engelle karşılaşıldığında kendini güncelleyen Öklid Mesafe Dönüşümü eklenmiştir.

Sonuç olarak bu tez kapsamında metasezgisel optimizasyon algoritmaları olan SA, GA ve CS algoritmaları ile global yol planlama algoritmaları geliştirilmiştir ve performansları karşılaştırılmıştır. Tezin bir sonraki aşamasında robotun hareketi esnasında karşılaşacağı daha önce hafızasında olmayan engellerden kaçınması için yeni bir yerel yol planlama yöntemi geliştirilmiştir. Geliştirilen algoritmada, haritalanmamış engellerden kaçınmak için metasezgisel bir algoritma olan PSO algoritması kullanılmıştır. Robot bir engelle karşılaştığında engeli aşıp tekrar global planladığı yola dönmektedir. Robot planladığı yola bağlanacağı noktayı geliştirilen pencere yöntemi ile belirler. Geliştirilen algoritma hem global hem de yerel yol planlama yapmaktadır. Algoritma haritalanmış ve haritalanmamış statik engeller içeren ortamlar için tasarlanmıştır. Yapılan simülasyonlar sonucunda algoritma başarılı bir şekilde çalışmıştır.

8.2. Öneriler

Geliştirilen metasezgisel yol planlama algoritması haritalanmış ve haritalanmamış statik engeller içeren ortamlarda çalışmaktadır. Gerçek dünya koşullarında statik engellerin yanı sıra dinamik engeller de bulunmaktadır. Algoritma dinamik engeller içeren ortamlarda da çalışacak şekilde güncellenebilir. Yapılan yol planlama sonucu elde edilen yol keskin dönüşler içerir. Bu gerçek dünya deneylerinde robotun fazla enerji harcamasına ve daha uzun sürede hedefine ulaşmasına sebep olmaktadır. Yol planlama işlemi yapıldıktan sonra bulunan noktalara uygun yörünge planlanarak robotun daha yumuşak dönüşler gerçekleştirmesi sağlanabilir. Mobil robot haritalanmamış engelleri sensörleri vasıtasıyla algılar. Sensör, engelin arka tarafı hakkında bilgi içermediği için algoritmanın çözümsüz kalacağı koşullar oluşabilir. Bu sorunun üstesinden gelmek için mobil robotun çalışacağı ortam koşulları düzenlenebilir.

Mevcut algoritma statik engel içeren ortam koşullarında çalışmaktadır. Gelecek çalışmalarda bu algoritmanın geliştirilerek dinamik engel içeren ortam koşullarında da çalışacak şekilde güncellenmesi düşünülmektedir.

KAYNAKLAR

Aggarwal, S., Kumar, N. (2020). Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges. *Computer communications*, *149*, 270-299. DOI: 10.1016/j.comcom.2019.10.014

Akarsu, M.E., Çetin, Ö. (2022). Effective Solutions for Common Problems of Artificial Potential Field Based Path Planning Algorithms for Mobile Robots. *Beykent Üniversitesi Fen ve Mühendislik Bilimleri Dergisi*, *15*(2), 105-120. DOI: 10.20854/bujse.1214752

Akka K., Khaber F. (2018). Mobile robot path planning using an improved ant colony optimization, *International Journal of Advanced Robotic Systems*, *15*(3). DOI: 10.1177/1729881418774673

Albayrak, M., Sümen, A.M. (2023). A Yıldız ve Akın Algoritmaları ile Otonom Sürü Sistemleri için Yol Bulma. *Bilişim Teknolojileri Dergisi*, *16*(4), 251-261. DOI: 10.17671/gazibtd.1236552

Amar L.B., Jasim W.M. (2021). Hybrid metaheuristic approach for robot path planning in dynamic environment. *Bulletin of Electrical Engineering and Informatics*, *10*(4), 2152-2162. DOI: 10.11591/eei.v10i4.2836

Amirsadri, S., Mousavirad, S.J., Ebrahimpour-Komleh, H. (2018). A Levy Flight-Based Grey Wolf Optimizer Combined with Back-Propagation Algorithm for Neural Network Training. *Neural Computing and Applications*, *30*, 3707-3720. DOI: 10.1007/s00521-017-2952-5

Asimov, I. (2022). *Ben, Robot* (18). Çeviren: Ekin Odabaş, Deniz Ofset Matbaacılık, İstanbul: İthaki Yayıncılık.

Atalı, G. (2018). Dağınık Mobil Robotlar için Yeni Bir Otonom Yol Planlama ve Engel Tespit Sisteminin Tasarımı. Doktora Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya, 543119.

Banks, A., Vincent, J., Anyakoha, C. (2007). A Review of Particle Swarm Optimization Part I: Background and Development. *Nat Comput*, *6*(4), 467-484. DOI: 10.1007/s11047-007-9049-5

Bayraktaroğlu, Z.Y. (2018). *Robot Teknolojisine Giriş* (1). Sarıyıldız Ofset, Ankara: Nobel Yayıncılık.

Bekdaş, G., Nigdeli, S.M., Yücel, M., Kayabekir, E. (2021). *Yapay Zeka Optimizasyon Algoritmaları ve Mühendislik Uygulamaları*. Ankara: Seçkin Yayıncılık.

Ben-Ari M., Mondada F. (2018). *Elements of Robotics* (e-book). Springer International Publishing.

Bingül, Z., Küçük S. (2019). *Robot Kinematiği* (4). Yazın Basın Yayın Matbaacılık, İstanbul: Umuttepe Yayınları.

Chopard, B., Tomassini, M. (2018). *An Introduction to Metaheuristics for Optimization* (e-book), Cham, Switzerland: Springer International Publishing.

Cong, Y.Z., Ponnambalam S.G. (2009). Mobile robot path planning using ant colony optimization. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore, 14-17 Temmuz 2009.

Contreras-Cruz, M.A., Ayala-Ramirez, V., Hernandez-Belmonte, U.H. (2015). Mobile Robot Path Planning Using Artificial Bee Colony and Evolutionary Programming. *Applied Soft Computing*, *30*, 319-328.

Cormen, T.H., Leiserson C.E., Rivest, R.L., Stein, C. (2022). *Introduction To Algorithms* (Four Edition). The MIT Press, London, England: Massachusetts Institute of Technology.

Deng, Y., Chen, Y., Zhang, Y., Mahadevan, S. (2012). Fuzzy Dijkstra Algorithm for Shortest Path Problem Under Uncertain Environment. *Applied Soft Computing*, *12*(3), 1231-1237. DOI: 10.1016/j.asoc.2011.11.011

Esheman, L.J., Schaffer, J.D. (1993). Real-Coded Genetic Algorithms and Interval-Schemata, *Foundations of Genetic Algorithms, 2*, 187-202
.

Eyüboğlu, M. (2022). Çok Yönlü İşbirlikçi Mobil Robotlar için Eş Zamanlı Hareket Planlama Algoritmasının Geliştirilmesi ve Uygulaması. Yüksek Lisans Tezi, Sakarya Uygulamalı Bilimler Üniversitesi, Lisansüstü Eğitim Enstitüsü, Sakarya, 739163.

Garip, Z. (2018). Mobil Robotların Yol Planlaması için Meta-sezgisel Hibrit Algoritmalar Geliştirilmesi ve Uygulanması. Doktora, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Sakarya, 781860.

Ghosh S., Panigrahi P.K., Parhi D.R. (2017). Analysis of FPA and BA meta-heuristic controllers for optimal path planning of mobile robot in cluttered environment. *IET Science, Measurement & Technology*, *11*(7), 817-828. DOI: 10.1049/iet-smt.2016.0273

Gonzales, G.V., Santos, E.D.D., Emmendorfer L.R., Isodi, L.A., Rocha L.A.O., Estrada, E.D.S.D. (2015). A Comparative Study of Simulated Annealing with Different Cooling Schedules for Geometric Optimization of a Heat Transfer Problem According to Constructal Design. *Scientia Plena*, *11*(8). DOI: 10.14808/sci.plena.2015.081321

Gul, F., Rahiman, W., Alhady, S.N.N., Ali, A., Mir, I., Jalil, A. (2021). Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot

using PSO–GWO optimization algorithm with evolutionary programming. *Intell Human Comput*, *12*(7), 7873-7890. DOI: 10.1007/s12652-020-02514-w

Han, J., Seo, Y. (2016). Mobile Robot Path Planning with Surrounding Point Set and Path Improvement. *Applied Soft Computing*, *57*, 35-47. DOI: 10.1016/j.asoc.2017.03.035

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), 100-1007. DOI: 10.1109/TSSC.1968.300136

Hoang, H.V., Sang, H.A., Tae, C.C. (2013). Dyna-Q-based Vector Direction for Path Planning Problem of Autonomous Mobile Robots in Unknown Environments. *Advanced Robotics*, *27*(3), 159-173. DOI: 10.1080/01691864.2012.754074

Hussein A., Mostafa H., Badrel-din M., Sultan O., Khamis A. (2012). Metaheuristic optimization approach to mobile robot path planning. *2012 International Conference on Engineering and Technology (ICET)*, Cairo, Egypt, 10-11 Ekim 2012.

Jung, J.H., Kim, D.H. (2020). Local Path Planning of a Mobile Robot Using a Novel Grid-Based Potential Method. *IJFIS*, *20*(1), 26-34. DOI: 10.5391/IJFIS.2020.20.1.26

Karaboğa, D. (2020). *Yapay Zeka Optimizasyon Algoritmaları* (7. Basım), Sarılyıldız Ofset, Ankara, Nobek Akademik Yayıncılık.

Karakaya, S., Ocak, H. (2020). Açısal Duruş Kontrolü Destekli Özgün bir Dinamik Pencere Yaklaşımı. *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi*, *7*(1), 184-200. DOI: 10.35193/bseufbd.705765

Kennedy, J., Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, *4*, 1942-1948, Perth, Australia. DOI: 10.1109/ICNN.1995.488968

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

Laubenbacher, R., Hinkelmann, F., Oremland, M. (2013). Chapter 5: Agent-Based Models and Optimal Control in Biology: A Discrete Approach, *Mathematical Concepts and Methods in Modern Biology*. Academic Press, Waltham: Elsevier.

Lin S., Liu A., Wang, J., Kong, X. (2023). An intelligence-based hybrid PSO-SA for mobile robot path planning in warehouse. *Journal of Computational Science*, *67*, 101938, DOI: 10.1016/j.jocs.2022.101938

Liu, Y., Jiang, Y. (2020). Robotic Path Planning Based on a Triangular Mesh Map. *Int. J. Control Autom. Syst.*, *18*(10), 2658-2666, DOI: 10.1007/s12555-019-0396-z

Mahdi, W., Medjahed, S.A., Ouali, M. (2017). Performance Analysis of Simulated Annealing Cooling Schedules in the Context of Dense Image Matching. *Computación y Sistemas*, *21*(3). DOI: 10.13053/cys-21-3-2553

Mareli, M., Twala, B. (2018). An Adaptive Cuckoo Search Algorithm for Optimisation. *Applied Computing and Informatics*, *14*, 107-115. DOI: 10.1016/j.aci.2017.09.001

Meraihi, Y., Gabis, A.B., Mirjalili, S., Ramdane-Cherif, A. (2021). Grasshopper Optimization Algorithm: Theory, Variants, and Applications. *IEEE Access*, *9*, 500001-50024. DOI: 10.1109/ACCESS.2021.3067597

Mohanty, P.K., Parhi, D.R. (2016). Optimal path planning for a mobile robot using cuckoo search algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, *28*(1-2), 35-52. DOI: 10.1080/0952813X.2014.971442

Noto, M., Sato, H. (2000). A Method for the Shortest Path Search by Extended Dijkstra Algorithm. *2000 IEEE International Conference onf Systems, Man and Cybernetics*, Nashville, Tennessee, USA. DOI: 10.1109/ICSMC.2000.886462

Orozco-Rosas, U., Montiel, O., Sepúlveda, R. (2019). Mobile Robot Path Planning Using Membrane Evolutionary Artificial Potential Field. *Applied Soft Computing*, *77*, 236-251. DOI: 10.1016/j.asoc.2019.01.036

Ouach, M.K., Eren, T., Özcan, E. (2023). PRM Path Smoothening by Circular Arc Fillet Method for Mobile Robot Navigation. *UMAG*, *16*(1), 1-19. DOI: 10.29137/umagd.1278980

Pattnaik S.K., Mishra D., Panda S. (2022). A comparative study of meta-heuristics for local path planning of a mobile robot. *Engineering Optimization*, *54*(1), 134-152. DOI: 10.1080/0305215X.2020.1858074

Sabudin, E.N., Omar, R., Debnath, S.K., Sulong, M.S. (2021). Efficient Robotic Path Planning Algorithm Based on Artificial Potential Field. *International Journal of Electrical and Computer Engineering (IJECE)*, *11*(6), 4840-4849. DOI: http://doi.org/10.11591/ijece.v11i6.pp4840-4849

Shebab, M., Khader, A.T., Al-Betar, M.A (2017). A Survey on Applications and Variants of the Cuckoo Search Algorithm. *Applied Soft Computing*, *61*, 1041-1059. DOI: 10.1016/j.asoc.2017.02.034

Shi, Y., Eberhart R.C. (1999). Empirical Study of Particle Swarm Optimization. *Proceedings of the 1999 Congress on Evolutinary Computation-CEC99*, *3*, 1945-1950, Washington, USA. DOI: 10.1109/CEC.1999.785511

Siddiqui R. (2018). *Path Planning Using Potential Field Algorithm*. https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08 (Ziyaret tarihi: 22 Mart 2024).

Stentz, A. (1994). Optimal and Efficient Path Planning for Partially-Known Environments. *In Proceedings IEEE International Congerence on Robotics and Automation*, San Diego, California, USA, 8-13 May 1994

Takahashi, M., Kita, H. (2001). A Crossover Operator Using Independent Component Analysis for Real-Coded Genetic Algorithms. *Proceedings of 2001 Congress on Evolutionary Comoutation IEEE*, Seoul, South korea, 643-649. DOI: 10.1109/CEC.2001.934452

Tezel, Y., Karakaya, S. (2024). Otonom Mobil Robotlar için Metasezgisel Yöntemler Kullanılarak Yol Planlama Algoritmasının Geliştirilmesi ve Performanslarının Karşılaştırılması. *5th International Engineering Research Symposium*, Düzce, Türkiye, 7-9 Mart 2024.

Tuncer A. (2013). Otonom Araçlar için Yol Bulma Probleminin Genetik Algoritmalar ve FPGA ile Çözümü ve Gerçekleştirilmesi. Doktora Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli, 335451.

Tuncer, A., Yıldırım, M. (2012). Dynamic Path Planning of Mobile Robots with Improved Genetic Algorithm. *Computers & Electrical Engineering*, *38*(6), 1564-1572. DOI: 10.1016/j.compeleceng.2012.06.016

URL-1: https://www.fanuc.eu/tr/tr/robotlar/robot-filtre-sayfas%c4%b1/boya-serisi, (Ziyaret tarihi: 15 Mart 2024)

URL-2: https://www.mars-amr.com/robots/c750, (Ziyaret tarihi: 15 Mart 2024)

URL-3: https://www.thymio.org/products/meet-thymio/, (Ziyaret tarihi: 15 Mart 2024)

URL-4: https://www.altinaysavunma.com/tr/urunler/bomba-imha-robot-kolu, (Ziyaret tarihi: 15 Mart 2024)

URL-5: https://baykartech.com/tr/uav/bayraktar-tb3/, (Ziyaret tarihi: 15 Mart 2024)

URL-6: https://www.mi.com/global/product/mi-robot-vacuum-mop-2-pro/, (Ziyaret tarihi: 15 Mart 2024)

URL-7: https://ifr.org/robot-history, (Ziyaret tarihi: 15 Mart 2024)

URL-8:https://www.kuka.com/tr-tr/%c3%bcr%c3%bcnler/robot-sistemleri/end%c3%bcstriyel-robot/kr-1000-titan, (Ziyaret tarihi: 20 Mart 2024)

URL-9:https://www.kuka.com/tr-tr/%c3%bcr%c3%bcnler/robot-sistemleri/end%c3%bcstriyel-robot/kr-delta-robotlar%c4%b1, (Ziyaret tarihi: 20 Mart 2024)

URL-10: https://www.ume.tubitak.gov.tr/tr/icerik/evrensel-sabitler-ve-degerleri, (Ziyaret tarihi: 2 Nisan 2024)

URL-11: https://medium.com/@seckindikbayir/guguk-ku%C5%9Fu-arama-optimizasyon-algoritmas%C4%B1-94049efea5b2, (Ziyaret tarihi: 4 Mart 2024)

Uzun, B. (2022). Otonom Mobil Robotlar için A* ve Olasılıksal Yol Haritası Tabanlı Hibrit Yol Planlama Algoritması Tasarımı ve Analizi. Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 729115.

Venkateswaran, C., Ramachandran, M., Kurinjimalar, R., Vidhya, P., Mathivanan, G. (2022). Application of Simulated Annealing in Various Field. *Materials and its Characterization*, *1*(1), 1-8. DOI: 10.46632/mc/1/1/1

Wahab M.N.A., Nefti-Meziani S., Atyabi A. (2020). A comparative review on mobile robot path planning: Classical or meta-heuristic methods?. *Annual Reviews in Control*, *50*, 233-252. DOI: 10.1016/j.arcontrol.2020.10.001

Wang, D., Tan, D., Liu, L. (2018). Particle Swarm Optimization Algorithm: an overview. *Soft Comput*, *22*(2), 387-408. DOI: 10.1007/s00500-016-2474-6

Xavier-de-Souza, S., Suykens, J.A.K., Vandewalle, J., Bolle, D. (2010). Coupled Simulated Annealing. *IEEE Trans. Syst.*, *40*(2), 320-335. DOI: 10.1109/TSMCB.2009.2020435

Xu, Y., Li, Q., Xu, X., Yang, J., Chen, Y. (2023). Research Progress of Nature-Inspired Metaheuristic Algorithms in Mobile Robot Path Planning. *Electronics*, *12*(15), 3263. DOI: 10.3390/electronics12153263

Yang, X., Deb, S. (2009). Cuckoo Search via Lévy Flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, Coimbatore, India, 210-214. DOI: 10.1109/NABIC.2009.5393690

Yıldırım, M.Y., Akay, R. (2022). Izgara Bazlı Yol Planlama için Matematik Tabanlı Metasezgisellerin Karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*, *32*, 521-530. DOI: 10.31590/ejosat.1039899

Yu, J., Yang, M., Zhao, Z., Wang, X., Bai, Y., Wu, J., Xu, J. (2022). Path Planning of Unmanned Surface Vessel in an Unknown Environment Based on Improved D*Lite Algorithm. *Ocean Engineering*, *266*(3), 1-14. DOI: 10.1016/j.oceaneng.2022.112873

Zexuan, Z., Fangxiao, W., Shan, H., Yiwen, S. (2015). Global Path Planning of Mobile Robots Using a Memetic Algorithm. *International Journal of Systems Science*, *46*(11), 1982-1993.

Zhang Z., Zhao Z. (2014). A Multiple Mobile Robots Path Planning Algorithm Based on A-star and Dijkstra Algorithm. *IJSH*, *8*(3), 75-86. DOI: 10.14257/ijsh.2014.8.3.07

Zhang, H., Lin, W., Chen, A. (2018). Path Planning for the Mobile Robot: A Review. *Symmetry*, *10*(10), 450. DOI: 10.3390/sym10100450

KİŞİSEL YAYIN VE ESERLER

Tezel, Y., Karakaya, S. (2024). Otonom Mobil Robotlar için Metasezgisel Yöntemler Kullanılarak Yol Planlama Algoritmasının Geliştirilmesi ve Performanslarının Karşılaştırılması. *5th International Engineering Research Symposium*, Düzce, Türkiye, 7-9 Mart 2024.

ÖZGEÇMİŞ

İlkokul eğitimini Şehit Büyükelçi Galip Balkar Ortaokulu'nda, lise eğitimini de Beşir Balcıoğlu Anadolu Lisesi'nde tamamlamıştır. 2016 yılında Kocaeli Üniversitesi Mühendislik Fakültesi, Mekatronik Mühendisliği bölümünü kazanmıştır ve 2021 yılında mezun olmuştur. 2021 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalın'nda yüksek lisans eğitimine başlamıştır. 2024 yılında Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi, Kontrol ve Otomasyon Mühendisliği Bölümü'nde Araştırma Görevlisi olarak göreve başlamış ve halen görevine devam etmektedir.