T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TEK VE ÇOK AMAÇLI ROBOT YOL PLANLAMA PROBLEMİ İÇİN HİBRİT BİR OPTİMİZASYON YÖNTEMİ

YÜKSEK LİSANS TEZİ

EŞREF BOĞAR

DENİZLİ, TEMMUZ - 2016

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TEK VE ÇOK AMAÇLI ROBOT YOL PLANLAMA PROBLEMİ İÇİN HİBRİT BİR OPTİMİZASYON YÖNTEMİ

YÜKSEK LİSANS TEZİ

EŞREF BOĞAR

DENİZLİ, TEMMUZ - 2016

KABUL VE ONAY SAYFASI

Eşref BOĞAR tarafından hazırlanan **"Tek ve Çok Amaçlı Robot Yol Planlama Problemi için Hibrit Bir Optimizasyon Yöntemi"** adlı tez çalışmasının savunma sınavı 12.07.2016 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Yrd. Doç. Dr. Selami BEYHAN Pamukkale Üniversitesi	.....
Üye Prof. Dr. Serdar İPLİKÇİ Pamukkale Üniversitesi	.....
Üye Yrd. Doç. Dr. Savaş ŞAHİN İzmir Kâtip Çelebi Üniversitesi	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 20.07.2016 tarih ve ..27/20.. sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Orhan KARABULUT

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

EŞREF BOĞAR

ÖZET

TEK VE ÇOK AMAÇLI ROBOT YOL PLANLAMA PROBLEMİ İÇİN HİBRİT BİR OPTİMİZASYON YÖNTEMİ
YÜKSEK LİSANS TEZİ
EŞREF BOĞAR
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: YRD. DOÇ. DR SELAMİ BEYHAN)

DENİZLİ, TEMMUZ - 2016

Yol planlama robotik alanında popüler bir konudur. Akıllı bir robotun başlangıç noktasından hedef noktasına engellere çarpmadan kendi kendine hareket edebilme yeteneğine sahip olması gerekmektedir. Bu tez çalışmasında bir mobil robotun yol planlaması için iki yöntem önerilmiştir. Önerilen yöntemler İyileştirilmiş Genetik Algoritma (İGA) ve Dijkstra Algoritması ile Genetik Algoritmayı birlikte içeren bir Hibrit Genetik Algoritma'dır (HGA). İGA'da, standart genetik algoritma operatörlerinin kullanımında RYPP'nin yapısı gereği bazı değişiklikler yapılmıştır. Ayrıca önerilen HGA'da, İGA aracılığıyla muhtemel çözümler aranırken, Dijkstra Algoritması ile de yerel arama yoğunlaştırılmaktadır. Önerilen algoritmalar hücrelere ayrılmış bir çevrede robotun sekiz yönlü hareketine izin vermektedir. Yöntemler uygulanırken RYPP'nin iki türü dikkate alınmıştır: Tek Amaçlı RYPP (TARYPP) ve Çok Amaçlı RYPP (RYPP). TARYPP ile rota uzunluğu en küçüklenmeye çalışılırken, ÇARYPP'de mesafe, düzgünlük ve güvenlik amaçları optimize edilmiştir. Dijkstra ve Bellman-Ford Algoritmaları, 0-1 Tam sayılı Doğrusal Programlama Modeli, İGA ve HGA kullanılarak TARYPP için çözümlere ulaşılmıştır. Ayrıca ÇARYPP için de İGA ve HGA kullanılmıştır. Farklı boyutlardaki problemler için sonuçlar elde edilmiş ve bu sonuçlar karşılaştırılmıştır. Sonuçlar, İGA ve HGA'nın kısa zamanda diğer kesin çözüm yöntemleri kadar iyi çözümler ürettiğini ve HGA'nın da İGA'dan iterasyon ve hesaplama süresi bakımından daha iyi çözümler verdiğini göstermektedir.

ANAHTAR KELİMELER: Mobil Robotlar, Yol Planlama, İyileştirilmiş Genetik Algoritma, Dijkstra Algoritması, Bellman-Ford Algoritması, Engelli Çevre, Hibrit Genetik Algoritma

ABSTRACT

A HYBRID OPTIMIZATION METHOD FOR SINGLE AND MULTI OBJECTIVE ROBOT PATH PLANNING PROBLEM
MSC THESIS
EŞREF BOĞAR
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
ELECTRICAL AND ELECTRONICS ENGINEERING

(SUPERVISOR: ASSIST. PROF. DR. SELAMİ BEYHAN)

DENİZLİ, JULY 2016

Path planning is a popular issue in mobile robotics. An intelligent robot must be able to move by itself from a start location to a target location without collision with obstacles. This thesis proposes two methods to solve the problem of path planning for a mobile robot in a static environment with obstacles. The proposed algorithms are an Improved Genetic Algorithm (IGA) and a Hybrid Genetic algorithm (HGA) which includes Genetic and Dijkstra algorithms together. In use of IGA, some changes are made in standard genetic algorithm's operators because of structure of robot path planning problem (RPPP). Additionally, proposed HGA provides diversification while searching possible solutions with IGA, but Dijkstra algorithm makes more and more intensification in local area. The proposed algorithms allows eight-neighbor movements in a grid environment. When applying methods; two types of RPPP are considered: Single Objective and Multi-Objective RPPP (SORPPP and MORPPP). While minimizing total path distance is intended in SORPP, distance, smootness and security objectives are optimized in MORPPP. SORPPP is solved by using Dijkstra and Bellman-Ford algorithms, 0-1 Integer Linear Programing Model, IGA and HGA. Additionally, IGA and HGA are used for MORPPP. Results are obtained for different dimensions of problems and reported with comparisons. Results indicate that IGA and HGA provide solutions as well as other exact solution methods in a short time for large dimensional problems and HGA gives better solutions than IGA in terms of iteration number and computation time both SORPPP and MORPPP.

KEYWORDS: Mobile Robots, Path Planning, Improved Genetic Algorithm, Dijkstra Algorithm, Bellman-Ford Algorithm, Environment With Obstacles, Hybrid Genetic Algorithm

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

P : Robotun geçmiş olduğu noktalar kümesi
$\boldsymbol{f_i(P)}$: P kümesindeki noktalardan geçen robotun i. amaç fonksiyonu değeri
$\boldsymbol{|p_i p_{i+1}|}$: Robotun bulunduğu p_i noktasından p_{i+1} noktasına olan öklid uzaklığı
n : Mobil robotun uğramış olduğu nokta sayısı
$\boldsymbol{C_1}$: Düzgünlük ceza sabiti
S : Çizgi segmenti sayısı
$\boldsymbol{w_i}$: i. amaç fonksiyonunun ağırlığı
$\boldsymbol{E^*}$: En yakın engel
$\boldsymbol{d_{ij}}$: (i,j) ayrıtının uzunluğu
$\boldsymbol{x_{ij}}$: i. düğümden j. düğüme geçilip geçilmediğini gösteren 0-1 karar değişkeni
$\boldsymbol{f_{normalize(i)}(P)}$: Normalize edilmiş i. amaç fonksiyonu değeri
$\boldsymbol{\theta(p_i p_{i+1}, p_{i+1} p_{i+2})}$: $p_i p_{i+1}$ ve $p_{i+1} p_{i+2}$ doğruları arasında kalan açı
$\boldsymbol{enkısa\_uzaklık(p_i p_{i+1}, E^*)}$: $p_i p_{i+1}$ doğrusunun en yakın engele (E^*) olan en kısa uzaklığı

KISALTMALAR LİSTESİ

RYP	**:**	Robot Yol Planlama
RYPP	**:**	Robot Yol Planlama Problemi
GA	**:**	Genetik Algoritma
İGA	**:**	İyileştirilmiş Genetik Algoritma
HGA	**:**	Hibrit Genetik Algoritma
TARYPP	**:**	Tek Amaçlı Robot Yol Planlama Problemi
ÇARYPP	**:**	Çok Amaçlı Robot Yol Planlama Problemi

ÖNSÖZ

Bu tez çalışmasında, bilgi, deneyim ve güvenini esirgemeyen danışmanım Yrd. Doç. Dr. Selami BEYHAN'a, bu süreçte her zaman yanımda olan eşim Zeynep'e ve aileme teşekkür ederim.

1. GİRİŞ

Günümüzde robotlar doğrudan veya dolaylı şekilde birçok alanda hayatımıza girmiştir. Çok farklı kullanım alanı olan ve gün geçtikçe hayatı kolaylaştırmada daha da fazla ihtiyaç haline gelen robotların, çalışma prensiplerinin kullanıldıkları farklı ortamlarda belirlenebilmesi önem arz etmektedir. Robot kelimesi insanlık diline 1900'lü yıllarda Karel Čapek tarafından girmiş ve kendi kendine çalışabilen işçiler olarak tanımlanmıştır (Çamoğlu 2011).

Robotlar sensörleri yardımıyla veya uzaktan bilgisayar kontrolü ile etrafını anlayıp yorumlayan ve çıkan sonuca göre karar alıp, hareket gerçekleştiren ya da durduran elektro-mekanik cihazlardır. Bazı robot türleri bilgisayarlar veya işlemciler tarafından kontrol edilen kollardan oluşurken, bu robotların bacaklar, raylar, tekerlekler veya paletler üzerinde hareket eden türleri de mevcuttur (Okutkan 2006).

Endüstriyel robotlar ve robot manipülatörler, robotların ilk uygulama alanları olmuş ancak güncel uygulamalar kendi kendine çalışabilen otonom robotlar üzerine yoğunlaşmıştır. Özellikle endüstride insan gücü ile üstesinden gelinemeyecek işlerde veya insan için uygun olmayan şartlarda robot manipülatörler tercih edilmektedir. Endüstri de montaj, taşıma, yükleme, boşaltma gibi alanlarda kullanılan sabit veya mobil yapıya sahip robot manipülatörler, verilen komut ve programlandıkları eylemler dışında bir hareket yeteneğine sahip değildirler. Bu nedenle robotik bilimi, kendi hareket kararını verebilen robotlara yönelmiştir. Otonom olarak hareket eden ve verilen görevi yerine getirmeyi hedefleyen bu gruba mobil robotlar denmektedir. Teknolojinin hızlı ilerleyişiyle birlikte, birçok alanda kullanılan bu mobil robotlardan özellikle son yıllarda askeri, afet ve arama çalışmaları, uzay araştırmaları, tıp ve sağlık bilimleri, hobi, eğlence ve eğitim gibi alanlarda istifade edilmektedir (Okutkan 2006).

Mobil robot uygulamalarının çoğunda insan kontrolü ve yönetimi söz konusudur. Bazı kritik işlevler gören robotların eylemlerinde ise tüm kontrol insanlarda olabilir. Ancak mevcut çalışmalar robotların hareketinde insan etkisini ve müdahalesini en aza ya da sıfıra indirme üzerine kaymıştır.

Engelleri tanıma, kendisine bir yol belirleyebilme ve çevreyi algılayabilme bir robotun otonom hareket edebilmesi için şarttır. Sensörler aracılığıyla robot etrafını algılayıp verileri işleyerek bir çevre haritası oluşturur veya önceden tanımlanmış bir haritaya göre hedefe ulaşmaya çalışır. Ortam engelliyse robotun aynı zamanda engellerden de sakınması gerekir. Tüm bu şartlar altında bazı uygulamalarda robotların belli bir hareketi yapması beklenirken, bazı uygulamalarda da robotların bir başlangıç noktasından istenen hedefe ulaşması amaçlanır. Zaman, para, enerji gibi kaynakların ve robotun etkin kullanımı için hedefe en kısa yoldan, en güvenli, en düzgün ve daha farklı performans göstergelerine göre ulaşılmak istenebilir (Geetha ve diğ. 2011). Tez çalışmasının bölümleri alttaki gibi özetlenmiştir.

Tez çalışmasının ikinci bölümünde robot yol planlama problemi (RYPP) ile ilgili literatürde yapılan çalışmalara yer verilmiştir. RYPP için problem tanımı verilerek, problemin çeşitli varyasyonlarından bahsedilmiştir. RYPP için tez kapsamında kabul edilen varsayımları, ızgara tabanlı haritalama ve mobil robot hareket mekanizmaları başlığında irdelenmiştir. Tek amaçlı ve çok amaçlı RYPP'nin özellikleri üzerinde durularak, amaçlar matematiksel olarak ifade edilmiştir. Çok amaçlı robot yol planlama problemi (ÇARYPP) 'nin çözümünde kullanılacak olan ağırlıklandırılmış toplam yöntemi ve amaçların skalarizasyonu hakkında bilgiler verilmiştir.

Üçüncü bölüm RYPP için kullanılan optimizasyon metotlarına ayrılmıştır. Tez kapsamında temel alınan Dijkstra Algoritması ve Genetik Algoritma örneklerle ve kullanım özellikleri bakımından ayrıntılı olarak açıklanmıştır. Ayrıca RYPP'yi çözmede kullanılan diğer kesin çözüm yöntemlerinden Bellman-Ford Algoritması ve 0-1 tam sayılı doğrusal programlama modeli de literatürden faydalanılarak anlatılmıştır.

Dördüncü bölümde, tek amaçlı ve çok amaçlı RYPP için iyileştirilmiş genetik algoritma (İGA) ve hibrit genetik algoritma (HGA) önerilmiştir. Önerilen yöntemlerin adımları ve uygulanış biçimlerine yer verilmiştir. Tek amaçlı robot yol planlama problemi (TARYPP) için en kısa yolun bulunması amacına yönelik problemin kodlanışı ve operatörlerin kullanımı örneklerle anlatılmıştır. Devamında

ise, ÇARYPP için en kısa yol, düzgünlük ve güvenlik amaçları dikkate alınarak önerilen algoritmaların uygulanışına yer verilmiştir.

Beşinci bölümde, TARYPP için elde edilen sonuçlar, Dijkstra Algoritması, Bellman-Ford Algoritması, 0-1 tam sayılı doğrusal programlama, önerilen İGA ve HGA için şekil, grafik ve tablolar üzerinden gösterilmiştir. Önerilen yöntemlerle, kesin çözümü veren yöntemlerle elde edilen sonuçlar karşılaştırılmıştır.

Altıncı bölümde de ÇARYPP için İGA ve HGA kullanılarak elde edilen sonuçlar mevcuttur. Sonuçlar iki algoritmanın performans göstergeleri bakımından farklarını içermektedir. Ayrıca problemin tek amaçlı ve çok amaçlı olarak ele alınışından dolayı amaç değerlerinde ortaya çıkan değişimler de, sonuçlar üzerinden tablolaştırılmıştır.

Yedinci bölümde ise elde edilen sonuçlar ve bulgular yorumlanarak gelecek çalışmalar için öneriler yapılmıştır.

2. ROBOT YOL PLANLAMA PROBLEMİ

2.1 Literatür Araştırması

Robot yol planlama, otonom robot teknolojisinde önemli bir yer teşkil etmektedir. Bu nedenle günümüzde robot yol planlama problemi (RYPP) ciddi çalışmaların temasını oluşturmaktadır. Otonom bir robotun hareket edebileceği güzergahın bulunması önemli bir problemdir ve literatürde RYPP başlığı altında incelenmektedir (Latombe 1991). Kısaca RYPP, otonom bir robotun belirli bir çevrede, engellere çarpmadan, başlangıç noktasından amaçlanan kriterler doğrultusunda (mesafe, zaman, güvenlik, düzgünlük, enerji vb.) hedef noktaya ulaşmasıdır.

Araştırmacılar RYPP'yi ele alırken öncelikle problemi somutlaştırmak için hücre ayrıştırması (cell decomposition) (Graf ve diğ. 2001), yol haritalama (roadmaps) (Zuo ve diğ. 2003) gibi yöntemler geliştirmişlerdir. Böylelikle karmaşık çevre yapıları basite indirgenerek hesaplama kolaylığı sağlanmıştır. En yaygın kullanılan haritalama yöntemi ızgara tabanlı haritalamadır. İlk olarak Moravec (1988), Elfes (1990) ve (Martin ve Moravec 1996) tarafından kullanılan ızgara tabanlı haritalama yöntemi, robotun hareket edeceği çevreyi karelere bölmektedir. Hücre adı verilen bu kareler engelin olup olmadığı; yani hücrenin dolu veya boş olması, ulaşılabilirlik gibi bilgileri tutar. Bu haritalardaki bilgilerin bilgisayar ortamında işlenmesi daha kolay olduğu için, bu çalışmada ızgara tabanlı harita sistemi kullanılmıştır.

Harita sisteminin oluşturulmasından sonra, robotun başlangıçtan hedef noktaya ulaşmak için izleyeceği yolun belirlenmesi gerekir. Hedefe engellere takılmadan ulaşmak için geliştirilmiş bazı algoritmalar mevcuttur. Literatürde en sık kullanılan yol bulma yöntemi Dijkstra Algoritması (Dijkstra 1959), Bellman-Ford Algoritması (Bellman 1958) ve A^* Algoritmasıdır (Hart ve diğ. 1968), (Hart ve diğ. 1972). Bu algoritmalar tek bir kaynaktan tek bir varış noktasına en kısa yolu bulmayı amaçlar. Bu yöntemlerin RYPP'ye uygulanmasında bazı avantajlar olduğu gibi

dezavantajlar da söz konusudur. Algoritmalar en kısa yolu bulmayı garanti eder ancak haritanın çok büyük olması durumunda gereken işlem zamanı ve ihtiyaç duyulan hafıza dezavantaj yaratabilmektedir. Bu tez çalışmasında Dijkstra ve Bellman-Ford algoritmaları RYPP için uygulanmış ve sonuçları karşılaştırılmıştır. Aynı zamanda önerilen hibrit genetik yöntemde de Dijkstra Algoritması kullanılmıştır.

Haritaların büyük olmasından kaynaklı ortaya çıkan zaman ve hafıza problemlerini aşmak için, RYPP'nin çözümünde en iyi çözümü garanti etmeyen ancak kısa sürede iyi çözümler üreten doğadan esinlenmiş çeşitli algoritmalar kullanılmıştır. 2000'li yıllarda RYPP için çok sayıda genetik algoritma tabanlı çalışmalar yapılmıştır (Gemeinder ve Gerke 2003), (Al-Taharwa ve diğ. 2008), (Mohanta ve diğ. 2011). Yapay sinir ağları ve karınca koloni optimizasyonu kullanarak çözüm elde eden algoritmalar da literatürde mevcuttur (Huang 2012) , (Zhi ve diğ 2012). (Chen ve Li 2006) ise statik engelli bir ortamda mobil robot yol planlama problemi için parçacık sürü algoritmasını kullanmışlardır. Yapay bağışıklık sistemi (Hu ve Xu 2007), diferansiyel evrim algoritması (Mo ve Li 2012) , memetik algoritma (Shahidi ve diğ. 2005) gibi farklı metasezgisel yöntemler de RYPP'yi çözmede kullanılmıştır. Her bir algoritmanın kendine özgü baskın ve zayıf yanları olmakla birlikte, farklı parametre değerlerinde farklı sonuçlar elde edebilmişlerdir. Bu tez çalışmasında genetik algoritma temel alınarak iyileştirilmiş genetik algoritma ve hibrit bir yöntem önerilmiştir.

Bahsedilen çalışmalarda RYPP genelde tek amaçlı olarak ele alınarak en kısa yolu bulma hedeflenmiştir. RYPP çözümünün uygulanabilirliğini artırmak ve gerçek hayat problemlerine uyarlamak için, birden fazla amaç oluşturulmuş ve çözülmüştür. En kısa yolu bulmaya ek olarak, robotun engellere çarpma olasılığını en aza indirgeme (güvenlik), manevra sayısını (düzgünlük) azaltma gibi amaçlar da eş zamanlı optimize edilmeye çalışılmıştır (Qiang ve diğ 2004), (Xiao ve diğ. 1997), (Zhou ve diğ. 2008). Bu çalışmalarda RYPP için, her bir amaca belirli ağırlıklar verilmiş, ağırlıklandırılmış toplam yöntemi (weighted sum) kullanılarak da pareto optimum sonuçlar elde edilmiştir. Elde edilen pareto optimum sonuçlarda birbirine baskın olmayan değerler mevcuttur. Bu değerler, verilen ağırlıkta, bir amacın kötüleştirmeden diğerinin iyileştirilemeyeceği anlamına gelmektedir. Yani

ÇARYPP'de, en kısa yol amacına ek olarak diğer amaçlar eklendiği için diğer amaçların iyileştirilmesinden kaynaklı en kısa yol amacının çözümünün tek amaçlıya göre aynı veya daha kötü olması beklenmektedir.

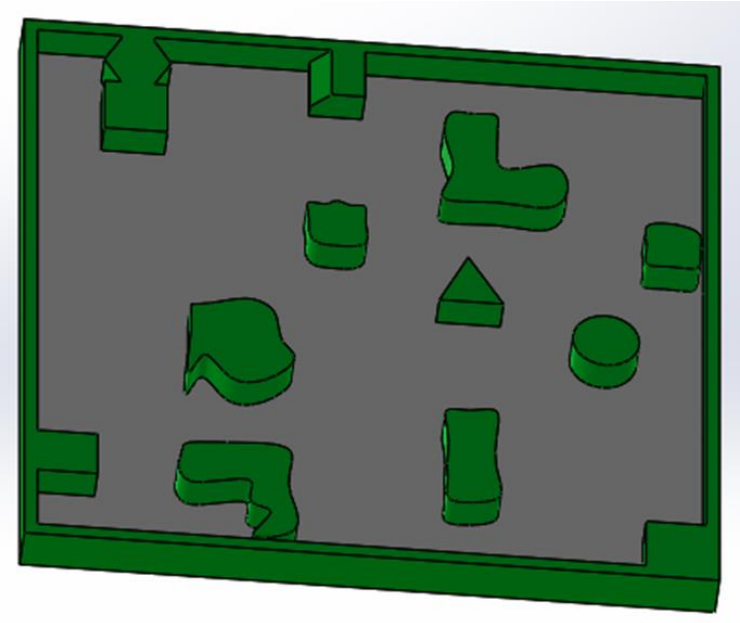
2.2 RYP için Problem Tanımı

Bir robotun belirli kısıtlar altında, kısa sürede, belirli bir amaç veya amaçlar doğrultusunda, bir başlangıç noktasından hedef noktaya ulaştırılmasına RYPP denilmektedir. RYPP tek bir amaca yönelik ise TARYPP, birden fazla amaca yönelik ise ÇARYPP olarak ifade edilmektedir. Örneğin, robotun statik bir çevrede engellere çarpmadan bir başlangıç noktasından en kısa yoldan hedef noktasına ulaşması isteniyorsa TARYPP olarak değerlendirilir. Hem en kısa yoldan hem de güvenli bir şekilde hedefe ulaşması istenirse de ÇARYPP olarak ele alınır.

Literatürde RYPP'nin çözümü aranırken problem tanımı ile ilgili bazı varsayımlar yapılmaktadır. Bu tez kapsamında yapılan varsayımlar aşağıdaki gibidir:

Izgara tabanlı haritalama:

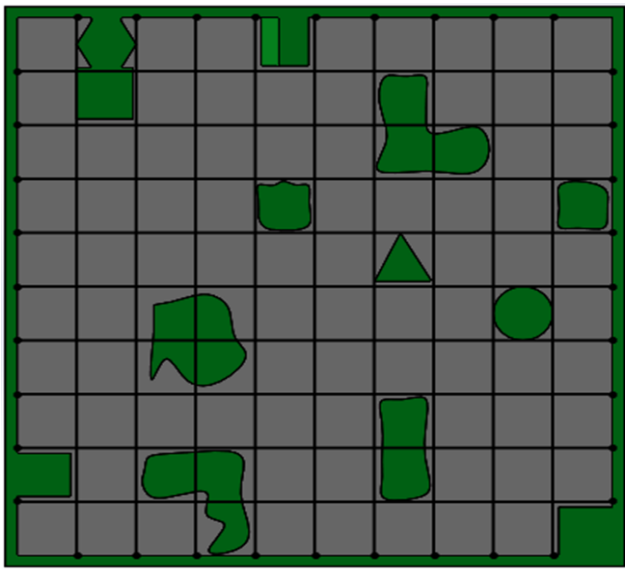
Bu tez çalışmasında ele alınan mobil robot statik bir çevrede hareket etmektedir. Bu çevre bir kamera yardımıyla çevrenin üstten çekilen görüntüsü ile belirlenmektedir. Şekil 2.1'de yeşil (koyu renkli) ve gri (açık renkli) renklerden oluşan 3 boyutlu bir çevre gösterilmiştir. Yeşil renk robotun hareket edemeyeceği yani engellerin mevcut olduğu bölgeleri, gri renk ise robotun hareket edebileceği bölgeleri göstermektedir.



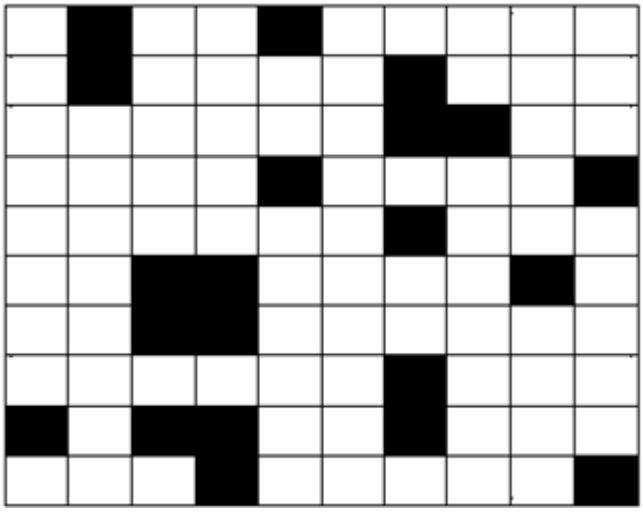
Şekil 2.1: Çevrenin 3 boyutlu görünümü

Kamera yardımıyla alınan görüntü Şekil 2.2’ de gösterildiği gibi 10x10’luk ızgaralara bölünmüştür. Sonsuz noktadan oluşan çevre, engellerin durumuna göre hücrelere bölünmektedir. Bu çevrede ise 100 hücreye bölünerek problemin karmaşıklığı azaltılmış ve problem somutlaştırılmıştır. Eğer çevre daha fazla sayıda hücreye bölünürse daha hassas sonuçlar elde edilecektir ancak problemin çözümü için gerekli olan süre artacaktır.

Izgaranın içinde engelli bir bölge mevcutsa o ızgaranın tamamı engelli bölge, hiç engelli bölge yoksa da engelsiz bölge olarak ifade edilmektedir.



Şekil 2.2: Izgara tabanlı harita



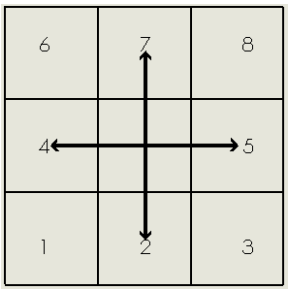
Şekil 2.3: Izgara tabanlı haritanın 2 boyutlu görünümü

Bu tezde yapılan çalışmaların daha anlaşılabilir olması için harita Şekil 2.3' te olduğu gibi 2 boyutlu olarak gösterilmektedir. İçi boş olan bölgeler (beyaz) engelsiz bölgeyi temsil etmekte iken içi dolu (siyah) noktalar ise engelli yani robotun hareket edemeyeceği bölgeleri temsil etmektedir.

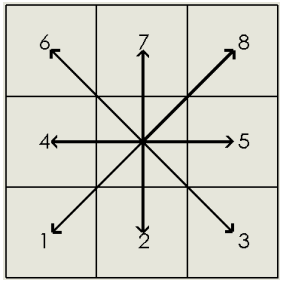
Mobil robot hareket mekanizması:

RYPP'yi çözmek için mobil robotun hareket davranışının bilinmesi gerekmektedir. Literatürde en çok kullanılan iki hareket türü mevcuttur (Ersson ve Hu 2001). Bunlardan birincisi, Şekil 2.4'te gösterilen, robotun bulunduğu noktadan kuzey, güney, doğu ve batı yönlerine olmak üzere dört komşulukta hareket

edebildiği, ikincisi ise Şekil 2.5 ‘te verilen, robotun kuzey, güney, doğu, batı, kuzey batı, kuzey doğu, güney batı ve güney doğu yönlerinde olmak üzere sekiz farklı komşulukta ilerleyebildiği hareket türüdür.



Şekil 2.4: Dört komşuluk



Şekil 2.5: Sekiz komşuluk

Bu tezde mobil robotun sekiz yönde ve engelsiz hücrelerin merkezinden komşu hücrenin merkezine hareket ettiği varsayılmıştır.

2.2.1 Tek Amaçlı Robot Yol Planlama Problemi

TARYPP, önceki bölümde tanımlandığı üzere bir robotun başlangıç noktasından tek bir amaç fonksiyonu altında hedef noktaya ulaşmasıdır. Bu tez çalışmasında RYPP tek amaçlı ise tek amaç, toplam kat edilen mesafenin en küçüklenmesidir. Toplam kat edilen mesafe (2.1) eşitliğinde verildiği gibidir:

$$f_{(P)} = \sum_{i=1}^{n-1} |p_i p_{i+1}| \tag{2.1}$$

P: Robotun başlangıç noktasından hedef noktaya ulaşıncaya kadar geçmiş olduğu noktaların kümesini ifade etmektedir. P= $\{p_1,\ p_2,\ \ldots,\ p_n\}$

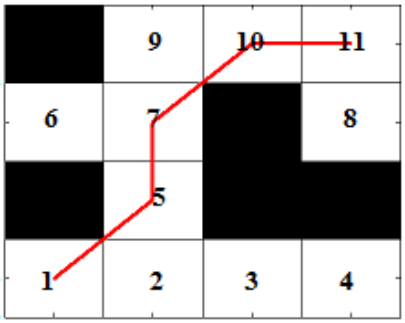
$\boldsymbol{f_{(P)}}$: Maliyet (mesafe) fonksiyonu

$|\boldsymbol{p_i p_{i+1}}|$: Robotun bulunduğu i. noktadan i+1. noktaya olan öklid uzaklığını ifade eder ve iki boyutlu uzayda uzaklık (2.2) eşitliği ile hesaplanır.

$$|p_i p_{i+1}| = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \qquad (2.2)$$

n: Mobil robotun uğramış olduğu nokta sayısı

Amaç: minimum $(f_{(P)})$



Şekil 2.6: 4x4'lük çevre örneği

Örneğin; mobil robot Şekil 2.6'da gösterildiği gibi 1 noktasından (başlangıç noktası) 11 noktasına (hedef nokta) 1→5→7→10→11 hücrelerine uğrayarak ulaşmaktadır.

Parametre değerleri:

$p_1 = 1,\ p_2 = 5,\ p_3 = 7,\ p_4 = 10,\ p_5 = 11$ dir.

P= $\{1, 5, 7, 10, 11\}$,

n = 5,

$|p_1 p_2| = \sqrt{2},\ |p_2 p_3| = 1,\ |p_3 p_4| = \sqrt{2},\ |p_4 p_5| = 1$ olup,

$f_{(P)} = 2\sqrt{2} + 2$ birimdir.

Bölüm 5'te literatürdeki bazı optimizasyon yöntemleri ve tez çalışmasında önerilen optimizasyon yöntemleri kullanılarak TARYPP farklı boyutlarda çözülüp sonuçlar karşılaştırılmıştır.

2.2.2 Çok Amaçlı Robot Yol Planlama Problemi

Mobil robotun başlangıçtan hedefe ulaşırken birden fazla amaç gözettiği problemlere ÇARYPP denilmektedir. Örneğin, mobil robotun başlangıç noktasından en kısa yoldan, güvenli bir şekilde ve fazla manevra yapmadan (düzgün bir şekilde) hedef noktaya gitmesi istenebilir. Bu durumda RYPP için üç farklı amaçtan söz edilir. Bu çalışmada da ele alınan ÇARYPP'nin mesafe, düzgünlük ve güvenlik amaç fonksiyonları sırasıyla denklem (2.3), (2.4) ve (2.5) ile ifade edilmiştir.

Amaç 1:

Robotun başlangıç noktasından hedef noktaya ulaşıncaya kadar kat ettiği toplam mesafe:

$$f_1(P) = \sum_{i=1}^{n-1} |p_i p_{i+1}| \tag{2.3}$$

olarak tanımlanır.

Mesafe fonksiyonun ($f_1(P)$) hesaplanması bölüm 2.2.1 de bahsedildiği gibidir.

Amaç 2:

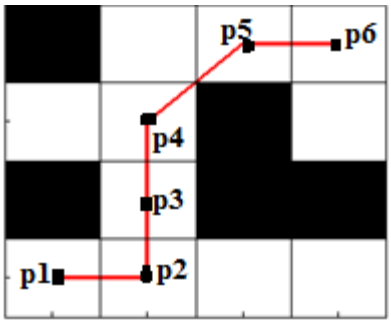
İkinci amaç fonksiyonu, robotun düzgün bir şekilde hedefe ulaşmasının sağlanmasıdır. Düzgünlük fonksiyonu denklem (2.4)'de verildiği gibi hesaplanır:

$$f_2(P) = \sum_{i=1}^{n-2} \theta(p_i p_{i+1}, p_{i+1} p_{i+2}) + C_1 S \tag{2.4}$$

C1: pozitif bir sabit sayı olup ceza kat sayısını ifade etmektedir.

S: çizgi segmenti sayısı

$\boldsymbol{\theta(p_i p_{i+1}, p_{i+1} p_{i+2})}$: $p_i p_{i+1}$ ile $p_{i+1} p_{i+2}$ doğruları arasında kalan açı



Şekil 2.7: Düzgünlük fonksiyonu için örnek

Şekil 2.7' de verilen küçük bir örnek için düzgünlük fonksiyonun hesabı aşağıdaki gibidir.

$\theta(p_1 p_2, p_2 p_3) = 90°$

$\theta(p_2 p_3, p_3 p_4) = 0°$

$\theta(p_3 p_4, p_4 p_5) = 45°$

$\theta(p_4 p_5, p_5 p_6) = 45°$

Çizgi segmenti sayısı (S) 4 tür.

Amaç 3:

RYPP'yi çözmek için kullanılan üçüncü amaç fonksiyonu ise robotun güvenli yani engellere mümkün olduğunca yaklaşmadan hedefe ulaşmasını sağlayan güvenlik fonksiyonudur ve (2.5) eşitliği ile hesaplanır.

$$f_3(P) = \frac{1}{\sum_{i=1}^{n-1} enkısa\_uzaklık(p_i p_{i+1}, E^*)} \tag{2.5}$$

$\boldsymbol{enkısa\_uzaklık(p_i p_{i+1}, E^*)}$ **:** $p_i p_{i+1}$ doğrusunun en yakın engele (E^*) olan en kısa uzaklığıdır.

ÇARYPP'de en kısa yol, düzgünlük ve güvenlik fonksiyonlarının sadece bir tanesi amaç olarak ele alınırsa $\min(f_1(P))$, $\min(f_2(P))$ ve $\min(f_3(P))$; TARYPP olarak değerlendirilir. Fakat iki veya daha fazla amaç fonksiyonu için çözüm elde edilmek istendiğinde her bir amaç, karar vericinin insiyatifine ya da ihtiyaçlara göre ağırlıklandırılır ya da önceliklendirilir. Örneğin, robot en kısa yolu kullanırken engellerin çok olduğu bölgeden veya çok fazla manevra yaparak hedefe ulaşabilir. Bu nedenle kullanıcının robotun yapacağı eylemlerin ağırlıklarına karar vermesi gereklidir.

Bu tez çalışmasında, bir sonraki bölümde anlatılacak olan Ağırlıklandırılmış Toplam Yöntemi ile ağırlıklandırılmış amaç fonksiyonları skalerleştirilerek ve uygun şekilde normalleştirilerek tek bir amaç fonksiyonuna dönüştürülmüştür.

2.2.2.1 Ağırlıklandırılmış Toplam Yöntemi

Çok amaçlı optimizasyon problemlerinin çözümünde en yaygın kullanılan yöntem Ağırlıklandırılmış Toplam Yöntemidir (Gass ve Saaty 1955). Her bir amaca bir ağırlık verilir ve amaçların ağırlıklandırılmış toplamı ile oluşturulan fonksiyon optimize edilmeye çalışılır.

Her w_i ($w_i \geq 0$) ağırlığı bir $f_i(x)$ (i=1,…,n ve n=amaç sayısı) amaç fonksiyonu ile ilişkilendirdiğinde, skaler amaç fonksiyonu denklem (2.6)'daki gibidir:

$$min \sum_{i=1}^{n} w_i f_i(x) \tag{2.6}$$

Amaçlar skalerleştirilirken uygun şekilde normalleştirme yapılması gerekmektedir. Amaçların benzer skalalarda yer alması, sonuçların anlamlı olması bakımından önemlidir. Örneğin, bir amaç 10 birimlik bir mesafeyi, bir diğer amaç 1000 gibi bir açı değerini ifade ediyorsa, bu iki değerin karşılaştırılması uygun olmayacaktır. Bu iki amaca eşit ağırlık verildiğinde, 10 sayısını 1000 ile karşılaştırmak sağlıklı sonuçlar vermeyecek, problem neredeyse ikinci amacı optimize etmeye dönük olacaktır. Bu nedenle ağırlıklandırma doğru yapılmalıdır.

3. OPTİMİZASYON METOTLARI

3.1 Dijkstra Algoritması

En kısa yol probleminin çözümünde en yaygın kullanılan yöntemlerden biri Dijkstra Algoritmasıdır. Bu algoritma bir düğümden, diğer tüm düğümlere olan en kısa güzergahı bulur. Bütün ayrıt uzunluklarının negatif olmadığı varsayımıyla çalışan ve düğümleri önce geçici sonra da kalıcı olarak etiketleyerek ilerleyen bir algoritmadır. Algoritma başlangıçtan bitişe en kısa güzergahı bulmaya ek olarak, her bir düğümün başlangıca göre en kısa güzergahını da vermektedir. Dijkstra algoritması en kısa güzergahları bulmanın yanında, bilgisayar ağları, lojistik gibi alanlarda da hızlı ve etkin bir şekilde hedefe ulaşmak için kullanılmaktadır.

Algoritmanın adımları aşağıda verildiği gibidir:

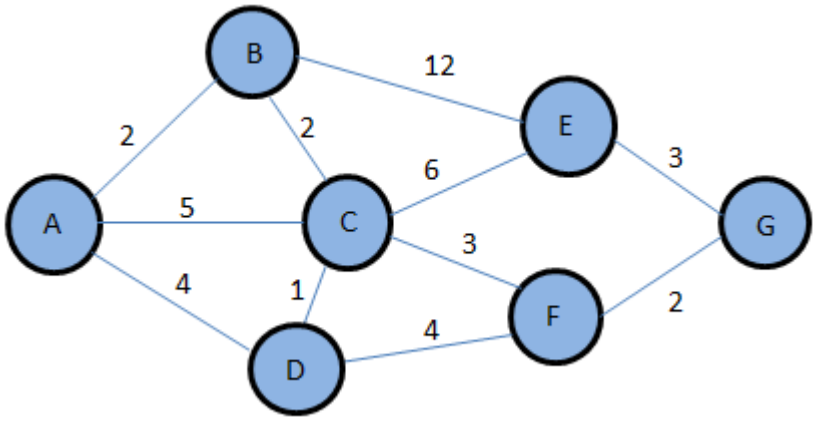
Adım 1: Başlangıç noktası geçerli düğümler kümesinde kalıcı olarak çözüme alınır ve bu düğüme komşu erişilebilir düğümler kümesi tespit edilir.

Adım 2: Geçerli düğümler kümesinden, tespit edilen erişilebilir düğümler kümesine en kısa yol bulunur ve saklanır. Eğer hedef düğüme ulaşıldıysa durulur aksi takdirde Adım 3'e gidilir.

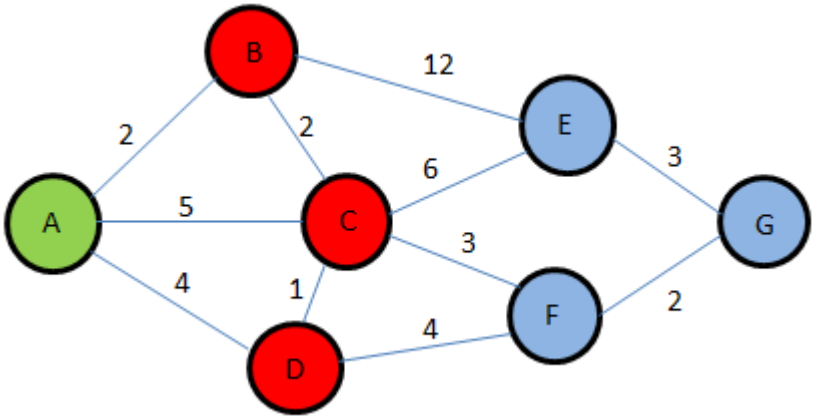
Adım 3: Seçilen en kısa yola ait erişilebilir düğüm, geçerli kümeye dahil edilir.

Adım 4: Mevcut geçerli kümeye ait erişilebilir düğümler tekrar bulunur ve 2. Adıma tekrar gidilir.

Örnek: Şekil 3.1'de verilen ağı kullanarak Dijkstra algoritması ile en kısa yol bulunacak olsun. A düğümünden B düğümüne olan uzaklık 2 birimken, B düğümünden E düğümüne olan uzaklık 12 birim verilmiştir. Bu örnekte amaç A düğümünden G düğümüne en kısa yolu bulmaktır.

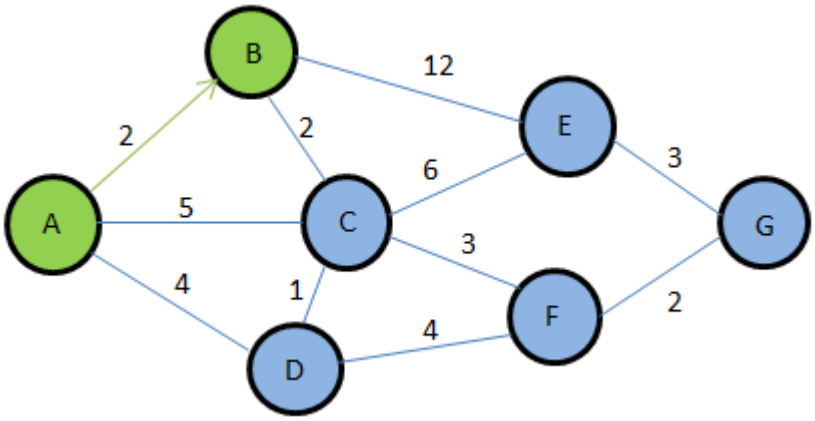


Şekil 3.1: Dijkstra algoritmasının anlatımı için kullanılan çizge



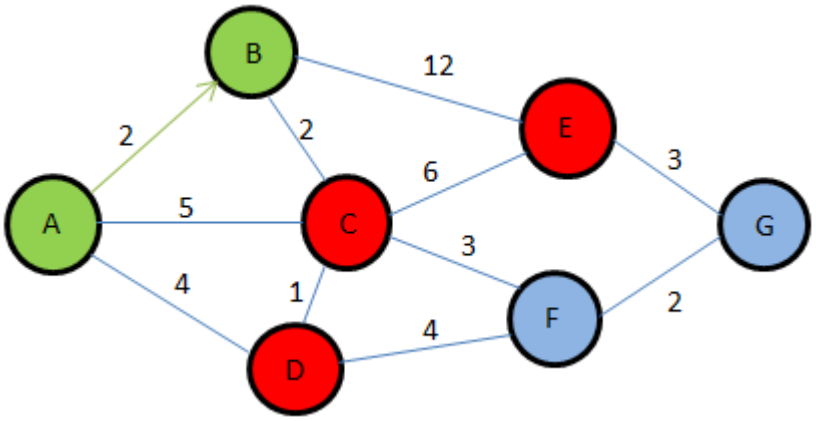
Şekil 3.2: Dijkstra algoritması görseli - 1

Şekil 3.2'de Adım 1'de bahsedildiği gibi A düğümü geçerli düğümler kümesine kalıcı olarak eklenmiştir. A düğümünden erişilebilen komşu düğümler kümesi " B, C ve D" olarak tespit edilmiştir.



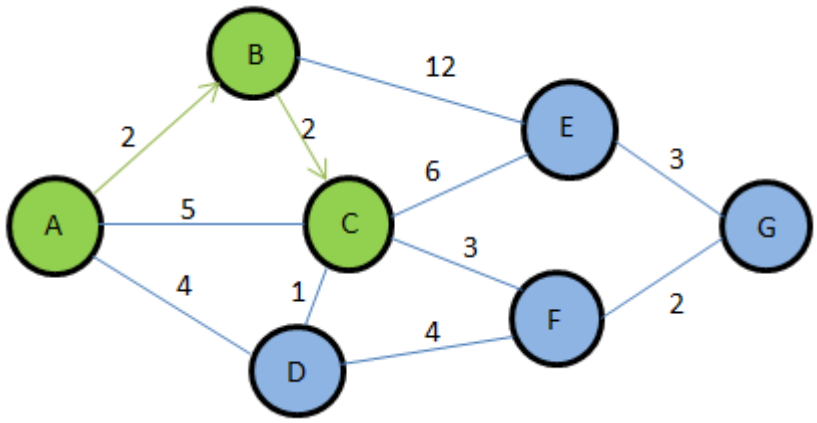
Şekil 3.3: Dijkstra algoritması görseli - 2

Şekil 3.3'te Adım 1'de tespit edilen erişilebilir düğümler kümesindeki B düğümü A düğümüne en yakın mesafede olduğu için saklanır. B düğümü geçerli düğümler kümesine dahil edilir.



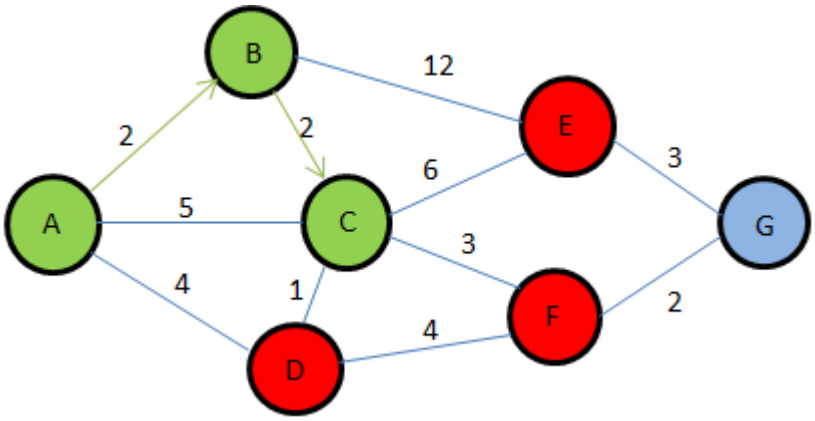
Şekil 3.4: Dijkstra algoritması görseli - 3

Geçerli düğümler kümesinde yer alan A ve B düğümlerinden erişilebilir düğümler Şekil 3.4'te verilmiştir. Bunlar "C, D ve E" düğümleridir.



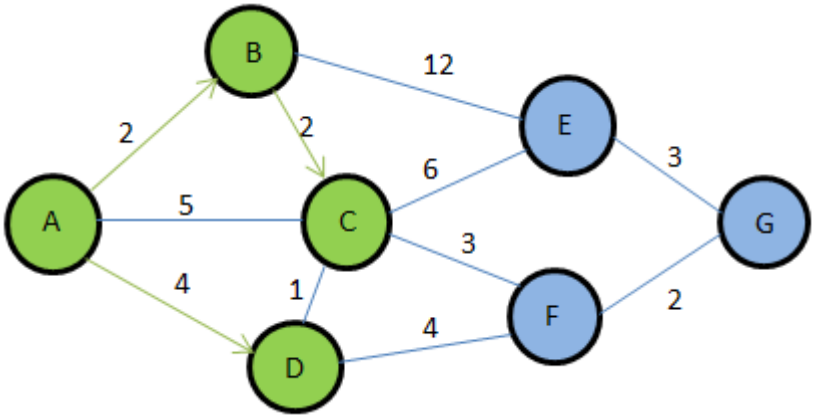
Şekil 3.5: Dijkstra algoritması görseli - 4

Geçerli düğümler kümesinden erişilebilir düğümler kümesine başlangıçtan olan en kısa yol B-C düğümleri arasındadır. C düğümü geçerli düğümler kümesine dahil edilir (Şekil 3.5).



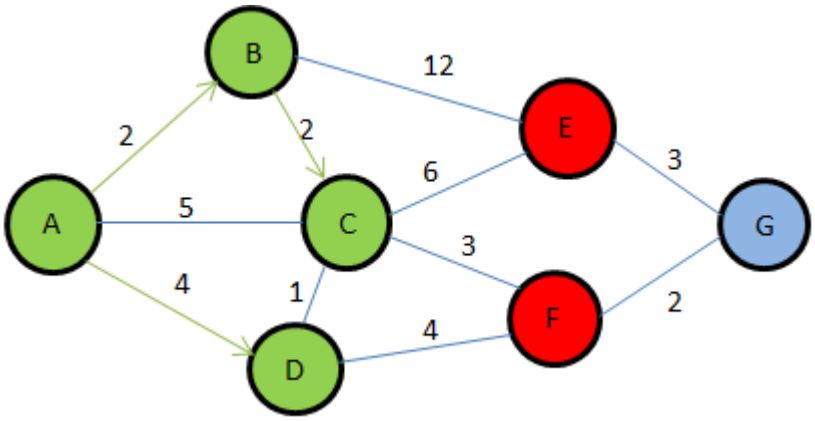
Şekil 3.6: Dijkstra algoritması görseli - 5

Şekil 3.6'da geçerli düğümler kümesi "A,B ve C", erişilebilir düğümler kümesi "D,E ve F" düğümlerinden oluşmaktadır.



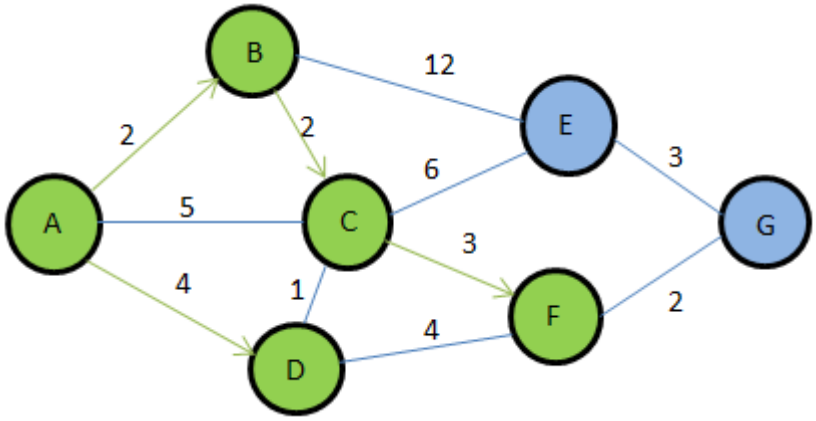
Şekil 3.7: Dijkstra algoritması görseli - 6

Şekil 3.7'de başlangıç A düğümünden erişilebilir düğümler kümesine en kısa yol A-D arasındadır. D düğümü geçerli düğümler kümesine eklenmiştir.



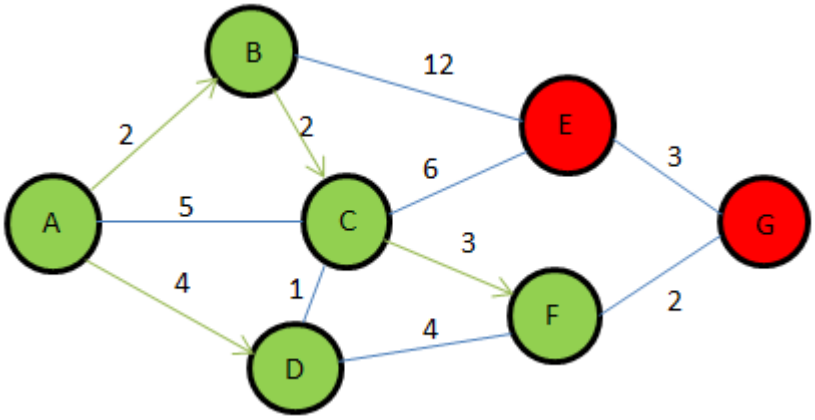
Şekil 3.8: Dijkstra algoritması görseli - 7

Geçerli düğümler kümesi “A,B,C ve D”, erişilebilir düğümler kümesi “E ve F” düğümlerinden oluşmaktadır (Şekil 3.8).



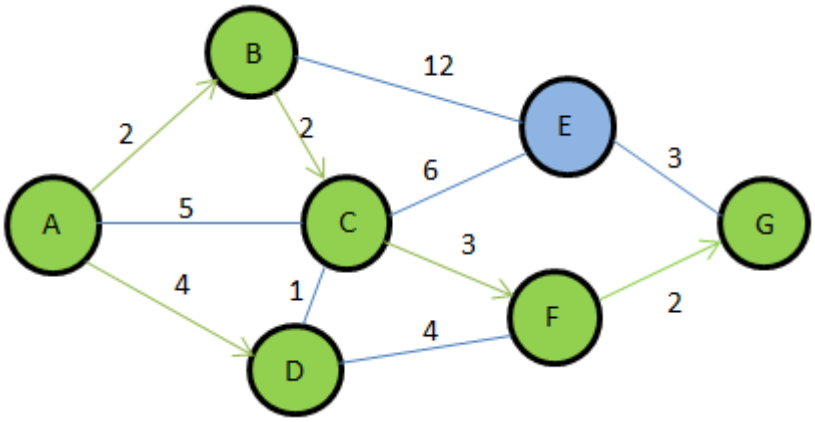
Şekil 3.9: Dijkstra algoritması görseli - 8

F düğümü geçerli düğümler kümesine eklenmiştir (Şekil 3.9).



Şekil 3.10: Dijkstra algoritması görseli - 9

Geçerli düğümler kümesi “A,B,C,D ve F”, erişilebilir düğümler kümesi “E ve G” düğümlerinden oluşmaktadır (Şekil 3.10).



Şekil 3.11: Dijkstra algoritması görseli - 10

Geçerli düğümler kümesinden erişilebilir düğümler kümesine en kısa yol F-G düğümü arasındadır. G düğümü kalıcı düğümler kümesine eklenmiştir. Hedef düğüm olan G’ye ulaşıldığı için algoritma sonlandırılmıştır (Şekil 3.11).

Başlangıç noktası olan A düğümünden hedeflenen G düğümüne giderken saklanan ayrıtlar oklar ile gösterilmiştir. En kısa yol A-B-C-F-G olup, toplam mesafe 9 birimdir. A-D ayrıtı saklanmış ancak D düğümünden G düğümüne gidilebilecek bir ayrıt saklanmadığı için bu yol kullanılmamıştır. Tablo 3.1’ de her bir adımda ki geçerli ve erişilebilen düğümler kümesi ile saklanan ayrıt ve başlangıç noktasından saklanan ayrıtın sonuna kadar olan toplam uzaklık bilgileri yer almaktadır.

Tablo 3.1: Dijkstra algoritması sonucu

Geçerli düğümler kümesi	Erişilebilir düğümler kümesi	Saklanan ayrıt	Toplam uzaklık
A	B, C, D	(A,B)	2
A, B	C, D, E	(B,C)	2+2=4
A, B, C	D, E, F	(A,D)	4
A, B, C, D	E, F	(C,F)	4+3=7
A, B, C, D, F	E, G	(F,G)	7+2=9

3.2 Bellman - Ford Algoritması

Bellman-Ford algoritması Bellman (1958) tarafından geliştirilmiştir. Bu algoritma bir kök düğüm için en az dallanan ağacı bulmakta kullanılır. Algoritma Dijkstra algoritmasında olduğu gibi en küçük değere sahip olan ayrıttan gitmek yerine bütün çevre üzerindeki kenarları test eder. Bellman-Ford Algoritması iki düğüm arasındaki ayrıtların negatif veya negatif ağırlıklı olduğu durumlarda da çözüm üretebilir. Örneğin iki düğüm arasındaki ayrıtın değeri mesafe olarak değil, belirli bir noktayı referans alan açı değerleri olarak tanımlandığında, bu değer negatif olabilir.

Bellman-Ford Algoritmasının çalışma adımları şu şekildedir:

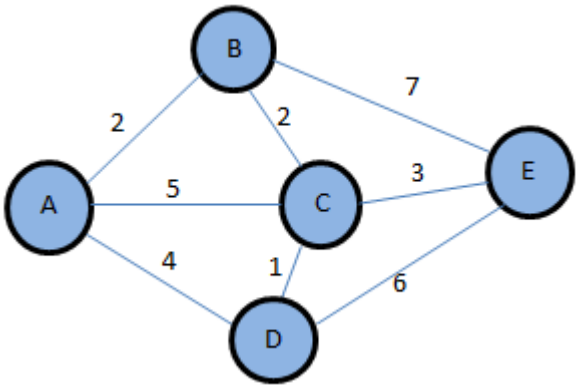
Adım 1: Başlangıç düğümüne 0 değeri atanır, diğer düğümler için maliyet ∞ kabul edilir.

Adım 2: Tüm ayrıtlar bir kural olmaksızın sıralanır.

Adım 3: Her bir düğümün değeri, kendisine gelinceye kadar olan en kısa yol uzunluğu dikkate alınarak negatif değerli ayrıtlar da skaler olarak toplanarak sürekli güncellenir.

Adım 4: Hedef düğüme ulaşıldığında algoritma sonlandırılır.

Örnek: Şekil 3.12'de verilen ağı kullanarak Bellman - Ford algoritması ile en kısa yol bulunacak olsun. A düğümünden B düğümüne olan uzaklık 2 birimken, B düğümünden E düğümüne olan uzaklık 7 birim verilmiştir. Bu örnekte amaç A düğümünden E düğümüne en kısa yolu bulmaktır.



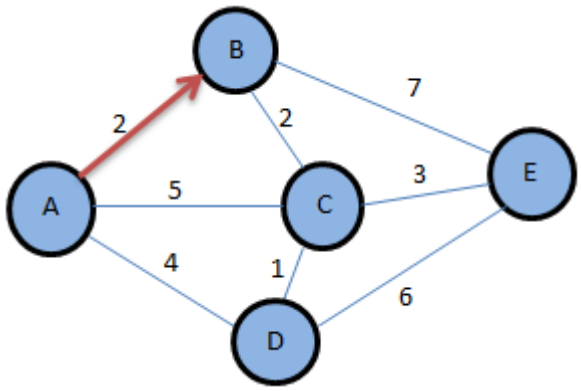
Şekil 3.12: Bellman - Ford algoritması serim gösterimi - 1

Tablo 3.2: Bellman –Ford algoritması aşama - 1

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	∞	-
C	∞	-
D	∞	-
E	∞	-

Tablo 3.2'de verildiği gibi maliyet başlangıç düğüm için 0 diğer düğümler için sonsuz kabul edilmiştir. Tüm ayrıtlar bir kural olmaksızın aşağıdaki gibi sıralanmıştır ve A→B ayrıtından işleme başlanmıştır.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E



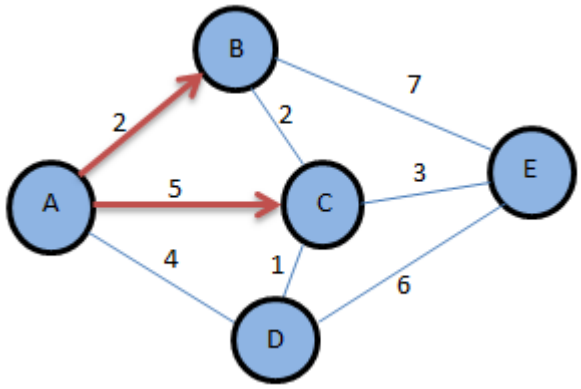
Şekil 3.13: Bellman - Ford algoritması serim gösterimi - 2

Tablo 3.3: Bellman –Ford algoritması aşama - 2

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	∞	-
D	∞	-
E	∞	-

Tablo 3.3'te A başlangıç düğümünden B düğümüne olan uzaklık Şekil 3. 13' te görüldüğü gibi 2 birim olarak güncellenmiştir.

A→B, **A→C**, A→D, B→C, C→D, D→E, C→E, B→E



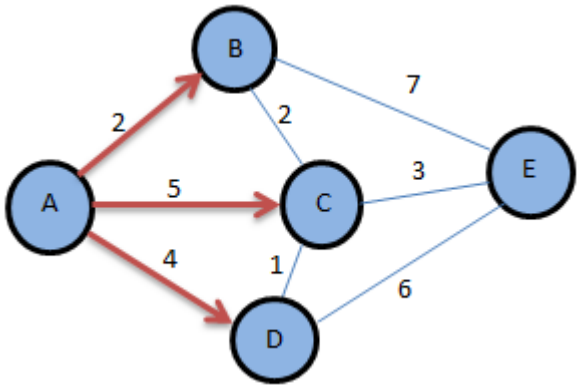
Şekil 3.14: Bellman - Ford algoritması serim gösterimi - 3

Tablo 3.4: Bellman –Ford algoritması aşama - 3

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	5	A→ C
D	∞	-
E	∞	-

Tablo 3.4'te A başlangıç düğümünden C düğümüne olan uzaklık Şekil 3. 14' te görüldüğü gibi 5 birim olarak güncellenmiştir.

A→B, **A→C**, **A→D**, B→C, C→D, D→E, C→E, B→E



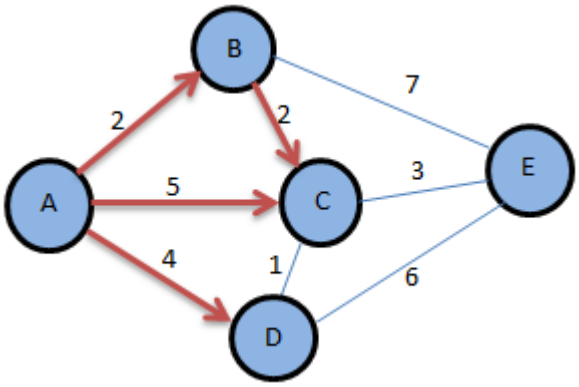
Şekil 3.15: Bellman - Ford algoritması serim gösterimi - 4

Tablo 3.5: Bellman –Ford algoritması aşama - 4

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	5	A→ C
D	4	A→ D
E	∞	-

Tablo 3.5'te A başlangıç düğümünden D düğümüne olan uzaklık Şekil 3. 15' te görüldüğü gibi 4 birim olarak güncellenmiştir.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E



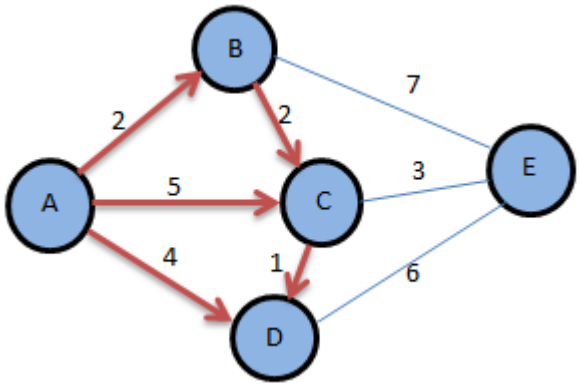
Şekil 3.16: Bellman - Ford algoritması serim gösterimi - 5

Tablo 3.6: Bellman –Ford algoritması aşama - 5

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	4	B→ C
D	4	A→ D
E	∞	-

Tablo 3.6'da A başlangıç düğümünden C düğümüne olan ve BC ayrıtından geçen toplam mesafe Şekil 3. 16' da görüldüğü gibi 4 birim olarak güncellenmiştir.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E



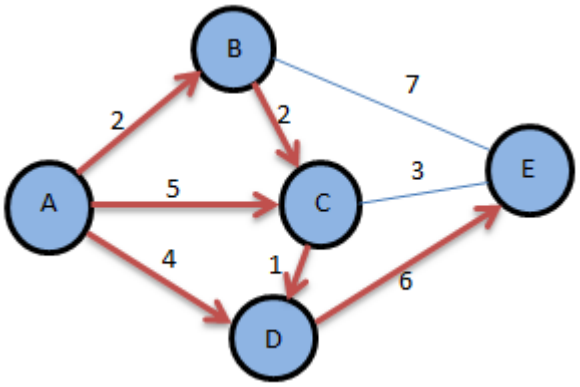
Şekil 3.17: Bellman - Ford algoritması serim gösterimi - 6

Tablo 3.7: Bellman –Ford algoritması aşama - 6

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	4	B→ C
D	4	A→ D
E	∞	-

Tablo 3.7'de her hangi bir güncelleme olmamıştır. Başlangıç noktasından D düğümüne olan en kısa yol Şekil 3.17'de görüldüğü gibi hala AD ayrıtından geçmektedir.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E



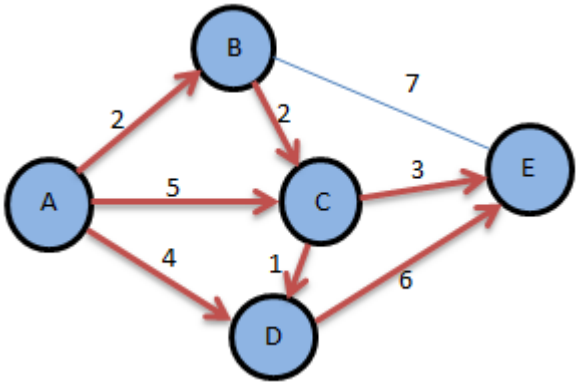
Şekil 3.18: Bellman - Ford algoritması serim gösterimi – 7

Tablo 3.8: Bellman –Ford algoritması aşama - 7

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	4	B→ C
D	4	A→ D
E	10	D→ E

Tablo 3.8'de A başlangıç düğümünden E düğümüne olan ve DE ayrıtından geçen toplam mesafe Şekil 3. 18' de görüldüğü gibi 10 birim olarak güncellenmiştir.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E



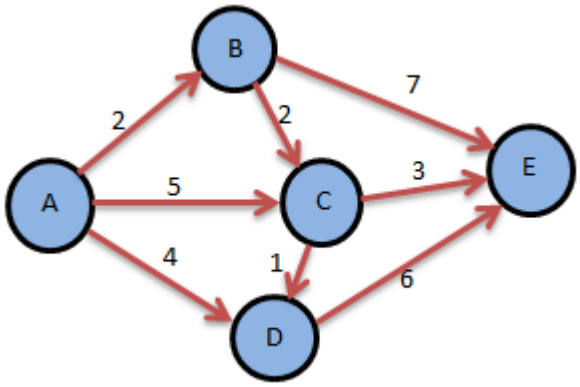
Şekil 3.19: Bellman - Ford algoritması serim gösterimi - 8

Tablo 3.9: Bellman –Ford algoritması aşama - 8

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	4	B→ C
D	4	A→ D
E	7	C→ E

Tablo 3.9'da A başlangıç düğümünden E düğümüne olan ve CE ayrıtından geçen toplam mesafe Şekil 3. 19' dA görüldüğü gibi 7 birim olarak güncellenmiştir.

A→B, A→C, A→D, B→C, C→D, D→E, C→E, B→E

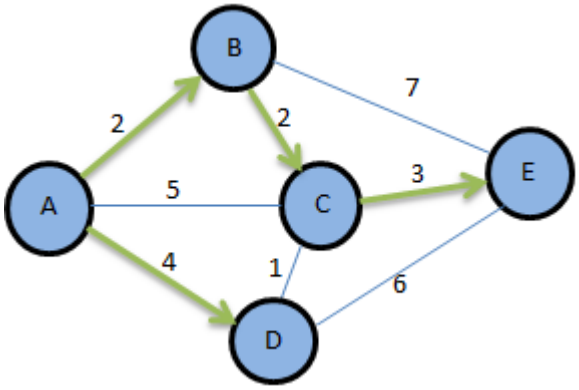


Şekil 3.20: Bellman - Ford algoritması serim gösterimi - 9

Tablo 3.10: Bellman –Ford algoritması aşama - 9

Düğüm	Toplam mesafe	Ayrıt
A	0	-
B	2	A→ B
C	4	B→ C
D	4	A→ D
E	7	C→ E

Tablo 3.10'da her hangi bir güncelleme olmamıştır. Başlangıç noktasından E düğümüne olan en kısa yol Şekil 3.20'de görüldüğü gibi hala CE ayrıtından geçmektedir.



Şekil 3.21: Bellman - Ford algoritması sonucu

Sonuç olarak Şekil 3.21'de görüldüğü gibi en kısa yol 7 birim olup A-B-C-E rotası izlenmektedir. AD ayrıtı başlangıçta iyi bir çözüm vaat ederken E düğümüne en kısa yoldan ulaşmasa başarısız kalmıştır.

3.3 RYPP için 0-1 Tam Sayılı Doğrusal Programlama Modeli

Robot yol planlama problemi, literatürde yer alan en kısa yol probleminin temel özelliklerini ve kısıtlarını taşımaktadır.

Düğüm sayısı n olan bir serimde, i. düğümü j. düğüme bağlayan ayrıtı (i,j) ile gösterecek olursak probleme ait parametre, karar değişkeni, amaç ve kısıtlar aşağıda verildiği şekilde olacaktır (Kara, 1991) :

d_{ij} : (i,j) *ayrıtının uzunluğu*

$$x_{ij} = \begin{cases} 1, & i.\text{düğümden } j.\text{ düğüme geçilirse} \\ 0, & \text{diğer durumlarda} \end{cases}$$

Başlangıç düğümden yalnız bir düğüme geçileceğinden (Denklem (3.1)):

$$\sum_{j=2}^{n} x_{1j} = 1 \tag{3.1}$$

Son (n.) düğüme yalnız bir düğümden gelineceğinden (Denklem (3.2)):

$$\sum_{j=1}^{n-1} x_{jn} = 1 \tag{3.2}$$

Her düğüme ya bir giriş bir çıkış olacak, ya da giriş çıkış olmayacağından (Denklem (3.3)):

$$\sum_{i} x_{ik} = \sum_{j} x_{kj} \qquad k \neq 1\,, k \neq n\,, \forall k \tag{3.3}$$

Amaç toplam mesafenin en küçüklenmesi olduğundan (Denklem (3.4)):

$$min\ Z = \sum_{i}\sum_{j} d_{ij} x_{ij} \tag{3.4}$$

Verilen 0-1 tam sayılı doğrusal programlama modeli, Simplex Algoritmasının çalışma prensibiyle GAMS, Lindo, Lingo gibi farklı programlarla çözülebilmektedir. Probleme ait karar modeli Tablo 3.2'deki gibi olacaktır:

Tablo 3.11: Karar modeli

$$\sum_{j=2}^{n} x_{1j} = 1$$

$$\sum_{j=1}^{n-1} x_{jn} = 1$$

$$\sum_{i} x_{ik} = \sum_{j} x_{kj} \qquad k \neq 1\,, k \neq n\,, \forall k$$

$$x_{ij} = 0\ veya\ 1$$

Kısıtları altında,

$$min\ Z = \sum_{i}\sum_{j} d_{ij}x_{ij}$$

3.4 Genetik Algoritma

Metasezgisel yöntemler, karmaşıklığı yüksek ve çözüm için fazla zaman gerektiren problemler için makul bir zamanda optimumu garanti etmeyen ancak kabul edilebilir sonuçlar üreten çözüm yöntemleridir. Metasezgisel yöntemler literatüre Glover (1962) tarafından kazandırılmış olup farklı sınıflandırmaları söz konusudur Talbi (2009) :

- Doğayı taklit edip etmemelerine göre
- Hafızalı olup olmamalarına göre
- Deterministik veya stokastik olmalarına göre
- Tek çözüm veya popülasyon temelli olmalarına göre
- İteratif veya açgözlü çözüme göre

Tez kapsamında ele alınan metasezgisel yöntem genetik algoritmadır. Genetik algoritma doğayı taklit etmesi bakımından evrimsel süreçten ilham almakta, önceki çözümleri yeni nesillere taşıdığından hafızalı, seçim süreçleri bakımından stokastik, birden fazla çözüm ile ilerlediğinden popülasyon temelli ve oluşmuş bir çözüm

üzerinden yeni nesiller üretilmesi mantığıyla çalıştığından açgözlü çözüm temelli bir metasezgiseldir.

Genetik algoritmanın farklı terimleri, bileşenleri ve operatörleri mevcuttur. İyi çözümler elde edebilmek için problemin genetik algoritmanın çözüm arama mantığına uygun bir şeklide ifade edilmesi gerekir. Genetik algoritmada, birden fazla çözümün yer aldığı çözümler kümesine popülasyon adı verilir. Burada yer alan her bir çözüme birey ya da kromozom denir. Her bir kromozom bir çözümü ifade eder ve genlerden oluşur. Bir gen çözümün bir parçasını veya kısmını saklar. Bir geni oluşturan her bir üyeye de alel adı verilir. Her bireyin bir uygunluk değeri vardır ve bu uygunluk değeri, problem için tanımlanan amaç fonksiyonu değerine eşittir. Her birey uygunluk değeri ile değerlendirilir. Bir popülasyona ait bireylerden belirli operatörler aracılığıyla yeni nesiller oluşturulur. Bireyden yeni bireyler oluşturan kromozomlara ebeveyn, çaprazlama ve mutasyon operatörleri ile oluşturulan bu yeni bireylere ise çocuk adı verilir. Yeni nesil, oluşturulan çocuklar ve mevcut nesil elemanları ile belirli seçim kriterlerine göre oluşturulur. Durdurma kriteri sağlanana kadar veya belirli bir iterasyon sayısı kadar iyi ve kaliteli çözümler elde edilerek popülasyon büyüklüğü sabit kalmak suretiyle yeni nesiller oluşturulur.

Genetik algoritmanın adımları aşağıdaki gibidir:

3.4.1 Kodlama

Kodlama aşamasında bir popülasyondaki bireyin gen yapısının belirlenerek çözümü ne şekilde saklayacağı belirlenir. Bu aşama bir nevi şifreleme gibi görülebilir ve çözümün yorumlanması için de bu şifrenin çözülmesi gerekir. Kodlama stratejisinin çözümün kalitesine, çözüm süresine ve mutasyon- çaprazlama operatörlerinin uygulanmasında büyük etkisi olduğundan, problemin yapısına uygun bir şekilde seçilmesi gerekmektedir. Farklı kodlama çeşitleri mevcuttur (Larranaga ve diğ. 1999):

İkili Kodlama: En yaygın ve eski kodlama türlerinden biri olan ikili kodlamada, genler veya geni ifade eden aleller (bir gen bir alelle ifade edilebileceği gibi birden fazla alelden de oluşabilir) 0 veya 1 değerini alırlar. Burada 0 ya da 1

değerini alacak dizinin (gen veya kromozom) uzunluğunun iyi belirlenmesi gerekir. Örneğin, bir kromozomda bir genin değerinin en fazla 3 olacağı biliniyorsa, ikili sisteme göre bu gende iki alel olması yeterli olacaktır. Kromozomun uzunluğunun da 3 olduğunu varsayarsak, kodlamaya karşı gelen çözüm Tablo 3.3'de verildiği gibi olacaktır:

Tablo 3.12: GA ikili kodlama örneği

İkili Kodlama	10	01	11
Çözüm	2	1	3

Permütasyon Tipi Kodlama: Özellikle tümleşik problemler için kullanılan en uygun kodlamalardan biri permütasyon tipi kodlamadır Çizelgeleme, sıralama, rotalama gibi tümleşik problemlerde, çözüm ardışık yapılacak faaliyetleri gösterir. Ancak bu tip kodlamanın bir dezavantajı çaprazlama esnasında tekrar eden elemanlar oluşabilir. Bu nedenle bir tamir etme (repair) stratejisinin kullanılması gerekebilir.

Tablo 3.13: GA permütasyon tipi kodlama örneği

Sıra	1	2	3	4	5
Birey	1	4	2	5	3

Örneğin Tablo 3.4'te birinci sırada birinci gen varken üçüncü sırada ikinci gen bulunmaktadır. Bir mobil robotun rotasının bu şekilde kodlandığı varsayılırsa, robot ilk önce 1. hücreyi sonrasında sırasıyla 4, 2, 5 ve 3. hücreyi ziyaret edecektir.

Rassal Anahtarlı Kodlama: Bean (1994) tarafından önerilen bu kodlama yönteminde, her bir gene (0,1) arasında bir rassal sayı değeri atanır. Atanan rassal değerlerin sıralanmasıyla da bir permütasyon ifade elde edilir (Tablo 3.5). Bu yöntemin en büyük avantajı uygun olmayan çözümler üretmeyişidir. Permütasyon tipi kodlamada çaprazlama sonucu bir eleman çözümde tekrar yer alabilirken, rassal anahtarlı kodlamada her bir aday çözümde bir defa yer alır.

Tablo 3.14: GA rassal anahtarlı kodlama örneği

Rassal Anahtarlı kodlama	0,43	0,24	0,78	0,60
Çözüm sırası	2	1	4	3

Tablo 3.5'te verilen bir rassal anahtarlamalı kodlama örneği için, birinci sırada ikinci gen ziyaret edilirken üçüncü sırada dördüncü gen ziyaret edilmektedir. Bir mobil robotun rotasının bu şekilde kodlandığı varsayılırsa, robot ilk önce 2. hücreyi sonrasında sırasıyla 1, 4 ve 3. hücreyi ziyaret edecektir.

3.4.2 Başlangıç Popülasyonu Üretme

Problemin boyutuna göre bir popülasyon büyüklüğü belirlenir. Bu büyüklük, arama uzayının kaç faklı noktasında bireylerin bulunduğunu ifade eder. Belirlenen popülasyon büyüklüğü kadar birey rassal olarak başlangıçta oluşturulur. Popülasyon temelli bir metasezgisel olan genetik algoritma, aynı anda birden fazla noktada arama yapabildiği için tek çözüm temelli metasezgisel yöntemlerden bu yönüyle ayrılır.

3.4.3 Uygunluk Değerinin Hesaplanması

Popülasyonda yer alan bir bireyin ifade ettiği çözümün uygunluk değeri, ele alınan problemde optimize edilmek istenen amaç fonksiyonunun değerine karşı gelmektedir. Bir bireyden alınacak olan bilgiler ışığında amaç fonksiyonu hesaplanır ve o bireyin kalitesi uygunluk fonksiyonunun değeri ile doğru orantılıdır. Bu nedenle uygunluk değeri yüksek olan bir bireyin sonraki nesillere aktarılma ve iyi çocuklar üretme olasılığı daha fazladır.

3.4.4 Seçim

Yeniden üreme aşamasına geçilmeden önce seçim operatörü ile hangi bireylerin yeniden üreme aşamasında kullanılacağının belirlenmesi gerekmektedir. Çeşitli seçim operatörleri mevcuttur:

Rulet Çemberi Seçim Yöntemi: Bu seçim yöntemine göre her bir bireyin uygunluk değerine göre bir seçilme olasılığı vardır. Bir bireyin uygunluk değerinin, popülasyondaki tüm bireylerin toplam uygunluk değerine oranı, o bireyin olasılık değerini verir. Bir bireyin olasılık değerini bir pastanın dilimine benzetecek olursak, bireyin uygunluk değeri ne kadar yüksek ise, pastadaki dilimin büyüklüğü de o kadar büyük olacak ve bireyin seçilme olasılığı da aynı oranda fazla olacaktır.

Turnuva Seçim Yöntemi: Rulet çemberi seçim yönteminde seçim işlemi oldukça zaman almaktadır. Özellikle popülasyon büyüklüğü fazla ise hesaplama zamanlarında artış olacaktır. Turnuva seçim yöntemi ise bu probleme bir alternatif olarak düşünülebilir. Popülasyondan belirli sayıda rasgele 'n' tane birey alınır ve uygunluk değeri en yüksek olan birey ebeveyn olarak seçilir. Eğer rasgele iki birey alınıp iyi olan seçilir ise bu 'ikili turnuva seçim' olarak adlandırılır.

Elitizim Stratejisi: Bu strateji ile popülasyonun belirli sayıda bireyi olduğu gibi sonraki nesle aktarılır. Sonraki nesle aktarılacak olan bu bireyler uygunluk değeri en yüksek olan bireylerdir. Örneğin elitizm oranı %10, popülasyon büyüklüğü 50 iken, her nesilde uygunluk değeri en yüksek olan ilk 5 birey bir sonraki nesle doğrudan geçecektir. Böylelikle iyi bireylerin çaprazlama veya mutasyon esnasında seçilmeyerek yok olması engellenmektedir. Ayrıca bu bireylerin kalması çaprazlama ve mutasyonda daha iyi bireylerin oluşmasında önemli rol oynamaktadır.

3.4.5 Yeniden Üreme

Bu adımda seçilen ebeveynlerden yeni nesil elde etmek için çaprazlama ve mutasyon işlemleri yapılmaktadır.

3.4.5.1 Çaprazlama

Genetik algoritmanın karakteristiğini oluşturan en önemli işlemlerden biri olan çaprazlamada, ebeveynlerin kalıtsal özelliklerini taşıyan çocuklar elde edilir. Çaprazlama operatörünün genetik algoritmanın performansı üzerine etkisi oldukça fazladır ve çeşitli çaprazlama türleri mevcuttur.

Tek Noktada Çaprazlama: Bu yöntemde ebeveynler rasgele belirlenen bir noktadan kesilir ve genlerin yerleri kesilen noktadan itibaren karşılıklı değiştirilir. Tablo 3.6'da verilen örnekte permütasyon tipi kodlanmış bireylerin 3. noktadan çaprazlandığını varsayacak olursak oluşan iki çocuk, tabloda verildiği gibi olacaktır.

Tablo 3.15: GA tek noktada çaprazlama örneği

Ebeveyn 1	1	2	**3**	4	5
Ebeveyn 2	2	1	**3**	5	4
Çocuk 1	1	2	3	5	4
Çocuk 2	2	1	3	4	5

N Noktada Çaprazlama: Bu yöntemde ebevenyler rasgele belirlenen bir noktadan değil, belirlenen 'N' sayısı kadar farklı noktadan kesilerek bireylerin karşılıklı yerleri değiştirilir ve çocuklar elde edilir. Tablo 3.7'de ikili kodlamaya göre kodlanmış iki ebeveynin 3 noktada çaprazlanmasına ait bir örnek mevcuttur:

Tablo 3.16: GA N noktada çaprazlama örneği

Ebeveyn 1	0110 / 1011 / 1010 / 1110
Ebeveyn 2	**1010** / **0011** / **1100** / 1**001**
Çocuk 1	0110 / **0011** / 1010 / 1**001**
Çocuk 2	**1010** / 1011 / **1100** / 1110

Uniform Çaprazlama: Bu yönteme göre ilk elemandan başlamak üzere çocuğun veya çocukların hangi alelleri hangi ebeveynden alacakları belirlenir. Çocuk ilk aleli, rasgele üretilen 1 değeri gelirse birinci ebeveynden, 0 gelirse ikinci ebeveynden alır. Bu işlem bireyin tüm alelleri için uygulanır. Eğer iki çocuk üretilmesi söz konusu ise birinci çocuğa, bir gelirse birinci ebeveynin aleli, ikinci çocuğun ilgili aleline ikinci ebeveynin aleli aktarılır.

3.4.5.2 Mutasyon

Mutasyon operatörü ile bir bireydeki bir ya da birkaç genin yerleri değiştirilir. Yeni nesiller üretilirken bir noktadan sonra kromozomlar birbirine çok benzer hale gelebilmektedir. Bu nedenle farklı kromozom üretimi de minimum düzeye dönüşebilmekte ve hep aynı bölgelerde arama yapılmaktadır. Bunun önüne geçebilmek, bireylerin çeşitliliğini artırmak için bireyler mutasyona uğratılmaktadır (Burke ve Kendall 2005). Bu sayede çaprazlama ile ziyaret edilemeyen arama uzayındaki bölgelere ulaşılabilmektedir. Ancak yeniden üretme sürecinde önemli olan mutasyon ile çaprazlama arasındaki dengenin sağlanmasıdır. Basit bir mutasyon örneği verecek olursak, Tablo 3.8'de ikili kodlanmış ve permütasyon tipi kodlanmış bireylerde rasgele seçilen iki genin yeri değiştirilerek mutasyon yapılmıştır.

Tablo 3.17: Basit mutasyon örneği

Birey	1	**2**	3	**4**	5
Mutasyona uğramış birey	1	**4**	3	**2**	5
Birey	**0**	1	0	**1**	1
Mutasyona uğramış birey	**1**	1	0	**0**	1

Özellikle tümleşik problemler için geliştirilmiş bazı mutasyon türleri aşağıda verilmiştir:

Ekleme Mutasyonu: Ekleme mutasyonunda, rasgele seçilen bir gen o noktadan alınıp, yine rasgele seçilen bir noktaya eklenir (Tablo 3.9) (Fogel 1988).

Tablo 3.18: Ekleme mutasyon örneği

Birey	1	**2**	3	4	5
Mutasyona uğramış birey	1	3	4	**2**	5

Basit Ters Çevirme Mutasyonu: Rasgele iki noktadan kesim yapılır ve bu iki nokta arasında kalan genler ters çevrilir (Tablo 3.10) (Holland 1975).

Tablo 3.19: Basit ters çevirme mutasyon örneği

Birey	1	**2**	**3**	**4**	5
Mutasyona uğramış birey	1	**4**	**3**	**2**	5

Ters Çevirme Mutasyonu: Basit ters çevirme mutasyonuna benzer olarak rasgele iki noktadan kesim yapılır ve bu arada kalan genler ters çevrilerek rasgele seçilen iki nokta arasına yerleştirilir (Tablo 3.11) (Fogel 1990).

Tablo 3.20: Ters çevirme mutasyon örneği

Birey	1	**2**	**3**	**4**	5
Mutasyona uğramış birey	1	5	**4**	**3**	**2**

3.4.6 Durdurma Kriteri

Algoritma durdurma kriterine göre sonlandırılır. Durdurma kriteri belirli bir iterasyon sayısı olabileceği gibi, belirli bir hedef değer de olabilir. Aynı zamanda belirli sayıda iterasyonda iyileşme olmaması, iyileşmenin belirli bir oranın altında kalması gibi durdurma kriterleri de tanımlanabilir.

4. ÖNERİLEN OPTİMİZASYON METOTLARI

4.1 Tek Amaçlı Robot Yol Planlama Problemi için Önerilen Optimizasyon Metotları

Bu bölümde, TARYPP için bir Genetik Algoritma ve Hibrit Genetik Algoritma (HGA) yöntemleri önerilmiştir. Önerilen yöntemlerin probleme uygulanış biçimi aşama aşama verilmiştir. Tek bir amacın ele alındığı bu bölümde, amaç mobil bir robotun en kısa yoldan hedefe ulaşmasıdır.

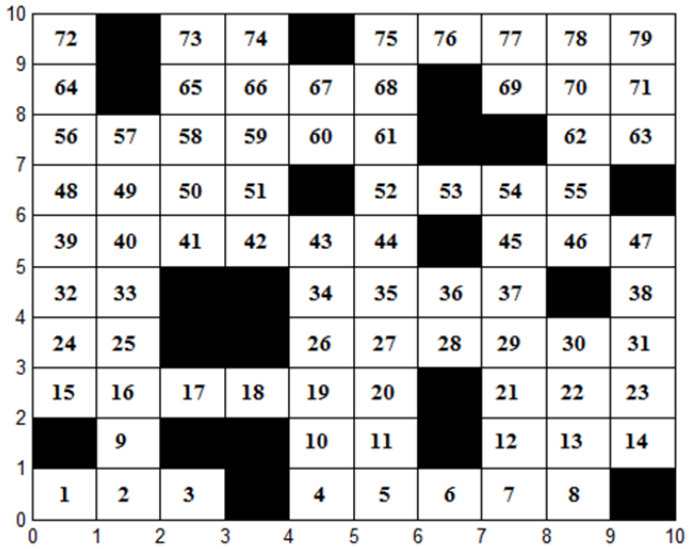
4.1.1 İyileştirilmiş Genetik Algoritma

RYPP için kesin çözüm veren algoritmalar Bölüm 3.1, 3.2 ve 3.3'te ele alınmıştır. Fakat problem boyutu arttığında verilen bu algoritmaların problemi çözme zamanları da artmaktadır. Bu nedenle optimum çözümü (en iyi çözümü) garanti etmeyen ancak kısa sürede makul çözümler üretebilen bir metasezgisel yöntem olan Genetik Algoritma ile çözüme gidilmiştir.

Problemin doğası gereği, bir robotun başlangıç noktasından hedef noktasına ulaşabilmek için izlediği yol, bir dizi oluşturmaktadır. Örneğin 1→2→3→4→5→6→7 şeklinde çözümü ifade eden bir dizide, robot başlangıç noktası 1'den 2'ye, 2'den 3'e, … ve 6. noktadan hedef olan 7. noktaya hareket etmektedir. Çözümün bu şekilde kesikli bir dizi olarak kolayca ifade edilebilir olması, Genetik Algoritmadaki çaprazlama ve mutasyon operatörlerinin etkin bir biçimde uygulanmasına avantaj sağlamaktadır. Gerek operatörlerin uygulanma kolaylığı gerekse kesikli bir problem olan RYP problemine iyi çözümler üretebilecek bir sistematiğe sahip olması nedeniyle, Genetik Algoritma çözüm yöntemi olarak seçilmiştir.

Şekil 4.1'de verilen 10x10'luk bir örnek problemde, robotun hareket edebileceği noktalar 1' den 79' a kadar numaralandırılmıştır. Önerilen Genetik

Algoritmanın aşamaları bu örnek üzerinden anlatılacaktır. Robotun başlangıç noktası 1 numaralı, hedef noktası ise 79 numaralı nokta ile gösterilmiştir.



Şekil 4.1: 10x10'luk örnek problemin hücre gösterimi

RYPP için uygulanan Genetik Algoritma adımları aşağıda verildiği gibidir:

Adım 1: Kodlama

Kodlama türü olarak permütasyon tipi kodlama kullanılmıştır. Robotun başlangıç noktası 1 ve hedef noktası 79 olmak üzere, hangi noktadan hangi noktaya hareket edileceğini gösteren bir dizi ile kodlama yapılmıştır.

[1→9→16→17→…→71→79]

Adım 2: Başlangıç Popülasyonu Üretme

Başlangıç popülasyonu üretilirken robotun başlangıç noktasından (1), uğramış olduğu bir noktadan tekrar geçmemek koşuluyla, hedef noktaya (79) ulaştığını ifade eden rassal bir dizi oluşturulmaktadır. Her bir noktadan ulaşılabilecek belirli noktalar vardır. Örneğin, başlangıç olan 1'den 2 veya 9'a gidebilmektedir. Herhangi bir nokta olan 16'dan ise 9, 15, 17, 24 ve 25 noktalarına gitmek mümkündür. Bir çözüm oluşturulurken, bir noktadan hangi komşu noktaya gidileceği rassal olarak belirlenir. Hedef nokta olan 79 a ulaşılıncaya kadar dizide yer alacak olan her bir eleman bu şekilde belirlenir ve diziye eklenir. Bu işlem popülasyon

büyüklüğüne (N) ulaşılıncaya kadar devam ettirilir. Algoritma boyunca popülasyondaki birey sayısı sabit kalmaktadır.

Tablo 4.1'de bir bireyin rassal olarak üretilmesi gösterilmiştir.

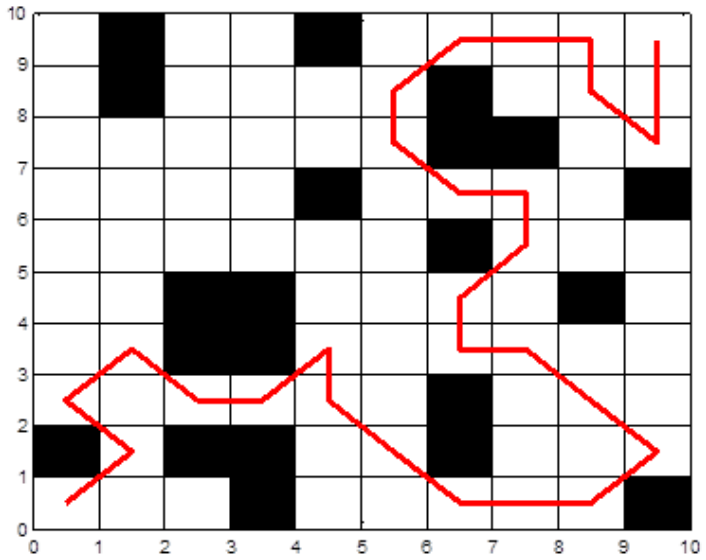
Tablo 4.1: Rassal başlangıç çözüm üretme

Mevcut Çözüm	Adaylar	Rassal olarak seçilen nokta
1	2,9	9
1, 9	2, 3, 15, 16, 17	15
1, 9, 15	16, 24, 25	25
…	…	…
1, 9, 15, 25, … , 71	62, 79	79

Şekil 4.2'de gösterilen rastgele belirlenmiş bir başlangıç çözümün dizi halinde gösterimi ise aşağıda verildiği gibidir:

[1,9,15,25,17,18,26,19,11,6,7,8,14,22,29,28,36,45,54,53,61,68,76,77,78,70,

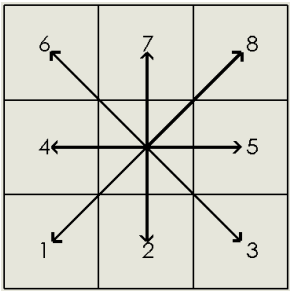
63,71,79]



Şekil 4.2: Rassal başlangıç çözüm

RYPP için problemin boyutu ve ortamdaki engelsiz nokta sayısı arttıkça, robotun hem uğrayabileceği nokta sayısı artmakta hem de buna bağlı olarak problemin çözüm süresi artmaktadır. Bu nedenle, popülasyon büyüklüğünü problemin boyutuna göre belirlemek önemlidir. Popülasyon büyüklüğü ne yerel optimuma takılacak kadar küçük, ne de çeşitliği sağlarken yerel arama yapamayacak kadar büyük olmalıdır.

Adım 3: Uygunluk Değerinin Hesaplanması



Şekil 4.3: Robotun sekiz olası hareketi

Şekil 4.3'te bir robotun hareket edebileceği olası sekiz nokta gösterilmektedir. Merkez noktada bulunan robot 1, 3, 6 ve 8 noktalarına hareket ederse $\sqrt{2}$ birim, 2, 4, 5 ve 7 noktalarına hareket ederse 1 birimlik yol kat etmektedir.

Algoritmada bireylerin uygunluk değerlerinin (amaç fonksiyonu) hesaplanması gerekmektedir. Popülasyondaki her bir birey için uygunluk değeri Denklem (4.1) ile bulunur.

$$d_{ij} : (i,j) ayrıtının\ uzunluğu$$

$$\text{Toplam mesafe} = \sum_i d_{ij} \qquad (her\ i'den\ j'ye\ olan\ gidişler\ için) \qquad (4.1)$$

Şekil 4.2'de rassal üretilen bireyin uygunluk değerinin hesaplanması aşağıda gösterilmektedir:

Birey→[1,9,15,25,17,18,26,19,11,6,7,8,14,22,29,28,36,45,54,53,61,68,76,77,

78,70,63,71,79]

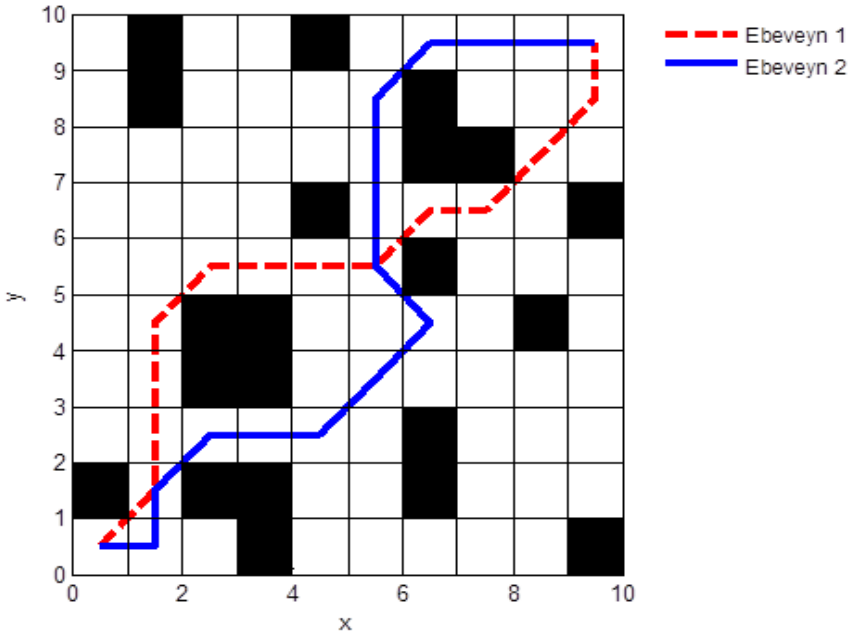
$$d_{19} = \sqrt{2},\ d_{9\,15} = \sqrt{2},\ d_{15\,25} = \sqrt{2}, \ldots,\ d_{63\,71} = 1,\ d_{71\,79} = 1$$

Robotun hareketini temsil eden bu bireyin uygunluk değeri dizideki tüm ayrıt uzunluklarının toplanmasıyla 33.799 birim olarak bulunmuştur. Popülasyondaki her bir birey için uygunluk değeri bu şekilde hesaplanmaktadır.

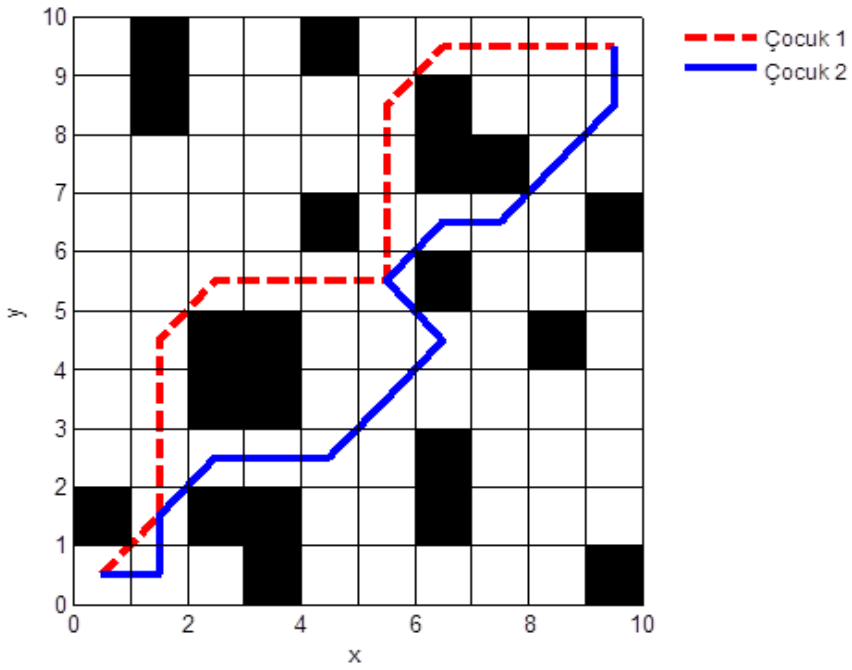
Adım 4: Çaprazlama

RYPP' de üretilen bireylerden yeni nesiller oluşturulurken çaprazlama operatörü kullanılmaktadır. RYPP için uygulanabilirliğinin kolay olması nedeniyle tek noktada çaprazlama kullanılmıştır. Standart Genetik algoritmada bireylerin uzunlukları (gen sayısı) eşit olduğu için, her birey her bireyle herhangi bir noktada kolayca çaprazlanabilmektedir. Ancak RYPP'de problemin yapısı gereği bireylerin uzunlukları farklı olabilmektedir. Örneğin, bir bireyin rotasında 20 nokta bulunurken, bir başka birey hedefe 37 noktaya uğrayarak ulaşabilir. Ayrıca, rotalarında başlangıç ve hedef noktası dışında ortak uğradıkları noktalar olabildiği gibi hiç ortak nokta olmayadabilir. Bu nedenle tez çalışmasında problemin yapısına uygun bir çaprazlama yöntemi önerilmiştir.

Şekil 4.4'te popülasyondan rastgele seçilen iki birey, Ebeveyn 1 ve Ebeveyn 2 gösterilmiştir. Bu iki bireyin farklı rotaları mevcuttur ancak başlangıç ve hedef noktası dışında ortak olarak 9. ve 44. hücreye uğramaktadırlar. Ortak ziyaret edilen noktalardan birisi rasgele seçilerek çaprazlama noktası olarak belirlenir. Bu örnekte çaprazlama noktası 44 numaralı hücre seçilmiş olsun. Bu nokta baz alınarak Şekil 4.5'te görüldüğü gibi tek noktada çaprazlama işlemi yapılmıştır.



Şekil 4.4: Çaprazlanacak ebeveynler ve noktalar



Şekil 4.5: Çaprazlama sonucu oluşan çocuklar

Ebeveyn 1, Ebeveyn 2, Çocuk 1 ve Çocuk 2'nin uygunluk değerleri sırasıyla, 15.0711, 17.0711, 16.2426 ve 15.8995 birimdir.

Çaprazlama sonucu yeni popülasyona aktarılmak üzere, toplam uygunluk değerleri göz önüne alınarak Ebeveyn 1 Çocuk 1 ile, Ebeveyn 2 de Çocuk 2 ile ikili

turnuva seçimine tabi tutulur. Ebeveyn 1'in uygunluk değeri Çocuk 1'in uygunluk değerinden daha iyi olduğu için Ebeveyn 1 popülasyonda kalır. Çocuk 2'in uygunluk değeri Ebeveyn 2'nin uygunluk değerinden daha iyi olduğu için de Ebeveyn 2 silinerek, Çocuk 2 popülasyona dahil edilir.

Adım 5: Mutasyon

Ebeveyn ve çocuklardan elde edilen bireylerin gen çeşitliliğinin sağlanması için popülasyonun belirli bir oranına mutasyon uygulanır. Şekil 4.6'da robotun hareketini temsil eden bir birey gösterilmektedir ve bu bireyin mutasyona uğramış hali Şekil 4.7'de gösterilmiştir.

Mutasyona uğrayacak birey aşağıdaki gibi olsun:

Birey → [1,2,9,17,18,19,27,**36**,44,53,54,62,71,79]

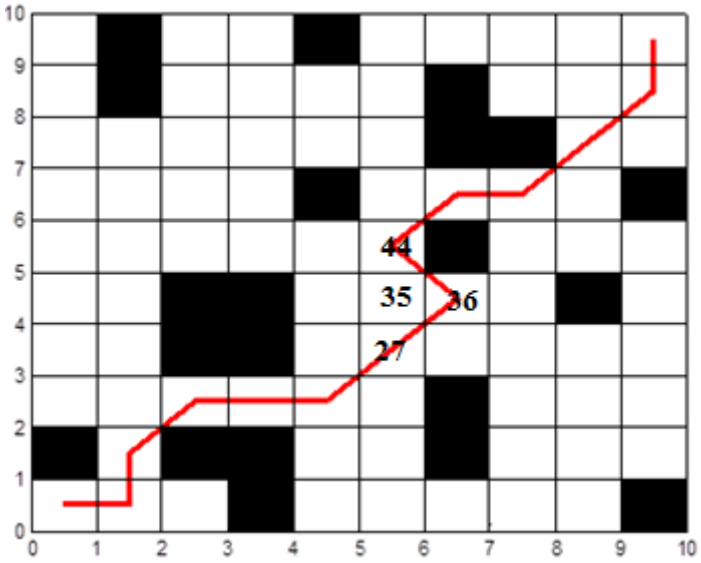
Rassal olarak bir nokta seçilir. Seçilen bu nokta 36 numaralı nokta olsun. Seçilen bu noktadan hemen önce ve sonra gelen hücreler arasında (27,44), 36 numaralı hücreye uğranmadan gidilebilecek daha yakın bir yol aranır. Bu yol aranırken, 27 ve 44 numaralı hücrelerin her ikisine de komşuluğu olan hücreler kümesi tespit edilir (34,35) ve bu kümeden rassal olarak bir eleman seçilir (35). Artık birey mutasyona uğramaya hazırdır. Bireyde 36 numaralı genin yerine 35 numaralı gen geçer. Eğer 27 ve 44 numaralı hücre doğrudan birbiriyle komşu ise de aradaki gen çözümden çıkarılır.

27→19,20,26,28,**34**,**35**,36

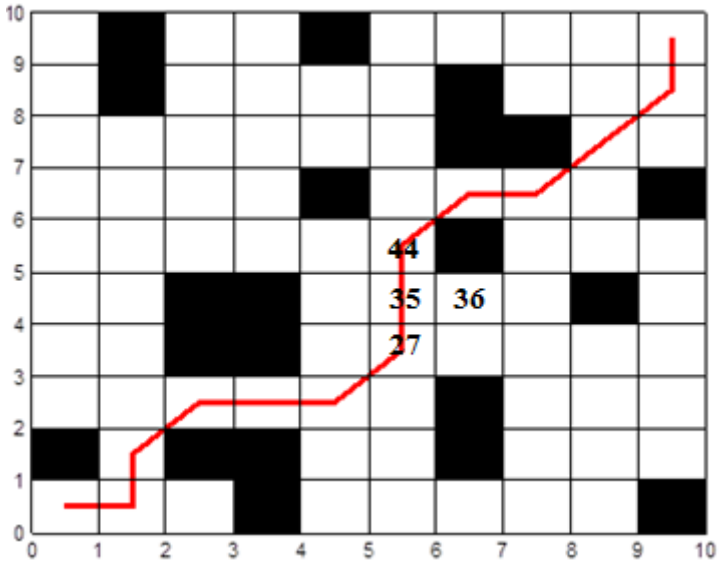
44→**34**,**35**,36,43,52,53

Mutasyona uğrayacak birey → [1,2,9,17,18,19,27,**36**,44,53,54,62,71,79]

Mutasyona uğramış birey → [1,2,9,17,18,19,27,**35**,44,53,54,62,71,79]



Şekil 4.6: Mutasyona uğrayacak birey



Şekil 4.7: Mutasyona uğramış birey

Adım 6: Durdurma kriteri

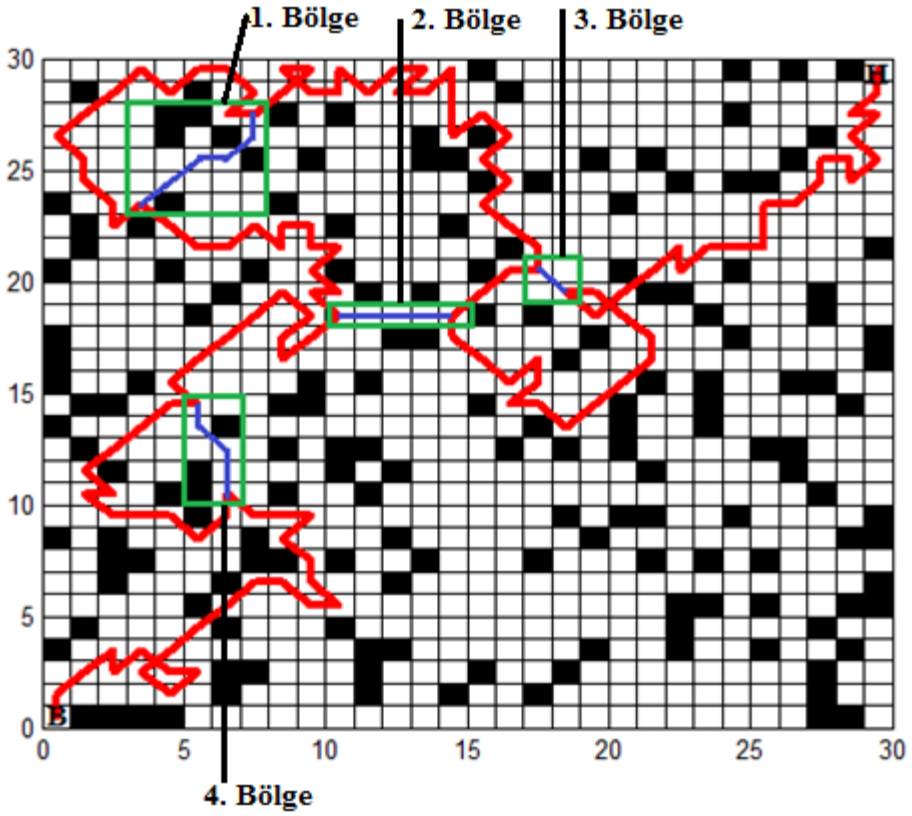
Belirlenen iterasyon sayısına ulaşıldığında algoritma sonlandırılır.

4.1.2 Hibrit Genetik Algoritma

Bu tez kapsamında Dijkstra Algoritması ve Genetik Algoritma kullanılarak hibrit bir çözüm önerilmiştir. Genetik algoritmanın içine, Dijkstra algoritması uygulanarak daha iyi bir çözüme daha hızlı bir yakınsama sağlamak hedeflenmiştir.

Bölüm 4.1.1' de verilen kodlama, başlangıç popülasyonu üretme, uygunluk değerini hesaplama ve mutasyon operatörleri HGA'da da aynıdır. Farklı olarak, mutasyon işleminden sonra, çözüme Dijkstra algoritması uygulanmaktadır. Böylelikle yöntem hibrit hale gelmektedir. Bu adım popülasyondaki tüm bireyler için uygulanmaktadır.

Öncelikle karar verici tarafından Dijkstra algoritması uygulanacak maksimum hücre sayısı belirlenir. Ardından, bireyin rassal olarak bir noktası seçilir. Rasgele seçilen bu nokta ile yine bireyin bir üyesi olan rasgele seçilmiş ikinci bir nokta arasına Dijsktra algoritması uygulanır. İkinci nokta, karar vericinin belirlediği hücre bölgesinde çözümün bir elemanı olmalıdır. Burada belirlenen maksimum hücre sayısını açacak olursak, bu sayı alanın büyüklüdür. Örneğin karar verici Dijkstra uygulanacak alanı 25 olarak belirlemişse, rasgele seçilen genin 2x2, 2x8, 5x5, 3x7, 3x8,..,1x25 boyutundaki ulaşılabilecek (iki nokta arasında yol olmayabilir) tüm komşuluklarına bakılır. Burada önemli olan seçilen komşuyla oluşturulan alanın 25 birim kareyi aşmamasıdır. Bu komşuluklarda, yine çözümde olan başka bir genin tespit edilmesi gerekmektedir. Rasgele seçilen nokta ile, 25 birimlik komşuluk içinde kalan ve yine çözümün bir parçası olan herhangi bir nokta arasına Dijkstra algoritması uygulanır.



Şekil 4.8: İGA içinde Dijkstra Algoritmasının uygulanışı

Şekil 4.8'de verilen bir çözüme Dijkstra algoritması uygulanacak olsun. Sol üst köşede yer alan 1. Bölgeyi ele alalım. Bu bölgenin sol alt köşesindeki hücre Dijkstra uygulayacağımız bölgenin başlangıç noktasıdır. Bu noktanın etrafında yer alan ve 25 birim kareye kadar olan komşuluklara bakılmış olsun ve bu komşulukta yer alan ikinci bir nokta olarak da 1. Bölgenin sağ üst köşesindeki hücre rasgele seçilmiş olsun. Seçilen bu nokta da Dijkstra algoritmasında hedef nokta olarak ele alınmaktadır. Başlangıç ve hedef noktaları belirli olan bu alana algoritma uygulandığında, çözümün bir bölümü iyileştirilmektedir. Algoritmanın uygulanacağı hücre üst sınırının, performansa etkisi büyüktür. Burada üst sınır olarak ifade edilen değer için (25), 5x5'lik bir alana bakılırken, 2x2'lik, 1x5'lik vb. alanlar da dikkate alınmaktadır. Şekil 4.8'de 2. Bölgede 1x5'lik bir komşuluğa Algoritma uygulandığında çözümde ciddi iyileştirmeye gidilmiştir. 3. Bölge ve 4. Bölge'de de aynı mantıkla kayda değer iyileştirmeler göze çarpmaktadır. Ancak şuna dikkat edilmelidir ki, üst sınırın problemin yapısına göre makul bir değer de olması

gerekmektedir. Çok büyük olması durumunda, arama uzayı çok büyük olacağından hesaplama zorlukları ortaya çıkacaktır.

Dijkstra algoritması uygulandıktan sonra İGA'da bir sonraki iterasyona geçilir veya durdurma kriteri sağlanmışsa algoritma sonlandırılır.

4.2 Çok Amaçlı Robot Yol Planlama Problemi için Önerilen Optimizasyon Metotları

ÇARYPP'nin çözümü için İGA ve HGA kullanılmıştır. Dijkstra ve Bellman-Ford algoritmalarının kullanımı düzgünlük amacının hesaplanışına uygun değildir. Bu algoritmalar bir düğümden başka bir düğüme geçişleri dikkate almaktadır. Ancak düzgünlük amacı hesaplanırken ardışık dört düğüm kullanıldığı için Dijkstra ve Bellman- Ford algoritmaları uygulanamamaktadır.

4.2.1 İyileştirilmiş Genetik Algoritma

Bu bölümde RYPP çok amaçlı olarak ele alınmıştır. Ele alınan amaçlar sırasıyla mesafe $(f_1(p))$, güvenlik $(f_2(p))$ ve düzgünlüktür$(f_3(p))$. Performans göstergeleri; bir rota boyunca (p: ziyaret edilen noktalar kümesi) kat edilen toplam yolun en küçüklenmesi, robotun engellere mümkün olduğunca uzak noktalardan geçmesi ve az sayıda manevra yaparak hedefe ulaşmasıdır. Bu amaçların hesaplanması Bölüm 2.2.2'de detaylı olarak anlatılmıştır (bkz. Denklem (2.3),(2.4),(2.5))

TARYPP için önerilen genetik algoritmanın tüm adımları ÇARYPP için de geçerlidir. Farklı olan tek adım Bölüm 4.1.1'de verilen TARYPP için İGA'nın 3. adımındaki uygunluk değerinin hesaplanma biçimidir.

Uygunluk değeri hesaplanırken, her bir amacın diğer amaçlarla kıyaslanabilmesi için normalize edilmesi gerekmektedir. Bu çalışmada Denklem 4.2'de verildiği gibi amaçlar normalize edilmiştir.

$$f_{normalize(i)}(p) = \frac{f_i\,(p) - f_{\min(i)}}{f_{\max(i)} - f_{\min(i)}} \qquad (i = 1,2,3) \tag{4.2}$$

Çok amaçlı RYPP'nin, her bir amaca eşit ağırlık verilerek (w_i) , skalerleştirilmiş amaç fonksiyonu Denklem 4.3'te verilmiştir:

$$min \sum_{i=1}^{3} w_i f_{normalize(i)}(p) \tag{4.3}$$

4.2.2 Hibrit Genetik Algoritma

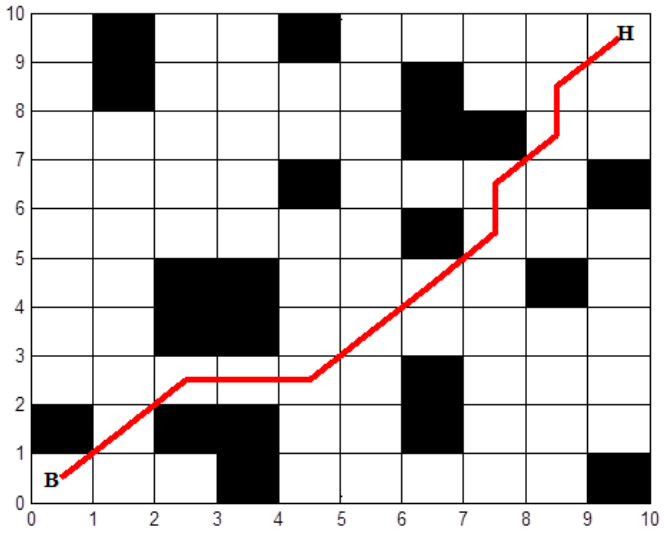
ÇARYPP için Bölüm 4.2.1'de önerilen İGA, bu bölümde HGA olarak ele alınmıştır. HGA'nın tüm adımları Bölüm 4.1.2'de verilen TARYPP için HGA ile aynıdır. Farklı olarak, problem çok amaçlı olduğu için, uygunluk değerinin hesaplanması adımı değişikliğe uğramıştır. Uygunluk değerinin hesaplanma şekli de, ÇARYPP için önerilen İGA ile aynıdır (bkz. Bölüm 4.2.1.)

5. TEK AMAÇLI ROBOT YOL PLANLAMA PROBLEMİ İÇİN BENZETİM SONUÇLARI

TARYPP optimizasyon algoritmaları Intel® Core™ i7-4712 MQ CPU @ 2.30 GHz, 8 GB RAM özelliklerine sahip bir bilgisayarda çözdürülmüştür. Bu tez kapsamında ele alınan tüm problemlerin çözümünde aynı özellikteki bilgisayar kullanılmıştır. Bölüm sonunda yer alan Tablo 5.2'de tüm metotlarla elde edilen sonuçlar yer almaktadır.

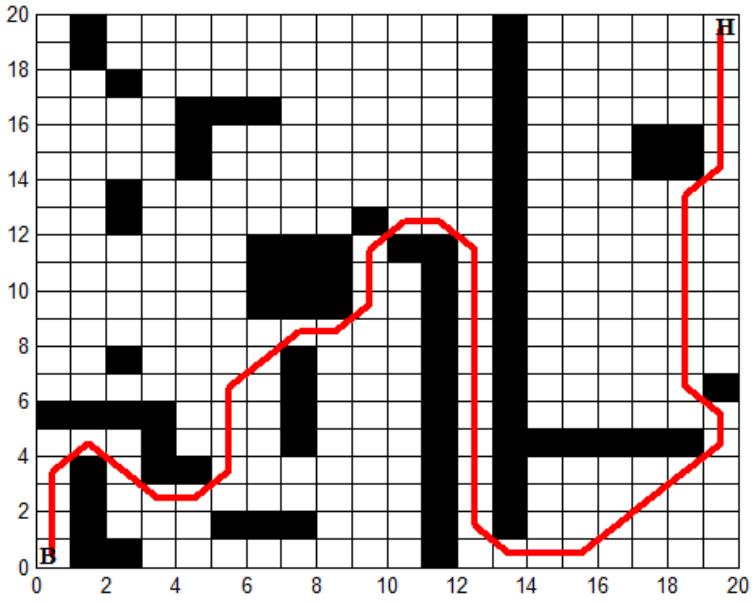
5.1 Dijkstra Algoritması

RYPP için iki farklı 10x10 ve 20x20'lik, engelli ortamda, bir robotun Dijikstra Algoritması ile bir başlangıç noktasından (B) hedef noktasına (H) en kısa yoldan ulaşması sağlanmıştır.



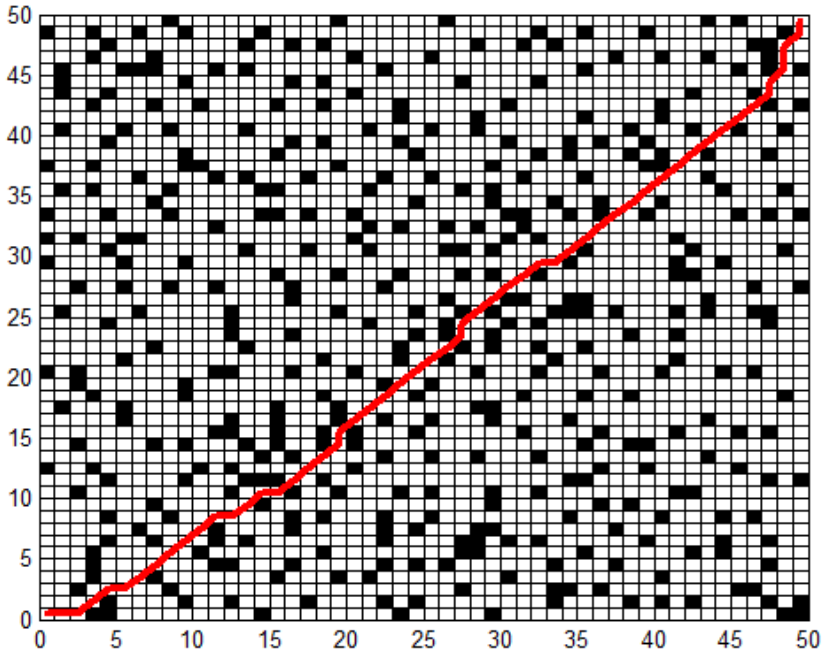
Şekil 5.1: 10x10'luk çevrede TARYPP için Dijkstra Algoritması sonucu

Şekil 5.1'de gösterildiği gibi 10x10'luk bir ortamda 79 engelsiz ve 21 engelli nokta mevcuttur. Robotun başlangıç (B) koordinatlarından (0.5,0.5) harekete başlayarak hedef (H) koordinatlarına (9.5,9.5) ulaştığı en kısa yol şekilde görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **13,8995** birimdir. Ele alınan problemin çözümüne **0,026975** saniyede ulaşılmıştır.



Şekil 5.2: 20x20'lik çevrede TARYPP için Dijkstra Algoritması sonucu

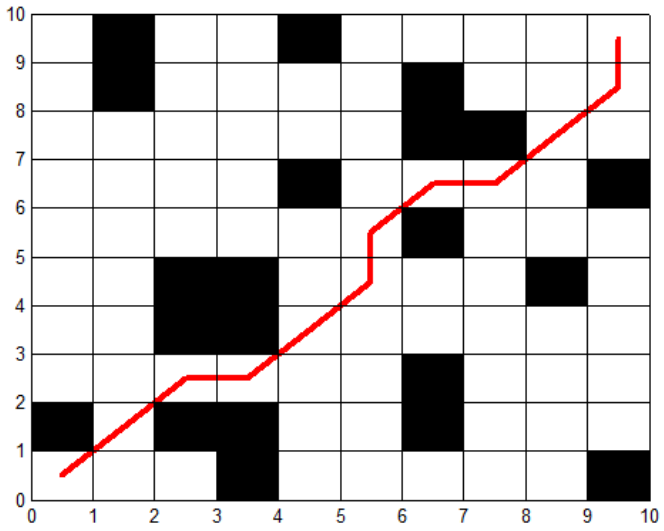
Şekil 5.2'de gösterildiği gibi 20x20'lik bir ortamda 318 engelsiz ve 82 engelli nokta mevcuttur. Robotun başlangıç (B) koordinatlarından (0.5,0.5) harekete başlayarak hedef (H) koordinatlarına (19.5,19.5) ulaştığı en kısa yol şekilde görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **59,8701** birimdir. Ele alınan problemin çözümüne **0,352001** saniyede ulaşılmıştır.



Şekil 5.3: 50x50'lik çevrede TARYPP için Dijkstra Algoritması sonucu

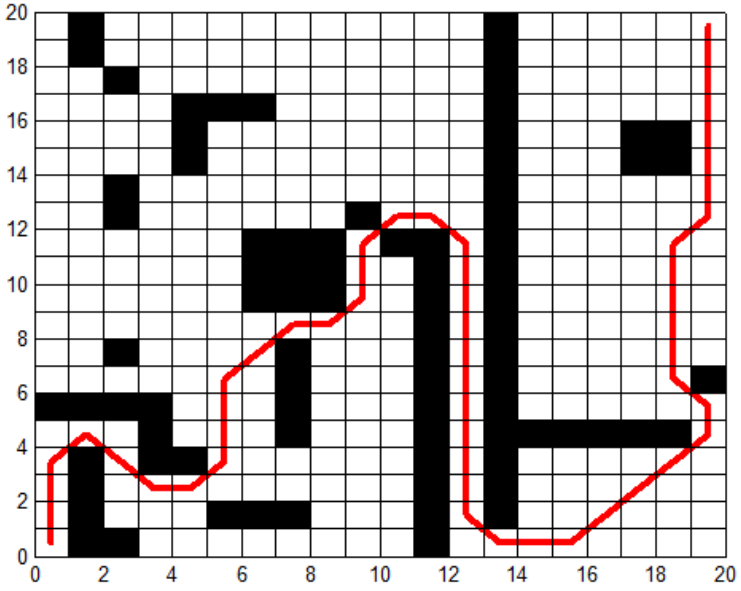
50x50'lik bir çevrede, robotun başlangıç noktasından (0.5,0.5) harekete başlayarak hedef noktaya (49.5,49.5) ulaştığı en kısa yol Şekil 5.3'te görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **72,8112** birimdir. Ele alınan problemin çözümüne **16,604** saniyede ulaşılmıştır.

5.2 Bellman-Ford Algoritması



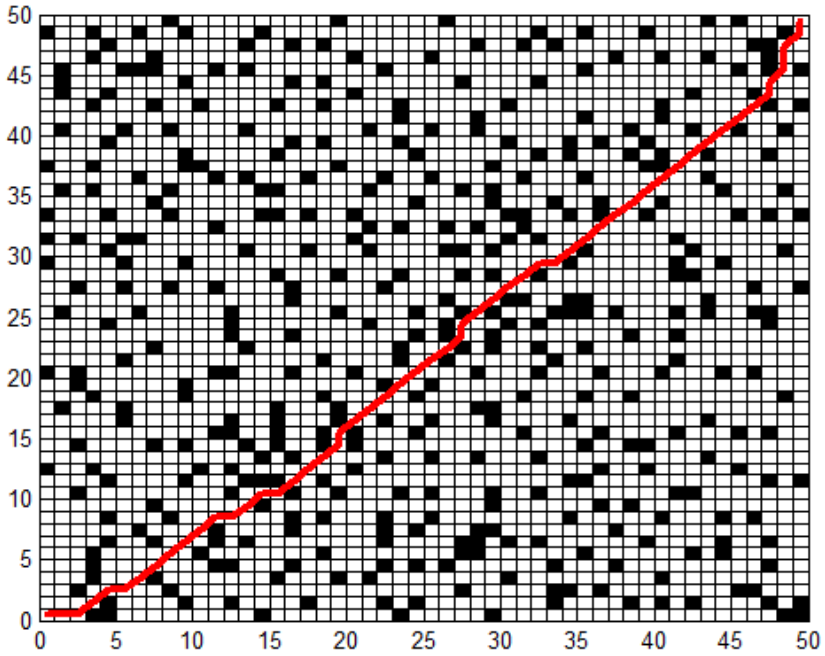
Şekil 5.4: 10x10'luk çevrede TARYPP için Bellman-Ford Algoritması sonucu

Şekil 5.4'te, Bölüm 5.1'de ele alınan 10x10'luk problemin Bellman - Ford Algoritması ile çözümü görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **13,8995** birimdir. Ele alınan problemin çözümüne **0,02734** saniyede ulaşılmıştır.



Şekil 5.5: 20x20'lik çevrede TARYPP için Bellman-Ford Algoritması sonucu

Şekil 5.5'te, bir önceki bölümde ele alınan 20x20'lik problemin Bellman - Ford Algoritması ile çözümü görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **59,8701** birimdir. Ele alınan problemin çözümüne **0,35194** saniyede ulaşılmıştır.



Şekil 5.6: 50x50'lik çevrede TARYPP için Bellman-Ford Algoritması sonucu

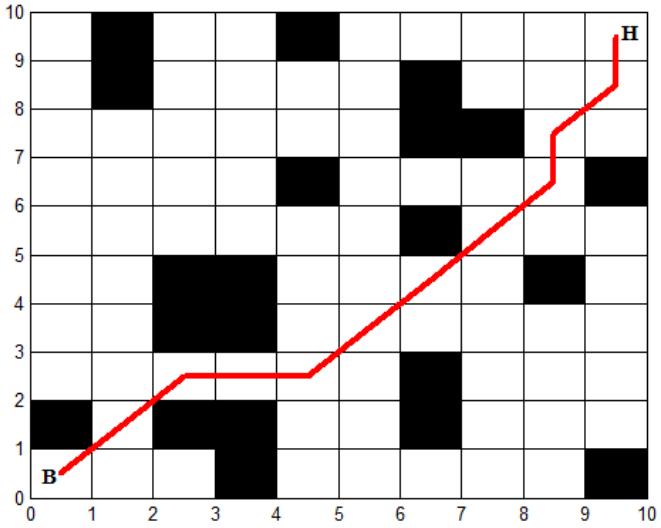
50x50'lik bir çevrede, robotun başlangıç noktasından (0.5,0.5) harekete başlayarak hedef noktaya (49.5,49.5) ulaştığı en kısa yol Şekil 5.6'da görülmektedir.

Problemin en iyi çözümüne ait en kısa yolun uzunluğu **72,8112** birimdir. Ele alınan problemin çözümüne **17,104** saniyede ulaşılmıştır.

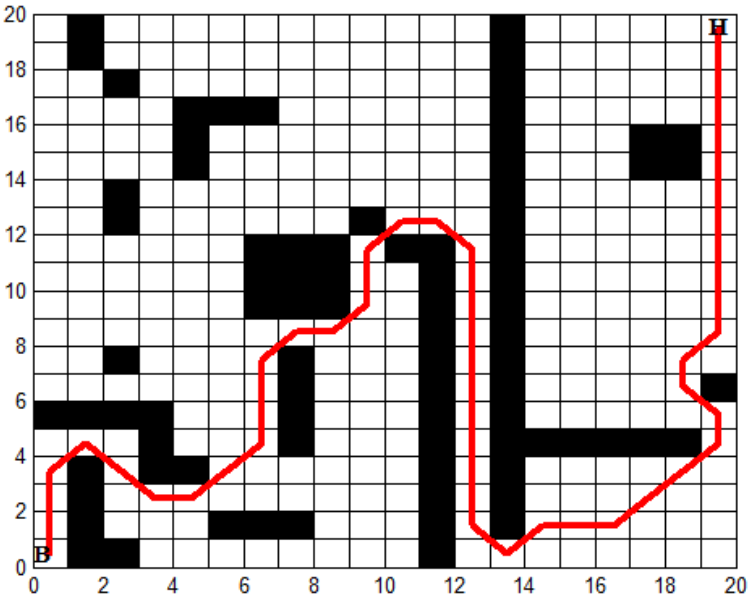
5.3 0-1 Tam Sayılı Doğrusal Programlama Modeli

Bölüm 6.1 ve 6.2'de ele alınan 10x10'luk ve 20x20'lik örnek problemlerin sonucuna 0-1 Tam Saylı Doğrusal Programlama Model ile de ulaşılmıştır. Matematiksel modelin sonuçları GAMS 24.3.3 programında CPLEX çözücüsü ile elde edilmiştir.

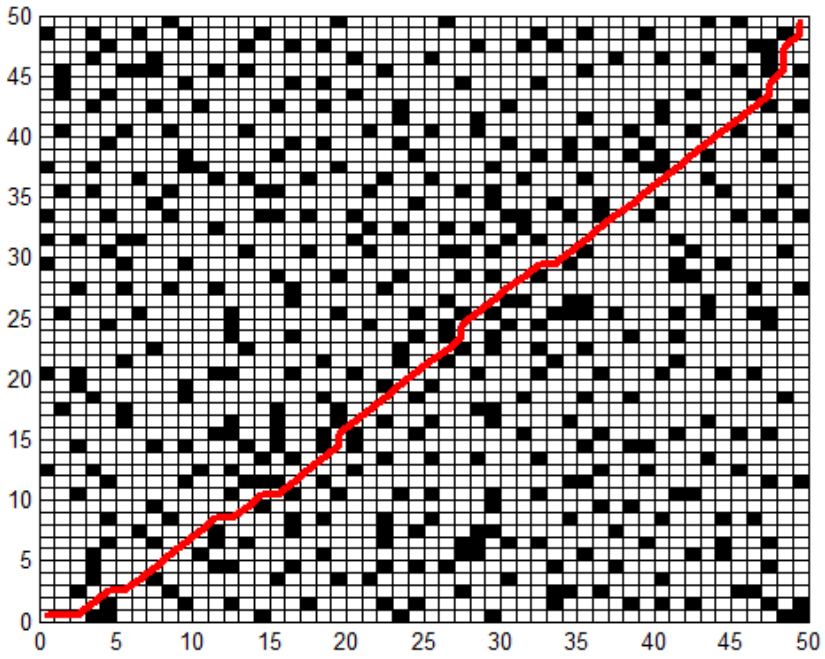
Şekil 6.5'te 10x10'luk, Şekil 6.6'da ise 20x20'lik problemler için elde edilen sonuçların gösterimi yer almaktadır. 10x10'luk problemin en iyi çözümüne ait en kısa yolun uzunluğu **13,8995** birimdir ve problemin çözümüne **0,09** saniyede ulaşılmıştır. 20x20'lik problemin en iyi çözümüne ait en kısa yolun uzunluğu **59,8701** birimdir ve problemin çözümüne **2.1341** saniyede ulaşılmıştır.



Şekil 5.7: 10x10'luk çevrede TARYPP için 0-1 Tam sayılı doğrusal programlama sonucu



Şekil 5.8: 20x20'luk çevrede TARYPP için 0-1 Tam sayılı doğrusal programlama sonucu



Şekil 5.9: 50x50'lik çevrede TARYPP için 0-1 Tam sayılı doğrusal programlama sonucu

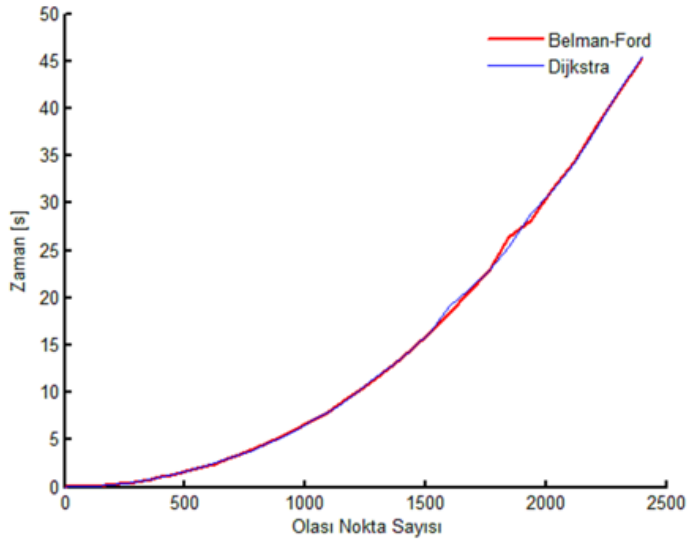
50x50'lik bir çevrede, robotun başlangıç noktasından (0.5,0.5) harekete başlayarak hedef noktaya (49.5,49.5) ulaştığı en kısa yol Şekil 5.9'da görülmektedir. Problemin en iyi çözümüne ait en kısa yolun uzunluğu **72,8112** birimdir. Ele alınan problemin çözümüne **126,32** saniyede ulaşılmıştır.

Her üç kesin çözüm yöntemi ile aynı sonuçlara ulaşılmıştır (Dijkstra, Bellman-Ford, 0-1 Tam sayılı doğrusal programlama modeli) ancak dikkat çeken nokta robotlar aynı amaç fonksiyonu değerine ulaşmalarına rağmen farklı rotalar çizebilmektedir. Bu da en iyi çözümü veren alternatif çözümler olduğunu göstermektedir.

0-1 Tam sayılı doğrusal programlama modeli CPLEX çözücü ile çözdürüldüğünde amaç fonksiyonu değeri diğer algoritmalarla aynı olsa da süre olarak daha kötü sonuçlar vermiştir. Dijkstra ve Bellman- Ford Algoritmaları da birbirine yakın sonuçlar vermiş birbirlerine karşı süre anlamında bir üstünlük sağlayamamışlardır. Bu nedenle hem bu iki algoritmayı karşılaştırmak hem de problemin boyutunun değişiminde nasıl sonuçlar elde edildiğini göstermek için Bölüm 5.4'te bu algoritmaların çözüm karmaşıklığı anlatılmaktadır.

5.4 Dijkstra ve Bellman-Ford Algoritmalarının Çözüm Karmaşıklığı

Dijkstra ve Bellman - Ford Algoritmaları 10x10'luk, 20x20'lik ve 50x50'lik engelli bir çevrede aynı amaç fonksiyonu değerine ulaşmakta ve en kısa yolu bulmayı garanti etmektedir.



Şekil 5.10: RYPP için Dijkstra ve Bellman –Ford algoritmalarına ait çözüm süreleri

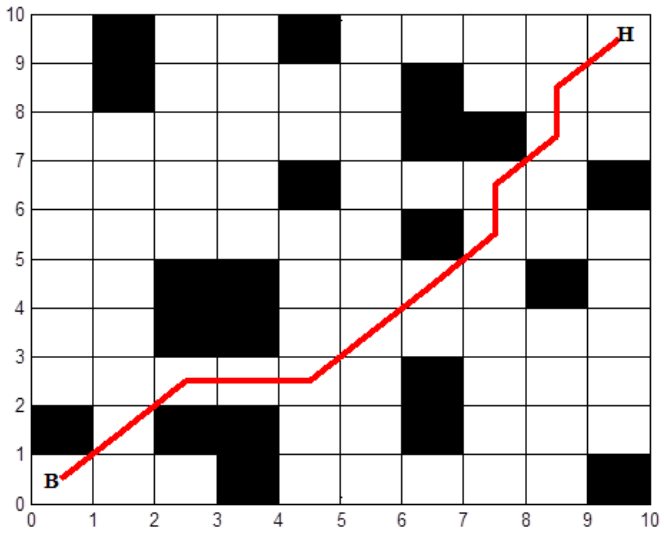
Dijkstra ve Bellman-Ford algoritmalarının farklı boyutlardaki aynı problemlerin çözüm sürelerine ait grafik Şekil 5.10'da verilmiştir. Problemin çözüm

süreleri her iki algoritmada yakın değerler vermektedir. Grafikte bahsedilen olası nokta sayısı, problem alanında yer alan ve robotun uğrayabileceği olası nokta sayısıdır. Bu olası nokta sayısı engelsiz bir ortamda 10x10'luk bir problem için 100 iken 20x20'lik bir problem için 400'dür. RYP probleminin karmaşıklığı grafikten de açıkça anlaşılacağı gibi üstel olarak artmaktadır. Bu nedenle problem boyutu arttıkça Dijkstra ve Bellman Ford Algoritmalarının kısa zamanda çözüme ulaşması zorlaşmakta ve daha zor problemler için farklı çözüm yöntemlerinin kullanılması kaçınılmaz hale gelmektedir.

5.5 İyileştirilmiş Genetik Algoritma

TARYPP için 10x10'luk örnek için elde edilen en iyi sonuç Şekil 5.11'de verilmiştir. Problemin çözümü için farklı iterasyon, popülasyon sayıları ve mutasyon oranlarında çözümler aranmış, en uygun parametre kombinasyonu aşağıdaki gibi belirlenmiştir:

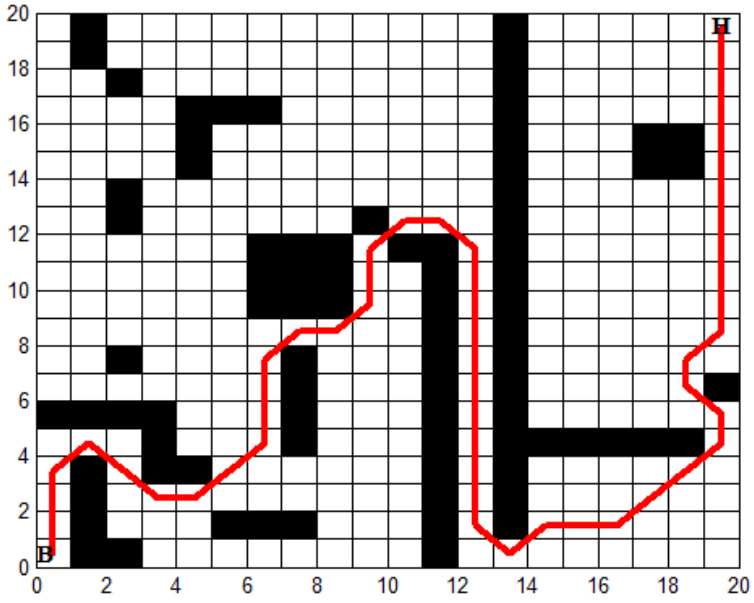
İterasyon sayısı=50, Popülasyon büyüklüğü=10, Mutasyon oranı =%10



Şekil 5.11: TARYPP 10x10'luk örnek için İGA ile elde edilen sonuç

Problemin en iyi çözümüne 47. iterasyonda 1,02 saniyede ulaşılmıştır ve amaç fonksiyonu değeri 13,8995 olarak hesaplanmıştır. Bu değer, aynı örnek için kesin

çözüm yöntemleri ile elde edilen sonuçlarla aynıdır yani problemin optimum çözümüne İGA ile de ulaşılmıştır.

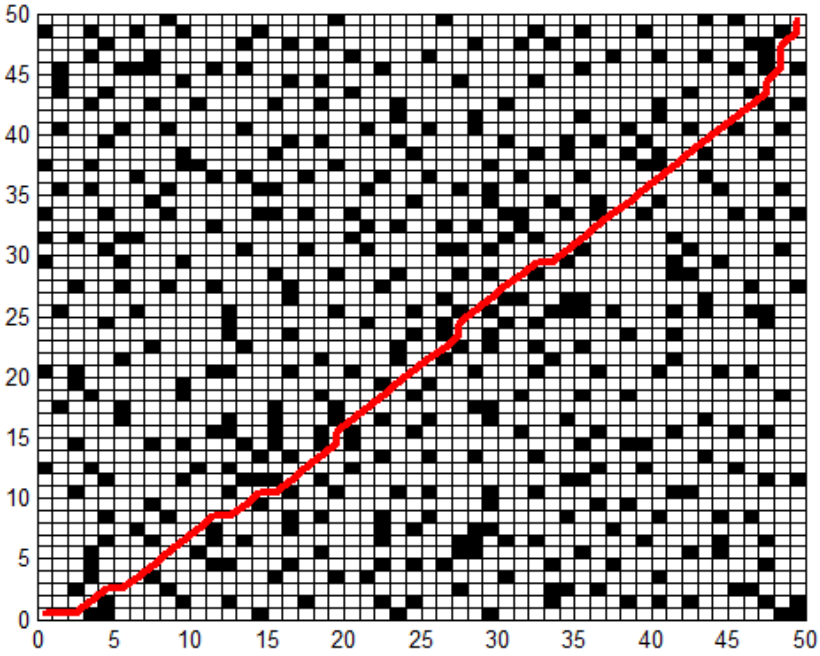


Şekil 5.12: TARYPP 20x20'lİk örnek için İGA ile elde edilen sonuç

TARYPP için 20x20'lik örnek üzerinde çalışılmıştır. Şekil 5.12'de bu problem için elde edilen en iyi sonuç verilmiştir. Problemin çözümü için farklı iterasyon, popülasyon sayıları ve mutasyon oranlarında çözümler aranmış, en uygun parametre kombinasyonu aşağıdaki gibi belirlenmiştir:

İterasyon sayısı=150, Popülasyon büyüklüğü =20, Mutasyon oranı =%8

Problemin en iyi çözümüne 102. iterasyonda **4,097** saniyede ulaşılmıştır ve amaç fonksiyonu değeri 59,8701 olarak hesaplanmıştır.



Şekil 5.13: TARYPP 50x50'lik örnek için İGA ile elde edilen sonuç

TARYPP için daha büyük boyutlu bir problem ele alınmış ve 50x50'lik örnek üzerinde çalışılmıştır. Şekil 5.13'te bu probleme için elde edilen en iyi sonuç verilmiştir. Problemin çözümü için farklı iterasyon, popülasyon sayıları ve mutasyon oranlarında çözümler aranmış, en uygun parametre kombinasyonu aşağıdaki gibi belirlenmiştir:

İterasyon sayısı=300, Popülasyon büyüklüğü=50, Mutasyon oranı =%5

Problemin en iyi çözümüne 281. iterasyonda 11,768 saniyede ulaşılmıştır ve amaç fonksiyonu değeri 72,8112 olarak hesaplanmıştır.

5.6 Hibrit Genetik Algoritma

Şekil 5.11, Şekil 5.12 ve Şekil 5.13'te yer alan çözümlere ulaşmak için HGA uygulanmıştır. Aynı sonuçlara ulaşılmış ancak, algoritma hibrit hale getirildiği için bu sonuçlara daha az iterasyonda ulaşılmıştır. Aynı parametre kombinasyonları için, 10x10'luk problemin çözümüne HGA ile 28. iterasyonda **0,583** saniyede, 20x20'lik problemin çözümüne 69. iterasyonda **2,811** saniyede, 50x50'lik problemin çözümüne

ise 153. iterasyonda **7,246** saniyede ulaşılmıştır. İGA ve HGA ile elde edilen bu sonuçlar iterasyon sayısı bakımından Tablo 5.1'de verildiği gibi karşılaştırılabilir.

Tablo 5.1: TARYPP için İGA ve HGA karşılaştırması

Metot	10x10'luk örnek		20x20'lik örnek		50x50'lik örnek	
	İterasyon Sayısı	Amaç Değeri	İterasyon Sayısı	Amaç Değeri	İterasyon Sayısı	Amaç Değeri
İGA	47	13,8995	102	59,8701	281	72,8112
HGA	28	13,8995	69	59,8701	153	72,8112

Dijkstra ve Bellman-Ford Algoritmaları, 0-1 tam sayılı doğrusal programlama modeli, İGA ve HGA ile farklı boyutlarda ancak aynı problemlerde elde edilen sonuçlar için çözüme ulaşma süreleri ve amaç fonksiyonu değerleri Tablo 5.2'de özetlenmiştir.

Tablo 5.2: TARYPP için optimizasyon metotları sonuçları

Metot	10x10'luk örnek		20x20'lik örnek		50x50'lik örnek	
	Süre (sn)	Amaç Değeri	Süre (sn)	Amaç Değeri	Süre (sn)	Amaç Değeri
Dijkstra	0,026975	13,8995	0,352001	59,8701	16,604	72,8112
Bellman-Ford	0,02734	13,8995	0,35194	59,8701	17,104	72,8112
0-1 tam sayılı model	0,09	13,8995	2,1341	59,8701	126,32	72,8112
İGA	1,02	13,8995	4,097	59,8701	11,768	72,8112
HGA	0,583	13,8995	2,811	59,8701	7,246	72,8112

Elde edilen sonuçlara göre küçük boyutlu problemlerde kesin çözüm yöntemleri olan Dijkstra ve Bellman-Ford algoritmaları ve 0-1 tam sayılı doğrusal programlama modeli TARYPP'ye daha kısa zamanda çözüm bulmaktadır. Problemlerin boyutu arttıkça bu yöntemlerin etkinliği azalmaktadır. İGA ve HGA küçük boyutlu problemlerde diğer yöntemlerle çözüm süresi bakımından rekabet edemese de problemlerin boyutu arttıkça ciddi bir üstünlük sağlamaktadır. Ayrıca HGA'nın da İGA'dan süre ve iterasyon sayısı bakımından üstün olduğu görülmektedir.

6. ÇOK AMAÇLI ROBOT YOL PLANLAMA PROBLEMİ İÇİN BENZETİM SONUÇLARI

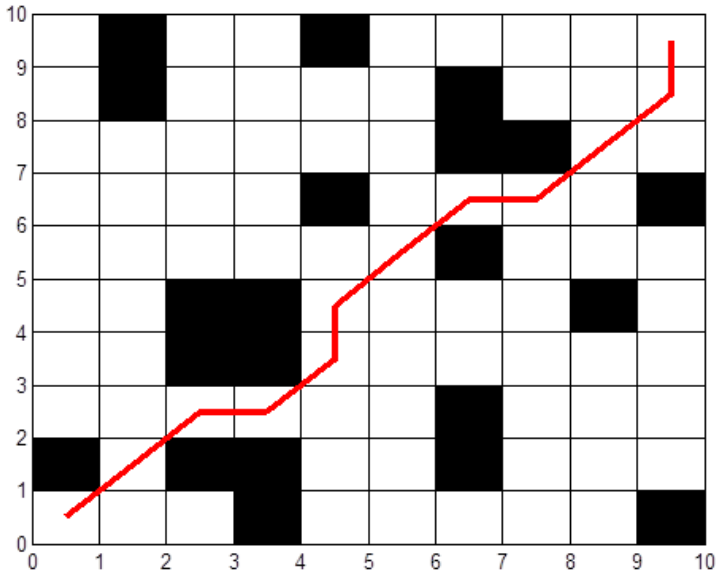
Bu bölümde ÇARYPP için önerilen İGA ve HGA metotları ile elde edilen sonuçlar yer almaktadır. Tablo 6.2 ve Tablo 6.3'te bu metotlarla bulunan sonuçlar özetlenmiştir.

6.1 İyileştirilmiş Genetik Algoritma

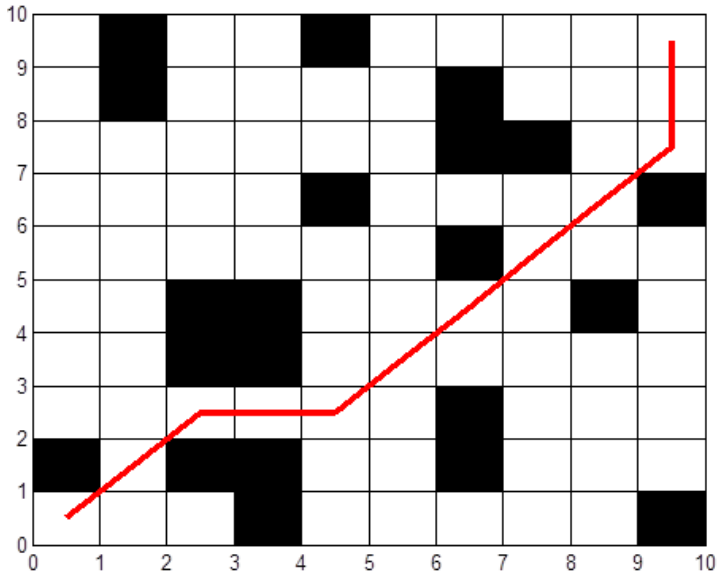
Bu bölümde ÇARYPP için önerilen İGA metodu ile elde edilen sonuçlar yer almaktadır.

Şekil 6.1'de 10x10'luk bir örnek için mesafenin en küçüklenmesine yönelik (TARYPP) elde edilen rota yer almaktadır. Aynı problem, mesafe, düzgünlük ve güvenlik amaçlarının eşit ağırlıklı olduğu durum için çözdürüldüğünde ise elde edilen sonuç Şekil 6.2'de verilmiştir. Bu 10x10'luk ÇARYPP için çözüme 28. iterasyonda **0,711** saniyede ulaşılmıştır. Bu çözüm için parametre kombinasyonu:

İterasyon sayısı = 100, Popülasyon büyüklüğü=10, Mutasyon oranı=%10'dur.



Şekil 6.1: ÇARYPP için 10x10'luk örneğin TARYPP olarak İGA ile elde edilen sonucu



Şekil 6.2: ÇARYPP 10x10'luk örnek için İGA ile elde edilen sonuç

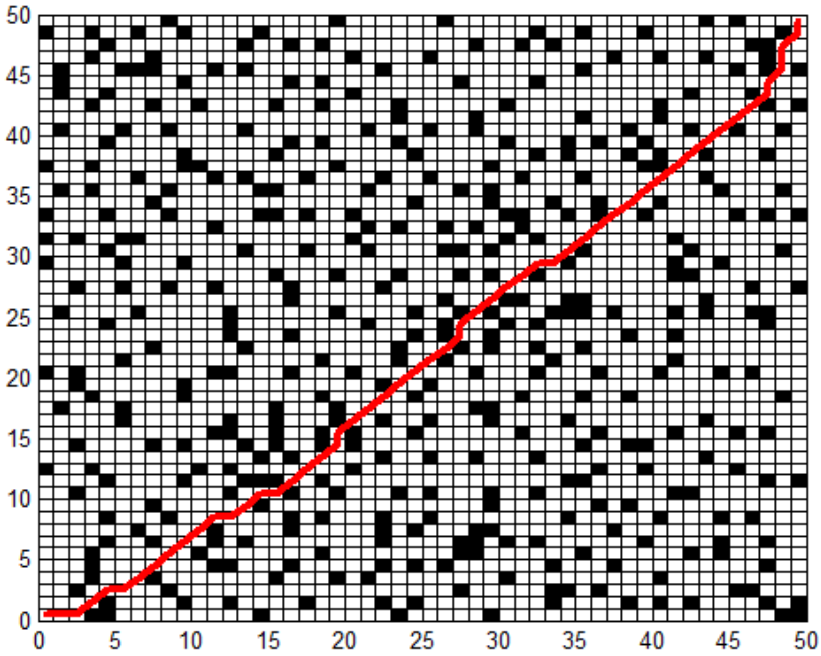
Şekil 6.1 (TARYPP) ve Şekil 6.2'de (ÇARYPP) gösterilen çözümlere ait amaçların ağırlıkları ve amaç fonksiyonu değerleri Tablo 6.1'de verildiği gibidir. Burada TARYPP için mesafe en küçüklenirken yani bu amacın ağırlığı 1 iken, diğer amaçlar göz ardı edilmektedir ve ağırlıkları 0'dır. Problem üç farklı amaç için amaçlara eşit ağırlıklar verilerek optimize edilmeye çalışıldığında ise, mesafe amacında bir değişim olmamasına rağmen, diğer iki amacın amaç fonksiyonu değerlerinin iyileştiği görülmektedir. Yani problemin mesafe amacı altında alternatif çözümleri mevcutken, diğer amaçlar dikkate alındığında tüm amaçları iyileştiren başka bir çözüme ulaşmak mümkün hale gelmektedir. Burada amaçlar arasında bir ödünleşme söz konusudur.

Tablo 6.1: TARYPP – ÇARYPP karşılaştırması

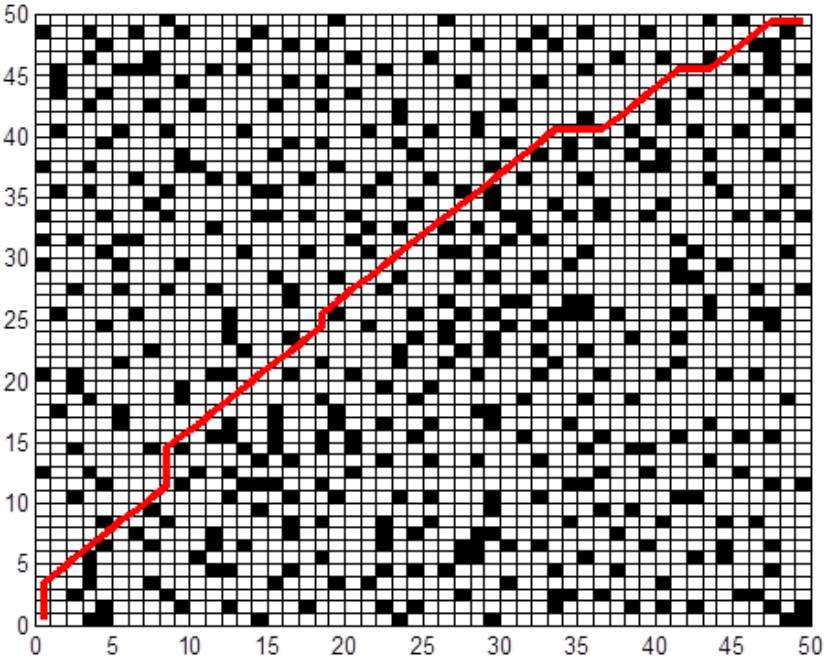
Ağırlıklar			Amaç Fonksiyonu Değerleri		
Mesafe (w_1)	Düzgünlük(w_2)	Güvenlik (w_3)	f_1 (P)	f_2 (P)	f_3 (P)
1	0	0	13,8995	12,5664	12
0,33	0,33	0,33	13,8995	6,2832	11,2251

ÇARYPP 50x50'lik bir problem amaçlara farklı ağırlıklar verilerek çözdürülmüştür. Şekil 6.3, 6.4, 6.5, 6.6, 6.7 ve 6.8'de farklı ağırlıklara ait çözümler yer almaktadır. Bu çözüm için parametre kombinasyonu:

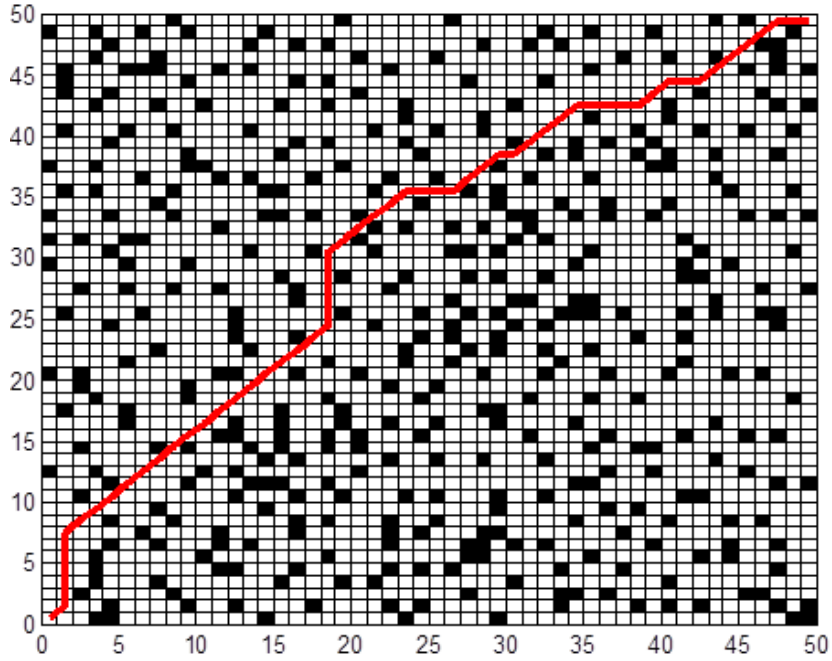
İterasyon sayısı = 500, Popülasyon büyüklüğü=50, Mutasyon oranı=%5'dur



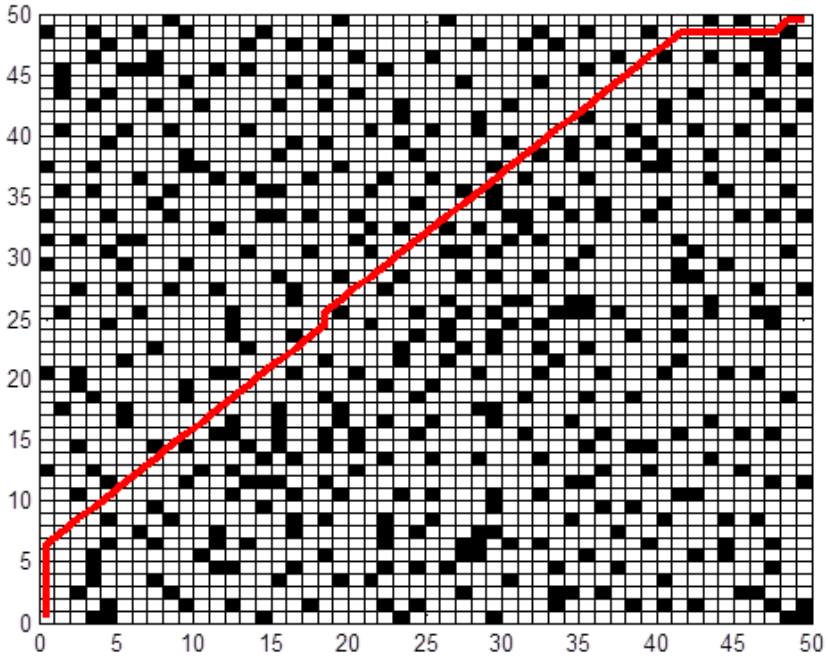
Şekil 6.3: 50x50'lik ÇARYPP için w_1=1, w_2=0, w_3=0 ağırlıklarında elde edilen sonuç



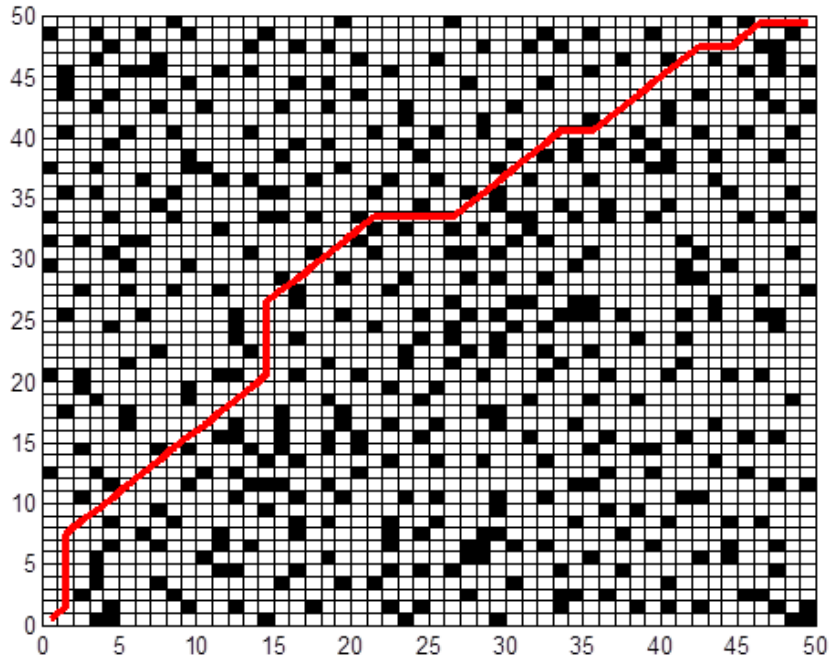
Şekil 6.4: 50x50'lik ÇARYPP için w_1=0.5, w_2=0.5, w_3=0 ağırlıklarında elde edilen sonuç



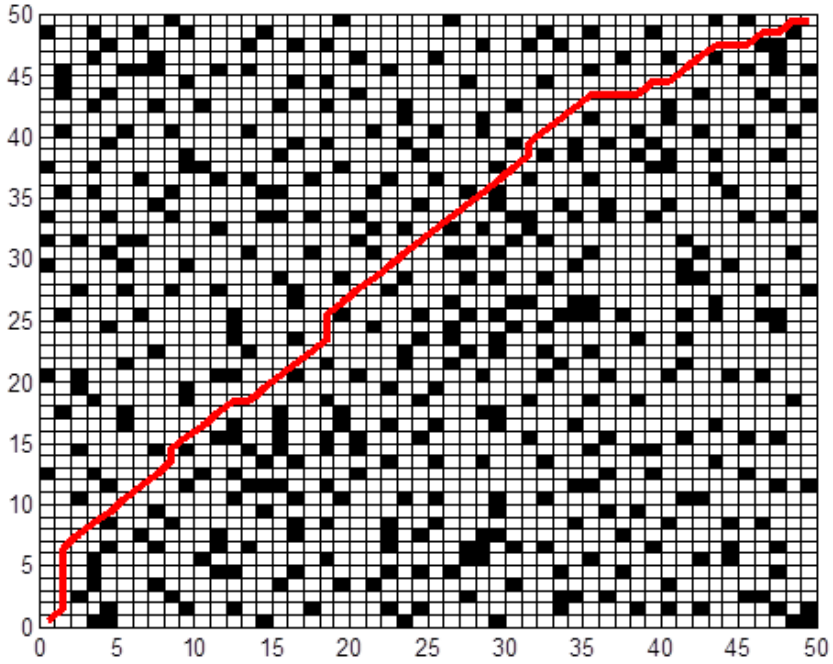
Şekil 6.5: 50x50'lik ÇARYPP için w_1=0.33, w_2=0.33, w_3=0.33 ağırlıklarında elde edilen sonuç



Şekil 6.6: 50x50'lik ÇARYPP için w_1=0, w_2=1, w_3=0 ağırlıklarında elde edilen sonuç



Şekil 6.7: 50x50'lik ÇARYPP için w_1=0, w_2=0.5, w_3=0.5 ağırlıklarında elde edilen sonuç



Şekil 6.8: 50x50'lik ÇARYPP için w_1=0, w_2=0, w_3=1 ağırlıklarında elde edilen sonuç

Tablo 6.2'de 50x50'lik ÇARYPP için elde edilen sonuçlar yer almaktadır. Her bir amacın 0 ile 1 aralığında bir ağırlığı vardır ve bu üç amacın ağırlıkları toplamı 1'dir. Bir amacın ağırlığının 1 diğerlerinin 0 olması demek, problemin aslında tek amaçlı olarak çözüldüğü anlamına gelmektedir. Herhangi iki amacın ağırlıkları toplamı 1 ve üçüncü amacın ağırlığı 0 ise de, problem iki amaçlı hale

gelmiş olmaktadır. Tablo 6.2'de farklı ağırlık kombinasyonlarına ait sonuçlar yorumlanacak olursa, herhangi bir amacın ağırlığı sabit tutulup diğer amaçlarla oynandığında, bir amacın iyileşirken diğerinin kötüleştiği görülmektedir. Örneğin güvenlik amacı sabit tutulduğunda yani ağırlığı 0 verildiğinde, uzaklık amacının ağırlığı azalıp düzgünlük amacının ağırlığı artarken, bu ağırlıklar doğrultusunda kat edilen yolun uzunluğunun arttığı veya sabit kaldığı (alternatif çözümlerin olması), düzgünlüğün ise amaç fonksiyonu değerinin azaldığı, yani daha düzgün bir rota izlendiği görülmektedir. Ayrıca tüm amaçların eşit ağırlıklı olduğu durumda ise, tek amaca göre elde edilen sonuç ile kıyasla, amaçlar arası bir ödünleşme olduğu dikkat çekmektedir. Hiçbir amaç fonksiyonu değeri olası en iyi değere ulaşamamış ancak hepsi belirli bir oranda ortak amaca hizmet ederek makul değerler almışlardır. Amaçlar arası ödünleşme yine mesafe amacının ağırlığı sabit tutulup, birbiriyle çelişen iki amaç olan düzgünlük ve güvenlik amaçlarının ağırlık değişiminde de göze çarpmaktadır. Düzgünlük amacının ağırlığı 1'den 0 a azalıp, güvenlik amacının ağırlığı 0'dan 1'e artarken, düzgünlük azalmakta ancak güvenlik artmaktadır. Daha fazla ağırlık seviyeleri ve kombinasyonları belirlenerek, farklı seviyelerde karar vericinin önceliğine göre daha farklı çözümler de elde etmek mümkündür.

Tablo 6.2: ÇARYPP için farklı ağırlıklarda elde edilen sonuçlar

Ağırlıklar			Amaç Fonksiyonu Değerleri		
Mesafe (w1)	Düzgünlük(w2)	Güvenlik (w3)	f1 (P)	f2 (P)	f3 (P)
1	0	0	72,8112	29,0597	52,5048
0,5	0,5	0	73,3970	18,0642	46,6627
0,33	0,33	0,33	76,3259	21,9911	46,5862
0	1	0	73,3970	11,7810	44.9557
0	0,5	0,5	73,3970	15,7080	42,5620
0	0	1	74,5685	31,4159	40,2623

6.2 Hibrit Genetik Algoritma

10x10'luk problemde Şekil 6.2'de elde edilen çözüme ulaşmak için HGA uygulanmıştır. Yine 50x50'lik problem için de, Tablı 6.2'de verilen değerlere ulaşmak için HGA uygulanmıştır. Yine aynı sonuçlaraa ulaşılmış ancak, TARYPP'de olduğu gibi algoritma hibrit hale getirildiği için aynı sonuçlara daha az iterasyonda ulaşılmıştır. Eşit ağırlıklar ve aynı rota için, 10x10'luk ve 50x50'lik

problemlerin çözümüne HGA ve İGA ile elde edilen sonuçlar iterasyon ve süre açısından Tablo 6.3'te karşılaştırılmıştır.

Tablo 6.3: ÇARYPP için İGA ve HGA karşılaştırması

	10x10'luk problem		50x50'lik problem	
Yöntem	İterasyon Sayısı	Süre (sn)	İterasyon Sayısı	Süre (sn)
İGA	28	0,711	304	13,217
HGA	17	0,295	182	8.946

7. SONUÇ VE ÖNERİLER

Mobil robotların kullanım alanı gün geçtikçe artmaktadır. Kendi kendine bir noktadan hedef noktaya ulaşabilme yeteneğine sahip olması beklenen bu mobil robotların, hareketleri esnasında izleyecekleri yolun belirli performans göstergelerine göre belirlenmesi zaman, enerji, sermaye ve işgücü gibi birçok kaynağın etkin kullanımı açısından önemli bir problem haline gelmiştir. Bu nedenle tez kapsamında RYPP için bir İGA ve HGA önerilmiştir. Önerilen İGA'da, çaprazlama ve mutasyon operatörlerinin kullanımı RYPP'nin yapısı gereği standart genetik algoritmadan farklılaştırılarak, bu operatörlerin kullanımı etkin hale getirilmiştir. HGA'da ise, İGA ile Dijkstra Algoritması birlikte kullanılmıştır. İGA ile kaliteli çözümler aranırken, aynı zamanda çözümlerin belirli noktalarına Dijkstra Algoritması uygulanıp bir yerel arama yapılarak daha iyi çözümlere hızlı bir yakınsama sağlanmıştır. Önerilen İGA ve HGA, TARYPP ve ÇARYPP'nin çözümünde kullanılmıştır.

TARYPP için amaç mobil robotun rotasının uzunluğunun en küçüklenmesidir. Farklı boyutlardaki problemler için Dijkstra ve Bellman-Ford Algoritmaları, 0-1 Tam Sayılı Doğrusal Programlama Modeli, İGA ve HGA ile çözümler elde edilmiştir. Problem boyutu arttıkça, önerilen yöntemlerin diğer yöntemlere göre çok daha kısa zamanda çözüme ulaştığı görülmüştür. Aynı zamanda, HGA'nın da daha iyi çözümlere hızlı bir şekilde ve daha az iterasyon ile yakınsaması nedeniyle İGA'dan daha iyi performans gösterdiği görülmüştür.

ÇARYPP'de ise optimize edilmek istenen üç amaç mevcuttur; mesafe, düzgünlük ve güvenlik. Ele alınan amaçlar Ağırlıklandırılmış Toplam Yöntemi kullanılarak uygun şekilde normalize edilip skalerleştirilmiş ve tek bir amaca dönüştürülmüştür. İGA ve HGA kullanılarak farklı boyutlardaki problemler için elde edilen çözümler iterasyon sayısı ve süre performansları bakımından kıyaslanmış HGA'nın İGA'ya göre daha başarılı olduğu kanısına varılmıştır. Aynı engelli çevrede, birden fazla amacın ve sadece mesafe amacının dikkate alındığı problemler tek ve çok amaçlı olarak çözdürülmüştür. Elde edilen sonuçlarda, mesafe amacı ile düzgünlük ve güvenilirlik amaçları arasında bir ödünleşme olduğu görülmüştür. Bir

diğer ifadeyle mesafenin amaç fonksiyonu değeri kötüleşirken, diğer amaçların amaç fonksiyonu değerleri iyileşmiştir.

Sonuç olarak, önerilen İGA ve HGA'nın TARYPP ve ÇARYPP için çözüm kalitesi ve süre bakımından iyi performans gösterdiği görülmüştür. Gelecek çalışmalarda, önerilen yöntemlerin dinamik sistemlere uygulanarak, mobil robotların sadece başlangıç noktasında değil herhangi bir noktada da rotalarını anlık belirleyebilmeleri üzerine çalışılmalar yapılması planlanmaktadır. Ayrıca önerilen yöntemlerin gerçek zamanlı sistemlerde test edilerek uygulanabilirlikleri noktasında geri bildirim alınması da gelecek çalışmalara yön verecektir.

8. KAYNAKLAR

Al-Taharwa, I., Sheta, A. and Al-Weshah, M., "A mobile robot path planning using genetic algorithm in static environment", *Journal of Computer Science*, vol:4, 4, pp. 341-344, (2008).

Bean J. C., "Genetic Algorithms and Random Keys for Sequencing and Optimization", *ORSA Journal on Computing*,vol:6, 2, pp. 154-160, (1994).

Bellman, R.. "On a routing problem", *Quarterly Applied Mathematics*, 16, pp. 87– 90, (1958).

Burke, E. K. and Kendall, G., *Search Methodologies*, USA:Springer, (2005).

Chen, X. and Li, Y. M., "Smooth path planning of a mobile robot using stochastic particle swarm optimization", *In Proceedings of IEEE International Conference on Mechatronics and Automation*, pp. 1722–1727, (2006).

Çamoğlu, D., *Bilgisayar kontrollü Robotik*, Ankara: Dikey Eksen, (2011).

Dijkstra E. W., "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol:1, 1, pp. 269-271, (1959).

Elfes, A., "Occupancy grids: Stochastic spatial representation for active robot perception", *Proceedings of 6th conference on Uncertainty in AI*, pp. 66-70, (1990).

Ersson, T. and Hu, X., "Path planning and navigation of mobile robots in unknown environments," *IEEE/RSJ International Conference*, 2, pp.858-864, (2001).

Fogel**,** D. B.," An Evolutionary Approach to the Traveling Salesman Problem" *Journal Biological Cybernetics* ,vol :60, no:2, pp. 139–144, (1988).

Fogel, D. B., “A Paralel Processing Approach to a Multiple Traveling Salesman Problem Using Evolutionary Programming”, *Fourth Annual Paralel Processing Symposium*, Fullterton, CA., pp. 318–326, (1990).

Gass, S. ve Saaty, T., “The computational algorithm for the parametric objective function”,*Naval Research Logistics Quart*, vol:2, pp. 39-45,(1955).

Geetha, S., Chitra, G. M. and Jayalakshmi, V., "Multi objective mobile robot path planning based on hybrid algorithm", *Electronics Computer Technology (ICECT), 2011 3rd International Conference*, Kanyakumari, pp. 251-255, (2011).

Gemeinder, M. and Gerke, M. , “GA-based path planning for mobile robot systems employing an active search algorithm”, *Applied Soft Computing*, vol:3, 2, pp. 149–158, (2003).

Glover, F., “Tabu Search: Part I”, *ORSA Journal on Computing*, vol:1, 13, pp. 196-206, (1962).

Graf, B., Wandosell, J. M. H. and Schaeffer, C., “Flexible path planning for nonholonomic mobile robots”, *In Proceeding of 4th European workshop on advanced Mobile Robots*, Fraunhofer Inst. Manufact. Eng. Automat., Lund, Sweden, pp. 199-206, (2001).

Hart, P. E., Nilsson, N. J. and Raphael, B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, *IEEE Transactions on Systems Science and Cybernetics*, vol:4, 2, pp. 100–107, (1968).

Hart, P. E., Nilsson, N. J. and Raphael, B. , “Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"", *SIGART Newsletter*, pp. 28–29, (1972).

Holland, J., *Adaptation in Natural and Artificial Systems,* Ann Arbor: University of Michigan Press, (1975).

Hu, X. Z. And Xu, Q. G. “Robot path planning based on artificial immune network”,*In Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics*, pp. 1053-1058, (2007).

Huang, Y., "Intelligent Technique for Robot Path planning Using Artificial Neural Network and Adaptive Ant Colony Optimization", *JCIT*, vol:7, 9, pp. 246 - 252, (2012).

Kara, İ., *Doğrusal Programlama*, Eskişehir: Bilim Teknik Yayınevi, (1991).

Larranaga, P., Kuıjpers, C.M.H., Murga, R.H. , Inza, I. and Dızdarevıc, S.,” Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators”, *Artificial Intelligence Review* ,13, pp. 129–170, (1999).

Latombe, J. C., *Robot motion planning*, Norwell MA: Kluwer, (1991).

Martin, C. M. and Moravec, H. P., "Robot evidence grids", *CMU Robotics Institute Technical Report CMU-RITR-96-06* , (1996).

Mo, H. W. and Li Z. Z., “Biogeography based differential evolution for robot path planning”, *In Proceedings of International Conference on Information and Automation*, pp. 1- 6, (2012).

Mohanta, J. C., Parhi, D. R., and Patel, S. K, “Path planning strategy for autonomous mobile robot navigation using Petri-GA optimization”, *Computers&Electrical Engineering*, vol:37, 6, pp. 1058-1070, (2011).

Moravec, H. P., “Sensor Fusion in Certainty Grids for Mobile Robots”, *Al Magazine*, vol:9, no:2, pp. 61-74, (1988).

Okutkan, O., ” Yapay Zeka ile Mobil Robot Kontrolü”, Yüksek Lisans, *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, 1-11, (2006).

Qiang, W., Jin, Y. and Jinge, W., ”A path planning approach to moving robot based on genetic algorithms”, *Journal of Harbin Institute of Technology*, vol:36, 7, 867-870, (2004).

Shahidi, N., Esmaeilzadeh, H., Abdollahi, M. and Lucas, C., “Memetic algorithm based path planning for a mobile robot”, *Engineering and Technology*, 1, pp. 34-37, (2005).

Talbi, E.G., *Metaheuristics From Design to Implementation*, New Jersey: Wiley, (2009).

Xiao, J., Michalewicz, Z., Zhang L. and Trojanowski, K., ”Adaptive Evolutionary Planner/Navigation for Mobile Robots,” *IEEE Trans. On Evolutionary Compution*, vol:1, 1, pp. 18- 28, (1997).

Zhi, W., Luo, Q. S. and Su, X. D., "A Novel Mobile Robot Path Planning Method based on ACO Algorithm using New Ants Meeting Judgment Strategy", *JDCTA*, vol:6, 1, pp. 302 - 309, (2012).

Zhou, W., Yi, Z. And Ruimin, Y., ”Mobile Robot Path Planning Based on Genetic Algorithm”, *Microcomputer Information*, vol:24, 26, pp.187-189, (2008).

Zuo, X. B., Cai, Z. X. and Sun, G. R., “Non-smooth environment modeling and globalpath planning for mobile robots”, *Journal of Central South China of Technology*, vol:10, 3, pp. 248-255, (2003).

9. ÖZGEÇMİŞ

Adı Soyadı	: Eşref BOĞAR
Doğum Yeri ve Tarihi	: ESKİŞEHİR / 29.06.1989
Lisans Üniversite	: Anadolu Üniversitesi
Elektronik posta	: esrefbogar@gmail.com
İletişim Adresi	: Siteler Mah. Öğretmenler Sitesi, 6212 Sokak, A2 Blok, Daire: 13 Pamukkale / DENİZLİ