T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİR YAPAY ZEKA METODU İLE AKILLI BİNA TAHLİYE SİSTEMİ TASARIMI

Enes Furkan SANCAK

YÜKSEK LİSANS TEZİ

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Danışman

Doç. Dr. Cüneyt YILMAZ

Şubat, 2021

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİR YAPAY ZEKA METODU İLE AKILLI BİNA TAHLİYE SİSTEMİ TASARIMI

Enes Furkan SANCAK tarafından hazırlanan tez çalışması 24.02.2021 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalı, Mekatronik Mühendisliği Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Cüneyt YILMAZ
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Doç. Dr. Cüneyt YILMAZ, Danışman
Yıldız Teknik Üniversitesi _____

Prof. Dr. Haydar LİVATYALI, Üye
Yıldız Teknik Üniversitesi _____

Doç. Dr. Zeki Yağız BAYRAKTAROĞLU, Üye
İstanbul Teknik Üniversitesi _____

Danışmanım Doç. Dr. Cüneyt YILMAZ sorumluluğunda tarafımca hazırlanan Bir Yapay Zeka Metodu ile Akıllı Bina Tahliye Sistemi Tasarımı başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Enes Furkan SANCAK

İmza

Aileme

ve

bu süreçte bana destek olan herkese

TEŞEKKÜR

Bu çalışma boyunca bana destek olan, değerli görüş ve önerileriyle bana yol gösteren, tez danışmanım Doç. Dr. Cüneyt YILMAZ'a teşekkürlerimi sunarım. Bu süreçte bana çok yardımcı olan manevi desteklerinden dolayı da annem, babam ve kardeşlerime teşekkürlerimi sunarım.

Enes Furkan SANCAK

İÇİNDEKİLER

SİMGE LİSTESİ

dA	A düğüm noktasının değeri
w	Akışın genişliği
L	Akışın uzunluğu
D	Alan bazında insan yoğunluğu (m2 kare başına bir insanın kapladığı alan)
N	Akıştaki insan sayısı
$x_{başlangıç}$	Başlangıç noktasının x koordinatı
$y_{başlangıç}$	Başlangıç noktasının y koordinatı
x_{hedef}	Hedef noktasının x koordinatı
y_{hedef}	Hedef noktasının y koordinatı
f	İnsanın yatay kesit alanı
ρ	İnsan yoğunluğu (m^2 başına düşen insan sayısı)
S	Kalabalıktaki bir insanın ilerleme hızı (Nelson) (m)
V	Kalabalıktaki bir insanın ilerleme hızı (Predtechenskii ve Milinskii) (m)
y_k	k. adımdaki yol
y_{k+1}	k+1. adımdaki yol
g	Oluşturulan mevcut rotayı kullanarak başlangıç noktasından şu anki düğüme gelmek için gereken toplam masraf
dS	S düğüm noktasının değeri
h	Sonraki düğümden hedefe ulaşmak için gereken tahmini masraf

KISALTMA LİSTESİ

ABTS	Akıllı Bina Tahliye Sistemi
GA	Genetik algoritma
IoT	Internet of Things
MLP	Multilayer perception
RFID	Radyo Frekans Identification (**r**adyo frekans tanımlama)
SPFA	Shortest Path Faster Algoritması
SEÖ	Stokastik evrensel örnekleme
UOS	Uygunlukla orantılı seçim

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Bir Yapay Zeka Metodu ile Akıllı Bina Tahliye Sistemi Tasarımı

Enes Furkan SANCAK

Mekatronik Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Doç. Dr. Cüneyt YILMAZ

Son yıllarda hızla gelişen teknolojinin etkisiyle birçok alanda olduğu gibi akıllı bina sistemleri alanında da büyük gelişmeler olmaktadır. Bu sistemlerden biri olan bina tahliye sistemleri, yangın ve deprem gibi acil durumlarda insanların en kısa zamanda güvenli bir şekilde binadan çıkarılmasına yönelik sistemler oldukları için büyük öneme sahiptir. Bu nedenle literatürde bu konu üzerine geçmişte yapılmış birçok çalışma mevcuttur. Ancak bu çözümlerde bulunan eksikliklerin giderilmesi, yeni teknolojilerin entegre edilerek bu çözümlerin daha iyi hale getirilmesi ve yenilikçi çözümlerin geliştirilmesi için bu alanda yeni çalışmalara ihtiyaç bulunmaktadır. Bu hedefe yönelik olarak, bu çalışmada, bir yapa zeka yöntemi ve güncel teknolojik çözümler kullanılarak bir akıllı bina tahliye sistemi geliştirilmiştir. Tasarlanan bu sistem donanımsal olarak, binanın belirli konumlarına yerleştirilmiş yangın ve kalabalık sensörleri, dinamik çıkış tabelaları ve bunların bağlı olduğu bir merkezi kontrol bilgisayarından oluşmaktadır. Dinamik çıkış tabelaları, merkezi kontrol sisteminden gelen sinyallere göre gösterdiği yönü dinamik olarak değiştirilebilen özel tabelalardır. Bir acil durum sırasında güvenli olmayan bölgeler sürekli olarak sensörler tarafından tespit edilerek merkezi kontrol yazılımına

iletilmektedir. Kontrol yazılımı da yapay zeka yöntemini kullanarak en kısa güvenli çıkış yollarını hızlıca hesaplayarak dinamik çıkış tabelalarını, en kısa ve güvenli olan çıkış rotasını gösterecek şekilde sürekli olarak güncellemektedir. Sonuç olarak bu çalışmada, bir yapay zeka yöntemi olan Genetik Algoritma ve güncel teknolojiler kullanılarak bir akıllı bina tahliye sistemi tasarlanmış, yapay zeka yönteminde kullanılan yöntem ve parametrelerin performansa etkileri incelenmiş, sanal ortamda bir benzetim yazılımı hazırlanarak sistem test edilmiş ve sistemin avantajları ve dezavantajları ortaya koyulmuştur.

Anahtar Kelimeler: Akıllı bina, bina otomasyonu, bina tahliye sistemi, genetik algoritma, yapay zeka

ABSTRACT

Designing a Smart Building Evacuation System with an Artificial Intelligence Method

Enes Furkan SANCAK

Department of Mechatronics Engineering

Master of Science Thesis

Advisor: Assoc. Prof. Dr. Cüneyt YILMAZ

In recent years, with the effect of current technological developments, major developments have taken place in the field of smart building systems. Building evacuation systems have an important place among smart building systems, because they are aiming safely evacuating people from buildings in an emergency case such as fire and earthquake etc. Although there are many studies on this subject in the literature, it is very important to carry out new studies in this field in order to improve the existing solutions by improving their deficiencies or to develop novel solutions. For this purpose, in this study, a smart building evacuation system has been developed using genetic algorithms which is an artificial intelligence method and current technological solutions. The hardware part of the designed system is composed of fire sensors, crowd sensors and dynamic exit signs which are placed in specific locations in the building and all of them are connected to a central computer. Dynamic exit signs are special signs which can change the directions they show dynamically according to the signals coming from the central control system. During an emergency, unsafe areas are continuously detected by the sensors and the data are forwarded to the central control software and the control software continuously

calculates the shortest safe exit routes using an artificial intelligence method to update the dynamic exit signs to direct the people through the shortest safety exit route. As a result, in this study, a smart building evacuation system was designed using current technologies and genetic algorithm, which is an artificial intelligence method, then the effects of the methods and parameters used in the artificial intelligence method on the performance were examined and a simulation software was developed to test the system, and finally, the advantages and disadvantages of the system were revealed.

Keywords: Smart building, building automation, building evacuation system, genetic algorithm, artificial intelligence

1
GİRİŞ

Deprem, yangın, terör saldırısı gibi acil durumlarda bir binadaki insanların en kısa sürede tahliye edilmesi, can kayıpları ve yaralanmaların önlenmesinde büyük bir öneme sahiptir. Böyle durumlarda, özellikle büyük binalarda, panik yaşayan insanların herhangi bir yardım almadan en güvenli ve en kısa rotayı bulabilmeleri çok zor olmakta, çok sayıda kişinin aynı yoldan gitmeye çalışması sonucu izdihamlar olabilmekte, binayı iyi tanımayanlar yolunu kaybederek tehlikeli bölgelere gidebilmektedir.

Günümüzde binaların çoğunluğunda, insanlara yardımcı olması için bina tahliye planları ve çıkış tabelaları kullanılmaktadır. Literatürdeki bir çalışmada Kobes ve ark. yangın sırasında insanların tahliye yollarını nasıl belirlediğini incelemiş ve duman olmadığında, insanların sadece %56.3'ünün çıkış tabelalarına bakarak tahliye yollarını belirlediği görülürken, duman artıp görünürlük seviyesi çok düşük olduğunda, çok daha fazla insanın (% 81.8'inin) çıkış tabelaları yardımıyla tahliye yollarını belirlediği tespit edilmiştir [1]. Bu çalışmanın sonuçları bir yangın durumunda bina tahliyesi için çıkış tabelalarının rolünün çok önemli olduğunu göstermektedir. Ancak hala birçok binada yaygın olarak kullanılmakta olan klasik sabit çıkış tabelaları, yangın gibi dinamik durumlarda, kalabalık ve büyük binalar için yetersiz kalmaktadır. Çünkü bir acil durum sırasında en kısa rota her zaman en iyi rota olmamaktadır. En kısa yol bazen tehlikeli veya çok kalabalık olabilmekte, bundan dolayı izdiham, can kayıpları ve yaralanmalar yaşanabilmektedir. Bu sorunların, binadaki tehlike bölgeleri göz önünde bulundurarak optimum rotayı hızlı bir şekilde hesaplayarak insanları aşırı kalabalığı önleyecek şekilde yönlendirebilen dinamik bir akıllı bina tahliye sistemi ile çözülmesi mümkündür.

1.1 Literatür Özeti

1.1.1 Akıllı Bina Tahliye Sistemleri

Chu ve Wu, RFID özellikli cep telefonları ve bulut bilişim teknolojisini kullanarak bir bina tahliye sistemi geliştirmişlerdir [2]. Sistemde yangın algılanan bölgeler rota hesaplamasında göz önünde bulundurulmakta ve RFID üzerinden insanların konumları izlenerek aşırı kalabalığı önleyecek şekilde insanlar alternatif rotalara yönlendirilmektedir. Her insan için bulut bilişim üzerinden konumuna göre bir rota hesaplanarak kişinin telefonuna iletilmekte ve telefondaki uygulama üzerinde çıkış rotası görüntülenmektedir.

Atila ve ark., özellikle yüksek binalardaki yangınlar için akıllı bireysel bina tahliye sistemi geliştirmişlerdir [3]. Binanın çeşitli bölgelerine yerleştirilen sensörler ile tehlikeli noktalar algılanmakta, ayrıca kullanıcıların yaş, cinsiyet, ekipman, engellilik, hastalık gibi çeşitli durumları da geliştirilen çok katmanlı algılama (multilayer perception – MLP) yöntemi ile değerlendirilerek bölgeler risk oranlarına göre sınıflandırılmakta, böylece her kullanıcıya özel rota oluşturulmaktadır. Hesaplanan rotalar cep telefonları aracılığıyla kullanıcılara iletilmektedir. Bu çalışmada tasarlanan sistem zamanla daha da geliştirilerek beş yıl sonra SmartEscape adında bir sistem ortaya çıkarılmıştır [4] .

Gökçeli ve ark. nesnelerin interneti (IoT) bazlı bir bina tahliye sistemi geliştirmişlerdir [5]. Binanın belirli bölgelerine yangın, gaz algılama ve insanların bina içi konumlarının tespiti için çeşitli sensörler yerleştirilmesini gerektiren bu sistemde; ayrıca konum tespiti için insanlar üzerinde RFID (radyo frekans tanımlama) etiketlerin bulunması gerekmektedir. Tüm cihazlar internet üzerinden merkezi bilgisayar ile haberleşmekte ve sonuçta hesaplanan tahliye rotasına insanlar sesli olarak yönlendirilmektedir. Çalışmada rota hesaplanmasında kullanılan algoritma ve performansı hakkında bilgi verilmemiştir.

Chou ve ark. yangın durumunda itfaiye kurtarma ekiplerini, binada mahsur kalmış ve kurtarılmayı bekleyen kişilere optimum şekilde ulaştırmayı hedefleyen akıllı yangın kurtarma sistemi geliştirmişlerdir [6]. Binadaki insanların ve itfaiye ekiplerinin bina içindeki konumlarının belirlenmesi için akıllı telefonlarının bluetooth bağlantısı özelliği kullanılmıştır. Bluetooth bağlantısı ile konum tespit sistemi, bina içinde belirli konumlara yerleştirilen bluetooth sensörler ve kullanıcıların akıllı telefonlarına yüklenen yazılımdan oluşmaktadır. Bu çalışmada en kısa rotaların hesaplanması için Dijkstra algoritması kullanılmıştır. Yukarıda bahsedilen diğer çalışmalardan farklı olarak, bu çalışmada kurtarılacak insanları değil kurtarma ekiplerini yönlendirmek hedeflenmiştir. Binada mahsur kalan, kendi başına çıkma imkanı olmayan insanların kurtarma ekipleri tarafından en hızlı şekilde bulunup kurtarılmasında ekiplere büyük destek sağlayacak bir sistem geliştirilmiştir.

1.1.2 En Kısa Yol Problemi

Bu çalışmada tasarlanan sistem için çözülmesi gereken en önemli problemlerden biri çıkış rotalarının en hızlı şekilde hesaplanmasıdır. En kısa yol problemi olarak da bilinen rota hesaplama problemi; otonom robotlar ve araçlar, navigasyon uygulamaları gibi birçok alanda karşılaşılan bir problem olduğu için bu konuda geçmişte birçok çalışma yapılmış, çok çeşitli algoritmalar geliştirilmiştir.

Dijkstra algoritması en kısa yol problemi için bilinen en popüler algoritmalardan biridir [7]. Birçok çalışmada doğrudan veya modifiye edilerek kullanılmış ve kullanılmakta olan bu algoritma, ağ üzerindeki bir başlangıç noktasından diğer tüm noktalara olan en kısa yolları bulmaktadır.

Samah ve ark. tarafından yapılan bir çalışmada Dijkstra algoritmasının lineer ve dinamik programlamaya göre çok daha kısa sürede çözüm sağladığı belirtilmiş ve Dijkstra algoritması modifiye edilerek geliştirilen tahliye sisteminde kullanılmıştır [8]. Bu çalışma Dijkstra algoritmasının bir bina tahliye sisteminde rota hesaplanması için kullanılabileceğini göstermektedir. Ancak sadece binanın bir katı üzerinde test edildiği için çok büyük ve karmaşık binalar için yeterli olup olmayacağı belirlenmemiştir.

Bellman-Ford algoritması, Dijkstra algoritması gibi başlangıç noktasından diğer tüm noktalara olan en kısa yolları bulmaya yarayan bir algoritmadır [9], [10]. Dijkstra algoritmasına göre çok daha yavaş olsa da Dijksra algoritmasının çözemediği negatif kenar ağırlıkları olan çizgeleri(graf) de çözebildiği için tercih edilebilmektedir.

A* algoritması, Nilsson tarafından ortaya çıkarılmış sezgisel yöntem kullanan bir en kısa yol bulma algoritmasıdır [11]. Sezgisel bir yöntem kullanması sayesinde iki nokta arasındaki en kısa rotayı Dijkstra algoritmasına göre çok daha hızlı bulabilmektedir, ancak Dijksra gibi her zaman kesinlikle en kısa yolu bulma garantisi verememektedir. A* algoritmasında kullanılan sezgisel yöntemin seçimi çok önemlidir çünkü probleme uygun bir sezgisel yöntem seçilmediğinde en kısa yol yerine daha uzun yolları bulma ihtimali bulunmaktadır.

Floyd-Warshall algoritması olarak bilinen bir diğer en kısa yol bulma algoritması, bir çizgedeki tüm nokta çiftleri arasındaki en kısa yolları tek seferde hesaplayabilmesi ile diğer algoritmalardan ayrılmaktadır [12], [13].

Johnson Algoritması, Floyd Warshall algoritması gibi bir çizgedeki tüm nokta çiftleri arasındaki en kısa yolları tek seferde hesaplamaktadır[14]. Negatif kenar ağırlıkları olan çizgedeki de çözebilmektedir. Bu algoritma, negatif kenarları elimine etmek için ilk önce Bellman-Ford algoritmasını kullanmaktadır. Daha sonra ise Dijkstra algoritmasını tekrar tekrar kullanarak her nokta çifti arasındaki mesafeleri hesaplar. Bölüm 2.1 de bahsedilen bu algoritmalar detaylıca açıklanacaktır.

1.1.3 En Kısa Yol Probleminin Çözümünde Yapay Zeka Kullanımı

Son zamanda gittikçe popülerleşmekte olan yapay zeka yöntemleri, klasik yöntemler kullanarak çözülmesi zor olan çeşitli problemleri hızlı bir şekilde çözebilen, güçlü araçlardır. Bu nedenle, birçok araştırmacı en kısa yol problemlerini çözmek için çeşitli yapay zeka yöntemlerini kullanmayı denemiştir. Örneğin, Ali ve ark. yaptıkları bir çalışmada yapay sinir ağlarını en kısa yol problemini çözmek için kullanmıştır [15].

Gen ve ark. 1997 yılında yayınladıkları bir çalışmada Genetik Algoritma kullanarak en kısa yol probleminin çözülebilmesinin mümkün olduğunu göstermişlerdir [16].

Gerçekleştirilen çeşitli benzetimler ile, geliştirilen Genetik Algoritmanın optimum çözümü hızlı bir şekilde ve yüksek olasılıkla bulabildiği ortaya koyulmuştur. Çalışma sonucunda geliştirilen Genetik Algoritma klasik algoritmalarla rekabet edecek kadar hızlı olmasa da, bu çalışmanın ileride daha verimli çözümler oluşturulması için bir temel görevi göreceği öngörülerek, en kısa yol problemine yeni bir yaklaşım ortaya koyulmuş olması, bu çalışmanın önemini göstermektedir.

Munetomo ve ark. 1998 yılında gerçekleştirdikleri bir çalışmada geliştirdikleri genetik adaptif rota algoritması için göç şeması (migration scheme) geliştirmişlerdir [17]. Göç şeması, alt popülasyonlar arasında kromozomların aktarımı şeklinde gerçekleşmektedir. Bu sayede erken yakınsamanın önüne geçilmesi hedeflenmiştir.

Inagaki ve ark. 1999 yılında, Genetik Algoritmanın sadece tek bir sonuç değil birden fazla sonuç üretilebilmesi avantajını kullanarak çoklu rota hesaplama algoritması geliştirmişlerdir [18]. Bu çalışma sayesinde en kısa rota dışında alternatif rotaların da bulunarak gerektiğinde kullanılabileceğini, Genetik Algoritmanın Dijkstra algoritması gibi tek çözüm üreten algoritmalara göre böyle bir avantajı olduğunu göstermişlerdir.

Ahn ve ark. 2002 yılında en kısa yol problemini çözümü için bir Genetik Algoritma geliştirmiş, ayrıca ihtiyaç duyulan kalitede çözüm elde edilebilmesi için gerekli olan optimum popülasyon büyüklüğünü belirlemeye yarayan bir denklem ortaya koymuşlardır [19]. Çalışma sonucunda yer alan benzetim sonuçlarına göre, geliştirdikleri Genetik Algoritmanın en kısa yol bulma konusunda Dijkstra algoritmasından daha hızlı olduğunu belirtmişlerdir. Ayrıca daha önce Munemoto ve Inagaki tarafından en kısa yol problemi için geliştirilen Genetik Algoritma çözümleri ile de karşılaştırmalar yaparak, geliştirilen algoritmanın bunlara göre çok daha hızlı bir şekilde sonuç verdiği ve çok daha kaliteli çözümler bulabildiğini göstermişlerdir.

Mohemmed ve ark. gerçekleştirmiş oldukları bir çalışmada en kısa yolu bulmak için parçacık sürüsü optimizasyonu yöntemini kullanmış ve başarılı sonuçlar almıştır [20]. Ghoseiri ve ark. ise karınca kolonisi optimizasyon algoritması ile iki amaçlı en kısa yol probleminin çözümü üzerinde çalışmışlardır. Ebrahimnejad ve ark. en kısa

yol problemini çözmek için yeni bir yapay arı kolonisi algoritması geliştirmişlerdir [21].

Atila ve ark. yüksek binalardaki yangınlar için yapay sinir ağları metodunu kullanarak akıllı bir bireysel tahliye sistemi tasarlamışlardır [3]. Bu sistem, çok katmanlı algılama (multilayer perception - MLP) yöntemini kullanarak, bina içindeki bölgelerin kullanıcıya özgü risk seviyelerini belirlemekte ve bu sonuçlara göre kullanıcıların cep telefonlarına bireysel çıkış rotaları ileterek güvenli ve en hızlı şekilde çıkışa ulaşmalarını sağlamaktadır. Risk seviyelerinin hesaplanmasında o bölgede ölçülen karbon monoksit miktarı, sıcaklık, görülebilirlik, yol uzunluğu, kalabalık miktarı gibi değerler dışında kullanıcıya özgü yaş, cinsiyet, vücut tipi, engellilik, hastalık, koruyucu ekipmana sahip olma durumu gibi bilgiler de değerlendirilmektedir. Kullanıcıya özgü rota oluşturma özelliği ile öne çıkan bu sistemin çözüm bulma hızı hakkında bilgi verilmemiştir.

Sharma ve ark. yaptıkları bir çalışmada statik ve dinamik ortamlarda Genetik Algoritma ile Dijkstra'nın en kısa yol algoritmasını karşılaştırmış ve Genetik Algoritmanın her iki ortamda da Dijkstra algoritmasından çok daha hızlı olduğunu ortaya koymuşlardır[18]. Saeed Behzadi ve Ali A. Alesheikh, en kısa yol problemini çözmek için bir Genetik Algoritma geliştirilmesi üzerine çalışmışlar ve gerçek yol haritaları üzerinde çeşitli benzetimler gerçekleştirmişlerdir[19]. Mohamed Amine Yakoubi ve Mohamed Tayeb Laskri, çalışmalarında temizlik robotunun rota planlaması için Genetik Algoritma kullanmışlardır [20]. Cedric Davies ve Pawan Lingras dinamik ve stokastik ağlardaki en kısa yolları yeniden yönlendirmek için Genetik Algoritmadan yararlanmışlardır [21]. Ayoub Bagheri, Muhammed-R. Akbarzadeh ve Mohammad-H Saraee tarafından gerçekleştirilen çalışmada, en kısa yol problemi için Genetik Algoritmanın Dijkstra algoritmasına göre çok daha hızlı sonuç verebildiğini gösteren sonuçlar elde edilmiştir. [22]. Ayrıca yapılan testler sonucunda; geliştirilen Genetik Algoritmanın daha önce Ahn [19], Munetomo [17] ve Inagaki [18] tarafından geliştirilen Genetik Algoritma çözümlerine göre çok daha düşük bir hata oranına sahip olduğu gösterilmiştir.

1.2 Tezin Amacı

Bu çalışmada, bir yapay zeka metodu olan "Genetik Algoritma" kullanarak çıkış rotalarını dinamik bir şekilde hesaplayabilen bir akıllı bina tahliye sisteminin tasarlanması amaçlanmıştır. Tasarlanan bu sistemin test edilebilmesi için bir benzetim yazılımının oluşturulması, bu yazılım üzerinde çeşitli benzetimler yapılarak geliştirilen yapay zeka destekli sistemin performansının ve bu performansa etki eden faktörlerin incelenmesi de çalışmanın amaçları arasındadır. Ayrıca, bu çalışma sonucunda Genetik Algoritmanın bu tür bir sistem için nasıl tasarlanması gerektiği, hangi yöntemlerin ve parametre değerlerinin en iyi performansı sağlayacağı, hangi büyüklükteki bir bina için artık Genetik Algoritmanın Dijkstra algoritmasına göre daha avantajlı hale geldiğinin tespit edilmesi amaçlanmıştır.

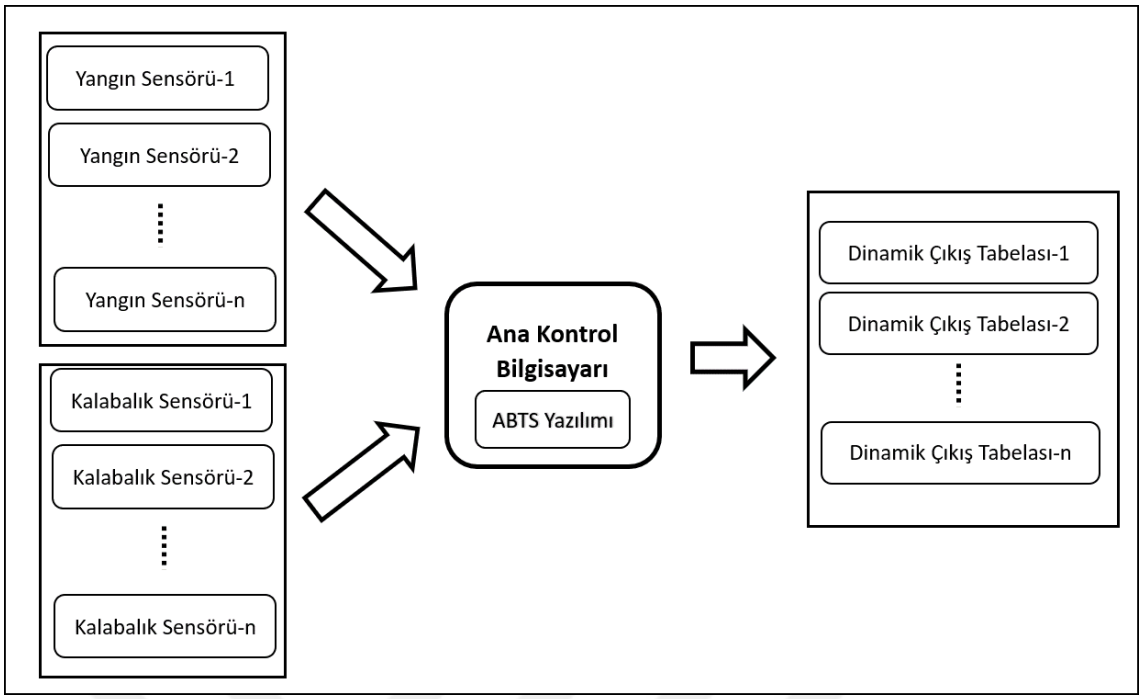
1.3 Orijinal Katkı

Günümüzde oldukça yüksek ve karmaşık yapıya sahip binalar giderek artmaktadır. Ayrıca acil durumlarda bina içindeki kalabalık oranı, yangın bölgeleri, zehirli gazların olduğu alanlar, tehlikeli bölgeler gibi çeşitli verilerin elde edilmesini sağlayan teknolojilerin de yaygınlaşması ile çok gelişmiş akıllı bina tahliye sistemlerinin tasarlanmasını mümkün kılmaktadır. Ancak bu durumda çok sayıda sensör içeren bir sistem oluşacağı için, Dijkstra algoritması gibi klasik algoritmaların çözüm süresinin yavaş kalacağı, bu durumda da sistemin uyartılara vaktinde tepki veremeyerek tahliyenin yavaşlamasına ve hatalara neden olabileceği öngörülmüştür. Akıllı bina tahliye sistemleri üzerine literatürde yer alan çalışmalar incelendiğinde ise, en kısa yol probleminin çözümü için genellikle Dijkstra algoritması gibi klasik algoritmaların tercih edildiği görülmüştür. Dijkstra algoritması dışındaki diğer çeşitli algoritmalar ve yapay zeka yöntemlerinin de akıllı bina tahliye sistemlerinde kullanımı üzerine çeşitli çalışmalar mevcut olsa da yapılan literatür taramalarında Genetik Algoritma ile akıllı bina tahliye sistemi geliştirilmesi konusunda bir çalışmaya rastlanmamıştır. En kısa yol problemi üzerine yapılan çalışmalar, özellikle büyük ve karmaşık çizge ağlarında Genetik Algoritmanın Dijkstra algoritmasına göre çok daha hızlı çözüm bulabildiğini göstermektedir[22]. Bu bilgiler ışığında, çok sayıda sensöre sahip büyük ve

karmaşık yapılı bir bina için tasarlanan akıllı tahliye sisteminde, yeterli tepki süresinin elde edilebilmesi amacıyla bir yapay zeka yöntemi olan Genetik Algoritmanın kullanılabileceği öngörülmüştür.

2
YÖNTEM

Geliştirilen Akıllı Bina Tahliye Sistemi (ABTS), binanın belirli bölgelerine yerleştirilmiş dinamik çıkış tabelaları, yangın ve kalabalık seviyesini ölçen sensörler ve ana kontrol yazılımının çalıştığı bir bilgisayardan oluşmaktadır. Yangın algılama işlemi genellikle duman ve sıcaklık sensörleri ile yapılmaktadır. Ancak günümüzde güvenlik kameraları ile görüntü işleme yöntemi kullanılarak yangının erken algılanması için de gelişmiş çözümler bulunmaktadır. İlk yangının başladığı tespit edildiğinde sistem aktifleşmekte, ana kontrol yazılımı sensör ağını kullanarak sürekli olarak binadaki tehlikeli bölgeleri tespit etmekte ve bu bölgeleri tehlikeli olarak işaretleyerek rota hesaplanırken kullanılmasını önlemektedir. Kalabalık sensörleri, bir ortamda bulunan kişi sayısını tespit etmekte ve ana kontrol yazılımına bu bilgiyi göndermektedir. Ana kontrol yazılımı da o noktadan çıkışa giden en kısa güvenli rotayı Genetik Algoritma yardımıyla hesaplayarak dinamik çıkış tabelalarını hesaplanan rotayı gösterecek şekilde güncellemektedir. Binada kalabalık sensörleri tarafından insan varlığı tespit edilen her noktada aynı işlemler tekrarlanmaktadır. Yeni bir yangın bölgesi algılandığında o bölge "tehlikeli" olarak işaretlenmekte ve önceden belirlenen tüm rotaların durumu kontrol edilmektedir. Bir rota eğer bu "tehlikeli" bölgeden geçiyor ise o rota tekrar hesaplanmakta ve ilgili tabelalar güncellenmektedir. Kalabalık sensörleri bir bölgede aşırı kalabalık oluşmaya başladığını tespit ettiğinde o bölgeden geçen tüm rotalar yoğunluğu daha az bir tarafa yönlendirilerek izdiham oluşması önlenmektedir. Bu sayede akıllı bina tahliye sistemi, tabelaları sürekli olarak güncelleyerek, hızlı ve güvenli bir şekilde binayı tahliye edebilmeleri için insanları yönlendirebilmektedir. Sistemin yapısı, Şekil 2.1' de bir blok diyagramı ile gösterilmiştir.



Şekil 2.1 Akıllı Bina Tahliye Sisteminin genel yapısı

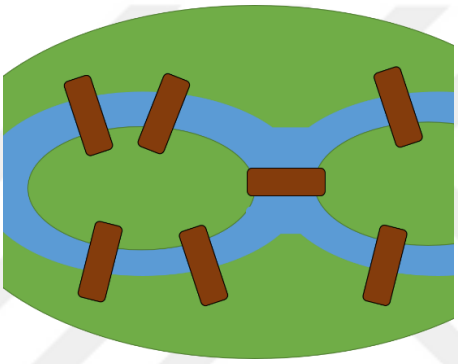
Tasarlanan bu sistemde, rotaların hesaplanması için bir algoritmaya ihtiyaç bulunmaktadır. Yangın gibi dinamik durumlarda bina içindeki tehlikeli ve kalabalık bölgeler sürekli ve hızlıca değişebileceği için algoritmanın yeterince hızlı olması gerekmektedir. Bu amaçla önce mevcut klasik en kısa yol algoritmaları incelenecek, daha sonra ise bu çalışmanın asıl amacı olan bir yapay zeka yöntemi ile en kısa yol probleminin çözümü konusu incelenerek geliştirilen çözüm detaylıca açıklanacaktır.

2.1 En Kısa Yol Algoritmaları

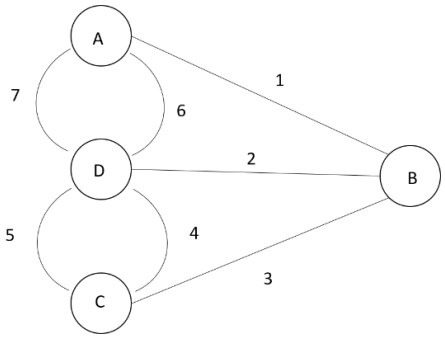
En kısa yol algoritmaları genelde çizge teorisine göre oluşturulan ağlar (network) üzerinde uygulanır. Algoritmalar incelenmeden önce çizge teorisi ve zaman karmaşıklığı gibi kavramların açıklanması, algoritmaların daha iyi anlaşılması konusunda faydalı olacaktır.

2.1.1 Çizge (Graf) Teorisi

Çizge teorisi, 1736 yılında "Königsberg'in yedi köprüsü" adıyla bilinen matematik probleminin Euler tarafından çözülmesi sırasında ortaya çıkmıştır. Euler, Şekil 2.2'de gösterilen "Königsberg kentinde bulunan yedi köprüden bir ve yalnız bir defa geçilerek tekrar başlanılan konuma dönülebilir mi?" sorusu ile ifade edilen bu problemin cevabının imkansız olduğunu matematiksel olarak ispatlamıştır. Problemi çözerken, adaları birer nokta, köprüleri ise onları birleştiren çizgiler olarak ifade ederek çizge teorisinin temellerini atmıştır. Şekil 2.3'te bahsedilen problemin çizge teorisine göre ifade edilmiş hali gösterilmiştir.



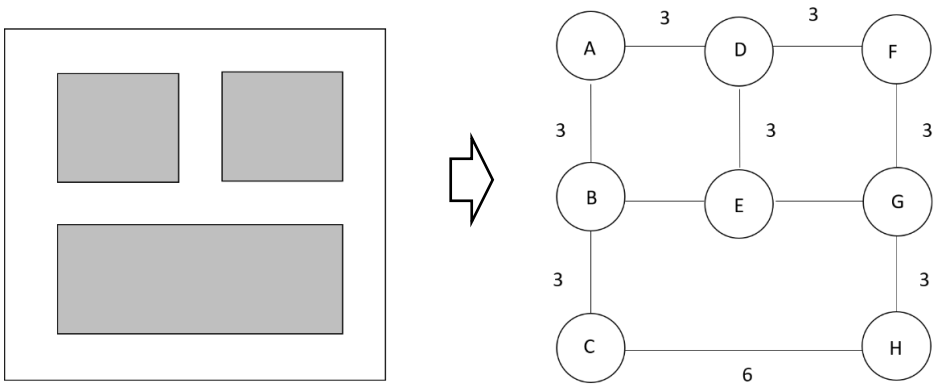
Şekil 2.2 Könisberg'in yedi köprüsü



Şekil 2.3 Könisberg'in yedi köprüsü probleminin çizge teorisine göre ifadesi

Çizge, düğümler (node) ve bu düğümleri birbirine bağlayan kenarlar (edge) ile oluşturulan ağ yapısına denir. Şekil 2.4'te solda bir kat planı sağda ise bu kat planına göre oluşturulan örnek bir çizge ağı gösterilmiştir. Çember şeklinde gösterilen

düğümler, çizgi şeklindeki kenarlar ile birbirine bağlanmıştır. Kenarların üzerinde ise maliyet (cost) değerleri yazılmıştır.



Şekil 2.4 Örnek bir kat planı için çizge ağı oluşturulması

Bir binanın kat planını çizge ağı şeklinde ifade etmek istersek ilk önce düğüm noktaları belirlenmelidir. Daha sonra belirlenen bu noktaları birbirine bağlayan koridorların uzunluğu ölçülerek kenarların maliyet değerleri tespit edilmelidir.

Çizgeler özelliklerine göre çeşitli sınıflara ayrılmaktadır. Bunlar:

Yönsüz (undirected) çizge: Çizgede bulunan kenarlarda yön sınırlaması yok ise yönsüz çizge olarak adlandırılır.

Yönlü (directed) çizge: Çizge üzerindeki kenarlarda sadece gösterilen yönde geçişe izin verildiği durumlarda çizge, yönlü çizge olarak tanımlanmaktadır.

Basit yönlü çizge: Herhangi iki düğüm arasında aynı yönlü sadece bir tane kenar bulunan çizgelerdir.

Multi yönlü çizge: Herhangi iki düğüm arasında birden çok aynı yönlü kenar içeren çizgelerdir.

Karışık yönlü çizge: Hem yönlü hem de yönsüz kenarlara sahip çizgelerdir.

Basit (simple) çizge: Basit çizgelerde her kenar iki düğümü birleştirmeli (yani döngü içermemeli) ve bu iki düğüm arasında birden fazla kenar olmamalıdır.

Çoklu (Multi) çizge: İki düğüm arasında birden fazla kenar içeren çizgelerdir.

Yalancı (Pseudo) çizge: Döngü içeren çoklu çizgeler yalancı (pseudo) çizge olarak adlandırılır.

2.1.2 Zaman Karmaşıklığı (Time Complexity)

Bir problemin çözümü için genellikle birbirinden farklı algoritmalar geliştirmek mümkündür. Bu algoritmalar aynı sonuca ulaşsa da çözüm süreleri farklı olmaktadır. Bu durumlarda algoritmaların karmaşıklığı incelenerek hangisinin daha kısa sürede problemi çözebileceğinin tespit edilebilmesi bilgisayar biliminde önemli bir ihtiyaçtır. Algoritmaların zaman karmaşıklığı incelenerek hangisinin daha hızlı çözüm bulabileceği kolaylıkla tespit edilebilmektedir.

Zaman karmaşıklığı, bir algoritmanın gerçekleştirdiği temel işlemlerin sayısına bağlı olarak hesaplanmaktadır. Her temel işlemin sabit sürelerde tamamlandığı kabul edilmektedir. Algoritmaya verilen girdinin büyüklüğü, çözüm süresini etkileyeceği için, zaman karmaşıklığı ifade edilirken büyük O notasyonu (big O notation) kullanılmaktadır. Büyük O notasyonuna göre bir algoritmanın zaman karmaşıklığı $O(1)$, $O(n)$,$O(\log n)$ gibi ifadelerle gösterilir. Burada n, algoritmaya verilen girdinin boyutunu ifade etmektedir. En sık karşılaşılan zaman karmaşıklığı değerleri şunlardır:

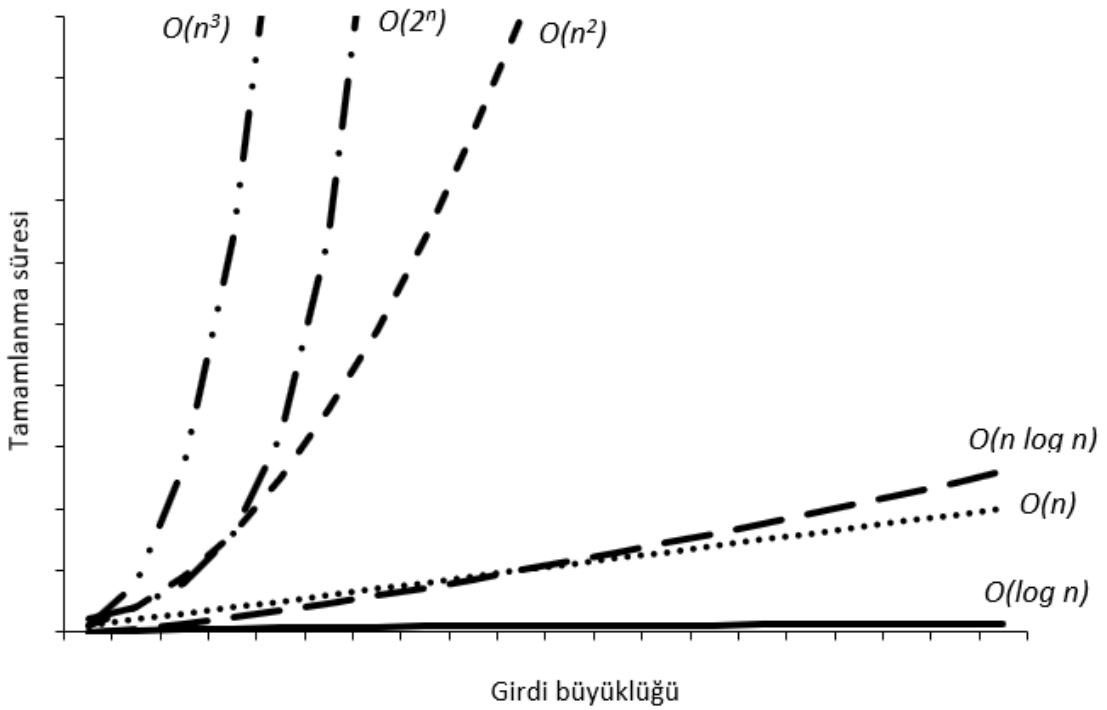
$O(1)$ = sabit: Tamamlanma süresi girdinin boyutuna bağlı olmayan algoritmalar için kullanılan zaman karmaşıklığı ifadesidir.

$O(n)$ = lineer: Girdinin boyutu arttıkça tamamlanma süresi de doğrusal olarak artan algoritmaların zaman karmaşıklığını ifade etmektedir.

$O(\log n)$ = logaritmik: Tamamlanma süresi girdi boyutuna göre logaritmik olarak artan algoritmalar için kullanılır.

$O(n^c)$ = çokterimli (polynomial) : Tamamlanma süresi girdi boyutuna göre çokterimli olarak artan algoritmalar için kullanılır. C değeri 1'den büyük olmalıdır.

$O(c^n)$ = üstel: Tamamlanma süresi girdi boyutuna göre üstel olarak artan algoritmalar için kullanılır. C değeri 1'den büyük olmalıdır. Çokterimli olana göre daha hızlı artar.



Şekil 2.5 Çeşitli zaman karmaşıklığına sahip algoritmalar için, girdi büyüklüğünün tamamlanma süresine etkisinin karşılaştırılması

Şekil 2.5'te, çeşitli zaman karmaşıklığına sahip algoritmaların tamamlanma sürelerinin (veya adım sayıları) problem büyüklüğüne göre nasıl değiştiği bir grafik üzerinde gösterilmiştir. Grafikte de görüldüğü gibi miktarının çok fazla artması durumundan en az etkilenen algoritmalar, logaritmik zaman karmaşıklığına sahip algoritmalardır. Bu nedenle büyük miktarda girdiye sahip problemleri, logaritmik zaman karmaşıklığına sahip algoritmaların diğerlerine göre çok daha kısa sürede çözebilmesi beklenmektedir. Bunlar dışında daha az yaygın olan farklı zaman karmaşıklıkları da bulunmaktadır. Bir algoritmanın denklemine bakarak zaman karmaşıklığını belirlemek için en hızlı artan kısmına bakılır. Örneğin, algoritmanın denklemi

$$f(n) = 5\,log(n) + 5n^3 + 6n + 8 \qquad (2.1)$$

ise, en hızlı artan kısmı grafikte de görüleceği üzere n^3 olduğundan

$$f(n) = O(n^3) \quad (2.2)$$

zaman karmaşıklığına sahip olduğu söylenebilir.

2.1.3 Dijkstra Algoritması

Edsger Wybe Dijkstra tarafından 1959 yılında geliştirilen Dijkstra algoritması, en kısa yol problemi için en bilinen algoritmalardan biridir. Dijkstra tarafından geliştirilen hali, sadece iki düğüm noktası arasındaki mesafeyi bulmaktadır ancak daha sonra geliştirilen birçok türevi bulunmaktadır. Bunlardan en yaygın olanı, bir başlangıç düğümünden diğer tüm düğümlere olan mesafeleri hesaplamaktadır.

Dijkstra algoritması $O(\ (|V|+|E|)\ log|V|\ )$ zaman karmaşıklığına sahiptir. Burada V parametresi düğüm sayısını, E parametresi ise kenar sayısını ifade etmektedir. Ancak algoritma yazılırken bir dizi (array) kullanılırsa, zaman karmaşıklığı $O\ (|V|^2)$ haline de getirilebilmektedir.

Dijkstra'nın algoritmasının döngüsü şu şekildedir:

1. İlk önce başlangıç düğümü hariç tüm düğüm noktalarına sonsuz maliyet değeri atanır. Başlangıç düğümü ise sıfır maliyetli kabul edilir. Düğümleri geçici ve kalıcı olarak ayırmak için iki liste oluşturulur.
2. Öncelikle başlangıç düğümü kalıcı düğümler listesine eklenir.
3. Kalıcı düğümler listesinin sonundaki düğümün tüm komşuları incelenerek hepsinin başlangıçtan itibaren kümülatif maliyet değerleri hesaplanır ve bu komşular geçici düğümler listesine eklenir.
4. Geçici düğümler listesindeki en düşük kümülatif maliyete sahip düğüm bulunarak kalıcı düğümler listesi taşınır. (Kalıcı düğümler listesine taşınan düğümler sonraki adımlarda artık incelenmez)

   Not: Geçici düğümler listesindeki bir düğüm eğer birden fazla yönden erişilebiliyor ise en düşük kümülatif maliyete sahip olan yön tercih edilir.
5. Tüm düğümler kalıcı listeye taşınana kadar adım 3-4 tekrarlanır.

2.1.4 *A** Algoritması

*A** algoritması, 1968 yılında Hart ve ark. [6] tarafından geliştirilmiştir. *A** algoritmasının Dijkstra algoritmasından en önemli farklı sezgisel bir yöntem içermesidir. Bu sayede çok daha hızlı sonuç verebilmektedir ancak sezgisel bir yöntem olduğundan dolayı her zaman en iyi sonucu vermeyi garanti edememektedir.

Algoritma, her adımda bir sonraki düğümü

$$f = g + h \tag{2.3}$$

fonksiyonuna göre belirler. Formüldeki "*g*" değişkeni, oluşturulan mevcut rotayı kullanarak başlangıç noktasından şu anki düğüme gelmek için gereken toplam masrafı belirtmektedir; "*h*" değişkeni ise sonraki düğümden hedefe ulaşmak için gereken tahmini masrafı ifade eder. Bu masrafın tahmin edilmesi için çeşitli sezgisel fonksiyonlar kullanmak mümkündür. Bunlar:

Öklid Mesafesi: Eğer çizgede hiç engel yok ise, "*h*" değeri Öklid mesafe formülü ile hesaplanabilir:

$$h = (x_{başlangıç} - x_{hedef})^2 + (y_{başlangıç} - y_{hedef})^2 \tag{2.4}$$

Manhattan Mesafesi: Bu yöntem genellikle grid ağ üzerinde çapraz harekete izin verilmediği durumlarda tercih edilmektedir. Bu yönteme göre "*h*" değeri, başlangıç ve hedef düğümlerinin koordinatlarının *x* değerleri arasındaki fark ile "*y"* değerleri arasındaki farkların mutlak değerleri birbirinden çıkarılarak hesaplanmaktadır.

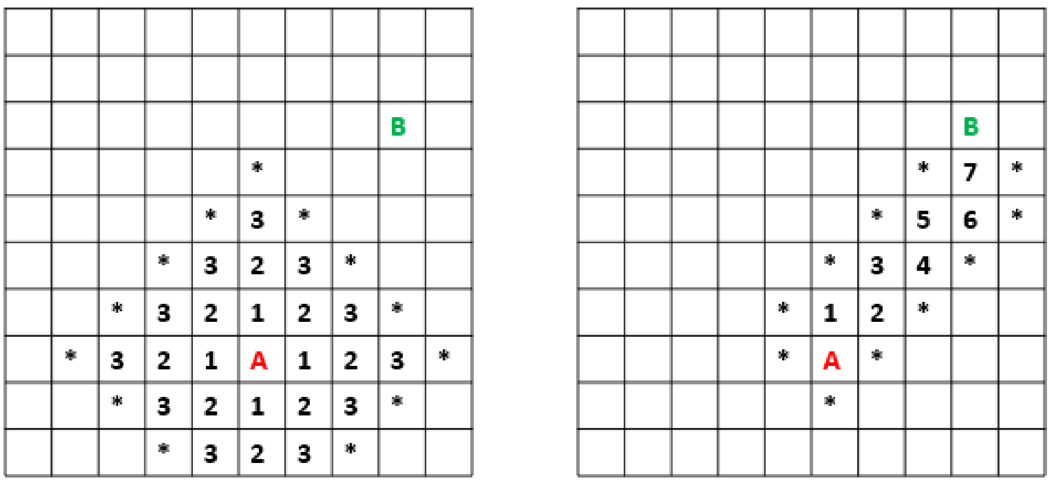
$$h = | x_{başlangıç} - x_{hedef} | + | y_{başlangıç} - y_{hedef} | \tag{2.5}$$

Çapraz Mesafe: Bu yöntem genellikle grid ağ üzerinde çapraz harekete izin verildiği durumlarda tercih edilmektedir. Bu yöntemde, başlangıç ve hedef noktalarının *x* koordinatları arasındaki fark ile *y* koordinatları arasındaki farkların mutlak değerleri karşılaştırılarak büyük olan değer *"h"* değeri olarak alınır:

$$h = maksimum\ (\ |\ x_{başlangıç} - x_{hedef}\ |,\ |\ y_{başlangıç} - y_{hedef}\ |\ ) \tag{2.6}$$

Kullanılan sezgisel fonksiyona göre A* algoritmasının zaman karmaşıklığı değişmektedir.

A* algoritması ile Dijksra algoritmasının çalışma yapıları arasındaki farkı Şekil 2.6 üzerinde görsel olarak görmek mümkündür. Görüldüğü gibi Dijksta algoritması her adımda tüm yönlerdeki düğüm noktalarını genişleyen bir daire şeklinde ararken A* algortimasının arama ağacı sezgisel yöntem aracılığı ile hedef noktasına doğru yönelerek genişletilmektedir. Bu nedenle eğer sadece bir adet hedef noktası var ise genellikle A* algoritması Dijksta algoritmasıdan daha hızlı çözüme ulaşmaktadır. Ancak eğer birden fazla hedef var ise tüm hedeflerin hesaplanmasını Dijkstra algoritması daha kısa sürede tamamlayabilmesi mümkün olabilir. Çünkü Dijkstra algoritması tüm düğüm noktalarını taradığı için tek döngüde tüm hedeflere giden rotaları bulabilmekte iken A* algoritması ise her hedef için ayrı bir döngü yapması gerekmektedir.



Şekil 2.6 Dijkstra (solda) ile A* (sağda) algoritmalarının A noktasından B noktasına giden en kısa yolu hesaplarken izledikleri adımların karşılaştırılması

2.1.5 Floyd-Warshall Algoritması

Dijksra algoritması bir başlangıç noktasından ağdaki diğer tüm noktalara olan mesafeleri hesaplarken, Floyd-Warshall algoritması ise ağ üzerindeki tüm düğümlerden diğer tüm düğümlere olan mesafeleri hesaplamaktadır.

Floyd-Warshall algoritması dinamik programlama prensibine göre çalışır. Her adımda tüm düğüm çiftleri arasındaki tüm olası rotalar hesaplanır ve sadece en iyi olanlar bir sonraki döngü için saklanır. Algoritmanın temel prensibi şudur: Eğer *A* düğümünden *C* düğümüne giden en kısa yol *B*'den geçiyorsa *A*'dan *B*'ye ve *B*'den *C*'ye giden alt yollar da en kısa yollar olmalıdır.

Floyd- Warshall algoritması $O(|V|^3)$ zaman karmaşıklığına sahiptir. Burada *V* düğüm sayısını ifade etmektedir. Algoritmanın çalışma yapısını açıklamak için 1 den *N*' ye kadar düğüm noktalarına sahip bir çizge düşünürsek; *k*. adımda *i*'den *j*'ye giden ve sadece *1*'den *k*'ye kadar olan düğümleri içeren en kısa yol $f(i,j,k)$ formatında bir fonksiyon ile hesaplanır. Sonraki adımda $(k+1)$, aynı şekilde sadece 1 den $k+1$' e kadar olan düğümler kullanılarak tüm *i,j* çiftleri arasındaki en kısa yollar hesaplanır. Bu adımda eğer

$$y_{k+1} = f(i,k+1,k) + f(k+1, j, k) \tag{2.7}$$

yolu, k. adımdaki

$$y_k = f(i,j,k) \tag{2.8}$$

yolundan daha kısa ise en kısa yol güncellenir, eğer daha kısa değilse

$$y_k = f(i,j,k) \tag{2.9}$$

olarak kalır.

Tüm en kısa yolları aynı anda bulabilmek için tüm düğüm noktası çiftlerinin maliyet değerleri Tablo 2.1'deki gibi bir tabloda tutulur. Aralarında bağlantı olmayan düğüm noktaları arasındaki maliyet sonsuz olarak yazılır. Bu tablo algoritmanın çalışma süreci boyunca her adımda güncellenir ve algoritma sona erdiğinde tüm düğüm noktaları arasındaki en kısa mesafeleri gösteren bir tablo elde edilir.

Tablo 2.1 Floyd-Warshall algoritmasının çözüm tablosu

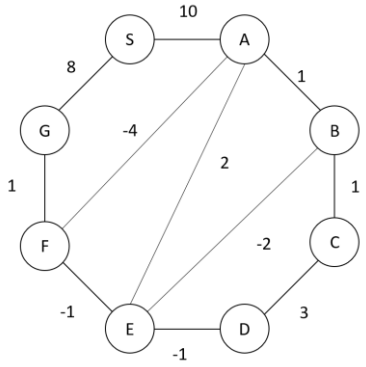
	a	**b**	**c**	**d**
a	0	9	7	15
b	∞	0	∞	6
c	∞	∞	0	5
d	∞	∞	∞	0

2.1.6 Bellman Ford Algoritması

Bellman Ford algoritması, çizge üzerindeki bir başlangıç noktası ile diğer tüm noktalar arasındaki en kısa yolları hesaplamak amacıyla kullanılmaktadır. Dijkstra ile aynı işleve sahip olan bu algoritma, $O(|V||E|)$ zaman karmaşıklığına sahip olduğu için Dijkstra algoritmasına göre daha yavaş olsa da Dijkstra algoritmasının çözemediği negatif ağırlıklı kenarlara sahip çizgeleri çözebildiği için genellikle bu tür çizgeler için tercih edilmektedir. Burada V değişkeni düğüm sayısını, E değişkeni ise kenar sayısını ifade etmektedir.

Tablo 2.2 Bellman Ford algoritmasının çözüm tablosu

	İterasyon							
Düğüm	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
S	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5
C	∞	∞	∞	∞	11	7	6	6
D	∞	∞	∞	∞	∞	14	10	9
E	∞	∞	12	8	7	7	7	7
F	∞	∞	9	9	9	9	9	9
G	∞	8	8	8	8	8	8	8



Şekil 2.7 Bellman Ford algoritması için örnek çizge ağı

Şekil 2.7'de görülen örnek bir çizge ağının Bellman Ford algoritması ile çözüm adımları Tablo 2.2'de gösterilmiştir. Her sütun, döngünün her yeni adımındaki düğümlerin mesafe değerlerini göstermektedir. Algoritmanın çalışma adımları şu şekildedir:

Bellman Ford algoritması ilk önce başlangıç düğümünden tüm diğer düğümlere olan mesafeleri sonsuz olarak kabul ederek başlar (kendine olan mesafesi ise "*0* "olarak işlenir). Ardından *|V|−1* adımlı bir döngüye başlar ve her adımda rahatlatma (relaxing) fonksiyonunu gerçekleştirir. Rahatlatma fonksiyonu şu şekilde çalışır:

Başlangıç düğümü *S*, değeri

$$dS = 0 \tag{2.10}$$

ona bağlı düğümlerden biri de *A*, değeri ise

$$dA = \infty \tag{2.11}$$

olsun. *S* ve *A* arasındaki kenarın ağırlığı ise

$$M = 10 \tag{2.12}$$

olsun. Eğer *A*'nın değeri (başlangıçtan A ya kadar olan mesafe),

$$dS + M \tag{2.13}$$

toplamından büyük ise

$$dA = dS + M \tag{2.14}$$

olarak güncellenir. Dolayısıyla

$$\infty > 0 + 10 \tag{2.15}$$

şartı sağlandığı için

$$dA = 0 + 10 = 10 \tag{2.16}$$

olarak güncellenir ve ayrıca *A* düğümünün öncesinde *S* düğümü olduğu da kayıt edilir. Bu bilgi daha sonra rotanın oluşturulması için gereklidir.

Bu döngü aynı şekilde $|V|-1$ kez tekrarlanır ve böylece tüm düğüm noktaları taranmış olur. Sonuç olarak başlangıç düğümünden tüm diğer düğümlere ulaşmak için en kısa yollar elde edilir.

2.1.7 Shortest Path Faster Algoritması (SPFA)

Daha hızlı en kısa yol algoritması anlamına sahip olan bu algoritma, Bellman Ford algoritmasının geliştirilmiş hali olarak tanımlanabilir. Algoritma 1959 yılında Edward F. Moore tarafından 'Algoritma D' adıyla geliştirilmiştir. Aynı şekilde bir başlangıç düğümü ile diğer tüm düğümler arasındaki en kısa yolları bulma amacıyla kullanılmaktadır. Zaman karmaşıklığı en kötü durumda Bellman Ford algoritması ile aynı, yani $O(|V||E|)$ olmaktadır. SPFA algoritması, yapılan geliştirme sayesinde Bellman Ford algoritmasındaki gibi her düğümü kör bir şekilde tek tek denemek yerine, aday düğümlerin listesini tutarak sadece bir düğüm rahatlatıldıysa (relaxed) onu bu listeye eklemektedir.

Bu şekilde rahatlatılacak hiç düğüm kalmayana kadar döngü devam etmektedir. Rahatlatma (relaxing) fonksiyonu Bellman Ford algoritması ile tamamen aynıdır. Yapılan geliştirme sayesinde SPFA algoritması, seyrek ve negatif ağırlıklı kenarlara sahip çizgeleri çözmede daha başarılı olmaktadır.

2.1.8 Johnson Algoritması

Johnson algoritması, bir çizgedeki tüm nokta çiftleri arasındaki en kısa yolları hesaplamak amacıyla Johnson tarafından 1977 yılında geliştirilmiştir [14]. Dağınık (sparse) ve yönlü (directed) çizgeler için kullanılabilmektedir.

Floyd Warshall algoritması da aynı işi yapabiliyor olsa da, Floyd Warshall algoritmasının $O(|V|^3)$ zaman karmaşıklığına sahip olması sebebiyle onun yerine genellikle $O(V^2 \log(V) + |V||E|)$ zaman karmaşıklığına sahip olan Johnson algoritması tercih edilmektedir. Seyrek çizgelerde Jonhson algoritması genellikle daha iyi olsa da, sık çizgelerde Floyd Warshall algoritması daha iyidir. Bunun nedeni Jonhson algoritmasının zaman karmaşıklığının çizgedeki kenar sayısına bağlı olmasıdır. Floyd Warshall algoritmasının zaman karmaşıklığı ise kenar sayısından etkilenmektedir.

Jonhson algoritması, negatif kenar ağırlıkları olan çizgeleri çözebilmektedir. Bu özelliği sağlamak için, negatif kenarları yeniden ağırlıklandırma (reweighting) yöntemini kullanarak elimine etmektedir. Bu işlem için Bellman-Ford algoritmasından yararlanmaktadır. Daha sonra, negatif kenarlar pozitife çevrildiği için, her nokta çifti arasındaki mesafeleri Dijkstra algoritmasını kullanarak hesaplayabilmektedir. Çıktı olarak ise ilk baştaki asıl çizgeye göre her nokta çifti için en kısa yolları içeren bir liste vermektedir.

Bellman Ford algoritmasının özelliği sayesinde negatif ağırlık döngüleri tespit edilebilir ve bu durumda Johnson algoritması çalışamayacağı için algoritma sonuca ulaşamadan sonlandırılır. Negatif ağırlık döngüsü, negatif ağırlığa sahip kenarların olduğu bir çizgede oluşan ve aynı döngüden her geçişte toplam ağırlığın azaldığı durumları ifade etmektedir.

Jonhson algoritmasının çalışma adımları şu şekildedir:

Çizgeye (G) yeni bir düğüm eklenir (S) ve bu düğüm ile diğer tüm düğümler arasına sıfır ağırlıklı kenarlar eklenir. Oluşan bu yeni çizge (G') üzerinde Bellman-Ford algoritması çalıştırılır ve çıktı kaydedilir.

$$h(v) = mesafe(s, v) \tag{2.17}$$

Bellman Ford algoritması kullanılarak elde edilen bu fonksiyon, çizgedeki herhangi bir düğümün (v), S düğümüne olan mesafesini vermektedir. Daha sonra yeniden ağırlıklandırma (reweighting) olarak adlandırılan işlem aşağıdaki ifade ile gerçekleştirilir (u,v herhangi iki düğümü ifade etmektedir) :

$$ağırlık\ '(u,v) = ağırlık\ (u,v) + h(u) - h(v) \quad (2.18)$$

Daha sonra S düğümü çıkarılır ve yeni ağırlıklar kullanılarak Dijkstra algoritması ile en kısa yollar (mesafe') hesaplanır. Ancak hesaplanan bu mesafeler asıl çizgeye göre hatalıdır. Bu nedenle son olarak aşağıdaki formül ile mesafe değerleri asıl çizgeye göre gerçek değerlerine çevrilir ve algoritma sona ermiş olur.

$$asıl\ mesafe(u,v) = mesafe'(u,v) + h(v) - h(u) \quad (2.19)$$

2.2 Bir Yapay Zeka Yöntemi ile En Kısa Yol Probleminin Çözümü

Önceki bölümde bahsedilen klasik algoritmalara alternatif olarak yapay zeka yöntemleri ile de en kısa yol probleminin çözümü mümkündür. Yapay zeka yöntemleri klasik algoritmalardan farklı olarak sezgisel yöntemler içermeleri sayesinde problemlere çok daha hızlı çözüm bulabilmektedirler. Ancak sezgisel yöntemler ile elde edilen sonuçların gerçekten optimum olduğu klasik yöntemlerdeki gibi kesin değildir, her zaman gerçek optimumu değil bazen optimuma yakın çözümleri bulabilmektedirler. Hızlı çözüm bulmanın önemli olduğu ve optimuma yakın sonuçların da yeterli kabul edilebildiği optimizasyon uygulamalarında yapay zeka yöntemleri tercih edilebilir bir alternatif olarak görülmektedir.

Günümüzde gittikçe sayıları artmakta olan yüksek ve karmaşık yapıya sahip binalarda çok sayıda sensör kullanılacağı ve bu nedenle yangın gibi dinamik durumlarda Dijkstra algoritması gibi klasik algoritmaların istenen hızda çözüm sağlayamayacağı öngörülmüştür. Bu nedenle çok daha hızlı çözüm verebilen bir en kısa yol algoritmasına ihtiyaç duyulacaktır. Böyle bir uygulamada hız avantajının daha öncelikli olacağı öngörülerek, her zaman gerçek optimum bulunamıyor, bazen optimuma yakın yollar bulunabiliyor olsa da bir yapay zeka yönteminin bu

uygulama için iyi bir alternatif olabileceği düşünülmüştür. (Tabii ki hata oranının da düşük olmasına dikkat edilmelidir.) Bu sebeple bu çalışmada, bir yapay zeka yönteminin böyle bir uygulamada kullanılabileceğinin gösterilmesi ve performansının incelenmesi hedeflenmiştir.

2.2.1 Kullanılacak Yapay Zeka Yönteminin Seçimi

Yapay zeka yöntemlerinin birçoğu doğadaki çeşitli canlıların problem çözme yöntemlerinden esinlenilerek geliştirilmiştir. Bunlara karınca kolonisi optimizasyonu, arı kolonisi optimizasyonu, balık sürüsü optimizasyonu, bakteri besin arama optimizasyonu, parçacık sürüsü optimizasyonu gibi çeşitli örnekler vermek mümkündür.

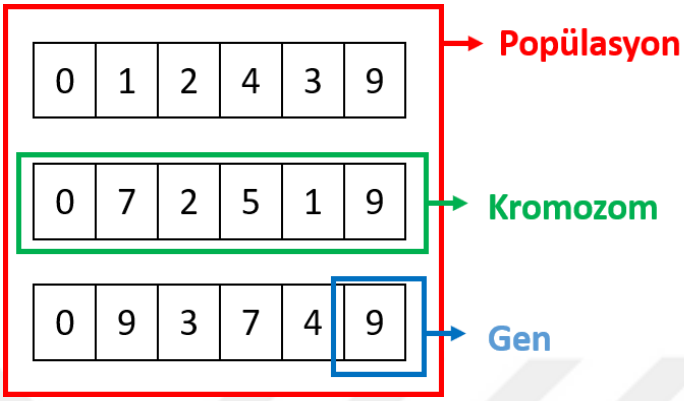
Yapılan literatür taramasında incelenen çalışmalarda, en kısa yol problemi için çeşitli yapay zeka yöntemlerinin kullanıldığı görülmüştür (Bölüm 1.1.3'te bu çalışmaların detayları açıklanmıştır). Bu çalışmaların değerlendirilmesi sonucunda en kısa yol problemi için genellikle yapay zeka yöntemleri arasında Genetik Algoritma yönteminin Dijkstra algoritmasına alternatif olarak tercih edildiği ve başarılı olduğu, ayrıca Genetik Algoritmanın bir akıllı bina tahliye sistemine uygulanması konusunda bir çalışma olmadığı görülmüş ve bu nedenlerle bu çalışmada Genetik Algoritmanın kullanılmasına karar verilmiştir.

2.2.2 Genetik Algoritma

Genetik Algoritma, 1975 yılında John Holland tarafından geliştirilen bir yapay zeka yöntemidir [22]. Genetik Algoritmayı optimizasyon problemlerini çözmek için kullanmak mümkündür ancak sezgisel bir yöntem olduğu için klasik yöntemlerdeki gibi kesin olarak optimum sonucun bulunacağı garanti edilememektedir. Ancak iyi tasarlandığında genellikle optimum veya optimuma çok yakın sonuçları bulabilmektedir. Bu dezavantajına rağmen hız avantajı nedeniyle klasik en kısa yol algoritmaları yerine tercih edilebilmektedir. Ayrıca Genetik Algoritmanın bir başka avantajı da sadece tek bir çözüm sunmak yerine en iyi çözümler kümesi

sunabilmesidir. Bu özellikleri çeşitli uygulamalar için önemli faydalar sağlayabilmektedir.

Genetik Algoritmanın temel kavramları popülasyon, kromozom ve gen kavramları arasındaki ilişki, Şekil 2.8'de bir şema üzerinde gösterilmiştir.



Şekil 2.8 Popülasyon, kromozom ve gen kavramları

Kromozom: "Gen" adı verilen birimlerden oluşan her "kromozom", çözülmek istenen problem için olası bir çözümü ifade edecek şekilde kodlanır. Kromozomlar genellikle ikili sistem veya tamsayılar kullanılarak oluşturulur.

Gen: Kromozomları oluşturan yapı taşları gen olarak adlandırılır. Gen olarak genellikle ikili sistem (0 ve 1 rakamları) veya tamsayılar kullanılmaktadır.

Popülasyon: Popülasyon kromozomların oluşturduğu kümeyi ifade etmektedir. Genetik algoritma döngüsü başlamadan önce bir başlangıç popülasyonu oluşturulmalıdır. Başlangıç popülasyonu, tamamen rastgele oluşturulan kromozomlar ile oluşturulabiliyor olsa da hızlıca çözüme ulaşabilmesi için genellikle başlangıç popülasyonunun belirli bir sezgisel fonksiyona göre oluşturulması tercih edilmektedir. Örneğin başlangıç ve bitiş noktaları belirli ise bu noktalar sabit tutularak, bunlar arasındaki diğer noktaların farklı kombinasyonları ile kromozomlar oluşturulabilir.

Uygunluk Fonksiyonu: Uygunluk fonksiyonu, kromozomları problemi çözme konusundaki performanslarına göre değerlendirir. Uygunluk fonksiyonu Genetik Algoritma tasarımının en önemli parçalarından biridir. Bu nedenle başarılı bir

Genetik Algoritma elde edebilmek için uygunluk fonksiyonunun iyi tasarlanması gerekmektedir.

Sonlandırma Kriteri: Sonlandırma kriteri Genetik Algoritma döngüsünün sona ermesi için gerekli görülen şartları ifade eder. Genellikle aşağıda verilen örneklerdeki gibi sonlandırma kriterleri tercih edilir:

- Son x nesildir daha iyi bir çözüm bulunamadı ise
- Uygunluk değeri belirli bir değerin üzerinde olan bir çözüm bulunana kadar
- Belirli bir nesil sayısına ulaşıldığında

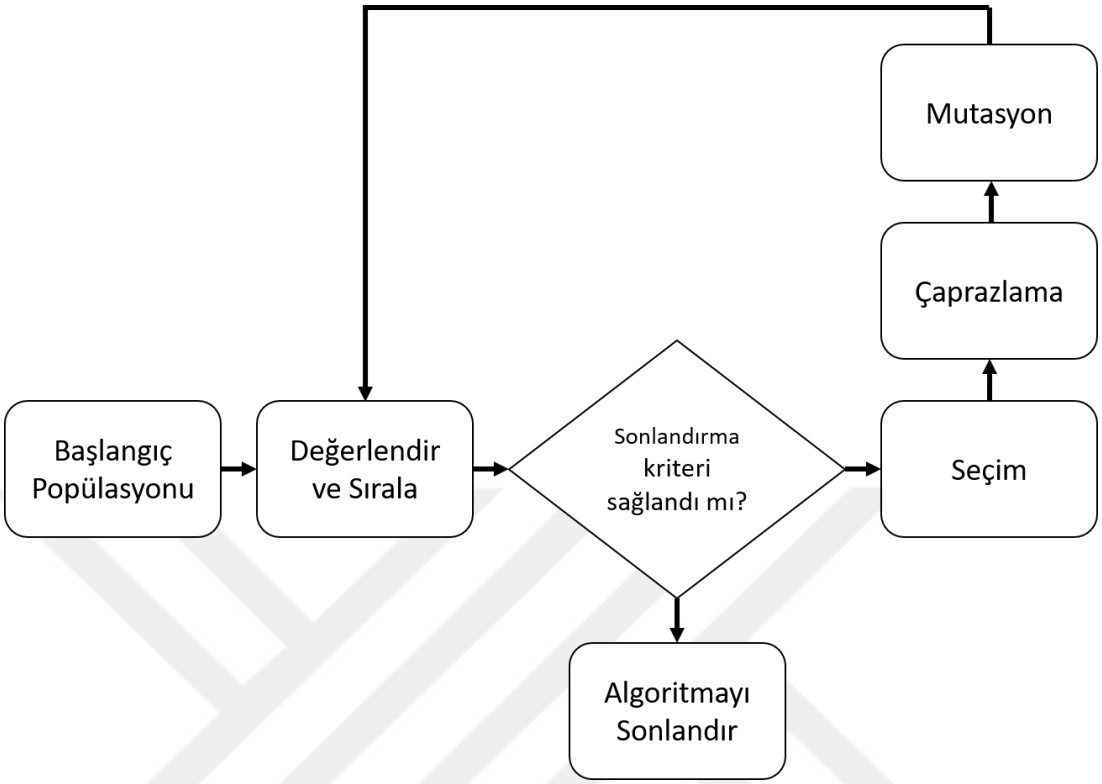
Seçim: Bir sonraki nesli oluşturmak için çaprazlama yapılacak ebeveynlerin popülasyondan seçilmesi işlemini ifade eder. Yaygın olarak kullanılmakta olan seçim yöntemleri ve bunların özellikleri ileride açıklanacaktır.

Çaprazlama: Seçilen ebeveyn kromozomlardan yeni kromozomların elde edilmesi için çaprazlama işlemi uygulanır. Çaprazlama işleminin genellikle her döngüde kesin değil belirli bir olasılıkla gerçekleşmesi tercih edilir. Çeşitli çaprazlama yöntemleri mevcuttur ve kullanılan çaprazlama yöntemine göre elde edilen yeni kromozomların ebeveynlerden aldıkları kısımlar değişmektedir. Bu yöntemler ileride detaylıca açıklanacaktır.

Mutasyon: Kromozom üzerinde rastgele değişiklikler yapılması işlemidir. Mutasyon işlemi her döngüde değil genellikle düşük bir olasılıkla uygulanmaktadır. Bu sayede popülasyondaki çeşitliliğin sağlanmasında ve lokal optimumlara takılmanın önlenmesinde rol oynamaktadır.

Genetik Algoritmanın genel çalışma döngüsünü gösteren blok diyagramı Şekil 2.9'da gösterilmiştir. Genetik Algoritma, başlangıç popülasyonunun oluşturulması ile döngüye başlar. İlk önce popülasyondaki kromozomlar uygunluk fonksiyonu ile değerlendirilir ve uygunluk değerlerine göre sıralanır. Daha sonra sonlandırma kriterinin sağlanıp sağlanmadığı kontrol edilir ve eğer kriter sağlanmadı ise seçim fonksiyonu ile ebeveyn seçimi yapılarak ardından popülasyonun yeni neslini oluşturmak için çaprazlama ve mutasyon işlemleri uygulanır. Ardından değerlendirme ve sıralama işlemlerinin uygulanması sonrasında sonlandırma

kriteri tekrar kontrol edilir ve sonlandırma kriteri sağlanıncaya kadar bu döngü aynı şekilde sürdürülür.



Şekil 2.9 Genetik algoritmanın blok diyagramı

Genetik algoritmanın tasarlanması için literatürde yer alan birçok yöntem mevcuttur ve tüm yöntemlerin çözülmek istenen probleme uygun olarak seçilmesi, Genetik Algoritmanın başarılı olmasında büyük bir öneme sahiptir. Mevcut yöntemleri kullanmak yerine ihtiyaç duyulursa yeni yöntemlerin geliştirilmesi de tercih edilebilmektedir. Şimdi sırasıyla bu yöntemleri inceleyelim:

Seçim Yöntemleri

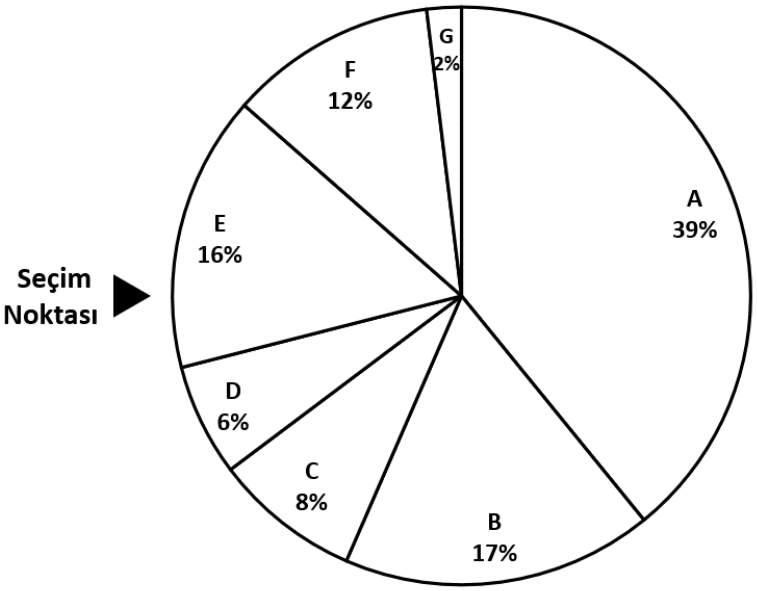
En bilinen seçim yöntemlerinden olan uygunlukla orantılı seçim (rulet seçimi), rank seçimi, turnuva seçimi ve stokastik evrensel örnekleme yöntemleri incelenecektir. Bunlar dışında literatürde çok çeşitli seçim yöntemleri de mevcuttur.

a) Uygunlukla orantılı seçim (Rulet seçimi)

Uygunlukla orantılı seçim (UOS) yönteminde (rulet seçimi olarak da bilinir), popülasyondaki her kromozomun seçilme olasılıkları uygunluk değerlerine göre ayarlanır. Daha yüksek uygunluk değerine sahip kromozomun rulet seçiminde daha yüksek seçilme olasılığı vardır. Örneğin Şekil 2.10'da *A* kromozomu dairenin en büyük kısmına sahiptir, çünkü Tablo 2.3'te görüleceği üzere en yüksek uygunluk değerine sahiptir. Bu seçim yöntemi, düşük uygunluk değerlerine sahip kromozomların da küçük bir ihtimalle de olsa seçilmesine izin verir, bu da popülasyondaki çeşitliliği artırır. Daha yüksek uygunluk değerine sahip kromozom için daha yüksek seçilme olasılığı prensibine dayandığından; bu yöntem minimizasyon problemleri için veya negatif veya sıfır uygunluk değerine sahip kromozomların bulunduğu problemler için kullanılamaz.

Tablo 2.3 Uygunlukla orantılı seçim için örnek popülasyon tablosu

Kromozom	Uygunluk Değeri
A	8,1
B	3,6
C	1,7
D	1,3
E	3,2
F	2,4
G	0,4



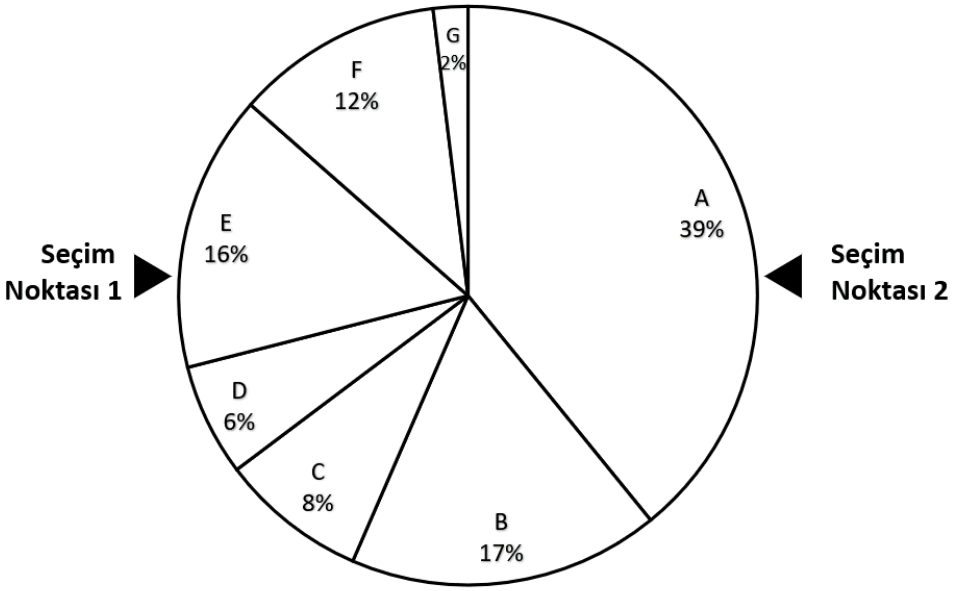
Şekil 2.10 Uygunlukla orantılı seçim çarkı

b) Stokastik Evrensel Örnekleme

Stokastik evrensel örnekleme (SEÖ) yöntemi, rulet seçim yönteminin daha gelişmiş bir versiyonu olarak gösterilebilir. Rulet seçim yönteminde her ebeveyni seçmek için fonksiyonun yeniden çalıştırılması gerekir, ancak stokastik evrensel örnekleme yönteminde tüm ebeveynler tek seferde seçilir. Bu durum rulet seçimi yöntemine kıyasla düşük uygunluk değerine sahip kromozomlara daha fazla seçilme ihtimali sağlar. Tablo 2.4'te görülen örnek bir popülasyon için Şekil 2.11'de gösterilen, rulet seçimindeki gibi bir daire oluşturulur ancak rulet seçiminden farklı olarak bu yöntemde her ebeveyn için bir seçim oku bulunur. Bu sayede tek döngü sonucunda tüm ebeveynler seçilmiş olur.

Tablo 2.4 Stokastik evrensel örnekleme için örnek popülasyon tablosu

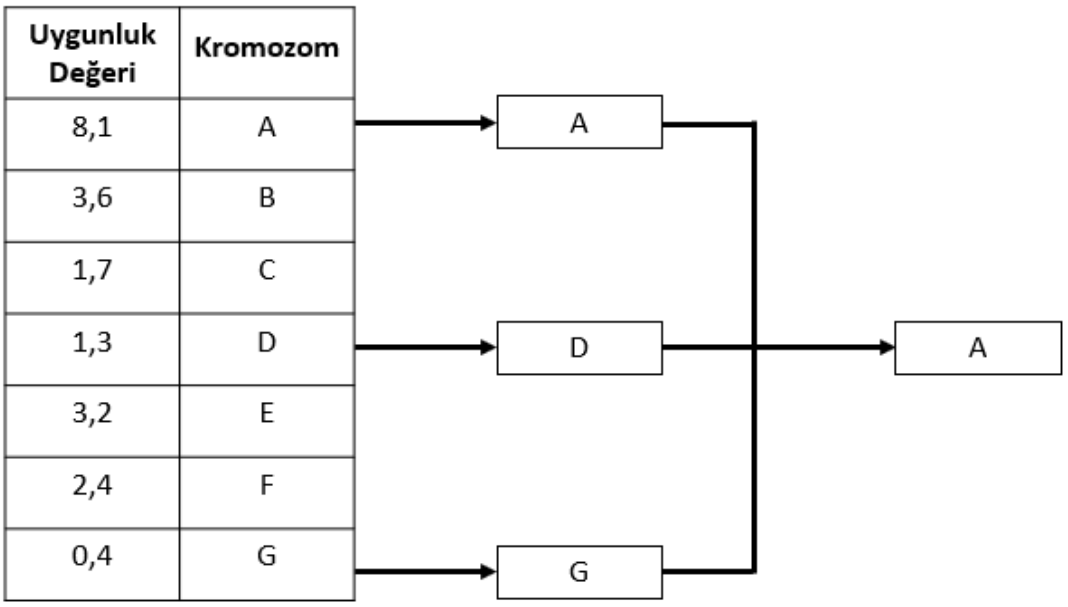
Kromozom	Uygunluk Değeri
A	8,1
B	3,6
C	1,7
D	1,3
E	3,2
F	2,4
G	0,4



Şekil 2.11 Stokastik evrensel örnekleme çarkı

c) Turnuva Seçimi

Turnuva seçimi yönteminde, turnuva için rastgele k adet kromozom seçilir (Şekil 2.12). Seçilenler arasından da en uygun olanı ebeveyn olarak seçilir. Turnuva büyüklüğünün arttırılması, düşük uygunluk değerlerine sahip kromozomların seçim ihtimalini azaltmaktadır. Turnuva seçimi yöntemi negatif uygunluk değerlerine sahip popülasyonlar için de kullanılabilir.



Şekil 2.12 Turnuva seçimi

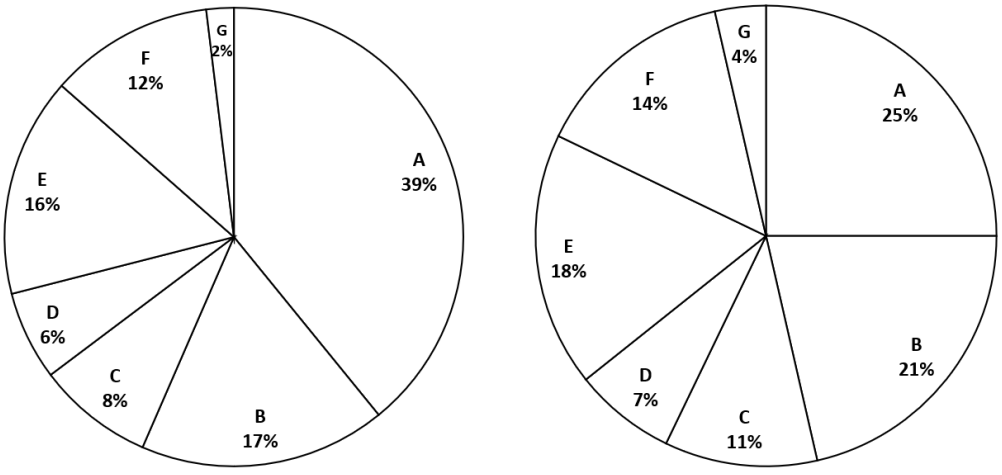
d) Rank Seçimi

Uygunlukla orantılı seçim yönteminde, en iyi kromozomların uygunluk değeri diğerlerine orantılı olarak çok yüksekse, diğerlerinin seçilme ihtimali çok düşük olacaktır. Rank seçim yöntemi bu sorunu çözmeyi amaçlamaktadır. İlk olarak tüm kromozomlar Tablo 2.5'te gösterildiği gibi sıralanır. Örneğin, en yüksek uygunluk değerine sahip kromozoma 1 değeri verilir, ikinci en iyi kromozom 2 değerini alır, diğerlerine de aynı şekilde sıra numarası verilir. Artık uygunluk değerleri değil verilen bu yeni numaralara göre seçilme ihtimalleri orantılanır. Bu işlem, Şekil

2.13'te görüldüğü gibi birinci durumda düşük uygunluk değerine sahip kromozomların seçilme ihtimalini arttırmaktadır.

Tablo 2.5 Rank seçimi için örnek popülasyon tablosu ve belirlenen rank değerleri

Kromozom	Uygunluk Değeri	Rank
A	8,1	7
B	3,6	6
C	1,7	3
D	1,3	2
E	3,2	5
F	2,4	4



Şekil 2.13 Rulet seçimi (solda) ile rank seçimi (sağda) yöntemlerinin karşılaştırılması

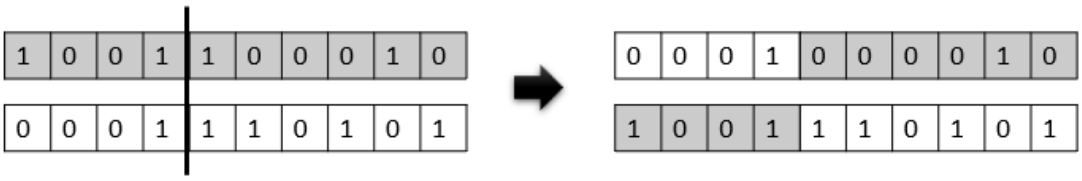
e) Rastgele Seçim

Bahsedilen seçim yöntemleri dışında uygulamaya özgü çeşitli yöntemler de geliştirilebileceği gibi tamamen rastgele bir seçim yapılması da tercih edilebilir. Ancak, tamamen rastgele seçimler yapıldığında kötü uygunluk değerine sahip kromozomların üst üste seçilmesi gibi riskler oluşacağı, bunun sonucunda da popülasyonun her nesilde daha iyiye gitmek yerine daha kötüye giderek istenmeyen sonuçların elde edilmesine yol açabileceği rahatça öngörülebilmektedir.

Çaprazlama Yöntemleri

a) Tek Noktadan

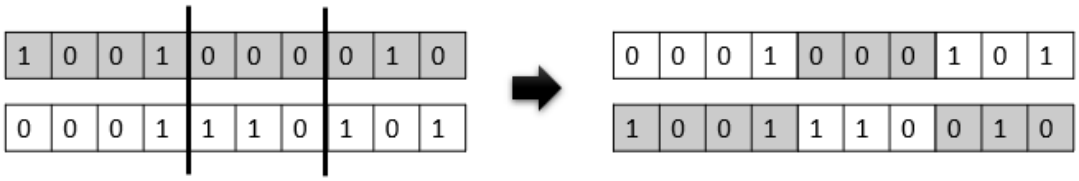
Tek noktadan çaprazlama işlemi, Şekil 2.14'te gösterildiği gibi iki kromozomun da aynı hizadan kesilerek birer parçalarının yer değiştirilmesi şeklinde gerçekleştirilir.



Şekil 2.14 Tek noktadan çaprazlama

b) Çok Noktadan

Çok noktadan çaprazlama işlemi, Şekil 2.15'te gösterildiği gibi iki kromozomun da aynı hizadan birden fazla yerden kesilerek parçalarının yer değiştirilmesi şeklinde gerçekleştirilir.



Şekil 2.15 Çok noktadan çaprazlama

c) Tekdüze (Uniform)

Tekdüze çaprazlama yönteminde ise her gen için tek tek diğer kromozomdaki karşılığı ile yer değiştirme yapılıp yapılmayacağı belirlenmektedir. Örnek bir uygulama Şekil 2.16'da gösterilmiştir.

Şekil 2.16 Tekdüze çaprazlama

Mutasyon Yöntemleri

Mutasyon işlemi popülasyondaki çeşitliliği sağlamak açısından çok önemlidir. Her döngüde mutasyon olmamalı, düşük bir olasılıkla gerçekleşmesi sağlanmalıdır. Her mutasyon yöntemi her problem için uygun olmayabilir. Bu nedenle mutasyon sonrasında istenmeyen sonuçların oluşmaması, oluşursa düzeltilmesi için önlemler alınmalıdır.

Çözülmek istenen probleme ve kromozom yapısına göre çeşitli mutasyon yöntemleri tasarlanabilir. En çok kullanılan mutasyon yöntemleri şunlardır:

a) Tek gen değiştirme

Bu yöntemde, Şekil 2.17'de gösterildiği gibi genlerden biri rastgele seçilir ve değeri değiştirilir. Kromozom 0 ve 1'lerden oluşuyor ise özel bir fonksiyona gerek yoktur ancak eğer tam sayılardan oluşuyor ise istenmeyen sonuçların önlenmesi için mutasyona uğratılan genin alabileceği değerlere kısıtlamalar getirilmesi gerekebilir.

Şekil 2.17 Tek gen değiştirme yöntemi

b) Yer değiştirme

Bu yöntemde, Şekil 2.18'de gösterildiği gibi rastgele seçilen iki genin yerleri değiştirilir.

Şekil 2 .18 Yer değiştirme yöntemi

c) Karıştırma

Bu yöntemde ise, Şekil 2.19'da gösterildiği gibi kromozom üzerinde rastgele seçilen bir bölgedeki genlerin sıralaması karıştırılır.

Şekil 2.19 Karıştırma yöntemi

d) Ters Çevirme

Bu yöntemde karıştırma ile benzer olarak, kromozom üzerinde rastgele bir bölge seçilir ve sıralamayı karıştırmak yerine Şekil 2.20'de gösterildiği gibi bölgenin sıralaması tersine çevrilir.

Şekil 2.20 Ters çevirme yöntemi

3
SİSTEM TASARIMI

3.1 Akıllı Bina Tahliye Sistemi İçin Genetik Algoritma Tasarımı

Genetik algoritma tasarımı her uygulama için özel olarak yapılmalıdır. Başarılı bir performans gösterebilmesi için Genetik Algoritmanın tüm fonksiyonları ve parametrelerinin uygulamaya uygun şekilde seçilmesine dikkat edilmesi gerekmektedir. Tasarım tamamlandıktan sonra yapılan benzetimlerle hangi yöntem ve parametrelerin daha iyi performans sağladığı belirlenerek Genetik Algoritmanın performansının iyileştirilmesi de mümkündür.

Akıllı bina tahliye sisteminde hızlı bir şekilde rotaların hesaplanabilmesi için Genetik Algoritma geliştirilmesine karar verilmiştir. Bu amaçla ilk önce rotaların kromozomlara nasıl kodlanacağı, başlangıç popülasyonun nasıl oluşturulacağı gibi temel basamaklar tamamlanmıştır. Ardından çeşitli yöntemler ve parametreler ile benzetimler yapma imkanı sağlayan bir benzetim yazılımı geliştirilmiş ve bu yazılımda yapılan benzetimler sonucunda en iyi performansı sağlayan yöntem ve parametreler belirlenmiştir. (Yapılan benzetimler ve sonuçları 4. bölümde paylaşılmıştır.) Aşağıda, akıllı bina tahliye sistemindeki rota hesaplamaları için geliştirilen Genetik Algoritmanın tasarım süreci ile ilgili detaylar açıklanmaktadır.

3.1.1 Rotaların Kromozomlara Kodlanması

Kromozomların nasıl kodlandığı algoritmanın performansını ciddi şekilde etkileyebileceğinden, problem için en uygun yöntemin seçilmesine önem verilmelidir. Kromozomlar genellikle tamsayılar veya 0 ve 1'lerden (ikili (binary) sistem) oluşan listeler şekilde kodlanır. Bu çalışmada geliştirilen yazılımın ilk versiyonunda, uygulama kolaylığı ve sonuçların rahatça gözlemlenebilmesi nedeniyle kromozomların tamsayı listesi olarak kodlanması tercih edilmiştir. Kat planındaki her düğüm noktası bir tamsayı değeri ile adlandırılmıştır. Bu sayede kromozomlar tam sayı listesi şeklinde oluşturulduğunda her kromozom bir rotayı

ifade etmektedir. Ancak bu durumda diğer parametreler ne kadar iyi seçilirse seçilsin Genetik Algoritmanın beklendiği kadar hızlı olamadığı görüldüğü için, sorunun kromozomların kodlanma yönteminden kaynaklanabileceği düşünülerek yeni bir versiyona geçilmiş, tüm yazılım güncellenmiştir. Yeni versiyonda kromozomların 0, 1, 2, 3 olmak üzere 4 rakam kullanılarak oluşturulması tercih edilmiştir. Bu rakamlar sırasıyla batı, kuzey, doğu, güney yönlerini temsil etmektedir. Kromozomlar artık düğüm noktalarını değil gidilecek yönleri içermektedir. Bu nedenle rotayı yazdırmak için, başlangıç düğümünden başlayarak bu yön komutları sırasıyla takip edilmekte ve uğranan her düğüm noktası rotaya eklenmektedir. Bu şekilde düğüm noktaları cinsinden rota elde edilebilmektedir.

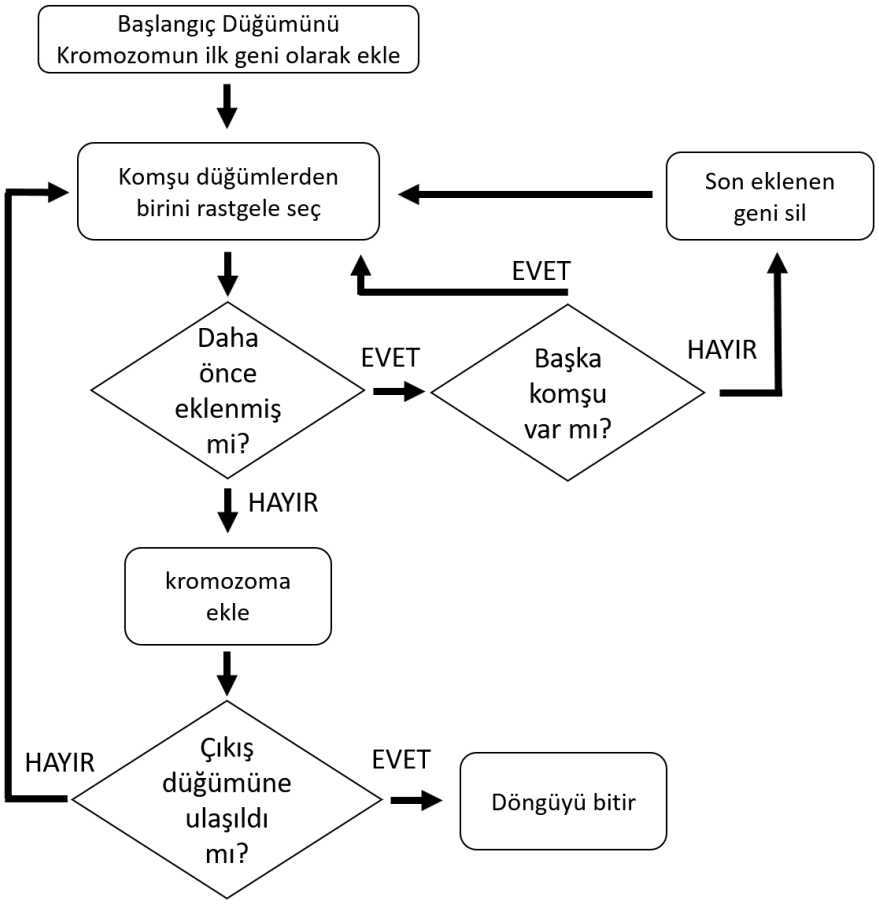
Kromozomların kodlanma yönteminin değişmesi nedeniyle yazılımın seçim, çaprazlama, uygunluk değerinin hesaplanması, gibi diğer kısımlarında da büyük değişiklikler yapılması gerektiği için yazılımda düzenleme yapmak yerine sıfırdan yeni bir yazılım oluşturulmuştur. Başlangıç popülasyonunun oluşturulması için özel bir fonksiyona ihtiyaç duymuyor olması sayesinde başlangıç popülasyonunun oluşturulma süresinde bile çok büyük hızlanma olduğu görülmüştür. Ayrıca çaprazlama ve mutasyon işlemleri için de anormal rotalar oluşturma riski olmadığı için özel fonksiyonlara gerek kalmamıştır. Bu sayede tüm işlemlerin çok daha hızlı tamamlanması sağlanmıştır. Ancak bu yöntemde kromozomlar tek başlarına bir rota ifade etmedikleri için onların öncelikle düğüm noktaları cinsinden rota ifadesine dönüştürülmeleri gerekmektedir. Bu ise biraz zaman alan bir işlem olsa da diğer kısımların oldukça hızlı olmasından dolayı toplam zaman açısından eski versiyona göre çok daha kısa sürede çözüme ulaşabildiği görülmüştür. Bu nedenle eski versiyon yarıda bırakılarak yeni versiyon üzerinden devam edilerek çalışma tamamlanmıştır.

3.1.2 Başlangıç Popülasyonunun Oluşturulması

İlk popülasyonu oluşturmak için, kromozomlar tamamen rastgele veya soruna dayalı bazı belirli kurallara göre üretilebilir. Ancak bina planındaki her düğüm

noktası arasında bağlantı olmadığı için istenmeyen rotaların oluşmasının önüne geçmek amacıyla başlangıç popülasyonu oluşturulurken rastgele listeler oluşturulması yerine belirli bir algoritmaya göre kromozomların oluşturulması gerekmektedir.

Bu nedenle geliştirilen yazılımın ilk versiyonunda başlangıç popülasyonunun oluşturulması için Şekil 3.1'de gösterilen algoritma geliştirildi:



Şekil 3.1 Başlangıç popülasyonunun oluşturulması

Kromozomlar aşağıdaki kurallar çerçevesinde oluşturulur. Bu sayede, başlangıç popülasyonundaki tüm kromozomların geçerli rotaları ifade etmesi sağlanmaktadır.

Kurallar:

- İlk gen her zaman başlangıç düğümüdür

- Son gen her zaman çıkış düğümüdür

- Kromozomların boyutu sabit değildir

- Daha önce geçtiği düğümlerden tekrar geçemez

- Eğer ilerleyebileceği uygun komşu düğüm kalmadı ise bir adım geri giderek başka bir yoldan devam etmeye çalışır. Ancak beş defa geri çekilme denemesinden sonra yine de bir yol bulamıyorsa bu rota iptal edilerek başlangıç noktasından tekrar başlanır.

- Üst kata geri dönmeyi önlemek için, merdivenler arasındaki yol tek yönlü olarak tanımlanmıştır (sadece aşağı doğru).

- En son, eğer rota üzerinde döngüler var ise bunları kaldırmak için bir onarım fonksiyonu çalıştırılır.

Yazılımın yeni versiyonuna geçildiğinde kromozomlar; batı, kuzey, doğu, güney yönlerini ifade eden 0,1,2,3 rakamlarından oluştuğu için, kromozomlar rastgele oluşturulduğunda yaşanan hatalı rotaların (komşu olmayan düğümler arasında geçiş içeren rotalar) oluşması sorunu da ortadan kalkmıştır. Bu nedenle artık kromozomların oluşturulması bu kurallara göre değil tamamen rastgele olarak yapılabilmektedir. Bu da Genetik Algoritmanın eski versiyona oranla oldukça hız kazanmasını sağlamıştır.

3.1.3 Uygunluk Fonksiyonunun Tasarımı

Uygunluk fonksiyonunun tasarımı Genetik Algoritmanın performansı için çok önemlidir. Uygunluk fonksiyonu, kromozomdaki genlerin toplam maliyeti ile ters orantılı bir uygunluk değeri verecek şekilde tasarlanmıştır. Bir kromozomdaki sırasıyla her düğüm noktası arasındaki mesafelerin toplamı "*t*" olarak ifade edilirse

1/t ifadesi uygunluk değerini vermektedir. Bu durumda en kısa rota en yüksek uygunluk değerine sahip olacaktır.

Yeni versiyonda her kromozom çıkışa ulaşamadığı için, uygunluk fonksiyonuna çıkış noktasına kuş uçuşu mesafe değerinin de etki etmesi sağlanarak kısa da olsa çıkışa ulaşamayan rotaların düşük uygunluğa sahip olması sağlanmıştır.

3.1.4 Seçim Fonksiyonunun Tasarımı

Çaprazlama yapılacak ebeveynlerin seçim işlemi, seçim yöntemlerinden biri kullanılarak yapılmaktadır. Tasarlanan benzetim uygulaması üzerinde seçim işlemi için turnuva seçimi, uygunlukla orantılı seçim, stokastik evrensel örnekleme ve rank seçimi seçenekleri eklenmiştir. Bu seçeneklerden biri ile popülasyon içinden iki adet kromozom seçilerek çaprazlama işlemi gerçekleştirilir.

3.1.5 Çaprazlama Fonksiyonunun Tasarımı

Çaprazlama fonksiyonunun tasarımı için öncelikle bilinen çeşitli çaprazlama yöntemlerini incelenmiştir ve ardından bu çalışmadaki probleme uygun bir çaprazlama fonksiyonu geliştirilmiştir. İlk versiyonda, rotaları belirten kromozomların uzunluklarının değişken olması ve her düğüm noktası arasında bağlantı olmaması nedeniyle klasik sabit bir noktadan çaprazlama, tekdüze (uniform) çaprazlama gibi yöntemler kullanılamamakta idi. Bu nedenle ilk versiyonda, iki kromozom arasında ortak genlerin tespit edilip bu noktalardan çaprazlama işlemini gerçekleştirecek bir çaprazlama fonksiyonu tasarlanmıştır. Önce iki kromozom arasındaki tüm ortak genler tespit edilmekte (aynı hizada olmalarına gerek yoktur) ardından bu genlerden biri rastgele seçilerek bu genlerin bulunduğu nokta yan yana getirilerek çaprazlama işlemi gerçekleştirilmektedir. Eğer ebeveynler arasında ortak gen tespit edilemezse ebeveynler çaprazlama yapılmadan çıktı olarak verilmektedir.

Ancak yeni versiyona geçilmesi ile, yeni bir kromozom oluşturmak yerine kromozom içinden rastgele olarak seçilen bir veya birkaç gen rastgele bir rakam ile (0, 1, 2 veya 3) değiştirilerek mutasyon işlemi daha hızlı gerçekleşecek hale getirilmiştir.

3.1.6 Mutasyon Fonksiyonunun Tasarımı

İlk versiyonda, mutasyon fonksiyonu her döngüde düşük bir ihtimalle uygulanır ve uygulandığında çaprazlama işlemi sonrası üretilen kromozomlardan birini, rastgele olarak yeni üretilmiş bir kromozom ile değiştirir. Yeni kromozom, ilk popülasyonu oluşturmak için kullanılan fonksiyon tarafından üretilmektedir.

Ancak yeni versiyona geçilmesi ile, yeni bir kromozom oluşturmak yerine rastgele bir gen değiştirilerek mutasyon işlemi daha hızlı gerçekleşecek hale getirilmiştir.

3.2 Kalabalık seviyesine göre ilerleme hızında oluşacak yavaşlamanın hesaplanması

Geliştirilen akıllı bina tahliye sisteminde bulunan kalabalık sensörleri, yerleştirildikleri koridorların kalabalık seviyelerini tespit etmektedir. Elde edilen bu ölçümler, rotalar hesaplanırken göz önünde bulundurularak aşırı kalabalık nedeniyle oluşacak yavaşlama ve izdihamların önüne geçilmesi amaçlanmaktadır. Bunu gerçekleştirebilmek için ölçülen kalabalık seviyesinin ne kadar yavaşlamaya neden olacağının hesaplanarak o koridorun mesafe değerine belirli bir ekleme yapılması gerekmektedir. Bu sayede Genetik Algoritma, rota hesaplaması yaparken o koridorun daha uzun hale geldiğini görecek ve daha kısa bir rota oluşturmaya çalışacaktır.

Kalabalık bir grubun ilerleme hızının tespiti konusu çok uzun zamandır üzerinde çalışılan bir konudur. Z. Fang ve ark. 2003 yılında yayınladıkları bir makalede [23], literatürdeki kalabalık seviyesi ile ilerleme hızı arasındaki ilişki üzerine yapılan çeşitli çalışmaları incelemişler ve bu çalışmalara ait formüllere yer vermişlerdir. Bu

formüller arasında hesaplama açısından en basit olan Nelson [24] tarafından geliştirilen formül olduğu görülmektedir. Bu çalışmada çok hızlı bir hesaplamaya ihtiyaç duyulduğu için ilk versiyonda bu formülün kullanılması tercih edilmiş idi. Formül *m/s* cinsinden kalabalıktaki bir insanın ilerleme hızını vermektedir:

$$S = k(1-a\rho) \tag{3.1}$$

Burada ρ parametresi insan yoğunluğunu (m^2 başına düşen insan sayısı) belirtmekte, *k* parametresi ise bulunulan yerin tipine göre değişen bir katsayıdır ve koridor için 1,4 olarak verilmiştir. Bir diğer parametre olan a parametresi ise bina içindeki durum için 0,266 olarak verilmiştir. İnsan yoğunluğu, yani m^2 başına düşen kişi sayısının hesabı için koridorun taban alanı ve bulunan kişi sayısına ihtiyaç bulunmaktadır. koridor uzunluk ve genişlikleri zaten sabittir ve sistemin kurulduğu binanın bilgilerinin kurulum sırasında yazılıma girilmesi gerekmektedir. Kişi sayısının ise, görüntü işleme ile insan sayımı yapabilen yeni nesil güvenlik kameraları aracılığıyla tespit edileceği planlanmıştır.

Bu formül ile örnek bir hesaplama yapmak için, *10m* uzunluğunda, *3m* genişliğinde bir koridorda 60 kişilik bir kalabalığın olduğunu varsayalım. Bu durumda m^2 başına düşen kişi sayısı

$$60 / 3 * 10 = 2 \tag{3.2}$$

olacaktır. Buna göre, kalabalığın ilerleme hızı ise

$$S = 1,4 (1 – 0,266 * 2) = 0,6552 \ m/s \tag{3.3}$$

olarak hesaplanacaktır.

Z. Fang ve ark. [23] makalelerinde ayrıca Predtechenskii ve Milinskii [25] tarafından geliştirilen ve daha kabul edilebilir olduğunu belirttikleri şu formüle de yer vermişlerdir.

$$V = 112D4 - 380D3 + 434D2 - 217D + 57 \ (m/dk) \quad (3.4)$$

Bu formül Nelson'un formülüne göre daha karmaşık olsa da daha kabul edilebilir görüldüğü için geliştirilen akıllı bina tahliye sistemi yazılımında bu formülün kullanılması tercih edilmiştir.

Bu formülde D parametresi, alan bazında insan yoğunluğunu ifade etmektedir (m^2/m^2) yani m^2 kare başına bir insanın kapladığı alanı belirtmektedir ve şu formül ile elde edilir:

$$D = Nf / wL \quad (3.5)$$

Bu formülde *N* parametresi akıştaki insan sayısını, f parametresi insanın yatay kesit alanını (yerde kapladığı alanı), *w* parametresi akışın genişliğini, *L* parametresi ise akışın uzunluğunu ifade etmektedir. Denklem 2.23 ile bir kalabalığın ilerleme hızını hesaplayabilmek için öncelikle bu parametreler kullanılarak *D* değerinin hesaplanması gerekmektedir. Bu çalışmada koridorların genişlikleri w, koridorların uzunlukları L parametresi olarak alınmıştır. Diğer bir parametre olan f parametresi ise *0,125* m^2/m^2 olarak kabul edilmiştir. Bahsi geçen makalede, bu değerin Thompson tarafından [26] kış kıyafetleri içindeki ortalama bir insanın yatay yansıması olarak kabul edildiği belirtilmiştir . *N* parametresinin ise görüntü işleme ile insan sayımı yapabilen yeni nesil güvenlik kameraları aracılığıyla tespit edileceği planlanmıştır.

Örnek bir koridor için ilerleme hızının (*V*) hesabı şu şekilde yapılabilir:

$$w = 2 \ m \quad (3.6)$$

$$L = 10 \ m \quad (3.7)$$

$$f = 0,125 \quad (3.8)$$

$$N = 40 \quad (3.9)$$

Bu bilgilere göre;

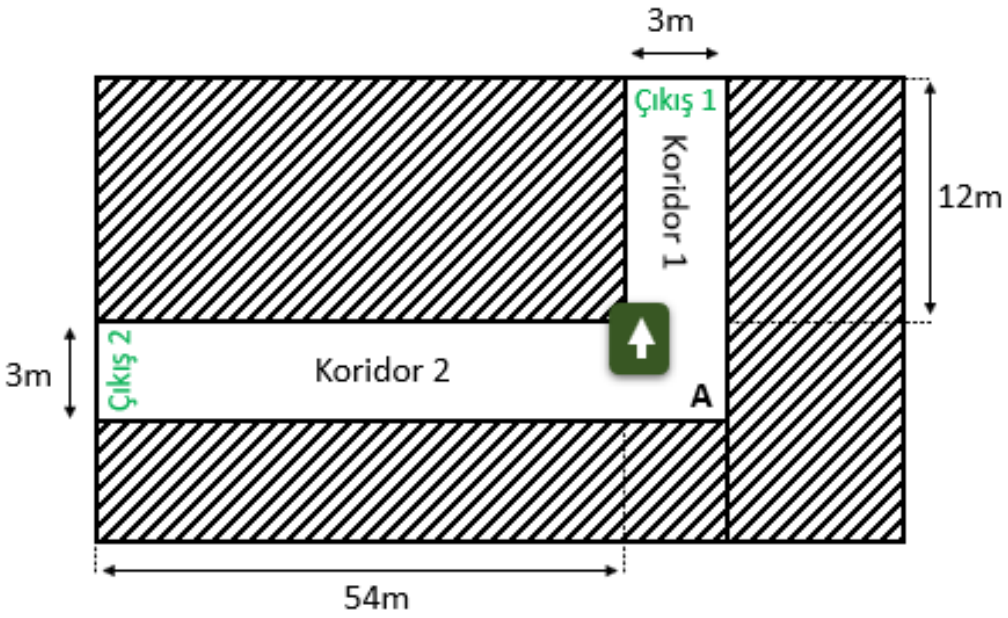
$$D = Nf / wL = 0{,}25 \tag{3.10}$$

$$V = 24{,}375 \quad m / dk = 0{,}40625\ m/s \tag{3.11}$$

olarak elde edilmektedir.

Bahsedilen bu iki formülden biri ile ilerleme hızı hesabı yapıldıktan sonra, bu hız değişiminin etkisini Genetik Algoritmaya kolayca aktarmak amacıyla koridorun uzunluk değeri değiştirilmektedir. Bu sayede bir koridorda kalabalık arttığında, ABTS yazılımı o koridorun uzunluğunun arttığını görüp daha kısa olan alternatiflere yönlendirebilmekte veya kalabalık azaldığında koridor uzunluğunun kısalması nedeniyle tekrar bu koridora yönlendirmeye karar verebilmektedir.

Örnek bir benzetim yapmak için Şekil 3.2'de gösterilen kat planındaki *A* noktasından çıkışlara ulaşmayı sağlayan alternatif iki koridoru ele alalım;



Şekil 3.2 Örnek kat planı

Koridorların boş olduğu ilk durumda sistem çıkışa en yakın olan koridor 1'e yönlendirecektir. Bir süre sonra koridor 1 kalabalıklaşmaya başlayacak ve belirli bir

değere ulaşıldığında artık koridor 2 daha hızlı hale gelecek ve sistem artık koridor 2'ye yönlendirmeye başlayacaktır. Bu yön değişiminin olacağı kalabalık miktarını hesaplayalım:

Kalabalık yok iken acil durumda ortalama bir insanın ilerleme hızını Park ve ark. ait bir çalışmayı [27] referans alarak 3 m/s olarak kabul edelim. Bu durumda kişi, 1. koridoru

$$12/3 = 4 \tag{3.12}$$

saniyede geçerek çıkışa ulaşacaktır. İkinci koridordan gitse idi

$$54 / 3 = 18 \tag{3.13}$$

saniyede çıkışa ulaşabilirdi. Bu durumda görüldüğü gibi koridor 1 daha avantajlıdır. Ancak koridor 1 de kalabalık oranı arttığında, kişi sayısının 72 olduğunu kabul edersek;

Yoğunluk değeri:

$$72 / (3 * 12) = 2 \tag{3.14}$$

olacaktır. Buna göre, Nelson formülüne göre kalabalığın ilerleme hızı ise

$$S = 1,4 (1 – 0,266 * 2) = 0,6552 \ m/s \tag{3.15}$$

olacaktır. Bu hız değeri ile koridor 1 i tamamlama süresi

$$12/0,6552 = 18,3 \tag{3.16}$$

saniye olacaktır. Sistem bu durumda koridorun uzunluk değerini

$$x/3 = 18,3 \tag{3.17}$$

denklemine göre

$$x = 54,9 \ m \tag{3.18}$$

olarak değiştirecektir. Bu durumda 2. koridor hala boş olduğu için, uzunluğu 54m, tamamlanma süresi de hala 18 saniye olduğundan daha avantajlı duruma gelmiş olacaktır ve bu nedenle sistem artık 2. koridora yönlendirmeye başlayacaktır. Tabii ki bir süre sonra 2. koridor da kalabalıklaşmaya başlayacak, bu sırada da 1. koridor da boşalacağı için sistem tekrar 1. koridorun uzunluk değerini düşürecek ve 2. koridora göre daha avantajlı hale geldiğinde tekrar 1. koridora yönlendirmeye başlayacaktır. Sistem bu şekilde tabelaları sürekli değiştirerek aşırı kalabalığın oluşmasını önlemeye çalışacaktır.

3.3 Akıllı Bina Tahliye Sistemi (ABTS) Benzetim Yazılımının Tasarımı

Tasarlanan Akıllı Bina Tahliye Sisteminin test edilmesi ve Genetik Algoritma parametreleri ve kullanılan yöntemlerde değişiklikler yapılarak performansa etkilerinin gözlemlenmesi amacıyla bir benzetim yazılımı geliştirilmiştir.

Yazılım C# programlama dili kullanılarak yazılmıştır.

ABTS benzetim yazılımının kullanıcı arayüzü Şekil 3.3'te görülmektedir. Binanın ilk 6 katını görsel olarak gösterebilmekte, daha fazla katlı binalar için ise yazılı olarak çözüm rotasını sunabilmektedir.

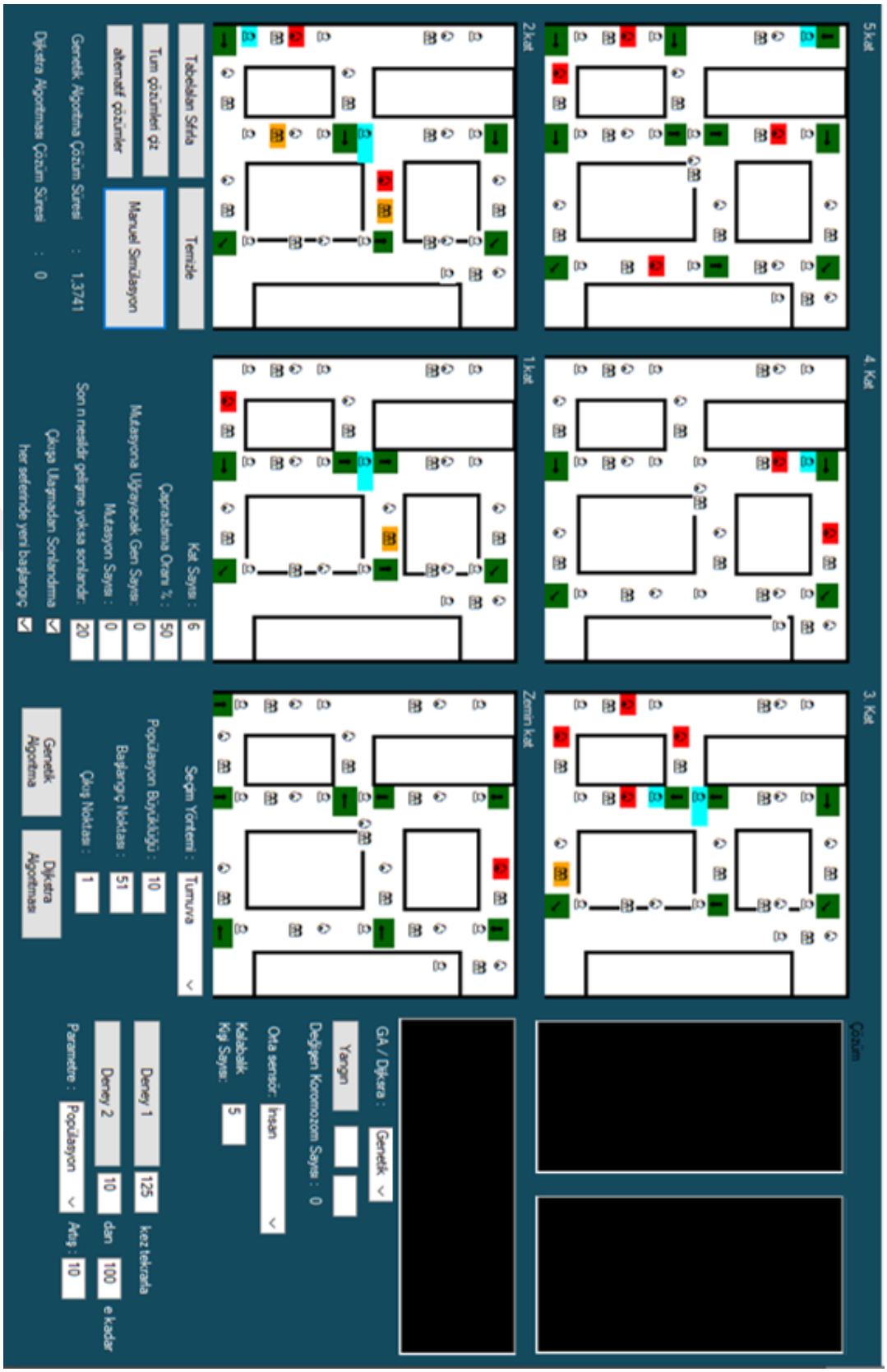
Manuel benzetim butonuna tıklandığında kat planları üzerinde yangın, kalabalık ve insan algılama butonları görülmektedir. Bu butonlara tıklandığında bu sensörden sinyal alındığı bilgisi yazılıma iletilmektedir ve yazılım da buna göre yeni rotalar hesaplayarak tabelaları oluşturmakta ve güncellemektedir.

Sağ kısımda kalabalık sensörü (kişi sayımı) girişi bulunmaktadır. Bu bölümde, kat planında herhangi bir kalabalık butonuna basıldığında o kalabalık sensörüne verilecek giriş (kişi sayısı) değeri ayarlanmaktadır.

Alt kısımda ayrıca Genetik Algoritma için parametre ve yöntemlerin seçimi için ayarlamaların yapılabildiği bir kısım yer almaktadır.

Bunlar dışında binanın kaç katlı olacağı, başlangıç ve çıkış noktası gibi ayarlar da yapılabilmektedir.

Ayrıca Genetik Algoritmaya alternatif olarak klasik en kısa yol bulma algoritmaları arasında en bilinenlerden biri olan Dijkstra algoritması ile rotaların hesaplanması için de bir seçenek sunulmuştur. Bu sayede farklı durumlarda Genetik Algoritma ile Dijkstra algoritmasının karşılaştırılabilme imkanı sağlanmıştır. Yapılan çeşitli karşılaştırma testleri örnek vaka ve benzetimler bölümünde bulunmaktadır.



Şekil 3.3 ABTS simülasyon yazılımı kullanıcı arayüzü

4

ÖRNEK VAKA VE BENZETİMLER

4.1 Kullanılan Yöntem ve Parametrelerin Genetik Algoritma Performansına Etkilerinin İncelenmesi

Benzetim 1: Seçim Yönteminin Etkisi

Benzetimin amacı:

Seçim yönteminin Genetik Algoritmanın hata oranı ve çözüm süresine etkisinin incelenmesi

Yöntem:

Genetik algoritma ve Dijkstra algoritması sırayla kullanılarak, 500 düğüm noktasına sahip bir binada, iki nokta arasındaki en kısa yolun bulunması işlemi, her bir popülasyon büyüklüğü için 100'er kez tekrarlanmıştır ve elde edilen çözüm sürelerinin ortalaması alınmıştır. Bu sayede benzetim sırasında oluşabilecek dış etkenlerden (yazılımın arka planındaki işlemler, bilgisayarda çalışan diğer işlemler, hatalar vs.) kaynaklanacak anlık hatalı süre ölçümlerinin önlenmesi amaçlanmıştır.

Sabit Tutulan Parametreler:

Düğüm Noktası Sayısı: 500

Popülasyon Büyüklüğü: 10

Çaprazlama Oranı: %50

Mutasyona Uğrayan Gen Sayısı: 10

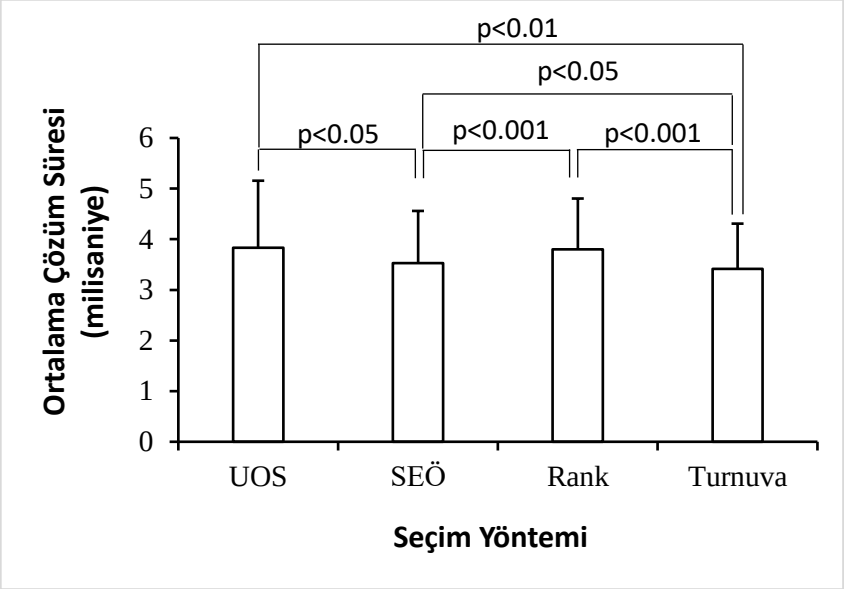
Mutasyona Uğrayan Kromozom Sayısı: 10

Sonlandırma kriteri: son 5 nesil boyunca bir gelişme yok ise sonlandır

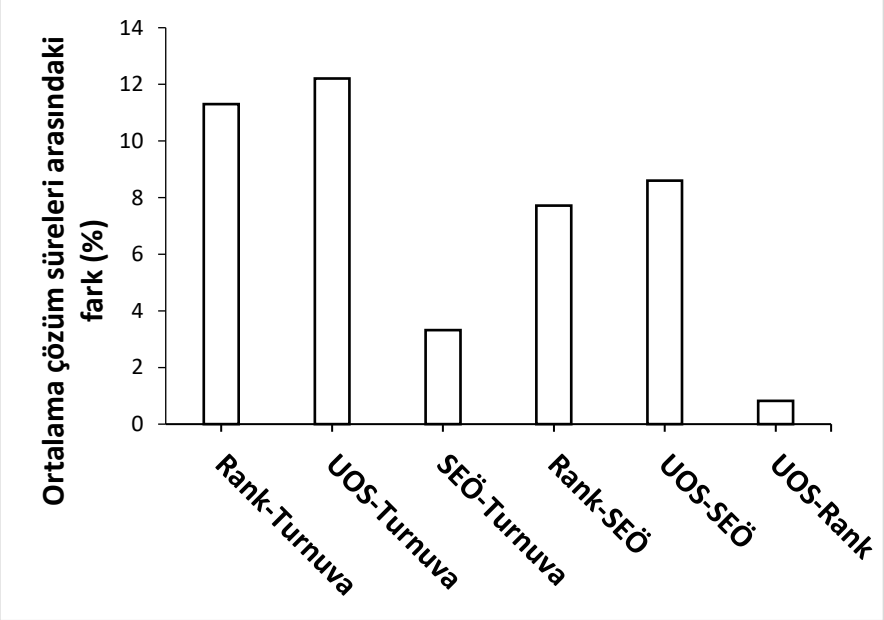
Benzetim Sonuçlarının Değerlendirilmesi:

Benzetim sonucunda elde edilen grafikte görüldüğü üzere (Şekil 4.1), test edilen dört farklı seçim yöntemi arasında en hızlı çözüm süresini veren seçim yöntemi turnuva seçimi olmuştur. Bu benzetimde sadece 1 adet rota oluşturulduğu için süre farkı çok düşük olsa da, çok sayıda düğüm noktasına sahip büyük ve karmaşık binalarda aynı anda çok fazla sayıda rota oluşturulması gerekeceği için bu süre farkı saniyeler seviyesine yükselecektir. Dinamik bir durumda bu rotaların çok hızlı bir şekilde anlık olarak güncellenmesi gerekeceği için bu saniyeler seviyesindeki süre farkı önemli bir sorun oluşturacaktır. Aradaki farkların daha anlaşılır bir şekilde gösterimi için yüzde cinsinden farklar Şekil 4.2'de gösterilmiştir.

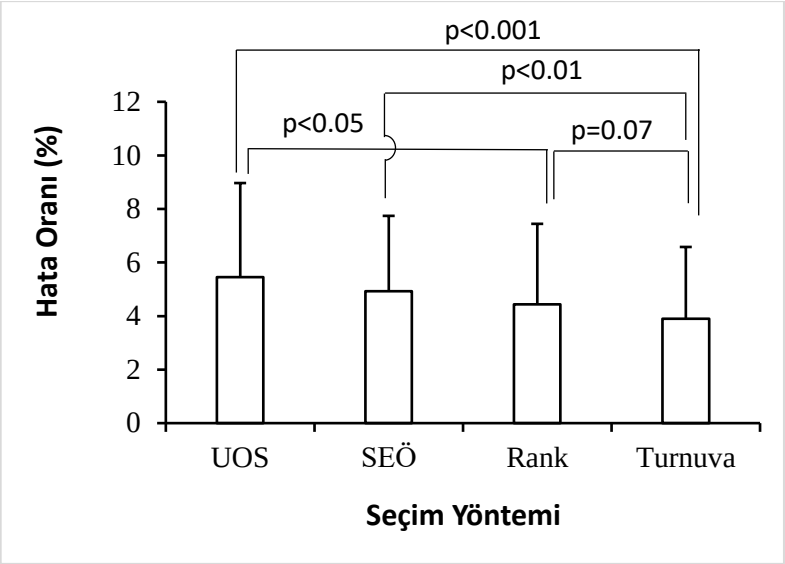
Çözüm süresinin dışında hata oranı da önemli bir kriter olduğu için onun da incelenmesi gerekli görülmüştür. Şekil 4.3'te de görüldüğü gibi hata oranı açısından da turnuva seçiminin aynı zamanda en düşük hata oranına da sahip olduğu belirlenmiştir. Burada da değerler birbirine yakın olduğu için aradaki farkların daha anlaşılır bir şekilde görülebilmesi için yüzde cinsinden farklar Şekil 4.4'te gösterilmiştir. Bu sonuçlar değerlendirildiğinde, turnuva yönteminin en uygun seçenek olduğu tespit edilmiştir.



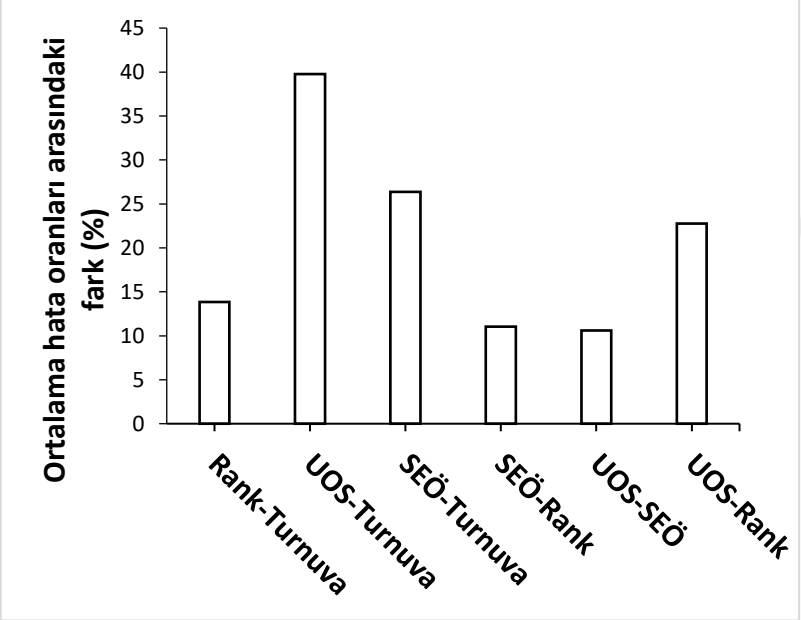
Şekil 4.1 Seçim yönteminin çözüm süresine etkisi



Şekil 4.2 Seçim yöntemlerinin çözüm süreleri arasındaki yüzde cinsinden farklar



Şekil 4.3 Seçim yönteminin hata oranına etkisi



Şekil 4.4 Seçim yöntemlerinin hata oranları arasındaki yüzde cinsinden farklar

Benzetim 2: Popülasyon Büyüklüğünün Etkisi

Benzetimin amacı:

Popülasyon büyüklüğünün Genetik Algoritmanın hata oranı ve çözüm süresine etkisinin incelenmesi

Yöntem:

Genetik algoritma ve Dijkstra algoritması sırayla kullanılarak, 500 düğüm noktasına sahip bir binada, iki nokta arasındaki en kısa yolun bulunması işlemi, her bir popülasyon büyüklüğü için 100'er kez tekrarlanmıştır ve elde edilen çözüm sürelerinin ortalaması alınmıştır.

Sabit Tutulan Parametreler:

Düğüm Noktası Sayısı: 500

Seçim yöntemi: Turnuva seçimi

Çaprazlama Oranı: %50

Mutasyona Uğrayan Gen Sayısı: 0

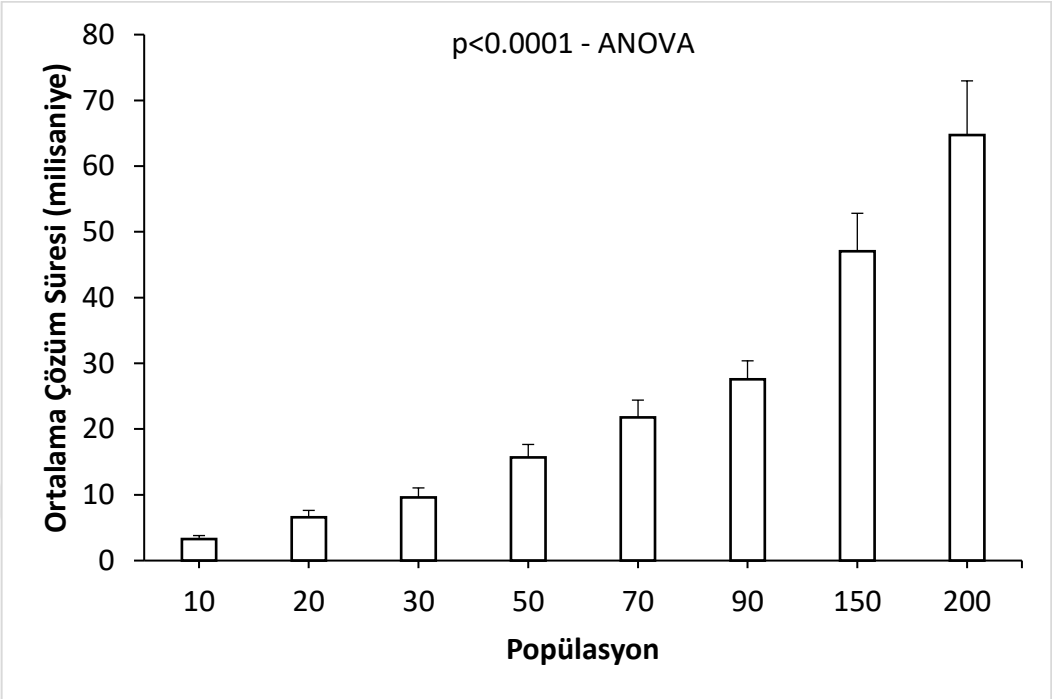
Mutasyona Uğrayan Kromozom Sayısı: 0

Sonlandırma kriteri: son 5 nesil boyunca bir gelişme yok ise sonlandır

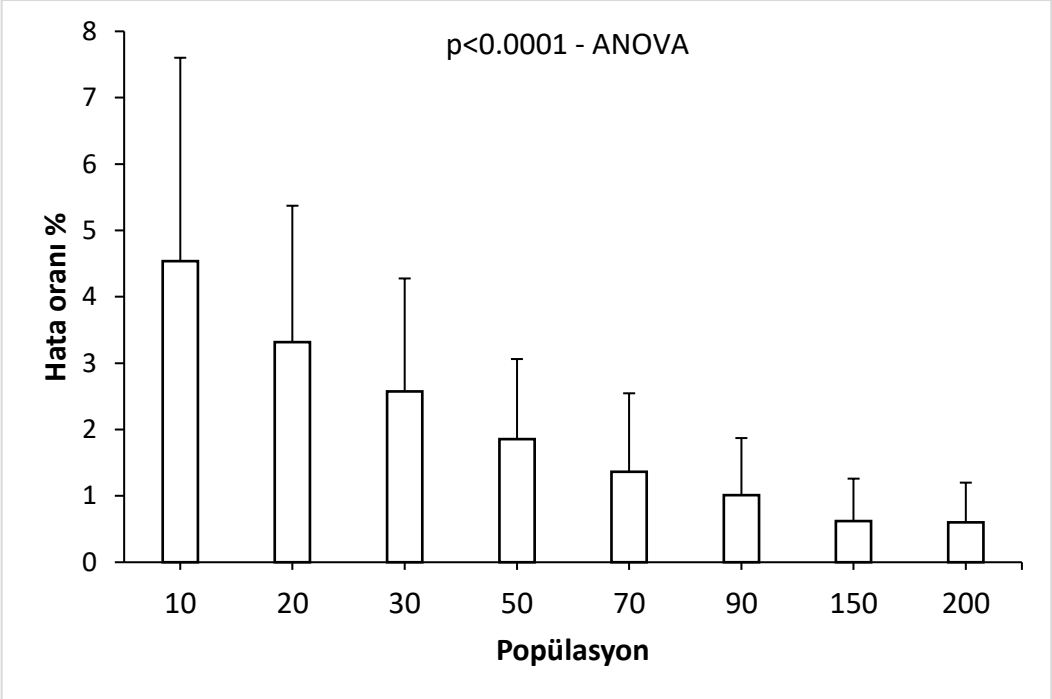
Benzetim Sonuçlarının Değerlendirilmesi:

Benzetim sonuçları ile oluşturulan grafikte (Şekil 4.5) görüldüğü gibi, popülasyon büyüklüğü arttıkça çözüm süresi uzamaktadır. Bunun nedeni, kromozom sayısı arttığı için bunların işlenmesinin daha fazla zaman almasıdır. Bu sonuçlara göre, hızlı bir çözüm bulabilmek için popülasyon büyüklüğünün düşük tutulması gerektiği görülmektedir. Ancak Şekil 4.6'da da görüldüğü gibi popülasyon büyüklüğü azaldıkça, Genetik Algoritmanın hata oranı da artmaktadır. Bu nedenle popülasyon

büyüklüğü seçilirken istenen çözüm süresini sağlayan en fazla popülasyon büyüklüğünün seçilmesi gerekmektedir.



Şekil 4.5 Popülasyon büyüklüğünün ortalama çözüm süresine etkisi



Şekil 4.6 Popülasyon büyüklüğünün hata oranına etkisi

Benzetim 3: Mutasyon Oranının ve Sayısının Etkisi

Benzetimin amacı:

Mutasyona uğrayan kromozom sayısının ve mutasyon sırasında değişikliğe uğrayan gen sayısının Genetik Algoritmanın hata oranı ve çözüm süresine etkisinin incelenmesi

Yöntem:

Genetik algoritma ve Dijkstra algoritması sırayla kullanılarak, 500 düğüm noktasına sahip bir binada, iki nokta arasındaki en kısa yolun bulunması işlemi, her bir popülasyon büyüklüğü için 100'er kez tekrarlanmıştır ve elde edilen çözüm sürelerinin ortalaması alınmıştır.

Sabit Tutulan Parametreler:

Düğüm Noktası Sayısı: 500

Popülasyon Büyüklüğü: 20

Seçim yöntemi: Turnuva seçimi

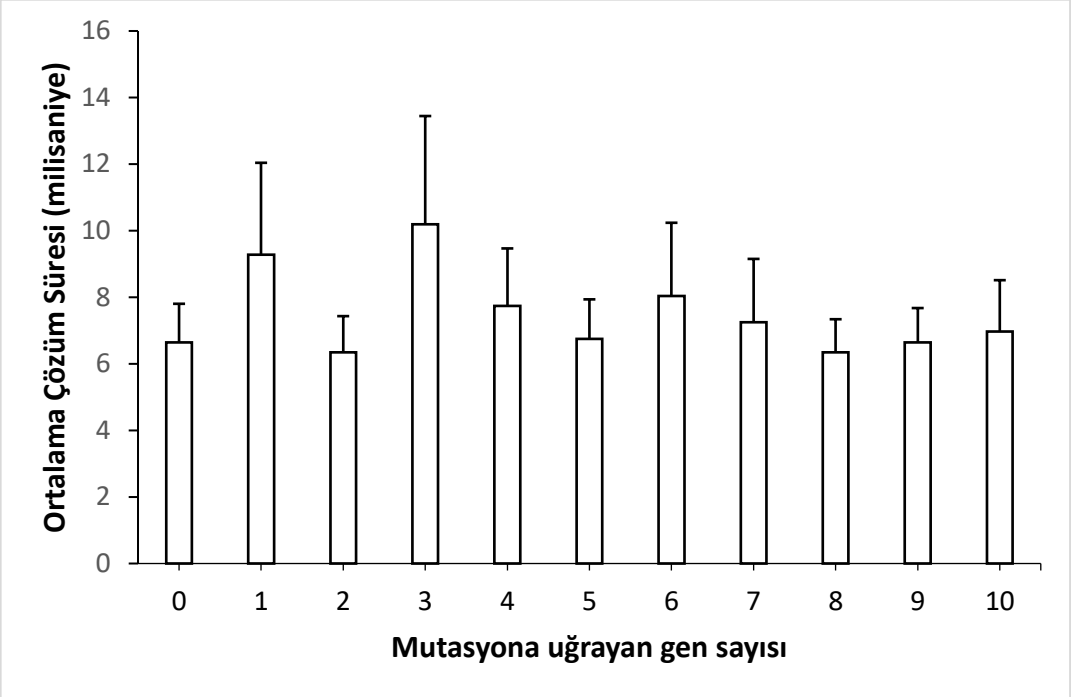
Çaprazlama Oranı: %50

Sonlandırma kriteri: son 5 nesil boyunca bir gelişme yok ise sonlandır

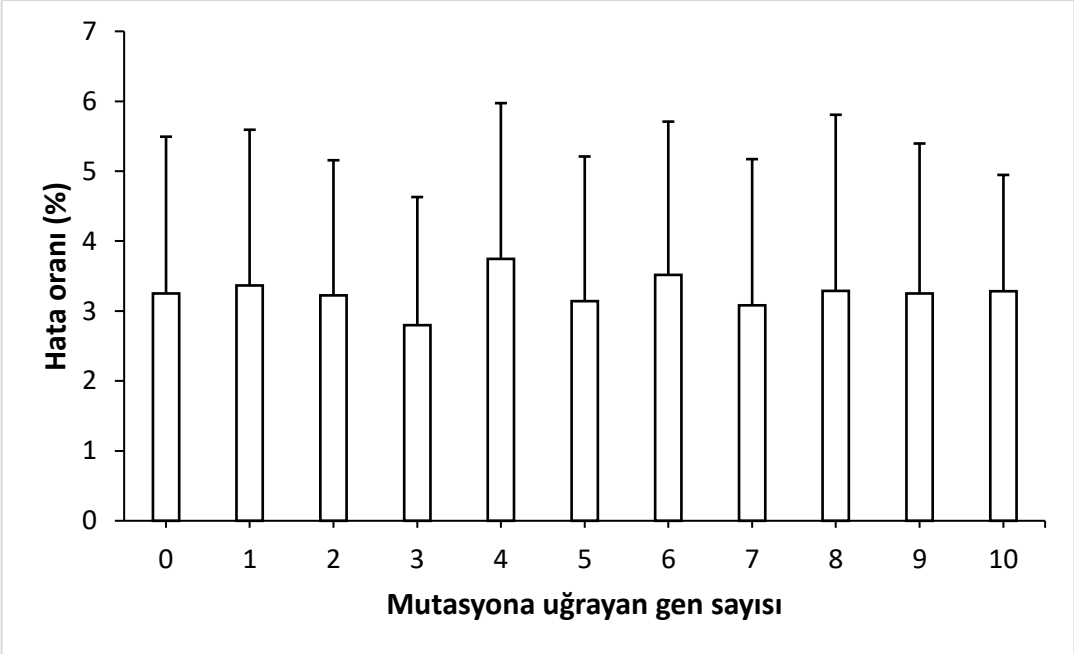
Benzetim Sonuçlarının Değerlendirilmesi:

Yapılan benzetimlerde, mutasyona uğrayan kromozom sayısı ve mutasyon sırasında değişikliğe uğrayan gen sayısı 0'dan 10'a kadar arttırılarak Genetik Algoritmanın çözüm süresi ve hata oranına etkileri gözlemlenmiştir. Aşağıdaki grafiklerde (Şekil 4.7, 4.8, 4.9, 4.9), mutasyon parametrelerindeki değişimin çözüm süresi veya hata oranına belirli bir etkisinin olmadığı görülmektedir. Sonuç olarak bu durumun bu çalışmadaki problem ve Genetik Algoritma tasarımıyla ilgili olduğu varsayılmış, bu

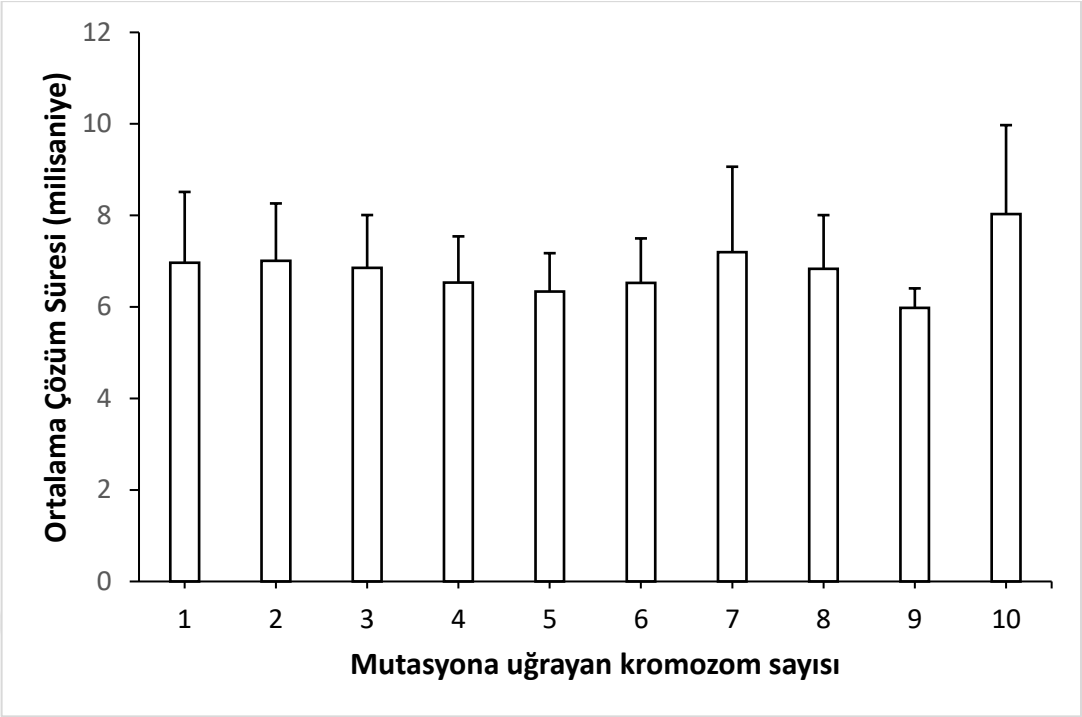
çalışmadaki problem için mutasyonun çok da etkili ver gerekli olmadığına karar verilmiştir.



Şekil 4.7 Mutasyon oranının ortalama çözüm süresine etkisi



Şekil 4.8 Mutasyon oranının hata oranına etkisi



Şekil 4.9 Mutasyona uğrayan kromozom sayısının ortalama çözüm süresine etkisi

Şekil 4.10 Mutasyona uğrayan kromozom sayısının hata oranına etkisi

Benzetim 4: Çaprazlama Oranının Etkisi

Benzetimin amacı:

Çaprazlama oranının Genetik Algoritmanın hata oranı ve çözüm süresine etkisinin incelenmesi

Yöntem:

Genetik algoritma ve Dijkstra algoritması sırayla kullanılarak, 500 düğüm noktasına sahip bir binada, iki nokta arasındaki en kısa yolun bulunması işlemi, her bir popülasyon büyüklüğü için 100'er kez tekrarlanmıştır ve elde edilen çözüm sürelerinin ortalaması alınmıştır.

Sabit Tutulan Parametreler:

Düğüm Noktası Sayısı: 500

Popülasyon Büyüklüğü: 20

Seçim yöntemi: Turnuva seçimi

Mutasyona Uğrayan Gen Sayısı: 0

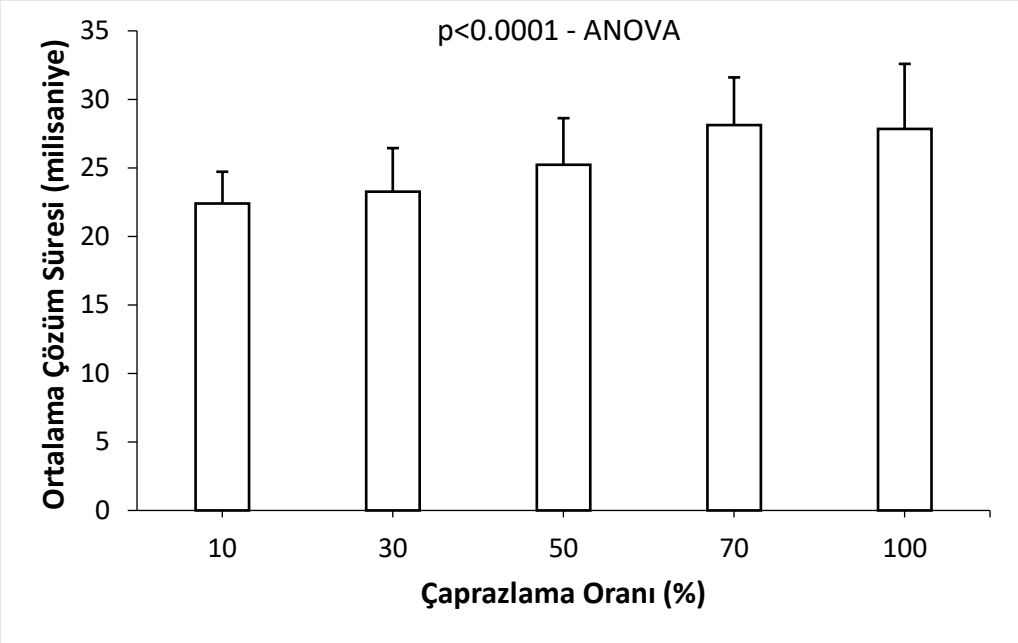
Mutasyona Uğrayan Kromozom Sayısı: 0

Sonlandırma kriteri: son 5 nesil boyunca bir gelişme yok ise sonlandır

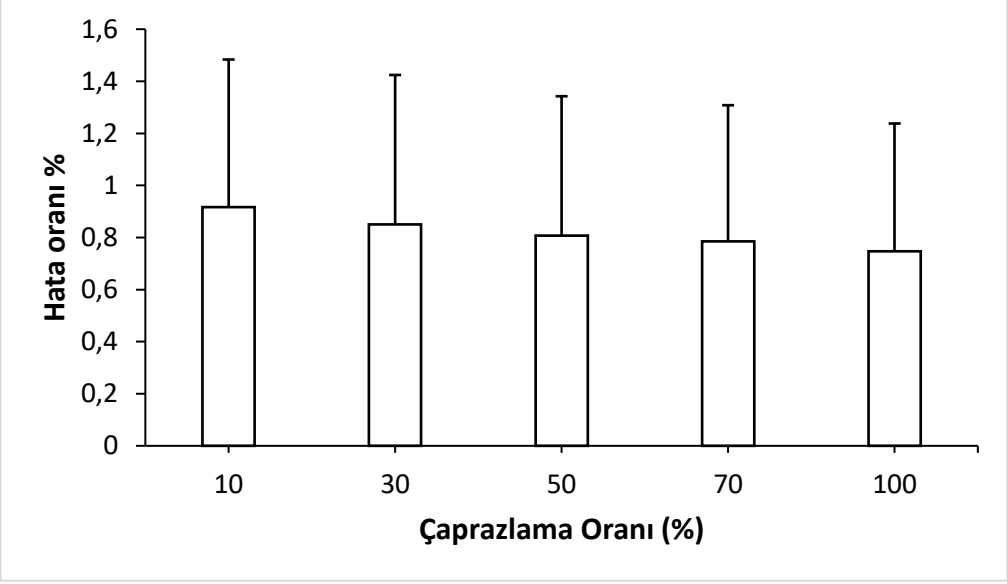
Benzetim Sonuçlarının Değerlendirilmesi:

Benzetim sonucunda elde edilen grafikte (Şekil 4.11), test edilen farklı çaprazlama oranları arasında en hızlı çözüm süresinin, popülasyonun %10 unun çaprazlandığı durumda elde edildiği görülmektedir. Bunun sebebi daha fazla çaprazlama işlemi yapıldığında daha fazla işlem süresi harcanmasıdır. Hata oranı açısından ise Şekil 4.12'de görüldüğü üzere en düşük hata oranı, popülasyonun %100 ünün çaprazlandığı durumda elde edilmiştir. Bunun sebebi ise çaprazlama oranının

artmasının çeşitliliği arttırarak daha iyi kromozomların daha hızlı ortaya çıkmasını sağlamasıdır. Bu sonuçlara göre çözüm süresi açısından çaprazlama oranının düşük tutulması mantıklı görünse de hata oranı çok daha önemli olduğundan popülasyonun %100 ünün çaprazlanmasının en uygun seçenek olduğu belirlenmiştir.



Şekil 4.11 Çaprazlama oranının ortalama çözüm süresine etkisi



Şekil 4.12 Çaprazlama oranının hata oranına etkisi

4.2 Genetik Algoritma ile Dijkstra Algoritmasının Çözüm Bulma Hızlarının Karşılaştırılması

Önceki bölümde yapılan benzetimler sonucunda optimum sonuç elde edebilmek için hangi Genetik Algoritma parametrelerinin kullanılması gerektiği belirlenmişti. Bu bölümdeki benzetimlerde ise, bu parametreler kullanılarak Genetik Algoritma ile Dijkstra algoritmasının farklı büyüklüğe sahip binalardaki çözüm süreleri karşılaştırılacaktır.

Benzetim 7: Düğüm Noktası Sayısındaki Artışın Genetik Algoritma ile Dijkstra Algoritmasının Çözüm Süreleri Arasındaki Farka Etkisi

Yöntem:

Genetik algoritma ve Dijkstra algoritması sırayla kullanılarak iki nokta arasındaki en kısa yolun bulunması işlemi, farklı yükseklikteki binalar için 100'er kez tekrarlanmıştır ve elde edilen çözüm sürelerinin ortalaması alınmıştır.

Sabit Tutulan Parametreler:

Popülasyon Büyüklüğü: 20

Seçim yöntemi: Turnuva seçimi

Çaprazlama Oranı: %100

Mutasyona Uğrayan Gen Sayısı: 0

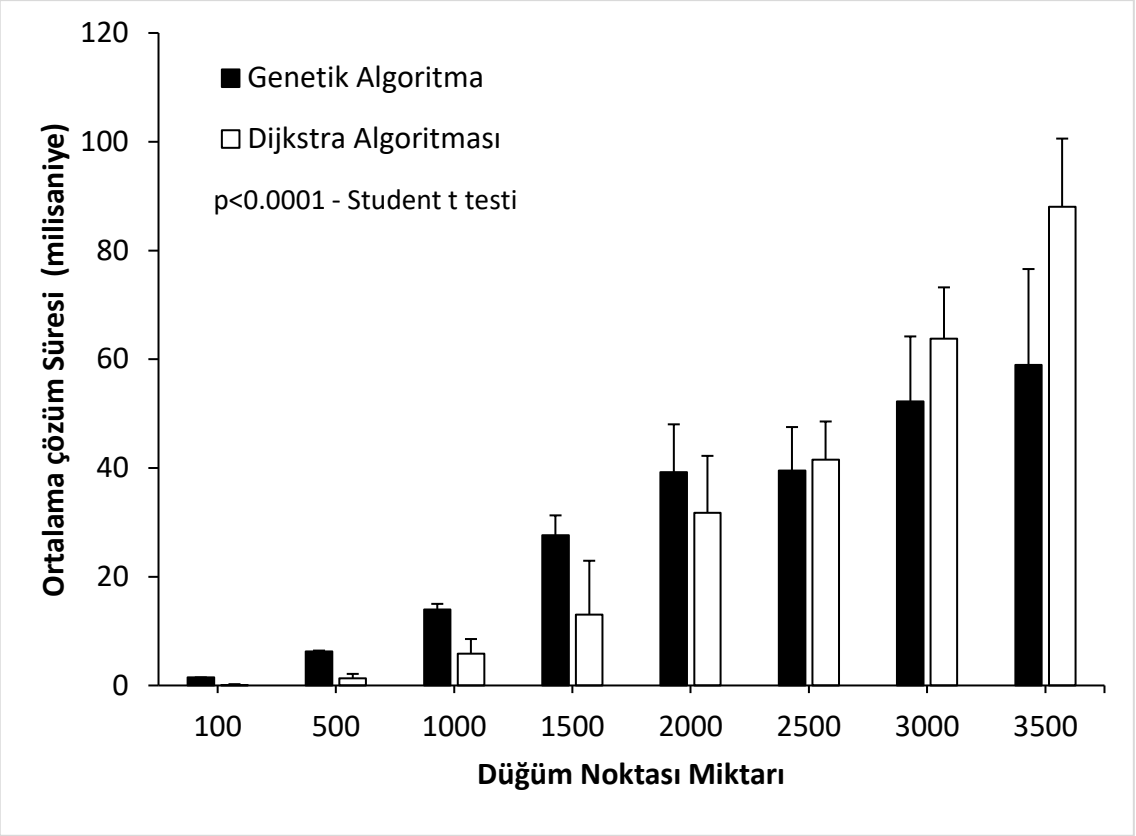
Mutasyona Uğrayan Kromozom Sayısı: 0

Sonlandırma kriteri: son 5 nesil boyunca bir gelişme yok ise sonlandır

Benzetim Sonuçlarının Değerlendirilmesi:

Yapılan benzetimlerin sonuçlarının gösterildiği grafikte (Şekil 4.13) de görüldüğü üzere, düğüm noktası miktarının az olduğu durumlarda Dijkstra algoritmasının Genetik algoritmaya göre çok daha kısa çözüm süresine sahip olduğu, ancak düğüm

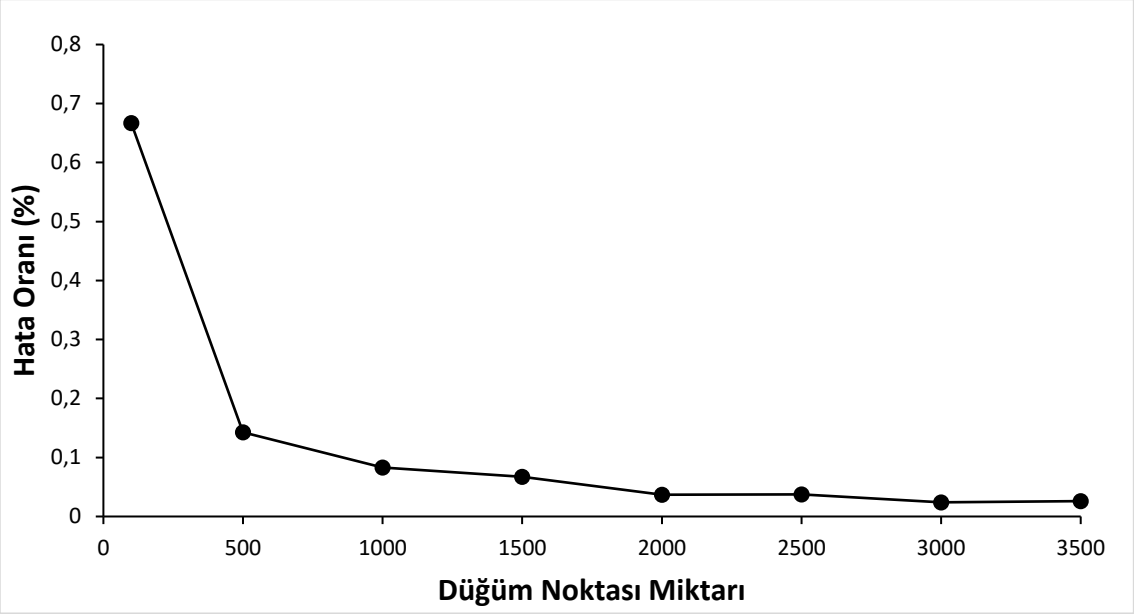
noktası miktarı arttırıldıkça, yani problemin büyüklüğü arttıkça; Dijkstra algoritmasının çözüm süresi ile Genetik algoritmanın çözüm süresi arasındaki farkın giderek azaldığı ve 2500 adet ve daha fazla düğüm noktasına sahip binalar için Genetik algoritmanın Dijkstra algoritmasına göre daha hızlı çözüm bulabildiği gözlemlenmiştir. Bu çalışmada kat planının çok basit olmasından dolayı kat başına 10 düğüm noktası düşmekte ve bu da 250 katlı binaya eşdeğer olmaktadır. Ancak günümüzdeki büyük binaların kat planları incelendiğinde çok daha karmaşık oldukları görülmektedir. Örneğin kat başına 50 düğüm noktası içeren 50 katlı yüksek bir bina 2500 düğüm noktasına sahip olacaktır. 50 katlı bir bina ise günümüzde birçok örneği bulunan gerçekçi bir bina örneği teşkil etmektedir.



Şekil 4.13 Artan düğüm noktası miktarına göre Genetik Algoritma ile Dijkstra algoritmasının çözüm sürelerinin karşılaştırılması

Bu sonuçlar, popülasyon miktarının 20 olarak seçilmesi sonucu elde edilmiştir. Buna rağmen hata oranının benzetim 1'deki hata oranlarına göre oldukça düşük olduğu görülmüştür. Bu durumun, düğüm noktası sayısının yüksek olmasından

kaynaklandığı düşünülerek düğüm noktası sayısına göre hata oranının nasıl değiştiğini incelemek için bir benzetim daha yapılarak, Şekil 4.14'teki grafik oluşturulmuştur.



Şekil 4.14 Artan düğüm noktası miktarına göre hata oranının değişimi

Şekil 4.14'teki grafikte de görüldüğü gibi, az miktarda düğüm noktasına sahip problemlerde popülasyon miktarının 20 olarak seçilmesi yüksek hata oranlarına sebep olmakta iken, yeterince çok düğüm noktasına sahip problemlerde popülasyon miktarının bu kadar düşük tutulması, o kadar yüksek hata oranlarına sebep olmamaktadır. Eğer bu durum olmasaydı, hata oranını düşürmek için popülasyon büyüklüğünün oldukça yükseltilmesi gerekecekti ve bu da aşırı bir yavaşlamaya sebep olacaktı. Bunun sonucunda da Genetik Algoritmanın Dijkstra algoritmasına göre daha hızlı hale gelmesi için gereken düğüm noktası miktarı çok daha yüksek olacaktır.

5
SONUÇ VE ÖNERİLER

Sonuç olarak, bu çalışmada bir akıllı bina tahliye sisteminde yapay zeka yöntemlerinden biri olan Genetik algoritmanın kullanılabilir olduğu ve nasıl tasarlanabileceği ve uygulanabileceği gösterilmiş, Genetik Algoritma parametrelerindeki değişimlerin ve problemin özelliklerinin çözüm süresi ve hata oranı üzerindeki etkileri incelenmiş ve en bilinen klasik en kısa yol bulma algoritması olan Dijkstra algoritması ile performansı karşılaştırılarak detaylı sonuçları ortaya koyulmuştur.

Bu sonuçlar göstermektedir ki, düğüm noktası miktarının az olduğu küçük ve karmaşık olmayan binalarda kullanılmak üzere geliştirilecek akıllı bina tahliye sistemleri için Dijkstra algoritmasının kullanılması çok daha uygun görünmekte iken; bina büyüklüğü ve karmaşıklığı arttıkça (düğüm noktası sayısı arttıkça), Genetik Algoritma, Dijkstra algoritmasına göre daha hızlı çözüm sağlamaya başlayacağı için tercih edilebilir bir hale gelecektir. Ancak burada hata oranının kesinlikle göz önünde bulundurulması, hata oranından kaynaklı mesafe artışının, hızlı çözüm süresi avantajını anlamsız kılacak derecede yüksek olmadığından emin olunması gerekmektedir. Çünkü Genetik algoritma çok daha hızlı çözüm bulabiliyor olsa da hata oranı sebebiyle rota mesafesini uzatacağı, bu durum da tahliye süresini arttıracağı için her problem için uygun bir seçenek olmayabilir. Veya Genetik Algoritmanın o probleme uygun tasarlanamaması nedeniyle yüksek hata oranları görülüyor olabilir.

Bu nedenle, bir akıllı bina tahliye sistemi geliştirilmek istendiğinde, bahsedilen tüm benzetim ve hesaplamalar yapılarak Dijkstra algoritmasının mı yoksa Genetik algoritmanın mı daha uygun bir seçim olacağına bu sonuçlar değerlendirilerek karar verilmesi önerilmektedir.

Geliştirilen bu sistemin gerçek bir bina üzerinde uygulanması için yenilikçi ve kolaylık sağlayacak bir yöntem olarak, bina bilgi sistemleri (BIS) ile entegrasyonunun sağlanarak tüm sensör verilerini oradan alması, tabelaları da BIS üzerinden kontrol edebilecek hale getirilmesi önerilmektedir.

Ayrıca binanın dijital ikizi üzerinde sanal bir sensör ve tabela ağı oluşturularak, bu sensörlerin gerçek bir acil durumu simule edecek pseudo random veriler üretmesinin sağlanması, bu sayede sistemin dinamik bir durumda davranışının test edilmesi de faydalı bir çalışma olacaktır.

KAYNAKÇA

[1] Kobes, M., Helsloot, I., De Vries, B., Post, J. G., Oberijé, N., ve Groenewegen, K. (2010). "Way finding during fire evacuation; an analysis of unannounced fire drills in a hotel at night". Building and Environment, 45(3): 537-548.

[2] Chu, L., ve Wu, S. J. (2011). "An integrated building fire evacuation system with RFID and cloud computing". Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 17-20.

[3] Atila, U. Karas, I. Turan, M. ve Rahman, A., (2013). "Design of an intelligent individual evacuation model for high rise building fires based on neural network within the scope of 3d GIS" , ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, 13-24.

[4] Atila, U. Ortakci, Y. Ozacar, K. Demiral, E. ve Karas, I.R.J.I.I.J.o.G.-I., (2018). "Smartescape: A mobile smart individual fire evacuation system based on 3d spatial model", ISPRS International Journal of Geo-Information, 7(6): 7-223.

[5] Gokceli, S. Zhmurov, N. Kurt, G.K. ve Ors, B., (2017). "IoT in action: Design and implementation of a building evacuation service", Journal of Computer Networks and Communications, 2017: 8595404.

[6] Chou, J.-S. Cheng, M.-Y. Hsieh, Y.-M. Yang, I.-T. ve Hsu, H.-T.J.A.i.C., (2019). "Optimal path planning in real time for dynamic building fire rescue operations using wireless sensors and visual guidance",Automation in Construction, 99: 1-17.

[7] Dijkstra, E.W., (1959). "A note on two problems in connexion with graphs", Numerische mathematik, 1: 269-271.

[8] Samah, K. Hussin, B. ve Basari, A.S.H., (2015). "Modification of Dijkstra's algorithm for safest and shortest path during emergency evacuation", Applied Mathematical Sciences, 9: 1531-1541.

[9] Bellman, R., (1958). "On a routing problem", Quarterly of applied mathematics, 16: 87-90.

[10] Ford Jr, L.R., (1956). "Network flow theory". Rand Corp Santa Monica Ca.

[11] Nilsson, N.J., (1984). "Shakey the robot". SRI International Menlo Park Ca.

[12] Floyd, R.W., (1962). "Algorithm 97: shortest path", Communications of the ACM, 5: 345.

[13] Warshall, S., (1962). "A theorem on boolean matrices", Journal of the ACM (JACM), 9: 11-12.

[14] Johnson, D.B.J.J.o.t.A., (1977). "Efficient algorithms for shortest paths in sparse networks", Journal of the ACM (JACM), 24: 1-13.

[15] Ali, M.K.M. ve Kamoun, F., (1993). "Neural networks for shortest path computation and routing in computer networks", IEEE transactions on neural networks, 4: 941-954.

[16] Gen, M. Cheng, R. ve Wang, D., (1997). "Genetic algorithms for solving shortest path problems", IEEE International Conference on Evolutionary Computation (ICEC'97), 401-406.

[17] Munetomo, M. Takai, Y. ve Sato, Y., (1998). "A migration scheme for the genetic adaptive routing algorithm", IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), Vol. 3, 2774-2779.

[18] Inagaki, J. Haseyama, M. ve Kitajima, H., (1999). "A genetic algorithm for determining multiple routes and its applications", 1999 IEEE International Symposium on Circuits and Systems (ISCAS) Vol. 6, 137-140.

[19] Ahn, C.W. ve Ramakrishna, R.S., (2002). "A genetic algorithm for shortest path routing problem and the sizing of populations", IEEE transactions on evolutionary computation, 6: 566-579.

[20] Mohemmed, A.W. Sahoo, N.C. ve Geok, T.K., (2008). "Solving shortest path problem using particle swarm optimization", Applied Soft Computing, 8: 1643-1653.

[21] Ebrahimnejad, A. Tavana, M. ve Alrezaamiri, H., (2016). "A novel artificial bee colony algorithm for shortest path problems with fuzzy arc weights", Measurement, 93: 48-56.

[22] Holland, J.H., (1992). "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence", MIT press.

[23] Fang, Z. Lo, S. ve Lu, J.J.F.S.J., (2003). "On the relationship between crowd density and movement velocity", Fire Safety Journal, 38(3): 271-283.

[24] Nelson, H.E. ve Mowrer, F.W.J.N., Quincy, (2002). "Emergency movement"., SFPE handbook of fire protection engineering, 367-380.

[25] Predtechenskii, V. ve Milinskii, A.J.A., New Delhi, (1969). "Planning for Foot Traffic Flow in Buildings. 1978", National Bureau of Standards, US Department of Commerce, and the National Science Foundation, Washington, DC.

[26] Thompson, P.A. ve Marchant, E.W.J.F.s.j., (1995). "A computer model for the evacuation of large building populations", Fire safety journal, 24(2): 131-148.

[27] Park, J. ve Arteaga, C., (2019). "Human responses of emergency evacuation using agent-based modeling". In International Conference on Smart Cities, Seoul, Korea, 1-6.

Tezden Üretilmiş Yayınlar

Konferans Bildirisi

1. Sancak E.F. ve Yılmaz C., (2020) "Genetik Algoritma Kullanarak bir Akıllı Bina Tahliye Sistemi Tasarımı", 2. Uluslararası Akademik Araştırmalar Kongresi (ICAR) Tam Metin Kitabı, 253:261.