T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

OPTİMİZASYON YÖNTEMLERİNİN SÜRÜ İNSANSIZ HAVA ARAÇLARI (İHA) ÜZERİNDE SİMÜLASYONU

Sefa ARBAÇ

DANIŞMAN
Dr. Öğr. Üyesi TARIK VELİ MUMCU

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

İSTANBUL-2022

Bu çalışma, 7.01.2022 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı, Elektrik-Elektronik Mühendisliği Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Dr. Öğr. Üyesi Tarık Veli MUMCU(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Aysel ERSOY
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Selin ÖZÇIRA
Yıldız Teknik Üniversitesi
Elektrik Elektronik Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

Bu tez, İstanbul Üniversitesi-Cerrahpaşa Bilimsel Araştırma Projeleri Yürütücü Sekreterliğinin ..... numaralı projesi ile desteklenmiştir.
Bu tez, ... numaralı ..... projesi ile desteklenmiştir.

ÖNSÖZ

Yüksek lisans eğitimim boyunca danışmanlığımı yürüten, yardımlarını, bilgi birikimini ve kıymetli zamanını hiçbir şekilde esirgemeyerek bu tez çalışmasının gerçekleştirilmesinde destekçim olan, yaptığı bilimsel eleştiriler ve yönlendirmelerle ufkumu açan değerli hocam Dr. Öğr. Üyesi Tarık Veli Mumcu' ya en kalbi duygularımla teşekkür ederim.

Hayatımın her aşamasında, karşılaştığım tüm zorluklarda yanımda olan ve beni bu günlere getiren sevgili aileme minnettarım.

Ocak 2022 Sefa ARBAÇ

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL LİSTESİ ..... **vii**
TABLO LİSTESİ ..... **ix**
SİMGE VE KISALTMA LİSTESİ ..... **x**
ÖZET ..... **xi**
SUMMARY ..... **xii**
1. GİRİŞ **1**
2. GENEL KISIMLAR **3**
3. MALZEME VE YÖNTEM **12**
3.1 METASEZGİSEL YÖNTEMLER 12
3.2 GENETİK ALGORİTMASI 14
3.2.1 Kodlama 15
3.2.2 Seçilim 18
3.2.3 Çaprazlama 20
3.2.3 Mutasyon 21
3.2.4 Genetik Algoritmasının Matematiksel Modeli 24
3.2.5 Uygunluk Değerlerinin Ölçeklendirilmesi 26
3.3 ÇOK AMAÇLI(MULTIOBJECTIVE) GENETİK ALGORİTMASI 27
3.3.1 Güç ve Mesafe Uygunluk Değerlerinin Hesaplanması 30
4. BULGULAR **32**
4.1 GAZEBO 32
4.2 ROBOTİK İŞLETİM SİSTEMİ 34
4.3 SİMÜLASYON ÇALIŞMASI 35
4.3.1 Sürü İHA Planlamasında Uygulanan Yöntemler 39
4.3.1.1 Mutasyon Fazında Uygulanan Takas Yönteminin Simülasyon Üzerindeki Etkisi 40
4.3.1.2 Mutasyon Fazında Uygulanan Karıştırma Yönteminin Simülasyon Üzerindeki Etkisi 41
4.3.1.3 Mutasyon Olasılık Parametresinin Simülasyon Üzerindeki Etkisi 42

4.3.1.4 Çaprazlama Fazında Uygulanan Tek Çözüm Yönteminin Simülasyon Üzerindeki Etkisi 44
4.3.1.5 Çaprazlama Fazında Uygulanan Çift Çözüm Yönteminin Simülasyon Üzerindeki Etkisi 45
4.4 UYGULANAN YÖNTEMLERİN KAŞILAŞTIRILMASI 47
5. TARTIŞMA VE SONUÇ 50
KAYNAKLAR.....52
EKLER..... Error! Bookmark not defined.
ÖZGEÇMİŞ 55

ŞEKİL LİSTESİ

Sayfa No

Şekil 1.1: İHA çeşitleri [1] 1

Şekil 2.1: Sürü İHA tasarım mimarisi 5

Şekil 2.2: DDDAS mekanizması 6

Şekil 2.3: DDDAS yazılımı ve MultiUAV2 etkileşimi. 7

Şekil 2.4: Sürü İHA test yatağının kavramsal mimarisi. 8

Şekil 2.5: Sürü İHA sınıflandırma örneği. 10

Şekil 3.1: Metasezgisel algoritmaların sınıflandırılması [23]. 13

Şekil 3.2: GA' da kullanılan operatörler ve farklı yöntemler [23]. 15

Şekil 3.3: Kromozomların ağaç yapısı ile kodlanması. 17

Şekil 3.4: Rulet Çarkı örneği [20]. 19

Şekil 3.5: Single point çaprazlama örneği. 21

Şekil 3.6: Displacement çaprazlama örneği. 23

Şekil 3.7: Simple inversion çaprazlama örneği. 23

Şekil 3.8: Scramble çaprazlama örneği. 24

Şekil 3.9: Karar ve amaç uzayları [29] 29

Şekil 3.10: Üç çözüme sahip amaç popülasyonu. 29

Şekil 4.1: Simülasyon mimarisi. 32

Şekil 4.2: Gazebo genel mimari [22]. 33

Şekil 4.3: ROS' u oluşturan bileşenler [34]. 34

Şekil 4.4: Sürü İHA' nın Gazebo simülasyonu. 36

Şekil 4.5: 1 Numaralı görevin görüntüsü. 36

Şekil 4.6: İHA görüntüsü. 37

Şekil 4.7: Takas yöntemi, Güç-Mesafe grafiği. 41

Şekil 4.8: Karıştırma Yöntemi, Güç-Mesafe grafiği. 42

Şekil 4.9: Tek Çözüm, Güç-Mesafe grafiği. 45

Şekil 4.10: Mutasyon, Takas yöntemi. 48

Şekil 4.11: Mutasyon, Karıştırma yöntemi. 48

Şekil 4.12: Çaprazlama, tek çözüm. 49

Şekil 4.13: Çaprazlama, çift çözüm 49

TABLO LİSTESİ

Sayfa No

Tablo 3.1: Kromozomların ikili kodlaması. 16

Tablo 3.2: Kromozomların permütasyon kodlaması. 16

Tablo 3.3: Kromozomların değer kodlaması. 17

Tablo 3.4: Sürü İHA simülasyonunda kullanılan kodlama. 18

Tablo 3.5: Örnek dört elemanlı popülasyon derece ve Rulet Çarkı değerleri 20

Tablo 4.1: Node ve Topic isimleri 38

Tablo 4.2: Takas yöntemi sonuçları. 40

Tablo 4.3: Karıştırma yöntemi sonuçları. 42

Tablo 4.4: Mutasyon olasılığı 0.3 43

Tablo 4.5: Mutasyon olasılığı 0.5 43

Tablo 4.6: Mutasyon olasılığı 0.7 43

Tablo 4.7: Tek çözüm yöntemi sonuçları 44

Tablo 4.8: Çift çözüm yöntemi sonuçları 46

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar **Açıklama**

HAPF : Hybrid Artificial Potential Field
ACO : Ant Colony Optimization
DDDAS : Dynamic Data Driven Applications System
ISOA : Intelligent Self-organized algorithm
MTWPS : Modified Two-Part Wolf Pack Search
ROS : Robot Operating System
TA : Tabu Search
MA : Microcanonical Annealing
GLS : Guided Local Search
GA : Genetic Algorithm
PSO : Particle Swarm Optimization
ACO : Ant Colony Optimization
MOGA : Multi Objective Genetic Algoritmassı
VTOL : Vertical Take-Off and Landing

ÖZET

YÜKSEK LİSANS TEZİ

OPTİMİZASYON YÖNTEMLERİNİN SÜRÜ İNSANSIZ HAVA ARAÇLARI (İHA) ÜZERİNDE SİMULASYONU

Sefa ARBAÇ

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman : Dr. Öğr. Üyesi Tarık Veli MUMCU

Havacılık alanında gün geçtikçe yerine getirilmesi gereken görevlerin karmaşıklığı artmaktadır. Bu karmaşık görevlerin tek bir hava aracının başarılı bir şekilde yerine getirmesi imkansızdır. Bu zorlu görevler için tek başına hava aracı kabiliyetlerini iyileştirme gereksinimlerinin yanında ortak bir amaç doğrultusunda, ahenk içerisinde hareket eden birden fazla hava aracının bulunduğu sistemlerin geliştirilmesi zorunlu hale gelmiştir. Birden fazla hava aracının bulunduğu sistemler "sürü İHA" olarak adlandırılmıştır. Sürü İHA sistemlerinde kontrol edilmesi gereken hava aracı sayısı, görev sayısı, görev özelliklerinin farklı olması sistemdeki farklılığı büyük oranda arttırmaktadır. Bunun sonucu olarak sürü İHA sistemlerindeki bu karmaşıklığı basitleştirmek ve en iyi çözümlere ulaşmak için optimizasyon algoritmalarına ihtiyaç duyulur.

Bu tez çalışması kapsamında, sürü İHA sistemlerinde görev planlama ve görev atama katmanlarında metasezgisel yöntemlerden Genetik Algoritmasının farklı yöntemleri uygulanmış ve sonuçları karşılaştırılmıştır. Sürü İHA sistemlerinin farklı çalışma alanlarında gerekli olan algoritmaların uygulanabileceği bir simülasyon ortamı önerilmiştir.

Ocak 2022, 71 sayfa.

Anahtar kelimeler: Sürü İnsansız Hava Aracı, Optimizasyon, Planlama, Genetik Algoritması

SUMMARY

M.Sc. THESIS

OPTIMIZATION METHODS SIMULATION ON SWARM UNMANNED AIR VEHICLES (UAV)

Sefa ARBAÇ

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Electrical and Electronic Engineering

Supervisor : Assist. Prof. Dr. Tarık Veli MUMCU

The complexity of the tasks to be fulfilled in the field of aviation is increasing nowadays. It is impossible for a single drone to successfully perform these complex missions. In addition to the requirements to improve aircraft capabilities for these challenging missions, it has become mandatory to develop systems with more than one drone moving in harmony for a common purpose. Systems with more than one drone are called "swarm UAVs". The number of drone to be controlled in swarm UAV systems, missions, and the difference in mission characteristics greatly increase the complexity of the system. As a result, optimization algorithms are needed to simplify this complexity in swarm UAV systems and to reach the best solutions.

Within the scope of this thesis, different methods of Genetic Algorithm, one of the metaheuristic methods, were applied in mission planning and task assignment layers in swarm UAV systems. The results of the various methods of Genetic Algorithm were compared. A simulation environment has been proposed in order to test the algorithms that can be applied for each working area of the swarm UAV systems.

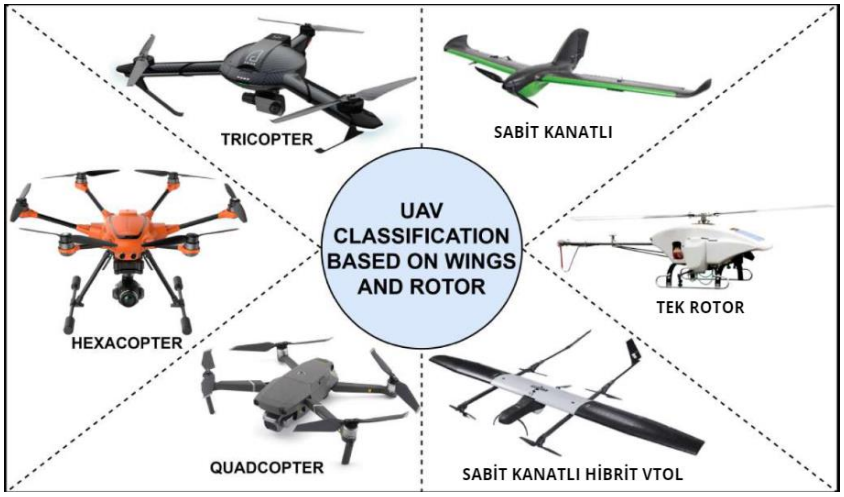
January 2022, 71 pages.

Keywords: Swarm Unmanned Aerial Vehicle, Optimization, Planning, Genetic Algorithm

1. GİRİŞ

İnsansız hava araçları (İHA) başta askeri amaçlar olmak üzere güvenlik, gözetleme, arama ve kurtarma, afet yönetimi gibi birçok alanda kullanılmaktadır. Son yapılan akademik yayınlarda, İHA' ların COVID-19 pandemisinin kontrol altında tutulması için toplanma yerlerinin gözetimi ve halka açık yerlerin dezenfekte edilmesi gibi görevlerde kullanıldığı belirtilmiştir [1]. İnsanlı havacılıkta, uçuş sırasında kritik bir hata meydana geldiğinde yaralanma veya ölüm riski vardır. İnsansız havacılık ile bu endişeler hafifletilir [1]. Bunun yanında pilotun yetiştirilmesi, yakıt ve bakım maliyetlerinden dolayı insanlı havacılık pahalıdır. Çeşitli kullanım alanlarına sahip insansız hava araçları üzerinde birçok çalışma literatüre girmektedir.

İnsansız hava araçları, ağırlıkları, yükseklik menzilleri, kullanıldığı uygulama, kanat veya rotor sayısı bakış açılarıyla sınıflandırılabilir. Şekil 1.1' de insansız hava aracı çeşitlerinin kanat ve rotor sayılarına göre sınıflandırılması görülmektedir. Sabit kanatlı İHA' lar, sabit kanatlı uçak tasarımını baz alırken, tek rotor tipi İHA' lar helikopter tasarımı taklit edilerek geliştirilmiştir. Sabit kanatlı hibrit VTOL, sabit kanatlı İHA' nın avantajlarına sahip olmasının yanında dikey olarak kalkış ve iniş yapabilen hava araçlarıdır. Tricopter, Hexacopter ve Quadcopter' ler çok rotorlu insansız hava araçları örnekleridir [1]. Bu tez çalışmasında simülasyon çalışması, birden fazla Quadcopter kullanılarak tasarlanmıştır.



Şekil 1.1: İHA çeşitleri [1].

Gün geçtikçe karmaşıklaşan görev gereksinimlerinden dolayı tek bir İHA’ nın bu görevleri icra kabiliyeti sınırlıdır. Bundan dolayı sürü İHA’ lar ile ilgili çalışmalar ve faaliyetler zorunlu hale gelmiştir. Sürü İHA' lar bir tür sürü robot sistemidir ve bu alanda yapılan araştırmalar, insansız ve otomatik teknolojilerin geliştirilmesinde önemli bir yönlendirici olmuştur [3]. Sürü İHA sistemleri, içerisinde kaynak kullanımı, hareket kabiliyeti, mesafe, fiziksel engeller gibi birçok kısıtlama bulunduran yapılardır. Bu tür sistemlerde düzenli bir modelin olmayışı, problemlerin tek bir formüle indirgenmesini engellemektedir. Bu nedenle Sürü İHA sistem tasarımındaki karmaşıklıkları gidermek için, genellikle balık, kuş, karınca gibi doğa canlılarının hareketleri taklit edilmiş ve problemlerin çözümünde sezgisel yöntemler kullanılmıştır. Sezgisel yöntemler, problemlere karşı kesin bir çözüm üretmek yerine, tahminlerde bulunarak yaklaşık çözüme veya en iyi çözüme ulaşmak için kullanılan yöntemlerdir.

Sürü davranışı her yerde kendini gösteren doğal bir kavramdır. Memelilerde, kuşlarda, balıklarda ve bakterilerde karmaşık sürü davranışları mevcuttur. Biyolojik sürüler, koordineli ve düzenli hareket etmekle birlikte, dış tehlikelere karşı hızlı ve doğru bir şekilde tepki verir. Böylece doğa sürüleri kendi kendine örgütlenme, iş birliği ve çevresel uyum özelliklerini barındırır [3].

Sürü İHA tasarım katmanları, görev planlama, görev atama, yol noktaları oluşturma, her bir hava aracı için kontrol mekanizması oluşturma, sürü içerisindeki hava araçlarının haberleşmesi ve uygulama katmanı olarak sınıflandırılabilir. Bu tez çalışması kapsamında, sürü İHA sistemlerinde görev planlama ve görev atama katmanlarında metasezgisel yöntemlerden biri olan Genetik Algoritmasının (GA) çeşitli yöntemleri uygulanmış ve sonuçları karşılaştırılmıştır. Sürü İHA sistemlerinde her çalışma katmanı için gerekli olan algoritmaların uygulanabileceği bir simülasyon ortamı önerilmiştir.

Tez çalışmasının ikinci kısmı, sürü robot ve sürü İHA alanlarında yapılan çalışmalar ele alınmıştır. Bu tez çalışmasının ana konusu olan görev planlama ve görev atama çalışmalarında bugüne kadar kullanılan yöntemler verilmiştir. Tez çalışmasının üçüncü kısmında, Genetik Algoritması ve Çok Amaçlı GA’ nın genel prensipleri ve bu tez çalışmasında uygulanan yöntemler verilmiştir. Dördüncü kısımda, simülasyon ortamı için gerekli olan araçlar hakkında bilgi ve uygulanan farklı yöntemlerin sonuçları verilmiştir. Tezin beşinci bölümünde, yapılan çalışmanın özeti verilmiş ve bu alanda ihtiyaç duyulan çalışmalar ile ilgili içgörüler paylaşılmıştır.

2. GENEL KISIMLAR

Bu bölümde, literatürde sürü teknolojileri ve çalışmamızda ele aldığımız sürü İHA alanlarında dikkat çeken bazı çalışmalar incelenmiştir.

Sürü veya filo kavramı, bakteri, karınca veya arı kolonileri gibi biyolojik olaylardan esinlenilerek robotik alanına uyarlanmıştır. Bu alanda yapılan çalışmalar, biyolojik canlılarda görülen desen oluşturma, taşıma gibi belli görevleri yerine getirmek için toplu bir şekilde hareket etme iç güdüsünü robotik anlamda kullanmaya yönelik çalışmalardır. Sürü zekası terimi ise, diğerlerinden bağımsız, tek başına otonom hareket eden birimlerin birleşiminden ortaya çıkan kolektif davranışı ifade eder. Cao ve diğ. [4] sürü zekâsını, "akıllı olmayan her bir birimin, bir araya gelerek akıllı davranışlar sergilemesidir" diye tanımlamışlardır. Yapılan çalışmalarda sürü zekâsı teriminin robotik alanına uyarlanırken "sürü robot", İHA alanına uyarlanırken "sürü İHA" olarak adlandırıldığı görülmüştür.

Sürü robot alanında, geçmişten günümüze kadar yapılan çalışmaları sınıflandırmak için birçok akademik yayın yapılmıştır. Bu çalışmaları sınıflandırmaya yönelik ilk çalışmalar Dudek ve diğ. [5] tarafından 1993 yılında yapılmıştır. Bu çalışma, sınıflandırma işlemi sırasında sürü büyüklüğü, iletişim altyapısı, sürü içerisindeki her birimin işlem görme kapasitesi gibi kriterleri esas almıştır. Mohan ve Ponnambalam [6] 2009 yılında yaptıkları sınıflandırma çalışmasında, sürü robotları biyolojik ilham, iletişim, kontrol yaklaşımı, haritalama ve yerini belirleme, nesne taşıma gibi başlıklar altında toplamış ve her bir başlık için açık araştırma konularına değinilmiştir.

"Sürü robot" teknolojileri açısından literatürde yapılan çalışmalarda, bazı araştırmacıların karıncaların yiyecek arama davranışlarından esinlenerek, nesne bulma görevinde yer alan bir grup robotun davranışını analiz ettiği görülmüştür. Yapılan bu çalışmada basit bir parametre uyarlamasının, grubun verimliliğini artırmasına ve grup üyeleri arasında iş bölümü ile ilgili optimizasyona yol açtığı gösterilmiştir. Çalışmada her bir robot kendisi ile ilgili lokal bilgileri kullanarak çevreye uyum sağladığı bir çözüm izlenmiştir [7]. Bundan dolayı, bu çalışma, sürü robot sistemlerinde, robotların bireysel olarak eğitilmesine yönelik çalışmalara örnek teşkil etmektedir.

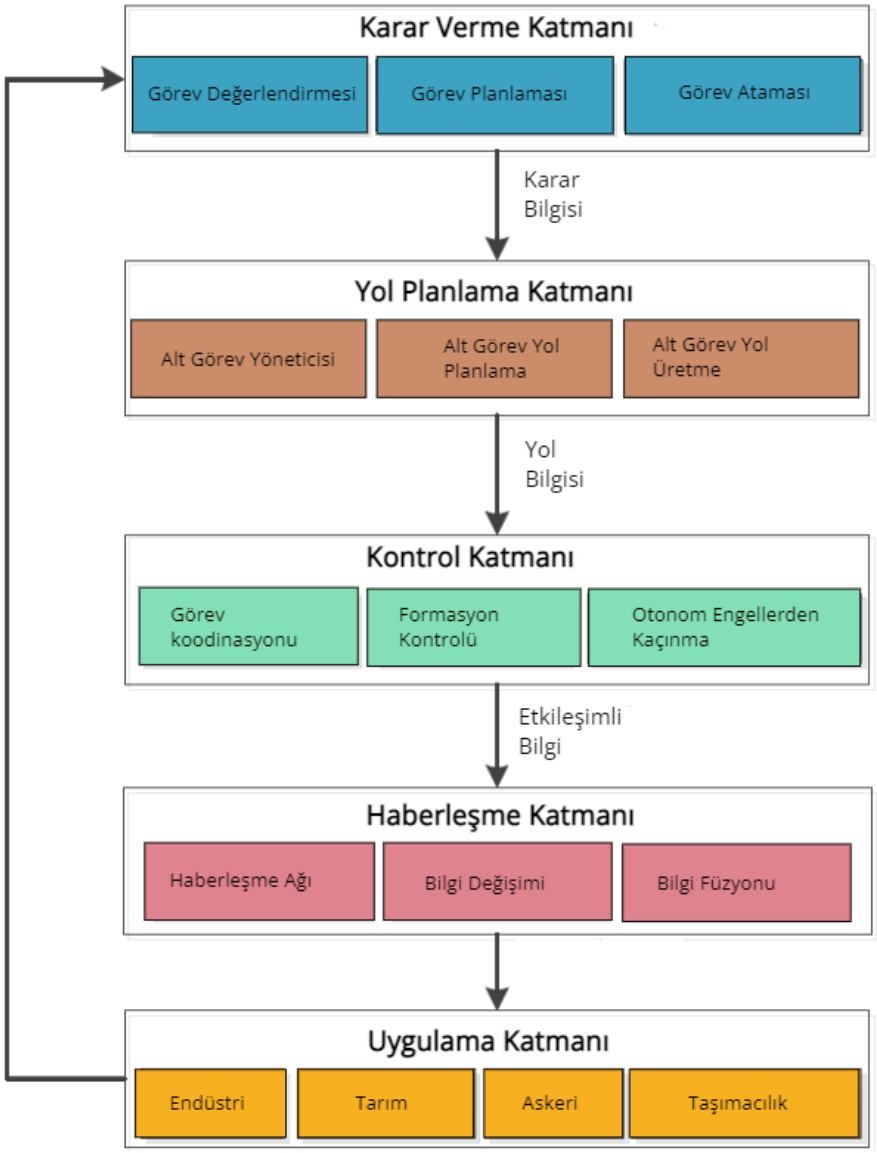
Kalla ve diğ. [8] tarafından yapılan çalışmada, sürü teknolojisini, bulunduğumuz yerde temizlik ve toplama gibi amaçlarla, ev eşyalarının bir yerden başka bir yere taşınması görevinde kullanılmak üzere, tüketici elektroniğine dahil etmeyi amaçlamışlardır. Çalışmada, sürü robot teknolojisini modellemek üzere iki adet robot kullanılmıştır. Robotların görevlerini icraları esnasında, robotlardan birinde arıza meydana gelmesi durumunda, en yakın robotun bu görevi yerine getirmesi sağlanmıştır.

Khaluf ve diğ. [9] tarafından yapılan çalışmada, dinamik olarak değişen bir ortamda mobil nesneleri takip etmek için robotik sürü algoritmaları kullanılmıştır. Bu çalışmada, robotlar nesnelerin hareketini toplu olarak izler ve hareketli nesne hakkında bir karar mekanizması oluşturmaya çalışır. Yapılan çalışmada nesne algılama olasılığını arttırmak için "kendi kendine dağıtım (Self-Deployment)" stratejisi önerilmiştir. Bu tür birlikte hareket etmeyi gerektiren görevlerde, robotların senkronize bir şekilde hareket etmesi gerekir. Bu çalışmada robotlar arası meydana gelebilecek haberleşme bağlantısı kesintilerinde, zaman senkronizasyonunu sağlamak için yeni bir yaklaşım sunulmuştur.

Sürü teknolojisi birçok alanda kullanıldığı gibi İHA' larda da etkin bir şekilde kullanılır. Ancak sürü içerisindeki hava aracı sayısının artması tasarlanacak sürü İHA sistemini karmaşık hale getirmektedir. Araştırmalar, sürü İHA sistemlerinin "hiyerarşik" bir yapıda tasarlanmasının bu ihtiyaca cevap oluşturduğunu göstermektedir [10]. Jovan ve diğ. [11] yaptıkları çalışmada İHA sürülerini kontrol etmek için hiyerarşik bir mimari önermişlerdir. Bu çalışmada, CoMPACT adı verilen, görev planlama aşamasından, görev yürütme aşamasına kadar altı koordineli katmandan oluşan bir mimari üzerinde çalışılmıştır. Jovan ve diğ. [12] yaptıkları diğer bir çalışmada başka bir mimari yaklaşım kullanılmıştır. Bu mimari yaklaşımda, sistem tasarım problemi beş katmana ayrılmıştır.

Şekil 2.1, mimari yapıda yer alan katmanların birbirleri ile olan ilişkilerini göstermektedir. Bu katmanlardan ilki karar verme katmanıdır. Karar verme katmanı, görevleri, konum, kısıtlama, gerekli sensörler gibi veriler ışığında bir değerlendirmeye tabi tutar. Daha sonrasında görevlerin planlamasını yaptıktan sonra hangi görevin hangi İHA tarafından yapılacağı ile ilgili, yol planlama katmanının kullanabileceği bilgiler üretir. Yol planlama katmanı, karar verme katmanından kendisine gönderilen "karar bilgisini" kullanarak her bir İHA için yol bilgisi üretmekten sorumludur. Kontrol katmanı, yol planlama katmanından her bir İHA için izlenecek yol bilgisini alır. İHA' lar arası görev koordinasyonunu, engellerden ve çarpışmalardan

kaçınarak sağlar. İletişim veya haberleşme katmanı, İHA' lar arası bilgi alışverişini kontrol eden katmandır. Uygulama katmanı, her uygulama senaryosunun ürettiği kendine özgü bilgilerle, karar verme katmanını besler [10]. Bu mimari yapısının kullanılması, katmanlar arası bir soyutlama getirir. Böylelikle her bir katman içerisinde diğer katmanları etkilemeyecek şekilde değişiklikler yapılabilir. Bu yapı araştırmacılar için büyük bir esneklik sağlar ve sürü İHA sistemlerinin tasarım süreçlerini oldukça basitleştirir.

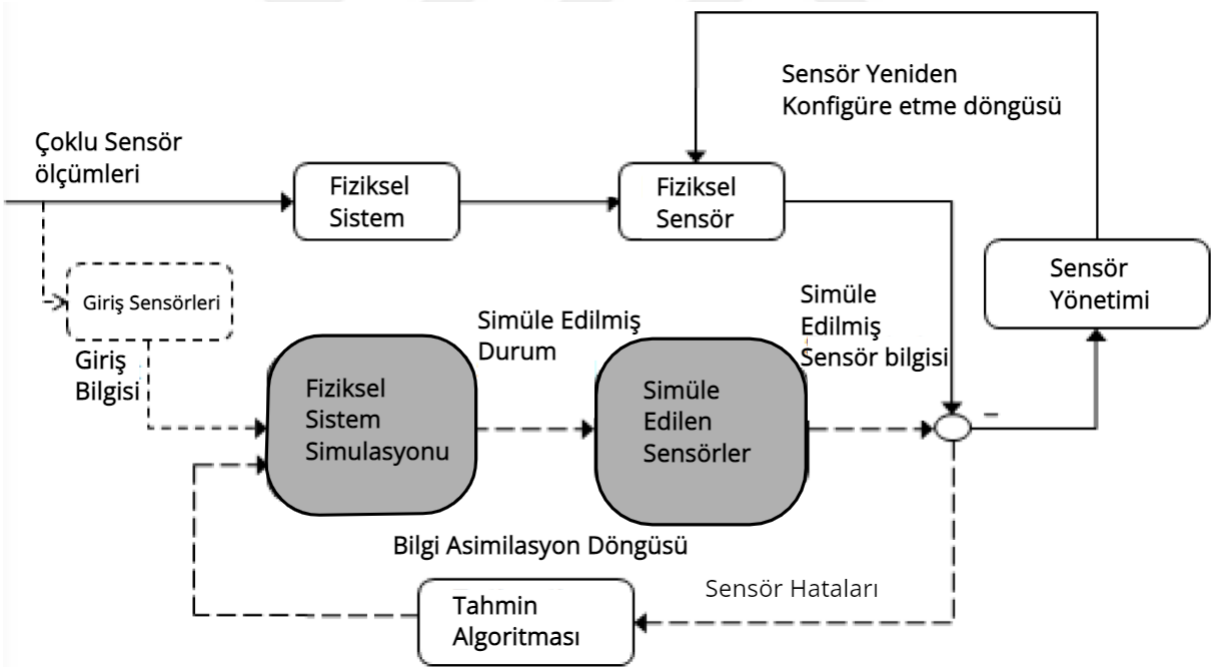


Şekil 2.1: Sürü İHA tasarım mimarisi.

Tez çalışması kapsamında, sürü İHA sistemleri oluşturulurken kullanılan katmanlardan ilki yani karar verme katmanında çalışmalar yapılmıştır. Literatürde karar verme katmanını

ilgilendiren birçok çalışma mevcuttur. Bu çalışmalar çoğunlukla Dinamik Veriye Dayalı Uygulama Sistemleri (DDDAS) yöntemlerini baz alan çalışmalardır.

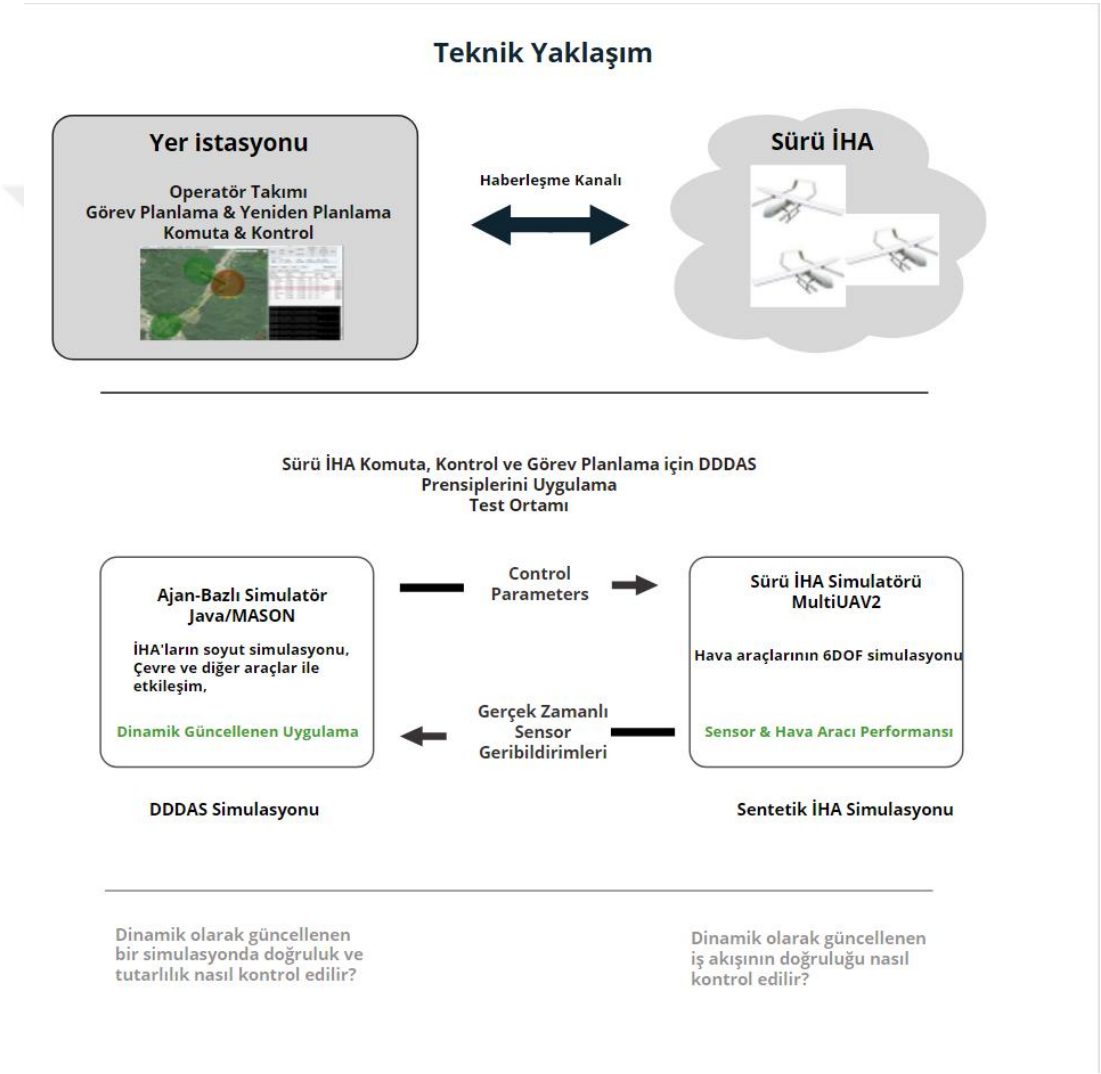
DDDAS, karmaşık sistemlerin model simülasyonlarını, entegre etmeye ve devreye almaya odaklanan bir sistem tasarım çerçevesidir. DDDAS paradigması ilk olarak 2005 yılında Darema tarafından öne sürülmüştür. Şekil 2.2' de DDDAS' in çalışma mekanizmasını gösteren yapı verilmiştir. Esasen DDDAS' te sistemin yapı taşını "Bilgi Asimilasyon Döngüsü" oluşturur. Bu döngü sensör hatalarını, fiziksel sistem simülasyonuna girdi olarak verilmesini sağlar ve böylelikle fiziksel simülasyon daha yakından takip edilir. Sistemde bilgi içeriğini zenginleştirip, geliştirmek için "Sensör Yeniden Konfigüre Etme" döngüsü kullanılır. Fiziksel ortam simülasyonundan sonra gelen simülasyon adımı, toplanan verilere rehberlik eder ve fiziksel sistem simülasyonunun doğruluğunu arttırır [13].



Şekil 2.2: DDDAS mekanizması.

Madey ve diğ. [14] tarafından yapılan çalışmada, DDDAS prensipleri sürü İHA sistemlerine entegre edilmiştir. Bu çalışmada DDDAS ilkeleri, Komuta & Kontrol ve Görev Planlama / Yeniden planlama görevleri için, gelecek nesil İHA' lara uygulanmıştır. DDDAS kullanılarak,

İHA sürüsü ile ilgili aşırı yük sorunu çözülmeye çalışılmıştır. Çalışmada İHA sürüsünden alınan sensör ve uçuş durum verilerini kullanarak, sürünün gelecekteki performansını tahmin eden bir yazılım sunulmuştur. Bu yazılımı test edebilmek için MultiUAV2 simulatörü kullanılmış, sensör verilerinin, hataların (beklenmedik olayların) ve yeni gereksinimlerin kaynağı olacak bir İHA sürü uçuş simülatörü oluşturulmuştur. Çalışmada tasarlanan MultiUAV2 ile DDDAS yazılımları arasındaki haberleşme Şekil 2.3' de gösterilmiştir.

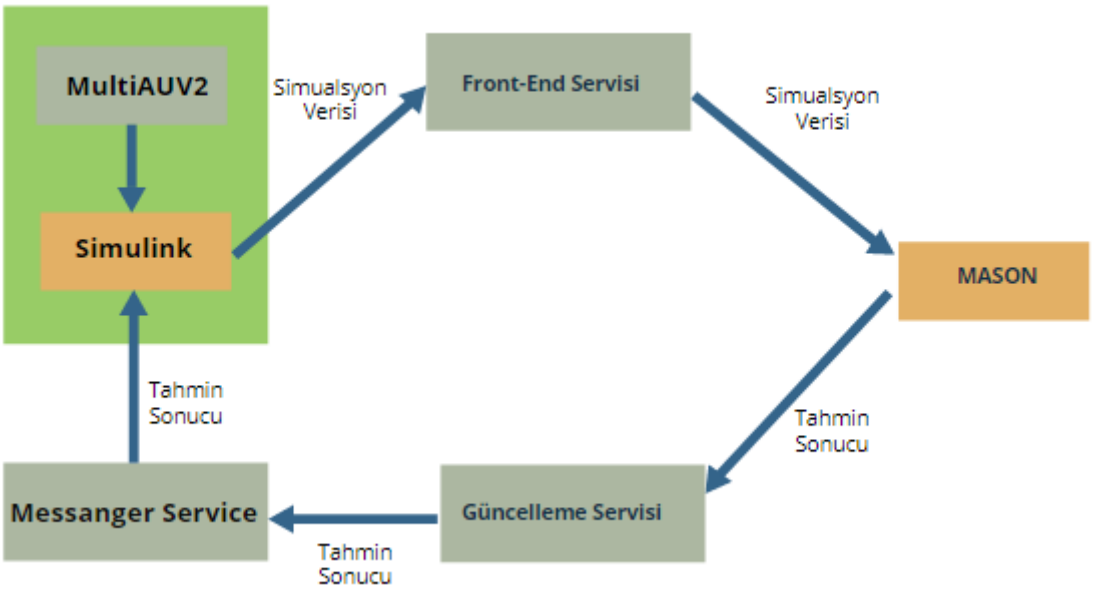


Şekil 2.3: DDDAS yazılımı ve MultiUAV2 etkileşimi.

Bunun yanında, çalışmada İHA sürü simülatörü ile DDDAS yazılımı arasındaki haberleşme için gerekli olan başka bir sistem yazılımı, hem çalışan DDDAS yazılımını dinamik olarak

güncellemek hem de İHA sürüsünü yönlendirmek amacıyla geliştirilmiştir. Bu çalışmada, DDDAS ile ilgili dört temel araştırma alanından üç alan ele alınmıştır:

- Uygulama Modelleme
- Sensör Kontrolü
- Sistem Yazılımı



Şekil 2.4: Sürü İHA test yatağının kavramsal mimarisi.

Şekil 2.4 İHA sürü simulasyonu ile DDDAS test ortamının etkileşim mimarisi göstermektedir. MultiUAV2 yazılımı, gerçek bir İHA sürüsü ve operasyon ortamını simüle ederken, Mason simülasyonu, dinamik olarak güncellenerek yeni kontrol algoritmalarını İHA sürüsüne geri gönderir. İHA sürüsünün simülasyonu sırasında, MultiUAV2 verileri Front-end web servisine gönderilir. Front-end servisi, sürü İHA davranış tahmin görevlerini yürütmek için MASON simülasyon yazılımını kullanır. Görevler tamamlandıktan sonra, MASON simülasyonu bir güncelleme servisine yeni kontrol parametreleri, yeni sürü İHA iş akışı spesifikasyonlarını veya yeni hedefleri gönderir. Servis, güncellemeleri bir mesajlaşma servisine göndererek mesajları

sıraya koyar. MultiUAV2, simülasyon parametrelerini dinamik olarak değiştirmek için mesajlaşma hizmetini periyodik olarak sorgular.

Çalışmada tahmin sonuçları, MultiUAV2 simulasyon girdileri için kullanılırken, simülasyon çıktısı, sensörleri ve gömülü oldukları cihazları yeniden yapılandırmak için kullanılmıştır.

DDDAS prensiplerinin dışında sürü İHA karar verme katmanında kullanılabilecek farklı yöntemler mevcuttur. Ziyang ve diğ. [15] tarafından yapılan çalışmada, sürü İHA' nın arama görevi için, görev planlama problemine çözüm olarak ISOA' yı (Intelligent Self-organized algorithm) önermişlerdir. ISOA, global optimizasyon problemini farklı yerel optimizasyon problemlerine bölerek, global görev planlama problemini çözmeyi amaçlar. Bu algoritma ile, kendi yerel optimizasyon problemini çözen her bir İHA' dan veriler toplanır ve ardından sürü İHA sistemi için verilecek olan kararlar bu veriler işlenerek alınır. Bu çalışmada görev planlama süreci iki kısma ayrılmıştır. Birinci kısım, İHA' nın manevra kabiliyeti, çarpışmadan kaçınma ve maksimum menzil kısıtlamaları altında yol noktaları oluşturma, diğeri ise tehditten kaçınmak için yol güzergahı belirlemedir. Bu çalışmada ilk olarak, görev planlamasını devreye almak ve yol güzergahı oluşturmak için geliştirilmiş bir dağıtılmış ACO algoritması kullanılmıştır. İHA' nın menzil kısıtlaması dikkate alınarak, İHA' yı menzil içerisinde tutabilmek için yeni bir kural tasarlanmıştır. İkinci aşamada, Dubins eğrisi, ACO tarafından oluşturulan güzergâh noktalarını sorunsuz bir şekilde birbirine bağlamak için kullanılır. Yapılan çalışmada son olarak, sürü İHA görev planlama problemi için önerilen ISOA' nın yakınsama performansını, dış duyarlılığını ve dahili ölçeklenebilirliğini analiz etmek için simülasyonlar gerçekleştirilmiştir. ISOA, gözetleme kapsamını maksimuma çıkaran ve arama-kurtarma görevlerinde başarı oranı yüksek bir algoritmadır.

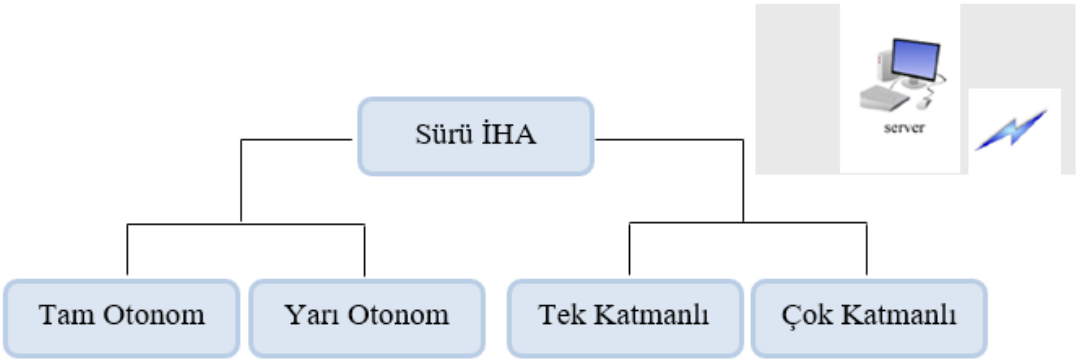
Karar verme katmanında kullanılan diğer bir yöntem ise hibrit yapay potansiyel alan (HAPF) yöntemidir. Zhen ve diğ. [16] tarafından yapılan çalışmada, hibrit bir yapay potansiyel alanı ve karınca kolonisi optimizasyonu (HAPF-ACO) yöntemi kullanılarak, belirsiz statik olmayan ortamlarda, hareketli hedefleri aramak ve bu hedefler ile ilgili işlem yapmak için İHA sürüsü iş birliğine dayalı bir çalışma sunulmaktadır. Bu çalışmada dinamik topoloji etkileşimi altındaki İHA sürüsünün arama görevi için, zamana duyarlı hedef olasılık haritası oluşturulmuştur. HAPF algoritmasına dayanarak, hedef olasılık haritasına göre tasarlanan hedef çekim alanı, uçuşa yasak bölge baz alınarak tasarlanmış tehdit itici alan ve çarpışmaları önlemek için İHA'

lar arası itici alan oluşturulmuştur. ACO algoritması ise İHA' ların küresel arama kabiliyetini geliştirmek için kullanılmıştır.

Sürü İHA sistemlerinde görev planlama alanında kullanılan diğer bir algoritma ise Genetik Algoritmasıdır. Wilhelm ve diğ. [17] tarafından yapılan çalışmada, çeşitli görev gereksinimlerini karşılayacak, sürü içerisinde hem sabit kanatlı hem de döner kanatlı hava aracı setini bulunduran bir sistem tasarımı için, Genetik Algoritmaya dayalı uyarlanabilir görev planlayıcısı önerilmiştir.

Sürü İHA sistemlerinde görev planlaması, genellikle belirli bir süre içerisinde yerine getirilmesi gereken bir hedef için görev planı oluşturma sürecidir. Görev planlayıcısının amacı zaman, güç tüketimi gibi kaynakları maksimize etmektir. Bu tür içerisinde birden fazla maksimize edilmesi gereken kaynakların var olduğu sistemleri çözümlemek için Ramirez ve diğ. [18] Çok amaçlı Genetik Algoritmasını kullanmışlardır. Çok Amaçlı Genetik Algoritması, büyük arama uzayı içerisinde çok küçük bir çözüm uzayına sahip problemler için, literatür çalışmasında geçen diğer algoritmalara göre daha iyi performans gösterir.

Sürü teknolojisinin İHA alanında kullanımını incelediğimizde kesin sınırları belirlenmiş bir sınıflandırmanın yapılmadığı görülür. Sürü İHA' lar birbirinden farklı yollar kullanılarak sınıflandırılabilir. Şekil 2.5 bu sınıflandırma örneklerinden bir tanesi olarak gösterilmiştir.



Şekil 2.5: Sürü İHA sınıflandırma örneği.

Yapılan çalışmalar, temelde İHA' ları tam ve yarı otonom olmak üzere ikiye ayırır. Başka bir bakış açısına göre, sınıflandırma, her İHA' nın kendi lideri olduğu tek katmanlı sürüler veya her katmanda lider bir İHA' nın yer aldığı, daha yüksek bir katmanda lider İHA' larına rapor veren, çok katmanlı sürüler olarak da düşünülebilir; yer kontrol istasyonunun sunucu olarak kullanıldığı bir organizasyonda, yer kontrol istasyonu bu hiyerarşideki en yüksek katmandır [19].

3. MALZEME VE YÖNTEM

Genel kısımlar bölümünde, sürü İHA alanında yapılan çalışma alanları ile ilgili verilen bilgiler ışığında, mevcut çözümler ile ilgili daha kısa sürede yakınsama gereksinimi olduğu görülür. Bu alanlardan biri olan, birçok görevin sürü İHA arasında istenen sonuçlara ulaşabilmek için nasıl paylaştırılacağı da büyük bir araştırma konusudur. Literatürde bu araştırma alanı sürü İHA sisteminde "Görev Planlama" olarak adlandırılmaktadır. Bir sürü İHA sisteminde, sürü İHA' yı belli görevler için devreye almanın ilk adımı görev planlamadır.

Literatürde görev planlama adımında birçok yöntem kullanılır. Bu alanda daha iyi sonuçlara ulaşabilmek için "Sırt çantası" ve "Seyahat" problemlerinin kombinasyonlarından oluşan yöntemlerin kullanıldığı görülür. Bunun yanında, "bildirimsel yaklaşım", "modifiye edilmiş iki parçalı kurt sürüsü (MTWPS)", "ISOA" gibi algoritmaların kullanıldığı gözlemlenmiştir. Planlama adımında bu tür algoritmaların kullanım amacı değişen hava koşullarına karşı daha verimli sürü İHA sistemlerinin elde edilmesi, belirli bir süre içerisinde daha büyük alanların gözetlenmesi veya belirli kaynak kullanımlarının minimuma indirgenmek istenmesidir. Toplam güç tüketimi veya toplam harcanan süre, minimize etmek istenen kaynaklar arasında gösterilebilir.

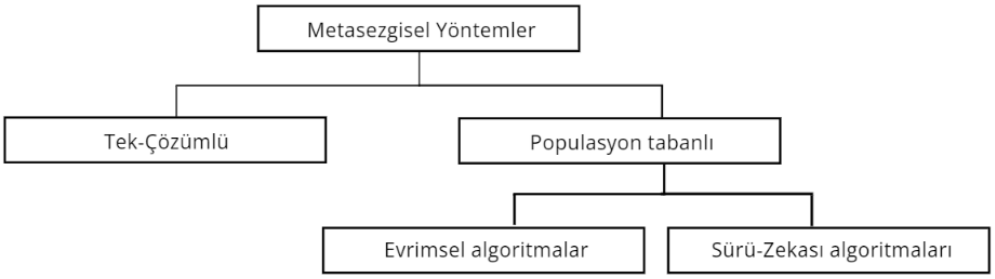
Bu çalışmamızda, planlama adımı için tasarladığımız simülasyonu, görevlerin minimum süre içerisinde minimum kaynak kullanımını sağlayan görev paylaşımını bulabilmek için meta-sezgisel yöntemlerden biri olan Çok Amaçlı Genetik Algoritması kullanılmıştır. Sürü İHA' ları simüle etmek için GAZEBO, MOGA algoritmasını gerçekleştirmek ve sürü İHA' nın kullanabileceği verilere dönüştürmek için MATLAB programı kullanılmıştır. MATLAB ile GAZEBO arasındaki haberleşme, Robotik İşletim Sistemi (ROS) altyapısı kullanılarak yerine getirilmiştir.

3.1 METASEZGİSEL YÖNTEMLER

Metasezgisel terimi, çok çeşitli optimizasyon problemlerinin çözümü için önerilen daha yüksek seviyeli sezgisel yöntemleri tanımlar. Glover, 1986 yılında yaptığı bir seminer konuşmasında sezgisel kelimesinin başına "meta" ön ekini getirmiştir. Bu zamandan itibaren lokal optimumu bulmak için kullanılan sezgisel yöntemlerin "ötesinde" çözümler üreten stratejiler anlamında meta-sezgisel yöntemler olarak kullanılmaya başlanmıştır [20].

Metasezgisel algoritmalar, kesin bir çözümü olmayan ve içerisinde birçok kısıtlamalar olan problemlerin çözümünde kullanılır. Bu kısıtlamalar ve problemin karmaşıklığı, problemin tek bir matematiksel modele indirgenmesini engeller. Bu tür çözümü kesin olmayan problemlerin çözülmesinde metasezgisel yöntemler kullanılır.

Metasezgisel algoritmaların çoğu biyolojik evrim sürecinden, sürü davranışlarından ve fizik yasalarından esinlenerek geliştirilmiş yöntemlerdir [21]. Bu algoritmalar genel olarak tek çözümlü ve popülasyon tabanlı metasezgisel algoritmalar olmak üzere iki kategoriye ayrılır. Tek çözümlü metasezgisel algoritmalar, çözüm kümesinde tek eleman içeren yöntemlerdir. İstenen optimizasyon seviyesine, bu tek elemanlı çözüm kümesi üzerinde işlem yapılarak ulaşılır [22]. Ancak, tek çözümlü metasezgisel yöntemlerden elde edilen çözüm, yerel optimumda sıkışıp kalabilir [23].



Şekil 3.1: Metasezgisel algoritmaların sınıflandırılması [23].

En çok kullanılan tek-çözümlü metasezgisel yöntemler şunlardır:

- Tabu Search (TA)
- Microcanonical Annealing (MA)
- Guided Local Search (GLS)

Popülasyon tabanlı metasezgisel yöntemler arama işlemi sırasında çözüm kümesinde birden fazla çözüm bulundurur ve istenen optimizasyon seviyesine ulaşabilmek için bu çözümler üzerinden işlemler gerçekleştirilir. Bu çözümler arasındaki çeşitlilik korunarak, çözümlerin yerel optimumda sıkışıp kalması engellenir. En çok kullanılan popülasyon tabanlı meta-sezgisel yöntemler şunlardır:

- Genetic Algorithm (GA)
- Particle swarm Optimization (PSO)
- Ant Colony Optimization (ACO)

Bu tez çalışmasının simülasyon fazında, popülasyon tabanlı bir algoritma olan Genetik Algoritması ve farklı varyantları uygulanmıştır.

3.2 GENETİK ALGORİTMASI

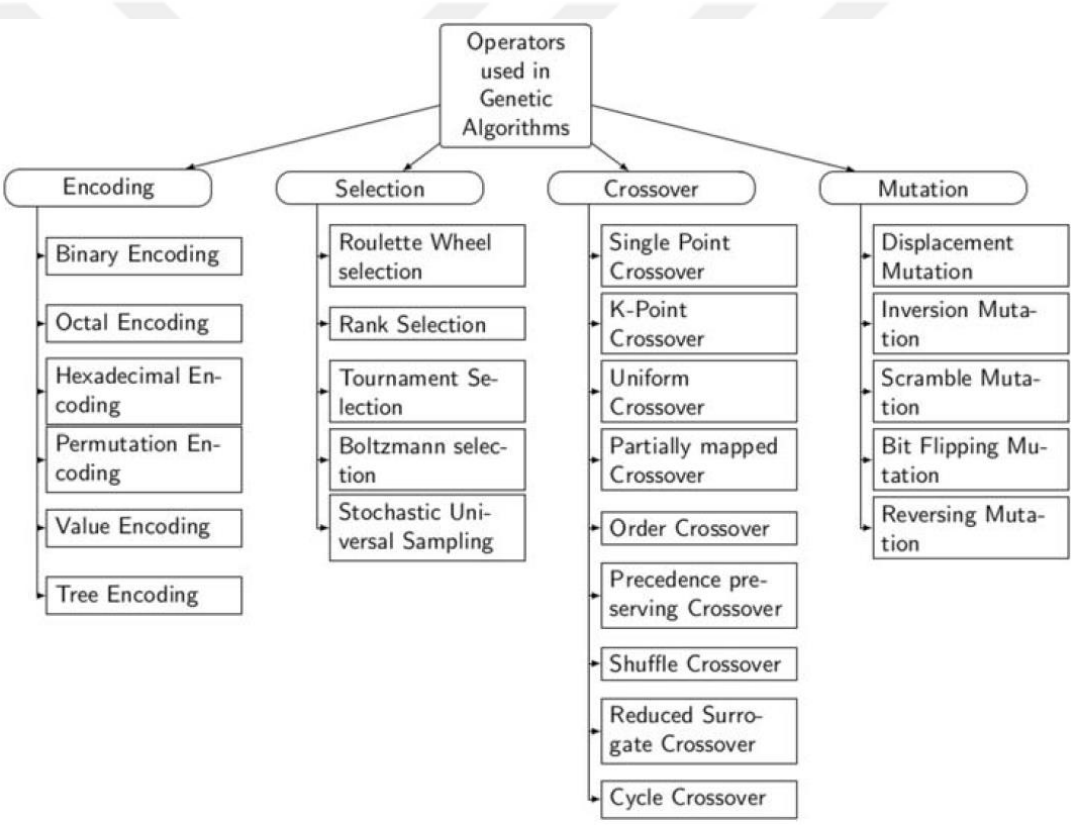
Genetik algoritması ilk olarak 1965 yılında J.H. Holland tarafından önerilmiştir. Genetik algoritması, mühendislik, finans, endüstri gibi alanlarda optimizasyon problemlerine çözüm aramak için kullanılır. Genetik algoritmasının kullanım amacı temelde Darwin'in evrim teorisini taklit ederek, mühendislik ve diğer bilimlerde karşılaşılan problemlere çözüm aramaya çalışmaktır. Genetik algoritması gerçek hayatta karşılaşılan problemlerin çözümlerini seçerken, evrim teorisinde gerçekleşen, ortam için daha güçlü canlıların hayatta kalmasından esinlenir. Bu yöntem yeni çözümlerin üretildiği her bir jenerasyonda zayıf çözümlerin devre dışı kalmasını sağlar.

GA temelde yeni çözümleri üretmek için rastgele seçimler yaptığı için stokastik bir algoritmadır. Stokastik bir algoritmanın çözümlerinin ne kadar güvenilir olduğu ile ilgili tartışmalar mevcuttur. Optimizasyon problemlerinde global optimumu bulmak için GA' nın kullanılmasındaki güven, GA' nın her jenerasyonda en iyi çözümleri koruması ve yeni çözüm uzaylarını bu en iyi çözümleri kullanarak oluşturmasından kaynaklanır. Böylelikle jenerasyon ilerledikçe popülasyon en iyi çözümlerden oluşmaya başlar [24].

GA' da her çözüm bir kromozomu ve her parametre bir geni temsil eder. GA uygunluk işlevi kullanarak popülasyondaki her bireyin uygunluğunu kontrol eder. Yeni jenerasyonda yer alan kötü çözümlerden kurtulmak için iyi çözümlerden rastgele çözümler seçilir [24]. GA' da Darwin teorisinde ilham alan operatörler şunlardır:

- Kodlama
- Seçim
- Çaprazlama
- Mutasyon

Kodlama adımında, verilen problemin bileşenleri, genler ve genlerin oluşturduğu kromozomları nasıl temsil edeceği ile ilgili bir mekanizma oluşturulur. Daha sonra bu mekanizma altyapısı kullanılarak GA tarafından kullanılacak ilk popülasyon oluşturulur. Seçim adımında kromozomlar, daha sonraki süreçlerde kullanılmak üzere uygunluk değerlerine göre seçilir [25]. Çaprazlama adımında, farklı yöntemler kullanılarak, ebeveynlerden (anne ve baba çözümlerinden) yeni bir çözüm (offspring) oluşturulur. Mutasyon adımında ise ebeveynlerde bulunmayan genlerin yeni oluşturulacak çözümlerde yer alması sağlanır. Şekil 3.2' de, GA' da kullanılan her bir operatör için kullanılabilecek yöntemler verilmiştir.



Şekil 3.2: GA' da kullanılan operatörler ve farklı yöntemler [23].

3.2.1 Kodlama

Oluşturulacak kromozom yapısı her problem alanı için farklılık gösterir. Literatürde kodlama için bazı yöntemler kullanılmıştır. Bu yöntemlerin başında ikili, sekizli, onaltılı, permütasyon, değer-bazlı ve ağaç kodlamaları gelir [23].

İkili kodlamada her gen “bir” veya “sıfır” olarak gösterilir. Bu kodlama türü çaprazlama ve mutasyon adımlarının hızlı bir şekilde devreye alınmasını sağlar. Ancak kodlama alanı kısıtlı olduğu için karmaşık problemlerde kullanılması önerilmemektedir. Tablo 3.1’ de ikili kodlama örneği görülmektedir.

Tablo 3.1: Kromozomların ikili kodlaması.

Kromozom	**Kod**
A	0011101010011010
B	1100010101000110

Sekizli kodlamada her gen 0 ile 7 arasında, onaltılık kodlamada 0 ile F arasında kodlanır. Permütasyon kodlamasında gen, bir dizideki konumu temsil eden sayı dizisi ile temsil edilir. Tablo 3.2’ de permütasyon kodlama örneği görülmektedir.

Tablo 3.2: Kromozomların permütasyon kodlaması.

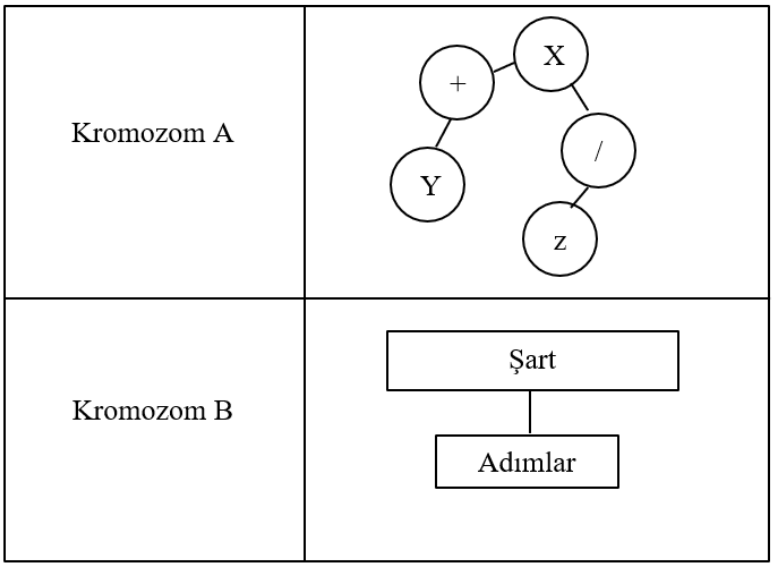
Kromozom	**Kod**
A	2 6 3 5 1 8 4 7 9
B	1 8 2 5 3 6 7 9 4

Değer kodlamasında gen, gerçek, tam sayı veya karakterlerle temsil edilir. Karmaşık problemlerin kodlamasında kullanılması avantaj sağlar. Tablo 3.3’ te kromozomların değer kodlama örneği sunulmaktadır.

Tablo 3.3: Kromozomların değer kodlaması.

Kromozom	Kod
A	1.53 5.31 2.45 3.87 5.0
B	ABCDEFGHLKMN

Ağaç kodlamasında gen, fonksiyon ağaçları veya komutlar ile temsil edilir. Şekil 3.3' te kromozomların ağaç yapısı ile kodlanması örneği verilmektedir.



Şekil 3.3: Kromozomların ağaç yapısı ile kodlanması.

Bu tez çalışmasında yapılan simülasyonda, her bir gen bir vektörden oluşur. Vektör, hangi görevin hangi İHA tarafından yapılacağı bilgisini tutar. İHA ve görevlerin kolayca numaralandırmaya imkân sağladığı için permütasyon kodlamasının kullanılması uygun görülmüştür. Tablo 3.4' te, 7 görevin ve 3 İHA' nın bulunduğu bir simülasyon çalışmasında kullanılan kodlama verilmiştir.

Tablo 3.4: Sürü İHA simülasyonunda kullanılan kodlama.

Kromozom	Kod
A	(5,2) (1,3) (2,3) (3,2) (4,1) (6,1) (7,2)
B	(5,1) (1,2) (2,3) (3,1) (4,2) (6,3) (7,3)

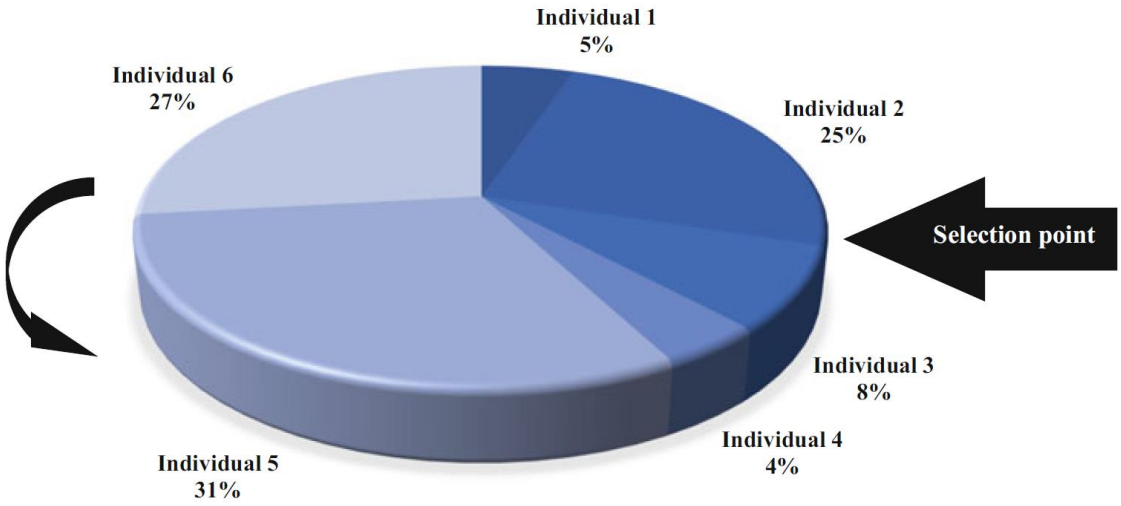
Her genin ilk elemanı yapılacak görevin numarasını temsil eder. İkinci elemanı ise bu görevin hangi İHA tarafından yapılacağı bilgisini tutar. A kromozomunda 1 numaralı İHA' nın yerine getirmesi gereken görevler sırasıyla 4 ve 6 numaralı görevlerdir.

3.2.2 Seçilim

Doğal seçilim evrimin temel adımlarından bir tanesidir. Temelde seçilim, popülasyondaki her bir bireyin bir sonraki jenerasyonda yer alıp almayacağı ile ilgili bir karar mekanizmasıdır. En çok kullanılan seçilim teknikleri şunlardır:

- Roulette Wheel (Rulet Çarkı)
- Tournament (Turnuva)
- Rank (Derece)

Rulet Çarkı yönteminde, her kromozom uyumluluk yüzdelerine göre bir çarka yerleştirilir. Daha sonra bu çark çevrilerek bir sonraki jenerasyonda yer alacak çözüm rastgele seçilir. Böylelikle çark üzerinde daha çok alana sahip olan bireyler bir sonraki jenerasyonda yer alma olasılıklarını arttırmış olurlar. Seçilme oranı düşük olan kromozomların seçilmesini tamamen önleme, popülasyondaki çeşitliliği azaltacağı için bulunan çözümler güvenli olmayabilir. Şekil 3.4' de bir Rulet Çarkı seçilim örneği görülmektedir.



Şekil 3.4: Rulet Çarkı örneği [20].

Rulet Çarkı yönteminde bir çözümün seçilme olasılığı, çözümün uyumluluk değerinin diğer tüm çözümlerin uyumluluk değerlerinin toplamına bölünerek bulunabilir. İlgili ifade 3.1 eşitliğinde verilmiştir.

$$p[i] = \frac{Fi}{\sum_{j=0}^{n} \mathrm{Fj}} \qquad (3.1)$$

F_i: i. Elemanın uygunluk değeri

F_j: j. Elemanın uygunluk değeri

P_i : i numaralı çözümün seçilme olasılığı

Seçilim sürecinde kullanılan bir diğer yöntem Turnuva yöntemidir. Bu yöntem ilk defa 1983 yılında Brindle tarafından önerilmiştir. Bu yöntemde, çözümlerin seçilme olasılıkları eşit kabul edilir ve çözüm popülasyonu içerisinden rastgele çözümler seçilir. Rastgele seçilimden sonra seçilen her bir çözüm diğer çözümler ile karşılaştırılır. En iyi uyumluluk değerine sahip olan çözüm bir sonraki jenerasyonda kullanılmak üzere yeni oluşturulacak popülasyona eklenir.

Seçilim sürecinde kullanılan diğer bir yöntem ise Bolztmann yöntemidir. Monte Carlo benzetiminde kullanılan örnek alma metotlarını baz alan bir yöntemdir [26].

De Jong ve Brindle, stokastik süreçlerinde, çözüme ulaşmada görülen hataları iyileştirmek için Rulet Çarkı yöntemini modifiye edip Derece yöntemini önermişlerdir. Derece yöntemi, seçilim esnasında çözümlerin uyumluluklarını baz alan bir yöntemden ziyade, derece değerlerini baz alan bir yöntem sunar. Derece yöntemi, çözümün yerel mimimuma erken zamanda yaklaşma problemini çözer. derece yöntemi kullanılarak yapılan seçimler iki adımdan oluşur. İlk adımda popülasyondaki elemanlar uygunluk değerlerine göre küçükten büyüğe doğru sıralanır. Daha sonrasında uygunluk değeri en düşük olan elemanın derece değeri 1 olarak atanır. Diğer elemanların derece değerleri atanırken ilk eleman baz alınarak düzenlenir. Derece tabanlı seçilim yönteminde, rulet çarkında işgal edilecek alanın yüzdesi ifade 3.2' de verilmiştir.

$$p[i] = \frac{ri}{\sum_{j=0}^{n} \mathrm{rj}} * 100 \qquad (3.2)$$

Burada $r_{i,,}$ i. bireyin serece değerini gösterir.

Uygunluk değelerinin sırasıyla f1 = 0.40, f2 = 0.05, f3 = 0.03 ve f4 = 0.02 olan dört elemanlı bir popülasyonda Derece değerleri Tablo 3.5' deki gibi olur. Bu tabloda Derece yüzdelerinin Rulet Çarkı ile karşılaştırıldığında birbirine daha yakın değerlerin atandığı görülür. Böylelikle oluşturulacak popülasyonda seçim şansı düşük olan elemanların seçilme olasılığı artmış olur.

Tablo 3.5: Örnek dört elemanlı popülasyon derece ve Rulet Çarkı değerleri.

Birey	Uygunluk	Rulet Çarkı	Derece	Derece Yüzdesi
1	0.4	80%	4	40%
2	0.05	10%	3	30%
3	0.03	6%	2	20%
4	0.02	4%	1	10%

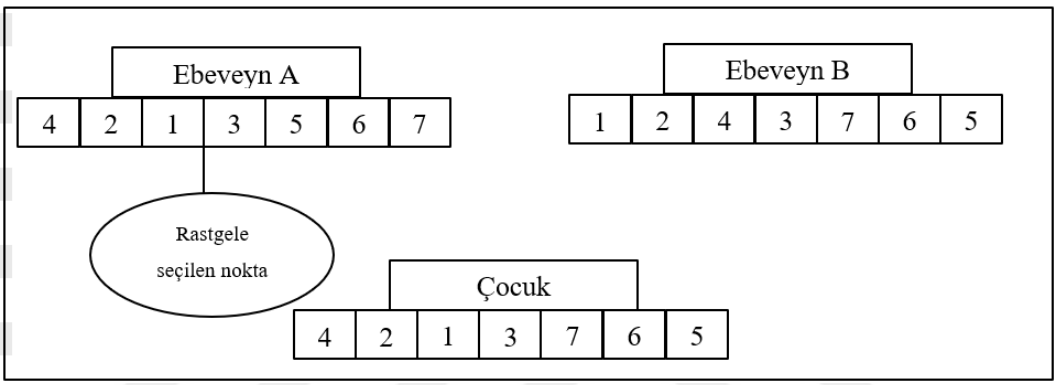
Bu tez çalışmasında Multi-Objektif Genetik Algoritması kullanıldığı için, seçilim esnasında Derece yönteminin kullanılması uygun görülmüştür. Bu yöntemi kullanmamızın diğer bir nedeni ise popülasyondaki çeşitliliği korumasıdır.

3.2.3 Çaprazlama

GA' da çaprazlama, ebeveynlerin genlerini kullanarak yeni çocukları (offspring) oluşturmak için uygulanan yöntemlerdir. En çok bilinen yöntemler şunlardır:

- Single Point
- Two-point
- Uniform

Single point çaprazlamada ebeveynlerden bir tanesinin genlerinin içerisinde rastgele bir nokta seçilir. Oluşturulacak çocuğun genlerinin ilk bölümü, bu noktaya kadar ilk ebeveynden seçilir. Geri kalan bölümü ise ikinci ebeveynden seçilir. Şekil 3.5' de single point çaprazlama örneği görülmektedir.



Şekil 3.5: Single point çaprazlama örneği.

Two point yönteminde ise iki faklı rastgele nokta seçilir ve çocuğun genleri sırasıyla bu iki farklı nokta arasında kalan genler seçilerek oluşturulur. Uniform çaprazlama tekniğinde ebeveynler bölümlere ayrılmaz. Çocuğun her bir geni için ebeveynlerden rastgele genler seçilir.

Bu tez çalışmasında, uygulamaya geçirilmesi kolay ve çok az kaynak tüketiminden dolayı single point yöntemi uygulanmıştır.

3.2.3 Mutasyon

GA' da mutasyon, farklılığı korumak için mevcut ebeveyn kromozomları içerisinde yer almayan genleri çocuğun kromozom yapısına eklemektir. En çok kullanılan mutasyon yöntemleri şunlardır:

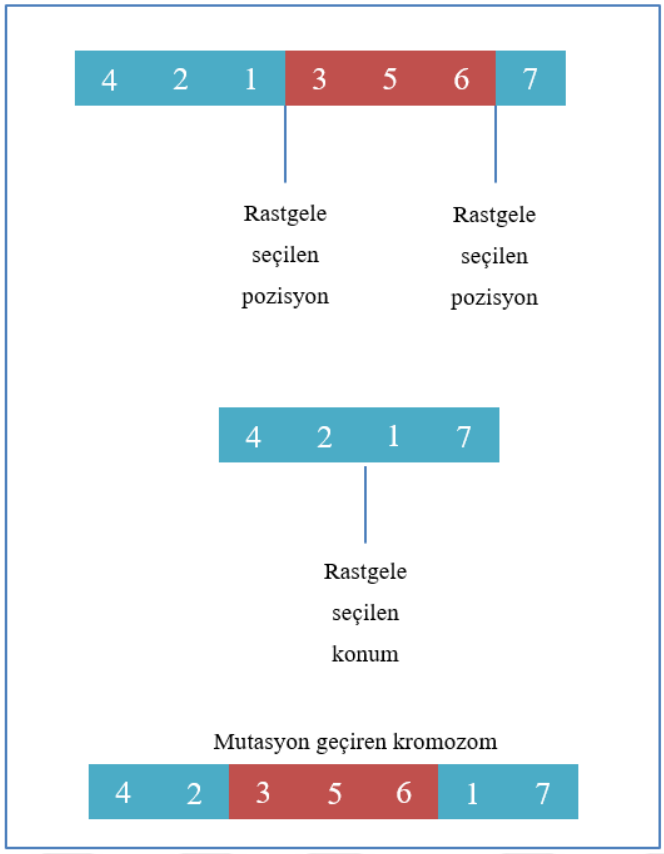
- Displacement

- Simple Inversion
- Scramble Mutation

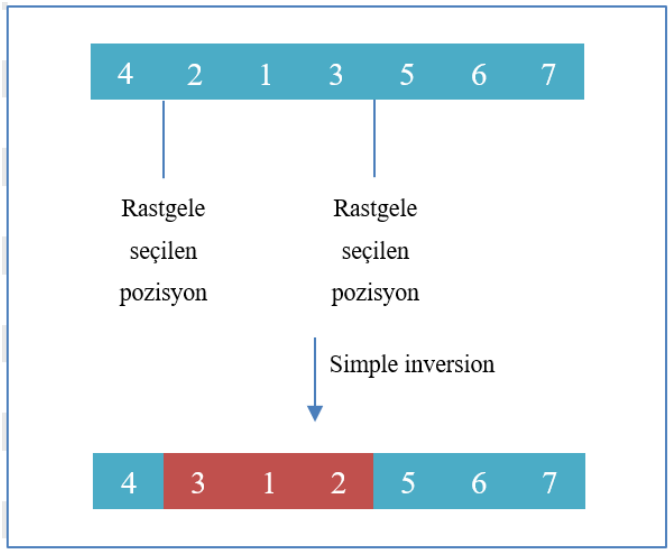
Displacement yönteminde verilen kromozom içerisinden rastgele iki gen seçilir. Seçilen iki gen arasında kalan genler, daha sonra orijinal kromozoma eklenmek üzere çıkartılır. Bu işlemden sonra orijinal kromozomun hangi alanına bu genin ekleneceği rastgele seçilir ve bu alana çıkartılan gen tekrar yerleştirilir. Şekil 3.6’ da displacement yöntemini gösteren bir örnek sunulmuştur.

Simple inversion metodunda kromozom içerisinden rastgele iki konum seçilir. Seçilen iki konum arasında kalan genler ters çevrilir. Şekil 3.7’ de simple inversion metodunu gösteren bir örnek verilmiştir.

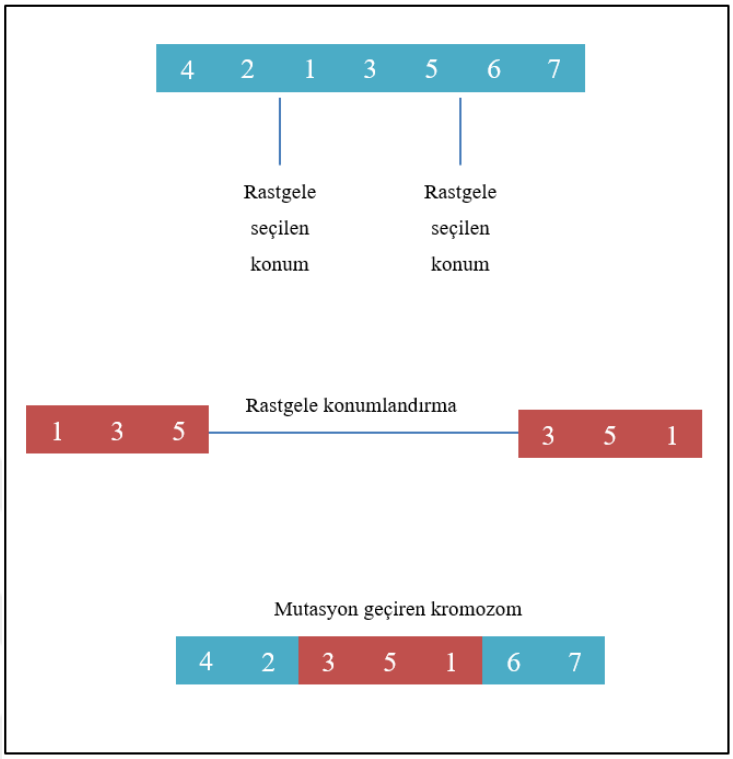
Scramble mutasyon metodu üç adımdan oluşur. Birinci adımda kromozom içerisinden rastgele iki konum seçilir. İki konum arasında kalan genler rastgele kendi içerisinde konumlandırılır. Daha sonra orijinal kromozoma eklenir. Şekil 3.8’ de scramble mutasyonuna örnek verilmiştir.



Şekil 3.6: Displacement çaprazlama örneği.



Şekil 3.7: Simple inversion çaprazlama örneği.



Şekil 3.8: Scramble çaprazlama örneği.

Bu yöntemlerden simple inversion yönteminin permütasyon kodlaması ile uyumluluğu akademik çalışmalarla kanıtlanmıştır. Bunun yanında simple inversion metodunda iki adımlı rastgele gen seçimi olmadığından kaynak kullanımı oldukça azdır. Scramble metodunda ise çeşitlilik son derece korunmasına rağmen kaynak kullanımı oldukça fazladır. Bu tez çalışmasında simple inversion ve scramble metotları uygulanıp sonuçları karşılaştırılmıştır.

3.2.4 Genetik Algoritmasının Matematiksel Modeli

Schema teoremi, GA' nın optimizasyon problemlerini çözerken dayandığı matematiksel modeli açıklamaya çalışan bir teoremdir. Schema teoremi, büyük çözüm kümelerine sahip uzayların örneklenerek optimal çözümlerin nasıl oluşturulabileceği ile ilgili GA' nın dizayn edilme aşamalarında hangi yöntemlerin kullanılabileceği ile ilgili araştırmacılara fikirler sunar. Bu teoremin başlangıç noktası, kromozomları baz alan bir desen oluşturmaktır. Bir problem için oluşturulacak kromozom kodlamasında ikili kodlama kullanıldığını varsayalım.

A kromozomu : 0100101100
B kromozomu : 0011001100
C kromozomu : 1100110011
D Kromozomu: 0100101001

Bu popülasyon için kullanılacak şema deseni H, 01***101** olsun. Bu desene uyumlu olan kromzomlar A ve D kromozomlarıdır. H deseninin ilk ve son bitlerinin konumlarının farkı l_H olarak tanımlanır. Desen içerisinde tanımlanmış bitlerin sayısı oH (Order of H) olarak tanımlanır. Yukarıdaki yapıda, H deseninin uzunluğu:

l_H = 8-1=7 (son tanımlanan bit (1)' in konumu 8, ilk tanımlanan bit (0)' ın konumu 1)

oH = 5 olarak hesaplanır (01101).

H bir şema olsun. mH(i) H desenine uyan kromozom sayısı, mH(i+1) bir sonraki jenerasyonda H' a uygun olacak kromozom sayısını gösterir. mH(i+1) beklentisi Schema teoremi ifade 3.3' te verilmiştir.

$$mH(i+1) = mH(i)\frac{F(H,i)}{F(i)}\left(1 - Pc\frac{lH}{l}\right)(1-Pm)^{oH} \quad (3.3)$$

F(H,i), H desenine uyan kromozomların ortalama uygunluk değeridir. İfade 3.4 ile gösterilir.

$$F(H) = \frac{1}{m}\sum_{i=0}^{n} F(H,i) \quad (3.4)$$

F(i), popülasyonda bulunan tüm kromozomların ortalama uygunluk değeridir. İfade 3.5 ile gösterilir.

$$F = \frac{1}{n}\sum_{i=0}^{n} F(i) \quad (3.5)$$

3.3 ifadesinde Pc çaprazlama olasılığı, Pm mutasyon olasılığını ifade eder. 3.3 ifadesinde kromozom ve mutasyon olasılığının 0 olduğunu varsayalım. mH(i+1)' in tamamen H desenine uyan kromozomların uygunluk değerine bağlı olduğu görülür. Göreceli uygunluk değerleri 1' den büyük olan şemaların sonraki jenerasyonlarda artarak devam edeceği açıktır.

lH, H deseninin uzunluğunun üye kromozom uzunluğuna eşit olduğu varsayalım. Bunun sonucu olarak mH büyük oranda kromozom olasılığına bağımlı hale gelir. Kromozom olasılığı arttıkça güçlü bireylerin bile artık sonraki jenerasyonlarda görülemeyeceği açıktır. Ayrıca oH (order of H) arttıkça, mutasyon meydana geldiğinde mH' de bozulmaların başlayacağı bu ifadeden anlaşılır.

3.2.5 Uygunluk Değerlerinin Ölçeklendirilmesi

Genetik algoritmasının uygulandığı çalışmalarda, açıklanan algoritmanın, seçilim sürecinde belli bir iterasyondan sonra birkaç "süper birey"' in ortaya çıkmasına neden olduğu gösterilmiştir [27]. Bu davranış sonraki jenerasyonlarda, kötü bireylerin hiç seçilmemesine ve çözüm arayışının lokal optimumda sıkışıp kalmasına neden olacaktır. Uygunluk ölçeklendirme yöntemleri, tanımlanan problemi çözmeyi amaçlar. Uygunluk ölçeklendirme, verilen problemin uygunluk fonksiyonu ne olursa olsun, her bir bireyin uygunluk değerinin tekrar hesaplandığı ve yeni jenerasyona bu değerlerle dahil edilmesini sağlayan bir adımdır [28].

Teoride ölçeklendirme yöntemleri üç ana kategoriye ayrılabilir:

- Linear
- Sigma Truncation
- Power Law

Linear ölçekleme yöntemleri, problemin özelliklerine bağlı olmayıp, popülasyonun maksimum, minimum ve ortalama (uygunluk) değerlerine bağlı olan yöntemlerdir. Sigma yöntemleri problemin sahip olduğu parametrelere bağımlıdır. Power Law ölçeklendirme yöntemi, işlenmemiş uygunluk değerlerini hesaba katarak yeni uygunluk değerlerini bulur [28].

Linear ölçekleme ifade 3.6' yı kullanarak her birey için uygunluk değerlerini hesaplar [28].

$$F' = a + b^{*}F \qquad (3.6)$$

Burada F = {f_1, f_2, f_3…F_N}, N büyüklüğündeki popülasyon bireylerinin uygunluk değerleri kümesi, F' = {f_1', f_2', f_3'…F_N'}, ölçekleme işleminden sonra oluşan yeni uygunluk değeleri kümesidir. İfadedeki a ve b katsayıları, 3.7 ve 3.8 ifadeleri ile bulunur.

$$a = \frac{Fort}{Fmax - Fort} \qquad (3.7)$$

$$b = \frac{Fort * Fmin}{Fmin - Fort} \quad (3.8)$$

Bu ifadelerdeki Fmax, küme içerisindek en büyük uygunluk değeri, Fmin en küçük değeri göstermektedir. Fort, tüm uygunluk değerlerinin ortalamasıdır ve ifade 3.9 ile bulunur.

$$\text{Fort} = \frac{\sum_{j=0}^{n} \text{Fi}}{\text{N}} \quad (3.9)$$

3.3 ÇOK AMAÇLI (MULTIOBJECTIVE) GENETİK ALGORİTMASI

Genetik algoritması kullanım amacına göre çeşitli varyantlara sahiptir. Bu varyantlar beş ana kategoride sınıflandırılır:

- Çok Amaçlı
- Gerçek ve İkili kodlu
- Paralel
- Kaotik
- Hibrit GA

Multi-objective GA (MOGA), genel olarak kullanılan ve kodlama, seçilme, çaprazlama ve mutasyon adımlarına sahip Genetik Algoritmasının modifiye edilmiş versiyonudur. MOGA çok amaçlı optimizasyon problemlerinde kullanılır. Çok amaçlı optimizasyon problemlerinde aynı anda birden fazla optimize edilmesi gereken hedefler veya maliyetler vardır. Çok amaçlı optimizasyon problemlerinde hedefler genellikle birbiriyle çatışır. Bu tür optimizasyon problemlerinde hedef veya maliyet optimizasyonu gibi amaçlardan birinde yapılan iyileştirme diğer amacı olumsuz etkileyebilmektedir. Bu nedenle, tüm amaçlar için fonksiyonları optimize eden tek bir çözüm yoktur [26]. Tek bir çözüm bulmak yerine, literatürde Pareto optimal veya Pareto cephesi olarak adlandırılan çözümler kümesi bulunur [29].

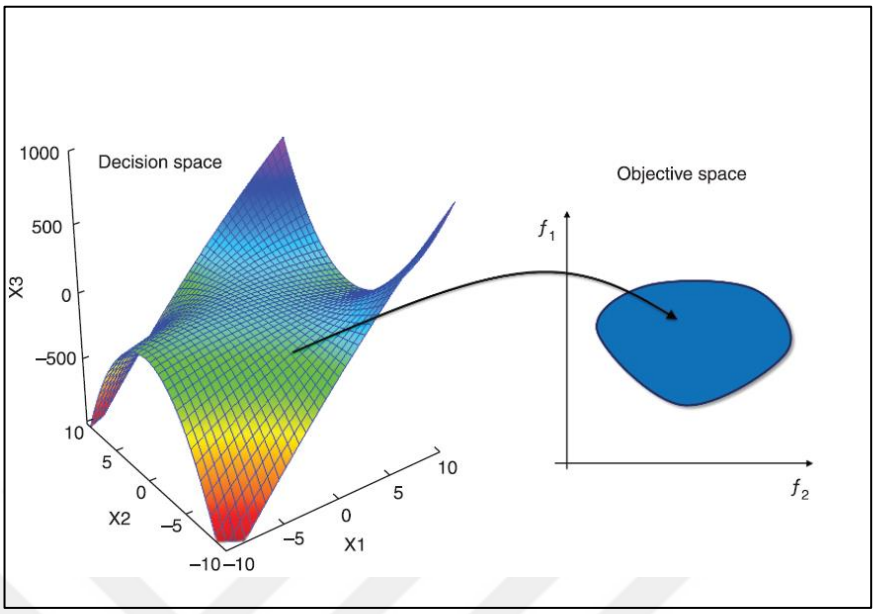
Bu tez çalışmasında uyguladığımız simülasyon çalışmasında, İHA' lar farklı hız değerlerine sahiptir. Her görev planlama isteğinde, İHA' ların bulunduğu konumlar ve durumlar farklıdır. İHA' ların bir tanesi havada operasyondayken diğerleri sabit bir yerde duruyor olabilir. Sabit

bir yerde duran ve hedefe yakın bir İHA' yı görev noktasına yönlendirmek, görev tamamlama süresini kısaltabilir ancak kalkış maliyetinden dolayı güç tüketimini arttırabilir. Bundan dolayı bu çalışmada temelde iki farklı amacımız vardır. Bu amaçlardan bir tanesi güç tüketimini minimuma indirgemek, diğeri ise görevlerin daha kısa sürede tamamlanmasını sağlamak. Sürü İHA' ların görev planlama adımında karşılaşılan çok amaçlı optimizasyon probleminin çözüm simülasyonu için MOGA kullanılmıştır.

Şekil 3.9, karar (decision) uzayında üç farklı çözüm içeren, amaç (objective) uzayında ise maksimize edilmesi gereken iki farklı amacı temsil eden bir yapıyı göstermektedir. Bunun yanında üç farklı çözümün amaç uzayında hangi değerlere denk geldiği gösterilmiştir. Maksimize edilmesi gereken her iki amaç fonksiyonu bizim için aynı oranda öneme sahip olsa da genellikle bu amaçlar için en uygun çözümü bulmak zordur [29].

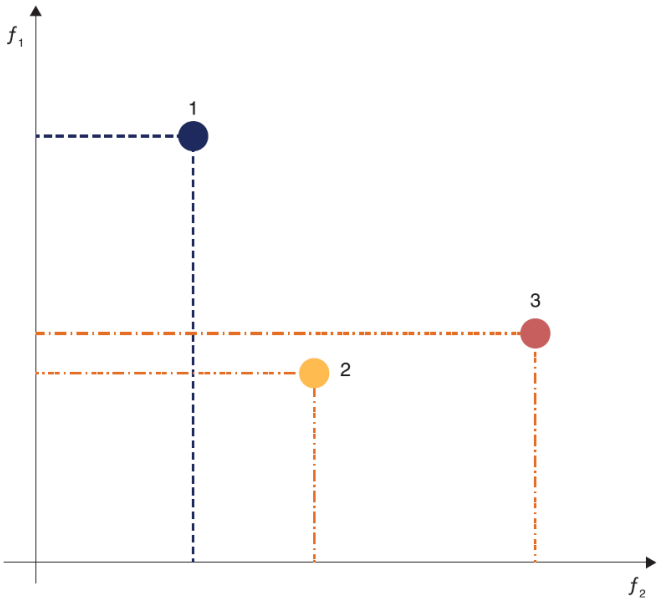
Şekil 3.10' da üç çözüme sahip amaç popülasyonu görülmektedir. Verilen bu örnekte, her iki amacında maksimize edilmek istendiği bir senaryoda, birinci çözüm ve ikinci çözüm karşılaştırıldığında f_1 amacı için birinci çözümün daha uygun olduğu görülür. Ancak f_2 amacı ele alındığında ikinci çözümün daha uygun olduğu görülmektedir. Bundan dolayı birinci çözüm için, ikinci çözüm tarafında baskılanmayan (non-dominated) denir. Aynı şekilde ikinci çözüm için de birinci çözüm tarafından baskılanmayan çözüm denir. İkinci ve üçüncü çözümler karşılaştırıldığında, f_1 ve f_2 amaçlarının her ikisi için de üçüncü çözümün daha uygun olduğu görülmektedir. Bu durumda ikinci çözüm, üçüncü çözüm tarafından baskılanmıştır (dominated) denir.

Birçok çözümden oluşan bir sistemde, en iyi çözümleri bulmak için olası tüm ikili karşılaştırmalar yapılır. Hangi çözümün hangisi tarafında baskılandığı veya hangi çözümlerin herhangi bir çözüm tarafından baskılanmadığı bu yöntemle bulunur. Çözüm kümemiz birbirini baskılamayan çözümlerden oluşan kümedir [29].



Şekil 3.9: Karar ve amaç uzayları [29].

Bu tez çalışmasında, amaçlardan biri güç tüketim fonksiyonu, diğeri ise verilen görevleri yerine getirmek için harcanan süredir. Simülasyon çalışmasında her iki amacın minimum değerlerini içeren çözüm kümesi MOGA yöntemleri kullanılarak bulunmuştur.



Şekil 3.10: Üç çözüme sahip amaç popülasyonu.

3.3.1 Güç ve Mesafe Uygunluk Değerlerinin Hesaplanması

GA' da uygunluk değerlerinin hesaplanması optimize edilecek problem alanı baz alınarak hesaplanır. Bu tez çalışmasında, sürü İHA' ların görevleri icra esnasında optimize edilmesi planlanan iki kaynak vardır. Bunlardan bir tanesi, sürü İHA' lar tarfından katedilmesi gereken toplam mesafe diğeri ise toplam harcanan güçtür.

Sürü İHA sisteminde bir çözümün aşağıdaki gibi kodlandığını varsayalım.

Kromozom A : [(5,2) (1,3) (2,3) (3,2) (4,1) (6,1) (7,2)]

Bu ifadede hangi numaralı görevin hangi hava aracı tarafından yerine getirilmesi gerektiğini gösterir. Bu kromozom için, sürü İHA sistemi tarafından katedilen toplam mesafe ifade 3.10 ile bulunur.

$$DF = D(t5,d2)+D(t1,d3)+D(t1,t2)+D(t5,t3)+D(t4,d1)+D(t6,d1)+D(t3,d2) \quad (3.10)$$

DF toplam katedilen mesafeyi gösterir. D(t5, d2) 5. Görev ile 3. İHA arasındaki mesafeyi temsil etmektedir. Sürü İHA sisteminde iki farklı görevin tek bir İHA tarafından yapılması gerektiği ihtimal dahilindedir. Verilen örnek kromozomda 5. Görev ve 3. Görev aynı hava aracı tarafından icra edilecektir. 3. Görevin yerine getirilmesi için gereken mesafe bundan dolayı D(t5,t3) olarak hesaplanır. Hesaplanan toplam mesafe aynı zamanda bu kromozom için mesafe uygunluk değerine eşit kabul edilmiştir.

Sürü İHA tarafında harcanan toplam güç hesaplamalarında, Hasini ve diğ. [30] tarafından yapılan çalışmalardan yararlanılmıştır. Bu çalışmada, İHA' nın farklı manevra hareketlerinde harcanan enerjinin hesaplanmasını ihtiva eden bir model geliştirilmiştir. Bu modele göre, hava aracının kalkış sırasında harcanan enerji ifade 3.11 ile hesaplanır. Hava aracının yatay hareketi sırasında harcanan enerji ifade 3.12 ile hesaplanır. Hava aracının havada asılı iken harcadığı enerji ifade 3.13 ile hesaplanır [30]. Burada H, İHA' nın bulunduğu yüksekliği göstermektedir. Sürü İHA sisteminde toplam harcanan güç, güç uygunluk değeri olarak kabul edilmiştir.

$$E = -0.432V2 + 3.786V - 1.224 \quad (3.11)$$

$$E = 308.709t - 0.852 \quad (3.12)$$

$$E = (4.917H + 275.204)t \quad (3.13)$$

Sürü İHA sisteminde bir çözümün aşağıdaki gibi kodlandığını varsayalım.

Kromozom A : [(5,2) (1,3) (2,3) (3,2) (4,1) (6,1) (7,2)]

Bu kromozom için, sürü İHA sistemi tarafından harcanan toplam güç ifade 3.14 ile bulunur.

$$PF = P(t5,d2)+P(t1,d3)+P(t1,t2)+P(t5,t3)+P(t4,d1)+P(t6,d1)+P(t3,d2) \quad (3.14)$$

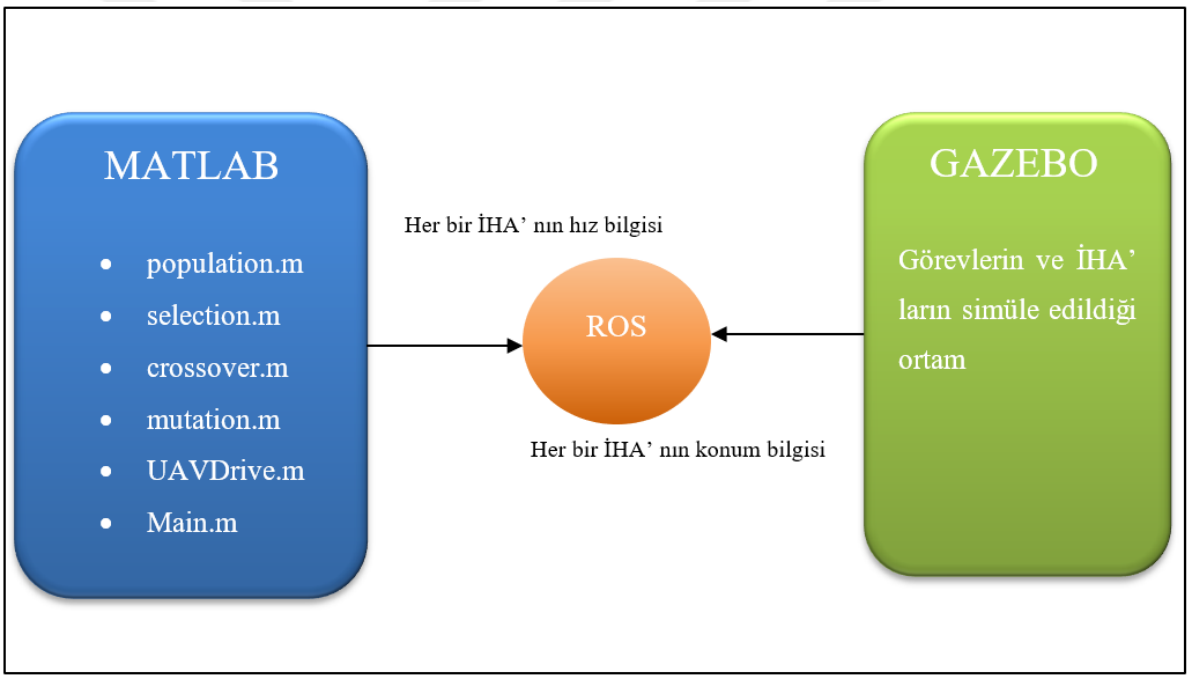
P(t,d) görevi için harcanan güç ifade 3.15 ile hesaplanır.

$$P(t,d) = P_{kalkış}(H_t, H_d) + P_{yatay}(L_t, L_d) + P_{gezinme} \quad (3.15)$$

Bu ifadede H_t ve H_d sırasıyla görevin icra edileceği yerin yüksekliğini ve d hava aracının yüksekliğini belirtmekte. Bu durumda $P_{kalkış}(H_t, H_d)$ ifadesi hava aracının kazanması gereken irtifaya sahip olmak için harcadığı güç olarak hesaplanır. L_t ve L_d, sırasıyla görev ve hava aracının x ve y koordinatlarındaki konumunu belirtmektedir. Bu durumda $P_{yatay}(L_t, L_d)$ ifadesi hava aracının görev mahalline ulaşması için harcadığı gücü ifade etmektedir. $P_{gezinme}$ ise hava aracının görevi icra esnasında havada asılı kalma güç maliyetidir.

4. BULGULAR

Tez çalışması kapsamında, sürü İHA sisteminin simülasyonu için Gazebo simülatörü, görev planlama algoritmalarının gerçeklemesi için MATLAB programı kullanılmıştır. Matlab programı içerisinde görev planlama algoritmaları çalıştırıldıktan sonra elde edilen veriler, Robotik İşletim Sistemi (ROS) üzerinden Gazebo simülatörüne gönderilir. Matlab ile Gazebo arasında, gerçek zamanlı bir haberleşme sistemi kurulmuştur. Her bir İHA' nın görev yerlerine ulaştırılması gerçek zamanlı olarak Matlab aracılığı ile yazılan algoritmalar ile gerçekleştirilir. Şekil 4.1' de simülasyon çalışmasının genel yapısı verilmiştir.



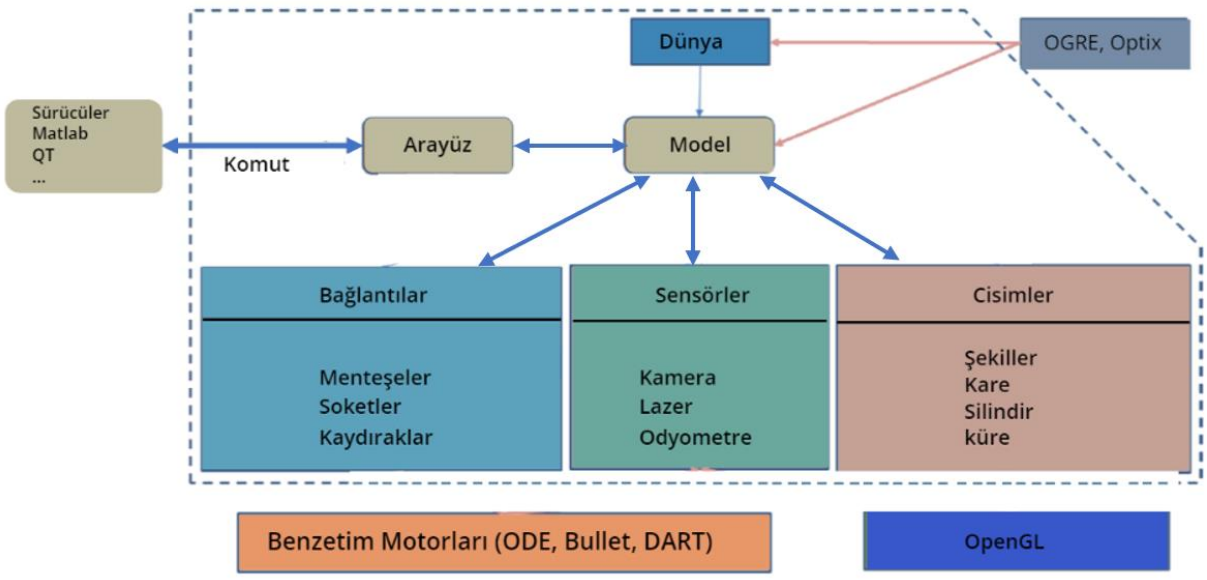
Şekil 4.1: Simülasyon mimarisi.

4.1 GAZEBO

Robot simülatörleri, gerçek hayatta tasarım aşamasına geçilmeden önce, bilgisayar ortamında tasarımı gerçekleştirmek için fiziksel ortamlar sunarlar. Yeni algoritmaların geliştirilmesi ve test edilmesine olanak sağlayan bu ortamlar, geliştirme süresini ve maliyetlerini azaltır [31].

Gazebo, bir doktora programının parçası olarak 2002 yılında Günay Kaliforniya üniversitesinde geliştirilmiştir. Gazebo' nun geliştirilmesindeki motivasyon, gerçek robotları simüle etme olanağı sunan yüksek kaliteli bir simülatör oluşturmaktır [32]. Gazebo simülatörünün en son versiyonu v11 Ocak 2020 yılında piyasaya sürülmüştür.

Robot çalışmalarında Gazebo simülatörünün çokça kullanılmasının nedeni, Gazebo' nun 3D ortamlar oluşturulmasına olanak sağlamasının yanında, robotik alanında kullanılan birçok sensörün simüle edilebildiği bir altyapı sağlamış olmasıdır. Gazebo' nun ROS ile entegrasyonu 2009 yılında Willow Garage' da gerçekleştirilmiştir. Gazebo ROS dışında diğer robotik platformlarına entegre edilebilir. Ancak robotik alanında Gazebo ROS' un en büyük tamamlayıcısı olarak görülmektedir. Şekil 4.2, Gazebo platformunun genel yapısını göstermektedir.



Şekil 4.2: Gazebo genel mimari [22].

Gazebo ilk sunulduğuna kullanılan bu yapı bu zamana kadar neredeyse hiç değişime uğramadan son versiyonda da kullanılmıştır. Simülatör programları katı cisimlerin dinamik hareketlerini taklit etmek için arka planda bir benzetim motoruna ihtiyaç duyarlar. ODE, Bullet, Dart gibi motorlarla Gazebo entegre bir şekilde çalışabilir. 2D ve 3D görüntüleri elde etmek için GLUT adı verilen program OpenGL vasıtasıyla kullanılır. Şekilde görülen komponentlerin

birbirinden bağımsız bir şekilde etkileşimde bulunması, Gazebo' nun bağımsız bir platform olmasına olanak sağlar.

Gazebo' da gzserver ve gzclient olmak üzere iki adet çalıştırılabilir program vardır. Ortamdaki fizik, sahneleme ve sensör olaylarını gzserver yürütürken, gzclient görselleştirme ve kullanıcı ile etkileşim olaylarını işler. İstemci ve sunucu (gzclient, gzserver) arasındaki haberleşmeyi sağlamak için boost::ASIO kütüphanesi kullanılır.

Gazebo' da İHA ve görevleri sanal bir ortamda test etmek ve tez çalışmasında kullanılan algoritmaların verimliliğini ölçmek, gerçek donanım gerektirmediğinden dolayı verimli bir yoldur.

4.2 ROBOTİK İŞLETİM SİSTEMİ

Robotik alanında araştırmacılar genellikle robot sisteminde yer alan alt bileşenlerin entegrasyonu ile ilgili teknik zorluklarla karşılaşırlar. Bu nedenle, tüm bileşenlerin basit bir şekilde birleştirilmesine ve haberleştirilmesine olanak sağlayan altyapıya ihtiyaç duyulur [33]. ROS robotik projelerinde bu ihtiyacı gideren, içerisinde birçok yazılım paketi ve araç içeren bir çerçeve sunar. ROS gibi açık kaynak kodlu projelerin geliştirilmesi ile birlikte robotik çalışmalar son yıllarda büyük değişimler geçirdi. Özellikle ROS içerisinde bulunan uygulamalar sayesinde bu alanda çalışan araştırmacılar için birçok araç erişilebilir hale geldi.

ROS isminde "işletim sistemi" geçmesine rağmen esasen içerisinde bir işletim sistemi barındırmaz. Bunun yerine, robot uygulamalarını devreye almak için ihtiyaç duyulan yapı taşlarını sağlayan bir yazılım geliştirme kitidir. Şekil 4.3 ROS' u oluşturan bileşenleri tasvir etmektedir.



Şekil 4.3: ROS' u oluşturan bileşenler [34].

ROS özünde, “ara katman yazılımı” veya “tesisat” olarak isimlendirilen bir mesajlaşma altyapısını kullanır. Haberleşme, bir robot uygulamasında, donanımla etkileşime girecek bir yazılım geliştirirken ihtiyaç duyulacak ilk bileşenlerin başında gelir. ROS, yayın/abonelik (publish/subscribe) modeli aracılığıyla dağıtılmış düğümler (nodes) arasındaki haberleşme ayrıntılarının tamamını yönetir [34]. Robotik ve açık kaynak kodlu yazılımlar üzerinde çalışan mühendisler ROS topluluğunu oluşturur.

ROS terminolojisinde kullanılan kavramlar ve açıklamaları aşağıda verilmiştir [34].

Master: Genellikle roscore olarak anılan Master, bağlantılar için istekte bulunan istemcilerin kayıtlarını tutar.

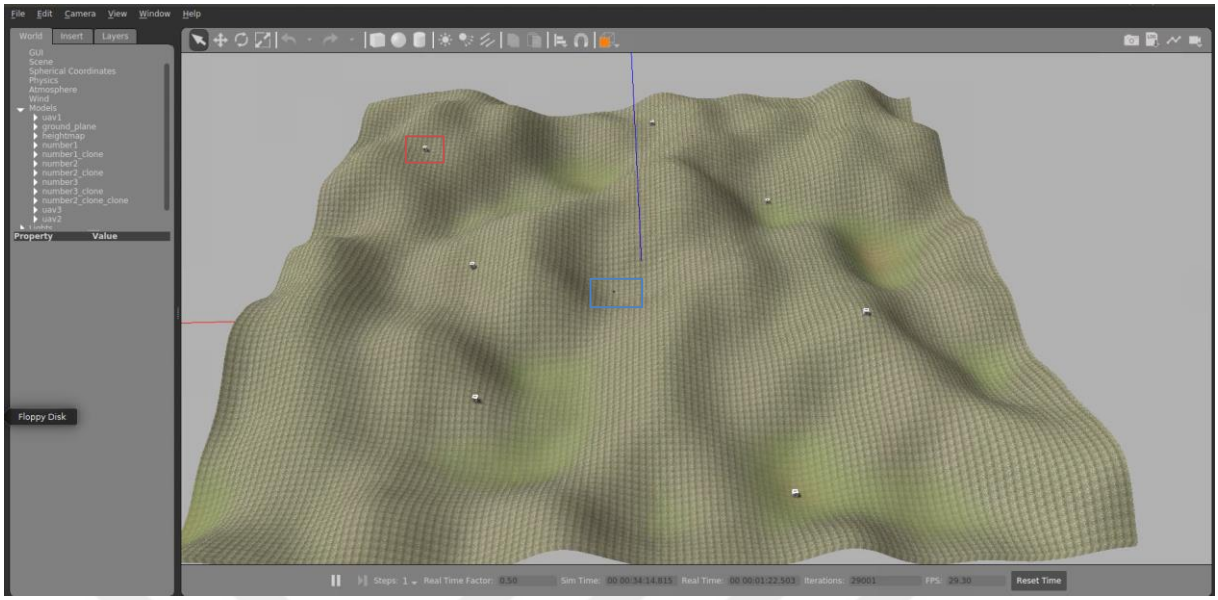
Host: ROS ağının üzerinde koştuğu bilgisayarı temsil eder ve IP adresi ile tanımlanır.

Node: ROS client API’ yını kullanan tüm uygulamalara verilen isimdir.

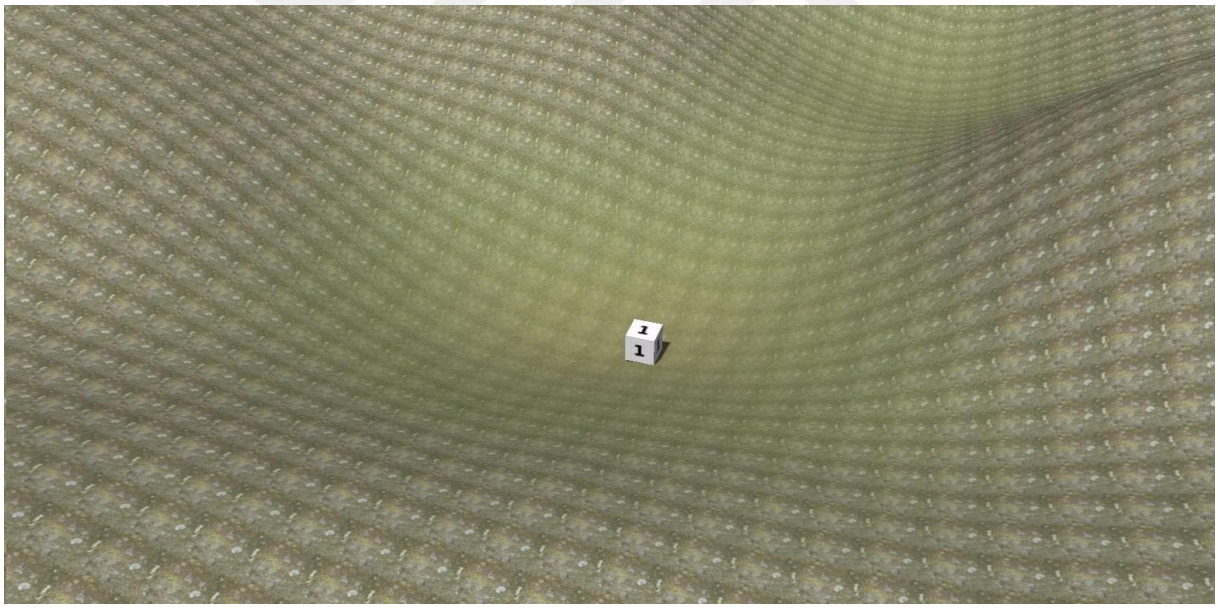
Topic: Publish/Subscribe yapısında iletişim kanalına verilen isimdir.

4.3 SİMÜLASYON ÇALIŞMASI

Planlama algoritmasının sonuçlarını analiz etmek için, 7 farklı görevin 3 farklı İHA tarafından icra edildiği bir durumun simülasyonunu uyguladığımız ortam Şekil 4.4’ de verilmiştir. Gazebo simülatöründe oluşturulan ortamda kırmızı kutu içerisinde gösterilen kare, 7 farklı görevden bir tanesini temsil etmektedir. Görevlerin tanımları için bu kareler 1’ den 7’ ye kadar numaralandırılmıştır. Bir numaralı görevin yakından görünüşü Şekil 4.5’ te verilmiştir.

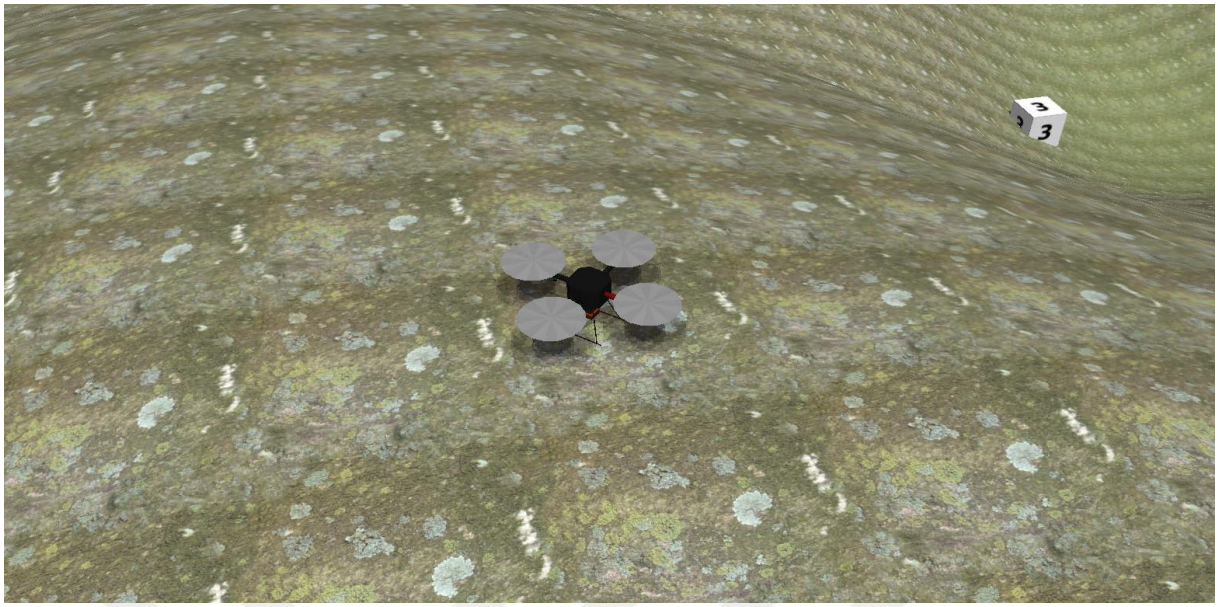


Şekil 4.4: Sürü İHA' nın Gazebo simülasyonu.



Şekil 4.5: 1 Numaralı görevin görüntüsü.

Simülasyon ortamında mavi kutu içerisinde gösterilen kare, 3 farklı İHA' dan bir tanesini temsil etmektedir. Kullanılan hava araçları Gazebo içerisinde, sol tarafta yer alan panel içerisinde görüldüğü gibi uav1, uav2 ve uav3 olarak isimlendirilmiştir. İHA' nın yakından görünüşü Şekil 4.6' da verilmiştir.



Şekil 4.6: İHA görüntüsü.

Simülasyon çalışmasında kullanılan İHA dört rotorlu bir hava aracıdır. Kullanılan hava aracı, Gazebo topluluğu tarafından geliştirilen ve sunulan açık kaynak kodlu bir projenin ürünüdür. Hava aracı üzerinde bulunan ve Gazebo tarafından simüle edilen birçok sensör mevcuttur. Bu sensörler aşağıdaki gibi sıralanabilir:

- Tarayıcı
- Kamera
- Sonar
- Odyometre
- GPS

Tarayıcı ve kamera sensörleri gözetleme ve haritalama çalışmalarında kullanılırken, sonar sensörü, ortamda eğer varsa engellerden kaçınmak için kullanılır. Odyometre ve GPS sensörleri hava aracının konumu hakkında bilgi almak için kullanılır. Bu tez çalışması kapsamında Odyometre sensörü, hava araçlarının konumlarını almak ve görev yerlerine yönlendirmek için kullanılmıştır.

Gazebo ortamında kullanılan hava aracının sahip olduğu Node ve Topic isimleri Tablo 4.1’ de verilmiştir.

Tablo 4.1: Node ve Topic isimleri.

NODE	TOPIC
/controller_spawner	/uav/command/attitude
/estop_relay	/uav/command/motor
/gazebo	/uav/command/pose
/gazebo_gui	/uav/command/pose_marker
/ground_truth_to_tf	/uav/command/thrust
/landing_action	/uav/command/twist
/pose_action	/uav/command/twist_limit
/rosout	/uav/command/wrench
/takeoff_action	/uav/command/yawrate
/teleop	/uav/controller/attitude/pitch/parameter_descriptions
/uav1/estop_relay	/uav/controller/attitude/pitch/parameter_updates
/uav1/ground_truth_to_tf	/uav/controller/attitude/pitch/state
/uav1/landing_action	/uav/controller/attitude/roll/parameter_descriptions
/uav1/pose_action	/uav/controller/attitude/roll/parameter_updates
/uav1/takeoff_action	/uav/controller/attitude/roll/state
/uav2/controller_spawner	/uav/fix/position/parameter_descriptions
/uav2/estop_relay	/uav/fix/position/parameter_updates
/uav2/ground_truth_to_tf	/uav/fix/status/parameter_descriptions
/uav2/landing_action	/uav/fix/status/parameter_updates
/uav2/pose_action	/uav/fix/velocity/parameter_descriptions
/uav2/takeoff_action	/uav/fix/velocity/parameter_updates
/uav3/estop_relay	/uav/fix_velocity
/uav3/ground_truth_to_tf	
/uav3/landing_action	
/uav3/pose_action	
/uav3/takeoff_action	

Simülasyon çalışmamızda, bütün görevlerin minimum güç tüketimi ile birlikte en kısa sürede tamamlanmasını sağlayan çözümü bulmak için MOGA yöntemleri uygulanmıştır. MOGA algoritmasının planlama aşamasında kullanımının gerçeklemesi, her adımda farklı yöntemlerin sürü İHA simülasyonu üzerindeki etkileri, Matlab aracılığı ile tasarlanmıştır.

Simülasyonun ilk aşamasında, farklı görevlerin hangi hava aracı tarafından tamamlanacağını gösteren bir çözüm kümesi oluşturulmuştur. Bu çözüm kümesi, görevlerin rastgele hava araçlarına atanmasıyla oluşturulur. Simülasyonda kullandığımız ilk çözüm kümesi eleman sayısı 100’ dür. Simülasyonun devamında sırasıyla, çaprazlama, mutasyon ve seçilim adımları uygulanmıştır. Seçilim sırasında, MOGA’ nın çeşitliliği arttırmak için önerdiği Kalabalık Mesafesi (Crowding Distance) hesaplanmış ve seçilim buna göre yapılmıştır.

En uygun çözüm bulunduktan sonra, görev bilgileri ROS üzerinde Gazebo' ya gönderilir. Hava aracının Gazebo simülasyon ortamındaki hareketleri 100ms' lik periyotlarla kontrol edilir. Hava aracının bu kontrolü Matlab aracılığı ile yazılan program tarafından gerçekleştirilmiştir. Hava aracının hedefe en kısa sürede ulaşıp görevi tamamlaması için PD kontrol algoritması kullanılmıştır. Hava aracı görev yerine ulaşırken, önce görev yerinin kendisine göre hangi yönde olduğunu tespit eder ve rotasını buna göre ayarlar. Daha sonra bu görev için çıkılması gereken yüksekliğe ve görev noktasına varma işlemlerini eş zamanlı olarak gerçekleştirir.

Hava aracı bir görevi bitirdikten sonra, kendisine atanmış başka görev olup olmadığını kontrol eder. Eğer yeni görev varsa, tamamlamış olduğu görev yerinden, yeni görev yerine hareketini başlatır. Gazebo, bütün hava araçları kendisine atanmış görevleri bitirdikten sonra simülasyonun sona erdiği bilgisini Matlab' a gönderir.

Simülasyonun ilk aşaması olan MOGA' nın uygulanması ve en iyi elemanlardan oluşan bir çözüm kümesinin bulunması adımlarından sonra oluşan çözüm kümesinin Derece (Rank) ve Kalabalık Mesafesi bilgilerini içeren çözüm elemanları Tablo 5.1' de verilmiştir.

4.3.1 Sürü İHA Planlamasında Uygulanan Yöntemler

Bu bölümde, simülasyonun ilk aşaması olan minimum güç ve süre gereksinimlerini sağlayan MOGA algoritmasının bazı katmanlarında uygulanan farklı yöntemler incelenmiştir. Uygulanan bu yöntemlerden bir tanesi mutasyon adımında yapılan değişikliklerdir. Mutasyon adımında iki farklı yöntem uygulanmıştır. Bunlardan bir tanesi takas (scramble mutation) yöntemi, diğeri ise karıştırma (scramble mutation) yöntemidir. Mutasyon adımında, ebeveynlerden oluşturulan çocuk çözümünün mutasyona uğrayıp uğramaması ile ilgili bir karar mekanizmasına ihtiyaç vardır. Bu karar mekanizması önceden belirlenmiş bir yüzde parametresine göre oluşturulur. Bu parametre literatürde "mutasyon şansı" olarak kullanılmıştır. Mutasyon adımında son olarak farklı mutasyon şansı parametrelerinin simülasyon üzerindeki etkisi incelenmiştir.

Mutasyon adımında uygulanan farklı yöntemlerin dışında, çaprazlama adımında bir parametre değişikliğinin simülasyon üzerindeki etkileri incelenmiştir. Literatürde her bir çaprazlama adımında tek çocuk çözümü veya iki farklı çocuk çözümünün oluşturulması ile ilgili çalışmalar mevcuttur. Bu tez çalışması kapsamında çaprazlama adımında, ebeveynlerden tek ve çift çözümler oluşturularak, simülasyon üzerindeki etkileri incelenmiştir.

4.3.1.1 Mutasyon Fazında Uygulanan Takas Yönteminin Simülasyon Üzerindeki Etkisi

MOGA' nın mutasyon fazında uygulanan "takas yöntemi" sonuçları aşağıdaki başlangıç parametreleri ile gerçekleştirilmiştir.

- Görev Sayısı : 7
- Hava Aracı sayısı : 3
- Çözüm Popülasyonu : 100
- Maksimum Iterasyon : 50
- Deneme Sayısı : 5

Tablo 4.2, Maksimum iterasyon sayısından sonra oluşan en iyi 10 çözümün "derece" ve "kalabalık mesafesi" içeren değerlerini göstermektedir.

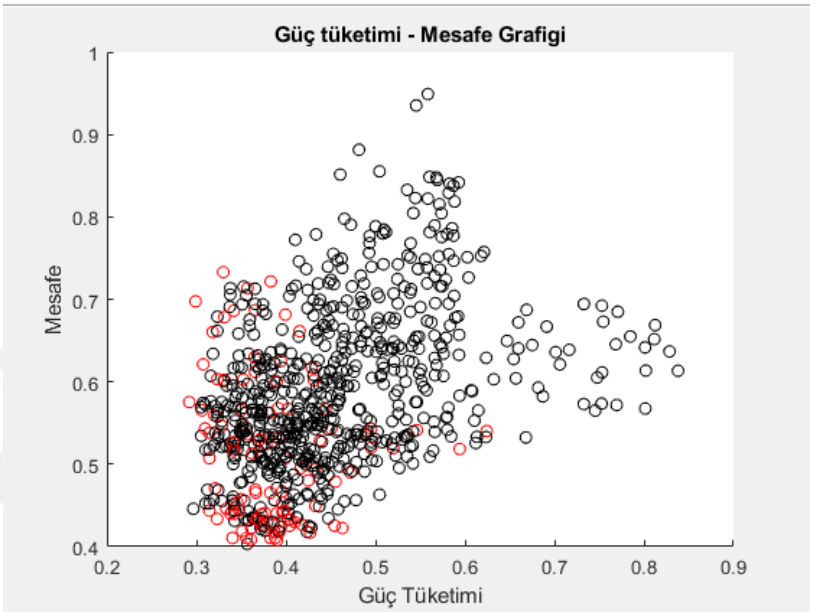
Tablo 4.2: Takas yöntemi sonuçları.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[2,1]	[1,1]	[4,3]	[6,3]	[5,3]	[3,1]	[7,3]	1	Inf
[2,1]	[1,1]	[5,3]	[4,3]	[3,2]	[7,2]	[6,2]	1	0.776
[2,1]	[5,2]	[1,1]	[3,2]	[4,3]	[6,3]	[7,3]	1	0.770
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[6,2]	[7,3]	1	0.746
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[6,1]	[7,3]	1	0.673
[2,1]	[1,1]	[4,3]	[5,2]	[6,2]	[7,3]	[3,1]	1	0.656
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[7,2]	[6,3]	1	0.650
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[6,1]	[7,3]	1	0.646
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[6,2]	[7,1]	1	0.594
[2,1]	[1,1]	[5,2]	[4,3]	[3,3]	[6,2]	[7,2]	1	0.447

G1' den G7' ye kadar olan ikililer hangi görevin hangi hava aracı tarafından yapılacağını belirtir. Tablo 4.2' de ilk 10 en iyi çözüm alındığından dolayı bu çözümlerin hepsi derece (rank) olarak 1 değerine sahiptir. Büyük kalabalık mesafesine sahip çözümler MOGA' ya göre bir sonraki jenerasyonlara katılma olasılıklarını arttırmış olurlar. MOGA' da, grafiğin en ucundaki çözümler için kalabalık mesafesi sonsuz olarak atanır ve daha sonrasında gelen çözümler büyükten küçüğe doğru sıralanır.

Her bir jenerasyonda oluşturulan çözümlerin, çözüm kümesindeki izdüşümleri Şekil 4.7' de verilmiştir. Bu grafikte x ekseni güç tüketimini, y ekseni ise her bir çözüm için katedilmesi

gereken mesafeyi göstermektedir. Kırmızı ile gösterilen noktalar her bir jenerasyonun sahip olduğu en iyi çözümleri göstermektedir. Şekilde kırmızı noktaların her bir iterasyonda (0,0) noktasına yaklaştığı görülür.



Şekil 4.7: Takas yöntemi, Güç-Mesafe grafiği.

4.3.1.2 Mutasyon Fazında Uygulanan Karıştırma Yönteminin Simülasyon Üzerindeki Etkisi

MOGA' nın mutasyon fazında uygulanan "karıştırma yöntemi" sonuçları aşağıdaki başlangıç parametreleri ile gerçekleştirilmiştir.

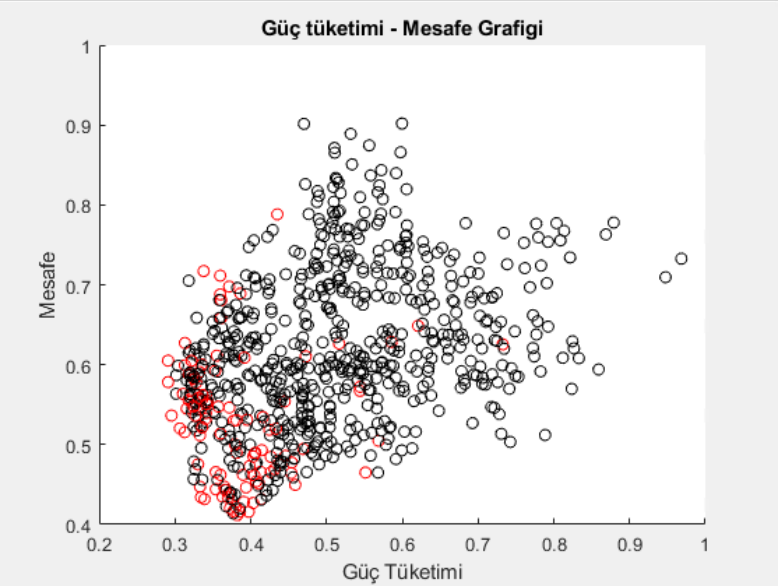
- Görev Sayısı : 7
- Hava Aracı sayısı :3
- Çözüm Popülasyonu : 100
- Maksimum Iterasyon : 50
- Deneme Sayısı :5

Tablo 4.3, Maksimum iterasyon sayısından sonra oluşan en iyi 10 çözümün "derece" ve "kalabalık mesafesi" içeren değerlerini göstermektedir.

Tablo 4.3: Karıştırma yöntemi sonuçları.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[4,3]	[3,3]	[2,1]	[1,1]	[5,2]	[6,2]	[7,3]	1	0.715
[4,3]	[3,3]	[5,1]	[2,1]	[1,2]	[6,2]	[7,1]	1	0.695
[4,3]	[3,3]	[2,1]	[5,3]	[6,3]	[1,1]	[7,1]	1	0.686
[4,3]	[3,3]	[2,1]	[1,1]	[5,2]	[7,2]	[6,3]	1	0.685
[4,3]	[3,3]	[2,1]	[1,1]	[5,2]	[6,2]	[7,3]	1	0.677
[4,3]	[2,1]	[1,1]	[5,3]	[6,3]	[3,2]	[7,1]	1	0.666
[4,3]	[5,2]	[3,3]	[2,1]	[6,1]	[1,2]	[7,1]	1	0.664
[4,3]	[3,3]	[1,3]	[2,1]	[5,2]	[6,2]	[7,2]	1	0.662
[4,3]	[3,3]	[5,1]	[1,1]	[6,2]	[2,2]	[7,1]	1	0.649
[4,3]	[3,3]	[2,1]	[1,1]	[5,2]	[6,2]	[7,2]	1	0.515

Her bir jenerasyonda oluşturulan çözümlerin, çözüm kümesindeki izdüşümleri Şekil 4.8' de verilmiştir.



Şekil 4.8: Karıştırma yöntemi, Güç-Mesafe grafiği.

4.3.1.3 Mutasyon Olasılık Parametresinin Simülasyon Üzerindeki Etkisi

Bu bölümde Şema teoreminde Pm olarak verilen mutasyon olasılığının simülasyon üzerindeki etkisi incelenecektir. Başlangıç parametreleri aşağıdaki gibidir. Mutasyon olasılıkları sırasıyla

0.3, 0.5 ve 0.7 olarak atanmıştır. Tablo 4.4, Tablo 4.5 ve Tablo 4.6 sırasıyla belirtilen mutasyon oranlarının sonuçlarını içermektedir.

- Görev Sayısı : 7
- Hava Aracı sayısı :3
- Çözüm Popülasyonu : 100
- Maksimum Iterasyon : 50
- Deneme Sayısı :5

Tablo 4.4: Mutasyon olasılığı 0.3.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[5,2]	[4,1]	[2,3]	[3,3]	[1,1]	[6,2]	[7,2]	1	Inf
[5,2]	[4,3]	[2,1]	[3,3]	[6,3]	[7,1]	[1,2]	1	0.835
[2,1]	[1,1]	[4,3]	[7,1]	[5,2]	[3,2]	[6,3]	1	0.760
[5,2]	[2,1]	[4,3]	[3,3]	[6,3]	[1,1]	[7,2]	1	0.720
[5,2]	[4,3]	[2,1]	[3,3]	[6,3]	[1,1]	[7,2]	1	0.689
[5,2]	[4,3]	[2,1]	[3,3]	[6,3]	[1,1]	[7,2]	1	0.670

Tablo 4.5: Mutasyon olasılığı 0.5.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[2,1]	[5,3]	[4,2]	[6,2]	[1,1]	[3,3]	[7,2]	1	Inf
[2,1]	[1,1]	[4,3]	[3,3]	[5,2]	[6,1]	[7,1]	1	0.577
[4,3]	[5,2]	[2,1]	[1,1]	[3,3]	[6,2]	[7,1]	1	0.571
[2,1]	[1,1]	[4,1]	[3,3]	[5,2]	[6,3]	[7,1]	1	0.548
[2,1]	[1,1]	[4,3]	[5,2]	[3,3]	[7,1]	[6,2]	2	Inf
[4,3]	[5,2]	[2,1]	[3,3]	[7,1]	[6,2]	[1,1]	2	0.682

Tablo 4.6: Mutasyon olasılığı 0.7.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[2,1]	[1,1]	[5,3]	[6,2]	[7,2]	[4,2]	[3,3]	1	Inf
[4,3]	[3,1]	[2,3]	[1,1]	[5,2]	[6,2]	[7,1]	1	0.618
[4,2]	[3,3]	[2,1]	[1,1]	[5,3]	[6,2]	[7,2]	1	0.559
[5,2]	[4,3]	[3,1]	[6,2]	[7,2]	[1,2]	[2,2]	2	Inf
[4,3]	[5,2]	[6,3]	[1,2]	[7,3]	[2,1]	[3,1]	2	0.965
[5,2]	[4,3]	[3,2]	[6,2]	[7,1]	[2,1]	[1,3]	2	0.815

4.3.1.4 Çaprazlama Fazında Uygulanan Tek Çözüm Yönteminin Simülasyon Üzerindeki Etkisi

MOGA' nın çaprazlama fazında uygulanan "tek çözüm" sonuçları aşağıdaki başlangıç parametreleri ile gerçekleştirilmiştir.

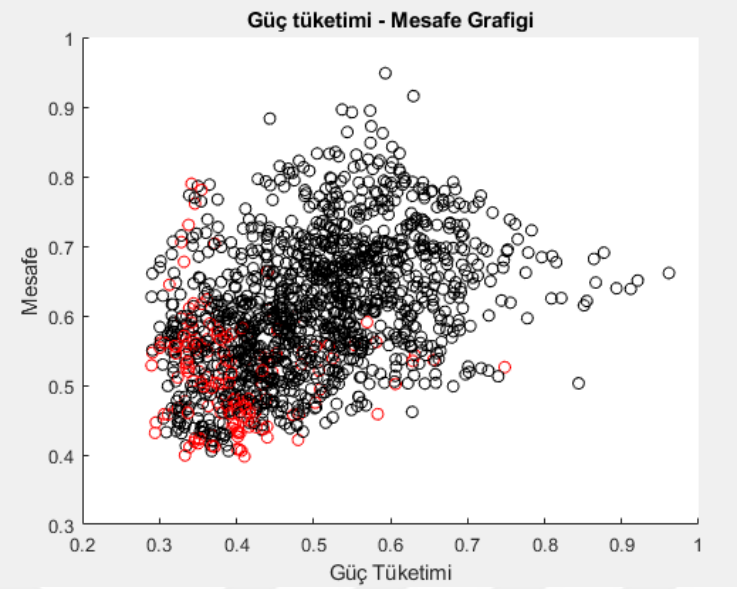
- Görev Sayısı : 7
- Hava Aracı sayısı :3
- Çözüm Popülasyonu : 100
- Maksimum Iterasyon : 50
- Deneme Sayısı :5

Tablo 4.4, maksimum iterasyon sayısından sonra oluşan en iyi 10 çözümün "derece" ve "kalabalık mesafesi" içeren değerlerini göstermektedir.

Tablo 4.7: Tek çözüm yöntemi sonuçları.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[4,3]	[3,3]	[5,2]	[2,1]	[1,1]	[7,1]	[6,1]	1	Inf
[4,3]	[3,3]	[2,1]	[5,1]	[1,2]	[6,2]	[7,3]	1	0.699
[4,3]	[2,3]	[3,1]	[1,1]	[5,2]	[6,2]	[7,3]	1	0.602
[4,3]	[3,3]	[1,1]	[5,1]	[7,2]	[2,1]	[6,2]	1	0.599
[4,3]	[5,2]	[3,3]	[6,2]	[1,1]	[7,2]	[2,1]	1	0.592
[4,3]	[5,2]	[3,3]	[6,2]	[1,1]	[7,2]	[2,1]	1	0.553
[4,3]	[5,2]	[3,3]	[6,2]	[7,1]	[1,2]	[2,1]	1	0.553
[4,3]	[5,2]	[3,3]	[6,2]	[7,1]	[2,2]	[1,1]	1	0.553
[4,3]	[3,3]	[1,2]	[5,1]	[2,1]	[7,1]	[6,1]	1	0.525
[5,2]	[4,3]	[3,3]	[2,2]	[6,1]	[1,3]	[7,1]	2	Inf

Her bir jenerasyonda oluşturulan çözümlerin, çözüm kümesindeki izdüşümleri Şekil 4.9' da verilmiştir.



Şekil 4.9: Tek çözüm, Güç-Mesafe grafiği.

4.3.1.5 Çaprazlama Fazında Uygulanan Çift Çözüm Yönteminin Simülasyon Üzerindeki Etkisi

MOGA' nın mutasyon fazında uygulanan "karıştırma yöntemi" sonuçları aşağıdaki başlangıç parametreleri ile gerçekleştirilmiştir.

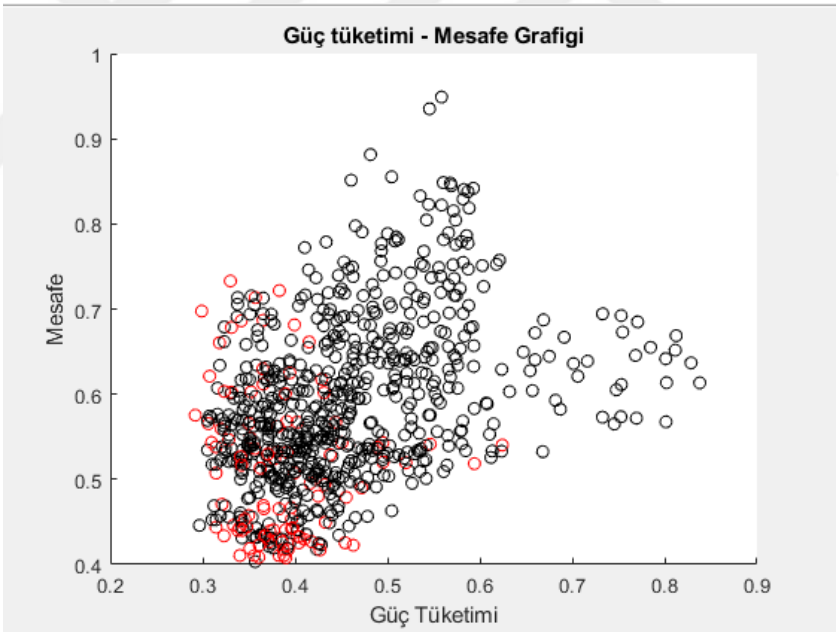
- Görev Sayısı : 7
- Hava Aracı sayısı :3
- Çözüm Popülasyonu : 100
- Maksimum Iterasyon : 50
- Deneme Sayısı :5

Tablo 4.5, Maksimum iterasyon sayısından sonra oluşan en iyi 10 çözümün "derece" ve "kalabalık mesafesi" içeren değerlerini göstermektedir.

Tablo 4.8: Çift çözüm yöntemi sonuçları.

G1	G2	G3	G4	G5	G6	G7	Derece	Kalabalık M.
[2,1]	[4,3]	[3,3]	[5,2]	[1,1]	[7,1]	[6,1]	1	Inf
[2,1]	[6,3]	[5,1]	[3,3]	[1,2]	[4,2]	[7,1]	1	0.616
[2,3]	[4,3]	[1,1]	[7,1]	[3,1]	[5,2]	[6,1]	1	0.564
[2,1]	[4,3]	[3,3]	[7,1]	[1,1]	[5,2]	[6,1]	1	0.564
[2,1]	[4,3]	[3,2]	[5,3]	[6,1]	[7,1]	[1,1]	1	0.564
[2,1]	[4,3]	[5,2]	[7,1]	[1,1]	[6,1]	[3,3]	2	Inf
[4,3]	[6,3]	[7,3]	[5,2]	[2,1]	[1,1]	[3,2]	2	0.738
[4,3]	[5,2]	[2,1]	[7,1]	[1,1]	[3,3]	[6,2]	3	Inf
[5,1]	[2,1]	[4,3]	[3,3]	[1,1]	[7,1]	[6,1]	3	0.769
[2,1]	[4,3]	[3,3]	[5,2]	[6,1]	[1,1]	[7,2]	3	0.711

Her bir jenerasyonda oluşturulan çözümlerin, çözüm kümesindeki izdüşümleri Şekil 4.10' da verilmiştir.



Şekil 4.10: Çift Çözüm, Güç-Mesafe grafiği.

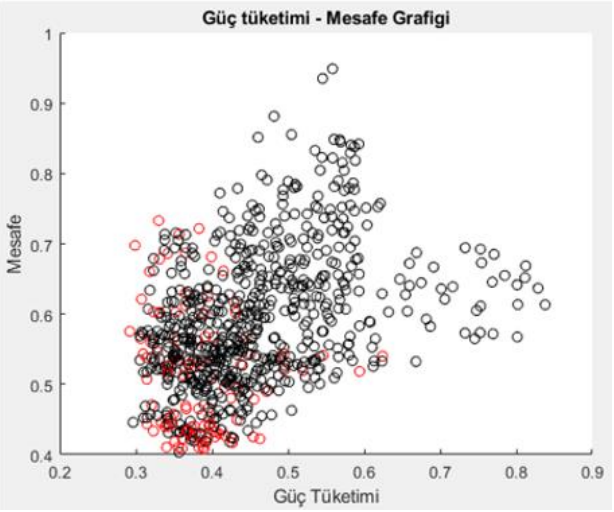
4.4 UYGULANAN YÖNTEMLERİN KAŞILAŞTIRILMASI

Bu bölümde MOGA fazlarından mutasyon ve çaprazlama adımlarında uygulanan farklı yöntemlerin simülasyon üzerindeki sonuçları incelenmiştir. Simülasyon çalışması aşağıdaki ortam ve programlarla yapılmıştır.

- İşletim Sistemi : 64-bit Windows 10
- İşlemci : Intel Xeon 2.30GHz
- RAM : 32GB
- Matlab Versiyonu : 9.6.0.1072779 (R2019a)
- Gazebo Versiyonu : 9.18.0
- ROS Versiyonu : Kinetic

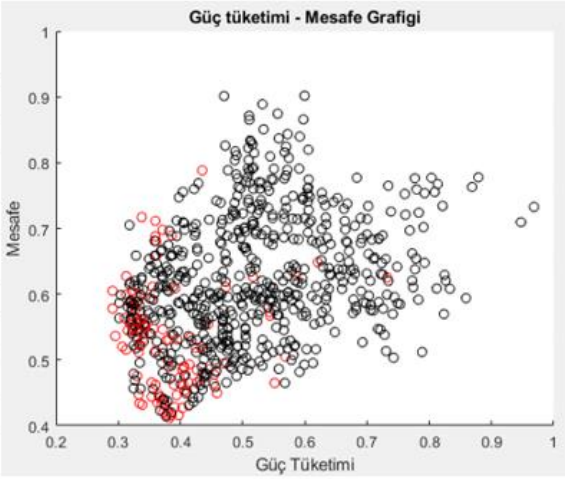
Mutasyon adımında uygulanan takas (swap) yönteminde, maksimum iterasyonun karıştırma (scramble) yöntemine göre daha kısa sürede tamamlandığı gözlemlenmiştir. Ancak derece ve kalabalık mesafesi değerleri açısından karıştırma yönteminde daha iyi sonuçlar elde edildiği gözlemlenmiştir. Şekil 4.10 ve Şekil 4.11' de mutasyon adımında uygulanan takas ve karıştırma yöntemlerinin, derece, kalabalık mesafesi değerleri ve Güç-Mesafe grafiği verilmiştir. Bu iki şekli karşılaştırdığımızda, ortalama kalabalık mesafesi değerinin karıştırma yönteminde daha düşük olduğu görülmektedir. Bu karıştırma yönteminden elde edilen sonuçların daha homojen dağıldığını gösterir. Güç tüketimi – Mesafe grafiklerini karşılaştırdığımızda, her iki yöntemde de mesafe ile ilgili optimizasyonun başarılı bir şekilde gerçekleştiği görülmektedir. Güç tüketimi açısından bakıldığında karıştırma yönteminde birçok çözümün 0.3 değerinin altında olduğu görülür. Sonuç olarak karıştırma yönteminin sürü İHA simülasyonunun daha verimli olduğu gözlemlenmiştir. Mutasyon olasılığı sırasıyla 0.3, 0.5 ve 0.7 olarak simülasyonda uygulanmıştır. En iyi sonuçların mutasyon olasılığının 0.5 olduğu değerde oluştukları gözlemlenmiştir. Mutasyon olasılığının 0.3 olarak değer aldığı simülasyonda, çözümlerin bir noktada toplandığı gözlemlenmiştir. Mutasyon olasılığının 0.7 olduğu durumda ise çözümlerin çok büyük güç tüketimi ve mesafe aralığında değer aldığı gözlemlenmiştir.

Derece	Kalabalık M.
1	Inf
1	0.776
1	0.770
1	0.746
1	0.673
1	0.656
1	0.650
1	0.646
1	0.594
1	0.447



Şekil 4.10: Mutasyon, Takas yöntemi.

Derece	Kalabalık M.
1	0.715
1	0.695
1	0.686
1	0.685
1	0.677
1	0.666
1	0.664
1	0.662
1	0.649
1	0.515

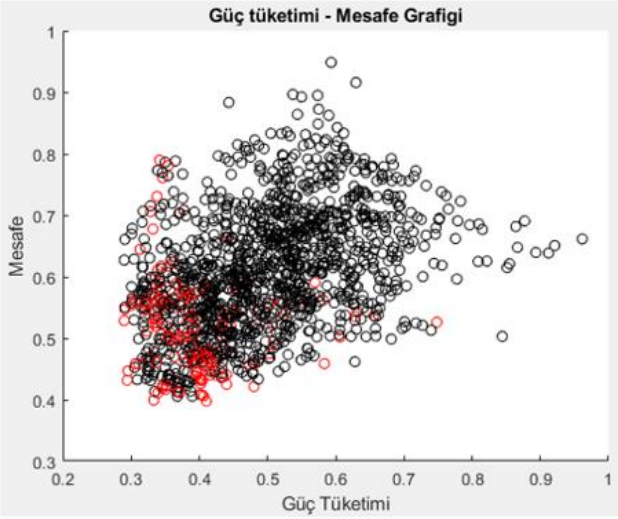


Şekil 4.11: Mutasyon, Karıştırma yöntemi.

Çaprazlama adımında uygulanan metodlar, mutasyon fazında daha iyi sonuçlar verdiği için karıştırma mutasyonu metodu ile birlikte uygulanmıştır. Çaprazlama adımında uygulanan tek çözüm yönteminde maksimum iterasyon sayısının çift çözüm metoduna göre daha uzun sürede tamamlandığı gözlemlenmiştir. Buna karşılık tek çözüm metodunda çözümlerin derece ve kalabalık mesafelerinin daha iyi olduğu gözlemlenmiştir. Şekil 4.12 ve Şekil 4.13' te çaprazlama adımında uygulanan tek ve çift çözüm yöntemlerinin, derece, kalabalık mesafesi değerleri ve Güç-Mesafe grafikleri verilmiştir. Derece açısından karşılaştırma yapıldığında tek

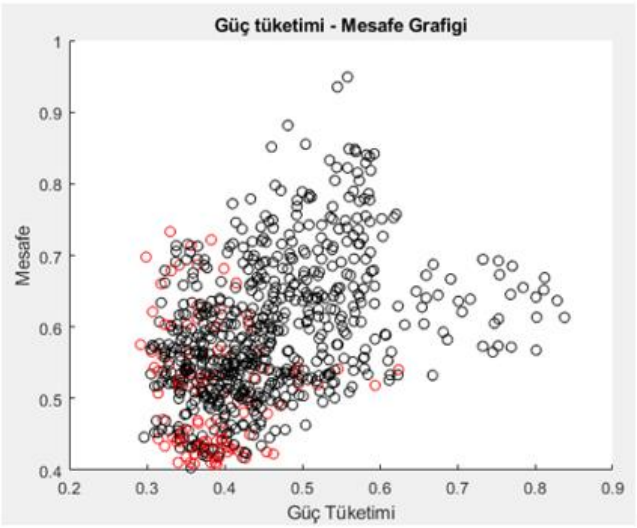
çözümün, çözüm uzayında daha iyi sonuçlar veren çözümlerin mevcut olduğu görülür. Ayrıca tek çözümün son iterasyondan sonra mesafe kullanımı açısından daha optimum bir çözüm sunduğu gözlemlenmiştir.

Derece	Kalabalık M.
1	Inf
1	0.699
1	0.602
1	0.599
1	0.592
1	0.553
1	0.553
1	0.553
1	0.525
2	Inf



Şekil 4.12: Çaprazlama, tek çözüm.

Derece	Kalabalık M.
1	Inf
1	0.616
1	0.564
1	0.564
1	0.564
2	Inf
2	0.738
3	Inf
3	0.769
3	0.711



Şekil 4.13: Çaprazlama, çift çözüm.

5. TARTIŞMA VE SONUÇ

Sürü İHA sistemlerinde hava aracı sayısı arttıkça, yol planlama, sistemde bulunan tüm hava araçlarının kontrolü, hava araçlarının haberleştirilmesi gibi birçok alanda optimize edilmesi gereken konu ortaya çıkmıştır. Bu kapsamda sürü İHA sisteminde, hava aracı sayısı arttıkça kullanılan kaynakların optimum düzeyde tutulması esasen planlama aşamasında yapılan çalışmalara bağlıdır. İyi geliştirilmiş bir planlama algoritması sürü İHA sistemindeki kaynak kullanımını optimum düzeyde tutabilir.

Bu tez çalışmasında sürü İHA sistemlerinin tasarlanmasını kolaylaştıran ve gerekli algoritmaların geliştirilip test edilmesini sağlayan bir simülasyon ortamı önerilmiştir. Sürü İHA sistemlerinin görev planlanmasında kullanılan Genetik Algoritmasının her bir adımında önerilen farklı yöntemler uygulanmış ve simülasyon üzerindeki etkileri karşılaştırılmıştır.

Tez kapsamında yapılan karşılaştırmaya göre mutasyon adımında önerilen yöntemlerden karıştırma (scramble) yönteminin takas yöntemine göre, daha iyi derece ve kalabalık mesafesi değerleri üretebildiği gözlemlenmiştir. Bunun sonucu olarak sürü İHA simülasyonunun daha kısa sürelerde ve daha az sorunla tamamlandığı gözlemlenmiştir. Buna karşılık takas (swap) yönteminin, karıştırma yöntemine göre daha kısa sürede çözüm jenerasyonlarını oluşturduğu gözlemlenmiştir.

Çaprazlama adımında önerilen yöntemlerden tek-çözüm yönteminin çift-çözüm yöntemine göre, daha iyi derece ve kalabalık mesafesi değerleri üretebildiği gözlemlenmiştir. Bunun sonucu olarak sürü İHA simülasyonunun daha kısa sürelerde ve daha az sorunla tamamlandığı gözlemlenmiştir. Buna karşılık çift-çözüm yönteminin, tek-çözüm yöntemine göre daha kısa sürede çözüme ulaştığı gözlemlenmiştir.

Seçilim aşamasından sonra güçlü bireylerin popülasyonda kalmasını sağlayan “Elitizm” yönteminin sürü İHA sistemlerinde uygulanmasının daha iyi sonuçlar üreteceği değerlendirilmektedir.

Bu tez çalışmasının sonucu olarak diğer bir öneri, MOGA yönteminde en iyi sonuçlar arasında rastgele bir çözüm seçmek yerine, en iyi çözümler arasında seçim yapmak için de bazı yöntemlerin geliştirilmesi Sürü İHA sistemlerindeki verimliliği artıracağı değerlendirilmektedir.

Sürü İHA sistemlerindeki karmaşık görevlerin yönetim senaryolarında genellikle bu tez çalışmasında olduğu gibi metasezgisel yöntemler kullanılır. Ancak yüksek hesaplama yükü nedeniyle en iyi çözümün bulunması uzun zaman alabilir. Bu nedenle özellikle sürü İHA sistemlerinin kritik görevlerinde, planlama algoritmasının daha hızlı yakınsayan ve doğru sonuçlar üreten yöntemlerden oluşması gerekir. Kara araçları ile karşılaştırıldığında, hava araçları daha hızlı hareket ederler. Bu nedenle planlama karar algoritmasının gerçek zamanlı gereksinimleri karşılaması gerekir. Genel olarak, sistem optimizasyonunun ve hızının hesaba katıldığı karar verme teknolojilerini gerçekleştirmek, gelecekte hala bir zorunluluktur.

KAYNAKLAR

[1] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, “Acomprehensive review of the covid-19 pandemic and the role of iot, drones, ai, blockchain, and 5g in managing its impact,” IEEE Access, vol. 8, pp. 90 225–90 265, 2020.

[2] Campion, M. Prakash, R. Faruque, Saleh. 2018, UAV Swarm Communication and Control Architectures: A Review. Journal of Unmanned Vehicle Systems. 7. 10.1139/juvs-2018-0009.

[3] Xiaoming Zhang, and Mohsin Ali, “A Bean Optimization-Based Cooperation Method for Target Searching by Swarm UAVs in Unknown Environments” School of Information and Computers, Anhui Agricultural University, Hefei 230031, China

[4] Cao, Y., Fukunaga, A., And Kahng, A. 1997.” Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots”* IEEE/RSJ IROS conference

[5] Dudek, G. Jenkin, M. Milios, E. Wilkes, David. (1993). A taxonomy for swarm robots. 441 - 447 vol.1.

[6] Yogeswaran Mohan, S. G. Ponnambalam “An Extensive Review of Research in Swarm Robotics”, 2009 World Congress on Nature & Biologically Inspired Computing

[7] Thomas H. Labella, Marco Dorigo and Jean-Louis Deneubourg, “Division of Labor in a Group of Robots Inspired by Ants’ Foraging Behavior”, ACM Transactions on Autonomous and Adaptive Systems, vol. 1, no. 1, September 2006, pp. 4-25.

[8] Praveen Kalla, Dr Ramji K, Dr Ravindranath P, “Swarm Home Robots” 2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)

[9] Yara Khaluf, Emi Mathews, Franz J. Rammig, “Self-organized Cooperation in Swarm Robotics” 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops

[10] Y. Zhou, B. Rao and W. Wang, "UAV Swarm Intelligence: Recent Advances and Future Trends," in *IEEE Access*, vol. 8, pp. 183856-183878, 2020, doi: 10.1109/ACCESS.2020.3028865.

[11] Boskovic J , Knoebel N , Moshtagh N , et al. Collaborative Mission Planning & Autonomous Control Technology (CoMPACT) system employing swarms of UAVs[C]// Aiaa Guidance, Navigation, & Control Conference. 2009.

[12] Boskovic J D , Prasanth R , Mehra R K . A Multi-Layer Autonomous Intelligent Control Architecture for Unmanned Aerial Vehicles[J]. Journal of Aerospace Computing, Information, and Communication, 2004, 1(12):605-628.

[13] E. Blasch, "DDDAS Advantages From High-Dimensional Simulation," 2018 Winter Simulation Conference (WSC), 2018, pp. 1418-1429, doi: 10.1109/WSC.2018.8632336.

[14] Gregory R. Madey, M. Brian Blake, Christian Poellabauer, Hongsheng Lu, R. Ryan McCune, Yi Wei “Applying DDDAS Principles to Command, Control and Mission Planning for UAV Swarms” Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA

[15] Zhen Ziyang, Xing Dongjing, Gao Chen “Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm” College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China Department of Automation, Tsinghua University, Beijing, 100084, China

[16] Ziyang Zhen, Yan Chen, Liangdong Wen, Bing Han, “An intelligent cooperative mission planning scheme of UAV swarm in uncertain dynamic environment” College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

[17] Jay Wilhelm, Jonathan Rojas, Gina Eberhart, Mario Perhinschi (2017). Heterogeneous Aerial Platform Adaptive Mission Planning Using Genetic Algorithms, Mechanical Engineering Department, Ohio University, 251 Stocker Center, Athens, OH 45701, USA

[18] Ramirez-Atencia C , R-Moreno M D , Camacho D . Handling swarm of UAVs based on evolutionary multi-objective optimization[J]. Progress in Artificial Intelligence, 2017.

[19] Anam Tahir, Jari Boling, Mohammad-Hashem Haghbayan, Hannu T. Toivonen, Juha Plosila “Swarms of Unmanned Aerial Vehicles — A Survey” in Journal of Industrial Information Integration · October 2019

[20] Mirjalili S. (2019) Genetic Algorithm. In: Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence, vol 780. Chapter 4, Genetic Algorithm.

[21] Bonabeau, E. Dorigo, M. Theraulaz, G. Swarm intelligence: from natural to artificial systems, 1992, Oxford University Press.

[22] 112. Kumar, V. Kumar, D. An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems, 2017

[23] A review on genetic algorithm: past, present, and future Sourabh Katoch1 & Sumit Singh Chauhan1 & Vijay Kumar1, Springer Science+Business Media, LLC, part of Springer Nature 2020

[24] Fred G. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research. Vol. 13(5):533-549.

[25] Holland, J. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 1992,

[26] M. Akbari, P. Asadi, M.K. Besharati Givi , G. Khodabandehlouie, 2014, Artificial neural network and optimization, woodhead.

[27] Vladik Kreinovich, Chris Quintana, Olac Fuentes, 1993, Genetic Algorithms:What fitness Scaling is Optimal?

[28] Farzad Sadjadi "Comparison of fitness scaling functions in genetic algorithms with applications to optical processing", Proc. SPIE 5557, Optical Information Systems II, (22 October 2004);

[29] Shojaeefard , M.H., Behnagh, R.A., Akbari , M. , Givi , M.K.B. and Farhani , F. (2013) Modelling and Pareto optimization of mechanical properties of friction stir welded AA7075/AA5083 butt joints using neural.

[30] Abeywickrama, Hasini & Jayawickrama, Beeshanga & He, Ying & Dutkiewicz, Eryk. (2018). Comprehensive Energy Consumption Model for Unmanned Aerial Vehicles, Based on Empirical Studies of Battery Performance. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2875040.

[31] Automatic tool for Gazebo world construction: from a grayscale image to a 3D solid model 2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France.

[32] Rivera, Zandra B., Marco C. De Simone, and Domenico Guida. 2019. "Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments" *Machines* 7, no. 2: 42.

[33] V. Mayoral *et al*., "The shift in the robotics paradigm — The Hardware Robot Operating System (H-ROS); an infrastructure to create interoperable robot components," *2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2017, pp. 229-236, doi: 10.1109/AHS.2017.8046383.

[34] ROS Website, 2021, https://www.ros.org/blog/ecosystem/ , [27.12.2021].

EKLER

EK 1. Matlab programının ana akışı.

```
while maxGenerationCount~=0
    Generation();
    maxGenerationCount = maxGenerationCount-1;

    maxGenerationCount

    group = {};
    i=1;
    PopSize = size(population(:,1));
    initialRankValue = 1;
    initialRank = 1;
    while i<=PopSize(1)
        initialRankValue = cell2mat(population(i, rankIndex));
        if initialRankValue == initialRank
            group = [group; population(i,:)];
            bAtLeastOneMember = 1;
            i=i+1;
        else
            if bAtLeastOneMember == 0
                i=i+1;
                continue;
            end

            % Her bir jenerasyonda çözümlerin sahip oldu?u güç tüketimi ve
            % kalabalık mesafesinin grafiğe dökümü
            powerFitnessValues = group(:,normPowerFitnessIndex);
            distanceFitnessValues = group(:,normDistanceFitnessIndex);
            if cell2mat(group(1,rankIndex)) == 1
                plot(cell2mat(powerFitnessValues),
cell2mat(distanceFitnessValues), 'ro');
            else
                plot(cell2mat(powerFitnessValues),
cell2mat(distanceFitnessValues), 'ko');
            end
            drawnow;
            initialRank = initialRank+1;
            group = {};
            bAtLeastOneMember = 0;

        end
    end
end

m = 5;

%Simülasyon Derece ve kalabalık mesafesine göre en iyi çözümü kullanarak
başlar.
operateDrones(population(1,:));
```

EK 2. Popülasyonun oluşturulması.

```
function [population] = PopulationCreator(number)

    global population;
    global InitialRank;
    global InitialCrowdingDistance;
    popCandidate = {};
    global nbrOfTasks;

    for j=1:number
        orderTasks = randperm(nbrOfTasks);
        for i=1:nbrOfTasks
            assignment = [orderTasks(i) randperm(3,1)];
            popCandidate = [ popCandidate assignment];
        end
        population = [population ; popCandidate InitialRank ...
                    InitialCrowdingDistance ...
                    GetDistanceFitness(popCandidate) ...
                    0 GetPowerFitness(popCandidate) 0];
        popCandidate = {};
    end
end
```

EK 3. Mesafe uyumluluk değerlerinin hesaplanması.

```
%Mesafe uyumlulu?unun hesaplanması
for i=1:nbrOfTasks
    task = cell2mat(popCandidate(i));
    totalDistance = totalDistance + norm(TCoordinates(task(1),:) ...
                                        - DCoordinates(task(2),:));
    for j=(i-1):-1:1
        compare = cell2mat(popCandidate(j));
        if(compare(2)==task(2))

          totalDistance = totalDistance - norm(TCoordinates(task(1),:) ...
                                                -
          DCoordinates(task(2),:));
                      totalDistance = totalDistance +
          norm(TCoordinates(compare(1),:) ...
                                                -
          TCoordinates(task(1),:));
              break;
        end
    end
end
```

EK 4. Gazebo ortamındaki örnek bir İHA' nın sürülmesi.

```
% Hedefe ulaşmak için uygulanan PD katsayıları
kpz = 1.2;
kdz = 1.3;
kp_zaxis = 1;
angle_diffPrev = 0;
kpx=1;

targetSize = size(UAV1Targets());
if targetSize(1) ~= 0
if size(UAV1Targets(1,:)) ~= 0
    inc_x = TCoordinates(cell2mat(UAV1Targets(1)),1)-UAV1x;
    inc_y = TCoordinates(cell2mat(UAV1Targets(1)),2)-UAV1y;
    inc_z = TCoordinates(cell2mat(UAV1Targets(1)),3)-UAV1z;

else
    % go to the main location :
    inc_x = DCoordinates(1,1)-UAV1x;
    inc_y = DCoordinates(1,2)-UAV1y;
    inc_z = DCoordinates(1,3)-UAV1z;
end

if inc_z < 0
   speed.Linear.Z = 0;
else
    speed.Linear.Z = kp_zaxis*inc_z;
end

if (abs(inc_x) < 1.2) && (abs(inc_y) < 1.2)
   speed.Linear.X = 0.0;
   speed.Angular.Z = 0.0;

   %task finished
   targetSize = size(UAV1Targets());
   if targetSize(2) == 0
       uav1Finished = 1;
   else
       UAV1Targets(:,1) =[];
   end
else

    angle_to_goal = atan2(inc_y, inc_x);
    angle_diff  = angle_to_goal - UAV1theta;

    speed.Angular.Z = kpz*angle_diff + kdz*(angle_diff - angle_diffPrev1);
    angle_diffPrev1 = angle_diff;
    if (abs(angle_diff) > 0.1)
        speed.Linear.X = 0.0;

    else
        if inc_x < 0
            speed.Linear.X = -kpx*inc_x;
        else
            speed.Linear.X = kpx*inc_x;
        end
    end
```

```
send(velPublisher1, speed);
  pause(0.1);
 end
 end
```