T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DRONE SÜRÜSÜ İLE HEDEF TAKİP OPTİMİZASYONU

Mustafa İlker EKMEN

YÜKSEK LİSANS

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Aralık-2020
KONYA

TEZ KABUL VE ONAYI

Mustafa İlker EKMEN tarafından hazırlanan "Drone Sürüsü ile Hedef Takip Optimizasyonu" adlı tez çalışması 25/12/2020 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Akif DURDU	…….
Danışman Prof. Dr. Ömer AYDOĞDU	…….
Üye Doç. Dr. Akif DURDU	…….
Üye Dr. Öğrt. Üyesi Ali ÜNLÜTÜRK	..……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mustafa İlker EKMEN

Tarih: 25/12/2020

ÖZET

YÜKSEK LİSANS TEZİ

DRONE SÜRÜSÜ İLE HEDEF TAKİP OPTİMİZASYONU

Mustafa İlker EKMEN

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ömer AYDOĞDU

2020, 70 Sayfa

Jüri
Prof. Dr. Ömer AYDOĞDU
Doç. Dr. Akif DURDU
Dr. Öğr. Üyesi Ali ÜNLÜTÜRK

Son yıllarda insansız hava araçları sınıfında yer alan Drone'ların askeri, sivil (hobi ve ticari) ve akademik amaçlı kullanımı gittikçe yaygınlaşmaktadır. Askeri alanda çoğunlukla keşif, arama, kurtarma ve saldırı görevlerinde, sivil kullanımda kargo taşımacılığı, görüntü alma, haritalama ve hobi faaliyetlerine yönelik alanlarda, akademik alanda ise yaygın olarak uçuş yönetimi ve kontrol yazılımlarının geliştirilmesi amacıyla kullanılmaktadır. Askeri amaçlı Drone kullanımının en önemli sorunlarından bir tanesi Drone ile komuta merkezi arasında ki mesafe sınırlamasıdır. Bu mesafe sınırlamasını aşabilmek için istenilen koordinatlara hareket eden otonom Drone'lar konusunda bazı çalışmalar yapılmış ve yapılan bu çalışmalar neticesinde başarılı sonuçlar elde edilmiştir. Ancak özellikle askeri amaçlar için tek başına kullanılan otonom Drone'lar belirli sorunlar ile karşı karşıya kaldığında komuta merkezine istenilen veri aktarımını sağlayamamakta ve icra etmesi gereken görevi başarı ile gerçekleştirememektedir. Drone'lara ait bu mesafe problemini aşabilmek için sürü algoritmaları üzerinde çalışmalar yapılmaktadır. Aynı zamanda çoklu Drone etkileşimi ve görev paylaşımı gibi konular üzerinde çalışmalar da oldukça popülerdir. Araştırmacılar özellikle askeri alanda keşif, arama, kurtarma ve saldırı görevleri için Drone'ların sürü halinde hareket edebilmesini sağlayan yeni algoritmalar üzerinde çalışmalarını yoğunlaştırmaktadır. Bu tez çalışmasında, hareketli bir hedefin Drone sürüsü ile en uygun biçimde takibi için, bir amaç fonksiyonu tanımlanmış ve bu amaç fonksiyonuna göre optimizasyon yapan hedef takip optimizasyon algoritmaları geliştirilmiştir. Çalışmada üç farklı optimizasyon algoritması ile, Drone sürüsü için tanımlanmış hedef takip ve imha görevleri Unity simülasyon ortamında test edilmiştir. Yürütülen çalışmada, Drone sürüsünü güden Parçacık Sürü Optimizasyonu (PSO), doğada arıların birlikte hareket ederek yaptıkları davranışların ortaya çıkardığı Yapay Arı Kolonisi Algoritması (YAK) ve doğadaki karıncaların davranışları incelenerek ortaya çıkarılan Karınca Kolonisi Algoritması (KKA) ele alınmıştır. Tanımlanan görev sırasında sürüdeki Drone'lardan bir tanesi zarar görse dahi diğer Drone'lar kullanılan algoritma yapısına göre en uygun pozisyonu alarak görevlerine devam edebilmektedir. Simülasyon çalışmasında Drone sürüsünün bir istasyondan kalkarak bir aracı takip emesi, dairesel bir yörüngede araç üzerinde konumlanması ve aracın imha edilmesi örnek görevi tanımlanmıştır. Bunun için Unity simülasyon ortamında bir adet hedef araç ve istenilen sayı da Drone'un kalkabileceği bir yer istasyonu tasarlanmıştır. Aracın konum bilgileri uydu üzerinden alındığı varsayılarak Drone'ların hedef araca göre konumları belirlenen optimizasyon algoritmaları ile sağlanmıştır. Çalışmada, optimizasyon algoritmalarının performansları görev için tanımlanan amaç fonksiyonu temel alınarak kıyaslanmıştır.

Anahtar Kelimeler: Amaç Fonksiyonu, Drone sürüsü, Hedef Takip, Karınca Kolonisi Algoritması, Parçacık Sürü Optimizasyonu, Yapay Arı Kolonisi Algoritması

ABSTRACT

MS THESIS

TARGET TRACKİNG OPTİMİZATİON WİTH DRONE SWARM

Mustafa İlker EKMEN

Konya Technical University
Graduate Education Institute
Department of Electrical and Electronics Engineering

Advisor: Prof. Dr. Ömer AYDOĞDU

2020, 70 Pages

Jury
Prof. Dr. Ömer AYDOĞDU
Assoc. Dr.Akif DURDU
Dr. Lec. Ali ÜNLÜTÜRK

In recent years, the use of drones, which are in the class of unmanned aerial vehicles, for military, civil (hobby and commercial) and academic purposes has become increasingly common. It is mostly used in reconnaissance, search, rescue and attack missions in the military field, cargo transportation in civil use, imaging, mapping and hobby activities, and in the academic field for the development of flight management and control software. One of the most important problems of using military drones is the limitation of the distance between the Drone and the command center. In order to overcome this distance limitation, some studies have been carried out on autonomous drones that move to the desired coordinates and successful results have been obtained as a result of these studies. However, when autonomous drones, which are used alone for military purposes, are faced with certain problems, they cannot provide the desired data transfer to the command center and cannot successfully perform the task they need to perform. Studies are carried out on herd algorithms to overcome this distance problem of drones. It is also very popular to work on topics such as multiple Drone interaction and task sharing. Researchers are focusing on new algorithms that enable drones to move in swarm, especially for reconnaissance, search, rescue and attack missions in the military field. In this thesis, an objective function has been defined for the most appropriate tracking of a moving target with the drone swarm and target tracking optimization algorithms have been developed that optimize according to this goal function. In the study, target tracking and destruction missions defined for the drone swarm were tested in Unity simulation environment with three different optimization algorithms. In the study, Particle Swarm Optimization (PSO), which drives drone swarm, Artificial Bee Colony Algorithm (YAK), which reveals the behaviors of bees by acting together in nature, and Ant Colony Algorithm (KKA), which is revealed by examining the behavior of ants in nature, are discussed. Even if one of the Drones in the herd is damaged during the defined mission, other Drones can continue their duties by taking the most appropriate position according to the algorithm structure used. In the simulation study, the sample task of the drone herd getting up from a station and following a vehicle, positioning it on the vehicle in a circular orbit and destroying the vehicle was defined. For this purpose, a target vehicle and a ground station where the desired number of Drones can take off were designed in the Unity simulation environment. Assuming that the location information of the vehicle is received via satellite, the position of the drones with respect to the target vehicle has been provided by the determined optimization algorithms. In the study, the performances of the optimization algorithms were compared based on the objective function defined for the task.

Keywords: Objective Function, Drone Swarm Target Tracking, Ant Colony, Particle Swarm, Artificial Bee Colony

ÖNSÖZ

Bu çalışmada doğadaki canlı davranışlarından esinlenilerek oluşturulmuş Parçacık Sürüsü, Yapay Arı Kolonisi ve Karınca Kolonisi Optimizasyon Algoritmaları Drone'ların sürü şeklinde birlikte yaptıkları uçuş görevlerine entegre edilmiştir. Bunun için Drone'ların matematiksel modeli elde edilmiş, görsel bir simülasyon ortamı oluşturulmuş ve optimizasyon algoritmalarının etkinliği ortaya konulmuştur.

Tez çalışmamda emekleri olan danışman hocam Prof. Dr. Ömer AYDOĞDU'ya, lisans dönemimde danışmanlığımı yapan ve her zaman desteklerini esirgemeyen Doç. Dr. Akif DURDU'ya, yüksek lisans eğitim dönemimde her türlü kolaylığı sağlayan Kahraman KÜÇÜK'e ve her zaman desteklerini yanımda hissettiğim aileme teşekkür ederim.

Muatafa İlker EKMEN
KONYA-2020

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

a_F	:	İvme
a	:	Araç Hızına Göre Oluşabilecek Ek Hasar
β	:	Yol Katsayısı
b	:	İtme Faktörü
c_1	:	Yerel en iyi katsayı
c_2	:	Küresel en iyi katsayı
d	:	Sürüklenme faktörü
e	:	Vektör
$Error_x$	:	Konum Hatası
$Error_y$	:	Konum Hatası
F_i	:	Kaldırma kuvveti
fit_i	:	İterasyonun en iyi pozisyonu
$f(x)$	:	Amaç fonksiyonu
$g(x)$	:	Küresel en iyi vektör
I	:	Atalet etkisi
it	:	İterasyon sayısı
J	:	Rotorun ataleti
l	:	Rotorla drone arasında ki mesafe
$L_{x,y,z}$	:	Açısal momentum
m	:	Drone sayısı
$p(x)$	:	Yerel en iyi vektör
th	:	Toplam hasar
$U_{1,2,3,4}$		Motorun giriş gerilimleri
W	:	Atalet etkisi
w	:	Açısal Hız
x	:	Drone'nun doğrusal pozisyonu
x_i	:	i iterasyonda ki konum
$x_n[t]$	:	Mevcut pozisyon vektörü
x_p	:	Parçacığın x ekseni koordinatı
y	:	Drone'nun doğrusal pozisyonu
y_p	:	Parçacığın y ekseni koordinatı
z	:	Drone'nun doğrusal pozisyonu
Ω_i	:	Pervanenin açısal hızı
$\Phi_{i,k}$	:	0 ile 1 arasında rastgele bir sayı
ϕ	:	Yalpalama açısı(roll-phi)(rad)
θ	:	Yunuslama açısı(rad)
Ψ	:	Sapma açısı(rad)

Kısaltmalar

AEPS	:	Arı Evrimsel Parçacık Sürüsü
ARP	:	Araç Rotalama Problemi
AYP	:	Araç Yönlendirme Problemi
İHA	:	İnsansız Hava Aracı
İSHA	:	İnsansız Savaşan Hava Aracı
KKA	:	Karınca Kolonisi Algoritması
KYAK	:	Kaotik Yapay Arı Kolonisi Algoritması
LQR	:	Lineer Karesel Regülatör
MPSO	:	Modifiye Parçacık Sürü Optimizasyonu
PSO	:	Parçacık Sürü Optimizasyonu
RPSO	:	Robotik Parçacık Sürü Optimizasyonu
SLPSO	:	Kendinden Uyarlamalı Parçacık Sürü Optimizasyonu
VRP	:	Araç Yönlendirme Problemi
YAK	:	Yapay Arı Kolonisi

1. GİRİŞ

Drone'lar dikey kalkış ve iniş yapabilen kullanımı pratik insansız hava araçlarıdır. Kullanım kolaylığı sayesinde popülaritesi giderek artmaktadır. Sivil ve askeri alanlardaki kullanımı günümüz dünyasında büyük önem arz etmektedir. Özellikle savaş durumlarında keşif veya saldırı amaçlı, sivil kullanımında ise arama-kurtarma gibi görevlerde kullanımı yaygındır. Drone'ların giderek yaygınlaşması onlara görev bazlı yeni özellikler eklemeyi mecbur kılmıştır. Örneğin bazı durumlarda Drone'ların otonom olarak uçması gerekebilir veya gerçekleştireceği göreve göre farklı özelliklere sahip olması istenebilmektedir.

Drone'lar iki farklı şekilde kullanılabilmektedir. İlk kullanım yöntemi, bir operatör ve yönlendirme kumandası vasıtasıyla Drone'ların istenilen görevleri yerine getirmesidir. Ancak bu kullanım şeklinde operatörün dikkati ve el hassasiyeti önemli olduğu için Drone'nun istenilen görevi yerine getirmesi esnasında birçok aksaklıklar meydana gelebilmektedir. Bu durum özellikle Drone'un kritik görevler için kullanılması esnasında arzu edilen başarıyı gerçekleştirememesine neden olabilmektedir. Drone'ların ikinci kullanım yöntemi ise, herhangi bir operatör olmaksızın, görev tanımlı algoritmalar sayesinde istenen görevleri tamamen otonom olarak gerçekleştirebilmesidir.

Kritik askeri operasyonlarda tek bir Drone'un görev alması yerine otonom olarak birlikte hareket edebilen ve yol planlaması gerçekleştirebilen Drone sürüsü sistemleri yaygın olarak kullanılmaktadır. Sürü Drone uygulamalarının en kritik aşaması yol planlamasının başarılı bir şekilde gerçekleştirilmesidir.

Yol planlama hesaplamaları yapılırken, edinilmesi gereken bilgiler araç üzerinde mevcut olan sensörler sayesinde anlık olarak alınmaktadır. Gerçek zamanlı yol planlama uygulamalarında süreç 3 aşamadan meydana gelmektedir. İlk olarak, yol planlama problemi bir optimizasyon problemi haline getirilir (Ahmadzadeh ve Ghanavati). İkinci aşamada problemin amacına ve yol planlaması yapılan bölgede olan veya oluşacak engellere göre en uygun amaç fonksiyonu tasarlanır. Üçüncü ve son aşamada, uygulama üzerinde kullanılacak olan algoritmanın yol planlama çalışmasında başarılı olup olmadığı gözlemlenir. Bu süreçte yol planlaması yapan aracın aldığı konumlar ve görülmesi gereken diğer ölçümler sürekli kayıt altına alınmalıdır.

Günümüzde yol planlama problemlerinde kullanılmak için doğada meydana gelen birçok canlı davranışı incelenerek ele alınmıştır. Doğal yaşamda ki sistemler

canlıların sosyal yaşamlarında ki çevresi ve diğer canlı organizmalarla oluşturdukları ilişkiye dayanmaktadır. Bu ilişkiye göre meydana gelen davranış biçimlerine "sürü zekâsı" ismi verilmektedir. Daha farklı bir ifade ile sürü zekâsı, kendi kendini idare edebilen yapıdaki basit bireyler topluluğunun ortak hareket ederek bir zekâ geliştirmesi olarak da tanımlanabilir. Sürü zekâsı temelli yöntemler gün geçtikçe geliştirilmekte ve otonom kontrol sistemlerin gelişmesine ve ilerlemesine katkıda bulunmaktadır. Sürü sistemi bir diğer adıyla otonom sistemler karmaşık ve farklı çevrelere kısa süre içerisinde uyum sağlamasından dolayı akıllı sistemler olarak nitelendirilebilir. Bundan dolayı otonom sistemlerin yol planlama problemleri bir optimizasyon problemi olarak ele alınabilir. Optimizasyon problemlerinde yaygın olarak sürü zekasına dayalı yöntemler kullanılmaktadır. Son yıllarda sürü zekâsı; ticaret, bilim ve mühendislik gibi farklı alanlarda çalışan araştırmacılar arasında giderek artan bir ilgi görmektedir. Arıların besin bulma ve kovanlarıyla ilgili davranışları, karıncaların yiyeceklerini en kısa yoldan yuvalarına taşımaları sürü zekasının en önemli örnekleridir. Problemin karmaşıklığından dolayı, nümerik yöntemlerle sonuca ulaşmanın zor olduğu durumlarda sürü zekasına dayanan çeşitli optimizasyon teknikleri ile karmaşık problemlerin çözülmesi mümkündür.

Bu tez çalışmasında doğadaki canlı davranışları incelenerek elde edilmiş Parçacık Sürü Optimizasyonu (PSO), Yapay Arı Kolonisi (YAK) Algoritması ve Karınca Kolonisi Algoritması (KKO) ele alınmıştır. Bu optimizasyon algoritmaları, Drone sürüsü için oluşturulan simülasyon ortamında belirlenmiş hedef takip sistemine göre test edilmiştir. Test sonucu elde amaç fonksiyonu değerleri göz önüne alınarak optimizasyon yöntemlerinin performansları birbirleri ile karşılaştırılmıştır.

Gerçekleştirilen tez çalışmasında yer alan Drone'lar birbirleriyle haberleşen bir ağ sistemi yapısına sahiptir. Drone'lar arasındaki haberleşmelerden ve Drone'ların çevre ile etkileşimlerinden arzu edilen bir toplu davranışın ortaya çıktığı varsayılmaktadır. Bu yaklaşım, sürü zekâsı alanında ve ayrıca sürü davranışının meydana geldiği doğadaki böcek, karıncalar ve diğer alanların biyolojik çalışmalarından ortaya çıkarılmıştır.

Sürü Drone'ların araştırılması; Drone'ların tasarımı, kontrolü ve sürü davranışlarının incelenmesini kapsamaktadır. Nispeten basit bireysel kurallar ile çok sayıda karmaşık sürü davranışı üretilebilir. Kilit bileşen, sürekli bir geri bildirim sistemi oluşturan grup üyeleri arasındaki iletişimdir. Sürü davranışı, diğerleriyle iş birliği içinde bireylerin sürekli değişimini ve tüm grubun davranışını içerir. Özellikle doğada sürü

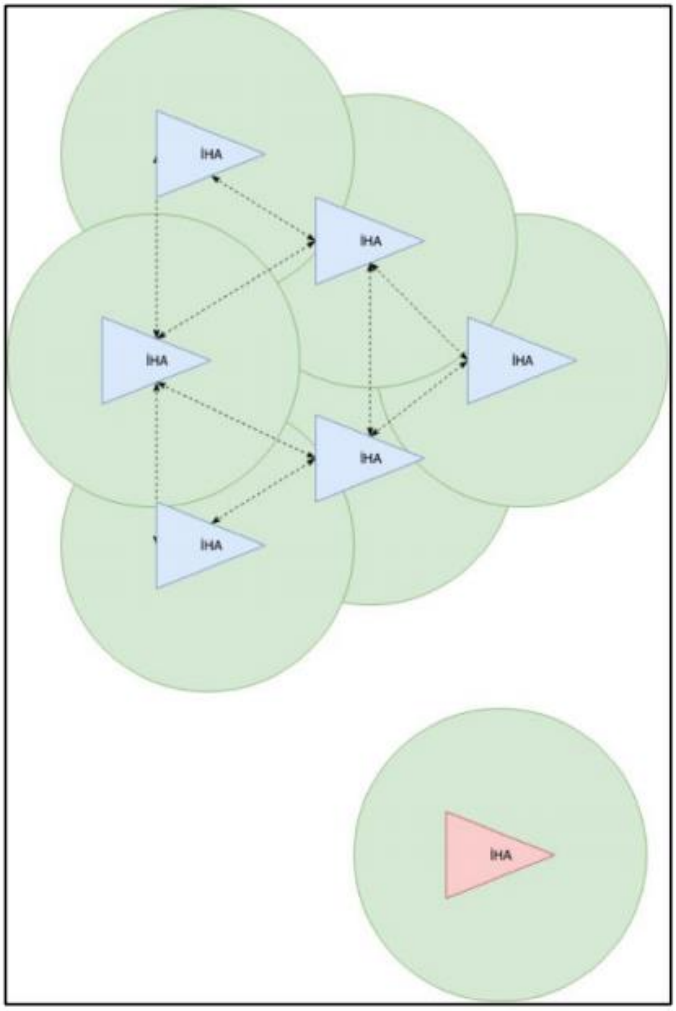
olarak davranış sergileyen ve belirli hedef noktalara çoklu şekilde yaklaşım gösteren canlılar, çoklu robot etkileşimi için temel referans kaynağı oluşturmaktadır. Literatürde en yaygın kullanılan sürü algoritmalarını aşağıda ki gibi sıralayabiliriz.

- Parçacık Sürü Algoritması,
- Yapay Arı Kolonisi Algoritması,
- Karınca Kolonisi Algoritması,
- Ateş Böceği Sürü Optimizasyonu,
- Yapay Balık Sürü Algoritması.

Ancak sürü algoritmaları sadece bunlarla kısıtlı değildir. Doğada birçok canlı kendi aralarında incelenerek farklı algoritma yapılarına referans olmuşlardır. Şekil 1.1'de ortak bir yol oluşturarak hareket eden Karınca Kolonisi görülmektedir.

Şekil 1.1. Karınca Kolonisine bir örnek(Anonim, 2019)

Drone sürüsü çalışmalarının gerçek zamanlı uygulamalarında her Drone'da iletişim modülü bulunmak zorundadır. İletişim modülleri belirli bir alan içerisine Drone'lar arasında veri aktarımı yapabilmeyi sağlar. Bu da sürü olarak hareket eden her Drone'nun kendi iletişim zinciri içerisindeki diğer Drone'lara veri aktarıp alabilme özelliğine sahip olmasını sağlar. Şekil 1.2'de iletişim modülünün çalışma mantığı gösterilmektedir.



Şekil 1.2. Örnek İHA iletişim haberleşmesi (Teknofest Teknik Şartnamesi, 2020)

1.1 Tezin Amacı

Bu tezin amacı, birden fazla Drone'nun zemin olarak referans alınan yol üzerinde hedef nokta olarak belirlenmiş hareketli kara aracını takip edecek şekilde otonom konumlandırılması ve hedefi en kısa sürede etkisiz hale getirmesidir. Bunun için ilk olarak simülasyon ortamında Parçacık Sürü Algoritması, Yapay Arı Kolonisi Algoritması ve Karınca Kolonisi Algoritması ile Drone Sürüsü hedef takip düzeni optimize edilecektir. Bu algoritmalar yol üstünde ki kara aracını referans alarak Drone'ların ne kadar kısa sürede kara aracının etrafında oluşturulacak konumların meydana gelmesinde fayda sağlayacaklardır. Algoritma türlerine göre yapılan konumlandırmadan sonra hedef noktaya belirli sayıda simülasyon ortamında puan

atışı yapılacaktır. Uzaklık ve hedefe verilen hasara göre elde edilen puana bakılarak algoritmalar arasında başarı performansları karşılaştırılacaktır.

1.2 Tezin Önemi

Drone sürülerinde optimal güdüm, gerçekleştirilmesi istenen amaca ulaşabilmek için çok önemlidir. Tez kapsamında simülasyon ortamında sürü algoritmalarının Drone sürülerinin güdümünde kullanımı amaçlanmıştır. Böylece otonom hareket edebilen Drone sürülerine yapay zekâ tabanlı karar verme mekanizması kazandırılarak, amaç fonksiyonunun en iyi biçimde gerçekleştirilmesi sağlanacaktır. Sürü algoritmaları içerisinde kullanılan birçok optimizasyon algoritması mevcuttur. Tez içerisinde bu sürü algoritmalarının başarı durumu karşılaştırılarak sonuçları incelenecektir.

Sürü algoritmaları, hayvan sürülerinin doğa üzerindeki davranışları incelenerek ortaya konmuş algoritma yöntemleridir. Sürüde ki canlılar birbirleri ile haberleşerek ortak bir amaç için hareket edebilmekte ve bulundukları ortamda sorunlara çözüm bulabilmektedir.

Tezin çıktıları aşağıdaki gibi özetlenebilir;

- Otonom Drone'ların birbirleriyle etkileşimli görev yapabilme sonuçları ortaya çıkacaktır.
- Drone sürüsü için ortak bir amaç fonksiyonu tanımlanarak görev başarı ölçütü ortaya konacaktır.
- Parçacık Sürü Algoritması (PSO), Karınca Kolonisi Algoritması (KKA), Yapay Arı Kolonisi (YAK) Algoritmasının başarı durumları karşılaştırılacaktır.
- Farklı senaryolar için PSA, KKA ve YAK'ın görev hızları incelenecektir.
- Farklı sayıdaki Drone sürücü için algoritmaların performansı görülebilecektir.

1.3 Tezin Yapısı

Yapılan çalışma iki aşamadan meydana gelmektedir. İlk aşamada Drone'ların matematiksel olarak doğrusal modelleri elde edilmektedir. Elde edilen doğrusal model durum uzayı formunda ifade edilerek simülasyon için hazır hale getirilmiştir.

İkinci aşamada ise belirlenen senaryoya göre, üç farklı optimizasyon algoritması için Drone sürüsünün hareketleri incelenmiş ve algoritmaların karşılaştırılmaları yapılmıştır. Senaryoya göre, Unity ortamında hazırlanan simülasyon çalışmasında takip edilecek hareketli hedef nokta, hareket halindeki bir araba ile temsil edilmiştir. Oluşturulan bu ortam içerisinde Drone'ların sürü halinde kalkış yaptığı yer istasyonu ve arabanın hareket edeceği yollar simülasyon ortamında oluşturulmuştur. Şekil 1.3'de tasarlanan simülasyon ortamından bir kısım gösterilmektedir. Drone'ların yer istasyonundan kalkış yapmasıyla birlikte her 3 saniyede bir yeni iterasyona geçilerek iterasyon sayısı artırılmaktadır. Bu sayede her iterasyonda Drone'ların konumlarının nereye geldiği ve araç ile aralarında ki mesafenin ne kadar olduğu kolaylıkla ölçülmektedir. Drone'lar ve hedef araç istenilen hızlarda simülasyon ortamında test edilebilmektedir.

Şekil 1.3. Tasarlanan simülasyon ortamından bir görüntü

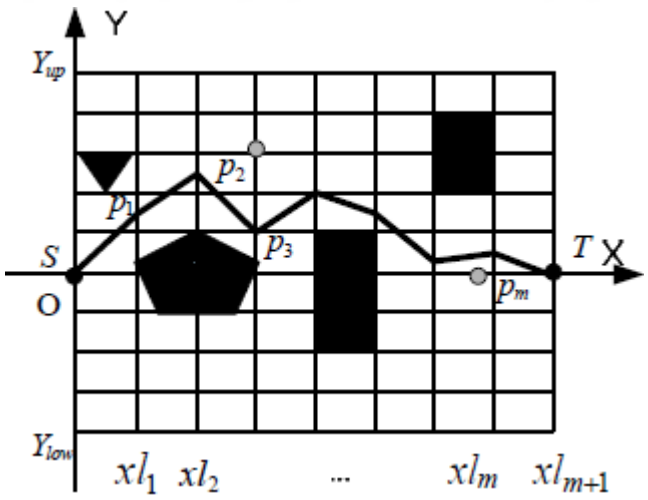
Tez 1. Bölüm olan Giriş ile birlikte 6 bölümden meydana gelmektedir. Buna göre;

- 2. Bölümde, geniş bir literatür taraması verilmiş ve kaynak araştırması yapılmıştır. Burada, son yıllarda yapılan benzer çalışmaların özetleri sunulmuştur.
- 3. Bölümde, tez çalışmasında kullanılan algoritmalar ve simülasyon ortamı detaylı şekilde anlatılmıştır.
- 4. Bölümde, Drone'nun matematiksel modelinin çıkarımı anlatılmaktadır.
- 5. Bölümde, tez çalışması için gerçekleştirilen uygulama ve sonuçları açıklanmaktadır.
- 6. Bölüm ise tezin sonuç kısmıdır.

2. KAYNAK ARAŞTIRMASI

Yapılan literatür taramasında, konu ile ilgili çalışmaların son yıllarda yoğunlaştığı görülmektedir. Aşağıda konu ile ilgili önemli çalışmaların özet bilgileri verilmiştir.

Gong ve arkadaşları (2011), tehlikeli kaynakların bulunduğu bir ortamda robotların geçeceği yolu planlamayı hedefleyen parçacık sürüsü optimizasyonuna dayanan küresel bir yol planlama yaklaşımı üzerinde durmuşlardır. İlk başta, mobil robotun bulunduğu ortamın haritasına bağlı olarak iki parametreli optimizasyon modeli, yani uzunluk ve tehlike derecesi ile ilgili bir çoklu amaç fonksiyonu tanımlamışlardır. İkinci olarak, modeli çözmek için geliştirilmiş çok amaçlı bir parçacık sürüsü optimizasyon algoritması geliştirmişlerdir. Algoritmalarının çalışmadaki performansını arttırmak için, uygulanabilir çözümlerin yanı sıra mümkün olmayan çözümleri de kaydederek farklı bir çalışma benimsenmişler ve sürü parçacıkları arasında küresel lideri ya uygulanabilir çözümler çalışmasından ya da mümkün olmayan çözümden seçmişlerdir. Paylaştıkları simülasyon sonuçları algoritmalarınının başarısını doğrulamaktadır. Şekil 2.1'de Gong ve arkadaşlarının üzerinde çalıştığı sistemin çevre modeli gösterilmektedir (Gong ve ark , 2011).



Şekil 2.1. Örnek Çevre Modeli(Gong, 2011)

Li ve Chou (2018), zorlu bir optimizasyon problemi olarak, mobil robot için yol planlaması üzerinde çalışmışlardır. Bu çalışmalarında, bu problemi çözmek için farklı öğrenme stratejilerine sahip kendinden uyarlamalı bir öğrenen parçacık sürüsü optimizasyonu (SLPSO) önermişlerdir. İlk olarak, yol planlama problemini minimizasyon için çok amaçlı optimizasyon problemine dönüştürmüşlerdir ve üç hedefi

göz önünde bulundurarak amaç fonksiyonunu formüle etmişlerdir. Daha sonra, parçacık sürüsü optimizasyonunun (PSO) arama yeteneğini artırmak için optimizasyon sürecinin farklı aşamalarında kendi kendini uyarlayan öğrenme mekanizması geliştirmişlerdir. Ayrıca, oluşturulan yolların fizibilitesini arttırmak için, her parçacığın hızını ve pozisyonunu kısıtlamak amacıyla yeni ihlal şemalarını da uygulamışlardır. Son olarak, sırasıyla simüle edilmiş bir robot ve gerçek bir robot ile deneyler yapmışlar ve sonuçlar, SLPSO'nun mobil robot yolu planlama probleminin çözümünde uygulanabilirliğini ve etkinliğini göstermiştir. Şekil 2.2'de Li ve Chou'nun Mobil Robot Yol Planlaması için oluşturdukları simülasyon ortamı gösterilmektedir (Chou ve Li, 2018).

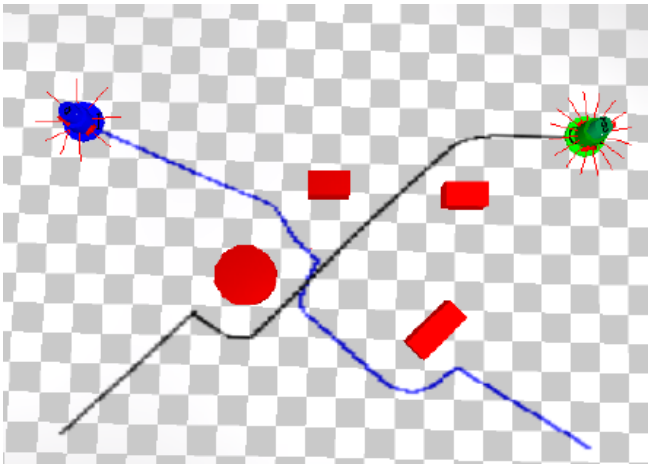
Şekil 2.2. Mobil Robot Yol Planlaması için Oluşturulmuş Simülasyon Ortamı(Chou ve Li, 2018)

Lee ve arkadaşları (2018), Quadcopter sürüsü için yeni bir arama yöntemi önermişlerdir. Önerilen yöntemde, doğadaki sürülerin davranışlarından esinlenilerek Quadcopterlerin etkili bir şekilde çevredeki hedefleri araması sağlanmıştır. Birbirleriyle iletişim kurarak Parçacık Sürü Optimizasyonu algoritmasını dinamik bir model kullanılarak uygulamışlardır. Ek olarak, oluşturdukları çevreyi Quadcopterlerin davranışlarıyla gerçek zamanlı olarak kullanmışlardır. Önerilen yöntemin etkisini, tekrarlanan test simülasyonları ile doğrulamışlardır. Sonuçlar, önerilen yöntemin oldukça pratik, doğru ve sağlam olduğunu göstermektedir (Lee ve ark, 2018).

Couceiro ve Ferraira (2011) ve arkadaşları, çalışmalarında biyolojik ilham tekniklerini çoklu etki alanına uyarlamak için sırasıyla RPSO (Robotik PSO) ve RDPSO (Robotik DPSO) olarak adlandırılan iki Parçacık Sürüsü Optimizasyonu (PSO) ve Darwinian Parçacık Sürüsü Optimizasyonu (DPSO) uzantılarını önermişlerdir. Bu yeni algoritmalar, dağıtılmış bir keşif görevi gerçekleştiren simüle edilmiş robot grupları için

önerilmiştir. Sosyal dışlanma ve sosyal içerme kavramları RDPSO algoritmasında yerel optimizasyondan kaçma yeteneğini artıran bir "ceza-ödül" mekanizması olarak kullanılmaktadır. Simüle edilmiş bir ortamda elde edilen deneysel sonuçlar, biyolojik ve sosyolojik çalışmanın, optimizasyon sorunlarının (ör. Arama ve kurtarma) robotik uygulamalarda zorluklarını karşılamak için yararlı olabileceğini göstermişlerdir (Couceiro ve Ferraira, 2011).

Ahmadzadeh ve Ghanayati (2012), bilinmeyen ortamlarda bir mobil robotun navigasyonu için akıllı bir yaklaşım önermektedirler. Optimizasyon problemlerinin uygun çözümlerini bulmak için Parçacık Sürüsü Optimizasyonu (PSO) yöntemini kullanmışlardır. İlk başta robot navigasyon problemi optimizasyon problemine dönüştürülür. Daha sonra PSO yöntemi, uygun minimum değeri bulmak için çözüm aralığını hedefin konumuna göre arar. PSO'daki her parçacık için bir amaç fonksiyonu hesaplanır. Algoritmanın her bir tekrarında, parçacığın küresel en iyi pozisyonu seçilir ve robot hedefe ulaşmak için bir sonraki hesaplanan noktaya hareket eder. Pratik olması için Robot'un sensörleri ile sınırlı bir çevre yarıçapındaki engelleri tespit edebileceği varsayılmaktadır. Ortamın dinamik olması sayesinde engeller düzeltilebilir veya taşınabilir olarak ayarlanmıştır. Şekil 2.3'de Ahmadzadeh ve Ghanavati'nin PSO ile hareketli engeller içeren çevre simülasyonu gösterilmektedir (Ahmadzadeh ve Ghanavati, 2012).



Şekil 2.3. PSO ile Hareketli Engeller İçeren Çevre Simülasyonu(Ahmadzadeh ve Ghanavati, 2012)

Bhagade ve Puranik (2012), seyahat eden satıcı sorunu için Yapay Arı Kolonisi Optimizasyonu'nu(YAK) önermişlerdir. Bu çalışmalarında, seyahat eden satıcı sorunu

VRP için en yakın komşu yöntemi kullanarak optimize etmişlerdir ve yapay arı koloni algoritması ile karşılaştırılan değerlendirme sonuçlarını sunmuşlardır. İzlenen yaklaşım, hedefe doğru ilerlemek için en kısa yolu en kısa sürede bulmak için en iyi sonuçları vermiştir. Böylece tur uzunluğu ile en uygun mesafe daha etkili bir şekilde elde etmişlerdir (Bhagade ve Puranik, 2012).

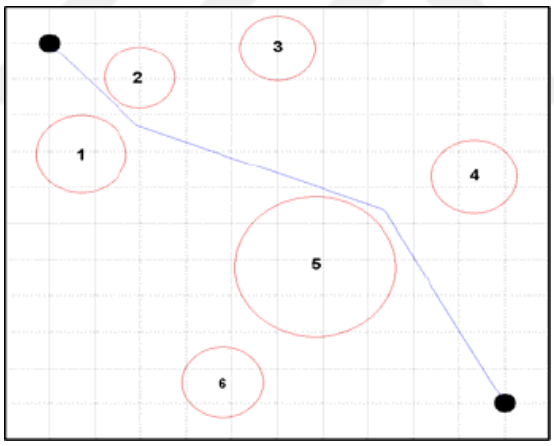
Akay ve Karaboga (2012) çalışmalarında, Yapay Arı Algoritmasının geniş bir alandaki sorunlarını çözmek için modelleri simüle eden birçok algoritma yöntemi önermişlerdir. Yapay Arı Kolonisi algoritması, bal arısı kolonilerinin yiyecek arama davranışını simüle eden istihbarat tabanlı algoritmalardan biridir. Çalışmalarında, Yapay Arı Kolonisi algoritmasının değiştirilmiş modelini, gerçek parametre optimizasyon problemlerine etkili bir şekilde çözmek için tanıtmış ve uygulamışlardır (Akay ve Karaboga, 2010).

Bhattachariee ve arkadaşları (2011) çalışmalarında, yapay arı kolonisi (YAK) optimizasyon algoritmasını kullanarak mobil robotların yol planlamasına alternatif bir yaklaşım önermektedirler. Burada üstlenilen sorun, tüm robotların yol uzunluğunu en aza indirgemek için robotların hareket yörüngesini, önceden belirlenmiş başlangıç pozisyonlarından dünya haritasındaki sabit hedef pozisyonlarına nihai bir hedefle belirlemeye çalışmaktadır. En iyi yolu elde etmek için YAK optimizasyon algoritması ile bir yerel yörünge planlama çalışması geliştirmişlerdir. Dünyadaki tüm robotların bir sonraki pozisyonları mevcut konumlarından hesaplanır. Deneyler, önerilen optimizasyon şemasının, ortalama toplam yol sapması ve ortalama keşfedilmemiş hedef mesafesi adı verilen standart metriklere göre iyi bilinen iki algoritmadan daha iyi performans gösterdiğini ortaya koymuşlardır (Bhattacharjee ve ark., 2011).

Ding ve arkadaşları (2015) çalışmalarında, havada asılı halde küçük ölçekli insansız helikopterin durum-uzay modelinin sistem tanımlaması için kaotik bir yapay arı kolonisi algoritması (KYAK) geliştirmeyi amaçlamışlardır. Önerilen algoritmanın etkinliğini ve belirlenen helikopter modelinin doğruluğunu göstermek için simülasyonlar sunmuşlardır (Ding ve ark., 2015).

Lin ve Huang (2009) çalışmalarında, öğrenme zekâsı ve kaotik dinamikleri temel almaktadırlar. Çalışmada, kaos ile birlikte geliştirilmiş yapay arı kolonisi

optimizasyonuna dayanan mobil robotlar için yol planlama sorununa yeni bir yaklaşım önermektedirler. Yapay arı kolonisi optimizasyonunda kaos, yapay arı kolonisi optimizasyonu ve kaotik arama davranışının popülasyona dayalı evrimsel arama yeteneğini makul bir şekilde birleştiren kaotik bir arı sürüsü optimizasyonu oluşturmak için geliştirilir. Bu kaotik değişken tüm arama alanı boyunca değişken olarak seyahat edebilir. Genel olarak, kaotik değişken özel karakterlere sahiptir, yani rastgele ve düzensiztir. Genel olarak, yapay arı kolonisi optimizasyonunun parametreleri yapay arı kolonisi optimizasyonunun yakınsamasını etkileyen anahtar faktörlerdir. Bununla birlikte, geleneksel yapay arı kolonisi optimizasyonunda kesinlikle rastgele oldukları için optimizasyonun çalışması tamamen çalışma alanında süreklilik sağlayamaz. Bu nedenle, bu çalışma yeni bir yöntem sunmaktadır. Küresel yakınsamayı geliştirmek için yapay arı kolonisi optimizasyonuna kesin, değişken ve stokastik özellik ile kaotik haritalama getirmektedir. Şekil 2.4'de Lin ve Huang'ın YAK algoritması ile yol planlama problemi için oluşturdukları çevre modeli gösterilmektedir (Huang ve Lin, 2009).



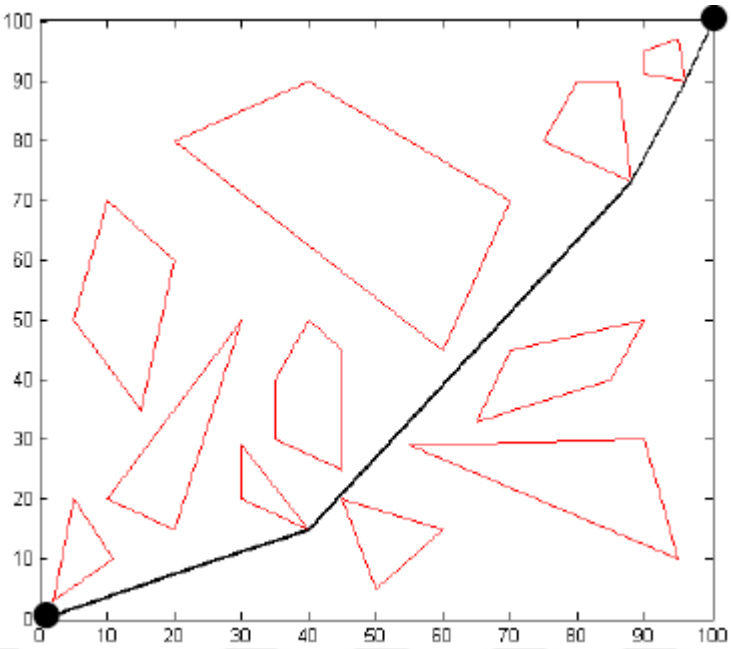
Şekil 2.4. Yapay Arı Kolonisi Algoritması ile Yol Planlama(Huang ve Lin, 2009)

Lei ve Shiru (2012), İnsansız Hava Araçlarının (İHA) yol planlamasını, kısıtlama koşulu ile karmaşık bir fonksiyon optimizasyonu problemi olabileceğini düşünmüşlerdir. Özellikle yapay arı kolonisi (YAK) algoritması, bu sorunu çözmek için etkili bir araç olarak bilinir. YAK algoritması, diğer popülasyon algoritmalarına göre daha az kontrol parametresine sahip olma avantajıyla nispeten baskın bir optimizasyon tekniğidir. Değişken ve kaotik haritanın stokastik özellikleri göz önüne alındığında, standart YAK için modifiye edilmiş bir başlangıç stratejisi önermektedirler, bu da ilk popülasyonun yanı sıra izci arı pozisyonunu oluşturmak için lojistik harita ve zıt tabanlı

öğrenmeyi kullanıyor. Ek olarak, kullanılan arı arama denklemi, yakınsama hızını arttırmak amacıyla ağırlık katsayıları eklenerek modifiye edilir. Ardından değiştirilmiş yapay arı kolonisi algoritmasını optimizasyonu probleminde ve yol planlama probleminde test etmişlerdir. Sonuçları daha üstün olan İHA yolu planlamasını iki boyutta çözmede algoritmanın performansını, standart ABC algoritması ile karşılaştırmışlardır (Lei ve Shiru, 2012).

Pan ve arkadaşları (2017), bu çalışmalarında, İHA rota planlama problemi için Kompakt yapay arı kolonisini (CYAK) sunmaktadır. Önerilen yöntemde, rota uzunluğu ve tehlikeye maruz kalma matematiksel amaç fonksiyonu olarak modellenmiş, ve kompakt algoritma konsepti rota planlama durumuna uyum sağlamak için uygulamışlardır. Bu algoritmada yapay arı koloni algoritmasının çözüm arama alanının gerçek tasarım değişkeni, nüfusun olasılıklı bir benzeri değiştirilir. Bu önerilen algoritma için arıların toplam davranışının rastgele bir olasılık temsilinden esinlenilmiştir. Gerçek nüfus, güncellenen olasılık vektörü ile değiştirilir. Literatürdeki diğer algoritmalarla karşılaştırıldığında hesaplama sonuçları, önerilen yöntemin İHA rota planlama problemi için etkili bir yolu sağlayabildiğini göstermişlerdir (Pan ve ark., 2017).

Saffari ve Mahjoob (2009), bu çalışmalarında, yapay arı algoritmasına dayanan mobil robotlar için yol planlama problemini çözmek için yeni bir yöntem sunmaktadırlar. Yöntem, kovanın çevresindeki yiyecek kaynaklarını bulmak için bal arılarının kolektif davranışlarından esinlenmiştir. Önerilen yöntem iki adım içerir. İlk adım, başlangıç noktasından hedefe ilk çarpışmasız yol oluşturmak için basit bir kural kullanmaktır ve ikinci adım, başlangıç yolunu optimize etmek için arı kolonisi algoritmasını kullanmaktır. Bilgisayar simülasyonlarının sonuçları, önerilen yöntemin etkili olduğunu ve mobil robotların gerçek zamanlı yol planlamasında kullanılabileceğini göstermektedirler. Şekil 2.5’de Saffari ve Mahjoob’un engelli ortamda yol planlama çalışmaları için oluşturdukları çevre ortamını göstermektedir (Mahjoog ve Saffari, 2009).



Şekil 2.5. Engelli Ortamda Yol Planlama(Mahjoog ve Safari, 2009)

Wang ve Li (2015) bu çalışmada, çok robotlu yol planlama problemlerinin çok amaçlı özelliğini yansıtan problemi geliştirmişlerdir ve bunu çok amaçlı yapay arı kolonisi algoritması ile tasarlamışlarıdır. İlk olarak, yemleme mekanizmasını optimize etmişlerdir ve kalabalıklık durumunu hesaplamak için yeni bir yöntem önermişlerdir. Ayrıca gıda kaynaklarının yeniden yapılandırılması ve ortadan kaldırılması mekanizması da sunulmaktadır. İkinci olarak, robotların yol bilgilerinin doğrudan Kartezyen koordinatları kullanılarak belirtildiği geliştirilmiş bir çevre haritası sunmuşlardır. Üçüncü olarak, yol planlaması için üç değişkenli amaç fonksiyonu tasarlamışlardır. Son olarak, simülasyon sonuçlarının geliştirilmiş çok amaçlı yapay arı kolonisi algoritmasının yol planlama problemlerini çözmek için etkili bir şekilde uygulanabileceğini göstermişlerdir (Li ve Wang, 2015).

Xu ve arkadaşları (2010), bu çalışmalarında, İnsansız Savaş Hava Aracı'nın (UCAV) yol planlamasının, farklı kısıtlamalar göz önünde bulundurularak üstün bir uçuş rotası aramakla ilgili oldukça karmaşık bir sorun olduğunu ortaya atmışlardır. Bundan dolayı geliştirdikleri Yapay Arı Kolonisi algoritmasını önermektedirler. Yapay Arı Kolonisi (YAK) algoritması, bal arılarının akıllı davranışlarıyla motive edilen yeni bir optimizasyon yöntemidir. Çeşitli savaş alanı ortamlarında UCAV yolu planlamasını çözmek için kaos teorisine dayanan optimizasyon algoritması ve önerilen kaotik YAK yaklaşımımızın uygulama prosedürünü de ayrıntılı olarak açıklamışlardır. Önerilen

yöntemin fizibilitesini, etkililiğini ve sağlamlığını göstermek için bir dizi deneysel çalışmanın karşılaştırmalı sonucunu sunmuşlardır (Xu ve ark., 2010).

Guan ve arkadaşları (2019), İHA'nın, bazı belirli kontrol noktalarını kontrol ederek veya göreve özgü bazı kısıtlamaları (örn. engellerden kaçınma, yakıt tüketimi, vb.) yerine getirerek başlangıç noktasından sonuna kadar optimum bir yolu bağımsız olarak hesaplamasına olanak tanımışlardır. Bunu Karınca Kolonisi Algoritması kullanarak gerçekleştirmişlerdir. Karınca kolonisi optimizasyonu (KKO) algoritması, karıncaların en uygun yolu bulmak için birlikte çalışabilmeleri nedeniyle büyük ilgi görmüştür. Bununla birlikte, KKO, özellikle sorunlu alanın büyük olduğu durumlarda, optimum bir yol bulmak için yavaşça yakınsar. Bu sorunu çözmek için, çift karınca kolonisine dayalı bir algoritma önermişlerdir. Daha spesifik olarak, erken aşamada feromonlar üretmek için genetik algoritmayı kullanmaktadırlar, böylece algoritmanın yakınsamasını hızlandırmaktadırlar. Sayısal sonuçlar önerilen algoritmanın etkinliğini doğrular (Guan ve ark., 2019).

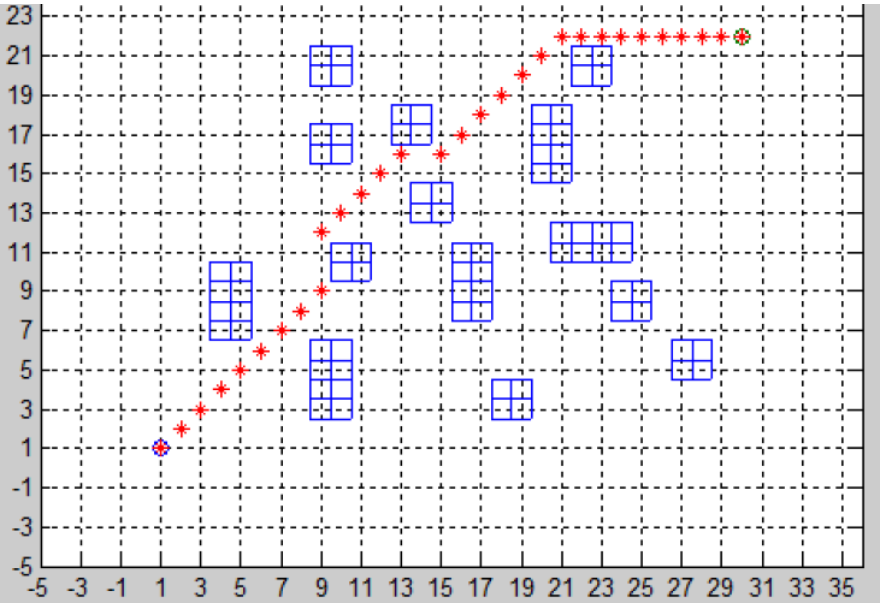
Ma ve Xiond (2019) çalışmalarında, çok sayıda düzensiz çalışma alanı için yol planlanmasında yeni bir yöntem önermişlerdir. Geliştirilmiş karınca kolonisi algoritması kullanılarak, çoklu bölgeler arasındaki en kısa toplam uçuş yolunu tasarlamışlardır. Sonuçlarına göre, bu yönteme göre optimize edilmiş uçuş yolunun yaklaşık% 4.46 oranında kısaltıldığını ve uygulama için geniş umutları olduğunu göstermişlerdir (Ma ve Xiond, 2019).

Raja ve Pugazhenthi (2012) bu çalışmalarında, çevrimiçi ve çevrimdışı ortamlar için bir mobil robotun yol planlamasındaki ilerlemesine genel bir bakış sunmaktadırlar. Mobil robotların yol planlamasında yaygın olarak kullanılan klasik ve evrimsel yaklaşımlar ele almışlardır. İncelemenin, evrimsel optimizasyon algoritmalarının hesaplama açısından verimli olduğunu ve bu nedenle gitgide klasik yaklaşımlarla birlikte daha fazla kullanılacağını göstermektedir. Ayrıca, hesaplamalı olarak etkili bir yol planlama algoritması geliştirmeyle ilgili zorluklara da değinmişlerdir (Pugazhenthi ve Raja, 2012).

Kou ve Ye (2011) araç yönlendirme problemi (VRP) için arı evrimsel parçacık sürüsü (BEPSO) temelli bir algoritma sundular ve VRP'ye uyguladılar. Bu algoritmada,

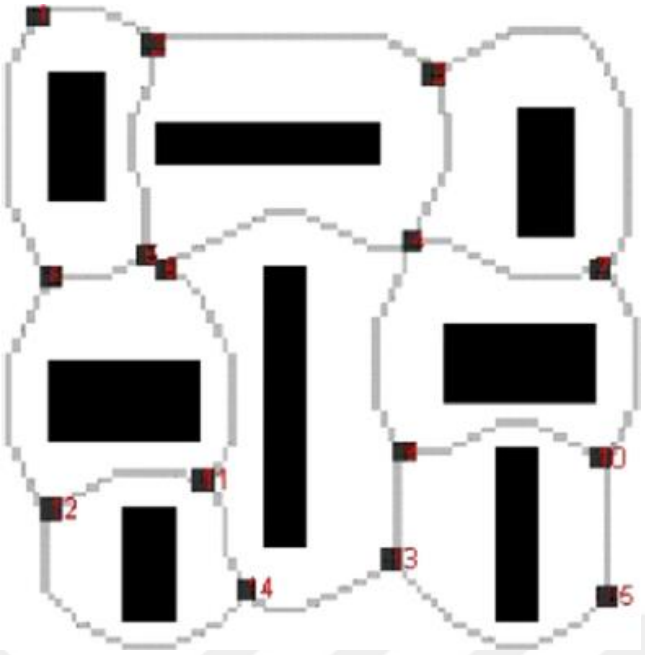
rastgele bir olasılıkla seçilen kraliçe arı olarak kabul edilen en iyi parçacık, diğer bireylerin uygulamada ki etkisini artırmaktadır. Aynı zamanda, bazı dronelar rastgele üretilit ve kraliçe arı ile çapraz konuma getirilir, bu da algoritmanın çeşitliliğini arttırmıştır. Bu simülasyon çalışmasında, bu algoritmanın daha iyi küresel arama yeteneğine sahip olduğunu ve algoritmanın fizibilitesini doğruladığını göstermiştir (Kou ve Ye, 2011).

Shiltagh ve Jalal (2013) bu çalışmalarında, modifiye parçacık sürüsü optimizasyonunun (MPSO) mobil robot navigasyon problemine, çalışma konumundan engelli çalışma ortamındaki bir hedef konuma geçmek için gereken minimum süreyi en kısa sürede belirlemek için uygulasını araştırmaktadır. Bu çalışmada, MPSO, küresel bir yol planlamasını optimize edilmiş algoritmaların kapasitesini artırmak için geliştirilmiştir. Önerilen algoritmalar, ortamın haritasını okur ve daha sonra optimal veya neredeyse çarpışmasız bir yol oluşturur. Mobil robot yolu planlaması için önerilen optimize edilmiş algoritmanın etkinliği simülasyon çalışmaları ile gösterilmiştir. Programlar MATLAB R2012a'da yazılmış ve 2.5 GHz Intel Core i5 ve 6 GB RAM'e sahip bir bilgisayarda çalıştığını belirtmişlerdir. MPSO'da sunulan gelişmeler esas olarak orijinal PSO ile ilişkili erken yakınsama sorununu çözmeye çalışmaktadır. MPSO'da PSO'nun birleşmesini sağlamak için bir hata faktörü modellenmiştir. MPSO, nüfusu pek çok olanaksız yol içerebilen başka bir sorunu ele almaya çalışır; imkansız yol problemini çözmek için MPSO'da değiştirilmiş bir prosedür gerçekleştirilmektedir. Sonuçlar, bu algoritmanın, yol planlamasını mesafe ve yürütme süresini en aza indirgemeyi tatmin edici sonuçlarla çözmek için büyük bir potansiyele sahip olduğunu göstermektedir. Şekil 2.6’da Shiltag ve Jalal tarafından oluşturulmuş çalışma ortamı gösterilmektedir (Jalal ve Shiltagh, 2013).



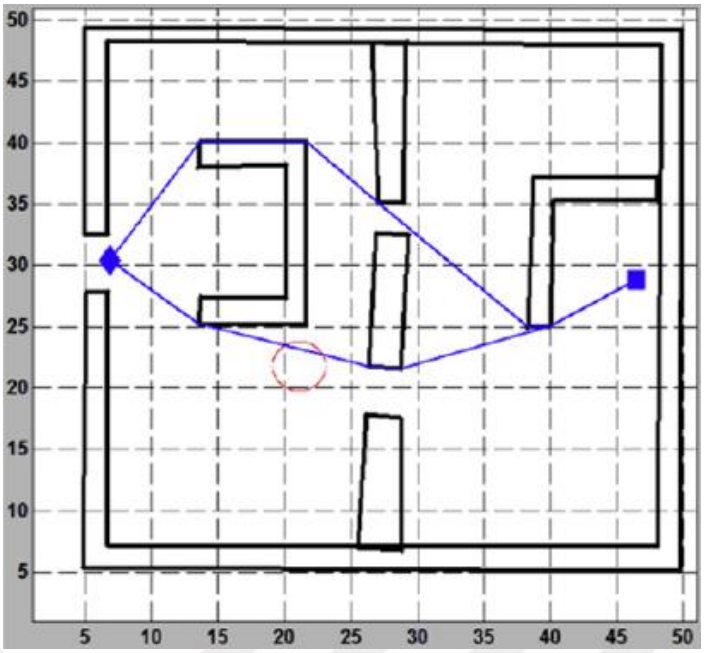
Şekil 2.6. MPSO Kullanılarak çalışma ortamı için oluşturulan yol(Jalal ve Shiltagh, 2013)

Mo ve Ju(2012) bu çalışmalarında, BBO, PSO ve yaklaşık voronoi sınır ağını (AVBN) statik bir ortamda birleştirerek yeni bir küresel yol planlama yöntemi sunmaktadır. Bu çalışmalarının amacı, BBO'daki nüfus çeşitliliğini artırmak için PSO'nun pozisyon güncelleme stratejisini uygulamak ve daha sonra AVBN modellemesi ile elde edilen yol ağındaki yolları optimize etmek için elde edilen biyocoğrafya parçacık sürüsü optimizasyon algoritmasını (BPSO) kullanmaktır. Simülasyondaki deneysel sonuçlar, önerilen yöntemin uygulanabilir ve etkili olduğunu göstermiştir. Şekil 2.7'de Mo ve Xu tarafından oluşturulan yol ağı gösterilmektedir (Mo ve Xu, 2012).



Şekil 2.7. Oluşturulan Yol Ağı(Mo ve Xu, 2012)

Zhang ve arkadaşları (2013) bu çalışmada, parçacığa dayalı çok amaçlı bir yol planlama algoritması önermektedirler. İlk olarak, yolun risk derecesini değerlendirmek için bir üyelik işlevi tanımlamışlardır. Ardından, kısıtlanmış bir nesnel parçacık sürüsü optimizasyonunu bu sorunu çözmek için geliştirmişlerdir. Rasgele örnekleme ve düzgün mutasyona dayalı parçacık güncelleme yöntemi gibi çeşitli yeni işlemler , iyileştirmeler, engellerle çarpışma sürelerine dayanan kısıtlı hakimiyet ilişkisi, etkinliğini artırmak için önerilen algoritmaya dahil edilir. Son olarak, simülasyon sonuçları yöntemin başarılı olduğunu göstermektedir. Şekil 2.8'de Zhang ve arkadaşları tarafından oluşturulan simülasyon ortamı gösterilmektedir (Zhang ve ark., 2012).

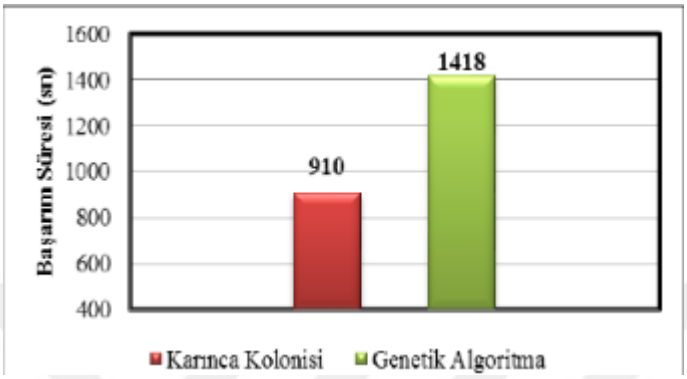


Şekil 2.8. Çevre Haritası ve Simülasyon Düzeneği(Zhang ve ark., 2012)

İnsansız Hava Araçları (İHA) kameralar, algılayıcılar, iletisim araçları ve diger faydalı yükleri tasıyabilen, uzaktan kumanda ile veya kendi kendine uçabilen hava araçlarıdır. Döner-rotorlu İHA'lar yapısal olarak sahip oldugu helikopter ve uçak modlarıyla, bu hava araçlarının havada asılı kalma ve yüksek seyir hızı gibi tercih edilen özelliklerini bünyesinde barındıran bir hava platformu olarak, kullanım alanlarındaki degisken ihtiyaçlara cevap verebilecek bir yapıya sahiptir. Ferit Çakıcı (2009) bu çalısmada, model uçak malzemeleriyle üretilebilecek döner-rotorlu bir mini İHA'nın kavramsal tasarımı yapılmıs, aerodinamik modeli olusturmus, helikopter, uçak ve geçis modları için denge noktaları hesaplanarak dogrusal modelleri elde edilmiştir, LQR (Linear Quadratic Regulator) yöntemiyle dogrusal kontrolcüler tasarlamıs ve bütün uçus zarfı için kazanç ayarlama (Gain-scheduling) yöntemiyle kontrol sisteminin benzetimi gerçeklestirmiştir. Elde edilen sonuçlar çerçevesinde, gerçek bir modelin prototipinin olusturulmasına yönelik fikirleri öne sürmüştür (Çakıcı, 2009).

Dikmen ve arkadaşları (2014) bu çalışmada, rota planlama problemlerinden olan gezgin satıcı probleminin (GSP) çözümünü gerçekleştirmek için yapay zekâ tekniklerinden olan karınca kolonisi ve genetik algoritmaların performansları karşılaştırmışlardır. Türkiye haritası üzerinde gerçekleştirdikleri çalışmada en iyi rotanın planlanması hedeflenmiştir. Her iki algoritmanın rota mesafesi yönünden başarım performansı ve bu rotayı hesaplama süresini incelemişlerdir.

Uygulamanın gerçekleştirilmesi ve deneysel sonuçların gözlemlenmesi için C# tabanlı bir arayüz tasarlamışlardır. Uygulama sonucunda karınca kolonisi algoritmasının hem rota mesafesi hem de başarım süresi yönünden genetik algoritmalara göre daha üstün olduğunu gözlemlemişlerdir. Şekil 2.9'da Dikmen ve arkadaşları tarafından oluşturulan çalışmanın karşılaştırmalı grafiği gösterilmektedir (Dikmen ve ark, 2014).



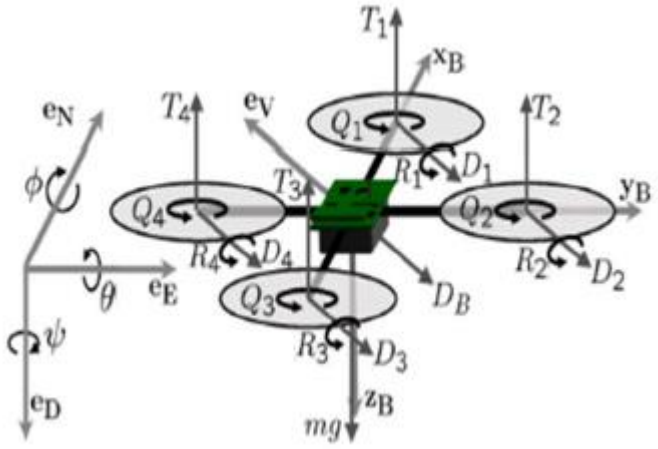
Şekil 2.9. KKA ve GA'nın Arama Süresi Yönünden Başarım Performansı(Dikmen ve ark., 2014)

Suvaydan (2011) tez çalışmasında da, mobil robotların yol planlaması problemini çözme işlemi için sezgisel yol planlama tekniklerinden biri olan karınca kolonisi algoritması kullanmış olup, buna göre bir alan içersinde engellere çarpma olmaksızın optimum yolun bulunmasını amaçlamıştır. Bu çalışma için görsel bir simülasyon programı hazırlamıştır. Bu simülasyon program ile başlangıç-bitiş koordinatları, engellerin sayısı ve boyutları ile karınca kolonisi algoritması parametre değerleri dışarıdan girilmektedir. Bu sayede parametre değerleri istenildiğinde değiştirilerek buna göre sonuçları sağlıklı bir şekilde değerlendirmesini sağlamıştır. Hazırlanan simülasyon programı ile 4 farklı çevre oluşturup, sonuçlar önerilen algoritmanın lokal feromen güncellemesine ve global feromen güncellemesine bağlı olarak elde etmiştir. Bununla birlikte karşılaştırmalar amacıyla yolların ceza fonksiyonu uygulamadan normal uzunlukları hesaplanmıştır ve bağıl hata değerlerini bulmuştur. Parametre değişikliklerine bağlı olarak program farklı engel boyutu ve sayısında defalarca çalıştırılarak algoritmanın performansını ve etkinliğini değerlendirmiştir (Suvaydan, 2011).

Yetgin (2011) bu çalışmada, Türkiye' de yurt içi taşımacılıkta mevcut coğrafi şartlar göz önüne alınarak karayolu ve demiryolunu içeren entegre bir sistem önermiştir.

Önerilen sistemde, karayolları ve demiryollarının mevcut durumları değerlendirilerek, demiryolu ile ulaşımın olduğu yerlere demiryolu ile, demiryolunun olmadığı yerlere karayolu ile ulaşılarak tüm ülkede en kısa turun bulunması hedeflemiştir. Bu anlamda gezgin satıcı problemleri ile en kısa yolların belirlenmesinde karınca kolonisi algoritmasını kullanmıştır. Yetgin'e göretaşımacılık faaliyetlerinin maliyetleri arasında en büyük payı enerji maliyetleri oluşturmaktadır. Bu nedenle bulunan en kısa yollar tüketilen enerji miktarı açısından incelemiştir. Ayrıca, çevreye yayılan karbondioksit miktarı da değerlendirilmiş ve güvenlik, arazi kullanımı, gürültü ve alt yapı maliyetleri yorumlamıştır (Yetgin, 2011).

Jafar ve arkadaşları (2016) bu çalışmada, Dörtlü helikopterin eğim, rulo, sapma ve itme komutunu arayüz ile modellenmesi ve doğrusal kontrolünü sağlamışlardır. Bu araştırma, 6 serbestlik derecesi (DOF) hareketinden oluşan dört helikopter dinamiğinin doğrusal olmayan matematiksel modelinin türetilmesiyle oluşturmuşlardır. Şekil 2.10'da Jafar ve arkadaşları tarafından Quadcopterin ataletsel koordinatları gösterilmektedir (Jafar ve ark., 2016). Doğrusal olmayan model daha sonra denge noktası etrafında doğrusal hale getirilmiş ve daha sonra üç açısal hareketin (Pitch, Yaw ve roll) stabilizasyon ve izleme kontrolünü sağlamak için 3 DOF kinematiği çıkarılmıştır. Her döndürme hareketinin açık döngü aktarım işlevi, dört helikopterin doğrusal modda herhangi bir sönümleme faktörü olmadan oldukça kararsız olduğunu göstermiştir. Orantılı bir integratör ve PID kontrolörü, dört rotoru simülasyondan bağımsız olarak kontrol ederek 3 DOF rotasyonlarını kontrol etmek ve stabilize etmek için tasarlamışlardır.



Şekil 2.10. Quadcopter'de Ataletsel Koordinatlar(Jafar ve ark., 2016)

Gerçek zamanlı 1 DOF işlemi, test teçhizatı üzerindeki kapalı döngü PID kontrolörü tarafından itme, ortalama RPM arasındaki ilişkinin değerlendirildiği ve istenen adım aralığı yanıtının elde edildiği tasarlanmış test teçhizatı üzerinde yapmışlardır. Yukarıdaki argümanları desteklemek için, simüle edilmiş sonuçlar ve deneysel sonuçları sunup karşılaştırmışlardır.

Wang ve Zhao (2016) bu çalışmada, bir Quadcopter'in davranışını dengelemek için kademeli bir PID geri besleme kontrol algoritmasını önermişlerdir. Quadcopter dinamiğinin matematiksel bir modeli Newton-Euler yöntemi uygulanarak geliştirmişlerdir. Daha sonra kontrolör tasarımı ve daha fazla geliştirme için gerekli olan doğrusal ve doğrusal olmayan durum uzay denklemlerini türetmişlerdir. Simülasyonlar, klasik PID kontrol şemasına kıyasla, kademeli PID algoritmasının etkililiğinin ve sağlamlığının daha iyi olduğunu göstermiştir (Wang ve Zhao, 2016).

3. MATERYAL VE YÖNTEM

Sürü çalışmalarında yol planlama algoritmaları için birçok araştırma yapılmıştır. Her algoritmanın kendilerine göre artıları ve eksikleri vardır. Bu algoritmaların seçiminde sürünün kullanılacağı çevre şartları, sürünün hangi görevde kullanılacağı, sürüdeki parçacık sayısı gibi birçok etken değerlendirmeye alınmalıdır. Bu tez çalışmasında ise, belirlenen görevi en iyi şekilde yerine getirebileceği düşünülen sürü algoritmaları çeşitleri üzerinde durulmuştur. Bu bağlamda, tez çalışmasında 3 adet sürü algoritmasının ortak bir görev üzerinde başarı durumları karşılaştırılacaktır. Karşılaştırılacak sürü algoritmaları;

- Parçacık Sürü Optimizasyonu,
- Yapay Arı Kolonisi Algoritması,
- Karınca Kolonisi Algoritması,

olarak belirlenmiştir.

3.1. Parçacık Sürü Optimizasyonu

Parçacık Sürü Optimizasyonu (PSO), birlikte hareket eden canlıların bir amaç doğrultusunda yaptıkları hareketlerin, diğer canlılarla iletişim kurarak sürüdeki diğer bireyleri etkilediğinin ve amaca uygun belirledikleri davranışların gözlemlenmesinden yola çıkarak Dr. Kennedy ve Dr. Eberhart tarafından 1995 yılında geliştirilmiş bir optimizasyon algoritmasıdır. PSO, bir maliyet işlevini en aza indirgemek için çalışır. Sürüyü modellemek için, her parçacık çok boyutlu bir alanda hareket eder. Bir parçacık, mevcut pozisyon vektörü $x_n[t]$, hız vektörü $v_n(t)$, istenen pozisyon vektörü $x(t+1)$ ve bir sonraki hız vektörü $v(t+1)$ ile temsil edilir. Algoritmada yerel en iyi vektör $p(x)$ ve küresel en iyi vektör $g(x)$ olarak tanımlanır. Vektörlerin boyutu çok boyutlu uzayın boyutuna bağlıdır. Denklem (3.1) ve (3.2)'de PSO için hız ve konum hesaplamaları gösterilmektedir (Couceiro ve Ferraira, 2011).

$$v_n[t+1] = w v_n[t] + c_1 r_1 (p(x) - x_n[t]) + c_2 r_2 (g(x) - x_n[t]) \quad (3.1.)$$

$$x_n[t+1] = x_n[t] + v_n[t+1] \quad (3.2.)$$

İfadelerdeki; w, r_1 ve r_2 katsayıları, sırasıyla yeni hız değerini belirlemede kullanılan atalet etkisi, küresel en iyi ve yerel en iyi ağırlıklar olarak tanımlanır. Genel olarak, atalet etkisi 1'den küçük bir değere ayarlanır. İfade de görülen c_1 ve c_2 , "sosyal" (küresel) ve "bilişsel" (yerel) bileşenleri temsil eden sabit tamsayı değerleridir. Bununla birlikte, her bileşen için farklı etkiler atanarak farklı sonuçlar elde edilebilir. Uygulamaya ve dikkate alınan sorunun özelliklerine bağlı olarak, bu parametrelerin doğru bir şekilde ayarlanması daha iyi sonuçların alınması sağlayacaktır. c_1 ve c_2 parametreleri, genellikle 0 ile 1 arasında rastgele bir sayıdır. Amaç, aynı bileşeni her parçacığın hız bileşeni ile çarpmak yerine, hız boyutu başına yeni bir rastgele bileşen ile çarpmaktır (Couceriro ve ark., 2011).

PSO üzerinde yapılan çalışmalar sayesinde yeni yöntemlerde ortaya çıkarılmıştır. Bu yöntemlerden bir tanesi Modifiye Parçacık Sürü Optimizasyonudur (MPSO). Temel olarak MPSO, tıpkı PSO gibi, arama alanında topluca küresel optimum arayışını araştıran bir parçacık popülasyonundan oluşur. İyileştirmeler temel olarak standart PSO ile ilişkili sorunları çözmeye çalışır. MPSO'da PSO'nun yakınlaşmasını sağlamak için bir hata faktörü modellenmiştir. MPSO'da, mümkün olmayan yollar atılmaz, ancak uygulanabilir yol olarak değiştirilebilir (Jalal ve Shiltag, 2013). MPSO'da bu hatalar PSO'nun yakınlaşmasını sağlamak için modellenmiştir, hata ile her parçacık için değiştirilmiş konum güncelleme denklemi aşağıdaki gibi ifade edilir.

$$x_2 = x_1 + vx_2 + r * Errorx \tag{3.3.}$$

$$y_2 = y_1 + vy_2 + r * Errory \tag{3.4.}$$

Denklem (3.3.) ve Denklem (3.4.)'de x ve y konumlarının hata payına göre yeniden hesaplanması formülize edilmiştir. Burada x_2 ve y_2, x_1 ve y_1 konumu boyunca güncellenmiş konumlardır, r rastgele bir sayıdır, bu parametrenin büyük değerleri genel aramaya, küçük değerler ise yerel aramaya yol açar. Errorx ve Errory, sırasıyla tüm parçacıklar için x ve y koordinatları boyunca konum hatasıdır ve geçerli parçacık konumu ile hedef konum arasındadır. Bu hatalar, parçacığın hedefe yönelme yönünü değiştirmek için kullanılır. Denklem (3.5.) ve (3.6.)'da konum hatalarının hesaplanması gösterilmektedir.

$$Errorx = x_p - x_t \tag{3.5.}$$

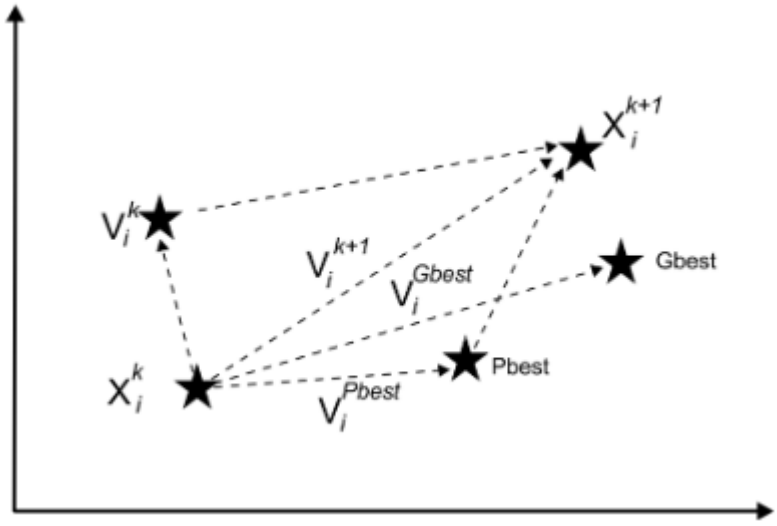
$$Errory = y_p - y_t \tag{3.6.}$$

Hata değeri, her bir parçacık için belirlenir; burada x_p ve y_p, ayrı parçacığın geçerli koordinatlarıdır ve xt ve yt, hedef koordinatlardır. Yukarıdaki denklemlere dayanarak, her bir parçacığın bir sonraki olası hızı ve konumu belirlenir.

PSO, bir problemin arama alanında keşfetmek için kullanılan bir tekniktir. Bu alanda PSO algoritması, hata durumunu küçültmek veya maksimize etmek için gerekli parametreleri araştırır. PSO algoritmasının üç aşaması vardır. Bu işlem bir durdurma koşulu gözlenene kadar tekrarlanır:

1. Her bir parçacığın uygunluğunu değerlendirilir.
2. En iyi fitness, yerel ve küresel en iyi pozisyonu güncellenir.
3. Her parçacığın hızı ve konumu güncellenir.

Şekil 3.1.'de PSO üyelerinin birbirleri ile olan etkileşimi gösterilmektedir.



Şekil 3.1. Örnek bir PSO üyelerinin etkileşimi

Parçacıkların hareketleri, hedef alanda kendi en iyi bilinen konumlarının yanı sıra tüm sürünün en iyi bilinen konumu tarafından yönlendirilir. İyileştirilmiş pozisyonlar keşfedildiğinde, bunlar sürünün hareketlerini yönlendirmek için gelecektir. Süreç tekrarlanır ve bu şekilde sonuçta tatmin edici bir çözüm bulunacağı umulur, ancak garanti edilemez. PSO'daki her parçacığın hızı ve konumu, sırasıyla (3.1.) ve (3.2.) denklemleri kullanılarak güncellenir (Ahmadzadeh ve Ghanavati, 2012).

PSO'da ilk olarak amaç fonksiyonu belirlenmesi gerekir. Bu amaç onksiyonuna göre sürü içerisinde bulunan elemanların gerçekleştirecekleri göreve göre elde

edecekleri sonuçları karşılaştırması ve başarı durumunu test etmesi sağlanır. Uygunluk fonksiyonu bulunurken görev ve çevre şartları dikkate alınması gerekir.

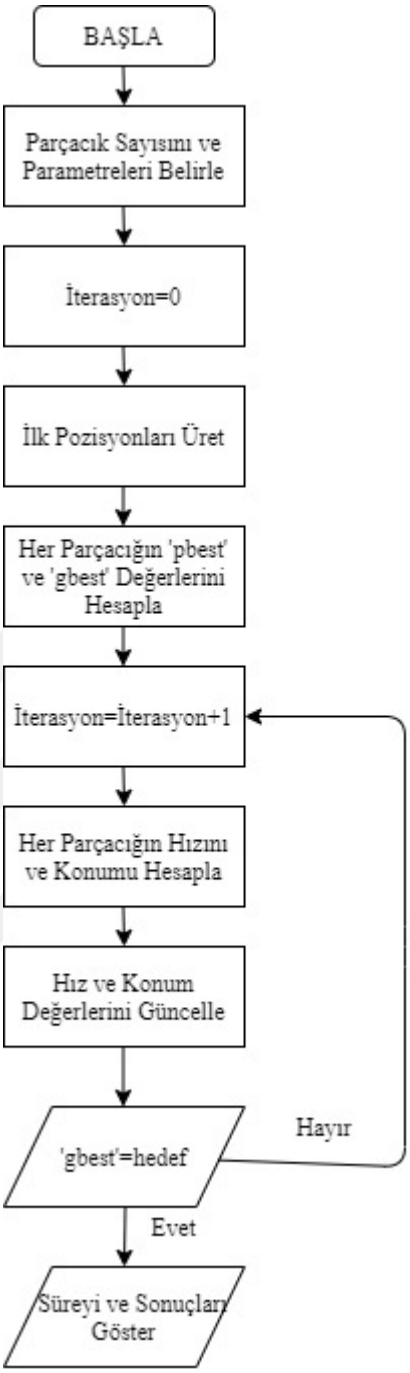
Parçacığın yapılan çalışma boyunca kendisinin en iyi durumuna ‘pbest’ denir. Çalışma boyunca sürüde bulunan parçacıkların hedefe en yakın olduğu konum ‘gbest’ olarak adlandırılır.

PSO’da sürünün eleman sayısı ve göreve uygun olacak şekilde sürü hakkında parametreler belirlenir. Sürüdeki parçacıkların konumu her iterasyonda amaç fonksiyonundan geçirilerek çözüme yakınlığı hesaplanır. Bu hesaplamalar sonucunda pbest ve gbest değerleri her iterasyon sonunda hesaplanır. Bu hesaplamalardan elde edilen değerler ile parçacığın değişim hızı bulunur ve parçacıklar elde edilen hız değerlerine göre yeni konumlarına hareket ederler. Yeni konumlarına gelen parçacıklar tekrar uygunluk fonksiyonundan geçerek yeniden hedefe ne kadar yaklaştıkları hesaplanır. Çözüm sağlanıncaya kadar bu döngü bu şekilde devam eder. PSO’nun ana parametreleri: Sürünün büyüklüğü, iterasyon sayısı, hızlanma katsayıları ve ağırlığından oluşmaktadır.

Algoritmanın adımları şöyle açıklanabilir.

- Sürü parçacıkları ilk konumlarına getirilir.
- Amaç fonksiyonuna göre hangi parçacığın en iyi konumda olduğu hesaplanır.
- Denklem (3.1.)’e göre her parçacığın bir sonraki iterasyon için hız değeri hesaplanır.
- Denklem (3.2.)’ye göre her parçacığın bir sonraki iterasyon için konumları hesaplanır.
- Yeni konumları göre küresel ve yerel en iyi pozisyonlar yeniden hesaplanır.
- İterasyon sayısı boyunca bu döngü devam ettirilir.

PSO algoritmasının şeması şekil 3.2.’de gösterilmektedir.



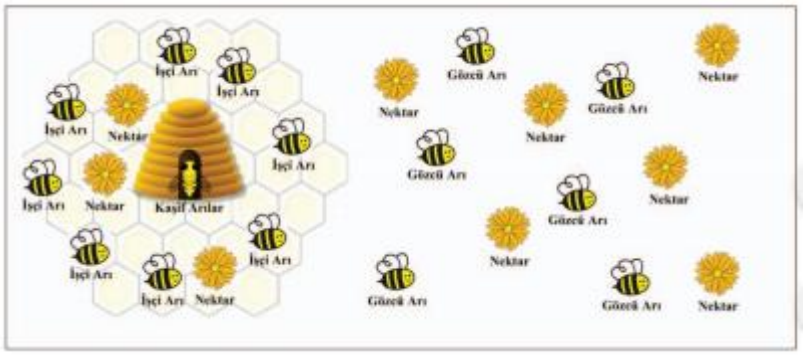
Şekil 3.2. PSO Algoritma Şeması

PSO algoritması kullanılarak yapılan hedef takip optimizasyonu Bölüm 5.2'de açıklanmaktadır.

3.2. Yapay Arı Kolonisi Algoritması

Yapay Arı Kolonisi (YAK) algoritmasında arıların doğadaki davranışları incelenerek, hareket ve birbirleri ile haberleşmesi dikkate alınmıştır.

YAK modelinde, koloni üç grup arıdan oluşur: çalışan arılar, izleyiciler ve izciler. Her besin kaynağı için yalnızca bir yapay çalışan arı olduğu varsayılmaktadır. Yani kolonide çalışan arı sayısı, kovan çevresindeki besin kaynaklarının sayısına eşittir. Çalışan arılar besin kaynaklarına giderler ve kovanlarına geri dönerler ve bu alanda dans ederler. Besin kaynağı terk edilen çalışan arı, izci olur ve yeni bir besin kaynağı aramaya başlar. Seyirciler çalışan arıların danslarını izler ve danslara göre besin kaynaklarını seçerler. Şekil 3.3.'de Yapay Arı Kolonisi yapısı gösterilmektedir.



Şekil 3.3. Yapay Arı Kolonisi Yapısı

Algoritmanın ana adımları aşağıda verilmiştir (Derviş Karaboğa, 2005)

- Çalışan tüm arılar için ilk besin kaynakları üretilir

- Çalışan her arı hafızasında bir besin kaynağına gider ve en yakın kaynağı belirler, ardından nektar miktarını değerlendirir ve kovandaki dans e

- Her izleyici, çalışan arıların dansını izler ve danslarına göre kaynaklarından birini seçer ve sonra o kaynağa gider. Çevresinde bir komşu seçtikten sonra nektar miktarını değerlendirir.

- Terk edilmiş besin kaynakları belirlenir ve keşifçiler tarafından keşfedilen yeni besin kaynakları ile değiştirilir.

- Şimdiye kadar bulunan en iyi gıda kaynağı kayıt edilir.

Nüfus bazlı bir algoritma olan YAK'da, bir gıda kaynağının konumu, optimizasyon problemine olası bir çözümü temsil eder ve bir gıda kaynağının nektar miktarı, ilgili çözümün kalitesine (uygunluğuna) karşılık gelir. Çalışan arı sayısı, popülasyondaki çözelti sayısına eşittir. İlk adımda, rastgele dağıtılmış bir ilk popülasyon (besin kaynağı pozisyonları) oluşturulur. Başlatma işleminden sonra, popülasyon, sırasıyla, çalışan, izleyen ve izci arıların arama süreçlerinin döngülerini tekrarlamaya tabi tutar. Çalışan bir arı, belleğindeki kaynak konumunda bir değişiklik yapar ve yeni bir besin kaynağı konumu keşfeder. Yenisinin nektar miktarı bir önceki

kaynağa göre daha yüksek olduğu takdirde arı yeni kaynak konumunu ezberler ve eskisini unutur. Aksi takdirde hafızasında birinin konumunu tutar. Tüm istihdam edilen arılar arama sürecini tamamladıktan sonra kaynakların konum bilgilerini dans alanında izleyenlerle paylaşırlar. Her izleyici, çalışan tüm arılardan alınan nektar bilgisini değerlendirir ve daha sonra kaynakların nektar miktarına bağlı olarak bir besin kaynağı seçer. İstihdam edilen arı durumunda olduğu gibi, hafızasındaki kaynak konumunda bir değişiklik yapar ve nektar miktarını kontrol eder. Arı, nektarının bir öncekinden daha yüksek olması şartıyla yeni pozisyonunu ezberleyerek eskisini unutur. Yapay izciler tarafından terk edilen kaynaklar belirlenir ve yeni kaynaklar rastgele üretilerek terk edilmiş olanlarla değiştirilir. Denklem (3.7)'ye göre arılar için başlangıçta yeni bir konum üretilir. Burada x_j^{min} ve x_j^{max} başlangıç değerlerinin belirlenmesi için maksimum ve minimum değerlerdir.

$$x_{ij} = x_j^{min} + rand(0,1) * (x_j^{max} - x_j^{min}) \quad (3.7.)$$

Sürüdeki her arı aşağıda tanımlanan formüle göre mevcut konumunun yakınında yeni bir konum üretir. Denklem (3.8)'de parçacıkların yeni konumlarının hesaplamaları formülüze edilmiştir.

$$v_{i,k} = x_{i,k} + \Phi_{i,k} * (x_{i,k} - x_{j,k}) \quad (3.8.)$$

Burada X_i iterasyonun başında rastgele seçilmiş arının ilk konumlarıdır. $\Phi_{i,k}$, ise -1 ile 1 arasında rastgele belirlenmiş bir sayıdır.

Tüm istihdam edilen arılar arama sürecini tamamladıktan sonra; besin kaynaklarının bilgilerini izleyen arılarla sallantılı danslarla paylaşırlar. İzleyen bir arı, çalışan tüm arılardan alınan nektar bilgisini değerlendirir ve nektar miktarı ile ilgili olasılıkla bir besin kaynağı seçer. Bu olasılıksal seçim, aşağıdaki denklem (3.9) olarak tanımlanan bir rulet çarkı seçme mekanizmasıdır:

$$P_i = \frac{fit_i}{\sum_j fitj} \quad (3.9.)$$

Burada fit_i iterasyonun en iyi pozisyonudur. Terkedilmiş kaynağın X_i olduğunu ve ardından keşif arısının aşağıdaki denklem ile yeni bir besin kaynağı keşfettiği varsayılır. Yeni besin kaynağının konumunun hesaplanması denklem (3.8)'de gibi gösterilmektedir.

Genel itibari ile YAK algoritması bu şekildedir. Algoritmanın genel adımları aşağıda açıklanmaktadır;

Adım 1: Denklem (3.7.)'ye göre $v_{i,k}$ komşu kaynağı üretilir ve f_i değeri hesaplanır.

Adım 2: $v_{i,k}$ ve $x_{i,k}$arasında açgözlü seçim işlemi uygulanır.

Adım 3: Denklem (3.9) kullanılarak kaynakların olasılıksal değeri hesaplanır.

Adım 4: Tükenen kaynak var ise Denklem (3.7.) ile yeni kaynak üretilir.

Adım 5: Bulunan en iyi kaynak koordinatları kaydedilir.

Adım 6; İterasyon değeri 1 artırılır.

Şekil 3.4'de YAK algoritmasının algoritma şeması gösterilmektedir.

Şekil 3.4. YAK Algoritma Şeması

Bölüm 5.3'de YAK algoritması kullanılarak gerçekleştiren uygulama ve sonuçları açıklanmaktadır.

3.3. Karınca Kolonisi Algoritması

Karınca kolonisi optimizasyon algoritması (KKO), grafiksel yöntemler ile iyi yollar bulmaya çalışan hesaplama problemlerini çözmek için ortaya atılmış olasılıksal bir tekniktir. Bu algoritmada kullanılan yapay karıncalar, gerçek karıncaların davranışlarından esinlenen çok yönlü davranışları tespit ederek ortaya çıkarır. Gerçek karıncaların feromon temelli iletişimi genellikle kullanılan en yaygın yöntemdir(Monmarché ve ark., 2010). Yapay Karıncaların ve yerel arama algoritmalarının kombinasyonları, araç yönlendirme ve internet yönlendirmesi gibi bir tür grafiği içeren çok sayıda optimizasyon görevi için tercih edilen bir yöntem haline geldi.

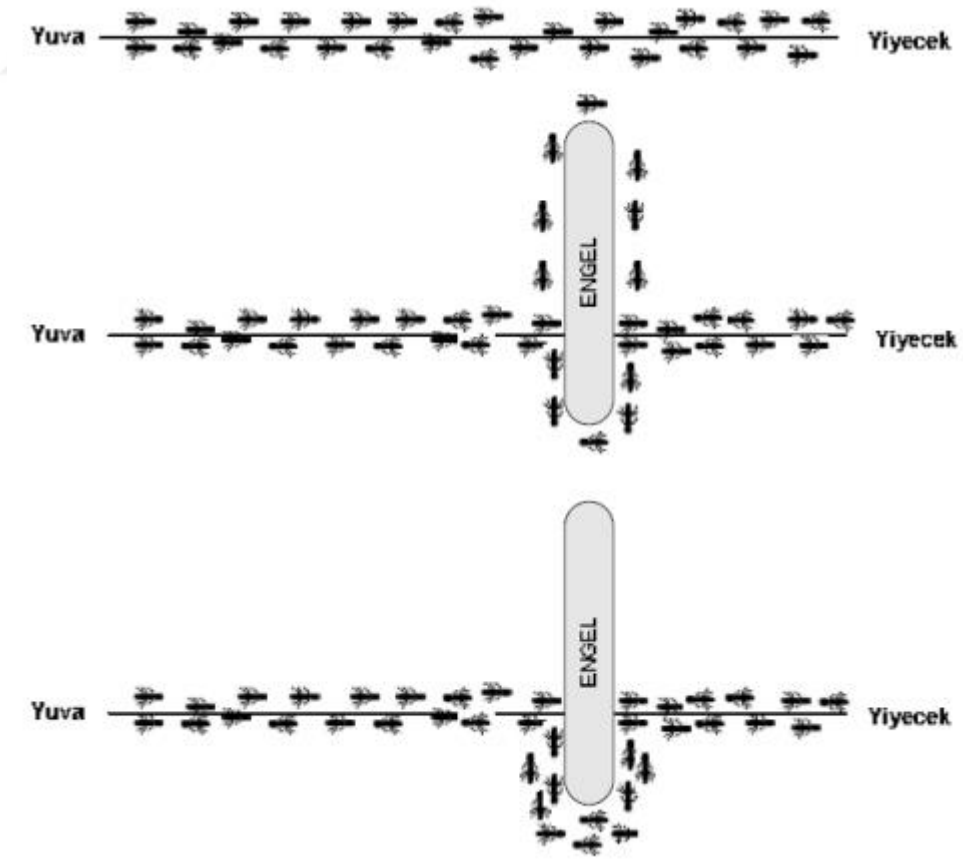
Örnek olarak, Karınca kolonisi optimizasyonu(Dorigo ve Stützle, 2004), bir karınca kolonisinin eylemlerine göre modellenen bir optimizasyon algoritmaları sınıfıdır. Yapay 'karıncalar', tüm olası çözümleri temsil eden bir parametre uzayında hareket ederek en uygun çözümleri bulurlar. Gerçek karıncalar çevrelerini keşfederken birbirlerini kaynaklara yönlendiren feromonlar bırakırlar. Simüle edilmiş 'karıncalar' benzer şekilde konumlarını ve çözümlerinin kalitesini kaydeder, böylece sonraki simülasyon yinelemelerinde daha fazla karınca daha iyi çözümler bulur(Dorigo ve Stützle, 2004). Bu yaklaşımın bir varyasyonu, başka bir sosyal böcek olan bal arısının yiyecek arama modellerine daha benzer olan arılar algoritmasıdır.

Bu algoritma, sürü zekası yöntemlerinde karınca kolonisi algoritmaları ailesinin bir üyesidir ve bazı meta-sezgisel optimizasyonları oluşturur. İlk olarak 1992 yılında Marco Dorigo tarafından doktora tezinde önerilen ilk algoritma, kolonileri ile bir besin kaynağı arasında bir yol arayan karıncaların davranışlarına dayalı olarak bir grafikte optimal bir yol aramayı hedefliyordu. Orijinal fikir o zamandan beri daha geniş bir sayısal problem sınıfını çözmek için çeşitlendi ve sonuç olarak, karıncaların davranışlarının çeşitli yönlerinden yararlanarak birkaç problem ortaya çıktı. Daha geniş bir perspektiften, KKO model tabanlı bir arama (Zlochin ve ark.) gerçekleştirir ve dağıtım algoritmalarının tahminiyle bazı benzerlikler paylaşır.

Doğal dünyada, bazı türlerin karıncaları (başlangıçta) rasgele dolaşırlar ve yiyecek bulduklarında, feromon izlerini bırakırken kolonilerine geri döner. Diğer karıncalar böyle bir yol bulurlarsa, muhtemelen gelişigüzel seyahat etmeye devam etmezler, bunun yerine yolu takip ederler, sonunda yiyecek bulurlarsa geri dönerler ve onu güçlendirirler (Fladerer ve ark.). Ancak zamanla feromon izi buharlaşmaya başlar ve böylece çekici gücünü azaltır. Bir karıncanın yoldan aşağı inip geri dönmesi ne kadar

çok zaman alırsa, feromonların o kadar çok buharlaşması gerekir. Karşılaştırıldığında, kısa bir yol daha sık ilerler ve bu nedenle feromon yoğunluğu daha kısa yollarda uzun yollardan daha yüksek olur. Feromon buharlaşması ayrıca yerel olarak optimal bir çözüme yakınsamadan kaçınma avantajına da sahiptir. Hiç buharlaşma olmasaydı, ilk karıncalar tarafından seçilen yollar, sonraki olanlar için aşırı derecede çekici olma eğiliminde olacaktır. Bu durumda çözüm uzayının keşfi kısıtlanacaktır. Gerçek karınca sistemlerinde feromon buharlaşmasının etkisi belirsizdir, ancak yapay sistemlerde çok önemlidir(Dorigo ve Stützle, 2004).

Genel sonuç, bir karınca koloniden bir besin kaynağına giden iyi (yani kısa) bir yol bulduğunda, diğer karıncaların bu yolu takip etme olasılığının daha yüksek olması ve olumlu geri bildirimin sonunda birçok karıncanın tek bir yol izlemesine yol açmasıdır. Karınca kolonisi algoritmasının amacı, bu davranışı, çözülecek problemi temsil eden grafikte dolaşan "simüle karıncalar" ile taklit etmektir. Şekil 3.5.'de karıncaların yiyeceklere karşı engel veya engelsiz ortamda davranışları gösterilmektedir.



Şekil 3.5. Örnek Karınca Kolonisi Davranışı

Feromon temelli iletişim, doğada yaygın olarak gözlemlenen en etkili iletişim yollarından biridir. Feromon, arılar, karıncalar ve termitler gibi sosyal böcekler tarafından kullanılır. Uygulanabilirliği nedeniyle, çoklu robot ve sürü robotik sistemlerinde yapay feromonlar benimsenmiştir.

Karınca kolonisi optimizasyon algoritmalarında, yapay bir karınca, belirli bir optimizasyon problemine iyi çözümler arayan basit bir hesaplama aracıdır. Bir karınca kolonisi algoritması uygulamak için optimizasyon probleminin ağırlıklı bir grafikte en kısa yolu bulma problemine dönüştürülmesi gerekir. Her yinelemenin ilk adımında, her karınca stokastik olarak bir çözüm oluşturur, yani grafikteki kenarların takip edilmesi gereken sıra. İkinci adımda, farklı karıncaların bulduğu yollar karşılaştırılır. Son adım, her bir kenardaki feromon seviyelerinin güncellenmesinden oluşur.

Her karınca, grafikte ilerlemek için bir çözüm oluşturmalıdır. Turunda bir sonraki kenarı seçmek için, bir karınca, mevcut olan onun kenarın uzunluğunun yanı sıra karşılık gelen feromon içinde gösterilen alacaktır Karıncaların konum değiştirme hesaplamaları deklem (3.10)'a göre yapılmaktadır.

$$P_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(n_{xy}^{\beta})}{\sum_{z}(\tau_{xz}^{\alpha})(n_{xz}^{\beta})} \qquad (3.10.)$$

p_{xy}^{k}: k karıncasının x düğümünden y düğümüne geçme olasılığı,

T_{xy}: x ve y değerleri arasında ki feromon değerleri,

η_{xy}: x ve y değerleri arasında ki sezgisel değerleri,

a: feromon katsayısı,

β: sezgisel katsayısı,

z: düğümler kümesidir.

Genel itibari ile KKO algoritması bu şekildedir. Algoritmanın genel adımları aşağıda açıklanmaktadır;

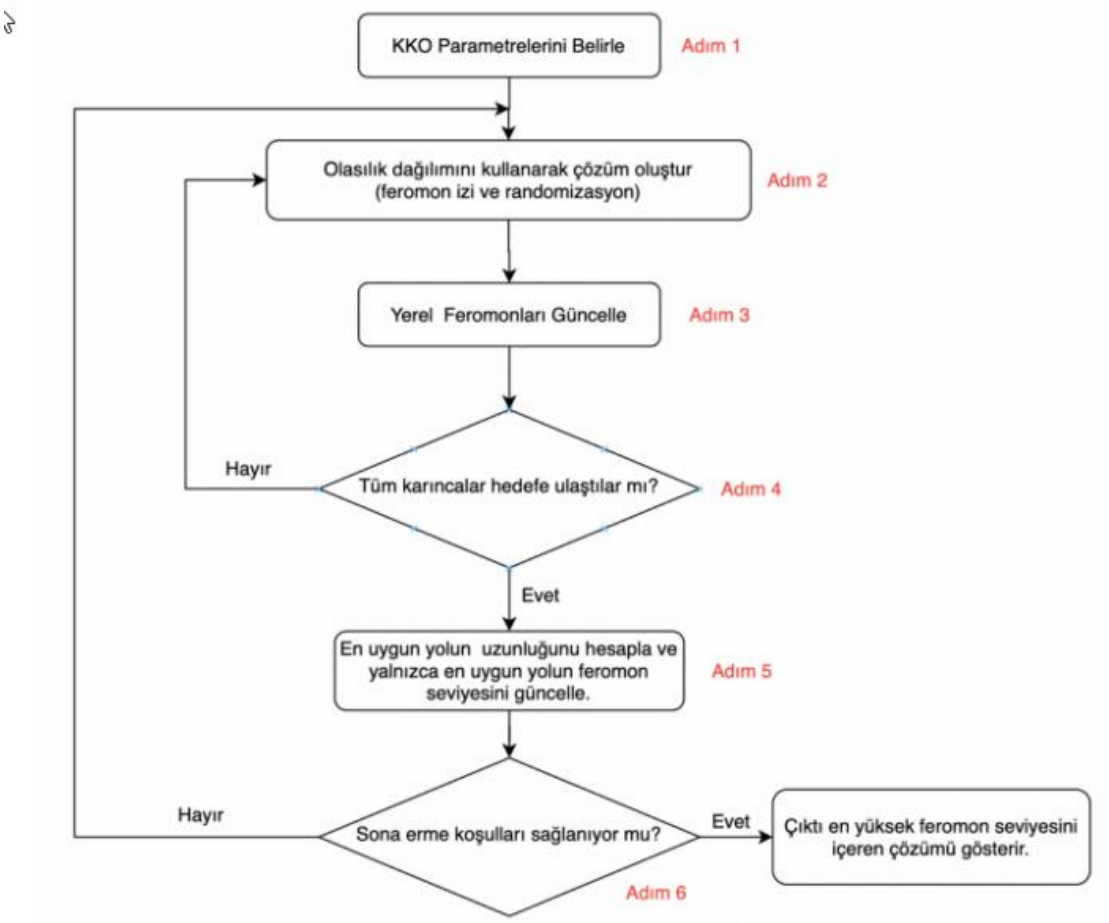
Adım 1: Karıncalar oluşturularak rastgele konumlara yerleştirilir.

Adım 2: Karıncalar alfa ve beta değerlerine göre rotaları oluştururlar.

Adım 3: Her karıncanın hedef ile arasında ki rota mesafesi hesaplanır.

Adım 4: Belirtilen iterasyon değerine kadar bu işlemler tekrar edilir.

Şekil 3.6.'da KKA şeması gösterilmektedir.



Şekil 3.6. KKA Yapısı

Bölüm 5.3'de KKO algoritması kullanılarak gerçekleştiren uygulama ve sonuçları açıklanmaktadır.

3.4. Unity Real-Time Development Platform

Unity3D, Unity Teknoloji firması tarafından geliştirilmiş kullanım durumuna göre ücretli veya ücretsiz olarak kullanılabilen bir grafik geliştirme motorudur. İçerisinde C# programlama dili bulunması sayesinde birçok başarılı simülasyon vb. uygulama gerçekleştirilebilir.

Unity3D grafik motoru, Unity Asset Store sayesinde Unity Technologies şirketi tarafından geliştirilen ve içerisinde yeteri miktarda kullanıma açık hazır tasarlanana

modelleride kullanıcılara sunmaktadır. Bu tez çalışmasında da kullanılan hedef araç ve Drone'ların tasarımları Asset Store üzerinden temin edilmiştir. Tez çalışmasında kullanılan modeller Şekil 3.7 ve Şekil 3.8'de gösterilmektedir.

Şekil 3.7. Hedef Araç

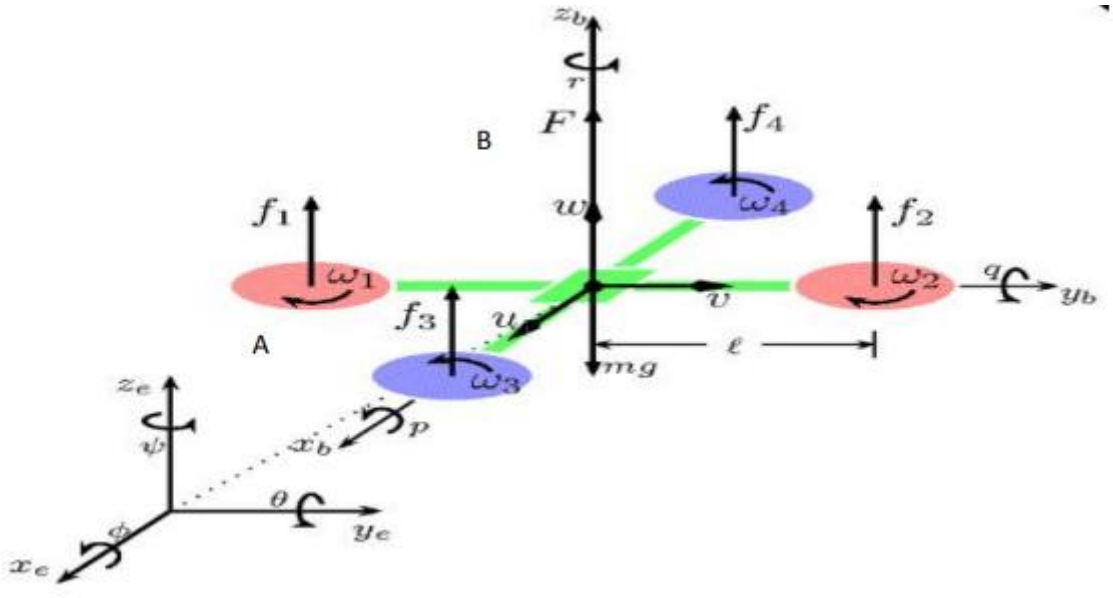
Şekil 3.8. Simülasyonda Kullanılan Drone

4. DRONE MATEMATİKSEL MODELİ

Sistem kontrollerinin sağlanması için öncelikli olarak dinamik bir sistemin oluşturulması gerekmektedir. Drone modellenmesi gerçekten karmaşık bir yapıdır. Bu yapıyı oluştururken özellikle simülasyon ortamında bazı değerleri net bir şekilde kabul etmek gerekir. Bu kabullerin yanı sıra sistemi gerçek hale dönüştürmek için Drone'un etkisi altında kaldığı fiziksel tepkileri dikkate almak gerekir. Drone'un dinamik modeli oluşturulurken, katı bir yapıda olduğu, 4 pervaneli simetrik yapıda olduğu, Drone'a etkiyen aerodinamik tepkilerin hızın karesi ile doğru orantıda olduğu, eylemsizlik matrisinin köşegen olduğu kabul edilmiştir.

Drone phi(ϕ), theta(θ), psi(ψ) açısal koordinatlarıında ve x, y, z eksenlerinde hareket edebilen 6 serbestlik dereceli insansız hava aracıdır(Kuzu, 2018). Bu çalışmada ki Drone sabit açıya sahip, bağımsız hıza sahip dört rotor içermektedir. 6 serbestlik derecesindeki hareketleri tanımlamak için Newton-Euler dinamik hareket denklemleri kullanılmaktadır.

Drone'un altı serbestlik derecesi yönelimini tanımlayan yunuslama (θ), yalpalama (ϕ), ve sapma (ψ) açıları ile 3 boyutlu uzayda doğrusal hareketi tanımlayan x, y, z eksenleri ile temsil edilir. Şekil 4.1'de Drone'un atalet referans ekseni B, yeryüzü referans ekseni A, Drone'a etkiyen ana kuvvetler f_1, f_2, f_3, f_4 mg ve pervanelere ait dönme yönleri gösterilmektedir(Kuzu, 2018).



Şekil 4.1. Drone'a etkiyen ana kuvvetler ve pervanelere ait dönme yönleri(Kuzu, 2018)

Drone'un kontrolünün sağlanması için motorların hızı ve dönüş yönleri değiştirilerek moment oluşturulur. Ω_i, pervanelerin açısal hızları ve i= (1, 2, 3, 4) olmak üzere pervanelerin hızlarından kaynaklı meydana gelen kaldırma kuvveti F_i Denklem (4.1)'de gösterilmektedir;

$$F_i = b\Omega_i^2 \tag{4.1}$$

olur. İfade de b itme faktörüdür ve sabit bir değerdir. Drone'a uygulanan toplam kaldırma kuvveti Denklem (4.2)'de gösterilmektedir;

$$F = b\sum_{i=1}^{4}\Omega_i^2 \tag{4.2}$$

Bu kuvvetten kaynaklı meydana gelen ivme Denklem (4.3)'de gösterilmektedir;

$$a_F = \frac{b}{m}\sum_{i=1}^{4}\Omega_i^2 \tag{4.3}$$

a_F ivmesinin A çerçevesine göre ifadesi R olarak toplam ivme ifadesi kuvvet dengesinden Denklem (4.4)'de ki gibi açıklanabilir;

$$\dot{v} = -ge + Re_{za_F} \tag{4.4}$$

Buradaki e_z= [0 0 1] Şeklinde bir vektör olup z eksenindeki büyüklükleri ifade etmek için kullanılır.

Drone, kendi eksenleri etrafında w açısal hızlarıyla döndüğü için açısal momentumlar meydana gelmektedir ve Denklem (4.5)'de ki gibi açıklanabilir;

$$L_{x,y,z} = I\omega \tag{4.5}$$

Burada 3x3 bir matris olan Drone gövdesinin x,y ve z eksenlerindeki ataletidir ve Denklem (4.6)'da ki gibi hesaplanabilir;

$$I = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \tag{4.6}$$

Tork, açısal momentumun zamana göre değişimidir. Bundan dolayı Drone'un açısal hızlarından meydana gelen tork Denklem (4.7) ve (4.8)'de gösterilmektedir;

$$T_B = \dot{L} \tag{4.7}$$

$$T_B = \omega * I\omega + I\dot{\omega} \tag{4.8}$$

Burada * vektörel çarpımdır.

Drone'un gövdesinin ve pervanelerin kendi eksenleri etrafından dönmesinden dolayı ortaya çıkan gyroskopik tork Denklem (4.9)'da gösterilmektedir;

$$T_B = \sum_{i=1}^{4} J(\omega x e_z)\Omega_i(-1)^i \tag{4.9}$$

Burada J, bir adet rotorun ataletidir.

Her bir pervanenin gerçekleştirmiş olduğu dönme hareketinden dolayı meydana gelen kaldırma kuvvetleri, Drone'a etkiyen torklar gerçekleştirir. Bir eksen boyunca oluşan tork, diğer eksende bulunan pervanelerin oluşturduğu torkların farkına eşittir(Kuzu, 2018). x, y ve z eksenleri boyunca pervanelerin gerçekleştirdiği torklar Denklem (4.10)'da ki gibi gösterilmektedir;

$$T_a = \begin{bmatrix} lb(\Omega_4^2 - \Omega_2^2) \\ lb(\Omega_3^2 - \Omega_1^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{4.10}$$

l, rotorla Drone'un merkezi arası mesafe ve d, sürüklenme faktörüdür. Buradan,

$$T_G + T_B = T_a \tag{4.11}$$

$$T_B = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} * \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \tag{4.12}$$

$$T_G = J\left(\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} * \begin{bmatrix}0\\0\\1\end{bmatrix}\right)(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) \tag{4.13}$$

çıkarımı yapılır. Tork dengesi denklem (4.14)'de ki hesaplanmaktadır;

$$J\left(\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} * \begin{bmatrix}0\\0\\1\end{bmatrix}\right)(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} * \begin{pmatrix}I_x & 0 & 0\\ 0 & I_y & 0\\ 0 & 0 & I_z\end{pmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} +$$

$$\begin{pmatrix}I_x & 0 & 0\\ 0 & I_y & 0\\ 0 & 0 & I_z\end{pmatrix}\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}lb(\Omega_4^2 - \Omega_2^2)\\ lb(\Omega_3^2 - \Omega_1^2)\\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)\end{bmatrix} \tag{4.14}$$

açısal ivmeler denklem (4.15), (4.16), (4.17) ve (4.18)'de ki gibi hesaplanmaktadır;

$$\ddot{\phi} = \dot{\psi}\dot{\theta}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J}{I_x}\dot{\theta}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + \frac{l}{I_x}b(\Omega_4^2 - \Omega_2^2) \tag{4.15}$$

$$\ddot{\phi} = \dot{\psi}\dot{\phi}\left(\frac{I_x - I_z}{I_y}\right) - \frac{J}{I_y}\dot{\phi}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + \frac{l}{I_y}b(\Omega_3^2 - \Omega_1^2) \tag{4.16}$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\left(\frac{I_x - I_z}{I_y}\right) + \frac{d}{I_y}(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{4.17}$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\0\\-g\end{bmatrix} + \begin{bmatrix}C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta S\phi + S\psi C\phi\\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta S\phi - C\psi C\phi\\ -S\theta & C\theta S\phi & C\theta C\phi\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\frac{b}{m}\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{4.18}$$

x, y, z denklem (4.19), (4.20) ve (4.21)'de ki gibi gösterilmektedir;

$$\ddot{x} = (C\psi S\theta S\phi + S\psi C\phi)\frac{b}{m}(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{4.19}$$

$$\ddot{y} = (S\psi S\theta C\phi - C\psi S\phi)\frac{b}{m}(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (4.20)$$

$$\ddot{z} = -g + (C\theta C\phi)\frac{b}{m}(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (4.21)$$

Drone sisteminin girişleri olan U_1, U_2,U_3 , U_4 motorların giriş gerilimleridir.

$$U = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix} \quad (4.22)$$

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (4.23)$$

$$U_2 = b(\Omega_4^2 - \Omega_2^2) \quad (4.24)$$

$$U_3 = b(\Omega_3^2 - \Omega_1^2) \quad (4.25)$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (4.26)$$

Pervanelerin açısal hızlarının karesi aşağıdaki denklemleri Denklem (4.27), (4.28), (4.29) ve (4.30)'daki gibi gösterilmiştir.

$$\Omega_1^2 = \frac{U_1}{4b} + \frac{U_4}{4d} + \frac{U_3}{2b} \quad (4.27)$$

$$\Omega_2^2 = \frac{U_1}{4b} + \frac{U_4}{4d} - \frac{U_2}{2b} \quad (4.28)$$

$$\Omega_3^2 = \frac{U_1}{4b} - \frac{U_3}{2b} - \frac{U_4}{4d} \quad (4.29)$$

$$\Omega_4^2 = \frac{U_1}{4b} + \frac{U_4}{4d} + \frac{U_2}{2b} \quad (4.30)$$

Drone'un genel açısal hızı denklemi denklem (4.31)'de ki gibi olur,

$$\Omega_r = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (4.31)$$

Sonuç olarak Drone'un hareket denklemleri Denklem (4.33), (4.33), (4.34), (4.35), (4.36) ve (4.37)'de ki olur;

$$\ddot{\phi} = \dot{\psi}\dot{\theta}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J}{I_x}\dot{\theta}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + \frac{l}{I_x}U_2 \tag{4.32}$$

$$\ddot{\theta} = \dot{\psi}\dot{\theta}\left(\frac{I_x - I_z}{I_y}\right) - \frac{J}{I_y}\dot{\phi}(-\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4) + \frac{l}{I_y}U_3 \tag{4.33}$$

$$\ddot{\psi} = \dot{\phi}\left(\frac{I_x - I_z}{I_y}\right) + \frac{1}{I_y}U_4 \tag{4.34}$$

$$\ddot{x} = (C\psi S\theta S\phi + S\psi C\phi)\frac{1}{m}U_1 \tag{4.35}$$

$$\ddot{y} = (S\psi S\theta C\phi - C\psi S\phi)\frac{1}{m}U_1 \tag{4.36}$$

$$\ddot{z} = -g + (C\theta C\phi)\frac{1}{m}U_1 \tag{4.37}$$

4.1 Drone'un Doğrusal Modeli

Doğrusal dinamikler, doğrusal olmayan dinamik sistemin denge noktası etrafında doğrusallaştırılması ile elde edilir(Kuzu, 2018). Doğrusal kontrol çalışmaları, kapalı çevrim sistemlerinin performans, kararlılık ve sağlamlılığı hakkında kesin ölçütlere ulaşmasını sağlamaktadır (Özbek, 2010). Elde edilen doğrusal olmayan matematiksel denklemler Taylor seri açılımından ilk terimler alınarak doğrusallaştırılır. Doğrusal olmayan denklemler bir çalışma noktası seçilerek o çalışma noktası etrafında doğrusallaştırılmıştır.

Drone'un havada sabit olarak durabilmesi için rotorun ürettiği toplam itki miktarının Drone'un toplam ağırlığına eşit olması gerekir. Açılar Denklem (4.38.), (4.39.) ve (4.40.)'da ki gibi gösterilir;

$$[\phi \quad \theta \quad \psi] = [0 \quad 0 \quad 0]U_1 = g \tag{4.38.}$$

$$\ddot{\phi} = \frac{U_3}{I_x} \quad (4.39.)$$

$$\ddot{\theta} = \frac{U_2}{I_y} \quad (4.40.)$$

$$\ddot{\psi} = \frac{U_4}{I_z} \quad (4.41.)$$

$$\ddot{x} = \frac{U_1}{m} sin\,\theta \quad (4.42.)$$

$$\ddot{y} = -\frac{U_1}{m} sin\,\phi \quad (4.43)$$

$$\ddot{z} = \frac{U_1}{m} - g \quad (4.44)$$

4.2. Doğrusal Durum Uzay Modeli

Drone'un hareket denklemleri doğrusal bir sisteme dönüştürülerek, durum uzay formuna çevrilmesi Drone'un kontrolü için gereklidir. Durum uzay modeli sistemin davranışlarını tanımlayan ve durum denklemleri olarak adlandırılan birinci derece denklemlerden oluşan bir denklem takımıdır(Kuzu, 2018). Durum denklemleri matris ve vektör gösterim yöntemleri kullanılarak ifade edilir. Durum değişkeni modeliyle, herhangi bir anda bir sistemin dinamik davranışı o sistemin durum değişkenleri cinsinden tanımlanır.

Genel olarak r girişli m çıkışlı bir sistemin durum denklemi Denklem (4.45.) ve çıkış denklemi Denklem (4.46.)'da ki gibi gösterilir;

$$\dot{x} = Ax + Bu \quad (4.45.)$$

$$y = Cx + Du \quad (4.46.)$$

Burada;

x: durum vektörü,

u: kontrol vektörü

A: sistem matrisi,

B: giriş matrisi,

C: çıkış matrisi,

D: doğrudan iletim matrisi

y: çıkış vektörü matrisidir.

Drone için durum vektörü Denklem (4.47.)'de ki gibi oluşturulabilir.

$$x = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \theta & \dot{\theta} & \phi & \dot{\phi} & \psi & \dot{\psi} \end{bmatrix} \tag{4.47.}$$

Drone için x, y, z konumları ve açısı çıkış olarak belirlenmiştir.

$$y = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T \tag{4.48}$$

Drone'un belirtilen giriş ve çıkışları için durum uzay gösterimi Denklem (4.49).'da ki gibi olmaktadır.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{g}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_x} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{4.49.}$$

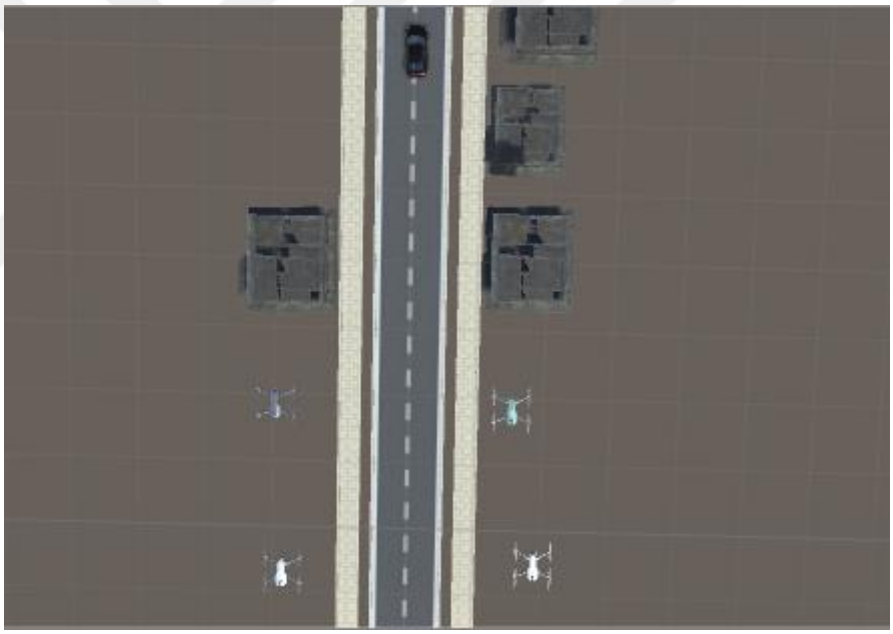
Denklem (4.49)'da elde edilen matematiksel çıkarım 4 rotorlu bir Drone için geçerlidir. Bu çıkarımda motorların voltaj değeri, Drone'nun ağırlığı gibi etkenler belirlenerek gerçek bir sistemde kullanılabilir. Tez çalışmasının bir sonraki bölümünde burada çıkarımını yaptığımız 4 rotorlu Drone'u simülasyon ortamında modelleyeceğiz.

5. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu bölümde, Drone sürüsü hedef takip optimizasyonu çalışmaları gerçekleştirilmiş ve sonuçları verilmiştir. Burada hedef takip optimizasyonu için, PSO, Karınca Kolonisi Algoritması ve Yapay Arı Kolonisi algoritmaları belirlenen amaç fonksiyonuna göre sırasıyla çalıştırılmış ve sonuçları elde edilmiştir. Çalışması yapılan 3 adet algoritmanın sonuçları bölüm 5.5'de açıklanmaktadır.

5.1 Deneysel Düzen ve Amaç Fonksiyonu

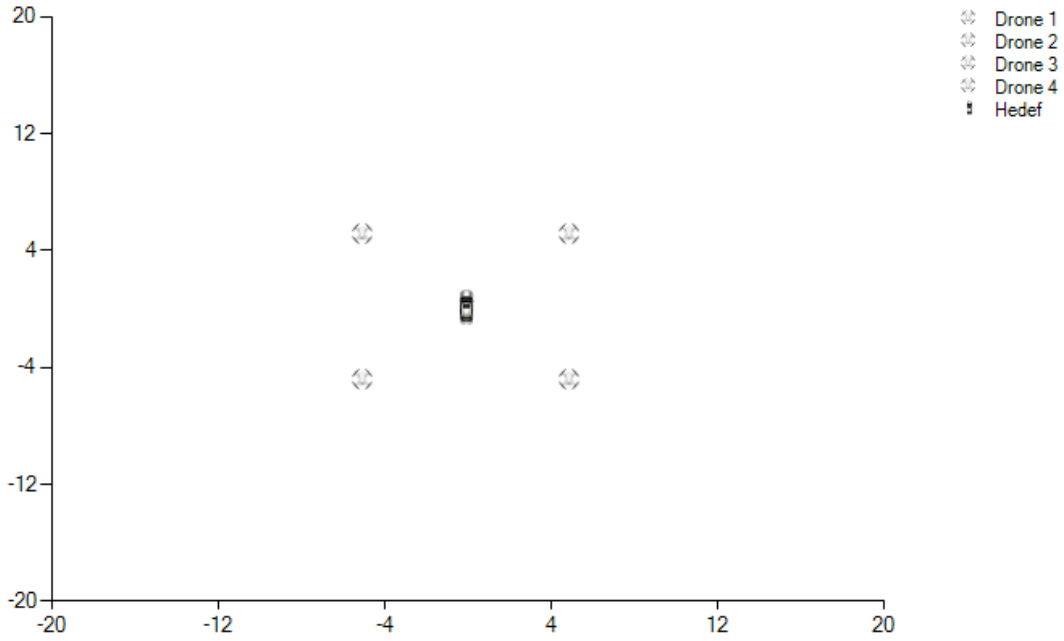
Bu simülasyon ortamında ilk olarak 4 adet Drone ve 1 adet hedef olarak tanımlanmış araç oluşturulmuştur. Şekil 5.1.'de oluşturulan simülasyon ortamının şekli gözükmektedir.



Şekil 5.1. Tasarlanan Simülasyon Ortamı

Aracın hızı sabittir ve saatte 10 birim olarak ayarlanmıştır. Simülasyon ortamında kullanılan Drone'lar 4 rotora sahiptir. Motorların gerilimi ve ağırlığının Drone'a etkisi matematiksel model çıkarımında anlatılmıştı. O yüzden simülasyon ortamında bu değerlerin etkisi yok sayılacaktır. Test düzeneğinde ki amaçlardan ilki Drone'ların hedef olarak belirlenmiş kara aracının üzerinde kare veya kareye benzer bir pozisyon almalarıdır. Bunda ki temel kriter 2 Drone'nun hedef araca göre y ekseninden

ileride diğer 2 Drone'nun ise hedef araca göre y ekseninden daha geride pozisyon almalarıdır. Örnek alarak beklenen pozisyon şekli Şekil 5.2.'de gösterilmektedir.



Şekil 5.2. Droneların Araca Göre Oluşturması Gereken Konumları

Test düzeneğinde ki ilk amaç konum şekli her algoritma için gerçekleştirildikten sonra hedef aracı imha etme testi gerçekleştirilecektir. Burada ki kıstas Droneların araca göre oluşturdukları konumlarının araca uzaklığını ölçerek simülasyon ortamında 1 atış yapmalarıdır. Merminin araca olan etkisi 5 birim uzaklıktan atıldığında maksimum kabul edilmektedir. Bu 5 birimlik değer tamamen tüm algoritmaların konumlarından yarattıkları etkiyi ölçebilmek için kabul edilen bir değerdir. Toplam da 4 adet Drone olduğu için ve her Drone'nun bir atış yaptığı varsayılarak, her Drone'nun araca verebileceği hasar değeri Denklem (5.1.) ve Denklem (5.2.)'de açıklanmaktadır.;

$$th = \alpha * \beta * (5 / x) \qquad (5.1.)$$

$$a = 1 - (V_{araç} / 100) \qquad (5.2.)$$

Burada

th: Oluşan Hasar,

x: Drone'nun araca göre mesafesidir.

a : Araç hızına göre oluşabilecek hasar

β : Yol katsayısı, 0.5 seçildi.

Bu şekilde 4 adet Drone'nun oluşturduğu hasarlar toplanarak araca verdikleri toplam hasar değeri hesaplanmış olacaktır.

Algoritmalarda testler için kullanılan amaç fonksiyonu Denklem (5.3.)'de gösterilmektedir. Amaç fonksiyonu belirlenirken algoritmaların hızı ve verdikleri hasarı ölçen bir fonksiyon türetilmiştir. Öncelikli olarak Drone'ların belirli konumlara gelmesi hedeflendiği için amaç fonksiyonuna bu hedeflenen konumların oluştukları zaman eklenmiştir. Daha sonra verdikleri hasarın integrali alınarak 0.5 ile çarpılmıştır. İntegral belli aralıklardaki toplam değişimi veya başka bir değişle biriken değişim miktarını bulmamızı sağladığı için alınmıştır. 0.5 ile çarpmanın sebebi ise verilen hasar değerinin puanlandırmada süreye göre daha yüksek seviyede olmasıdır.

$$f(x) = it - (1/2)\int (2 - th)^6 \, dth \tag{5.3.}$$

Bu denklemde; it iterasyon sayısını, th ise toplam hasarı ifade etmektedir.

5.2 Parçacık Sürü Optimizasyonu Algoritması ile Hedef Takip Optimizasyonu

Parçacık Sürü Optimizasyonu algoritması yapısı detaylı olarak bölüm 3.1'de anlatılmıştır. Burada hedef takip optimizasyonu için bu algoritmanın nasıl kullanıldığı ve parametre değerlerinin neler olduğu açıklanacaktır.

Algoritmanın çalışma prensibine göre başlangıçta Drone'lar kalkış noktasına yerleştirilir. Drone'lar PSO algoritmasından belirlenen Denklem (3.1.)'e göre pozisyonlarını belirlerler. Her iterasyonda konum değişikliği değişken olup sürüde ki en iyi konuma iterasyonlar içerisinde ki en iyi konuma ve hedef aracın konumuna bağlı olarak değişmektedir. Bu döngü belirlenen iterasyon değeri kadar kendi içerisinde devam etmektedir.

Parçacık sürü optimizasyonu algoritmasında kullanılan parametre değerleri;

w=0.7 (Genellikle 1'e yakın seçilir.)

c1=0.6 (0 ile 1 aasında rastgele bir değerdir.)

c2=0.6 (0 ile 1 arasında rastgele bir değerdir.)

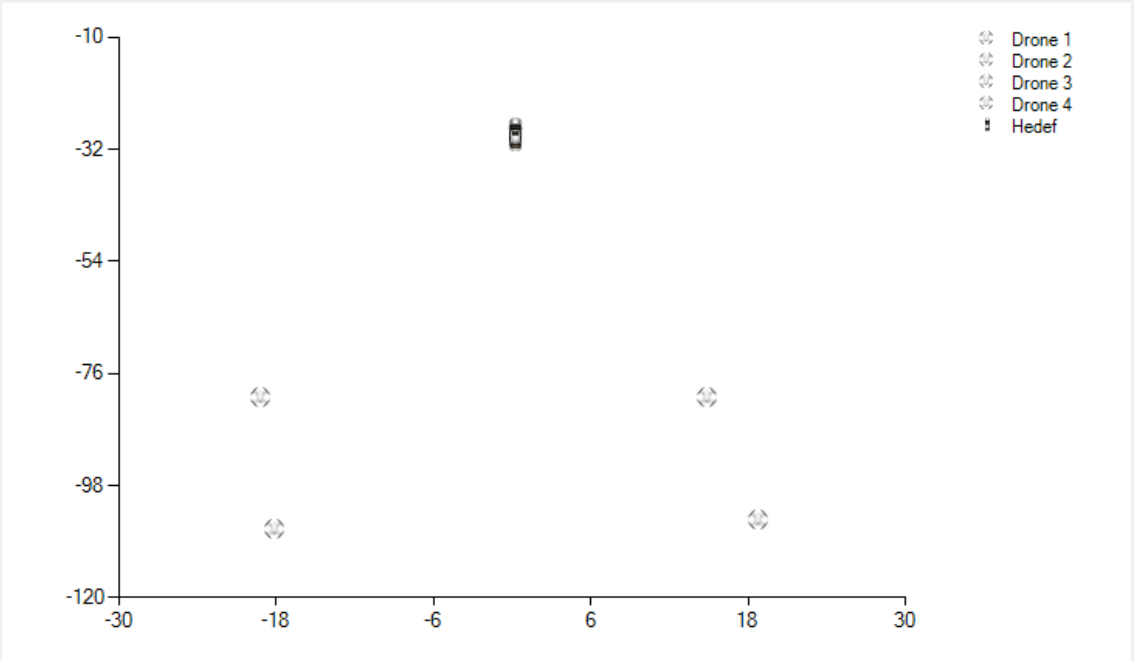
PSO algoritmasına göre Drone'ların oluşturduğu konumlar aşağıdaki tablo ve şekillerde gösterilmektedir.

Tablo 5.1.'de 1. İterasyon, 10. İterasyon, 29. İterasyon ve 60. İterasyonda hedef araç ve Drone'ların konumları yazmaktadır.

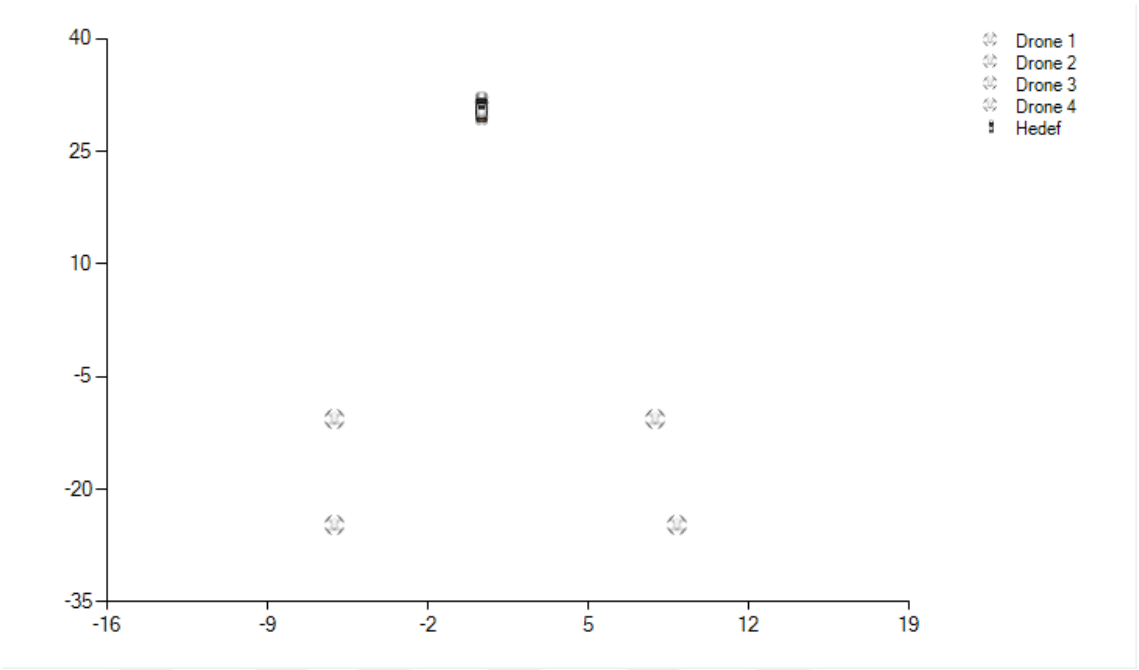
Şekil 5.3.'de 1. İterasyonda Drone'ların araca göre konumları, Şekil 5.4'de 10. İterasyonda Drone'ların araca göre konumları, Şekil 5.5'de 17. İterasyonda Drone'ların araca göre konumları değerleri, Şekil 5.6'de 29. İterasyonda Drone'ların araca göre konumları ve Şekil 5.7'da 60. İterasyonda Drone'ların araca göre konumları gösterilmektedir.

Tablo 5.1. İterasyon Sonuçları

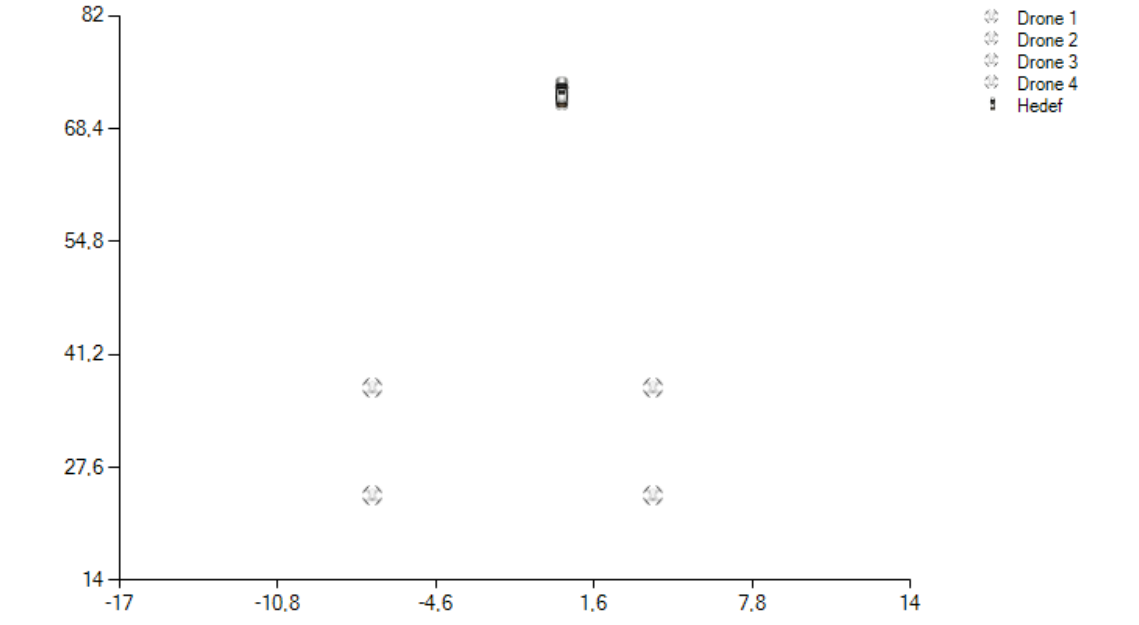
	1. İterasyon		10. İterasyon		17. İterasyon		29. İterasyon		60. İterasyon	
	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni
Hedef Araç	0.4	-29.50	0.4	30.44	0.4	72.44	0.4	144	0.4	276
Drone 1	19	-105	9	-25	4	24	5	107	8	240
Drone 2	15	-81	8	-11	4	37	5	121	7	285
Drone 3	-18	-107	-6	-25	-7	24	-8	107	-4	246
Drone 4	-19	-81	-6	-11	-7	37	-8	121	-6	286



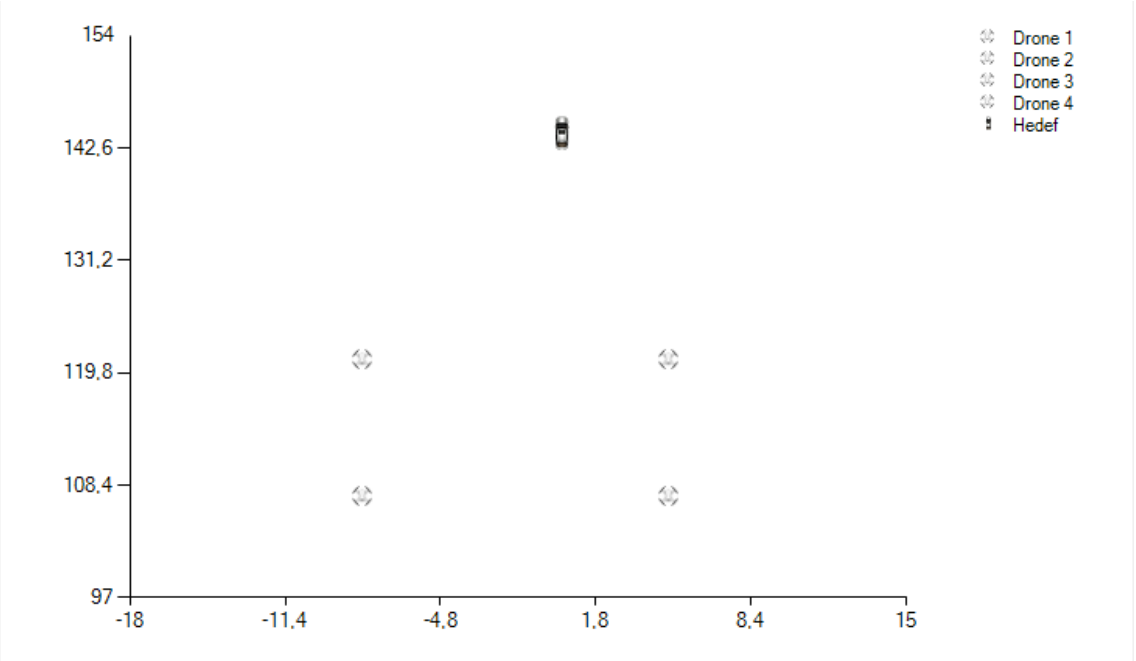
Şekil 5.3. 1. İterasyon Sonuçları



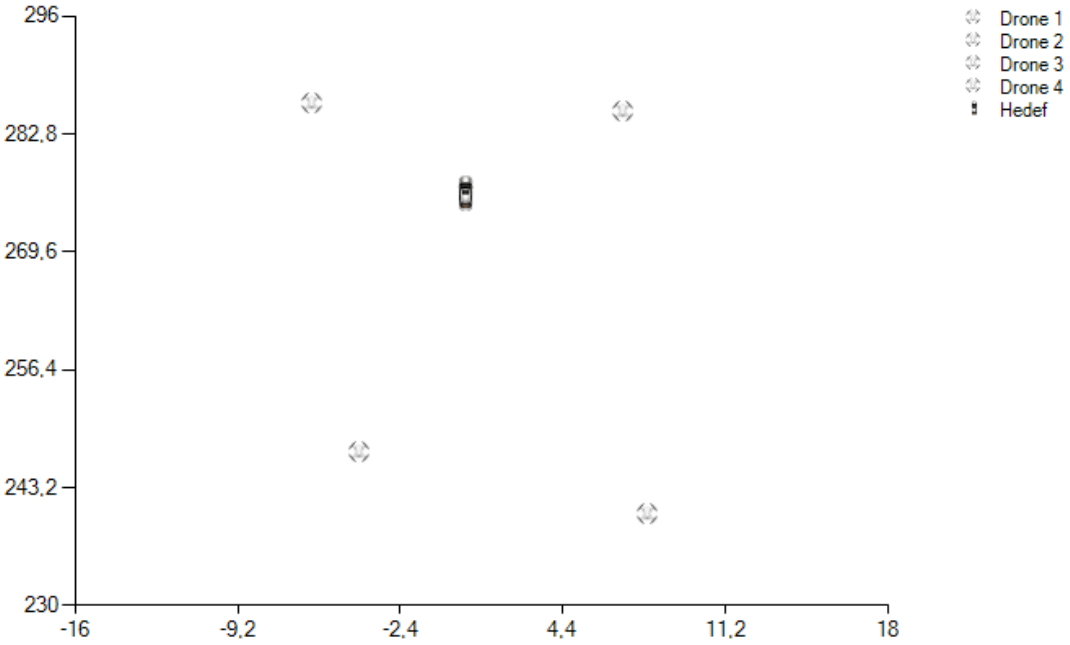
Şekil 5.4. 10. İterasyon sonuçları



Şekil 5.5. 17. İterasyon sonuçları



Şekil 5.6. 29. İterasyon sonuçları



Şekil 5.7. 60. İterasyon Sonuçları

Parçacık sürü optimizasyonu algoritması kullanılarak 60. iterasyondan sonra Dronelar hedef olarak belirlenen araba etrafında dairesel yörünge üzerinde kare benzeri pozisyonlarını almışlardır. Droneların araca göre konumları Şekil 5.3, Şekil 5.4, Şekil 5.5, Şekil 5.6 ve Şekil 5.7'de gösterilmektedir. PSO algoritmasına göre mermi atışları

60. İterasyondan itibaren başlayacaktır. Buna göre 1. merminin verdiği hasarın hesaplanması;

- Şekil 5.7.'de (-6, 286) konumunda duran Drone'nun araca göre uzaklığı 11.96 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.189$ hasar meydana getirmektedir.
- Şekil 5.7.'de (7, 285) konumunda duran Drone'nun araca göre uzaklığı 11.16 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.198$ hasar meydana getirmektedir.
- Şekil 5.7.'de (-4, 246) konumunda duran Drone'nun araca göre uzaklığı 30.32 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.072$ hasar meydana getirmektedir.
- Şekil 5.7.'de (8, 240) konumunda duran Drone'nun araca göre uzaklığı 36.56 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.0585$ hasar meydana getirmektedir.

Bölüm 5.5'de araç üzerinde oluşan toplan hasar hesaplanarak diğer algoritmalarla karşılaştırılacaktır.

5.3 Yapay Arı Kolonisi Algoritması ile Hedef Takip Optimizasyonu

Yapay Arı Kolonisi Algoritma yapısı detaylı olarak bölüm 3.2 de anlatılmıştır. Burada hedef takip optimizasyonu için bu algoritmanın nasıl kullanıldığı ve parametre değerlerinin neler olduğu açıklanacaktır.

Algoritmayı detaylandırmadan önce projenin verileri oluşturulur. Bu çalışmada daha önceden de bahsedildiği gibi 4 adet birbiri ile haberleşen Drone'lar ve 1 adet takip edilecek hedef araç bulunmaktadır. İlk olarak Drone'ların konumları kalkış noktasında oluşturulur. Algoritmanın çalışma prensibine göre başlangıçta rastgele yiyecek kaynakları oluşturulur. Biz ise bu başlangıç aşamasını Drone'ların kalkış noktası olarak kabul etmekteyiz. Her Drone kendi bulunduğu konumun araca göre uzaklığını kendi içinde hesaplayarak yakınlık ve uzaklık durumuna göre nektar miktarı üretir. Bu nektar miktarının değeri aynı zamanda aracın yakınlığını tespit işe yaramaktadır. Drone'lar diğer Drone'ların nektar miktar miktarlarını kontrol ederek bir sonraki iterasyon için hangi konuma gitmeleri gerektiğini hesaplamaktadırlar(Bknz. Bölüm 3.2). Her

Drone'nun konumu amaç fonksiyonundan geçirilerek hedefe yakınlığı ve aldığı pozisyon kayıt edilmektedir. Bu döngü belirlenen iterasyon değeri kadar kendi içerisinde devam etmektedir. YAK algoritmasının başlangıcında kullanılan parametreler aşağıdaki gibidir;

x_j^{min} = -105 (Başlangıç Koordinatının Minumum Değeri)

x_j^{max} = 29 (Başlangıç Koordinatının Maximum Değeri)

m = 4 (Kullanılan Drone Sayısı)

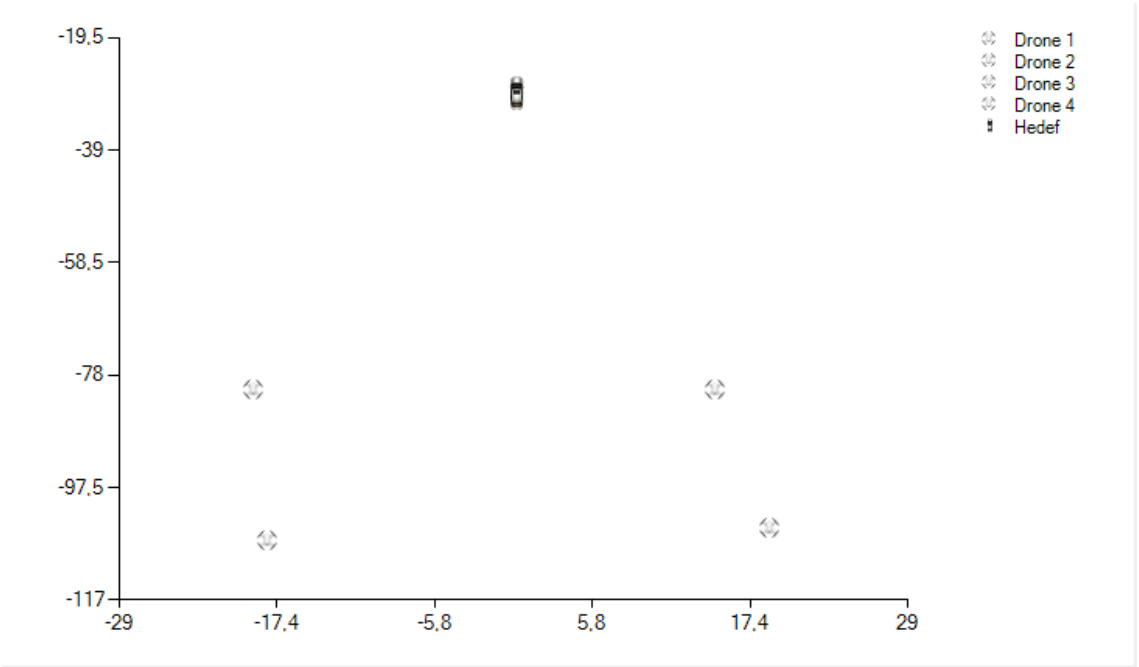
YAK algoritmasına göre Drone'ların oluşturduğu konumlar aşağıdaki tablo ve şekillerde gösterilmektedir.

Tablo 5.2'de iterasyon sonucu oluşan konum değerleri gösterilmektedir. 78. İterasyonda Drone'lar hedeflenen pozisyona gelmişlerdir.

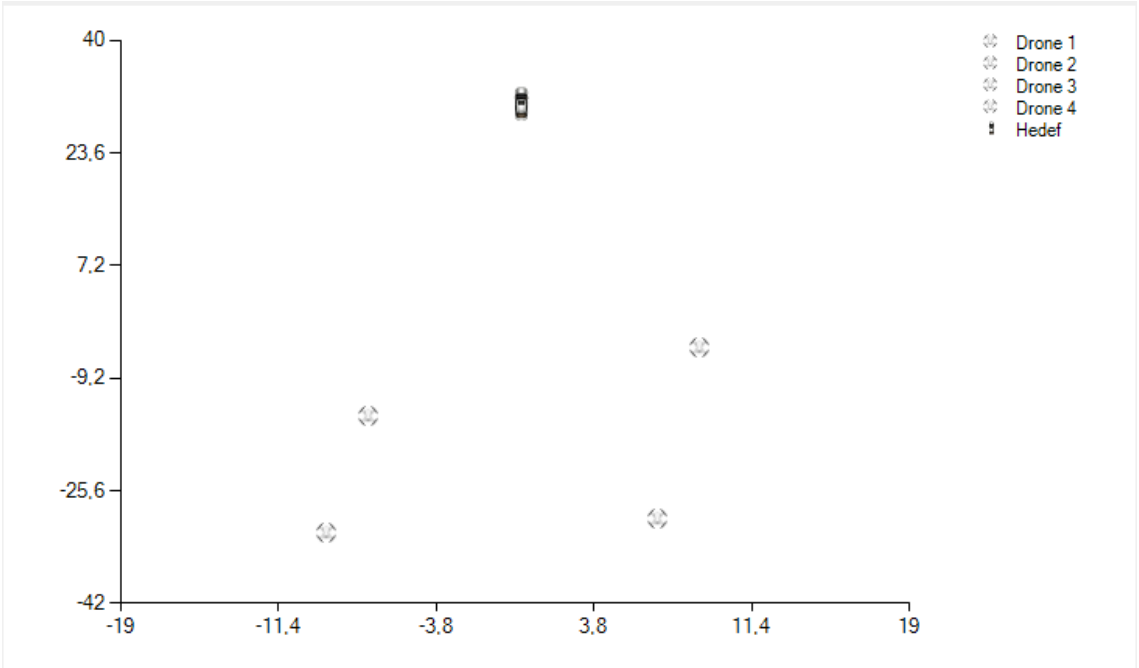
Şekil 5.8'de 1. İterasyonda Drone'ların araca göre konumları, Şekil 5.9'da 10. İterasyonda Drone'ların araca göre konumları, Şekil 5.10.'da 17. İterasyonda Drone'ların araca göre konumları değerleri, Şekil 5.11.'de 29. İterasyonda Drone'ların araca göre konumları ve Şekil 5.12'de 78. İterasyonda Drone'ların araca göre konumları gösterilmektedir.

Tablo 5.2. İterasyon Sonuçları

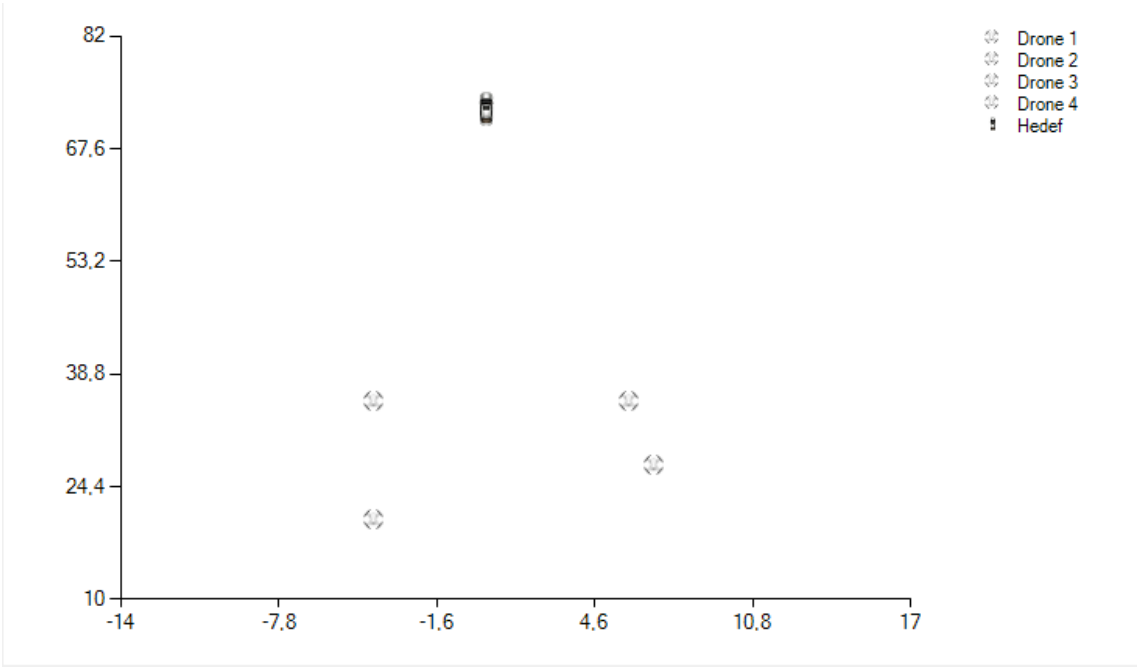
	1. İterasyon		10. İterasyon		17. İterasyon		29. İterasyon		78. İterasyon	
	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni
Hedef Araç	0.4	-29.50	0.4	30.44	0.4	72.44	0.4	144	0.4	312
Drone 1	19	-105	7	-30	7	27	4	103	3	297
Drone 2	15	-81	9	-5	6	35	4	114	11	331
Drone 3	-18	-107	-9	-32	-4	20	-7	108	-5	299
Drone 4	-19	-81	-7	-15	-4	35	-9	125	-8	322



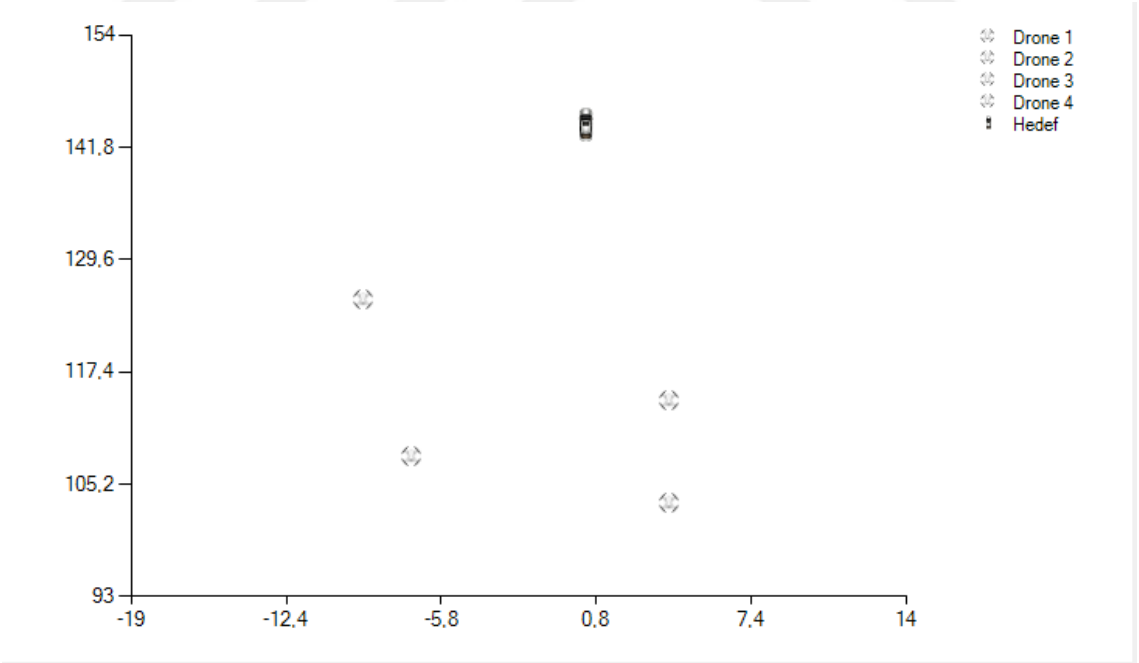
Şekil 5.8. 1. İterasyon Sonuçları



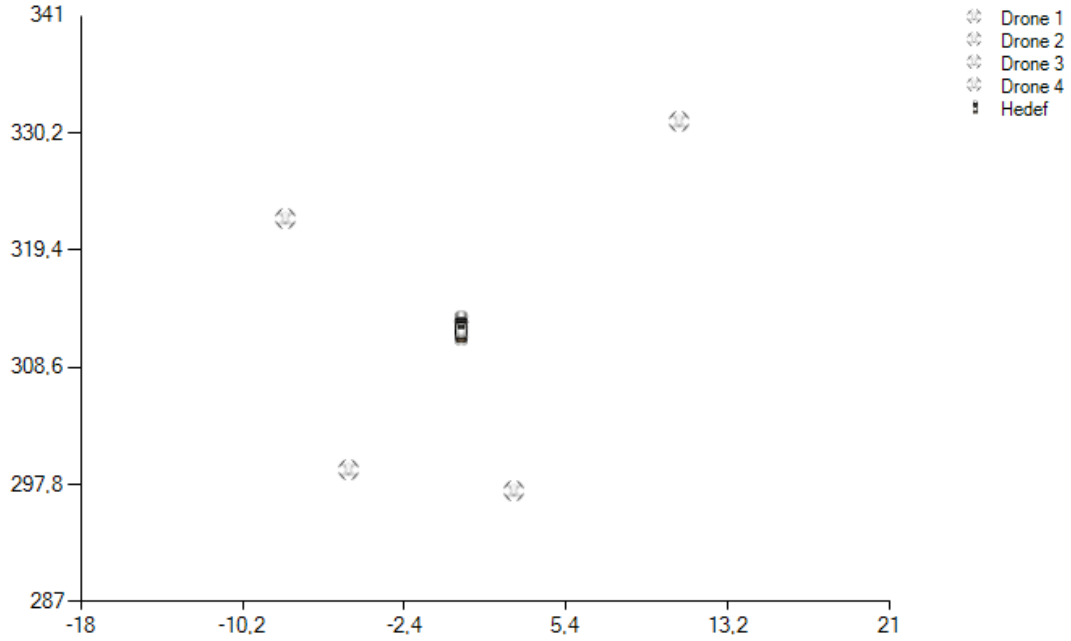
Şekil 5.9. 10. İterasyon Sonuçları



Şekil 5.10. 17 İterasyon Sonuçları



Şekil 5.11. 29. İterasyon Sonuçları



Şekil 5.12. 78. İterasyon Sonuçları

YAK algoritması kullanılarak 78. iterasyondan sonra Dronelar hedef olarak belirlenen araba etrafında alması gereken pozisyonlarını almıştır. Droneların araca göre konumları Şekil 5.8., Şekil 5.9., Şekil 5.10, Şekil 5.11. ve Şekil 5.12'de gösterilmektedir. Simülasyon ortamında ki mermi atışları YAK algoritması için simülasyon ortamında ki mermi atışı 78. İterasyonda ki konumlarında başlayacaktır. 1. merminin verdiği hasarın hesaplanması;

- Şekil 5.12'de (-8, 322) konumunda duran Drone'nun aracın tam üst noktasına göre uzaklığı 13.03 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den $th = 0.171$ hasar meydana getirmektedir.
- Şekil 5.12'de (11, 331) konumunda duran Drone'nun aracın tam üst noktasına göre uzaklığı 21.74 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.1035$ hasar meydana getirmektedir.
- Şekil 5.12'de (5, 299) konumunda duran Drone'nun aracın tam üst noktasına göre uzaklığı 14.07 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.1575$ hasar meydana getirmektedir.
- Şekil 5.12'de (3, 297) konumunda duran Drone'nun aracın tam üst noktasına göre uzaklığı 15.22 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.144$ hasar meydana getirmektedir.

Bölüm 5.5'de araç üzerinde oluşan toplan hasar hesaplanarak diğer algoritmalarla karşılaştırılacaktır.

5.4 Karınca Kolonisi Algoritması ile Hedef Takip Optimizasyonu

Karınca Kolonisi Algoritma yapısı detaylı olarak bölüm 3.3 de anlatılmıştır. Burada hedef takip optimizasyonu için bu algoritmanın nasıl kullanıldığı ve parametrelerinin neler olduğu açıklanacaktır.

Algoritmanın çalışmalarına başlarken öncelikli olarak projenin verileri ortaya çıkarılır. Bu çalışmada daha önceden de bahsedildiği gibi 4 adet birbiri ile haberleşen Drone'lar ve 1 adet takip edilecek araç bulunmaktadır. İlk olarak Drone'ların konumları kalkış yapacağı yerlerde belirlenir. KKA algoritması PSO ve YAK algoritmasına göre çalışma şekli biraz farklıdır. KKA algoritmasında yollar hesaplanırken rastgele olarak noktalar belirlenir ve her bir karınca kendisine en yakın noktaya göre hareket eder. Algoritmanın yapısına uygun olarak başlangıçta feromon değerleri 0 olarak alınır. Drone'lar hareketlenerek ilk iterasyonu gerçekleştirirler. Bu iterasyona göre sürüde konumu araca göre en yakın olan Drone algoritmadaki adlandırmaya göre feromon değeri üretir. Her Drone bulunduğu konum için uzaklığa bağlı olarak feromon değeri üretir. Bu üretilen feromon değerlerinin büyüklüğüne göre Drone'lar bir sonraki iterasyonlarda yeni konumlarını almaktadırlar. Maksimum iterasyon sayısına göre bu durum sürekli tekrarlanır. Her iterasyondan sonra hedef araç konumuna göre Drone'ların araca yakınlığı ve oluşturdukları pozisyonlar kayıt altına alınmaktadır. Algoritmaların her test aşamasında ki iterasyonlarında formülasyona bağlı olarak farklı konumlar ortaya çıkabilmektedir.

KKO algoritmasında kullanılan parametre değerleri aşağıda ki gibidir;

Karınca Sayısı: 4,

Alfa Değeri: 1,

Beta Değeri: 3.5,

Feromon Buharlaşma Değeri: 0.8.

Karınca sayısı Drone sayısı demektir. Alfa, Beta ve Feromon Buharlaşma Değeri istenilen değerlerde seçilebilir. Ancak genel olarak sırasıyla 1, 3.5 ve 0.8 değerleri kullanılır.

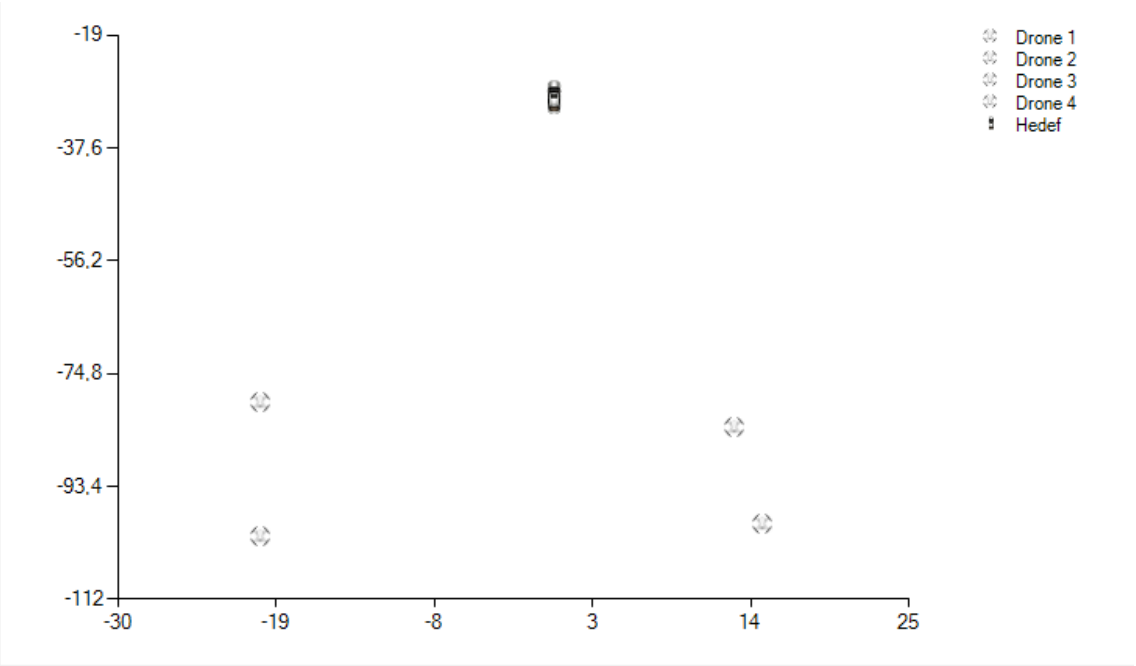
KKO algoritmasına göre Drone'ların oluşturduğu konumlar aşağıdaki tablo ve şekillerde gösterilmektedir.

Tablo 5.3'de 1. İterasyon, 10. İterasyon, 29. İterasyon ve 71. İterasyonda hedef araç ve Drone'ların konumları yazmaktadır.

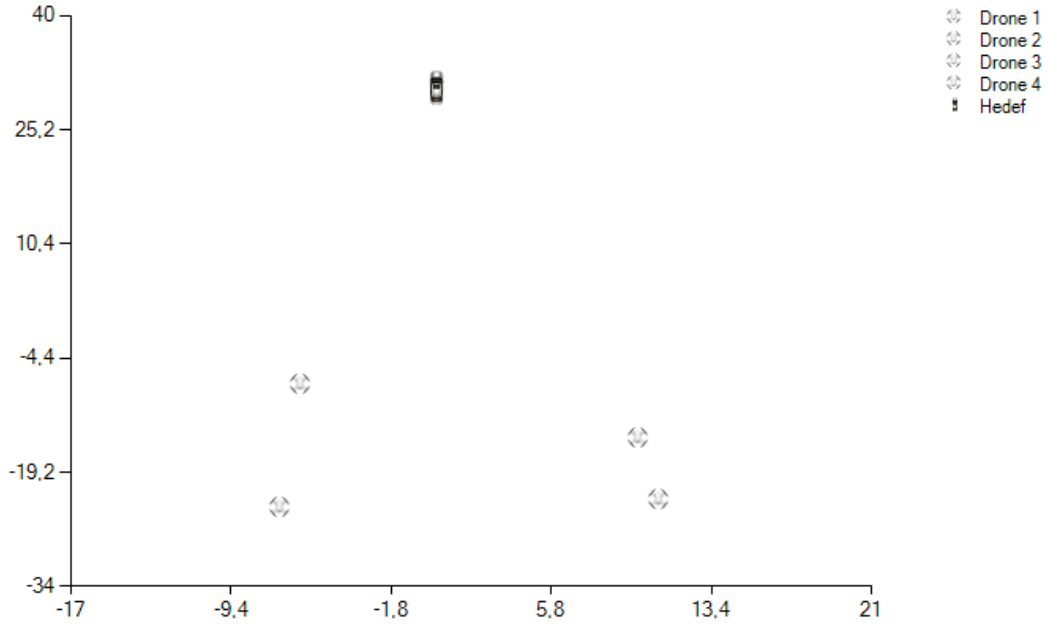
Şekil 5.13'de 1. İterasyonda Drone'ların araca göre konumları, Şekil 5.14'de 10. İterasyonda Drone'ların araca göre konumları, Şekil 5.15'de 17. İterasyonda Drone'ların araca göre konumları değerleri, Şekil 5.16'da 29. İterasyonda Drone'ların araca göre konumları ve Şekil 5.17'de 71. İterasyonda Drone'ların araca göre konumları gösterilmektedir.

Tablo 5.3. İterasyon Sonuçları

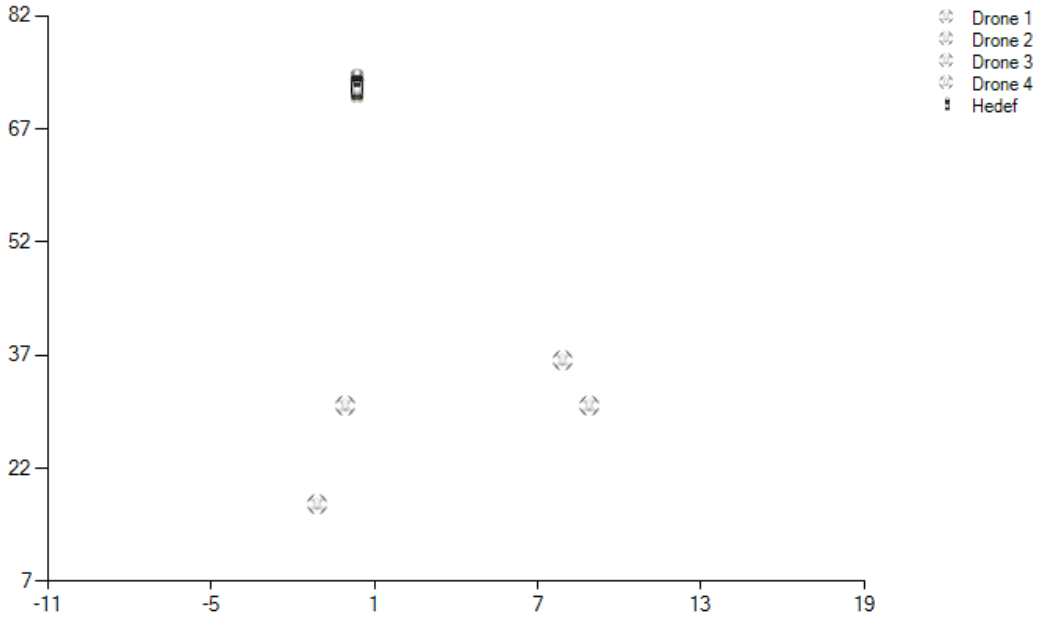
	1. İterasyon		10. İterasyon		17. İterasyon		29. İterasyon		71. İterasyon	
	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni	X Ekseni	Y Ekseni
Hedef Araç	0.4	-29.50	0.4	30.44	0.4	72.44	0.4	144	0.4	294
Drone 1	15	-100	11	-23	9	30	6	100	6	283
Drone 2	13	-84	10	-15	8	36	5	111	7	301
Drone 3	-20	-102	-7	-24	-1	17	-4	104	-9	290
Drone 4	-20	-80	-6	-8	0	30	-10	120	-5	304



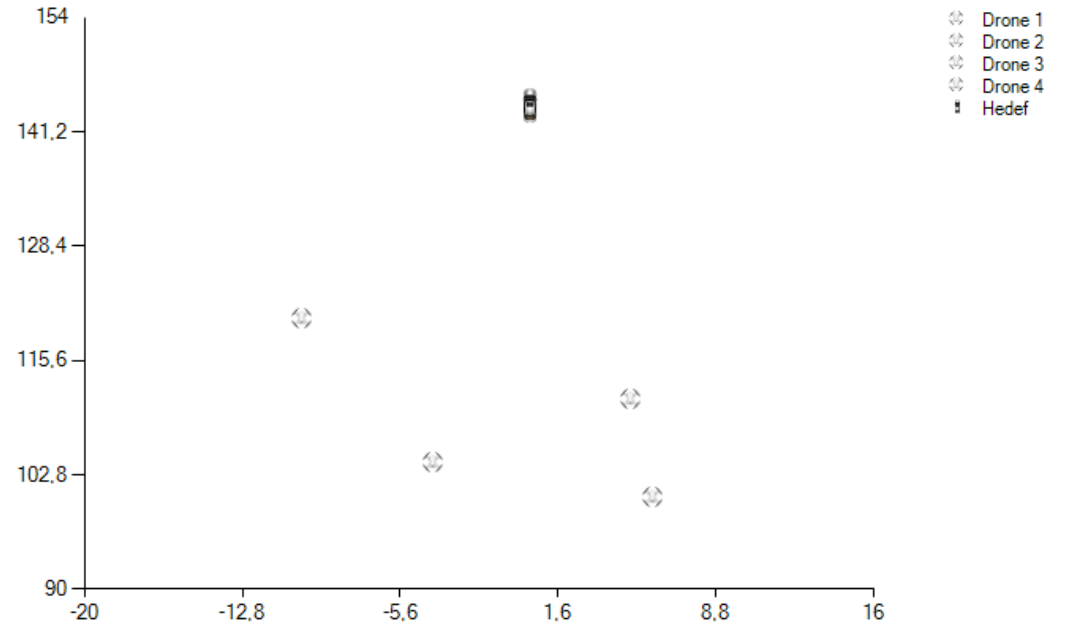
Şekil 5.13. 1. İterasyon sonuçları



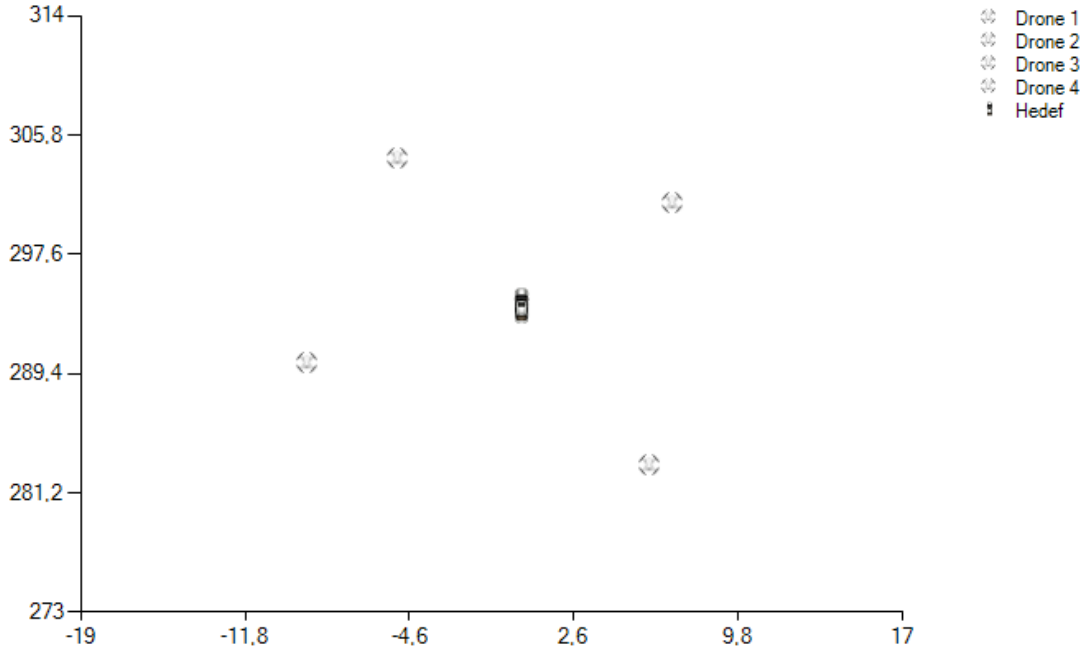
Şekil 5.14. 10. İterasyon Sonuçları



Şekil 5.15. 17. İterasyon Sonuçları



Şekil 5.16. 29. İterasyon Sonuçları



Şekil 5.17. 71. İterasyon Sonuçları

KKO algoritması kullanılarak 71. iterasyondan sonra Dronelar hedef olarak belirlenen araba etrafında alması gereken pozisyonlarını almıştır. Droneların araca göre konumları Şekil 5.13, Şekil 5.14, Şekil 5.15, Şekil 5.16 ve Şekil 5.17'de

gösterilmektedir. Simülasyon ortamında ki mermi atışları KKO algoritması için 71. İterasyonda ki konumlarında başlayacaktır. 1 merminin verdiği hasarın hesaplanması;

- Şekil 5.17'de (-5, 304) konumun da sol en üst noktada duran Drone'nun aracın tam üst noktasına göre uzaklığı 11.36 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.198$ hasar meydana getirmektedir.
- Şekil 5.17'de (7, 301) konumun da duran Drone'nun aracın tam üst noktasına göre uzaklığı 12.43 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den $th = 0.18$ hasar meydana getirmektedir.
- Şekil 5.17'de (-9, 290) konumun da duran Drone'nun aracın tam üst noktasına göre uzaklığı 10.21 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.2205$ hasar meydana getirmektedir.
- Şekil 5.17'de (7, 301) konumun da duran Drone'nun aracın tam üst noktasına göre uzaklığı 11.93 birimdir. Drone'nun bu mesafeden yaptığı atışla vermiş olduğu hasar Denklem (5.1.)'den, $th = 0.189$ hasar meydana getirmektedir.

Bölüm 5.5'de araç üzerinde oluşan toplan hasar hesaplanarak diğer algoritmalarla karşılaştırılacaktır.

5.5 Sonuçların Karşılaştırılması

Bu bölümde PSO, YAK ve KKO algoritmasının oluşturulan simülasyon test düzeneğinde elde edilen sonuçları karşılaştırılacaktır. Bölüm 5.2, Bölüm 5.3 ve Bölüm 5.4'de 4 adet Drone kullanılarak yapılmış örnek hesaplamalar ve alınan sonuçlar gösterilmiştir. Burada ayrıca algoritmalarda 3 adet Drone ve 5 adet Drone kullanılması halinde oluşan sonuçlarda ele alınacaktır. Bunun için 5 adet Drone içeren sonuçlar Bölüm 5.4'de uygulanan adımlar yürütülerek tekrar elde edilmiştir. Buna göre Tablo 5.4'de 5 adet Drone kullanıldığı zaman PSO algoritmasında Drone'ların başarılı hedef pozisyon konum değerleri gösterilmektedir. Tablo 5.5'de 5 adet Drone kullanıldığı zaman YAK algoritmasında Drone'ların başarılı hedef pozisyon konum değerleri gösterilmektedir. Tablo 5.6'da 5 adet Drone kullanıldığı zaman KKA algoritmasında Drone'ların başarılı hedef pozisyon konum değerleri gösterilmektedir. Tablo 5.11'de ise 5 adet Drone kullanımında algoritmaların kaçıncı iterasyonda hedef pozisyona geldiğini göstermektedir. Çalışmada 3 adet Drone'dan oluşan sürü için benzer sonuçlar tekrar

elde edilmiştir. Buna göre Tablo 5.7, 3 adet Drone kullanıldığı zaman PSO algoritmasında Drone'ların başarılı hedef pozisyon konum değerlerini göstermektedir. Tablo 5.8, 3 adet Drone kullanıldığı zaman YAK algoritmasında Drone'ların başarılı hedef pozisyon konum değerlerini göstermektedir. Tablo 5.9, 3 adet Drone kullanıldığı zaman KKA algoritmasında Drone'ların başarılı hedef pozisyon konum değerlerini göstermektedir. Tablo 5.12 ise 3 adet Drone kullanımında algoritmaların kaçıncı iterasyonda hedef pozisyona geldiğini göstermektedir.

Çalışmada sürü için 3, 4, 5 adet Drone kullanılması durumunda elde edilen tüm sonuçlar göz önüne alındığında, algoritmaların kendi aralarındaki başarı durumlarını aşağıdaki gibi inceleyebiliriz. 4 adet Drone kullanımında algoritmaların kendi aralarında ki başarı durumları Tablo 5.10'de, 5 adet Drone kullanımında algoritmaların kendi aralarında ki başarı durumları Tablo 5.11'de ve 3 adet Drone kullanımında algoritmaların kendi aralarında ki başarı durumları Tablo 5.12'de verilmektedir.

Tablo 5.4. 5 Adet Drone'nun PSO Algoritmasına Göre Konumları

	PSO					
Dronelar	Drone 1	Drone 2	Drone 3	Drone 4	Drone 5	Hedef Araç
X Eksenine Göre Konumu	-20,56	30,36	-14,6	-14,65	28,45	0,4
Y Eksenine Göre Konumu	156,45	144,32	176,89	163,11	147,44	162

Tablo 5.5.: 5 Adet Drone'nun YAK Algoritmasına Göre Konumları

	YAK					
Dronelar	Drone 1	Drone 2	Drone 3	Drone 4	Drone 5	Hedef Araç
X Eksenine Göre Konumu	-11,56	28,15	-22,15	-15,87	34,56	0,4
Y Eksenine Göre Konumu	169,11	138,45	199,41	184,45	142,52	182

Tablo 5.6.: 5 Adet Drone'nun KKA Algoritmasına Göre Konumları

	KKA					
Dronelar	Drone 1	Drone 2	Drone 3	Drone 4	Drone 5	Hedef Araç
X Eksenine Göre Konumu	-32,45	28,98	0,78	-29,23	26,63	0,4
Y Eksenine Göre Konumu	178,23	190,02	226,23	201,55	201,58	214

Tablo 5.7. 3 Adet Drone'nun PSO Algoritmasına Göre Konumları

	PSO			
Dronelar	Drone 1	Drone 2	Drone 3	Hedef Araç
X Eksenine Göre Konumu	-15,68	12,34	-5,56	0,4
Y Eksenine Göre Konumu	288,47	296,74	342,63	324

Tablo 5.8. 3 Adet Drone'nun YAK Algoritmasına Göre Konumları

	YAK			
Dronelar	Drone 1	Drone 2	Drone 3	Hedef Araç
X Eksenine Göre Konumu	-18,25	21,14	-1,42	0,4
Y Eksenine Göre Konumu	146,47	158,47	192,11	179

Tablo 5.9. 3 Adet Drone'nun KKA Algoritmasına Göre Konumları

	KKA			
Dronelar	Drone 1	Drone 2	Drone 3	Hedef Araç
X Eksenine Göre Konumu	-7,45	6,98	-3,21	0,4
Y Eksenine Göre Konumu	306,54	320,11	402,69	375

Tablo 5.10. Algoritmaların 4'lü Drone'a Göre Hedef Pozisyona Ulaştıkları İterasyon Sayıları

	PSO	YAK	KKO
60. İterasyon	✓		
71. İterasyon			✓
78. İterasyon		✓	

Tablo 5.11. Algoritmaların 5'li Drone'a Göre Hedef Pozisyona Ulaştıkları İterasyon Sayıları

	PSO	YAK	KKA
48. İterasyon	✓		
51. İterasyon		✓	
69. İterasyon			✓

Tablo 5.12. Algoritmaların 3'lü Drone'a Göre Hedef Pozisyona Ulaştıkları İterasyon Sayıları

	PSO	YAK	KKA
68. İterasyon		✓	
82. İterasyon	✓		
99. İterasyon			✓

Tablo 5.13. 4 Adet Drone'a Göre Algoritmaların Simülasyon Ortamındaki Test Ortamına göre Aldıkları Sonuçlar

	PSO				YAK				KKA			
Dronelar	Drone 1	Drone 2	Drone 3	Drone 4	Drone 1	Drone 2	Drone 3	Drone 4	Drone 1	Drone 2	Drone 3	Drone 4
Hedefe Göre Mesafe(metre)	11,96	11,16	30,32	36,56	13,03	21,74	14,07	15,22	11,36	12,43	10,21	11,93
1 Atışta Hedefe Verdiği Hasar	0,189	0,198	0,072	0,0585	0,171	0,1035	0,1575	0,144	0,198	0,18	0,2205	0,189
Toplam Hedefe Verilen Hasar	0,5175				0,576				0,7875			
Amaç Fonksiyonu Sonuçları	54,7				73,83				69,41			

Tablo 5.14. 5 Adet Drone'a Göre Algoritmaların Simülasyon Ortamındaki Test Ortamına göre Aldıkları Sonuçlar

	PSO					YAK					KKA				
Dronelar	Drone 1	Drone 2	Drone 3	Drone 4	Drone 5	Drone 1	Drone 2	Drone 3	Drone4	Drone 5	Drone 1	Drone 2	Drone 3	Drone4	Drone 5
Hedefe Göre Mesafe(metre)	10,08	18,88	15,56	12,65	38,59	19,11	15,63	18,88	29,91	22,55	18,91	15,56	23,39	25,32	36,63
1 Atışta Hedefe Verdiği Hasar	0,223	0,119	0,144	0,177	0,0583	0,117	0,143	0,119	0,075	0,099	0,118	0,144	0,096	0,088	0,061
Toplam Hedefe Verilen Hasar	0,7213					0,553					0,507				
Amaç Fonksiyonu Sonuçları	45,81					46,41					63,47				

Tablo 5.15. 3 Adet Drone'a Göre Algoritmaların Simülasyon Ortamındaki Test Ortamına göre Aldıkları Sonuçlar

	PSO			YAK			KKA		
Dronelar	Drone 1	Drone 2	Drone 3	Drone 1	Drone 2	Drone 3	Drone 1	Drone 2	Drone 3
Hedefe Göre Mesafe(metre)	21,68	32,36	19,55	14,05	11,23	19,33	27,09	11,89	20,02
1 Atışta Hedefe Verdiği Hasar	0,103	0,069	0,115	0,16	0,2	0,116	0,083	0,189	0,249
Toplam Hedefe Verilen Hasar	0,287			0,476			0,521		
Amaç Fonksiyonu Sonuçları	69,6			61,74			93,77		

6. SONUÇLAR VE ÖNERİLER

6.1 Sonuçlar

Bu çalışmada ilk olarak Drone sürüsü uygulamalarında kullanılmak üzere bir Drone'nun matematiksel modeli elde edilmiştir. Ayrıca çalışmada 3, 4, 5 elemanlı Drone sürüsü için üç farklı sürü algoritması kullanılarak, hareketli bir hedef için davranışları incelenmiştir. Bu amaçla görev tanımın bağlı olarak birçok parametreli amaç fonksiyonu tanımlanmıştır. Bu amaç fonksiyonu temel alınarak optimizasyon algoritmaları yürütülmüştür. Sürü algoritmalarının temelinde doğadaki canlıların davranışları yatmaktadır. Bundan dolayı Drone sürüsü güdümü amacıyla optimizasyon algoritmalarının seçiminde buna dikkat edilmiş ve aşağıda verilen optimizasyon algoritmalarının amacımıza uygun olacağı düşünülmüştür;

- Parçacık Sürü Optimizasyonu Algoritması
- Yapay Arı Kolonisi Algoritması
- Karınca Kolonisi Algoritması

Çalışmada ilk olarak parçacık sürü optimizasyonu algoritması üzerinde durulmuştur. 3, 4 ve 5 adet eş Drone sürüsünün hedef olarak belirlenen aracı takip etmesi ve hedef üzerinde uygun pozisyon alarak simülasyon ortamında toplam da 4 atış karşılığı aldığı puanı kayıt altına alınmıştır.

İkinci olarak yapay arı algoritması üzerinde durulmuştur. 3, 4 ve 5 adet eş Drone sürüsünün hedef olarak belirlenen aracı takip etmesi ve hedef üzerinde uygun pozisyon alarak simülasyon ortamında toplam da 4 atış karşılığı aldığı puanı kayıt altına alınmıştır.

Üçüncü ve son algoritma karınca kolonisi algoritmasıdır. Bu algoritmada da aynı şekilde 3, 4 ve 5 adet eş Drone sürüsünün hedef olarak belirlenen aracı takip etmesi ve hedef üzerinde uygun pozisyon alarak simülasyon ortamında toplam da 4 atış karşılığı aldığı puanı kayıt altına alınmıştır.

Sezgisel algoritmalar yapısı gereği her görev tanımında ve her yeniden yol planlamasında farklı sonuçlar verebilmektedir. Algoritmalarda gidilecek konumlar rastgele seçilen parametre değerlerinden dolayı değişiklik gösterebilmektedir. Bu durum zaten doğadaki canlıların davranışlarıyla da benzerlik gösterir. Yapılan çalışmada da her üç algoritmanın her deneme için farklı yollar ürettiği görülmüştür.

Üç adet algoritma aynı çalışma koşullarında toplamda farklı iterasyon sonuçlarında hedeflenen konuma varmışlardır. Hız olarak PSO algoritması 4'lü ve 5'li Drone sürüsü testlerinde daha başarılı sonuçlar vermiştir, konumlandırma olarak da 5'li

Drone sistemlerinde KKA algoritmasının daha başarılı sonuçlar verdiği gözlenmiştir. Amaç fonksiyonu sonuçlarına bakılarak 4'lü ve 5'li Drone çalışmalarında PSO algoritmasının daha başarılı sonuçlar verdiği Tablo 5.13 ve Tablo 5.14'de gözükmektedir. Kullanılan Drone sayısının da başarı durumunu etkilediği net bir şekilde gözükmektedir.

6.2 Öneriler

Tez çalışmasında Drone'ların sürü olarak etkileşimini artırmaya yönelik çalışma gerçekleştirilmiştir. Askeri veya sivil Drone kullanımlarında, belirli görevleri başarılı bir şekilde gerçekleştirilebilmesi için problem optimizasyon haline getirilmiştir. Optimizasyon haline getirilen çalışma da PSO, YAK ve KKO algoritması kullanılarak sonuçlar elde edilmiştir. Çalışma daha farklı sürü algoritmaların kullanımında uygundur.

Bu çalışmayı yaparken dikkat edilecek en önemli konulardan bir tanesi amaç fonksiyonunun doğru seçilmesi olacaktır. Çünkü görev tanımının başarı durumu bu amaç fonksiyonuna göre değerlendirilmektedir. Amaç fonksiyonunun her görev tanımına göre güncellenmesi gerekir. Diğer bir yandan bu tür sürü sistemlerinde kullanılan Drone sayısı da başarı sonuçlarını net bir şekilde etkilemektedir. Bundan dolayı çalışmada Drone sayısı artırılıp azaltıldığında sonuçlar tekrar gözlemlenerek değerlendirmeler yapılabilir.

Algoritmaları gerçek zamanlı çalışmalar içinde uygulamak mümkündür. Gerçek zamanlı uygulamalarda Drone'ların birbiri ile çarpışma yaşanmaması için gerekli güvenlik kontrollerinin hem yazılım hem de algoritma tarafında dikkate alınması gerekir.

KAYNAKLAR

A. Jafar, S. M. A., Nisar Ahmed, 2016, Mathematical Modeling and Control Law Design for 1DOF Quadcopter Flight Dynamics, *IEEE*.

Ahmadzadeh, S. ve Ghanavati, M., 2012, Navigation of Mobile Robot Using The PSO Particle Swarm Optimization, *Journal of Academic and Applied Studies (JAAS)*, 2(1), 32-38.

Akay, B. K., Derviş, 2010, A modified Artificial Bee Colony algorithm for real-parameter optimization, *Information Sciences*, 120-142.

Ashita S., B. P. V., Puranik, 2012, Artificial Bee Colony (ABC) Algorithm for Vehicle Routing Optimization Problem, *International Journal of Soft Computing and Engineering (IJSCE)*, 2 (2).

Bhattacharjee, P. R., Pratyusha;, Goswami, I., Konar, A. ve K. Nagar, A., 2011, Multi-Robot Path-Planning Using Artificial Bee Colony Optimization Algorithm, *IEEE*.

Couceiro, M. S. R., Rui; ve Ferraira, N. M., 2011, A Novel Multi-Robot Exploration Approach based on Particle Swarm Optimization Algorithms, *Proceedings of the 2011 IEEE International Symposium on Safety,Security and Rescue Robotics*.

Çakıcı, F., 2009, Modeling, Stability Analysis and Control System Design of a Small-Sized Tiltrotor UAV, *ODTÜ, Yüksek Lisans Tezi*.

Dikmen, H. D., Hüseyin; Elbir, Ahmet; Ekşi, Ziya; Çelik, Fatih, 2014, Gezgin Satıcı Probleminin Karınca Kolonisi ve Genetik Algoritmalarla Eniyilemesi ve Karşılaştırılması, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 8-13.

Ding L., W. Hongtao, Yao Yu, 2015, Chaotic Artificial Bee Colony Algorithm for System Identification of a Small-Scale Unmanned Helicopter, *Hindawi Publishing Corporation International Journal of Aerospace Engineering*, 2015.

Dorigo, M. S. T., 2004, Ant Colony Optimization.

Gong, D.-w. Z., Jian-hua; Zhang, Yong, 2011, Multi-objective Particle Swarm Optimization for Robot Path Planning in Environment with Danger Sources, *JOURNAL OF COMPUTERS*, 6, 1554-1561.

Guan, Y. G., Mingsheng, Bai Yufan, 2019, Double-ant Colony Based UAV Path Planning Algorithm, *ICMLC '19*.

Kou, M. ve Ye, C. C., Zihao, 2011, A Bee Evolutionary Particle Swarm Optimization Algorithm for Vehicle Routing Problem, *IEEE*.

Kuzu, F., 2018, Dört Pervaneli (Quadcopter) İnsansız Hava Aracına Farklı Kontrol Yöntemlerinin Uygulanması, *Fırat Üniversitesi Yüksek Lisans Tezi*.

Lee, K.-B. ve Kim, Y.-J. H., Young-Dae, 2018, Real-Time Swarm Search Method for Real-World Quadcopter Drones, *Applied Sciences*.

Lei, L. S., Qu, 2012, Path Planning For Unmanned Air Vehicles Using An Improved Artificial Bee Colony Algorithm, *Proceedings of the31 st Chinese Control Conference*.

Li, G. C., Wusheng, 2018, Path planning for mobile robot using self-adaptive learning particle swarm optimization, *SCIENCE CHINA*, 61.

Lin, J.-H. H., Li-Ren, 2009, Chaotic Bee Swarm Optimization Algorithm for Path Planning of Mobile Robots, *Proceedings of the 10th WSEAS International Conference on Evolutionary Computing*.

Ma, F. X., Feng, 2019, Research on Path Planning of Plant Protection UAV Based on Grid Method and Improved Ant Colony Algorithm, *ACMME 2019*.

Monmarché, N., Guinand, F. ve Siarry, P., 2010, Artificial Ants From Collective Intelligence to Real-life Optimization and Beyond.

Özbek, N. S., 2010, İnsansız Hava Araçlarında Farklı Kontrol Tekniklerinin Performans Karşılaştırması, *TOBB Ekonomi ve Teknoloji Üniversitesi Yüksek Lisans Tezi*.

Pan, T.-S., Thi, K. D., Pan, J.-S. ve Nguyen, T.-T., 2017, An Unmanned Aerial Vehicle Optimal Route Planning Based on Compact Artificial Bee Colony, *Springer International Publishing*.

Raja, P. P., S., 2012, Optimal path planning of mobile robots: A review, *International Journal of Physical Sciences*, 7(9), 1314-1320.

Saffari, M. H. ve Mahjoog, M. J., 2009, Bee Colony Algorithm for Real-Time Optimal Path Planning of Mobile Robots, *IEEE*.

Shiltagh, N. A. J., Lana Dalawr, 2013, Optimal Path Planning For Intelligent Mobile Robot Navigation Using Modified Particle Swarm Optimization, *International Journal of Engineering and Advanced Technology (IJEAT)*, 2 (4).

Suvaydan, F., 2011, Mobil Robotlar İiçin Yol Planlama Problemi ve Karınca Kolonisi Algoritması ile Yol Planlama Problemlerinin Optimal Çözümü, *Düzce Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi*.

Wang, P. M., Zhihong, Cao, Zhenwei, Zheng, Jinchuan; ve Zhao, Y., 2016, Dynamics Modelling and Linear Control of Quadcopter, *Proceedings of the 2016 International Conference on Advanced Mechatronic Systems*.

Wang, Z. L., Min Dou, Lianghang ve Li Y. Z., Qingying Li Jie, 2015, A Novel Multi-objective Artificial Bee Colony Algorithm for Multi-robot Path Planning,

Proceeding of the 2015 IEEE International Conference on Information and Automation.

Xu, C. D., Haibin; Liu, Fang, 2010, Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning, *Aerospace Science and Technology*, 535-541.

Yetgin, M., 2011, Türkiye'de Çok Modlu Taşımacılıkta En Kısa Yolların Belirlenmesi, *Gazi Üniversitesi Yüksek Lisans Tezi*.

Zhang, Y., Gong, D.-W. ve Zhang, J.-H., 2012, Robot path planning in uncertain environment using multi-objective particle swarm optimization, *Neurocomputing*.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Mustafa İlker EKMEN
Uyruğu : T.C
Doğum Yeri ve Tarihi : Konya 12.09.1995
Telefon : 05545985655
Faks :
E-Posta : ilkerekmen99@gmail.com

EĞİTİM

Derece		Adı	İlçe	İl	Bitirme Yılı
Lise	:	**Muhittin Güzelkılınç Anadolu Lisesi**	**Meram**	**Konya**	**2014**
Üniversite	:	**Selçuk Üniversitesi**	**Selçuklu**	**Konya**	**2018**
Yüksek Lisans	:	**Konya Teknik Üni.**	**Selçuklu**	**Konya**	**2020**
Doktora	:				

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2018-2019	Hidrokon	Ar-Ge Mühendisi
2020-…	Ortana Elektronik	Yazılım Mühendisi

UZMANLIK ALANI

Gömülü Yazılım

YABANCI DİLLER

İngilizce