REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Abbès LAGHROUR - Khenchela
Faculté des Sciences et de la technologie
Département de Génie Industriel

Mémoire de fin d'études

Pour l'Obtention de Diplôme Master

Filière : Automatique

Spécialité : Automatique et informatique industrielle

Réalisé par :

Nabti Chouiab et Keziz Idrisse

Intitulé :

Grey Wolf Optimization (GWO) algorithm Based Unmanned Combat Aerial Vehicles (UCAVs) Path Planning

Dirigé par : Dr. ALLOUANI Fouad

Soutenue le 18/06/2023 devant les membres du Jury :

Bououden Sofiane	Prof. à l'université de Khenchela	Président
Mme. Ben Ferroudj Hafiza	MAA à l'université de Khenchela	Examinatrice
Allouani Fouad	MCA à l'université de Khenchela	Rapporteur

Année universitaire 2022/2023

Abstract:

This master thesis explores the application of the Grey Wolf Optimization (GWO) algorithm for Unmanned Combat Aerial Vehicle (UCAV) path-planning. UCAV path-planning is essential for ensuring vehicle safety and efficiency, but current methodologies often fail due to the high complexity and real-time demands of these systems. Despite several swarm intelligence algorithms having been proposed to address this issue, their effectiveness varies based on UCAV scale and complexity. The thesis aims to apply and assess the GWO algorithm in this context, hoping to overcome the challenges inherent in UCAV path-planning. The work is organized into an overview of UAVs and UCAVs, the development of a UCAV path-planning model, an analysis of metaheuristic algorithms, and finally the application of GWO to UCAV path-planning through simulation tests.

Résumé

Ce mémoire de master explore l'application de l'algorithme d'Optimisation du Loup Gris (GWO) pour la planification de trajets de Véhicules Aériens de Combat Non Habités (UCAV). La planification de trajets pour les UCAV est essentielle pour assurer la sécurité et l'efficacité du véhicule, mais les méthodologies actuelles échouent souvent en raison de la grande complexité et des exigences en temps réel de ces systèmes. Malgré plusieurs algorithmes d'intelligence en essaim proposés pour résoudre ce problème, leur efficacité varie en fonction de l'échelle et de la complexité des UCAV. Ce mémoire vise à appliquer et évaluer l'algorithme GWO dans ce contexte, dans l'espoir de surmonter les défis inhérents à la planification de trajets des UCAV. Le travail est organisé en une vue d'ensemble des UAV et UCAV, le développement d'un modèle de planification de trajets pour les UCAV, une analyse des algorithmes métaheuristiques, et enfin l'application du GWO à la planification de trajets des UCAV à travers des tests de simulation.

Contents

Contents

General Introduction

General Introduction

Path-planning, a critical component in the Uninhabited Combat Air Vehicle (UCAVs) or Unmanned Combat Aerial Vehicles (UCAVs) system, aims to ascertain a path with the minimum distance while ensuring the safety of the vehicle. A safe route ensures that the aircraft can reach the destination without any collisions or being detected by radar. The development of UCAVs has garnered significant attention due to its ability to accomplish challenging tasks in complex environments. Past studies have introduced a variety of models to simulate the intelligent behavior needed for an aircraft to autonomously determine a suitable path [1,2].

Under this model, a plethora of methods have been proposed to produce an optimal path, considering factors such as angle, velocity, and height [3]. For instance, You et al., in [4], introduced a three-dimensional (3D) path-planning approach based on situational space to meet UCAVs' tactical requirements for target tracking and collision avoidance. In [5], a two-stage method to address the terrain-following (TF)/terrain-avoidance (TA) path-planning problem for UCAVs is outlined. However, these methods often falter due to the high complexity and the need for UCAVs to respond in real-time. Swarm intelligence algorithms have been widely proposed to overcome this complexity. For example, Pehlivanoglu et al. [7] introduced a multi-frequency vibrational genetic algorithm for UCAV path-planning, and Zhang et al. employed the grey wolf optimizer to handle UCAV path-planning [8]. Notwithstanding, each swarm intelligence algorithm has its strengths and weaknesses, and different swarm intelligence algorithms yield varied performance for cases with different UCAV scales. It is improbable that a single swarm intelligence algorithm can address all complex problems without a detailed analysis from multiple perspectives [19].

Therefore, in this master thesis, we showcase the application of the Grey Wolf Optimization (GWO) algorithm, a swarm intelligence algorithm, to the UCAV path-planning problem.

This master thesis is organized as follows:

- Chapter I contains two main sections, the first offering a comprehensive overview of Unmanned Aerial Vehicles (UAVs), their classifications, and applications, and the second providing a brief description of UCAVs, their main types, and components.
- Chapter II details the steps undertaken to develop a mathematical model for UCAV path planning, highlighting the most prevalent method.
- Chapter III discusses the basic aspects and types related to metaheuristic algorithms.
- Chapter IV showcases the application of the Grey Wolf Optimization (GWO) algorithm through simulation tests for the UCAV path-planning problem.

Finally, the thesis concludes with a broad summary and discussion of the topics covered.

Chapter I:

Unmanned Aerial Vehicles UAVs; An Overview

I.1. Introduction

The present chapter is divided into two main parts; the first talks about Unmanned Aerial Vehicles (UAVs) in terms of; A general overview of these latter, its classification and its applications, quadrotors as a specific kind of UAVs and finally its advantage, disadvantages and its typical uses. The second part contains a brief description of Unmanned Combat Aerial Vehicles (UCAVs), in terms of its main types and its main components.

I.2. Unmanned Aerial Vehicles (general case)

I.2.1. Overview of Unmanned Aerial Vehicles (UAVs)

An Unmanned Aerial Vehicle (UAV) or drone is an aircraft without passenger or pilot on-board which can fly autonomously based on pre-programmed flight plans, through the help of complex dynamic automation systems, or be controlled remotely by a pilot at a ground control station. Depending on its mission, endurance, and size, each UAV can carry different payloads. UAVs are equipped with multiple sensors (such as Global Positioning System (GPS), Inertial Measurement Unit (IMU), Differential Global Positioning System (DGPS), accelerometer, magnetometer, pito tube, etc. and are connected in real time to many other systems at the ground control station.

This allows the UAV to gain important information such as speed, position, heading, altitude, movement direction, temperature, speed and wind direction, the amount of fuel or remaining energy to perform the task, etc. UAV can perform a variety of tasks in military, civilian, and entertainment applications. Moreover, since they are free of screw on board, they can be designed to be smaller, can perform tasks that require greater flexibility, and can carry more payloads.

I.2.2. UAVs classification

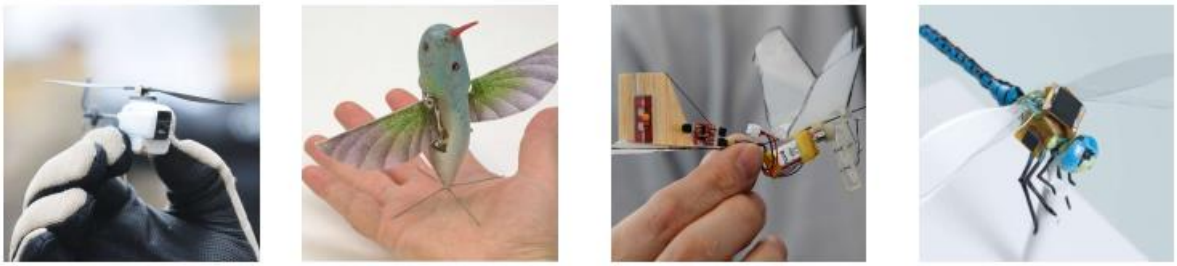
There are many characteristics for classifying UAVs [21], [22] based on their variety intended use.

I.2.2.1. Classification according to size

A. Very small UAVs: are the unmanned aircraft with dimensions ranging from the size of a large insect to 30 [cm] - 50 [cm] long. The very small UAVs can be sub-divided into

- Micro or nano UAVs: insect size is up to 15 [cm],
- Mini UAVs: insect size is between 15 [cm] and 50 [cm].

The representative of very small UAVs is insect-like UAVs. Their popular micro design always has flapping or rotary wings. They are extremely small in size, very lightweight, and can be used for spying and biological warfare. Some of nano UAVs are shown in Fig. I.1 Examples of mini-UAVs are Xbird250 Quadro, Parrot bebop, Parrot Disco, and Eagle VTOL Mapping in Fig. I.2.



(a) Black Hornet (b) Hummingbird (c) Hummingbird (d) Dragonfly robot

Figure I.1. Nano UAV robots.

(a) Xbird 250 quadro (b) Parrot bebop (c) Parrot Disco (d) Eagle VTOL Mapping

Figure I.2. Mini UAV robots.

B. Small UAVs: are the UAVs class that have at least one dimension greater than 50 cm and no larger than 2 meters. Most of them are hand-launched by throwing in the air while some other are of fixe d-wing model. Examples of small UAVs are RQ 11 Raven with a wingspan of 1.4 m, RQ 7 Shadow, RS 16 is a crossover UAV between a small and a medium sized system, and Turkish Bayraktar which weight is 5 kg and has a range of 20 km in Fig. I.3.

(a) RQ 11 Raven (b) RQ 7 Shadow (c) RS 16 (d) Turkish Bayraktar

Figure I.3. Small UAV robots.

C. Medium UAVs: are the class of UAVs that are too heavy to be carried by one person but are still smaller than a light aircraft. They usually have a wingspan of about 5-10 m and can carry payloads of 100 to 200 kg. Some medium fixed-wing UAVs are shown in Fig. I.4. There are also numbers of medium sized UAVs in the form of rotary-based.

(a) UK Watchkeeper (b) RQ 2 Pioneer (c) RS 20 (d) Eagle Eye

Figure I.4. Medium UAV robots.

D. Large UAVs: applies to the large UAVs and used mainly by the military. Some of large UAVs such as Atomics MQ-1 Predator, Global Hawk, Harfang, and MQ lC Warrior are shown in Fig. I.5.

(a) Atomics MQ-1 Predator (b) Global Hawk (c) Harfang (d) MQ 1C Warrior

Figure I.5. Large UAV robots.

I.2.2.2. Classification of UAVs according to its range of action

A. High-altitude long-endurance (HALE): these types of UAVs can flight over 15000 [m] altitude and more than 24 [hr] of endurance. These UAVs are usually used for surveillance mission. Two examples of HALE UAVs are the Northrop Grummans Global Hawk and the Agilis HALE UAV illustrated in Fig. I.6.

(a) Northrop Grummans Global Hawk (b) Agilis HALE UAV

Figure I.6. High-attitude long-endurance (HALE) UAV robots.

B. Medium-altitude long-endurance (MALE): the altitude for these types of UAVs is about 5000 [m] to 15000 [m] with 24 [hr] maximum of endurance. These UAVs are also used for surveillance mission but with shorter ranges. Two examples of MALE UAVs are Persistent UAS Platforms and Agilis HALE UAV illustrated in Fig. I.7.

(a) Persistent UAS Platforms (b) WK-450 Watchkeeper

Figure I.7. Medium-altitude long-endurance (MALE) UAV robots.

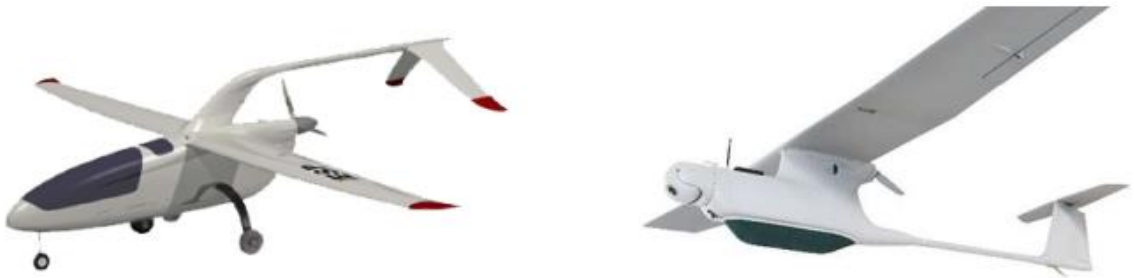
C. Medium-Range or tactile UAV (TUAV): the range of flight for this type of UAVs is between 100 [km] and 300 [km]. The TUAVs IAI Heron (Machatz-1) and AAI (Textron) RQ-7 Shadow are shown in Fig. I.8 as examples of this kind of UAVs.

(a) IAI Heron (Machatz-1)

(b) AAI (Textron) RQ-7 Shadow

Figure I.8. Medium-Range or tactile UAV (TUAV) robots.

D. Close-range UAV: The close-range UAVs can operate within the range less than 100 [km]. This type of UAVs is mainly used for traffic monitoring, powerline inspection, surveillance, or crop-spraying in precision agriculture. The close-range UAV Optimus UAV and RemoEye 006 are shown in Fig. I.9.



(a) Optimus UAV

(b) RemoEye 006

Figure I.9. Close-range UAV robots.

D. Mini-UAV (MUAV): The operation range for mini-UAVs is up to 30 [km] and their weight is less than 20 [kg]. The mini-UAVs Orbiter Mini UAV and Skylark 1 LE Mini are shown in Fig. I.10.

(a) Orbiter Mini UAV

(b) Skylark 1 LE Mini

Figure I.10. Mini-UAV robots.

E. Micro UAV (MAV): the wingspan is less than 150 [mm]. This type of UAVs are mainly used in urban environnements or for indoor applications. Due to the small size, the micro-UAVs are very vulnerable to the wind (e.g., see Fig. I.11).

(a) Micro UAV NX70

(b) Honeywell RQ-16 T-Hawk

Figure I.11. Micro-UAV robots.

F. Nano Air Vehicle (NAV): the size is of 10 [mm]. The Nano UAVs, the Hornet 2-b (Prox Dynamics) and Nano UAV Hornet 3 Prox Dynamics are illustrated in Fig. I.12, both of its are complete with a camera and a video transmitter.

(a) The Hornet 2-b (Prox Dynamics)

(b) Nano UAV Hornet 3 Prox Dynamics

Figure I.12. Nano UAV robots.

G. Remotely Piloted Helicopter (RPH): are UAVs that are capable of vertical take-off and landing automatically with pre-programmed programs. This type of aircraft is often used in missions that require hovering flight such as inspection or surveillance. Remotely piloted helicopters Long Endurance Fuel Power Fixed Wing gasoline VTOL UAV and Terrafugia TF-2 Tiltrotor are shown in Fig. I.13.

(a) Long Endurance Fuel Power Fixed Wing gasoline VTOL UAV

(b) Terrafugia TF-2 Tiltrotor

Figure I.13. Remotely Piloted Helicopter (RPH).

I.2.3. Applications of UAVs

Nowadays, the use of UAVs with diverse capabilities for both civilian and military applications are growing very fast. Moreover, there is a significant interest in the development of novel drones that can operate autonomously in different types of complicated environments and locations. UAVs can perform various missions in complicated environments both outdoor and indoor. Drones can be equipped with various sensors and cameras. Consequently, UAVs are capable of performing accurate and reliable tasks of intelligence, surveillance, and reconnaissance missions. The missions of UAVs can be categorized as for the military or civil, type of the flight zones (outdoor and indoor), and type of the environments (air/space) as shown in the Fig. I.14.

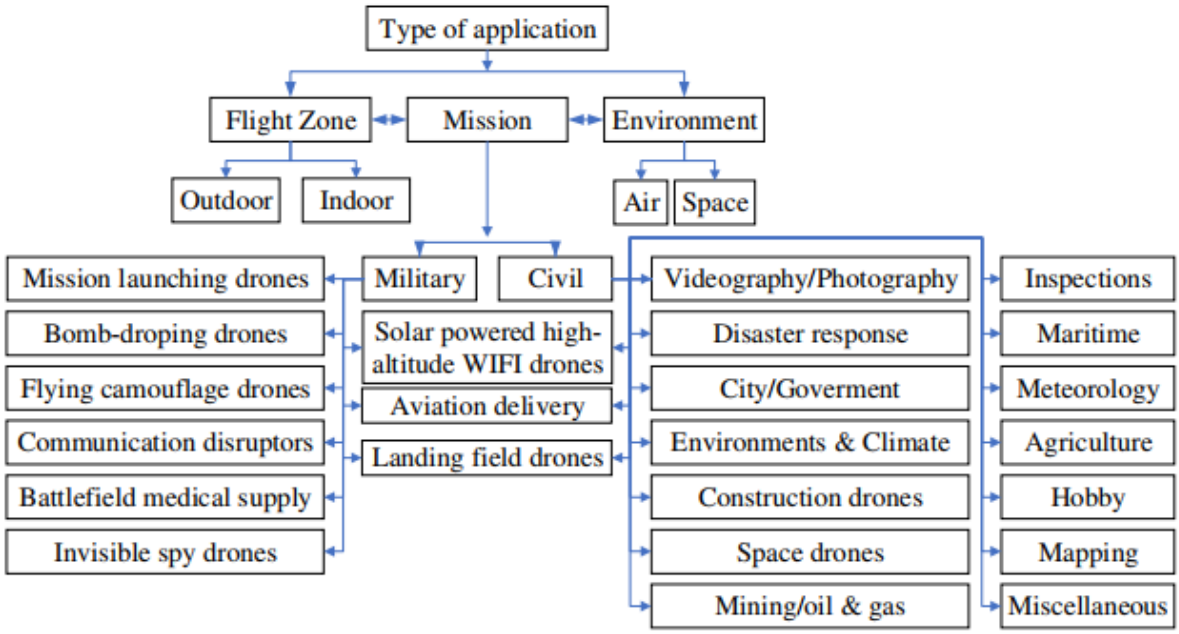


Figure I.14. Classifications of drones in termers of application

A. Search and rescue operations

A really important application of UAV is to search and rescue missions. A search and rescue UAV is usually used by emergency services, such as police officers, firefighters, or rescue teams. Every second is vital in search and rescue operations. In order to most effectively deploy rescue missions, we need to understand an get a quick overview of the situation. While manned aerial vehicles and helicopters require time to be ready for operating the mission, drones can be operated instantaneously without any preparation time. Unmanned Arial vehicles UAVs can provide real-time visual information and data in the aftermath of an earthquake or hurricane. They can also become an eye in the sky to locate a lost person in the mountain for example. Thanks to high mobility, quick deployment, able to operate in complex environments including small environments such as caves, this type of UAVs is ideal for searching over vast areas for missing persons in need of rescue and in environment.

B. Environmental protection

Besides rescue or military applications, UAV is also increasingly applied in environmental management such as managing agricultural lands, real-time weather condition and forecast, observing the effects of climate change, monitoring the biodiversity of different ecosystems from rain forests to the oceans, monitoring crops pest, and tracking wildlife in different areas [23].

C. Mailing and delivery

Unmanned delivery service is attracting the attention of big companies in the world such as Amazon, Google [24],[25]. More and more companies are using drones to deliver products to their customers. To accomplish this, the UAV is equipped with the ability to vertically take off and land.

D. Drone in Precision Agriculture (PA)

With increasing populations, the demand for food is increasing. The latest advanced technologies should be applied in agriculture to meet this growing demand. Several types of robots have been designed for using in PA such as mobile robots for harvesting, crops monitoring. With the introduction of low-cost drones with advanced capabilities, the use of UAVs in precision agriculture is growing really fast. Drones are playing a significant role in optimizing agriculture operations. Thanks to the help of drones, agricultural activities such as crop management, crop counting, crop health monitoring, and spraying pesticides can be operated more cost-effectively and more quickly as compared to conventional methods, which typically rely on the use of piloted aircraft.

I.2.4. Advantage, disadvantages, and typical uses of UAVs

In the previous subsection, we have examined the classification and applications of UAVs. In this sub-section, we examine the advantages and disadvantages of UAVs for the fixed-wing UAV, single rotor UAV, and multirotor UAV.

A. Fixed Wing UAV

Fixed-wing UAVs use a wing like a normal aeroplane to provide the lift by the aircraft's forward motion. Fixed-wing UAVs can be self-propelled, pure gliders (vehicles whose free flight does not rely on a method of propulsion) or a mixture of the two. Fixed-wing UAVs can be distinguished to the other types of UAVs that they cannot stay hovering in one place with vertical lift rotors in the air. Fixed wing UAVs are well known in the military, as they are often used when manned flight is considered too risky or difficult. They are also used in the commercial industry, monitoring, etc.

Advantages:

The operating time of fixed-wing UAVs can be a couple of hours and can up to 16 hours or more.

- Fixed-wing UAVs can fly at a high altitude.
- Fixed-wing UAVs have the ability to carry more weight.

Disadvantages:

- Fixed-wing UAVs are usually expensive.
- Training is usually required for flying.
- Launcher or long runway are needed to get a fixed-wing UAVs into the air.
- It is required the runway, parachute, or special method for landing (recovering) the UAVs.
- Fixed -wing UAVs can't hover in the air.

Typical uses:

The fixed-wing UAVs are usually used for commercial purposes such as aerial mapping, inspection, agriculture, construction, security, and surveillance due to the ability of long endurance, high altitude, and long operating fly time.

B. Single Rotor UAV

While a multi-rotor UAV has many different rotors to hold it up, a single rotor UAV has just one rotor, plus a tail rotor to control its heading.

Advantages:

- A single rotor UAVs have the benefit of much greater efficiency over a multi-rotor,
- Long flying time if they are powered by a gas mot or.
- Single rotor UAVs are able to hover vertically in the air.
- Single Rotor UAV are strong and durable.
- Heavy payload capability.

Disadvantages:

- Single Rotor UAVs are harder to fly than multi-rotor UAV types.
- Single rotor UAVs can be expensive.
- Single rotor UAVs have a higher complexity.
- Single rotor UAVs can be dangerous because of their heavy spinning blade.

Typical uses:

Surveying, research, Aerial LIDAR laser scan.

C. Multi-rotors UAVs

A multirotor UAV is one of the robots that has the ability to achieve agile motion in the air. Unlike fixed-wing UAV, multirotor UAVs rely on the rotation of rotors to generate power, which will generate powerful air flow. This is the popular choice for aerial photography, film making and surveillance. It can also be used by professionals and hobbyists alike because of its small size and ready to fly out of the box capabilities. The multirotor UAVs classification is shown in Fig. I.15. Multirotor UAVs carry several rotors on their body and can be further classified based on the number of them on the platform of the drone. There are tri-copters (3 rotors as in Fig. I.15a, I.15b), quadcopters (4 rotors as in Fig. I.15c, I.15d), hexa-copters (6 rotors as in Fig. I.15e, I.15f, I.15g, I.15h, I.15i) and octocopters (8 rotors as in Figures I.15j, I.15k, I.151, I.15m, I.15n, I.15O). The most commonly used of Multirotor UAVs are quadcopters.

Advantages:

- Simpler rotor mechanics required for flight control, easy for maintenance.
- Multirotor UAVs are easy control and maneuver.

- Multirotor UAVs have the ability to hover in the air.
- Multirotor UAVs can take off and land vertically.
- Multirotor UAVs are very stable.

Disadvantages:

- Low flying time (usually 15-30 minutes).
- Multirotor UAVs are limited on payload capability.
- Multirotor UAVs's energy is spent on fighting gravity and stabilizing in the air.

Typical uses:

Aerial photography and video aerial inspection, leisure, agriculture, construction, security.

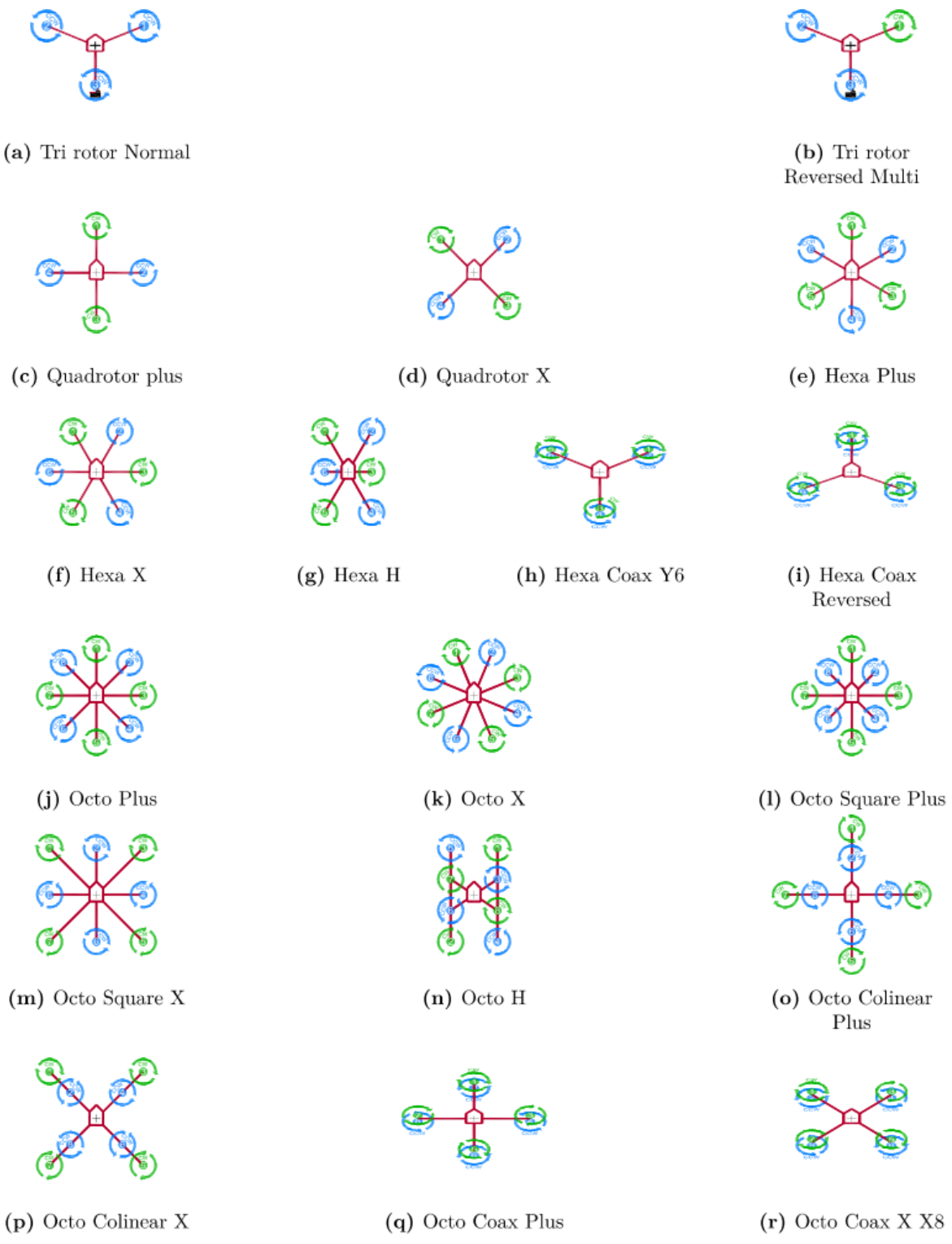
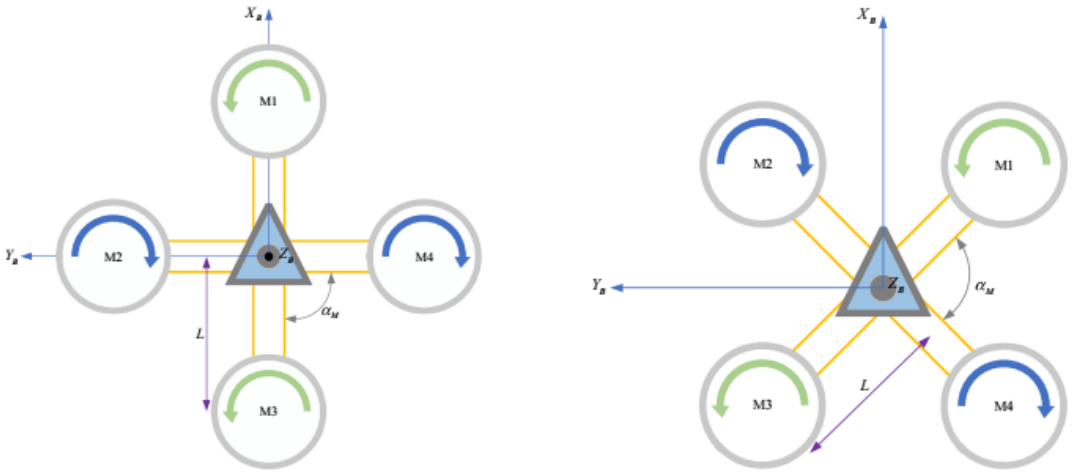


Figure I.15. Multi-rotors classification according to the principle of flight.

I.3. Quadrotors

A quadcopter is a helicopter that consists of a rigid cross-frame equipped with four rotors M_1, M_2, M_3, and M_4 as shown in Fig. I.16. There are two main configurations for quadrotor: "+" configuration as in Fig. I.15a, and "X" configuration as in Fig. I.15b. The motors are equidistant from the centre of the quad by about L. Each rotor M_i (for $i = 1,..,4$) generates thrust f_i (for $i = 1,..,4$). The direction of rotation of the rotors are fixed and f_i is always a positive quantity. The main thrust is the sum of all the thrusts which are generated by the rotors.



(a) Quadrotor + configurations (b) Quadrotor X configurations

Figure I.16. A quadrotor main configuration.

In order to avoid the yaw drift due to the reactive torques, the quadrotor aircraft is configured such that the set of rotors M_2, M_4 (left-right) revolves in clockwise (CW) direction, while the pair of rotors M_1, M_3 (front-rear) rotates in counter-clockwise (CCW) direction.

Advantages:

- Rotor mechanics are simplified as it depends on four fixed pitch rotors unlike the variable pitch rotor in the helicopter.
- Due to the symmetry in the configuration, the gyroscopic effects are reduced leading to simpler control.
- Stationary hovering can be more stable in quadrotors than in helicopters due to the presence of four propellers providing four thrust forces shifted a fixed distance from the center of gravity instead of only one propeller centred in the middle as in the helicopters structure.
- More advantages are the vertical take-off and landing capabilities, better manoeuvrability and smaller size due to the absence of a tail, these capabilities make quadrotors useful in small area

monitoring and buildings exploration.

Disadvantages:

- Quadrotors consume a lot of energy due to the presence of four separate propellers.
- Quadrotors have a large size and heavier than some of their counterparts again to the fact that there are four separate propellers.

I.4. Unmanned Combat Aerial Vehicles (UCAVs)

A UCAV is an aircraft that expands tactical mission options for revolutionary new airpower as an integrated part of a system of systems solution. A UCAV can conduct a mission from ordinary airfields as part of an integrated force package complementary to manned tactical and support assets. The controllers of the UCAV can observe rules of engagement and make critical decisions to use or refrain from using force. The role of the UCAV is "the first day of the war" force enabler which complements a strike package. In this role, UCAVs accomplish pre-emptive destruction of sophisticated enemy integrated air defences (IADs) in advance of the strike package and enable the attacking forces by providing reactive suppression against the remaining IADs. UCAVs can provide continuous vigilance with an immediate lethal strike capability to prosecute high value and time-critical targets. A UCAV system can provide "no-win" tactical deterrence against which an enemy's defences would be ineffective, thereby ensuring air superiority and kill high-value targets with point precession.

I.4.1. Comparison of UCAVs

A UCAV may seem like a futuristic fighter aircraft but in reality, it is a lot different. The first and most obvious difference is that it is unmanned. In today's world where we value human life so much and cannot bear losses in any form, this piece of technology can bring revolution. Since we need not carry the load of a human, we may use the difference of weight, to carry the equipment or improve on the engine which can prove as a gamechanger in a situation. It can be launched for any action in a very short amount of time. It can do all the activities that any normal fighter aircraft may have performed, but by giving the luxury to the pilot of sitting in a room and controlling the aircraft to do the task assigned. If we compare the size of the UCAV to any other fighter plane, it varies but the models being used in the present are smaller and more compact. They have a very high success rate which tends to grow exponentially if used on a much larger scale. UAV/UCAV drones are used where there are small areas, areas in remote locations where it is hard to fly. Higher-resolution data can be obtained as compared to traditional aircraft surveying. These drones also provide great efficiency and cost savings for the organizations it is being utilized by while confronting traditional workflows and expectations. UCAVs demand only very small stretches to take off and land, no longer than a few square feet in the case of vertical take-off and landing (VTOL) designs.

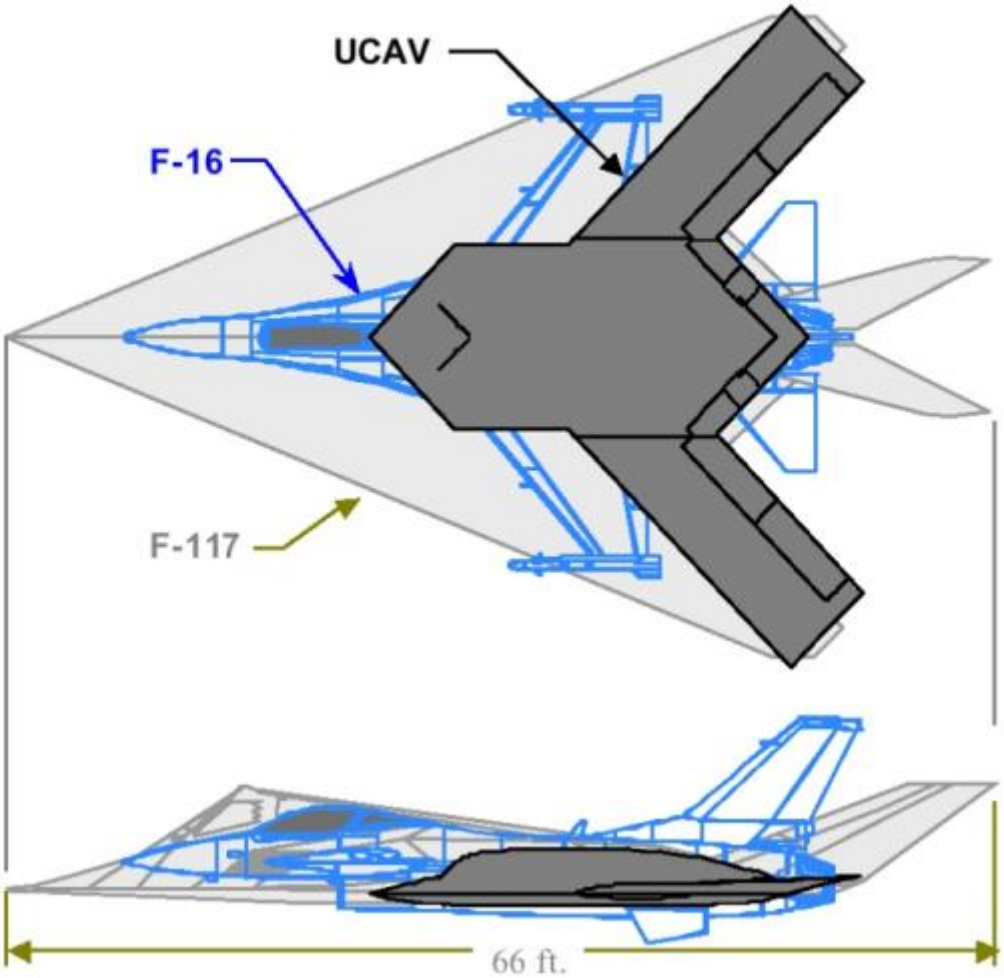


Figure I.17. Comparison of the size of a UCAV with other aircrafts; F-16 -General Dynamics F-16 Fighting Falcon, F-117- Lockheed Martin F-117 Nighthawk and UCAV- Lockheed Martin RQ-170 Sentinel.

I.4.2. Components of a UCAV

The components (see the Fig. I.18) of a UCAV are the same as that of a UAV but with the capability to carry ordinance on board. UCAVs have several main components such as:

Figure I.18. Various devices and sensors mounted on a UAV.

I.4.2.1. The main body

The body of a UCAVs comprises a fuselage, plane wings, tail rotor and canopy, multi-rotor frame and arms.

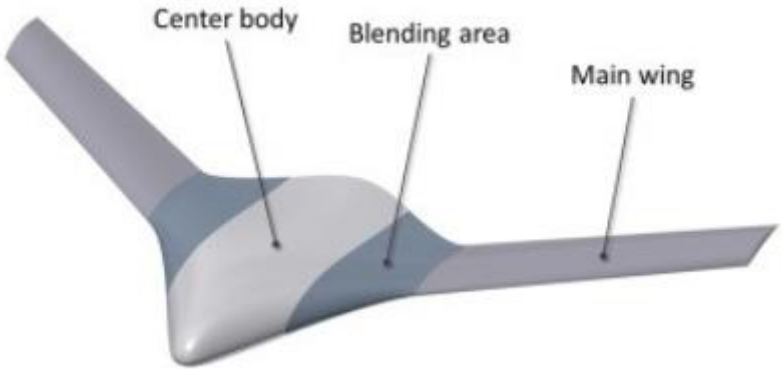


Figure I.19. The main body for a UAV.

I.4.2.2. Power supply with the corresponding platform

Smaller UCAVs usually fly on batteries while the larger ones use fuel or even solar power (Not being used very widely as of now). Most of the UCAVs used for traditional purpose use engines such as or comparable to Pratt & Whitney F100-PW-220U (see Fig. I.20) engine used by X-47B Pegasus.

Figure I.20. Pratt & Whitney F100-PW-220U.

I.4.2.3. Computer operations

UCAVs hardware systems are constantly being specialized, operation numbers increased and accelerated to support the operating system with no failures.

I.4.2.4. Sensors

It mainly has three types of sensors (see Fig. I.21):

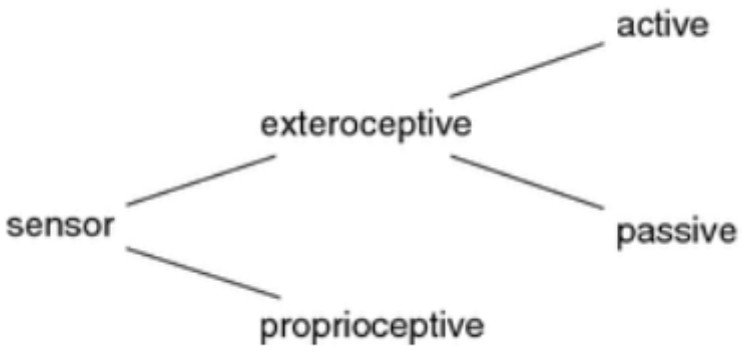


Figure I.21. Classification of sensors.

- Proprioceptive

Measurements of movement relative to the robot's internal frame of reference (also called dead reckoning).

- Exteroceptive

Measurements of the layout of the environment and objects relative to a robot's frame of reference.

- Ex-proprioceptive

Measurement of the position of the robot body or parts relative to the layout of the environment.

I.4.2.5. Actuators

Installed actuators are determined by the type of UCAVs (several electronic controllers, engines, propellers and others) (see Fig. I.22).

Figure I.22. Northwest UAV Pegasus Actuator.

I.4.2.6. Software

The uninterrupted flight is enabled with installed software safely leading the UCAVs on its way and giving the information where to go and when to react.

I.4.2.7. Loop principles

It has open loops for elementary types and closed loops for larger and more sophisticated systems.

I.4.2.8. Flight controls

It has similar flight controls like aerial vehicles flown by a pilot. However, automatic flight control is much more demanding.

I.4.2.9. Telecommunications

The connection is quite regular (antenna and analog-digital converter) and enables the transmission of data needed for the flight to proceed. As the radio signal can be transmitted from ground control, a remote system as well as another manned aerial vehicle.

I.4.2.10. UCAV software

UCAV software is designed to tell the UCAVs where to go and what to do while flying from A to B. To understand and connect all necessary information of the UCAVs, the software part becomes a very complex system. The software installed in the UCAVs operates in a layer like a system. Furthermore, the layers are divided into tiers that perform in various time slots. The layers have to be combined properly to control the flight patterns, altitude and other important information for the UCAVs to work and act

accurately. These combinations of layers are called the flight stack or autopilot. The information received has to be analyzed during the flight. To achieve a unified component's communication, a generic architecture must be designed and promoted.

I.4.2.11. Flight control system

Current flight control is basically in this behavior layer. Some advanced control arithmetic is also realized in this layer. In this paper, as far as system nonlinear strong coupling characters, corresponding control rules are designed by the use of a dynamic inversion technique. It resolved the control problem when the speed changes as the result of flight at an angle of attack, and sideslip angle, roll angle. The flight control system structure of nonlinear dynamic inversion is shown in Fig. I.23.

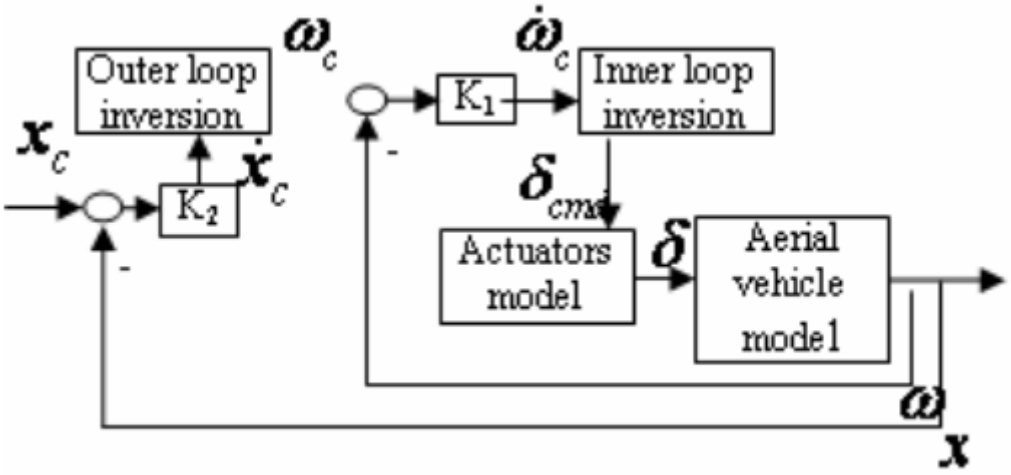


Figure I.23. Flight control system structure.

I.4.2.12. Software layers

A Firmware: It operates from machine code to processor and afterward to memory access.

A Middleware: It conducts flight control, navigation, and telecommunication.

An Operating System

It monitors optic flow, avoids interference while Simultaneous Localization & Mapping (SLAM) searches for the solution and decides what the action according to received information would be. Requirements of firmware and middleware are time-critical. The requirement of the operating system is computer intensive.

I.4.2.13. Autonomous control structure

Based on the artificial brain theory; an autonomous control system may be classified into three layers namely: -

- Perception fusion layer

In this layer, the information is transmitted from a multi-sensors system to a relative system and was fused ultimately.

- Cognition and decision layer

In this layer, a cognitive model was designed based on three-phase memory mechanisms that had ultra-short-time memory and long-time memory.

- Behavior control layer

In this layer, a planning function was added that increased behavior intelligence. The flight control was simulated using the dynamic inversion method. And the result showed the design of the execution sub-system was reasonable through tracking the mobile trajectory.

An autonomous control structure based on cognitive science is designed, as showed in Fig. I.24 and it is divided into three layers which are perception fusion layer, cognition and decision layer, and behavior control layer. UCAV could perceive online situations, execute assigned missions autonomously and make a decision during aviation in principle.

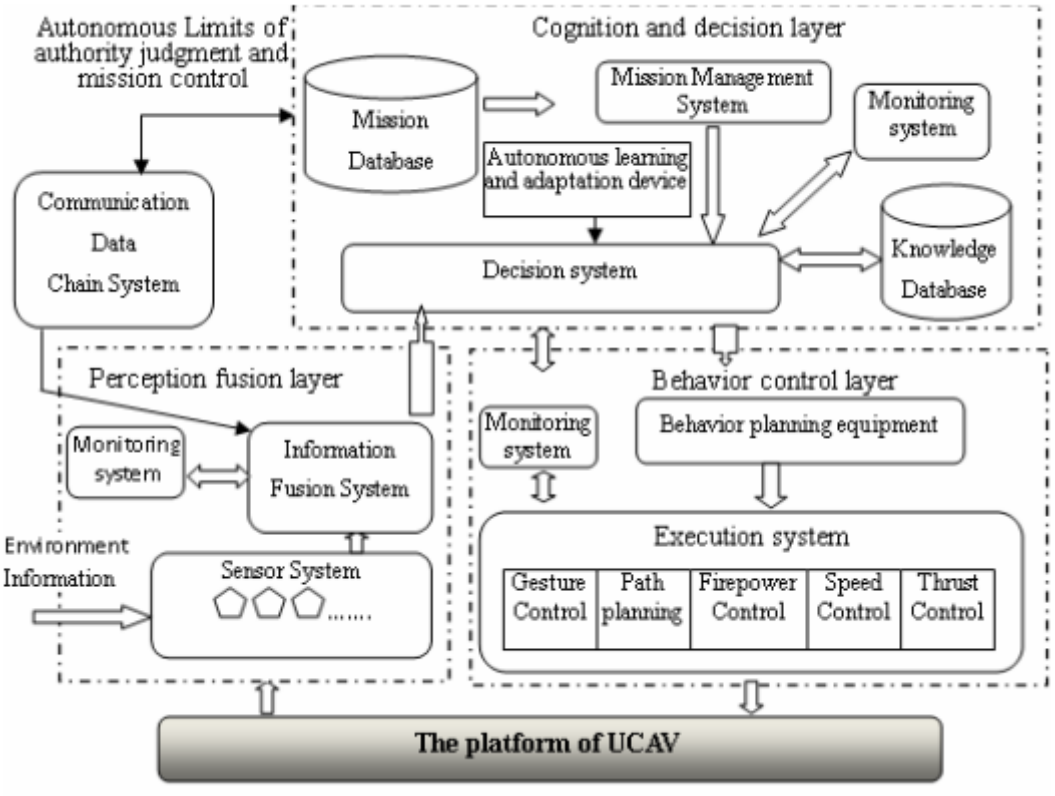


Figure I.24. UCAV autonomous control structure.

I.4.2.14. Weapons on Board

There are several types of weapons that are carried by different UCAVs: - Here examples from the General Atomics MQ-9 Reaper Drone have been taken, which has proved its operational capability in many of the operations.

A. AGM-114 Hellfire

The AGM-114 Hellfire (Fig. I.25) is an air-to-surface missile (ASM) first developed for anti-armor use, but later models were developed for precision drone strikes against other target types, and have been used in a number of targeted killings of high-profile individuals.

Figure I.25. AGM-114 Hellfire Missile.

B. GBU-12 Paveway II

The GBU-12 Paveway II (Fig. I.26) is an American aerial laser-guided bomb, based on the Mk 82 500-pound general-purpose bomb, but with the addition of a nose-mounted laser seeker and fins for guidance. A member of the Paveway series of weapons, Paveway II entered into service c. 1976.

Figure I.26. GBU-12 Paveway Missile.

C. MBDA Brimstone

MBDA Brimstone (Fig. I.27) is an air-launched ground attack missile developed by MBDA.

Figure I.27. A Brimstone missile.

I.4.3. Parameter for classification of a UCAV

I.4.3.1. Types

A UCAV can be classified into two types:

A. MALE or Medium Altitude Long Endurance UCAV: It can fly up to an altitude of 30,000 feet and travel over 200 km.

B. HALE or High-Altitude Long Endurance UCAV: It can go beyond 30,000 feet and has a range of a few thousand.

I.4.3.1. Speed

- Mach 0.18 (222 kmph)-Mach 1 (1,234 kmph)
- Ratio of a drone's speed to speed of sound.
- This figure is never specific and changes from model to model.

I.4.3.1. Cost of manufacturing*:* Here is the approximate cost of one of the most widely used drone, the General Atomics MQ-1 Predator UCAV.

Program cost: US $2.38 billion (2011) Unit cost: US $4.03 million (2010)

- This figure is approximate and varies as per the aircraft model.

I.4.4. Uses of UCAV's

I.4.4.1. Fighting Terrorism

UCAVs have played a vital role in fighting terrorism in terror-stricken regions. A MQ-1 Predator UAVs armed with Hellfire missiles (Fig. I.28) have been used by the U.S. as platforms for hitting ground targets. Between the years 2006 and 2009, UCAV-launched missiles allegedly had killed between 750 and 1,000

people in Pakistan occupied regions, according to the report. Out of these, about 20 people were said to be leaders of various terror groups. As per the report Overall, 66% to 68% of the people killed were militants, and 31% to 33% were civilians. U.S. officials disputed the percentage for civilians.

Figure I.28. A MQ-1 Predator in Afghanistan equipped with AGM-114 Hellfire missile.

I.4.4.2. Killing a High-Value Target in Foreign Territory

UCAVs can and are being used to take down high-value targets with pinpoint precision in foreign territory. They make less noise which makes them less vulnerable to being spotted and leave the region quickly. A top Irani General, Qasem Soleimani was struck on 3rd January 2020 by U.S. UCAV strike. An MQ-9 Reaper UCAV (Fig.14) fired at least two Hellfire missiles at vehicles carrying Iranian Major General Qassem Soleimani, Iraqi militia leader Abu Mahdi al-Muhandis and their entourage upon arrival at Baghdad International Airport.

I.4.4.3. Increasing the operational range of the Navy

The navy may use it to increase its range and launch it from an aircraft carrier (if a carrier-based version is used). This would also decrease the response time since it needs no pilot to dress up to take charge. It may be operated from the same operations room from where the rest of the aircraft carrier is being operated and engagements are been ordered. Even if the drone is lost, no rapid rescue is needed since it is unmanned.

I.4.4.4. Air Superiority and Air Supremacy

The few moments at the start of any conflict would decide which side gains the advantage throughout. In this matter, the UCAV would prove crucial since it may be used in long-range strikes to destroy the air defense capability so as the paratroopers could be dropped, or in close quarters for air-to-air conflicts so as to respond immediately, preventing the enemy from penetrating deep inside the territory before the principal fighters and bombers could arrive to suppress the enemy.

I.4.4.5. Electronic Warfare

Any compact electronic warfare equipment may be outfitted along with other ordinances so as to be used to wreak havoc on enemy intelligence by using electronic warfare, in which we deny the opponent the advantage of and ensure friendly unimpeded access to, the EM spectrum. Due to this, the enemy may not be able to use its radar and other critical equipment, to coordinate assaults or other operations planned prior. This would bring down the command structure and would make the gathering of intelligence difficult for the enemy and provide an advantage to the party making use of this technology.

I.4.4.6. Healthcare in Hostile Regions

Since the early days, medicine has had a profound impact on preventing disease, increasing life expectancy, and raising general standards of living. However, many hostile regions lack access to even basic healthcare as they do not have a stable healthcare service due to ever-mounting tensions inside the country most of the time. While medical supplies can be delivered by traditional means, certain circumstances call for quick and safe access to drugs, blood, and medical technology for the war-stricken public and soldiers—a need UCAVs could fulfill.

I.5. Conclusion

In this chapter we have talked firstly about, Unmanned Aerial Vehicles (UAVs) in terms of; A general overview of these latter, its classification and its applications, quadrotors as a specific kind of UAVs and finally its advantage, disadvantages and its typical uses. A brief description of Unmanned Combat Aerial Vehicles (UCAVs), in terms of its main types and its main components was also presented.

In the following chapter, we will present in detail, how a mathematical model for UCAV path planning is obtained.

Chapter II:

Mathematical model for UCAV path planning

II.1. Introduction

Path planning for UCAV is a new low altitude penetration technology to achieve the purpose of terrain following and terrain avoidance and flight with evading threat, which is a key component of mission planning system. The goal for path planning is to calculate the optimal or suboptimal flight route for UCAV within the appropriate time, which enables the UCAV to break through the enemy threat environments, and self-survive with the perfect completion of mission. In this chapter, we present in detail referring to the work done in [26-31], the different steps followed to develop a mathematical model relative to an UCAV path planning.

II.2. UAVs Path planning; A general presentation

Path planning has been and continues to be an extensively researched topic. The problem has been addressed in situations ranging from controlling the motion of computer-generated entities through virtual worlds in video games to the autonomous motion of a rover through a physical building with no prior knowledge of the environment. Much research has focused on path planning for UAVs as it is a relatively complex problem. A representative sample of these path planners will be discussed. Many UAV path planners simplify the problem by limiting the vehicle to maneuver only in a horizontal plane and treating altitude as a constant.

Formulating the problem with only four degrees of freedom facilitates the application of ground vehicle path planning algorithms directly to UAVs. Two-dimensional path planners also lend themselves well to visualization on 2D computer screens. Reducing the problem from six degrees of freedom to four also exponentially reduces the computational complexity of the problem. Simpler computations are attractive because they are easier to program, and take less time to execute. In the case of UAVs, the benefits of simplicity do not justify the cost of neglecting the 3D maneuvering capabilities of these vehicles. Recently, many researchers have realized the limitations of two-dimensional UAV path planning and begun implementing three-dimensional path planners, taking advantage of the additional two degrees of freedom, when generating alternate paths.

With that freedom come new challenges in path computation and visualization. To date, no research group has produced a path planner that satisfactorily addresses all of the challenges presented by this problem. Instead, groups of similar algorithms have emerged that address one aspect of the problem well while neglecting other areas. Offline path planners tend to be the most sophisticated or complete branch of solutions to the problem. They are designed to generate the entire path of a UAV from takeoff to touchdown before it ever leaves the base. Within reason, there is no time constraint dictating how quickly a path must be generated for the UAV mission. The tradeoff between computational time and accuracy then swings far in favor of accuracy. Path planners of this type attempt to use all available

information about mission objectives, enemy locations, and the environment in the computations. They also include better models of the vehicle's dynamics to ensure path feasibility. A drawback to this group of methods is that they are too slow to deal with flight situations where a UAV would need to re-plan a portion of its path in real time.

The counterbalance to offline path planners is online, or real time, algorithms. Real time path planning is important in allowing a UAV to react dynamically to its environment. Time is very important for these algorithms because a solution must be reached before the UAV encounters the obstacle that necessitated the re-planning. Completeness, especially in terms of dynamics or the number of objectives included in the problem formulation is often sacrificed to meet the time constraint. Some online path planners are designed specifically to re-plan and update a portion of the UAV's flight path that has become undesirable for some reason. Other path planners work by continuously updating or extending the UAV's path in short increments as it flies, which is necessary for missions involving tracking or following other entities. Both groups make important contributions to the field of path planning and a more comprehensive solution to the problem will likely involve coupling these groups in some fashion.

Another method of grouping path planners is by the objectives and constraints included in the cost function formulation. Objectives may be based strictly on path characteristics, such as path length, avoiding enemy missile sites, and passing through reconnaissance areas. Others may incorporate characteristics of the UAV flying the mission. Each UAV has a minimum radius of curvature for turning, an achievable range of flight velocities, and maximum climbing and diving angles that are properties of its design. For path planners that do not incorporate flight dynamics, it is usually assumed that the vehicle will take the path waypoints and fly as close to them as possible given its capabilities. Often, the line between objectives and constraints in path planning optimization problems is blurred. Because unconstrained optimization problems are much simpler to code and solve, many constraints are actually treated as components of the cost function, just like objectives. Formulating the problem in this way is made possible by making the cost of violating a constraint prohibitively high so that a path that violates a constraint will never be an optimal path.

There are a wide range of objectives and constraints included in path planning optimization, as seen in the following two examples. The choice of what to include is based solely upon the opinion of the developer as there is no standard yet for path planning objectives and constraints. One example is a path planner designed to minimize aircraft radar cross section for an unmanned combat aerial vehicle to avoid enemy missiles. Another considers four separate objectives; shortest path, terrain avoidance, minimum and maximum elevation above terrain, and minimum radius of curvature for the path. Even path planners that have similar objectives may formulate their cost functions very differently. Cost

function formulation is a very important part of path optimization and there is still plenty of work to be done in determining the best mathematical representation of the problem.

II.3. Mathematical model for UCAV path planning

Primarily, it should be noted that, the development presented in the following is the most used in the work carried out by a large number of researchers (see the following references [26-31], for this reason we have showed it, despite it's not used in our work presented in this Master dissertation.

As a key component of mission planning system, UCAV path planning is a new low altitude penetration technology to achieve the purpose of terrain following and terrain avoidance and flight with evading threat. The goal for path planning is to find an optimal or near-optimal flight path for UCAV to break through the enemy threat environments, and self-survive with the perfect completion of mission. In our work presented in this dissertation, we use the mathematical model for UCAV path planning described as follows.

II.3.1. Threat resource model in UCAV path planning

In this model, S and T are defined as the starting point and the target point (see Fig. II.2), respectively. There are some installations in the combat field, for instance, radars, missiles, and artilleries. The effects of such installations are presented by circles in the combat field of different radiuses and threat weights. If part of its path falls in a circle, an UCAV will be vulnerable to the threat with a certain probability proportional to the distance away from the threat center. Moreover, when the fight path is outside a circle, it will not be attacked. The UCAV flight mission is to calculate an optimal path from S to T.

Meanwhile all the given threat regions in the combat field and the fuel consumption should be considered.

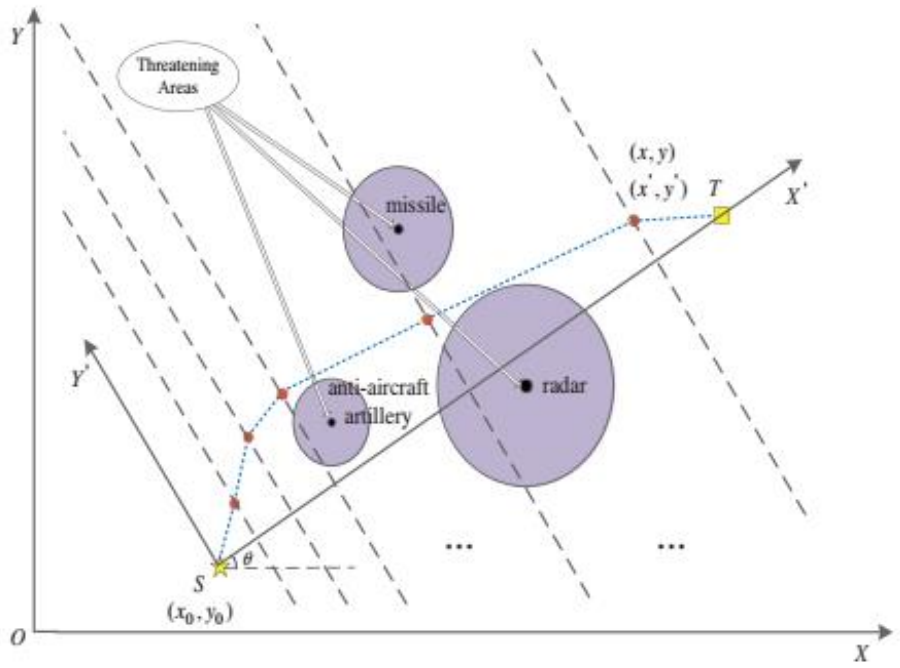


Figure II.1. UCAV battle field model.

To make this problem more specific, we draw a segment ST connecting the starting point S and the target point T. Then, ST is divided into D equal portions and vertical coordinate Y' is optimized on the vertical line for each node to get a group of points composed by vertical coordinate of D points. Obviously, it is easy to get the horizontal abscissas of these points. We can get a path from start point to end point through connecting these points (see the red circle in Fig. II.2) together, so that the route planning problem is transformed into a *D*-dimensional function optimization problem. In Fig. II.1, the original coordinate system is transformed into new coordinate whose horizontal axis is the connection line from the starting point to the target point according to the transformation formula shown in Eq. (II.1), where the point (x, y) is a coordinate in the original coordinate system O_{XY}, the point (x', y') is coordinate in the new rotating coordinate system $O_{X'Y'}$, θ is the rotation angle of the coordinate system.

$$\theta = arcsin\frac{y_2 - y_1}{|\overrightarrow{AB}|} \tag{II.1}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

II.3.2. Performance Indicator

A performance indicator of path planning for UCAV mainly contains the completion of the mandate of the safety performance indicator and fuel performance indicator, that is, indicators with the least threat and the least fuel.

Minimum of performance indicator for threat:

$$\min J_t = \int_0^L w_t dl \qquad L \text{ is the length of the path.}$$

Minimum of performance indicator for fuel:

$$\min J_f = \int_0^L w_f dl \qquad L \text{ is the length of the path.}$$

Then the total performance indicators for UCAV route:

$$J = \lambda J_t + (1-\lambda) J_f$$
$$= \lambda \int_0^L w_t dl + (1-\lambda) \int_0^L w_f dl \tag{II.2}$$

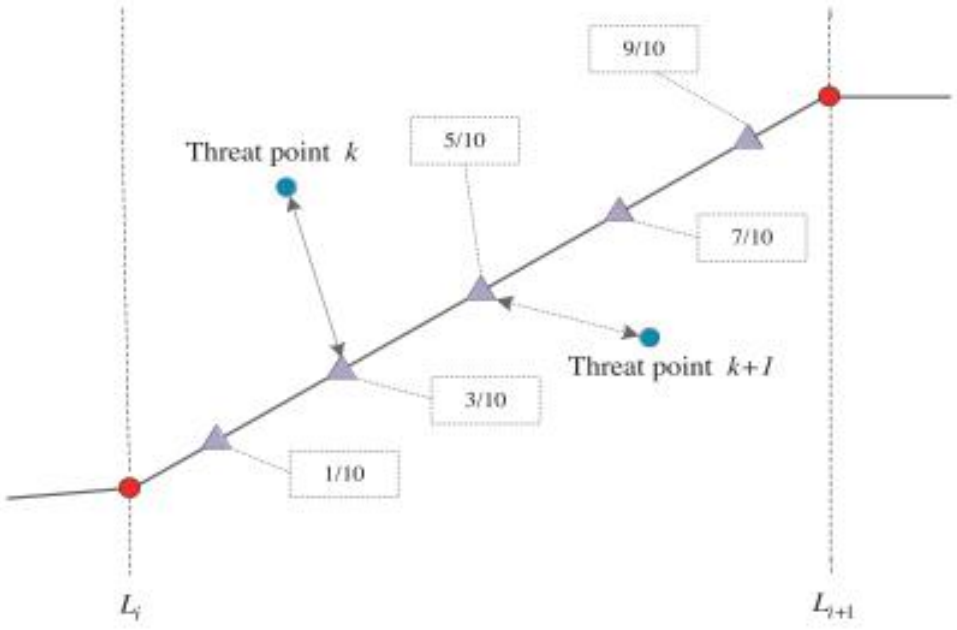


Figure II.2. Computation of threat cost.

where J is the weighted sum of flight cost for this flight path, w_t and w_f are variables close related with the current path point and changing along with *l*, which respectively present the threat cost and fuel cost of each line segment on the route, and L is the total length of the generated path. $\lambda \in [0,1]$

denotes the weighting parameter. When λ is close to 1, a shorter path is needed to be planned, and less attention is paid to the radar's exposed threat. In addition, when λ is close to 0, it requires avoiding the threat as far as possible on the cost of sacrifice the trajectory length. In this work (see the reference [30]), λ is set to 0.5 according to [30]. The optimized path is founded only when function J reaches its minimal value.

When the UCAV is flying along the path $L_{i\rightarrow i+1}$, the total threat cost generated by N_t threats is calculated using w_t where $L_{i\rightarrow i+1}$ is calculated as follows:

$$L_{i\rightarrow i+1} = \int_0^{length_i} \sum_{k=1}^{N_t} \frac{t_k}{[(x-x_k)^2+(y-y_k)^2]} dl \tag{II.3}$$

So, in order to simplify the integral operations in (II.2), the detection cost w_t, $L_{i\rightarrow i+1}$ from the point along L_i to the one along L_{i+1} is calculated at five sample points [30], as shown in Fig. II.2.

If the distance from the threat point to the end of the each subsegment is within threat radius, the threat cost is calculated as follows [30]:

$$w_t, L_{i\rightarrow i+1} = \frac{length_i}{5} \times \sum_{k=1}^{N_t} \left[t_k \times \left(\frac{1}{d_{0.1,i,k}^4} + \frac{1}{d_{0.3,i,k}^4} + \frac{1}{d_{0.5,i,k}^4} + \frac{1}{d_{0.7,i,k}^4} + \frac{1}{d_{0.9,i,k}^4} \right) \right] \tag{II.4}$$

where N_t is the number of threatening areas, $length_i$ is the i_{th} sub-path length, $d_{0.1,i,k}^4$ is the distance from the 1/10 point on the path and the k_{th} threat center, and t_k is the threat level of k_{th} threat. It is assumed that the speed of UCAV is a constant. It is common practice to assume that the fuel cost J_f is in direct proportion to $length$, which means w_f will be a constant [30]. This in no problem if we set $w_f \equiv 1$ because the polynomial $(1-\lambda)$ in (II.2) is always attached with w_f.

II.4. Conclusion

In this chapter we have talked firstly about, the UAVs path planning problem in a general sense, then we have presented in detail, the different steps followed to develop a mathematical model relative to an UCAV path planning. More precisely, a UCAV battle field model under a schematic representation inspired from reality and a performance evaluation function of route optimization are presented. In the next chapter, we will talk about metaheuristic algorithms in a general sense.

Chapter III:

Optimization Metaheuristics

III.1. Introduction

Optimization can exist in almost every aspect of life including engineering, industry, business or even social science. Metaheuristics are widely recognized as efficient approaches for optimization problems of such areas. Because of simplicity and robustness of produced results, a great interest has been devoted to metaheuristics. In the following, we present some basic aspects related to this kind of algorithm.

III.2. Metaheuristic algorithms

Metaheuristics are generally applied to problems for which there is no satisfactory problem specific algorithm to solve them. In order to find desirable solutions, for the multi-population metaheuristics, the optimization process begins with the creation of an initial group of random solutions (population) that satisfy the restriction of the problem to be solved. Then, the set containing all the solutions is iteratively evaluated by one or more target functions associated with the problem and one iterates over generations to minimize or maximize the objective(s). These iterations run until the solution found meets some pre-defined criteria. This final solution (near optimal solution) is said to be an "optimal" solution and the system reaches a converged state [32]. Note that, though this procedure is quite simple, finding solutions for real-life problems requires considering and addressing several issues, from which the most important are: local optimal avoidance, computational cost of function evaluation, constraints handling, multiplicity of objective functions, and uncertainties.

Basically, to search for an optimal solution of a given problem, a metaheuristic algorithm can be formulated as:

$$\min_{X\in\Omega_x} / \min_{X\in\Omega_x} Q(X), \;\; X = (x_1, \dots \dots x_C) \in \Re^C \qquad \text{(III.1)}$$

where X is the design vector that encodes $\boldsymbol{C}$ decision variables of the problem, usually, a design vector is also called candidate solution. Ω_X is the feasible search space or solution space limited by the lower and upper bounds.

A metaheuristic will be successful on a given optimization problem if it can provide a balance between two cornerstones features: the exploration and the exploitation (also referred to as diversification and intensification, respectively). Exploration is the ability to expand search in wide spread domain to explore unvisited areas such that parts of search space with high-quality solutions are identified. Exploitation, via accumulated search experience, is important to intensify the search in some promising areas. The main differences among the existing metaheuristics concern the particular way in which they try to achieve this balance [33]. Some promising approaches, which can be used to

maintain the trade-off balance, are: parameter tuning, population size control, and diversity maintenance through deterministic, adaptive, and self-adaptive techniques.

Almost all metaheuristics share the following characteristics: they are nature-inspired (based on principles of physics, biology or ethology); they make use of stochastic elements (random variables); they do not use the gradient or Hessian matrix of the objective function; existing several parameters need to be configured to the problem at hand [34]. The classification of metaheuristics can be performed with respect to different aspects concerning the search path they follow, the use of memory, the kind of neighborhood exploration used or the number of solutions maintained during iterations. For a formal classification of metaheuristics, we refer the reader to [35].

In order to analyze the performance of metaheuristic algorithms, researchers have commonly used benchmark test functions such as those used in [36]. Then, some validation criteria such as best, worst, mean and standard deviation of objective function values obtained over specific number of runs are used to make a comparison. For specific engineering problems, which are also solved by using metaheuristics, the quality of the final solution is evaluated by using validation criteria in those domains (such as Dice coefficient, Hausdorff distance, etc.).

The idea of solving optimization problems through heuristic approaches was envisioned more than forty years ago when Operations Research was in its infancy during World War II. The formal kick off of metaheuristic research took place when initial metaheuristic methods like Simulated Annealing and Tabu Search were introduced in 1980s. However, the boom of this field of research was witnessed in 1990s after the wider applications of Genetic algorithms (GA), Ant colony optimization (ACO) and Particle swarm optimization (PSO). Despite the success of metaheuristic methods on diversified areas of science, engineering and technology, there remains a sufficient gap that needs to be filled in order to reach maturity level as compared to other established fields of research.

Now that the above description has already established preliminary knowledge about metaheuristics, the upcoming sections explore more about some attractive categories of metaheuristics.

III.3. Single-solution based metaheuristics

Single-solution based metaheuristics, also called *trajectory methods*, are based on the evolution of a single solution during the search process. Typically, these methods start with a single initial solution and move away from it, describing a trajectory in the search space. Usually, basic single solution-based algorithms are more exploitation oriented. Many methods in this category can be found in the literature; however, in this section we present two common ones used not only in image segmentation but also in other applications, named Simulated Annealing (SA) and Tabu search (TS).

III.3.1. Simulated annealing

Simulated annealing (SA) is a stochastic optimization technique, first introduced by Kirkpatrick [37] and independently by Cerny [38]. SA is inspired by the annealing technique in which high temperature metal if cooled at an appropriate cooling rate will reach an absolute minimum energy state related to complete atomic ordering of metal. If the high temperature metal is cooled at a fast rate, the atoms will reach a sub-optimal energy state. The hypothesis of this method is that: system energy at higher temperature (T) is allowed to move uphill as well as downhill, but as temperature (T) goes down gradually, the energy is allowed to move downhill only. Thus, evolution of states is sensitive to coarser energy variation when (T) is large and to finer variation when (T) is small.

Transposing the process of annealing to the optimization process is based on the following analogies: the objective function to be optimized is similar to the energy of a material, and the temperature is represented by a controllable parameter defining the cooling scheme. The algorithm begins with selecting an initial solution and later generating a new state, randomly generating a new solution $(X' = X + \Delta X)$ in the neighborhood of the current solution (X); this is called a neighbour solution. This new state is evaluated and compared with the previous solution in terms of fitness value (f). If the solution of the new state is better than the previous one, it is accepted; but if it is not, it is accepted with a probability, $P(T, f_{X'}, f_X) = exp\left(-\frac{f_{X'}+f_X}{T}\right)$. The temperature (T) is gradually decreased during the search process.

By repeatedly following this Metropolis rule of acceptance, a sequence of solutions is generated, which constitutes a Markov chain (in the sense that each solution depends on only that one which immediately precedes it). With this formalism in place, it is possible to show that, when the chain is of infinite length, the system can reach (in practice, can approach) an equilibrium points at the temperature considered. The process is terminated when the system is "solidified" (either the temperature has reached zero or no more moves causing an increase in energy have been accepted).

It is clear that, at high temperature, $exp\left(-\frac{f_{X'}+f_X}{T}\right)$ is close to 1, and therefore the majority of the moves are accepted and the algorithm becomes equivalent to a simple random walk in the configuration space. However, at low temperature, $exp\left(-\frac{f_{X'}+f_X}{T}\right)$ is close to 0, and therefore the majority of the moves that increase the energy are rejected. At an intermediate temperature, the algorithm intermittently allows transformations that degrade the objective function: hence it leaves a chance for the system to be pulled out of a local minimum. Note that the cooling schedule for T is critical to the efficiency of SA. If T is reduced too rapidly, a premature convergence to a local minimum may occur. In contrast, if it is too slow, the algorithm is very slow to converge. Given a "sufficiently large" number of iterations at each temperature, SA is proved to converge almost surely to the global optimum [37]. The Simulated Annealing method is presented in Algorithm III.1.

Algorithm III.1: The general SA algorithm [37]

Initialization: Initialize randomly a solution $\mathbf{X}$ for the system to be optimized; initialize the temperature T
Results : The optimal solution $\mathbf{X}^*$
$k \leftarrow 1$
repeat
1 Apply random perturbations to the state: $\mathbf{X}' \leftarrow (\mathbf{X} + \triangle\mathbf{X})$
2 Evaluate changes in the energy: $\triangle \mathrm{f}_{\mathbf{X}} \leftarrow (\mathrm{f}_{\mathbf{X}'} - \mathrm{f}_{\mathbf{X}})$
3 **if** $\triangle \mathrm{f}_{\mathbf{X}} < 0$ **then**
3.1 Keep the new state: $\mathbf{X} \leftarrow \mathbf{X}'$
else
3.2 Accept the new state with probability: $P(T, \mathrm{f}_{\mathbf{X}'}, \mathrm{f}_{\mathbf{X}}) = \exp(-\frac{\mathrm{f}_{\mathbf{X}'} - \mathrm{f}_{\mathbf{X}}}{T})$
4 Decrease T: $T \leftarrow (T - \triangle T)$
5 $k \leftarrow k + 1$
until *termination conditions are met*

SA has been successfully applied to many engineering problems [39], though it has been found too slow to converge to the global optimum.

III.3.2. Tabu search

Tabu search (TS) is also a single-solution based metaheuristic, which was formalized in 1986 by Glover [40]. Originally, the method was developed for very large combinatorial optimization and was later extended to continuous optimization [41]. Tabu search uses a set of strategies and learned information to mimic human insights for problem-solving. Essentially, TS is a greedy local search (also known as hill-climbing) method that explores the solution space beyond local optimality and adopts a memory structure that imitates human behavior, and uses past experiences to improve decision-making.

In TS, a neighborhood structure is introduced to the solution space $\mathcal{A}(X)$ in the following way: Each solution $X \in \mathcal{A}(X)$ has an associated set $\mathcal{N}(X) \subset \mathcal{A}(X)$, with $X \notin \mathcal{N}(X)$, called the *neighborhood* of X. $\mathcal{N}(\boldsymbol{X})$ is defined as the set of all $X' \in \mathcal{A}(X)$ that can be obtained directly from X by a modification called a move $m(X; X')$ from X to X'. The number of solutions in a neighborhood may be very large and the quality of these solutions may vary a lot. Usually, the sizes of neighborhoods are much larger than can be evaluated by the algorithm, and thus only the most attractive part of a neighborhood is actually explored.

In order to guide the search process in an intelligent manner, TS procedure incorporates a flexible memory structure as its essential component. Various types of memory structures can be used to remember specific properties of the trajectory through the search space that the algorithm has undertaken. The memory structure, D, is formally defined using so-called *tabu lists*: $T_i, i \in \{1, \ldots, p\}$

with lengths $L_i, i \in \{1, \ldots., p\}$, and $D = T_1 \cup \ldots..\cup T_P$ The main role of this structure in TS is to prevent cycling back to some previously generated solutions and to diversify the search process, i.e., induce the search of new subregions of the solution space $\mathcal{A}(X)$. However, the disadvantage of the use of a tabu list is that a forbidden attribute may be part of an attractive solution of a neighborhood that has not been visited so far. To cope with this problem, an aspiration criterion, involving a set of rules, is used to override tabu restrictions (if the aspiration criterion is satisfied, the move is allowed).

A typical aspiration criterion is to keep a solution that is better than the best solution found so far. It should be noted that the length of the tabu list controls the memory of the search process. If the length of the list is low, the search will concentrate on small areas of the search space (intensification). On the contrary, a high length forces the search process to explore larger regions (diversification), because it forbids revisiting a higher number of solutions.

The TS procedure starts from an initial feasible solution and at each step moves from the current solution to the best one in its neighborhood, trying to reach an optimal solution. At step (k), a subset $\mathcal{N}'(X^{(k)})$ of the modified neighborhood $\mathcal{N}(X^{(k)}, D)$, which is defined according to D, is constructed and the best solution in $\mathcal{N}'(X^{(k)})$ is chosen as the next solution $X^{(k+1)}$ (even if it brings no improvement in terms of fitness value). Then, this solution $X^{(k+1)}$ will replace the previous one X^* (the best solution discovered in the neighbourhood so far) if it is found to be better in terms of fitness value.

At the end of the step (k), the memory structure D and the aspiration criterion are updated, preparing for next iteration $(k+1)$. If the stopping criteria are met, the TS algorithm terminates and the best solution found is returned. The TS procedure can be expressed in the most general way as in Algorithm III.2.

Algorithm III.2: The general TS algorithm

Initialization: Generate an initial solution $\mathbf{X}^{(0)}$; create the memory structure $\mathcal{D}$
Results : The optimal solution $\mathbf{X}^*$
$k \leftarrow 1$
repeat
1 Define $\mathcal{N}(\mathbf{X}^{(k)}, \mathcal{D})$ and evaluate fitness values, $\mathrm{f}_{\mathcal{N}(\mathbf{X}^{(k)}, \mathcal{D})}$
2 Generate a set $\mathcal{N}'(\mathbf{X}^{(k)})$ as a subset of $\mathcal{N}(\mathbf{X}^{(k)}, \mathcal{D})$
3 Determine $\mathbf{X}^{(k+1)}$ by optimizing the objective function over $\mathcal{N}'(\mathbf{X}^{(k)})$
4 **if** $\mathrm{f}_{\mathbf{X}^{(k+1)}}$ *better than* $\mathrm{f}_{\mathbf{X}^*}$ **then**
4.1 $\mathbf{X}^* \leftarrow \mathbf{X}^{(k+1)}$
5 Update the memory structure $\mathcal{D}$, and the aspiration criteria
6 $k \leftarrow k + 1$
until *the stopping criteria are satisfied*

The stopping criteria can have the following forms. First, the TS procedure is terminated if the number of consecutive iterations, performed without any improvement of the currently best objective

function value, is greater than a specified number. Second, if the optimal fitness value of the objective function is known in advance, then the process can be interrupted as soon as this value is reached.

TS has been applied to many optimization problems: vehicle routing [42], continuous optimization [43], multi-criteria optimization [44], stochastic programming [45] and real-time decision problems [46].

III.4. Population-based metaheuristics

Population-based metaheuristics deal with a set of solutions rather than with a single solution. In this section, we present the most studied population-based methods related to Evolutionary Computation (EC) and Swarm Intelligence (SI). EC algorithms are inspired by Darwin's evolutionary theory, where a population of individuals is modified through recombination and mutation operators. SI is a branch of biologically inspired algorithms which is focused on the collective behavior of swarms in order to develop metaheuristics which mimic the swarm's problem solution abilities.

III.4.1. Evolutionary Computation (EC)

From a conventional point of view, an EC method is an algorithm that simulates at some level of abstraction a biological, natural or social system. Evolutionary computation (EC) [47] methods are derivative-free procedures, which do not require that the objective function must be two-times differentiable or uni-modal. Therefore, EC methods as global optimization algorithms can deal with non-convex, non-linear, and multimodal problems subject to linear or nonlinear constraints with continuous or discrete decision variables. Despite the existing differences in these methods, they all share a common underlying idea of simulating the evolution of individual structures via processes of evolutionary operators (*selection, recombination, mutation*). The general scheme of an evolutionary algorithm can be given in Algorithm III.3 in a pseudocode form.

Algorithm III.3: The general scheme of an evolutionary algorithm [47]

Initialization: Initialize the population $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_P\}$ randomly; evaluate their fitness values
Results : The optimal solution $\mathbf{X}^*$
$k \leftarrow 1$
repeat
1 *Select* parents
2 *Recombine* pairs of parents
3 *Mutate* a few individuals
4 *Evaluate* new individuals
5 *Select* individuals for the next generation
6 $k \leftarrow k + 1$
until *the stopping criteria are satisfied*

EC methods do not require hypotheses on the optimization problem nor any kind of prior knowledge on the objective function. They obtain knowledge about the structure of an optimization problem by utilizing information obtained from the possible solutions (i.e., candidate solutions) evaluated in the past. This knowledge is used to construct new candidate solutions which are likely to have a better quality. Figure III.1 presents a graphical representation of a basic cycle of an EC method.

In order to solve an optimization problem by using evolutionary computation method, a population $X = \{X_1, \ldots., X_P\}$ with P candidate solutions (individuals) evolve from the initial point $(k = 1)$ to a total number of iterations, N_{iter}. In its initial point, the algorithm begins by initializing the set of P candidate solutions with values that are randomly and uniformly distributed between the prespecified lower (x_{min}) and upper (x_{max}) limits. In each iteration, a set of evolutionary operators are applied over the population $X^{(k)}$ to build the new population $X^{(k+1)}$. The quality of each candidate solution $X_i^{(k)}$ i is evaluated by using an objective function representing the fitness value, $f_{X_i^{(k)}}$. During the evolution process, the best candidate solution X^* seen so far is preserved since it represents the best available solution.

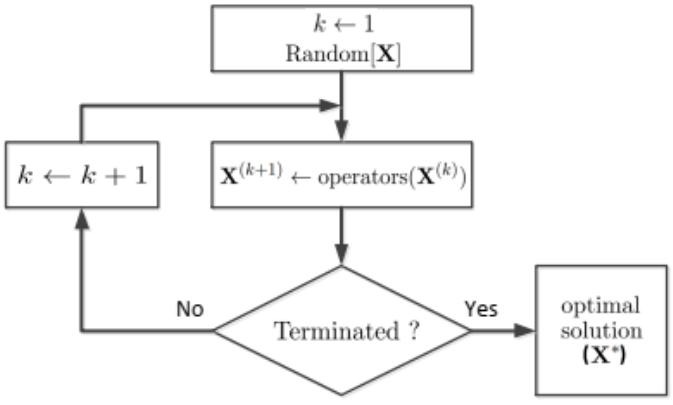


Figure III.1. The basic cycle of an EC method.

Over the years, Evolutionary algorithms have been applied with a good measure of success to many areas such as combinatorial optimization problems [48], constrained optimization problems [49], Data mining and Knowledge discovery [50], etc.

III.4.2. Swarm Intelligence (SI)

Swarm Intelligence (SI) is an innovative distributed intelligent paradigm for solving optimization problems that takes inspiration from the collective intelligence behavior of self-organized and decentralized systems, e.g., artificial groups of simple agents. The members of a swarm must be active, dynamic and simple (with no or very little inherent knowledge of the surroundings). The algorithms

in the class of swarm intelligence primarily consist of two phases, namely the *variation phase* and the *selection phase*. These phases are responsible for maintaining the balance between exploration and exploitation and forcing the entire swarm, i.e., the set of potential solutions, to update their positions. While the variation phase explores different areas of the search space, the selection phase works for the exploitation of the previous experiences [51].

A group of homogeneous agents exhibits the swarm intelligence if and only if it follows two conditions: self-organization and division of labour. According to Bonabeau et al. [52], self-organization is categorized into four strategies: positive feedback, negative feedback, fluctuations and multiple interactions. While positive feedback is revealed to the input system to promote formation of appropriate structures (provides diversity), negative feedback balances the positive feedback and provides stabilization to the collective pattern (refers to exploitation). Fluctuations meanwhile provide new situations in the process and help to get rid of stagnation. Multiple interactions improve the overall intelligence of the swarm by sharing information among individuals within their searching area. The second condition, division of labor, is defined as the simultaneous execution of various simple and feasible tasks by individuals. This division allows the swarm to be capable of handling changed conditions in the search space.

In this section, we briefly present the most notable swarm intelligence techniques for obtaining approximate solutions to optimization problems: Ant colony optimization (ACO), Particle swarm optimization (PSO) and Cuckoo search (CS). These optimization methods will be explained below.

A. Ant colony optimization

Ant colony optimization (ACO) was introduced by M. Dorigo and colleagues [53] for the solution of hard combinatorial optimization problems. ACO is inspired from the way how the ant colonies find the shortest route between the food source and their nest. When searching for food, these ants initially explore the area surrounding their nest by performing a randomized walk. Along their path between

food and nest, ants deposit a chemical pheromone trail on the ground that guides other ants to the food source [53]. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. After some time, the shortest path presents a higher concentration of pheromones, and therefore attracts more ants. Ant system [53] exploited this characteristic of real ant colonies to build solutions for an optimization problem and exchange information on their quality through a communication scheme that is reminiscent of the one adopted by real ants.

ACO involves solution construction on a graph. Many ants travel through the solution space adding solution components to partial solutions until they reach a complete solution. The selection of the components depends on the pheromone content of the paths and a heuristic evaluation [53]. At each step of construction, the m^{th} ant selects the next node using a probabilistic action selection rule, which dictates the probability at which the m^{th} ant will choose to go from the current node (i) to next node (j). At the k^{th} generation (or iteration), the probability is defined as follows:

$$p_{ij}^{m}(k) = \begin{cases} \frac{[\tau_{ij}(k)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^m}[\tau_{il}(k)]^{\alpha}[\eta_{il}]^{\beta}} & if\ \mathrm{j} \in\ \mathcal{N}_i^m \\ 0 & \text{Otherwise} \end{cases} \quad \text{(III.1)}$$

where τ_{ij} is the pheromone content of the arc from node (i) to node (j), $\mathcal{N}_i^m$ is the set of nodes that remain to be visited by the m^{th} ant positioned at node (i) to make the solution feasible. η_{ij} is the heuristic information for going from node (i) to node (j). The heuristic information is a measure of the cost of extending the current partial solution. The constants α and β represent the influence of pheromone content and heuristic, respectively. Once a solution is built, it is evaluated and amount of pheromone is relatively deposited to the quality of the solution. The ants deposit pheromone on the arcs they visited as follows:

$$\tau_{ij}(k+1) = \tau_{ij}(k) + \sum_{m=1}^{p} \Delta\tau_{ij}^{m}(k) \quad \text{(III.2)}$$

where $\Delta\tau_{ij}^{m}$ is the amount of pheromone ant m will add to the arc going from node (i) to node (j), and P is the total number of ants. The amount of pheromone added is defined by:

$$\Delta\tau_{ij}^{m}(k) = \begin{cases} \frac{Q}{L_m(k)} & if\ arc\ is\ in\ the\ path\ of\ ant\ m \\ 0 & \text{Otherwise} \end{cases} \quad \text{(III.3)}$$

Where is a constant and $L_m(k)$ is the total cost of the path solution (the penalized objective function value for ant m) at the k^{th} iteration. All arcs in the same path will have the same cost value.

Once all ants of the colony have completed the construction of their solution, pheromone evaporation, which provides an effective strategy to avoid rapid convergence to local optima and to favor the exploration of new regions of the search space, is performed as follows:

$$\tau_{ij}(k+1) = (1-\rho)\tau_{ij}(k) + \rho\tau_{ij}(0) \quad \text{(III.4)}$$

Where $\rho \in (0,1)$ is a pheromone decay parameter. The basic ACO algorithm can be summarized as in in Algorithm III.4.

Algorithm III.4: The ACO algorithm [53]

Initialization: Initialize the pheromone matrix $\mathcal{T} = [\tau_{ij}]$ and the number of ants P; and set:
$\mathbf{X}^* \leftarrow$ Null
Results : The optimal solution, $\mathbf{X}^*$
$k \leftarrow 1$
repeat
1 Initialize the set of solutions obtained by ants
/* Solution construction */
2 **for** *each ant* **do**
2.1 Choose next node by applying the state transition rule given by Eq. **(III.1)**
2.2 Update pheromone on ant's path
2.3 Build solution by the selected items
2.4 Evaluate solution and update the best one, $\mathbf{X}^*$
/* Update pheromones */
3 Update and evaporate the pheromone matrix, $\mathcal{T}$, according to solutions and $\mathbf{X}^*$
4 $k \leftarrow k+1$
until *the stopping criteria are satisfied*

ACO has several advantages including offering positive feedback resulting in rapid solution finding, and having distributed computation which avoids premature convergence. However, ACO has some drawbacks such as slower convergence compared with other heuristic-based methods and lack of a centralized processor to guide it towards good solutions. Even though the convergence is frequently obtained, the time for convergence is uncertain. In addition, ACO shows its poor performance within problems with large search spaces.

Over the years, a number of ACO variants, which share the same characteristic idea, have been created with the aim to improve overall performance and many successful applications are now available [54].

B. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population-based stochastic optimization technique regarded as a global search strategy, originally introduced by Kennedy and Eberhart [55]. In the PSO algorithm, each member of the population, called *particle*, represents a potential solution to the optimization problem; and the population, called *swarm*, is evolved through successive iterations. The quality of a candidate solution is evaluated by the fitness value, associated with each particle. Each particle in the swarm with size of P, denoted by i, has a position vector $\mathrm{X}_i = (x_{ir})_C$, a velocity vector $\mathrm{V}_i = (v_{ir})_C$, its own best position $pBest$ found so far, and interacts with neighboring particles through the best position $gbest$ discovered in the neighborhood so far. At the k^{th} iteration, each particle is moved according to equations (III.5) and (III.6):

$$\mathrm{V}_I^{(k+1)} = w^{(k)}\mathrm{V}_I^{(k)} + c_1 r_1 \left[pBest^{(k)} - \mathrm{X}_i^{(k)}\right] + c_2 r_2 \left[gBest^{(k)} - \mathrm{X}_i^{(k)}\right] \quad \text{(III.5)}$$

$$\mathrm{X}_i^{(k+1)} = \mathrm{X}_i^{(k)} + \mathrm{V}_I^{(k+1)} \quad \text{(III.6)}$$

where c_1 and c_2 are acceleration coefficients that scale the influence of the *cognitive* and *social* components; r_1 and r_2 are two random values, uniformly distributed in $[0,1]$; and $w^{(k)}$ is inertia weight at the k^{th} iteration. The higher $w^{(k)}$ is, the higher the possibility of searching in the global solution space is, and the smaller $w^{(k)}$ is, the higher the possibility of searching in the local solution space is.

There are two basic criteria for assessing performance of the PSO algorithm, named the convergence speed and the ability to find global optima. To optimize both criteria, keeping balance between global exploration and local exploitation is crucial. From equations (III.5) and (III.6), it is clear that the performance depends not only on the controlling parameters $\{w^{(k)}, c_1, c_2\}$, but also on the size and structure of the neighborhood.

C. Cuckoo Search (CS)

The Cuckoo search (CS) algorithm is one of the latest metaheuristic approaches introduced by Yang and Deb in 2009 [56]. This algorithm is inspired by the behavior of cuckoo species, such as brood parasites, in combination of the Levy flight behavior of some birds and fruit flies. CS employs three basic rules or operations in its implementation. First, each cuckoo is only allowed to lay one egg in each iteration, and the nest is chosen randomly by the cuckoo to lay its egg in. Second, the eggs and nests with high quality are carried forward to the next generation. Finally, the number of available host nests is fixed and the egg laid by a cuckoo is discovered by a host bird using a probability of $p_a \in [0,1]$. In

other words, the host can choose whether to throw the egg away or abandon the nest and build a new nest completely.

The main merits of the CS over other optimization algorithms are as follows: the number of parameters needed to be configured in the initial search is very little, and the inexperienced user can easily interact with it. CS has the strength points of TS in exploitation through random walk and of EC in exploration through Levy flights. It is an efficient metaheuristic algorithm that balances between the local search strategy (exploitation) and the whole space (exploration) [56].

However, CS also comprises several limitations such as low performance of local search, easily trapping into local optimum and lack of diversity of the local search and global search [56].

III.5. Multi-objective metaheuristics

Many sectors of industry (mechanics, chemistry, telecommunication, environment, transport, etc.) are concerned with complex problems of large dimensions that must be optimized. These optimization problems are seldom mono-objective; on the contrary, they frequently have several contradictory criteria or objectives that must be satisfied simultaneously. Multi-objective optimization (MOO) is a discipline centered on the resolution of these kinds of problems. For them, instead of seeking for a single solution, we seek for a set of non-dominated solutions representing the compromise solutions between different conflicting objectives, called the Pareto optimal set. A solution that belongs to this set is said to be a Pareto optimum and, when the solutions of this set are plotted in the objective space, they are collectively known as the *Pareto front*.

Furthermore, many real-world multi-objective optimization problems (MOPs) typically need computationally expensive methods for computing the objective functions and constraints. In this context, deterministic techniques are generally not applicable, which leads therefore to using approximate methods. Among them, metaheuristics are nowadays used extensively to deal with MOPs.

Algorithm III.5: The general scheme of a basic MOO evolutionary algorithm

Initialization: Initialize population and controlling parameters; evaluate fitness values of all individuals; build an initial Pareto set approximation
Results : Pareto set approximation
$k \leftarrow 1$
repeat
1 Evolve the population using evolutionary operators (crossover, mutation, and selection) to generate a new population
2 Evaluate the individuals of the created population in terms of fitness value
3 Build a new Pareto set approximation using dominance concept
4 Maintain non-dominated solutions (external archives, crowding distance, adaptive grid, density estimation, ranking relations,...)
5 $k \leftarrow k + 1$
until *the stopping criteria are satisfied*

The success of a MOO metaheuristic algorithm depends mainly on the definition of some issues: (1) solution representation; (2) choice of objective function; and (3) design of the operators. To assess its performance, three items are interesting to measure: (1) the number of elements of the Pareto optimal set found; (2) the distance of the Pareto front produced by the algorithm with respect to the theoretical Pareto front (assuming we know it); and (3) the spread of solutions found. The main steps of a basic MOO evolutionary algorithm can be summarized as in Algorithm III.5.

Over the years, many methods have been proposed including both evolutionary approaches, MOEAs, (mostly) and non-evolutionary approaches [35].

An excellent repository ((http://delta.cs.cinvestav.mx/~ccoello/EMOO/)) in which the state-of-the-art MOO algorithms can be found is available for research community. Nowadays, many MOO metaheuristic algorithms have been successfully used in different areas, for instance, scheduling, data mining, circuits and communications, control systems and robotics, manufacturing, and image processing.

III.6. Hybrid metaheuristics

Recently, quite an impressive number of algorithms have been reported that do not purely follow the paradigm of a single traditional metaheuristic. On the contrary, they combine various algorithmic components, often originating from algorithms of various research areas on optimization. These approaches are commonly referred to as *hybrid metaheuristics*. The main motivation behind the hybridization of different algorithms is to exploit the complementary character of different optimization strategies, while simultaneously trying to minimize any substantial disadvantages. It is mostly due to the *no free lunch* theorems [57] that there cannot exist a general optimization strategy which is globally better than any other. In fact, to solve a problem at hand most effectively, one almost always needs a specialized algorithm that includes several mechanisms.

All the existing metaheuristics share some ideas and differ among each other by certain characteristic *key* components; making a toolbox of these components, from which we can pick in the design of an optimization algorithm (hybrid algorithm), is the most appropriate approach tailored to the specific characteristics of one problem at hand [58]. Unfortunately, developing an effective hybrid approach is in general a difficult task which requires expertise from different areas of optimization. Moreover, the literature shows that it is non-trivial to generalize, because the hybridization of algorithms might work well for specific problems, but it might perform poorly for others.

The work on hybrid algorithms is relatively recent and can be subdivided into two different categories: collaborative hybrids and integrative hybrids [59]. In collaborative hybrids, two or more algorithms are combined in manner of running either in sequential or parallel. On the other hand, in integrative hybrids, one algorithm is regarded as a subordinate, embedded in a master metaheuristic.

Good resources for studying on hybrid metaheuristics can be found in [60-64]. It should be noted that though hybrid approach offers a great advantage of increasing the diversity in a population and hence enhancing the search capability of the developed algorithm, there are some drawbacks. First, the hybridization process usually creates extra components, and hence the complexity of the hybrid method is increased. Second, the developed algorithm usually uses a higher number of (internal or implicit) iterations. In addition, most hybrid algorithms require an increasing number of tuning parameters, and due to its complicated structure, a hybrid algorithm is harder to be analyzed.

However, in reality, hybrid algorithms have been proved successful in solving a wide range of applications, such as power systems [61], scheduling [62], telecommunications [63], data clustering [64], and many others.

III.7. Conclusion

In this chapter we have talked firstly about, metaheuristic algorithms, what do these words mean exactly! After, a simple explanation of the so-called Single-solution based metaheuristics, is presented with an illustration of two examples (the most famous) of algorithms belong to this methods kind.

The population-based metaheuristics family, with its two main types, Evolutionary Computation (EC) and Swarm Intelligence (SI), is also presented. Finally, a brief overview of multi-objective metaheuristics and Hybrid metaheuristics is illustrated.

In the next chapter, we will show in detail the application of an algorithm belongs to the Swarm Intelligence algorithms, called Artificial bee colony algorithm (ABC), in the problem of the UCAVs path planning.

.

Chapter IV:

Grey Wolf Optimizer (GWO) Based UCAVs Path Planning

IV.1. Introduction

Unmanned combat aerial vehicles (UCAVs) have been of great interest to military organizations throughout the world due to their outstanding capabilities to operate in dangerous or hazardous environments. UCAV path planning aims to obtain an optimal flight route with the threats and constraints in the combat field well considered. In this chapter, a Grey Wolf Optimizer (GWO) algorithm is applied in this optimization scheme. Simulation results confirm that GWO algorithm is adaptable and very suitable for the UCAV path planning problem.

IV.2. Grey Wolf Optimizer (GWO) algorithm

In this section the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

IV.2.1. Inspiration

Grey wolf (Canis lupus) belongs to Canidae family. Grey wolves are considered as apex predators, meaning that they are at the top of the food chain. Grey wolves mostly prefer to live in a pack. The group size is 5–12 on average. Of particular interest is that they have a very strict social dominant hierarchy as shown in Figure IV.1. The leaders are a male and a female, called alphas. The alpha is mostly responsible for making decisions about hunting, sleeping place, time to wake, and so on. The alpha's decisions are dictated to the pack. However, some kind of democratic behavior has also been observed, in which an alpha follows the other wolves in the pack. In gatherings, the entire pack acknowledges the alpha by holding their tails down. The alpha wolf is also called the dominant wolf since his/her orders should be followed by the pack [20]. The alpha wolves are only allowed to mate in the pack. Interestingly, the alpha is not necessarily the strongest member of the pack but the best in terms of managing the pack. This shows that the organization and discipline of a pack is much more important than its strength. The second level in the hierarchy of grey wolves is beta. The betas are subordinate wolves that help the alpha in decision-making or other pack activities. The beta wolf can be either male or female, and he/she is probably the best candidate to be the alpha in case one of the alpha wolves passes away or becomes very old. The beta wolf should respect the alpha, but commands the other lower-level wolves as well. It plays the role of an advisor to the alpha and discipliner for the pack. The beta reinforces the alpha's commands throughout the pack and gives feedback to the alpha. The lowest ranking grey wolf is omega. The omega plays the role of scapegoat. Omega wolves always have to submit to all the other dominant wolves. They are the last wolves that are allowed to eat. It may seem the omega is not an important individual in the pack, but it has been observed that the whole pack face internal fighting and problems in case of losing the omega. This is due to the venting of violence and frustration of all wolves by the omega(s). This assist satisfying the entire pack and maintaining the dominance structure. In some cases, the omega is also the babysitters in the pack. If a wolf is not an

alpha, beta, or omega, he/she is called subordinate (or delta in some references). Delta wolves have to submit to alphas and betas, but they dominate the omega. Scouts, sentinels, elders, hunters, and caretakers belong to this category. Scouts are responsible for watching the boundaries of the territory and warning the pack in case of any danger. Sentinels protect and guarantee the safety of the pack. Elders are the experienced wolves who used to be alpha or beta. Hunters help the alphas and betas when hunting prey and providing food for the pack. Finally, the caretakers are responsible for caring for the weak, ill, and wounded wolves in the pack. In addition to the social hierarchy of wolves, group hunting is another interesting social behavior of grey wolves. According to [20] the main phases of grey wolf hunting are as follows:

-Tracking, chasing, and approaching the prey,

-Pursuing, encircling, and harassing the prey until it stops moving,

-Attack towards the prey.

These steps are shown in Figure IV.2. In this work this hunting technique and the social hierarchy of grey wolves are mathematically modeled in order to design GWO and perform optimization.

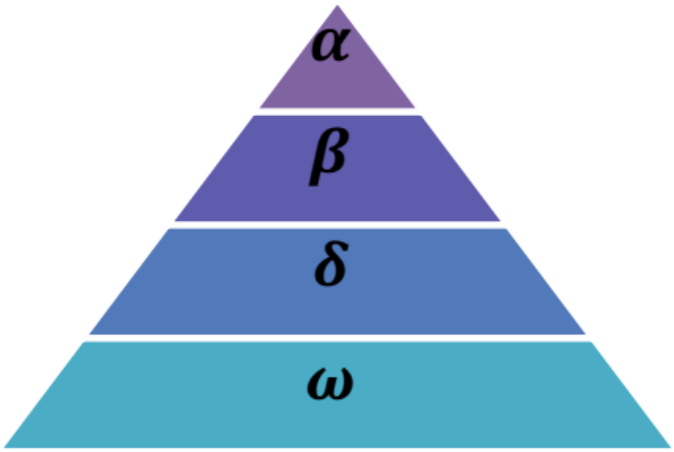


Figure IV.1. Hierarchy of grey wolf (dominance decreases from top down).

IV.2.2. Mathematical model and algorithm

In this subsection the mathematical models of the social hierarchy, tracking, encircling, and attacking prey are provided. Then the GWO algorithm is outlined.

IV.2.2.1. Social hierarchy

In order to mathematically model the social hierarchy of wolves when designing GWO, we consider the fittest solution as the alpha (α). Consequently, the second and third best solutions are named beta (β) and delta (δ) respectively. The rest of the candidate solutions are assumed to be omega (ω). In

the GWO algorithm the hunting (optimization) is guided by α, β, and δ. The ω wolves follow these three wolves.

Figure IV.2. Hunting behavior of grey wolves: (A) chasing, approaching, and tracking prey (B–D) pursuiting, harassing, and encircling (E) stationary situation and attack [?].

IV.2.2.2. Encircling prey

As mentioned above, grey wolves encircle prey during the hunt. In order to mathematically model encircling behavior the following equations are proposed:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_p}(t) - \vec{X}(t)\right| \quad \text{(IV.1)}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A} \cdot \vec{D} \quad \text{(IV.2)}$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\overrightarrow{X_p}$ is the position vector of the prey, and $\vec{X}$ indicates the position vector of a grey wolf. The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows:

$$\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} \cdot \vec{a} \quad \text{(IV.3)}$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \quad \text{(IV.4)}$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and $\overrightarrow{r_1}$, $\overrightarrow{r_2}$ are random vectors in [0,1].

To see the effects of equations (IV.1) and (IV.2), a two-dimensional position vector and some of the possible neighbors are illustrated in Figure IV.3 (a). As can be seen in this figure, a grey wolf in the position of (X, Y) can update its position according to the position of the prey (X^*, Y^*). Different places around the best agent can be reached with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors. For instance, $(X^* - X, Y^*)$ can be reached by setting $\vec{A} = (1,0)$ and $\vec{C} = (1,1)$. The possible updated positions of a grey wolf in 3D space are depicted in Figure IV.3 (b). Note that the random vectors $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ allow wolves to reach any position between the points illustrated in Figure IV.3. So a grey wolf can update its position inside the space around the prey in any random location by using equations (IV.1) and (IV.2).

The same concept can be extended to a search space with n dimensions, and the grey wolves will move in hyper-cubes (or hyper-spheres) around the best solution obtained so far.

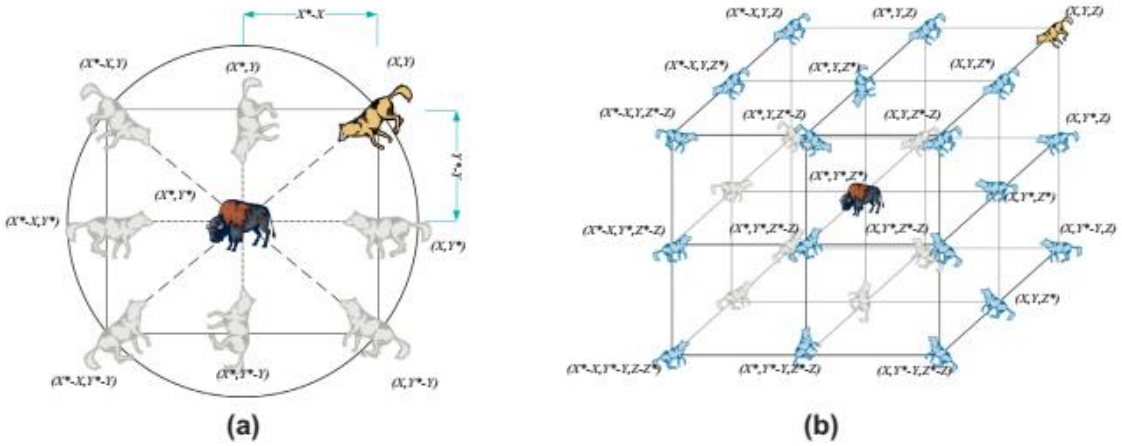


Figure IV.3. 2D and 3D position vectors and their possible next locations.

IV.2.2.3. Hunting

Grey wolves have the ability to recognize the location of prey and encircle them. The hunt is usually guided by the alpha. The beta and delta might also participate in hunting occasionally. However, in an abstract search space we have no idea about the location of the optimum (prey). In order to mathematically simulate the hunting behavior of grey wolves, we suppose that the alpha (best candidate solution) beta, and delta have better knowledge about the potential location of prey. Therefore, we save the first three best solutions obtained so far and oblige the other search agents (including the omegas) to update their positions according to the position of the best search agents. The following formulas are proposed in this regard.

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \vec{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X}\right| \quad \text{(IV.5)}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot (\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot (\overrightarrow{D_\beta}), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot (\overrightarrow{D_\delta}) \quad \text{(IV.6)}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad \text{(IV.7)}$$

Figure IV.4 shows how a search agent updates its position according to alpha, beta, and delta in a 2D search space. It can be observed that the final position would be in a random place within a circle which is defined by the positions of alpha, beta, and delta in the search space. In other words, alpha, beta, and delta estimate the position of the prey, and other wolves updates their positions randomly around the prey.

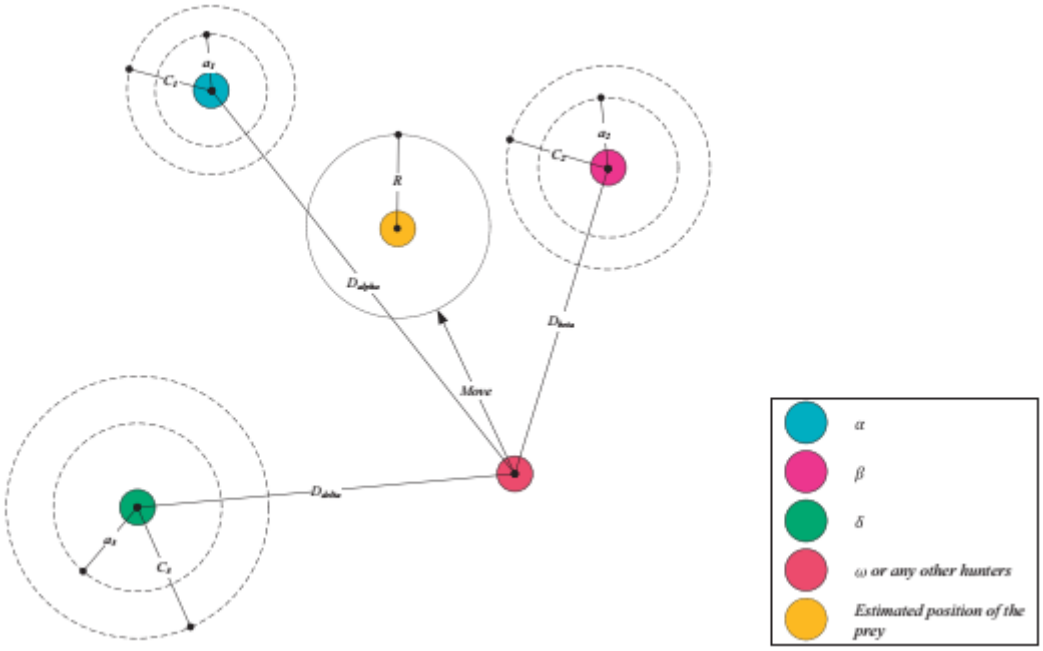


Figure IV.4. Position updading in GWO.

IV.2.2.4. Attacking prey (exploitation)

As mentioned above the grey wolves finish the hunt by attacking the prey when it stops moving. In order to mathematically model approaching the prey we decrease the value of $\vec{a}$. Note that the fluctuation range of $\vec{A}$ is also decreased by $\vec{a}$. In other words $\vec{A}$ is a random value in the interval $[-2\alpha, 2\alpha]$ where a is decreased from 2 to 0 over the course of iterations. When random values of $\vec{A}$ are in $[-1,1]$, the next position of a search agent can be in any position between its current position and the position of the prey. Figure IV.5(a) shows that $|A| < 1$ forces the wolves to attack towards the prey. With the operators proposed so far, the GWO algorithm allows its search agents to update their position based

on the location of the alpha, beta, and delta; and attack towards the prey. However, the GWO algorithm is prone to stagnation in local solutions with these operators. It is true that the encircling mechanism proposed shows exploration to some extent, but GWO needs more operators to emphasize exploration.

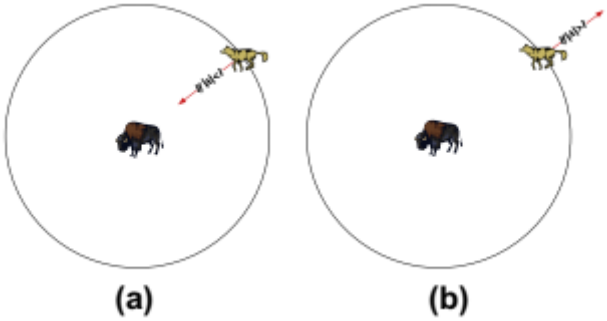


Figure IV.5. Attacking prey versus searching for prey.

IV.2.2.5. Search for prey (exploration)

Grey wolves mostly search according to the position of the alpha, beta, and delta. They diverge from each other to search for prey and converge to attack prey. In order to mathematically model divergence, we utilize $\vec{A}$ with random values greater than 1 or less than -1 to oblige the search agent to diverge from the prey. This emphasizes exploration and allows the GWO algorithm to search globally. Figure IV.5(b) also shows that $|A| > 1$ forces the grey wolves to diverge from the prey to hopefully find a fitter prey. Another component of GWO that favors exploration is $\vec{C}$. As may be seen in Eq. (IV.4), the $\vec{C}$ vector contains random values in $[0,2]$. This component provides random weights for prey in order to stochastically emphasize $(C > 1)$ or deemphasize $(C < 1)$ the effect of prey in defining the distance in Eq. (IV.1). This assists GWO to show a more random behavior throughout optimization, favoring exploration and local optima avoidance. It is worth mentioning here that C is not linearly decreased in contrast to A. We deliberately require C to provide random values at all times in order to emphasize exploration not only during initial iterations but also final iterations. This component is very helpful in case of local optima stagnation, especially in the final iterations. The C vector can be also considered as the effect of obstacles to approaching prey in nature. Generally speaking, the obstacles in nature appear in the hunting paths of wolves and in fact prevent them from quickly and conveniently approaching prey. This is exactly what the vector C does. Depending on the position of a wolf, it can randomly give the prey a weight and make it harder and farther to reach for wolves, or vice versa. To sum up, the search process starts with creating a random population of grey wolves (candidate solutions) in the

GWO algorithm. Over the course of iterations, alpha, beta, and delta wolves estimate the probable position of the prey. Each candidate solution updates its distance from the prey. The parameter a is decreased from 2 to 0 in order to emphasize exploration and exploitation, respectively. Candidate solutions tend to diverge from the prey when $|\vec{A}| > 1$ and converge towards the prey when $|\vec{A}| < 1$. Finally, the GWO algorithm is terminated by the satisfaction of an end criterion. The pseudo code of the GWO algorithm is presented in Figure IV.6. To see how GWO is theoretically able to solve optimization problems, some points may be noted:

- The proposed social hierarchy assists GWO to save the best solutions obtained so far over the course of iteration.
- The proposed encircling mechanism defines a circle-shaped neighborhood around the solutions which can be extended to higher dimensions as a hyper-sphere.
- The random parameters A and C assist candidate solutions to have hyper-spheres with different random radii.
- The proposed hunting method allows candidate solutions to locate the probable position of the prey.
- Exploration and exploitation are guaranteed by the adaptive values of a and A.
- The adaptive values of parameters a and A allow GWO to smoothly transition between exploration and exploitation.
- With decreasing A, half of the iterations are devoted to exploration $(|A| \gg 1)$ and the other half are dedicated to exploitation $(|A| < 1)$.
- The GWO has only two main parameters to be adjusted (a and C).

There are possibilities to integrate mutation and other evolutionary operators to mimic the whole life cycle of grey wolves.

```
Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
Xα=the best search agent
Xβ=the second best search agent
Xδ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update Xα, Xβ, and Xδ
    t=t+1
end while
return Xα
```

Figure IV.6. Pseudo code of the GWO algorithm.

IV.3. Grey Wolf Optimizer (GWO) algorithm-based UCAVs Path Planning

At the beginning, we present the formulation of the treated problem (UCAVs Path Planning), used in our work, which is presented in detail in the following (for more information, reader can see [65]).

IV.3.1. UCAVs Path Planning problem, other formulation

Path-planning of UCAVs is a numerical problem where emergencies should be considered. It aims to arrive at the destination safely by finding a superior path to avoid any threat, which accounts for artificial threats and natural constraints. The UCAV path-planning problem is usually represented by the classic 2D model, as shown in Figure IV.7. As depicted in this figure, we can clearly see that a path can be found to link up the starting point and the terminal end, and threats presented in circles are avoided. In the first, a segment *ST* that directly connected the starting and terminal point is drawn, then it is divided into $(D+1)$ equal parts by D perpendicular lines. After that, these D lines are taken as a new axis, onto which a series of points are set and connected one by one to form a path. Under this model, the whole path is divided into $D+1$ steps represented by a vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where x_{ij} is the jth dimension of the ith solution vector, and it indicates a discrete position of each step in the vertical axis.

The way of representing the path (a solution) is using a vector in the form of $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. For example, considering the coordinate $(0,0)$ is the starting point and $(T, 0)$ is the terminal point, then the distance between the starting and the terminal point is T. A path with D steps can be performed by that vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, thus the coordinate of the first step is $\left(\frac{T}{D}, x_{i1}\right)$, the coordinate of

the second step is $\left(\frac{T}{D} \times 2, x_{i2}\right)$, and so on. With all the steps linked up, the whole path can be formed. Since the paths cannot be away from the battle area, the restriction of each solution path on each case can be set manually related to the size of the battlefield.

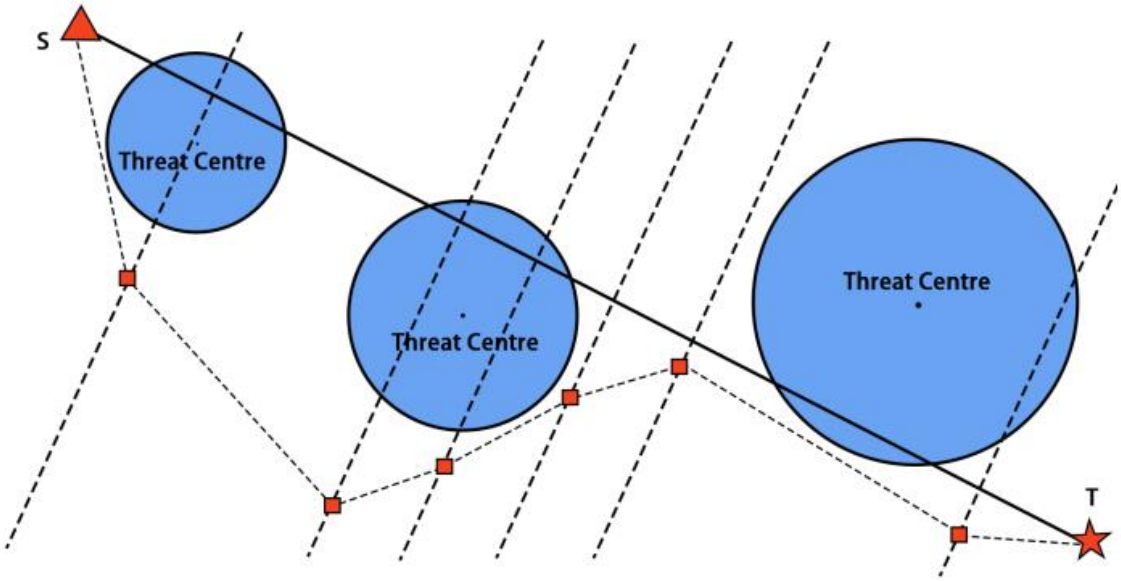


Figure IV.7. The schematic diagram for combat field modeling. The X and Y axles are the length and width of the horizontal battlefield. Note that all nodes are linked up to form a feasible path. During the flight, all threats are supposed to be avoid.

To measure those paths, a probability density model is formulated as Equation (IV.8). Unlike the traditional cost function model (described in section II.3 in the chapter 2), there is no distinct boundary where the damage risk remains zero. As distance increases, the probability of being attacked by a threat regularly decreases, but it's going to in no way be zero. $\|D_{kj}\|$ indicates the distance between the jth step and the kth threat center, δ is a parameter to control the shape of the density function:

$$Cost_{T1} = \sum_{k=1}^{n} exp\left(-\left(\sum_{j=1}^{D} \|D_{kj}\|\right)\delta\right) \quad \text{(IV.8)}$$

Indeed, the UCAV speed and the entire distance of a path should be taken into consideration. Simplify, it is assumed that the UCAV maintains a constant speed, and then the fuel consumption corresponds with the total distance of the path. Under that, the complete cost function can be described as:

$$Cost = \gamma \times \sum_{k=1}^{n} exp\left(-\left(\sum_{j=1}^{D} \|D_{kj}\|\right)\delta\right) + (1-\gamma) \times \frac{\sum_{j=1}^{D} d_{ij}}{\|ST\|} \quad \text{(IV.9)}$$

where γ is the weighting factor ranging in $[0,1]$, and γ is set to 0.5. $\|ST\|$ is the length of the segment ST. $\sum_{j=1}^{D} d_{ij}$ refers to the length of the whole flight path, and d_{ij} is the length of the jth step of the ith solution.

IV.3.2. Experimental results and discussion

In order to investigate the feasibility and effectiveness of the proposed method (GWO algorithm-based UCAVs Path Planning problem) in this work, an experiment is conducted with a simple comparison with the results obtained using a PSO algorithm.

The GWO algorithm parameters used in simulation test are adapted as indicted in Table IV.1.

Parameter	Value
Number of search agents	30
Max number of iterations	10000/1000

Table IV.1. The GWO algorithm parameters used in simulation.

In order to showcase the efficacy and robustness of our newly introduced approach, SMOBHC, we performed tests on three different battlefields. Each of these fields contained a randomly distributed set of threats, ten in one, thirty in the second, and twenty in the third, under certain predefined conditions. These tests were conducted on a PC equipped with an Intel(R) Core (TM) i7-4510U CPU operating at 2.60 GHz, a 6 GB RAM, and a 64-bit Windows 10 operating system. The algorithms utilized in these tests were compiled with MATLAB 2022b.

In our inaugural experiment, we employed two battlefields that specifically had ten and twenty threats, respectively. These threats were visualized as circles with varying positions (threat centers) and radii, as indicated in Figures IV.8 and IV.9. It's important to mention that the ideal path (optimal path) should circumvent these threats and reach the terminal safely. Moreover, according to Figures IV.8 and IV.9, the starting point and the end point coordinates were set to be (0,0) and (0.500) respectively.

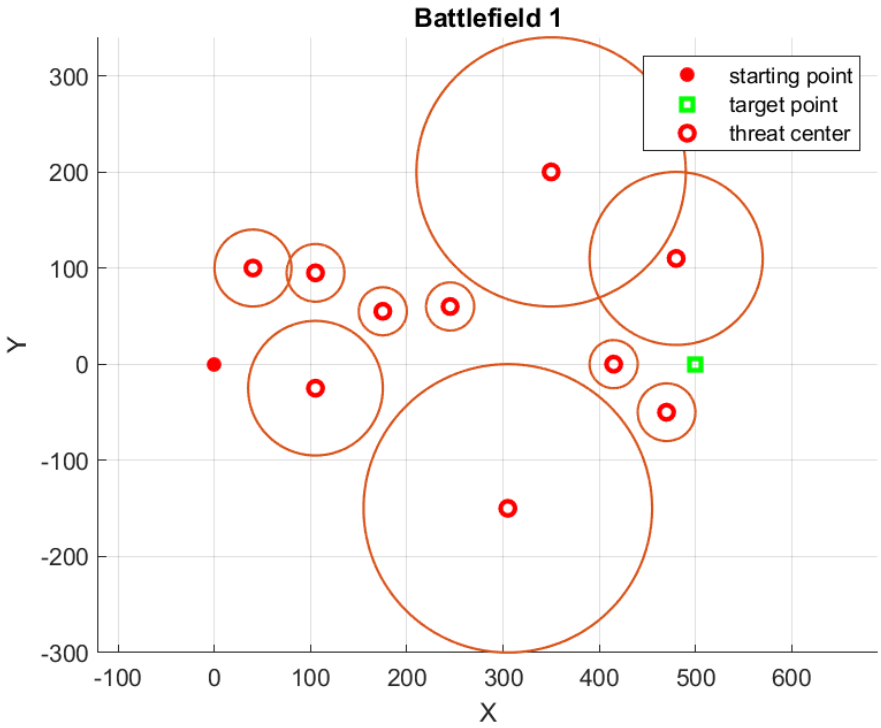


Figure IV.8. First battlefield with 10 threats, starting point (red) and target point (green).

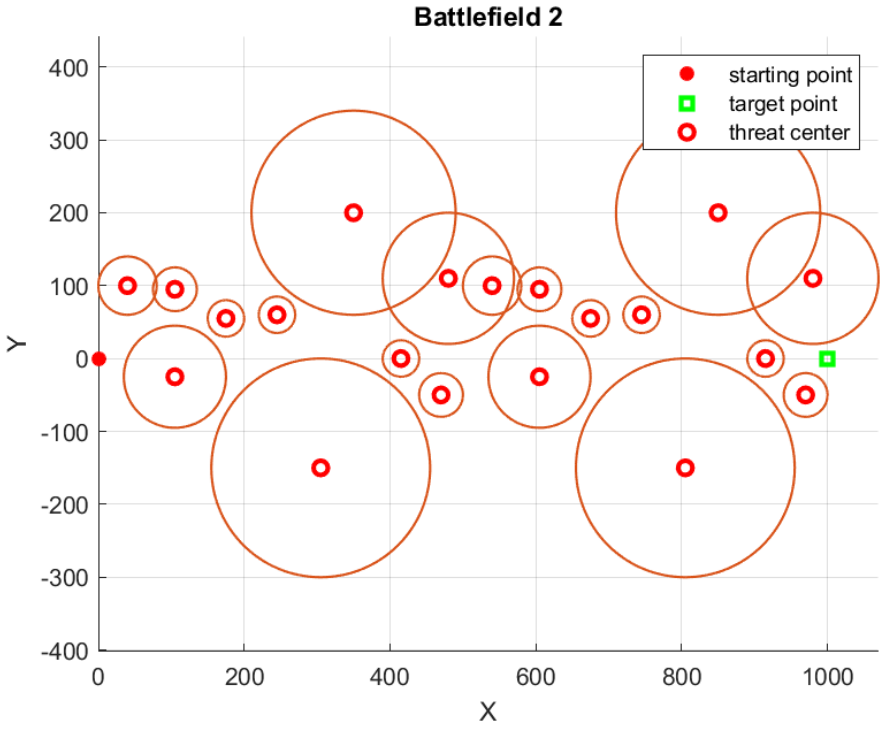


Figure IV.9. Second battlefield with 10 threats, starting point (red) and target point (green).

To validate the efficacy of the GWO algorithm for this particular problem, we contrasted its achieved outcomes with those produced by a conventional PSO algorithm. Indeed, all the results from our

simulations are comprehensively presented in Figures IV.10, IV.11, IV.12, IV.13, IV.14, IV.15 and Table IV.3. All PSO control parameters used in simulation tests are illustrated in Table IV.2 indicted below.

Parameter	Value
Swarm size	40 particles
Learning factor c_1 and c_2	1.4962 (c_1 and c_2)
Inertia weight w	0.7298
$maxGeneration$	10000/1000

Table IV.2. PSO control parameters used in simulation.

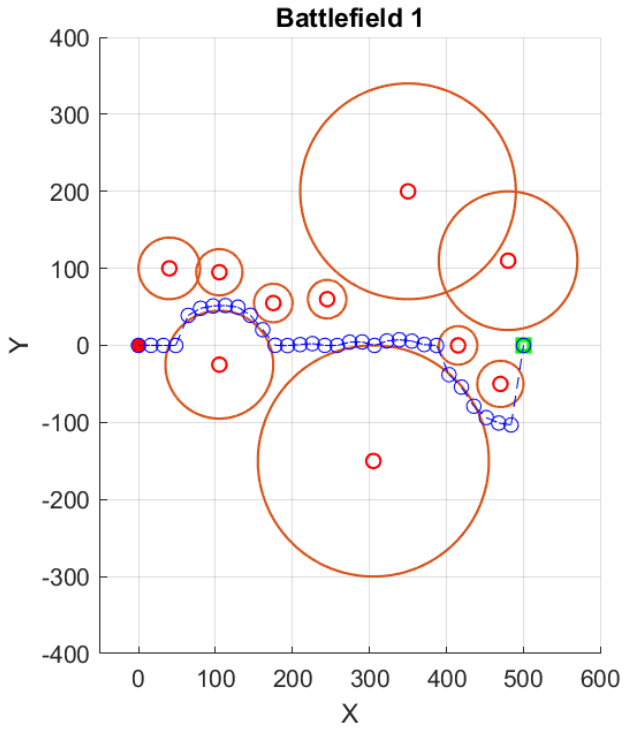


Figure IV. 10. Optimal path obtained by the GWO algorithm (drawn in bleu) for the first battlefield.

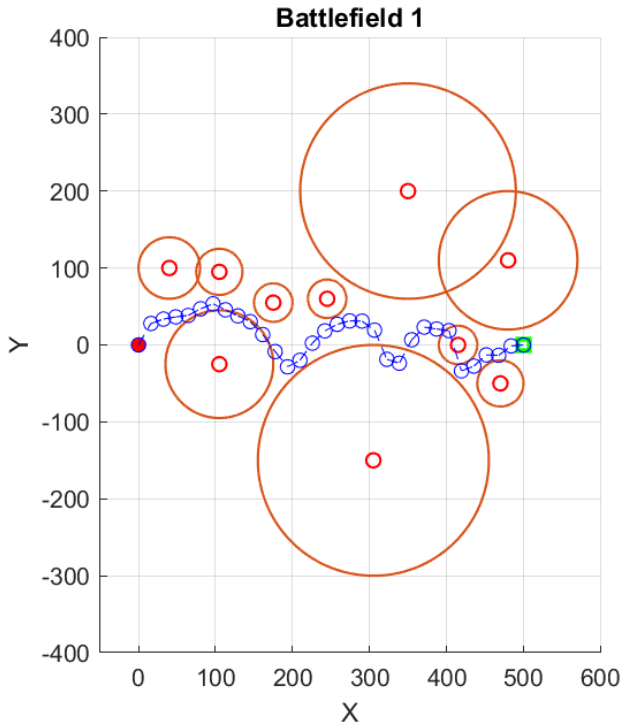


Figure IV.11. Optimal path obtained by the PSO algorithm (drawn in bleu) for the first battlefield.

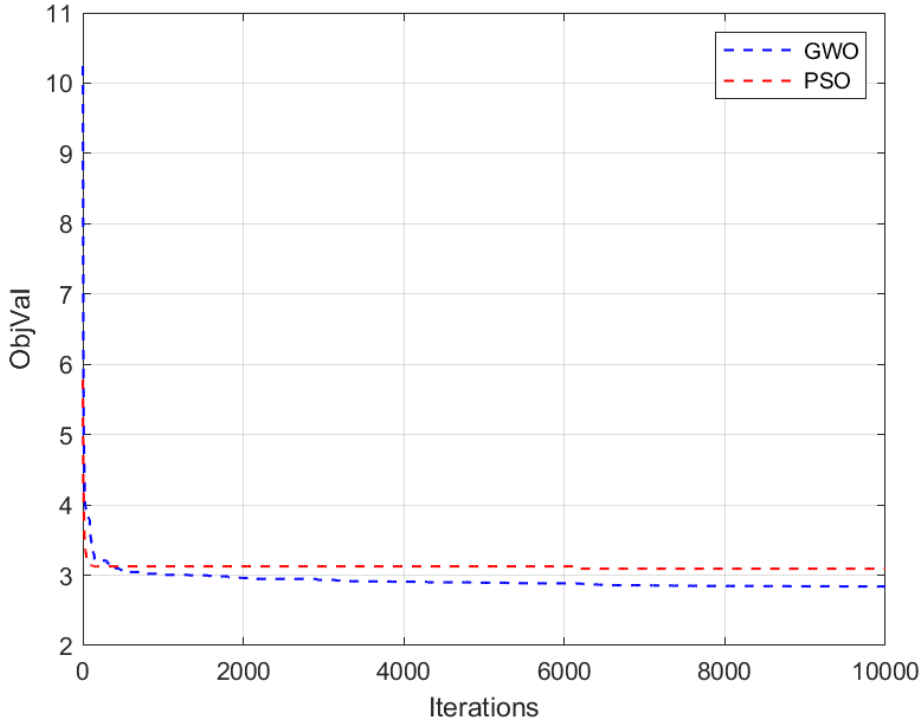


Figure IV.12. Comparative convergence curves of GWO and PSO algorithms for the first battlefield.

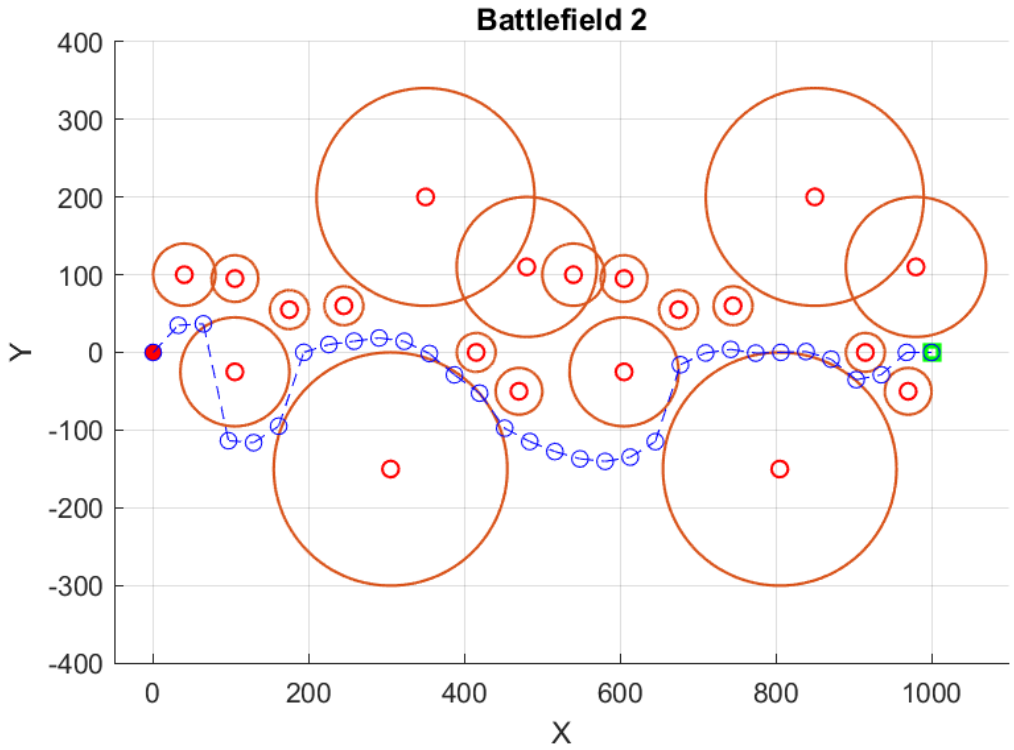


Figure IV.13. Optimal path obtained by the GWO algorithm (drawn in bleu) for the second battlefield.

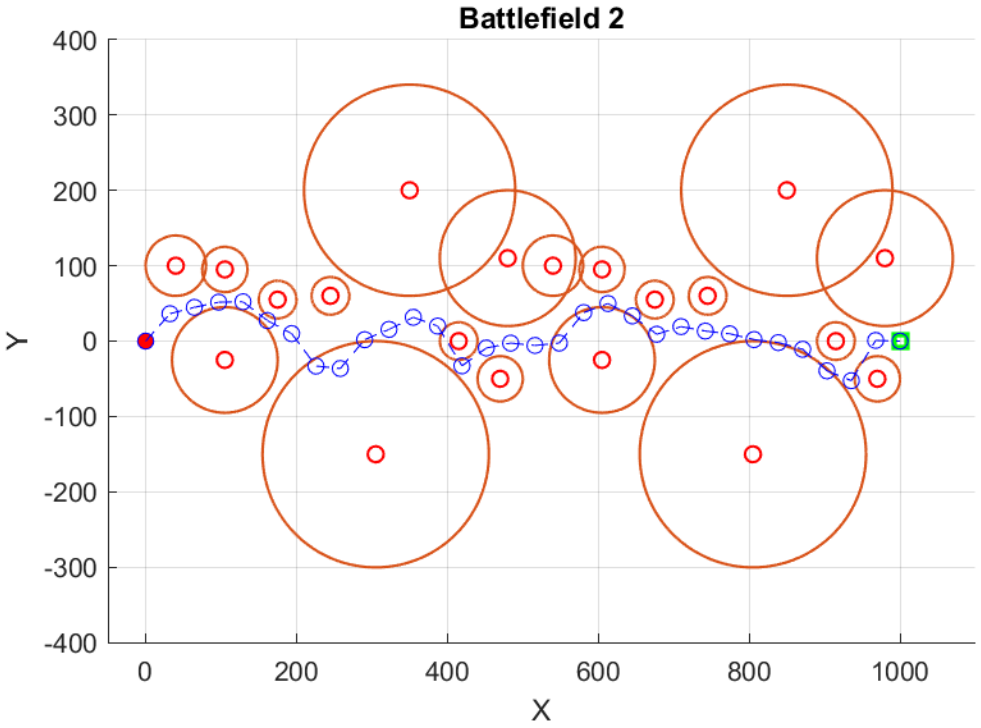


Figure IV.14. Optimal path obtained by the PSO algorithm (drawn in bleu) for the second battlefield.

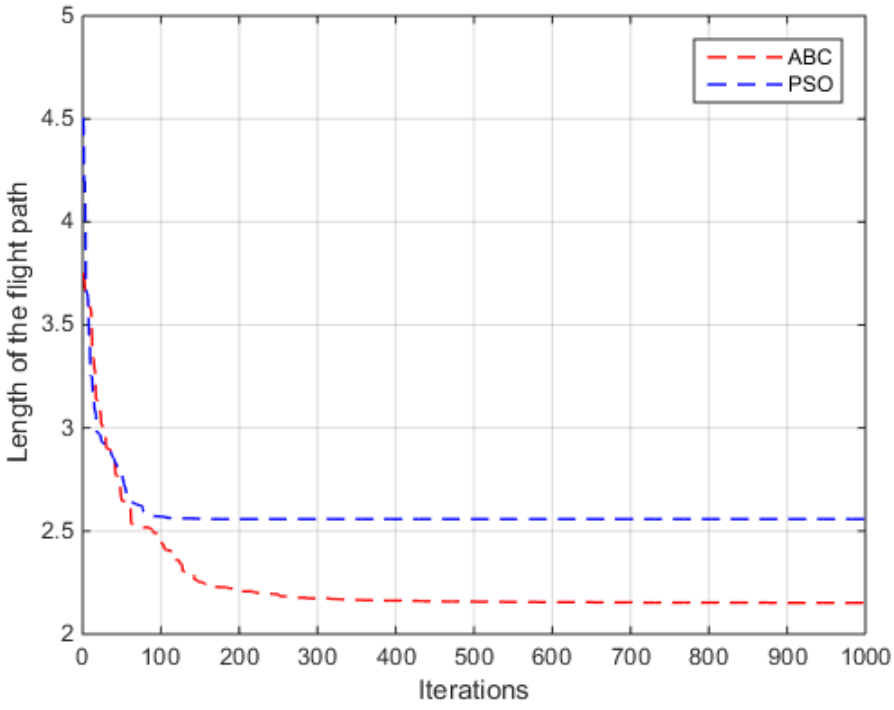


Figure IV.15. Comparative convergence curves of GWO and PSO algorithms for the second battlefield.

Algorithm	Optimal Obtained path cost	
	Battlefield 1	Battlefield 2
GWO	**2.8404**	**2.8254**
PSO	3.0918	2.9951

Table IV.3. Optimal path cost achieved using GWO and PSO algorithms across both battlefields.

Our analysis of the acquired results indicates a conspicuous observation: the Grey Wolf Optimization (GWO) algorithm demonstrates a significant capacity to address problems of this nature with satisfactory precision. This precision is ascertained by the value of the optimally derived flight path length.

The GWO algorithm's efficacy isn't limited merely to its problem-solving ability. In fact, it exhibits an advantageous superiority in comparison to the Particle Swarm Optimization (PSO) algorithm. This superiority of the GWO was not confined to one, but both simulation tests conducted as a part of our study, highlighting its consistency and reinforcing its robustness in solving such problems.

Furthermore, based on the visual data presented in Figure IV.11 and Figure IV.14, a certain level of acceptance can be conferred upon the convergence rate of the GWO when juxtaposed against the PSO. The comparative study of the figures underlines the superior adaptability and performance of the GWO.

The rate at which it converges towards the optimal solution effectively accentuates its efficiency, further substantiating its supremacy over the PSO.

In conclusion, the analysis of the acquired data, along with the supporting visual representations, strongly attests to the effective application of the GWO algorithm in problem-solving scenarios of this kind. Its satisfactory precision, consistent superiority in simulations, and acceptable convergence rate all demonstrate its effectiveness in comparison to the PSO algorithm.

IV.4. Conclusion

In the course of this elaborate chapter, we focus on the application of the Grey Wolf Optimization (GWO) algorithm to resolve a significant problem in the world of Unmanned Combat Aerial Vehicles (UCAVs), that is, to discover the most suitable, or optimal, flight path. The chapter commences with a thorough introduction and elucidation of the GWO algorithm, providing a broad overview of its fundamental principles and operational dynamics. This lucid discourse lays the groundwork for understanding the complex mechanisms underpinning the algorithm. Following this extensive theoretical exploration, the topic of discussion transitions into the specifics of UCAVs path planning problems. In this section, a fresh formulation of the problem is introduced. This new perspective stands apart from, and is wholly unlike, the formulation detailed in Chapter II, offering a unique approach to tackle the same problem.

Further deepening our exploration, the GWO algorithm is applied to the problem defined above within the contexts of two distinct battlefields. This dual-application allows for a robust comparison and a thorough understanding of the algorithm's effectiveness under varying conditions. The results procured from this rigorous application unequivocally underscore the efficacy of this approach, aptly named the GWO-based UCAVs Path Planning. Both the accuracy of the solutions and the rate of convergence towards them were notable, further emphasizing the utility of this method.

In order to add another layer of depth to our experiment, we also incorporate the use of an alternative population-based metaheuristic algorithm known as Particle Swarm Optimization (PSO). However, in both test scenarios performed, the GWO algorithm consistently outperforms PSO, further bolstering our confidence in the GWO's superior capabilities.

Conclusions

Conclusions

In this Master thesis, we have employed the Grey Wolf Optimization (GWO) algorithm, a form of swarm intelligence algorithm, to determine an optimal path for the two-dimensional path planning problem associated with Unmanned Combat Aerial Vehicles (UCAVs) in challenging combat environments. The impetus for this research stems from the satisfactory balance that the GWO algorithm provides between exploration and exploitation capabilities.

The chosen algorithm was subjected to two test cases, with its outcomes compared to those obtained through another stochastic optimization algorithm known as Particle Swarm Optimization (PSO). This comparative analysis was conducted to validate the results obtained from the GWO. The findings highlighted the efficacy of the GWO in addressing the UCAV path planning problem.

One notable advantage of the GWO algorithm is its competent ability to avoid local optima, thus improving the likelihood of discovering viable approximations of the optimal weighted sum cost for this path. Moreover, the accuracy of the acquired optimal values for the weighted sum cost was found to be highly satisfactory, attributable to the effective exploitation capabilities of the GWO algorithm.

Despite the exhibited effectiveness of the GWO algorithm in several aspects such as accuracy, convergence rate, and avoidance of local optima, it remains necessary to juxtapose it with more than one method and, if feasible, implement it on an actual prototype.

Our future work will endeavor to address these two aspects. On one hand, we plan to apply the GWO algorithm to solve the UCAV path planning problem in three-dimensional space. On the other hand, we intend to experiment with other contemporary algorithms such as the Whale Optimization Algorithm [66], the Ant Lion Optimizer [67], the Virus Colony Search Algorithm [68], among others.

References

[1] Kabamba, P.T.; Meerkov, S.M.; Zeitz, F.H., III. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking. *J. Guid. Control Dyn.* 2006, *29*, 279–288.

[2] Sud, A.; Andersen, E.; Curtis, S.; Lin, M.; Manocha, D. Real-time path planning for virtual agents in dynamic environments. In Proceedings of the 2007 IEEE Virtual Reality Conference, Charlotte, NC, USA, 10–14 March 2007; pp. 91–98.

[3] Xin, H.; Chen, Q.; Wang, Y.; Jia, G.; Hou, Z. An Optimal Path Planning Method for UCAV in Terminal of Target Strike. In Proceedings of the 2019 Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 3672–3676.

[4] You, S.; Gao, L.; Diao, M. Real-time path planning based on the situation space of UCAVS in a dynamic environment. *Microgravity Sci. Technol.* 2018, *30*, 899–910.

[5] Chen, H.X.; Nan, Y.; Yang, Y. A two-stage method for UCAV TF/TA path planning based on approximate dynamic programming. *Math. Probl. Eng.* 2018, *2018*, 1092092.

[6] Wei, Z.; Huang, C.; Han, T.; Dong, K.; Li, Y. UCAVs online collaborative path planning method based on dynamic task allocation. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 872–877.

[7] Pehlivanoglu, Y.V. A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. *Aerosp. Sci. Technol.* 2012, *16*, 47–55.

[8] Zhang, S.; Zhou, Y.; Li, Z.; Pan, W. Grey wolf optimizer for unmanned combat aerial vehicle path planning. *Adv. Eng. Softw.* 2016, *99*, 121–136.

[9] Wang, G.G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* 2016, *49*, 231–238.

[10] Yi, J.H.; Lu, M.; Zhao, X.J. Quantum inspired monarch butterfly optimisation for UCAV path planning navigation problem. *Int. J. Bio-Inspired Comput.* 2020, *15*, 75–89.

[11] Pan, J.S.; Liu, N.; Chu, S.C. A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning. *IEEE Access* 2020, *8*, 17691–17712.

[12] Huang, H.; Zhuo, T. Multi-model cooperative task assignment and path planning of multiple UCAV formation. *Multimed. Tools Appl.* 2019, *78*, 415–436.

[13] Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Three dimensional path planning using Grey wolf optimizer for UAVs. *Appl. Intell.* 2019, *49*, 2201–2217.

[14] Paszkiel, S.; Sikora, M. The Use of Brain-Computer Interface to Control Unmanned Aerial Vehicle. In Proceedings of the Conference on Automation, Warsaw, Poland, 27–29 March 2019; pp. 583–598.

[15] Parpinelli, R.S.; Lopes, H.S. New inspirations in swarm intelligence: A survey. *Int. J. Bio-Inspired Comput.* 2011, *3*, 1–16.

[16] Yang, X.S.; Cui, Z.; Xiao, R.; Gandomi, A.H.; Karamanoglu, M. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*; Elsevier: Waltham, MA, USA, 2013.

[17] Blum, C.; Li, X. Swarm intelligence in optimization. In *Swarm Intelligence*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 43–85.

[18] Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. In *Nature-Inspired Computing and Optimization*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 475–494.

[19] Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* 1997, *1*, 67–82.

[20] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer. Advances in Engineering Software, 2014, 69, 46–61.

[21] Dalamagkidis K. "Classification of UAVs". In: Valavanis K. Vachtsevanos G. (eds) Handbook of Unmanned Aerial Vehicles, Springer Dordrecht, 2014, pp. 83–91.

[22] Garcia Carrillo L.R. et al. *Quad Rotorcraft Control. Vision-Based Hovering and Navigation*. Springer-Verlag London, 2013. isbn: 1430-9491.

[23] Goran Tmusic et al. "Current Practices in UAS-based Environmental Monitoring". In: *Remote Sensing* 12 (Mar. 2020), p. 1001. doi: 10.3390/rs12061001.

[24] Asma Troudi et al. "Sizing of the Drone Delivery Fleet Considering Energy Autonomy". In: *Sustainability* 10 (Sept. 2018), p. 3344. doi: 10.3390/su10093344.

[25] Anne Goodchild and Jordan Toy. "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing *CO*2 emissions in the delivery service industry". In: *Transportation Research Part D: Transport and Environment* (Mar. 2017). doi: 10.1016/j.trd.2017.02.017.

[26] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A bat algorithm with mutation for UCAV path planning, Sci. World J. 2012 (2012).

[27] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A modified firefly algorithm for ucav path planning, International Journal of Hybrid Information Technology 5 (3) (2012) 123{144.

[28] Li B, Gong G L, Yang W L (2014) An improved Artificial Bee Colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning. The Scientific World

Journal, vol. 2014, Article ID 232704, 10 pages
[29] Sen Zhang , Yongquan Zhou , Zhiming Li , Wei Pan.(2016), “Grey wolf optimizer for unmanned combat aerial vehicle path planning”, Advances in Engineering Software, Volume 99, September, Pages 121–136.
[30] Xu C, Duan H, Liu F (2010) Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning. Aerosp Sci Technol. doi:10.1016/j.ast.2010.04.008
[31] W. Zhu, H.B. Duan, Chaotic predator–prey biogeography-based optimization approach for UCAV path planning, Aerosp. Sci. Technol. 32 (1) (2014) 153–161.
[32] Hussain, K., Salleh, M. N. M., Cheng, S., and Shi, Y. (2018). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, pages 1{43.
[33] Birattari, M., Paquete, L., Strutzle, T., and Varrentrapp, K. (2001). Classification of metaheuristics and design of experiments for the analysis of components
[34] Boussa¨ıD, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information sciences*, 237:82{117.
[35] Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

[36] H´eliodore, F., Nakib, A., Ismail, B., Ouchraa, S., and Schmitt, L. (2017). *Metaheuristics for intelligent electrical networks*, chapter Performance Evaluation of Metaheuristics. Wiley Online Library
[37] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671{680.
[38] Cern`y, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. ˇ *Journal of Optimization Theory and Applications*, 45(1):41{51.
[39] Chibante, R. (2010). *Simulated Annealing: Theory with Applications*. BoD{Books on Demand.

[40] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533{549.
[41] Cvijovi´c, D. and Klinowski, J. (1995). Taboo search: an approach to the multiple minima problem. *Science*, 267(5198):664{666.
[42] Toth, P. and Vigo, D. (2003). The granular Tabu search and its application to the vehicle routing problem. *Informs Journal on Computing*, 15(4):333{346.
[43] Battiti, R. and Tecchiolli, G. (1996). The continuous reactive Tabu search: blending combinatorial optimization and stochastic search for global optimization. *Annals of Operations Research*, 63(2):151{188.
[44] Jaeggi, D. M., Parks, G. T., Kipouros, T., and Clarkson, P. J. (2008). The development of a multi-objective Tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, 185(3):1192{1212.
[45] Løkketangen, A. and Woodruff, D. L. (1996). Progressive hedging and Tabu search applied to mixed integer (0, 1) multistage stochastic programming. *Journal of Heuristics*, 2(2):111{128.
[46] Gendreau, M., Guertin, F., Potvin, J.-Y., and S´eguin, R. (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157{174.
[47] Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.
[48] Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268{308.
[49] Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245{1287.
[50] Freitas, A. A. (2003). A survey of evolutionary algorithms for data mining and knowledge discovery. In *Advances in Evolutionary Computing*, pages 819{845. Springer.
[51] Bansal, J. C., Singh, P. K., and Pal, N. R. (2019). *Evolutionary and swarm intelligence algorithms*. Springer.
[52] Bonabeau, E., Marco, D. d. R. D. F., Dorigo, M., Theraulaz, G., et al. (1999). *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford University Press.
[53] Drigo, M. (1996). The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):1{13.
[54] Mohan, B. C. and Baskaran, R. (2012). A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, 39(4):4618{4627.
[55] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks IV*, volume 1000, pages 1942{1948.
[56] Yang, X.-S. and Deb, S. (2009). Cuckoo search via l´evy flights. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 210{214. IEEE.
[57] Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67{82.
[58] Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In *International Workshop on Hybrid Metaheuristics*, pages 1{12. Springer.
[59] Ting, T., Yang, X.-S., Cheng, S., and Huang, K. (2015). Hybrid metaheuristic algorithms: past, present, and

future. In *Recent Advances in Swarm Intelligence and Evolutionary Computation*, pages 71{83. Springer.
[60] Blum, C., Roli, A., and Sampels, M. (2008). *Hybrid metaheuristics: an emerging approach to optimization*, volume 114. Springer.
[61] Katsigiannis, Y. A., Georgilakis, P. S., and Karapidakis, E. S. (2012). Hybrid simulated annealing{ Tabu search method for optimal sizing of autonomous power systems with renewables. *IEEE Transactions on Sustainable Energy*, 3(3):330{338.
[62] Behnamian, J., Ghomi, S. F., and Zandieh, M. (2009). A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications*, 36(8):11057{11069.
[63] Shankar, T., Shanmugavel, S., and Rajesh, A. (2016). Hybrid HSA and PSO algorithm for energy efficient cluster head selection in wireless sensor networks. *Swarm and Evolutionary Computation*, 30:1{10.
[64] Garc´ıa, M. L. L., Garc´ıa-R´odenas, R., and G´omez, A. G. (2014). Hybrid metaheuristic optimization algorithms for time-domain-constrained data clustering. *Applied Soft Computing*, 23:319{332.
[65] B. Li, L. G. Gong, and C. H. Zhao, "Unmanned combat aerial vehicles path planning using a novel probability density model based on Artificial Bee Colony algorithm," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP '13)*, pp. 620–625, Beijing, China, June 2013.
[66] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Software 2016;95:51–67.
[67] Mirjalili S. The ant lion optimizer. Adv Eng Softw 2015;83:80–98.
[68] M. D. Li, H. Zhao, X. W. Weng, T. Han, A novel nature-inspired algorithm for optimization: Virus colony search, Advances in Engineering Software 92 (2016). 65-88.