Coordination of scout drones (UAVs) in smart-city to serve autonomous vehicles

Amylia Ait Saadi

THESE DE DOCTORAT

NNT : 2023UPASG064

Coordination of scout drones (UAVs) in smart-city to serve autonomous vehicles

Coordination de drones éclaireurs (UAV) dans la smart-city pour servir les véhicules autonomes

Thèse de doctorat de l'université Paris-Saclay et de l'université M'Hamed Bougara de Boumerdes

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : informatique
Graduate School : Informatique et sciences du numérique. Référent : Université de Versailles Saint-Quentin-en-Yvelines

Thèse préparée dans les unités de recherche **LISV (Université Paris-Saclay, UVSQ) et LIST (Université M'Hamed Bougara de Boumerdes)**, sous la direction de **Amar RAMDANE-CHERIF**, Professeur des universités, la co-direction de **Yassine MERAIHI**, Maître de conférence-HDR, la co-direction de **Assia SOUKANE**, Directrice de recherche.

Thèse soutenue à Boumerdes, le 30 octobre 2023, par

Amylia AIT SAADI

Composition du Jury

Membres du jury avec voix délibérative

Dalila ACHELI Professeure des universités, Université M'Hamed Bougara de Boumerdes - Lab LAA	Présidente	
Walid Khaled HIDOUCI Professeur des universités, Ecole Supérieur d'Informatique – LAB LCSI	Rapporteur & Examinateur	
Marie-Ange MANIER Professeure des universités, Université de technologie de Belfort Montbéliard - LAB FEMTO-ST	Rapportrice & Examinatrice	
Dana MARINCA MCF, Université Paris Saclay - Lab DAVID	Examinatrice	

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Coordination de drones éclaireurs (UAV) dans la smart-city pour servir les véhicules autonomes

Mots clés : Véhicule aérien sans pilote (VAP), problème de planification de trajectoire d'UAV, méta-heuristique, problème de placement d'UAV.

Résumé : Le sujet des véhicules aériens sans pilote (VAP) est devenu un domaine d'étude prometteur tant dans la recherche que dans l'industrie. En raison de leur autonomie et de leur efficacité en vol, les drones sont considérablement utilisés dans diverses applications pour différentes tâches. Actuellement, l'autonomie du drone est un problème difficile qui peut avoir un impact à la fois sur ses performances et sur sa sécurité pendant la mission. Pendant le vol, les drones autonomes sont tenus d'investiguer la zone et de déterminer efficacement leur trajectoire en préservant leurs ressources (énergie liée à la fois à l'altitude et à la longueur de la trajectoire) et en satisfaisant certaines contraintes (obstacles et rotations d'axe). Ce problème est défini comme le problème de planification de trajectoire UAV qui nécessite des algorithmes efficaces pour être résolus, souvent des algorithmes d'intelligence artificielle.

Dans cette thèse, nous présentons deux nouvelles approches pour résoudre le problème de planification de trajectoire UAV. La première approche est un algorithme amélioré basé sur l'algorithme d'optimisation des vautours africains, appelé algorithmes CCO-AVOA, qui intègre la carte chaotique, la mutation de Cauchy et les stratégies d'apprentissage basées sur l'opposition d'élite. Ces trois stratégies améliorent les performances de l'algorithme AVOA original en termes de diversité de solutions et d'équilibre de recherche exploration/exploitation.

Une deuxième approche est une approche hybride, appelée CAOSA, basée sur l'hybridation de Chaotic Aquila Optimization avec des algorithmes de recuit simulé. L'introduction de la carte chaotique améliore la diversité de l'optimisation Aquila (AO), tandis que l'algorithme de recuit simulé (SA) est appliqué comme algorithme de recherche locale pour améliorer la recherche d'exploitation de l'algorithme AO traditionnel. Enfin, l'autonomie et l'efficacité du drone sont abordées dans une autre application importante, qui est le problème de placement du drone. La question du placement de l'UAV repose sur la recherche de l'emplacement optimal du drone qui satisfait à la fois la couverture du réseau et la connectivité tout en tenant compte de la limitation de l'UAV en termes d'énergie et de charge. Dans ce contexte, nous avons proposé un algorithme hybride et efficace appelé IMRFO-TS, basé sur la combinaison de l'algorithme d'amélioration de l'optimisation de la recherche de nourriture des raies manta, qui intègre une stratégie de contrôle tangentiel et d'algorithme de recherche taboue.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : Coordination of scout drones (UAVs) in smart-city to serve autonomous vehicles

Keywords : Unmanned Aerial Vehicle (UAV), the UAV path planning problem, meta-heuristics, the UAV placement problem.

Abstract : The subject of Unmanned Aerial Vehicles (UAVs) has become a promising study field in both research and industry. Due to their autonomy and efficiency in flight, UAVs are considerably used in various applications for different tasks. Actually, the autonomy of the UAV is a challenging issue that can impact both its performance and safety during the mission. During the flight, the autonomous UAVs are required to investigate the area and determine efficiently their trajectory by preserving their resources (energy related to both altitude and path length) and satisfying some constraints (obstacles and axe rotations). This problem is defined as the UAV path planning problem that requires efficient algorithms to be solved, often Artificial Intelligence algorithms.

In this thesis, we present two novel approaches for solving the UAV path planning problem. The first approach is an improved algorithm based on African Vultures Optimization Algorithm (AVOA), called CCO-AVOA algorithms, which integrates the Chaotic map, Cauchy mutation, and Elite Opposition-based learning strategies. These three strategies improve the performance of the original AVOA algorithm in terms of the diversity of solutions and the exploration/exploitation search balance.

A second approach is a hybrid-based approach, called CAOSA, based on the hybridization of Chaotic Aquila Optimization with Simulated Annealing algorithms. The introduction of the chaotic map enhances the diversity of the Aquila Optimization (AO), while the Simulated Annealing (SA) algorithm is applied as a local search algorithm to improve the exploitation search of the traditional AO algorithm. Finally, the autonomy and efficiency of the UAV are tackled in another important application, which is the UAV placement problem. The issue of the UAV placement relays on finding the optimal UAV placement that satisfies both the network coverage and connectivity while considering the UAV's limitation from energy and load. In this context, we proposed an efficient hybrid called IMRFO-TS, based on the combination of Improved Manta Ray Foraging Optimization, which integrates a tangential control strategy and Tabu Search algorithms.

Résumé :
Le sujet des véhicules aériens sans pilote (VAP) est devenu un domaine d'étude prometteur tant dans la recherche que dans l'industrie. En raison de leur autonomie et de leur efficacité en vol, les drones sont considérablement utilisés dans diverses applications pour différentes tâches. Actuellement, l'autonomie du drone est un problème difficile qui peut avoir un impact à la fois sur ses performances et sur sa sécurité pendant la mission. Pendant le vol, les drones autonomes sont tenus d'investiguer la zone et de déterminer efficacement leur trajectoire en préservant leurs ressources (énergie liée à la fois à l'altitude et à la longueur de la trajectoire) et en satisfaisant certaines contraintes (obstacles et rotations d'axe). Ce problème est défini comme le problème de planification de trajectoire UAV qui nécessite des algorithmes efficaces pour être résolus, souvent des algorithmes d'intelligence artificielle.
Dans cette thèse, et dans ce domaine, nous présentons deux nouvelles approches. La première consiste à proposer une approche hybride, appelée CAOSA, qui est basée sur l'hybridation de l'algorithme chaotique Aquila avec des algorithmes de recuit simulé pour résoudre le problème de la planification de la trajectoire. L'objectif est de déterminer le chemin optimal en considérant la longueur du chemin afin d'optimiser la contrainte énergétique et les contraintes liées aux obstacles, altitude et rotations du drones. L'introduction de la carte chaotique améliore la diversité de l'optimisation Aquila (AO), tandis que l'algorithme de recuit simulé (SA) est appliqué comme algorithme de recherche locale pour améliorer la recherche d'exploitation de l'algorithme AO traditionnel. L'algorithme CAOSA a été testé dans trois (3) scénarios en comparaison à plusieurs algorithmes d'optimisation tels que : optimisation des essaims de particules, algorithme de chauve-souris, algorithme de luciole, algorithme de libellule, optimisation du loup gris, l'algorithme du recuit simulé, algorithme sinus-cosinus, algorithme d'optimisation de baleine, et optimisation d'Aquila. Les résultats de simulation ont démontré la supériorité de notre algorithme, dans la plupart des cas, en termes de temps d'exécution et longueur du chemin.

La deuxième approche est une amélioration de l'algorithme méta-heuristique d'optimisation du vautour africain (AVOA), nommé AVOA basé sur l'opposition chaotique de Cauchy (CCO-AVOA), pour résoudre le problème de planification de trajectoire des drones dans un environnement 3D. Dans cette contribution, l'objectif est de minimiser la consommation énergétique du drone en minimisant la longueur du chemin et des déplacements en altitude, tout en considérant les contraintes des obstacles et rotations des drones. L'efficacité du CCO-AVOA proposé est validée dans différents environnements avec différents nombres de points de cheminement et de menaces en tenant compte prennent en compte la valeur de fitness, le coût du trajet, le coût de la hauteur, le coût des obstacles, le coût de l'angle de l'UAV et les mesures de temps d'exécution. Comparés à dix méta-heuristiques bien connues, les résultats de simulation démontrent l'efficacité de l'approche CCO-AVOA proposée dans la plupart des cas en obtenant un chemin court, fluide, le moins coûteux et sans collision avec une meilleure stabilité pour les drones dans des environnements complexes.

Le concept de ville intelligente consiste à améliorer la qualité de vie des résidents et à fournir des services efficaces en intégrant des technologies avancées de l'information et de la communication, des robots autonomes, des appareils Internet des objets (IoT), etc. Les véhicules aériens sans pilote (UAV) sont une classe de robots mobiles volants qui apportent

de nombreux avantages aux villes intelligentes en raison de leur mobilité, de leur accessibilité, de leur autonomie et bien d'autres avantages. Leur intégration permet d'accomplir des tâches difficiles et complexes que les humains ou d'autres entités ne sont pas en mesure d'accomplir. Dans la plupart des applications, plusieurs drones connectés sont nécessaires pour créer un réseau dans lequel les missions sont accomplies et les tâches sont partagées. Ce réseau peut être établi en trouvant le placement optimal du drone qui répond à certaines exigences telles que la couverture des utilisateurs, la connectivité du drone, l'énergie et la répartition de la charge. Dans ce contexte, nous présentons un algorithme hybride, appelé IMRFO-TS, basé sur l'hybridation de l'optimisation améliorée de la recherche de nourriture des raies manta (IMRFO) avec l'algorithme recherche taboue (TS) pour résoudre le problème de placement des drones dans une ville intelligente. Premièrement, la stratégie de contrôle tangentiel est intégrée à l'algorithme MRFO original pour améliorer sa vitesse de convergence et explorer efficacement l'espace de recherche. Deuxièmement, l'algorithme TS est hybridé avec l'algorithme IMRFO pour augmenter la capacité d'exploitation d'IMRFO et améliorer la meilleure solution (qualité de placement) obtenue après chaque itération. Les performances de l'algorithme IMRFO-TS proposé sont validées à l'aide de 52 benchmarks prenant en compte la valeur de fitness, la couverture, la connectivité, consommation d'énergie et paramètres de répartition de la charge. Comparé à huit métaheuristiques d'optimisation bien connues telles que le MRFO original, l'algorithme recherche taboue, algorithme de luciole, l'algorithme des chauves-souris, l'optimisation du loup gris, l'algorithme sinus cosinus, l'algorithme d'optimisation baleine, et algorithme de recherche de reptiles, les résultats des expériences ont révélé la supériorité significative de l'algorithme IMRFO-TS proposé en obtenant des solutions prometteuses (positions optimales des drones) dans la majorité des cas.

ACKNOWLEDGEMENTS

During my university career, I had the great honour of meeting illustrious intellectual personalities and the pleasure of appreciating their contribution to science, their dedication to work and their socio-professional qualities. What I experienced in our university corresponds perfectly with my expectations both in terms of teaching and supervision. The availability of professors and support staff for educational activities allowed me to obtain the necessary insights for the conduct of my research work in a completely appropriate framework. Their kindness is quite remarkable. At the same time, I acknowledge all my consideration to the teaching body and more particularly to:

- My Thesis Directors, MERAHI Yassine and RAMDANE-CHERIF Amar. For their recommendations, sound advice and the regularity of their interventions as I progressed in my work.

- My professors, in particular, SOUKANE Assia, BENMASSAOUD GABIS Asma and ACHELI Dalila. who know how to be there whenever they are called.

Also and throughout my university career, my parents and my in-laws have supported and encouraged me to carry out my work. To this must be added the unfailing support of my husband, whose presence is more than encouraging and comforting. Finally, I must point out that the staff of my university have spared no effort to provide their help and assistance to allow me to progress normally in the search for quality in what I was undertaking and efficiency in my steps. May all these illustrious personalities find here the expression of my sincere gratitude and the testimony of my deep respect.
My father-in-law used to say: there is no truth without brilliance nor beautiful roses without thorns. With hindsight, I agree with this reflection. Indeed, thanks to the contribution of our teachers, we have perceived the brilliance of science and harvested its fruits with the delicacy that befits the circumstances. The merit of this thesis belongs to them because they knew how to be there to teach us, enlighten us, guide us and direct us.
These dedications, therefore, go to those, many of whom have contributed to the culmination

of this work through which they can identify themselves as having contributed directly or indirectly to their knowledge and knowledge. Their invaluable help on the intellectual and material levels is felt through the results of our research. It is also manifested by the echo of their efforts felt elsewhere and the satisfaction born of their contribution.
If, for me, it is indeed a well-deserved consecration after several years of hard work, I have no reservations to make that it is also theirs. There are many names to mention. I limit myself to those who have marked me because at some point they were able to demonstrate their efficiency and provide invaluable advice. I am particularly thinking of:

- My grandmother and uncles AIT SAADI Said, Mohammed, and Karim.
- SI MOHAND Family
- My colleagues YAHIA Selma and MEKHMOUKH TALEB Sylia.

I also do not forget my parents (EL-Hocine and Fatma Zohra) and sisters (Sara and Amina) who used simply the right words to motivate me to move forward in everything I undertook. They are there, attentive to my results and happy when I progress in my studies. Their joy was immense at the announcement of my results and that filled me with joy.
Finally, I think of my in-laws and Yaniss, my husband. He was by my side at the right time. He passed on to me the notions of rigour and perseverance. He often returned to the quality of research and the effectiveness of actions. They avoid wasting time, he confides to me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ABC Ant Bee Colony

ACO Ant Colony Optimization

AEFA Artificial Electric Field Algorithm

AI Artificial Intelligence

AO Aquila Optimizer

AOA Arithmetic Optimization Algorithm

APF Artificial Potential Field

A* A-Star

AVOA African Vultures Optimization Algorithm

BA Bat Algorithm

BBO Bibliography-Based Optimizer

BLP Bi-Level Programming

CFO Central Force Optimization

CSA Crow Search Algorithm

EMA Exchange Market Algorithm

FWA Fire Work Algorithm

DA Dragonfly Algorithm

DE Differential Evolution

DoS Deny Of Service

DTBO Driving Training-Based Optimization

EOBL Elite Opposition Based-Learning

ES Evolution Strategy

FA Firefly Algorithm

GA Genetic Algorithm

GCS Ground Control Station

GLS Guided Local Search

GNSS Global Navigation Satellite System

GPS Global Position System

GSA Gravitational Search Algorithm

GSO Glowworm Swarm Optimization

GWO Grey Wolf Optimization

HALE High Altitude, Long Endurance

HGSO Henry Gas Solubility Optimization

HS Harmony Search

HTOL Horizontal take-off and Landing

IMU Inertial Measurement Unit

KH Krill Herd

LALE Low Altitude, Long Endurance

LASE Low Altitude, Short Endurance

LCA League Championship Algorithm

LIDAR Light Detection And Ranging

MALE Medium Altitude, Long Endurance

MO Multi-Objective

MPA Marine Predator Algorithm

MRFO Manta Ray Foraging Optimization

MVO Multi-Verse Optimizer

OBL Opposition Based-Learning

PRM Probabilistic Road Map

PSO Particle Swarm Optimization

RC Radio Control

RRT Rapid-exploring Random Trees

SA Simulated Annealing

SAR Synthetic Aperture Radar

SCA Sine Cosine Algorithm

SI Swarm Intelligence

SO Single Objective

SSA Salp Swarm Algorithm

TLBO Teaching Learning Based Optimization

TS Tabu Search

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicles

USV Unnamed Surface Vehicle

UUV Unmanned Underwater Vehicles

VD Voronoi Diagram

VG Visibility Graph

VTOL Vertical take-off and Landing

WHO Wild Horse Optimizer

WOA Whale Optimization Algorithm

GENERAL INTRODUCTION

These days, the Unmanned Aerial Vehicles (UAVs) has attracted the attention of research, industries in all the world due to the benefits that they offer from efficiency and manoeuvrability. With the rapid growth of the technologies implemented in the UAVs, UAVs are being deployed in various fields including both the military and civil applications. There design allows their access to hostile environments, such as: war, forest, volcano, etc. With the required and dedicated payloads, the UAV can perform search and rescue, object reconnaissance, target tracking, etc. These kind of applications require generally from the UAV the determination of the path required to take.

The autonomy of the UAV in taking decisions in finding the optimal path one the critical and fundamental issue in the UAV applications. This issue is known in the literature as the UAV path planning problem. It consists of determining the shortest path that the UAV need to take to accomplish its mission. As the real environment is full of objects and obstacles in the route, it is necessary to ensure and maximize the safety of the UAV from obstacles and threats until its reaches the final destination. Contrary to the other robots, the UAVs cannot perform sharp turns and hard manoeuvrability due to their dynamic constraints [1]. Therefore, for an efficient and optimal use of the UAV, all these requirement need to be considered. In the literature, several methods were proposed to solve the issue of the UAV path planning, such as classical and Artificial Intelligence approaches, which are reviewed in [2]. As the UAV path planning problem is considered as an optimization problem, the meta-heuristic algorithms can successfully optimize it. However, meta-heuristics present some drawbacks that need to be improved. Addressing the UAV path planning problem and the meta-heuristics limitations constitute our main objective in this thesis.

This thesis is mainly based on three major contributions. In the first contribution, we have proposed an improved optimization approach based on African Vulture Optimization algorithm (AVOA) for solving the UAV path planning problem, named a Chaotic Cauchy Opposite African Vulture Optimization algorithm (CCO-AVOA). Our approach integrates three strategies, as follows: logistic chaotic map, Cauchy mutation, and elite opposition

based learning (EOBL) strategies.
In the second contribution, we have suggested a hybrid approach, called a Chaotic Aquila Optimization-Simulated Annealing algorithm (CAOSA), based on the hybridization of Chaotic Aquila Optimization with Simulated Annealing for solving the UAV path planning.
In the last contribution, we proposed a novel hybrid algorithm for solving the UAV placement in smart cities, called Improved Manta Ray Foraging Optimization-Tabu Search (IMRFO-TS). Our approach is based on the hybridization of Improved Manta Ray Foraging Optimization, which includes a non-linear control strategy, with the Tabu Search algorithm.
The organization of our thesis :
Chapter 1 introduces an overview of the Unmanned Aerial Vehicles (UAVs) including their system architecture, classification, advantages and weakness. In chapter 2, we addressed the highlight of the optimization problems, their classification, the main algorithms used for solving them and their characteristics.
Chapter 3 surveys the UAV path planning problem, the step required for solving it, the main objectives and constraints, and finally the approaches proposed in the literature during the last two decades.
In chapter 4, we suggested our improved approach (CCO-AVOA) algorithm for solving the UAV path planning, and demonstrated its efficiency in comparison with several well-known meta-heuristic algorithms.
Chapter 5 presents the application of our hybrid approach (CAOSA) algorithm for solving the UAV path planning in complex environment, and evaluated the performance of CAOSA algorithm considering several parameters comparing to other optimization algorithms.
In chapter 6, we addressed the UAV placement problem and proposed our IMRFO-TS algorithm for solving it. The efficiency of our approach is presented and validated compared to various methods.
Finally, we conclude this thesis with a summary of our findings and objectives in the future regarding this research.

CHAPTER 1

AN OVERVIEW OF UNMANNED AERIAL VEHICLES (UAVS)

1.1 Introduction

Unmanned Aerial Vehicles (UAVs), known as drones represent the new generation of flying mobile robots, which are widely used to offer services in complex areas. Nowadays, UAVs are often referred to as smart robots, since they can perform complex tasks in autonomous way such as reconnaissance, rescue, assistance, etc. The reason behind this performance is the technologies embedded in the UAV systems that are mainly based on nano-technologies and Artificial Intelligence (AI). With the rapid growth of the technologies, the use of UAVs is expected to increase in the future and more challenges will appear with.
This chapter introduced a general overview of UAVs, their system architecture, classification, and their main advantages and weakness.

1.2 Unmanned Aerial Vehicles (UAVs)

Unmanned Aerial Vehicles (UAVs) refer to the class of mobile robots which are able to fly in an autonomous way without any human intervention and control onboard [3]. One of the main UAVs features is their ability in accessing dangerous and hard areas that other entities cannot. This feature allows the UAV today, to offer various services in both military and civil applications. In addition, the ability of the UAV to carry different types of payload according to its capacity increases quickly their deployment.
With the rapid growth of intelligent systems and technologies, UAVs today are becoming more and more autonomous and able to take decisions, which is a challenging subject. An example of these decisions is the ability of the UAV to select in an efficient way the path that it should take during its mission, under some requirements, such as the UAV resources and the environmental conditions.

1.3 UAV System Architecture

The system architecture of the UAV consists of several units and components, such as an onboard computing Unit, a communication unit, a driving unit, sensors and actuators, a power unit, and a payload.

1.3.1 On-board computing Unit

The onboard computing unit can be represented as the UAV's brain. It is mainly built with electronic devices divided into two a high-level control part and low-level control. The low-level control part is responsible for the flight and navigation control in the take-off, ascent, cruise, descent, and landing phases. This control part is handled by a component called the flight controller, which is a microcontroller. This module receives the UAV's position information from the Global Navigation Satellite Systems (GNSS) compass. It also receives other information in this context from different sensors, such as cameras. Afterwards, the flight controller uses an efficient algorithm to adapt the UAV position according to the desired one, by performing control on the following parameters:

- Position control: altitude and horizontal coordinates
- Velocity control: vertical and horizontal speed
- Attitude control: pitch, roll, and yaw angles

By the end, the flight controller sends the required commands to specific modules, called actuators to set the UAV motions.
The second control part is called the high-level control. The high-level control subsystem is responsible for managing and processing other tasks than flight control. It includes a microcontroller and memory managed by an operating system software, like computers. The configuration of this subsystem is oriented and defined according to the dedicated mission or application, such as mission planning, task management, etc. The microcontroller can collect different types of data from both basic and customized sensors for processing. It can also request information from the flight controller if necessary for some tasks. Mainly, the high-level control system performs high computational complexity despite remaining optional.

1.3.2 Communication unit

The communication unit can be described as a set of components used by the UAV to communicate with external entities. In the communication unit, we can find several data link interfaces derived from the following modules:

- Radio Control (RC) receiver: RC receiver is an electronic component, which consists of an antenna operating in the $2.4GHz$ frequency. This receiver is mainly used to receive data from the ground control station (GCS). Through this wireless interface, the GCS sends commands to control the UAV flight.
- Modem Radio : The modem radio is used for the telemetry and information exchange with the ground control station (GCS) using the $915MHz$ frequency. Through this module, the UAV sends its information and surroundings.

1.3.3 Driving unit

The driving unit gathers the hardware devices of the aircraft, which are used for driving and navigation. The driving unit includes various devices as follows:

- Propellers: the propellers are the basic modules used to carry the UAV in the sky by generating airflow and spinning. The pressure gap within the propellers creates a thrust force which makes the UAV moves in the horizontal plane
- Rotors: the work concept of the rotor is similar to the propeller. However, rotors produce a lift force in the vertical plane to control the UAV attitude.
- Motors: motors in the UAV system are used to make the UAV flies by spinning the propellers. This kind of motor is used for the rotor wings UAVs.
- Frame: The frame represents the structure that holds all the UAV components

1.3.4 Sensors and actuator

Sensors

The UAV contains several devices used to collect and measure various parameters. These devices are called sensors. Their main usage is to ensure a stable and safe UAV flight. UAV's sensors are divided into two categories: active and passive sensors. In the active category, we can find the primary sensors following :

- Inertial Measurement Unit (IMU): this unit gathers information related to the inertial movement from different sensors. This information allows the flight controller in keeping the UAV stable in the sky. In the IMU, we can find the accelerometer sensors, which inform the linear acceleration of the UAV in any axes. The magnetometer is used to indicate the direction of the magnetic field. Finally, the gyroscope is used to determine the UAV's angular velocity.
- Global Navigation Satellite System (GNSS)/Global Position System (GPS): These components determine the real-time positioning information, which helps in determining both the position and speed.

Besides the previous sensors, in the UAV we can find other sensors, such as Light Detection And Ranging (LIDAR), which is used to detect the presence of obstacles. infrared sensors are also used in the UAV to determine the environment temperature.
In the UAV, we can also find the passive sensors. But, most of them are optional, such as electro-optical cameras, optical flow sensors, etc.

Actuator

By definition, the actuator is a device that transforms an energy signal into a mechanical one. In the UAV, the actuator is placed after the flight controller to receive the control signal. Afterwards, the actuator transforms the electrical signal into a motion signal and redirects it to the motors used in the driving unit. In the UAV, we can find the following actuators:

- Linear actuator: the linear actuator transforms the received signal into a straight linear motion. It operates when linear control of the UAV surface is required.
- Rotary actuator: this kind of actuator converts the control signal into a rotary motion. The rotary actuator is involved in the direction changes and rotation is needed.

1.3.5 Power unit

The power unit is responsible for providing the required energy to the different UAV hardware components. The power unit includes mainly three components as follows:

- Power source: The power source is defined as the raw material, which provides electrical energy. For the UAV, the power can be extracted from different sources, such as electrical, chemical, and Hybrid.
- DC converter: the DC converter is an electrical block placed at the output of the power source. This block allows fixing the voltage at a specific value for the components' electrical security purposes.
- DC Bus: the DC bus consists of several parallel electric wires, that transfer the current to the other units.

1.3.6 Payload

The term payload refers to any additional component held by the UAV to perform a customized task. The payload is one of the secondary UAV components that can be replaced or removed. The UAV can carry several types of payload depending on its physical characteristics such as weight, size, power source, etc. Among the payload, we can find the following types:

- Electro-optical camera: the Electro-optical camera uses visible and infrared sensors for object detection. Generally, it is used in the military, surveillance and tracking applications.
- Synthetic Aperture Radar (SAR): It is another type of imaging system which provide high-resolution quality. The SAR system uses the reflection of the transmitted signal over a region for object detection. The SAR system is mainly used in search and rescue operations.
- Transponder: the transponder is an electronic device which uses radio frequency communication for information exchange with other aircraft vehicles. Based on the information, the UAV can be aware of the aircraft positions for collision avoidance. Therefore, the transponder provides safety to the drone.
- Deliverable payload: this type of payload can be any transportable object, where The UAV is used as a cargo carrier. This kind of payload can be mostly found in transportation and mail delivery services.

1.4 UAV Classification

The UAV is characterized by several criteria that can be used to determine the different UAV classes such as weight and range, aerodynamic, landing/taking-off mode, altitude and endurance, operational mode, engine type and power source, and application, as shown in Figure 1.1.

1.4.1 Classification based on weight and range

In the literature, the most common UAV class in research papers is based on the UAV's weight. In fact, the UAV's weight can be considered the most effective metric for many reasons. Firstly, based on the UAV weight, the kinetic energy produced can be predicted, since they are correlated [4]. In addition, the UAV mission is conditioned and determined based on its weight due to the fact that the weight can impact the UAV's safety at some points. Based on the UAV weight, the UAVs can be classified into nine sub-classes as follows:

Nano UAV

The first UAV sub-class is the Nano UAVs, which represent the lightest UAVs. The weight of the UAV belonging to this class is generally under $200g$. The nano UAVs can be launched by hand and are easily portable and carried. Their main use case is reconnaissance, surveillance, and other local applications that can be performed in a small range of less than $5Km$. In the nano UAV, we can find both fixed wings and rotary wings like FLIR Black Hornet III shown in Figure 1.2a.

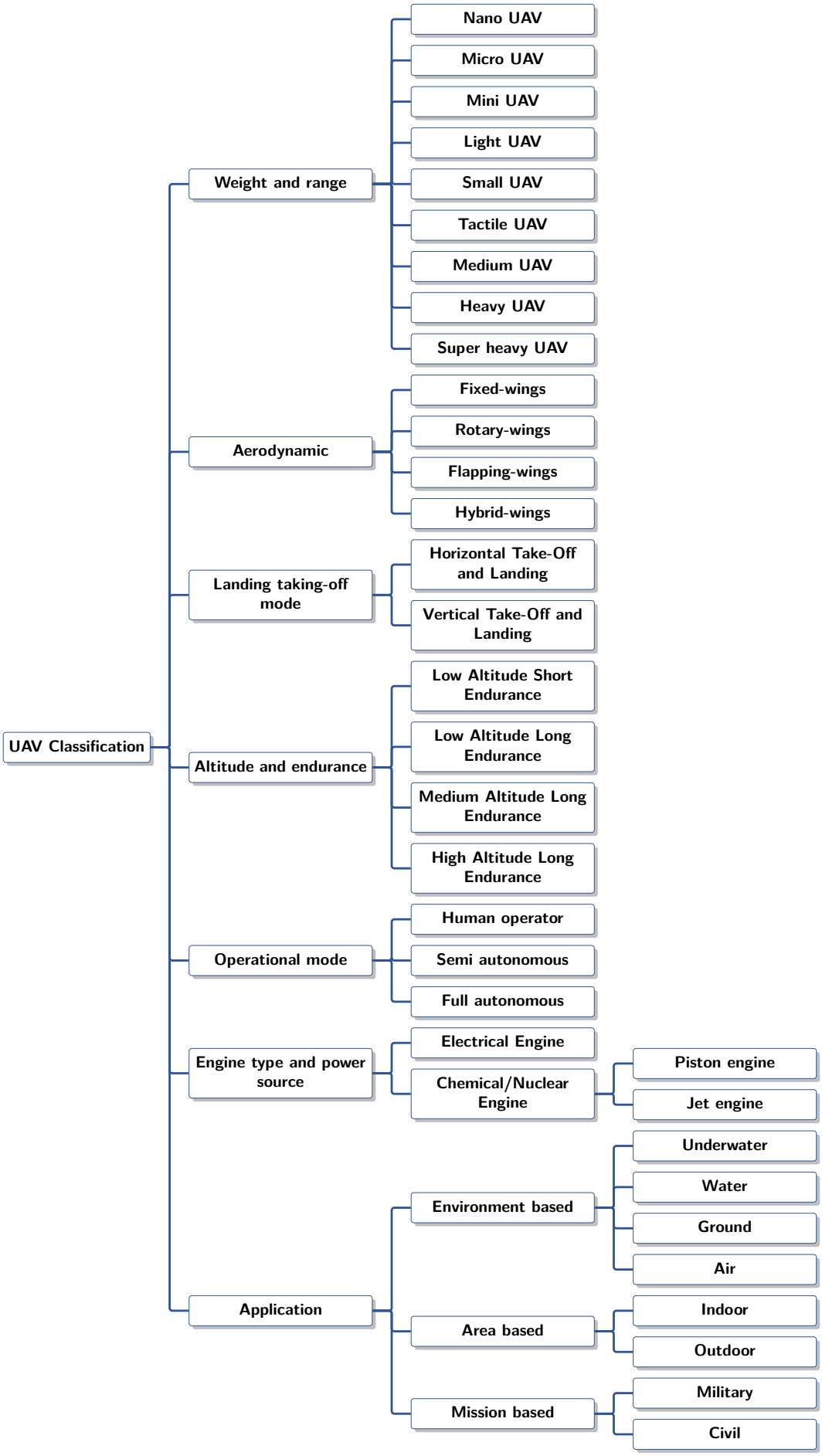


Figure 1.1: Classification of meta-heuristic algorithms

Micro UAV

The micro UAV sub-class comes after the nano UAVs, which are bigger and their weight is varied between $200g$ and $2Kg$. Micro UAVs are also portable UAVs and can perform the same applications as nano UAVs. however, the micro UAVs are superior to the nano UAVs in terms of maximum range, whereas the micro UAVs are limited to the $25Km$. Therefore, they can cover larger areas. In the micro UAV sub-class, we can find both fixed-wings and rotary-wings UAVs, such as Parrot bebop 2 illustrated in Figure 1.2b.

Mini UAV

The mini UAV sub-class gathers larger UAVs, which their weight is included in the range of $2-20Kg$. The mini UAVs are able to several applications in a wider range than the previous UAV sub-classes, where their maximum range is limited to the $40km$. Based on the mini UAVs' weight, they can perform both ground and sky levels flights. Note that UAVs under $9Kg$ can be carried by hand and perform only applications at the ground level, due to the fact that their altitude range is limited. As for the mini UAVs, which weight exceeds the $9Kg$, they require tools for launching, such as catapult. RQ 14 Dragon Eye is an example of a mini UAV, which is represented in Figure 1.2c.

Light UAV

The light UAVs are heavier than the previous UAV sub-classes mentioned before. The light UAV's weight does not exceed the $50Kg$. The main advantage of the light UAV is their ability to carry light payloads for distances that do not exceed the $70Km$. This feature makes the light UAV suitable for mail and small package delivery applications. Figure 1.2d shows an example of a light UAV, called RQ-21 Black-jack.

Small UAV

The small UAVs come after the light UAVs in terms of weight. The small UAVs' weight is included in the range of $50-150Kg$ and can perform tasks in larger areas which are less than $150Km$. Due to their weight, Most of the UAVs in this weight range are fixed-wing UAVs. In addition, this UAV type can carry more payloads than the previous UAVs at both ground and sky level. Yamaha RMAX represented in Figure 1.2e, belongs to the small UAV class.

Tactile UAV

The tactile UAVs are larger UAVs weighing more than $150Kg$, and less than $600Kg$. Mainly tactile UAVs are used for weapon carrying and heavier payloads within the same area range as the small UAVs. The tactile UAVs also share with the small UAVs the wings type, which

is the fixed wings. But, we can find a few rotary wings UAVs in this class, such as Volocopter 2X UAV displayed in Figure 1.2f.

Medium UAV

The Medium UAVs include the UAVs whose weight is between 600 and $1000Kg$. The medium UAVs can accomplish flights with a maximum range of $250Km$ at a middle altitude level. Starting from the medium UAV sub-class, the UAVs require a dedicated and improved runway for landing and taking-off [5]. Baykar Bayraktar TB2 represented in Figure 1.2g, is a type of medium UAV.

Heavy UAV

The heavy UAV's weight is limited by the medium UAV's maximum weight and the super heavy UAV's minimum weight, which means in the range of 1000 and $2000Kg$. The heavy UAVs can fly over large areas that do not exceed $1000Km$ at higher speeds. The wing type adopted to the heavy UAVs is the fixed wings due to the important weight and payload type like the MQ-8B Fire Scout UAV illustrated in Figure 1.2h.

Super heavy UAV

The last UAVs sub-category is called the super heavy UAVs, which are the largest. The super heavy UAVs' weight exceeds the $2000Kg$. The super heavy UAVs perform long-distance flights that can reach the $1500Km$. Moreover, they can fly at high-level altitudes due to their high speed. The main application of this UAV sub-category is wide-area surveillance, infrastructure inspection, penetrating attacks, etc. Global Hawk UAV shown in Figure 1.2i, is a super heavy UAV, used in military applications.

1.4.2 Classification based on aerodynamic

In this class, the UAVs are classified based on their aerodynamics, which is responsible for generating and producing thrust. The thrust allows the UAV to move forward and control the flight. Based on the aerodynamic criteria, we can find four (4) types of UAVs: fixed wings, rotary wings, flapping wings, and hybrid wings. Figure 1.3 illustrates an example of each UAV in this class.

Fixed-wings

The design of the fixed-wing UAV is similar to the migratory bird [6]. The structure of the fixed-wing UAV is simple and less complicated. It is equipped with two rigid wings to provide lift when the UAV moves forward [7]. This kind of UAV is considered energy efficient since they do not require much energy except in moving forward. Moreover, they can fly

(a) Nano UAV: FLIR Black Hornet III

(b) Micro UAV: Parrot bebop 2

(c) Mini UAV: RQ 14 Dragon Eye

(d) Light UAV: RQ-21 Blackjack

(e) Small UAV: Yamaha RMAX

(f) Tactile UAV: Volocopter 2X

(g) Medium UAV: Baykar Bayraktar TB2

(h) Heavy UAV: MQ-8B Fire Scout

(i) Super heavy UAV: Global Hawk

Figure 1.2: Example of the UAVs in weight/range class

long distances and at high altitude levels, like the Sagem Crecerelle UAV shown in Figure 1.3a. However, their design does not allow them to hover in the same place. Therefore, they cannot maintain low speed. In addition, their manoeuvrability is limited due to their physical characteristics and engine. For these reasons, fixed-wing UAVs are not suitable for applications which require precision like inspection and monitoring. Thus, they can be used in delivery services, military applications, remote-control racing, product spraying in agriculture, etc.

Rotary-wings

The design and mechanical concept of the rotary wings UAV is more complex than the previously mentioned category. In the rotary-wings UAV, the lift is produced around the rotation centre of one or many rotors. The lift type provided by the rotors allows the UAV to hover at a fixed altitude and retain good stability. Therefore, the rotary-wings UAV can maintain a specific localization, which makes it optimal for stationary applications such as monitoring, surveillance, and wireless coverage. The DJ Mini 2 shown in Figure 1.3b is an example of a rotary-wings UAV.

Flapping-wings

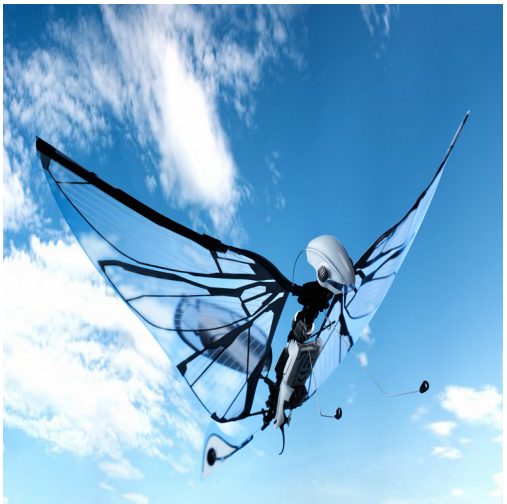
In the context of flapping wings UAV, the UAV's design is inspired by the biology of birds and insects, like the Metafly UAV presented in Figure 1.3c. The concept of this type of UAV is to flap the wings to produce the lift. Unlike the fixed-wings UAV, the flapping-wings UAV can produce thrust and hover sometimes [8]. These features allow for the UAV to establish a stable flight, especially in winds. Moreover, with the design of wings, these UAVs perform better manoeuvrability than fixed-wings UAVs. But, not as much as rotary wings. In addition, the flapping-wings UAVs are not really considered optimal regarding energy consumption, since the wings' actuation requires much power. We can find this type of UAV in many applications, such as military, surveillance, aerial photography, etc.

Hybrid-wings

The design of hybrid-wings UAV is based on the combination of fixed, rotary, and flapping-wings UAVs. The purpose of such design is to take benefits from both aerodynamics types and bring additional improvement. For example, a hybrid rotary with fixed wings UAV uses two propulsion systems during the flight, like the Skywalker X8 tilt-rotor displayed in Figure 1.3d. It can use the rotor for hovering and lifting to have better flight stability. For fast forwarding, the hybrid UAV uses the conventional fixed-wings system. Thus, this design gathers the benefits from both rotary and fixed wings systems.

(a) Fixed-wings: Sagem Crecerelle

(b) Rotary-wings: DJ Mini 2



(c) Flapping-wings: Metafly

(d) Hybrid-wings: Skywalker X8 tilt-rotor

Figure 1.3: Example of the UAVs in the aerodynamic class

(a) HTOL: Bormatec maja

(b) VTOL: Eagle eye

Figure 1.4: Example of the UAVs in the landing/taking-off class

1.4.3 Classification based on landing/taking-off mode

In this part, the UAVs are classified based on their landing/taking-off mode, which includes: Horizontal take-off and Landing (HTOL) and Vertical take-off and Landing (VTOL). Figure 1.4 represents examples of UAV in the landing/taking-off mode class.

Horizontal Take-Off and Landing

The fixed-wings UAVs mainly operate in the horizontal directions for both landings and taking off. This mode requires the UAV's high speed for forwarding and smooth landing to ensure its own safety. Both horizontal landing and taking-off are complex and require a specific runway. However, their main advantage is the power source conservation, which makes them suitable for Long distance travels. Among the HTOL UAVs, we have the Bormatec maja UAV presented in Figure 1.4a.

Vertical Take-Off and Landing

This kind of take-off is usually performed by rotary wings UAVs due to their aerodynamic model. In this mode, the UAV flies in the vertical direction and easily hovers anywhere. This landing mode is considered simple and does not require any specification for landing, unlike HTOL. For these reasons, the UAVs that belong to the vertical take-off class are suitable for military applications. However, the UAV's speed in this class is limited due to retreating propellers slowing down [9]. We can find in the mode several UAVs such as Eagle UAV illustrated in Figure 1.4b.

1.4.4 Classification based on altitude and endurance

Other criteria that can be used for UAV classification, are endurance and altitude. Both of them can be gathered under one criterion since they are correlated. Endurance is defined as the maximum amount of time the UAV can be in flight, while altitude refers to the maximum altitude level that the UAV can reach during flight. Based on these features, the UAVs can be classified into four (4) sub-categories as follows:

Low Altitude, Short Endurance (LASE)

The first sub-class is the Low Altitude, Short Endurance, or LASE. The LASE UAVs category represents the UAV that flies at a closer altitude to the ground level in the range of $300m$. In addition, the LASE UAVs are characterized by their short endurance that does not exceed 3 hours. Generally, the LASE UAVs do not require a runway and they can be used in rough terrain. In the LASE UAVs, we can find both VTOL and HTOL UAVs, like the DJI Agras MG-IP UAV displayed in Figure 1.5a.

Low Altitude, Long Endurance (LALE)

The Low Altitude Long Endurance (LALE) UAVs are bigger than the LASE UAVs and can carry heavier payloads. The LALE UAVs can reach a maximum altitude level of $4500m$. In terms of flight time, the LALE UAVs' endurance is limited in the range of $3 - 10H$. This feature does not allow them to perform longer missions to the power limitation. Among the LALE UAVs, we have the Scan Eagle UAV shown in Figure 1.5b.

Medium Altitude, Long Endurance (MALE)

The Medium Altitude Long Endurance (MALE) UAVs represent the category of UAVs that fly at high altitude levels that do not exceed $9000m$. Generally, the MALE UAVs are used in military applications due to their altitude level flight that makes their detection hard. The endurance of the MALE UAVs is bigger than the LALE UAVs and can reach $200h$ in one flight. Wing Loong II is a MALE UAV represented in Figure 1.5c.

High Altitude, Long Endurance (HALE)

The High Altitude Long Endurance (HALE) UAVs constitute the larger UAVs in terms of altitude level and endurance. The HALE UAVs can fly at altitude levels up to $21000m$ for a period of $50H$. Most of the HALE UAVs belong to the fixed-wings UAVs category since they are heavy, such as the Zephyr UAV displayed in Figure 1.5d.
Figure 1.5 shows an example of the UAVs that belong to the Altitude and Endurance category.

(a) LASE UAV: DJI Agras MG-1P

(b) LALE: Scan eagle

(c) MALE: Wing Loong II

(d) HALE: Zephyr

Figure 1.5: Example of the UAVs in the altitude/endurance class

1.4.5 Classification based on operational mode

The UAV operational mode can be defined as the level of control of the UAV during the flight. The UAV operational mode is different from one drone to another depending on the application's needs. Mainly, the UAV has three modes: human operator, semi-autonomous, and fully autonomous modes.

Human operator

The human operator mode represents the lowest level of control, which can be handled by the UAV. In this mode, the flying is completely controlled by a human operator through the remote controller. The control commands are directly received by the RC receiver of the communication unit and executed on the driving unit components. For this control mode, the human operator should master the UAV guidance.

Semi autonomous

The semi-autonomous operator is the second operational mode of the UAV. This operational mode of UAVs is considered smarter than the first type since they have a superior level of autonomy. The semi-autonomous UAVs are able to perform some tasks without any human intervention such as obstacle detection and avoidance, trajectory planning, etc. However, the human operator can involve at any moment and interrupt the tasks. Human intervention can be beneficial to the UAV in some situations, especially when it fails or is unable to accomplish the mission at some point.

Full autonomous

The fully autonomous operational UAVs are the most intelligent UAVs compared to the previous ones. This kind of UAV is able to perform completely the requested tasks without any intervention. for example for path planning, they can determine their own localization in the area. Afterwards, they computed their own trajectory in an efficient way using an optimal approach. Finally, accomplish their task. Even though, in this mode, the operations are handled by the drone, the human operator can monitor the UAV flight via the GCS.

1.4.6 Classification based on engine type and power source

The UAV power source differs from one UAV to another. It can be related to the UAV size, where we can find power sources which can be integrated into small UAVs and others not. It can be also related to the UAV application, where some applications require more time (more power source) than others. Figure 1.6 illustrates the different UAVs in this class.

Electrical Engine

The electrical engine is one of the most engine types used in UAVs, especially in light and small UAVs [4] since the electrical engine does not take up much space in the UAV system. For this type of engine, the main source which generates the power is the battery. The energy is transferred from the battery to the engine until the UAV driving unit maintains the UAV operational mode. Regarding the battery model, we can find several types such as solar batteries, lithium-ion batteries, nickel metal hybrid batteries, etc. DJI S1000+ shown in Figure 1.6a, is a type of UAV equipped with an electrical engine, which uses the battery as a power source.

Chemical/Nuclear Engine

1. Piston engine
   The piston engine is also an engine type widely used in the heavy UAVs category due to its payload capacity. The piston engine is an internal combustion system that generates initially the power from the combusted air or fuel. This power is exerted on the piston for moving back and forward in the cylinders. The piston motion's power is transmitted to the crankshaft to produce torque. which includes: consists of several pistons, are connected with a crankshaft. The pistons in this engine move in several cylinders, which generate power. Several piston engine types are used today in UAVs, among them:

   (a) Two stoke: In the two stoke engine, two cylinders are used in the engine to produce the power. This engine is light and can be used in the light UAV [10], such as MLB Bat 3 shown in Figure 1.6b.

   (b) Four stoke: the four stoke engine is bigger compared to the two stoke engine, which consists of four cylinders to produce more power. The four-stroke engine takes more space than the previous type which makes it suitable for UAVs that exceed $50Kg$. Seeker 400 displayed in Figure 1.6c, uses the four stoke engine as an engine.

   (c) Rotary engine: the motion concept of the rotary engine is different from the previous engines. In the rotary engine, the crank-shaft is stationary and the power is produced by the rotary movement of both engine housing and the cylinders. This feature produces good balance and better compression compared to the Stoke engine. As shown in Figure 1.6d, RQ-6 Outrider is one of the rotary engine UAVs.

   (d) Wankel engine: The concept of the Wankel engine is derived from both stoke and rotary engines. The piston in the Wankel engine is triangular and converts the fuel combustion energy to a mechanical one by the rotary movements inside the

engine housing. The Sikorsky cypher UAV presented in Figure 1.6e, produces power using the Wankel engine.

2. Jet engine
   The jet engine is another type of engine used in the UAV, especially the Medium and High Altitude UAVs to produce the necessary energy. The jet engine came after the piston engine to bring improvement, mainly the speed efficiency. The jet engine takes the fuel and gas as a power source like the piston engine, which is converted based on the thrust produced in the opposite sense of the combusted fuel. In the jet category, we can find the following engines:

   (a) Rocket engine: the rocket engine is a type of jet engine that uses liquid fuel or solid or both as a power source. It is also known as a reaction engine, due to the fact that the produced power is generated from the burnt fuel and oxidizer in the combustion room. The combustion products are expelled through a nozzle that generates trust. The WZ-8 UAV shown in Figure 1.6f, uses the rocket engine in its power unit.

   (b) Turboprop engine: the turboprop engine is equipped with a propeller connected to a turbine to generate power. The turbine takes the power from the compressor rotations and then transfers it to the driving shaft which pushes the propeller to turn by itself. The size of the turboprop engine is not big and can be used for light UAVs. We can find this type of engine also on the super heavy UAV such as MQ-9A Reaper, presented in Figure 1.6g.

   (c) Turbojet engine: the power of the turbojet engine as the jet engine types, is derived from the fuel combustion. The particularity of the turbojet engine is to use the energy of the exhaust gas and transfers it to the turbines that drives also the compressor. To generate trust, the nozzle bushes the exhaust gas from the back to the front. The WZ-7 Soar Dragon shown in Figure 1.6h, uses the turbojet engine.

   (d) Turbofan engine: the concept of the turbofan engine is similar to the turbojet engine, they both share the same core. The difference between the turbofan engine and the turbojet engine is the presence of fans. Due to the fans, the air bypasses the combustion room and improves trust generation. Figure 1.6i illustrates an example of an UAV equipped with a turbofan engine, called the RQ-3 Darkstar.

1.4.7 Classification based on application

Nowadays, UAVs are deployed in several fields, where they perform tasks for a dedicated application. Thus, they can be classified based on the application criteria, as follows:

(a) Electrical engine: DJI S1000+

(b) Two stoke engine: MLB Bat 3

(c) Four stoke engine: Seeker 400

(d) Rotary engine: RQ-6 Outrider

(e) Wankel engine: Sikorsky cypher

(f) Rocket engine: WZ-8

(g) Turboprop engine: MQ-9A Reaper

(h) Turbojet engine: WZ-7 Soar Dragon

(i) Turbofan engine: RQ-3 Darkstar

Figure 1.6: Example of the UAVs in the engine type/power source class

Environment based

In this sub-category, the UAVs are classified based on the environment, where they perform the missions. Generally, UAVs can be deployed in four environments: underwater, on the water, ground, and air.

1. Underwater
   Underwater drones can be referred to the Unmanned Underwater Vehicles (UUV). They are a class of UAVs equipped with the mandatory systems and technologies for underwater environment adaptation. Similar to air UAVs, they can be either autonomous or remote-controlled Underwater UAVs.

2. Water
   Water UAVs are defined as the category of UAVs that operate at sea level. They are also known as Unnamed Surface Vehicles (USVs). In terms of navigation, the USVs are similar to surface vehicles such as boats, etc. The particularity of USVs is their autonomy and intelligence.

3. Ground
   Ground UAVs, sometimes known as the Unmanned Ground Vehicles (UGV), are the type of UAVs that operate at the ground level. Ground UAVs are suitable for dangerous and harsh environments, where it is hard for humans to operate.

4. Air
   The air UAVs are the typical flying UAVs, also known as drones. The UAVs operate at the air level, above the water and ground levels. The UAVs can perform tasks at different levels of altitude depending on their characteristics and mission objectives.

Area based

Besides the environment, the UAV use cases depend also on the area where it is deployed, including the indoor and outdoor environments.

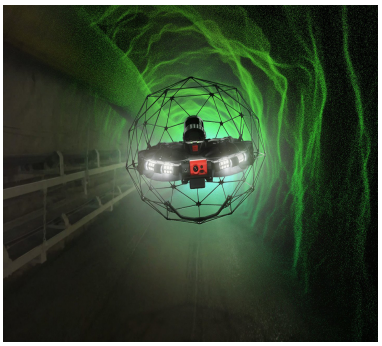
1. Indoor
   Indoor UAVs are generally the UAVs deployed in a closed environment, such as offices, buildings, parking, etc. Most of these areas are cluttered and full of obstacles.

2. Outdoor
   The outdoor environment can be seen as a free space compared to the indoor environment, where the obstacles are less and the UAV is freer to navigate. The outdoor environment for the UAV can be a rural terrain, forests, etc.

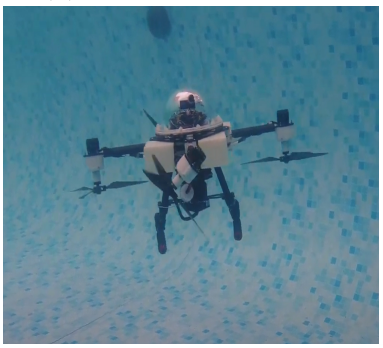
Figure 1.7 illustrates an example of Environment and area based UAVs.

(a) Water: Z-boat 1800

(b) Ground: Huuver



(c) Air: Elios 3



(d) Underwater: TJ-FlyingFish

(e) Indoor: Parrot Mambo

(f) Outdoor: Boeing X-50 Dragonfly

Figure 1.7: Application based drone: Environment and area based UAV

Mission based

UAVs are also used for different types of missions, depending on their nature. Mainly, UAVs are used in both military and civil missions. For each mission, we can find several applications and operations, as follows:

1. Military
   In the beginning, UAVs were only used for military purposes, and the first UAV appeared in late World Ware I [11]. In military applications, several tasks can be allocated to the UAV, such as:
   (a) Missile lunching
   (b) Bomb-dropping
   (c) Spy
   (d) Battlefield
   (e) flying camouflage
   (f) Target acquisition
2. Civil
   Only at the beginning of the new millennium, the UAV started to be commercialized

(a) Missile lunching UAV: SARISA SRS-1X

(b) Bomb-dropping UAV: Defendtex d40

(c) Spy UAV: RQ 170

(d) Battlefield UAV: Area-I Altius-600

(e) Flying camouflage UAV: EHang

(f) Target acquisition UAV: Rheinmetall KZO

Figure 1.8: Application-based drone: Military Mission-based UAV

and used for non-military applications. Starting from that period, the number of UAVs deployed in the market is continuously growing due to their advantages and their easy deployment in several fields, such as:

(a) Agriculture

(b) Photography/Videography

(c) Meteorology

(d) Mining

(e) Medical

(f) Delivery

(g) Wireless coverage

(h) Construction

Figures 1.9 and 1.8 display examples of application-based UAVs.

(a) Agriculture UAV: DJI Agras T40

(b) Photography UAV: DJI Mavic 3 Cine

(c) Meteorology UAV: Meteodrone MM-641/SSE

(d) Mining UAV: Aibotix Aibot X6

(e) Medical UAV: EHang

(f) Delivery UAV: Wingcopter 198

(g) Wireless communication UAV:

(h) Construction UAV: DJI Matrice 300 RTK

Figure 1.9: Application-based drone: Civil Mission based UAV

1.5 UAV advantages and characteristics

1.5.1 Mobility

One of the biggest reasons for the widespread use of UAVs is their mobility. The feature allows the UAV to navigate and manoeuvre freely in a space that might be dangerous for humans or unreachable by other robots.

1.5.2 Cost efficiency

As the use of drones is continuous in various fields, the industrial markets are working on providing better quality/price rates for customers. With the fact that UAVs today attract manufacturers, the competition between them leads to the price reduction of UAVs. In addition, the deployment of UAVs saves more cost instead of traditional tools, aircraft, etc.

1.5.3 Easy deployment and control

Another advantage of UAVs is their easy deployment. Generally, UAVs do not need any requirements in the area where they are supposed to operate. In addition, for the human operator UAVs, the operator does need any technical background or training to control them.

1.5.4 Autonomy

Artificial Intelligence (AI) is taking part also in UAV technologies. The trend today is to make robots including the UAV smarter and more autonomous. The autonomy of the UAV is considered an important aspect and feature, especially in the area where only the UAV can access it since it reflects on its performance and the quality of service that can give. With the development and research progress, we can see that UAVs are able to take and control their own decision in critical cases.

1.6 Limitation and challenges

1.6.1 Resource limitations

Power source limitation

One of the biggest issues in the UAVs is their lack in terms of power source. As known, the UAV's size is defined and specified by the constructor. Each element in the UAV has a specific size and location. Therefore, the allocated space of the power source element in the power unit is limited. In addition, the energy consumption of the UAV is not steady and depends on various parameters such as flight time, flight conditions (wind speed, altitude, etc), the UAV movement (Landing, take-off, rotations), etc. Some flights or applications

require more power than others. Regardless of the nature of the power source (electrical or chemical), the amount of power produced for consumption is limited. Therefore, it needs to be minimized and optimized as much as possible.

Wireless contact limitation

Another UAV limitation is the limitation of the wireless contract with the control station. This issue is more frequent for the UAVs controlled by the GSC. In fact, the communication between the UAV and the GSC is established only within the communication range. Outside, the UAV cannot be controlled. In addition, the communication is established through a wireless channel, which is not secure and may lead to losing legitimate control. contact

1.6.2 Physical limitations

Payload limitation

As mentioned before, the amount of charge that the UAV can carry is predefined. Therefore, the payload in the UAV is limited and any potential overload may damage the UAV.

Versatility limitation

In most cases, the UAV is equipped with a set of devices and payload dedicated to the UAV's mission. The UAV's high-level control in the onboard unit is programmed according to the dedicated task. So, the UAV is aware only of the mission information, and the other tasks are not guaranteed. For this reason, the flexibility of the UAV is limited.

1.6.3 Obstacle avoidance Challenges

Obstacle detection and avoidance is one of the complex tasks required by the UAV to preserve its safety, especially the autonomous one. The important and difficult part of this task is to identify the different obstacles presented in the flying area in the first place. Then, deploying efficient strategies and methods to avoid the detected obstacles. In real applications and environments, this issue gets harder due to the complexity of obstacles in terms of mobility, randomness, and non-uniform structure. Therefore, usually, the robustness of the UAV system is determined based on its ability of hardware and software to solve this problem.

1.6.4 Security limitations

Privacy limitation

The design of the UAV allows him to freely fly in space. This feature is considered one of the vulnerabilities since only the human or control station can control its movement. With embedded technologies and sensors, the UAV can record videos and images without any

authorization requirements. It can also be used for spying, tracking and attack applications. Therefore, the UAV can be easily used for privacy violations.

Security in communications limitation

In the UAV communication system, the messages are exchanged using a wireless link. As with all wireless communications, UAV wireless communications suffer from a lack of privacy and security. First, wireless communications are easy to intercept, which facilitates network identification and access. Second, the UAV communication system is more vulnerable to malicious attacks due to its open links, such as Deny Of Service (DoS), man-in-the-middle, etc. However, these issues can be addressed using different strong techniques such as authentication, encryption and hashing.

1.7 Conclusion

In this chapter, we presented a summary of the UAVs, their architecture, and their advantages and limitations.
As mentioned, UAVs offer a set of advantages, such as mobility which make them particular over the other robots. Despite that, they also present limitations and challenges like the autonomy in the flight and the ability to determine their own trajectory in an efficient way, which is today widely addressed in the research field. This challenging problem is known in the literature as the UAV path planning problem which is classified with the optimization problem category.
Therefore, in the next chapter, we will provide the definition of the optimization problem, as well as the different classes. Finally, we overview the different algorithms used to solve it.

CHAPTER 2

OPTIMIZATION ALGORITHMS

2.1 Introduction

Optimization is an important component and interesting aspect in various fields including, research, mathematics, computer science, management, industrial engineering, etc. By definition, optimization is the discipline or process that aims to achieve the best or most effective use of something in a particular situation. An optimization problem is a problem that needs to be optimized using different methods and approaches, such as the UAV path planning problem.
Considering the interest of the optimization problem, in this chapter, we will give a brief definition of the optimization problems. Followed by their classification criteria such as the decision variables type, number of objectives, constraints, and computational complexity. In the end, we provide the methods used to solve optimization problems which are classified into: classical and Artificial Intelligence (AI) algorithms.

2.2 Optimization problem

2.2.1 Definition

An optimization problem in computer science is defined as a set of tasks that need to be solved by a specific type of algorithm called, optimization algorithms. The optimization algorithms evaluate different possible solutions and return the optimal solution found. Mainly, the optimization problem depends on three parameters such as the search space of the problem domain, the objective function, and the problem constraints. Mathematically, an optimization problem can be expressed as the following equation [12]:

$$P = (D, f, C) \tag{2.1}$$

Where P stands for the optimization problem. D represents the search space of the problem domain. f is the objective function. C donates the problem's constraints.
The search space domain D is a set of variables or data used for solution exploration, generally called decision variables expressed as in Eq (2.2).

$$D = \{X_1, X_2, \ldots, X_n\} \tag{2.2}$$

Where n donates the problem domain dimension.

2.2.2 Objective function

Definition

An objective function is a mathematical representation of one or more objectives that need to be optimized. Depending on the problem nature, the objectives may be subject to some constraints C or not. The objective function attempts to output a real number as an optimal solution based on the relationship between the decision variables in D. Mathematically, it is expressed in the following equation [12]:

$$\begin{aligned} f : D &\longrightarrow \mathbb{R} \\ \forall \quad d \in D, \quad f(d) &= fitness(\mathbb{R}) \end{aligned} \tag{2.3}$$

Directions

Mainly, the objective function has two purposes. First, the objective function is used to evaluate and determine the quality of the possible solution using Eq (2.2). Secondly, is to guide and direct the search towards the promising area [13]. In the literature, we can find two directions of the objective function as follows:

1. **Minimization**
   In the case of problem minimization, the objective function is optimized by finding the global minimum solution $\overline{d}$ using Eq (2.4). The lowest fitness value is the best performance result.

$$\begin{aligned} &\textbf{Minimize} \quad f \\ \forall d \in D, \quad &f(\overline{d}) \leq f(d) \end{aligned} \tag{2.4}$$

2. **Maximization**
   In the case of problem maximization, the best performance represents the highest fitness value. The possible solutions are evaluated for returning the global maximum

solution as the following equation:

$$\begin{aligned} &\textbf{Maximize} \quad f \\ &\forall d \in D, \quad f(\overline{d}) \geq f(d) \end{aligned} \tag{2.5}$$

Where $\overline{d}$ refers to the optimal solution.

2.3 Optimization problem classification

Understanding the optimization problem type and nature is the main key element for their resolution. According to their features, the optimization problem can be classified, as shown in Figure 2.1 into the following classes:

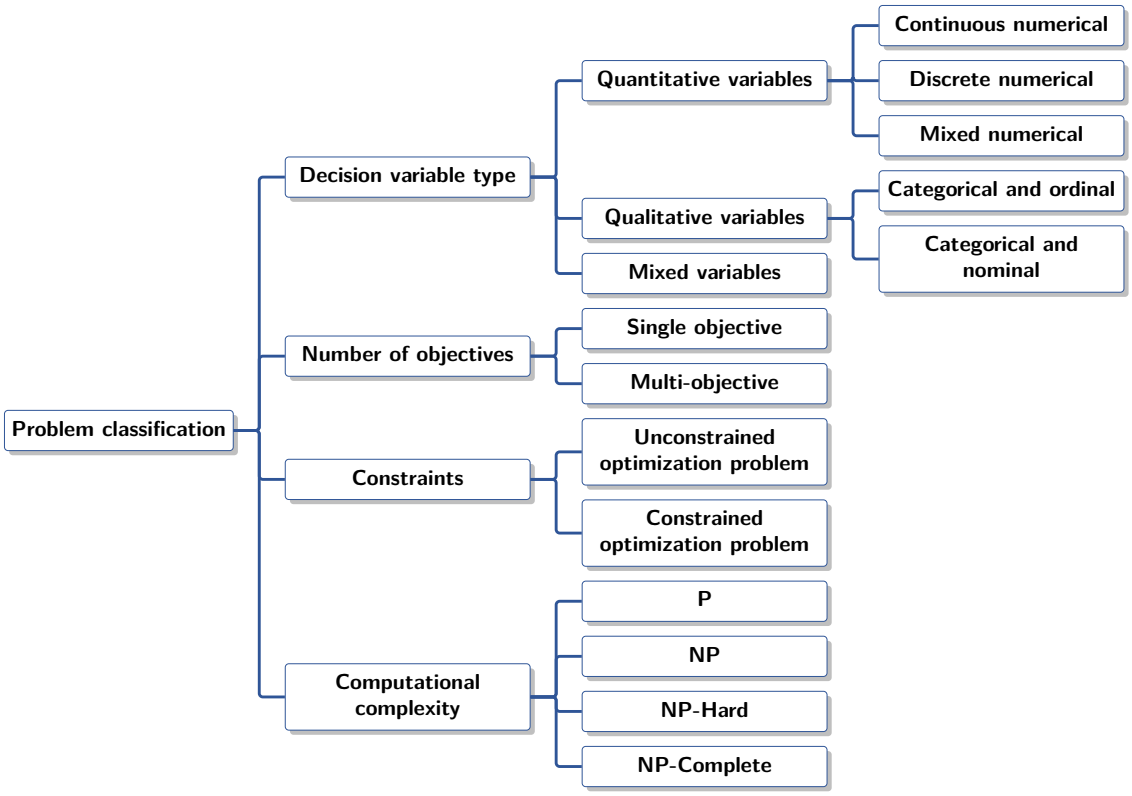


Figure 2.1: Classification of optimization problems

2.3.1 Classification based on decision variable type

In optimization problems, decision variables are a set of unknown and controllable quantities, which belong to a specific domain. Using optimization algorithms, the decision variables can be changed and optimized to suit the defined objective. In the literature, three types of decision variables are presented as follows:

Quantitative variables

Numerical variables are the most common variable type. they include continuous, discrete, and mixed numerical values.

1. Continuous numerical variable:
   The continuous numerical variables are a set of uncounted numbers of values, which can take any numerical value in $\mathbb{R}$ domain.

2. Discrete numerical variable:
   Discrete numerical variables represent the measured variables, that can take any value from a discrete range.

3. Mixed numerical variable:
   The mixed variables are the numerical variables that gather both continuous and discrete variables.

Qualitative variables

Qualitative variables in optimization problems are less known and discussed in the research. Unlike quantitative variables, qualitative variables do not take any numerical values but are described rather by labels or categories. Depending on the categories' features, we can have either categorical and ordinal or categorical and nominal variables.

1. Categorical and ordinal variables:
   The particularity of the categorical and ordinal variables is the order of the variables. In this type of variable, the category's values are ordered in a particular way, which has significance.

2. Categorical and nominal variables:
   In the categorical and nominal variables, the ordering of the values within the category is not impacting and does not have a logical meaning.

Mixed variables

Mixed or hybrid variables are a set of variables that contain both quantitative and qualitative variables. Generally, this type of variable is used in solving mixed optimization problems such as Traveling Salesman Problem.

2.3.2 Classification based on number of objectives

Considering the number of objectives, the optimization problem can have two possible behaviours either a single objective problem or a multi-objective problem behaviour as follows:

Single objective

The single objective (SO) problem is defined as the optimization problem that uses in the fitness function, either a single objective or combines several objectives into one. The advantage of this type of optimization is the ease of determining the best solution, which corresponds to the fittest fitness value. However, the SO problem in most cases, cannot provide alternative solutions that trade-off different objectives.

Multi-objective

Contrary to the single objective problems, the multi-objective (MO) problem includes two or more objectives simultaneously. The MO problem deals with conflicting objectives and handles the interaction among the different objectives [14]. Due to these features, there is no single optimal solution to the MO problem. The MO methods return generally compromised solutions, known as Pareto-optimal solutions in the literature.

2.3.3 Classification based on constraint

In an optimization problem, the constraints are defined as a set of restrictions used to determine whether a solution is feasible or not according to the problem's rule. Based on the constraint criteria, the optimization problem is divided into unconstrained and constrained optimization problems.

Unconstrained optimization problem

Unconstrained optimization problems are a class of problems, which consider the problem of finding the fittest solution depends only on the variables without any constraints or restrictions. In some applications, some constrained optimization problems are converted to unconstrained problems for simplification by replacing the constraints with penalty terms in the objective function. Thus, the constraint problem is solved as an unconstrained problem.

Constrained optimization problem

By definition, constrained optimization problems are the type of problems that optimize an objective function under some constraints on the variables. Based on the type of constraint, two decisions are possible when a constraint violation occurs. If the constraints are hard meaning mandatory, the solution that does not satisfy the constraint is discarded. If the constraints are soft or desirable, the solution is accepted rather than rejected by the algorithm. However, this solution is less optimal and gives bad performance.

2.3.4 Classification based on computational complexity

In optimization problems classification, another criterion can be used which is the computational complexity. Computational complexity refers to the amount of resources required by an algorithm such as time, to solve a problem. It also represents the degree of difficulty of an algorithm in solving optimization problems. Therefore, the computational complexity reflects the performance of algorithms and can be used for evaluating and camping the efficiency of the different algorithms. The optimization problems can be divided into four computational complexity classes as follows:

P problems

The P problems known as Polynomial-time problems are a class of problems that can be easily tractable and solved using a polynomial deterministic algorithm.

NP problems

NP problems category, which is larger than the P class, includes the set of problems which can be solved using non-deterministic polynomial algorithms. In addition, their solutions can be verified in polynomial time.

NP-Hard problems

NP-hard problems are another category of problem which are non deterministic polynomial time hardness problems. By definition, problem A is classified as an NP-hard problem if and only if an NP-Complete problem A is reducible to B (B is at least hard as A). For the NP-hard problems, they might be some polynomial time algorithms used to solve them.

NP-Complete problems

NP-complete problems present the problems that are NP-hard and belong to the NP problems. The NP-complete problems can also be described as the hardest problems in the NP category. Because, till now, no polynomial time algorithms were found to solve it quickly. However, they can be quickly verified.
Figure 2.2 illustrates the different classes of the optimization problem based on the computational complexity.

2.4 Optimization problem algorithms

In the literature, several algorithms were proposed for solving the optimization problems. According to the algorithms' nature and their search method for the optimal solution, the

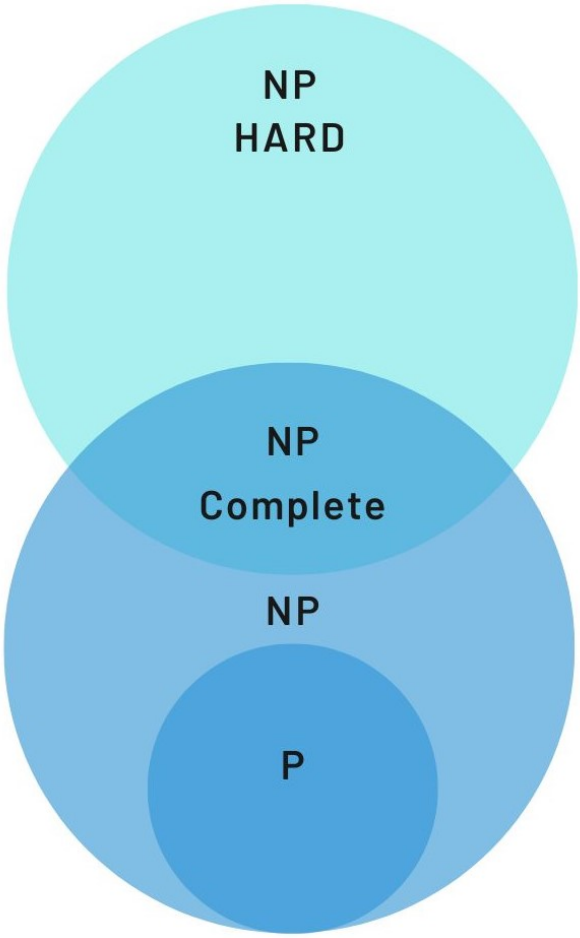


Figure 2.2: Optimization problem classes

optimization algorithms are classified into two main categories, including the classical and the Artificial Intelligence (AI) approaches, as illustrated in Figure 2.3.

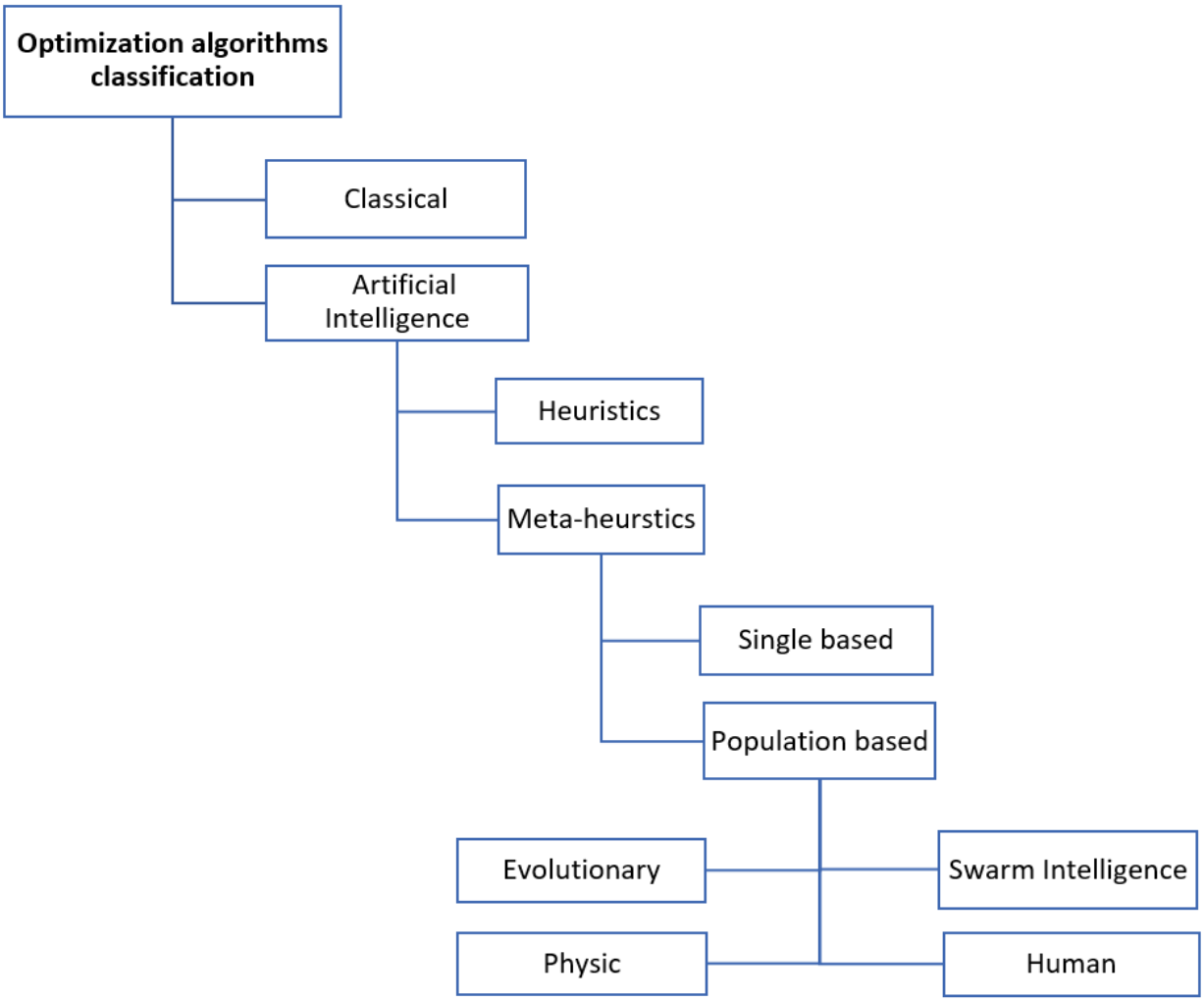


Figure 2.3: Classification of optimization algorithms

2.5 Classical algorithms

The classical algorithms are the first approaches that appear in the literature, used for solving optimization problems. The classical algorithms are also known in the literature as exact approaches which are characterized by their efficiency in terms of finding the optimal solution. However, in large-scale optimization problems, the classical algorithms may take a longer time to determine the optimal solution. In the classical category, we can find several algorithms such as brute force, branch and bound, branch and cut, dynamic programming, integer programming, etc.

2.6 Artificial Intelligence algorithms

Besides the classical algorithms, artificial intelligence or approximate algorithms are also used for solving optimization problems. The approximate algorithms were basically proposed to overcome the lack of the exact algorithms. Among their advantages, the flexibility and cost efficiency. In addition, the AI algorithms can quickly generate the solution. But, their main weakness is the inability to guarantee the optimal solution. The Artificial Intelligence approaches are divided into 2 main sub-classes: Heuristics and meta-heuristics.

2.6.1 Heuristic algorithms

Heuristic algorithms can be defined as a set of approximate algorithms, which depend on the problem characteristics. The heuristic algorithms' nature requires the knowledge of the problem particularities and their adaptation to the problem. Therefore, they are less flexible and cannot be used in large-scale problems. Still, they can generate good solutions in a reasonable time. Among the heuristic algorithms, we have A-Star (A*) algorithm, Greedy algorithm, etc.

2.6.2 Meta-heuristic algorithms

The meta-heuristic algorithms are the most used algorithms in solving optimization problems due to their advantages. Firstly, they are optimal in terms of cost and time. The particularity of meta-heuristic is their problem-independent characteristic which allows them to solve broad range of problems. In addition, they can offer an efficient solution. However, like the heuristics, meta-heuristics cannot guarantee the optimal solution. Based on their behaviour, the meta-heuristics are divided into two main categories: single-based and population-based meta-heuristics.

Single based algorithms

Single-based meta-heuristics are also known in the literature as trajectory algorithms. Their main feature is the generation of a single solution at each run [15]. The single solution meta-heuristics uses the concept of neighbourhood search starting from an initial solution, to determine better solutions. The well-known single-based algorithms are Simulated Annealing (SA), Tabu Search (TS), Hill Climbing, Guided Local Search (GLS), etc.

Population based algorithms

In contrast to single-based meta-heuristics, population-based meta-heuristics can generate multiple solutions at the same run. According the to algorithms nature, population-based methods are divided into four classes as follows:

1. **Evolutionary based**
   The evolutionary-based algorithms are defined as the set of algorithms inspired by the phenomena of evolution in nature such as Darwin's evolution. The process used by evolutionary algorithms in search goes through three main steps: selection, crossover, and mutation. The popular evolutionary algorithms are Genetic Algorithm (GA), Differential Evolution (DE), Evolution Strategy (ES), Bibliography-Based Optimizer (BBO), etc.

2. **Swarm Intelligence based**
   The second population-based meta-heuristic class is the Swarm Intelligence (SI) algorithms, which are inspired by the social behaviour of swarms such as butterflies, birds, bees, fish, ants, etc. The most popular methods of SI algorithms include the following algorithms: Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Glowworm Swarm Optimization (GSO), Grey Wolf Optimization (GWO), Firefly Algorithm (FA), Marine Predator Algorithm (MPA), Aquila Optimizer (AO), etc.

3. **Physic based**
   The physic-based meta-heuristics are another class of algorithms inspired by nature which is basically modelled on the laws and processes of physic. The well-known algorithms in the physic-based class are Artificial Electric Field Algorithm (AEFA), Gravitational Search Algorithm (GSA), Multi-Verse Optimizer (MVO), Henry Gas Solubility Optimization (HGSO), Arithmetic Optimization Algorithm (AOA), etc.

4. **Human based**
   The human-based meta-heuristics are completely different from the others in terms of inspiration's nature. They are inspired by the behaviour of human beings or phenomena related to them rather than nature. In the human-based meta-heuristics, we can find the following algorithms: Teaching Learning Based Optimization (TLBO), Harmony Search (HS), Driving Training-Based Optimization (DTBO), Exchange Market Algorithm (EMA), League Championship Algorithm (LCA), etc.

2.7 Conclusion

This chapter highlights the important points and aspects related to optimization problems that include their mathematical definition, the main characteristics and criteria that affect their nature, concluding with the different approaches that can be used to solve them according to problem type and goals.
In the next chapter, we will point out our overview of the context of the UAV path plan-

ning problem and present the details from objectives and constraints and the methods used previously in the literature.

CHAPTER 3

A SURVEY OF THE UAV PATH PLANNING

3.1 Introduction

Path planning is one of the most important steps in the navigation and control of Unmanned Aerial Vehicles (UAVs). The main objective of UAV path planning is to determine the optimal path that satisfies the UAV requirements such as its resources (power, memory, etc) and safety and considering the flying area conditions from obstacles and forbidden areas, as shown in Figure 3.1 [16].

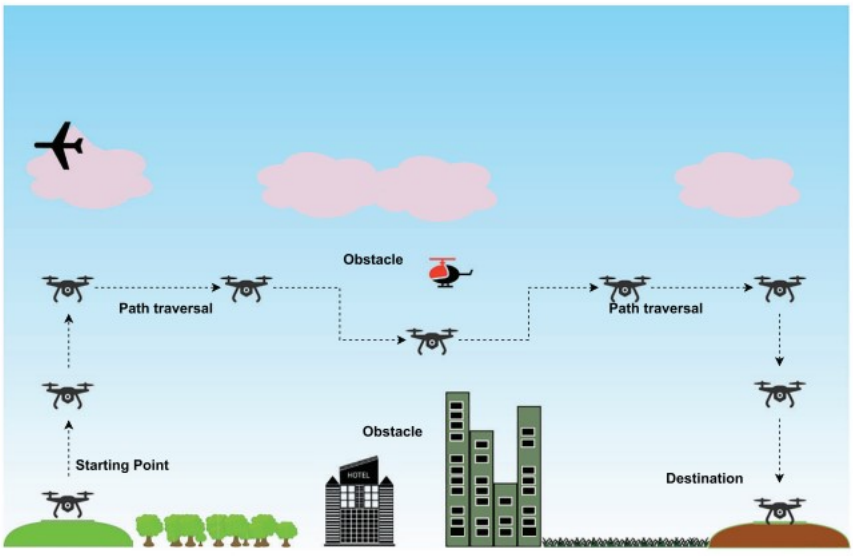


Figure 3.1: The UAV path planning concept

In view of the importance of this topic, we will provide a brief definition of UAV path planning, as well as the different objectives and constraints associated to the problem. In addition, this chapter surveys the methods proposed in the literature in the last two decades as, followed by a deep analysis and cross-comparison between the different approaches.

3.2 UAV path planning steps

Mainly, solving the UAV path planning problem involves four main steps as shown in Figure 3.2. The first step consists of representing the environment area, where the UAV performs its flight. Afterwards, the objectives and constraints related to the subject are formulated into an objective function that can be solved by optimization approaches in the third step. Finally, the best solution provided by the approaches represents the best path found.

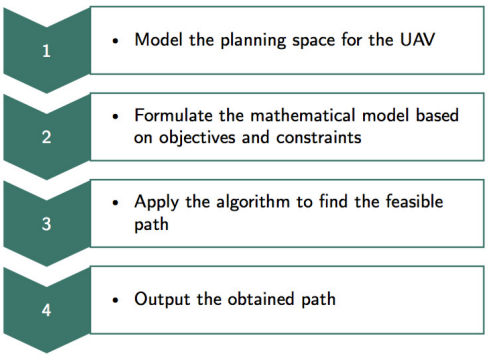


Figure 3.2: The path planning steps

3.3 Objectives

Path optimization

This objective includes the path length and its smoothness. The path length is the travelling distance between the source and destination nodes. Smoothness in fact that the path ensures continuity as long as possible by avoiding turns and obstacles.

Time-efficiency

Time efficiency, often called planning time, is the required time for UAVs to generate the full and optimal path between source and destination points.

Collision avoidance

Collision avoidance is the ability of UAVs to detect and avoid any obstacle in order to move without any physical damage.

Cost-efficiency

This objective represents the sum of multiple computation costs of UAVs such as hardware and software costs, fuel costs, memory costs, battery charging costs, and CPU costs.

3.4 Constraints

Altitude

Altitude is directly related to the safety of UAVs. In fact, when UAVs fly at a low level of altitude, then the number of obstacles increases. Consequently, collision probability increases. On the other hand, flying at a high level of altitude increases the energy consumption of UAVs. The altitude is given in meters.

Climb/descend angles

Climb and descend angles are presented as necessary angles of UAVs for taking off, landing, and moving between multiple heights. They are measured in degrees. Climb and descend angles influence energy consumption, especially for fixed wings.

Energy consumption

Energy consumption is represented as the power consumed by UAVs in terms of fuel, battery, chemical substances, or solar panels. It depends on travelling time, distance, altitude, and also the nature of the served environment. The energy consumption is given in Watts

Obstacles and threats

Obstacles are defined as any objects that interrupt UAV's path. Obstacles can be static (such as buildings, mountains, trees...) or dynamic like moving objects and vehicles. Radars and missiles are considered real threats since the radar can detect UAVs and missile damages by their attacks.

UAV's axes

UAVs in motion performs multiple rotations in three dimensions. These rotations revolve around three main axes, which are the pitch, roll, and yaw angle axes.

Velocity

Velocity is a constraint for UAVs in the context of path planning. It affects directly the fuel consumption and the safety of UAVs during the path-planning process. It is preferred to take into consideration a rational velocity to preserve energy.

3.5 Related Works

In the literature, several path-planning approaches were proposed in the UAV area. We propose in the following a classification based on the type of the used Algorithm. As shown in Figure 3.3, we distinguish five categories: methods based on classical approaches, methods using heuristics, methods using meta-heuristics, those applying machine learning, and hybrid methods.

3.5.1 Classical Approaches

Various classical approaches were developed for solving the UAV path planning problem including Rapid-exploring Random Trees(RRT) [17], Voronoi Diagram (VD) [18], Artificial Potential Field (APF) [19], Visibility Graph (VG) algorithm [20], Dijkstra algorithm [21], and Probabilistic Road Map (PRM) algorithm [22]. They are summarized in Figure 3.4.

Rapid-exploring Random Trees(RRT)

Yang et al. [23] developed a novel approach, called Gaussian Process-based RRT (GP-RRT), based on the integration of the Gaussian Process (GP) map-building model into the RRT algorithm for tackling the UAV path planning problem. In the work of Kothari and Postlethwaite [24], an enhanced RTT algorithm, called Chance Constraint-RRT (CC-RRT), based on the introduction of Chance Constraint approach [25] into RRT algorithm for solving the real-time UAV path planning problem. Lin and Saripalli [26] proposed an improved RTT algorithm based on the integration of Dubin's curves strategy into the RTT algorithm for optimizing the UAV path planning in an indoor environment. Xinggang et al. [27] suggested a Variable Probability-based Bidirectional RRT (VPB-RRT) algorithm for solving the UAV path planning problem. Yang et al. [28] proposed a novel model, called EPF-RRT, based on the integration of Potential Field into the original RTT for solving the UAV path planning problem. Zu et al. [29] suggested an improved RRT (IRRT) algorithm for solving the multi-UAVs trajectory planning problem. Sun et al. [30] developed an Improved RRT (Im-RRT) algorithm based on the integration of dynamic p_g value and dynamic step length in RRT algorithm for optimizing the UAV path planning. Meng et al. [31] proposed a novel approach based on the RRT algorithm for optimizing the UAV path planning. Wen et al. [32] proposed a Heuristic Dynamic Domain RTT (HDDRRT) algorithm based on the combination of Dynamic Domain RTT [33] and an extension of RTT (RRT*) [34] algorithms for enhancing the online UAV path planning. Lee et al. [35] presented a Bidirectional Spline-RRT* (BS-RRT*) algorithm based on the integration of the spline method [36] into the RRT algorithm for solving the UAV path planning problem. Aguilar et al. [37] suggested a novel approach based on the combination of RRT* Goal [37] and RTT* Limit [37] algorithms for the UAV path planning problem. Meng et al. [38] developed an Informed RRT* (IRRT*)

Figure 3.3: The UAV path planning approaches

algorithm based on the integration of the oblique cylinder subset method into the RRT* algorithm for optimizing the UAV path planning. Mechali et al. [39] suggested a Rectified RRT* algorithm based on integrating the smoothing method into RRT* algorithm for solving the UAV path planning problem.

Voronoi Diagram (VD)

Bortoff [40] introduced the Voronoi Diagram algorithm for solving the UAV path planning problem. In another work, Chen et al. [41] developed an improved Voronoi Diagram algo-

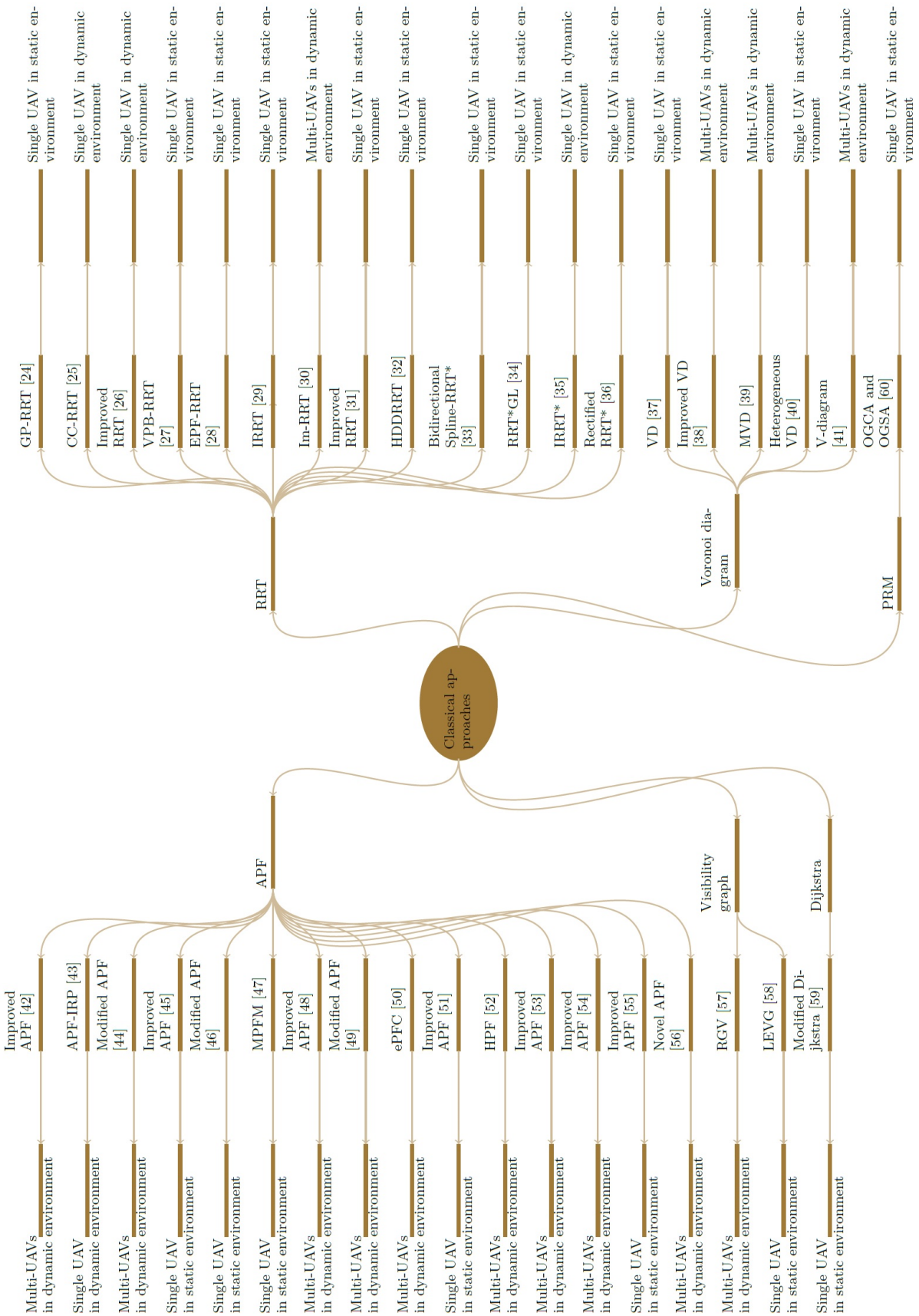


Figure 3.4: Classical approaches

rithm based on the integration of Consistence theory [42] into Voronoi Diagram for solving the multiple UAVs path planning problem. Baek and Han [43] suggested a modified Voronoi Diagram algorithm (mVD) for solving the UAV path planning problem. Feng and Murray [44] proposed a heterogeneous Voronoi Diagram algorithm for solving the UAV path planning problem. Chen and Zhao [45] developed a modified Voronoi Diagram algorithm based on the integration of cubic spline and crowding mechanisms into the Voronoi Diagram algorithm for optimizing multi-UAV path planning in a 2D dynamic environment.

Artificial Potential Field (APF)

An improved APF was proposed by Moon et al. [46], which is the hybrid of APF and A* for solving the multi-UAV path planning problem in a 3D dynamic environment. Qian et al. [47] suggested a novel approach, called APF-Improved Rolling Plan (APF-IRP), for UAV's real-time path planning optimization. Budiyanto et al. [48] proposed a modified APF algorithm for optimizing the Quad-Copter UAV path planning in static and dynamic environments. Chen et al. [49] developed an enhanced APF approach based on the integration of optimal control method into APF for solving the UAV path planning problem. In their proposal, Liu and Zhao [50] suggested a modified APF technique based on the integration of virtual waypoint into the APF algorithm for solving the UAV path planning problem. Mac et al. [51] proposed a Modified Potential Field Method (MPFM) for solving the UAV path planning problem. Authors in [52] suggested an improved APF algorithm based on the integration of selective avoidance strategy into the APF algorithm for optimizing the multi-UAV path planning. Sun et al. [53] proposed an enhanced APF algorithm for solving the multi-UAVs path planning problem. Distance factor and Jump strategy approaches were integrated into APF algorithm to avoid the local minimum point. Woods and Hung [54] suggested an extended APF Controller (ePFC) for optimizing the UAV tracking mission in a 3D dynamic real environment. Zhiyang and Tao [55] developed an improved APF algorithm based on the introduction of the advanced search method in APF algorithm for solving the UAV path planning problem. Dai et al. [56] proposed a Hierarchical Potential Field (HPF) approach for optimizing multi-UAVs path planning. Authors in [57] developed an improved approach based on the integration of B-spline Interpolation strategy [58] into APF algorithm for multi-UAVs path planning problem. Feng et al. [59] proposed an Improved APF based on the introduction of the Formation Control method into the standard APF algorithm for UAV tracking missions and collision avoidance. In the proposal of Yingkun [60], the author presented an enhanced APF algorithm for solving the Agriculture UAV path planning problem. Abeywickrama et al. [61] proposed a modified APF algorithm for handling the multiple UAVs path planning problem. A virtual target point is created to increase attractive force which pushes UAVs in altitude to avoid the obstacles in 3D flight.

Visibility Graph (VG)

D'Amato et al. [62] developed a Bi-level optimization algorithm based on the Visibility graph algorithm for solving the cooperative UAV path planning problem in a 2D dynamic environment. In their work, D'Amato et al. [63] proposed a Layered Essential Visibility Graph (LEVG) algorithm based on the integration of Dubins curves into a Visibility graph algorithm for solving the fixed-wing UAV path planning problem in a 3D environment.

Dijkstra algorithm

Maini and Sujit [64] applied a novel approach based on the Dijkstra algorithm for enhancing the UAV path planning in a complex environment.

Probabilistic Road Map (PRM)

Wang et al. [65] proposed an Obstacle-free graph construction algorithm (OGCA) and Obstacle-free graph search algorithm (OGSA) for solving the UAV path planning problem. The two algorithms are improved versions of PRM* [34] and A* [66] algorithms respectively.

3.5.2 Heuristic Approaches

For solving the UAV path planning, many heuristic-based algorithms were proposed as shown in Figure 3.5.

A-Star Algorithm

In path planning, A-Star (A*) algorithm is a popular heuristic algorithm, it was firstly introduced by Hart et al. [67].

Dong et al. [68] proposed a Virtual Force A∗ (HVFA) algorithm based on the introduction of virtual force method [69] into A∗ algorithm for solving the UAV path re-planning problem. The performance of HVFA algorithm was assessed in 4 experiences. Geng et al. [66] suggested A∗ algorithm for solving multi-UAVs path planning problem. The effectiveness of A∗ was evaluated in a 3D environment. Wang et al. [70] proposed an improved A∗ algorithm, called Dubins-Sparse A-Star (Dubins-SAS) algorithm, based on the integration of Dubins curve into SAS algorithm [71] for optimizing the UAV path planning. In their proposal, Tianzhu et al. [72] suggested an improved A∗ algorithm for optimizing the UAV path planning in a 3D environment. Zhang and Meng [73] developed a Sparse A∗ Search (SAS) algorithm for solving the UAV path planning problem. Chen et al. [74] proposed an A∗ algorithm for optimizing UAV path planning. Zhang and Hsu [75] developed an improved path planning algorithm based on the integration of GNSS error distribution into A∗ algorithm for solving Global Navigation Satellite System localization map error for UAVs. Primatesta et al. [76] proposed a RiskA∗ algorithm for optimizing the UAV path planning in an urban

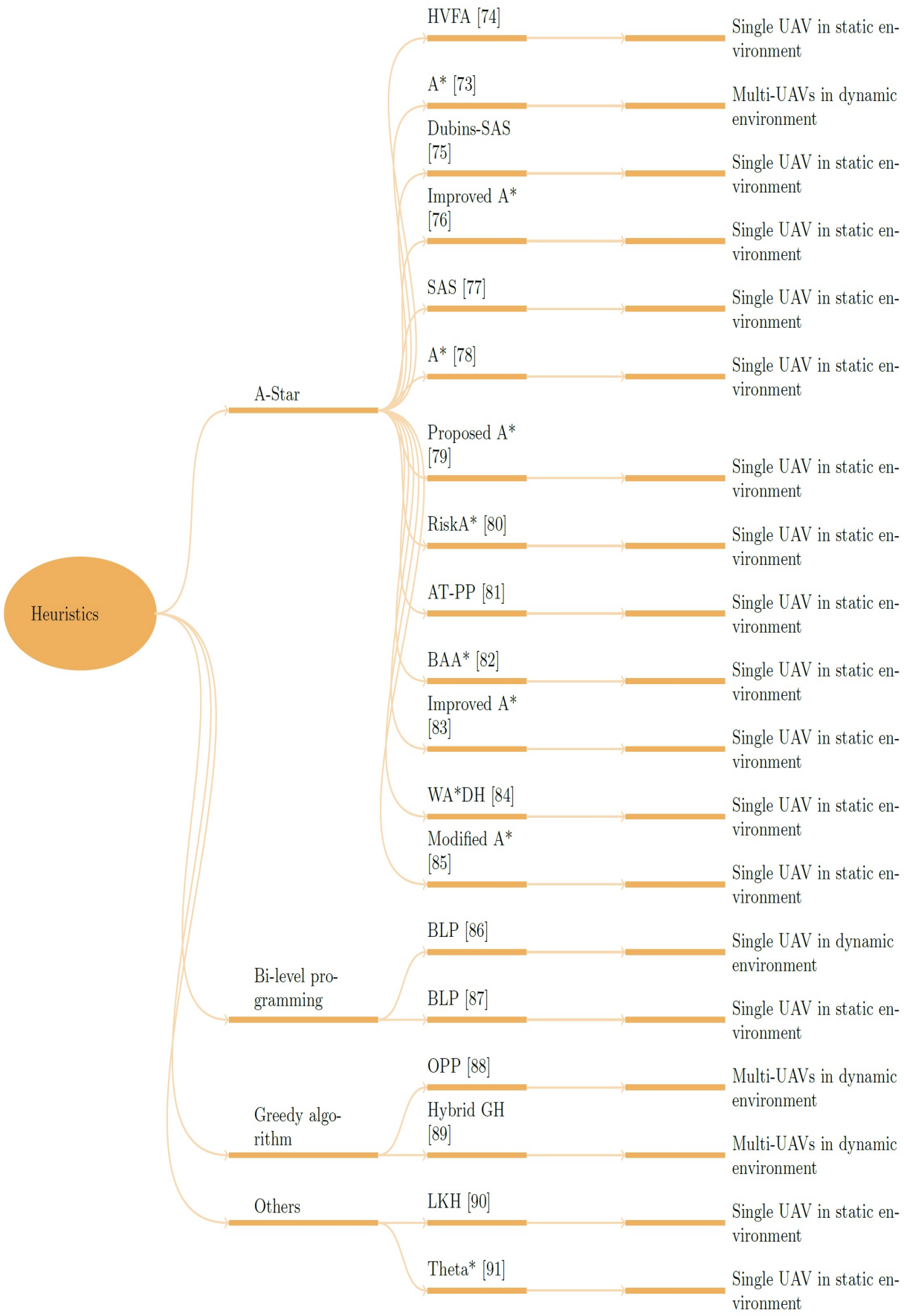


Figure 3.5: Heuristic approaches

environment. Madani et al. [77] developed three approaches based on the A∗ algorithm for optimizing Quality of Service (QoS) in UAV path planning. Average Throughput-Path Planning (AT-PP) and Maximum Throughput-PP (MT-PP) algorithms were designed for QoS processing, while Improved Path Smoothing (IPS) was used for path planning optimization. Wu et al. [78] proposed a Bi-directional Adaptive A∗ (BAA∗) algorithm for solving the UAV path planning problem. Directional, adaptive step and adaptive weight search strategies were introduced to improve the expansion process, path smoothness, and exploration speed. Zhang et al. [79] proposed an Improved A∗ algorithm, called the Learning Real-Time A-star algorithm (LRTA-Star) based on the combination of Model-based predictive control [80] and A∗ algorithm for real-time penetration path planning.

Bi-Level Programming algorithm

In the work of Liu et al. [81], authors proposed an improved Bi-Level Programming (BLP) algorithm based on the integration of multiple optimization strategies into BLP algorithm [82] for solving the real-time UAV path planning problem. Kang et al. [83] proposed a BLP algorithm for solving the UAV path planning problem. The performance of the BLP algorithm was validated in a 2D environment.

Greedy algorithm

Ahmed et al. [84] proposed an Optimal Path Planning (OPP) algorithm based on the hybridization of two variants of the greedy algorithm, called, Greedy Least Cost (GLC) and First Detect Fist Reserve (FDFR) to enhance the energy consumption for the UAV path planning. In the work of Silva et al. [85], a Greedy heuristic (GH) algorithm was proposed for solving the UAV path planning problem.

Others

Fritas et al. [86] proposed Lin–Kernighan heuristic (LKH) algorithm for improving the UAV path planning in biological pest control applications. In their proposal, De Filippis et al. [87] developed a novel heuristic approach as an extension of the A∗ algorithm, called the Theta∗ algorithm, for optimizing the UAV path planning problem in a 3D environment.

3.5.3 Meta-heuristic Approaches

Meta-heuristic algorithms are widely used for optimizing the UAV path planning as shown in Figure 3.6. They are divided into two main categories: single-based and population-based approaches.

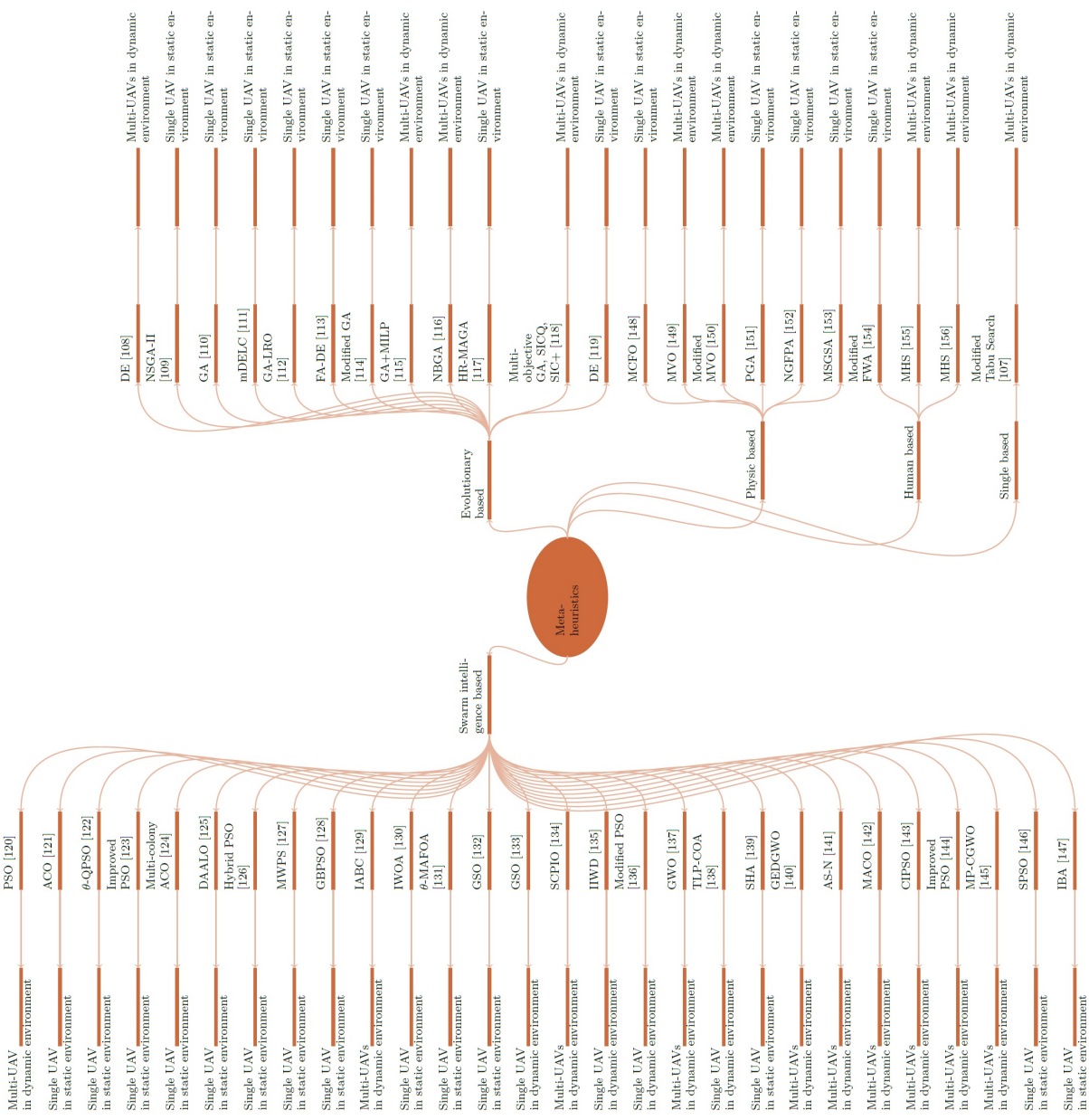


Figure 3.6: Meta-heuristic approaches

Single-based approaches

Du et al. [88] presented a modified Tabu search algorithm based on the integration of the Nawaz-Enscore-Ham (NEH) method [89] into Tabu Search for solving the multiple UAVs path planning problem.

Population-based approaches

1. **Evolutionary based**
   In their proposal, Brintaki and Nikolos [90] suggested a Differential Evolution algorithm (DE) [91] for solving the multi-UAVs path planning problem. An improved Non-dominated Sorting Genetic Algorithm (NSGA-II) based on the introduction of the B-spline method into NSGA-II [92] for solving the UAV path planning problem was proposed by Mittal and Deb [93]. Roberge et al. [94] used GA for solving the fixed-wing UAV path planning problem. Authors in [95], presented a novel approach, called improved Differential Evolution (mDELC), based on the integration of improved Level Comparison strategy into Differential Evolution algorithm. Li et al. [96] proposed a path planning technique, called Genetic Algorithm-Local Rolling Optimization (GA-LRO), based on the introduction of Local rolling mechanism into GA and for optimizing the UAV path planning. Adhikari et al. [97] suggested a Fuzzy Adaptive Differential Evolution algorithm (FA-DE) based on the integration of Fuzzy logic mechanism [98] into Differential Evolution for solving the UAV path planning problem. Authors in [99] developed an improved GA algorithm for optimizing the UAV path planning. A novel approach based on the integration of Mixed Integer Linear Programming (MILP) into GA algorithm for improving the UAV path planning in a complex environment was proposed by Dai et al. [100]. A novel approach, called Neighborhood Based Genetic Algorithm (NBGA), was proposed by Xiao et al. [101] for solving multi-UAVs dynamic path planning and UAV/UGV coordination. Yang et al. [102] developed a Hierarchical Recursive-Multi Agent Genetic Algorithm (HR-MAGA) for solving the UAV path planning problem. In the work of Hayat et al. [103], authors proposed two approaches, called Simultaneous Inform and Connect with QoS (SICQ) and SIC following QoS (SIC+) algorithms, for optimizing the UAV path planning. Chawra and Gupta [104] suggested DE algorithm for optimizing multi-UAVs path planning for data collection in cluster-based Wireless Sensor Network.

2. **Swarm Intelligence based**
   Fu et al. [105] presented a novel variant of Particle Swarm Optimization called Phase-encoded Quantum Particle Swarm Optimization algorithm (θ-QPSO) based on the combination of Phase-encoded PSO (θ-PSO) [106] and Quantum PSO (QPSO) [107] for solving the UAV path planning problem. In the work of Liu et al. [108], authors proposed an improved Particle Swarm Optimization (PSO) algorithm based on the

introduction of Adaptive Sensitive Decisions into the PSO algorithm for solving the UAV path planning problem in a 3D static environment. Cekmez et al. [109] suggested a Multi-Colony Ant Colony Optimization (Multi-colony ACO) algorithm, for solving the UAV path planning problem. Yao and wang [110] proposed an improved Ant Lion Optimizer, called, Dynamic Adaptive Ant Lion Optimizer (DAALO), for solving the UAV path planning problem. Wu et al. [111] proposed a modified PSO algorithm based on PSO and Bezier curves model for optimizing real-time UAV path planning. Yongbo et al. [112] proposed a Modified Wolf Pack Search (MWPS) algorithm for solving the 3D UAV path planning problem. Huang and Fei [113] developed a Global Best Particle Swarm Optimization (GBPSO) algorithm for solving the fixed-wing UAV path planning problem. Tian et al. [114] suggested an Improved Artificial Bee Colony (IABC) algorithm for Multi-UAVs dynamic tracking planning. Wu et al. [115] suggested an Improved Whale Optimization Algorithm (IWOA) and Restrained Interfered Fluid Dynamic System (RIFDS) for solving the UAV path planning problem. An approach called, θ-Mutation Adaptation Fruit Fly Optimization Algorithm (θ-MAFOA) was proposed by Zhang et al. [116], which is based on the integration of mutation adaptation mechanism and phase angle-based encoded strategy into Fruit Fly Optimization Algorithm (FOA) for solving the UAV path planning problem. In [117], Pandey et al. proposed a path planning approach based on Glowworm Swarm Optimization (GSO) [118] solving the UAV path planning problem. Goal et al. [119] developed a path planning technique based on GSO algorithm for solving the UAV path planning in a 3D dynamic environment. Zhang et al. [120] presented a novel bio-inspired technique, called Social-Class Pigeon Inspired Optimization (SCPIO) algorithm, for optimizing multi-UAVs coordination and path planning. In another work, Sun et al. [121] proposed an Improved Intelligent Water Drop Algorithm (IIWD) for solving the UAV path planning problem. In the work of Muliawan et al. [122], an improved PSO algorithm was proposed for optimizing the UAV path planning in autonomous Spraying Task application. Dewangan et al. [123] used Grey Wolf Optimizer (GWO) algorithm for solving the multiple UAVs path planning problem in a 3D complex environment. Cai et al. [124] proposed a Tri-Level Programming-Cognitive behaviour Optimization Algorithm (TLP-COA) hybridizing COA [125] and TLP solution for solving the real-time UAV path planning problem. In another work, a Self-Heuristic Ant was proposed by Zhang et al. [126], which is based on Ant-Colony Optimization for solving the UAV path planning problem. In another work, Wang et al. [127] suggested an improved Grey Wolf Optimization (GWO), called Gaussian Estimation of Distribution Grey Wolf Optimizer (GEDGWO), based on the integration of Gaussian Estimation of Distribution (GED) strategy into GWO for solving the multi-UAV multi-target urban tracking problem. Yue and Chen [128] proposed a hybrid approach, called ant colony algorithm with punitive measures (AS-N) based on the integration of penalty strategy on Ant

Colony Optimization Algorithm for optimizing the UAV path planning. In a similar work, Li et al. [129] developed a novel approach, termed Modified Ant Colony Optimization (MACO), which joined metropolis criterion to ACO algorithm for multi-UAVs path planning. Shao et al. [130] suggested a Comprehensively Improved PSO (CIPSO) algorithm for solving Multi-UAVs path planning problem. In another work, Liu et al. [131] presented a modified PSO algorithm based on the introduction of Spatial Refined Voting Mechanism into PSO for solving multi-UAVs path planning problem. Yang et al. [132] proposed an improved GWO algorithm, called Multi Population-Chaotic GWO (MP-CGWO), for solving the multi-UAVs path planning problem.

3. **Physic-based techniques**
   Chen et al. [133] developed a Modified Central Force Optimization (MCFO) algorithm for solving the UAV path planning problem. Similarly, Kumar et al. [134] proposed a Multi-Verse Optimizer (MVO) algorithm for enhancing the Quality of Service (QoS) in the UAV path planning. In their proposal, Jain et al. [135] proposed a modified MVO algorithm for solving the UAV path planning problem.

4. **Human based**

Authors in [136] proposed a Modified Firework Algorithm (FWA) based on the introduction of new feasibility rules into FWA for solving the UAV path planning problem. Wu et al. [137] developed an improved path planning algorithm, called the Modified Harmony Search algorithm (MHS), based on the integration of the Pythagorean Hodograph (PH) curve into the Harmony Search algorithm for optimizing the multiple UAVs path planning. In the work of Binol et al. [138], authors presented a modified Harmony Search algorithm for solving multi-UAVs path planning problem.

3.5.4 Machine learning Approaches

Neural Network

Nikolos et al. [139] proposed a Radial Basis Functions Artificial Neural Network (RBF-ANN) algorithm for optimizing the UAV path planning. In the proposal of Wu et al. [140], authors presented an improved Deep Q-Network (DQN) model based on the integration of the Lazy training method into the deep Q-network [141] technique for optimizing the UAV path planning. Yan et al. [142] proposed an improved technique, called Dueling Double Deep Q-networks (D3QN) algorithm, based on deep Q-networks algorithm for optimizing the UAV path planning. Shiri et al. [143] proposed a neural network-based Opportunistic Hamilton-Jacobi-Bellman (oHJB) approach for solving the remote UAV online path planning problem.

Supervised learning

Chen et al. [144] presented a novel approach based on Support Vector Machine (SVM) for solving the UAV path planning problem. Yoo et al. [145] suggested Gaussian Process (GP) regression for optimizing the UAV path planning. In their proposal, Koo et al. [146] applied the polynomial regression model for optimizing the UAV path planning in data collection missions. Radmanesh et al. [147] suggested a high-dimensional regression technique with Partial Differential Equation (PDE) for optimizing the multiple UAVs path planning. The proposed technique adopted 3 models for multi-UAVs control including centralized, decentralized, and sequential models.

Reinforcement learning

Ragi and Chong [148] suggested an improved Partially Observable Markov Decision Process (POMDP) based on the integration of Nominal Belief-state Optimization strategy (NBO) into POMDP approach for optimizing the UAV path planning in a dynamic environment. Zhang et al. [149] developed a novel approach, called Cooperative and Geometric Learning Algorithm (CGLA), for optimizing multiple UAV path planning. Yijing et al. [150] proposed an Adaptive and Random Exploration (ARE) approach based on a Q-learning algorithm for optimizing the UAV path planning. Authors in [151] presented a Deep Reinforcement Learning Echo State Network (Deep-RL-ESN) model based on the integration of Deep ESN (DESN) into RL algorithm for solving the multi-UAVs online path planning. Luo et al. [152] suggested Deep-State action reward state action (Deep-Sarsa) algorithm based on the reinforcement Deep-Sarsa learning approach for optimizing the UAV path planning. Yan and Xiang [153] proposed an enhanced Q-learning algorithm based on the integration of ε-greedy and Boltzmann approaches into the Q-learning algorithm for solving the UAV path planning problem. In their proposal, Zhang et al. [154] suggested a novel approach based on the integration of state machine and differential-equation models into a Q-learning algorithm for optimizing the QuadRotor UAV path planning. In the work of Xie et al. [155], authors developed an improved Q-learning algorithm, called MARER Q-learning, based on the integration of the Experience Relay function into the Q-learning algorithm for solving the UAV path planning problem.

Unsupervised Learning

In the proposal of Pierre et al. [156], authors proposed an improved Kohenen's Self-Organizing Map (SOM) technique based on the hybridization of competitive learning and particle physics for optimizing the UAV path planning. Choi et al. [157] presented Two-layer approaches based on clustering and Predictive Control models for optimizing fixed-wing UAV path planning.

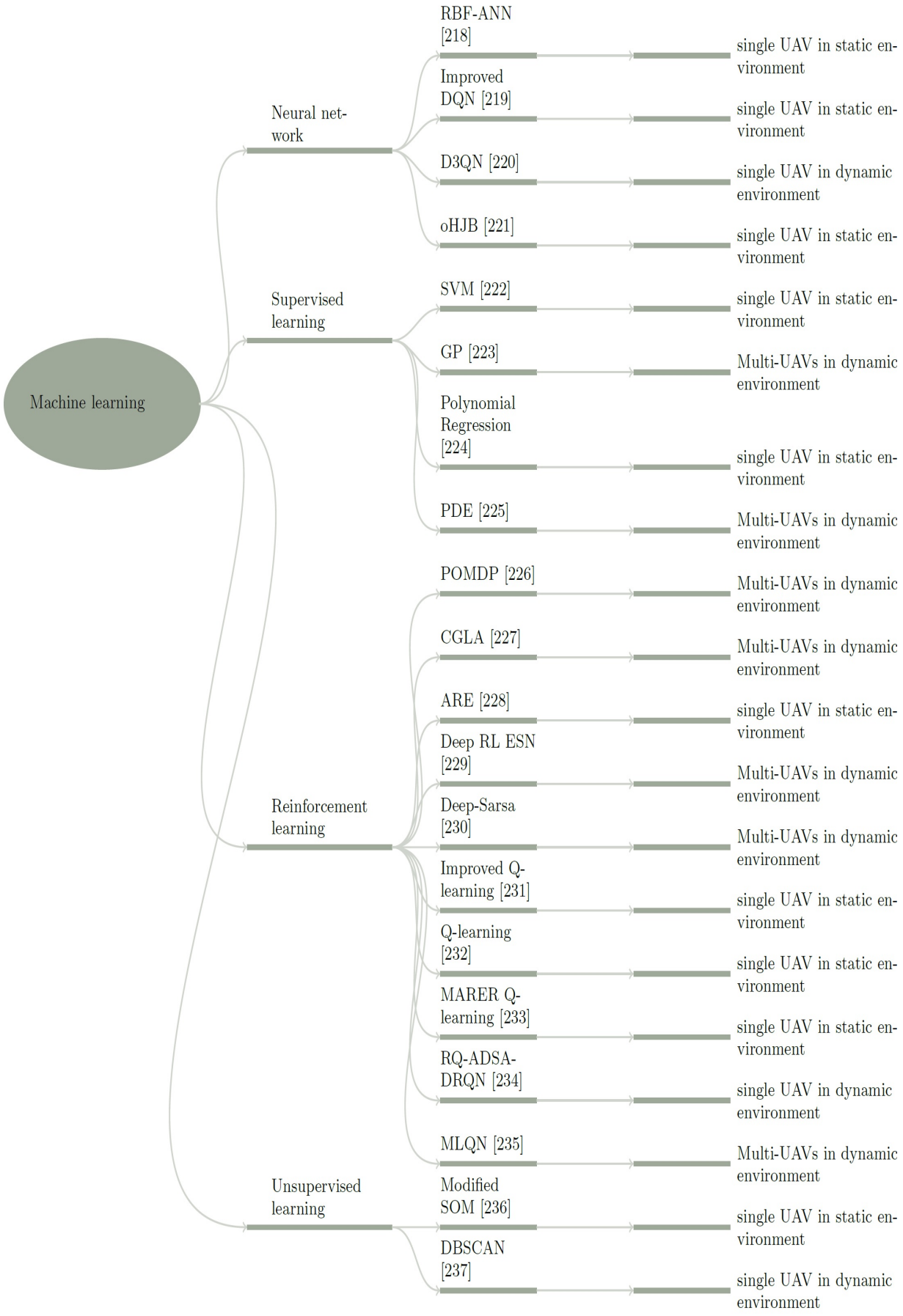


Figure 3.7: Machine learning approaches

3.5.5 Hybrid Approaches

Classical with classical approaches

Authors in [158] proposed a dynamic improved Voronoi Diagram algorithm based on the combination of the Voronoi Diagram and Dijkstra algorithm for solving the UAV path planning problem. An Improved RRT-Connect (IRRTC) algorithm was proposed by Zhang et al. [159], which is based on the hybridization of RRT-Connect with an Artificial Potential Field algorithm for optimizing the UAV path planning in a 2D static environment. Wang et al. [160] proposed an improved RTT (i-RRT) algorithm based on the combination of RRT with APF and the curve smoothing method for optimizing the UAV path planning in a 3D environment. In their work, Shen and Li [161] developed an improved APF algorithm based on the combination of APF and RRT algorithms for solving the UAV path planning problem. Similarly, Debnath et al. [162] proposed an Elliptical Concave Visibility Graph (EC*o*VG) model based on the combination of VG and Dijkstra algorithms for solving the UAV path planning problem.

Heuristics with Classical approaches

Chandler et al. [163] developed an improved Voronoi Diagram algorithm by combining the Voronoi Diagram algorithm with A* and Rendezvous approach for optimizing the Multi-UAVs path planning. In their proposal, Yan et al. [164] developed an improved PRM algorithm aggregating original PRM with octree and A* algorithms for solving the UAV path planning problem in a 3D environment. Xue et al. [165] suggested a hybrid approach, called D* lite-IPRM, hybridizing PRM with D* lite algorithm for optimizing the UAV path planning in both 2D and 3D environments. Ahmad et al. [166] proposed a Visibility roadmap algorithm based on the combination of visibility Graph algorithm with A* heuristic for optimizing the path planning for Small UAVs (SUAVs). In a similar work, Naazare et al. [167] suggested a hybrid approach based on the Visibility Graph algorithm and A* algorithm. The effectiveness of the proposed algorithm was evaluated using AirRobot AR200 in both simulation and real area. Experimental results showed that the proposed algorithm provides an optimal path.

Meta-heuristics with Classical approaches

Qu et al. [168] hybridized Dijkstra, APF, and GA algorithms for solving the fixed-wing UAV path planning problem. The hybrid method used the Dijkstra algorithm to seek the shortest path, APF algorithm aimed to find the feasible path by guiding the UAV toward a smooth area with avoiding threats, and the purpose of the GA algorithm is to enhance the path. Pehlivanog [169] developed a modified Multi-frequency Vibrational Genetic Algorithm (mVGA^v) based on the hybridization of the multi-frequency Genetic Algorithm and

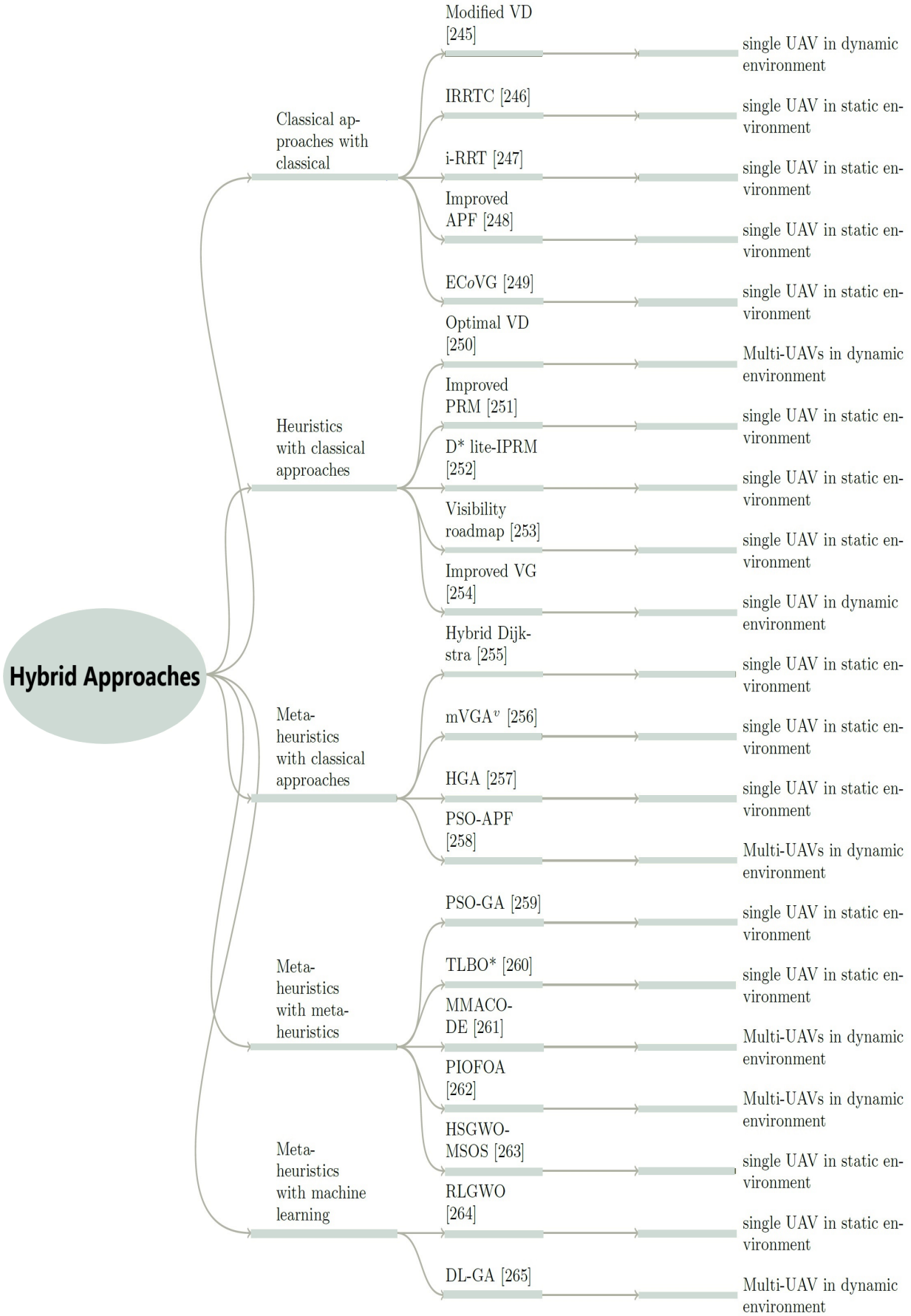


Figure 3.8: Hybrid approaches

Voronoi Diagram model for solving the UAV path planning problem. A hybrid technique, called Hybrid Genetic Algorithm (HGA), was introduced by Arantes et al. [170], which is based on the combination of GA and Visibility Graph for solving the UAV path planning problem. Girija and Ashok [171] proposed a novel algorithm, called PSO-APF, based on the combination of PSO and APF algorithms for optimizing UAV path planning.

Meta-heuristics with meta-heuristics

In [172], a hybrid PSO-GA was proposed in an attempt to do real-time UAV path planning problems. In the work of Ghambari et al. [173], authors developed a novel approach, called TLBO* based on the combination of Teaching Learning-based Optimization (TLBO) with Genetic Algorithm for solving the UAV path planning problem. Authors in [174] proposed a hybrid approach, called Maximum-Minimum Ant Colony Optimization-DE algorithm (MMACO-DE), based on the combination of ACO and DE for solving the multi-UAVs path planning problem in a dynamic environment. Ge et al. [175] presented a modified Pigeon-Inspired Optimization, called, Pigeon Inspired optimization Fruit fly optimization algorithm (PIOFOA) based on the combination of Pigeon-inspired optimization (PIO) and Fruit fly optimization algorithm(FOA) for solving the UAV path planning problem in a 3D dynamic environment. In another work, Qu et al. [176] proposed a hybrid path planning algorithm (HSGWO-MSOS) based on the hybridization of Simplified GWO with a Modified Symbiotic Organism Search for solving the UAV path planning problem.

Meta-heuristics with machine learning

Qu et al. [177] hybridized GWO and Reinforcement Learning model for solving the UAV path planning problem.

3.5.6 Discussion

From this study, important points can be summarized as follow:
Figures 3.9, 3.10 and 3.11 give respectively a synthesis of the addressed environments, the evaluated metrics as well as the proportions at which the different approaches are used.
We can see that the static environment is the one that is mostly taken into account. This is expected because of the regularity it offers. The use of dynamic and multi-UAVs environments are not negligible either and many propositions consider them.

Regarding the metrics, it seems that in the UAV path planning problem, the path quality is more priority than the cost it could generate. In fact, as shown in Figure 3.10, the most evaluated measures are path optimization and collision avoidance.

Various approaches were used for solving the UAV path planning problem since 2000. As shown in Figure 3.11, 34.3% of them are based on Meta-heuristics, 25.3% used classical techniques, whereas hybrid approaches, machine learning, and heuristics are used in only

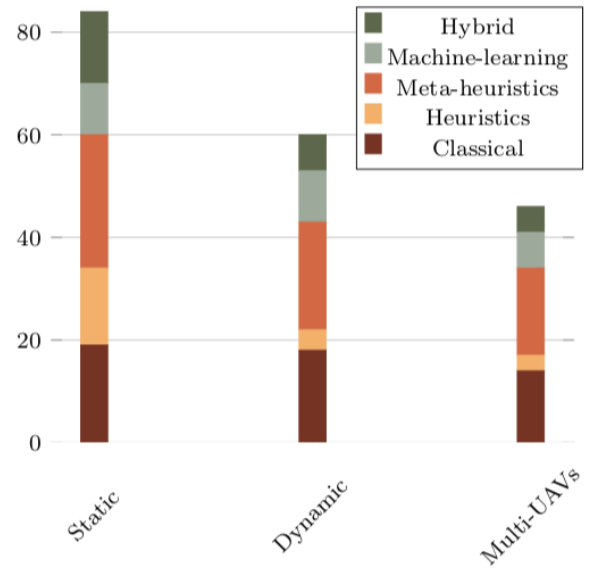


Figure 3.9: Experimented environment of UAV path planning

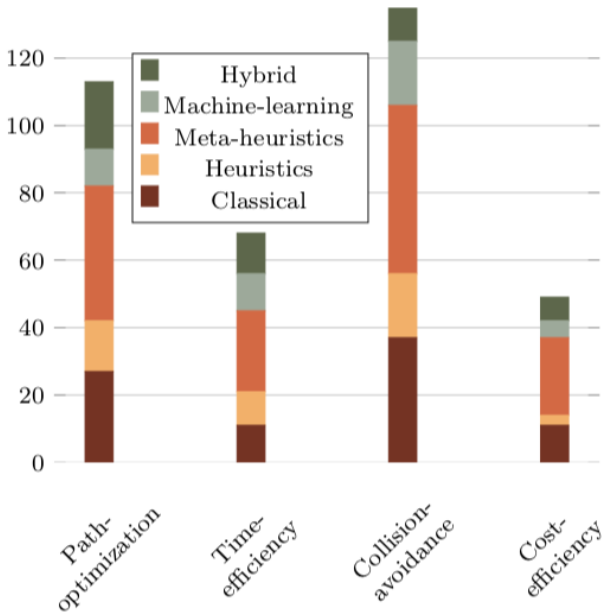


Figure 3.10: Optimization results of UAV path planning

14.5% and 13% of cases, respectively. The strengths and weaknesses of each approach are given in Table 3.1 where we can notice the following results:

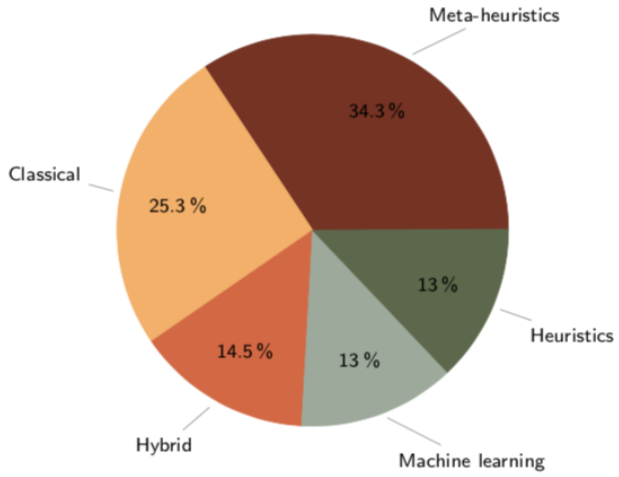


Figure 3.11: The optimization approaches for UAV path planning

Several classical approaches were applied to solving the UAV path planning problem such as RRT, VD, APF, VG, Dijkstra, and PRM. The main reason for their success is their easy implementation while giving good results (Path optimization, fast solution generation, and time efficiency) for static environments with simple obstacles. This makes them suitable and reliable for real-time planning. However, no theoretical guarantee is provided about the optimality of the given solutions due to the presence of constraints. Moreover, classical algorithms require full information about the environment to be performed. They provide poor performance in complex and dynamic environments with different threats and obstacles. Heuristic algorithms appeared to overcome the drawbacks of classical approaches. They are partially intelligent and require only some information about the environment to be performed. Heuristics generate a short and safe path for UAV path planning with a reasonable computational time. Heuristics are more optimal in static and simple environments. Nevertheless, they do not respond well in the case of dynamic and complex environments. Moreover, heuristics return generally bad solutions for UAV path planning problem in the case of multi-objective resolutions. To deal with the heuristic limits faced in multi-objective cases, meta-heuristics are the alternative which is widely used in UAV path planning. They are known to present many advantages as they are intelligent approaches which makes them appropriate for dynamic and complex environments. In UAV path planning problem, meta-heuristics give good quality solutions (path optimization: length, smoothness, cost, energy, and computational time) for complex and dynamic environments while handling UAV's constraints. Despite their advantages, meta-heuristics remain not suitable for real-time path planning.

Machine learning algorithms are also appropriate for complex and dynamic environments since they are intelligent approaches and their behaviour is near to human one. Although

Table 3.1: Strengths and Weaknesses of UAV path planning approaches

Approach	Strengths	Weaknesses
Classical	– Good results (Path optimization, fast solution generation, and time-efficiency) for static environments with simple obstacles – Easy implementation for various static environments – Low resources requirements and low computational cost	– Failing in local minima problems – No guarantee about the optimality of results due to the presence of constraints – Poor performance in complex and dynamic environments with different threats and obstacles – Requiring the full information about the environment to be performed
Heuristics	– Good solutions for UAV path planning with mono-objective resolutions – Appropriate for static environments under different constraints – Partially intelligent – Reasonable response time	– Not always reaching optimal solutions – Bad solutions for UAV path planning with multi-objective resolutions – Stuck in local optima – Not suitable for dynamic and complex environments
Meta-heuristics	– Good results for UAV path planning with multi-objective resolutions – Reasonable execution time – Intelligent approaches with easy implementation – Good quality solutions (path optimization: length, smoothness, cost, energy , and computational time) for complex and dynamic environments	– Optimal solutions not guaranteed – Not suitable and reliable for real time planning – Control parameters tuning – More resources requirements – No theoretical converging property
Machine learning	– Optimal results (path length, smoothness, time efficiency, and collision avoidance) – Intelligent approaches – Suitable for dynamic environments and sudden changes	– Requirement of large datasets of the environment to provide optimal solutions, especially with supervised learning – Requirement of several training before providing an accurate final model. – High computational time – High computational cost

machine learning algorithms present several advantages, they have some weaknesses. Firstly, the time required for processing is higher, so it is not appropriate for real-time planning. Secondly, high execution time requires high resource consumption which impacts the cost of the applied approach. Thirdly, Machine learning approaches require large datasets of the environment to provide optimal solutions, especially with supervised learning.

3.6 Conclusion

This chapter presents a survey in the context of UAV path planning which includes its main objectives, constraints, and the different approaches used in the literature.
Based on the analysis and considering the advantages and limitations presented by each class of algorithms, we will present in the next chapter our first approach, which is a meta-heuristic based method to solve the UAV path planning problem.

CHAPTER 4

AN IMPROVED APPROACH FOR SOLVING UAV PATH PLANNING PROBLEM

4.1 Introduction

This chapter describes the details of solving the UAV path planning problem in 3D environment using our improved approach, called the Chaotic Cauchy Opposition based African Vulture Optimization (CCO-AVOA) algorithm. The proposed CCO-AVOA approach integrates three major strategies: Logistic chaotic function, Elite Opposition Based Leaning (EOBL), and Cauchy mutation strategies. These techniques aim to improve the diversity of the original AVOA algorithm and its efficiency in achieving global optima.
At the beginning of this chapter, we selected and presented the environment model for UAV path planning. Afterwards, we formulated the UAV path planning problem into an objective function that requires to be optimized. Then, we reviewed the African Vulture Optimization Algorithm (AVOA), Logistic chaotic function, Elite Opposition Based Leaning (EOBL), and Cauchy mutation strategies. Finally, we conclude this chapter with the evaluation results of our proposed CCO-AVOA algorithm compared to different meta-heuristics.

4.2 Environment modeling

In this work, we consider a 3D area of $[L, W, H]$ where the UAV performs its tasks. The UAV starts its flight from a source node S located at (x_s, y_s, z_s) to reach its final destination T at the location (x_t, y_t, z_t). In our model, as shown in Figure 4.1, we consider several obstacles and threats O_j present in the area and located at $(x_o^j, y_o^j, z_o^j), \quad j = \{1, 2, \ldots, m\}$. Mathematically, they can be represented as follows:

$$(x - x_o^j)^2 + (y - y_o^j)^2 + (z - z_o^j)^2 = R^j \tag{4.1}$$

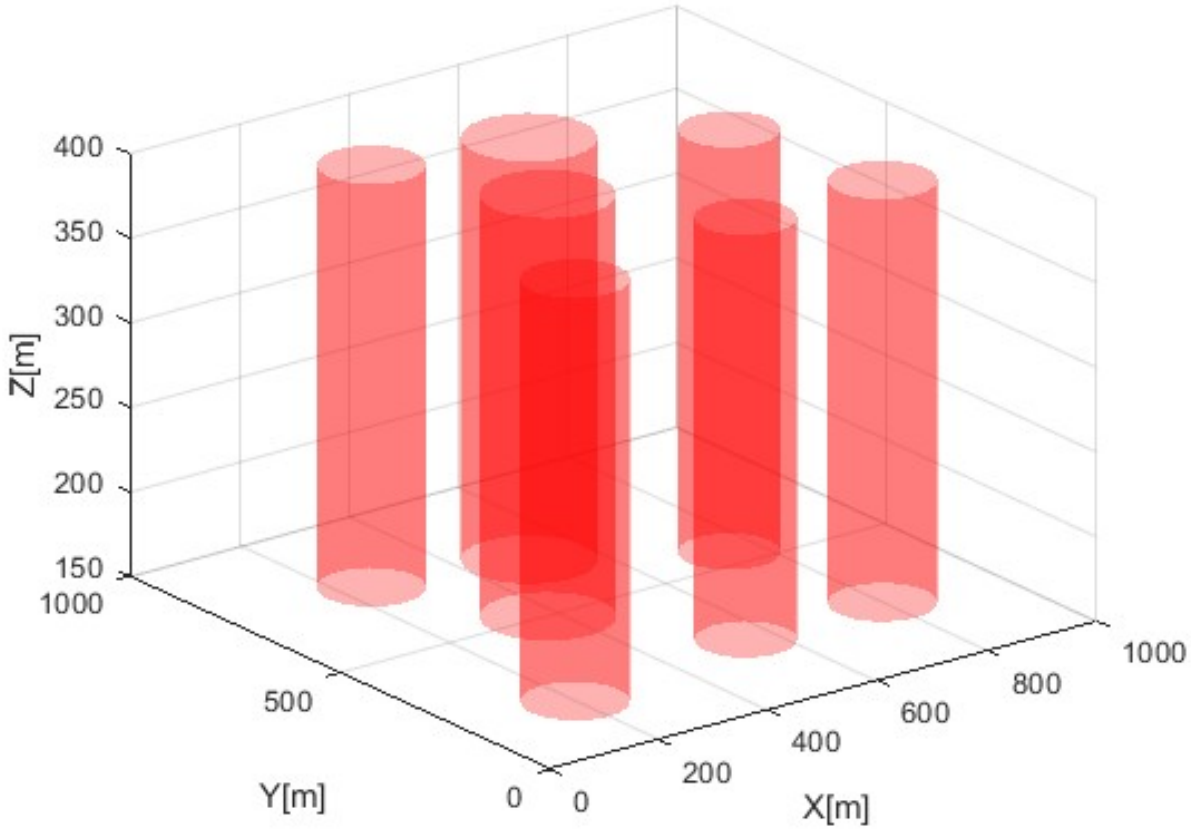


Figure 4.1: Environment model in 3D

where R^j represents the *j-th* obstacle's radius. The goal of the path planning is to find a set of n feasible waypoints $W = \{W_1, W_2, \ldots, W_n\}$. Each node W_i takes a 3D position at (x_i, y_i, z_i). To form the path, every two adjacent waypoints are linked forming $(n-1)$ segment lines $\overrightarrow{\mathrm{W_iW_{i+1}}}$. The total path formed from the source to the target nodes can be expressed as follow: $P = \{S, (x_1, y_1, z_1), \ldots, (x_n, y_n, z_n), T\}$.

To limit the search space and handle better the UAV's rotations, we convert the original coordination system into a spherical one. Every segment line $\overrightarrow{\mathrm{W_iW_{i+1}}}$ is represented as a vector with magnitude r_i and directions (θ_i, ϕ_i) along the azimuth and elevation planes, respectively. The transformation of the waypoints can be done mathematically as formulated in Equation (4.2).

$$\begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sin(\theta).\cos(\phi) & \sin(\theta).\sin(\phi) & \cos(\theta) \\ -\cos(\theta).\cos(\phi) & \cos(\theta).\sin(\phi) & -\sin(\theta) \\ -\sin(\phi) & \cos(\phi) & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.2}$$

The new transformed path can be represented as follows: $P^s = \{S, (r_1, \theta_1, \phi_1), \ldots, (r_n, \theta_n, \phi_n), T\}$.

4.3 Fitness function

The UAV path planning problem is a multi-objective constraint problem, which can be expressed by the following equation:

$$\begin{aligned} \min \quad & f_k(x), \quad k = 1, 2, \ldots, n \\ & g_l(x) \leq 0, \quad l = 1, 2, \ldots, m \end{aligned} \tag{4.3}$$

Where $f(x)$ and $g(x)$ stand for objectives and constraints functions, respectively.
For simplification, we transformed the problem into a single weighted function that gathers the objectives and constraints as the following expression:
$Cost = \sum_{k=1}^{n} \omega_k . f_k(x) + \sum_{l=1}^{m} \omega_{l+n} . g_l(x)$.
Where n and m stand for the number of objectives and constraints, respectively. ω denotes a weight coefficient defined as follows:

$$\begin{cases} \omega_i \geq 0 \\ \sum\limits_{i=1}^{n+m} \omega_i = 1 \end{cases} \tag{4.4}$$

In our work, our objective is to ensure the UAV safety and preserves the UAV resources. Therefore, we introduce to our objective function the path length and height objectives, which are related to the UAV power source, and the UAV's angle and obstacle constraints related to the UAV safety. The total objective function is expressed mathematically as follows:
$Cost = \omega_1 . f_P + \omega_2 . g_O + \omega_3 . f_H + \omega_4 . g_s, \quad \omega_1 = \omega_2 = \omega_3 = \omega_4 = 0.25$.
Where f_P represents the objective function related to the path length. g_O represents the constraint function penalizing the collision with obstacles. f_H stands for height objective function. g_A is the function related to UAV's angle constraints.

4.3.1 Path length

Considering the power source limitation of UAVs, a shorter path saves more time and energy during the flight. Assuming that the total path involves n waypoints excluding the start and destination waypoints, the cost related to the path length f_P is calculated as in Equation (4.5).

$$f_P = \sum_{i=0}^{n} \|\overrightarrow{\mathrm{W_i W_{i+1}}}\| \tag{4.5}$$

4.3.2 Obstacles avoidance

In the process of UAV path planning, ensuring the safety of UAVs from colliding with obstacles is the main requirement. In our model, obstacles are characterized by their 2D positions (x_j, y_j) and radius R_j. For each segment line $\overrightarrow{\mathrm{W_iW_{i+1}}}$, the safety violation is calculated based on its distance from the obstacle's radius D_i. The total obstacles cost g_O is formulated as follows [178]:

$$g_O = \sum_{j=0}^{m-1} O_j \tag{4.6}$$

where m is the total number of obstacles and O_j is the cost related to the *j-th* obstacles expressed in the following equation:

$$O_j = \begin{cases} 0, & \text{if } D_i > R_j + D_s + U_s \\ \infty, & \text{if } D_i \leq R_j + U_s \\ R_j + D_s + U_s - D_i, & \text{Otherwise} \end{cases} \tag{4.7}$$

Where D_s stands for safety distance and U_s represents the UAV size.

4.3.3 Height

Instability in flight and altitude level changes affect both the energy consumption and control system of UAVs. For these reasons, the UAV should maintain a stable flying height. The cost related to height for the total flight f_H is expressed as follows:

$$f_H = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \overline{z})^2} \tag{4.8}$$

With

$$\overline{z} = \frac{1}{n}\sum_{i=1}^{n}(z_i) \tag{4.9}$$

4.3.4 UAV's angle

UAV's manoeuvrability is considered a constraint in path planning since it impacts its physical safety in case of large turns. Moreover, successive turns involve more power source consumption. Considering these facts, UAV's manoeuvrability angles should be limited. During the flight, UAVs make direction changes in both azimuth and elevation planes with the respective turning (θ) and climbing (ϕ) angles as shown in Figure 4.2. For each plane, a cost is associated with each angle produced by a change of direction. The total UAV's angle cost g_A is expressed in Equation (4.10).

$$g_A = g_C + g_T \tag{4.10}$$

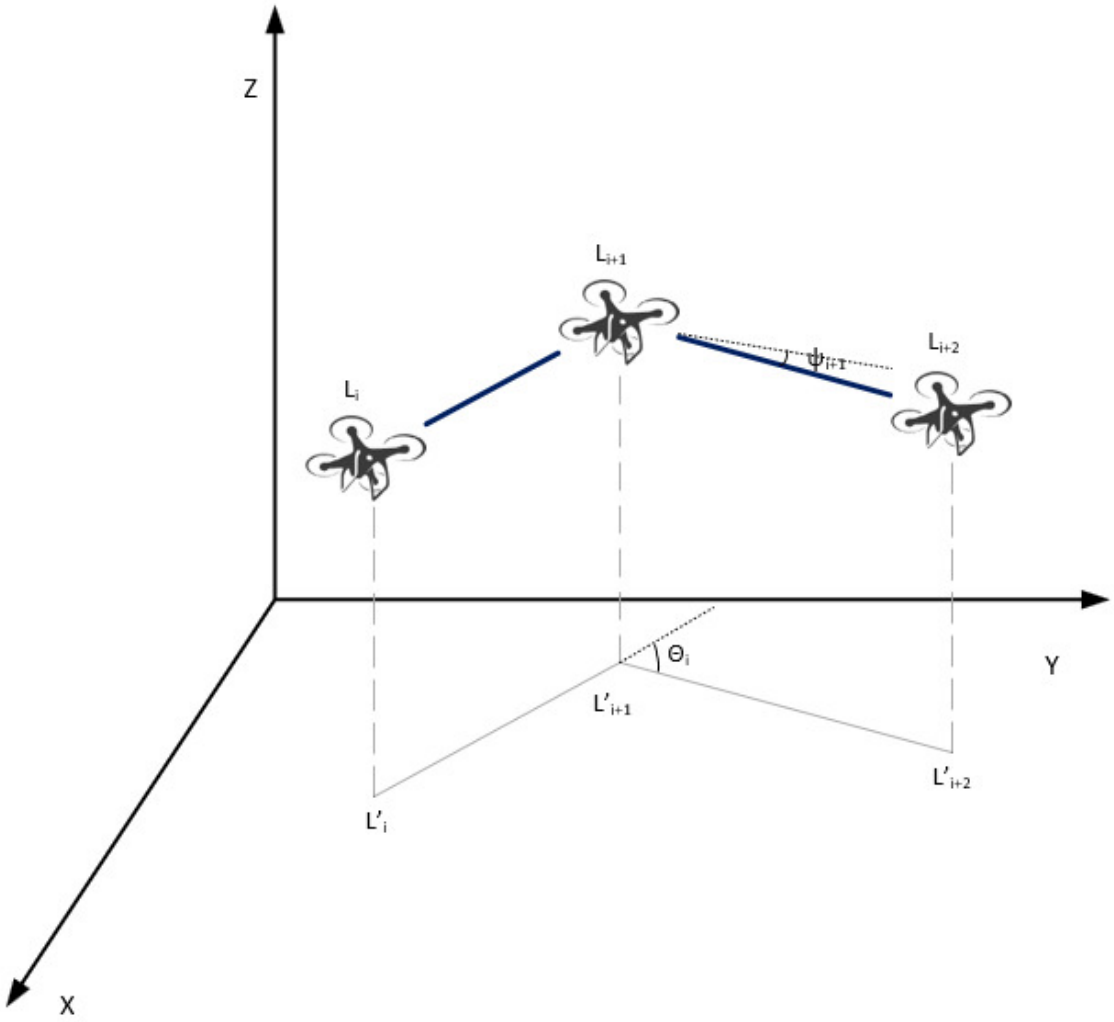


Figure 4.2: Caption

Where g_T and g_C are the costs related to both turning and climbing angles constraints formulated in Equations (4.11) and (4.13) respectively.

$$g_T = \sum_{i=0}^{n-1} \Theta i \tag{4.11}$$

With

$$\Theta_i = \begin{cases} \theta_i - \theta_{max}, & \text{if } \theta_i > \theta_{max} \\ 0, & \text{Otherwise} \end{cases}$$

Where θ_{max} denotes the maximum turning angle which depends on the physical design of the UAV. θ_i is the *i-th* turning angle produced by the projection of two consecutive segments of line over the horizontal plane $\overrightarrow{\mathrm{W'_i W'_{i+1}}}$ and $\overrightarrow{\mathrm{W'_{i+1} W'_{i+2}}}$. It is mathematically formulated in Equation (4.12).

$$\theta_i = \arccos \frac{\overrightarrow{\mathrm{W'_i W'_{i+1}}} . \overrightarrow{\mathrm{W'_{i+1} W'_{i+2}}}}{\|\overrightarrow{\mathrm{W'_i W'_{i+1}}}\| . \|\overrightarrow{\mathrm{W'_{i+1} W'_{i+2}}}\|} \tag{4.12}$$

The following equation expresses the total climbing angle constraint cost. For a given segment line $\overrightarrow{\mathrm{W_i W_{i+1}}}$, the associated cost is mathematically represented in Equation (4.14).

$$g_C = \sum_{i=1}^{n} \Phi i \tag{4.13}$$

With

$$\Phi_i = \begin{cases} |\phi_i| - \phi_{max}, & \text{if } |\phi_i| > \phi_{max} \\ 0, & \text{Otherwise} \end{cases} \tag{4.14}$$

Where ϕ_{max} is the maximum climbing angle allowed. ϕ_i stands for the *i-th* climbing angle formed by the segment line $\overrightarrow{\mathrm{W_i W_{i+1}}}$ and its 2D projection $\overrightarrow{\mathrm{W'_i W'_{i+1}}}$ as formulated in Equation (4.15).

$$\phi_i = \arctan\left[\frac{z_{i+1} - z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}\right] \tag{4.15}$$

4.4 African Vulture Optimization Algorithm

African Vulture Optimization Algorithm (AVOA) is a new nature-inspired meta-heuristic algorithm developed recently by Abdollahzadeh et al. in 2021 [179]. This algorithm was designed by modelling and simulating the living habits and foraging behaviour of African vultures according to the following criteria:

- The African vulture population has N vultures and the position space of each vulture is specified in d dimensions.
- The vulture population can be physically divided into three groups according to their living habits. The position of the vultures is determined by the fitness value of the

feasible solution; the best solution is recognized as the first-best vulture, the second solution is recognized as the second-best vulture, and the remaining vultures are assigned to the third group.

- In the population, the three groups are created in such a way that the most crucial natural function of vultures can be formulated: living in groups to find food. Therefore, different species of vultures play different roles.
- Similarly, the fitness value of the possible solution may reflect the advantages and disadvantages of the vultures. Thus, the weakest and hungriest vultures correlate with the worst vultures. The strongest and most numerous vultures, on the other hand, correspond to the best vulture at present. In general, all AVOA vultures seek to be near the best vultures while avoiding the worst.

Based on the vulture concepts and the four assumptions presented above, the AVOA algorithm was formulated in 5 distinct phases as follows.

4.4.1 Phase 1: Population Grouping

Once the initial random population of the AVOA algorithm is generated, the fitness values of all the solutions are evaluated, where the best solution is chosen as the best vulture in the first group, the second-best solution is placed in the second group, and the rest of the vultures are placed in the third group. Since the first and second best vultures have steering effects, equation ((4.16)) is designed to select the vulture to steer towards in the current iteration.

$$R(i) = \begin{cases} \text{Best Vulture }_1 \text{ if } p_i = L_1 \\ \text{Best Vulture }_2 \text{ if } p_i = L_2 \end{cases} \quad i = 1, \ldots, N. \tag{4.16}$$

where $R(i)$ denotes the best vulture selected, Best Vulture $_1$ means the first best vulture, and Best Vulture $_2$ means the second best vulture. L_1 and L_2 are two random values in the interval [0,1] and their sum must be equal to 1. The probability of selecting the best solution from each group p_i is obtained according to the Roulette wheel mechanism, and its calculation formula is shown in (4.17)

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i}, \quad i = 1, \ldots, N. \tag{4.17}$$

where F_i is the fitness of the first and second groups of vultures and n represents the total number of both groups of vultures.

4.4.2 Phase 2: The Rate of Satiety of Vultures

When the group of vultures feels full, they have enough energy to search for food over greater distances, but when they are starving, they do not have the energy to maintain their long-

distance flight. Therefore, hungry vultures will become particularly aggressive and stay near vultures that have food instead of searching for food themselves. The transition from the exploration stage to the exploitation stage can therefore be built based on this behaviour. The satiety degree of vultures (which is used as an indicator of the transition of vultures from exploration to exploitation), F, can be modelled as follows:

$$F = (2 \times \text{ rand }_1) \times z \times \left(1 - \frac{\text{It}}{\text{Max}_{\text{It}}}\right) + t \tag{4.18}$$

where rand $_1$ is a random value between 0 and 1, and z is a random value in the interval [-1,1] which changes at each iteration. t is calculated as (4.19)

$$t = h \times \left[\sin^{\omega}\left(\frac{\pi}{2} \times \frac{\text{It}}{\text{Max}_{\text{It}}}\right) + \cos\left(\frac{\pi}{2} \times \frac{\text{It}}{\text{Max}_{\text{It}}}\right) - 1\right] \tag{4.19}$$

Where It is the current iteration, Max_{It} is the maximum number of iterations, and h denotes a random number between [- 2,2]. w is a constant that controls the exploration phase. F_i will gradually decrease as the number of iterations increases, according to (4.19). Vultures enter the exploration phase and search for new food at different locations when the value of $|F_i|$ is greater than 1. Otherwise, vultures enter the exploitation phase, searching for food in the immediate vicinity.

4.4.3 Phase 3: Exploration Stage

In the wild, vultures have very good eyesight, which allows them to efficiently find food and dying animals. However, vultures may have difficulty locating food, as they spend a great deal of time examining their environment before flying great distances in search of food. The vultures in AVOA can inspect randomly selected locations, using two distinct strategies. Each strategy is selected using a parameter called $P1$, which is assigned a value between 0 and 1. The mathematical model can be expressed as follows.

$$P(i+1) = \begin{cases} R(i) - D(i) \times F, P1 \geq \text{rand}_{p1} \\ R(i) - F_i + \text{rand}_2 \times ((ub - lb) \times \text{rand}_3 + lb), \quad i = 1, \ldots, N. \\ P1 < \text{rand}_{p1} \end{cases} \tag{4.20}$$

where $R(i)$ is one of the best vultures chosen in the current iteration using (4.16), F_i is the satiation rate of the vultures in the current iteration calculated using (4.18), rand_2 is a random number in the range [0, 1], and lb and ub are the variables' lower and upper bounds. rand_3 is used to provide a high random coefficient as a means of increasing diversity and exploring different search areas. $D(i)$ represents the distance between the vulture and the

current optimum one. It is calculated by

$$D(i) = |X \times R(i) - P(i)|, \quad i = 1, \ldots, N. \tag{4.21}$$

here $P(i)$ is the position of the i^{th} vulture, and X is a random value between 0 and 2.

4.4.4 Phase 4: Exploitation (First Stage)

In this phase, the efficiency stage of the AVOA is explored. The AVOA starts the first phase of operation if $|F_i|$ is in the interval [0.5, 1]. Here, two behaviours are realized: siege combat and rotary flight. The parameter P2 in the interval [0,1] allows us to decide which strategy to choose, which must be valued before the search operation. At the beginning of this phase, rand_{p2}, a random number between 0 and 1 is produced. If this number is greater than or equal to the P2 parameter, the siege combat strategy is slowly applied. Otherwise, the rotation flight technique is carried out. This behaviour can be simulated as follows:

$$P(i+1) = \begin{cases} Eq.(4.23) & \text{if } P_2 \geq \text{ rand }_{p_2} \\ Eq.(4.24) & \text{if } P_2 < \text{ rand }_{p_2} \end{cases}, \quad i = 1, \ldots, N. \tag{4.22}$$

$$P(i+1) = D(i) \times (F + \text{ rand }_4) - d(t), \quad i = 1, \ldots, N. \tag{4.23}$$

$$P(i+1) = R(i) - (S_1(i) + S_2(i)), \quad i = 1, \ldots, N. \tag{4.24}$$

where rand $_4$ is a random number in [0, 1] and $d(t)$ is the distance between the i^{th} vulture and the current best vulture, which is calculated as follows:

$$d(i) = R(i) - P(i), \quad i = 1, \ldots, N. \tag{4.25}$$

$S_1(i)$ and $S_2(i)$ are calculated as (4.26) and (4.27):

$$S_1(i) = R(i) \times \left(\frac{\text{rand }_5 \times P(i)}{2\pi} \right) \times \cos(P(i)), \quad i = 1, \ldots, N. \tag{4.26}$$

$$S_2(i) = R(i) \times \left(\frac{\text{rand }_6 \times P(i)}{2\pi} \right) \times \sin(P(i)), \quad i = 1, \ldots, N. \tag{4.27}$$

here rand $_5$ and rand $_6$ are random numbers between 0 and 1.

4.4.5 Phase 5: Exploitation (Second Stage)

This phase of the algorithm is performed if $|F_i| < 0.5$. Most of the vultures in the population were satiated, but the two best types of vultures became weak and hungry after prolonged exertion. At this point, the vultures attack the food and many types of vultures gather

around the same food source. Therefore, in the later exploitation phase, there is also a parameter p3 between 0 and 1. This parameter is used to determine whether the vultures exhibit aggregation or attack behaviour. Before the vultures act, a rand_3 will be generated in the interval [0,1]. When rand_{p3} is greater than or equal to the parameter P_3, vultures exhibit aggregation behaviour. On the other hand, when rand_{p3} is less than the parameter P_3, the vulture adopts an attack behaviour. This phase is mathematically formulated as follows:

$$P(i+1) = \begin{cases} Eq.(4.29) & \text{if } P_3 \geq \text{ rand }_{p3} \\ Eq.(4.30) & \text{if } P_3 < \text{ rand }_{p3} \end{cases}, \quad i = 1, \ldots, N. \tag{4.28}$$

$$P(i+1) = \frac{A_1(i) + A_2(i)}{2}, \quad i = 1, \ldots, N. \tag{4.29}$$

$$P(i+1) = R(i) - |d(i)| \times F_i \times \text{Levy}(d), \quad i = 1, \ldots, N. \tag{4.30}$$

where d is the problem dimensions. A_1 and A_2 are respectively calculated as (4.31) and (4.32)

$$A_1(i) = \text{ Best Vulture }_1(i) - \frac{\text{BestVulture }_1(i) \times P(i)}{\text{BestVulture }_1(i) - (P(i))^2} \times F, \quad i = 1, \ldots, N. \tag{4.31}$$

$$A_2(i) = \text{ BestVulture }_2(i) - \frac{\text{Best Vulture }_2(i) \times P(i)}{\text{BestVulture }_2(i) - (P(i))^2} \times F, \quad i = 1, \ldots, N. \tag{4.32}$$

The efficiency of the AVOA was increased using the Lévy flight (LF) mechanism, which is derived using the following equation

$$\text{Levy(d)} = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{4.33}$$

where β is a fixed number set to 1.5, $r1$ and $r2$ are a random number between 0 and 1. σ can be calculated as (4.34)

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma(1+\beta 2) \times \beta \times 2\left(\frac{\beta-1}{2}\right)} \right)^{\frac{1}{\beta}} \tag{4.34}$$

where $\Gamma(x) = (x-1)!$. The pseudo-code of the AVOA is described in Algorithm 1.

4.5 Chaotic map

In general, chaos can be defined as a deterministic and arbitrary type of strategy observed in a nonlinear dynamic system. Chaos has a special property that includes its ability to

Algorithm 1 The pseudo-code of the African vultures Optimization Algorithm

Initialization Step, P_i, $i = 1, 2, \ldots .N$.

while $It < \text{Max}_{\text{It}}$ **do**
    Calculate the fitness of Vulture
    Set $\boldsymbol{P}_{\text{BestVulture}_1}$, as the location of Vulture (First best location of BestVulture category 1)
    Set $\boldsymbol{P}_{\text{BestVulture}_2}$, as the location of Vulture (Second best location of BestVulture category 2)
    for each Vulture (Pi) **do**
        Select $R(i)$ using Eq. (4.16)
        Update the F using Eq. (4.18)
        if $|F| \geq 1$ **then**
            if $P_1 \geq \text{rand}_{p1}$ **then**
                Update the location Vulture using Eq. (**??**)
            else
                Update the location Vulture using Eq. (**??**)
            end if
        else
            if $|F| < 1$ **then**
                if $F \geq 0.5$ **then**
                    if $P_2 \geq \text{rand}_{p2}$ **then**
Update the location using Eq. (4.23)
                    else
Update the location Vulture using Eq. (4.24)
                    end if
                else
                    if $P_3 \geq \text{rand}_{p3}$ **then**
Update the location Vulture using Eq. (4.29)
                    else
Update the location Vulture using Eq. (4.30)
                    end if
                end if
            end if
        end if
    end for
end while

generate regular, unpredictable, ergodic, and non-repeating numbers. So far, many meta-heuristic algorithms contain specific random parameters that need to be tuned. However, parameter tuning is a difficult task as it can vary with different data sets. Thus, chaos can be an effective way to solve the problem. The main idea is to replace the random initialization variables with chaotic map variables. This can maintain population diversity, allow the optimization algorithm to escape the local trap, and improve the overall exploration capability of the algorithm. A large number of chaotic maps are available in the field of optimization. The ten best-known chaotic maps are the Logistic map, Tent map, Sine map, Gauss map, Chebyshev map, Piecewise map, Iterative map, Singer map, and Circle map. In our work, we have applied the logistic map. This classical logistic map appears in the nonlinear dynamics of biological populations highlighting a chaotic behaviour [180]. It is one of the simplest and most representative chaotic maps, which generates a more uniform value between [0, 1]. Therefore, we adopted this chaotic map to replace the random variables of the AVOA position. It can be expressed as follows

$$L_{t+1} = aL_t\left(1 - L_t\right) \tag{4.35}$$

where L_t is the t^{th} chaotic number, with t denoting the number of iterations. Obviously, $L \in (0, 1)$ under the conditions that the initial $L_0 \in (0, 1)$ and $L_0 \notin \{0.0, 0.25, 0.75, 0.5, 1.0\}$. $a = 4$ (in experiments).

4.6 Cauchy mutation

The cauchy mutation is a random mutation operation based on the Cauchy distribution that is introduced in the design of fast evolutionary programming [181]. It has been integrated into several meta-heuristic algorithms such as Bat Algorithm (BA) [182], Krill herd (KH) [183], and Differential evolution (DE) [184]. The idea behind the inclusion of a Cauchy mutation operator in meta-heuristic approaches is to disrupt the crossing points. In addition, the selection probability operator is introduced to allow the algorithm to accept the current bad solution with some probability so that the algorithm can skip the local optimal solution and find the best optimal solution. The Cauchy density function is expressed as follows [182]:

$$f_t(x) = \frac{1}{\pi}\frac{t}{t^2 + x^2} \; -\infty < x < +\infty \tag{4.36}$$

where $t > 0$ is a scale parameter.

Then, the Cauchy distributed function is described as (4.37) [182]

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x}{t}\right) \tag{4.37}$$

The Cauchy mutation will result in a larger mutation step size, which gives the algorithm a

better overall search capability and improve the diversity of the population.

4.7 Elite Opposition-based learning

Elite Opposition-based learning (EOBL) is an innovative approach in computer intelligence proposed in [185]. It is an improved version of the OBL technique that aims to improve the performance of meta-heuristic optimization algorithms. Its strategy is based on finding a more efficient solution between the current individuals generated by the optimization algorithm, and its corresponding opposite solution. The evaluation function is applied for both solutions, and the best one is selected to proceed to the next iteration. This technique can be integrated with various meta-heuristics algorithms such as Particle Swarm Optimization (PSO) [186], Grey Wolf Optimizer (GWO) [187], Crow Search Algorithm (CSA) [188], and Salp Swarm Algorithm (SSA) [189]. Mathematically, OBL can be modelled as follows: let P be a real candidate solution in one dimension space, where $P \in [ub, lb]$. The opposite value P^o is defined as:

$$P^o = lb + ub - P \tag{4.38}$$

The EOBL strategy relies on the elite individual to drive the population toward the global solution. In particular, EOBL relies on the elite individual of the current population to generate the complementary opposites of the current population located within the research boundaries. The population is then guided by the elite individual to finally reach the promising region, in which the global optimum can be found. Therefore, the application of the EOBL technique will enhance the diversity of the population and improve the overall search of the optimization algorithm. Supposing that P_{el} is the elite candidate, the elite opposite position, P^o_{el}, can be obtained by the following equation

$$P^o_{el} = S \times (d_a + d_b) - P_{el} \tag{4.39}$$

where $S \in (0, 1)$ is a generalized factor used to control the opposition magnitude. d_a and d_b are dynamic boundaries, which can be defined as:

$$d_a = \min(P_{el}), d_b = \max(P_{el}) \tag{4.40}$$

However, the corresponding opposite may exceed the search bounds [lb, ub]. To solve this problem, the transformed individual is assigned a random value in $[lb, ub]$ as follows:

$$P^o_{el} = \text{rand}(lb, ub), \quad \text{if} \quad P^o_{el} < lb \quad \text{OR} \quad P^o_{el} > ub \tag{4.41}$$

4.8 The proposed CCO-AVOA algorithm for solving the UAVs path planning problem

In this section, we illustrate the structure of our proposed CCO-AVOA algorithm in detail. The core idea of the CCO-AVOA algorithm is the incorporation of three strategies including the chaotic map, EOBL concept, and Cauchy mutation into the original AVOA algorithm to enhance its optimization performance. Firstly, the chaotic sequence generated by the Logistic chaotic map according to (Eq.4.35) is used in three cases as described below:

- In this first case, the random parameters rand_{p1}, rand_{p2}, and rand_{p3} are replaced by the logistic chaotic map L_t. The corresponding parameters determine the behaviour of the African vultures. The new position's update is formulated as follows:

$$P(i+1) = \begin{cases} R(i) - D(i) \times F, P1 \geq \mathrm{L}_t \\ R(i) - F + \text{rand}_2 \times ((ub - lb) \times \text{rand}_3 + lb), \quad i = 1, \ldots, N. \\ P1 < \mathrm{L}_t \end{cases} \tag{4.42}$$

$$P(i+1) = \begin{cases} Eq.(4.23) & \text{if } P_2 \geq \mathrm{L}_t \\ Eq.(4.24) & \text{if } P_2 < \mathrm{L}_t \end{cases} \quad i = 1, \ldots, N. \tag{4.43}$$

$$P(i+1) = \begin{cases} Eq.(4.29) & \text{if } P_3 \geq \mathrm{L}_t \\ Eq.(4.30) & \text{if } P_3 < \mathrm{L}_t \end{cases} \quad i = 1, \ldots, N. \tag{4.44}$$

- In Elite Opposition-based Learning, the generalized factor S which is a random value in the range $[0, 1]$ is replaced by L_t. The improved Elite opposite P^o_{el} of a given position P_{el} is given by:

$$P^o_{Iel} = L_t \times (d_a + d_b) - P_{el} \tag{4.45}$$

- In the last case, the logistic map is used in the Cauchy mutation strategy to improve the stochastic behaviour of the random variable generated from the Cauchy distribution. Figure 4.3 displays both standards and improved Cauchy distribution during the iteration. The improved Cauchy distribution is obtained by the following Cauchy operator expressed in Equation (4.46).

$$CM = \tan\left((L_t - \frac{1}{2}).\pi\right) \tag{4.46}$$

As a second improvement, the Improved Elite Opposition-based Learning is adopted to enhance the exploration search of the chaotic AVOA algorithm. For each iteration, the CCO-AVOA algorithm searches for the elite opposite of positions using Equations (4.40)

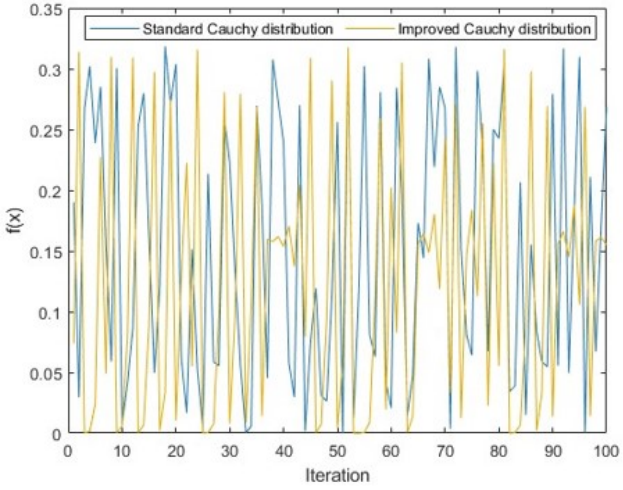


Figure 4.3: The effect of introducing logistic map into Cauchy distribution

and (4.45). Then, the $P_{el} \cup P^o_{Iel}$ positions are evaluated based on their fitness values. Finally, The best N positions from $\{P_{el} \cup P^o_{Iel}\}$ are selected to replace the old positions.
As the last improvement, we integrated an improved Cauchy mutation strategy to improve the quality of the solution and avoid trapping into local optima. The concept of the Cauchy mutation method is to perturb the candidate solutions to find eventually the global optimal solution. In the CCO-AVOA algorithm, the Cauchy mutation strategy is applied to the selected solutions from the previous improvement. The new mutated solution P'_i can be found using equation (4.47). Both solutions are evaluated and the best solutions are chosen. This process will repeat until the maximum number of iterations is reached.

$$P'_i = P_i + P_i.CM, \quad i = 1, \ldots, N. \tag{4.47}$$

Where P_i is the current solution. CM is the Cauchy operator defined in Equation (4.46).

The pseudo-code of the proposed approach for UAV path planning is expressed in Algorithm 2.

4.9 Simulation results and analysis

In this section, we evaluated the performance of the suggested CCO-AVOA algorithm for solving the UAV path planning problem using several scenarios. For all scenarios, the UAV performs its flight in a 3D area of $1000m \times 1000m \times 400m$. In this area, the UAV travels from an initial position of $(200, 100, 150)$ to a final position at $(800, 800, 150)$, where obstacles are set up differently for each scenario. In table 4.1, The details of the scenarios are summarised. To validate its efficiency, we performed a comparison with various meta-heuristics such as the original AVOA, Particle Swarm Optimization (PSO), Bat Algorithm (BA), Grey Wolf Optimizer (GWO), Sine Cosine Algorithm (SCA), Whale Optimization Algorithm (WOA),

Algorithm 2 The pseudo-code of the proposed Chaotic Cauchy Opposition-based African Vulture Optimization Algorithm

Set the path planning parameters: Start and destination points, obstacles information, maximum turning and climbing angles
Transform the original coordination into a spherical coordinate system
Initialization Step, P_i, $i = 1, 2, \ldots .N$.
Initialize the logistic chaotic map L_0
while $It < \mathrm{Max_{It}}$ **do**
    State Apply EOBL using Eq. (4.45) and (4.40), and select the N fittest solution from $\{P_{el} \cup P^o_{Iel}\}$
    State Apply CM strategy using Eq. (4.46) and (4.47), and select the N fittest solution from $\{P_{el} \cup P'_i\}$
    Set $\boldsymbol{P}_{\mathrm{BestVulture_1}}$, as the location of Vulture ((First best location of BestVulture category 1)
    Set $\boldsymbol{P}_{\mathrm{BestVulture_2}}$, as the location of Vulture (Second best location of BestVulture category 2)
    for each Vulture (Pi) **do**
        Select $R(i)$ using Eq. (4.16)
        Update the F_i using Eq. (4.18)
        if $|F_i| \geq 1$ **then**
            Update the position using Eq. (4.42)
        else
            if $|F_i| < 1$ **then**
                if $F_i \geq 0.5$ **then**
                    Update the position using Eq. (4.43)
                else
                    Update the position using Eq. (4.44)
                end if
            end if
        end if
    end for
end while
return The best path BestVulture
Transform BestVulture into original coordinates

Table 4.1: Scenario settings

Scenario	Parameter	Value
Common settings	UAV size (Diagonal wheelbase) (U_s)	$1m$
	Safety distance (D_s)	$10m$
	Number of waypoints (n)	[10,20,30,40]
	Height's limit	[100,200]
	Maximum climbing angle	45°
	Maximum turning angle	45°
Scenario 1	Obstacle's position and radius	(400,500,100,80)
		(600,200,150,70)
		(500,350,150,80)
		(350,200,150,70)
Scenario 2	Obstacle's position and radius	(200,200,150,80)
		(380,500,150,100)
		(550,250,150,75)
		(250,750,150,80)
		(750,550,150,75)
		(800,250,150,80)
		(500,700,150,100)
Scenario 3	Obstacle's position and radius	(200,200,150,80)
		(400,450,150,100)
		(550,250,150,75)
		(250,750,150,80)
		(750,550,150,75)
		(800,250,150,80)
		(500,700,150,100)
		(680,375,150,75)
		(900,700,150,80)
		(600,900,150,75)
Scenario 4	Obstacle's position and radius	(200,200,150,80)
		(400,450,150,100)
		(550,250,150,75)
		(250,750,150,80)
		(750,550,150,75)
		(800,250,150,80)
		(500,700,150,100)
		(680,375,150,75)
		(900,700,150,80)
		(600,900,150,75)
		(700,750,150,70)
		(550,500,150,70)
		(320,300,150,70)

Dragonfly Algorithm (DA), Aquila Optimizer (AO), and Wild Horse Optimizer (WHO) algorithms. The parameter settings of the selected algorithms are presented in Table 4.2. Experiments were conducted on MATLAB 2022b, 64-bit, and implemented on the computational environment of Intel Core i7-10700 CPU 2.90GHz, 32 GB RAM system. All performance results are obtained based on the average of 50 independent runs.

4.9.1 Case of four obstacles

Fitness analysis

Table 4.3 reports fitness results in the case of using four obstacles. As shown, the proposed CCO-AVOA algorithm provides the minimum fitness values regarding the best, worst, mean, and standard deviation in most cases. From the best fitness results, the CCO-AVOA algorithm offers the optimal and safe path. Regarding the standard deviation, we can notice that our proposed algorithm is the best in terms of stability, which explains its robustness against the local optimal issue. As for SCA, WOA, AO, AVOA, and WHO algorithms, they provide good and acceptable results in this scenario. Although the DA algorithm was able to

Table 4.2: Algorithms parameters

Algorithm	**Parameter**	**Value**
Common settings	Population size N	50
	Maximum number of iterations T	200
	Number of separate runs	50
CCO-AVOA	Control parameter α	0.8
	Control parameter β	0.2
	Control parameter γ	2.5
	Control parameter P_1	0.6
	Control parameter P_2	0.4
	Control parameter P_3	0.6
	Initial value L_0	0.7
PSO	Inertia maximum weight ω_{Max}	0.9
	Inertia minimum weight ω_{Min}	0.4
	Acceleration parameter C_1	2
	Acceleration parameter C_2	2
BA	Minimum frequency f_{Min}	0
	Maximum frequency f_{Min}	2
	Initial loudness A_0	1
	Initial pulse emission rate r_0	1
	Loudness constant α	0.5
	Emission rate constant γ	0.5
GWO	Control parameter a_{min}	0
	Control parameter a_{max}	2
SCA	Control parameter r_1	[2,0]
WOA	Control parameter a_{min}	0
	Control parameter a_{max}	2
DA	Inertia maximum weight ω_{Max}	0.9
	Inertia minimum weight ω_{Min}	0.4
AO	Exploitation adjustment α	0.1
	Exploration adjustment β	0.1
AVOA	Control parameter α	0.8
	Control parameter β	0.2
	Control parameter γ	2.5
	Control parameter P_1	0.6
	Control parameter P_2	0.4
	Control parameter P_3	0.6
WHO	Stallion percent PS	0.2
	Crossover percent PC	0.13

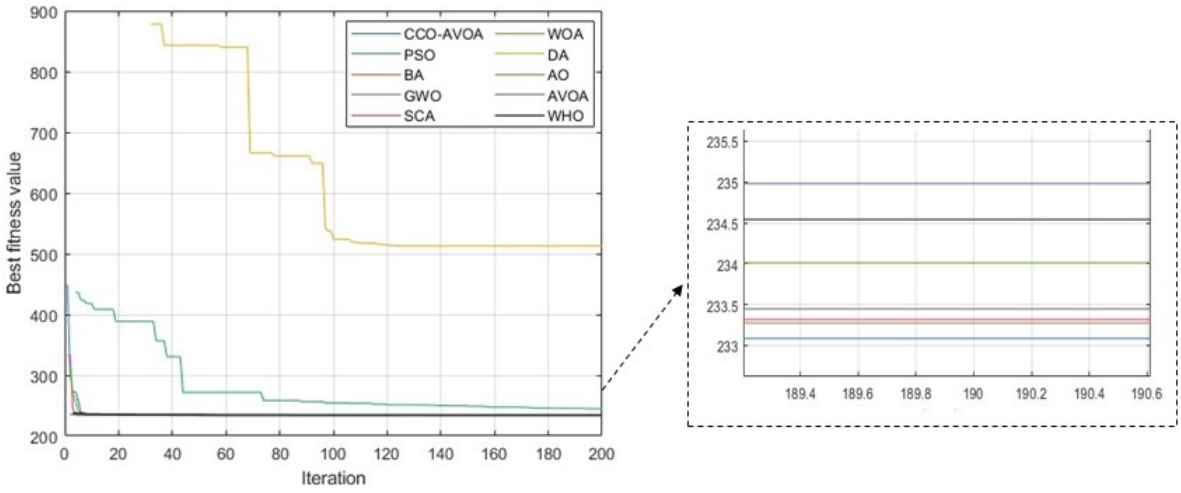


Figure 4.4: Example of convergence curves the first case for $n = 40$

provide a safe path in all cases of n, it cannot be considered a good path. As we can see, the mean and worst fitness values are important and affect the quality of the path. Moreover, the DA algorithm is not stable and may fail according to its standard deviation results. The results of the GWO algorithm can be considered good only in the case of $n = \{10, 20\}$. In other cases, GWO is unable to provide a safe path which is demonstrated by the worst fitness values. As for the PSO algorithm, except in the case of $n = 10$, it provides the highest fitness value, which makes it easy to fall into local optimal as in the case of $n = 40$. This issue can be confirmed by its standard deviation results. In this scenario, the BA algorithm presented the worst results regarding the fitness value. It can be seen that it provides results only in the case of $n = 10$. In other cases, at some runs, it was not able to provide a collision-free path which is justified by its infinite fitness value. In those cases, BA is trapped in local optima due to its instability.

Figure 4.4 illustrates an example of convergence curves of the tested algorithms in the case of $n = 40$. We can notice that the CCO-AVOA algorithm convergences to the best cost quickly in the earlier iterations. Moreover, the best cost, in this case, is provided by the proposed CCO-AVOA algorithm, which demonstrates its efficiency in terms of convergence speed and rate. The convergence of GWO, SCA, WOA, AO, AVOA, and WHO is considered good. As for the PSO and DA algorithms, we can see that their convergence is slow during the iterations and does not reach the optimal cost. Both of them require more iterations for searching. In this figure, we can notice that the convergence curve of the BA algorithm is not displayed, which is explained by the fact that the BA algorithm did not provide any optimal cost in this case.

Performance analysis

Figure 4.5 displays the average execution time taken by the algorithms to provide their optimal solution. As noticed, the proposed approach takes more time to process in case of

Table 4.3: Fitness results of the first case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Best	233.06	**232.52**	246.14	233.25	233.08	233.08	242.49	233.16	233.07	233.07	233.26	$n = 10$
Worst	233.25	238.39	325.03	235.21	233.45	233.92	302.49	234.45	**233.18**	234.44	235.51	
Mean	**233.09**	233.96	303.68	234.18	233.19	233.34	252.07	233.56	**233.09**	233.29	233.91	
Std	0.03	1.78	24.57	0.53	0.08	0.23	16.66	0.29	**0.02**	0.3	0.46	
Best	**233.06**	234.51	283	233.47	233.09	233.09	234.16	233.17	233.08	233.09	233.38	$n = 20$
Worst	**233.54**	242.84	/	235.43	234.16	234.15	440.15	234.52	234.13	235.38	236.52	
Mean	**233.14**	238.33	/	234.48	233.34	233.57	264.35	233.54	233.17	233.69	234.29	
Std	**0.09**	2.17	/	0.53	0.26	0.31	36.54	0.27	0.16	0.53	0.6	
Best	**233.08**	237.33	285.3	233.25	233.09	233.1	233.87	233.1	**233.08**	233.22	233.33	$n = 30$
Worst	233.59	250.73	/	/	235.58	234.93	733.58	235.17	**233.55**	235.32	235.46	
Mean	**233.21**	242.16	/	/	233.4	233.55	316.71	233.74	233.22	233.41	234.36	
Std	**0.12**	2.28	/	/	0.39	0.42	100.26	0.56	0.14	0.51	0.58	
Best	**233.08**	242.44	305.04	233.16	233.1	365.32	234.91	233.11	233.09	288.62	233.31	$n = 40$
Worst	**233.81**	/	/	/	234.43	438.35	620.08	235.16	234.15	/	235.73	
Mean	**233.2**	/	/	/	233.46	380.45	315.49	233.84	233.22	/	234.31	
Std	**0.13**	/	/	/	0.33	9.26	82.39	0.61	0.16	/	0.64	

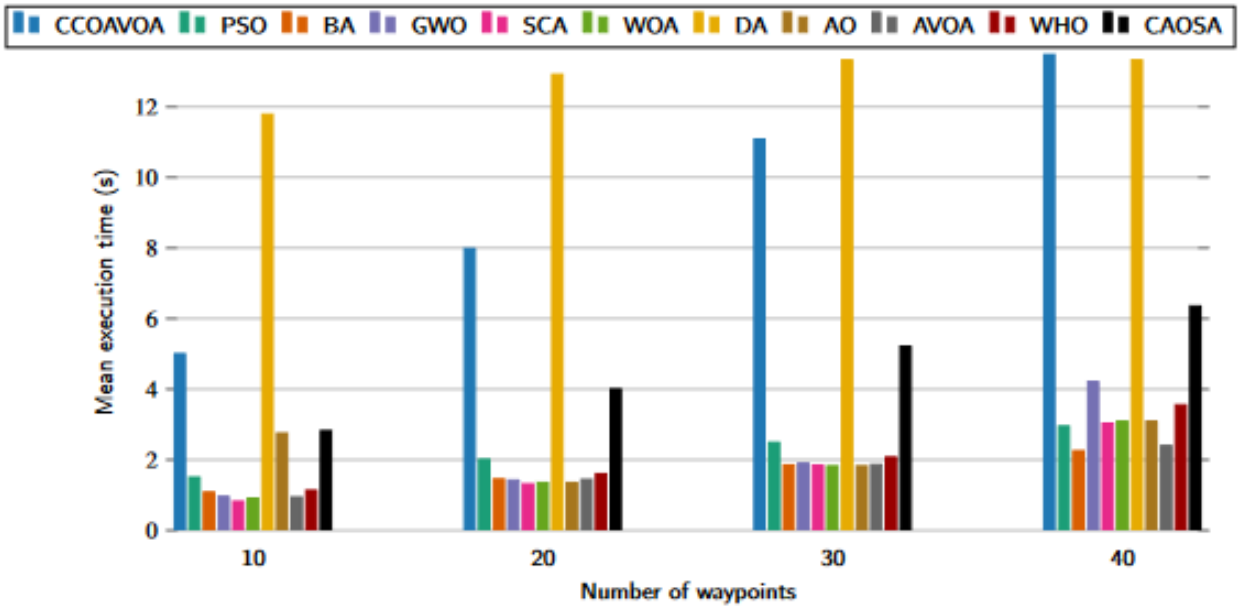


Figure 4.5: The average execution time in the first case

Table 4.4: Cost results in the first case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Path cost	**925.8**	929.45	1075.6	928.74	926.05	926.39	979.42	926.82	925.8	926.57	928.23	$n = 10$
Height cost	**0.04**	3.11	18.68	1.48	0.2	0.45	14.42	0.72	0.05	0.11	0.77	
Obstacles cost	6.51	3.29	15.56	**6.5**	6.51	**6.5**	4.32	6.68	6.51	6.51	6.66	
Smoothness cost	**0**	**0**	104.9	**0**	**0**	**0**	10.11	0	**0**	**0**	0	
Path cost	**925.92**	941.32	1189.8	930.1	926.52	927.29	1002.6	926.9	926.03	928.1	929.6	$n = 20$
Height cost	**0.12**	9.42	29.53	1.32	0.31	0.49	14.15	0.61	**0.12**	0.16	0.08	
Obstacles cost	6.51	**2.57**	/	6.51	6.52	6.51	4.42	6.67	6.51	6.51	6.59	
Smoothness cost	**0**	**0**	342.56	**0**	**0**	**0**	36.22	0	**0**	**0**	**0**	
Path cost	**926.17**	955.15	1295.6	928.16	926.76	927.19	1071.9	927.72	926.23	929.37	930.17	$n = 30$
Height cost	0.15	10.2	26.36	1.77	0.28	0.5	16.01	0.5	**0.12**	0.23	0.67	
Obstacles cost	6.5	**3.29**	/	/	6.57	6.51	5.76	6.73	6.51	6.51	6.61	
Smoothness cost	**0**	**0**	1021.7	**0**	**0**	**0**	173.12	0	**0**	**0**	**0**	
Path cost	**926.18**	926.18	1304.4	928.65	926.97	927.25	1088.2	927.72	926.25	930.5	929.54	$n = 40$
Height cost	**0.13**	23.86	28.94	1.26	0.31	0.43	15.15	0.62	0.14	0.27	0.87	
Obstacles cost	**6.51**	/	/	/	6.55	**6.51**	4.51	7.07	**6.51**	/	6.82	
Smoothness cost	**0**	528.9	1414	**0**	**0**	**0**	154.06	0	**0**	**0**	**0**	

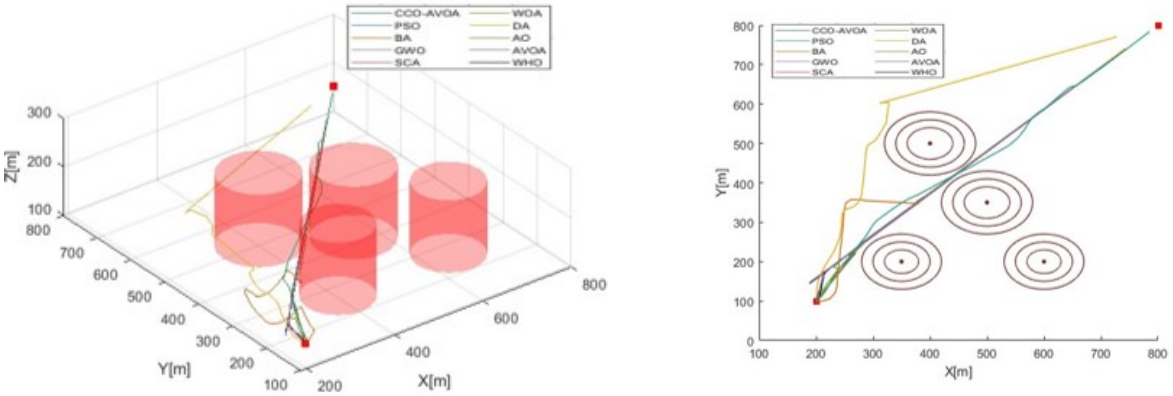
different numbers of waypoints. This can be explained by the fact that the CCO-AVOA approach explores more candidate solutions in the initialization and update phase. In this scenario, the mean execution time of CCO-AVOA is increased from 5 s to 14.19 s for a variation in the number of waypoints from 10 to 40. It can be explained by the fact that the number of waypoints is proportional to the problem's complexity. Therefore, the CCO-AVOA algorithm takes additional time to generate a path in case of handling high problem complexity. Regarding the DA algorithm, we can see that the average execution time is high compared to the other algorithms, which can be explained by the random search behavior in the exploration phase. As for the rest, the average execution time does not exceed 4.5 s during this case.

Table 4.4 details the performance results of the algorithms in case of using 4 obstacles. From the table, we can notice that our proposed approach offers the shortest path for different values of n. Therefore, the CCO-AVOA algorithm is the suitable method in terms of path length. The SCA, WOA, AO, and AVOA algorithms also provide good results in the same case. As for GWO and WHO algorithms, their results are acceptable only in the case of $n = \{10, 20\}$ and $n = \{10, 20, 30\}$, respectively. In other cases, both of them failed to provide a feasible path. In this scenario, the path given by the DA algorithm is the longest path which cannot be considered acceptable. The BA algorithm could provide a feasible path only in the case of $n = 10$. Besides the safety, the remaining path is long and not optimal.

Regarding the obstacle cost, we can see the results from the proposed approach, and both WOA and AVOA algorithms are approximately the same. In this scenario, the DA algorithm provides the smallest obstacle's cost handling better UAV safety. The PSO algorithm also gave optimal results only for $n = \{10, 20, 30\}$. As for the GWO algorithm, it offers good results in a few waypoints' numbers. The results given by the BA algorithm are not considered optimal since the path provided is too closer to the obstacle and does not respect the safety distance. Eventually, using the BA algorithm, the UAV collides with obstacles for $n > 10$.

By analyzing the height cost, it can be seen that the proposed approach provides the best values in the case of $n = 10, 40$. Using our CCO-AVOA algorithm, the UAV remains flying approximately at the same height level preserving its power source. SCA, WOA, AO, AVOA, and WHO algorithms also offered good height results in the scenario. The results given by GWO are also good and acceptable. As for DA, PSO, and BA algorithms, the height cost is important and remarkably affects energy consumption.

Concerning the UAV's angle cost, the proposed approach with GWO, SCA, AO, AVOA, and WHO algorithms provide good results. Using these algorithms, the UAV can perform its turns with complete safety. As for the PSO algorithm, it offers good results for $n \leq 40$. Both BA and DA algorithms solutions cannot be acceptable since they affect UAV safety.



(a) The 3D view

(b) The top view

Figure 4.6: The path given by the algorithms in the first case for $n = 40$

4.9.2 Case of seven obstacles

Fitness analysis

Table 4.5 reports the fitness results given by the algorithms in the case of using seven obstacles under various numbers of waypoints. From the results, we can see that our CCO-AVOA algorithm achieves the best results in most cases. It gives the smallest fitness value with the best stability while dealing with this case of environmental complexity. The AVOA algorithm offers good fitness results in this scenario. The results given by AO and WHO algorithms are acceptable since they can offer a safe path. But their solution cannot be considered optimal considering their mean fitness value. As for GWO and WOA algorithms, we can notice their failure in ensuring UAV safety starting from $n > 30$. In this case, we can conclude that both GWO and WOA algorithms cannot handle high complexity. Regarding the PSO algorithm, it succeeded in finding a good path in all cases of n where Its results are acceptable. By analyzing fitness results for $n = 20$, we can notice that the values are great compared to the other cases. Therefore, the PSO algorithm can be applied to a few waypoints. In this scenario, we can see that both BA and DA algorithms did not fail according to the fitness results. However, both of them provide great results, which are not optimal. By seeing their standard deviation values, both of them are unstable and may fail. Figure 4.7 shows an example of the convergence curves of the cost minimization by using the ten methods. We can see that the BA algorithm converges quickly to a premature solution due to the lack of exploring all waypoints in the search area. The convergence of PSO, SCA, WOA, DA, and AO is slow, which indicates that they are still searching for the best solution. GWO, AVOA, and WHO algorithms, in this case, converge in a similar way. The convergence of our CCO-AVOA approach is efficiently fast, which demonstrates its balancing capabilities between the exploration and exploitation phases. Moreover, the proposed approach provides the optimal fitness cost.

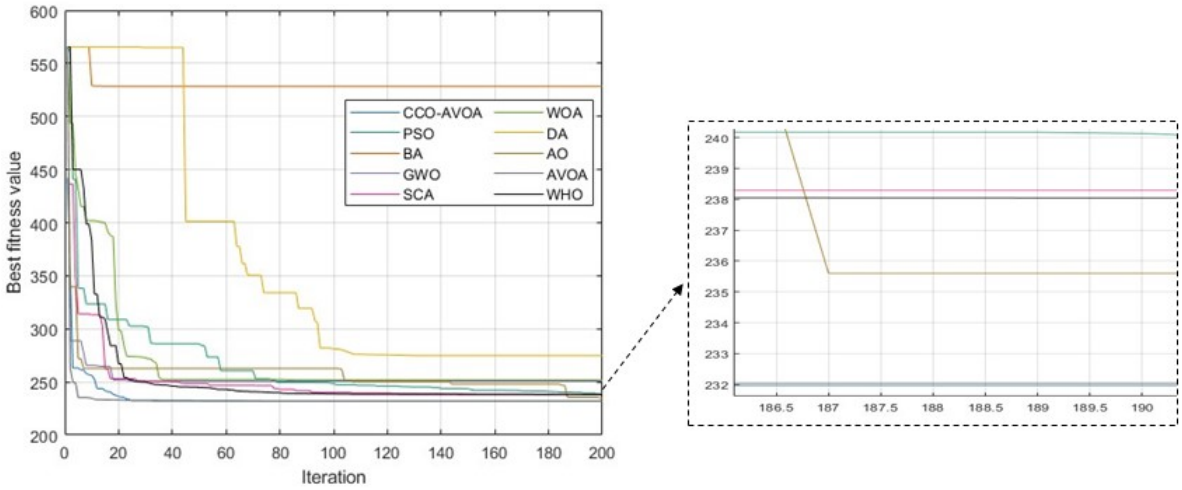


Figure 4.7: Example of convergence curves the second case for $n = 40$

Table 4.5: Fitness results of the second case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Best	331.92	**231.91**	242.91	234.31	232.07	232.06	232.05	232.39	231.92	231.93	232.23	
Worst	**232.1**	232.17	381.4	245.23	249.28	254.12	249.13	246.12	**232.1**	240.46	239.06	$n = 10$
Mean	232.02	**231.99**	286.5	237.78	235.64	238.84	240	235.25	232.04	232.46	233.98	
Std	**0.04**	0.06	38.05	2.68	3.68	5.25	3.48	3.1	**0.04**	1.53	1.51	
Best	**231.94**	232.1	284.78	240.85	232.05	233.44	232.86	232.43	231.95	232.68	232.82	
Worst	**232.13**	237.14	429.52	252.8	372.62	246.91	291.45	246.91	232.14	239.38	237.51	$n = 20$
Mean	**232.03**	233.07	360.25	244.35	269.97	239.03	249.21	237.35	232.04	236.31	234.39	
Std	0.05	1.57	29.8	2.64	57.96	3.21	11.75	4.4	**0.04**	2.4	1.27	
Best	**231.90**	232.46	373.25	242.33	232.67	236.55	242.25	232.53	231.91	232.05	232.81	
Worst	**232.11**	238.21	497.21	492.28	249.18	370.07	387.46	238.77	232.2	241.74	243.82	$n = 30$
Mean	**232.02**	236.19	421.58	366.38	236.83	251.75	292.23	234.49	232.06	237.01	237.44	
Std	**0.04**	1.96	31.66	70.91	3.01	24.67	40.56	1.07	0.05	2.32	3.63	
Best	**231.93**	232.83	381.5	270.1	232.08	238.92	244.19	232.62	231.94	232.49	232.7	
Worst	**232.21**	239.82	453.83	/	382.74	/	428.73	247.9	232.24	242.78	275	$n = 40$
Mean	**232**	237.15	410.69	/	344.6	/	305.74	234.56	232.06	237.85	237.28	
Std	**0.05**	1.69	15.93	/	55.14	/	48.99	2.6	0.07	2.57	7.37	

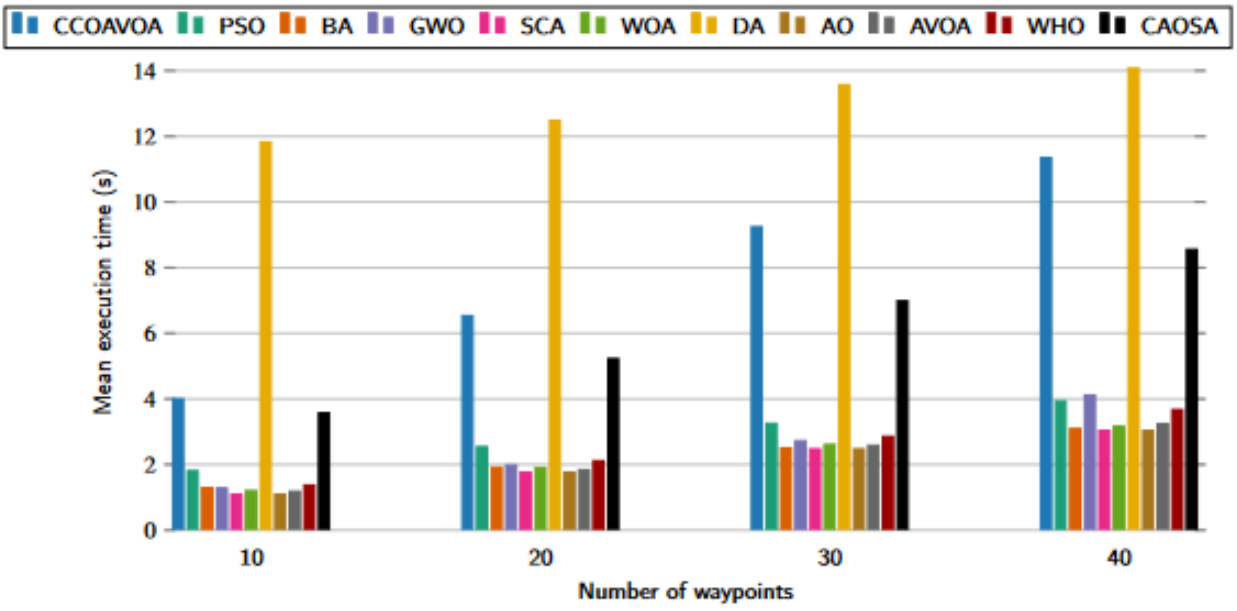


Figure 4.8: The average execution time in the second case

Table 4.6: Performance results in the second case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Path cost	928.06	**927.85**	1040	944.76	935.4	945.51	942.73	934.75	928.09	928.81	931.77	$n = 10$
Height cost	0.06	0.08	23.96	9.77	0.87	8.32	16.24	2.98	**0.05**	0.8	1.47	
Obstacles cost	3.61 e^{-4}	7.4 e^{-3}	5.59	2.59	6.31	1.53	1.02	3.26	**2.19** e^{-6}	0.23	2.68	
UAV's angle cost	**0**	**0**	76.43	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
Path cost	**928.04**	929.95	1152.62	962.33	1065.8	944.92	972.73	940.76	928.06	935.28	935.08	$n = 20$
Height cost	**0.09**	2.21	23.13	12.76	3.63	6.94	15.45	3.66	**0.09**	9.83	1.88	
Obstacles cost	1.85 e^{-4}	0.1	9.79	2.3	7.54	4.26	0.05	4.99	**2.36** e^{-5}	0.14	2.78	
UAV's angle cost	**0**	**0**	251.93	0	2.93	**0**	8.64	**0**	**0**	**0**	**0**	
Path cost	**927.99**	936.12	1437.5	1231.4	940.44	985.72	1017.4	933.19	928.08	939.22	942.83	$n = 30$
Height cost	**0.1**	8.58	12.93	14.47	1.14	9.31	17.43	1.59	0.14	8.71	5.08	
Obstacles cost	5.76 e^{-4}	0.07	5	22.11	5.72	9.94	2.32	3.19	**2.27** e^{-4}	0.12	1.87	
UAV's angle cost	**0**	**0**	230.91	197.52	**0**	2.04	131.74	**0**	**0**	**0**	**0**	
Path cost	**927.89**	940.89	1502.8	939.07	1365.5	1381	1022.2	933.24	928.03	943.72	936.9	$n = 40$
Height cost	**0.17**	7.68	9.59	0.65	7.72	8.65	16.03	1.94	0.19	7.66	3.51	
Obstacles cost	1.61 e^{-4}	0.05	4.12	/	4.15	/	2.65	1.94	**2.27** e^{-6}	0.01	3.62	
UAV's angle cost	**0**	**0**	126.26	21.74	1	11.38	182.1	1.11	**0**	**0**	5.09	

Performance analysis

Figure 4.8 illustrates the average execution time required for providing solutions by the ten algorithms. From the figure, we can see that the DA takes more time compared to the others to find its best solution in all cases and reaches the $14s$ for $n = 40$. The DA requires a lot of time to generate a solution which demonstrates its lack of exploring new areas. As for the proposed algorithm, its mean execution time exceeds the $10s$ only in the case of the high number of n. In other cases, the mean execution time is less than $10s$. These values can be considered good and acceptable given the solution provided in the context of UAV path planning. In addition, the CCO-AVOA approach explores three types of positions, including the original solutions, their opposite elite solutions, and the mutated solutions. This extensive search generates additional time compared to the original AVOA algorithm. Regarding the other algorithms, their average execution time is low and less than $5s$ in all cases.

Table 4.6 reports the average cost results in the case of using seven obstacles under a various number of waypoints. From the results, we can notice that the proposed CCO-AVOA approach achieves the best results regarding the path cost, in most cases. For $n = 20, 30, 40$, it provides the shortest path with the least amount of power source consumption. As for PSO, AO, AVOA, and WHO algorithms, their path costs are good and acceptable but less than the cost given by the proposed approach. In this case of obstacles, the WOA algorithm provides good path cost only in the case of $n \leq 20$. Otherwise, the path length is relatively high. The path cost given by GWO, SCA, and DA algorithms is optimal for $n = 10$. In other cases, their cost is increased and high. In this scenario, for all cases of n, the BA algorithm provides the worst path cost, which was expected from its fitness results.

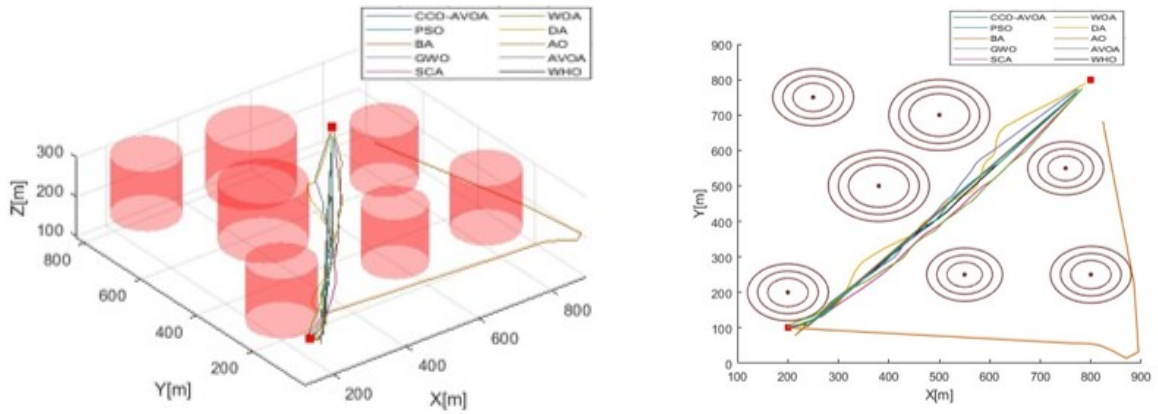
Regarding the height cost, we can notice from the reported results that our CCO-AVOA algorithm offers the smallest height cost. The solution given by our approach introduces the least variation in the height levels, which preserves the energy better. The AVOA algorithm also gives good results in this scenario. Its height cost is close to our approach's cost. Results provided by the AO algorithm are good in all cases of n where the height changes during the flight do not exceed $5m$. As for the SCA algorithm, its height cost is good only for $n \leq 30$.

The PSO algorithm provides good height cost in the case of $n = \{10, 20\}$. In other cases, we can notice that the cost is greater than $5m$. Regarding the WHO algorithm, its height cost is optimal only in the first case of n. BA, GWO, and DA algorithms provide, in all cases of n, an important and remarkable height cost that cannot be considered optimal. Their results involve additional energy consumption.
Concerning the obstacle cost, it can be noticed that our approach provides good results. Using our approach, the total crossing distance between the obstacles and the UAV's safety distance is less than $0.6mm$ in all cases. These results demonstrate that our proposed algorithm ensures UAV safety and responds better to the obstacle's constraint. In this scenario, the AVOA algorithm provides the least obstacle cost for all cases of n. Results given by PSO, DA, AO, and WHO algorithms are acceptable and less than 5 in all cases. We can see from the table that the obstacle cost given by both GWO and WOA algorithms is increased from one case of n to another. Both of them, in this case of obstacles, are not able to search and explore all the points (n) to avoid better obstacles. Both of them fail to ensure a safe path in the last case of n, where the problem is highly complex. Regarding the SCA algorithm, we can notice its instability during this case of obstacles. This behavior can be justified by the exploration-oriented nature algorithm, where it achieves its global optima in some cases of n. As expected from the fitness results, the BA algorithm provides remarkable results.
From the results given in table 4.6, we can see that our approach did not present any violation regarding the UAV's angle constraints. In all cases of n, the CCO-AVOA algorithm provides the best UAV's angle cost value. Therefore, our approach considers better energy optimization and UAV physical safety. PSO, SCA, AO, AVOA, and WHO algorithms offer good and acceptable results in this scenario. As for GWO and WOA algorithms, we can see that their results are not good in most cases. The GWO algorithm cannot handle the UAV's angle constraint for an important number of waypoints. For $n > 20$, its given path involves dangerous and aggressive turns threatening UAV safety. The WOA algorithm provides its worst results regarding the UAV's cost in the last case of n. In all cases of n, the BA algorithm gives the worst results compared to the other algorithms. We can see that the UAV's angle cost is important and unacceptable. Therefore, the BA algorithm is not suitable for this kind of scenario.

4.9.3 Case of ten obstacles

Fitness analysis

Table 4.7 summarized the fitness results given the algorithms in the case of using ten obstacles. We can see that the CCO-AVOA approach outperforms the other algorithms in most cases. Our approach achieves the best results with good stability due to its strong exploration capabilities. PSO, AO, AVOA and WHO algorithms perform well in this scenario according to their results. For all cases of n, the DA algorithm was able to offer a feasible



(a) The 3D view

(b) The top view

Figure 4.9: The path given by the algorithms in the second case for $n = 40$

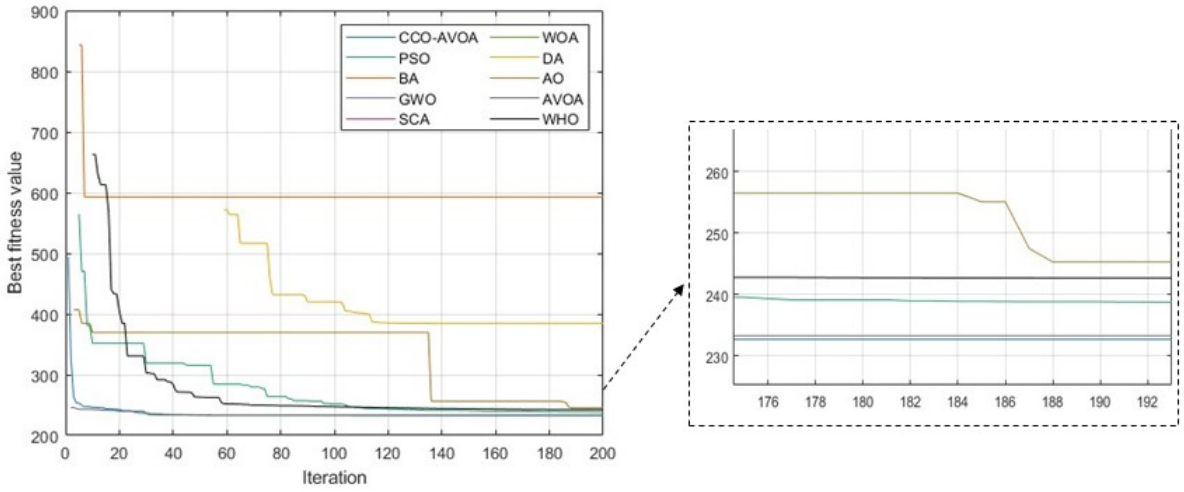


Figure 4.10: Example of convergence curves in the third case for $n = 40$

solution. However, it cannot be considered optimal due to its high fitness costs. We can notice that the DA algorithm shows less stability over cases which is explained by the lack of exploring the search area. GWO, SCA, and WOA algorithms present failure in some cases of n. In those cases, we can say that the algorithms were not able to find the corresponding waypoints. Therefore, the algorithms formed their path only from a few considered waypoints. Regarding the BA algorithm, we can see from its worst fitness that it cannot handle this environmental complexity. Regardless of the number of waypoints, the BA algorithm fails in some tests. Figure 4.10 shows the best fitness values over iterations given by the state-of-the-art algorithms. In this example, we can notice that the BA algorithm converges in earlier iterations to a suboptimal value. BA algorithm in this case considered only a few waypoints, which reduces the search dimension resulting in its quick convergence. We can also notice that the DA algorithm starts its convergence from the 60-th iteration. In this case, the DA algorithm was not able to explore and stagnates in the suboptimal solution. As for PSO, WOA, and WHO algorithms, their convergence becomes slower with iteration

Table 4.7: Fitness results of the third case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Best	232.26	232.26	244.2	240.53	233.98	233.65	233.75	235.63	232.3	**232.22**	233.29	$n = 10$
Worst	233.93	**233.16**	/	264.72	/	/	243.53	240.56	233.95	240.32	239.3	
Mean	233.04	**232.44**	/	248.29	/	/	239.66	235.67	233.09	234.74	235.22	
Std	0.62	**0.18**	/	6.65	/	/	2.58	1.12	0.65	2.58	1.26	
Best	**232.34**	232.72	344.61	248.31	233.61	234.57	239.21	234.71	232.64	237.21	234.17	$n = 20$
Worst	233.52	238.28	/	/	263.57	306.79	347.27	252.47	**233.5**	246.6	241.16	
Mean	**232.83**	234.88	/	/	244.74	248.26	265.41	238.44	233.01	237.21	236.66	
Std	0.31	2.21	/	/	7.79	11.44	23.91	3.02	**0.27**	2.98	1.59	
Best	**232.3**	233.98	376.99	245.17	233.97	234.91	242.97	234.7	232.56	233.61	235.57	$n = 30$
Worst	**233.47**	239.76	/	337	/	/	352.45	248.47	233.75	251.7	281.74	
Mean	**232.65**	237.41	/	275.2	/	/	285.67	238.52	233.23	239.91	241.13	
Std	**0.24**	1.73	/	26.05	/	/	30.64	2.49	0.29	3.58	6.4	
Best	**232.38**	234.75	399.91	249.83	234.04	242.88	256.61	234.73	232.83	236.15	236.24	$n = 40$
Worst	**233.52**	242.75	/	394.65	419.59	293.17	437.41	256.83	233.96	248.94	287.28	
Mean	**233.04**	239.09	/	295.81	286.92	255.45	343.59	238.09	233.27	241.32	242.25	
Std	0.72	1.68	/	27.71	55.55	11.65	51.84	3.68	**0.26**	2.31	8.52	

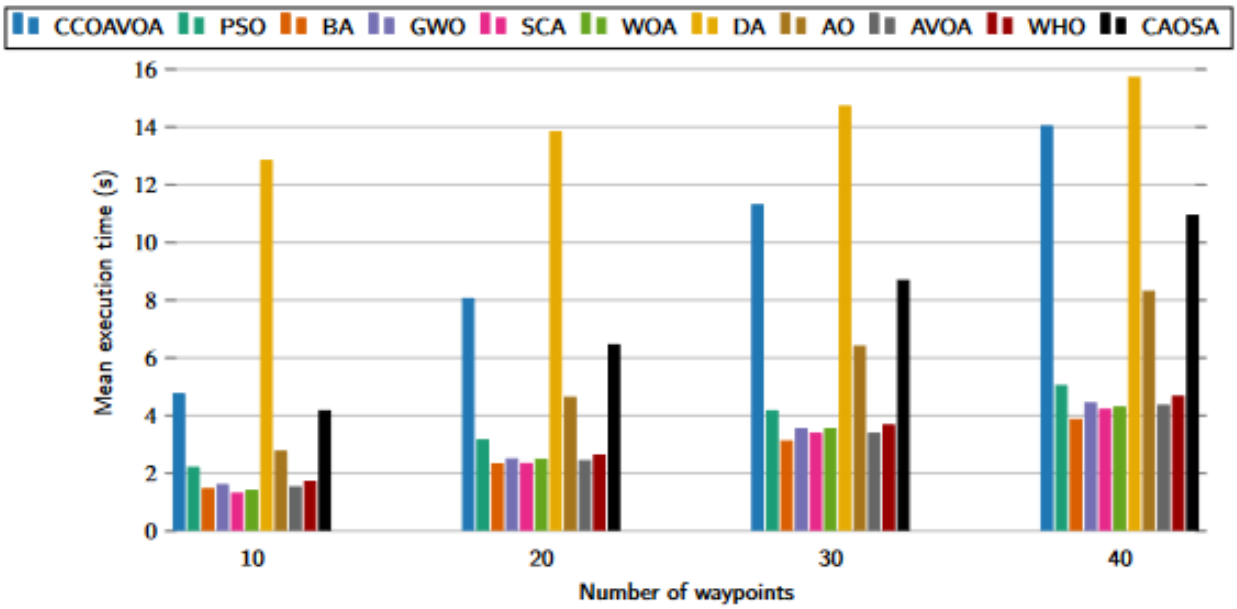


Figure 4.11: The average execution time in the third case

changes. The convergence of the AO algorithm is not steady in this case, which indicates its weakness in finding the global solution. The CCO-AVOA and AVOA algorithms in this scenario show the same behaviour in the convergence. However, in the earlier iterations, we can notice that the CCO-AVOA convergences to the optimal value before the original AVOA algorithm.

Performance analysis

Figure 4.11 illustrates the average execution time for the algorithms in the case of ten obstacles. From the figure, it can be seen that the DA algorithm takes the longest time to provide the solution in all cases. Our approach presents additional time compared to the original AVOA algorithm due to its extensive search for the global optimal solution, which involves both Elite opposition-based and Cauchy mutation strategies. Regarding the quality of the solution and the requirements of the UAV path planning, the execution time of our approach

Table 4.8: Performance results in the third case

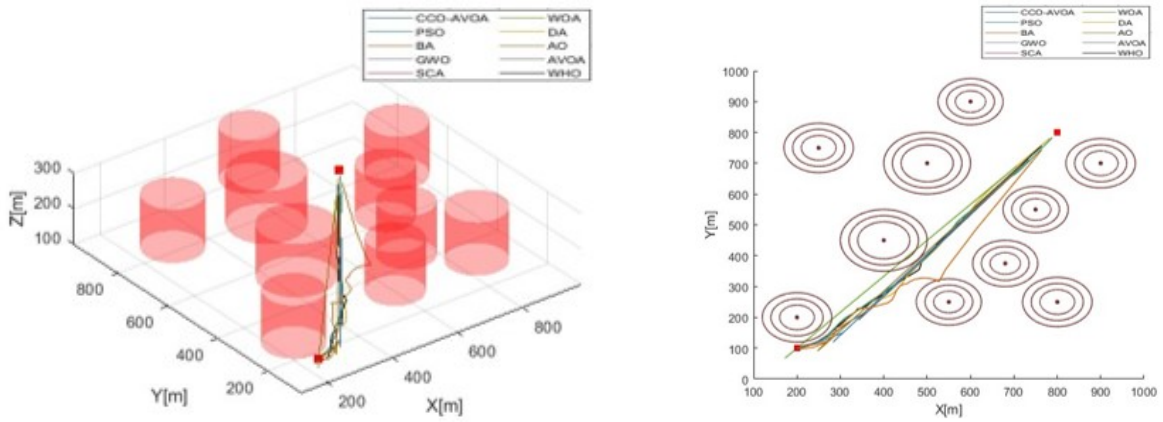
Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Path cost	932.11	**929.5**	997.08	965.67	925.63	929.13	944.36	938.1	932.13	933.78	936.6	$n = 10$
Height cost	**0.06**	0.25	25.2	17.44	0.23	2.24	13.07	1.62	0.22	3.87	2	
Obstacles cost	**5.6** e^{-3}	0.02	/	2.98	/	/	1.23	2.96	0.03	1.3	2.28	
UAV's angle cost	**0**	**0**	8.63	7.06	**0**	0.95	13.07	**0**	**0**	**0**	**0**	
Path cost	**930.93**	933.9	1233.1	925.33	956.19	963.76	996.22	945.14	931.7	939.2	939.77	$n = 20$
Height cost	0.39	5.52	19.65	0.76	5.79	11.07	18.43	4.02	**0.34**	9.49	3.56	
Obstacles cost	3.83 e^{-4}	0.1	/	/	16.98	6.18	2.01	3.55	**1.5** e^{-4}	0.15	3.38	
UAV's angle cost	**0**	**0**	314.64	3.17	**0**	12.03	44.96	**0**	**0**	**0**	**0**	
Path cost	**930.16**	939.87	1210.2	981.1	926.33	928.81	1010.5	946.37	932.03	947.03	951.05	$n = 30$
Height cost	0.43	9.68	29.15	15.1	0.61	1.55	18.72	3.81	**0.42**	12.61	7.37	
Obstacles cost	**2.8** e^{-4}	0.1	/	17.55	/	/	3.25	3.9	1.9 e^{-3}	0.1	2.73	
UAV's angle cost	**0**	**0**	1141.9	87.06	0	11.69	110.23	**0**	**0**	**0**	3.39	
Path cost	**931.52**	946.17	1239.6	982.98	972.76	977.87	1027.9	944.04	932.69	953.45	954.71	$n = 40$
Height cost	0.65	10.03	13.97	16.04	7.6	10.83	14.86	3.99	**0.37**	11.82	6.82	
Obstacles cost	**1.12** e^{-4}	0.15	/	25.43	45.72	14.63	6.34	3.44	3.3 e^{-4}	0.03	3.23	
UAV's angle cost	**0**	**0**	405.21	158.78	121.59	18.48	325.32	0.9	**0**	**0**	4.22	

is acceptable. The average execution time provided by the other algorithms is optimal and less than $7s$. Table 4.8 details the average performance results given by the algorithms in the case of ten obstacles. By seeing the average path cost, we can conclude that our approach outperforms the other algorithm in terms of providing the shortest feasible path in most cases. In spite of the fact that GWO and SCA algorithms respectively offer the least path cost for $n = 20$ and $n = 30$, their path can't be taken into consideration because they failed in handling the fundamental requirement of the UAV path planning. Both of them fail to ensure a safe path in those cases. The paths generated by PSO, AO, AVOA, and WHO algorithms are relatively good and satisfy the UAV path planning requirements. The DA algorithm produces the longest feasible path in this scenario. In cases where GWO, SCA, and WOA succeeded, their solutions involve additional distance compared to the others. In all cases, the BA algorithm presents the longest path, which cannot be taken into account due to its failure.

Concerning the height cost, we can notice that, in this case, the original AVOA algorithm achieves the smallest cost in most cases. Nevertheless, our approach provides good results. Both CCO-AVOA and AVOA algorithms offer stability during the flight, which preserves the energy source. PSO and WHO algorithms optimize the height cost only in the first case of n. In other cases, the cost is relatively increased. Results given by the AO algorithm can be considered good in all scenarios, where we can see that the average total height change is less than $5m$. GWO, SCA, and WOA algorithms provide unstable cost, where we can notice an important gap from one case of n to another. During this test, both BA and DA algorithms were not able to provide a good height cost.

Considering the obstacles' cost, the CCO-AVOA algorithm offers optimal results. Our approach can be considered the safest method in this complex scenario, which involves the least crossing between the safety distance and the obstacles' radius. After our approach, the original AVOA algorithm gives good results in the context of UAV safety. The results produced by PSO, DA, AO, and WHO algorithms are within the safety distance range. Therefore, these algorithms are suitable and applied in UAV path planning. In this scenario, we can say

that the GWO algorithm gives good results only in the first case of n, where the obstacles' cost is acceptable. In the remaining other cases, either it fails, or the total safety violation is important and dangerous. The performance of both the SCA and WOA algorithm is poor in this scenario, which makes them unsuitable and risky for a complex environment. Their instability can easily lead to failure and obstacle collision. According to the table results, the BA is not suitable for this case, whatever the number of waypoints.
By analyzing the UAV's angle costs, we can see that our approach presents optimal results in all cases. From the results, we can conclude that the path's turns provided by our approach do not exceed the maximum allowed angles in elevation and azimuth planes. Therefore, the angles' constraint is respected and considered. Algorithms such as PSO, AVOA, and WHO provide similar results as our approach in this context. The AO algorithm presents the non-zero cost only for $n = 40$, which can be negligible and acceptable. As for the rest, their cost results are high and involve UAV safety violations. Therefore, the results cannot be accepted, and the algorithms do not suit this scenario.



(a) The 3D view (b) The top view

Figure 4.12: The path given by the algorithms in the third case for $n = 40$

4.9.4 Case of thirteen obstacles

Fitness analysis

Table 4.9 represents the fitness results obtained by the algorithms in case of using 13 obstacles in the area. From the table, we can notice that the algorithms in this case are divided into 3 categories, including algorithms that totally fail to provide a safe path such as BA, GWO, and WOA algorithms. The second category includes PSO, SCA, DA, AVOA, and WHO algorithms, that at some tests were not able to generate a feasible path. Finally, algorithms that offer a collision free path, including AO, CAOSA, and our CCO-AVOA approaches. Comparing the results of these algorithms, we can see that the best approach is the CCO-AVOA algorithm, since it provides the smallest best, worst, and mean fitness values in most

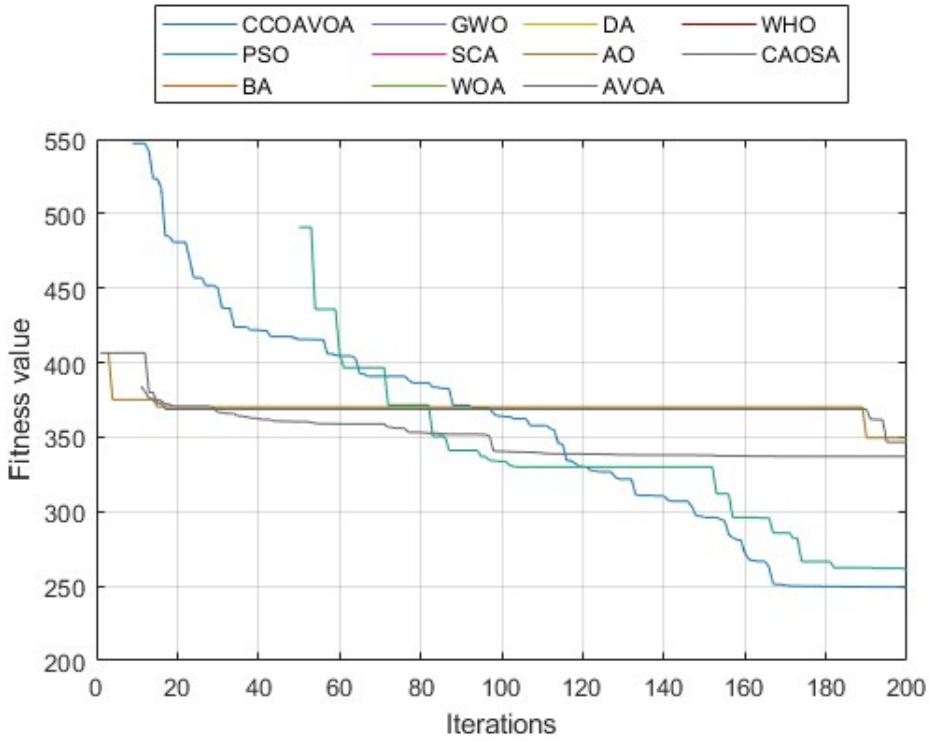


Figure 4.13: Example of convergence curves in the fourth case for $n = 40$

of cases. Regarding the standard deviation, its value is little high compared to the one given by both AO and CAOSA algorithms. But, it is acceptable and can be justified by the case complexity. In this case, both AO and CAOSA algorithms give good results in all cases of n. By comparing both of them, the CAOSA algorithm offers better fitness results compared to AO algorithm. Figure 4.13 displays the fitness cost over iterations given by the algorithms in case of using 13 obstacles in the area. From the figure, we can notice that only the convergence curve of the CCO-AVOA, AVOA, PSO, AO, and CAOSA algorithms is shown. It can be explained by the failure of the other algorithms to converge in this case. By comparing the displayed converge curves, we can see clearly that the best convergence is given by our approach. First, our algorithm converges in an efficient way during the search stages. In the beginning and in the middle of iterations, its convergence is efficiently slow due to the search of new area, which demonstrates its exploration capabilities. Before the end of iterations, our algorithm converges quickly to the best fitness value, which shows its exploitation efficiency. The convergence of PSO algorithm in case starts from the 50-th iteration, which explain its inability to find a solution before. In addition, its convergence is not steady. However, its best fitness value is good. The convergence behaviour of the AVOA, AO, and CAOSA algorithm in this scenario is similar and slow.

Table 4.9: Fitness results of the fourth case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Best	**235.92**	242.58	/	/	/	/	272.91	293.23	248.78	240.67	322.42	$n = 10$
Worst	354.79	/	/	/	/	/	/	372.94	353.92	/	367.46	
Mean	**266.21**	/	/	/	/	/	/	359.76	336.89	/	353.28	
Std	38.54	/	/	/	/	/	/	12.81	13.29	/	**8.74**	
Best	**237.05**	242.41	/	/	/	/	251.41	340.64	335.24	238.5	337.88	$n = 20$
Worst	352.36	/	/	/	/	/	/	373.35	/	**285.76**	371.49	
Mean	279.55	/	/	/	/	/	/	365.43	/	**247.32**	359.7	
Std	44.64	/	/	/	/	/	/	**7.77**	/	7.88	8.68	
Best	**240.28**	245.66	/	/	373.2	/	369.43	357.01	335.33	244.35	340.24	$n = 30$
Worst	**360.28**	/	/	/	/	/	/	373.91	/	/	372.6	
Mean	**323.06**	/	/	/	/	/	/	369.01	/	/	361.84	
Std	35.26	/	/	/	/	/	/	**3.73**	/	/	9.01	
Best	249.78	**247.12**	/	/	371.9	/	540.72	359.97	335.75	274.03	341.22	$n = 40$
Worst	**356.56**	/	/	/	/	/	/	380.18	/	/	374.31	
Mean	**336.14**	/	/	/	/	/	/	369.9	/	/	364.67	
Std	20.75	/	/	/	/	/	/	**3**	/	/	6.86	

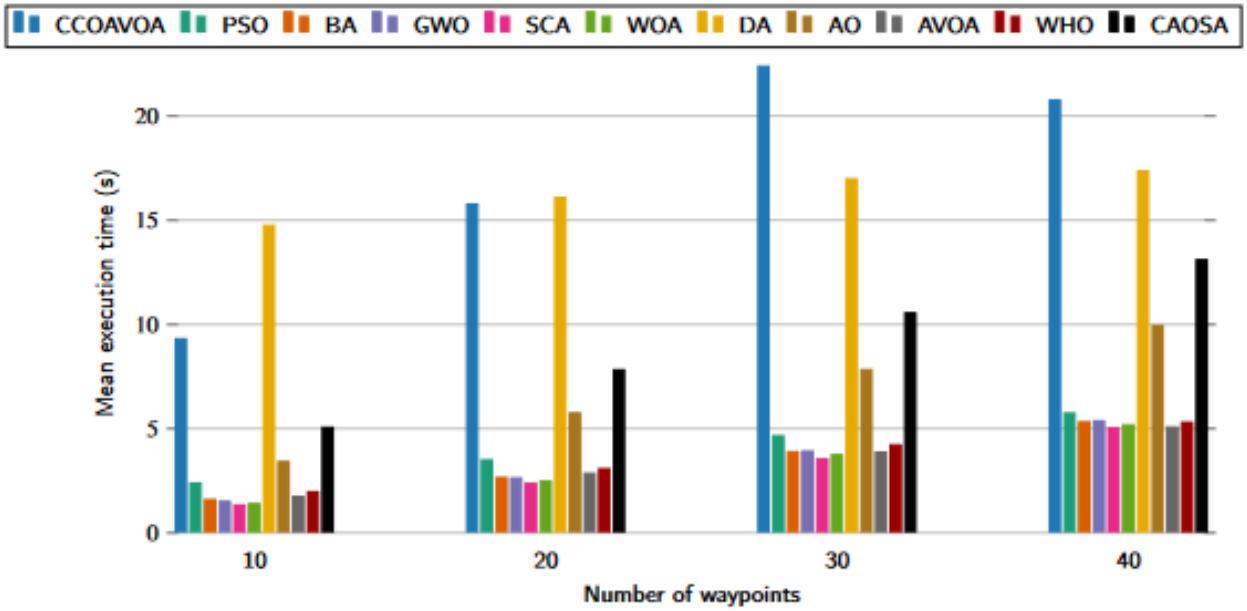


Figure 4.14: The average execution time in the fourth case

Table 4.10: Performance results in the fourth case

Cost results	CCO-AVOA	PSO	BA	GWO	SCA	WOA	DA	AO	AVOA	WHO	CAOSA	Number of waypoints
Path cost	1035.8	1006.9	**921.95**	**921.95**	**921.95**	**921.95**	1016.7	1343.3	1315.2	1059.6	1330.7	$n = 10$
Height cost	15.41	17.97	0	0	0	0	**4.32**	15.07	15.88	19.26	15.26	
Obstacles cost	**4.66**	/	/	/	/	/	/	16.09	12.51	/	14.74	
UAV's angle cost	8.95	29.8	**0**	**0**	**0**	**0**	28.7	64.55	3.96	161.64	52.38	
Path cost	1089	1129.3	**921.95**	**921.95**	**921.95**	**921.95**	958.5	1360.8	1310.3	970.62	1348	$n = 20$
Height cost	12.75	24.02	**0**	**0**	**0**	**0**	2.35	**11.91**	11.75	13.21	11.84	
Obstacles cost	5.68	/	/		/	/	/	15.92	/	1.54	14.15	
UAV's angle cost	10.81	449.88	**0**	**0**	**0**	**0**	25.14	73.08	**0**	3.91	64.85	
Path cost	1268.4	1058.1	**921.95**	**921.95**	930.81	**921.95**	930.66	1361	1030.3	992.03	1355.1	$n = 30$
Height cost	10.91	20.2	**0**	**0**	0.26	**0**	0.19	**10.62**	10.54	17.03	10.72	
Obstacles cost	**10.5**	/	/	**0**	/	/	/	17.48	/	/	14.44	
UAV's angle cost	2.45	493.85	**0**	**0**	1.88	**0**	1.86	86.91	**0**	35.93	67.12	
Path cost	1313.3	1019.3	**921.95**	**921.95**	930.91	**921.95**	931.23	1366	1313.4	1174.1	1358.1	$n = 40$
Height cost	10.86	15.71	**0**	**0**	0.22	**0**	0.66	**9.99**	9.47	20.46	9.77	
Obstacles cost	**13**	/	/	/	/	/	/	14.71	/	/	15.66	
UAV's angle cost	7.73	134.8	**0**	**0**	1.88	**0**	43.18	88.93	2.97	1359.9	75.16	

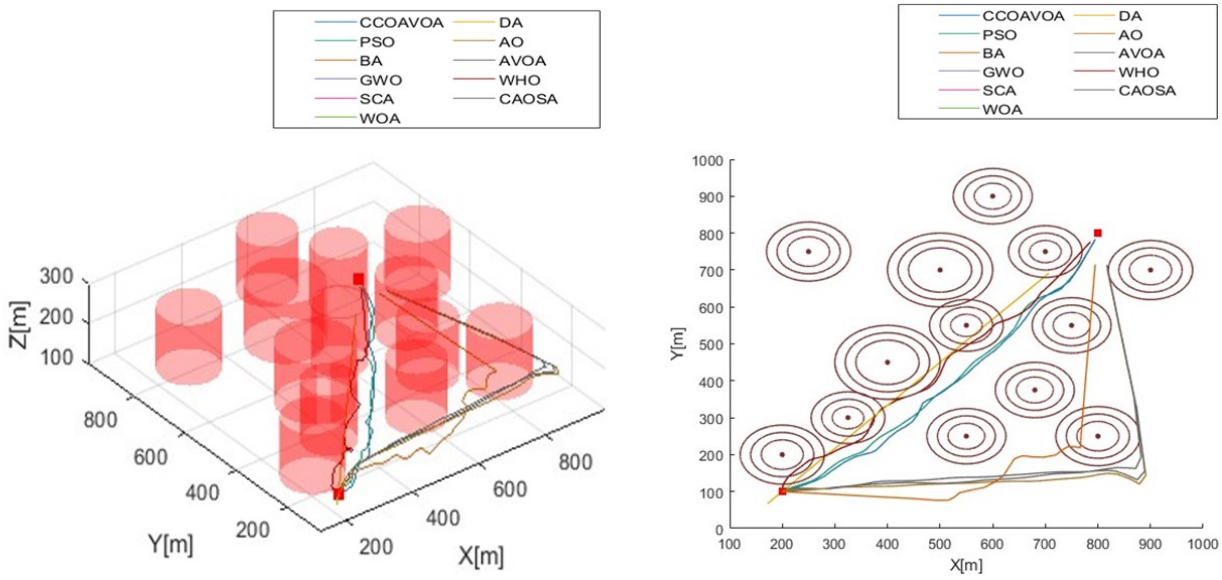
Performance analysis

Figure 4.14 illustrates the average time required by the algorithms to provide a path in this case of obstacles. From the figure, we can see that the average execution time given by the algorithm is high compared to the previous cases. It can be explained by the fact that this case is the most complicated which required additional processing. Concerning our approach, its mean execution time is included within the range $9.3 - 22.38$s. These values are relatively high due to extensive exploration search processed by Elite Opposition based learning and Cauchy mutation strategies introduced to the AVOA in the CCOAVOA algorithm. However, they can be acceptable since our approach offers the best results and they respect the UAV requirement. In this case, the execution time of DA algorithm is less than the previous cases due to its failure. Still, its execution time is greater than the one produced by our approach. The average execution time of the other algorithms does not exceed 6s in all cases, except AO and CAOSA algorithms, where we can notice a rise in the execution time from $n = 20$ and $n = 30$ for CAOSA and AO algorithms, respectively. The hybrid nature of the CAOSA algorithm produced additional time compared to original AO algorithm, which is expected. Table 4.10 details the performance results obtained from the different state-of-the-art algorithms in the last scenario. By analysing the average path cost, we can see that BA, GWO, SCA,WOA share the same cost, which is stable regardless the value of n. In fact, this value represents the length of the direct path between the source and destination, which is the shortest. Despite that, the path cannot be taken into account since it does not respect the obstacles constraint. The path cost given by PSO, DA algorithms cannot be taken into consideration due to their failure in some tests in all cases of n. AVOA and WHO algorithms are acceptable only in the case of $n = 10$ and $n = 20$, respectively, where WHO algorithm offers the shortest path. In other cases of n, both of them failed. The CCOAVOA, AO, and CAOSA algorithms provide optimal path in this scenario. However, our approach is the best one.

As for the height cost, we can see from the table that it is greater and different from the previous cases. The average height cost given by BA, GWO, WOA and WHO in all cases of n is null. Despite that, it cannot be consider optimal. This value is produced based on the failure against obstacles in the environment. SCA algorithm generates a significant cost in case of $n = \{30, 40\}$. PSO and DA algorithms also give a considerable in all cases of n. Still, the actual path are not safe and collide with obstacles. The cost given by both AVOA and WHO algorithms can be taken into consideration only in case of $n = 20$ and $n = 30$, respectively. In both cases, the path is feasible and the cost is acceptable. CCO-AVOA, AO, and CAOSA algorithms offer a significant height cost in all cases of n. By comparing there results, we can see that the provided average height costs are approximately similar in all cases of n. However, in this case, the smallest cost is given by AO algorithm.

By examining the obstacle cost, we can see that the cost given by PSO, BA, GWO, SCA,

WOA, DA, and WHO algorithms in this case is not specified, which is explained by their failure to maintain a safe distance from the obstacles. These algorithms cannot handle the complexity of this case and cannot ensure a safe path for the UAV. Therefore, they cannot be used. As for the AVOA algorithm, it only succeeded in the first case of n to offer a safe path. Still, the obstacle cost of AVOA is not the optimal one. By comparing the CCO-AVOA, AO, and CAOSA algorithms, the smallest and best cost is given by the proposed CCO-AVOA algorithm in all cases of n. Theses results demonstrate the efficiency of our approach against the complexity of the environment. CAOSA and AO algorithms also provide good results in this case, but the CAOSA algorithm gives better results than AO algorithm.
Regarding the UAV's angle cost, we can notice from the table that the smallest cost in this case is given by our approach as expected, which proves the ability of our approach in preserving both the UAV safety and power source. As for the results given by both AO and CAOSA algorithms, is high compared by the one provided by the CCO-AVOA algorithm. The cost given by BA, GWO, SCA, and AVOA is either small in some cases or null. The results seem good but they cannot be taken into consideration since they failed. PSO, DA, and WHO algorithms show a high and unacceptable angle cost.



(a) The 3D view

(b) The top view

Figure 4.15: The path given by the algorithms in the fourth case for $n = 40$

4.9.5 Comparison between cases

Figures 4.6a, 4.9a, and 4.12a illustrate the path given by the algorithms in three scenarios for 40 waypoints. The obstacles and radars are represented as cylinders with various radius. Figures 4.6b, 4.9b, and 4.12b display the $2D$ view of the given paths.

Fitness analysis

From the results, we can notice that both obstacles and waypoints number impact the UAV path planning process. Regarding the fitness results, from tables 4.3, 4.5 and 4.7, it can be concluded that the variation of fitness values is proportional to the number of obstacles. In fact, the Occurrence of obstacles in the environment gives rise to its complexity which explains the rise of fitness values. By analyzing each case, we can see that the effect on the number of waypoints n is also proportional to the fitness results. By increasing the number of waypoints which represents the size of the UAV path planning problem, the problem of complexity is increased. This fact is justified by the obtained results. In our tests, the proposed CCO-AVOA approach is suitable for both environment and problem complexity since it provides a safe path regardless of the number of obstacles and waypoints. As for GWO and WOA algorithms, we can say that both of them are suitable only for a few obstacles and fewer waypoints. Apart from this case, they both fail to provide a safe path which leads to UAV safety infractions. PSO, SCA, and WHO algorithms present a failure for the high number of waypoints ($n = 40$). So, It may be appropriate to use them for less number of waypoints.

Performance analysis

Concerning the execution time, we can see that the algorithms take more time to provide results when both the number of obstacles and waypoints are increased. In both cases, the process of UAV path planning is getting more complicated which requires additional time and computation to avoid obstacles and cover more points to form the best path.
Comparing the performance results provided in Table 4.4, 4.6, and 4.8, it can be seen that the different costs are changing from one case to another. We can notice that varying the number of the waypoints does not impact the path cost since the line formed between the source and destination remains the same regardless of the node's density. We can notice some changes in the path cost while n is varying in the case of using PSO, BA, SCA, GWO, WOA, and DA algorithms. As a result of their instability that affects the path cost. As for obstacles variation, we can that the path cost is proportional to the number of obstacles which is increased from one case to another. With the presence of obstacles within the line formed between the source and destination, the algorithms search for points to avoid them which are around them and separated from the line SE with a specific distance causing additional path length that impacts the path cost.
Concerning the height cost, we can notice that it is not impacted by the obstacles' number changes. In fact, the obstacle cost computation is based on the $2D$ distance between the UAV and the obstacle centre. Therefore, the UAV's height is not involved. As for the waypoints' number changes, we can say that the height cost is not relevant, where we can an instability in its variation.

Regarding the obstacles cost, it can be seen that it is impacted by both waypoints and obstacles number. To begin with, increasing the number of the waypoint improves the possibility of exploring more points by algorithms in the searching area. As a result, the distance between the obstacle and UAV candidate positions is computed with better precision and more complexity to ensure its safety. Some algorithms such as PSO, BA, GWO, SCA, and WOA cannot handle this level of complexity. As we can see, for a high number of waypoints, they are not able to find a safe path explaining their failure to explore all the waypoints. As for the impact of the number of obstacles, we can conclude that it is correlate with the obstacle cost. The presence of obstacles in the area reduces the probability to avoid them causing additional violations of UAV safety.
By analyzing the UAV's angle results, we can notice that it's impacted neither by the obstacle nor the node's number. In most cases, the algorithms can handle UAV manoeuvrability. The UAV's angle cost becomes important while using BA, GWO, SCA, WOA, and DA algorithms, only in complex cases where they provide important obstacles cost or fail to offer a feasible path.
Based on our simulation results, we have derived the following main conclusions:

- Among the studied SI in this work, CCO-AVOA is the one that in general achieved the best results for the UAV path planning problem. It has been the best approach in terms of fitness, path, obstacles, UAV's angle, and height costs in the majority of cases evaluated such as 4, 7, and 10 obstacles, and 10, 20, 30, and 40 waypoints. Only in the case of 7 obstacles and 10 waypoints, the PSO algorithm outperforms the proposed CCO-AVOA.

- The dominance of the objective function components changes with the change in the number of obstacles and waypoints. In the case of 7 obstacles and 10 waypoints, the obstacles cost is the most dominant component. Nevertheless, if 40 waypoints are used, the path cost is more dominant than the obstacle cost due to the fact that the available waypoints are enough for UAV safety. In the case of using 3 obstacles, obstacles cost is not relevant for different values of n. Finally, for 10 obstacles, the obstacle cost maintains its dominance and the path cost is not pertinent.

- Based on the path, UAV's angle, and height costs, the CCO-AVOA is the best approach in terms of energy efficiency

- The proposed CCO-AVOA algorithm outperforms the state-of-the-art meta-heuristics, especially in complex scenarios, where the difference is pertinent. BA, GWO, SCA, and WOA algorithms fail in a high number of obstacles and nodes, while PSO cannot handle high problem complexity.

4.10 Conclusion

In this chapter, we have proposed an efficient optimization technique called Chaotic Cauchy Opposition-based African Vulture Optimization Algorithm (CCO-AVOA) for solving the UAVs path planning problem in a 3D environment. The introduced CCO-AVOA is based on the integration of three strategies (i.e., chaotic map, Cauchy mutation, and Elite Opposition-based Learning) into the standard AVOA to enhance its optimization ability. In order to assess and investigate the effectiveness of CCO-AVOA, the experiments are conducted on 12 scenarios with various numbers of waypoints and threats, considering the fitness value, path cost, height cost, obstacles cost, UAV's angle cost, and execution time parameters. The proposed CCO-AVOA algorithm was compared with other meta-heuristics, such as classical AVOA, PSO, BA, GWO, SCA, WOA, DA, AO, WHO, and CAOSA, and it consistently outperforms them in 11 out of 12 scenarios.
In the next chapter, we will present a novel hybrid algorithm to solve the UAV path planning in a different environment than the previous one.

CHAPTER 5

A HYBRID APPROACH FOR SOLVING UAV PATH PLANNING PROBLEM

5.1 Introduction

This chapter details the application of our hybrid algorithm, called hybrid Chaotic Aquila Optimization with Simulated Annealing (CAOSA) algorithm to solve the UAV path planning problem. Starting by representing the environment area, where the UAV performs its flight. Afterwards, the objectives and constraints related to the subject are formulated into an objective function that can be solved by our optimization approach. Then, we will present the details of our approach which are based on the integration of the singer chaotic map into original Aquila Optimization and the application of the Simulated Annealing algorithm as a local search operator. Finally, we conclude this chapter with the simulation results and conclusion.

5.2 Environment model

The environment model is an important step in the path planning. It consists of describing mathematically the nature of the environment and the different objects presented when the UAV flies. To be closer to the real environment, in our work, we considered a 3D map build based on real elevation data as illustrated in Figure 5.1. Mathematically, the map is represented as follows:

$$Map = \begin{bmatrix} Xm_1 & Xm_2 & \dots & Xm_k \\ Ym_1 & Ym_2 & \dots & Ym_l \\ Zm_1 & Zm_2 & \dots & Zm_m \end{bmatrix} \tag{5.1}$$

Where Xm_1 and Xm_k donate the map limits over the X plane.Ym_1 and Ym_l represent the map limits over the Y plane. Zm_1 and Zm_m are the map limits over the Z plane.

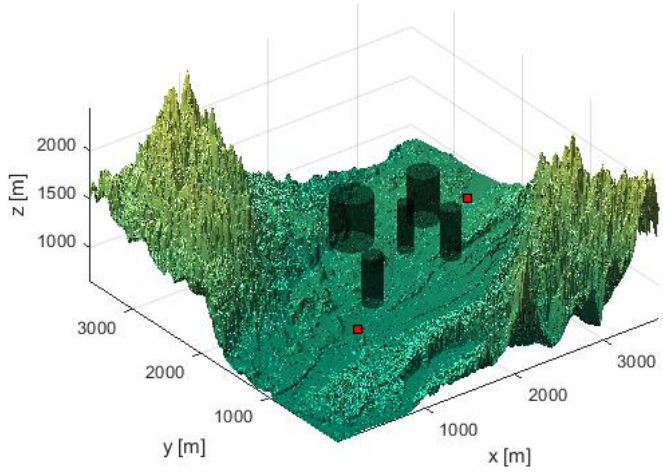


Figure 5.1: Environment model

For a given application, the UAV starts its flight from a starting point S located at (x_s, y_s, z_s) and reaches the final destination T at (x_t, y_t, z_t). During the flight, several objects are present in the area, which are across the UAV path. These objects can be either obstacles such as buildings and walls or threats such as radar and missiles. Both kinds of objects represent a threat to the UAV. The obstacles can damage the UAV if it collides with them. Radar and missiles can detect the presence of the UAV within their range, which represents a security violation, especially in military applications. In our work, we considered the integration of m the obstacles and threats $O_j, \quad j = \{1, 2, \ldots, m\}$ in the model. For simplification, we represented them as geometric shapes, which are cylinders. Each object O_j is located at (x_o^j, y_o^j, z_o^j) and characterized by its range R_j.
The objective of the UAV path planning is to determine the path from S to T without colliding with the m obstacles and threats. For processing, the path formed between the source S and the destination E is first divided into $(n+1)$ line segments $L_i, \quad i = \{1, 2, \ldots, n\}$, as shown in Figure 5.2. Each line segment connects two adjacent nodes located at (x_i, y_i, z_i) and $(x_{i+1}, y_{i+1}, z_{i+1})$ to form the total path $P = \{S, (x_1, y_1, z_1), (x_2, y_2, z_2), \ldots (x_n, y_n, z_n), T\}$. To handle better the UAV rotations, each line segment L_i is represented as a vector $(\overrightarrow{\mathrm{W_i W_{i+1}}})$ with a specific magnitude r_i and directions in the azimuth and elevation planes (θ_i, ϕ_i), respectively. So, the new total path can be represented as follows:
$P^s = \{S, (r_1, \theta_1, \phi_1), \ldots, (r_n, \theta_n, \phi_n), E\}$.

5.3 Objective function

The second step of the UAV path planning process consists of defining mathematically the parameters that impact it according to the applied algorithm. For meta-heuristic approaches, the set of parameters is introduced into a function, called an objective function, which gathers

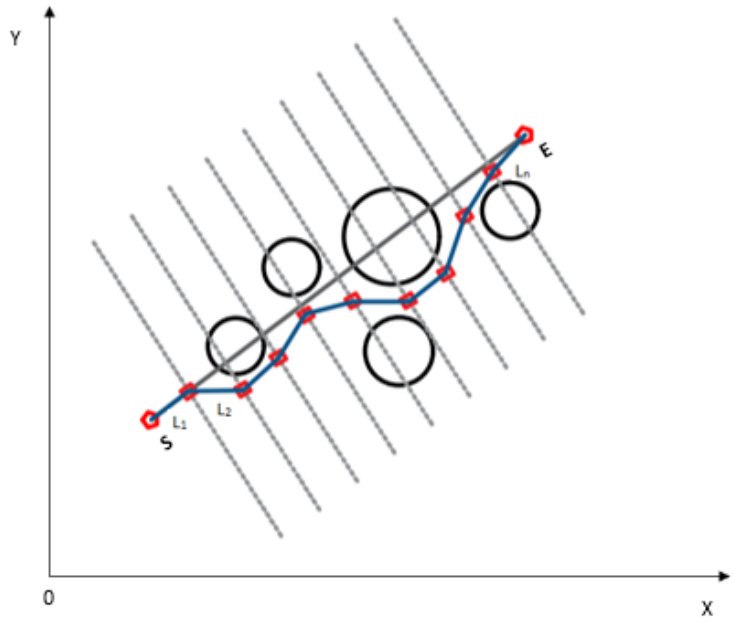


Figure 5.2: The 2D view of the path determination

objectives, constraints, or both.

The UAV starts its mission with a limited power source, which can be an electric battery, fuel, etc. During the flight, this power source is consumed with time, and it is related to several parameters, such as the travelled distance and the UAV height changes. In order to optimize this resource, we introduced the distance or the path length as a parameter to be minimized in our objective function. When the UAV performs changes in the height level, the power source consumption increased. To consider this issue, we defined height level boundaries and include them as an optimization parameter to our objective function. During the flight, other parameters also impact the UAV safety aspect. First of all, the objects and obstacles are risky for the UAV in case of collision. To avoid them, we added the related parameter as a constraint into consideration in the objective function design. Not only obstacles, but UAV rotations are also dangerous at some levels. brutal and successive turns increase the risk of the UAV's physical damage. Therefore, the UAV rotation is limited and optimized by our objective function.

Based on the parameters mentioned above, our objective function includes two objectives, which are: path length and height changes, and two constraints, including obstacle avoidance and the UAV rotation. therefore, the mathematical representation of the objective function is expressed as a linear weighted sum as the following equation:

$$Cost = \omega_1.f_P + \omega_2.g_O + \omega_3.f_H + \omega_4.g_s, \quad \omega_1 = \omega_2 = \omega_3 = \omega_4 = 0.25. \tag{5.2}$$

Where $Cost$ is the total cost. f_P and f_H are the cost related to the path length and the height changes objectives, respectively. g_O and g_s stand for the obstacles and UAV rotation

constraints. w is a weight coefficient in the range of $[0-1]$, which defines the importance of each parameter.

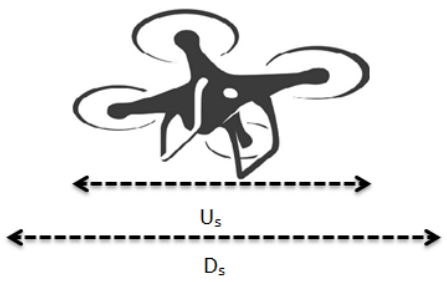
5.3.1 Path length

The path length parameter is defined as the travelled distance by the UAV between two points. The path length affects proportionally the power source consumed by the UAV. Therefore, the path length should be minimized. By definition, the path total path length is expressed as the following equation:

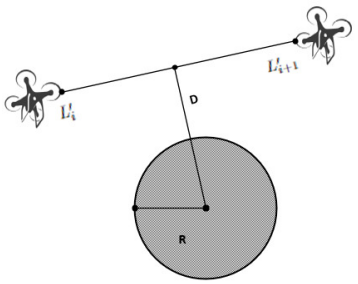
$$f_P = \sum_{i=0}^{n} \|\overrightarrow{\mathrm{W_i W_{i+1}}}\| \tag{5.3}$$

5.3.2 Obstacle avoidance

Obstacles represent a threat to the UAV's physical safety during the flight. for this reason, UAVs should avoid as much as possible colliding with them. In our model, the safety violation is calculated based on the 2D distance between the UAV in the projected segment $\overrightarrow{\mathrm{W'_i W'_{i+1}}}$ over the (O, X, Y) plane and the obstacle centre $(x_j^o, y_j^o), \quad j = \{1, 2, \ldots, m\}$ as shown in Figure 5.3b. To ensure better safety, we defined a safety distance D_s as shown in Figure 5.3a, greater than the UAV's size U_s for safety violation computation. The final obstacle avoidance cost can be calculated as follows:



(a) Safety distance representation



(b) Distance to obstacle computation

Figure 5.3: Obstacle cost determination

$$g_O = \sum_{i=0}^{n} \sum_{j=0}^{m-1} O_j(\overrightarrow{\mathrm{L_i L_{i+1}}}) \tag{5.4}$$

where $O_j(\overrightarrow{\mathrm{W_i W_{i+1}}})$ is the j-th obstacle cost at the $\overrightarrow{\mathrm{W_i W_{i+1}}}$ segment defined as the following equation:

$$O_j(\overrightarrow{\mathrm{L_i L_{i+1}}}) = \begin{cases} 0, & \text{if } D > R_j + D_s \\ \infty, & \text{if } D < R_j + U_s \\ D_s - D, & \text{otherwise} \end{cases} \tag{5.5}$$

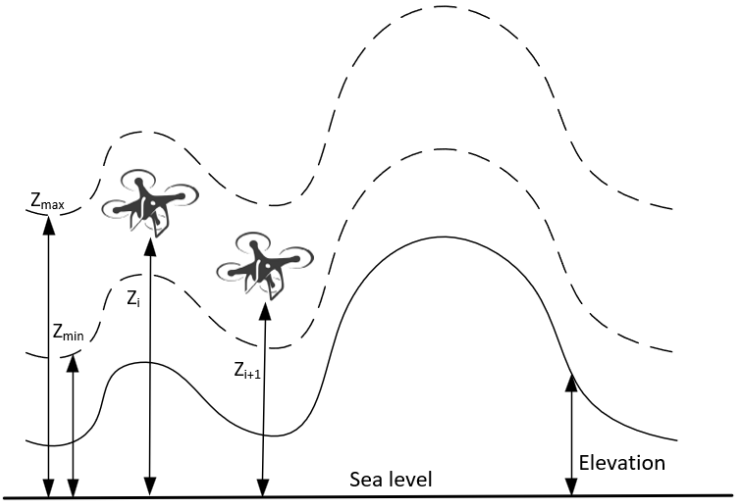


Figure 5.4: Height limits

Where D stands for the distance between the projected segment $\overrightarrow{W_i'W_{i+1}'}$ and the j-th obstacle radius R_j.

5.3.3 Height

Height is another parameter to consider in the UAV path planning problem. Both high and low height level impact the UAV during the flight. Low height level presents a risk to UAV safety due to the presence of obstacles and objects in the environment. High height level increases, on the one hand, the power source consumption. On the other hand, it might push the UAV out of the way. As a compromise, we delimited the UAV's height to ensure its safety, as illustrated in Figure 5.4. The constraint related to UAV's height is expressed as the following equation:

$$f_H = \sum_{i=1}^{n} Z_i \tag{5.6}$$

where Z_i donates the height cost at the i-th point, which is calculated as in Equation 5.7.

$$Z_i = \begin{cases} |z_i - z_{max}|, & \text{if } z_i > z_{max} \\ 0, & \text{if } z_{min} \leq z_i \leq z_{max} \\ |z_i - z_{min}|, & \text{if } 0 < z_i < z_{min} \\ \infty, & \text{if } z_i \leq 0 \end{cases} \tag{5.7}$$

Where z_{min} and z_{max} stand for the minimum and maximum height, respectively.

5.3.4 UAV's rotation axes

Since the UAV is a flying mobile robot, it is able to move in any direction in a 3D area. However, some movements and turns can cause damage to its physical system. Firstly, the large turns for UAVs will lead to their destruction. Secondly, successive turns will increase fuel consumption. Finally, brutal landing and taking off are risky for their own physical safety. For these reasons, we associated a cost related to the UAV manoeuvrability in both azimuth and elevation directions to guarantee its safety and resource optimization. The final UAV's rotation axes cost can be computed as the following equation:

$$g_A = g_C + g_T \tag{5.8}$$

Where g_C and g_T are the costs related to the manoeuvrability along the elevation and azimuth planes, respectively.
The climbing cost is defined as the cost related to the angle delimited by the path segments $\overrightarrow{\mathrm{W_iW_{i+1}}}$ and its projection on the horizontal plane (OXY), as shown in Figure 4.2. It is expressed as the following equation:

$$g_C = \sum_{i=1}^{n} \Psi i \tag{5.9}$$

With,

$$\Psi_i = \begin{cases} |\psi_i| - \psi_{max}, & \text{if } |\psi_i| > \psi_{max} \\ 0, & \text{Otherwise} \end{cases} \tag{5.10}$$

Where ψ_i donates the i^{th} climb angle calculated as in Equation 5.11. ψ_{max} is the maximum allowed climb angle.

$$\psi_i = \arctan\left[\frac{z_{i+1} - z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}\right] \tag{5.11}$$

the turning cost is the cost related to the angle formed by two adjacent path segments over the horizontal plane (OXY). Let $\overrightarrow{\mathrm{L'_iL'_{i+1}}}$ and $\overrightarrow{\mathrm{L'_{i+1}L'_{i+2}}}$ be the projection of $\overrightarrow{\mathrm{L_iL_{i+1}}}$ and $\overrightarrow{\mathrm{L_{i+1}L_{i+2}}}$ on (OXY). The cost related to the i-th turning angle Θi is given by the following equation:

$$g_T = \sum_{i=0}^{n-1} \Theta i \tag{5.12}$$

With,

$$\Theta_i = \begin{cases} \theta_i - \theta_{max}, & \text{if } \theta_i > \theta_{max} \\ 0, & \text{Otherwise} \end{cases}$$

The maximum turning angle θ_{max} is a specific parameter related to UAV's physical characteristics. θ_i represents the i-th turning angle expressed as follows:

$$\theta_i = \arccos \frac{\overrightarrow{W'_i W'_{i+1}} . \overrightarrow{W'_{i+1} W'_{i+2}}}{\|\overrightarrow{W'_i W'_{i+1}}\| . \|\overrightarrow{W'_{i+1} W'_{i+2}}\|} \tag{5.13}$$

5.4 Aquila Optimization (AO)

The Aquila Optimizer algorithm, proposed by Abualigah et al. [190] in 2021, is a recent Swarm Intelligence meta-heuristic that can be applied to solve many optimization problems. The Aquila algorithm mimics the behaviour of Aquila in hunting and catching the prey based on four strategies: expanded exploration, narrowed exploration, expanded exploitation, and narrowed exploitation [190].

1. Expanded exploration
   In this step, the Aquila makes a high soar with a vertical stoop. It is mathematically presented as follows:
   $$P_i^{(t+1)} = X_{best}^t \times \left(1 - \frac{t}{T}\right) + \left(X_m^t - X_{best}^t \times rand\right), \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \tag{5.14}$$
   where $P_i^{(t+1)}$ is the position of the current individual i at the $t+1$ iteration, P_{Best} is the current best position obtained at iteration t^{th} . N is the population size. $rand$ is a random number in the range of $[0, 1]$ and T is the maximum number of iterations. P_m^t is the position mean value of the current position at the current iteration t which is given by:
   $$P_m^t = \frac{1}{N} \sum_{i=1}^{N} X_i^t, \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \tag{5.15}$$
2. Narrowed exploration
   Narrowed exploration is the most used method by Aquila. Here, the Aquila catches the prey by contouring the flight with a short glide attack. This method is expressed in Equation (5.16)
   $$P_i^{(t+1)} = P_{best}^t \times Levy(Dim) + P_r^t + (y - x) \times rand, \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \tag{5.16}$$
   where $Levy(Dim)$ denotes the levy flight distribution of the space dimension Dim. P_r^t is a random solution. y and x represent the spiral shape of Aquila's movement in the search space which is presented mathematically as follows:
   $$\begin{aligned} x &= r \times \cos\theta \\ y &= r \times \sin\theta \end{aligned} \tag{5.17}$$

Where

$$r = r_1 + U \times D_1$$

$$\theta = -\omega \times D_1 + \frac{3 \times \pi}{2}$$

r_1 is a fixed value in the range of [1-20]. U and ω take small values fixed to 0.00565 and 0.005, respectively. D_1 is an integer number from 1 to the space dimension Dim.

3. Expanded exploitation
   In this method, the Aquila performs a low flight with a slow descend attack. It is formulated in Equation (5.18).

$$\begin{aligned} P_i^{(t+1)} = &\left(P_{best}^t - P_m^t\right) \times \alpha - rand \\ &+ ((UB - LB) \times rand + LB) \times \delta, , \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \end{aligned} \tag{5.18}$$

   where α and δ represent the exploitation parameters adjustment. UB and LB are the upper and lower bounds, respectively.

4. Narrowed exploitation This method is called walking and grabbing prey. It is used for hunting large prey. It is expressed as follows:

$$\begin{aligned} P_i^{(t+1)} = &QF \times P_{best}^t - (G_1 \times P(t) \times rand) \\ &- G_2 \times Levy(Dim) + rand \times G_1, \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \end{aligned} \tag{5.19}$$

   where QF represents the quality function which is represented in Equation (5.20). G_1 and G_2 are Aquila's motions and the flight slope, respectively. They can be expressed using Equations (5.21) and (5.22).

$$QF = \frac{2 \times rand - 1}{\sqrt{1 - T}} \tag{5.20}$$

$$G_1 = 2 \times rand - 1 \tag{5.21}$$

$$G_2(t) = 2 \times (1 - \frac{t}{T}), \quad t = 1, \ldots, T \tag{5.22}$$

The pseudo-code of the Aquila Optimization algorithm is presented in 3

5.5 Simulated Annealing (SA)

Simulated Annealing, introduced in 1983 by Kirkpatrick et al. [191], is a single-based meta-heuristic used for solving optimization problems. It is inspired by the annealing theory,

Algorithm 3 The pseudo-code of the Aquila Optimization algorithm

Initialize AO parameters: Maximum number of iterations T, Population' size N, Dimension Dim, $\alpha, \delta, etc.$
Initialize the population of AO: $P_i (i = 1, 2, ..., N)$
while $(t < T)$ **do**
    Calculate the fitness value $F(P(t))$
    Determine the best solution P_{best}
    for $i = 1, 2, \ldots, N$ **do**
        Update the mean value of the current solution $P_m(t)$
        Update x, y, G_1, $G_2(t)$, $Levy(Dim), etc.$
        if $t \leq (\frac{2}{3}.T)$ **then**
            if $rand \leq 0.5$ **then**
                Update the current solution $P(t)$ using Equation (5.14)
                Evaluate the fitness value $F(P(t))$
                Update the solutions $P(t)$ and P_{best} according to the fitness value
            else
                Update the current solution $P(t)$ using Equation (5.16)
                Evaluate the fitness value $Cost(P(t))$
                Update the solutions $P(t)$ and P_{best} according to the fitness value
            end if
        else
            if $rand \leq 0.5$ **then**
                Update the current solution $P(t)$ using Equation (5.18)
                Evaluate the fitness value $Cost(P(t))$
                Update the solutions $P(t)$ and P_{best} according to the fitness value
            else
                Update the current solution $P(t)$ using Equation (5.19)
                Evaluate the fitness value $Cost(P(t))$
                Update the solutions $P(t)$ and P_{best} according to the fitness value
            end if
        end if
    end for
    $t = t + 1$
end while
return The best solution P_{best}

which simulated the cooling process of metal atoms. SA starts with an initial solution X and Temperature Tmp. For each iteration t, SA searches the neighbour of the current solution P' by making a disturbance. This solution is accepted only in two cases: Firstly, if it minimizes the value of the objective function ($Cost(P') < Cost(P)$). Secondly, if the Boltzmann probability $B_p = e^{\Delta Cost/Tmp}$ is greater than a rand value, in the case where $Cost(P') > Cost(P)$. At the end of the iteration, the temperature Tmp decreases with a cooling factor Cf. This process is repeated until reaching the maximum number of iterations. The pseudo-code of the SA algorithm is presented in Algorithm 4

Algorithm 4 The pseudo-code of Simulated Annealing algorithm

Initialize SA parameters: Initial Temperature Tmp_0, cooling factor Cf and nbre of neighborhoods ns.
Generate initial solution P
while $(t < Maximum\ number\ of\ iterations)$ **do**
    while $(it < n)$ **do**
        Generate a random neighbor P' of P
        Calculate $\Delta Cost = Cost(P') - Cost(P)$
        Generate a random uniform variable r
        if $(\Delta Cost < 0)$ **then**
            $P = P'$
        else
            if $(\exp^{-\Delta Cost/Tmp} > r)$ **then**
                $P = P'$
            end if
        end if
        $it = it + 1$
    end while
    $Tmp = Tmp.Cf$
    $t = t + 1$
end while
return The best solution P_{best}

5.6 Chaotic strategy

Many meta-heuristic algorithms use in their process random values that need to be adjusted. Generally, these arbitrary values are generated randomly without a particular distribution. The main idea is to replace these random initialization variables with chaotic functions. This can enhance the searching capability and increases the convergence rate. Furthermore, chaos presents particular and important characteristics such as regularity, non-periodicity, ergodicity, and unpredictability [192]. Several chaotic maps are available in the literature. The ten well-known chaotic maps are the Logistic map, Tent map, Sine map, Gauss map, Sinusoidal map, Chebyshev map, Piecewise map, Iterative map, Circle map, and Singer map. [193]. The Singer map is one of the most representative chaotic functions generating a more uniform initial value between [0, 1]. So, we adopted the Singer map to replace the random variable that controls Aquila's position. It is mathematically expressed in Equation (5.23) [193] as follows:

$$y_{t+1} = \mu(7.86y_t - 23.31y_t^2 + 28.75y_t^3 - 13.302875y_t^4), \quad t = 1, \ldots, T$$
$$\mu = 1.07 \quad (5.23)$$

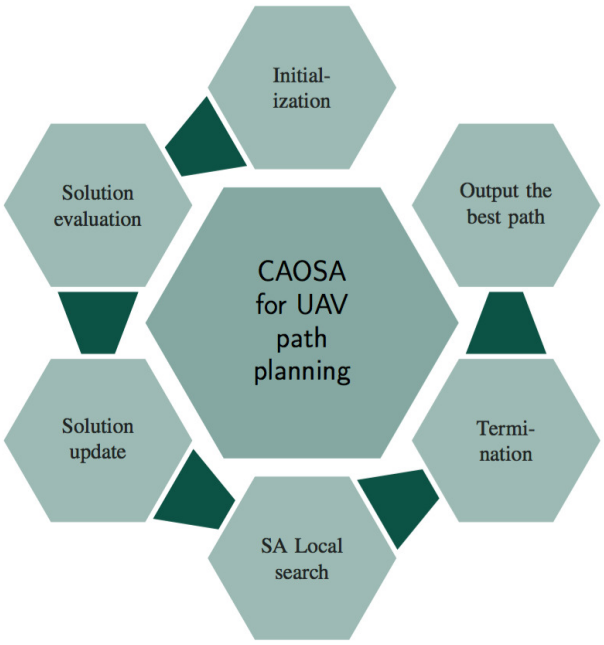


Figure 5.5: The CAOSA algorithm process for the UAV path planning

5.7 The proposed CAOSA algorithm for solving the UAV path planning problem

This section details the application of the proposed CAOSA algorithm in the context of the UAV path planning problem. To bring significant improvement to the original AO algorithm, two main strategies are introduced. First, the random variable, which controls the search type selection is replaced by the singer chaotic map. This strategy enhances the solution diversity and does not allow repetitive solutions. As a second improvement, the SA algorithm is applied as a local search method to improve the exploitation capabilities and search for better solutions.

Algorithm 5 describes the pseudo-code of the CAOSA algorithm.

The implementation of the CAOSA algorithm, as shown in Figure 5.5, includes six main steps: population initialization, position evaluation, position update, applying local search, termination, and output of the best path.

5.7.1 Population initialization

Population initialization represents the first step of implementing the CAOSA algorithm for the UAV path planning. In this step, the candidate positions of the UAV are initialized in the 3D environment within the range $[X, Y, Z]$, where X, Y, and Z are the environment limits over x, y, and z axes, respectively. Mathematically, the candidate positions are expressed as follows:

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \vdots \\ (x_N, y_N, z_N) \end{bmatrix} \tag{5.24}$$

Where N is the population size.
For better search efficiency, the positions are transformed to the new coordinate, the new coordinate positions are represented as the following equation:

$$P^n = \begin{bmatrix} (r_1, \theta_1, \phi_1) \\ (r_2, \theta_2, \phi_2) \\ \vdots \\ (r_N, \theta_N, \phi_N) \end{bmatrix} \tag{5.25}$$

In this step, the random control parameter R is initialized with the initial value of the singer chaotic map. Therefore, R is defined as follows:
$R = y(0) = 0.7$

5.7.2 Position evaluation

After the initialization phase, each position P_i is evaluated according to the objective function defined in Equation 5.2. Based on the fitness result, the best position obtained is selected according to the following equation:

$$P_{best} = \arg\min Cost(P) \tag{5.26}$$

5.7.3 Position update

In this step, the CAOSA processes the positions update using different types of search controlled by two parameters: $(\frac{2}{3}T)$ and R. If the number of iteration t is less than $(\frac{2}{3}T)$, either equation (5.14) or (5.16) is used. Otherwise, Equation (5.18) or (5.19) is applied. The selection between the equation within the same case of t is controlled by the R parameter based on its value. Therefore, enhancing the randomness of the R value with the Singer chaotic map allows the diversity and exclusivity of positions, which is the goal of the first improvement in the CAOSA algorithm.
After the position update, the new position is evaluated and the best position is determined as in Equation 5.26 for each iteration.

5.7.4 Apply local search

To improve the exploitation search of the Chaotic AO algorithm, the SA algorithm is applied to search for potential better solutions in the area. Firstly, the SA algorithm takes the best position P_{best} found in the previous step as an initial solution. Afterwards, the SA algorithm searches in the neighbourhood of P_{best} for a better solution based on objective function evaluation as in Equation 5.2. If SA finds better positions, the P_{best} is updated by the new position found. This process is repeated until the maximum number of iterations is reached.

5.7.5 Termination

At this point, the maximum number of iterations is reached. The CAOSA algorithm finished the processing of finding the best UAV positions and the best path is given.

5.7.6 Output the best path

Finally, the best path found is transformed to the original cartesian form. The final best path is expressed in the following format:
$P_{best} = \{S, (x_{b1}, y_{b1}, z_{b1}), (x_{b2}, y_{b2}, z_{b2}), \dots (x_{bn}, y_{bn}, z_{bn}), T\}$.

5.8 Simulation results

This section presents the performance evaluation of the proposed CAOSA algorithm for solving the UAV path planning problem. To demonstrate the effectiveness of the proposed approach, the CAOSA algorithm was compared to nine well-known meta-heuristics such as: Simulated Annealing (SA), Particle Swarm Optimization (PSO) [194], Bat Algorithm (BA) [195], Firefly Algorithm (FA) [196], Grey Wolf Optimization (GWO) [197], Sine Cosine Algorithm (SCA) [198], Whale Optimization Algorithm (WOA) [199], Drangonfly Algorithm (DA) [200], and original AO algorithms. The algorithms were evaluated based on the fitness, path cost, and execution time results using different scenarios as described in Table 5.3 and 5.2. The algorithms parameters setting are detailed in Table 5.1. All simulations were conducted on MATLAB R2021b software with the computer processor Intel Core i7 2.90GHz, RAM 32 GB, and 64-bit Windows 11 operating system. The reported results represent the average of 50 independent runs.

5.8.1 Convergence analysis

Case 1

Table 5.4 summarizes the fitness results produced by the algorithms in the first scenario. From the table, we can notice that the proposed CAOSA approach offers the best results

Algorithm 5 The pseudo-code of CAOSA for UAV path planning

Load map and path planning parameters: Start and target points, maximum turning and climbing angles
Transform the original coordination into a new coordinate system
Initialize AO and SA parameters
Initialize the population of AO: P
Initialize Singer map: R
while ($t < Maximum\ number\ of\ iterations$) **do**
    Calculate the fitness value $Cost(P(t))$
    Determine the best solution P_{best}
    for $i = 1, 2, \ldots, N$ **do**
        Update the mean value of the current solution $X_m(t)$
        Update x, y, G_1, G_2, $Levy(Dim), etc.$
        if $t \leq (\frac{2}{3}.T)$ **then**
            if $R \leq 0.5$ **then**
                Update the current solution $P(t)$ using Equation (5.14)
                Evaluate the fitness value $Cost(P(t))$
                Update the solution $P(t)$ and the best solution P_{best} according to the fitness value
            else
                Update the current solution $P(t)$ using Equation (5.16)
                Evaluate the fitness value $Cost(P(t))$
                Update the solution $P(t)$ and the best solution X_{best} according to the fitness value
            end if
        else
            if $R \leq 0.5$ **then**
                Update the current solution $P(t)$ using Equation (5.18)
                Evaluate the fitness value $Cost(P(t))$
                Update the solution $P(t)$ and the best solution P_{best} according to the fitness value
            else
                Update the current solution $P(t)$ using Equation (5.19)
                Evaluate the fitness value $Cost(P(t))$
                Update the solutions $P(t)$ and the best solution X_{best} according to the fitness value
            end if
        end if
    end for
    Call SA on P_{best} as initial solution
    Update the best solution P_{best}
    Update R using Equation (5.23)
    $t = t + 1$
end while
return The best solution P_{best}
Transform P_{best} into original coordinates

Table 5.1: Algorithms parameters

Algorithm	Parameter	Value
CAOSA	Exploitation adjustment α	0.1
	Exploration adjustment β	0.1
	Initial temperature Tmp_0	0.025
	Final temperature Tmp_f	0
	Cooling Factor Cf	0.99
SA	Initial Temperature Tmp_0	0.025
	Final temperature Tmp_f	0
	Cooling Factor Cf	0.99
PSO	Inertia maximum weight ω_{Max}	0.9
	Inertia minimum weight ω_{Min}	0.4
	Acceleration parameter C_1	2
	Acceleration parameter C_2	2
BA	Minimum frequency f_{Min}	0
	Maximum frequency f_{Min}	2
	Initial loudness A_0	1
	Initial pulse emission rate r_0	1
	Loudness constant α	0.5
	Emission rate constant γ	0.5
FA	Light absorption coefficient γ	1
	Initial light intensity coefficient I_0	2
	Initial attraction coefficient β_0	2
	Mutation coefficient	0.2
	Mutation coefficient damping ratio	0.98
GWO	Control parameter a_{min}	0
	Control parameter a_{max}	2
SCA	Control parameter r_1	[2,0]
WOA	Control parameter a_{min}	0
	Control parameter a_{max}	2
DA	Inertia maximum weight ω_{Max}	0.9
	Inertia minimum weight ω_{Min}	0.4
AO	Exploitation adjustment α	0.1
	Exploration adjustment β	0.1

Table 5.2: Path planning parameters

Parameter	**Value**
N	50
T	200
n	30
U_s	0.302 m
D_s	0.906 m
z_{min}	550 m
z_{max}	750 m
ψ_{max}	45°
θ_{max}	45°

regarding the best, worst, mean and standard deviation fitness. In this case, we can conclude that the best path is given by our approach. Moreover, the standard deviation results demonstrate the good stability of the CAOSA algorithm, which reflects its strength while dealing with the UAV path planning constraints. AO, SCA, and WOA algorithms provide good fitness results in this case of obstacles. However, the standard deviation value of both SCA and WOA is a little high and may lead to local optima stagnation. GWO, DA, FA, PSO, and BA algorithms provide present high fitness results, which proves their poor

Table 5.3: Description of experiences

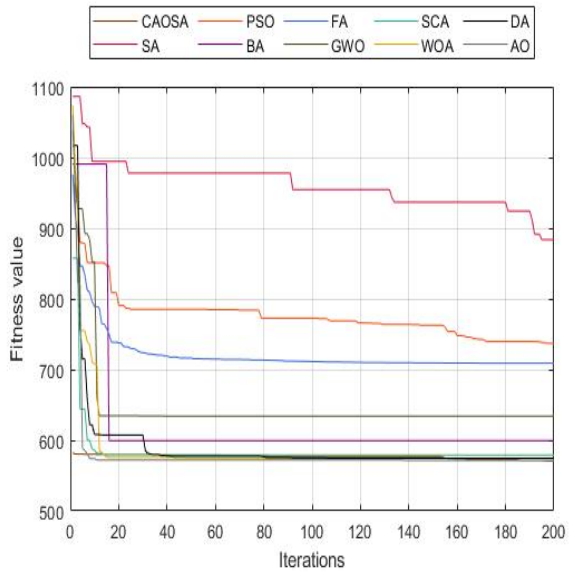
Cases	Start point	End point	Obstacles Center	Obstacles Radius
1	(1000,1000,600)	(3000,2000,700)	[1400,1300]	100
			[2500,2000]	150
			[2200,1200]	100
			[1700,2000]	200
			[2000,1600]	75
2	(500,800,600)	(2500,2500,700)	[1500,1300]	100
			[2500,2000]	150
			[2200,1200]	100
			[1700,2000]	200
			[2000,1600]	75
			[1000,1500]	100
			[800,1000]	80
			[2200,2300]	75
3	(500,800,600)	(2500,2500,700)	[1500,1300]	100
			[2500,2000]	150
			[2200,1200]	100
			[1700,2000]	200
			[2000,1600]	150
			[1000,1500]	100
			[800,1000]	80
			[2200,2300]	75
			[1400,1600]	80
			[2150,1900]	75

performance. Moreover, their instability shows their easy trapping into local optima and may fail in offering a feasible path. In this scenario, the SA algorithm provides the worst fitness value, which explains by its local search behaviour and inability to explore better solutions.

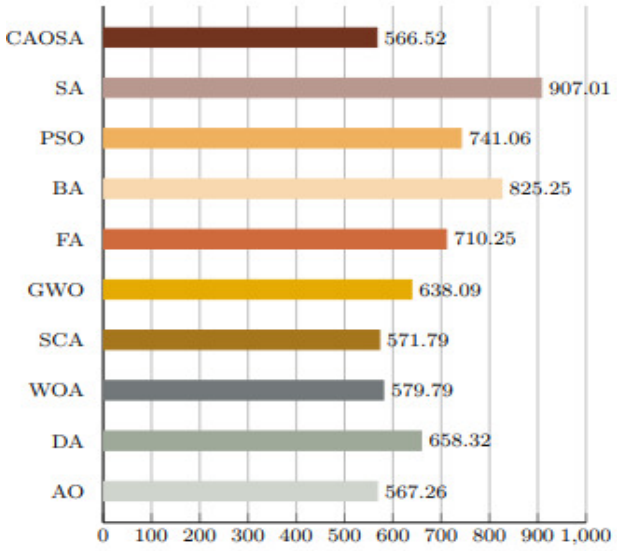
Figure 5.6a illustrates the best fitness value given by the algorithms over the iterations. We can see that the BA algorithm converges quickly to its best solution in the earlier stage of iterations. However, it converges to a sub-optimal value due to the lack of exploration. Therefore, the convergence of the BA algorithm cannot be considered optimal. As for PSO and SA algorithms, we can see that both of them showed the same convergence behaviour. Their convergence is slow and requires more iterations to reach the best solution. The convergence of the FA algorithm is slow at the beginning of iterations due to the exploration search stage and becomes slow before the maximum number of iterations is reached. In this case, we can say that the FA algorithm reaches its best solution. Still, its solution is not the optimal one. GWO shows the same behaviour as the BA algorithm. it stuck to local optima and converges to a sub-optimal solution in the earlier iterations. As for DA, SCA, WOA, and AO algorithms, we can see that their convergence is slow and it requires additional iterations. Regarding the proposed CAOSA approach, we can see that in the earlier iterations, its convergence is slow due to the exploration of new solutions in the area inherited by the exploration behaviour of the AO algorithm. Before the end of iterations, the proposed approach achieves the best solution which demonstrates its good exploitation capabilities adopted from the SA algorithm. Therefore, we can say the convergence speed of the COASA algorithm is efficiently fast. Moreover, the CAOSA approach outperforms in terms of convergence efficiency, since it provides the best fitness over iterations, as shown in Figure 5.6b.

Table 5.4: Results obtained from different algorithms in the first case

Fitness	**CAOSA**	**SA**	**PSO**	**BA**	**FA**	**GWO**	**SCA**	**WOA**	**DA**	**AO**
Best	**565.1**	675.75	716.96	573.81	703.29	577.09	565.99	565.58	565.26	566.01
Worst	**570.73**	2084.1	996.25	1100.3	721.058	768.42	610.1	665.3	660.1	571.41
Mean	**566.52**	907.01	741.06	825.25	710.25	638.09	571.53	579.79	658.32	567.26
STD	**1.51**	227.89	37.61	174.40	2.48	27.34	12.67	17.41	103.99	1.57



(a) The convergence curve of the compared algorithms in the first case



(b) The average cost in the first cases

Figure 5.6: Fitness results of the first case

Case 2

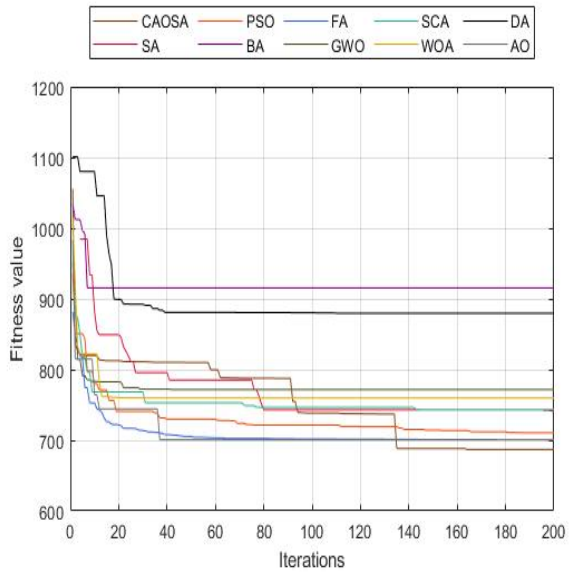
Table 5.5 reports the fitness results obtained by the algorithm in the case of using 7 obstacles in the area. Table results show that the CAOSA approach provides the best best, worst, and mean fitness values compared to the other algorithms. For the standard deviation, the PSO algorithm gives the smallest value. Still, the stability of our algorithm is good since the standard deviation value is not great. The results given by the AO and PSO are acceptable. SCA, GWO, WOA, DA, and BA algorithms show high fitness values, that demonstrate their poor performance and the lack of efficiency in this scenario. Except for the PSO algorithm, their instability is clearly determined by the great standards deviation values. As for FA and SA algorithms, they provide the worst results and their inability to handle the UAV path planning, in this case, is proved by their failure at some tests to offer a feasible path. In this case, we can say that both SA and FA were trapped in the local optima.

Figure 5.7a displays the converge curve of the algorithms in the second case of obstacles. From the figure, we can notice the DA, WOA, GWO, and AO algorithms converge to their best fitness value in the earlier iterations. However, their convergence can not be considered

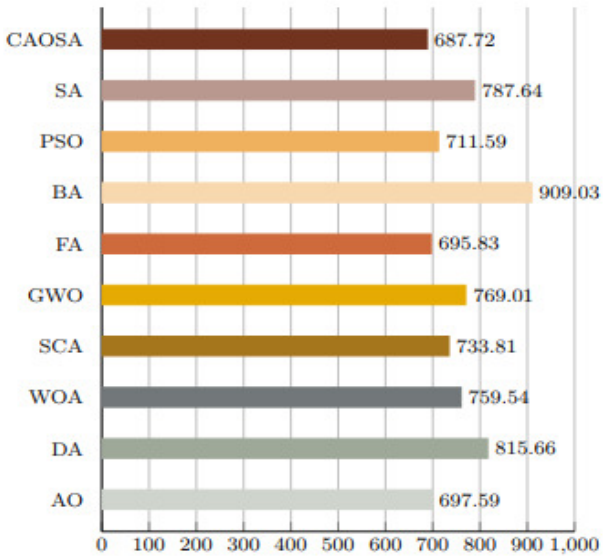
as optimal since they did not provide the best fitness value. Moreover, their quick and earlier convergence demonstrates their lack of finding a better solution. PSO and FA algorithms share the same convergence variation, where we can see that their convergence is steady in the earlier stage of iteration, and become stable before the maximum iterations are reached. The convergence of both SCA and SA algorithms is slow and the best fitness cost is not reached yet. Regarding the proposed approach, we can notice that its convergence is efficiently fast, where it is slow in the exploration search and fast in the exploitation search. This behaviour demonstrates its ability to balance between the type of search. In terms of the fitness value, our approach provides the best value over the iterations as shown in Figure 5.7b.

Table 5.5: Results obtained from different algorithms in the second case

Fitness	**CAOSA**	**SA**	**PSO**	**BA**	**FA**	**GWO**	**SCA**	**WOA**	**DA**	**AO**
Best	**677.39**	713.06	700.52	720.44	688.39	731.61	713.06	683.22	680.39	683.82
Worst	**713.42**	-	740.56	1155.4	-	808.74	769.57	838.12	1018.1	731.45
Mean	**687.72**	787.64	711.59	909.03	695.83	769.01	733.81	759.54	815.66	697.59
STD	9.08	190.92	**7.34**	101.93	165.28	17.10	21.62	33.1	80.28	8.65



(a) The convergence curve of the compared algorithms in the second case



(b) The average cost in the second cases

Figure 5.7: Fitness results of the second case

Case 3

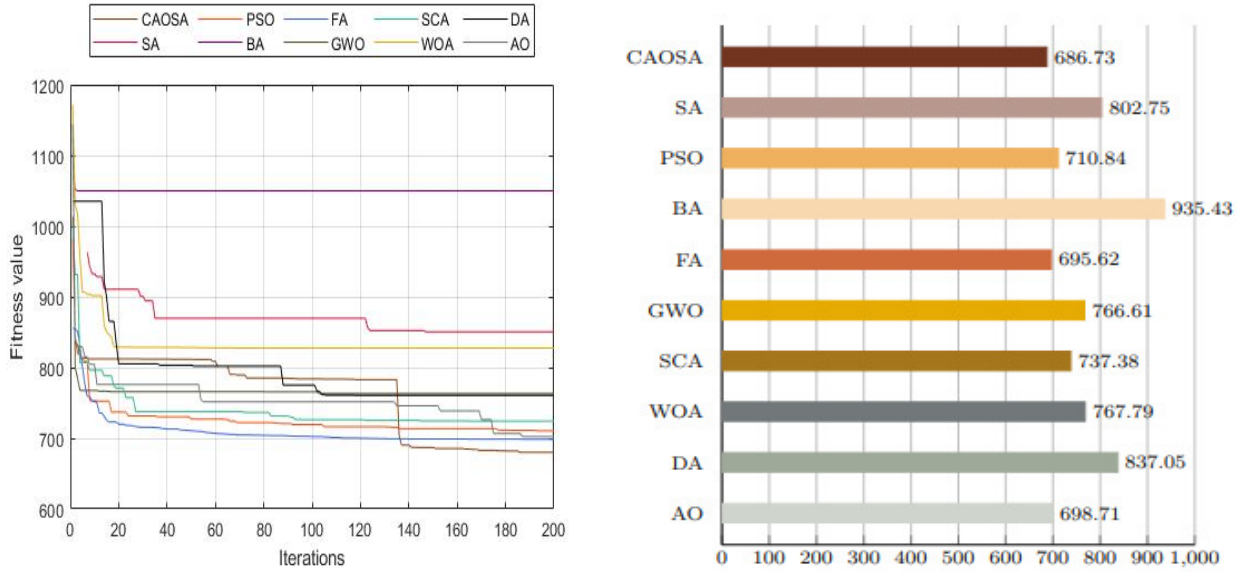
Table 5.6 represents the fitness results given by the algorithm in the last case of obstacles. From the table, we can see that our approach offers the best fitness results compared to the other algorithms. Our CAOSA approach produces the smallest best, worst, and mean fitness

values. As for the standard deviation, the PSO algorithm provides the smallest value. Still, the standard deviation of our approach is good and closer to the one given by PSO. The CAOSA fitness results prove its efficiency in optimizing the UAV path planning problem with good stability. The AO algorithm fitness values are relatively high. But, they are acceptable. Fitness results given by SCA, GWO, and WOA algorithms are high and not optimal, which reflects their poor performance in this scenario, even though they are to provide a safe path. Both DA and BA algorithms present unacceptable fitness results. Therefore, both of them are not able to optimize the UAV path planning objectives and constraints. As for both SA and FA algorithms, they are not suitable for the UAV path planning problem. Because, at some tests, they both failed to provide a feasible and collision-free path.

Figure 5.8a displays the best fitness value over the iterations given the ten algorithms. From the figure, we can notice that the BA algorithm is the fastest algorithm in convergence rate. However, it converges to a non-optimal value due to its lack of exploration search. The convergence of WOA, DA, GWO, and AO algorithms is also quick. We can see that they converge early before half of the stopping criteria. Their convergence behaviour proves their inability to avoid the local optima stagnation. As for the FA algorithm, we can see that its convergence is steady in the earlier stage of iterations, showing its good exploration ability. However, before the ending, the FA algorithm is not able to explore better, which is shown by the convergence stability. The PSO algorithm shows the same behaviour as FA at the beginning of the iterations. However, the PSO algorithm keeps looking for better solutions and does not reach its best solution yet. As for the SA algorithm, its convergence is relatively quick in the beginning. But, it stabilizes at an earlier point of iterations. The convergence of the SCA algorithm is slow and requires more iterations. As for our CAOSA algorithm, its convergence is relatively slow in the beginning due to the strong performance in the exploration search and becomes fast and stable before the end of iteration, which demonstrates that the best solution is reached. In addition to the convergence speed, the CAOSA algorithm is the best algorithm, considering the fact that it offers the best fitness cost as shown in Figure 5.8b.

Table 5.6: Results obtained from different algorithms in the third case

Fitness	**CAOSA**	**SA**	**PSO**	**BA**	**FA**	**GWO**	**SCA**	**WOA**	**DA**	**AO**
Best	**676.73**	745.77	700.15	707.20	688.71	726.63	693.69	714.69	683.15	681.95
Worst	**704.30**	-	727	1111.9	-	804.27	775.5	827.46	1046	733.22
Mean	**686.73**	802.75	710.84	935.43	695.62	766.61	737.38	767.79	837.05	698.71
STD	6.14	122.87	**5.93**	81.06	97.46	15.93	17.29	27.36	70.21	13.58



(a) The convergence curve of the compared algorithms in the third case

(b) The average cost in the third cases

Figure 5.8: Fitness results of the third case

5.8.2 Performance analysis

Case 1

Table 5.7 reports the average path cost and execution time given by the algorithms in the case of using four obstacles. From the table, we can see that the proposed approach provides the smallest path cost compared to the other algorithms. AO algorithm in this case gives also a good path cost, which is closer to the one given by the CAOSA algorithm. As for SCA, WOA, and GWO algorithms, they give good and acceptable path costs. The other algorithms provide high path cost, whereas the SA algorithm gives the worst cost value. They show bad performance results, as expected from their fitness results.
Regarding the average execution time, we can notice that the FA algorithm shows the highest value, which exceeds the $61s$. This execution time cannot be considered acceptable, since the UAV resources are limited and should be optimized. The execution time required for the DA algorithm is a little high but acceptable. As for our approach, its execution time is relatively superior to the rest of the algorithms, due to the hybrid nature of the algorithm. The local search step in the CAOSA algorithm generates additional processing time, which explains the time difference gap with the original AO algorithm. As for the other algorithms, their mean execution time does not exceed $4.5s$.

Case 2

Table 5.8 summarises the mean path cost and execution time provided by the algorithm in the second case of obstacles. As shown in the table, the best path cost, in this case, is given

Table 5.7: Path cost and execution in the first case

Algorithm	**Path cost**	**Execution time**
CAOSA	**2275.29**	6.37
SA	3332.28	**1.86**
PSO	2939.68	2.93
BA	2901.23	2.27
FA	2852.49	61.74
GWO	2498.11	2.27
SCA	2319.14	2.08
WOA	2333.84	2.16
DA	2588.2	11.79
AO	2277.75	4.38

Table 5.8: Path cost and execution in the second case

Algorithm	**Path cost**	**Execution time**
CAOSA	**2747.41**	8.07
SA	3057.49	**2**
PSO	2846.28	3.62
BA	3131.30	2.87
FA	2783.26	77.56
GWO	3046.28	3.03
SCA	2918.41	2.76
WOA	2988.02	2.81
DA	3065.49	11.72
AO	2772.76	5.78

by our CAOSA approach. Its performance result demonstrates its efficiency in providing the shortest path under environmental complexity and UAV constraints. In this case, the difference gap with the other algorithms is clear, where the cost given by AO is not closer to the one given by our approach. But, still, the AO path cost is good and acceptable. As for the other algorithms, they produce high path costs, which reflect their bad performance in this kind of environment complexity.

As for the execution time, the results show that the FA algorithm gives the worst result, which exceeds the $77s$. FA algorithms take a longer time to produce the path, which is not acceptable in the UAV path planning context. The DA algorithm also takes a longer time, less than FA. But, higher than the other algorithm and exceed the $11s$. Our CAOSA algorithm produces less execution time compared to both FA and DA algorithms. However, it requires additional time than the rest algorithms due to the supplementary process of searching for better solutions. As for the rest, their average execution time is good and within the range of $[2 - 5.78]$.

Table 5.9: Path cost and execution in the third case

Algorithm	**Path cost**	**Execution time**
CAOSA	**2742.69**	9.32
SA	3090.94	**2.41**
PSO	2843.28	4.12
BA	3264.96	3.31
FA	2782.44	91.26
GWO	3035.89	3.54
SCA	2929.1	3.21
WOA	3032.38	3.26
DA	3180.15	12.19
AO	2752.03	6.77

Case 3

Table details the performance results given by the algorithms in the last case of obstacles. We notice from the table that our CAOSA approach outperforms the other algorithms in terms of path cost, as expected. The CAOSA algorithm maintains its best performance even in this case of environment complexity, which demonstrates its efficiency. In this case, AO provides good results, and also FA when it does not fail. As for the other algorithm, their path cost is high compared ti the one given by our approach. In this case, we can conclude that the other algorithm are not able to balance between finding the shortest path length and staying far from obstacles. Their lack is reflected on their performance and clearly noticed in the high environment complexity. Considering the execution time, we can see that the FA algorithm is the worst algorithm also in this case, where its average processing time almost reaches the $100s$, which is exorbitant. the DA algorithm also produced a high average execution time, but less than the one given by the FA algorithm. As for the rest original algorithms, their average execution time does not exceed $7s$. Our hybrid CAOSA algorithm provides an execution time slightly high than the original AO algorithm, due to additional processing performed by the local search algorithm. Nevertheless, it is acceptable.

Comparison between cases

Figure 5.9 displays the average path length provided by the algorithm in all cases. From the figure, we can notice that the number of obstacles affects the UAV path planning problem. By comparing the three cases, we can see that the path length is proportional to the number of obstacles. In fact, the presence of additional obstacles in the line formed between the start and destination points requires the algorithms to find points outside the obstacle range, but not far from the segment line. Generally, these points are around the obstacles. The gap between the line formed by the start and destination and the obstacles' nearby points represent the path length gap.

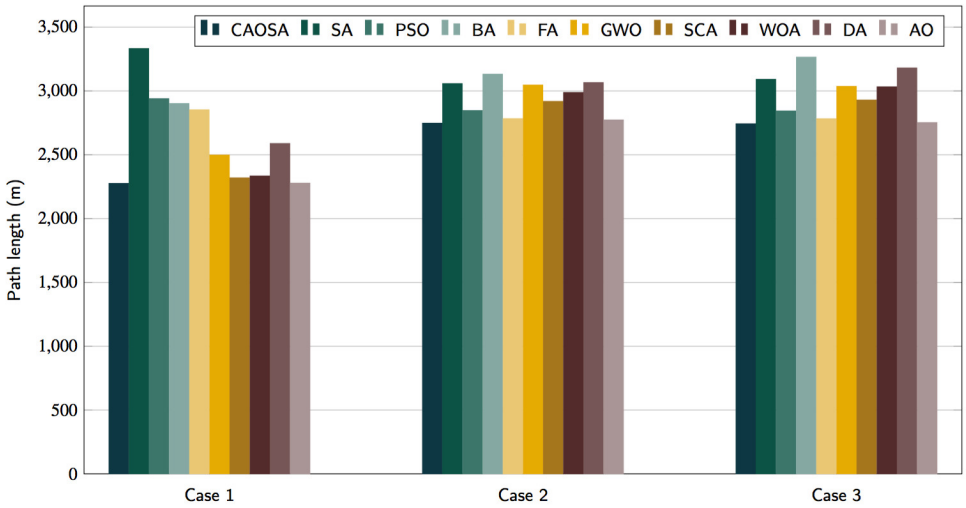


Figure 5.9: The average path cost produced in all cases

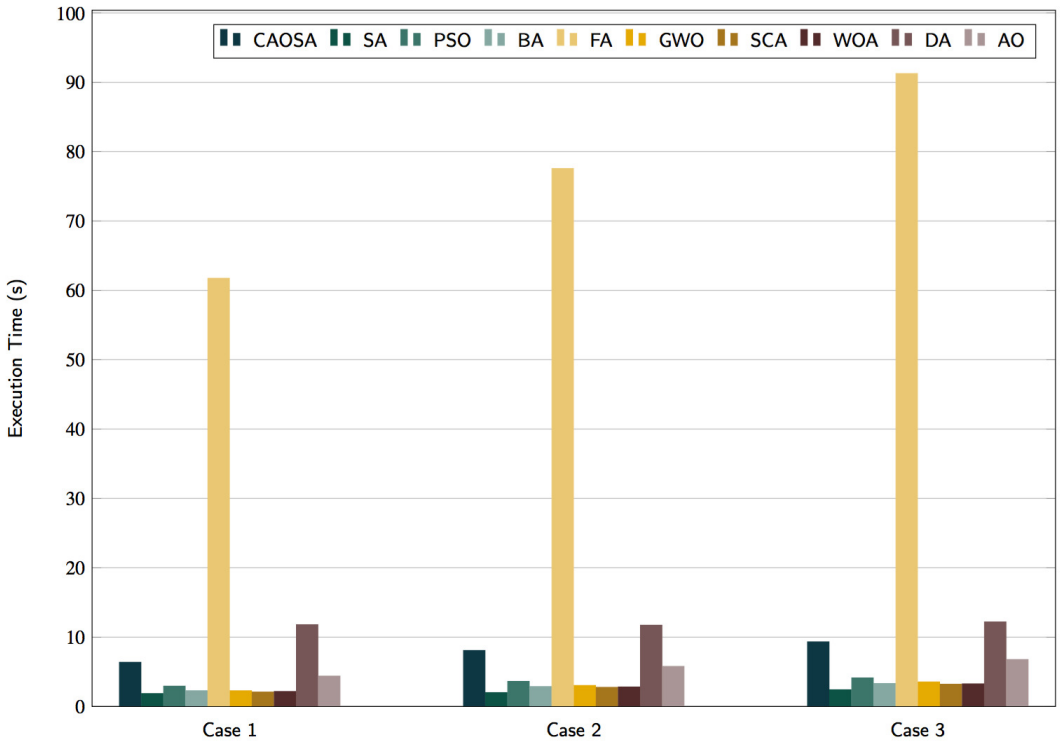


Figure 5.10: The average execution time produced in all cases

Therefore, more the obstacles are present in the start/destination line, the greater gap. Figure 5.10 illustrates the average execution time produced by the algorithms in all cases. From the figure, we can conclude that the number of obstacles impacts the performance of the algorithms. We can notice that the processing time is increasing with the number of obstacles. In can be explained by the fact that the obstacles reflect the environment's complexity. Since the environmental complexity is part of the UAV path planning problem complexity, the number of obstacles impacts the UAV path planning problem complexity. Therefore, the problem complexity is increased with the number of obstacles, which increases the processing done by algorithms. Thus, the execution time.

5.9 Conclusion

In this chapter, a Chaotic hybrid Aquila Optimization with Simulated Annealing (CAOSA) algorithm is introduced to tackle the issue of the UAV path planning problem. The CAOSA algorithm was evaluated in different environments using various numbers of obstacles. To validate its efficiency, the CAOSA algorithm was compared to nine well-known meta-heuristics, such as SA, PSO, BA, FA, GWO, SCA, WOA, DA, and original AO. The algorithms were evaluated according to their fitness, path cost, and execution time results. Simulation results proved the superiority of the proposed CAOSA algorithm compared to the other algorithms. Moreover, results showed that our proposed approach can deal successfully with high-complexity optimization problems.
In the next chapter, we will present a novel hybrid algorithm to optimize the usage of UAV in another application, which consists of determining the optimal UAV placement for maximum users' coverage.

CHAPTER 6

A HYBRID APPROACH FOR UAV PLACEMENT

6.1 Introduction

The current trend of technology research is to enhance the quality and services in smart cities by deploying the UAVs network as an access layer. The main purpose of this deployment is to determine the optimal placement of UAVs that meets some requirements such as user coverage, UAV connectivity, energy, and load distribution.
In this chapter, we present a hybrid algorithm, called IMRFO-TS, based on the hybridization of Improved Manta Ray Foraging Optimization (IMRFO) with the Tabu Search (TS) algorithm for solving the UAV placement problem in a smart city. First, a brief survey of the existing research related to the present study is presented. Then, the system model used in this work and the mathematical model used to solve the UAV placement problem are given. Followed by a description of the Mantra ray Foraging algorithm, Tabu search algorithm, and control parameter strategy. Afterwards, we focus on the proposed Hybrid Mantra ray Foraging with Tabu search algorithms for solving the UAV placement problem. Finally, we report the simulation results and comparisons followed by conclusions and future works.

6.2 Literature review

Since the UAV placement is an NP-hard problem [201], various meta-heuristics for solving this problem have been proposed in the literature. We will focus in this section on some previous works (see Table 6.1) proposed regarding this topic based on meta-heuristics.
Wang et al. [202] proposed an improved Tabu Search algorithm based on the integration of the complementary concept into the TS algorithm for solving the UAV placement issue. The proposed algorithm was evaluated using a different number of UAVs and end nodes. Simulation results demonstrated the efficiency of the improved Tabu Search algorithm compared to Tabu Search and branch-and-cut optimizer approaches considering the solution quality

and execution time.

Strumberger et al. [203] used the Elephant Herding Optimization (EHO) Algorithm to solve the UAV Placement problem. The EHO algorithm was assessed in four scenarios using a different number of drones and users. Simulated results show that EHO offers maximum coverage for users using a limited number of drones.

Authors in [204] applied the Tabu Search algorithm for solving the UAV placement problem. The Tabu search Algorithm was evaluated using 4 UAVs and random users under different connectivity requirements. Simulation results showed that the TS algorithm gives good results regarding the connectivity metric.

In the work of Reina et al. [205], authors proposed an ameliorated version of the GA algorithm, called the Multi-Layout Multi-subpopulation Genetic Algorithm (MLMPGA), for solving the UAV placement issue. The efficiency of the MLMPGA algorithm was evaluated using different UAVs and users. The obtained results demonstrated that the MLMPGA algorithm outperforms GA, PSO, and HCA methods in terms of fitness value, coverage, fault tolerance, and redundancy.

The authors in [206] used both SA and GA for solving the UAV placement problem taking into account coverage and energy metrics. The performance of SA and GA are assessed using Java and Python, respectively, in an area of 80 kilometres squared with a different number of targeted users. Simulation results showed that the GA algorithm outperforms SA regarding fitness value and execution time.

In [207], A Social Spider Optimization (SSO) Algorithm was employed to solve the problem of UAV deployment. An evaluation of its effectiveness was conducted in three different areas serving a different number of users, and a comparison was made with the Random Search (RS) method and the Uniform Distribution application. SSO outperforms other meta-heuristics regarding fitness value, execution time, and covered users in simulations.

Yang et al. [208] applied the Differential Evolution (DE) algorithm to address the problem of UAV placement. The performance of DE algorithm was validated in one case using 5 UAVs and 100 users. DE outperforms both Genetic Algorithm (GA) and PSO algorithms in terms of fitness value.

Gupta and Varma [209] used four meta-heuristics for optimizing the UAV placement in emergency cases, named Multi-objective Particle Swarm Optimization (MOPSO), Non-dominated Sorting Genetic Algorithm II (NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA2), and Pareto Envelope-based Selection Algorithm (PESA-II). The four algorithms were validated in three scenarios (small, medium, and large-scale) with different users. Test results demonstrated that, based on fitness values, SPEA2 is suitable for a small number of users, while NSGA-II is appropriate for a medium and large number of users.

In another work, the same authors [210] suggested two hybrid algorithms for an optimal UAV placement, called HWWO-HSA and HGA-SA. The HWWO-HSA scheme is based on the hybridization of Water Wave Optimization (WWO), Harmony Search (HS), and SA al-

gorithms. The GA-SA algorithm is the result of the hybridization of GA with SA algorithms. Both proposed algorithms were tested in three scenarios using twelve different numbers of users. Results confirmed the superiority of the proposed algorithms in terms of fitness values compared to WWO, HS, SA, and GA methods.

Sawalmeh et al. [211] suggested GA-based and PSO-based clustering algorithms for solving the UAV placement issue. Both algorithms were assessed using various numbers of drones and ground users. Simulation results proved that the PSO-based clustering algorithm outperforms GA-based, ABC-based, and k-mean clustering algorithms in terms of computational complexity and cost.

Authors in [212] applied the Particle Swarm Optimization (PSO) algorithm for solving the UAVs-based Quality of Service (QoS) placement problem. The PSO algorithm was evaluated in four different areas with multiple users. Simulation results showed that the PSO algorithm provides good results in finding best positions of UAVs regarding the coverage and the requirements of QoS.

In [213], two optimization approaches in a Continuous-data range (OFSAC-PSO and OFSAC-EML) and two approaches in a Discrete-data range (OFSAD-PSO and OFSAD-EML) are developed for optimizing the UAV placement problem. Both approaches are based on Particle Swarm Optimization (PSO) and ElectroMagnetism-Like (EML) algorithms. These algorithms were applied in one area using 2601, 676, and 7734 users. Results showed that the OFSAC-PSO algorithm outperforms WOA, OFSAC-EML, OFSAD-PSO, and OFSAD-EML based on the compared metrics.

Ouamri et al. [214] applied the GWO algorithm for solving the UAV placement issue. The effectiveness of the GWO algorithm was evaluated using 10 UAVs and 200 users. Simulation results showed the performance of the GWO algorithm by covering more than 85% of users.

In [215], authors proposed a multi-objective Cuckoo Search algorithm for optimal placement of UAVs. The proposed algorithm was evaluated using a different number of drones and ground users. The proposed algorithm gives competitive results regarding the coverage, energy, and network efficiency metrics.

Wei et al. [216] applied the Deep Q-learning (DQN) approach for solving the UAV, Unmanned Surface Vehicle (USV), and Unmanned Underwater Vehicle (UUV) placement issue. The efficiency of DQN algorithm was assessed in one scenario using 1 UAV, 1 USV, and 3 UUVs. Compared to ACO, Double DQN, and Dueling DQN algorithms, the DQN approach outperforms in terms of success rate, training time, and energy optimization.

Liu et al. [217] proposed a Proximal Stochastic Gradient Descent Based Alternating approach for solving the UAV placement problem. The proposed approach was tested in an area where the users are grouped into 15 cells served by Base Stations. the proposed approach achieves good results in terms of fitness value and coverage rate.

Table 6.1: Comparative table of some existing UAV Placement works

Algorithms	Reference	Coverage	Energy	Connectivity	Load balancing
Improved TS	Wang et al. [202]	✘			
EHO	Strumberger et al. [203]	✘			
TS	Ur Rahman et al. [204]	✘	✘		
MLMPGA	Reina et al. [205]	✘			
SA and GA	Al-Turjman et al. [206]	✘	✘		
SSO	Chaalal et al. [207]	✘			
MOPSO, NSGA-II, SPEA2, and PESA-II	Gupta and Varma [209]	✘	✘		
DE	Yang et al. [208]	✘			✘
HWWO-HSA and HGA-SA	Gupta and Varma [210]	✘	✘		✘
PSO-C and GA-C	Sawalmeh et al. [211]	✘			
PSO	Mayor et al. [212]	✘			
OFSAC-PSO, OFSAC-EML, OFSAD-PSO, and OFSAD-EML	Recep Ozdag [213]	✘			
GWO	Ouamri et al. [214]	✘			
MOCS	Mahajan et al. [215]	✘	✘	✘	
DQN	Wei et al. [216]	✘	✘	✘	
ProxSGD-based alternating	Liu et al. [217]	✘	✘		
IMRFO-TS	Proposed method	✘	✘	✘	✘

6.3 UAV Placement Problem formulation

The purpose of this section is to describe the UAV placement problem in more detail. To begin with, we will present the system model we used in this study. Afterwards, we will

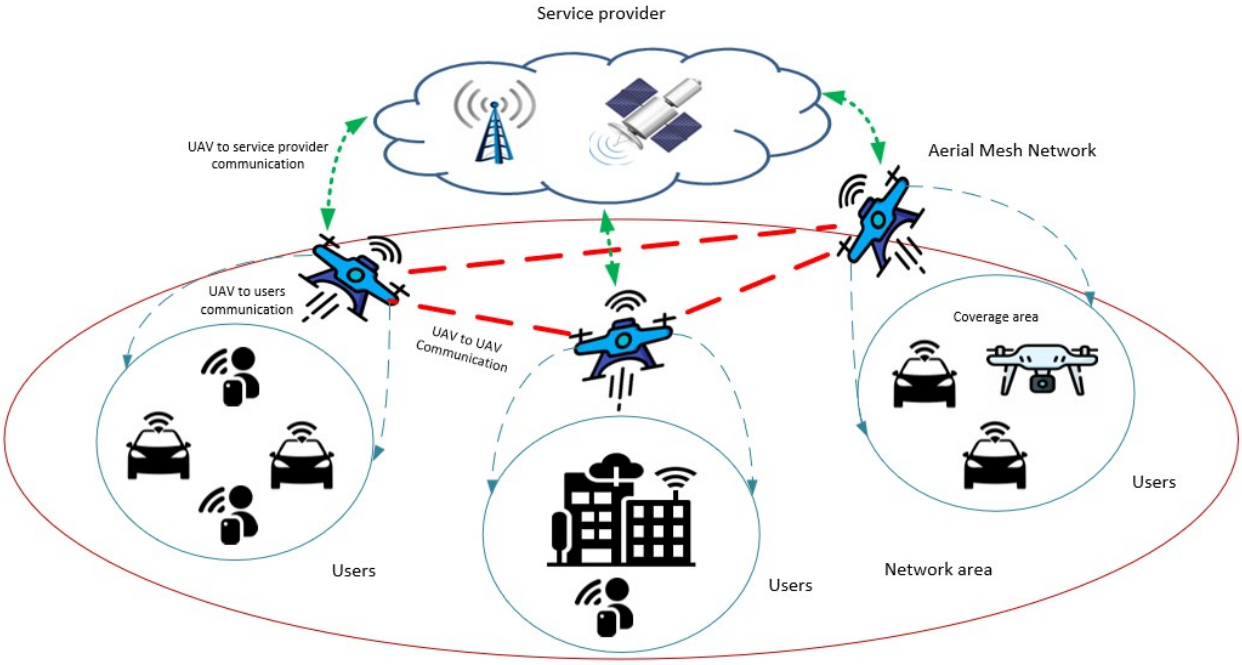


Figure 6.1: UAV-based wireless network architecture

mathematically define the objective function that includes user coverage, UAV connectivity, energy, and load distribution requirements.

6.3.1 UAV System model

We consider in an urban area a set of users $G = \{g_1, g_2, \ldots, g_k\}$, where k is the total number of users and a set of UAVs $U = \{u_1, u_2, \ldots, u_n\}$, where n is the total number of UAVs. Each UAV $u_j, j \in \{1, 2, \ldots, n\}$ is located at the coordinate (x_j, y_j, z_j) within the boundaries of the served area $[L, W, H]$, where L, W and H represent the length, width and height limits, respectively. Similarly, each user $g_i, i \in \{1, 2, \ldots, k\}$ is located within the same area at the coordinate (x_i, y_i).

In this system model, we assume that each UAV $u_j, j \in \{1, 2, \ldots, n\}$, is equipped with an antenna for wireless communications characterized by a maximum transmission range R_{max}. The purpose of this communication is to form links with users to provide services and also to share information with other UAVs forming an Aerial Mesh Network (AMN) as shown in Figure 6.1.

6.3.2 Objective function

To solve the UAV placement problem, an objective function is designed to optimize simultaneously the four considered objectives which are user coverage, UAV connectivity, energy consumption, and equal load balancing. Their definitions are given below:

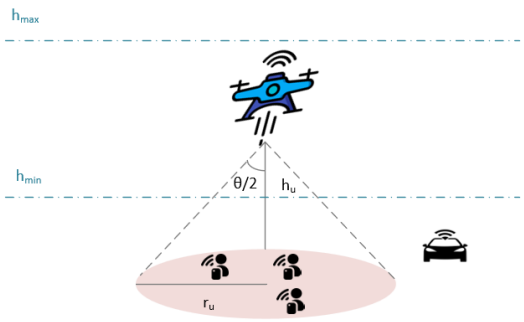


Figure 6.2: UAV Coverage

User coverage

UAVs are said to cover a user when they are located at a Euclidean distance d shorter than or equal to their coverage range in the horizontal plane (r) from it. Therefore, the coverage model considers a boolean variable for the connectivity between the UAV and the user. It is mathematically given by Equation (6.1).

$$c_{g_i,u_j} = \begin{cases} 1, & if \quad d(g_i,u_i) \leq r_j \\ 0 & otherwise. \quad i \in \{1,\ldots,k\}, \quad j \in \{1,\ldots,n\}. \end{cases} \tag{6.1}$$

Where $d(g_i,u_j)$ is the Euclidean distance between the UAV u_j and the user g_i in the horizontal plane formulated in Equation (6.2). r_j stands for the UAV coverage range related to both altitude and visibility angle as shown in Figure 6.2. It can be calculated using Equation (6.3)

$$d(g_i,u_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad i \in \{1,\ldots,k\}, \quad j \in \{1,\ldots,n\}. \tag{6.2}$$

$$r_j \leq z_j . \tan(\frac{\theta}{2}), \quad j = 1,\ldots,n. \tag{6.3}$$

Where z_j represents the altitude level of the UAV at the position (x_j, y_j). θ is the visibility angle. The total user coverage in this model Cv is defined as follows:

$$Cv = \sum_{j=1}^{n}\sum_{i=1}^{k} c_{g_i,u_j}, \quad i \in \{1,\ldots,k\}, \quad j \in \{1,\ldots,n\}. \tag{6.4}$$

Our first objective is to find the (x_j, y_j, z_j) coordinate that maximizes Cv by:

$$\max Cv = \sum_{j=1}^{n}\sum_{i=1}^{k} \max c_{g,u_j}, \quad i \in \{1,\ldots,k\}, \quad j \in \{1,\ldots,n\}. \tag{6.5}$$

UAV connectivity

The nodes in a network are said to be connected when they have a path between them. A connected network has no unreachable nodes. The connectivity between nodes or users can be done by connecting the deployed UAVs. The connectivity model considers also a Boolean decision expressed in Equation (6.6) based on the distance between UAVs.

$$ct_{u_j,u_s} = \begin{cases} 1, & if \quad d(u_j,u_s) < 2R_{max} \\ 0 & otherwise. \quad s \in \{1,\ldots,n\}, \quad j \in \{1,\ldots,n\}. \end{cases} \tag{6.6}$$

Where $d(u_j,u_s)$ is the Euclidean distance between the UAV u_j and u_s.

$$d(u_j,u_s) = \sqrt{(x_j-x_s)^2+(y_j-y_s)^2+(z_j-z_s)^2}, \quad s \in \{1,\ldots,n\}, \quad j \in \{1,\ldots,n\}. \tag{6.7}$$

The total connectivity of the UAV network Cn is given by Equation (6.8).

$$Cn = \sum_{j=1}^{n}\sum_{s=1}^{n} ct_{u_j,u_s} \tag{6.8}$$

The second objective is to find the (x_j,y_j,z_j) coordinate that maximizes Cn by:

$$\max Cn = \sum_{j=1}^{n}\sum_{s=1}^{n} \max ct_{u_j,u_s} \tag{6.9}$$

Energy consumption

Energy is an important parameter to take into consideration for the efficient deployment of UAVs network. Till today, modelling the UAV's energy consumption is a challenging issue that researchers have focused on. According to the measurements and tests done in [218], the closest model to reality of the energy consumed by the UAV u_j during the flight is formulated as follows [219]:

$$E_j = (\alpha + \beta.z_j)T + P_{Max}(\frac{z_j}{S_j}), \quad j \in \{1,\ldots,n\}. \tag{6.10}$$

Where α is the minimum power required for the UAV to fly, β is the motor speed multiplier. T represents the flight time. P_{Max} denotes the maximum motor power. z_j and S_j are the height and the speed of the UAV u_j, respectively. The total energy consumed by the UAV network is given in Equation (6.11)

$$En = \sum_{j=1}^{n} E_j, j \in \{1,\ldots,n\}. \tag{6.11}$$

The third objective is to determine the adequate UAV's height z_j to minimize the energy consumption by:

$$\min En = \sum_{j=1}^{n} \min E_j, j \in \{1, \ldots, n\}. \tag{6.12}$$

Load distribution

The physical characteristics of UAVs make them limited in terms of resources such as memory, wireless resources, etc. To ensure the optimal use of these resources and improve the homogeneity of the UAVs network, we introduced an equal load distribution to balance the exploitation of these resources. It is formulated mathematically in Equation (6.13).

$$ELD = \frac{1}{n} \sum_{j=1}^{n} (g^{u_j} - \frac{k}{n})^2, j \in \{1, \ldots, n\}. \tag{6.13}$$

Where g^{u_j} is the total number of users covered by UAV u_j. The last objective is to minimize as much as possible the equal load distribution based on the UAV location by Equation (6.14).

$$\min ELD \tag{6.14}$$

Based on the four objectives defined, we formulated the objective function as a linear weighted function expressed in the following equation:

$$f(P_i) = \omega_1.Cv + \omega_2.Cn + \omega_3.En + \omega_4.ELD, \quad \omega_1 = \omega_2 = 0.5, \quad \omega_3 = \omega_4 = -0.5. \tag{6.15}$$

Where Cv represents the user coverage cost. Cn expresses the UAV connectivity cost. E and ELD represent energy and equal load distribution costs respectively. ω is a weight coefficient.

6.4 Manta Ray Foraging Optimization Algorithm

Manta Ray Foraging Optimization Algorithm (MRFO) is a new nature-inspired meta-heuristic proposed in 2020 by Zhao et al. [220] which was applied for solving various optimization problems. The main inspiration for the MRFO algorithm comes from the behaviour of manta rays in catching their prey. MRFO employs three main foraging strategies including chain foraging, cyclone foraging, and somersault foraging [220].

6.4.1 Chain foraging

In this strategy, Manta rays form a head-to-tail chain and start foraging by moving one after another. Except for the first individual, other manta rays move toward the food and the closest manta ray for cooperation. The mathematical expression of chain foraging is

expressed as follows:

$$P_i^{t+1} = \begin{cases} P_i^t + r.(P_{Best} - P_i^t) + \alpha.(P_{Best} - P_i^t), i = 1 \\ P_i^t + r.(P_{i-1}^t - P_i^t) + \alpha.(P_{Best} - P_i^t), i = 2, \ldots, N, t \in \{1, \ldots, T\}. \end{cases} \tag{6.16}$$

$$\alpha = 2.r\sqrt{|\log(r)|} \tag{6.17}$$

Where P_i^t represents the $i-th$ individual position at the iteration t. r is a random vector in the range $[0-1]$. α is a weight coefficient. P_{Best} represents the best position found so far.

6.4.2 Cyclone foraging

This strategy is characterized by the spiral movement of Manta rays toward the food and the individual in front of it. It is mathematically represented in the following equation:

$$P_i^{t+1} = \begin{cases} P_i^t + r.(P_{Best} - P_i^t) + \beta.(P_{Best} - P_i^t), i = 1 \\ P_i^t + r.(P_{i-1}^t - P_i^t) + \beta.(P_{Best} - P_i^t), i = 2, \ldots, N, t \in \{1, \ldots, T\}. \end{cases} \tag{6.18}$$

$$P_{rand} = Lb + r.(Ub - Lb) \tag{6.19}$$

$$P_i^{t+1} = \begin{cases} P_i^t + r.(P_{rand} - P_i^t) + \beta.(P_{rand} - P_i^t), i = 1 \\ P_i^t + r.(P_{i-1}^t - P_i^t) + \beta.(P_{rand} - P_i^t), i = 2, \ldots, N, t \in \{1, \ldots, T\}. \end{cases} \tag{6.20}$$

$$\beta(t) = 2.e^{(r_1 \frac{T-t+1}{T})}.\sin(2.\pi.r_1), t \in \{1, \ldots, T\}. \tag{6.21}$$

Where p_{rand} represents a random position in the space delimited by the lower and upper bounds Lb and Ub respectively.β donate a weight coefficient. r_1 is a random value within the range $[0-1]$. T is the maximum number of iterations.

6.4.3 Somersault foraging

In this case, the food position is shown as a pivot. The manta rays swim to and fro around the food and somersault to a new position. This behaviour is expressed as follows:

$$P_i^{t+1} = P_i^t + S.(r_2.P_{Best} - r_3.P_i^t), i \in \{1, \ldots, N\}, t \in \{1, \ldots, T\}. \tag{6.22}$$

Where S represents the somersault factor which is fixed to 2. r_2 and r_3 are random numbers in the range $[0-1]$.
The pseudo-code of the Manta Ray Foraging Optimization algorithm is presented in 6

Algorithm 6 The pseudo-code of Manta Ray Foraging Optimization algorithm

Initialize MRFO parameters: Maximum number of iterations T, Population' size N, Dimension Dim, $\alpha, \delta, etc.$
Initialize the population of MRFO: $P_i(i = 1, 2, ..., N)$
Calculate the fitness value $f(P_i)$
Determine the best position P_{best}
while $(t < T)$ **do**
    for $i = 1, 2, \ldots, N$ **do**
        if $rand < 0.5$ **then**
            if $(\frac{t}{T}) < rand$ **then**
                P_{rand}=$Lb + rand.(Ub - Lb)$
                Update the position $P_i(t + 1)$ using Equation (6.20)
            else
                Update the position $P_i(t + 1)$ using Equation (6.18)
            end if
        else
            Update the position $P_i(t + 1)$ using Equation (6.16)
        end if
        Calculate the fitness value $f(P_i(t + 1))$
        Update the best position P_{best}
        Update the position $P_i(t + 1)$ using Equation (6.22)
        Calculate the fitness value $f(P_i(t + 1))$
        Update the best position P_{best}
    end for
    $t = t + 1$
end while
return The best position P_{best}

6.5 Tabu Search Algorithm

Tabu Search (TS) algorithm is a famous meta-heuristic proposed by Glover [221], widely applied for solving optimization problems [222, 223, 224, 225]. TS algorithm belongs to the single-based meta-heuristic category, which explores one candidate solution at each run [15]. Due to its characteristics, TS performs better as a local search algorithm by a neighbourhood search approach. Starting from a candidate or initial solution, TS searches for potential other solutions nearby. Then, these solutions are evaluated one by one according to their fitness value. To make sure that the TS algorithm doesn't explore solutions more than once, it creates a list to store the evaluated solutions. This list is updated if solutions don't meet evaluation criteria. In the end, the best solution which is not in the tabu list is selected. The pseudo-code of the Tabu Search algorithm is presented in 7

Algorithm 7 The pseudo-code of Tabu Search algorithm

```
Initialize TS parameters: Maximum number of iterations T, Neighbors' number ns, Tabu List TL.
Generate initial solution P
Set the current solution P as the best solution P_best
while (t < T) do
    for i = 1, 2, ... , n do
        Generate a random neighbor P' of P
        Evaluate neighbor' solution P'
        Update TL
        Update the best solution P_best
    end for
    t = t + 1
end while
return The best solution P_best
```

6.6 Non-linear control adjustment strategy

By definition, meta-heuristics are optimization algorithms that search for the optimal solution based on two strategies: exploration and exploitation search strategies. Exploration involves searching the unexplored areas of the feasible region, while exploitation involves searching the neighbourhoods of the promising region [226]. The performance of meta-heuristics depends mainly on their ability to balance both strategies. Mathematically, this balance is represented by a control parameter. In the MRFO algorithm, the term $(\frac{t}{T})$ defines the parameter that controls exploration and exploitation search. The variation of this parameter during the algorithm's execution is illustrated in Figure 6.3. To enhance the performance of the MRFO algorithm, we adopted a non-linear strategy using tangential variation as shown in Figure 6.4. As we can see from the figure, in the later stage of iterations, the control parameter $(\gamma(t))$ is not close to 1. So, its value is not greater than *rand*.

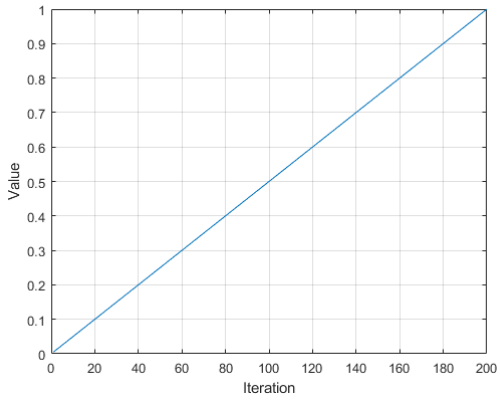


Figure 6.3: Linear control strategy

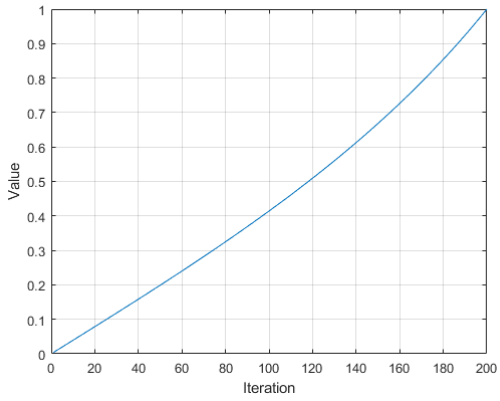


Figure 6.4: Tangential control strategy

In this case, the algorithm remains in the exploration phase, searching for new and better solutions. Therefore, This improvement can enhance the performance of the MRFO algorithm by improving its exploration ability. Mathematically, it is represented by the following equation:

$$\gamma(t) = \tan\left(\frac{\pi}{4}.\frac{t}{T}\right), \quad t \in= \{1, \ldots, T\}. \tag{6.23}$$

Where t and T are the current and the maximum number of iterations, respectively.

6.7 Hybrid Manta Ray Foraging Optimization Algorithm for UAV Placement

This section describes the implementing steps of our proposed algorithm (IMRFO-TS) for solving the UAV placement problem based on the hybridization of Improved MRFO and Tabu Search algorithms. To begin with, a non-linear control strategy is incorporated into the original MRFO for enhancing its exploration capability. Furthermore, to improve its search balance, the Tabu Search algorithm is combined as a local search algorithm with Improved MRFO for better exploitation. The pseudo-code of the proposed IMRFO-TS

algorithm is given in Algorithm 8.
The application of the IMRFO-TS algorithm for UAV placement involves 5 basic phases such as initialization, evaluation, update, local search application, termination, and solution representation.

6.7.1 Initialization

Similar to classical optimization functions, the important parameters should be given in a reasonable range for better efficiency starting with the search space. In UAV placement, the search space is set to be the 2D dimension of the area where UAVs are deployed, in addition to the height limitations of UAVs. In this space, each individual of the population represents a potential solution that is randomly initialized. The total population can be represented as follows:

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,j} \\ P_{2,1} & P_{2,2} & \dots & P_{2,j} \\ \vdots & \vdots & \vdots & \vdots \\ P_{N,1} & P_{N,2} & \dots & P_{N,j} \end{bmatrix} \tag{6.24}$$

Where N is the total population size. $P_{i,j}$ denotes the *i-th* individual position that is represented in Equation (6.25)

$$P_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j}), \quad i \in= \{1, \dots, N\}, \quad j \in= \{1, \dots, n\}.. \tag{6.25}$$

H_{max} represents the maximum flight altitude that is related directly to both visibility angle (θ) and maximum transmission range (R_{max}). It is mathematically expressed in Equation (6.26).

$$H_{max} = \frac{R_{max}}{\tan(\theta/2)} \tag{6.26}$$

6.7.2 Evaluation

At this level, the population is initialized and the IMRFO-TS starts evaluating the individuals. For each individual, the fitness value is calculated using Equation (6.27).

$$Cost_i = f(P_i) \tag{6.27}$$

IMRFO-TS saves at this point the best individual found so far that corresponds to the best cost. The problem of UAV placement belongs to the category of maximization problems. Therefore, the best individual is defined as the individual where the maximum cost is reached. It is formulated in Equation (6.28).

$$P_{Best} = \arg\max f(P_i) \tag{6.28}$$

6.7.3 Update

At the beginning of each iteration, the predefined control parameter ($\frac{t}{T}$) is replaced and updated using the proposed control strategy defined in Equation (6.23). This new control parameter strategy constitutes the first improvement for traditional MRFO to push better the exploration search strategy. According to the parameter values, individuals of the improved MRFO algorithm are updated in the same way as the original MRFO algorithm. The cost of each updated individual is calculated, and the best individual found is stored.

6.7.4 Local search application

To bring more balance to the improved MRFO algorithm, Tabu Search algorithm is added at the end of each iteration. Based on the previous best solution, Tabu Search makes its searches for better solutions in the neighbourhood. If any better individual is found, the best individual is updated and the others are stored in its list.

6.7.5 Termination

The proposed IMRFO-TS algorithm repeats the update and local search application phases until the maximum number of iterations is reached. Once the termination criteria are satisfied. IMRFO-TS displays the best solution found.

6.7.6 Solution representation

This phase is an extra step that aims to present the solution in a format that facilitates the use of the solution for other purposes such as displaying and plotting. The best solution is given using the following representation:

$$P_{best} = \Big[(x_1, y_1, z_1) \quad (x_2, y_2, z_2) \quad \dots \quad (x_n, y_n, z_n)\Big] \tag{6.29}$$

6.7.7 Time complexity of the proposed IMRFO-TS algorithm

The time complexity of the IMRFO-TS depends on the problem dimension, the population size, and the maximum number of iterations. For every iteration, chain foraging, cyclone foraging, somersault foraging, and local search strategies are implemented. Therefore, the total time complexity of the IMRFO-TS algorithm is given as follows:
$O(IMRFO{-}TS) = O(T(O(chainforaging{+}cycloneforaging){+}O(somersaultforaging){+}O(localsearch)))$
$O(IMRFO - TS) = O(T(Nd + Nd) + T(nd))$
$O(IMRFO - TS) = O(Td(N + n))$
Where T stands for the maximum number of iterations. N and d represent the population

Algorithm 8 The pseudo-code of Hybrid Manta Ray Foraging Optimization algorithm for UAV Placement

Initialize MRFO parameters: Maximum number of iterations T, Population' size N, Dimension Dim, $\alpha, \delta, etc.$
Initialize the population of MRFO: $P_i(i = 1, 2, ..., N)$
Calculate the fitness value $f(P_i)$
Determine the best position P_{Best}
while $(t < T)$ **do**
    Update γ using Equation 6.23
    for $i = 1, 2, \ldots, N$ **do**
        if $rand < 0.5$ **then**
            if $\gamma < rand$ **then**
                X_{rand}=$Lb + rand.(Ub - Lb)$
                Update the position $P_i(t + 1)$ using Equation (6.20)
            else
                Update the position $P_i(t + 1)$ using Equation (6.18)
            end if
        else
            Update the position $P_i(t + 1)$ using Equation (6.16)
        end if
        Calculate the fitness value $f(P_i(t + 1))$
        Update the best position P_{Best}
        Update the position $P_i(t + 1)$ using Equation (6.22)
        Calculate the fitness value $f(P_i(t + 1))$
        Update the best position P_{Best}
    end for
    Call TS on P_{Best} as initial solution
    Update the best solution P_{Best}
    $t = t + 1$
end while
return The best position P_{Best}

Table 6.2: Description of simulation parameter

Simulation Parameter	**Value**
Area size	$[1000, 1000]$
Total number of users	$[20, 60, 100, 140]$
Total number of UAVs	$[4 - 16]$
UAVs transmission range	$250m$
Visibility angle	$120°$
E_0	$100Kj$
α	30
β	1
P_{Max}	85 $Watt$
s	2 m/s

size and the problem dimension, respectively. n donates the number of neighbourhoods defined in TS algorithm.

6.8 Numerical experiments and results

In this section, we introduce the simulation experiments, performance evaluation, and results of the proposed IMRFO-TS algorithm and other well-known meta-heuristic techniques, namely: TS [221], BA [195], FA, GWO [197], SCA [198], WOA [199], MRFO [220], and RSA [227]. These experiences aim to demonstrate the efficiency of the proposed IMRFO-TS algorithm by investigating how the IMRFO-TS adapts to the problem of UAV placement under various requirements. For this purpose, several scenarios have been considered using a different number of UAVs and users distributed randomly and non-uniformly in a square area of 1000×1000 m^2. The details of the simulation parameters are summarized in Table 6.2. Before starting scenarios implementation, all optimization algorithms parameters are adapted and fixed according to Table 6.3. Afterwards, simulations are run on MATLAB R2021b software with the computer processor Intel Core i7 2.90GHz, RAM 32 GB, and 64-bit Windows 11 operating system. The population size and the maximum number of iterations are set to 50 and 200, respectively.

We set different complexity ranges for the UAV placement problem. Starting with the simplest case using 4 UAVs and 20 users and ending with 140 users served by 16 UAVs. For each case, algorithms are evaluated according to their fitness value, coverage, connectivity, energy, and load distribution metrics. The evaluation was running 50 times and the results are presented in this section.

Table 6.3: Algorithms parameters

Algorithm	**Parameter**	**Value**
IMRFO-TS	α	rand
	β	rand
	Number of neighborhood n	50
	Tabu length TL	25
TS	Number of neighborhood n	50
	Tabu length TL	25
BA	Minimum frequency f_{Min}	0
	Maximum frequency f_{Min}	2
	Initial loudness A_0	1
	Initial pulse emission rate r_0	1
	Loudness constant α	0.5
	Emission rate constant γ	0.5
FA	Light absorption coefficient γ	1
	Initial light intensity coefficient I_0	2
	Initial attraction coefficient β_0	2
	Mutation coefficient	0.2
	Mutation coefficient damping ratio	0.98
GWO	Control parameter a_{min}	0
	Control parameter a_{max}	2
SCA	Control parameter r_1	[2,0]
	Control parameter r_2	$rand$
	Control parameter r_3	$rand$
	Control parameter $r4$	$rand$
WOA	Control parameter a_{min}	0
	Control parameter a_{max}	2
MRFO	α	rand
	β	rand
RSA	Exploitation adjustment α	0.1
	Exploration adjustment β	0.1

6.8.1 Case of 20 users

Table 6.4 reports the obtained results of 20 users covered by different numbers of UAVs. It can be observed that the proposed IMRFO-TS algorithm provides the best fitness value in most cases. The IMRFO-TS algorithm reaches its maximum when $U = \{4, 5\}$ which is considered as the appropriate use case of the IMRFO-TS algorithm. MRFO and GWO algorithms present approximately the same best fitness value when U is set at 4. TS, BA, SCA, WOA, RSA algorithms didn't reach their optimal value. Therefore, more UAVs are required for the global solution. FA algorithm, in this scenario, achieves its best fitness value of 0.06 when $U = 10$. However, this value is small compared to the value of 0.28 provided by the IMRFO-TS algorithm. For all algorithms, we can notice that the fitness value is decreasing while the number of UAVs U increases. It can be explained by the fact that the total energy consumed by UAVs is also increasing. Figure 6.5 displays the convergence curve given by the algorithms in the case of using 4 UAVs for serving 20 users. In this case, we can notice that TS, BA, FA, and WOA algorithms converge quickly in the earlier iterations. This phenomenon can be explained by the fact that these algorithms cannot explore new solutions and only consider their sub-optimal solutions as the best. As for SCA and RSA algorithms, we can notice that their convergence behaviour, in this case, is similar and too slow. MRFO algorithms converge slowly and did not reach their best fitness value yet. The convergence of the GWO algorithm is approximately steady during the iterations and it requires more iterations to reach the optimal solution. Our IMRFO-TS algorithm converges to the best fitness cost with an optimal speed. In the earlier iterations, the convergence of our proposed approach is enough quick due to its good exploration capabilities and improved non-linear control strategy. In the last stage of iterations, we can see that the convergence becomes slow and almost stable. At this stage, the IMRFO-TS searches for better solutions in the area by performing a local search. To validate the efficiency of the proposed approach, we performed several statistics tests, including the Friedman test as presented in Table 6.5. From the table, we can see that the proposed IMRFOTS provides optimal descriptive results, where it gives the highest minimum, maximum, and mean fitness values in the first scenario. Regarding the standard deviation, our method gives good results and is closer to the one given by both MRFO and GWO. In this case, we can say that our approach is stable along the test. By analysing the mean rank, we can see that the IMRFOTS gives the best rank compared to the other algorithms. The asymptomatic signification is less than 0.001 which represents 0.1%. In this case, the signification level indicates that the median of all the algorithms is not equal and the null hypothesis is rejected. From these results, we can conclude that our approach clearly outperforms the state-of-the-art algorithms.

Figure 6.6 illustrates the whisker plots of the fitness provided by the algorithms in the first case of users. From the figure, we can notice that both the median fitness and distribution of the algorithms are not all identical. For the IMRFO-TS algorithm, the statistical data is

Table 6.4: Results obtained from different algorithms in the first case

Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA	
Fitness	**0.28**	-0.64	-0.68	-0.34	0.25	-0.26	-0.26	0.26	-0.42	
Coverage	**95.4**	67.9	46.8	69.7	95.2	81.4	83.4	**95.4**	66.1	
Connectivity	**100**	81	80	98.5	99.5	87.5	74	89.5	90	$U = 4$
Energy	**35.78**	59.64	41.55	43.42	36.63	54.6	52.98	38.38	45.42	
Load	0.49	3.44	3.57	2.62	0.56	2.18	2.07	**0.43**	2.77	
Fitness	**0.28**	-0.46	-0.68	-0.18	0.18	-0.38	-0.21	0.25	-0.26	
Coverage	**93.6**	69.1	55.3	82.6	89.8	65.2	76.7	91.1	82.9	
Connectivity	94	86.8	64.4	95.2	**98.8**	76	95.2	97.6	80	$U = 5$
Energy	**27.49**	53.28	40.39	61.37	32.88	39.08	54.21	28.26	59.03	
Load	**0.5**	2.88	2.65	1.9	0.83	2.53	2.02	0.6	2.07	
Fitness	**0.15**	-0.3	-0.41	-0.25	0.02	-0.27	-0.19	**0.15**	-0.4	
Coverage	87.3	60.8	54	68.7	86.9	55.5	68.5	**89.7**	34	
Connectivity	**100**	98	98.67	85	95.67	92	98.67	96	**100**	$U = 6$
Energy	**18.63**	38.85	33.07	53.3	29.78	33.1	42.75	21.09	17.72	
Load	1.12	2.41	2.85	1.99	1.44	2.24	2	**0.98**	2.75	
Fitness	**0.23**	-0.34	-0.68	-0.1	0.09	-0.14	-0.16	0.13	-0.34	
Coverage	**91.7**	49.1	55.5	68.5	83	80.5	60.2	83.8	45.8	
Connectivity	99.71	90	**100**	98.86	**100**	96.57	99.14	99.71	94.57	$U = 7$
Energy	22.75	41.77	33.21	47.51	23	54.02	34.96	**21.80**	35.09	
Load	**0.76**	2.33	2.24	1.6	1.22	1.79	1.87	1.11	2.41	
Fitness	**0.18**	-0.3	-0.27	-0.09	0.08	-0.14	-0.13	0.16	-0.23	
Coverage	**91.3**	52.7	64	68.7	84.6	63.9	67.5	91.2	67	
Connectivity	**100**	85.25	99.75	**100**	99.75	96.25	99.5	**100**	73.75	$U = 8$
Energy	**23.61**	41.05	41.86	50.55	24.48	33.69	49.39	23.87	30.09	
Load	**0.95**	2.16	2.31	1.54	1.29	1.82	1.7	1.03	2.03	
Fitness	0.12	-0.21	-0.16	-0.05	0.08	-0.12	-0.12	**0.15**	-0.17	
Coverage	88.3	79.3	89.5	71.9	85.9	64	55.8	**92.9**	43.8	
Connectivity	**100**	97.56	**100**	**100**	99.78	96.89	99.78	**100**	99.56	$U = 9$
Energy	**20.75**	51.32	51.04	56.58	24.04	35.68	28.80	24.39	30.96	
Load	1.2	2.1	2.06	1.34	1.31	1.73	1.75	**1.07**	1.82	
Fitness	**0.08**	-0.13	-0.11	0.06	0.13	-0.08	-0.02	**0.08**	-0.11	
Coverage	79.8	74.8	81.7	85.1	**86.1**	63.4	73.2	74.5	54.10	
Connectivity	**100**	**100**	**100**	**100**	**100**	97.8	**100**	**100**	**100**	$U = 10$
Energy	14.39	40.78	40.57	50.22	23.12	29.03	37.89	**12.94**	27.35	
Load	1.34	1.86	**1.11**	1.12	1.87	1.66	1.42	1.31	1.69	
Fitness	0.11	-0.11	-0.14	0.05	0.12	-0.07	-0.06	**0.13**	-0.14	
Coverage	**79.8**	66.5	69.3	76.1	79.4	60.7	49.7	74.8	46.6	
Connectivity	**100**	92.18	**100**	**100**	**100**	98.18	**100**	**100**	**100**	$U = 11$
Energy	17.34	42.74	48.81	53.23	21.18	29.92	30.14	**14.41**	35.25	
Load	1.15	1.6	1.78	**1.04**	1.1	1.59	1.43	1.09	1.7	
Fitness	0.1	-0.08	-0.11	-0.04	0.11	-0.07	-0.03	**0.16**	-0.13	
Coverage	65.4	46.9	70.3	64.2	73	44.1	67.4	**81.5**	14.5	
Connectivity	**100**	99.67	**100**	**100**	**100**	99.67	98.83	**100**	**100**	$U = 12$
Energy	**14.28**	37.88	50.12	30.3	21.42	21.91	43.39	23.07	5.48	
Load	1.13	1.42	1.65	1.49	1.05	1.49	1.35	**0.95**	1.61	
Fitness	**0.15**	-0.06	-0.08	-0.01	0.12	-0.05	0.01	0.09	-0.06	
Coverage	**79.2**	38.4	32.7	71.4	74.8	38.5	63.7	69.5	53.1	
Connectivity	**100**	99.69	100	99.54	99.85	99.07	99.85	100	99.54	$U = 13$
Energy	25.94	29.61	34.91	43.18	23.1	**17.98**	34.4	22.09	27.99	
Load	**0.94**	1.32	1.32	1.3	1.03	1.41	1.26	1.1	1.48	
Fitness	0.11	-0.08	-0.09	0.03	0.09	-0.04	0	**0.13**	-0.03	
Coverage	63.9	66.5	35.4	72	61.2	29.2	48.8	**76.4**	52	
Connectivity	**100**	99.43	93.14	99.57	93.14	99.43	98.86	**100**	**100**	$U = 14$
Energy	13.83	40.77	35.77	53.4	20.61	**10.30**	27.55	21.34	25.11	
Load	1.09	1.17	1.3	1.06	1.05	1.34	1.22	**1.03**	1.38	
Fitness	**0.17**	-0.07	-0.08	0.01	0.13	0	0.01	0.12	0.01	
Coverage	**89**	62.8	51.1	62.5	73.9	41.3	65.7	72.3	53.9	
Connectivity	**100**	99.47	99.87	**100**	99.87	99.2	99.87	**100**	**100**	$U = 15$
Energy	33.83	44.19	42.25	48.93	20.31	**16**	44.91	20.19	30.17	
Load	**0.87**	1.47	1.41	1.1	1.02	1.24	1.15	1.05	1.21	
Fitness	**0.2**	-0.03	-0.03	0.03	0.17	0.01	0.04	0.12	0.01	
Coverage	**89.2**	72.1	75	55.3	83.4	43.4	63.7	61.3	78.6	
Connectivity	**100**	99.62	**100**	**100**	**100**	99.87	99.35	99.87	**100**	$U = 16$
Energy	20.73	40.18	37.43	36.46	20.21	**19.64**	29.9	14.74	38.03	
Load	**0.9**	1.44	1.5	1.1	0.96	1.19	1.15	1	1.35	

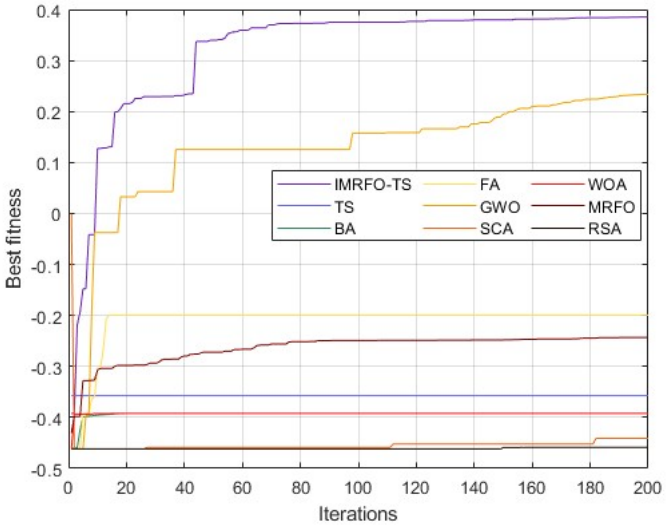


Figure 6.5: Convergence curve in the case of 20 users and 4 UAVs

Table 6.5: The Friedman tests of the fitness cost in the first case

	Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA
Descriptive statistics	N	13	13	13	13	13	13	13	13	13
	Minimum	0.08	-0.6	-0.6	-0.3	0.2	-0.3	-0.2	0.08	-0.4
	Maximum	0.28	0	0	0.06	0.25	0	0.01	0.26	0.01
	Mean	0.17	-0.18	-0.21	-0.04	0.11	-0.08	-0.06	0.14	-0.14
	STD	0.07	0.19	0.25	0.11	0.05	0.10	0.08	0.06	0.15
Rank	Mean rank	8.46	2.31	2.31	4.96	7.5	3.85	4.54	8.04	3.04
Statistic tests	N	13								
	Chi-square	91.27								
	Df	8								
	Asymp. sig	< 0.001								

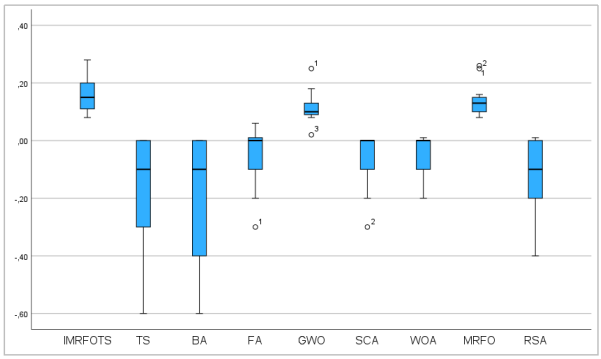


Figure 6.6: The whisker plot of the fitness results obtained in the first case

Multiple Comparison						
Dependent variable: Fitness						
Algorithm (I)	Algorithm (J)	Mean difference (I-J)	SE	Sig	95 % confidence interval	
					Lower bound	Upper bound
IMRFO-TS	*TS*	0.38	0.5	< 0.001	0.28	0.49
	BA	0.44	0.53	< 0.001	0.33	0.54
	FA	0.23	0.53	< 0.001	0.13	0.34
	GWO	0.5	0.53	0.4	-0.06	0.15
	SCA	0.29	0.53	< 0.001	0.18	0.4
	WOA	0.25	0.53	< 0.001	0.15	0.36
	MRFO	0.02	0.53	0.74	-0.09	0.12
	RSA	0.34	0.53	< 0.001	0.23	0.45

Table 6.6: The post hoc test of the mean fitness of each algorithm in the firth case

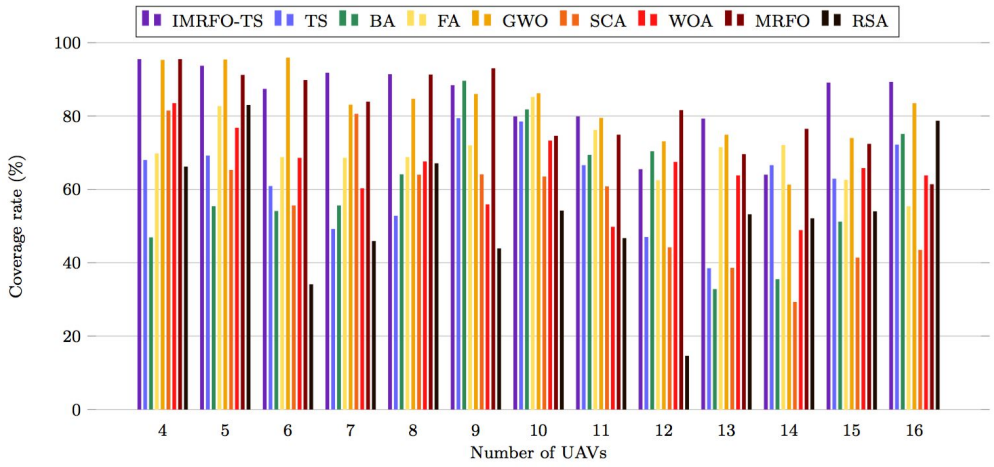


Figure 6.7: The average coverage rate in the first case

normally distributed and has no outliers. Both TS and BA algorithms share the median and their data distribution is negative skew. FA, SCA, WOA, and original MRFO data distribution is negative skew and present outliers, except the MRFO algorithm. The distribution of both GWO and RSA algorithms is positively skew. In addition, the GWO algorithm has outliers.

The post hoc tests of the fitness for the state-of-the-art algorithms in the case of using 20 users are presented in Table 6.6. From the table, we can conclude that a statistical significance between IMRFO-TS and TS, BA, FA, SCA, WOA, and RSA algorithms exists based on the significant value and the confidence interval. As for both GWO and MRFO algorithms, no statistical significance exists since their confidence interval contains zero and the significance value exceed 0.5. Therefore, the null hypothesis is accepted for both GWO and MRFO. Remaining the rest algorithms, it is rejected.

Figure 6.7 illustrates the mean coverage rate of the algorithms using various numbers of UAVs. In most cases, the IMRFO-TS algorithm provides the best coverage quality. The best coverage quality given by IMRFO-TS is 95.4% achieved when $U = 4$. In this case, both GWO and MRFO provided their best rates which are 95.4% and 95.2%, respectively. TS, SCA, WOA, and RSA algorithms provide a poor coverage quality of less than 85% which makes them inappropriate for this scenario. BA and FA achieved 89.5% and 85.1% of users

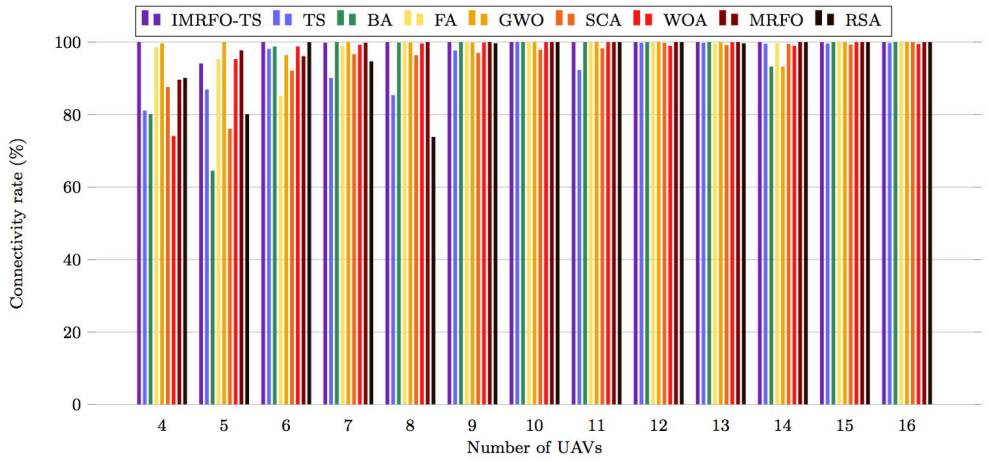


Figure 6.8: The average connectivity rate in the first case

coverage, respectively in cases $U = 9$ and $U = 10$, respectively. By varying the number of UAVs, it can be noticed that the coverage quality is decreasing due to the coverage radius minimization related to UAVs' heights.

Figure 6.8 depicts the mean connectivity rate obtained by the algorithms under different numbers of UAVs. It can be seen that the proposed IMRFO-TS algorithm improves significantly the connectivity rate compared to the other algorithms. Starting from $U = 8$, the proposed approach achieves full connectivity between UAVs. When $U < 8$, the connectivity rate provided by the IMRFO-TS is considered high since it is varied between 94% and 100%. After IMRFO-TS, MRFO and GWO algorithms provide good connectivity quality of more than 89%. Both of them achieve full connectivity 5 and 7 times, respectively. Both TS and WOA algorithms offer more than 85% of UAV connectivity starting from $U = 5$. BA, SCA, and RSA are considered optimal in terms of UAV connectivity when $U \geq 6$. The connectivity rate provided by FA is good and more than 85% regardless of the number of UAVs.

Figure 6.9 illustrates the mean energy consumed by different UAVs in the case of 20 users. We can see that the power source of UAVs is decreasing when the number of UAVs is rising up. It can be justified by the fact that algorithms tend to minimize the cost of using UAVs and improve their efficiency by reducing energy costs. The total energy consumed by UAVs is around 20% and continuously decreases until it reached its minimum value of 13.83% when $U = 14$. The total energy consumed in the case of using IMRFO-TS algorithm solutions is reasonable and acceptable since the algorithm is improving the coverage rate. MRFO and GWO presented close results and their total energy values are good. IMRFO-TS, MRFO, and GWO make a good balance between the energy constraint and the coverage objective. The solutions found by the SCA algorithm focused on the energy aspect with less consideration of the coverage quality. The energy values provided by RSA are a little high and can be acceptable for$U \geq 9$. TS, BA, FA, and WOA showed poor performance in all cases. Their

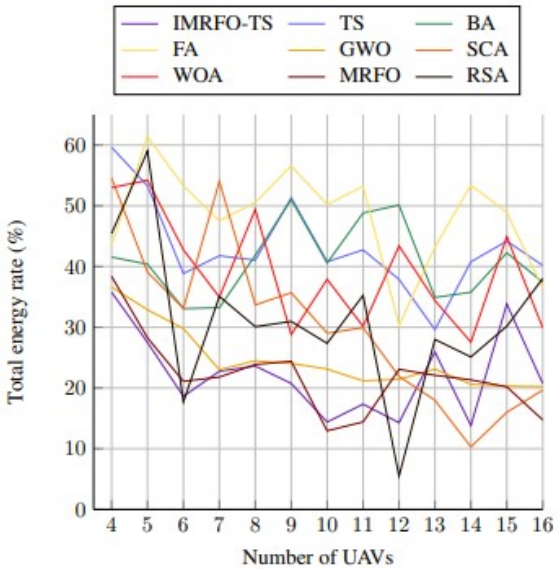


Figure 6.9: The average total energy consumption in first case

total energy consumption is considered high and not efficient which makes them inappropriate for this case of users. Figure 6.10 presents the mean load distribution. It can be seen that all algorithms provided good results regarding this metric. In most cases, the mean load distribution is around 1 which is optimal since it represents 1.67% of total users. IMRFO-TS and MRFO in this scenario provided closer results. But the proposed IMRFO-TS outperforms MRFO algorithm by one case difference. Comparing the algorithms, BA provided the highest load value. RSA and TS are ranked after BA, respectively. Fa results are less than BA, RSA, and TS results, but more than SCA and WOA results, respectively. Finally, both IMRFO-TS and MRFO algorithms are the most suitable methods for preserving UAVs resources according to the load distribution results.

6.8.2 Case of 60 users

Table 6.7 details the results obtained of 60 users under various numbers of UAVs. It can be noticed that the proposed IMRFO-TS outperforms the compared algorithm by giving the highest fitness value in most cases. The IMRFO-TS algorithm achieves its maximum value of 0.23 when the number of UAVs is set to 4. MRFO and GWO algorithms provide an equal maximum fitness value of 0.2 using 5 and 6 UAVs respectively. The fitness value of TS, BA, SCA, WOA, and RSA algorithms is continuously increasing by rising up the number of UAVs. Consequently, these algorithms didn't reach the optimal value yet and need more UAVs. Figure 6.11 shows the convergence curve of the algorithms in the case of 4 UAVs used. As expected, The convergence of TS, BA, and FA algorithms is efficient in terms of speed. However, their convergence can not be considered optimal regarding the rate since their fitness results are not optimal. The convergence of RSA, SCA, and MRFO algorithms is slow in this case due to their weakness in exploration. As for the WOA algo-

Table 6.7: Results obtained from different algorithms in the second case

Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA	
Fitness	**0.23**	-1.88	-1.88	-1.59	0.17	-1.02	-0.07	-0.38	-1.75	
Coverage	**98.8**	65.63	63.6	74.3	98.2	82.3	93.8	90.4	68.73	
Connectivity	74.5	75	77.5	**100**	85.5	69.5	73	80.5	50	$U = 4$
Energy	50.84	52.44	40.62	56.62	57.80	61.81	62.65	44.04	**38.62**	
Load	**0.3**	8.44	8.53	7.52	0.56	4.97	1.31	2.8	7.78	
Fitness	**0.06**	-1.71	-1.48	-1.05	0.2	-1.42	-1.16	-0.42	-1.99	
Coverage	95.65	58.37	61.2	71.73	97.4	68.2	72.5	88.2	56.13	
Connectivity	84.4	62	**100**	99.6	98.4	73.2	94	78.8	77.2	$U = 5$
Energy	41.89	49.36	**38.28**	62.02	51.83	44.18	56.35	38.61	43.47	
Load	1.14	7.54	7.17	5.3	**0.65**	6.67	5.73	2.95	8.85	
Fitness	0	-1.58	-1.46	-1.21	0.11	-1.11	-0.86	**0.2**	-1.39	
Coverage	91.84	59.1	57.5	69.47	96.6	72.8	69	96	62.4	
Connectivity	98	73.33	76	92.67	99.67	63	86	**99.67**	98.67	$U = 6$
Energy	**32.04**	49.33	45.58	65.74	51.73	44.96	40.71	40.84	44.34	
Load	1.94	7.16	6.73	5.82	1	5.34	4.6	**0.75**	6.71	
Fitness	-0.11	-1.37	-1.39	-1.03	-0.34	-1.21	-1.22	-0.12	-1.31	
Coverage	**91.87**	51.5	39.1	61.07	86.4	59.7	53.8	91.4	49.67	
Connectivity	97.43	95.43	**100**	99.71	98.86	82.29	89.14	98	68.86	$U = 7$
Energy	36.87	42.38	42.03	49.39	37.45	41.62	39.46	**33.42**	42.69	
Load	**1.99**	6.53	6.53	5.24	2.85	5.84	5.92	2.04	6.01	
Fitness	-0.05	-1.13	-1.33	-0.87	-0.23	-1.03	-1.05	**0**	-1.29	
Coverage	92.9	62.3	41.9	62.67	88.9	55.5	51.8	98.5	35.7	
Connectivity	**99.75**	87.75	71	100	98.5	86.5	98.75	94.07	96	$U = 8$
Energy	**33.68**	45.69	41.9	47.17	39.27	33.92	38.35	34.33	37.32	
Load	1.80	5.58	4.98	4.65	2.42	5.22	5.33	**1.57**	6.1	
Fitness	**-0.16**	-1.1	-1.20	-0.69	-0.35	-0.98	-0.68	-0.27	-1.02	
Coverage	**85.47**	62.07	48.2	71.57	80.2	46.4	66.4	83	35.87	
Connectivity	**100**	83.33	**100**	99.78	98.22	92.89	99.78	96.89	**100**	$U = 9$
Energy	31.29	40.82	52.05	52.51	34.15	**26.23**	49.08	26.33	27.15	
Load	**2.21**	5.43	5.76	3.95	2.83	5.05	3.89	2.6	5.19	
Fitness	**-0.23**	-0.97	-1.04	-0.7	-0.5	-0.85	-0.9	-0.26	-1.02	
Coverage	**83.03**	57.87	51.7	62.63	72.4	49.2	48.27	80.3	32.07	
Connectivity	**100**	90.6	99.6	98.4	98.8	93.8	99.8	**100**	99	$U = 10$
Energy	34.48	46.01	43.27	42.17	28.98	**27.21**	37.52	31.88	29.6	
Load	**2.43**	4.91	5.23	3.98	3.4	4.54	4.71	2.52	5.08	
Fitness	-0.23	-0.89	-0.91	-0.71	-0.33	-0.75	-0.72	-0.29	-0.76	
Coverage	79.87	69.53	34.8	56.47	76.9	50	50.1	78.7	49.27	
Connectivity	99.45	**100**	**100**	97.45	**100**	95.64	99.45	99.45	98.73	$U = 11$
Energy	31.11	39.46	40.46	48.23	29.05	**28.6**	34.14	28.8	33.65	
Load	**2.42**	4.85	4.59	3.92	2.79	4.17	4.05	2.93	4.19	
Fitness	**-0.27**	-0.64	-0.89	-0.62	-0.32	-0.72	-0.68	-0.32	-0.74	
Coverage	**77.8**	68.33	31.4	60.5	74.9	41.8	45.27	72.2	36.27	
Connectivity	99.83	91.67	92.33	**100**	99.33	97.17	**100**	99.67	99.5	$U = 12$
Energy	34.87	38.03	40.87	51.15	23.92	21.05	31.32	23.13	**20.85**	
Load	**2.52**	3.8	4.4	3.59	2.78	4.07	3.86	2.76	4.09	
Fitness	**-0.23**	-0.74	-0.76	-0.46	-0.36	-0.73	-0.6	-0.36	-0.76	
Coverage	75.07	35	60.6	67.77	65.4	31	50.9	64.5	20.13	
Connectivity	99.84	**100**	**100**	92.15	99.54	96.92	99.85	**100**	99.85	$U = 13$
Energy	25.81	36.19	37.74	44	25.17	22.11	39.91	25.87	**15.66**	
Load	**2.41**	3.92	4.28	3.01	2.83	4	3.49	2.81	4.1	
Fitness	**-0.29**	-0.75	-0.74	-0.51	-0.4	-0.56	-0.63	-0.3	-0.68	
Coverage	**71.36**	52.9	44.4	59.9	60.7	54.6	36.2	66.7	39.8	
Connectivity	**100**	98.71	99	**100**	99.57	97.29	99.71	99.86	99.71	$U = 14$
Energy	32.79	44.04	31.36	52.45	**23.43**	33.76	27.76	25.77	35.55	
Load	**2.55**	4.08	4.08	3.11	2.95	3.42	3.58	2.63	3.77	
Fitness	**-0.17**	-0.65	-0.68	-0.49	-0.29	-0.63	-0.58	-0.35	-0.59	
Coverage	73.67	43.43	25.6	55.87	69.2	27.1	33.3	62.9	33.47	
Connectivity	**99.87**	**99.87**	**99.87**	**100**	**99.87**	98.4	99.73	**99.87**	99.37	$U = 15$
Energy	22.53	43.76	36.44	49.13	25.18	**17.86**	25.39	28.67	33.97	
Load	**2.19**	3.6	3.62	3.04	2.59	3.6	3.41	2.74	3.33	
Fitness	**-0.25**	-0.63	-0.68	-0.36	-0.29	-0.49	-0.46	-0.35	-0.56	
Coverage	61.87	41.47	**68.7**	61.4	59.2	47	63.6	48.7	31.2	
Connectivity	**100**	99.37	99.75	**100**	99.87	97.18	99.87	99.75	99.75	$U = 16$
Energy	19.65	40.74	47.15	45.38	19.97	38.48	47.31	**14.35**	16.04	
Load	**2.46**	3.53	3.92	2.59	2.53	3.02	3	2.74	3.39	

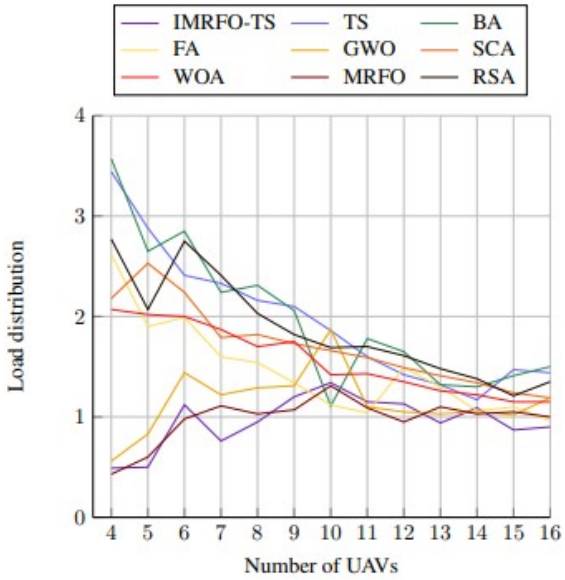


Figure 6.10: The average load distribution in the first case

Table 6.8: The Friedman tests of the fitness cost in the second case

	Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA
Descriptive statistics	N	13	13	13	13	13	13	13	13	13
	Minimum	-0.28	-1.88	-1.88	-1.59	-0.5	-1.42	-1.22	-0.42	-1.99
	Maximum	0.23	-0.63	-0.68	-0.36	0.2	-0.49	-0.07	0.2	-0.56
	Mean	-0.13	-1.08	-1.11	-0.79	-0.23	-0.88	-0.74	-0.25	-1.07
	STD	0.15	0.43	0.38	0.35	0.23	0.27	0.31	0.18	0.45
Rank	Mean rank	8.69	2.27	1.38	5.62	7.54	4.31	4.92	7.69	2.58
Statistic tests	N	13								
	Chi-square	94.86								
	Df	8								
	Asymp. sig	< 0.001								

rithm, we can see that its convergence is not steady in this case and its behaviour changes among the iterations, which demonstrates its inability to achieve the balance between the exploration/exploitation search. Regarding the GWO algorithm, we can see that it is still performing the exploration search and additional iterations are required in this case. From the figure, we can notice that the best fitness value is offered by our IMRFO-TS approach, which demonstrates its efficiency in terms of convergence rate. As for the convergence speed, we can consider it optimal due to the fact that the IMRFO-TS algorithm is enough fast to identify the area where the optimal solution is located. Moreover, our approach shows slow convergence to search for better solutions in the area.

Table 6.8 describes the statistical test results of the algorithms in the second case. It can be seen that the IMRFO-TS algorithm gives the highest results regarding the minimum, maximum, and means of fitness. Therefore, our approach is the best in terms of finding the optimized UAV positions. Moreover, the IMRFO-TS is the most stable in this case, which is proved by the standard deviation results. In this scenario, the best mean rank is given by our approach, which demonstrates its efficiency. As for the significance level, it is less than 0.1% in this case. Therefore, the null hypothesis is rejected and the difference between the

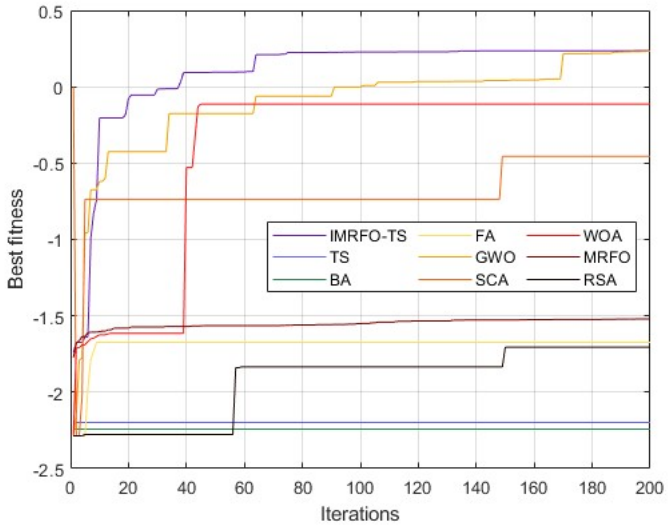


Figure 6.11: Convergence curve in the case of 60 users and 4 UAVs

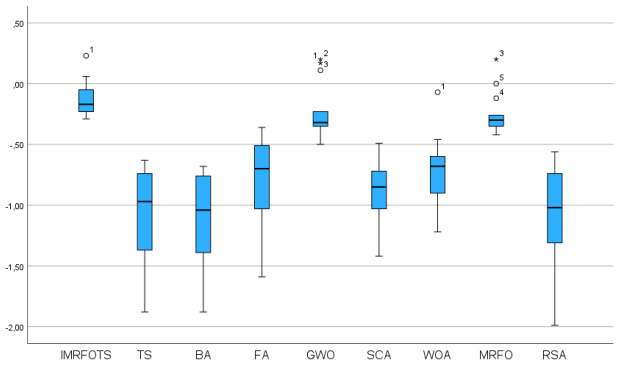


Figure 6.12: The whisker plot of the fitness results obtained in the second case

algorithms in terms of performance exists.

Based on the data presented in Table 6.8 we created a whisker plot of the fitness results for the algorithms as presented in Figure 6.12. In this case, the median fitness given by the algorithms is not identical at all. As for the data distribution, IMRFO-TS, GWO, and RSA algorithms have positive skew distribution. The other algorithms present negative skew distribution. The IMRFO-TS, GWO, WOA, and MRFO algorithms present outliers in this scenario. Table 6.9 summarized the post hoc of results of the algorithms in the case of using 60 users in the area. The table shows that statistical significance with algorithms exists. The significance value for TS, BA, FA, SCA, WOA, and RSA algorithms represent 0.1%, which is less than 5%. In addition, the confidence interval is strictly positive. Therefore, the statistical significance exists and the null hypothesis is rejected. The significance value for both GWO and MRFO algorithms exceeds 0.5. In this case, the statistical significance does not exist and the null hypothesis is accepted.

Regarding the coverage metric, IMRFO-TS gives the best coverage quality when $U = 4$ as shown in Figure 6.15. IMRFO-TS is close to covering all users with a coverage rate of

Multiple Comparison						
Dependent variable: Fitness						
Algorithm (I)	**Algorithm (J)**	**Mean difference (I-J)**	**SE**	**Sig**	**95 % confidence interval**	
					Lower bound	**Upper bound**
IMRFO-TS	*TS*	0.95	0.13	< 0.001	0.7	0.86
	BA	0.98	0.13	< 0.001	0.73	1.23
	FA	0.66	0.13	< 0.001	0.41	0.91
	GWO	0.09	0.13	0.45	-1.55	0.34
	SCA	0.75	0.13	< 0.001	0.5	1
	WOA	0.61	0.13	< 0.001	0.36	0.86
	MRFO	0.12	0.13	0.36	-0.13	0.37
	RSA	0.94	0.13	< 0.001	0.69	1.19

Table 6.9: The post hoc test of the mean fitness of each algorithm in the second case

98.8%. After IMRFO-TS, the MRFO algorithms provide a good coverage rate of 98.5% for $U = 8$. GWO and WOA provide their best coverage rate of 98.6% and 93.8%, respectively for $U = 4$. The coverage quality provided by TS, BA, FA, SCA, and RSA algorithms is poor and less than 85%. By varying the number of UAVs, the coverage quality is decreased due to the impact of energy resources. IMRFO-TS ensures a coverage quality of more than 85% when the number of UAVs is less than 9. GWO and MRFO can provide that quality until $U = 8$. WOA gives only a good quality when $U = 4$. The quality of the other algorithms is not acceptable and continues to decrease.

Figure 6.16 illustrates the mean connectivity rate. It can be noticed that the connectivity quality increases as the number of UAVs increases. In fact, adding more UAVs gives rise to the size of the network and produces more mesh links between them. For the IMRFO-TS algorithm, the use of 4 and 5 UAVs is not the optimal one. Starting from $U = 6$, the connectivity quality given by IMRFO-TS is increased from 97.43% to 100% where it achieves the full connectivity 4 times when $U = \{9, 20, 14, 16\}$ After BA algorithm in case of $U = \{5, 7, 9, 11, 13\}$. GWO and FA algorithms provide a good connectivity rate greater than 85%. WOA and MRFO algorithms improve the connectivity quality when the number of drones is greater than 6. Tabu search starts to give good connectivity results starting from $U = 7$. SCA and RSA reach more than 85% of connectivity when $U > 8$. The connectivity quality provided by BA is not stable when $U < 9$.

Figure 6.13 illustrates the mean energy provided by the algorithms. We can see that the energy is decreasing when the number of UAVs is increasing. Algorithms provide low heights UAV positions that minimize the energy. The IMRFO-TS archives the best energy value of 19.65% when $U = 16$ after MRFO and RSA with the value of 14.35% and 16.04% respectively. Except the case of $U = \{4, 5\}$, IMRFO-TS provides mean energy around 30%. In the scenario, solutions given by both SCA and RSA tend to optimize the energy metric more than the coverage. starting from $U = 9$, we can see that both of them give a good energy percentage. But the coverage quality is less than 50% in most cases. Consequently, SCA and RSA are not able to balance between different objectives. IMRFO-TS provides the best balance between coverage and energy. It works on minimizing the energy while keeping a coverage quality over 70% in most cases. MRFO and GWO algorithms also a give

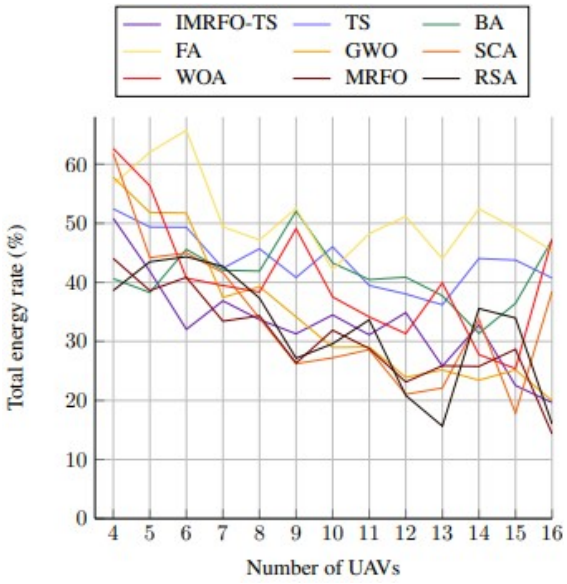


Figure 6.13: The average total energy consumption in the second case

good balance after the IMRFO-TS algorithm. The best energy results given by the WOA algorithm are achieved when $U = \{14, 15\}$, but without a good balance. The results given by TS, BA, and FA algorithms are not considered optimal. The mean energy rate is high compared to the others.

Figure 6.14 represents the mean load distribution. We can notice that the IMRFO-TS algorithm provides the smallest load distribution value in most cases. Therefore, our proposed algorithm splits approximately an equal number of users between UAVs which preserves better UAVs resources. After the IMRFO-TS algorithm, MRFO and GWO algorithms provide good results regarding the load. The load results given by TS, BA, FA, SCA, WOA, and RSA algorithms are high but acceptable since it is less than 8% of total users in most cases.

6.8.3 Case of 100 users

Table 6.10 summarized the obtained results in the case of 100 users using different numbers of UAVs. According to the table, the IMRFO-TS algorithm provided the best fitness value in most cases as expected. The proposed method achieved the optimal fitness of 0.19 which is the greatest value in the case of $U = 4$. In the same case, both GWO and MRFO algorithms provided their best fitness value of -0.08 and -0.18, respectively. The other algorithms did not converge to their optimal value and their mean fitness is improving by adding more UAVs. IMRFO-TS, GWO, and MRFO algorithms need a few numbers of UAVs to converge, unlike TS, BA, FA, SCA, WOA, and RSA algorithms. Consequently, IMROF-TS, GWO, and MRFO are the most optimal regarding the deployment cost of UAVs. Figure 6.17 illustrates the best fitness over iterations given by the algorithms in the case of 4 UAVs. As

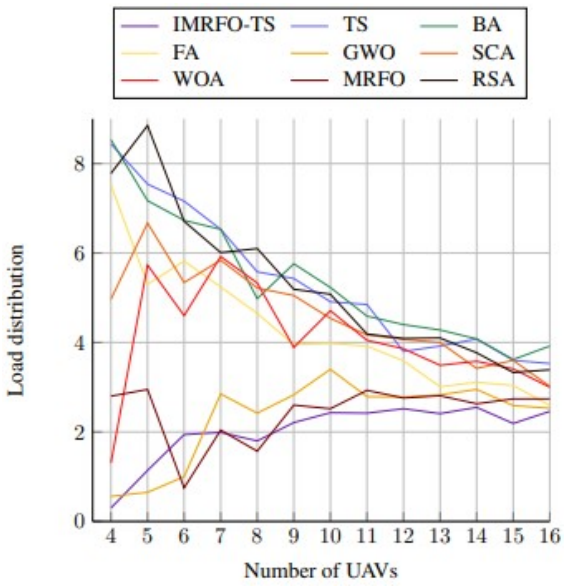


Figure 6.14: The average load distribution in the second case

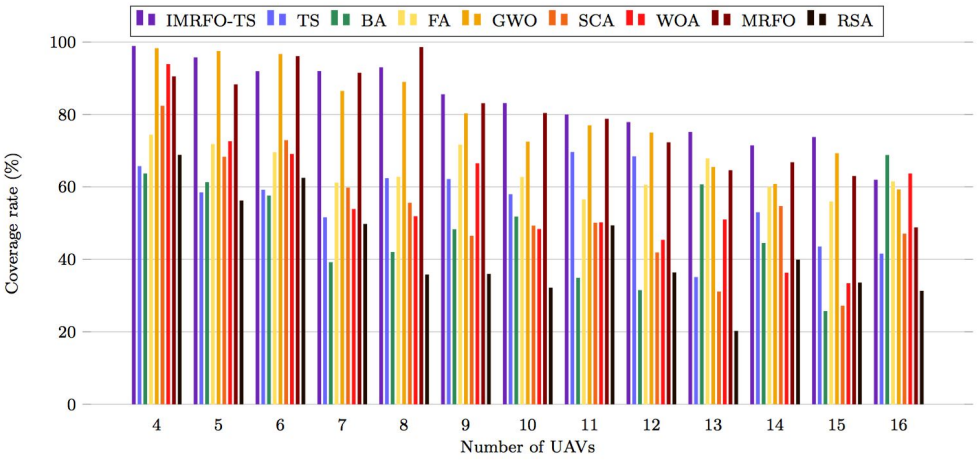


Figure 6.15: The average coverage rate in the second case

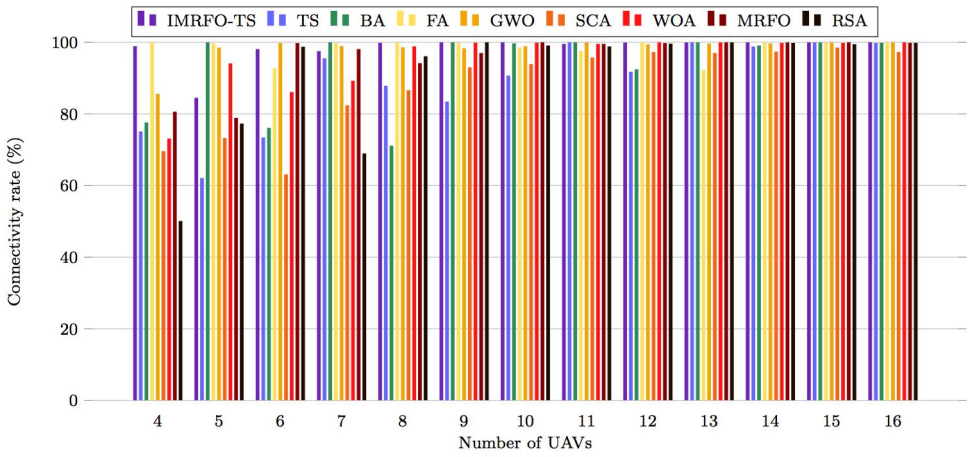


Figure 6.16: The average connectivity rate in the second case

Table 6.10: Results obtained from different algorithms in the third case

Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA	
Fitness	**0.19**	-3.04	-3.6	-2.85	-0.08	-1.57	-2.74	-0.18	-2.97	
Coverage	**99.1**	70.58	55.16	76.14	96.7	83.18	76.04	92.22	73.58	
Connectivity	69	60.5	76	**100**	94	58	78	76.5	50	$U = 4$
Energy	**44.29**	42.23	56.23	64.64	58.47	57.34	48.87	51.34	54.91	
Load	**0.45**	13.07	1.8	12.51	1.67	1.62	1.74	3.6	12.56	
Fitness	**-0.23**	-3.05	-3.58	-2	-0.71	-2.75	-1.34	-0.34	-3	
Coverage	78.4	71.9	37.66	77.52	91.12	60.84	83.78	**99.82**	59	
Connectivity	94.2	89.6	**99.6**	**99.6**	84.8	68.4	90.8	91.2	60	$U = 5$
Energy	**39.82**	54.21	42.46	57.1	48.28	44.90	67.52	47.10	43.81	
Load	2.27	13.27	1.74	9.22	1.81	**1.64**	1.79	3.61	12.74	
Fitness	**0.11**	-2.67	-2.72	-1.38	-0.07	-2.65	-1.92	-0.24	-2.55	
Coverage	**98.16**	63.64	50.58	85.86	96.94	53.28	68.1	93.54	53.54	
Connectivity	99.3	77.3	85.67	**100**	97.67	63	94.33	90.67	91	$U = 6$
Energy	37.389	50.46	**30.18**	54.07	49.52	38.97	51.08	41.55	44.18	
Load	**1.17**	11.64	1.77	6.83	1.85	1.65	1.77	3.62	11.21	
Fitness	**-0.35**	-2.29	-2.36	-1.98	-0.54	-2.04	-1.63	-0.37	-1.74	
Coverage	91.62	65.64	75.46	53.62	90.24	62.76	72.54	**96.57**	79.88	
Connectivity	98.28	88.29	62.29	84.86	95.53	62.29	**100**	90.32	87.14	$U = 7$
Energy	**35.67**	56.48	53.45	41.05	45.13	38.74	41.6	43.78	44.78	
Load	2.95	10.12	1.88	8.88	1.84	**1.72**	1.79	2.93	8.16	
Fitness	**-0.25**	-2.25	-2.31	-1.64	-0.57	-2.06	-1.96	-0.35	-2.45	
Coverage	**92.28**	48.48	50.4	64.84	88.88	49.58	53.1	88.36	27.66	
Connectivity	98.75	61.5	95.5	**100**	97	70	98.5	98.5	97.25	$U = 8$
Energy	**30.9**	40.27	31.98	52.04	40.97	32.16	47.17	36.73	36.53	
Load	2.64	9.7	1.84	7.68	1.89	**1.73**	1.79	3.66	10.68	
Fitness	**-0.36**	-2.03	-1.85	-1.66	-0.72	-1.83	-1.62	-0.4	-2.12	
Coverage	**88.4**	47.56	43.66	58.56	83.14	57.28	58.9	87.48	45.28	
Connectivity	99.33	84.89	**100**	93.11	96.89	91.11	98.11	97.78	98.22	$U = 9$
Energy	40.43	42.29	39.96	47.64	**35.36**	46.23	44.65	35.93	44.12	
Load	2.91	9.02	1.81	7.67	1.88	**1.77**	1.79	3.65	9.46	
Fitness	**-0.65**	-1.84	-1.62	-1.3	-0.66	-1.73	-1.5	-0.78	-1.69	
Coverage	79.66	46.38	63	64.66	**81.4**	40.36	53.84	75.88	41.32	
Connectivity	97.2	94.2	**100**	**100**	99.2	87.6	98.2	99.4	**100**	$U = 10$
Energy	30.45	39.94	33.88	54.47	35.06	**27.23**	50.85	28.2	35.29	
Load	4.10	8.34	1.88	6.29	1.9	**1.77**	1.83	3.67	7.81	
Fitness	**-0.43**	-1.62	-1.71	-1.2	-0.8	-1.52	-1.41	-0.54	-1.61	
Coverage	**82.76**	49.2	31.52	55.34	75.08	50.04	49.36	82.04	43.58	
Connectivity	99.45	99.82	99.45	99.45	**99.82**	86.91	99.09	99.64	94.91	$U = 11$
Energy	33.24	46.77	35.79	32.12	**28.44**	31.8	37.53	33.81	38.59	
Load	3.24	7.52	**1.84**	6.02	1.93	1.85	**1.84**	3.79	7.46	
Fitness	**-0.74**	-1.6	-1.5	-1.45	-0.84	-1.4	-1.23	-0.81	-1.62	
Coverage	**73.80**	55.9	43.18	44.48	68.56	42.56	53.74	71.96	18.98	
Connectivity	98.3	99.33	**100**	94.17	98.5	90.5	80.5	99.17	99.33	$U = 12$
Energy	35.74	42.25	38.89	49.17	28	28.59	39.77	34.02	**15.83**	
Load	4.36	7.51	**1.87**	6.7	1.94	1.88	1.89	3.84	7.49	
Fitness	-0.53	-1.41	-1.57	-1.12	-0.8	-1.41	-1.16	**-0.48**	-1.38	
Coverage	77.48	33.16	23.68	48.62	54	28.6	48.9	**80.34**	38.56	
Connectivity	99.54	93.54	92.77	99.69	**99.85**	96.77	93.54	**99.85**	**99.85**	$U = 13$
Energy	38.39	39.04	39.19	45.99	**24.86**	20.08	33.42	31.97	25.47	
Load	3.54	6.52	**1.86**	5.49	1.96	1.88	1.89	3.89	6.65	
Fitness	**-0.65**	-1.35	-1.27	-1.07	-0.78	-1.18	-1.02	-0.66	-1.27	
Coverage	69	62.6	40.6	53.2	62.74	47.38	53.7	**69.74**	35.68	
Connectivity	**100**	94.29	71.43	98.86	**100**	93.14	99.71	99.57	99.57	$U = 14$
Energy	29.82	47.77	37.31	48.52	**23.55**	32.24	31.38	25.8	27.25	
Load	3.9	6.48	**1.92**	5.3	1.98	1.95	1.93	3.93	6.14	
Fitness	**-0.62**	-1.31	-1.29	-0.96	-0.7	-1.21	-0.96	**-0.62**	-1.18	
Coverage	**68.28**	45.98	23.9	59.68	62.56	28.16	54	67.2	29.26	
Connectivity	**100**	97.87	**100**	98.67	**100**	96	**100**	99.87	**100**	$U = 15$
Energy	37.22	46.28	41.68	50.15	27.11	18.66	40.11	25.48	**23.69**	
Load	3.87	6.23	**1.91**	4.91	2.02	1.95	1.98	3.97	5.79	
Fitness	**-0.64**	-1.22	-1.14	-0.9	-0.65	-1.09	-0.91	-0.64	-1.16	
Coverage	64.54	40.46	55.58	57.08	**65.48**	28.14	59.82	61.62	19.1	
Connectivity	99.75	99.62	**100**	**100**	99.5	96.12	99.87	99.87	99.75	$U = 16$
Energy	25.86	43.28	48.51	45.51	25.13	**16.98**	33.13	24.64	17.85	
Load	3.97	5.83	2.01	4.71	2.05	**1.98**	2.09	4.02	5.66	

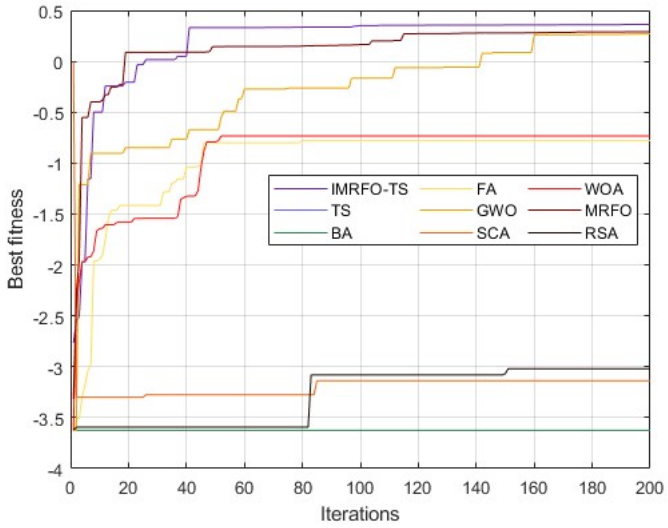


Figure 6.17: Convergence curve in the case of 100 users and 4 UAVs

Table 6.11: The Friedman tests of the fitness cost in the second case

	Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA
Descriptive statistics	**N**	13	13	13	13	13	13	13	13	13
	Minimum	-0.74	-3.05	-3.6	-2.85	-0.84	-2.75	-2.74	-0.81	-3
	Maximum	0.19	-1.22	-1.14	-0.9	-0.07	-1.09	-0.91	-0.18	-1.16
	Mean	-0.4	-1.98	-2.04	-1.5	-0.61	-1.73	-1.49	-0.49	-1.89
	STD	0.29	0.64	0.83	0.54	0.25	0.53	0.5	0.2	0.63
Rank	**Mean rank**	8.85	1.81	1.96	5.19	7.23	3.73	5.42	7.92	2.88
Statistic tests	**N**	13								
	Chi-square	93.97								
	Df	8								
	Asymp. sig	< 0.001								

Shown in the figure, the convergence of TS, BA, and SCA algorithms is stable over the iterations due to their lack of finding new solutions. RSA algorithms converge slowly and need more iterations. FA and WOA converge in this case in the same manner, where both of them reach their best in the earlier iterations. As for GWO and MRFO algorithms, we can see that their convergence is a little stable by the end of iterations, which demonstrates their limitation in exploitation. In this case, the proposed IMRFO-TS approach shows the best performance in terms of both speed and rate due to non-linear control and local search mechanisms.

Table 6.11 details the statistical results given by the algorithms in the third case. We can notice from the table that the best descriptive statistic results is given by the IMRFO-TS algorithm. Our approach gives the highest minimum, maximum, and mean values. As for the standard deviation, the original MRFO algorithm gives the smallest value. However, the STD given by our approach is close to the one given by MRFO. Therefore, the stability of our approach is optimal and proved the robustness of our approach in avoiding trapping into local optima. Regarding the mean rank, our approach is classified first and the rank gap with other algorithms is clearly noticed. By analysing the asymptomatic significance, we can see that it is small and $< 0.1\%$. Therefore, the null hypothesis is rejected also in this

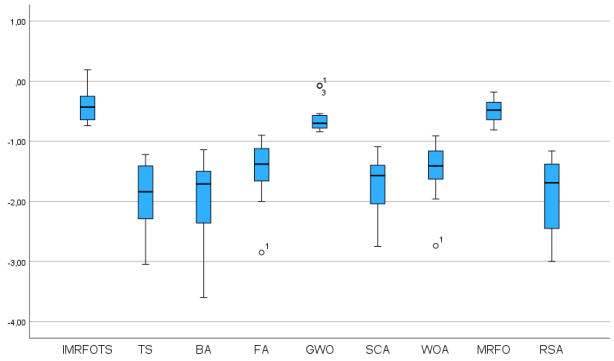


Figure 6.18: The whisker plot of the fitness results obtained in the third case

Multiple Comparison						
Dependent variable: Fitness						
Algorithm (I)	**Algorithm (J)**	**Mean difference (I-J)**	**SE**	**Sig**	**95 % confidence interval**	
					Lower bound	**Upper bound**
IMRFO-TS	*TS*	1.58	0.21	< 0.001	1.17	1.99
	BA	1.64	0.21	< 0.001	1.23	2.05
	FA	1.1	0.21	< 0.001	0.69	1.51
	GWO	0.21	0.21	0.31	-1.2	0.62
	SCA	1.33	0.21	< 0.001	0.92	1.74
	WOA	1.1	0.21	< 0.001	0.68	1.51
	MRFO	0.1	0.21	0.64	-0.31	0.51
	RSA	1.5	0.21	< 0.001	1.08	1.9

Table 6.12: The post hoc test of the mean fitness of each algorithm in the third case

case.

Figure 6.18 displays the whisker plot of the fitness results in the case of using 100 users. From the figure, the median difference between the algorithms is clearly noticed in this case. In addition, The data distribution is not the same for all the algorithms. The IMRFO-TS, BA, FA, SCA, MRFO, and RSA algorithms show a negatively skewed distribution. The distribution of the other algorithms is positively skewed. Moreover, the FA, GWO, and WOA algorithms present outliers.

Table 6.12 details the post hoc results using the least significant difference (LSD) test for the algorithms. The table reveals that there is a significant difference between the IMRFO-TS algorithm and the rest. By analysing the significant value, we can see that it is less than 0.5% for all algorithms, except both GWO and MRFO algorithms. In this case, we can say that only GWO and MRFO algorithms are not significant. In addition, the confidence interval for the mean difference between IMRFO-TS and both GWO and MRFO algorithms contains zero. Thus, the null hypothesis that there is no statistical significance between them is rejected. As for the rest, the significant value is less than 5% and the confidence interval is positive. Therefore, statistical significance exists.

Figure 6.23 illustrated the mean coverage rate in a scenario where 100 users are deployed. It can be noticed that the proposed IMRFO-TS provides the best coverage quality in most cases compared to the other algorithms. When $U < 10$, IMRFO-TS offers more than 88% of user coverage. Moreover, the coverage rate increases until 99.1% for $U = 4$. In other cases,

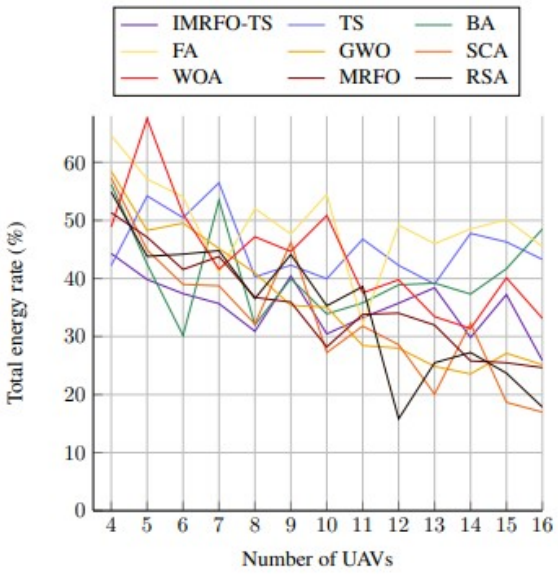


Figure 6.19: The average total energy consumption in the third case

the coverage rate given is acceptable and around 70%. Similar to the IMRFO-TS algorithm, MRFO covers more than 88% of users when $U < 10$. However, the MRFO algorithm in this case of users outperforms the GWO algorithm by giving better coverage quality. The coverage rate provided by GWO is more than 85% for $U < 9$. Apart from this number of UAVs, MRFO, and GWO algorithms have coverage rates of around 70% and 60%, respectively. The coverage results given by TS, BA, FA, SCA, WOA, and RSA algorithms are poor and around 40%.

Figure 6.24 depicts the mean UAV connectivity in the case of 100 users. It can be seen that the IMRFO-TS algorithm provides the best connectivity rate in most cases. In the case of $U \leq 6$, the IMRFO-TS method achieves more than 98% of UAVs connectivity and reaches the full connectivity for $U = \{14, 15\}$. The other algorithms also presented good connectivity results and greater than 85% in most cases. However, the IMRFO-TS algorithm outperforms the other meta-heuristics.

Regarding the energy metric, Figure 6.19 presents the mean total energy consumed by different numbers of UAVs serving 100 users. We can notice that the total energy consumed when IMRFO-TS is used is acceptable and around 35%. IMRFO-TS gives its best energy value of 25.86% for $U = 16$. From $U = 12$, SCA and RSA algorithms improve the energy consumption over other metrics where their both give their best of 16.98% and 15.83%, respectively. MRFO presents similar results to the IMRFO-TS algorithm. However, GWO results are better and around 30% of the total energy used. The results given by TS, BA, and WOA are approximately similar and around 40% which is high and inefficient. Using the FA algorithm, 50% of energy is used and unacceptable practically.

Figure 6.20 illustrates the average load distribution given by different UAVs for 100 users. It can be seen that the most optimal load is provided by the proposed approach for $U = 4$.

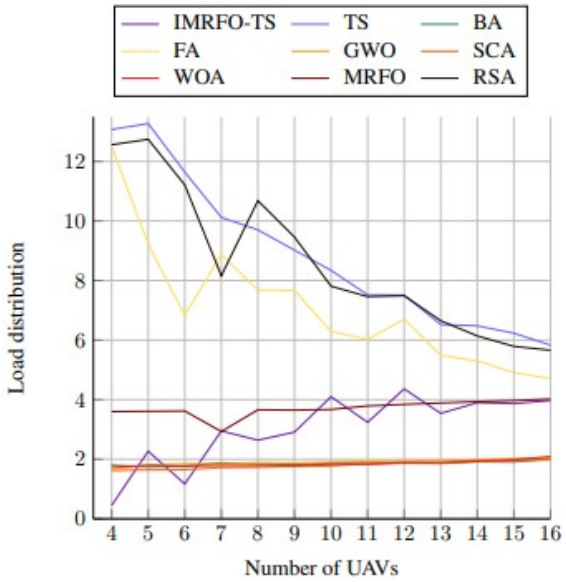


Figure 6.20: The average load distribution in the third case

For $U \neq 4$, the load increases from 1.17 to 4.36 which represents a gap of 4.5%. This gap is considered optimal and acceptable in view of the coverage quality offered. In this scenario, GWO provides good load results and MRFO offers close results to the one given by IMRFO-TS. The obtained load values by BA, SCA, and WOA algorithms are small and good. But, we cannot consider them optimal because using these algorithms most of the users and not covered which explains the small load values. As for TS, FA, and RSA algorithms, the gap in user distribution between UAVs is remarkable. But, it can be acceptable since it represents less than 10% in most cases.

6.8.4 Case of 140 users

Table 6.13 shows the obtained results by the meta-heuristics in the case of 140 users. As can be noticed, the proposed IMRFO-TS outperforms state-of-the-art meta-heuristics in most cases. IMRFO-TS gives the most fitness value of 0.3 for $U = 7$. In this case, GWO and MRFO algorithms provide the best fitness value of 0.17 and 0.16, respectively. As expected, the other algorithms did not converge yet to their optimal values and need more UAVs. Figure 6.25 illustrates the convergence curve of the algorithms in the case of using 4 UAVs. From the figure, we can notice that TS, BA, and FA algorithms converge to their best fitness in the earlier iterations.In this case, these algorithms are trapped in local optima and not able to explore new solutions. As for SCA and RSA algorithms, we can see that their convergence is slow. both of them reach their best solution in the last iterations. By analysing the convergence of the GWO algorithm, we can see that it is relatively slow and it requires more iterations in this case. As for the WOA algorithm, it converges to a good fitness value with efficient speed. The convergence behaviour of the original MRFO algorithm is similar to the

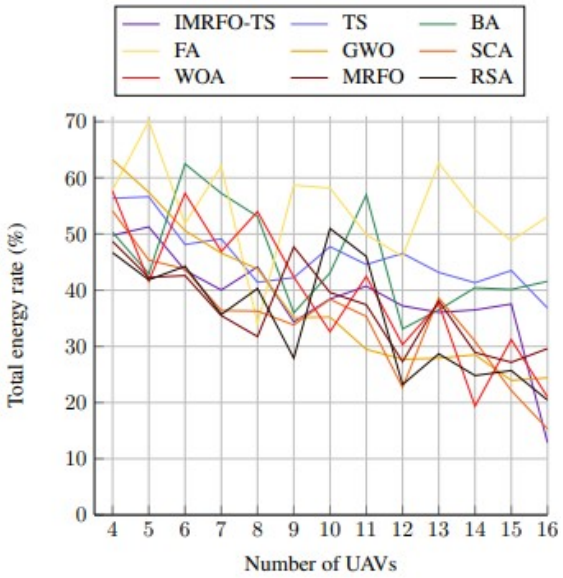


Figure 6.21: The average total energy consumption in the fourth case

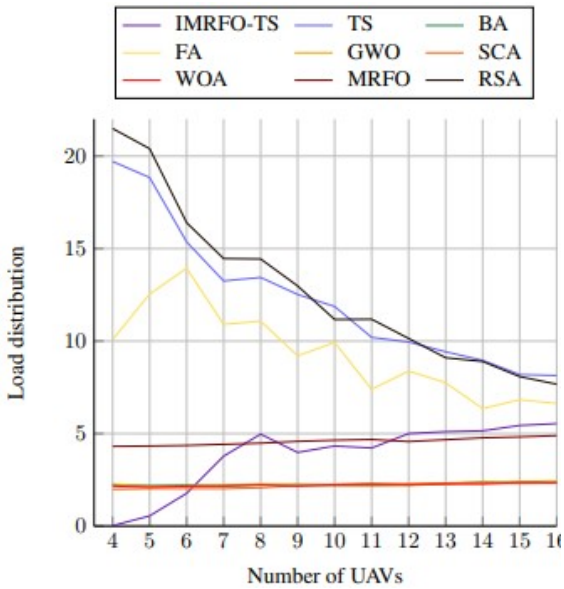


Figure 6.22: The average load distribution in the fourth case

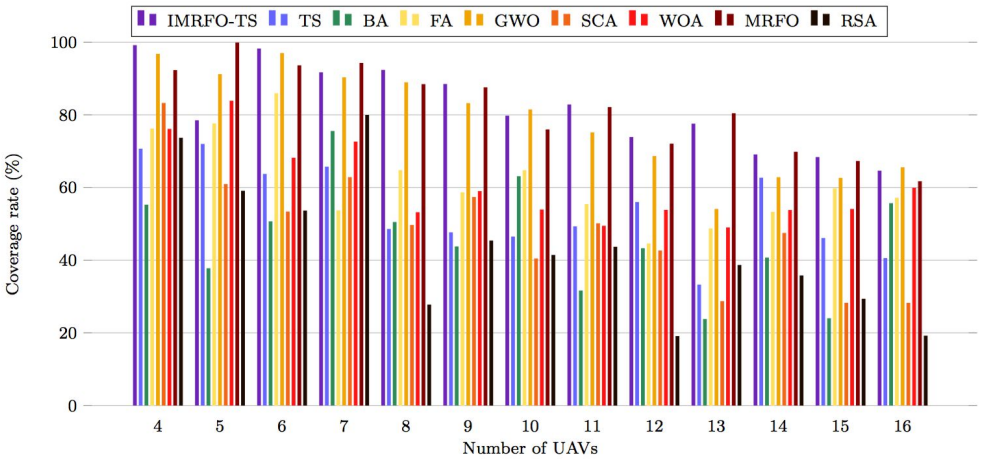


Figure 6.23: The average coverage rate in the third case

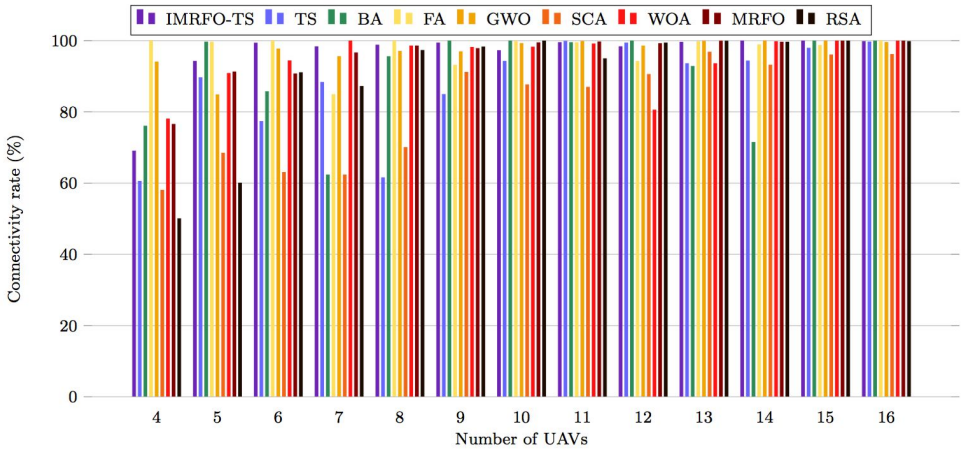


Figure 6.24: The average connectivity rate in the third case

Table 6.13: Results obtained from different algorithms in the fourth case

Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA	
Fitness	**0.30**	-4.70	-4.23	-2.33	0.17	-3.77	-1.57	0.16	-5.18	
Coverage	94.76	65.18	78.83	84.26	**99.97**	74.79	87.27	99.46	56.93	
Connectivity	**99.97**	82	75	50	88	59.5	72.5	54.5	67	$U = 4$
Energy	**49.76**	56.35	50.31	57.85	63.19	54.1	57.64	48.66	46.68	
Load	0.02	19.71	2.23	10.08	2.27	1.99	2.16	4.31	21.49	
Fitness	**0.21**	-4.52	-4.79	-3	-0.34	-3.84	-3.32	-0.8	-4.86	
Coverage	**89.2**	58.7	64.21	80.19	96.2	66.44	66.99	92.6	45.66	
Connectivity	**99.13**	73.2	77.2	42.8	92	56.8	79.6	82	91.6	$U = 5$
Energy	51.26	56.62	42.91	70.07	57.42	45.34	**41.6**	42.28	41.85	
Load	0.55	18.84	2.19	12.54	2.15	2.02	2.10	4.33	20.4	
Fitness	**-0.06**	-3.54	-3.6	-3.22	-0.49	-3.3	-2.68	-0.13	-3.88	
Coverage	**96.1**	83.87	84.13	56.69	94.04	65.86	72.29	95.99	50.67	
Connectivity	99.3	83	83.67	**100**	82.67	57.67	94.67	90	83.67	$U = 6$
Energy	43.52	48.12	62.48	51.84	50.6	43.77	57.22	**42.59**	44.25	
Load	**1.77**	15.36	2.22	13.94	2.17	2.02	2.15	4.36	16.4	
Fitness	**-0.58**	-3.08	-3.69	-2.43	-0.68	-3.24	-2.1	-1.39	-3.33	
Coverage	**91.87**	57.44	57.79	78.93	92.7	55.36	73.5	85.86	51.19	
Connectivity	91.71	85.71	94.29	**100**	85.14	70.57	76.86	90.57	98.29	$U = 7$
Energy	**40.03**	49.13	57.24	62.07	46.59	36.38	46.9	35.48	35.68	
Load	**3.78**	13.25	2.15	10.9	2.21	2.02	2.18	4.42	14.46	
Fitness	-0.88	-3.1	-3.23	-2.49	-0.71	-2.9	-2.42	-0.95	-3.4	
Coverage	87.63	51.91	61.73	56.8	90.9	53.34	67.41	86.86	40.47	
Connectivity	98.5	93.25	64.75	88.5	96.5	68	97	**98.75**	84.25	$U = 8$
Energy	44.15	41.4	53.13	34.05	43.79	36.28	53.99	31.74	40.29	
Load	4.97	13.44	2.23	11.08	2.27	2.07	2.22	4.48	14.44	
Fitness	**-0.61**	-2.89	-2.77	-2	-1.32	-2.62	-2.48	-0.8	-2.98	
Coverage	**88.79**	45.96	59.69	79.21	79.87	52.43	52.14	87.44	33.63	
Connectivity	99.78	93.56	**100**	99.11	96.89	70.67	96	90.89	99.56	$U = 9$
Energy	34.35	42.24	35.96	58.69	35.11	**33.81**	42.36	47.71	27.89	
Load	3.98	12.52	2.26	9.2	2.23	2.20	2.16	4.58	12.98	
Fitness	**-0.71**	-2.73	-2.63	-2.26	-1.28	-2.17	-2.12	-0.79	-2.57	
Coverage	**86.88**	52.21	53.13	59.11	76.74	59.6	55.47	84.61	50.73	
Connectivity	**99.4**	90.8	90.4	87.8	99.4	79.2	89.6	99.4	90.2	$U = 10$
Energy	38.44	47.74	43.08	58.18	35.21	38.38	**32.59**	39.59	50.98	
Load	4.33	11.87	2.23	9.93	2.25	2.18	2.24	4.64	11.17	
Fitness	**-0.69**	-2.33	-2.49	-1.56	-1.24	-2.18	-1.95	-0.9	-2.57	
Coverage	**85.89**	40.14	69.04	66.44	73.09	52.99	53.43	81.71	36.8	
Connectivity	98.73	91.64	**100**	97.64	98.73	83.09	86.73	99.09	98.91	$U = 11$
Energy	40.71	44.57	56.97	49.89	**29.47**	35.33	42.46	37.41	46	
Load	4.22	10.2	2.3	7.39	2.27	2.18	2.25	4.68	11.18	
Fitness	**-0.9**	-2.21	-2.3	-1.83	-1.23	-2.11	-1.81	-1.21	-2.28	
Coverage	**79.27**	65.04	31.21	53.2	69.55	37.9	51.76	71.49	26.3	
Connectivity	98.17	93.5	**100**	**100**	98.67	89.5	99.17	99.83	99.67	$U = 12$
Energy	**37.21**	46.54	33.11	46.01	27.69	22.69	30.3	27.28	23.21	
Load	5	9.95	2.22	8.38	2.28	2.20	2.26	4.57	10.14	
Fitness	**-0.93**	-2.11	-2.12	-1.69	-1.24	-1.89	-1.76	-1.1	-2.01	
Coverage	**74.44**	43.14	34.37	64.79	65.49	48.63	49.57	71.81	34.29	
Connectivity	**99.23**	93.38	92.77	86.46	99.54	93.69	99.23	99.54	100	$U = 13$
Energy	**36.05**	43.16	36.5	62.64	27.91	38.61	37.58	38.08	28.67	
Load	**5.1**	9.42	2.29	7.75	2.33	2.27	2.29	4.67	9.1	
Fitness	**-0.95**	-2.05	-1.89	-1.32	-1.19	-1.78	-1.71	-1.11	-1.99	
Coverage	**72.94**	20.66	65.44	61.47	63.41	40.67	38	69.57	23.53	
Connectivity	99	97.14	**100**	99.86	99.29	93.57	99.14	99.71	96	$U = 14$
Energy	36.48	41.32	40.43	54.34	28.49	30.9	19.36	28.91	24.8	
Load	**5.15**	8.96	2.38	6.35	2.38	2.32	2.27	4.77	8.9	
Fitness	**-1**	-1.82	-1.8	-1.45	-1.17	-1.79	-1.65	-1	-1.76	
Coverage	**68.23**	37.11	25.71	50.14	59.34	30.26	37.56	65.51	30.31	
Connectivity	99.86	99.6	**100**	99.87	99.87	92.93	98.53	100	100	$U = 15$
Energy	37.54	43.5	40.13	48.77	23.95	22.14	31.2	27.16	25.69	
Load	5.44	8.19	2.33	6.83	2.40	**2.31**	2.36	4.82	8.08	
Fitness	**-1.05**	-1.78	-1.84	-1.42	-1.12	-1.71	-1.59	-1.12	-1.65	
Coverage	**59.54**	39.74	29.81	49.31	56.79	23.14	29.4	58.61	27.1	
Connectivity	**100**	99.87	94.12	**100**	**100**	96.25	98.62	99.62	99.37	$U = 16$
Energy	**12.87**	36.81	41.57	53.09	24.4	15.31	20.98	29.6	20.44	
Load	**5.54**	8.14	2.35	6.63	2.44	2.33	2.35	4.89	7.67	

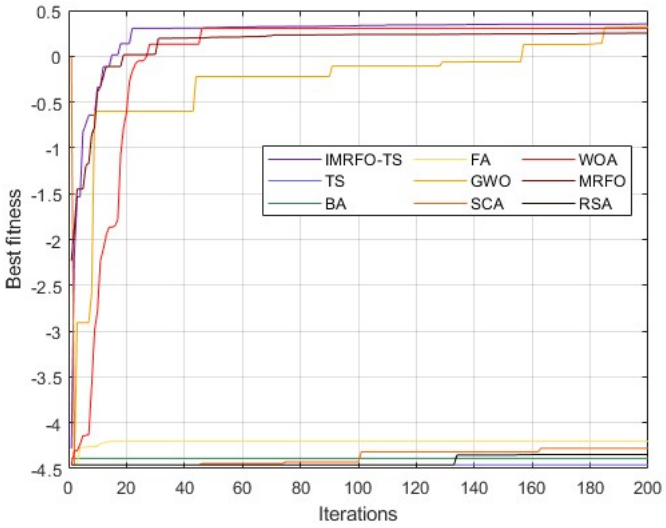


Figure 6.25: Convergence curve in the case of 140 users and 4 UAVs

Table 6.14: The Friedman tests of the fitness cost in the fourth case

	Results	IMRFO-TS	TS	BA	FA	GWO	SCA	WOA	MRFO	RSA
Descriptive statistics	N	13	13	13	13	13	13	13	13	13
	Minimum	-1.05	-4.7	-4.79	-3.22	-1.32	-3.84	-3.32	-1.39	-5.18
	Maximum	0.3	-1.78	-1.8	-1.32	0.17	-1.71	-1.57	0.16	-1.65
	Mean	-0.6	-2.84	-2.87	-2.08	-0.91	-2.56	-2.09	-0.86	-2.96
	STD	0.46	0.95	0.96	0.61	0.46	0.77	0.52	0.42	1.14
Rank	Mean rank	8.888	2.31	1.92	5.31	7.27	3.85	5.46	7.62	2
Statistic tests	N	13								
	Chi-square	96.64								
	Df	8								
	Asymp. sig	< 0.001								

WOA algorithm. However, in terms of efficiency, WOA performs better than the MRFO algorithm. In this case, the convergence speed of our proposed approach is sufficiently fast due to the good balance between the exploration and exploitation phases. Moreover, in terms of efficiency, our approach outperforms state-of-the-art meta-heuristics.

Table 6.14 displays the Friedman test results provided by the algorithms, performed in the last case of users. As expected, the optimal descriptive statistic results are provided by the IMRFO-TS algorithm. These results demonstrated the efficiency of our proposed approach in dealing with a high number of users, which reflects the high complexity problem. Moreover, with the optimal standard deviation results, we can conclude that the IMRFO-TS is stable along the scenario. This characteristic proved that our approach is likely getting closer to the global optimal solution. The highest mean rank in this case is also provided by our approach. Both MRFO and GWO offer high mean ranks. However, the mean rank gap is important and clearly noticed. Regarding the significance value, we can notice that it is less than 0.1% also in this case. Therefore, the null hypothesis is rejected.

The whisker plot of the fitness results provided by the algorithms in the last case is presented in Figure 6.26. As expected, the median fitness varies from one algorithm to another. Therefore, the algorithms' statistics are not identical, as proved by the significance value. The

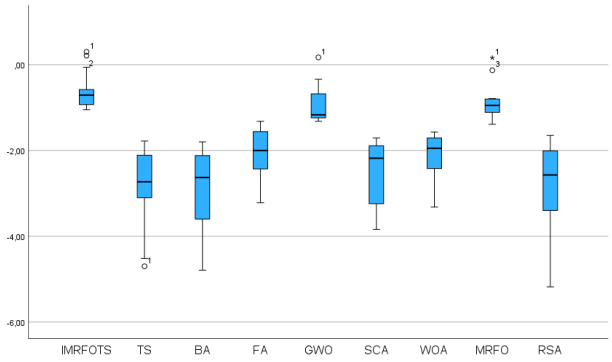


Figure 6.26: The whisker plot of the fitness results obtained in the fourth case

Multiple Comparison						
Dependent variable: Fitness						
Algorithm (I)	**Algorithm (J)**	**Mean difference (I-J)**	**SE**	**Sig**	**95 % confidence interval**	
					Lower bound	**Upper bound**
IMRFO-TS	*TS*	2.23	0.33	< 0.001	1.59	2.88
	BA	2.26	0.33	< 0.001	1.62	2.91
	FA	1.47	0.33	< 0.001	0.83	2.12
	GWO	0.31	0.33	0.35	-0.34	0.95
	SCA	1.96	0.33	< 0.001	1.31	2.6
	WOA	1.19	0.33	< 0.001	0.54	1.83
	MRFO	0.25	0.33	0.44	-0.39	0.9
	RSA	2.35	0.33	< 0.001	1.71	3

Table 6.15: The post hoc test of the mean fitness of each algorithm in the fourth case

distribution of the IMRFOTS, BA, SCA, WOA, MRFO, and RSA algorithms is negatively asymmetric. As for the rest, their distribution is positively skewed. None of the algorithms present a normal distribution in this case. Regarding the outliers, IMRFO-TS, TS, GWO, and original MRFO have. Table 6.15 represents the post hoc results of the fitness provided by the algorithms in the last case of users. According to the table, we can notice that the significance value is not the same for all the algorithms. Thus, statistical significance exists for some algorithms. Except for GWO and MRFO algorithms, the significance value is 0.1% which is less than 5%. Therefore, the hypothesis that no statistical significance exists is rejected. Figure 6.27 depicts the average coverage rate provided by different algorithms under various numbers of UAVs. As it can be seen, the use of 6 UAVs is the most optimal case of IMRFO-TS regarding the coverage metric. In this case, IMRFO-TS covers approximately 96% of users. Up to 11 UAVs, IMRFO-TS ensures coverage quality superior to 85% users. GWO and MRFO algorithms offer good coverage quality up to 99 for $U = 4$. GWO and MRFO can cover more than 85% of users for $U \leq 8$ and $U \leq 9$, respectively. WOA achieves a coverage rate of 87.27% in the case of 4 UAVs. In other cases, it provides less than 85% of user coverage. The other algorithms present a weak coverage rate that is not acceptable in real-life applications.

Figure 6.28 shows the average UAVs connectivity rate. As noticed, the proposed IMRFO-TS provides optimal UAV connectivity results. In most cases, IMRFO-TS offers more than 98% of UAV connectivity and achieves the 100% of connectivity for $U = 16$. GWO and MRFO

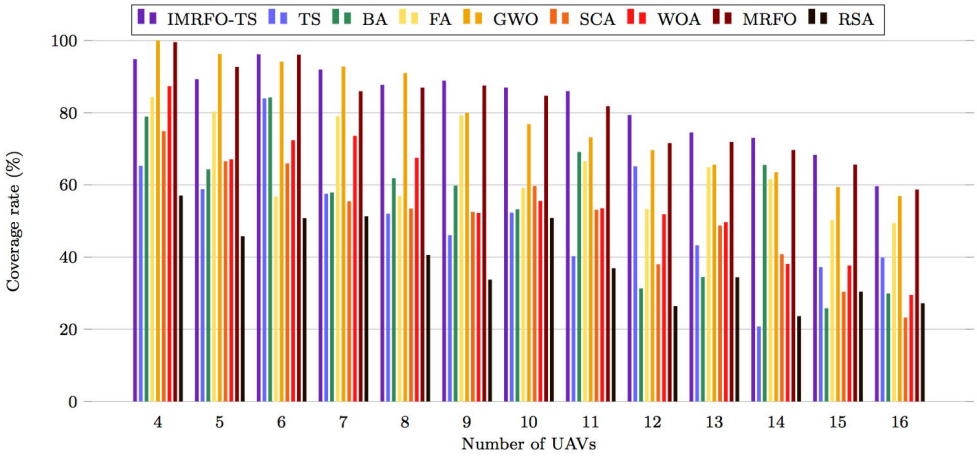


Figure 6.27: The average coverage rate in the fourth case

also give good connectivity results but not good as IMRFO-TS results. The connectivity quality given by BA, FA, and RSA algorithms is optimal starting from $U = 7$. As for WOA and SCA algorithms, they require more than 8 and 12 UAVs, respectively to offer good connectivity quality.

Figure 6.21 illustrates the average total energy consumed by different UAVs serving 140 users. As we can see, the proposed approach provides the optimal and smallest value of 12.87% in the case of $U = 16$. In other cases, the energy rate is remarkable and around 38%, as a result of the coverage quality offered. In this scenario, IMRFO-TS tend to compromise between the coverage rate and energy consumption which are both related to UAVs' heights. It results can be acceptable since IMRFO-TS is able to balance the two objectives. For $U \geq 7$, results given by GWO and MRFO are closer to IMRFO-TS results and acceptable. Results given by BA can be accepted only in the case of $U = \{12, 13\}$. SCA and WOA offer optimal results regarding the energy consumption for $U \geq 7$ and $U \geq 10$. FA algorithm presents the worst results that cannot be acceptable.

Figure 6.22 depicts the average load distribution given by the algorithms under various numbers of UAVs. From the results, it can be seen that the proposed IMRFO-TS provides the smallest load value of 0.02 for $U = 7$. The second best load (2.15) value is given by GWO in the case of $U = 7$. By varying the number of UAVs, we can notice that the load distribution obtained by the IMRFO-TS algorithm is decreasing. It can be explained by the fact that the number of covered users is decreasing due to the impact of the energy metric. IMRFO-TS results are considered optimal since the gap of load distribution between UAVs does not exceed 3%. In this scenario, the load results of BA, GWO, SCA, and WOA algorithms are optimal and less than the one given by IMRFO-TS. But, these values are small only because the coverage rate is reduced. Regarding TS, FA, and RSA results, load values are remarkable and not optimal.

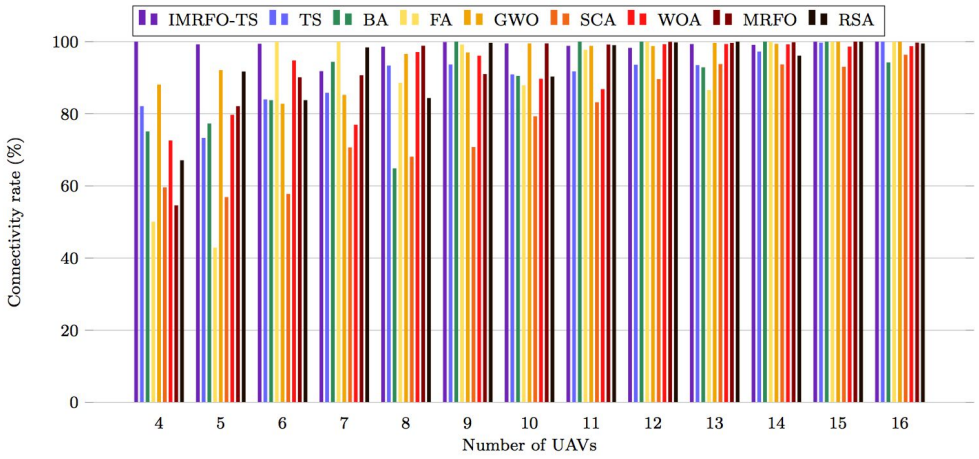


Figure 6.28: The average connectivity rate in the fourth case

6.8.5 Comparison between different cases

Comparing tables 6.4, 6.7, 6.10, and 6.13, we can notice that varying the number of users affects the optimization results provided by the algorithms. By increasing the number of users, the performance of most algorithms, except IMRFO-TS is decreasing. It can be explained by the fact that the complexity of the problem is increasing with the number of users which make it hard for them to maintain the same solution quality. In the most complex scenario ($G = 16$), IMRFO-TS provides its best fitness value compared to the other scenarios. The reason behind this is that the improvement and hybridization strategies of the IMRFO-TS algorithm are efficient and make a difference compared to other algorithms.

The coverage quality is also impacted by the number of users. Comparing the optimal coverage value given by the algorithms using different users, we can conclude that the coverage rate is increasing with the number of users. It can be justified by the fact that more users are present within the coverage area. On another side, to keep the same coverage quality, additional UAVs are required. In fact, the coverage rate is calculated based on the distance between UAVs and users. Additional users involve additional distances to take into consideration. Since the users are randomly distributed to the area and the coverage radius is limited, the probability of the calculated distances exceeding the coverage radius is increasing with the number of users. From one case of users to another, IMRFO-TS requires only 1 more UAV which is considered optimal in terms of costs.

As for the connectivity rate, the number of users is not really affecting on UAVs connectivity. Most algorithms were able to keep the same connectivity quality for different numbers of users. The connectivity metric is impacted only by the number of UAVs since the network links are established only between UAVs.

Regarding the energy metric, it can be noticed that it changes according to the number of users. The energy consumption is related to users via the coverage metric. As mentioned before, the coverage rate is increasing with the users which implies the rise of UAVs' height.

As a result, the energy efficiency is decreasing. From this fact, we can conclude that the energy and coverage metrics are trade-offs. In most user cases, the proposed IMRFO-TS approach offers a good balance between them. MRFO and GWO also present good results. SCA and RSA algorithms tend to optimize the energy parameter over the coverage. The other algorithms provide high-energy results.
The load distribution metric is impacted directly by the number of covered users. By adding more users, the coverage quality is improved. As a result, the UAVs are able to share more users which reduces the gap of the load distribution. Considering the optimal load distribution results, IMRFO-TS outperforms all meta-heuristics by giving the smallest value.

6.9 Conclusion

In this chapter, we have introduced a hybrid solution (IMRFO-TS) based on the combination of improved MRFO with the TS algorithm for solving the UAV deployment issue in a smart city. The effectiveness of the proposed IMRFO-TS algorithm is assessed using 52 benchmarks in comparison with the original MRFO and eight competitor optimization meta-heuristics such as TS, BA, FA, GWO, SCA, WOA, and RSA. Evaluation of the simulation results demonstrated the performance and efficiency of the proposed IMRFO-TS algorithm by finding optimal locations of UAVs.

GENERAL CONCLUSION

The UAV path planning represents a key factor for UAV control and navigation. It is required to ensure the UAV's safety and the efficiency of its resources to accomplish its dedicated mission. In addition, the UAV path planning reflects the path quality and the level of autonomy provided by the UAV system. The challenge in UAV path planning is to determine an efficient approach to ensure both the UAV's safety and its resource preservation. The objective of our thesis is to solve the issue of UAV path planning in different situations and constraints using optimal approaches. In view of this topic, we brought the following contributions:

The first contribution consists of proposing an improved approach to the African Vulture Optimization algorithm for solving UAV path planning. Our enhanced approach is called Chaotic Cauchy Opposite African Vulture Optimization (CCO-AVOA) algorithm, which integrates three different strategies, including logistic chaotic map, improved Elite Opposition Based Learning (EOBL), and Cauchy mutation strategies. The efficiency of our CCO-AVOA algorithm was demonstrated in a complex environment where several obstacles are presented, compared to several well-known meta-heuristics.

In the second contribution, we addressed the issue of UAV path planning in a different environment, which is built based on a real Digital Elevation Map (DEM), which is more complex. To solve this level of complexity, we suggested a hybrid approach, called Chaotic Aquila Optimization with Simulated Annealing algorithm (CAOSA) based on the hybridization of Chaotic Aquila Optimization with Simulated Annealing algorithm. For validation, our approach was evaluated in different case of obstacles and compared to various optimization algorithms.

In the last contribution, we pointed out our research on the deployment of the UAV as a wireless network to offer services to users in smart cities. Our contribution relays on proposing an optimization method, called an Improved Manta Ray Foraging Optimization with Tabu Search (IMRFO-TS) algorithm for solving the UAV placement problem. Our hybrid method includes a tangential control strategy and a local search strategy ensured by the

Tabu Search algorithm.
From our perspective, preserving UAV resources is still a challenging issue that we aim in future address it and work on it. In our future works, we will focus our research on studying the behaviour of the energy consumed by the UAV under different constraints such as winds, speed, etc. Furthermore, we will investigate on the use of machine learning approaches to optimize the use of UAVs in other applications, such as surveillance and monitoring.

BIBLIOGRAPHY

[1] Jong-Jin Shin and Hyochoong Bang. Uav path planning under dynamic threats using an improved pso algorithm. *International Journal of Aerospace Engineering*, 2020:1–17, 2020.

[2] Amylia Ait Saadi, Assia Soukane, Yassine Meraihi, Asma Benmessaoud Gabis, Seyedali Mirjalili, and Amar Ramdane-Cherif. Uav path planning using optimization approaches: A survey. *Archives of Computational Methods in Engineering*, 29(6):4233–4284, 2022.

[3] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: a review. *Applied geomatics*, 6:1–15, 2014.

[4] Nourhan Elmeseiry, Nancy Alshaer, and Tawfik Ismail. A detailed survey and future directions of unmanned aerial vehicles (uavs) with potential applications. *Aerospace*, 8(12):363, 2021.

[5] Martin E Dempsey and S Rasmussen. Eyes of the army–us army roadmap for unmanned aircraft systems 2010–2035. *US Army UAS Center of Excellence, Ft. Rucker, Alabma*, 9, 2010.

[6] Hakan Ucgun, Ugur Yuzgec, and Cuneyt Bayilmis. A review on applications of rotary-wing unmanned aerial vehicle charging stations. *International Journal of Advanced Robotic Systems*, 18(3):17298814211015863, 2021.

[7] Jack Saunders, Sajad Saeedi, and Wenbin Li. Autonomous aerial delivery vehicles, a survey of techniques on how aerial package delivery is achieved. *arXiv preprint arXiv:2110.02429*, 2021.

[8] Chunjin Yu, Daewon Kim, and Yi Zhao. Lift and thrust characteristics of flapping wing aerial vehicle with pitching and flapping motion. *Journal of Applied Mathematics and Physics*, 2(12):1031–1038, 2014.

[9] Gaurav Singhal, Babankumar Bansod, and Lini Mathew. Unmanned aerial vehicle classification, applications and challenges: A review. 2018.

[10] Leszek Cwojdziński and Mirosław Adamski. Power units and power supply systems in uav. *Aviation*, 18(1):1–8, 2014.

[11] Roger Clarke. Understanding the drone epidemic. *Computer Law Security Review*, 30(3):230–246, 2014.

[12] Fernando Peres and Mauro Castelli. Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development. *Applied Sciences*, 11(14):6449, 2021.

[13] Alexander EI Brownlee, John R Woodward, and Jerry Swan. Metaheuristic design pattern: surrogate fitness functions. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1261–1264, 2015.

[14] Dragan Savic. Single-objective vs. multiobjective optimisation for integrated decision support. 2002.

[15] Yassine Meraihi, Asma Benmessaoud Gabis, Seyedali Mirjalili, and Amar Ramdane-Cherif. Grasshopper optimization algorithm: theory, variants, and applications. *IEEE Access*, 9:50001–50024, 2021.

[16] Muhammad Toaha Raza Khan, Malik Muhammad Saad, Yang Ru, Junho Seo, and Dongkyun Kim. Aspects of unmanned aerial vehicles path planning: Overview and applications. *International Journal of Communication Systems*, 34(10):e4827, 2021.

[17] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[18] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1):153–174, 1987.

[19] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[20] Emo Welzl. Constructing the visibility graph for n-line segments in o (n2) time. *Information Processing Letters*, 20(4):167–171, 1985.

[21] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[22] Lydia Kavraki and J-C Latombe. Randomized preprocessing of configuration for fast path planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2138–2145. IEEE, 1994.

[23] Kwangjin Yang, Seng Keat Gan, and Salah Sukkarieh. A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav. *Advanced Robotics*, 27(6):431–443, 2013.

[24] Mangal Kothari and Ian Postlethwaite. A probabilistically robust path planning algorithm for uavs using rapidly-exploring random trees. *Journal of Intelligent & Robotic Systems*, 71(2):231–253, 2013.

[25] Abraham Charnes and William W Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[26] Yucong Lin and Srikanth Saripalli. Path planning using 3d dubins curve for unmanned aerial vehicles. In *2014 international conference on unmanned aircraft systems (ICUAS)*, pages 296–304. IEEE, 2014.

[27] Wu Xinggang, Guo Cong, and Li Yibo. Variable probability based bidirectional rrt algorithm for uav path planning. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 2217–2222. IEEE, 2014.

[28] Hongji Yang, Qingzhong Jia, and Weizhong Zhang. An environmental potential field based rrt algorithm for uav path planning. In *2018 37th Chinese Control Conference (CCC)*, pages 9922–9927. IEEE, 2018.

[29] Wei Zu, Guoliang Fan, Yang Gao, Yao Ma, Haiying Zhang, and Haitao Zeng. Multi-uavs cooperative path planning method based on improved rrt algorithm. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1563–1567. IEEE, 2018.

[30] Qinpeng Sun, Meng Li, Tianhe Wang, and Chenpeng Zhao. Uav path planning based on improved rapidly-exploring random tree. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 6420–6424. IEEE, 2018.

[31] LI Meng, SUN Qinpeng, and ZHU Mengmei. Uav 3-dimension flight path planning based on improved rapidly-exploring random tree. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 921–925. IEEE, 2019.

[32] Naifeng Wen, Lingling Zhao, Xiaohong Su, and Peijun Ma. Uav online path planning algorithm in a low altitude dangerous environment. *IEEE/CAA Journal of Automatica Sinica*, 2(2):173–185, 2015.

[33] Anna Yershova, Léonard Jaillet, Thierry Siméon, and Steven M LaValle. Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 3856–3861. IEEE, 2005.

[34] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[35] Dasol Lee and David Hyunchul Shim. Path planner based on bidirectional spline-rrt for fixed-wing uavs. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 77–86. IEEE, 2016.

[36] Xi-An Han, YiChen Ma, and XiLi Huang. The cubic trigonometric bézier curve with two shape parameters. *Applied Mathematics Letters*, 22(2):226–231, 2009.

[37] Wilbert G Aguilar, Stephanie Morales, Hugo Ruiz, and Vanessa Abad. Rrt* gl based optimal path planning for real-time navigation of uavs. In *International Work-Conference on Artificial Neural Networks*, pages 585–595. Springer, 2017.

[38] Jiawei Meng, Vijay M Pawar, Sebastian Kay, and Angran Li. Uav path planning system based on 3d informed rrt* for dynamic obstacle avoidance. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1653–1658. IEEE, 2018.

[39] Omar Mechali, Limei Xu, Mingzhu Wei, Ilyas Benkhaddra, Fan Guo, and Abdelkader Senouci. A rectified rrt* with efficient obstacles avoidance method for uav in 3d environment. In *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 480–485. IEEE, 2019.

[40] Scott A Bortoff. Path planning for uavs. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, volume 1, pages 364–368. IEEE, 2000.

[41] Xia Chen, Guang-yao Li, and Xiang-min Chen. Path planning and cooperative control for multiple uavs based on consistency theory and voronoi diagram. In *2017 29th Chinese Control and Decision Conference (CCDC)*, pages 881–886. IEEE, 2017.

[42] Xu Wei, Duan Fengyang, Zhang Qingjie, Zhu Bing, and Sun Hongchang. A new fast consensus algorithm applied in rendezvous of multi-uav. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 55–60. IEEE, 2015.

[43] Jaeuk Baek, Sang Ik Han, and Youngnam Han. Energy-efficient uav routing for wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 69(2):1741–1750, 2019.

[44] Xin Feng and Alan T Murray. Allocation using a heterogeneous space voronoi diagram. *Journal of Geographical Systems*, 20(3):207–226, 2018.

[45] Xia Chen and Miaoyan Zhao. Collaborative path planning for multiple unmanned aerial vehicles to avoid sudden threats. In *2019 Chinese Automation Congress (CAC)*, pages 2196–2201. IEEE, 2019.

[46] Sangwoo Moon, Eunmi Oh, and David Hyunchul Shim. An integral framework of task assignment and path planning for multiple unmanned aerial vehicles in dynamic environments. *Journal of Intelligent & Robotic Systems*, 70(1-4):303–313, 2013.

[47] Xue Qian, Cheng Peng, Cheng Nong, and Zou Xiang. Dynamic obstacle avoidance path planning of uavs. In *2015 34th Chinese Control Conference (CCC)*, pages 8860–8865. IEEE, 2015.

[48] Almira Budiyanto, Adha Cahyadi, Teguh Bharata Adji, and Oyas Wahyunggoro. Uav obstacle avoidance using potential field under dynamic environment. In *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pages 187–192. IEEE, 2015.

[49] Yong-bo Chen, Guan-chen Luo, Yue-song Mei, Jian-qiao Yu, and Xiao-long Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016.

[50] Yuecheng Liu and Yongjia Zhao. A virtual-waypoint based artificial potential field method for uav path planning. In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pages 949–953. IEEE, 2016.

[51] Thi Thoa Mac, Cosmin Copot, Andres Hernandez, and Robin De Keyser. Improved potential field method for unknown obstacle avoidance using uav in indoor environment. In *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 345–350. IEEE, 2016.

[52] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Algorithm for energy efficient inter-uav collision avoidance. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–5. IEEE, 2017.

[53] Jiayi Sun, Jun Tang, and Songyang Lao. Collision avoidance for cooperative uavs with optimized artificial potential field algorithm. *IEEE Access*, 5:18382–18390, 2017.

[54] Alexander C Woods and Hung M La. A novel potential field controller for use on aerial robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(4):665–676, 2017.

[55] Liu Zhiyang and Jiang Tao. Route planning based on improved artificial potential field method. In *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 196–199. IEEE, 2017.

[56] Jiyang Dai, Yunxia Wang, Cunsong Wang, Jin Ying, and Jinyou Zhai. Research on hierarchical potential field method of path planning for uavs. In *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 529–535. IEEE, 2018.

[57] Wenbin Bai, Xiande Wu, Yaen Xie, Yu Wang, Han Zhao, Kefan Chen, Yong Li, and Yong Hao. A cooperative route planning method for multi-uavs based-on the fusion of artificial potential field and b-spline interpolation. In *2018 37th Chinese Control Conference (CCC)*, pages 6733–6738. IEEE, 2018.

[58] PV Sankar and Leonard A. Ferrari. Simple algorithms and architectures for b-spline interpolation. *IEEE transactions on pattern analysis and machine intelligence*, 10(2):271–276, 1988.

[59] Yunduo Feng, Yanxuan Wu, Haozhe Cao, and Jiankun Sun. Uav formation and obstacle avoidance based on improved apf. In *2018 10th International Conference on Modelling, Identification and Control (ICMIC)*, pages 1–6. IEEE, 2018.

[60] Zhang Yingkun. Flight path planning of agriculture uav based on improved artificial potential field method. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1526–1530. IEEE, 2018.

[61] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Potential field based inter-uav collision avoidance using virtual target relocation. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2018.

[62] Egidio D'Amato, Massimiliano Mattei, and Immacolata Notaro. Bi-level flight path planning of uav formations with collision avoidance. *Journal of Intelligent & Robotic Systems*, 93(1-2):193–211, 2019.

[63] Egidio D'Amato, Immacolata Notaro, Luciano Blasi, and Massimiliano Mattei. Smooth path planning for fixed-wing aircraft in 3d environment using a layered essential visibility graph. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 9–18. IEEE, 2019.

[64] Parikshit Maini and PB Sujit. Path planning for a uav with kinematic constraints in the presence of polygonal obstacles. In *2016 international conference on unmanned aircraft systems (ICUAS)*, pages 62–67. IEEE, 2016.

[65] Jingjing Wang, YF Zhang, L Geng, Jerry YH Fuh, and SH Teo. A heuristic mission planning algorithm for heterogeneous tasks with heterogeneous uavs. *Unmanned Systems*, 3(03):205–219, 2015.

[66] L Geng, YF Zhang, Jingjing Wang, Jerry YH Fuh, and SH Teo. Cooperative mission planning with multiple uavs in realistic environments. *Unmanned Systems*, 2(01):73–86, 2014.

[67] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[68] Zhuoning Dong, Zongji Chen, Rui Zhou, and Rulin Zhang. A hybrid approach of virtual force and a search algorithm for uav path re-planning. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 1140–1145. IEEE, 2011.

[69] ZN Dong, Pei Chi, RL Zhang, and ZJ Chen. The algorithms on three-dimension route plan based on virtual forces. *Journal of System Simulation*, 20(S):387–392, 2009.

[70] Zhu Wang, Li Liu, Teng Long, Chenglong Yu, and Jiaxun Kou. Enhanced sparse a* search for uav path planning using dubins path estimation. In *Proceedings of the 33rd Chinese Control Conference*, pages 738–742. IEEE, 2014.

[71] Robert J Szczerba, Peggy Galkowski, Ira S Glicktein, and Noah Ternullo. Robust algorithm for real-time route planning. *IEEE Transactions on Aerospace and Electronic Systems*, 36(3):869–878, 2000.

[72] Ren Tianzhu, Zhou Rui, Xia Jie, and Dong Zhuoning. Three-dimensional path planning of uav based on an improved a* algorithm. In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pages 140–145. IEEE, 2016.

[73] Zhang Chengjun and Meng Xiuyun. Spare a search approach for uav route planning. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 413–417. IEEE, 2017.

[74] Tianyou Chen, Guofeng Zhang, Xiaoguang Hu, and Jin Xiao. Unmanned aerial vehicle route planning method based on a star algorithm. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1510–1514. IEEE, 2018.

[75] Guohao Zhang and Li-Ta Hsu. A new path planning algorithm using a gnss localization error map for uavs in an urban area. *Journal of Intelligent & Robotic Systems*, 94(1):219–235, 2019.

[76] Stefano Primatesta, Giorgio Guglieri, and Alessandro Rizzo. A risk-aware path planning strategy for uavs in urban environments. *Journal of Intelligent & Robotic Systems*, 95(2):629–643, 2019.

[77] Afshin Mardani, Marcello Chiaberge, and Paolo Giaccone. Communication-aware uav path planning. *IEEE Access*, 7:52609–52621, 2019.

[78] Xueli Wu, Lei Xu, Ran Zhen, and Xiaojing Wu. Bi-directional adaptive a* algorithm toward optimal path planning for large-scale uav under multi-constraints. *IEEE Access*, 8:85431–85440, 2020.

[79] Zhe Zhang, Jian Wu, Jiyang Dai, and Cheng He. A novel real-time penetration path planning algorithm for stealth uav in 3d complex dynamic environment. *IEEE Access*, 8:122757–122771, 2020.

[80] Abdul Afram, Farrokh Janabi-Sharifi, Alan S Fung, and Kaamran Raahemifar. Artificial neural network (ann) based model predictive control (mpc) and optimization of hvac systems: A state of the art review and case study of a residential hvac system. *Energy and Buildings*, 141:96–113, 2017.

[81] Wei Liu, Zheng Zheng, and Kai-Yuan Cai. Bi-level programming based real-time path planning for unmanned aerial vehicles. *Knowledge-Based Systems*, 44:34–47, 2013.

[82] Patrice Marcotte and Gilles Savard. Bilevel programming: A combinatorial perspective. In *Graph theory and combinatorial optimization*, pages 191–217. Springer, 2005.

[83] Minyang Kang, Yang Liu, Yijie Ren, Yijing Zhao, and Zheng Zheng. An empirical study on robustness of uav path planning algorithms considering position uncertainty. In *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 1–6. IEEE, 2017.

[84] Shaimaa Ahmed, Amr Mohamed, Khaled Harras, Mohamed Kholief, and Saleh Mesbah. Energy efficient path planning techniques for uav-based systems with space discretization. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6. IEEE, 2016.

[85] Jesimar da Silva Arantes, Márcio da Silva Arantes, Claudio Fabiano Motta Toledo, Onofre Trindade Júnior, and Brian Charles Williams. Heuristic and genetic algorithm approaches for uav path planning under critical situation. *International Journal on Artificial Intelligence Tools*, 26(01):1760008, 2017.

[86] Heitor Freitas, Bruno S Faiçal, Alef Vinicius Cardoso e, and Jó Ueyama. Use of uavs for an efficient capsule distribution and smart path planning for biological pest control. *Computers and Electronics in Agriculture*, 173:105387, 2020.

[87] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path planning strategies for uavs in 3d environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264, 2012.

[88] Yi-Chen Du, Min-Xia Zhang, Hai-Feng Ling, and Yu-Jun Zheng. Evolutionary planning of multi-uav search for missing tourists. *IEEE Access*, 7:73480–73492, 2019.

[89] Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[90] Athina N Brintaki and Ioannis K Nikolos. Coordinated uav path planning using differential evolution. *Operational Research*, 5(3):487–502, 2005.

[91] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[92] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[93] Shashi Mittal and Kalyanmoy Deb. Three-dimensional offline path planning for uavs using multiobjective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3195–3202. IEEE, 2007.

[94] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on industrial informatics*, 9(1):132–141, 2012.

[95] Xiangyin Zhang and Haibin Duan. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Applied Soft Computing*, 26:270–284, 2015.

[96] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, and Zhong Ming. A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, 2016.

[97] Debesh Adhikari, Eunjin Kim, and Hassan Reza. A fuzzy adaptive differential evolution for multi-objective 3d uav path optimization. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2258–2265. IEEE, 2017.

[98] Lotfi A Zadeh. Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific, 1996.

[99] Zhangjie Fu, Jingnan Yu, Guowu Xie, Yiming Chen, and Yuanhang Mao. A heuristic evolutionary algorithm of uav path planning. *Wireless Communications and Mobile Computing*, 2018, 2018.

[100] Rui Dai, Sneha Fotedar, Mohammadreza Radmanesh, and Manish Kumar. Quality-aware uav coverage and path planning in geometrically complex environments. *Ad Hoc Networks*, 73:95–105, 2018.

[101] Chunhui Xiao, Yuanyuan Zou, and Shaoyuan Li. Uav multiple dynamic objects path planning in air-ground coordination using receding horizon strategy. In *2019 3rd International Symposium on Autonomous Systems (ISAS)*, pages 335–340. IEEE, 2019.

[102] Qiansheng Yang, Jing Liu, and Liqiang Li. Path planning of uavs under dynamic environment based on a hierarchical recursive multiagent genetic algorithm. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[103] Samira Hayat, Evşen Yanmaz, Christian Bettstetter, and Timothy X Brown. Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Autonomous Robots*, 44(7):1183–1198, 2020.

[104] Vrajesh Kumar Chawra and Govind P Gupta. Multiple uav path-planning for data collection in cluster-based wireless sensor network. In *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, pages 194–198. IEEE, 2020.

[105] Yangguang Fu, Mingyue Ding, and Chengping Zhou. Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for uav. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(2):511–526, 2011.

[106] Zhong Wei-Min, Li Shao-Jun, and Qian Feng. θ-pso: a new strategy of particle swarm optimization. *Journal of Zhejiang University-SCIENCE A*, 9(6):786–790, 2008.

[107] Jun Sun, Bin Feng, and Wenbo Xu. Particle swarm optimization with particles having quantum behavior. In *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, volume 1, pages 325–331. IEEE, 2004.

[108] Yang Liu, Xuejun Zhang, Xiangmin Guan, and Daniel Delahaye. Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization. *Aerospace Science and Technology*, 58:92–102, 2016.

[109] Ugur Cekmez, Mustafa Ozsiginan, and Ozgur Koray Sahingoz. Multi colony ant optimization for uav path planning with obstacle avoidance. In *2016 international conference on unmanned aircraft systems (ICUAS)*, pages 47–52. IEEE, 2016.

[110] Peng Yao and Honglun Wang. Dynamic adaptive ant lion optimizer applied to route planning for unmanned aerial vehicle. *Soft Computing*, 21(18):5475–5488, 2017.

[111] Keyu Wu, Tao Xi, and Han Wang. Real-time three-dimensional smooth path planning for unmanned aerial vehicles in completely unknown cluttered environments. In *TENCON 2017-2017 IEEE Region 10 Conference*. IEEE, 2017.

[112] Chen YongBo, Mei YueSong, Yu JianQiao, Su XiaoLong, and Xu Nuo. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing*, 266:445–457, 2017.

[113] Chen Huang and Jiyou Fei. Uav path planning based on particle swarm optimization with global best path competition. *International Journal of Pattern Recognition and Artificial Intelligence*, 32(06):1859008, 2018.

[114] Guozun Tian, Lan Zhang, Xin Bai, and Bing Wang. Real-time dynamic track planning of multi-uav formation based on improved artificial bee colony algorithm. In *2018 37th Chinese Control Conference (CCC)*, pages 10055–10060. IEEE, 2018.

[115] Jianfa Wu, Honglun Wang, Na Li, Peng Yao, Yu Huang, and Hemeng Yang. Path planning for solar-powered uav in urban environment. *Neurocomputing*, 275:2055–2065, 2018.

[116] Xiangyin Zhang, Xingyang Lu, Songmin Jia, and Xiuzhi Li. A novel phase angle-encoded fruit fly optimization algorithm with mutation adaptation mechanism applied to uav path planning. *Applied Soft Computing*, 70:371–388, 2018.

[117] Prashant Pandey, Anupam Shukla, and Ritu Tiwari. Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. *International Journal of System Assurance Engineering and Management*, 9(4):836–852, 2018.

[118] KN Krishnanand and Debasish Ghose. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm intelligence*, 3(2):87–124, 2009.

[119] Utkarsh Goel, Shubham Varshney, Anshul Jain, Saumil Maheshwari, and Anupam Shukla. Three dimensional path planning for uavs in dynamic environment using glow-worm swarm optimization. *Procedia computer science*, 133:230–239, 2018.

[120] Daifeng Zhang and Haibin Duan. Social-class pigeon-inspired optimization and time stamp segmentation for multi-uav cooperative path planning. *Neurocomputing*, 313:229–246, 2018.

[121] Xixia Sun, Su Pan, Chao Cai, Yanfang Chen, and Jie Chen. Unmanned aerial vehicle path planning based on improved intelligent water drop algorithm. In *2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, pages 867–872. IEEE, 2018.

[122] I Wayan Muliawan, Muhammad Anwar Ma'Sum, Noverina Alfiany, and Wisnu Jatmiko. Uav path planning for autonomous spraying task at salak plantation based on the severity of plant disease. In *2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, pages 109–113. IEEE, 2019.

[123] Ram Kishan Dewangan, Anupam Shukla, and W Wilfred Godfrey. Three dimensional path planning using grey wolf optimizer for uavs. *Applied Intelligence*, 49(6):2201–2217, 2019.

[124] Yawei Cai, Hui Zhao, Mudong Li, and Hanqiao Huang. 3d real-time path planning based on cognitive behavior optimization algorithm for uav with tlp model. *Cluster Computing*, 22(2):5089–5098, 2019.

[125] Mudong Li, Hui Zhao, Xingwei Weng, and Tong Han. Cognitive behavior optimization algorithm for solving optimization problems. *Applied Soft Computing*, 39:199–222, 2016.

[126] Chao Zhang, Chenxi Hu, Jianrui Feng, Zhenbao Liu, Yong Zhou, and Zexu Zhang. A self-heuristic ant-based method for path planning of unmanned aerial vehicle in complex 3-d space with dense u-type obstacles. *IEEE Access*, 7:150775–150791, 2019.

[127] Xiaofei Wang, Hui Zhao, Tong Han, Huan Zhou, and Cong Li. A grey wolf optimizer using gaussian estimation of distribution and its application in the multi-uav multi-target urban tracking problem. *Applied Soft Computing*, 78:240–260, 2019.

[128] Longwang Yue and Hanning Chen. Unmanned vehicle path planning using a novel ant colony algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):136, 2019.

[129] Bo Li, Xiaogang Qi, Baoguo Yu, and Lifang Liu. Trajectory planning for uav based on improved aco algorithm. *IEEE Access*, 8:2995–3006, 2019.

[130] Shikai Shao, Yu Peng, Chenglong He, and Yun Du. Efficient path planning for uav formation via comprehensively improved particle swarm optimization. *ISA transactions*, 97:415–430, 2020.

[131] LIU Yang, Xuejun Zhang, Yu Zhang, and GUAN Xiangmin. Collision free 4d path planning for multiple uavs based on spatial refined voting mechanism and pso approach. *Chinese Journal of Aeronautics*, 32(6):1504–1519, 2019.

[132] Liuqing Yang, Jin Guo, and Yanbin Liu. Three-dimensional uav cooperative path planning based on the mp-cgwo algorithm. *Int. J. Innov. Comput. Inf. Control*, 16:991–1006, 2020.

[133] Yongbo Chen, Jianqiao Yu, Yuesong Mei, Yafei Wang, and Xiaolong Su. Modified central force optimization (mcfo) algorithm for 3d uav path planning. *Neurocomputing*, 171:878–888, 2016.

[134] Puneet Kumar, Sahil Garg, Amritpal Singh, Shalini Batra, Neeraj Kumar, and Ilsun You. Mvo-based 2-d path planning scheme for providing quality of service in uav environment. *IEEE Internet of Things Journal*, 5(3):1698–1707, 2018.

[135] Gatij Jain, Gaurav Yadav, Dhruv Prakash, Anupam Shukla, and Ritu Tiwari. Mvo-based path planning scheme with coordination of uavs in 3-d environment. *Journal of Computational Science*, 37:101016, 2019.

[136] Adis Alihodzic. Fireworks algorithm with new feasibility-rules in solving uav path planning. In *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, pages 53–57. IEEE, 2016.

[137] Jiayang Wu, Jin Yi, Liang Gao, and Xinyu Li. Cooperative path planning of multiple uavs based on ph curves and harmony search algorithm. In *2017 IEEE 21St international conference on computer supported cooperative work in design (CSCWD)*, pages 540–544. IEEE, 2017.

[138] Hamidullah Binol, Eyuphan Bulut, Kemal Akkaya, and Ismail Guvenc. Time optimal multi-uav path planning for gathering its data from roadside units. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2018.

[139] Ioannis K Nikolos, Eleftherios S Zografos, and Athina N Brintaki. Uav path planning using evolutionary algorithms. In *Innovations in intelligent machines-1*, pages 77–111. Springer, 2007.

[140] Juan Wu, Seabyuk Shin, Cheong-Gil Kim, and Shin-Dug Kim. Effective lazy training method for deep q-network in obstacle avoidance and path planning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1799–1804. IEEE, 2017.

[141] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[142] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. *Journal of Intelligent & Robotic Systems*, pages 1–13, 2019.

[143] Hamid Shiri, Jihong Park, and Mehdi Bennis. Remote uav online path planning via neural network based opportunistic control. *IEEE Wireless Communications Letters*, 2020.

[144] Yanhong Chen, Wei Zu, Guoliang Fan, and Hongxing Chang. Unmanned aircraft vehicle path planning based on svm algorithm. In *Foundations and Practical Applications of Cognitive Systems and Information Processing*, pages 705–714. Springer, 2014.

[145] Jaehyun Yoo, H Jin Kim, and Karl H Johansson. Path planning for remotely controlled uavs using gaussian process filter. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 477–482. IEEE, 2017.

[146] Kwang Min Koo, Kyung Rak Lee, Sung Ryung Cho, and Inwhee Joe. A uav path planning method using polynomial regression for remote sensor data collection. In *Advances in Computer Science and Ubiquitous Computing*, pages 428–433. Springer, 2018.

[147] Reza Radmanesh, Manish Kumar, Donald French, and David Casbeer. Towards a pde-based large-scale decentralized solution for path planning of uavs in shared airspace. *Aerospace Science and Technology*, page 105965, 2020.

[148] Shankarachary Ragi and Edwin KP Chong. Uav path planning in a dynamic environment via partially observable markov decision process. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2397–2412, 2013.

[149] Baochang Zhang, Wanquan Liu, Zhili Mao, Jianzhuang Liu, and Linlin Shen. Cooperative and geometric learning algorithm (cgla) for path planning of uavs with limited information. *Automatica*, 50(3):809–820, 2014.

[150] Zhao Yijing, Zheng Zheng, Zhang Xiaoyi, and Liu Yang. Q learning algorithm based uav path learning and obstacle avoidence approach. In *2017 36th Chinese Control Conference (CCC)*, pages 3397–3402. IEEE, 2017.

[151] Ursula Challita, Walid Saad, and Christian Bettstetter. Deep reinforcement learning for interference-aware path planning of cellular-connected uavs. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[152] Wei Luo, Qirong Tang, Changhong Fu, and Peter Eberhard. Deep-sarsa based multi-uav path planning and obstacle avoidance in a dynamic environment. In *International Conference on Sensing and Imaging*, pages 102–111. Springer, 2018.

[153] Chao Yan and Xiaojia Xiang. A path planning algorithm for uav based on improved q-learning. In *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*, pages 1–5. IEEE, 2018.

[154] Tianze Zhang, Xin Huo, Songlin Chen, Baoqing Yang, and Guojiang Zhang. Hybrid path planning of a quadrotor uav based on q-learning algorithm. In *2018 37th Chinese Control Conference (CCC)*, pages 5415–5419. IEEE, 2018.

[155] Ronglei Xie, Zhijun Meng, Yaoming Zhou, Yunpeng Ma, and Zhe Wu. Heuristic q-learning based on experience replay for three-dimensional path planning of the unmanned aerial vehicle. *Science Progress*, 103(1):0036850419879024, 2020.

[156] Djamalladine Mahamt Pierre, Nordin Zakaria, and Anindya Jyoti Pal. Self-organizing map approach to determining compromised solutions for multi-objective uav path planning. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 995–1000. IEEE, 2012.

[157] Youngjun Choi, Hernando Jimenez, and Dimitri N Mavris. Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories. *Robotics and Autonomous Systems*, 98:158–173, 2017.

[158] Xia Chen and Xiangmin Chen. The uav dynamic path planning algorithm research based on voronoi diagram. In *The 26th chinese control and decision conference (2014 ccdc)*, pages 1069–1071. IEEE, 2014.

[159] Denggui Zhang, Yong Xu, and Xingting Yao. An improved path planning algorithm for unmanned aerial vehicle based on rrt-connect. In *2018 37th Chinese Control Conference (CCC)*, pages 4854–4858. IEEE, 2018.

[160] Hai Wang, Zhi Sun, Dazi Li, and Qibing Jin. An improved rrt based 3-d path planning algorithm for uav. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 5514–5519. IEEE, 2019.

[161] Hong Shen and Ping Li. Unmanned aerial vehicle (uav) path planning based on improved pre-planning artificial potential field method. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 2727–2732. IEEE, 2020.

[162] Sanjoy Kumar Debnath, Rosli Omar, Susama Bagchi, Marwan Nafea, Ranesh Kumar Naha, and Elia Nadira Sabudin. Energy efficient elliptical concave visibility graph algorithm for unmanned aerial vehicle in an obstacle-rich environment. In *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 129–134. IEEE, 2020.

[163] Phillip Chandler, Steven Rasmussen, and Meir Pachter. Uav cooperative path planning. In *AIAA guidance, navigation, and control conference and exhibit*, page 4370, 2000.

[164] Fei Yan, Yan Zhuang, and Jizhong Xiao. 3d prm based real-time path planning for uav in complex environment. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1135–1140. IEEE, 2012.

[165] Qian Xue, Peng Cheng, and Nong Cheng. Offline path planning and online replanning of uavs in complex terrain. In *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, pages 2287–2292. IEEE, 2014.

[166] Zahoor Ahmad, Farman Ullah, Cong Tran, and Sungchang Lee. Efficient energy flight path planning algorithm using 3-d visibility roadmap for small unmanned aerial vehicle. *International Journal of Aerospace Engineering*, 2017, 2017.

[167] Menaka Naazare, Diego Ramos, Joerg Wildt, and Dirk Schulz. Application of graph-based path planning for uavs to avoid restricted areas. In *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 139–144. IEEE, 2019.

[168] Yaohong Qu, Yintao Zhang, and Youmin Zhang. A global path planning algorithm for fixed-wing uavs. *Journal of Intelligent & Robotic Systems*, 91(3-4):691–707, 2018.

[169] Y Volkan Pehlivanoglu. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. *Aerospace Science and Technology*, 16(1):47–55, 2012.

[170] Márcio da Arantes, Jesimar da Arantes, Claudio Fabiano Motta Toledo, and Brian C Williams. A hybrid multi-population genetic algorithm for uav path planning. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 853–860. ACM, 2016.

[171] S Girija and Ashok Joshi. Fast hybrid pso-apf algorithm for path planning in obstacle rich environment. *IFAC-PapersOnLine*, 52(29):25–30, 2019.

[172] Vincent Roberge, Mohammed Tarbouchi, and François Allaire. Parallel hybrid metaheuristic on shared memory system for real-time uav path planning. *International Journal of Computational Intelligence and Applications*, 13(02):1450008, 2014.

[173] Soheila Ghambari, Lhassane Idoumghar, Laetitia Jourdan, and Julien Lepagnot. An improved tlbo algorithm for solving uav path planning problem. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2261–2268. IEEE, 2019.

[174] Zain Anwar Ali, Han Zhangang, and Di Zhengru. Path planning of multiple uavs using mmaco and de algorithm in dynamic environment. *Measurement and Control*, page 0020294020915727, 2020.

[175] Fawei Ge, Kun Li, Ying Han, Wensu Xu, et al. Path planning of uav for oilfield inspections in a three-dimensional dynamic environment with moving obstacles based on an improved pigeon-inspired optimization algorithm. *Applied Intelligence*, pages 1–18, 2020.

[176] Chengzhi Qu, Wendong Gai, Jing Zhang, and Maiying Zhong. A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (uav) path planning. *Knowledge-Based Systems*, page 105530, 2020.

[177] Chengzhi Qu, Wendong Gai, Jing Zhang, and Maiying Zhong. A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (uav) path planning. *Knowledge-Based Systems*, 194:105530, 2020.

[178] Amylia Ait-Saadi, Yassine Meraihi, Assia Soukane, Amar Ramdane-Cherif, and Asma Benmessaoud Gabis. A novel hybrid chaotic aquila optimization algorithm with simulated annealing for unmanned aerial vehicles path planning. *Computers and Electrical Engineering*, 104:108461, 2022.

[179] Benyamin Abdollahzadeh, Farhad Soleimanian Gharehchopogh, and Seyedali Mirjalili. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Computers & Industrial Engineering*, 158:107408, 2021.

[180] Yantao Li, Shaojiang Deng, and Di Xiao. A novel hash algorithm construction based on chaotic neural network. *Neural Computing and Applications*, 20(1):133–141, 2011.

[181] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102, 1999.

[182] Fábio AP Paiva, Cláudio RM Silva, Izabele VO Leite, Marcos HF Marcone, and José AF Costa. Modified bat algorithm with cauchy mutation and elite opposition-based learning. In *2017 IEEE Latin American conference on computational intelligence (LA-CCI)*, pages 1–6. IEEE, 2017.

[183] Gai-Ge Wang, Suash Deb, Amir H Gandomi, and Amir H Alavi. Opposition-based krill herd algorithm with cauchy mutation and position clamping. *Neurocomputing*, 177:147–157, 2016.

[184] Musrrat Ali and Millie Pant. Improving the performance of differential evolution algorithm using cauchy mutation. *Soft Computing*, 15(5):991–1007, 2011.

[185] Yongquan Zhou, Rui Wang, and Qifang Luo. Elite opposition-based flower pollination algorithm. *Neurocomputing*, 188:294–310, 2016.

[186] Junliang Shang, Yan Sun, Shengjun Li, Jin-Xing Liu, Chun-Hou Zheng, and Junying Zhang. An improved opposition-based learning particle swarm optimization for the detection of snp-snp interactions. *BioMed research international*, 2015, 2015.

[187] Alaa A Alomoush, Abdulrahman A Alsewari, Hammoudeh S Alamri, Khalid Aloufi, and Kamal Z Zamli. Hybrid harmony search algorithm with grey wolf optimizer and modified opposition-based learning. *IEEE Access*, 7:68764–68785, 2019.

[188] Shalini Shekhawat and Akash Saxena. Development and applications of an intelligent crow search algorithm based on opposition based learning. *ISA transactions*, 99:210–230, 2020.

[189] Timea Bezdan, Aleksandar Petrovic, Miodrag Zivkovic, Ivana Strumberger, V Kanchana Devi, and Nebojsa Bacanin. Current best opposition-based learning salp swarm algorithm for global numerical optimization. In *2021 Zooming Innovation in Consumer Technologies Conference (ZINC)*, pages 5–10. IEEE, 2021.

[190] Laith Abualigah, Dalia Yousri, Mohamed Abd Elaziz, Ahmed A Ewees, Mohammed A A Al-Qaness, and Amir H Gandomi. Aquila optimizer: a novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering*, 157:107250, 2021.

[191] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[192] Xiaoming Zhang and Tinghao Feng. Chaotic bean optimization algorithm. *Soft Computing*, 22(1):67–77, 2018.

[193] Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. Biogeography-based optimisation with chaos. *Neural Computing and Applications*, 25(5):1077–1097, 2014.

[194] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[195] Xin-She Yang and Amir Hossein Gandomi. Bat algorithm: a novel approach for global engineering optimization. *Engineering computations*, 2012.

[196] Xin-She Yang. Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178. Springer, 2009.

[197] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[198] Seyedali Mirjalili. Sca: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, 96:120–133, 2016.

[199] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in engineering software*, 95:51–67, 2016.

[200] Seyedali Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural computing and applications*, 27(4):1053–1073, 2016.

[201] Sérgio Sabino and António Grilo. Topology control of unmanned aerial vehicle (uav) mesh networks: A multi-objective evolutionary algorithm approach. In *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, pages 45–50, 2018.

[202] Haibo Wang, Da Huo, and Bahram Alidaee. Position unmanned aerial vehicles in the mobile ad hoc network. *Journal of Intelligent & Robotic Systems*, 74(1):455–464, 2014.

[203] Ivana Strumberger, Nebojsa Bacanin, Slavisa Tomic, Marko Beko, and Milan Tuba. Static drone placement by elephant herding optimization algorithm. In *2017 25th Telecommunication Forum (Telfor)*, pages 1–4. IEEE, 2017.

[204] Shams ur Rahman, Geon-Hwan Kim, You-Ze Cho, and Ajmal Khan. Positioning of uavs for throughput maximization in software-defined disaster area uav communication networks. *Journal of Communications and Networks*, 20(5):452–463, 2018.

[205] DG Reina, Hissam Tawfik, and SL Toral. Multi-subpopulation evolutionary algorithms for coverage deployment of uav-networks. *Ad Hoc Networks*, 68:16–32, 2018.

[206] Fadi Al-Turjman, Joel Poncha Lemayian, Sinem Alturjman, and Leonardo Mostarda. Enhanced deployment strategy for the 5g drone-bs using artificial intelligence. *IEEE Access*, 7:75999–76008, 2019.

[207] Elhadja Chaalal, Laurent Reynaud, and Sidi Mohammed Senouci. A social spider optimisation algorithm for 3d unmanned aerial base stations placement. In *2020 IFIP Networking Conference (Networking)*, pages 544–548. IEEE, 2020.

[208] Lei Yang, Haipeng Yao, Jingjing Wang, Chunxiao Jiang, Abderrahim Benslimane, and Yunjie Liu. Multi-uav-enabled load-balance mobile-edge computing for iot networks. *IEEE Internet of Things Journal*, 7(8):6898–6908, 2020.

[209] Manali Gupta and Shirshu Varma. Optimal placement of uavs of an aerial mesh network in an emergency situation. *Journal of Ambient Intelligence and Humanized Computing*, 12(1):343–358, 2021.

[210] Manali Gupta and Shirshu Varma. Metaheuristic-based optimal 3d positioning of uavs forming aerial mesh network to provide emergency communication services. *IET Communications*, 15(10):1297–1314, 2021.

[211] Ahmad Sawalmeh, Noor Shamsiah Othman, Guanxiong Liu, Abdallah Khreishah, Ali Alenezi, and Abdulaziz Alanazi. Power-efficient wireless coverage using minimum number of uavs. *Sensors*, 22(1):223, 2021.

[212] Vicente Mayor, Rafael Estepa, and Antonio Estepa. Qos-aware multilayer uav deployment to provide vowifi service over 5g networks. *Wireless Communications and Mobile Computing*, 2022, 2022.

[213] Recep Ozdag. Multi-metric optimization with a new metaheuristic approach developed for 3d deployment of multiple drone-bss. *Peer-to-Peer Networking and Applications*, 15(3):1535–1561, 2022.

[214] Mohamed Amine Ouamri, Marius-Emil Oteşteanu, Gordana Barb, and Cedric Gueguen. Coverage analysis and efficient placement of drone-bss in 5g networks. *Engineering Proceedings*, 14(1):18, 2022.

[215] Prateek Mahajan, Anusha Kumar, GSS Chalapathi, and Rajkumar Buyya. Efta: An energy-efficient, fault-tolerant, and area-optimized uav placement scheme for search operations. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[216] Wei Wei, Jingjing Wang, Zhengru Fang, Jianrui Chen, Yong Ren, and Yuhan Dong. 3u: Joint design of uav-usv-uuv networks for cooperative target hunting. *IEEE Transactions on Vehicular Technology*, 2022.

[217] Yaxi Liu, Wei Huangfu, Huan Zhou, Haijun Zhang, Jiangchuan Liu, and Keping Long. Fair and energy-efficient coverage optimization for uav placement problem in the cellular network. *IEEE Transactions on Communications*, 2022.

[218] Yongguo Mei, Yung-Hsiang Lu, Y Charlie Hu, and CS George Lee. A case study of mobile robot's energy consumption and conservation techniques. In *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 492–497. IEEE, 2005.

[219] Luigi Di Puglia Pugliese, Francesca Guerriero, Dimitrios Zorbas, and Tahiry Razafindralambo. Modelling the mobile target covering problem using flying drones. *Optimization Letters*, 10(5):1021–1052, 2016.

[220] Weiguo Zhao, Zhenxing Zhang, and Liying Wang. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence*, 87:103300, 2020.

[221] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[222] Meng Qiu, Zhuo Fu, Richard Eglese, and Qiong Tang. A tabu search algorithm for the vehicle routing problem with discrete split deliveries and pickups. *Computers & Operations Research*, 100:102–116, 2018.

[223] Xiaochen Chou, Luca Maria Gambardella, and Roberto Montemanni. A tabu search algorithm for the probabilistic orienteering problem. *Computers & Operations Research*, 126:105107, 2021.

[224] Fatos Xhafa, Christian Sánchez, Admir Barolli, and Makoto Takizawa. Solving mesh router nodes placement problem in wireless mesh networks by tabu search algorithm. *Journal of Computer and System Sciences*, 81(8):1417–1428, 2015.

[225] Jan Van Belle, Paul Valckenaers, Greet Vanden Berghe, and Dirk Cattrysse. A tabu search approach to the truck scheduling problem with multiple docks and time windows. *Computers & Industrial Engineering*, 66(4):818–826, 2013.

[226] Surafel Luleseged Tilahun. Balancing the degree of exploration and exploitation of swarm intelligence using parallel computing. *International Journal on Artificial Intelligence Tools*, 28(03):1950014, 2019.

[227] Laith Abualigah, Mohamed Abd Elaziz, Putra Sumari, Zong Woo Geem, and Amir H Gandomi. Reptile search algorithm (rsa): A nature-inspired meta-heuristic optimizer. *Expert Systems with Applications*, 191:116158, 2022.