Low carbon footprint intrusion detection in IoT systems based on Deep Learning algorithms combined with a metaheuristic

Mohamed Sassi

CY CERGY PARIS UNIVERSITÉ

DOCTORAL SCHOOL (ED 407)

ECONOMIC, MANAGEMENT, MATHEMATICS, PHYSICS AND COMPUTER SCIENCES (EM2PSI)

A THESIS

PRESENTED AS A REQUIREMENT TO AIM FOR THE DEGREE OF 3rd CYCLE DOCTORATE IN SYSTEMS AND INFORMATION AND COMMUNICATION PROCESSING (STIC)

Low carbon footprint intrusion detection in IoT systems based on Deep Learning combined with a metaheuristic

By:

Sassi Mohamed

Defended at 3 may 2023

Approved by supervisory committee:

Jourdan	Laeitia	Prof. Univ Lille	**President**
Siarry	Patrick	Prof. Univ Paris-Est Créteil	**Rapporteur**
Barkaoui	Kamel	Prof. Univ CNAM	**Rapporteur**
Labadi	Karim	Prof. ECAM-EPMI	**Examinator**
Chelouah	Rachid	Prof. Univ CY-Tech	**Supervisor**
Davadie	Philippe	DER PJGN	**Invited**

Résumé

Face à la problématique croissante de la cybersécurité des objets connectés (IoT) au sein des réseaux Wi-Fi d'entreprise et étatique tel que la Gendarmerie nationale, cette thèse expose notre méthode de conception d'un Système de Détection d'Intrusion (IDS) contre les attaques spécifiques 802.11 et embarquable dans un IoT aux ressources de calcul et de mémoire limitées. Notre méthode est basée sur l'optimisation métaheuristique, la sélection des dimensions et les algorithmes Deep Learning.

Le problème d'optimisation de sélection des dimensions est un problème discret de type NP-Hard. Dans la deuxième partie de cette thèse nous avons donc conçu une nouvelle métaheuristique, Harris Hawk Optimization Encirclement Attack Synergy (HHO-EAS), en mesure de trouver une bonne solution acceptable optimale ou proche de l'optimale en un temps raisonnable à cette catégorie de problème d'optimisation. HHO-EAS est issue d'une stratégie d'hybridation de la métaheuristique Harris Hawk Optimization (HHO) avec l'ambition de supprimer ses faiblesses dans la résolution des problèmes d'optimisation hautement multimodaux et de grande dimension. Notre stratégie d'hybridation est entièrement bio-inspirée par une synergie de chasse gagnant-gagnant inopinée entre deux prédateurs pendant les périodes hivernales extrêmement difficiles : le corbeau et le loup. Les facultés exploratoires des corbeaux combinées à la capacité des loups à capturer des proies plus grosses qu'eux avec rapidité et efficacité, permettent à ces deux prédateurs de détecter et d'attraper de bonnes proies très rares et très difficiles à chasser en période hivernale rigoureuse. Afin de modéliser mathématiquement cette synergie de chasse gagnant-gagnant avec les équations d'encerclement et d'attaque et de l'intégrer dans HHO, nous avons utilisé la logique floue pour créer un système d'inférence floue (FIS) de type Mamdani. HHO-EAS a été testé d'une part avec HHO, GWO et PSO sur un benchmark général de 19 fonctions bien connues et d'autre part avec HHO sur un benchmark spécifique des 20 fonctions les plus complexes du CEC 2017. Les résultats expérimentaux obtenus sur ces deux benchmarks démontrent la supériorité de HHO-EAS sur HHO pour des problèmes d'optimisation hautement multimodaux et de grande dimension et a ainsi validé notre stratégie d'hybridation entièrement bio-inspirée.

Disposant de la métaheuristique HHO-EAS en mesure de traiter les problèmes d'optimisation NP-Hard, la troisième partie de cette thèse est une application concrète de la métaheuristique HHO-EAS pour concevoir un IDS contre les attaques spécifiques 802.11 au profit des IoT. Nous avons réalisé une

méthode de sélection des dimensions de type Wrapper plus avancée que celle que l'on peut retrouver dans la littérature. Notre méthode est basée sur des algorithmes métaheuristiques, les algorithmes Deep Learning et l'exploitation de la puissance de calcul de la technologie GPU pour le calcul des valeurs de la fonction objectif. Pour ce faire nous avons dans un premier temps hybridé la métaheuristique HHO-EAS pour créer Binary HHO-EAS (BHHO-EAS) adaptée à l'optimisation du problème multi-objectif NP-Hard de sélection des dimensions de type Wrapper dans un espace de recherche binaire. Puis nous avons dans un deuxième temps intégré BHHO-EAS à notre nouvelle méthode de sélection des dimensions Wrapper combinée à l'algorithme Deep Learning Convolutional Neural Network (CNN). Afin d'être cohérent avec l'évolution technologique des systèmes d'information des entreprises et des nouvelles attaques Wi-Fi, nous avons appliqué notre méthode au nouveau dataset AWID3. L'objectif ultime est de concevoir un CNN-IDS, dans un contexte de Green computing, pouvant être embarqué dans un IoT et qui détecte les sept attaques spécifiques au 802.11: Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)Assoc), Rogue Access Point (Rogue AP), Evil Twin, Key Reinstallation Attack (Krack) et Kr00k. Les résultats de nos travaux ont conduit à une sélection de 8 dimensions parmi les 253 du dataset AWID3 et un CNN-IDS avec de très bonnes performances de prédiction des attaques supérieures à 99,70%. Nos travaux ont ainsi prouvé la capacité de notre méthode pilotée par BHHO-EAS à fournir une bonne solution au problème d'optimisation multi-objectif NP-Hard de sélection des dimensions et a contribué au domaine de la cybersécurité des réseaux Wi-Fi d'entreprise en concevant un CNN-IDS compétitif et suffisamment léger pour être intégré dans un IoT. Dans le cadre d'un Proof of Concept (PoC) technique, un prototype de CNN-IDS, nommé E-CNN-IDS, a été embarqué et testé en condition réelle dans un Raspberry Pi 4 Model B. E-CNN-IDS a démontré d'excellentes performances de détection des attaques spécifiques 802.11.

Mots-clés *: Métaheuristique, HHO-EAS, Synergie, Corbeau, Loup, Logique floue, Encerclement, Attaque, Entreprise, Cybersécurité, Wi-Fi, IoT, IDS, Sélection des dimensions Wrapper, Optimisation multi-objectif, CNN, GPU, AWID3.*

Abstract

Faced with the growing problem of cybersecurity of Internet of Things (IoT) within company and state Wi-Fi networks such as the National Gendarmerie, this thesis presents our method for designing an Intrusion Detection System (IDS) against 802.11 specific attacks and embeddable in an IoT with limited computing and memory resources. Our method is based on metaheuristic optimization, feature selection and Deep Learning algorithms.

The feature selection optimization problem is a discrete NP-Hard type problem. In the second part of this thesis we have therefore designed a new metaheuristic, Harris Hawk Optimization Encirclement Attack Synergy (HHO-EAS), able to find a good solution in an acceptable time to this category of optimization problem. HHO-EAS comes from a hybridization strategy of the Harris Hawk Optimization (HHO) metaheuristic with the ambition to remove its weaknesses in solving highly multimodal and high-dimensional optimization problems. Our hybridization strategy is entirely bio-inspired by an unexpected win-win hunting synergy between two predators during extremely difficult winter periods: the crow and the wolf. The exploratory faculties of crows, combined with the ability of wolves to capture prey larger than themselves with speed and efficiency, allow these two predators to detect and catch good prey that is very rare and very difficult to hunt in rigorous winters. In order to mathematically model this win-win hunting synergy with encirclement and attack equations and integrate it into HHO, we used fuzzy logic to create a Mamdani-like fuzzy inference system (FIS). HHO-EAS was tested on the one hand with HHO, GWO and PSO on a general benchmark of 19 well-known functions and on the other hand with HHO on a specific benchmark of the 20 most complex functions of the CEC 2017. The experimental results obtained on these two benchmarks demonstrate the superiority of HHO-EAS over HHO for highly multimodal and high-dimensional optimization problems and thus validated our fully bio-inspired hybridization strategy.

Having the HHO-EAS metaheuristic able to deal with NP-Hard optimization problems, the third part of this thesis is a concrete application of the HHO-EAS metaheuristic to design an IDS against 802.11 specific attacks for the benefit of IoT. We have carried out a more advanced Wrapper feature selection method than which can be found in the literature. Our method is based on metaheuristic, Deep Learning

and the exploitation of the computing power of GPU technology for the calculation of the objective function values. To do this, we first hybridized the HHO-EAS metaheuristic to create Binary HHO-EAS (BHHO-EAS) adapted to the NP-Hard multi-objective optimization problem of feature selection in a binary search space. Then, in a second step, we integrated BHHO-EAS with our new Wrapper feature selection method combined with the Deep Learning Convolutional Neural Network (CNN) algorithm. In order to be consistent with the technological evolution of company information systems and new Wi-Fi attacks, we applied our method to the new AWID3 dataset. The ultimate goal is to design a CNN-IDS, in a Green computing context, that can be embedded in an IoT and that detects the seven 802.11 specific attacks: Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)Assoc), Rogue Access Point (Rogue AP), Evil Twin, Key Reinstallation Attack (Krack) and Kr00k. The results of our work led to a selection of 8 features among the 253 of the AWID3 dataset and a CNN-IDS with very good performance in attack prediction above 99.70%. Our work thus proved the ability of our method, driven by BHHO-EAS, to provide a good solution to the NP-Hard multi-objective optimization problem of feature selection and contributed to the field of cybersecurity in company Wi-Fi networks by designing a competitive CNN-IDS that is light enough to be integrated into an IoT. As part of a technical Proof of Concept (PoC), a CNN-IDS prototype, named E-CNN-IDS, was embedded and tested in real conditions in a Raspberry Pi 4 Model B. E-CNN-IDS demonstrated excellent performance in detecting 802.11 specific attacks.

Keywords: *Metaheuristics, HHO-EAS, Synergy, Crow, Wolf, Fuzzy Logic, Encirclement, Attack, Company, Cybersecurity, Wi-Fi, IoT, IDS, Wrapper feature selection, Multi-objective Optimization, CNN, GPU, AWID3.*

Acknowledgements

First and foremost, I would like to thank my thesis director Rachid Chelouah who believed in me from the start and enabled me to fulfill my dearest professional dream: a thesis in the favorite and most strategic fields of our century, which are Cybersecurity and Artificial Intelligence. During these four years, Rachid has always been available seven days a week, even during his holidays. He was able to guide me, encourage me and give me the necessary impulses during my thesis. I send you a thousand thanks Rachid.

At the beginning of my thesis, I focused on metaheuristic algorithms because I was sure that they represented the keystone of my research. I have therefore studied with great interest several scientific works on metaheuristics. However, only one author was able to bring these algorithms to life, reveal their real potential and inspire me with the strategy to adopt for the design of my metaheuristics: Patrick Siarry, university professor at the University of Paris-Est Créteil. I would never even hope that I would have the great honor of having Patrick Siarry become a member of the thesis follow-up committee and the thesis jury. I would like to express my most sincere thanks to Patrick Siarry.

I would also like to thank Kamel Barkaoui, university professor at the CNAM, for agreeing to be the rapporteur for my thesis.

Finally, I would like to thank my wife and four children for their unwavering support and patience over the past four years. In addition to my extremely time-consuming professional responsibilities, they accepted that I devote all my rest days and holidays to my thesis work. To my wife, I dedicate all of my thesis work to her.

Contents

List of Abbreviations

ABC:	**A**rtificial **B**ee **C**olony
ACO:	**A**nt **C**olony **O**ptimization
AP:	**A**ccess **P**oint
AWID:	**A**egean **W**i-Fi **I**ntrusion **D**ataset
BA:	**B**at **A**lgorithm
BHHO-EAS:	**B**inary **H**arris **H**awk **O**ptimization **E**ncirclement **A**ttack **S**ynergy
BOA:	**B**ase **O**ptimization **A**lgorithm
BSSID:	**B**asic **S**ervice **S**et **ID**edntifier
CNN:	**C**onvolutional **N**eural **N**etwork
CNN-IDS:	**C**onvolutional **N**eural **N**etwork **I**ntrusion **D**etection **S**ystem
DE:	**D**ifferential **E**volution
DOE:	**D**esign **O**f **E**xperiment
E-CNN-IDS:	**E**mbedded **C**onvolutional **N**eural **N**etwork **I**ntrusion **D**etection **S**ystem
EMA:	**E**lectromagnetism **M**ecanism **A**lgorithm
FIS:	**F**uzzy **I**nference **S**ystem
FISREE:	**F**uzzy **I**nference **S**ystem **R**abbit **E**scaping **E**nergy
GA:	**G**enetic **A**lgorithm
GP:	**G**enetic **P**rogramming
GRASP:	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
GSA:	**G**ravitational **S**earch **A**lgorithm
GWO:	**G**rey **W**olf **O**ptimizer
HHO:	**H**arris **H**awk **O**ptimization
HHO-EAS:	**H**arris **H**awk **O**ptimization **E**ncirclement **A**ttack **S**ynergy
HMS:	**H**uman **M**ental **S**earch
ICA:	**I**mperialist **C**ompetitive **A**lgorithm
IDS:	**I**ntrusion **D**etection **S**ystem

IoT:	**I**nternet **o**f **T**hings
IP:	**I**nternet **P**rotocol
IWO:	**I**nvasive **W**eed **O**ptimization
LCA:	**L**eague **C**hampionship **A**lgorithm
MAC:	**M**edia **A**ccess **C**ontrol
MOOP:	**M**ulti-**O**bjective **O**ptimization **P**roblem
OSI:	**O**pen **S**ystems **I**nterconnection
PoC:	**P**roof **o**f **C**oncept
PSO:	**P**article **S**warm **O**ptimization
REE:	**R**abbit **E**scaping **E**nergy
SA:	**S**imulated **A**nnealing
SCA:	**S**ine **C**osine **A**lgorithm
SLC:	**S**occer **L**eague **C**ompetition
SSA:	**S**quirrel **S**earch **A**lgorithm
SSID:	**S**ervice **S**et **ID**entifier
STA:	**Sta**tion
TLBO:	**T**eaching **L**earning **B**ased **O**ptimization
VNS:	**V**ariable **N**eighborhood **S**earch
Wi-Fi:	**Wi**reless **Fi**delity
WWO:	**W**ater **W**ave **O**ptimization

General introduction

Technological progress has enabled companies as well as state administrations to integrate a movement of upward digitization in their work organization methods. IoT plays a key role in this evolution: Smart phones, drones, connected watches, tablets, laptops, connected screens, connected lamps, nano computer, etc. Along with Bluetooth, Wi-Fi (IEEE 802.11 standard) is one of the wireless protocols most used by IoT. However, this protocol is subject to several 802.11 specific cyber-attacks and the wireless specificity of this protocol only facilitates the action of cyberattackers. Moreover, given that in companies' information systems the Wi-Fi wireless network coexists with the wired network, hackers can use it as a gateway to extend their attacks on the latter. Therefore, in companies' information systems, the Wi-Fi network represents an attack surface that can cause very serious consequences for the company. The literature does very little about the consequences of these Wi-Fi cyberattacks on the companies:

- Unavailability of computer services hosted on its network: Web servers, E-mail, data exchange, etc.;
- Theft or destruction of sensitive data: customer files, passwords, medical data, etc.;
- Damage to the image of the company;
- Financial loss;
- Judicial backlash due to illicit access to company computer systems and dissemination of sensitive company data.

To deal with these constantly evolving threats, IoT systems must be able to have a smart cyber-defense system allowing the detection of attacks. These systems are called Intrusion Detection System (IDS). The design of the IDS must exploit a distinct approach from traditional signature-based IDSs in order to satisfy two main objectives:

- Maximize real-time detection performance against known and unknown 802.11 specific attacks;
- Request a minimum of computing and memory resources in order to allow the IDS to be embedded in an IoT.

Mathematically, we must therefore solve a multi-objective optimization problem. But, as this manuscript demonstrates, the complexity of this multi-objective optimization problem belongs to the NP-Hard class. In order to solve this complex optimization problem, we have focused in this thesis on the potential of metaheuristic algorithms. These algorithms are the first important point of this thesis and constitute the keystone of our work. Unlike classical optimization methods, metaheuristics, such as Particle Swarm Optimization (PSO), Differential Evolution (DE) or Harris Hawk Optimization (HHO) consider the optimization problem as a black box whatever its internal mathematical specificities. Furthermore, they make it possible to obtain in an acceptable time a good solution to NP-Hard optimization problems. However, metaheuristics are likely to show weaknesses when faced with the growth of the dimension and the multimodality of the optimization problem. This particularly concerns feature selection problems that fall victim to the curse of dimensionality. This problem is one of the components of the multi-objective optimization problem of this thesis.

In order to defend ourselves against cyber threats targeting IoT systems exploiting Wi-Fi within the companies' information systems and state administration information systems, the work presented in this thesis manuscript provides contributions in the following three fields: metaheuristic algorithms, feature selection algorithms and cybersecurity.

In the field of metaheuristic algorithms, we designed two new metaheuristics: HHO-EAS (Harris Hawk Optimization – Encirclement Attack Synergy) [Sassi M et al., 2023] and BHHO-EAS (Binary HHO-EAS).

With the ambition of meeting the cybersecurity challenges of this thesis, the new metaheuristic HHO-EAS was designed via a new hybridization strategy entirely bio-inspired by the hunting synergy between crows and wolves during the harsh winter periods. We will see on two benchmarks that HHO-EAS, compared to three well-known metaheuristics including PSO, GWO and HHO, demonstrates very good performances on high-dimensional (greater than 100) and highly multimodal optimization problems. The promising results obtained at this stage of our works made it possible to validate the use of HHO-EAS for the feature selection problems and the design of a CNN-IDS (Convolutional Neural Network Intrusion Detection System) against the 802.11 specific attacks.

The feature selection optimization problems of high-dimensional datasets are of NP-Hard type within a discrete binary search space. In order to allow HHO-EAS to solve this category of problem, we hybridized it while keeping its algorithmic and mathematical strategy to create BHHO-EAS.

In the field of feature selection algorithms and cybersecurity, to design our CNN-IDS, we improved the Wrapper feature selection system by integrating the ability to exploit all metaheuristics, all Deep Learning algorithms, all kind of dataset as well as GPU technology. In the applicative part of this manuscript which is the second important point of this thesis, our system thus allowed us on one hand, to benefit from the superior predictive skills of Deep Learning algorithms instead of traditional Machine Learning algorithms commonly observed in the literature for the feature selection. And on the other hand, during the feature selection process driven by the metaheuristic BHHO-EAS, the latter is able to exploit the computing power of the GPUs when calculating the objective function values for each agent of BHHO-EAS and of the CPUs for updating their positions in the binary search space.

In a supervised learning context, we have focused our attention on the Deep Learning Convolutional Neural Network (CNN) algorithms for their excellent attack prediction skills and their inherent capabilities in their architecture to be embedded in an IoT.

Concerning the dataset, thanks to the excellent work of the researchers from the Greek Aegean University, the new AWID3 dataset (Aegean Wi-Fi Intrusion Dataset 3) was created in 2021. Much more complex, adapted to the technical specificities of companies' information systems and richer in Wi-Fi attacks than its predecessor AWID2, the AWID3 dataset makes it possible to create IDSs against very recent 802.11 specific attacks targeting companies' information systems. Furthermore, at the time of writing this manuscript, we are the first to applied a metaheuristic optimization method to the feature selection of AWID3 dataset. Indeed, in the literature, no feature selection via metaheuristic optimization methods were used on the AWID3 dataset.

The results of our feature selection method and design of a CNN-IDS against 802.11 specific attacks were compared to the work of Chatzoglou E. et al. in [Chatzoglou E et al., 2022a] recognized by the community of researchers and which are the only ones to be in the same dominants as ours. For a fair comparison between the 4 best IDS of [Chatzoglou E et al., 2022a] and our CNN-IDS, the performance measures are based on *AUC* and *F1* metrics as well as the *Confusion matrix* and the number of features

selected from the AWID3 dataset. The experimental results demonstrated that our CNN-IDS provides extremely competitive results compared to the 4 best IDS of [Chatzoglou E et al., 2022a]. And in order to prove the ability of our CNN-IDS to be embedded in an IoT environment with limited computing and memory resources, we carried out a technical Proof of Concept (PoC) on a Raspberry Pi 4.

In line with our objectives set out above, our thesis manuscript is organized into three chapters. First of all, in order to understand the mathematical challenges of optimization problems and the advantages of metaheuristics in solving these problems, we provided in the first chapter a general presentation of mono-objective and multi-objective optimization problems as well as metaheuristics algorithms.

In the second chapter we present our design strategy of the new metaheuristic HHO-EAS for solving high-dimensional and highly multimodal optimization problems and its performances on general and specific benchmarks. Our design strategy is based on a hybridization entirely bio-inspired by the hunting synergy between crows and wolves.

The third chapter is the application of HHO-EAS to IoT cybersecurity within companies' information systems. We explain our hybridization method of HHO-EAS to design BHHO-EAS, thus allowing it to solve the NP-Hard multi-objective optimization problems of feature selection. Thus, BHHO-EAS drives our Wrapper feature selection method and selects the most relevant AWID3 feature subset that maximizes the prediction performance of CNN-IDS while minimizing its complexity in order to be integrated into an IoT with limited computing and memory resources. We will conclude this thesis by recalling our contributions and detailing the promising perspectives for our future work.

The second and third chapters of our thesis have both been the subject of an article submitted to the journal Artificial Intelligence Review. The first article has been validated and accepted for publication [Sassi M et al., 2023]. We have also written Chapter 3 of the book *Optimization and Machine Learning: Optimization for Machine Learning and Machine Learning for Optimization* [Sassi M, 2022] which describes the synergy between metaheuristics and the feature selection process for the design of an IDS.

Chapter 1

Generalities about optimization and metaheuristics

1.1. Introduction

Optimization problems are encountered on a daily basis regardless of the sector of activity. A business manager wants his schedule for the week to take into account all the assignments of the week while taking into account the constraints imposed by his employees and minimizing operating costs. The traveler salesman will want to deliver to all their customers as quickly as possible by choosing the shortest route and consuming as little fuel as possible. Finally, electronics engineers, forced by the inflation of metal prices, seek to create electronic boards using as little metal as possible while connecting all the components of the electronic board.

In order to satisfy these needs, engineers implement optimization methods. Metaheuristics are among the most powerful and exploited optimization methods to solve optimization problems categorized as NP-Hard. This class of problems are insurmountable for other "classical" optimization methods. In this chapter, after mathematically defining optimization and the computational complexity theory in the sections 1.2 and 1.3, we will provide a general description of mono-objective optimization in section 1.4. Since real-world optimization problems must satisfy multiple objectives, which is the case of the subject of this thesis, we will present in section 1.5 a general description of the multi-objective optimization problems (MOOPs). Then we will provide in section 1.6 a general review of the keystone algorithms of this thesis: metaheuristics.

1.2. Definition of optimization

Optimization is defined as "the search for the minimum or the maximum of a given function" [Siarry P et al., 2002a]. It is therefore a question of finding a data vector representing candidate decision variables which minimizes or maximizes the objective function(s). The objective function(s) mathematically model the optimization problem to be solved. In addition, optimization problems are very often limited by equality or inequality constraints that must be taken into account in the decision variables of the problem. We thus obtain a constrained optimization problem for which the decision variables fluctuate in a restricted search space. And depending on the possibilities of assigning the values of the decision

variables, whether they are limited by discrete values in $\mathbb{Z}_+$ or unlimited by continuous values in $\mathbb{R}$, the problem is respectively categorized as a combinatorial, integer variable or continuous [Singh P et al., 2021].

This general definition applies to both categories of optimization problem depending on whether the problem must satisfy one or more objective functions. If there is only one objective function, the optimization is said to be mono-objective. On the other hand, if there are several objective functions, the optimization will be multi-objective. But before addressing these two categories of problem it is necessary to appreciate their level of complexity.

1.3. Computational complexity theory of optimization problems

In chapter 3 of this thesis we analyze and deal with feature selection problem whose computational complexity is categorized as *NP-Hard*. In order to measure the complexity of this category of problem it is essential above all to define the fundamental notions of the theory of the complexities of optimization problems.

We distinguish two main classes of problem: P and NP [Du D et al., 2014; Izadkhah H, 2022; Rabbouch B et al., 2023]. These two classes discriminate the problems according to whether it is possible or not to solve them by a deterministic polynomial algorithm. More precisely does there exist a polynomial of degree m which dominates the time of resolution of the problem or not?

A problem belongs to class P (Polynomial) if there is a deterministic algorithm that can solve this problem exactly in polynomial time. An example of a well-known class P problem is that of Linear Search. As a consequence of the above, a candidate solution to this problem of class P can be verified in polynomial time is therefore P a subset of the problem class NP ($P \subset NP$).

A problem belongs to the class NP (Non-deterministic Polynomial) if it can be solved by a non-deterministic polynomial algorithm and we can verify a candidate solution to this NP problem in polynomial time. A classic example of an NP problem is the Traveling Salesman Problem TSP.

Among the NP class problems we can find the most complex problems: *NP-Complete* (NPC) and *NP-Hard*. NP-Hard problems find several definitions in the literature that can overlap globally. The one

that is the most precise and the most adequate to this thesis is the one that follows [Siarry P, 2014a]. We distinguish the cases where the optimization problem is discrete or continuous:

- A continuous optimization problem is class NP-*Hard* if we don't know algorithm to find the exact solution with certainty in a limited computation time. This case is encountered for high-dimensional and highly multimodal engineering optimization problems.
- A discrete optimization problem of dimension D is NP-*Hard* if there is no integer m allowing to dominate the solving time by D^m. This is the case for the feature selection problem whose resolution time is an exponential function of the dimension of the problem.

As for problems the NPC problem class they are both NP and NP-*Hard*.

We are now able to appreciate the complexity of an optimization problem, especially if it is NP-*Hard*, whether it is mono-objective or multi-objective.

1.4. Mono-objective optimization

1.4.1. Mathematical expression of a mono-objective problem

As we have just stated above, optimization in a d-dimensional search space consists of finding a subset of optimal decision variables in order to minimize or maximize an objective function that mathematically models the optimization problem [Cuevas E et al., 2021b].

A mono-objective optimization problem under constraints is mathematically expressed as follows in (1.1) [Houssein EH et al., 2021b; Khanduja N et al., 2021; Singh P et al., 2021]:

- One mono-objective function f to minimize (or to maximize) in a d-dimensional search space;
- A vector $\vec{g}$ of p inequality constraints g_k;
- A vector $\vec{h}$ of q equality constraints h_j.

$$\begin{cases} Minimize\ (or\ Maximize)\colon f(x) \\ Inequality\ constraints\colon g_k(x) \leq 0\ , k \in \{1,..,p\} \\ Equality\ constraints\colon h_j(x) = 0\ , j \in \{1,..,q\} \\ x = (x_1, \dots, x_i, \dots., x_d), i \in \{1,..,d\} \\ x_i \in [l_i; u_i]\ , x \in S^d \\ \vec{g}(x) = \left(g_1, \dots, g_k, \dots., g_p\right) \\ \vec{h}(x) = \left(h_1, \dots, h_j, \dots., h_q\right) \end{cases} \quad (1.1)$$

l_i and u_i represent respectively the lower and upper bounds of the search space of the decision variable x_i. l_i and u_i are also the components of the vectors $\boldsymbol{L}$ and $\boldsymbol{U}$ of dimension d. $\boldsymbol{L}$ and $\boldsymbol{U}$ thus create a hyper rectangular domain S^d. Furthermore, to be more comprehensive, the space where the search space and all the constraints are taken into account is the realizable value space $\aleph^d$.

The optimization problem can be split into two categories depending on the existence of local optima in the objective function f. An optimization problem with a unique global optimum is said to be unimodal. While an optimization problem with one or more local optima and one or more global optima is said to be multimodal.

1.4.2. Definition of global and local optimum

We distinguish three types of optimum: global optimum, strict local optimum and weak local optimum [Siarry P et al., 2002a].

- A global optimum x_g is a global minimum (or a global maximum) of the objective function f if

  $\forall\ x \in S^d, x \neq x_g \mid f(x_g) \leq f(x)$ (or $f(x_g) \geq f(x)$ if maximum)

- Similarly, a local optimum x_w is a weak local minimum (or a weak local maximum) of the objective function f if there exists a neighborhood $N(x_w)$ of x_w defined by: $x \in N(x_w), \exists\ \varepsilon > 0 \mid \|x - x_w\| < \varepsilon$.

  Thus $\forall\ x \in N(x_w), x \neq x_w \mid f(x_w) \leq f(x)$ (or $f(x_w) \geq f(x)$ if local maximum).

- And a local optimum x_s is a strict local minimum (or a strict local maximum) of the objective function f if there exists a neighborhood $N(x_s)$ of x_s such that

  $\forall\ x \in N(x_s), x \neq x_s \mid f(x_s) < f(x)$ (or $f(x_s) > f(x)$ if strict local maximum)

We will exploit these mathematical definitions in what follows by a simple example of optimization problem.

1.4.3. Illustration of global and local optimums in a real case of optimization problem

In order to illustrate the global and local minima on a concrete case of an optimization problem, let us take the example of the function $f(x) = \sum_{j=0}^{20} a_j . x^j$. f is a polynomial whose coefficients a_j are created using the least squares method of [Krueger M, 1990]. Moreover, $\forall x \in [0,1], f(x) \in [0,1]$. The curve of function f is illustrated in Fig. 1.1.

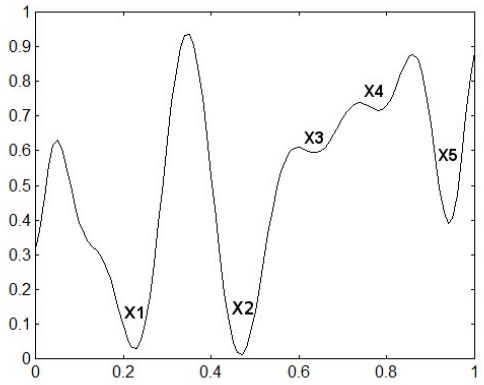


Fig. 1.1. Polynomial function f with one variable [Chelouah R, 2000]

The particularity of the function f is that it has in $[0,1]$, five diversified local minima.

- A global minimum $X2$ equal to 0.4666 with $f(X2) = 0.01048$;
- A strict local minimum $X1$ with $f(X1)$ almost equal to $f(X2)$ and positioned in a wider valley ;
- Three strict local minimums $X3$, $X4$ et $X5$ with values $f(X3), f(X4)$ and $f(X5)$ far from equaling $f(X2)$ and positioned in wide valleys for $X3$, $X4$ and a narrow valley for $X5$;
- No weak local minimum.

To solve mono-objective optimization problems, mathematics offers a wide range of possibilities.

1.4.4. Taxonomy of mono-objective optimization methods

The range of mathematical methods for solving mono-objective problems is substantial. As we explained in the definition of optimization above, these methods can be divided into two categories: combinatorial or discrete methods and continuous methods. Combinatorial methods exploit a

permutation of finite set of numbers as decision variables. The *Traveling Salesman* problem is a classic example. And discrete methods exploit discrete values as decision variables as in the *Binary feature selection* problem. It includes exact and approximate methods. Exact methods such as *Branch and Bound* [Hillier FS et al., 1995] or *Simplex method* [Nash JC, 2000] make it possible to obtain the exact global optimum of certain "simple" optimization problems with a search space that is not very complex and of small and medium dimension. On the contrary, approximate methods such as *Heuristics* and *Metaheuristics* [Ahmed F et al., 2021; Houssein EH et al., 2021b]are used for so-called "difficult" optimization problems.

Continuous methods as their name suggests exploit continuous decision variables. They make it possible to deal with linear and nonlinear optimization problems. Linear optimization problems are solved by linear programming methods. And nonlinear optimization problems also include so-called "difficult" problems that can be solved, according to their mathematical specificities, either with local methods such as *Gradient based* ones or global methods such as *Metaheuristics*.

Fig. 1.2 provides the taxonomy of mono-objective optimization methods that we have just exposed.

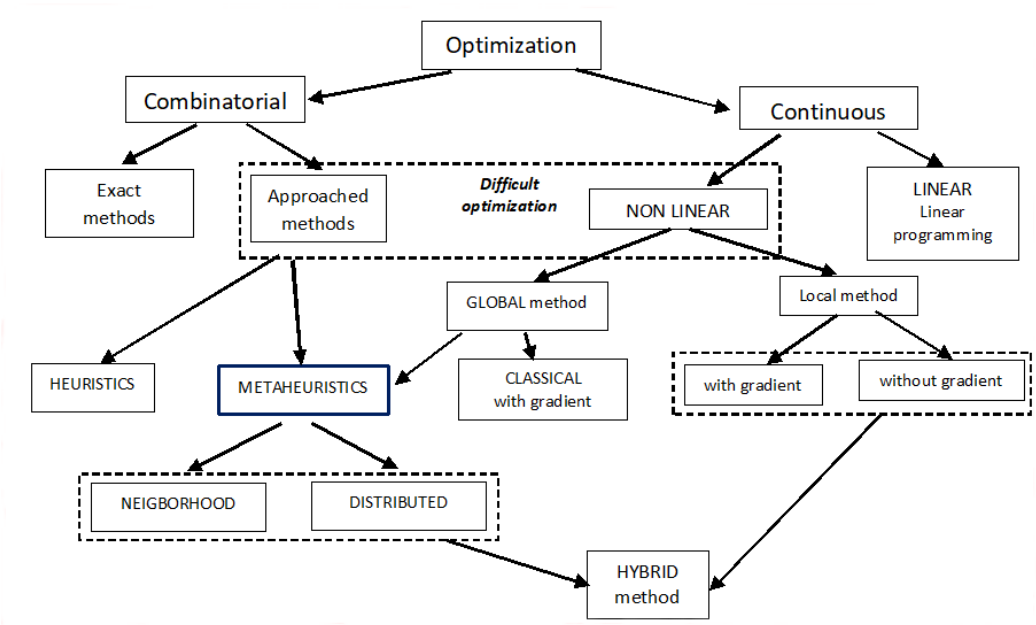


Fig. 1.2. Taxonomy of optimization methods for mono-objective problems [Chelouah R, 2000; Siarry P et al., 2002a]

We now have an exhaustive view of mono-objective optimization problems and their methods for solving them. However, real-world optimization problems are not satisfied with only a single objective to be satisfied but with a multitude of contradictory objectives. These are called multi-objective problems.

1.5. Multi-objective Optimization

Real-world engineering optimization problems typically require consideration of multiple adversarial optimization sub problems and just like single-objective problems, this requires taking into account several decision variables. For example, a carpenter who has to build a plastic board with the least possible deformation and the smallest possible section. These optimization problems are said to be multi-objective [Alkebsi K et al., 2020; Houssein EH et al., 2021b; Abdollahzadeh B et al., 2022; Toloo M et al., 2022a]. This is also the case of the optimization problem which constitutes the mathematical modeling of the objectives of our thesis: satisfying both strategic objectives of maximizing the cybersecurity performance of an IDS and integrating it into an IoT with limited computing and memory resources. In order to better understand this vital contemporary problem, it is essential to have the theoretical fundamentals of multi-objective optimization.

1.5.1. Mathematical expression of a multi-objective problem

The mathematical expression of a multi-objective optimization problem (MOOP) is similar in form to that of a single-objective optimization problem. The fundamental difference is that we no longer have to deal with single objective but with a vector of functions composed of several objective functions.

A multi-objective optimization problem under constraints in real life is mathematically expressed as follows in (1.2) [Al-Tashi Q et al., 2020; Rodríguez MA et al., 2020; Houssein EH et al., 2021b; Khanduja N et al., 2021; Panagant N et al., 2021; Yu X et al., 2022]:

- A vector $\vec{f}$ of n objective function f_i to minimize or to maximize in a d-dimensional search space, with $n>1$;
- A vector $\vec{g}$ of p inequality constraints g_k;
- A vector $\vec{h}$ of q equality constraints h_j.

$$\begin{cases} Minimize\ (or\ Maximize): \vec{f}(x) \\ Inequality\ constraints: \vec{g}(x) \leq 0 \\ Equality\ constraints: \vec{h}(x) = 0 \\ x = (x_1, \dots, x_i, \dots, x_d), i \in \{1, .., d\} \\ x_i \in [l_i; u_i], x \in S^d \\ \vec{f}(x) = (f_1, \dots, f_l, \dots, f_n), l \in \{1, .., n\} \\ \vec{g}(x) = (g_1, \dots, g_k, \dots, g_p), k \in \{1, .., p\} \\ \vec{h}(x) = (h_1, \dots, h_j, \dots, h_p), j \in \{1, .., q\} \end{cases} \quad (1.2)$$

As in mono-objective optimization, l_i and u_i represent respectively the lower and upper bounds of the research space of the decision variable x_i. And S^d, $\aleph^d$.and $\mathcal{F}^d$ are respectively the search space, the realizable value space and the objective function space. These three spaces are illustrated in Fig. 1.3 for *d*=2. Fig. 1.3 illustrates the evolution of the search space S^2 to $\aleph^2$ when we apply the constraints of the optimization problem then to the objective function space $\mathcal{F}^2$ when we apply the two objective functions (f_1, f_2). We also provide, via the color codes *green triangle*, *yellow square*, *red circle* and *blue star*, the Pareto fronts of the four types of bi-objective optimization problem *Min-Min*, *Min-Max*, *Max-Max* and *Max-Min*.

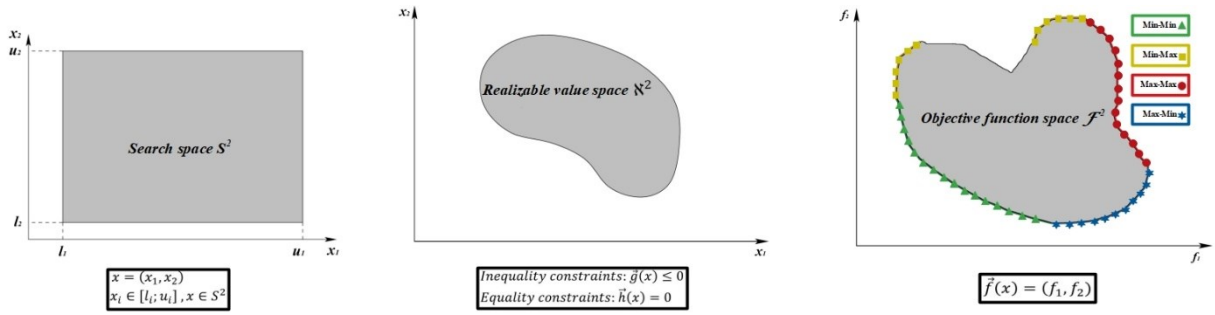


Fig. 1.3. Search space, realizable value space and objective function space for *d*=2 and *n*=2

Compared to mono-objective optimization problems, multi-objective problems are distinguished by several mathematical particularities.

1.5.2. Particularities of multi-objective optimization problems

The first particularity that we have already mentioned above is that multi-objective optimization problems require minimizing or maximizing *n* objective functions which are generally contradictory, while taking into consideration constrained function vectors. This concretely means that a decision vector

x which decreases the value of one of the component objective functions will increase the value of another.

The second feature stems from the first. Given the contradictory nature of objective functions, solving a multi-objective problem generates several non-optimal solutions. Indeed, each of these solutions represents a compromise between the different objectives of the problem. So, they cannot minimize or maximize all the *n* objectives. On the contrary, some of the objectives will be devalued in favor of others which will come closer to the expected results. Therefore, we can notice that the notion of good solution is not as obvious as for mono-objective problems.

The third particularity is linked to the vector character of the objective function $\vec{f}$. This creates an *n*-dimensional objective function space. Fig. 1.3 illustrates this space for $d = 2$ and $n = 2$.

The fourth and last particularity is the introduction of a decision maker in the strategy of solving a multi-objective problem. We will detail the role of this decision maker in what follows.

1.5.3. Role of the decision maker in multi-objective optimization

The decision maker is a crucial actor in the process of multi-objective optimization. If the decision maker has a good strategy, he can come up with very competitive solutions. There are three main approaches for the decision maker [Siarry P et al., 2002a; Donoso Y et al., 2016; Toloo M et al., 2022b] : A priori, A posteriori and Interactive.

A. A priori

In the A priori case, the objective functions do not have the same importance for the decision maker. Thus he makes arbitration choices on the n objectives to be achieved before the start of the optimization in order to define the tradeoff between the n objectives. This is the case, for example in the weighted sum method (or scalar method). The decision maker sets a weight for each objective function according to their importance and aggregates them. We therefore move from a multi-objective problem to a mono-objective problem. For example, for a bi-objective optimization problem (f_1, f_2) the decision maker will assign them the weights (w_1, w_2) and aggregate them to obtain the unique objective function $f = w_1. f_1 + w_2. f_2$.

B. A posteriori

Contrary to the A priori approach, in the A posteriori approach the decision maker has no appreciation on the objective functions, thus he gives them the same importance. We therefore remain in a multi-objective context until the end of the optimization process. This approach can be applied with a metaheuristic algorithm such as Multi-Objective Particle Swarm Optimization (MOPSO) [Bouchaala A et al., 2022]. However, the complexity and computation time of this approach is much greater than the A priori approach. At the end of the optimization process, the decision maker must choice the best solution in line with his target. It is therefore necessary for him to have a method of ranking Pareto solutions. Taguchi's quadratic loss function, based on socio-economic criteria, allows this classification [STeele S et al., 1988; Baron C et al., 2005; Chelouah R et al., 2007; Chelouah R et al., 2009]. Taguchi's quadratic loss function consists of a weighted sum of the costs of the decision variables. This Loss function considers that deviations from the target is a loss which increases quadratically with respect to the target [Yu FJ et al., 2009]. For the case of a 2-dimensional search space, the Taguchi's quadratic loss function is represented by the 2 equations (1.3) and (1.4) below [Chelouah R et al., 2009]:

$$C = O + \Delta = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} o_1 \\ o_2 \end{pmatrix} + \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix} = \begin{pmatrix} o_1 \\ o_2 \end{pmatrix} + \begin{pmatrix} k_1(e_1 - x_1)^2 \\ k_2(e_2 - x_2)^2 \end{pmatrix} \quad (1.3)$$

$$G = \sum_{i=1}^{2} c_i = \sum_{i=1}^{2} o_i + \sum_{i=1}^{2} \delta_i = \sum_{i=1}^{2} o_i + \sum_{i=1}^{2} k_i(e_i - x_i)^2 \quad (1.4)$$

- C is the cost of the solution (x_1, x_2) belonging to the Pareto front;
- O is the vector of optimal costs;
- Δ is the vector of the additional costs;
- G is the total cost of the solution (x_1, x_2) ;
- (e_1, e_2) is the target desired by the decision maker;
- k_1 and k_2 are constants called Taguchi's quality loss coefficients which represent respectively the weights of the cost $(e_1 - x_1)^2$ and $(e_2 - x_2)^2$.

C. Interactive

In this approach, the decision maker is present throughout the optimization process by deciding on the tradeoff to carried out on the n function objectives. This is the case with Fandel's method, which guides the choice of objective weights [Eschenauer H et al., 1990].

In order to have a synthetic knowledge of the methods of multi-objective optimization **A priori**, **A posteriori** and **Interactive**, we propose the taxonomy below represented by Fig. 1.4 [Yassa S, 2014]. We provide in gray the optimization methods that we used in the research work of our thesis.

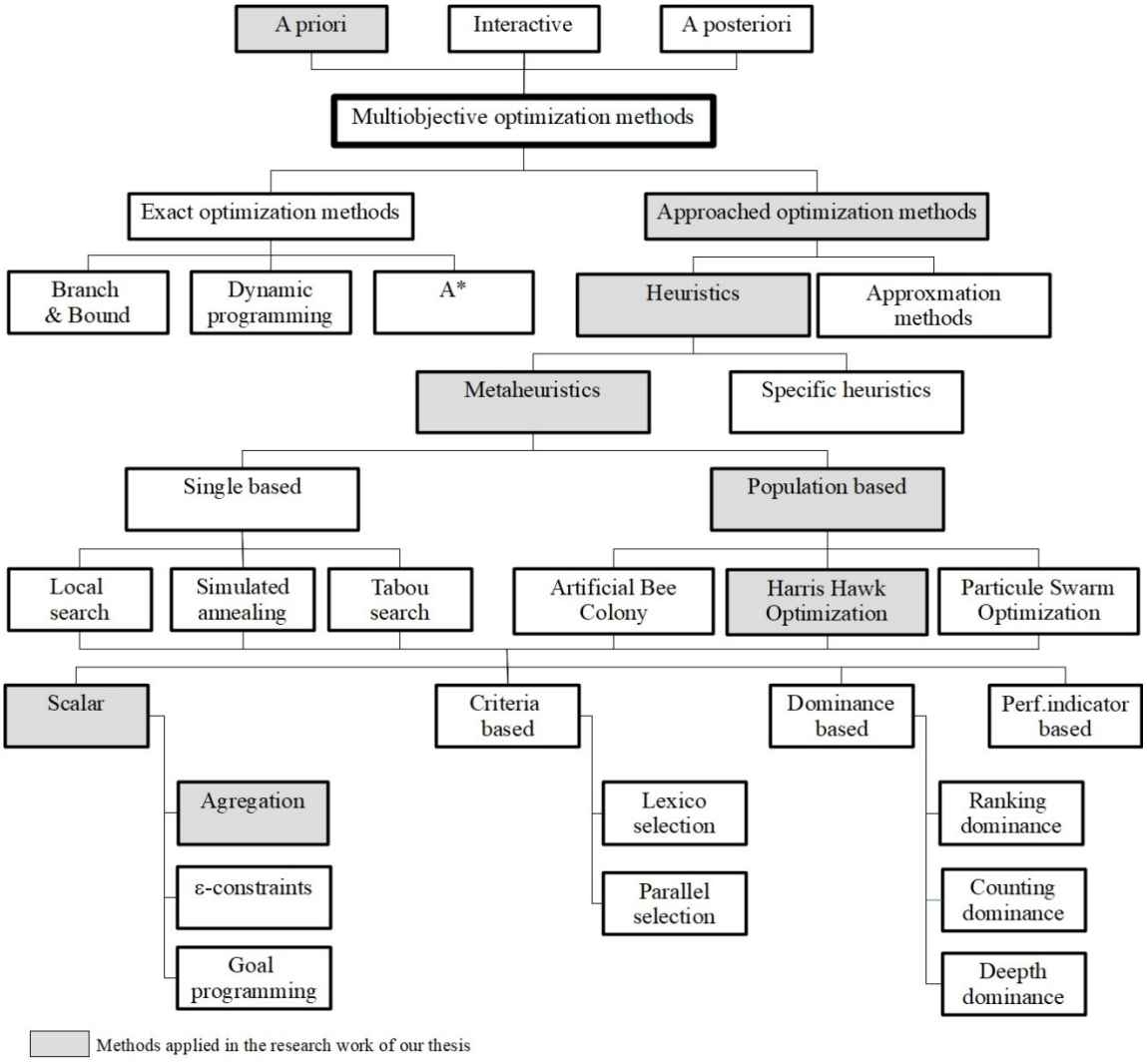


Fig. 1.4. Taxonomy of multi-objective optimization methods [Yassa S, 2014]

At the end of the optimization process, the **A priori** and **Interactive** approach therefore make it possible to obtain a single solution whose quality will translate the strategy of the decision maker. While the **A posteriori** approach provides several solutions. However, only solutions having a dominance relationship on others, in the Pareto optimal front (or very close to it) must be taken into account. In order to identify, in a context of multi-objective optimization, the concepts of Pareto global optimum, Pareto local optimum and Pareto front it is necessary to define these mathematical concepts.

1.5.4. Definition of dominance in a context of multi-objective optimization

We will first define the Pareto Dominance between two decision-making vectors, the Pareto front then the Pareto local and global optimum [Siarry P et al., 2002a; Al-Tashi Q et al., 2020; Rodríguez MA et al., 2020; Sharma S et al., 2022].

D. Pareto dominance

The mathematical concept of Pareto dominance was first defined in 1881 by the Irish economist and lawyer Edgeworth FY for the resolution of the economic problems of balance between taxation and purchasing power of each individual [Chiandussi G et al., 2012]. Then in 1896 the Italian engineer and economist Pareto V theorized it in order to extend multi-objective optimization problems to other scientific dominant [Emmerich MTM et al., 2018]. This is why the mathematical term *Edgeworth-Pareto dominance* is mentioned in some literature for multi-objective optimization problems.

Pareto V issued the following postulate in 1896: "*There is a balance such that one cannot improve one criterion without deteriorating at least one of the other criteria*." From this postulate we can conclude that there is no single solution as in mono-objective optimization problems but several solutions. The concept of Pareto dominance thus makes it possible to create an ordering relation between two candidate solutions to be evaluated against each other in multi-objective context.

Either two solutions $(S_\alpha, S_\beta) \in (\aleph^d)^2$, the solution S_α dominates the solution S_β, mathematically noted as $S_\alpha \prec S_\beta$, if:

- S_α has at least the same performance on the n objective functions than S_β,
- S_α has performance strictly superior than S_β on at least one of the n objective functions.

Mathematically Pareto dominance is expressed by (1.5):

$$\begin{cases} \vec{f}(S_\alpha) = \left(f_1(S_\alpha), \dots, f_l(S_\alpha), \dots., f_n(S_\alpha)\right) \\ \vec{f}(S_\beta) = \left(f_1(S_\beta), \dots, f_l(S_\beta), \dots., f_n(S_\beta)\right) \\ S_\alpha \prec S_\beta \; and \; \vec{f}(S_\alpha) \leq \vec{f}(S_\beta) \; if \; and \; only \; if \; \forall l \in \{1, .., n\} \; f_l(S_\alpha) \leq f_l(S_\beta) \\ and \; \exists \, u \in \{1, .., n\} \; f_u(S_\alpha) < f_u(S_\beta) \end{cases} \quad (1.5)$$

Like mono-objective problems, the notion of Pareto dominance introduces a Pareto global and Pareto local optimality and a weak and a strict Pareto dominance.

a. Pareto global and local optimality

The definition of a Pareto global and a Pareto local optimum in a multi-objective context is quite similar to that in a single-objective context:

- A vector $S_\alpha \in \aleph^d$ is a Pareto global optimum if $\forall\, S_\beta \in \aleph^d,\ S_\alpha \prec S_\beta$;
- A vector $S_\alpha \in \aleph^d$ is a Pareto local optimum local if there exists a real $\varepsilon > 0$ such that $\forall\, S_\beta \in \aleph^d \cap B(S_\alpha, \varepsilon), S_\alpha \prec S_\beta$, knowing that $B(S_\alpha, \varepsilon)$ is a hyperball with center S_α and radius ε.

Local optimality is therefore limited to a restricted area B dimensioned by ε around S_α, while global optimality concerns the entire space $\aleph^d$. But optimality can be weak or strict.

b. Weak and a strict dominance in the sense of Pareto

Thus for two solutions $(S_\alpha, S_\beta) \in (\aleph^d)^2$, the solution S_α weakly dominates the solution S_β, mathematically noted as $S_\beta \preccurlyeq S_\alpha$, if and only if $\forall l \in \{1,..,n\}\ f_l(S_\alpha) \leq f_l(S_\beta)$.

And the solution S_α strongly dominates the solution S_β, mathematically noted as $S_\alpha \prec S_\beta$, if and only if $\forall l \in \{1,..,n\}\ f_l(S_\alpha) < f_l(S_\beta)$.

Finally, using the mathematical definition of Pareto dominance, we can say that a vector $S_\alpha \in \aleph^d$ is non-dominated if $\forall\, S_\beta \in \aleph^d\ S_\beta \nprec S_\alpha$.

The mathematical definitions that we have stated above make it possible to determine the quality of a solution in $\aleph^d$ and the best solutions are positioned on the Pareto front.

E. Pareto front

The Pareto front is constituted by the subset Π^* of Pareto optimal solutions or Pareto optimal set. The solutions in Π^* dominate the other solutions but do not dominate each other. This means that if $S_\alpha \in \Pi^*$ there is no solution $S_\beta \in \aleph^d$ that dominates S_α. The Pareto front thus constitutes the ideal compromise on all the objectives of a multi-objective optimization problem.

The Pareto front Πf^* is the image of Π^* in the objective function space. Π^* is defined by (1.6) and Πf^* is defined by (1.7). For example, Fig. 1.3 illustrates the Pareto front with green triangles for a *Min-Min* bi-objective optimization problem of the two objective functions $f_1$and f_2.

$$\Pi^* = \{S_\alpha \in \aleph^d | \nexists S_\beta \in \aleph^d, S_\beta \prec S_\alpha\} \quad (1.6)$$

$$\Pi f^* = \{\vec{f}(S) | S \in \Pi^*\} \quad (1.7)$$

As can be visually guessed from Fig. 1.3, there are several fronts starting from the Pareto optimal front which constitute the Pareto ranks.

F. Pareto ranks

The Pareto ranks allow to sort subsets of solutions, that do not dominate each other, according to their dominance relation on other subset of solutions. This technique is used in some multi-objective optimization algorithms such as the NSGA-II metaheuristic [Deb K et al., 2002].

We will call $S\wp$ the set of solutions and Sk the subset of solutions of rank k. So, for n subsets and therefore n ranks, $S\wp = \bigcup_1^n Sk$.

The creation of ranks is performed as follows. Rank 1 is made up of the Pareto optimal solutions $S1$ which dominates all the other solutions. To create the second rank 2 with $S2$, we remove the solutions $S1$ from $S\wp$ and we recalculate the Pareto optimal solutions. We proceed in this way until there are no more solutions. At the end, all the solution subsets Sk will have been integrated at a rank k. Fig. 1.5 illustrates the result of this process for a set of 15 solutions which are distributed over 4 ranks.

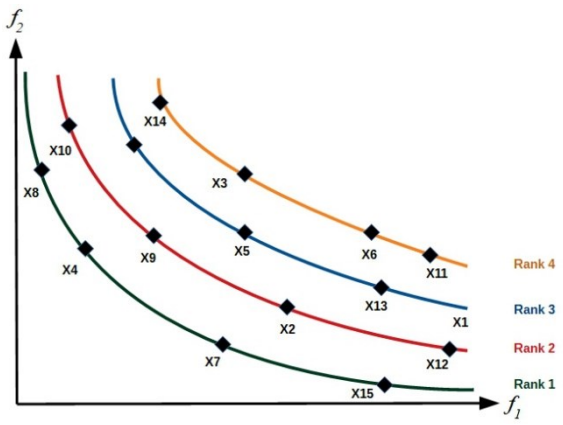


Fig. 1 5. Sorting of the 15 solutions on 4 Pareto dominance rank

To illustrate the concept of Ranking Pareto dominance, we will use the example of Siarry P et al. [Siarry P et al., 2002b] for a bi-objective optimization problem: maximize f_1 and minimize f_2.

For this problem we obtain 5 solutions detailed in Table 1.1.

Solutions	Fonction objectif f_1	Fonction objectif f_2
A	8	5
B	9	2
C	12	1
D	11	2
E	16	2

Table 1.1. The 5 solutions to the bi-objective problem f_1 and f_2 [Siarry P et al., 2002b]

Graphically the 5 solutions are distributed in objective function space (f_1, f_2) according to Fig. 1.6.

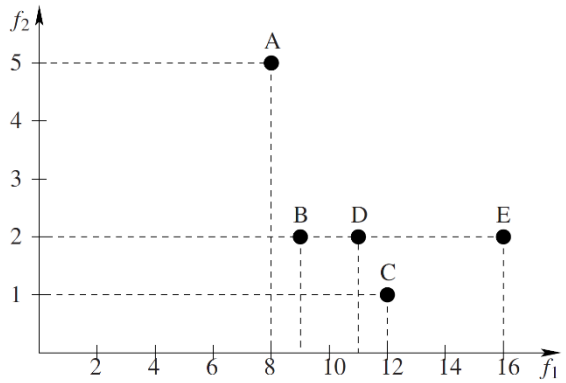


Fig. 1.6. Representation of the 5 solutions in the objective function space (f_1, f_2) [Siarry P et al., 2002b]

In order to sort the 5 solutions in accordance with the Pareto dominance criteria, we will use, like Siarry P et al., the symbols $+$, $-$ and $=$ if respectively a solution is better, less good or equivalent than another on the 2 objective functions f_1 et f_2. At the end we obtain the result in Table 1.2.

	A	B	C	D	E
A		(-,-)	(-,-)	(-,-)	(-,-)
B	(+,+)		(-,-)	(-,=)	(-,=)
C	(+,+)	(+,+)		(+,+)	(-,+),
D	(+,+)	(+,=)	(-,-)		(-,=)
E	(+,+)	(+,=)	(+,-)	(+,=)	

Table 1.2. Evaluation of the 5 solutions A, B, C, D et E

Based on the results obtained in Table 1.2, it appears that solutions C and E dominate solutions A, B and D but they do not dominate each other. C and E therefore constitute the Pareto optimal solutions and will belong to the rank 1. To determine rank 2, solutions C and E are removed from the set of 5 solutions. We obtain Table 1.3 made up of solutions A, B and D. We repeat the same processing as in Table 1.2. Thus, solution D dominates solutions A and B and constitutes the Pareto optimal solution for the rank 2.

	A	B	D
A		(-,-)	(-,-)
B	(+,+)		(-,=)
D	(+,+)	(+,=)	

Table 1.3. Evaluation of the 3 solutions A, B and D

We remove D from Table 1.3 and we get Table 1.4.

	A	B
A		(-,-)
B	(+,+)	

Table 1.4. Evaluation of the 2 solutions A and B

In Table 1.4, B dominates A. Consequently, B will belong to the rank 3 and A to the rank 4. This last iteration finalizes the sorting of the 5 solutions in ranks 1, 2, 3 and 4 as illustrated in Fig. 1.7.

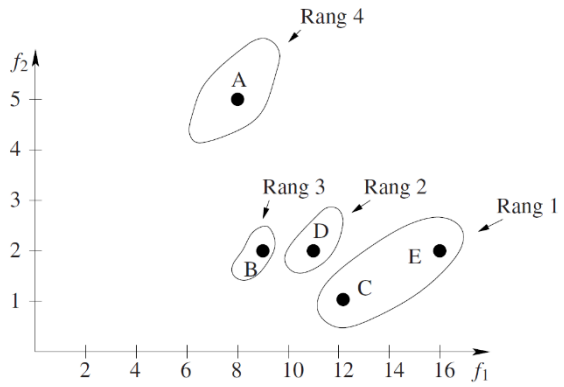


Fig. 1.7. Pareto rank of the 5 solutions [Siarry P et al., 2002b]

This section has provided the general concepts of multi-objective optimization. We have seen that MOOPs, which represent real-world engineering problems, are much more complex mathematically. And as we have already specified above, the mathematical modeling of the optimization problem of the aims of this thesis is a MOOP. But we will also see in chapter 3 that this optimization problem is discrete, non-linear and NP-Hard. However, metaheuristics, faced with the curse of the dimensionality [Segera D et al., 2023], and contrary to classic optimization methods, will make it possible to take up the challenge of searching in the large space for feasible solutions $\aleph^d$, without going through it entirely, a good solution in an acceptable time very close to the Pareto front.

1.6. Metaheuristics

Metaheuristic algorithms appeared in the 1980s with the ambition to solve complex optimization problems. Glover F was the first to use the term metaheuristic in 1986 [Sörensen K et al., 2018; Alabas-Uslu C et al., 2020]. The etymology of Metaheuristic has Greek origins: Meta which means beyond and Heuristic which means to find. The etymology therefore means that a metaheuristic is an algorithm that positions itself above heuristics with the objective of implementing them in order to obtain a global solution to an extremely large amount of real-world optimization problems [Toloo M et al., 2022b]. A heuristic and a metaheuristic are therefore two optimization methods that should not be confused. To do this, we will provide firstly a definition of these two optimization methods before explaining the common specificities of metaheuristics and the curses of their parameters. We will then discuss the mathematical interest of metaheuristics, their inherent limits as well as a general classification. Finally, we will end this part with three vital concepts for metaheuristics: exploration, exploitation and hybridization.

1.6.1. Definition of a heuristic

A heuristic is an optimization method specific to a given optimization problem, especially for difficult combinatorial optimization problems. The solution obtained by the heuristic is close to the optimum but not necessarily optimal [Siarry P, 2014a; Sörensen K et al., 2018; Rabbouch B et al., 2023].

1.6.2. Definition of a metaheuristic

According to Cuevas E et al. [Cuevas E et al., 2021c] "*A metaheuristic is a solution method that articulate interactions between some improvements in local heuristics and high-level strategies, aimed at escaping from local optimum in a solution space, aimed at a global optimum*". But the qualities of metaheuristics are much better defined by Osman IH et al. [Osman IH et al., 1996]: "*A metaheuristic is an iterative generation process that guides a subordinate heuristic by intelligently combining different concepts to explore and exploit the search space, learning strategies are used to structure information in order to efficiently find the near-optimal solutions* ".

According to these definitions, we can affirm that metaheuristics are above all powerful, stochastic generic heuristics, alternating between exploration and exploitation phases with an imperative of a

balanced tradeoff between these two phases and not to be trapped in a local optimum [Jerebic J et al., 2021]. The vast majority of metaheuristics share common specificities.

1.6.3. Common specificities of metaheuristics

The main specificities of metaheuristics which constitutes the essence of their strengths but also of their weaknesses are seven in number:

- Global;
- Stochastic, which helps to deal with the curse of dimensionality;
- Generic even if the context is continuous or discrete;
- Do not need the gradient of the objective function in the case of continuous problems;
- Finds an acceptable solution in a reasonable time without guaranteeing optimality;
- With difficulties in the choice of assignment of the values of the control parameters;
- Inspired by analogies from nature.

We will develop in the next subsection the issue of the parameters tuning of metaheuristics because of the challenge it represents for the research community in the field of metaheuristic optimization.

1.6.4. The curse of parameters

The number of quantitative parameters of a metaheuristic, the time-consuming cost and the difficulty of their configuration is one of the main weaknesses of metaheuristics [Siarry P, 2014b; Siarry P, 2014c]. In addition, the closer we get to ad-hoc tuning of the parameters of a metaheuristic, for a given optimization problem instance, the more we benefit from the following effects:

- Increased performance of metaheuristics;
- Decrease in the spatial distance between the final solution and the global optimum;
- Reduced execution time to obtain the final solution;
- Reduced processing cost of a problem.

Given the role of parameters in the performance of a metaheuristic, the issue of parameters tuning is therefore far from being a secondary action.

Just like the curse of dimensionality, metaheuristics are also plagued by the "curse of parameters". Indeed, increasing the number of parameters in a metaheuristic only exponentially increases the number

of possible parameter vectors in the parameter search space. Consider a striking example among the recent metaheuristics: Squirrel search algorithm (SSA) [Mohit J et al., 2019]. SSA requires the configuration of six parameters including three integers and three real:

- The maximum number of iteration T_{max};
- The population size N;
- The nutrition food resources N_{fs}
- The gliding distance d_g ;
- The predator presence probability P_{dp} ;
- The gliding constant G_c.

The author of SSA himself confirms that a bad tuning of his parameters will induce bad results. From the point of view of this problematic, the ideal metaheuristic would be the one whose parameters are dynamic or self-adaptive during the iterations.

In general in the literature, the fine optimal tuning of the metaheuristic parameters is mainly empirical based on expert knowledge and following several tests carried out on function benchmarks[Agushaka JO et al., 2022]. But this empirical method does not allow, in the face of the curse of the parameters, to obtain the optimal parameter vector. And if a researcher approaches it during his work, it is essential for him to justify the choice of his parameters for a metaheuristic and a problem instance so that the research community can rely on these results and conclusions for their future work [Kazikova A et al., 2020].

In order to face the issue of the parameter tuning, researchers have developed a wide variety of parameter tuning methods that aim to maximize the performance of metaheuristics for a given optimization problem instance [Huang C et al., 2020].

In [Talbi EG, 2009] Talbi EG provided a complete taxonomy of parameter tuning methods split into two main paradigms: Offline parameter tuning and Online parameter tuning. We explain this taxonomy in Fig. 1.8 below.

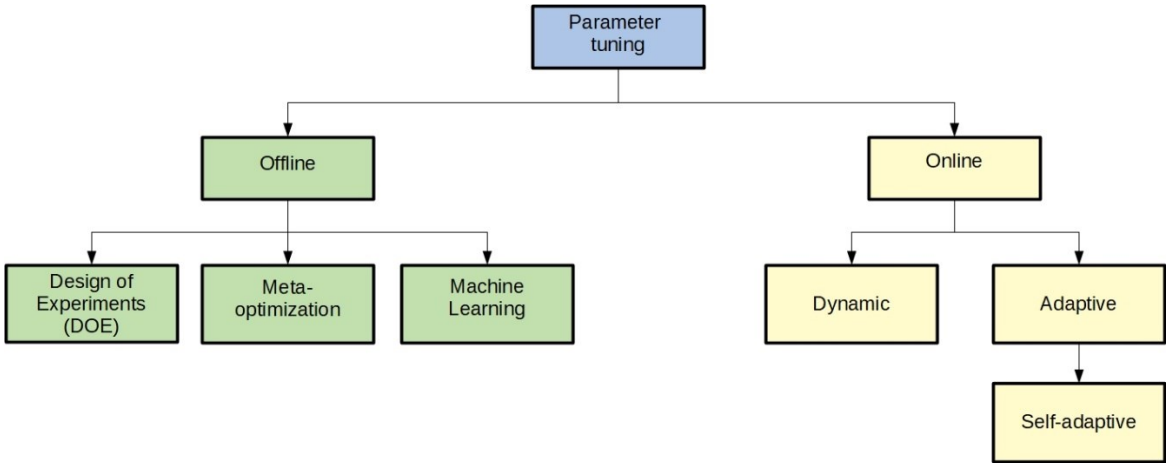


Fig. 1.8. Parameter tuning taxonomy

Before describing the Offline and Online paradigms let us mathematically define the components of the metaheuristic parameter optimization problem:

- A metaheuristic M;
- A parameter search space C;
- An instance of problem I;
- A performance metric P for the measure of the performance of a vecteur $c \in C$ of an instance of metaheuristic $M(c)$ and an instance of problem I.

The objective is to obtain the optimal parameter vector $c^* \in C$ which makes it possible to obtain the metaheuristic $M(c^*)$ with the best performance on the instance of problem I. Mathematically this is expressed by $max\{P: c \in C\}$.

A. Offline parameter tuning

In the Offline parameter tuning paradigm, the vector of parameters c is defined before the execution of the metaheuristic M, without modification during the execution, by exploiting three groups of methods: Design of Experiments (DOE), Meta-Optimization and Machine Learning.

a. Design of Experiments

As its name indicates, the DOE methods exploit the experience acquired during the analysis of the variations of P during the changes of values of the components of the vector parameter c [Pereira I et al., 2013]. However, the computation time of this method increases with the number of parameters.

Unsupervised learning algorithms are used to support DOE methods in order to select and define the most influential metaheuristic parameters on an instance of optimization problems. In [Ramos ICO et

al., 2005] Ramos ICO et al. exploited the logistic regression algorithm to implement the DOE method on evolutionary algorithm ProtoG applied to the Traveling Salesman Problem (TSP). The authors clustered the parameter search space in order to detect the most influential parameter cluster on the TSP problem. This allowed to reduce the time cost for the tuning of the parameters.

b. Meta-Optimization

The meta-Optimization method uses metaheuristic algorithms to determine the vector of parameter c closest to the optimal vector c^*. In this method, these metaheuristics are positioned at the upper-level and the metaheuristics M for which the optimal parameter vector must be determined are positioned at the low-level [Pereira I et al., 2013; Huang C et al., 2020]. Thus, meta-level metaheuristics see the metaheuristic set M and problem instance I as the black box optimization problem to be solved by finding the closest parameter vector c to c^* in the parameter search space C. The value of the objective function to this black box optimization problem, for a given vector of parameter c, is that obtained for the best solution acquired by the base-level metaheuristic.

In [Crawford B et al., 2013] Crawford B et al. used an upper level metaheuristic to tune the parameters of a low level metaheuristic. The upper level metaheuristic is Genetic Algorithm (GA) and the lower metaheuristic are Ant Colony System (ACS) and Scatter Search (SS) applied to the Set Covering optimization problem.

c. Machine Learning

Machine Learning algorithms are able to predict the parameter vector c as close as possible to the optimal parameter vector c^* for a problem instance I. Machine Learning algorithms therefore make it possible to adapt a metaheuristic to different problem instance before execution.

The work of Rasku J et al. in [Rasku J et al., 2016], based on Vehicle Routing Problem (CVRP) instance benchmark, combined three Machine Learning algorithms: Principal Component Analysis (PCA), the clustering algorithm DBSCAN and Random Forest (RF) Algorithm. This combination of algorithms made it possible to determine which vector c (Initial temperature and cooldown factor) is necessary for the SA metaheuristic to solve an instance of problem VRP.

B. Online parameter tuning

Unlike the Offline paradigm, the Online paradigm updates in real time the parameters values during the execution of the metaheuristic in a dynamic or adaptive way.

a. Dynamic

Dynamic methods update the parameters at each iteration, via random or deterministic variables regardless of the search strategy.

In [Eiben AE et al., 2011] the authors take up the two levels of the upper and lower metaheuristics which they position respectively in the design layer and algorithm layer. Unlike Meta-Optimization which selects the parameters only during the initialization of a metaheuristic, in [Eiben AE et al., 2011] evolutionary algorithms (EA) (GA, EDA, ES, etc.) positioned in the design layer as that meta-EA select the parameters of a metaheuristic stochastically and dynamically at each iteration. This is also the case of the metaheuristic Harris Hawk Optimization (HHO) [Heidari AA et al., 2019] which randomly updates its two parameters J and E.

b. Adaptative

Adaptive or even self-adaptive methods constitute a privileged path for researchers. Algorithms in the field of artificial intelligence able to self-regulate the parameters of a metaheuristic in order to maximize its performance during iteration constitute a decisive advance in the field of the optimization of NP-Hard problems. Alabas-Uslu C et al. [Alabas-Uslu C et al., 2020] implemented, for instances of the combinatorial optimization problem VRP, a self-adaptive local search algorithm (SALS) which self-adaptes its unique parameter θ. To do this SALS calculates on the one hand the value α_1 by the ratio of the values of the objective function of the best solution $X_b^{(i)}$ at iteration i and of the initial solution X_z and on the other hand the value α_2 by the ratio between the improvement number of the best solution at iteration i and the value i.

The curse of the parameters is therefore not a fatality thanks to the work of the research community. Self-adaptive methods exploiting reinforcement learning algorithms is the preferred choice for researchers to increase in the coming years the mathematical interest of metaheuristics in solving real NP-Hard optimization problems.

1.6.5. Mathematical interest of metaheuristics for the NP-Hard optimization problems

Compared to other optimization methods, metaheuristics have unique and exceptional mathematical qualities. As specified by Siarry P [Siarry P, 2016] and Cuevas E et al [Cuevas E et al., 2021c], compared to the classical optimization methods that we saw in 2.3, each metaheuristic has its own algorithmic strategy which allows it to analyze the search space in order to obtain a final solution as close as possible to the global optimum. Metaheuristics thus allow the resolution of a very wide range of optimization problems, too complex for "conventional" optimization methods, even if the problems are linear or non-linear, discrete or continuous, non-convex, non-differentiable, unimodal or multimodal, low or high dimension [Cuevas E et al., 2021b; Houssein EH et al., 2021b; Houssein EH et al., 2021c]. Indeed, real-world optimization problems are mostly nonlinear, discontinuous, highly multimodal, high-dimensional with non-smooth constraints and noisy. While other classical optimization methods require the objective function to be twice differentiable, metaheuristics do not require this. Indeed the metaheuristics are gradient-free and are not interested in the continuity or the differentiability of the problem or in the smoothing [Teghem J, 2012; Qian L et al., 2020; Cuevas E et al., 2021b]. Based on their current state variables, they only need the return value of the objective function $f(x)$, for an input candidate solution x created by the metaheuristic. Thus $f(x)$ is like a "breadcrumb trail" for metaheuristics in the search for the global optimum of the optimization problem. This is the main reason why the mathematical modeling of an optimization problem in objective functions will condition the quality of the final results. Furthermore, in order to avoid or extricate themselves from the pitfalls of local optima, metaheuristics temporarily deteriorate their situation during certain iterations with the aim of obtaining a better one in the following iterations.

Thus, metaheuristics see optimization problems as a "black box" [Qian L et al., 2020; Cuevas E et al., 2021b; Houssein EH et al., 2021c] regardless of its content. And this ability to consider the optimization problem as a black box is far from trivial. On the contrary, it opens up a wide range of technical possibilities that do not exist with conventional optimization methods. Indeed, it is a major asset for solving very complex optimization problems of the *NP-Hard* type.

Based on the description of metaheuristics and their mathematical specificities, we are in a position to propose in Fig. 1.9 a general diagram of a metaheuristic.

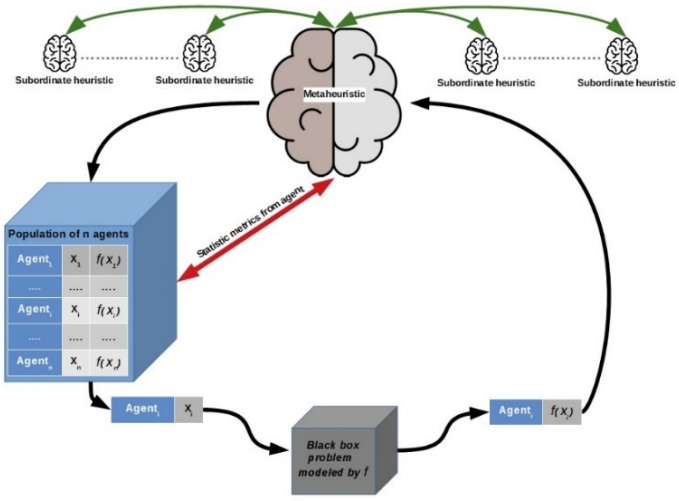


Fig. 1.9. General diagram of a metaheuristic

However, a metaheuristic should not be considered as the keystone of all optimization problems because it also has its limits that the No Free Lunch theorem clearly reminds us.

1.6.6. The limits of metaheuristics via the No Free Lunch theorem

When discussing the superior potential of metaheuristics, it is necessary to consider them as a whole and not individually. Wolpert et al. [Wolpert DH et al., 1996; Wolpert DH et al., 1997] were able to theorize its limits by the No Free Lunch theorem. Indeed, the No Free Lunch (NFL) theorem reminds us of the individual limits of a metaheuristic [Adam SP et al., 2019; Wolpert DH, 2021]. According to the NFL, each metaheuristic performs better than others on some subset of optimization problems and performs poorly on others. There is thus no metaheuristic superior to other if we consider the whole space of optimization problems. Therefore, the metaheuristics will have on average the same performance over the whole space of optimization problems and for each optimization problem it is necessary to choose the most suitable metaheuristics to solve it. And, in order to support the selection of appropriate metaheuristics to the problem to be solved, a synthetic classification of the latter is desirable.

1.6.7. Classification of metaheuristics

The immense amount of metaheuristics that can be found in the literature are generally inspired by nature. The researchers, through a metaphorical process, were able to model at the mathematical and algorithmic level the optimization processes resulting from nature which effectively made it possible to overcome the problems of survival and to prolong existence [Hussain K et al., 2019a; Moshtaghi HR et al., 2021]. These sources of inspiration have thus made it possible to design a profusion of classification of metaheuristics [Dhiman G et al., 2017; Abdel-Basset M et al., 2018; Abd EM et al., 2021; Avjeet S et al., 2021; Cuevas E et al., 2021c; Cuevas E et al., 2021d; Moshtaghi HR et al., 2021; Houssein EH et al., 2022a; Houssein EH et al., 2022d; Muazu AA et al., 2022; Vinod Chandra SS et al., 2022]. However, this profusion only complicates the search for a common denominator reconciling all its classifications.

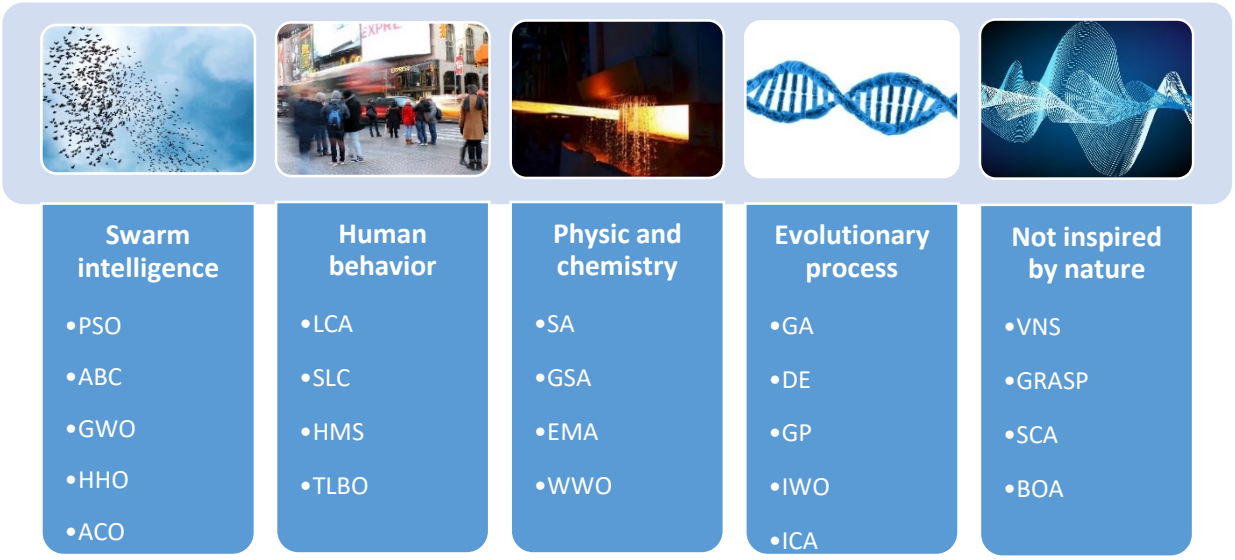


Fig. 1.10. Classification of metaheuristics into five groups (Image source: *https://www.westend61.de*, *https://www.gettyimages.fr*, *https://www.soustraiter.fr*, *https://www.infogm.org*, *https://www.freepik.com*)

Our analysis of the state of the art has enabled us to highlight five major groups capable of federating these classifications: **Swarm intelligence**, **Human behavior**, **Physic**, **Evolutionary** and **Not inspired by nature**. Fig. 1.10 represents these five groups with for each examples of well-known metaheuristics.

We can complete this classification by splitting the metaheuristics according to the size of their agent: single-based metaheuristic such Simulated Annealing (SA) and population based metaheuristic such as HHO [Hussain K et al., 2019a; Houssein EH et al., 2022a].

The list of metaheuristics in Fig. 1.10 is obviously not exhaustive. The articles [Moshtaghi HR et al., 2021] and [Abdel-Basset M et al., 2018] provide a broad view of the oldest to the most recent metaheuristics. And as we mentioned in 2.6.1 each of these metaheuristics has its own exploration and exploitation strategy with the ultimate aim of finding the global optimum of the optimization problems.

1.6.8. Exploration and exploitation

For a metaheuristic, exploration (or diversification) and exploitation (or intensification) are the two pillars of its optimization process with opposite strategies for finding a good final solution [Siarry P, 2014d; Ghazali R et al., 2018; Cuevas E et al., 2021a]. A balance between these two phases is the key to a successful metaheuristic which allows it to find solutions close to the global optimum and to avoid or escape from the traps of the local minima of the objective function. [Morales-Castañeda B et al., 2020]. This process is called the strategic oscillation between exploration and exploitation [Teghem J, 2012]. However, balance should not be confused with equality. Indeed, having the same number of exploration and exploitation phases does not give the assurance of obtaining better results. Balancing exploration and exploitation means designing and implementing a strategy that will create the right ratio between the exploration and exploitation phases as iterations occur. Of course, the strategy is intimately linked to the algorithmic specificities and mathematical operators of each metaheuristic. We will describe below these two pillars of optimization enriched by exploration and exploitation strategies implemented by well-known metaheuristics in the literature. We will close this part with the attempts in the literature to measure the exploration and exploitation phases.

A. Exploration

Exploration is a global, diverse search for unvisited promising areas in the search space [Cuevas E et al., 2021a; Jerebic J et al., 2021]. This process is based on stochastic "jumps". A good exploration therefore increases the probability of finding the area of the search space containing the global optimum and of thus obtaining a final solution closest to or equal to this global optimum. Thus a good exploration decreases the probability of being trapped in a local optimum.

For example, in Artificial Bee Colony (ABC) [Karaboga D et al., 2007], exploration is the responsibility of bee scouts who are assigned a randomly selected food source. The more scout bees

there will be following the abandonment by an exploiting bee of a bad source of food (position) and the more exploration will be favored.

In the case of Simulated Annealing (SA) [Kirkpatrick S et al., 1983], it's the acceptance criterion of Metropolis with the probability $e^{\frac{-\Delta E}{T}}$ which makes it possible to accept a degradation of the position in the search space in order to explore other potentially promising valleys . ΔE represents the energy variation (or variation of the objective function) and T the temperature of the solid. The higher the temperature, the greater the probability of accepting a degradation of the position. This has the effect of encouraging exploration.

However, if the exploration is excessive within the framework of respecting the exploration-exploration balance, it will degrade the speed of convergence. The complement to exploration is exploitation which has an opposite research logic.

B. Exploitation

Exploitation aims to stochastically intensify local search in promising area located during exploration. Consequently, the exploitation makes it possible to refine the best promising area already found in exploration phases [Cuevas E et al., 2021a; Jerebic J et al., 2021]. Generally, the exploitation consists in "mutating" a solution x^* by a movement around x^*in order to obtain, in its neighborhood $N(x^*)$, a better solution $x \in N(x^*)$.

In ABC the mutation of a solution s is carried out in four steps:

- Random selection of dimension k of s to mutate;
- Selection of an influential food source s_i ;
- Selection of a random value ε in $[-1,1]$;
- Mutation operation on the dimension k of s according to the equation $s^k = s^k + \varepsilon * (s^k - s_i^k)$.

If the mutation of s provides a better food source, then the mutation is retained. Otherwise it is rejected and s gets a penalty. The exploiting bees with the most penalty, in proportion to the number of scout bees compared to the size of the bee population, will become scouts in the next iteration.

In Bat Algorithm (BA) [Yang XS, 2010] the mutation operates beforehand on the speed v of a bat of position x and ultrasound emission frequency f. The mutation equation is $v = v + (x - x^*) * f$. Then the position x undergoes the movement $x = x + v$.

Moreover, a good exploitation makes it possible to increase the rate of convergence towards a good solution. Nevertheless, excessive exploitation increases the probability of being trapped in a local optimum.

To better understand the exploration and exploitation process we will use the metaphor of the dig for diamonds. Gemologists (diamond diggers), first explore promising areas of the mine with probes and other drilling mining machines. This is the exploration phase in search of a promising diamond area. Then as soon as a promising area is detected, a new more refined search is carried out in its vicinity using hydraulic drills, shovels, pickaxes, rotary cutters and diamond testers. This is the exploitation phase. If the area fails, gemologists revert to the exploration phase so they don't get stuck in barren local drilling. The global logic is similar in metaheuristics but with "intelligent" stochastic components that allow them to extract themselves from local optima with an exploration-exploitation balance imperative. The literature assures us and repeats to us that a good exploration-exploitation balance guarantees better efficiency in the search for a good solution to the optimization problem. But how to know if the metaheuristic respects this imperative of balance? To do this, methods for measuring exploration and exploitation are needed.

C. Exploration and exploitation measurement methods

Measurements of exploration and exploitation are not easy because of the great heterogeneity of metaheuristics. However, the literature provides some methods of indirect and direct measurement of exploration and exploitation, particularly for population-based metaheuristics [Gabor T et al., 2017; Ghazali R et al., 2018; Hussain K et al., 2019b]. The indirect measurement method assesses the level of exploration or exploitation based on other metrics. The methods known in the literature are those of Swarm diversity measuring the positions of agents relative to each other, the measurement of Entropy [Gabor T et al., 2017; Hussain K et al., 2019b]and more simplistically, the monitoring of the improvement of the best value obtained from the objective function [Tilahun SL, 2017]. As for direct

measurement, as its name suggests, it directly measures the level of exploration and exploitation. One of the most recent methods is that of measuring the Attraction basins in the search space [Jerebic J et al., 2021].

But these methods of measurement only make it possible to note the effectiveness or the impotence of the operations of exploration and exploitation of a metaheuristic. If one wishes to act on the observed weaknesses, methods such as hybridization must be implemented.

1.6.9. Metaheuristics hybridization

Glover F was the first to create a hybrid metaheuristic in his work in [Glover F et al., 1998]. However, the literature does not clearly define the action framework of a metaheuristic hybridization. But this lack of definition can be seen as an advantage for the research community. Indeed, no definition can thus limit, by strict frameworks, the spirit of innovation of researchers to create new metaheuristics by hybridization. On the other hand, the literature specific to metaheuristic hybridization allows us to provide a common denominator that describes the major effect sought in hybridization: The main purpose of hybridization and to exploit the mutually beneficial synergy of at least two distinct algorithmic strategies with the purpose of obtaining a metaheuristic with increased performance for solving NP-Hard optimization problems [Talbi EG, 2013; Ting TO et al., 2015; Hassan A, 2019; Raidl GR et al., 2019; Cuevas E et al., 2021e].

The hybridization process must therefore allow a metaheuristic to increase its exploration and exploitation capacities with an imperative of balance between them in order to find a good solution close to the global solution by limiting its weaknesses which are premature convergence or slow convergence [Ting TO et al., 2015].

The hybridization of a metaheuristic generally exploits other metaheuristics, machine learning algorithms or other exact mathematical methods [Talbi EG, 2013].

In order to better understand the fundamentals of hybridization, we will focus in what follows on the hybridization between at least two well-known metaheuristics in the literature and explain the basic principles of the design of a hybrid metaheuristic which are based on the strategies of control, the order of execution and the level of hybridization [Talbi EG, 2013; Ting TO et al., 2015; Raidl GR et al., 2019].

A. Control strategy

The control strategy of a metaheuristic can be collaborative or integrative. A hybridization is collaborative when two algorithms act jointly "as equals" sequentially or in parallel during the optimization process. In this strategy, generally one of the algorithms acts on exploration and the second on exploitation.In [Ting TO et al., 2006] the authors created a hybrid metaheuristic to evaluate the flow of electric power in power transmission network. They used GA and PSO metaheuristics in a collaborative strategy. GA manages global search and PSO acts at the local search level.

On the other hand, the integrative strategy creates a link of subordination between two metaheuristics. In this strategy, a slave algorithm is integrated into a master algorithm and it is the master algorithm that directs the optimization process. Birogul S proposes in [Birogul S, 2019] an integrative hybridization of the DE metaheuristic in HHO to obtain HHODE. The five mutation equations of DE have been integrated into the exploratory equations of HHO in order to increase its exploration capacity. Tests on the CEC2005 and CEC2017 benchmarks confirmed the increased performance of HHODE compared to HHO.

In the two control strategies that we have just described, several types of order of execution are possible.

B. Order of execution strategy

After the control strategy, the hybridization of a metaheuristic requires an order of execution strategy. Three strategies are possible: sequential, parallel and interleaved.

The integrative control strategy uses a sequential or interleaved execution order. This is the case of HHODE [Birogul S, 2019] which has a sequential execution order. The Hybrid HHO (H-HHO) metaheuristic in [Abualigah L et al., 2021], used for data clustering, just like HHODE, integrates DE into HHO with an interleaved execution order to reduce its weaknesses during the exploration phases and also to operation.

On the other hand, the parallel execution order is exploited in the collaborative control strategy. In [Hijazi NM et al., 2021] the authors combine three metaheuristics with a parallel execution order to design a hybrid metaheuristic for feature selection: Gray Wolf Optimization (GWO) [Mirjalili S et al., 2014], Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

Within the sequential or interleaved order of execution, the level of hybridization strategy depends on the strength of the fusion of the metaheuristics which acts on the hybridization level.

C. Hybridization level strategy

The designer of a hybrid metaheuristic can choose to combine these metaheuristics with a high-level or low-level strategy. In a high-level strategy, the algorithms and equations of the metaheuristics are not modified and maintain their independence. Metaheuristics are therefore weakly-coupled and communicate via an ad hoc interface. This high-level strategy is present in a context of collaborative metaheuristics with sequential execution as in [Ting TO et al., 2006].

In the low-level strategy, on the contrary, the metaheuristics are strongly-coupled and therefore inter-dependent. Their algorithms and equations are changed. This strategy is found in a context of integrated metaheuristics with an execution order interleaved as in [Abualigah L et al., 2021].

The Fig. 1.11 below provides a summary diagram of hybridization strategies.

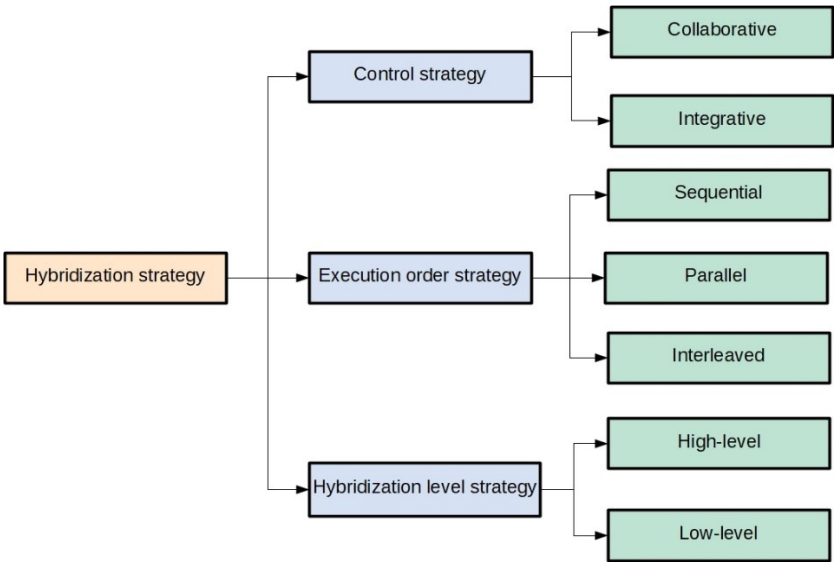


Fig. 1.11. Summary diagram of hybridization strategies

1.7. Conclusion

Optimization problems are ubiquitous in the technical environments of engineers and researchers. They can be mono-objective without constraints for the most basic ones, but the vast majority of real-world optimization problems are multi-objective with several constraints and having to satisfy multiple contradictory aspirations by compromise. We have seen that there is a plethora of methods for solving optimization problems depending on the category and complexity classes to which the problem belongs.

The problems that pose the most difficulty to the research community, as their names suggests, are problems of the *NP-Hard* class. Fortunately, metaheuristic algorithms appeared in the 1980s opened up promising new perspectives for solving *NP-Hard* problems. One of their major assets is to consider the optimization problem as a black box without worrying about the internal mathematical specifics of the problem. The other major asset of metaheuristics is their intelligent stochastic nature which allows them, through balanced exploration and exploitation processes, to extract themselves from local optima and to get as close as possible to the global optimum. Thanks to their strengths, metaheuristics have managed to solve many *NP-Hard* engineering problems.

The spirit of innovation of the researchers has made it possible to design a very large number of metaheuristics mainly inspired by nature and by hybridization. As the No Free Lunch Theorem states, each metaheuristic can excel on some subset of optimization problems and be deficient in others. For the cybersecurity objectives of our thesis, we have taken advantage of a very recent nature-inspired and population-based metaheuristic: Harris Hawk Optimization (HHO). However, this metaheuristic does not escape to the NFL theory. It has weaknesses for high-dimensional and highly multimodal optimization problems.

In the chapter 2 we will develop our unseen hybridization strategy of HHO totally bio-inspired by the hunting synergy between crows and wolves to design the new metaheuristic HHO-EAS. Our hybridization strategy aims to increase HHO-EAS skills during the exploration and exploitation phases and to achieve better performance than HHO for high-dimensional and highly multimodal optimization problems.

Chapter 2

Conception of the metaheuristic HHO-EAS for highly multimodal and high-dimensional optimization problems

2.1. Introduction

As we explained in chapter 1, the conception of metaheuristics is mostly inspired by nature, and they are divided into four categories: Bio-inspired, Physics and Chemistry, Evolutionary and Swarm Intelligence [Houssein EH et al., 2022a; Houssein EH et al., 2022b; Houssein EH et al., 2022d]. These metaheurtics are single-based such Simulated Annealing (SA) or population-based such as Harris Hawk Optimization (HHO).

HHO is a very recent metaheuristic inspired of Harris Hawk's pack hunting techniques. It was created by Heidari AA et al. in 2019 [Heidari AA et al., 2019]. However, as we explained in chapter 1, the No Free Lunch theorem reminds us that no metaheuristic can claim to outperform all metaheuristics on all optimization problems [Wolpert DH et al., 1997]. According to this theorem such a metaheuristic does not exist. Thereby, like all metaheuristics, HHO has some weaknesses, particularly for the optimization of highly multimodal and high-dimensional problems. Indeed, for highly multimodal and high-dimensional problems, HHO fails to maintain a good balance between exploration and exploitation to end up trapped in a local optimum [Chen H et al., 2020; Alabool HM et al., 2021].

However, these weaknesses are far from being inevitable. So, in order to minimize the HHO's weaknesses we designed a hybridization strategy totally inspired by a partnership between two predators that everything separates.

In northern Europe and in Yellowstone Park in Wyoming United-state, during the hard winter months when good preys are hard to find and capture, an atypical alliance between two predators was born: the crows of the family corvidae and the wolves of the family canidae [Stahler D et al., 2002; Milner R, 2020]. These two predators have combined the smart and adaptive exploration of the hunting ground of crows with the exceptional abilities of wolves to capture rapidly and in an organized way prey even if it is larger than them.

In order to improve the optimization capabilities of HHO for highly multimodal and high-dimensional optimization problems, we mathematically modeled this win-win hunting synergy. We designed a

Mamdani-like Fuzzy Inference System (FIS) to model the crows' smart and adaptive exploration and the Encirclement and Attack equations to model wolves action. Then we integrated them into HHO to create a new metaheuristic: Harris Hawk Optimization – Encirclement Attack Synergy (HHO-EAS). These work has been the subject of a scientific article validated for a publication in the journal Artificial Intelligence Review.[Sassi M et al., 2023].

Thus, the main contributions of our work are twofold:

- We have designed the new metaheuristic HHO-EAS overall more efficient than HHO for the optimization of highly multimodal and high dimension problems;
- Unlike classical hybridization strategies that combine metaheuristics with each other, our hybridization strategy is based on another paradigm entirely inspired by the win-win hunting synergy observed in nature between two animals: the crows and the wolves.

This chapter, is articulated as follows. In the section 2.2 we will describe the HHO algorithm with its exploration and exploitation techniques, its weaknesses and the main works aimed at improving it by hybridization.

The section 2.3 will present our contribution in order to design HHO-EAS by detailing our hybridization strategy bio-inspired by the win-win hunting synergy between the crows and wolves.

In the section 2.4 we will demonstrate firstly, with a general benchmark of 19 well-known unimodal, multimodal and composite functions, the overall superiority of HHO-EAS over HHO as well as over two other population-based metaheuristics, GWO and PSO. Secondly we will focus only on HHO-EAS and HHO with a specific benchmark of the 20 most complex optimization problems of the CEC 2017 close to real life optimization problems: 10 hybrid and 10 composite functions.

We will conclude this chapter in the section 2.5 and we will open the future applications of HHO-EAS in the chapter 3.

2.2. Harris Hawk Optimization

The metaheuristic HHO, created by Ali Asghar Heidari in 2019 [Heidari AA et al., 2019] is inspired by the Harris Hawks' pack hunting strategy. HHO is a population-based metaheuristic allowing global and stochastic optimization guided by intelligent mechanisms during the exploration and exploitation

phases [Shu PW et al., 2020]. Furthermore, HHO is almost an "autonomous" metaheuristic, that is to say with very few parameters to configure in its algorithm. However, HHO as all metaheuristic, presents weaknesses that some works have attempted to reduce.

2.2.1. HHO Inspiration

Biologists Jennifer O. Coulson and Thomas D. Coulson studied the cooperative hunting techniques of the Harris Hawks. Their articles make reference in this field [Coulson J et al., 2012; Coulson JO, 2013]. Their research has provided insight into the Harris Hawks' hunting strategies. The Harris Hawks' hunting techniques inspired HHO's exploration and exploitation equations.

Harris Hawks are raptors with some intelligence and have the particularity of living in packs unlike other Hawks. They practice cooperative hunting.

They live between southwestern North America and central and southern of South America. Their living areas are semi-desert.

Their strategy for finding and capturing preys is organized in groups and structured in two stages: Search and Attack.

2.2.2. Search phase (Exploration)

They begin as a single group by gathering on the cacti, tree branches, utility poles, etc.

Then they split into two groups. The first group flies from perch to perch like leapfrog to get a different point of view. This allows them to better explore the surrounding terrain in search of prey with a preference for rabbits. They sometimes practice the backend to get a better point of view (One Harris Hawk perches on top of another).

The second group, to find a prey, flies over a potentially promising areas in packs. The two groups maintain eye contact to be ready to go into attack phase if they detect a good prey.

2.2.3. Attack Phase (Exploitation)

The Harris Hawk's key hunting tactic is the surprise pounce to attack a prey such as rabbit.

After detecting the rabbit, each Harris Hawk pounces on it, if one of the Harris Hawks misses the target because the prey fled in the opposite direction, the Harris Hawk in the opposite direction takes over

in the attack. Thus, Harris Hawks cooperatively attack the prey from several directions. This attack resembles to a Harris Hawk flurry on the prey from all possible sides.

Harris Hawk's research and attack techniques are all modeled by the exploration and exploitation equations in the HHO algorithm.

2.2.4. HHO Algorithm

HHO metaheuristic, like all metaheuristics, is structured in two phases: exploration and exploitation.

The exploration phases are guided by two exploratory equations inspired by the prey search phase. And the exploitation phases are guided by four equations inspired by the attack phase. These four equations make it possible to search for better solutions in a neighborhood far or close to promising areas.

The exploration, the exploitation and the transition of HHO between exploration and exploitation are all driven by the Rabbit Escaping Energy (REE) variable. But that's not all. REE also makes it possible to control the magnitude of local searches in the near or far neighborhood of the best solution at iteration t.

Our analysis of this algorithm allows us to affirm that REE is the central element of HHO because it's this that coordinates the action of the Harris Hawk agents at each iteration. In addition, REE has a major influence in the balance between exploration and exploitation as well as in the search for a good solution.

A. Rabbit Escaping Energy

REE is defined by the equations (2.1) et (2.2). REE's amplitude decreases linearly in term of the Iteration. Through the equations (2.1) and (2.2), REE ensures a smooth transition between the exploration and exploitation phases. It is mathematically represented by the function E.

$$E = 2E_0\left(1 - \frac{t}{T_{max}}\right) \tag{2.1}$$

$$E_0 = 2r - 1 \tag{2.2}$$

In (2.1) the iteration is represented by t and T_{max} is the maximum number of iteration.

In (2.2) r is an uniform random variable defined in $[0,1]$, therefore $E_0 \in [-1,1]$.

Based on (2.1) and (2.2), $E \in [-2,2]$ and $|E|$ reaches zero at the end of the iteration T_{max}. We can also notice that $|E| < 1$ from half of the iteration $\frac{T_{max}}{2}$. (2.3) summarizes the membership intervals of E according to the iteration t.

$$\begin{cases} if\ t\ \in \left[0, \frac{Tmax}{2}\right], E \in [-2,2] \\ if\ t\ \in \left[\frac{Tmax}{2}, Tmax\right], E \in [-1,1] \\ if\ \ t = \ Tmax\ , E = 0 \end{cases} \quad (2.3)$$

The REE evolution curve is provided by the Fig. 2.1. As noted above, REE drives the exploration and exploitation phases.

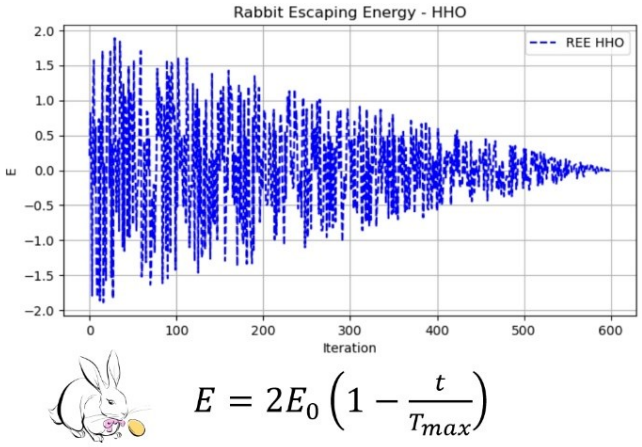


Fig. 2.1. REE evolution curve

B. Exploration phases

The exploration phases take place when the of REE's amplitude is greater than or equal to 1:$|E\ | \geq 1$.

The exploration phases model the two behaviors of Harris Hawks when searching for prey. Harris Hawks either jump from perch to perch in search of a better vantage point to detect a rabbit, or they group together by flying around a promising area and by perching close to each other in order to be ready to migrate in attack formation if they detect a rabbit. The second situation therefore exploits the rabbit's position X_r, and the Harris Hawks swarm average position X_m.

The exploration process is done using the exploratory equations (2.4). These equations make it possible, by stochastic leap, to diversify the search for unexplored promising areas in the search space and to calculate the Harris Hawk's next position.

These two situations are equally likely. Thus, for a uniform random variables q in $[0,1]$, the first situation will occur if $q \geq 0.5$ and the second situation if $q < 0.5$.

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)|, & q \geq 0.5 \\ (X_r(t) - X_m(t)) - r_3(LB + r_4(UB - LB)), & q < 0.5 \end{cases} \quad (2.4)$$

- r_1, r_2, r_3 et r_4 are uniform random variables in $[0,1]$;
- LB and UB are respectively the lower and upper bounds of the search space, which makes it possible to create a random position in the search space with $r_3(LB + r_4(UB - LB))$;
- X_r is the rabbit's position, so the best solution at iteration t;
- X_{rand} is a position chosen at random from the Harris Hawk population;
- $X_m(t)$ is the average position of the Harris Hawk population and is calculated by (2.5):

$$X_m(t) = \frac{\sum_{i=1}^{N} X_i}{N} \quad (2.5)$$

C. Exploitation phases

The exploitation phases take place when: $|E| < 1$.

The rabbit is under fire from Harris Hawk attacks. There are four exploitation phases (attacks), each of which models a collective Harris Hawk attacks: Soft besiege, Hard besiege, Soft besiege with progressive rapid dives and Hard besiege with progressive rapid dives. Each of these four phases will seek better solutions in a neighborhood near or far from $X_r(t)$.

The four phases are identified by the couple ($|E|, r_5$) with r_5 a uniform random variable in $[0,1]$.

The random variable r_5 let's us to know if the rabbit manages to escape $r_5 < 0.5$ or if his escape attempt failed $r_5 \geq 0.5$. The success and failure are equally likely.

$|E|$ allows us to measure the rabbit exhaustion during the Harris Hawk attacks and therefore its ability to be able to escape. If $|E| \geq 0.5$ the rabbit has enough energy to run away. On the other hand if $|E| < 0.5$, the rabbit is exhausted and will have more difficulty avoiding Harris Hawk attacks.

Thus the four exploitation situations (attacks) are identified as follows:

- **Soft Besiege (SB)**: if $|E| \geq 0.5$ and $r_5 \geq 0.5$;
- **Hard Besiege (HB)** : if $|E| < 0.5$ and $r_5 \geq 0.5$;
- **Soft Besiege with Progressive Rapid Dives (SBPRD)**: if $|E| \geq 0.5$ and $r_5 < 0.5$;
- **Hard Besiege with Progressive Rapid Dives (HBPRD)**: if $|E| < 0.5$ and $r_5 < 0.5$.

One of the strengths of HHO is the integration of the Levy Flight equation in SBPRD and HBPRD. Given that the position of the global solution is unknown in the search space, just like the position of a prey is unknown in the hunting ground, the Levy Flight allows Harris Hawk to perform effective local research by random short-range jumps in the vicinity of X_r. This method of research, by exploiting the Levy Flight distribution, increases search diversity in the distant neighborhood of X_r.

We are going to describe in detail each of the four exploitation phases explained above.

a. Soft besiege

The rabbit is not yet exhausted and is hopping in all directions in an attempt to escape. This therefore creates uncertainty in his position. Unfortunately for him he cannot escape. Harris hawks take advantage of this and perform surprise pounces to capture their prey. This attack is modeled by the equations (2.6) and (2.7) which allow to calculate the next position:

$$J = 2.(1 - r_6) \tag{2.6}$$

$$\begin{cases} X(t+1) = \Delta X(t) - E.|J.X_r(t) - X(t)| \\ \Delta X(t) = Xr(t) - X(t) \end{cases} \tag{2.7}$$

r_6 is a uniform random variable in $[0,1]$, so J belongs to the interval $[0,2]$. J models the uncertainty of the rabbit position and helps to extend the search area in the neighborhood of $Xr(t)$.

The search magnitude in the neighborhood of $Xr(t)$ is managed by E. Since $|E| \geq 0.5$, the search takes place in a far neighborhood from $Xr(t)$.

b. Hard besiege

The rabbit is exhausted and no longer has the energy to perform run, jump and escape. So there is less uncertainty in the rabbit position. Harris hawks perform thus surprise pounces all around the rabbit to capture it. This close attack is modeled by equation (15). As $|E| < 0.5$, the search is performed in a close neighborhood of $Xr(t)$.

$$X(t+1) = X_r(t) - E.|\Delta X(t)| \tag{2.8}$$

c. Soft besiege with progressive rapid dives

The rabbit has enough energy and it manages to escape. So it performs random zigzag movements. Like the rabbit still has enough energy and that he avoids skillfully short-range attacks, Harris Hawks

perform a soft besiege in order to try to catch the rabbit with surprise pounces disordered. Then, the Harris Hawk agents correct their trajectory progressively.

In order to model the random rabbit zigzag movements, the Levy Flight (LF) equation (2.12) is used in the Harris Hawk movement strategy. The LF movement is a great way to widen the search areas stochastically during exploitation.

Equations (2.9) to (2.12) take into account two situations.

In the first situation equation (2.9) is sufficient to try to capture the rabbit and is satisfactory to calculate the next Harris Hawk position Y.

In the second situation the rabbit is much more skill than Harris Hawks. In this situation, the next Harris Hawk position Z is calculated with the equations (2.9), (2.10), (2.11) and (2.12).

$$Y = X_r(t) - E.|J.X_r(t) - X(t)| \tag{2.9}$$

$$Z = Y + S.LF \tag{2.10}$$

$$X(t+1) = \begin{cases} Y \ \ if \ \ F(Y) < F(X) \\ \\ Z \ \ if \ \ F(Z) < F(X) \end{cases} \tag{2.11}$$

$$\begin{cases} LF = 0.001.\frac{u.\sigma}{|v|^{\frac{1}{\beta}}} \\ \\ \sigma = \left(\frac{\Gamma(1+\beta).sin\left(\frac{\pi.\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right).\beta.2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \end{cases} \tag{2.12}$$

u and v are two vectors whose components are random variables in $[0,1]$ and β is a constant equal to 1,5.

d. Hard besiege with progressive rapid dives

The rabbit is exhausted and it is therefore in a difficult situation. However, it manages to escape. We observe again a random zigzag movement escape.

Like SBPRD, in HBPRD the calculation of the next Harris Hawk position uses the Levy Flight equation (2.12) and depend on two situations: the first situation uses equation (2.13) and the second

situation uses equations (2.12), (2.14) and (2.15). Since the Harris Hawk attacks strategy is in close formation, the next position calculation is based on the average Harris Hawk positions X_m. The average position X_m is calculated with equation (2.5).

$$Y = X_r(t) - E.|J.X_r(t) - X_m(t)| \tag{2.13}$$

$$Z = Y + S.LF \tag{2.14}$$

$$X(t+1) = \begin{cases} Y & if \quad F(Y) < F(X) \\ Z & if \quad F(Z) < F(X) \end{cases} \tag{2.15}$$

Appendix 6 provides the flowchart of HHO. The schematic representations of the updating Harris Hawk position provided in the literature are unreadable and incomprehensible views in 3-dimension on a 2-dimensional plan [Heidari AA et al., 2019; Hussain K et al., 2019c]. We propose in our flowchart a mapping more pragmatic of the updating of the Harris Hawk positions in the exploration and exploitation phases. Moreover, this new representation highlights REE as the central element of HHO as the coordinator of exploration and exploitation phases. The pseucode Algorithm 1 below implements the HHO algorithm.

Algorithm 2.1 Pseudocode of HHO

In: Population size *N*, number of iteration T_{max}, objective function *F*

Out: $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

Initialization of the Harris Hawk positions $\vec{X}_i, i \in \{1, .., N\}$, t=0

While (t < T_{max}) **do**:

Calculate $F(\vec{X}_i)$ of each Harris Hawk position $\vec{X}_i$

Define the best position of rabbit $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

For each Harris Hawk i **do**:

Update E_0, E, and J with (2.1), (2.2) and (2.6)

If($|E| \geq 1$) **then**: *[Exploration]*

Calculate q and **Update** position with (2.4)

Else: *[Exploitation]*

Calculate r_5

If $|E| \geq 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.6) and (2.7)

If $|E| < 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.8)

If $|E| \geq 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.9) to (2.12)

If $|E| < 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.12) to (2.15)

End for

t=t+1

End while

Return $\overrightarrow{X_r}$

In subsection 2.2.4 we have seen all the qualities of HHO metaheuristic inspired by Harris Hawks' pack hunting technique. It is a very recent and almost autonomous metaheuristic (two parameters to initialize) with sophisticated exploration and exploitation equations.

The experimental results in the literature demonstrate correct performances in search of the global solution for small and medium-dimension optimization problems [Heidari AA et al., 2019].

However, as the No Free Lunch Theorem [Wolpert DH et al., 1996] reminds us «There is no metaheuristic algorithm that can solve all optimization problems». This means that a metaheuristic can demonstrate good performance on a group of optimization problems, while getting mediocre results on another group. So, the perfect metaheuristic does not exist and HHO does not escape the rule. This opens up a wide range of innovation to researchers to improve HHO and to reduce its weaknesses. These weaknesses were particularly manifested in highly multimodal and high-dimensional optimization problems. Indeed, in this category of problems, HHO fails to maintain proper balance between exploration and exploitation phases to get stuck in one of the local optimum with low accuracy. Thus the Harris Hawks have a lot of weaknesses "to hunt and catch an acceptable prey" (acceptable solution) in a reasonable time.

2.2.5. The HHO's weaknesses and improvement attempts

Many studies in the literature have attempted to correct the HHO's weaknesses by hybridization strategies [Alabool HM et al., 2021]. These weaknesses relate more particularly to exploration and

exploitation capacities and the balance between these two phases. These weaknesses are accentuated by the multimodality of the optimization problem and with the curse of the dimensionality by increasing the dimension.

Four main strategies have been implemented to correct the HHO's weaknesses: Improve the exploratory equations, modify the REE function, increase the diversification of the population and adapt the exploitation area by altering the exploitation equations. These four strategies used individually or combined contribute to the final objective of correcting the HHO's weaknesses that we have just explained above.

HHO skills were exploited by Elkadeem M.R. et al. [Elkadeem MR et al., 2019] to resolve the optimization of renewable energy distribution planning in photovoltaic and volume panels. In order to improve HHO exploration capacities, the latter was hybridized with Particle Swarm Optimization (PSO) to create HHO-PSO. Indeed, PSO, although less evolved than HHO, is able to explore the search space quite quickly. It's this particularity that has been integrated into the exploratory equations of HHO. The experimental results testify to the good exploratory faculties of HHO-PSO compared to HHO.

In a research by Birogul S. [Birogul S, 2019] HHO was hybridized with the metaheuristic Differential Evolution (DE) to create HHODE. HHODE has allowed to better solve the optimization problem of Electrical Power Flow (OPF). Again, exploration is the improvement line of HHO by replacing the exploratory equations for $q \geq 0.5$. This equation is replaced by the five mutational equations of the most used algorithm in DE to benefit from their diversification capabilities. The experimental results demonstrated the superiority of HHODE on HHO.

Houssein E.H. et al. [Houssein EH et al., 2020] introduced a hybridization of HHO with the Cuckoo Search (CS) and chaotic maps to create the CHHO-CS metaheuristic. CHHO-CS was used for the feature selection of the chemical composition descriptors. CHHO-CS uses CS to achieve better X_{rand} and X_r position to improve explorations phases and the balance between exploration and exploitation. Then chaotic maps were incorporated into the REE function to change the shape of its amplitude to an inverted parabolic. Thus E evolve in $[-2,2]$ in the two half of the iteration. Thereby hybridization allows to reorganize the exploration and exploitation phases in the two halves of the iteration.

Jia H. et al. [Jia H et al., 2019] proposed DHHO/M in order to use it in satellite images segmentation. DHHO/M improves the exploration phases of HHO by replacing on the one hand a part the exploration equation for $q \geq 0.5$ with the DE best/2 operation. And on the other hand by adding to the REE a function which stochastically creates "pulses" during the second half of the Iteration. Those pulses increase the amplitude of REE. This makes it possible to have $|E| \geq 1$ during the second half of the Iteration, so new exploration phases could allow DHHO/M to get out from the local optimum traps. Tests on satellite image datasets validated the performance of DHHO/M against HHO.

The authors Gupta S. et al. [Gupta S et al., 2020] combined three of the four methods mentioned above to improve HHO by creating the metaheuristic m-HHO. The linear management of the amplitude of the REE function has been replaced by a decreasing quadratic exponential function. This function grants less iterations to the exploration phases while accelerating convergence. The Levy Flight (LF) function multiplied by a parameter α was added to the two exploitation equations (2.9) and (2.13). This parameter α controls the magnitude of the step of LF and varies throughout iterations in the same way as the REE function. This modification makes it possible to reduce disturbances in the exploitation phases as the iterations progress. The diversification phase of the algorithm has been implemented by the use of the Opposition-Based Learning (OBL) method applied to a greedy selection of the N_{op} best opposed solutions of the HHO population. N_{op} is a decreasing staircase function according to iterations which allows a wide diversification of the population at the start of iterations in order to promote exploration to then focus on a smaller portion of the population at the end of the iteration in order to promote exploitation. Tests on a benchmark of 33 functions provided better results for m-HHO with a better local optimums avoidance.

Fan Q. et al. [Fan Q et al., 2020] created QRHHO by integrating the Quasi-Reflection-Based Learning (QRBL) mechanism into HHO in order to diversify the population. QRBL is a variant of the Opposition-Based Learning (OBL) mechanism. QRBL is used for the initialization phases and the updating of the population at each iteration. The calculation of QRBL is performed on each position of the N agents of the population to provides N new QRBL agents. Then a greedy selection is performed on the original population and the QRBL population in order to keep only the N best agents. Tests on a benchmark of

23 functions have demonstrated that QRHHO has better exploitation, exploration and convergence capabilities than HHO.

Chen H. et al. [Chen H et al., 2020] have integrated into HHO three mechanisms allowing to correct the exploration, the exploitation and the precision of the final solution. To do this, they created the metaheuristic CMDHHO which uses respectively the topological multi-population strategy, the chaotic sequence of the Logistic map and the three DE operations: mutation, crossover and selection.

The 30 functions of the CEC2017 benchmark made it possible to highlight that the three mechanisms include at the same time in CMDHHO allowed a significant reduction of HHO's weaknesses to increase the convergence speed and the final solution quality, but without knowing which of the mechanisms introduced brings the most performance or the least performance or even no improvement.

Gao Z.M. et al. [Gao ZM et al., 2019] have used in HHO, the chaotic sequence of the Tent map to diversify at each iteration the best position X_r. It is a diversification method applied only to Rabbit position. This strategy is divided into three steps. In the first step the components of X_r, initially in the search space $[LB, UB]$, are mapped in $[0,1]$. In the second step the components are transformed into chaotic value with the Tent chaotic map. In the last step they are switched in the initial search space $[LB, UB]$. The reiteration of this process is equal to the number M of chaotic iteration. Therefore, M new Rabbit positions are created. A greedy selection of the best position among the M is finally made to replace the position X_r if its fitness value is better.

Experimental results on a benchmark of 18 functions compared HHO and the chaotic version of HHO. These results highlighted the superiority of chaotic version of HHO by a faster convergence and a better precision of the final solution.

All the work that we have just described above prove that HHO has great weaknesses and that a large leeway exists to improve it. However, these works only use classical hybridization paradigms to improve HHO. Contrary to these works, our hybridization paradigm is entirely inspired by an unexpected hunting synergy observed in nature. In section 2.3, we will detail our contribution in which we will explain in detail our hybridization strategy inspired from the hunting synergy between the crows of and the wolves which has allowed us to design HHO-EAS.

2.3. Our contribution HHO-EAS a new population based metaheuristic inspired from the nature

2.3.1. Inspiration from the nature

In northern Europe, near Romania and in the Yellowstone Park in Wyoming, winters are very harsh. During this season sufficiently nutritious prey is scarce and the chances of survival are dwindling for predators. It was in this hostile environment that an unusual alliance was born between the wolves and the crows [Stahler D et al., 2002; Milner R, 2020].

Fig. 2.2. Synergy between crow and wolf while hunting [Milner R, 2020].

Individually, the crows and wolves struggle to feed themselves by hunting in such difficult climatic conditions.

In the snow, wolves have a good ability to hunt by detecting and catching a prey near them, even if it is larger than them. However, they have great difficulty in locating their prey in a large hunting ground, especially if the hunting ground is very hilly and rugged. They will therefore have exhausted themselves trying to catch a prey in vain. Then, they give up due to excessive complexity of the hunt, and consequently reduces their chances of survival.

Crows, unlike wolves, have a good aerial view on the entire hunting ground. This makes it easier for them to quickly explore the hunting ground and to detect attractive prey. However, attractive preys are often too big for their hunting ability. Crows therefore cannot catch them away. Consequently, they will have exhausted themselves in vain in search of this prey. Thus, they will suffer the same fate as the wolves.

However, if the crows and wolves associate their skills, their desperate situation could favorably change. Crows will use their aerial exploration faculties to detect promising prey, even if stronger than them. As soon as they will have detected a good prey they will communicate his position to the wolves

and ask them to exploit this information to attack and catch this prey. Once the prey is acquired, the wolves honorably cedes a part of the booty to the crows.

This win-win synergy between the aerial exploration abilities of the crows and the attack skills of the wolves allows them both to better feed themselves during the winter and ensure their survival.

The hunting strategy between crows and wolves is structured in three phases.

First phase: The crows initiate the hunt and invite the wolves to join them. The crows perform a careful exploration adapted to the hunting ground whatever its complexity. The initial field of exploration is wide and then gradually narrows to an area with a promising prey. So the exploration phases are present until the crows have targeted a promising prey. The wolves' encirclement and attack phase is in the last phase of the hunt. During the exploration phases, crows can ask wolves to check the feasibility of the attack and the capture of a prey.

Second phase: Finally, the crows' exploration abilities allowed them to detect a promising region with a good prey. They ask wolves to attack this prey. The wolves move towards the position of the prey provided by the crows to encircle and attack it.

Third phase: Wolves and crows benefit together of the outcome of their alliance by sharing the prey.

The hunting synergy between the crows and the wolves brings out four key points of the adopted strategy:

- A good management of hunting by the crows and a constant communication between crows and wolves;
- A smart exploration, gradual and adapted to the hunting ground carried out by the crows in order to find a good prey;
- A signal that allows to coordinate the missions of each during the hunt and asking the wolves to attack the prey;
- An encirclement and attack of the prey made by the wolves.

Based on the three phases of the hunting synergy between crows and wolves and its four key points we were able to enforce a hybridization strategy to design a new HHO-EAS metaheuristic.

2.3.2. *Our contribution HHO-EAS*

Subsection 2.2.5 explained some researches to improve HHO by hybridization. These improvements act mainly on exploratory and exploitation equations and on the alteration of the REE function.

The HHO-EAS metaheuristic provides news axis of improvement for exploration and exploitation in order to improve the performance of HHO. Inspired by the hunting synergy between crows and wolves, HHO-EAS integrates the four key points listed above in HHO:

- A new REE function designed by fuzzy logic techniques that provides a smart distribution of exploration and exploitation phases;
- Progressive and decreasing exploration phases, better suited to highly multimodal problems thanks to fuzzy logic;
- A new Harris Hawk hierarchy, $\alpha - \beta - \delta$;
- A coordinated exploitation between Harris Hawk γ and the Harris Hawk hierarchy, $\alpha - \beta - \delta$, who lead the encirclement and attack.

A. A smart distribution of exploration and exploitation phases managed with a Fuzzy Inference Sytem (FIS)

In subsection 2.3.1, three of the four points of the hunting synergy between the crows and the wolves are:

- A good management of hunting by the crows and a constant communication between crows and wolves;
- An intelligent exploration, gradual and adapted to the hunting ground in order to find a good prey;
- A signal that allows to coordinate the missions of each during the hunt and asking the wolves to attack the prey.

We will model this strategy by creating a Fuzzy Inference System (FIS) based on a Mamdani type. We name it ***FISREE*** (*Fuzzy Inference System Rabbit Escaping Energy*). Our algorithm manages the amplitude e of REE, so it coordinates the exploration and exploitation phases of HHO-EAS. Unlike HHO that coordinates the exploration and exploitation phases at each iteration with a simplistic linear function,

our algorithm will implement a smart coordination process of exploration and exploitation phases suited to highly multimodal and high-dimensional optimization problems. As we could see in subsection 2.2.4, the REE function is the beating heart of the HHO algorithm. This function makes it possible to coordinate the exploration and exploitation phases while ensuring a smooth transition and a balance between its two phases. It therefore contributes to performance of HHO. However, as explained in subsection 2.2.5, REE can be improved to better promote the distribution of exploration and exploitation phases during iterations and to tend towards a better exploration-exploration balance. Thus this allows to increase the probability of finding a promising area in the search space containing a good solution and to avoid the pitfalls of a poor exploration-exploitation balance, either premature convergence or too slow convergence.

In HHO, the amplitude of REE is the crisp variable e and is expressed over two phases in a decreasing and linear fashion :

$$\begin{cases} if\ t\ \in \left[0, \frac{Tmax}{2}\right], e \in [1,2] \\ if\ t\ \in \left[\frac{Tmax}{2}, Tmax\right], e \in [0,1] \end{cases}$$

In $\left[0, \frac{Tmax}{2}\right]$, the amplitude e allows several exploration phases and in $\left[\frac{Tmax}{2}, Tmax\right]$ there's no more exploration phases. The second phase is therefore suddenly deprived of exploration phases. We can easily deduce that in HHO, there is a premature disappearance of the exploration phases, which is not adapted to the high-dimensional and highly multimodal optimization problems [Jia H et al., 2019; Zhang Y et al., 2020]. However, in the collaborative hunting between the crows and the wolves, there is an intelligent exploration, gradual and adapted to the hunting ground to find a prey. Indeed, the crows gradually decrease their fields of exploration to manage to detect a promising area. In this strategy the exploration phases constitute the major part of the hunting process and the wolf attack phase, with only exploitation phases, is the very last part of the hunt. To model this strategy in HHO-EAS, with ***FISREE*** we insert in the amplitude e three level: ***Small***, ***Medium*** and ***Big***.

Before detailing the architecture of ***FISREE***, it is necessary to recall the general concepts of fuzzy logic and to fix its theoretical fundamentals. These fundamentals are 8 in number: Universe of discourse,

Fuzzy set, Membership functions, Fuzzy Rules, Fuzzification, Defuzzification, Knowledge base and Inference Engine [Zadeh LA, 1965; Klir G et al., 1995; Sabri N et al., 2013; Mitiku T et al., 2018; Pourjavad E et al., 2019].

B. General concepts

Fuzzy logic is a subset of the artificial intelligence created by ZADEH Lotfi in 1965 [Zadeh LA, 1965]. Fuzzy logic makes it possible to process physical signals from our environment by the decision-making process of human perception which may lack precision. Mathematically, Fuzzy logic enables rational, flexible and repeatable decisions to be made about crisp output variables based on crisp input variables and fuzzy rules between input and output linguistic variables [Klir G et al., 1995; Ross TJ, 2016; Reddy PVS, 2021].

C. Theoretical fundamentals

a. Universe of discourse

Universe of discourse χ of a fuzzy set L is the interval of definition of the crisp values x in which membership function μ^L is defined.

b. Fuzzy set

A Fuzzy set L, defined in the universe of discourse χ is characterized by its membership function μ^L. It is represented by the couple x et $\mu(x)^L$, with x a crisp value in the universe of discourse χ. The mathematical expression of the fuzzy set L is :

$$\boldsymbol{L = \{(x, \mu(x)^L) |\, x \in \chi\}}\ .$$

c. Membership function

A membership function μ^L is associated with the fuzzy set L . For each crisp value x of the universe of discourse χ, μ^L provide a real value in $[0, 1]$ which represents the degree of membership of x in the fuzzy set L.

The membership μ^L is defined by the mathematical expression : $\begin{cases} \boldsymbol{\chi \rightarrow [0, 1]} \\ \boldsymbol{x \rightarrow \mu^L(x)} \end{cases}$

There is a multitude of membership functions: Gaussian, sigmoid, trapezoid, triangular, etc. What all of its membership functions have in common is that their values are in $[0,1]$. Membership functions must agree with the definition space of the ***crisp input*** and ***crisp output*** value that they describe.

d. Fuzzy rules

Fuzzy rules are the expression of the expert human knowledge. This makes it possible to express, in human language, a fuzzy logical relationship between input and output linguistic variables and input and output fuzzy sets.

A fuzzy rule is expressed by the basic logical expressions ***IF antecedent THEN consequent***. The ***antecedent*** is the fuzzy logic equation involving the input linguistic variables and input fuzzy sets. The ***consequent*** is the fuzzy logic equation involving the output linguistic variables and output fuzzy sets. Knowing that the fuzzy sets represent linguistic values of the linguistic variables.

e. Fuzzification

The Fuzzification process is the mapping of a crisp value in a Fuzzy Set. Thus, the Fuzzification makes it possible to transform a crisp input value, from the input universe of discourse, in membership degrees of the fuzzy sets.

f. Defuzzification

The Defuzzification is the opposite process of Fuzzification. It is the process we use to convert a fuzzy set, taking into account the activated fuzzy rules, in order to provide a crisp output value belonging to the output universe of discourse.

g. Knowledge base

The Knowledge base is a data base containing the expert knowledge of the Fuzzy Inference System. It contains the Fuzzy rules and the Membership function associated to the Fuzzy sets.

h. Inference Engine

Inference engine allow to create the output fuzzy set from the input fuzzy set and the fuzzy knowledge base containing the fuzzy rules. To do this, the Inference Engine acts in two steps:

1st step: Knowing the degree to which the inputs crisp values belong to each membership function in the Fuzzification process, the Inference Engine calculates the degree for which each rule is fired;

2nd step: From the triggered rules and their degree of firing, the Inference Engine creates the output fuzzy set concerned by the rules triggered.

We now know the fundamental blocks that will allow us to build our ***FISREE***. We can now explain step by step its implementation.

D. Variables and membership functions of FISREE

In order to design and to develop the architecture of ***FISREE***, it is necessary to first define its input and output variables and its input and output membership functions. The choices made for each of these elements are fundamental for the efficiency of ***FISREE***.

We describe below the design choices of ***FISREE***:

- The Input and Crisp output value;
- The Output universe of discourse, knowing that input universe of discourse for ***Iteration*** is [0,1];
- The Input and Output fuzzy set;
- The Input and Output Membership functions.

a. Input and crisp output value

Like HHO that uses the iteration as numerical input variable for the function REE (2.1) to pace exploration and exploitation, HHO-EAS will use the iteration as a crisp input value to pace ***FISREE***. In order to have an interval of value between 0 and 1 we divide the iteration by the maximum number of iteration : $\frac{t}{T_{max}}$.

It is the amplitude e that is the crisp output value calculated by ***FISREE***.

Next, we need to determine for ***FISREE***, the bounds of the output universe of discourse the most optimal in order to model the crow exploration strategy.

b. Output universe of discourse

In order to determine which is the most optimal output universe of discourse for ***FISREE***, we performed the efficiency evaluation of the six following universe of discourse:

$[0,2], [0.5,2], [0,2.5], [0.5,2.5], [0.5,3]$ and $[0,3]$. These universes of discourse were determined after analyzing the state of the art of the attempts to improve HHO by acting on the amplitude of REE [7,16,17,20]. Each universe of discourse acts differently on the employability of REE and therefore on the distribution of exploration and exploitation phases. The output universe of discourse most consistent with the strategy of crows with a gradual decline in the field of exploration is $[0.5,2.5]$. It was therefore chosen as the output interval of discourse for the amplitude e.

c. Input and Output linguistic variables

The input linguistic variable ***Iteration*** has three input fuzzy set linked to the three linguistic values: ***Start, Middle*** and ***End***.

As ***Iteration***, the output linguistic variable e has three fuzzy set linked to the linguistic values: ***Small, Medium*** and ***Big.***

d. Membership functions

The choice of membership functions is vital for the ***FISREE***. They act not only on ***FISREE*** performance but also on the calculation time of the output value e.

For ***FISREE*** we want membership functions:

- Which are compatible with large variations of ***input*** and ***output*** numeric variable in a very short time interval;
- Requiring a short calculation time in order to not to increase the HHO-EAS complexity.

Based on the state of the art [Zhao J et al., 2002; Monicka JG et al., 2011] the membership functions adapted to our needs are triangular or trapezoid functions. The most efficient for ***FISREE*** are the triangular ones.

e. Triangular membership functions

The triangular membership function is normal (there is at least one x belonging to the universe of discourse such as $\mu(x) = 1$), symetric and convex fuzzy set. It is defined by the equations (2.16) depending on three real values *a*, *b* and *c* with $a < b < c$.

$$\mu(x) = \begin{cases} 0 \; if \; x \leq a \\ \frac{x-a}{b-a} \; if \; a < x \leq b \\ \frac{c-x}{c-b} \; if \; b < x \leq c \\ 0 \; if \; c \leq x \end{cases} \tag{2.16}$$

The real values a, b and c define the **Support**, the **Core** and the **Boundary** of the membership function:

- **Support** is $\{(x \in [a\,, c])|\; \mu(x) > 0\}$;
- **Core** is $\{(x \in [a\,, c])|\; \mu(x) = 1\}$, for a triangular membership function the core is equal to b;
- **Boundary** is $\{(x \in [a\,, c]): 0 < \mu(x) < 1\}$;

Therefore, for the triangular membership function, b is the *argmax* value in $[a\,, c]$ for $\mu(x)$.

Thanks to the results obtained in [Zhao J et al., 2002; Monicka JG et al., 2011], we selected and tested two membership functions with different linguistic weighting terms $[a\,, b, c]$ in a universe of discourse normalized $[0\,, 1]$:

(1). $[0\,, 0, 0.5], [0\,, 0.5, 1], [0.5, 1, 1]$;

(2). $[0\,, 0, 0.4], [0.1\,, 0.5, 0.9], [0.6, 1, 1]$.

We have chosen (2) because, for the progressive decrease of the exploration phases, this allows on the three portions of the input universe of discourse: $\left[0, \frac{T_{max}}{10}\right]$, $[\frac{2T_{max}}{5}, \frac{3T_{max}}{5}]$, and $[\frac{9T_{max}}{10}, T_{max}]$ to obtain three constant levels of amplitude e followed or preceded by non-linear progressive decrease of the amplitude. Thus, we have associated to the input and output fuzzy sets three triangular membership functions with:

- ***Crisp input*** value in the universe of discourse $[0\,,1]$;
- ***Crisp output*** value in the universe of discourse $[0.5\,,2.5]$;
- For the linguistic variable ***Iteration*** , the three triangular membership functions have the linguistic weighting terms defined by ***Start*** $[0\,, 0, 0.4]$, ***Middle*** , $[0.1\,, 0.5, 0.9]$, ***End*** $[0.6, 1, 1]$;

- For the amplitude linguistic variable $\boldsymbol{e}$, the three triangular membership functions have the linguistic weighting terms defined by ***Small*** [0.5,0.5,1.3], ***Medium*** [0.7,1.5,2.3], ***Big*** [1.7,2.5,2.5].

Fig. 2.3 provides with the graphical representation of the six input and output membership functions for the linguistic variables ***Iteration*** and $\boldsymbol{e}$.

We now have all the fundamental building blocks to design the ***FISREE's*** architecture.

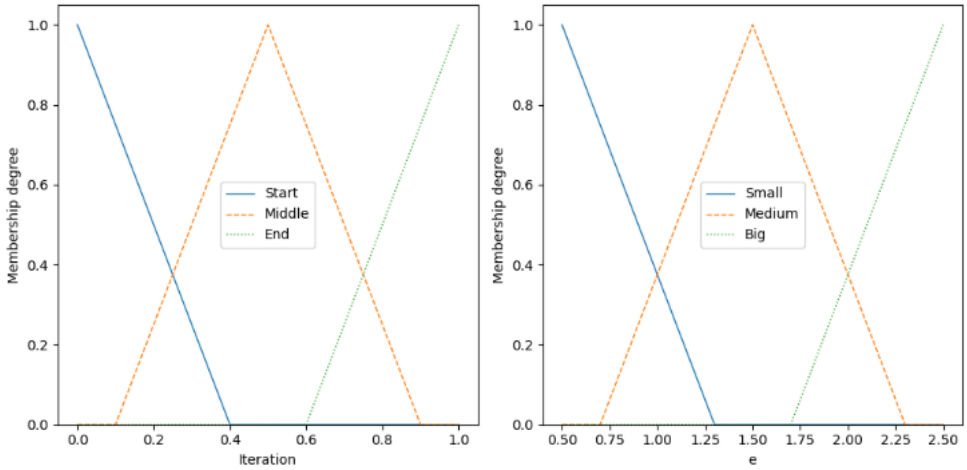


Fig. 2.3. Membership functions for linguistic variable ***Iteration*** and ***e***.

E. FISREE's architecture

FISREE manage the amplitude of REE with five decreasing phases: ***Big***, ***Big and Medium***, ***Medium***, ***Medium and Small*** and ***Small***.

FISREE provide therefore a new distribution of exploration and exploitation phases during Iteration from 0 to T_{max} in order to obtain a better exploration-exploitation balance in highly multimodal and high-dimensional optimization problems and to increase the probability of obtaining an area containing a good solution. This area will then be enhanced by the exploitation phases.

The ***FISREE's*** architecture is composite of the four main blocks of a Fuzzy Inference System. Each of them plays an essential and complementary role in the calculation of the crisp value $\boldsymbol{e}$:

- The Fuzzification block;
- The Knowledge base block;
- The Inference engine block;

- The Deffuzzification block.

Fig. 2.4 below provides the ***FISREE's*** architecture.

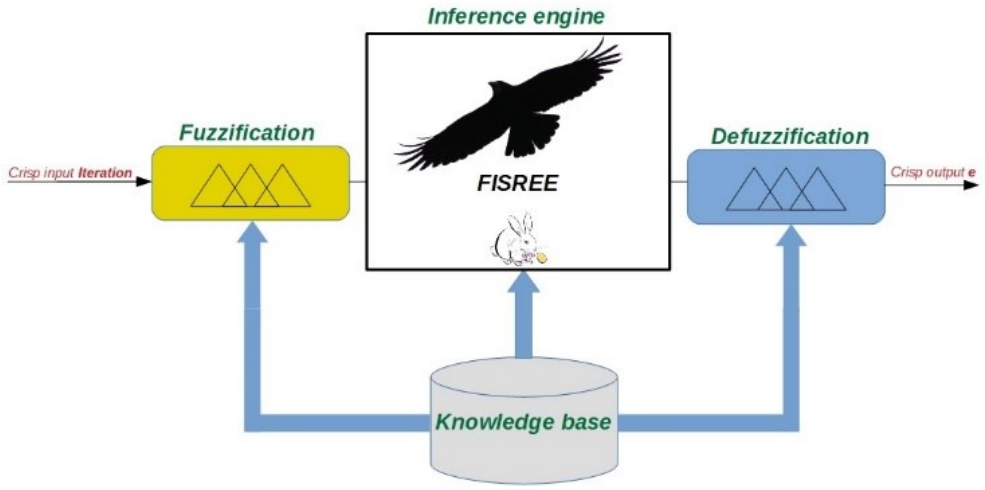


Fig. 2. 4. *FISREE's* architecture.

These four blocks were designed in order to obtain the desired non-linear decreasing monotonous variations for the amplitude e.

The objective is to obtain, like the exploration strategy of the hunting ground by the crows, a redistribution of gradually decreasing and balanced exploration phases compared to exploitation phases, until the final step of the optimization process with only exploitation phases.

a. Fuzzification block

The Fuzzification block receives the crisp input value *Iteration*. This block transforms it into a linguistic variable ***Iteration*** with different degree of membership associated to the fuzzy sets ***Start***, ***Middle*** and ***End*** thank to the three triangular membership functions detailed above and loaded in the Knowledge Base block.

b. Knowledge Base block

As clarified above, the fuzzy rules and the triangular membership functions express the knowledge base that reflects the human expert choices during the execution of the HHO-EAS metaheuristic. We detailed previously the input and output triangular membership functions. Never the less we have not explained our "fuzzy strategy" expressed by our fuzzy rules.

FISREE has three fuzzy rules. In order to create a decreasing amplitude e (crisp output variable) we implement three decreasing rules between the linguistic variables ***Iteration*** and e :

IF (Iteration) IS (Start) THEN (e) is (Big)

IF (Iteration) IS (Middle) THEN (e) is (Medium)

IF (Iteration) IS (End) THEN (e) is (Small)

These three rules make it possible to determine the linguistic value e from the linguistic value ***Iteration*** .

These three fuzzy rules will be activated by the ***Inference engine block*** taking into account the membership degree of the crisp value associated to each fuzzy set ***Start, Middle*** and ***End.***

c. Inference engine block

Inference engine allow to create the fuzzy set of the linguistic variable e from the fuzzy set of the linguistic variable ***Iteration,*** its three triangular membership functions and the three fuzzy rules in the knowledge base.

The results from the fuzzy rules activations are then aggregated and transmitted to the ***Defuzzification*** block.

d. Defuzzification block

For the Defuzzification process in ***FISREE***, with a Mamdani-type FIS, we use the centroid method to provide a amplitude value e. This method use the weighted average of the output fuzzy sets expressed by:

$$e = \frac{\int \mu^L(y).\, y.\, dy}{\int \mu^L(y).\, dy}$$

The ***FISREE's*** architecture is now operational. We can therefore visualize the results on the amplitude e, the REE function and the distribution of exploration and exploitation phases from the iteration 0 to T_{max} with $T_{max} = 600$ and $T_{max} = 10000$ in accordance with the T_{max} values used in general and

specific tests. We will see that ***FISREE*** makes it possible to redistribute the exploration and exploitation phases independently of T_{max} in order to model the smart strategy of the crows with the progressive decreasing of exploration phases.

F. Graphical results

a. Evolution of the amplitude e during the iterations

Fig. 2.5 allows us to visualize the progression of the numeric variable e for HHO and HHO-EAS.

We can see in Fig. 2.5, whatever T_{max} value, that unlike HHO which has a single decreasing linear management of e, ***FISREE*** allows HHO-EAS to manage the numeric variable e on five phases: Three constant levels at the beginning in the middle and at the end of the iterations, preceded by nonlinear decreasing phases.

Each phases represents the following fuzzy sets of e:

Phase 1: ***Big;***

Phase 2: ***Big and Medium;***

Phase 3: ***Medium;***

Phase 4: ***Small and Medium;***

Phase 5: ***Small.***

This is the result of the operations carried out by ***FISREE*** from the crisp input value of *Iteration*. It can be noted that HHO-EAS is still able to perform exploration phases after $\frac{T_{max}}{2}$. These five variations of the amplitude e allow HHO-EAS to have a smart and adapted management of the REE function to the highly multimodal and high-dimensional optimization problems.

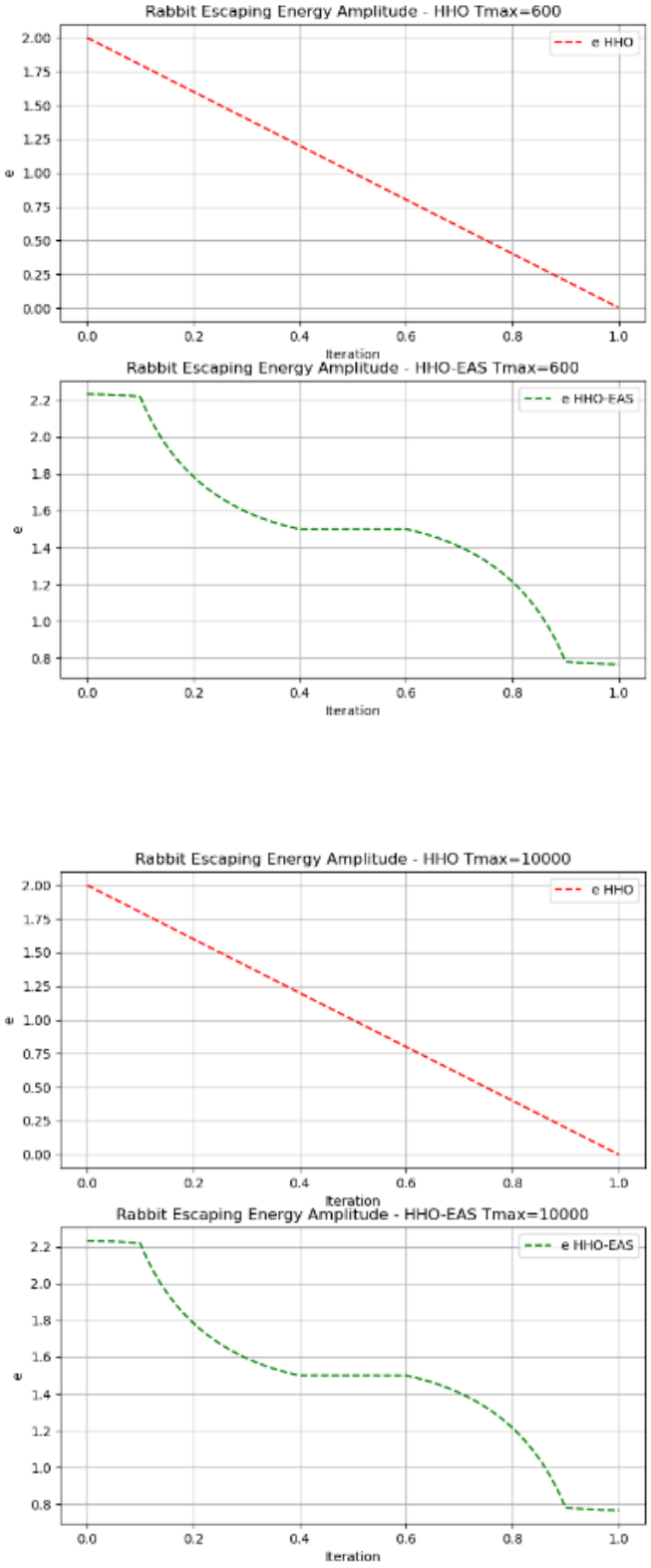


Fig. 2.5. Graphical representation of the amplitude e in HHO and HHO-EAS for T_{max}=600 and T_{max}=10000

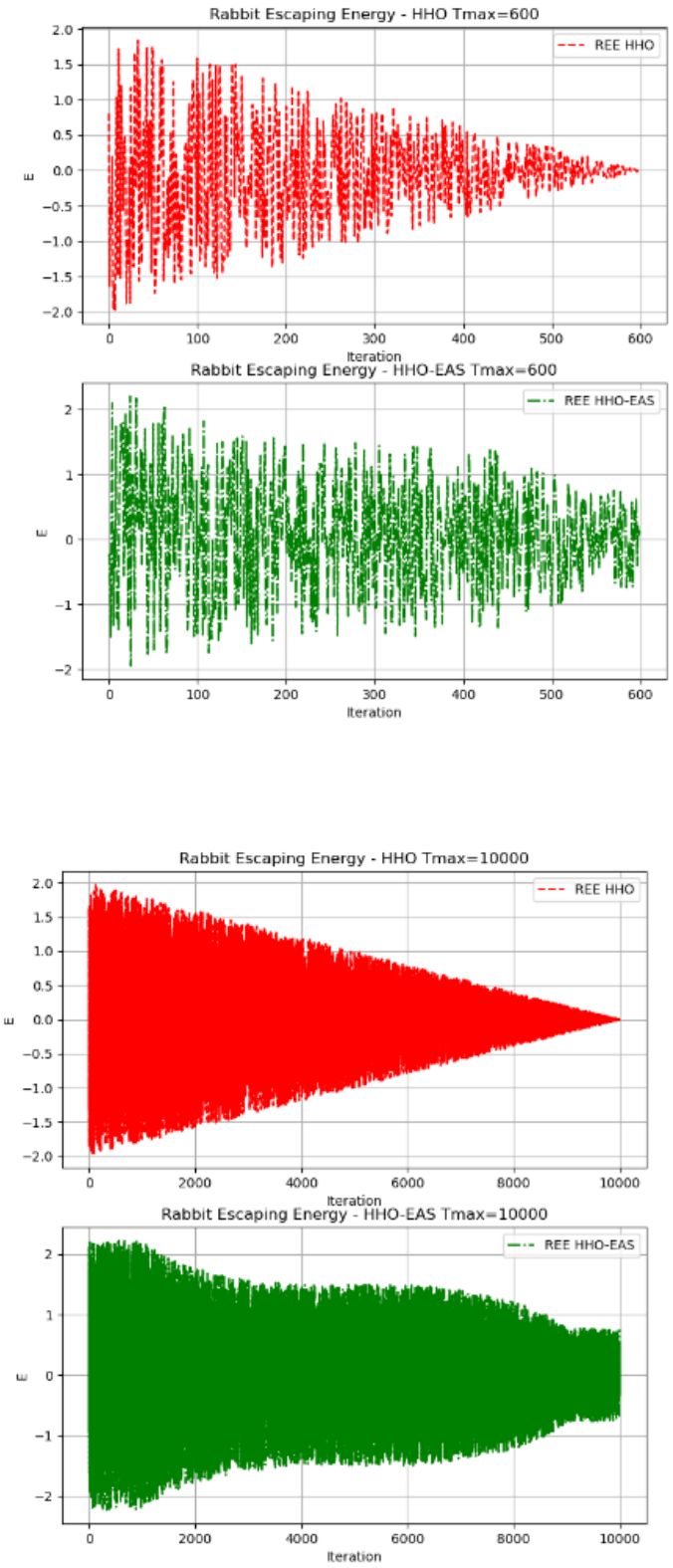


Fig. 2.6. Graphical representation of REE function in HHO and HHO-EAS for T_{max}=600 and T_{max}=10000

b. Evolution of the REE function during the Iteration

To analyze the evolution of REE in HHO-EAS, we replace in the equation (2.1) the linear decreasing amplitude *e* by the output crisp value *e* produced by ***FISREE***.

Fig. 2.6 make it possible to compare the REE managed by HHO and HHO-EAS. We can see that the wrap of the REE function is the same for $T_{max} = 600$ and $T_{max} = 10000$. Which confirms that the variation of the amplitude *e* is independent of T_{max}.

As we had expected, we observe several peaks of amplitude greater than 1 up to iteration 540 for $T_{max} = 600$ and 9000 for $T_{max} = 10000$. After 540 and 9000, there is no more peak greater than 1, leaving the field open for the exploitation phases only. Like the hunting synergy between the crows and the wolves, the exploration phases are present in the major part of the optimization process and end with only the exploitation phases in the last phase of the optimization.

We will see in what follows that ***FISREE*** carries out a new distribution of the exploration and exploitation phases in accordance with the crows' exploration strategy.

c. Exploration and exploitation distributrion

In order to model the crows' exploration strategy, ***FISREEE*** provide a new distribution of the exploration and exploitation phases in order to increase, for the highly multimodal and high-dimensional optimization problems, the probability of finding a promising area with good solution and avoiding to be trapped in a local optimum.

In accordance with our ***FISREE*** conception choices explained above, the latter performs a distribution in the five following intervals: $\left[0, \frac{T_{max}}{10}\right]$, $[\frac{T_{max}}{10}, \frac{2T_{max}}{5}]$, $[\frac{2T_{max}}{5}, \frac{3T_{max}}{5}]$, $[\frac{3T_{max}}{5}, \frac{9T_{max}}{10}]$ and $[\frac{9T_{max}}{10}, T_{max}]$. These intervals represent respectively the linguistic values: ***Start, Start and Middle, Middle***, ***Middle and End*** and ***End***. Whatever T_{max} the fuzzy inference system ***FISREE*** allows us to obtain the same proportion between exploration and exploitation phases in these five intervals. Thus, ***FISREE*** allows to maintain an stable strategy independently of T_{max}.

Knowing that the exploration phases disappear halfway through the iterations for HHO, we have split the interval $[\frac{2T_{max}}{5}, \frac{3T_{max}}{5}]$ into two: $[\frac{2T_{max}}{5}, \frac{T_{max}}{2}]$ and $[\frac{T_{max}}{2}, \frac{3T_{max}}{5}]$. This allows us to observe the total loss of exploration phases in HHO unlike HHO-EAS which maintains its strategy of progressive decrease

in its exploration phases. Of course, ***FISREE*** will keep the same logic of distribution in these two intervals.

So we will analyze in Fig. 2.7 the quotient between the exploration and exploitation phases over the six key intervals: $\left[0, \frac{T_{max}}{10}\right]$, $[\frac{T_{max}}{10}, \frac{2T_{max}}{5}]$, $[\frac{2T_{max}}{5}, \frac{T_{max}}{2}]$, $[\frac{T_{max}}{2}, \frac{3T_{max}}{5}]$, $[\frac{3T_{max}}{5}, \frac{9T_{max}}{10}]$ and $[\frac{9T_{max}}{10}, T_{max}]$ with $T_{max} = 600$ and $T_{max} = 10000$ in accordance with the T_{max} values used in general and specific tests.

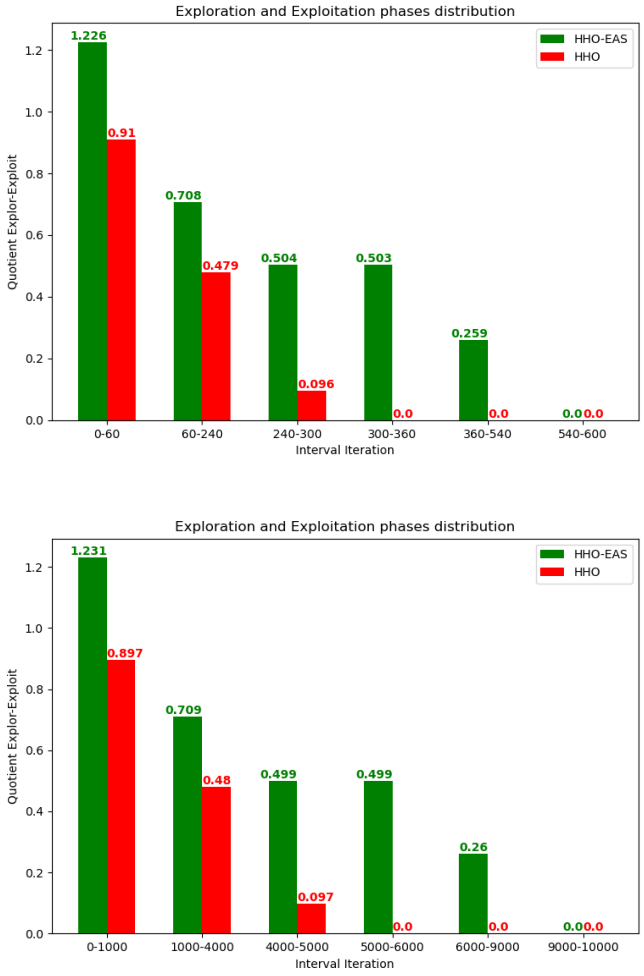


Fig. 2.7. Distribution of Exploration and Exploitation phases over the six key iteration intervals for *Tmax*=600 and *Tmax*=10000

d. Analysis of the distribution exploration and exploitation phases strategy implemented by FISREE

First of all, as we had planned, we can note that the quotients between the exploration and exploitation phases are almost identical for $T_{max} = 600$ and $T_{max} = 10000$. The exploration phases are the majority in the interval $\left[0, \frac{T_{max}}{10}\right]$ for HHO-EAS with a ratio of 1.23 unlike HHO where they are

minority with a ratio of 0.90. This is one of the assets of HHO-EAS compared to HHO because it allows, from the beginning of the optimization process, to cover a greater portion of the search space and thus provide a better investment in the search for promising areas such as the crows at the beginning of the hunt. In addition, the number of exploration phases gradually decreases in the interval $\left[0, \frac{T_{max}}{2}\right]$ while maintaining a ratio greater than 0.5. This allows HHO-EAS to continue to explore new areas of the search space in order to have a better probability of finding promising areas while leaving little by little the field free for exploitation.

For HHO the ratio decreases too brutally in $[\frac{T_{max}}{10}, \frac{T_{max}}{2}]$ from 0.48 to 0.1. If the promising areas have not been detected at the beginning of iterations, HHO will have much more trouble doing it in $[\frac{T_{max}}{10}, \frac{T_{max}}{2}]$.

As expected, HHO no longer provides exploration phases when the iterations are greater than $\frac{T_{max}}{2}$, (300 and 5000 in our case). In contrast, for HHO-EAS the exploration phases represent a ratio of 0.5 in $[\frac{T_{max}}{2}, \frac{3T_{max}}{5}]$ and 0.26 in $[\frac{3T_{max}}{5}, \frac{9T_{max}}{10}]$. These exploration phases could allow to move towards better research areas contrary to HHO. As we wanted, the exploration phases disappear in $[\frac{9T_{max}}{10}, T_{max}]$ with 100% of exploitation phases. This part is the last of the optimization process when the wolves attacks the prey detected by the crows.

So Thanks to our design choices, ***FISREE*** manages to reproduce the crows' exploration strategy followed by the final attack of wolves. Thus HHO-EAS performs a progressive and careful exploration more adapted to the hunting ground of the optimization problem. The initial field of exploration is wide in $\left[0, \frac{T_{max}}{10}\right]$ and then gradually narrows toward an area with promising prey, to conclude with the final attack of wolves in $[\frac{9T_{max}}{10}, T_{max}]$ with 100% of exploitation phases.

Other design choices in fuzzy logic are certainly possible for modeling other exploration strategy model. The choices that have been taken to create ***FISREE*** all come from an analysis of the state of the art with a dual objective: to best model the crows' exploration strategy of the hunting ground and minimize the complexity of ***FISREE*** so not to weigh down on the execution of the algorithm HHO-EAS.

In the next subsection, we will explain the wolves attack modeling. In HHO-EAS, this attack will be led not by wolves but by a new hierarchy of Harris Hawk $\alpha - \beta - \delta$. It's this new hierarchy which

pilots the Harris Hawks' attack in the exploitation phases. This model is based on the encirclement and attack equations.

G. Exploitation improvement with the encirclement and attack equations

The second part of the hybridization models the encirclement and attack of the prey by the wolves after the exploration driven by the crows. After the detection of a promising area with a potential prey, the wolves encircle, attack and share the prey with the crows in the last phase of the hunt. Unlike other methods aimed at orienting, at each iteration, the metaheuristics towards the most appropriate dimension such as sensitivity analysis [Loubiere P et al., 2016; Loubiere P, 2016], the encirclement and attack equations act on all dimensions in order to position the agents in a portion of the search space more promising. This hybridization improves the quality of the local search even if the optimization problem is high-dimensional and highly multimodal. Indeed, in this category of optimization problem, for the population based metaheuristics, the agents tend to gather around the agent having the best position even if it is in a local optimum. This can therefore generate premature convergence towards this local optimum. With this second hybridization, our ambition is to avoid this defect, to position Harris Hawks in more promising areas and then catch "a good prey".

This hybridization is high-level and precedes the exploitation phases of HHO, with a subordinated execution to ***FISREE*** when the crisp output value is in $[-1, 1]$.

Like the crows that can't hunt alone of good prey in the exploitation phases without the help of the wolves, the Harris Hawk agents γ will benefit from the help of the hierarchy Harris Hawk agents $\alpha - \beta - \delta$ to obtain improved exploitation phases with solutions of better qualities.

In order to models the action of the wolves in this win-win hunting synergy, HHO-EAS acts on two axes:

- We have completed the organization of the Harris Hawk pack by integrating their hierarchy $\alpha - \beta - \delta$ observed in nature [Dawson JW et al., 1991] ;
- We inserted the encirclement and attack equations synchronized on the crisp output value provided by ***FISREE*** which also ensures the magnitude of the encirclement around the $\alpha - \beta - \delta$ Harris Hawks.

a. General concepts of encirclement and Attack equations

Several metaheuristics inspired by predators' hunting techniques model their strategy with the encirclement and attack equations: Sea Lion Optimization [Masadeh R et al., 2019], Whale Optimization [Mirjalili S et al., 2016], Spotted Hyena Optimizer [Dhiman G et al., 2017] and Gray Wolf Optimization [Mirjalili S et al., 2014].

The main effect of these equations is to create virtuous encirclement areas in the neighborhood of the top three solutions $\alpha - \beta - \delta$ and to attack them by repositioning the other agents in a more promising area consisting of a linear combination of three new positions. Each of these three new positions belong to one of the three areas around $\alpha - \beta - \delta$. Thereby, the processes of encirclement and attack guide, on all dimensions, the agents toward better positions in the exploitation phases.

In the literature the encirclement and attack equations have produced very good experimental results in the exploitation phases [Mirjalili S et al., 2014; Zhang X et al., 2020a]. On the other hand, the results in the exploration phases are very mixed or even mediocre for high-dimensional and highly multimodal optimization problems. However, in HHO-EAS the encirclement and attack equations have the only mission to improve exploitation phases. They are therefore in adequacy with our goals.

b. Encirclement equations

The encirclement equations are described by the equations (2.17) and (2.18).

$$\begin{cases} \vec{S} = \left| K\vec{V}.\overrightarrow{X_p}(t) - \vec{X}(t) \right| & (2.17) \\ \vec{X}(t) = \overrightarrow{X_p}(t) - \overrightarrow{W}.\vec{S} & (2.18) \end{cases}$$

- K is a non-zero integer;
- $\vec{V}$ is a vector whose the components are uniform random values defined in $[0,1]$;
- $\overrightarrow{X_p}(t)$ is the prey position;
- $\overrightarrow{X}(t)$ is the predator position;
- $\overrightarrow{W}$ is a vector whose components evolve randomly in the interval $[-w, w]$. Generally $\overrightarrow{W}$ is defined by the combination of a random variable and a dependent function of the iteration.

Equation (2.17) creates a vector $\vec{S}$ whose components represent the distances between each component of the vectors $K\vec{V}.\overrightarrow{X_p}(t)$ and $\vec{X}(t)$.

$K\vec{V}$ models the uncertainties of the hunting ground in order to access the prey. The hunting ground can keep the prey away from predators if the ground is too rough and too hilly. Thus the more the distance between $K\vec{V}.\overrightarrow{X_p}(t)$ and $\vec{X}(t)$ increases, the more the predator is likely to be away from the prey. Conversely, the more this distance decreases, the closer the predator gets to the prey. Mathematically the components of the vector $\vec{S}$ sizes the stride of the movement and therefore the speed of predators in the encirclement area around the neighborhood of the prey.

Once $\vec{S}$ and $\overrightarrow{W}$ have been calculated, the encirclement area or hypercube (for a dimension N) around the prey $\overrightarrow{X_p}$ is created by (2.18).

In order to illustrate the action of the encirclement equations on the search for a solution, let us take the simple case of a two-dimensional search space.

c. Encirclement equations for a bidimensional search space

The two components of the vector $\vec{S}$ (2.17) is given by equations (2.19).

$$\begin{cases} S_1 = \left|KV_1.X_{p1}(t) - X_1(t)\right| \\ S_2 = \left|KV_2.X_{p2}(t) - X_2(t)\right| \end{cases} \tag{2.19}$$

KV_1 and KV_2 are random values in $[0, K]$.

Once the components (W_1, W_2) of $\overrightarrow{W}$ are calculated with $(W_1, W_2) \in [-w, w]^2$, thus $W_1.S_1 \in [-w.S_1, w.S_1]$ and $W_2.S_2 \in [-w.S_2, w.S_2]$, the equation (2.20) create the encirclement area around the prey's position $\overrightarrow{X_p}$. The encirclement process is illustrated by Fig. 2.8.

$$\begin{cases} X_1(t) = X_{p1}(t) - W_1.S_1 \\ X_2(t) = X_{p2}(t) - W_2.S_2 \end{cases} \tag{2.20}$$

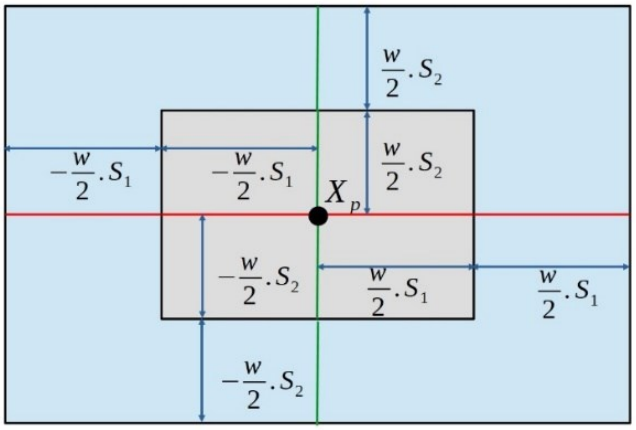


Fig. 2.8. Encirclement area around the prey in a two dimensional search space

The calculation of the new position $\vec{X}(t)$ of the predator is stochastic due to the randomness of the vectors $\vec{V}$ and $\vec{W}$. However it is possible to determine the neighborhood in which $\vec{X}(t)$ will be.

Each metaheuristic using the encirclement equations defines two neighborhoods in the encirclement area around the prey:

- A remote neighborhood (bleu area) for the exploration with $(W_1, W_2) \in \left[-w, -\frac{w}{2}\ [\cup]\frac{w}{2}, w\right]^2$;
- A near neighborhood (grey area) for the exploitation with $(W_1, W_2) \in \left[-\frac{w}{2}, \frac{w}{2}\right]^2$.

The encirclement and attack equations, in HHO-EAS, will only be employed in the grey area for the exploitation.

For example for the metaheuristic GWO [Mirjalili S et al., 2014] , the encirclement and attack equations are illustrated by (2.21):

$$\begin{cases} \vec{W} = 2.f(t).\vec{r} - f(t).\vec{I} \\ f(t) = 2 - 2\left(\frac{It}{Itmax}\right) \\ \vec{r} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}, (r_1, r_2) \in [0,1]^2 \quad and \quad \vec{I} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{cases} \tag{2.21}$$

We detail in what follows the mathematical modeling in HHO-EAS of the encirclement and attack equations driven by the Harris Hawk hierarchy $\alpha - \beta - \delta$.

d. Encirclement and attack equations driven by the Harrys Hawk hierarchy $\alpha - \beta - \delta$

The positions of the Harris Hawk hierarchy $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ will be surrounded, and attacked in order to get a new position in a more promising area. This significantly improve the convergence rate and the quality of the solution as we will see from the experimental results in section 2.4.

$\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ represent the three best Harris Hawk positions. The three best positions $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$ are also the three best prey in descending order of quality with each iteration. For the encirclement equations we have $K = 2$ and $\overrightarrow{W} = \vec{E}$. Each components of the vector $\vec{E}$ are provided by the crisp output value of ***FISREE*** multiplied by a uniform random value provided by equation (2.2). Being in the exploitation phases, their value are in $[-1,1]$. As a result, the attack will be done in the exploitation areas (grey area) around $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$ and $\overrightarrow{X_\delta}$.

The encirclement equations applied to the hierarchy positions X_α, X_β et X_δ provide three new positions in each exploitation area: $\overrightarrow{X_1}$, $\overrightarrow{X_2}$ and $\overrightarrow{X_3}$ with the equations (2.22) to (2.27). The attack equation (2.28) repositions the Harris Hawk position $\vec{X}_i$ at the center of gravity $\overrightarrow{X_e}$ of the three new positions $\overrightarrow{X_1}$, $\overrightarrow{X_2}$ and $\overrightarrow{X_3}$.

$$\overrightarrow{S_\alpha} = \left|K\overrightarrow{V_\alpha}.\overrightarrow{X_\alpha}(t) - \vec{X}_i(t)\right| \quad (2.22)$$

$$\overrightarrow{S_\beta} = \left|K\overrightarrow{V_\beta}.\overrightarrow{X_\beta}(t) - \vec{X}_i(t)\right| \quad (2.23)$$

$$\overrightarrow{S_\delta} = \left|K\overrightarrow{V_\delta}.\overrightarrow{X_\delta}(t) - \vec{X}_i(t)\right| \quad (2.24)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha}(t) - \vec{E}.\overrightarrow{S_\alpha} \quad (2.25)$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta}(t) - \vec{E}.\overrightarrow{S_\beta} \quad (2.26)$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta}(t) - \vec{E}.\overrightarrow{S_\delta} \quad (2.27)$$

$$\overrightarrow{X_e}(t) = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \quad (2.28)$$

Like the wolves which offers the crows good prey during final attack, the hierarchy $\alpha - \beta - \delta$ with the encirclement and attack equations (2.22) to (2.28) offer a much more attractive position to the Harris

Hawk for a more productive exploitation. Fig. 2.9 illustrate the repositioning of a Harris Hawk agent towards $\overrightarrow{X_e}$.

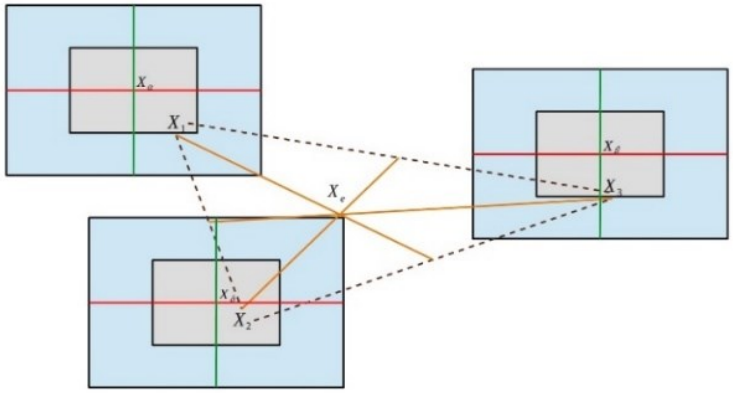


Fig. 2.9. New position resulting from encirclement and attack

H. Computational complexity of HHO-EAS

In order to better compare HHO-EAS and HHO metaheuristics, it is essential to calculate for each of them their computational complexity.

The computational complexity of HHO is based on 3 components of its algorithm: initialization, fitness evaluation and the updating of Harris Hawk positions [Heidari AA et al., 2019]. For a population size *N*, a maximum number of iterations T_{max} and an optimization problem of dimension d, the computational complexity of these 3 components are:

- Initialization : $O(N)$
- Fitness evaluation : $O(T_{max} \times N)$
- Updating of Harris Hawk's position: $O(T_{max} \times N \times d)$

The total computational complexity of HHO is therefore $O(N \times (1 + T_{max} + T_{max} \times d))$.

The computational complexity of HHO-EAS also takes into account the 3 first components that we have just discussed. However, in the updating phase of Harris Hawk positions we need to add the computational complexity of the encirclement and attack equations:

- Encirclement equations: $O(6 \times T_{max} \times N \times d)$
- Attack equation: $O(T_{max} \times N \times d)$

The total computational complexity of HHO-EAS is thereby $O(N \times (1 + T_{max} + 8 \times T_{max} \times d))$

For high-dimensional problems, the computational complexity of HHO-EAS tends to be 8 greater than HHO.

This increase in computational complexity is one of the corollaries of our hybridization strategies also observed in other works aimed at improving HHO [Chen H et al., 2020; Alabool HM et al., 2021].

We now have all the components to model the hunting synergy between crows and wolves in HHO-EAS: The Fuzzy Inference System FISREE for the exploration strategy and the encirclement and attack equations for the exploitation strategy.

Like the crows, the initial field of exploration is wide and then gradually narrows to a promising area by leaving more and more scope for the exploitation phases. In those promising areas, during the exploitation phases, as the wolves, the hierarchy $\alpha - \beta - \delta$ with the encirclement and attack equations, provides a better position to each Harris Hawk and so a better final solution to the optimization problems.

Appendix 7 provides the flowchart of HHO-EAS and the mapping of the updating of the Harris Hawk positions in exploration and exploitation phases. The pseucode Algorithm 2 below implements the HHO-EAS algorithm. Section 2.4 will make it possible to experimentally validate the superiority of HHO-EAS over HHO.

Algorithm 2.2 Pseudocode of HHO-EAS

In: Population size *N*, number of iteration T_{max}, objective function *F*

Out: $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

Initialization of the Harris Hawk positions $\vec{X}_i, i \in [\![1, N]\!]$, $K = 2$, t=0

While (t < T_{max}) **do**:

  Calculate $F(\vec{X}_i)$ of each Harris Hawk position $\vec{X}_i$

  Define the best position of rabbit $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

  For each Harris Hawk i **do**:

    Update e, E_0, E, and J with ***FISREE***, (2.2) and (2.6)

    If($|E| \geq 1$) **then**: *[Exploration]*

      Calculate q and **Update** position with (2.4)

    Else: *[Exploitation]*

      Update $\overrightarrow{V_\alpha}$, $\overrightarrow{V_\beta}$ and $\overrightarrow{V_\delta}$

Define:

$\overrightarrow{X_\alpha}$=First best position

$\overrightarrow{X_\beta}$=Second best position

$\overrightarrow{X_\delta}$=Third best position

Calculate encirclement equations with (2.22), (2.23), (2.24), (2.25), (2.26) and (2.27)

Calculate $\overrightarrow{X_e}$ with attack equation (2.28)

Update Harris Hawk position $\vec{X}_i$ by $\overrightarrow{X_e}$

Calculate r_5

If $|E| \geq 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.6) and (2.7)

If $|E| < 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.8)

If $|E| \geq 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.9) to (2.12)

If $|E| < 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.12) to (2.15)

End For

t=t+1

End While

Return $\overrightarrow{X_r}$

2.4. Experiment and discussion

To examine the performances of the new metaheuristic HHO-EAS, we have set up an analytical environment with all the necessary hardware, software and statistic components:

- An analytical workstation with enough computing power;
- A programming language adapted to scientific calculations and artificial intelligence;
- General and specific benchmarks composed of sufficiently diversified functions in order to validate the design strategy of HHO-EAS and its superiority over HHO;
- Experimental metrics of performance;
- A non-parametric statistical hypothesis Wilcoxon test to validate statistically the experimental results.

2.4.1. Analytical working station setup

We implemented HHO-EAS and did the experimental tests on a computing station with the following technical specifications:

- Operating system: UBUNTU 20.04 LTS 64 bits
- Hardware:
    - SSD 2,0 To
    - RAM 31,3 GoProcessor Intel Xeon 3.50 Ghz, total cores 8 and total threads 16

2.4.2. Programming language

The HHO-EAS metaheuristic has been developed with the Python programming language.

2.4.3. Benchmark functions

In order to validate the superiority of HHO-EAS over HHO, we have carried out general and specific experimental tests on benchmarks with sufficiently diversified optimization problems set.

A. General benchmark

General tests have for ambitions to analyze the convergence behavior, the exploitation and exploration performance of HHO-EAS as well as the balance between exploration and exploitation, knowing that a good balance exploration-exploitation allows the avoidance of local optimums. To do this we have, unimodal, multimodal and compound optimization problems to measure respectively the performances of exploitation, exploration and the search strategy with the ability to maintain a balance between exploitation and exploration. In order to analyze the scalability of HHO-EAS on a dimension spectrum small, medium and high we have performed the tests on a wide range of dimensions: 2, 30, 100, and 1000. Indeed, the increase in the dimension causes the deterioration of performances of any metaheuristic, this is called the curse of the dimensionality.

Regarding test functions, we used a diversified benchmark of 19 well-known functions frequently used in the literature to validate the metaheuristics' performances [Yao X, Liu Y, Lin G, 1999; Digalakis JG et al., 2000; Mirjalili S et al., 2014; Hayashida T et al., 2017; Cortés-Toro EM et al., 2018; Wang GG et al., 2018; Vanaret C et al., 2020; Zhang X et al., 2020a; Zhang X et al., 2020b]. This benchmark

consists of 4 unimodal functions *(U1 –U4*), 13 multimodal functions (*M1-M13*) that have several local optima and 2 composite functions (*C1-C2*).

We have put HHO-EAS in competition with HHO and with two other population-based metaheuristics well-known in the literature and with well-established optimization techniques: GWO [Mirjalili S et al., 2014] and PSO [El-Shorbagy M, Hassanien AE, 2018].

Mathematical formulas and graphical representations of these functions are provided in Appendix 1.

B. Specific benchmark

Unlike general tests, the specific tests focus on HHO-EAS's performances compared to HHO in 20 very complex environments close to real life. To do this, we have exploited the benchmark CEC 2017 [Wu G et al., 2017].

Thus, for the specific tests we used the 20 most complex problems from the CEC 2017: 10 hybrid functions (*F11 - F20*) and 10 compound functions (*F21 - F30*).

These 20 optimization problems make it possible to assess search strategy, the exploration performance and the balance between exploration and exploitation of HHO-EAS in complex environment which is the key to the avoidance of local optimums. In addition, this will allow us to confront our exploration and exploitation strategy inspired from the hunting synergy between wolves and a crows in the real life optimization problems.

Unlike the general tests above, the dimensions of the CEC 2017 functions are limited to six: 2, 10, 20, 30, 50 and 100. Given that we want to validate the performance of HHO-EAS in the most complex environments possible, we have chosen the maximum dimension of CEC 2017: 100. The details of its 20 functions are provided in Appendix 4, Table A4.1.

2.4.4. Metaheuristic parameters

A. General tests

To compare fairly, HHO-EAS, HHO, GWO and PSO on general tests, for the dimension 2, 30, 100 and 1000 we have configured the same spatial and temporal complexity for each metaheuristic. Our experimental parameters are inspired by Heidari AA et al. in [Mirjalili S et al., 2014]. We maintained the same population size than [Mirjalili S et al., 2014] in order to have the same spatial complexity. Nevertheless, this is not the case for the Number of independent executions, the Maximum number of

iteration and the PSO settings. Regarding PSO settings in [Mirjalili S et al., 2014], the authors use a constant inertial factor equal to 0.3 and cognitive and social constants equal to 1. This choice seemed too simplistic to us and could penalize the PSO metaheuristic during experimental tests. We have therefore chosen parameters that can better promote the exploration and exploitation capacity of PSO: a variable inertial factor decreasing linearly from 0.9 to 0.2 during the iterations and cognitive and social constants equal to 1.48. As for the Number of independent executions and the Maximum number of iteration, after several tests we have selected other values which gave us the best results. So we arrived at the following parameters:

- Number of independent executions: 50;
- Population size: 30 (spatial complexity);
- Maximum number of iteration T_{max}: 600 (temporal complexity);
- For PSO: Inertia factor Min wMin= 0.2, Inertia factor Max wMax=0.9, Cognitive coefficient c1=1.48 and Social coefficient c2=1.48.

B. Specific tests

To compare HHO-EAS and HHO we use the same spatial complexity as in the general tests. To take into account the complexity of the functions of the CEC 2017 benchmark, we used a much greater temporal complexity and identical for HHO-EAS and HHO:

- Number of independent executions: 50;
- Population size: 30 (spatial complexity);
- Maximum number of iteration: 10000 (temporal complexity).

In the general and specific tests, for each metaheuristic, the results from each of the 50 independent executions are stored. The comparison between the metaheuristics is founded on the average of these 50 independent results.

2.4.5. Performance metrics

The performance metrics for the assessment of metaheuristics on general and specific benchmarks tests are ***AVG, MIN, MAX*** and ***STD***: Average result, Minimum (Best) result, Maximum (Worst) result and Standard deviation. The best results are in bold in Appendix 2 and 4.

For each of the dimension 2, 30, 100 and 1000, a summary of the results with the signs $+$, $-$ and $\approx$ is provided at the end of each table for the four metrics ***AVG, MIN, MAX*** and ***STD***. So for each test functions, if HHO, GWO or PSO has better results than HHO-EAS on one of the metrics we increment $+$, if the results are worse we increment $-$ and if they are equal we increment $\approx$.

2.4.6. Experimental results on the general benchmark tests

The experimental results on the general benchmark tests are provided in Appendix 2 and 3. They show the exploitation and the exploration performances as well as the local optima avoidance ability of HHO-EAS compared to HHO, GWO and PSO. Appendix 2 details, for the dimensions 2, 30, 100 and 1000, the performance metrics for the 19 functions of the benchmark from Table A2.1 to Table A2.4 and Appendix 3 makes it possible to visualize the convergence curves.

A. Exploitation performances

Unimodal functions *U1* to *U4*, have only one global optimum and therefore help to determine the exploitation performances of HHO-EAS, HHO, GWO and PSO metaheuristics.

The experimental results show that HHO-EAS provide very good results and a suitable convergence rate for unimodal functions from *U1* to *U4* with the dimensions 30, 100 and 1000. These good exploitation performances for medium and large dimensions are due to the encirclement and attack equations. Those equations make it possible to obtain results closer to the global minimum than HHO, GWO and PSO.

On the other hand, for small dimensions such as 2, except for *U3*, GWO has the best results for *U1*, *U2* and *U4*. Which confirms these excellent exploitation capacities for small dimensions. However, for dimension 2, HHO-EAS maintains overall better exploitation performance than HHO.

B. Exploration performances

Unlike unimodal functions, multimodal functions from *M1* to *M13* have global optimums and several local optimums. The curse of the dimensionality increases these local optimums exponentially depending on the dimension. The multimodal functions are therefore good support for evaluating the exploration performance of HHO-EAS, HHO, GWO and PSO.

As we could expect the performance of metaheuristics degrades with the increase of the dimension. With regard to the dimensions, 30, 100 and 1000, from *M1* to *M13*, HHO-EAS surpasses HHO, GWO and PSO or at least demonstrates performances as competitive as HHO. We can note a correct convergence rate for M2, M5, M6, M8, M11 and M13.

This is the experimental demonstration of the effectiveness of ***FISREE*** that implements an efficient exploration and exploitation phases distribution which helps HHO-EAS to limit blockages in local optima and to have a good convergence to a better final solution.

Concerning the dimension 2, PSO has the best results for the multimodal functions *M1, M2, M5, M8*, *M9, M10* and *M11*. For *M3* and *M13,* the four metaheuristics displays equivalent results. This confirms that PSO is able to effectively and fairly quickly explore the search spaces with small dimensions. And for the functions *M6* and *M7*, HHO-EAS displays the best results. Nonetheless, the exploration capabilities of HHO-EAS for the dimension 2 remains overall superior to HHO.

C. Local optima avoidances and balance between exploitation and exploration

The complexity of composite functions *C1* and *C2* is a challenge for HHO-EAS, HHO, GWO and PSO metaheuristics that will test their ability to maintain the balance between exploitation and exploration. Indeed, a good balance between exploitation and exploration will better avoid local optimums. The composite functions *C1* and *C2* will therefore make it possible to evaluate the validity of the distribution of the exploration and exploitation phases implemented by ***FISREE*** and its ability to maintain a good balance between these two phases despite the increase of the dimension.

As with multimodal functions, the performances of metaheuristics degrade with the increase of the dimension. Notwithstanding, the experimental results are without calls. Whatever the dimension, 2, 30, 100 or 1000, for *C1* and *C2*, HHO-EAS surpasses HHO, GWO and PSO with a convergence rate much higher and a much greater ability to avoid local optima. This experimentally demonstrates once again the

validity of the conception strategy of ***FISREE*** which makes it possible to maintain a good balance between exploration and exploitation and particularly better than HHO with better avoidance of local optima.

To conclude, the results on the general benchmark show that our hybridization strategy is not efficient for the small dimensions such as 2. Whether for unimodal or multimodal functions, the encirclement and attack equations and FISREE do not do not allow to erase the weaknesses of HHO although HHO-EAS maintains overall better performances than it. But this observation is absolutely not in contradiction with our hybridization strategy. Our hybridization strategy explained in section 2.3, was mainly designed to increase the performance of HHO for higher dimensions. Indeed, the experimental results for the medium and large dimensions 30, 100 and 1000 confirm the superiority of HHO-EAS over HHO as well as over GWO and PSO. Consequently, these experimental results corroborate the success of our hybridization strategy to obtain a better metaheuristic than HHO for highly multimodal and high-dimensional optimization problems.

In what follow, the specific tests performed on the CEC 2017 benchmark will focus on the performances of HHO-EAS and HHO in the 20 more complex environments of the CEC 2017 and more close to the problems encountered in real life. They will support the results obtained in general tests.

2.4.7. Experimental results on the CEC 2017 specific benchmark tests

As noted above, specific tests aim to make a close-up on the performances of HHO-EAS and HHO in complex environments close to the real life problems. These 20 functions will put to the tests in difficult conditions all the abilities of HHO-EAS and HHO: exploitation, exploration, balance exploitation-exploration, local optimum avoidances and convergence.

These environments consist of the most complex functions in a search space with the largest dimension of the CEC 2017: 10 hybrid functions from *F11* to *F20*, 10 composite functions from *F21* to *F30* and a dimension equal to 100.

Appendix 4 details the performance metrics obtained for these twenty functions and makes it possible to visualize the convergence curves.

Experimental results demonstrate that HHO-EAS has better performances than HHO out of 90% of hybrid functions and 90% of the composite functions: respectively *F11* to *F19*, *F21, F23* to *F30*. However, even for the hybrid function *F20* and the composite function *F22*, HHO-EAS provides competitive results.

We can also note, for the twenty functions *F11* to *F30*, that the convergence rate of HHO-EAS is, from the first iterations, greater than that of HHO. It's thanks to the exploration and exploitation strategy of HHO-EAS. ***FISREE*** pilots the distribution of the exploration and exploitation phases as well as the balance exploration-exploitation that provides areas of the search space more promising that HHO. Then thanks to the encirclement and attack equations the exploitations of the promising areas are better "optimized" to provide a final solution with a better quality in 90% of cases.

All the results we have obtained above in the general tests and the specific tests allow us to validate our hybridization strategy inspired by the hunting synergy between crows and wolves. Those results must now be statistically validated by one of the most used non parametric statistical test: The Wilcoxon tests with the calculation of p-value.

A. Wilcoxon test

In order to analyze whether the performances of HHO-EAS are revealing of its superiority over HHO, GWO and PSO and to validate statistically the experimental results, we have made a paired non parametric statistical Wilcoxon test with 5% significance for obtaining **p-value**. We used the same calculation method as [Gardeux V, 2011] and we have included in these test the **Holm adjustment** method.

For the general tests we ask the null hypothesis **H0g**: "*There is no distinction between the distribution of solutions found by HHO-EAS and the distributions of solutions found by HHO, GWO and PSO*". And for the specific tests we ask the null hypothesis **H0s**: "*There is no distinction between the distribution of solutions found by HHO-EAS and the distributions of solutions found by HHO*".

To do these tests we used the statistical software *R*. The results are provided in Appendix 5, Table A5.1 and Table A5.2. These tables provide respectively the results per pair of Metaheuristic:

- HHO-EAS\HHO, HHO-EAS\GWO and HHO-EAS\PSO for the general tests;

- HHO-EAS\HHO for the specific tests.

B. Wilcoxon tests for general results Table A5.1

The calculated p-value for the dimensions 30, 100 and 1000 are all less than 0.05. We can therefore reject the null hypothesis **H0g** with an error probability of 5% for these dimensions. Regarding the W^+ values, for the dimension 30, 100 and 1000, they are all very large, for the metaheuristics HHO, GWO and PSO. This confirms the conclusions of the general tests: HHO-EAS dominates HHO, GWO and PSO for medium and high dimensional search spaces.

For the dimension 2, the results are different. We have a p-value less than 0.05 for the pairs HHO\HHO-EAS and GWO\HHO-EAS. But this is not the case for PSO\HHO-EAS with a p-value equal to 0.3684. So we have to accept the null hypothesis **H0g** for this pair. This means that for dimension 2, there is no distinction between the distribution of solutions found by HHO-EAS and PSO. Thus, statistically, the performance of HHO-EAS is not different from PSO. These results are consistent with the results observed in the general tests for the dimension 2, since the general results are mitigated between HHO-EAS and PSO.

C. Wilcoxon tests for specific results Table A5.2

The Wilcoxon test on the specific results for the pair HHO\HHO-EAS rejects the null hypothesis **H0s** since it is less than 0.05. In addition, the W^+ value statistically attests of the superiority of HHO-EAS over HHO, so the performances of HHO-EAS are much better than HHO.

2.5. Conclusion

In this chapter we have presented HHO-EAS, a new population based metaheuristic, hybrid of HHO resulting from a hybridization strategy entirely inspired by the hunting synergy observed in nature between the crows and the wolves. The aim of HHO-EAS is to produce better results than HHO in the highly multimodal and high-dimensional optimization problems. HHO-EAS implements a win-win hunting synergy between the good exploration strategy managed by the Fuzzy Inference System ***FISREE*** which models the exploration of the crows and the exploitation strategy of the Encirclement and Attack equations which models the attack technique of the wolves. These equations make it possible,

on the one hand, to prevent premature convergence when agents gather prematurely near a local optimum on the other hand to upgrade the Harris Hawk positions in a more promising area.

The exploitation and the exploration performances as well as the balance between these two phases in HHO-EAS have been evaluated on a general and on a specific benchmark. The general benchmark consists of 19 well known functions: 4 unimodal functions, 13 multimodal functions and 2 composite functions with the dimensions 2, 30, 100 and 1000. The specific tests aim to make a close up on the performance of HHO-EAS and HHO in the 20 most complex environments of the CEC 2017 close to real life. The specific tests consist of the 10 hybrid functions and the 10 composite functions from the CEC 2017 with a dimension equal to 100. The tests on the general benchmark have proved that HHO-EAS is overall superior to HHO whether in the exploration phases with the multimodal functions, the exploitation phases with the unimodal functions and the balance between these two phases with the composite functions. In addition, HHO-EAS has a better convergence rate, the best scalability on all dimensions and generally gives the assurance of having a better quality solution and faster than HHO. On the specific benchmark, in the 20 most complex environments of the CEC 2017, HHO-EAS has shown that its abilities are mainly superior to those of HHO on hybrid and compound functions. Indeed, in these 20 most complex problems of CEC 2017, HHO-EAS demonstrated a success of 90% on hybrid functions and 90% on composite functions. As well as the results obtained on the general benchmark, the specific benchmark makes it possible to experimentally validate the exploration and exploitation strategy implemented in HHO-EAS inspired by the hunting synergy between the crows and the wolves.

The most important result that emerges from our research and which will serve us in the next chapter is that HHO-EAS is better suited than HHO to deal with the real life optimization problems as evidenced by experimental results for dimensions 100 and 1000. Indeed, the real life optimization problems are generally highly multimodal, high-dimensional and NP-Hard such as feature selection optimization problems. Thus, HHO-EAS, combined to Deep Learning algorithms, opens a wide range of perspectives to create computer resource efficient Intrusion Detection Systems (IDS) in IoT environments with excellent attack prediction performance. The next chapter is the concrete application of HHO-EAS to process the NP-hard feature selection optimization problems in a high-dimensional binary search space. The new metaheuristic, named Binary HHO-EAS (BHHO-EAS), have the ambition to achieve results

higher than those obtained in research performed in the same field such as [Too J et al., 2019] or [Gardeux V et al., 2011; Gardeux V, 2011; Gardeux V et al., 2012] who uses a metaheuristic with line search methods for the feature selection. BHHO-EAS will perform the feature selection on a very recent cybersecurity dataset AWID3 which contains modern attacks against Wi-Fi networks. BHHO-EAS will thus make it possible to obtain the most relevant features of the AWID3 dataset and to design an IDS based on Deep Learning algorithms with high detection capabilities and resource-efficient to be embedded in an IoT with limited computing and memory resources.

Chapter 3

Elaboration of an intrusion detection system against 802.11 specific attacks with BHHO-EAS and the Wrapper feature selection method

3.1. Introduction

At this stage of this thesis, the work carried out in chapter 2 allows us to have the HHO-EAS metaheuristic. Its performances in the general and specific benchmarks have demonstrated that it is able to find a good solution in a reasonable time to NP-Hard problems. The skills of HHO-EAS will be decisive in this chapter for the design of an IDS embedded in IoTs with limited computing and memory resources against 802.11 specific attacks: Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)Assoc), Rogue Access Point (Rogue AP), Evil Twin, Key Reinstallation Attack (Krack) and Kr00k. Via HHO-EAS, we will use metaheuristic optimization process for the Wrapper feature selection of the AWID3 dataset (Aegean Wi-Fi Intrusion Dataset 3) [University of the Aegean,]. The Wrapper feature selection method is explicitly defined in subsection 3.4.1.

Works of the same field in the literature, use the Wrapper feature selection method with classical Machine Learning algorithms such as KNN, SVM, OPF, Decision Tree, Random Forest, etc. [Agrawal P et al., 2021]. However, they deprive themselves of Deep Learning algorithms that are much more efficient to conceive a IDS than conventional Machine Learning algorithms. To this end, our contributions in this chapter is threefold:

- We have designed the new metaheuristic BHHO-EAS from the HHO-EAS metaheuristic for the optimization of NP-Hard Wrapper features selection multi-objective problems in a binary search space: maximize the detection capabilities of the IDS against 802.11 specific attacks and minimize features in order to design an embeddable IDS in IoT;
- At the time of writing this manuscript, we are the first to have created and implemented a Wrapper feature selection method that is more advanced and more complex than the classic Wrapper method mentioned above. Our method integrates a Convolutional Neural Network (CNN) combined with the computing power of GPU technology instead of conventional

classical Machine Learning algorithms and CPU technology [Agrawal P et al., 2021; Houssein EH et al., 2022c; Houssein EH et al., 2022a; Houssein EH et al., 2022b];

- We are also the first to have applied a feature selection process by metaheuristic optimization driven by BHHO-EAS on the AWID3 dataset, much more complex than its predecessor AWID2 [University of Aegean, 2014].

Like chapter 2, the work presented in this chapter has also been the subject of a second scientific article submitted for publication in the journal Artificial Intelligence Review.

This chapter is articulated as follows. Section 3.2 provide an overview of the Intrusion Detection Systems (IDS). Having the ambition to exploit our IDS in companies and public administration information systems, we have developed in this section a synthetic specification. This specification will guide not only the conception strategy of the IDS but also the preprocessing of the AWID3 dataset in subsection 3.5.4. Section 3.3 presents the related works which aim to design an IDS using the feature selection with the AWID3 dataset. Section 3.4 explains the design of the new BHHO-EAS metaheuristic to apply our Wrapper feature selection method in the section 3.5 on the AWID3 dataset and to conceive a CNN-IDS. In section 3.6, we will analyze the experimental results. In this experimental section we will provide a technical PoC of our work by embedding and testing a prototype of CNN-IDS, named E-CNN-IDS (Embedded CNN-IDS), in a Raspberry Pi 4 Model B with very limited memory and computing resources. In section 3.7 we will conclude this chapter.

3.2. Overview of Intrusion Detection Systems and specification of the CNN-IDS

To deal with the constantly evolving threats targeting IoTs using Wi-Fi, they must be able to have a smart cyber-defense system allowing the detection of attacks. These systems are called Intrusion Detection System (IDS). Before providing a description of IDSs, it is first necessary to define what an Intrusion is in cybersecurity terms and the main modules that constitute an IDS.

3.2.1. Definition of an Intrusion

An intrusion is defined as an illegitimate access to a system in violation of the Confidentiality, Integrity and Availability (of its resources [Sengupta N et al., 2020]. Thus, an Intrusion jeopardizes the fundamentals of the cybersecurity of an information system. Furthermore, an Intrusion can target not only

networks but also host systems. Consequently, network and host must be defended by an IDS whose constituent modules are adapted to the technical specificities of their environment.

3.2.2. Description of the main modules required for IDSs

An Intrusion Detection System audits a host system and/or the network in order to identify actions that are harmful and antagonistic to the security policy [Drewek-Ossowicka A et al., 2021]. It is therefore an "autonomous process" of intrusion detection which is to find events of violation of security policies or standard security practices in computer networks [Kim K et al., 2018a].

Intrusion detection is based on four main modules as illustrated in Fig. 3.1 [Sengupta N et al., 2020]:

- Collection of system and/or network data;
- Preprocessing of collected data;
- Analysis and recognition of an attack with the intrusion detection engine;
- Actions and logging following the detection of an attack.

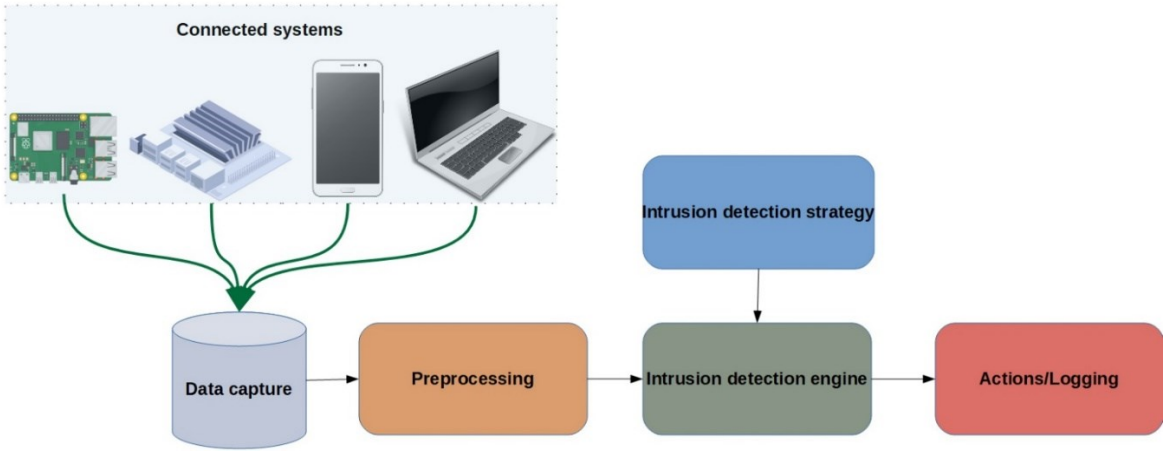


Fig. 3.1. Architecture of the main modules of an IDS

Regarding the action and logging module, the action could be a simple alert sent to the administrator or a more sophisticated action adapted to the detected attack such as a logical isolation from the rest of the network with the IP address (Internet Protocol) or with the MAC address (Medium Access Control) of the compromised agent. In the latter case, the IDS is an IPS (Intrusion Prevention System) because it actively participates in blocking attacks.

Among these four modules, it is the Intrusion detection engine module with its design and detection strategies which are the capital elements of an IDS. The design of this module is the main objective of

this chapter and the data to be processed by this module and the positioning of the IDS depends on the class to which the IDS belongs.

3.2.3. The three main classes of IDS and their detection method

There are three main classes of IDS: Network IDS (NIDS), Hosted IDS (HIDS) and Hybrid IDS [Kim K et al., 2018a; Sengupta N et al., 2020; Drewek-Ossowicka A et al., 2021; Mirjalili S et al., 2023]. The NIDS analyzes the data coming from the network packets captured and made up of several features specific to the protocols used. While the HIDS analyzes data from the host in which it is embedded: network packets captured crossing the host, CPU usage, RAM usage, storage disk usage, temperature, access to system files, etc. The third class of IDS, Hybrid IDS, bring together NIDS and HIDS.

These three IDS classes have two types of detection method: signature-based and anomaly-based [Azizjon M et al., 2020; Jiyeon K et al., 2020; Riyaz B et al., 2020; Park D et al., 2021; Chatzoglou E et al., 2022a]. Signature-based detection uses a signature database describing the pattern of an attack. If an attack conforms to one of the pattern, it will be detected by the IDS. This detection method has the advantage of detecting all the attacks listed in the signature database. However, it has three major drawbacks:

- Unknown attacks (zero-day) are not detected;
- The attack signature database requires regular updates;
- For IoT with limited memory resources, the signature database is highly likely to saturate the memory in the medium term or even in the short term.

Anomaly-based detection does not have these three major drawbacks [Jiyeon K et al., 2020; Riyaz B et al., 2020; Park D et al., 2021]. This detection method analyzes the data in order to detect an abnormal deviation from the states referenced as normal. To do this, the IDS must have been previously trained with reference data (or dataset) labeled with normal states and abnormal states. This detection method has the advantage of detecting unknown attacks (zero-day) and, while being scalable, of not requiring a signature database. However, the major drawback of this method is that it generates some detection errors: false positives and false negatives. Despite this drawback and in view of the problems inherent in signature-based detection, anomaly-based detection is the most suited for our CNN-IDS as well as to

the constantly changing attack detection requirements and hardware constraints of IoT. But determining the method of detection is not enough. As the CNN-IDS is intended to be integrated into company information systems and state administrations, we have developed a defense strategy and a synthetic specification.

3.2.4. CNN-IDS defense strategy and specification

The defense strategy that we have defined and which will guide the design of our IDS is as follows: focus the defense on the Data Link Layer. Indeed, the seven specific attacks 802.11 are initiated from the Physical layer and are finalized at the Data Link Layer [Chatzoglou E et al., 2021; Chatzoglou E et al., 2022a]. Once the attacker has taken over the Data Link Layer, it will be easier for him to escalate the higher layers in order to have greater privileges and a much greater capacity for harm. So if we detect and block the attack at the Data Link Layer, "it will be dead in the bud" and will have no chance of succeeding on the upper layers. Our defense strategy is combined with synthetic specifications we have developed.

The requirements of our specifications were built from the knowledge and experience of information system experts. The proof of the consistency of our specifications is that it is in total agreement with the requirements explained in the state of the art concerning the IDS designed from AWID3 [Chatzoglou E et al., 2022a; Chatzoglou E et al., 2022b]. The specifications are articulated in 5 requirements. These 5 requirements guide not only the conception strategy of the IDS but also the preprocessing of the AWID3 dataset. Therefore, the 5 requirements have been incorporated into the preprocessing of the AWID3 dataset in subsection 3.5.4. The 5 requirements are:

1. **Detection of 802.11 specific attacks**

The IDS must detect known and unknown 802.11 specific attacks in real time and must be sufficiently independent of the technical specificities of the implementation of the attacks in order to better generalize the detection capacities of the IDS.

2. **Compatibility with companies' information systems**

The IDS must be usable on a large majority of company information systems. Consequently, it must not depend on the architecture of the network and the entities that the information system hosts.

3. Hardware adhesion

The IDS must be able to be embedded and be operated in IoTs or STAs (Stations) with heterogeneous hardware specificities and with limited computing and memory capacities.

4. 802.11 frames processing

802.11 frames are of 3 types: management, control and data. The IDS must be able to exploit the 3 types of 802.11 frame for the detection of attacks.

5. Temporal and sequential components

The need to embed the IDS in environments with limited computing and memory resources requires minimizing the processing of superfluous data. Moreover, the embedded IDS is not intended to take time series into account. Consequently, the temporal and sequential features must not be taken into account in the detection of 802.11 attacks.

In this section we have defined the fundamental concepts of IDS. We have also explained our defense strategy as well as our specifications adapted to the imperatives of CNN-IDS and information systems. The section 3.5 will contribute to the design of the CNN-IDS via metaheuristic optimization and Deep Learning algorithms. But first of all, we present in the section 3.3 the related works which aim to design an IDS using the feature selection with the AWID3 dataset.

3.3. Related work

As we specified above, the AWID3 dataset is very recent. Only two articles to date have been published with the objective of creating an IDS from the AWID3 dataset. Also, AWID3 is much more complex than its predecessor AWID2. Indeed, AWID3 was built in a context of company information system architectures with WPA2-Enterprise, frame protection (PMF) with the 802.11w amendment and the 802.11ac standard (or Wi-Fi 5). In [Chatzoglou E et al., 2021], the authors Chatzoglou E. et al., comprehensively provide the architecture of the information system on which AWID3 was built. This information system consists of 17 nodes including 10 IoT terminals, 3 physical or virtual servers with heterogeneous operating systems and 4 active Wi-Fi network management systems:

- **IoT terminals**

- 2 workstations running on Ubuntu 20.04 desktop;
- 5 workstations running on Microsoft Windows 10 Pro or Enterprise;
- 3 smartphones including a Samsung S20 FE running on Android v10, a Samsung Note 4 running on Android v6.0.1 and an iPhone 6s running on iOS v14.2.

- **Servers**
  - 1 Microsoft Windows 2019 servers host the Active Directory (AD) domain controller;
  - 1 Fedora 33 server hosting FreeIPA identity management to use Kerberos authentication, DNS server and Samba server for file sharing via SMB protocol;
  - 1 Ubuntu 20.04 server.
- **Active Wi-Fi network management system**
  - 1 Access Point (AP) ASUZ RT-AC68U;
  - 1 monitor node on Kali Linux v2020.4 operating in Man in the Middle between the STA and the AP. For capturing network packets, it uses Wireshark v3.2.7 for the fixed STA and Mitmproxy for mobile STA;
  - 1 dockerized DVWA for testing attacks on web applications hosted in a Microsoft Azure cloud;
  - 1 FreeRADIUS v3.0.20 server installed on Ubuntu 20.04 server virtual server with an exchange encrypted by 802.1X certificate between this server and its clients.

Furthermore, AWID3 contains modern 802.11 attacks such as Krack and Kr00k, integrates features and attacks on the entire OSI stack from physic layer to application layer. Whereas AWID2 was built on a WEP based infrastructure logic in a standard at home environment with only 802.11 attacks such as ReAssociation or Deauthentication. In addition, to the OSI level, only the physics and data link layers are taken into account in AWID2.

Consequently, it would make no technical and functional sense to provide the related work of IDS built with AWID2 when our work exploits AWID3 to do so. This section will therefore describe the only 2 works existing for AWID3 of Chatzoglou E. et al. [Chatzoglou E et al., 2022a] and [Chatzoglou E et al., 2022b] with the aim of designing an IDS.

Chatzoglou E. et al. in [Chatzoglou E et al., 2022a], designed by supervised learning, IDSs agains t 7 attacks specific 802.11 : *Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)A ssoc), Rogue Access Point (Rogue AP), Evil Twin, Key Reinstallation Attack (Krack)* and *Kr00k*. To do this, the authors performed feature selection on the AWID2 and AWID3 datasets based on the analysis of human experts in 802.11 technology and empirical observations. Those works are divided into 4 main parts.

In the first part the authors selected only features common to AWID2 and AWID3 to create two new ones. The arguments of the authors justifying the choice of features is extremely exhaustive and technically of a very good level. The expert's analysis led to the selection of 16 features common to AWID2 and AWID3 in order to train Machine Learning and Deep Learning algorithms. 6 Machine Learning and 2 Deep Learning models were used: Logistic Regression (LR), LinearSVC, Stochastic Gradient Descent Classifier (SGDClassifier), Light Gradient Boosting Machine (LightGBM), Decision Trees (DT), Random Forest (RF), Extra Trees (ET), Multi-Layer Perceptron (MLP) and Denoising stacked Autoencoders (AE).

Since the authors did not balance their dataset, they mainly relied on the AUC and F1 metrics to evaluate the performance of the 6 Machine Learning models and the 2 Deep Learning models. We define AUC and F1 metrics in subsection 3.6.2 and explain why this choice of metric is relevant. For the sake of completeness, the authors have included the 3 metrics Accuracy, Recall and Precision also explicitly defined in subsection 3.6.2.

The experimental results were split by model family: Machine Learning first and then Deep Learning. On the AWID2 dataset, DT provided the best AUC results for (95.16%) and LightGBM for F1 (95.36%). The worst results were obtained by SGDClassifier for AUC (85.99%) and F1 (84.56%). As for the Deep Learning models, MLP obtained the best performance on AUC (81.79%) and F1 (81.98%) compared to AE which provided mediocre AUC (80.7%) and F1 (81.98%).

On the AWID3 dataset, DT and LightGBM still obtained the best results respectively for AUC (99.49%) and F1 (99.55%). Once again SGDClassifier demonstrated the worst performance for AUC (95.17%) and F1 (96.60%). The ranking of Deep Learning models remains unchanged. MLP had the best AUC (96.47%) and F1 (97.55%) and AE the worst AUC (95.39%) and F1 (96.78%). We can notice

that overall the 6 Machine Learning models and the 2 Deep Learning models had better results with the AWID3 dataset than with the AWID2 dataset. This is largely due to the sample count deficiency in AWID 2 compared to AWID3 that allows models to converge to their optimal performance.

In the second experimental part, the authors selected 3 new features belonging exclusively to AWID3, speculating that they would allow models to better distinguish attack classes. Consequently, only the AWID3 dataset has been used in this part. Here again the authors relied on the knowledge of human experts and their empirical observations. In this part only the Machine Learning ET model and the Deep Learning MLP model were used. ET had the best performance for AUC (99.24%) and F1 (99.37%) and although the performance of MLP increased on AUC (97.23%) and F1 (97.97%) it is unacceptable compared to ET.

In the third part the authors proceeded to the reduction of the subset of the 16 features common to AWID2 and AWID3 hoping to obtain better results. In this part, the authors used only 4 Machine Learning models belonging to the group of tree classifiers: LightGBM, DT, RF and ET. To do this, they constructed 4 feature subsets, Set 1 to Set 4, with a nmber of 4 features. This time the authors, used the feature selection based on empirical observations and human expert knowledge, confirmed by the Feature Permutation Importance technique on the LightGBM and RF models.

For the AWID2 dataset, the ET model had the best performance on AUC and F1 on the 4 subsets with AUC values ranging from 91.67% to 95.03% and F1 ranging from 92.14 to 95.19%. LightGBM obtains the worst results on the 4 subsets, except for Set 2 where it is DT who has the worst F1. For the AWID3 dataset, the DT model wins with an AUC ranging from 91.09% to 97.16% and F1 ranging from 90.04% to 95.19%. On the other hand, unlike AWID2, the ET model had the worst results on the 4 subsets.

Finally, the fourth part aims to analyze the transferability of features within AWID2 and between AWID2 and AWID3 as a test dataset. The authors constructed 4 new datasets using the same feature selection method as in the first three parts: datasets 30 (30F), 27 (27F), 13 (13F) and 5 (5F). Only the 3 classifiers ET, DT and LightGBM, having provided at least once the best performances were chosen for this experimental part. The central question of this part is: "Is the selected set of 16 features directly

transferable between datasets and does it improve the performance of ET, DT and LightGBM classifiers?".

The 30F dataset was created by adding to the AWID2 dataset of 16 features and the 14 most used AWID2 features in the state of the art. This dataset allowed the 3 classifiers to have their best performance compared to the other AWID2 datasets. The following 4 experiments with the 4 AWID2 datasets 30F, 27F, 13F and 5F consist in training the 3 Machine Learning models mentioned above with these AWID2 datasets and then testing them on the AWID3 counterpart. Except for the ET and DT models respectively on the 13F and 5F dataset, the performance of the models at the end of these 4 experiments is very poor. This experimentally confirms our analysis at the beginning of this section concerning the technical and functional difference between the creation environments of AWID2 and AWID3. Thus an IDS created on AWID2 is generally not compatible in a company information system.

Thus the work in [Chatzoglou E et al., 2022a] has made it possible to highlight, for the benefit of the research community, 2 bases of essential information for the design of IDS specific 802.11 and the selection of features on the AWID2 and AWID3 datasets.

If AWID2 was one of the main precursor datasets for the creation of IDS against 802.11 specific attacks by supervised learning, it clearly showed in [Chatzoglou E et al., 2022a] its technical and functional limits compared to AWID3.

On a technical level, the number of samples in AWID2 does not allow the models to converge towards acceptable performance unlike AWID3. The comparison of the results of the 6 Machine Learning models LR, LinearSVC (LSVC), SGDClassifier, LightGBM, DT, RF, ET and of the 2 Deep Learning models MLP, AE with the AWID2 and AWID3 datasets are proof of this. In addition, functionally AWID2 does not take into account the new 802.11 standards, the new protection and security protocol reforms as well as the new 802.11 attacks. This makes all IDS created with AWID2 unsuitable for company information systems.

But beyond the datasets AWID2 and AWID3, Chatzoglou E. et al. demonstrated to us the technical capabilities of feature selection based on empirical observations and human expert knowledge. This selection technique has shown the fullness of its effectiveness on IDSs design with the 16+3 features subset of AWID3 with very good AUC and F1 performance metrics above 99% for the ET model. It is

the only IDS in [Chatzoglou E et al., 2022a] with sufficient credibility to be deployed on a company information system against attacks specific 802.11.

In their second work Chatzoglou, E. et al completed their cyber defense surface by creating an IDS against attacks targeting the application layer.

In [Chatzoglou E et al., 2022b], Chatzoglou E. et al. attempted to design an IDS specific to attacks targeting the application layer using 6 subsets of AWID3 features. The authors exploited the richness of AWID3 which integrates features and attacks from layers higher than the Data link layer.

The IDS focuses on the 6 attacks: *Botnet*, *Malware*, *SSH*, *SQL Injection*, *SSDP amplification* et *Website spoofing*. The authors again used feature selection based on empirical observations and human expert knowledge to create applicative IDSs from subsets of AWID3 features and an artificially created feature:

1. 16 features specific 802.11,
2. 17 non-specific 802.11 features,
3. Reduced set of 6 features 802.11 from the 16 features specific 802.11,
4. Reduced set of 3 features Non-802.11 from the 17 features specific Non-802.11,
5. An artificial feature called *Insider*.

Targeting the correlation between the IP from client station, the *Insider* feature was created by the authors, via Algorithm n° 1 in [Chatzoglou E et al., 2022b], in order to better detect Botnet-type attacks. In this type of attack, the hacker has taken possession of a client station and uses it in server mode for his distributed attacks to other clients.

These groups of features do not incorporate any features of the application layer because these are anonymized or encrypted, therefore unusable by the IDS.

From the features initially selected or created, the authors created by combination 5 subsets of additional features:

1. Reduced set of 6 features 802.11 with set of 3 features Non-802.11,
2. 16 specific 802.11 features with set of 17 non-specific 802.11 features,

3. Reduced set of 6 features with *Insider* feature,
4. Reduced set of 6 features 802.11 with reduced set of 3 features Non-802.11 and with *Insider* feature,
5. 16 specific 802.11 features with set of 17 Non-specific 802.11 features and with *Insider* feature.

To create the subset of the 16 802.11 specific features, the authors used the work of [Chatzoglou E et al., 2022a]. The features of the non-specific 802.11 dataset include 17 features from the Network and Transport layers. As in [Chatzoglou E et al., 2022a] the reduced datasets of 6 specific 802.11 features and 3 non-specific 802.11 features were created by the selection method based on empirical observations confirmed by the Feature Permutation Importance technique on the tree-based LightGBM model.

3 Machine Learning models and 2 Deep Learning models were used in the experimental part: DT, LightGBM, Bagging, MLP and AE. These 5 models were tested on datasets n° 1 and n° 2. As in [Chatzoglou E et al., 2022a], given that the AWID3 dataset is highly unbalanced, the authors relied on the AUC and F1 metrics to evaluate the performance of the 5 models.

On the two datasets n° 1 and n° 2, for the Machine Learning models, Bagging had the best results for the AUC and F1 metrics:

- 90.77 % et 88.07 % on the specific 802.11 dataset,
- 76.28 % et 66.02 % on the non-specific 802.11 dataset.

DT had the worst results with

- 88.98 % et 86.34 % on the specific 802.11 dataset,
- 76.21 % et 66.02 % on the non-specific 802.11 dataset.

For Deep Learning models, MLP demonstrates the best performance for AUC and F1:

- 75.53 % et 69.40 % on the specific 802.11 dataset,
- 74.67 % et 64.48 % on the non-specific 802.11 dataset.

AE performs worse results on the 802.11 specific dataset with 74.96% and 68.50% and surprisingly identical results to MLP on the non-specific 802.11 dataset.

We can notice that even the best results on the two datasets n° 1 and n° 2 are well below 90% and therefore a source of many detection errors. The application IDS created from the 5 models are therefore mediocre and unsuitable for deployment in a company information system.

For datasets n° 3, n° 4 and n° 6 to n° 10, only the 2 Machine Learning models LightGBM and Bagging were used. Overall Bagging obtains higher performances than LightGBM more particularly on dataset n° 3, n° 6, n° 8 and n° 9 with an AUC ranging from 90.71% to 95.46% and F1 ranging from 87.84% to 93.33%. For dataset n° 7 and n° 10 the results are divided: Bagging has the best AUC from 95.29% to 96.70% and LightGBM has the best F1 from 94.48% to 95.99%.

Application IDSs created on datasets n° 6 to n° 10 have metrics greater than 90%. The results are therefore much more acceptable than for the IDS created from datasets n° 1 to n° 4. However, even the best results of the work of [Chatzoglou E et al., 2022b] cannot have sufficient credibility to be deployed in a company information system where detection errors are very detrimental to the company.

To conclude this section, works [Chatzoglou E et al., 2022a] and [Chatzoglou E et al., 2022b], employing a feature selection based on empirical observations and expert human knowledges in 802.11 technology, have produced mixed results overall. The best results in [Chatzoglou E et al., 2022a], demonstrated that this feature selection technique makes it possible to create IDSs against 802.11 specific attacks with performance in line with the requirements of an enterprise information system. But this is far from being the case in [Chatzoglou E et al., 2022b] where the application IDS have insufficient performance.

However, the main pitfall of these two works, admitted by the authors, is that they did not use "*optimization or dimensionality reduction techniques*". Our work goes beyond the work of Chatzoglou E. et al. and fills these gaps by using the new BHHO-EAS metaheuristic to perform for the first time on the AWID3 dataset, at the time of writing this manuscript, a Wrapped feature selection in order to create an IDS based on a Deep Learning CNN model against attacks specific 802.11. Furthermore, the work of Chatzoglou E. et al. we will serve as references in section 3.6 in order to compare the performance of our IDS to theirs and demonstrate the superiority of our method.

3.4. Design of BHHO-EAS for a efficient Wrapper feature selection process

3.4.1. Description of the feature selection process

The features selection process consists in selecting the most relevant, informative and smallest possible subset of features $\widehat{\Phi}$ from the source set of features as illustrated in Fig. 3.2. The subset $\widehat{\Phi}$ will thus contain the features with the minimum of redundancy and of noisy as well as the maximum of relevance [Agrawal P et al., 2021; Houssein EH et al., 2022b; Houssein EH et al., 2022a; Houssein EH et al., 2022c; Segera D et al., 2023].

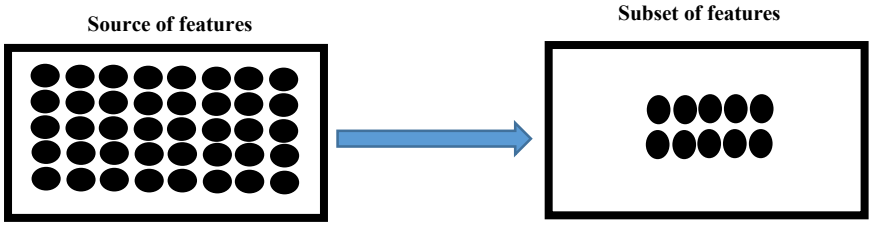


Fig. 3.2. Feature selection process

Concretely with the use of a binary vector $B_i = \left(b_i^1, \dots, b_i^d\right)$ the feature F^j is selected if $b_i^j = 1$. For example with $d = 6$ and $B_i = (0,1,0,0,1,0)$ the subset of features selected is (F^2, F^5) as illustrated in Table 3.1.

Features	F^1	F^2	F^3	F^4	F^5	F^6
Binary vector	0	1	0	0	1	0
Features selected	-	F^2	-	-	F^5	-

Table 3.1. Example of features selection for $d = 6$

Therefore, the positions of the agents in the BHHO-EAS metaheuristic will be binary positions and the feature selection optimization problem is a discreet problem in a binary search space. In order to understand the level of difficulty of this problem a mathematical explanation is necessary.

A. Mathematical theory of the feature selection

The binary discrete search space is a search space whose vector components can only take the discrete values 0 or 1.

As we mentioned above, the mathematical representation of a binary vector is $B_i = \left(b_i^1, \dots, b_i^d\right)$. The variable d is the dimension of the binary search space $\mathcal{B}^d$. $\mathcal{B}^d$ is represented by a hypercube with 2^d vertices. Each binary vector is consequently one of the vertices of this hypercube. For example, for d=3,

$\mathcal{B}^3$ is a cube with eight binary vectors positioned on the eight vertices as illustrated bellow Fig. 3.3. It means that the size of a hypercube $\mathcal{B}^d$ increases exponentially with respect to the dimension d. This mathematical fatality is conducive to the curse of the dimensionality for a high dimensional optimization problem in a binary search space. Therefore, the feature selection is a *NP-Hard* optimization problem. Fortunately, metaheuristic algorithms are the most suitable methods for solving this category of problem as a prime contractor of the feature selection. Thus, an efficient metaheuristic for the resolution of the *NP-Hard* optimization problems of feature selection, will browse the vertices of the hypercube $\mathcal{B}^d$ by exploration and exploitation phases in a reasonable time in search of a good solution close to the optimal solution. The nature-inspired and population-based metaheuristics [Agrawal P et al., 2021; Segera D et al., 2023] will provide to the design of the IDS all the benefits of the feature selection.

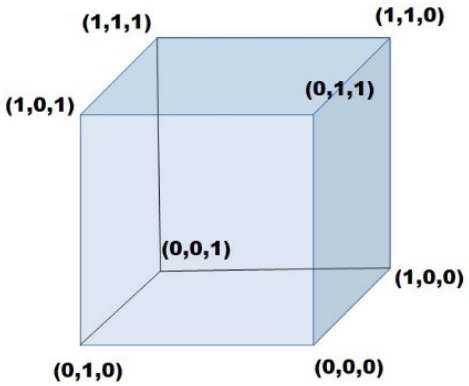


Fig. 3.3. Binary search space $\mathcal{B}^3$

B. The benefits of the feature selection to the design of a IDS

If the feature selection is an NP-Hard problem to solve, the final solution $\widehat{\Phi}$ to that problem will provide benefits commensurate with the complexity of that problem [Banka H et al., 2017a; Banka H et al., 2017b; Almomani A et al., 2019; Hodashinsky IA et al., 2019; Sassi M, 2022; Tansel D et al., 2022; Segera D et al., 2023]. The benefits will concern all the components related to the design of a IDS in the context of supervised learning: the dataset, the Deep Learning algorithm and the STA (IoT). We count ten main benefits. The feature selection process:

- Selects relevant data and remove the redundant data;
- Reduces the noise such as irrelevant and fallacious data;
- Reduces the risk of error in the data samples since the size of features is reduced;

- Minimizes the consequences of the curse of the dimensionality which exists if the number of features in the dataset is greater than or equal to 100, which increases the noise in the dataset and therefore the detection errors rate of the IDS [Gardeux V, 2011] ;
- Simplifies the representation pattern and the interpretation of the data;
- Increases or at least maintains the performance of Deep Learning algorithms;
- Reduces the computational load of the training, validation and testing phases as well as the number of Epochs during the training phase thanks to the reduction in the size of the dataset;
- Improves the generalization capacity of the Deep Learning model and therefore makes it more resilient to unknown attacks;
- Decreases the complexity of the Deep Learning model architecture, which is an essential asset for compacting an IDS and saving computing and memories resources of an IoT;
- Reduces processing time, this is another asset for better responsiveness in the field of cybersecurity.

Now that we know the benefits of the feature selection process, we need to choose how to do it, more precisely which feature selection method to use.

C. Feature selection methods

The Filter, Wrapper, Hybrid and Embedded methods are the four main methods used in the literature for feature selection [Hodashinsky IA et al., 2019; Sharma M et al., 2021; Houssein EH et al., 2022b; Houssein EH et al., 2022a; Houssein EH et al., 2022c; Segera D et al., 2023]. Each of these methods has a separate evaluation step.

a. Filter method

Filter selection methods have the advantage of being independent of the Machine Learning algorithm for the evaluation of the feature subset Φ and of being not very complex. They exploit the intrinsic statistical properties of features to assign them a value. This value will allow features to be evaluated and selected based on a threshold. The disadvantages are that this method can select interrelated features and is less precise than the wrapper method. The most used filter methods are the Hisher test, the χ^2test, the

Welsh test [Gardeux V, 2011] or even Mutual information, Correlation-based, Fast Correlation-based, Branch and bound, Consistency, Principal Component Analysis and Information Gain [Banka H et al., 2017a; Almomani A et al., 2019; Sakr MM et al., 2019; Houssein EH et al., 2022c; Houssein EH et al., 2022a].

b. Wrapper method

Unlike the filter method, the wrapper method is dependent on the Machine Learning algorithm for the evaluation of a subset of feature Φ [Banka H et al., 2017a; Almomani A et al., 2019; Houssein EH et al., 2022c; Houssein EH et al., 2022a; Houssein EH et al., 2022b; Tansel D et al., 2022]. In addition, it will provide a final solution $\widehat{\Phi}$ that is more precise and better adapted to the Machine Learning algorithm. With this method, the whole $\{Machine\ Learning\ algorithm, Dataset\}$ is considered as the "black box" optimization problem for a metaheuristic. On the other hand, the major drawback of the wrapper method is that it requires much more computational time and resources than the filter method.

c. Embedded method

The embedded method is quite similar to the wrapper method [Houssein EH et al., 2022c; Houssein EH et al., 2022a]. Like the wrapper method, the embedded method uses the Machine Learning algorithm to evaluate the feature subset Φ and provide a final solution $\widehat{\Phi}$ more precise and better adapted to the Machine Learning algorithm. The main difference with the wrapper method is that the embedded method uses an internal method that evaluates Φ during the Machine Learning model training process. The evaluation of Φ is thus embedded in the training process of the Machine Learning model, selects the features and trains the Machine Learning model at the same time [Banka H et al., 2017a; Almomani A et al., 2019; Agrawal P et al., 2021]. The L1 penalty with LASSO regularization is one of the best known evaluation methods for the embedded method.

d. Hybrid method

The hybrid method uses both the upstream filter and downstream wrapper methods [Banka H et al., 2017a; Houssein EH et al., 2022a; Houssein EH et al., 2022b]. The filter method selects a subset of features Φ before implementing the wrapper method. Then the wrapper method considers the subset Φ as the initial feature set and will in turn select and evaluate a feature subset Φ'. However, this method

inherits the lack of precision of the filter method. Indeed, if the filter method selects a subset of features Φ which is not optimal with several correlations between features, the result of the quality of the solution $\widehat{\Phi}$ provided by the wrapper method will decrease accordingly [Almomani A et al., 2019].

Among these four feature selection methods, the wrapper method is therefore the most efficient and accurate for our work because it is more efficient than the filter method and therefore the hybrid method [Sakr MM et al., 2019; Davahli A et al., 2020a]. In our works, we do not use a classic Machine Learning algorithm but the Deep Learning CNN algorithm which is very efficient in the classification and therefore in the detection of attacks. We will justify our choice in subsection 3.5.2. Moreover, unlike the embedded method, the wrapper method meets our technical specifications which require that the BHHO-EAS metaheuristic be the driving force in the search for the best subset of feature $\widehat{\Phi}$ of the AWID3 dataset. But the wrapper method must be implemented with the most appropriate feature selection initialization.

3.4.2. Binary vector initialization strategies

The initialization strategy is sometimes neglected in the literature while the initial position of the agents in the search space is a crucial element in the optimization strategy [Qian L et al., 2020; Agushaka JO et al., 2022]. The initialization in a metaheuristic algorithm must make it possible to position its agents in order to effectively cover the search space and to increase the probabilities that at least one of the agents is in a promising zone containing the optimal solution. This would reduce the search time and the quality of the final solution.

For binary metaheuristics there are three main binary vector initialization strategies [Almomani A et al., 2019; Gad AG et al., 2022]:

- Random initialization;
- Great initialization;
- Small initialization.

A. Random initialization

The Random selection method offers the greatest probability of being positioned from the start on a promising zone of the hypercube by randomly initializing each component of the binary vectors to ***0*** or

1. During the iterations the components at ***0*** or at ***1*** can respectively change to ***1*** or to ***0*** in order change the subset of feature Φ and to obtain the most optimal subset $\widehat{\Phi}$. It is this method that we favored in our work compared to the Great initialization and Small initialization methods.

B. Great initialization

The Great initialization method, for each binary vector, randomly selects one of its components to set it to ***0*** and leaving all the other components at ***1***. The positioning of the bit at ***0*** must be different from one binary vector to another. The objective, if the size of the population of the metaheuristic is greater than or equal to the dimension of the optimization problem, is that the component at ***0*** traverses at least once the set of dimensions of the binary vectors.

As iterations go, the bits at ***1*** will change to ***0*** to remove features and reduce the size of the subset Φ [Almomani A et al., 2019; Gad AG et al., 2022].

C. Small initialization

The Small initialization method is the opposite of the Great initialization method. This method will randomly set to 1, a single component of the binary vector and all the other components to ***0***. As the iterations progress, the bits at ***0*** will change to ***1*** to select new features and increase the size of the subset Φ [Schiezaro M et al., 2013; Almomani A et al., 2019; Gad AG et al., 2022].

Fig. 3.4. Illustrates the three initialization methods for a population of four agents and a dimension equal to eight.

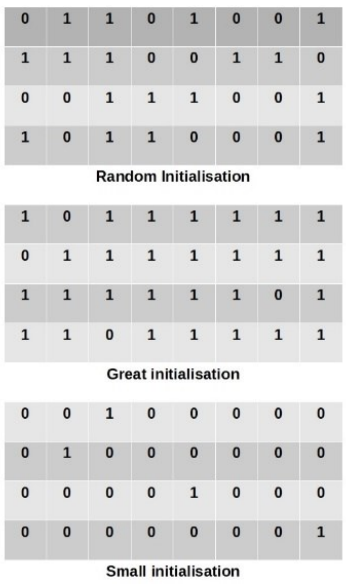


Fig. 3.4. The three main binary vector initialization strategies

We now have all the theoretical elements of the feature selection process. In order to have a synthetic vision we provide in the following its flow chart.

3.4.3. Flow chart of the feature selection process

The feature selection process is divided into four steps [Banka H et al., 2017b]:

- Selection of a feature subset Φ;
- Evaluation of the subset Φ;
- Termination tests of the Selection process;
- Validation of the subset Φ.

The flowchart in four steps are illustrated in Fig. 3.5.

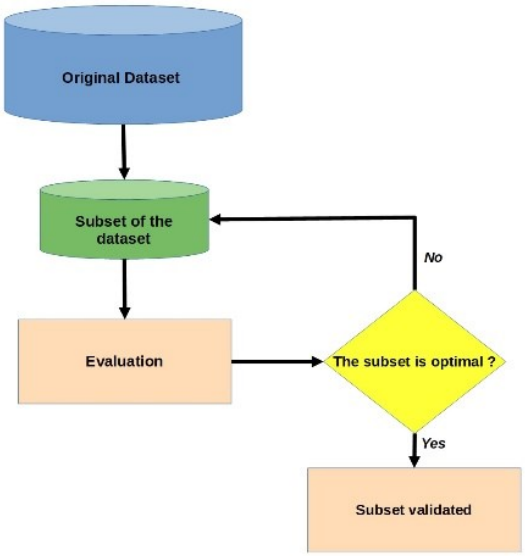


Fig. 3.5. Steps of the feature selection

A. Selection and validation

The selection is made by one of four methods: filter, wrapper, hybrid and embedded. As explained above, each of these methods has a separate evaluation step.

B. Termination tests of the selection process

There are three criteria for stopping the feature selection process:

- Maximum number of iterations reached;
- No improvement after a certain number of iterations;
- The optimal subset of features is already obtained at initialization.

C. Feature subset validation

The validation of the feature subset $\widehat{\Phi}$ consists in confirming the relevance of the selection thanks to performance metrics. In our work, the evaluation step will exploit the *Accuracy* metric of the IDS.

We now have the fundamentals of the feature selection process. We know what this process is, what are its advantages, its four main steps and how to implement it with filter, wrapper, embedded or hybrid methods. In the next subsection 3.4.4 we will design the BHHO-EAS metaheuristic which will be the prime contractor for our wrapper feature selection. Our wrapper feature selection method is unique since it uses the Deep Learning algorithms and the computing power of GPU technology for the design of the CNN-IDS.

3.4.4. Design of the binary metaheuristic BHHO-EAS

To adapt HHO-EAS metaheuristic to optimization in a binary discrete search space it is necessary it is necessary to allow him to discretize the continue search space.

There are several methods of discretization or encoding such as Nearest Integer (NI) [Burnwal S et al., 2013], Smallest Position Value (SPV) [Verma RS et al., 2012; Demyana IE et al., 2016], Great Value Priority (GVP) [Congying L et al., 2011], Random-Key (RK) [Huiqin C et al., 2011] or the modified position equation (MPE) method that was used on PSO [Pan QK et al., 2008]. The later required the modification of the equations for calculating the position of the agents. In our case we adopt the binary discretization which will limit the number of possible states to 0 or 1 [Sassi M, 2022].

We have designed a modular binary discretization method, independent of metaheuristics and which allows to preserve their algorithmic and mathematical integrity as well as their exploration and exploitation strategy.

We therefore performed a high-level hybridization, integrated in the HHO-EAS metaheuristic [Ting TO et al., 2015]. Our hybridization is composed of two modules $M1$ and $M2$ [Sassi M, 2022]. These two modules are suitably positioned in HHO-EAS so that at each iteration the continuous positions X_i of an agent i are converted into a binary position V_{bi} [Raidl GR et al., 2019].

A. Module M1

For a search space of dimension d and an agent population of size N, the module $M1$ make it possible to go from a continuous real space $\mathbb{R}^d$ to an continuous intermediate space $[0,1]^d$. It normalizes the real values x_i^j, components j of the positions X_i of an agent i, in order to obtain a probability value p_i^j via a transfer function T. Thus the module $M1$ converts the position vector $X_i = (x_i^1, \dots, x_i^d)$ to the intermediate vector $P_i = (p_i^1, \dots, p_i^d)$ which will act on the selection of the features set $(f^1, .., f^d)$.

Equation (3.1) summarizes the mathematical operation of $M1$ performed by a transfer function T.

$$\begin{cases} i \in [\![1, N]\!], j \in [\![1, d]\!] \\ X_i \in \mathbb{R}^d, P_i \in [0,1]^d \\ X_i = (x_i^1, \dots, x_i^d) \overset{T}{\Rightarrow} P_i = (p_i^1, \dots, p_i^d) \end{cases} \tag{3.1}$$

Referring to the state of the art, the most used transfer functions T for feature selection are: *S-shaped*, *V-shaped, U-shaped* and *Q-shaped* [Too J et al., 2019; Kumar V et al., 2021; Nadimi-Shahraki, M.H. et al., 2021; Tansel D et al., 2022]. Recent work has proven the superiority of *Q-shaped* transfer functions [Too J et al., 2019].

Table 3.2, Table 3.3, Table 3.4 and Table 3.5 provide the mathematical formulation of transfer functions and Fig. 3.6, Fig. 3.7, Fig. 3.8 and Fig. 3.9 provide their graphical representations. For a search space of dimension d, the transfer functions *S-shaped* provide the probability of selecting a feature. On the other hand, the *V-shaped*, *U-shaped* and *Q-shaped* transfer functions provide the probability of modifying the choice made at the previous iteration: if a feature has been selected, it will no longer be at the next iteration and vice versa.

The next step will be to go from the intermediate space $[0,1]^d$ to the binary space $\{0,1\}^d$ with the module $M2$.

Name	S-shaped functions
S1	$\frac{1}{1+e^{-2x}}$
S2	$\frac{1}{1+e^{-x}}$

S3	$\frac{1}{1+e^{(-\frac{x}{2})}}$
S4	$\frac{1}{1+e^{(-\frac{x}{3})}}$

Table 3.2. S-shaped functions

Name	V-shaped functions
V1	$\left\lvert \operatorname{erf}(\frac{\sqrt{\pi}}{2}x) \right\rvert$
V2	$\lvert \tanh(x) \rvert$
V3	$\frac{\lvert x \rvert}{\sqrt{1+(x)^2}}$
V4	$\left\lvert \frac{2}{\pi} \arctan(\frac{\pi}{2}x) \right\rvert$

Table 3.3. V-shaped functions

Name	U-shaped functions
U1	$\alpha \left\lvert x^{\frac{3}{2}} \right\rvert$
U2	$\alpha \lvert x^2 \rvert$
U3	$\alpha \lvert x^3 \rvert$
U4	$\alpha \lvert x^4 \rvert$

Table 3.4.U-shaped functions

Name	Q-shaped functions
Q1	$\begin{cases} \left\lvert \frac{x}{0.5x_{max}} \right\rvert, if\ \lvert x \rvert < 0.5x_{max} \\ 1, else \end{cases}$
Q2	$\begin{cases} \left(\frac{x}{0.5x_{max}}\right)^2, if\ \lvert x \rvert < 0.5x_{max} \\ 1, else \end{cases}$
Q3	$\begin{cases} \left(\frac{\lvert x \rvert}{0.5x_{max}}\right)^3, if\ \lvert x \rvert < 0.5x_{max} \\ 1, else \end{cases}$

Q4	$\begin{cases} \left(\frac{\lvert x\rvert}{0.5x_{max}}\right)^{\frac{1}{2}}, if\ \lvert x\rvert < 0.5x_{max} \\ 1, else \end{cases}$

Table 3.5. Q-shaped functions

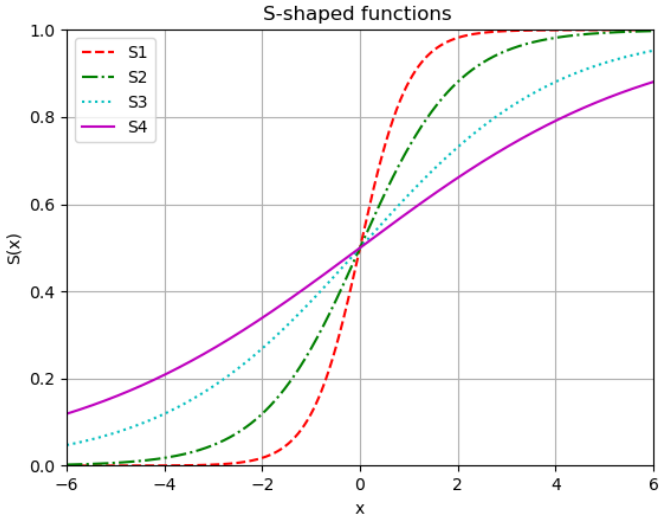


Fig. 3.6. S-shaped functions

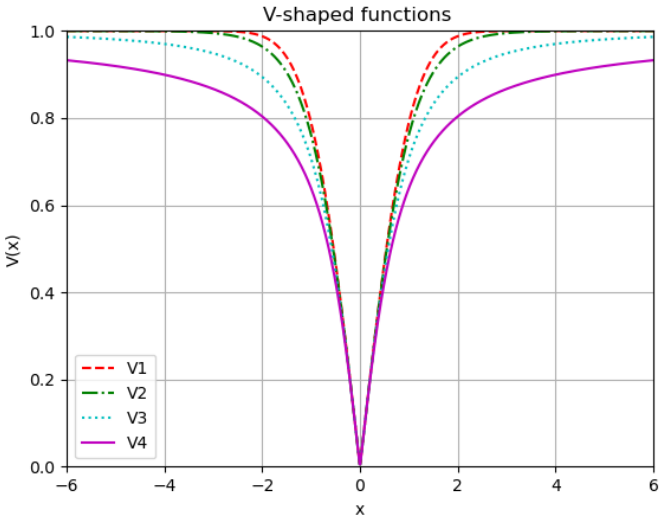


Fig. 3.7.V-shaped functions

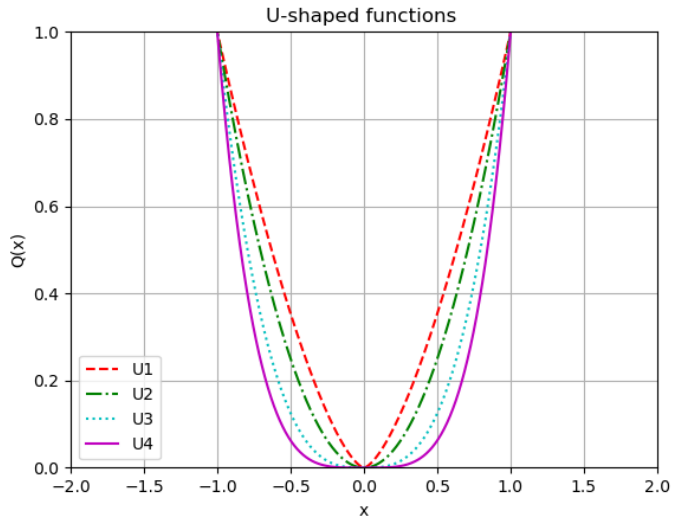


Fig. 3.8. U-shaped functions ($\alpha = 1$)

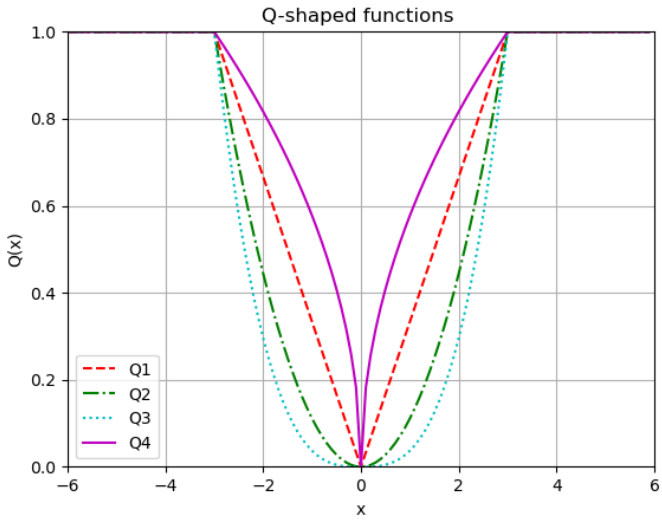


Fig. 3.9. Q-shaped functions ($x_{max} = 6$)

B. Module M2

The module $M2$, by the use of a binarization rule R, makes it possible to pass from the intermediate space $[0,1]^d$ to the binary space $\{0,1\}^d$. There are two main binarization rules: $R1$ and $R2$ .These two rules convert the probability values p_i^j of the vector P_i into binary values b_i^j in order to create the binary vector B_i. The mathematical equations of the rules $R1$ and $R2$ are respectively detailed in (3.2) and (3.3). Each of these rules applies to a specific category of transfer function T:

- Rule $R1$ for *S-shaped* functions;
- Rule $R2$ for *V-shaped*, *U-shaped* and *Q-shaped* functions.

$$R1: b_i^j = \begin{cases} 1 \; if \; rand \; \leq \; p_i^j \\ 0 \; else \end{cases} \tag{3.2}$$

$$R2: b_i^j = \begin{cases} \sim(b_i^j) \; if \; rand \; \leq \; p_i^j \\ b_i^j \; else \end{cases} \tag{3.3}$$

Thus in rule $R2$, if the binary component $b_i^j = 1$ and the random value $rand \; \leq \; p_i^j$ then $\sim(b_i^j) = 0$. And if $b_i^j = 0$ and $rand \; \leq \; p_i^j$ then $\sim(b_i^j) = 1$. On the other hand, if $rand > p_i^j$, b_i^j will not change its value.

Equation (3.4) summarizes the mathematical operation of $M2$ performed by a logical rule R.

$$\begin{cases} i \; \in [\![1, N]\!], j \in [\![1, d]\!] \\ P_i \in [0,1]^d, \; B_i \; \in \{0,1\}^d \\ P_i = \left(p_i^1, \dots, p_i^d\right) \overset{R}{\Rightarrow} B_i = \left(b_i^1, \dots, b_i^d\right) \end{cases} \tag{3.4}$$

Thus the combination of the module $M1$ and the module $M2$ creates a mathematical process of binary discretization of the position vectors X_i in order to get B_i. Equation (3.5) summarizes the overall process of binary discretization for a transfer function T and a logical rule R.

$$\begin{cases} i \; \in [\![1, N]\!], j \in [\![1, d]\!] \\ M1: X_i = \left(x_i^1, \dots, x_i^d\right) \overset{T}{\Rightarrow} P_i = \left(p_i^1, \dots, p_i^d\right) \\ M2: P_i = \left(p_i^1, \dots, p_i^d\right) \overset{R}{\Rightarrow} B_i = \left(b_i^1, \dots, b_i^d\right) \end{cases} \tag{3.5}$$

In order to obtain the new metaheuristic BHHO-EAS, it now remains to integrate the two modules $M1$ and $M2$ in HHO-EAS and to initialize the binary position of the agents with the Random initialization method. Algorithm 2 describes the pseudo-code of BHHO-EAS. The modules $M1$ and $M2$ are in blue color in Algorithm 2. The flow chart of BHHO-EAS is provided in Appendix 8. This flow chart specifies the components of the BHHO-EAS algorithm which exploits the GPU or the CPU for its calculations. The calculation of the new position of the agents at each iteration is identical to HHO-EAS. The pseudo-code and the flow chart confirm that there is no alteration of the HHO-EAS algorithm and that the modules $M1$ and $M2$ were added at the end of the calculation of the new positions X_i of each agent i.

In the next section 3.5, with the aim of conceive the CNN-IDS against 802.11 specific attacks, we will demonstrate that BHHO-EAS is the central intelligence in the wrapper feature selection method applied to the AWID3 dataset.

Algorithm 3.1 Pseudocode of BHHO-EAS

In: Population size *N*, Number of Iteration *ItMax*, Objective function *F,* Transfer function *T,* Rule *R*

Out: $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

Initialization of the Harris Hawk's binary positions $\vec{B}_i, i \in [\![1, N]\!]$, $K = 2$, t=0

While (t < *ItMax*) **do**:

Calculate $F(\vec{X}_i)$ of each Harris Hawk's position $\vec{X}_i$

Define the best position of rabbit $\overrightarrow{X_r}$ and $F(\overrightarrow{X_r})$

For each Harris Hawk agent i **do**:

Update e, E_0, E, and J with ***FISREE***, (2.2) and (2.6)

If($|E| \geq 1$) *[Exploration]* **then**:

Calculate q and **Update** position with (2.4)

Else *[Exploitation]:*

Update $\overrightarrow{V_\alpha}$, $\overrightarrow{V_\beta}$ and $\overrightarrow{V_\delta}$

Define:

$\overrightarrow{X_\alpha}$=First best position

$\overrightarrow{X_\beta}$=Second best position

$\overrightarrow{X_\delta}$=Third best position

Calculate encirclement equations with (2.22), (2.23), (2.24), (2.25), (2.26) and (2.27)

Calculate $\overrightarrow{X_e}$ with attack equation (2.28)

Update Harris Hawk's position $\vec{X}_i$ by $\overrightarrow{X_e}$

Calculate r_5

If $|E| \geq 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.6) and (2.7)

If $|E| < 0.5$ and $r_5 \geq 0.5$ **then**:

Update position with (2.8)

If $|E| \geq 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.9) to (2.12)

If $|E| < 0.5$ and $r_5 < 0.5$ **then**:

Update position with (2.12) to (2.15)

End For

For each Harris Hawk agent position X_i **do**:

$P_i = T(X_i)$ with $T \epsilon M1$ (3.1)

$B_i = R(P_i)$ with $R \epsilon M2$ (3.2), (3.3)

t=t+1

End While

Return $\overrightarrow{X_r}$

3.5. Application of BHHO-EAS to design a CNN-IDS specific to 802.11 attacks

3.5.1. Mathematical modeling of the Wrapper feature selection process in a multi-objective optimization problem

The wrapper feature selection method provides the optimal subset of features $\widehat{\Phi}$ by pursuing a dual objective: minimizing the size of the feature subset Φ and maximizing the detection performance of the CNN. The wrapper feature selection is therefore a NP-Hard multi-objective optimization problem [Almomani A et al., 2019; Davahli A et al., 2020a; Houssein EH et al., 2021b; Sharma M et al., 2021; Houssein EH et al., 2022a].

The two objectives to be optimized are:

- **1st objective Minimize** : Select the feature subset Φ with the smallest possible size represented by a binary vector V_b. But reducing the size of Φ consists in minimizing the Hamming distance $|\boldsymbol{V_b}|$ between the binary vector V_b and the null vector;
- **2nd objective Maximize**: We need to increase the *Accuracy* metric ($\boldsymbol{Acc}$) to maximize the CNN's ability to predict the *Attacks* and *Normal* samples and reduce the detection errors. *Accuracy* metric is more explicitly detailed in subsection 3.6.2.

The mathematical transcription of this multi-objective optimization problem is as follows:

- Maximize $\boldsymbol{Acc}$
- Minimize $|\boldsymbol{V_b}|$.
- With:

$$\boldsymbol{V_b} = (\boldsymbol{b^1}, \dots, \boldsymbol{b^d}),$$

$$|\boldsymbol{V_b}| = \sum_{\boldsymbol{j=1}}^{\boldsymbol{d}} \boldsymbol{b^j},$$

$$\boldsymbol{Acc}(\boldsymbol{V_b}) = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{FN} + \boldsymbol{TN} + \boldsymbol{FP}}$$

The solution $\widehat{V_b}$ to this multi-objective problem is generally not unique and not optimal because they represent a compromise on the two objectives Min-Max to be achieved. Indeed, the excessive reduction

of the number of features could decrease the *Accuracy*. And conversely the increase in *Accuracy* tends to generate an increase in the number of features.

As we specified in chapter 1, among the solutions obtained, those which have a relation of dominance over the other solutions and none of them, constitute the Pareto optimal front.

To solve this multi-objective problem with an A priori approach, we will use the weighted sum methods (or aggregation method) [Siarry P et al., 2002a; Donoso Y et al., 2016]. This method is the most widely used in the literature for solving the multi-objective wrapper feature selection optimization problems [Mafarja MM et al., 2017; Almomani A et al., 2019; Bonab MS et al., 2020; Davahli A et al., 2020a; Davahli A et al., 2020b]. The two main advantages of this method and which motivated its use for our works are the following:

- It allows to go from a multi-objective problem to a single-objective problem that can be solved by the BHHO-EAS metaheuristic;
- It is particularly effective in obtaining a good final solution thanks to the A priori intervention of an expert.

In our case, the A priori intervention requires a cybersecurity expert in 802.11 technology. The expert will give either a preponderance to the *Accuracy* $\boldsymbol{Acc}$ with the weight ω_1 in order to maximize the detection performances of the CNN-IDS, or will favor the minimization of $|\boldsymbol{V_b}|$ to save the IoT's computing resources via the weight ω_2. Our expert choice is as follows: In the context of IoT cybersecurity, the priority is to have the most efficient CNN-IDS possible in the detection of 802.11 specific attacks with the fewest false positives and false negatives. Consequently, the weight assigned to $\boldsymbol{Acc}$, ω_1 will be preponderant with respect to the weight ω_2 assigned to $|\boldsymbol{V_b}|$.

In order to obtain a single equation to be minimized from a Min-Min multi-objective optimization problem, we replace the *Accuracy* by the detection Error $\boldsymbol{Err}$. $\boldsymbol{Acc}$ and $\boldsymbol{Err}$ are linked by the equation: $\boldsymbol{Err = 1 - Acc}$. And since the functions to be optimized must be normalized in the method of weighted sums, we divide $|\boldsymbol{V_b}|$ by the dimension of the optimization problem $\boldsymbol{d}$.

The multi-objective optimization problem Min-Min is therefore expressed by the mono-objective equation (3.6) to be minimized:

$$F(V_b) = \omega_1.(1 - Acc(V_b)) + \omega_2.\frac{|V_b|}{d}. \quad (3.6)$$

Furthermore we know that in the weighted sum method $(\omega_1, \omega_2) \in [0,1]^2$ and $\omega_1 + \omega_2 = 1$. The equation (3.6) then becomes equation (3.7) with $\omega_2 = \omega$ and $\omega_1 = 1 - \omega$:

$$F(V_b) = (1 - \omega).(1 - Acc(V_b)) + \omega.\frac{|V_b|}{d} \quad (3.7)$$

The theoretical bases of the Wrapper feature selection multi-objective optimization problem are laid. We must now design the architecture of the wrapper feature selection with the triptych {BHHO-EAS, CNN, AWID3}. We have already designed the BHHO-EAS metaheuristic in section 3.4. In what follows we will first define the choice of the Deep Learning model CNN before approaching the analysis and the preprocessing of the AWID3 dataset.

3.5.2. Convolutional neural network for the creation of a CNN-IDS

The Convolutional Neural Network (CNN) is one of the neural networks of Deep Learning algorithms. Deep Learning algorithms are a subset of Machine Learning algorithms themselves being a subset of Artificial Intelligence [Houssein EH et al., 2021a; Monteiro ACB et al., 2021; Pedroza M et al., 2021]. Fig. 3.10 below schematizes these inclusions.

Deep Learning algorithms are much more efficient than classic Machine Learning algorithms for classification tasks [Zhang S et al., 2020]. Deep Learning provides us with many neural networks that mimic the neural and visual abilities of animals: Feedforward Neural Network (FNN), Recurrent Neural Network (RNN), Convolutional Neural Networks (CNN), etc.

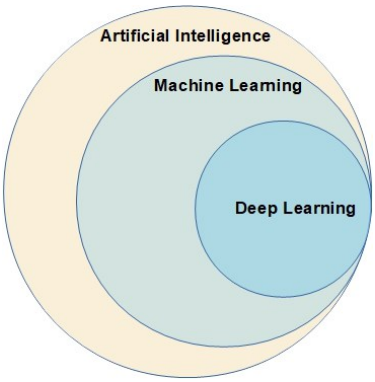


Fig. 3.10. Inclusion of ML and DL in AI

Before exhaustively describing CNN and its inherent attack detection capabilities and its ability to be embedded into an IoT, we will provide a general overview of Machine Learning.

A. General overview of Machine Learning

Michel T, computer science professor at the University of Carnegie Mellon and researcher in artificial intelligence, was one of the first researchers to provide a definition of Machine Learning : *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P, improves with experience E"* [Mitchell T, 1997]. This definition is one of the theoretical foundations of Machine Learning.

Machine Learning consists of three main branches [Hussain F et al., 2020; Sarker IH, 2021]: Reinforcement Learning, Unsupervised Learning, and Supervised Learning. We will define below each of these three branches.

a. Reinforcement Learning

Reinforcement Learning is the branch of Machine Learning which is closest to human behavior because it is based on trials and errors. This branch of Machine Learning requires an Agent, an Environment, the States of the Environment and the Rewards obtained by the Agent's Actions for a given State of the Environment. Based on the State of the Environment and the value of the Reward, the Agent learns via Reinforcement Learning algorithms to optimize these decisions by choosing the Action which will modify the state of the environment and will maximize his Rewards in the long term. A multitude of Reinforcement Learning algorithms exist such as Q-Learning, Sarsa, Deep Q-Learning, etc. [Zhang H et al., 2020].

b. Unsupervised Learning

Having no information on the data label, Unsupervised Learning algorithms are trying to highlight the internal data structure in order to determine internal characteristics common to data. This is the case of clustering or reduction of dimensions such as Principal Component Analysis (PCA) [Zhang S et al., 2020; Dara S et al., 2022; Verma KK et al., 2022].

c. Supervised Learning

In Supervised Learning, all data are labeled. This means that each record of the data has its known exit. For example, an image dataset representing cats or dogs will be labeled cat or dog respectively. The

Supervised Learning algorithms will create a function that will associate a record with its label [Zhang S et al., 2020; Dara S et al., 2022; Verma KK et al., 2022]. The design of the CNN-IDS is in the context of the Supervised Learning.

Fig. 3.11 provides a general taxonomy of the three branches of the Machine Learning with examples of algorithms for each branch.

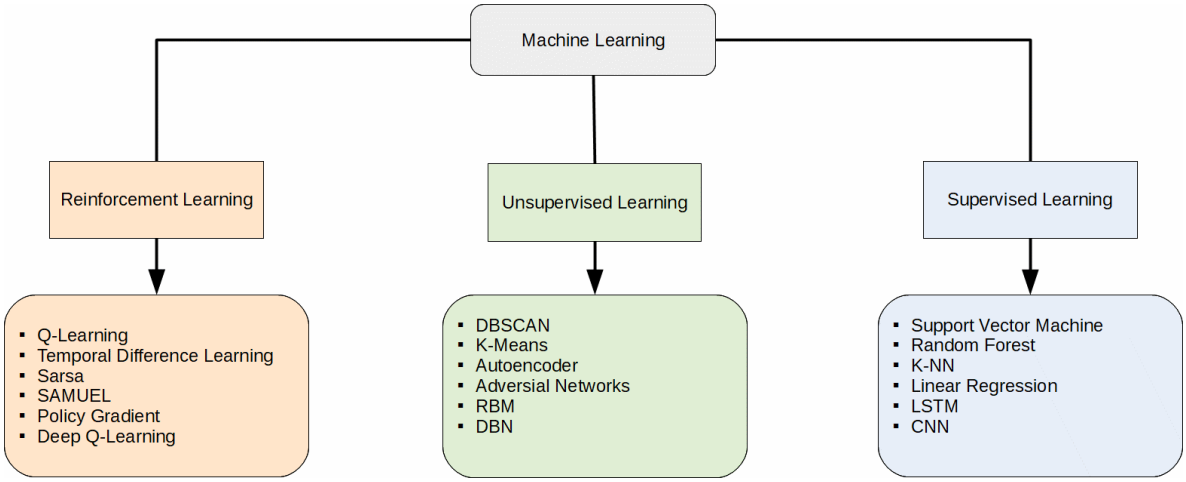


Fig. 3.11. General taxonomy of Machine Learning algorithms

B. General skills of CNNs for intrusion detection and to be embedded into an IoT

CNN models have a common general architecture. They are composed of three main blocks [Ramos-Michel A et al., 2021]:

- **The convolution layers**: They consist of several maps of features having distinct convolution kernels of smaller dimensions than the images to be processed as input. These kernels are the equivalent of weights in classical neural networks. Within the same feature map, the convolution kernels are identical. Thanks to convolution kernels, these feature maps can extract one after another and layer by layer the main high-level features of an image from its low-level features to transmit them to the Pooling layers;
- **Pooling layers:** In a rectangular area of reduced dimension, the Pooling layers sub-sample the information coming from the convolution layers by different mathematical aggregation operations. The most exploited operations in CNNs are the maximum, the average, the weighted average or the $\mathcal{L}^2$ norm. The Pooling layer therefore makes it possible to reduce the dimensions of an image, the amount of memory and calculation required while keeping the main high-level

information. This last point also makes it possible to minimize the overfitting of the CNN model. The data aggregated by the last pooling layer is then passed to the fully connected layers;

- **Fully connected layers**: After a flattening (or serialization) of the data of the last Pooling layer, this layer learns to process the high-level nonlinear features coming from the combination [Convolution layer - Pooling layer], classifies it as a classic neural network would.

The general architecture of CNNs is detailed in Fig. 3.12.

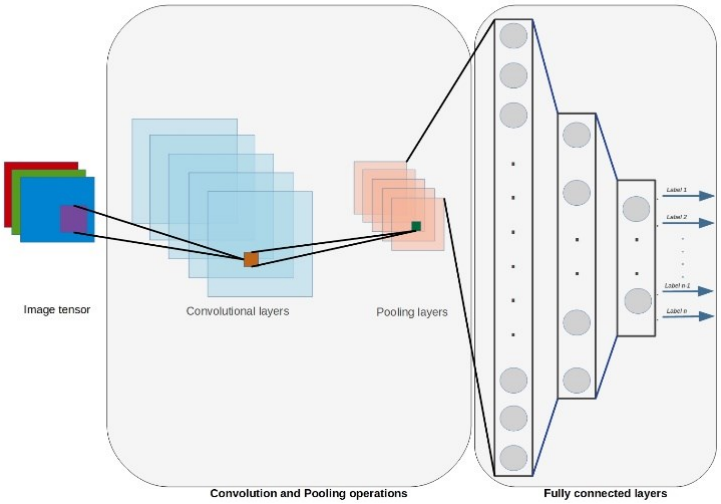


Fig. 3.12. General architecture of Convolutional Neural Network

Now that we know the technical specifications of CNN, let's answer the following question: why did we choose CNN to design our IDS? We chose CNN for two main reasons.

The first reason is based on the three properties inherent in the architecture and operation of CNN [Goodfellow I et al., 2016; Gaspar A et al., 2021; Marquez Casillas ES et al., 2021; Monteiro ACB et al., 2021; Ramos-Michel A et al., 2021]. Initiated by work on the Neocognitron by Kunihiko Fukushima in 1980 [Fukushima K, 1980], the CNN has mathematical processes based on convolution (or cross-correlation) making it possible to extract the high-level characteristics of images from low-level characteristics such as the human visual cortex would. This is the mathematical property of equivariance and invariance of CNNs. This is a major asset for the recognition of attacks with comparable motives. The other two properties are sparse interactions and parameter sharing. These two properties make it possible to reduce the number of parameters, the necessary calculations and de facto will relieve the

material resource necessary for the execution of the CNN model. These two properties thus contribute to the ability of the CNN-IDS to be embedded in an IoT.

The second reason is based on our analysis of the state of the art. Recent literature proves that CNNs are much more efficient than "classic" Machine Learning algorithms for the detection of known and unknown attacks [Kim K et al., 2018b; Kim K et al., 2018c; Azizjon M et al., 2020; Jiyeon K et al., 2020; Riyaz B et al., 2020; Hu J et al., 2021; Park D et al., 2021; Tressa M, 2021; Tufan E et al., 2021; Mohammadpour L et al., 2022]. CNNs are therefore a major asset for the recognition of attack patterns coming from the Wi-Fi network flow composed of several complex low-level characteristics.

Thus the inherent qualities of the CNN that we have just exposed above make it the appropriate Deep Learning algorithm for the design of our CNN-IDS. And in order to classify the attack vectors by the CNN-IDS from the AWID3 dataset, they will first be transformed into 1D images in the preprocessing phases.

3.5.3. Analysis of the AWID3 dataset

The AWID3 dataset (Aegean Wi-Fi Intrusion Dataset 3) was created in 2021 by researchers from the Greek Aegean University [Chatzoglou E et al., 2021]. It is available in PCAP and CSV format [University of Aegean, 2014]. As we specified in section 3.3, unlike its predecessor AWID2, the AWID3 dataset is much more complex and fully in line with the technical requirements of a company information system.

For the design of our CNN-IDS we used AWID3 in CSV format. It is composed of 28,800,413 samples. These samples are structured around 253 features representing the protocols from the Physical layer to the Application layer of the OSI stack. A label is added to the 253 features to identify an attack. The 253 features are composed of several types of data: integer, binary, float and string, knowing that hexadecimal values are transformed into integer values. We will see in what follows that these features can be classified into two classes of data: categorical and numeric.

The attacks are spread over 13 groups of class. These 13 groups of attacks represent both the classic 802.11 specific attacks found in AWID 2 and upper layer attacks: *Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)Assoc), Rogue Access Point (Rogue AP), Evil Twin, Key*

Reinstallation Attack (Krack), Kr00k, Botnet, SSH brute force, Malware, SQL injection, SSDP amplification and *Website spoofing*. 7 of these 13 groups of attacks are specific to 802.11 technology and more particularly target the MAC layer in order to succeed in their offensive towards higher layers. These 7 groups of attacks are: *Deauthentication (Deauth), Disassociation (Disas), ReAssociation ((Re)Assoc), Rogue Access Point (Rogue AP), Evil Twin, Key Reinstallation Attack (Krack)* and *Kr00k*. This represents 15,155,345 samples of the dataset [Chatzoglou E et al., 2022a]. The other 6 groups of attacks target the upper layers: *SSH brute force, Malware, SQL injection, SSDP amplification* and *Website spoofing.* This represents 13,645,068 samples of the dataset [Chatzoglou E et al., 2022b].

As we specified in subsection 3.2.4, we focus on the samples linked to the 7 groups of 802.11 specific attacks that must be preprocessed beforehand before being used.

3.5.4. Preprocessing of the dataset AWID3

As it stands, the AWID3 dataset cannot be used by a Deep Learning algorithm and does not meet the 5 requirements of the specifications provided in subsection 3.2.4. Significant data preprocessing work was carried out beforehand, followed by a balancing of the data by subsampling in order to have a 1:1 ratio between the Normal and Attacks samples. The preprocessing ends with an encoding phase of the categorical features and a normalization phase of the numerical features by using respectively One-Hot-Encoding (OHE) and Min-Max [Neu DA et al., 2022].

The OHE encoding converts categorical features into numerical values via a binary vector whose size is the numbers of categorical values. In this vector representation, all the components are at 0 except the position of the feature which is at 1. For example, the feature *wlan.fc.tytpe* has 3 possible values: *management*, *control* and *data*. Their respective binary representations are: $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. On the other hand, Min-Max makes it possible to grant the same importance to all the numerical features in the supervised learning process whatever their scale of value. In addition, it helps to reduce noise and outlier values. The scale values of each numerical feature are thus converted into a numeric value in [0,1] by the following formula: $\frac{x-x_{min}}{x_{max}-x_{min}}$.

This preprocessing phase of the AWID3 dataset allows our CNN model to be trained and to be integrated in the Wrapper feature selection process driven by our BHHO-EAS metaheuristic.

The preprocessing of the AWID3 dataset follows a 4-step workflow taking into account the CNN-IDS design specifications:

- Application of the 5 requirements of the CNN-IDS specifications;
- Data cleaning;
- Data balancing by a subsampling of the *Normal* samples to get a ratio 1:1 between *Normal* and *Attack* samples;
- Data encoding with OHE and Min-Max methods.

A. Application of the 5 requirements of the CNN-IDS specifications

In the first phase of preprocessing, we apply the 5 requirements of the specifications explained in subsection 3.2.4.

R1. Detection of 802.11 specific attacks

Only the seven 802.11 specific attacks are taken into account as well as the Physic and Data Link layer features. These features are 66 in number.

R2. Compatibility with companies' information systems

For the CNN-IDS to be independent of the information system, the IDS must not be linked to the Wi-Fi infrastructure. We therefore remove the RSN-related features *wlan.rsn* (Robust Security Network) which make it possible to determine which security protocol is used: WPA, WPA2 and WPA3. We also remove the features *wlan.ta*, *wlan.ra*, *wlan.bssid* and *wlan.ssid* attached to the Wi-Fi network infrastructure. Furthermore, these features can be very easily usurped or modified by attackers and keeping these features would link the dataset to a Wi-Fi network and would generate a risk of overfitting the CNN model and would harm to its generalization.

R3. Hardware adhesion

The features should not be linked to the hardware of an STA. This particularly concerns the datarate fields and the MAC adress: *radiotap.datarate, wlan_radio.data_rate*, *wlan.sa* and *wlan.da*.

R4. 802.11 frames processing

The features must be common to the three types of frames: Management, Control and Data. This allows the CNN-IDS to exploit the three types of frames for the detection of 802.11 attacks. Consequently, the features linked only to management frames with the *wlan_mgt* root are deleted.

R5. Temporal and sequential components

The need to embed the IDS in environments with limited computing and memory resources requires minimizing the demands of superfluous data. Moreover, the IDS is not intended to take time series into account. Consequently, the temporal and sequential components are not taken into account. The CNN-IDS is therefore timeless and each frame must be independent of the previous one. Consequently, the temporal and sequential features must not be taken into account in the detection of 802.11 attacks: The following features are deleted:

- *wlan.seq,*
- *frame.number,*
- *frame.time,*
- *frame.time_delta,*
- *frame.time_delta_displayed,*
- *radiotap.timestamp.ts,*
- *wlan_radio.start_tsf,*
- *wlan_radio.end_tsf,*
- *frame.time_epoch,*
- *radiotap.mactime,*
- *wlan_radio.timestamp,*
- *wlan_radio.start_tsf,*
- *wlan_radio.end_tsf,*
- *frame.time_relative.*

At the end of this first phase of preprocessing based on the 5 requirements of the specifications, there remain 18 features explained in Table 3.7. The correlation heatmap of these 18 features is detailed in the Fig. 3.13.

A cleaning of the data represented by these 18 features is necessary before the encoding phase.

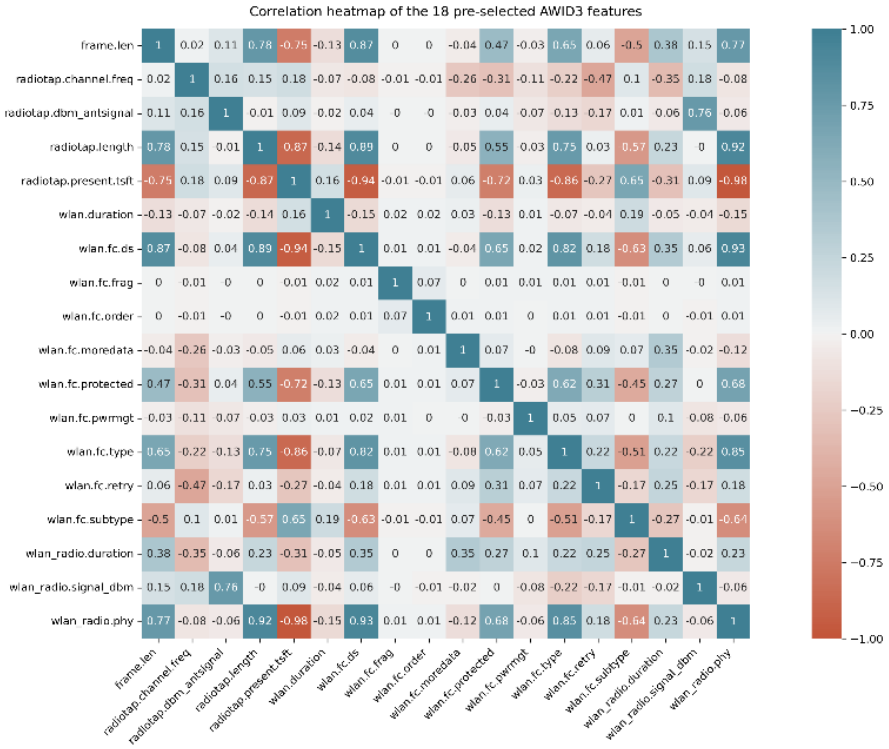


Fig. 3.13. Correlation heatmap of the 18 pre-selected AWID features

B. Data cleaning

In the second phase of the preprocessing we have proceeded to the cleaning of the data by deleting the features presenting the following specificities in the distribution of their values:

- Having more than 50% missing or outlying values such as *'NaN', '?', 'Null', 'Inf'*;
- Having more than 90% of zero variance such as features:
  - *radiotap.present.flags,*
  - *radiotap.present.rxflags,*
  - *radiotap.present.dbm_antsignal,*
  - *radiotap.present.channel,*
  - *radiotap.present.antenna*.

For the remaining features, we replaced the missing values by the average of their values for float type features and the median for integer, binary and non-ordinal string type features. The average and the median are calculated in relation to all the correct values of the feature. The hexadecimal data with '*0x*', the exponent syntax of 10 with the symbol '*E*' or '*e*' and the value of combining

several data such as *radiotap.present.tsft* or *radiotap.dbm_antisignal* have been converted to numerical value.

The next preprocessing step will create the final dataset balanced between *Normal* and *Attack* samples.

C. Data balancing by subsampling

To respect the specifications, with the objective of creating a compact and embeddable CNN-IDS in an IoT, we have chosen to reduce the complexity of the neural network by detecting the 3 main classes of 802.11 attacks: *Impersonation*, *Flooding* and *Normal*. The Impersonation attack class represents the 3 attacks *Krack*, *Evil_Twin* and *RogueAP*. The Flooding attack class represents the 4 attacks *Deauth*, *Disas*, *(Re)Assoc* and *Kr00k*.

Contrary to [Chatzoglou E et al., 2022a] which kept a very strongly unbalanced dataset in favor of *Normal* samples and at the expense of *Attack* samples, we balanced our dataset so as not to be reduced only to *AUC* and *F1* metrics. We can thus use the *Accuracy* metric for the resolution of the multi-objective Wrapper feature selection optimization problem. To do this we have down sampled the *Normal* samples in order to have a 1:1 ratio between the *Normal* and *Attack* samples. Indeed, as Table 3.6, indicates, the CSV datasets of the seven 802.11 specific attacks are mostly made up of *Normal* samples, which can bias the CNN-IDS towards this population of samples.

CSV file attacks 802.11	Normal ratio	Attack ratio
Krack	96,52 %	3,48 %
Evil_Twin	97,23 %	2,77 %
RogueAP	99,93 %	0,07 %
Deauth	97,61 %	2,39 %
Disas	96,27 %	3,73 %
(Re)Assoc	99,70 %	0,3 %
Kr00k	93,38 %	6,61 %

Table 3.6. Proportion between *Normal* and *Attack* samples per CSV file

After the subsampling process, we obtain the final dimension of the dataset with 935,010 samples and 18 features including 467,505 *Normal* samples and 467,505 *Attack* samples distributed between 311,378 *Flooding* samples and 156,127 *Impersonation* samples. Our final dataset is therefore well balanced between *Normal* and *Attack* samples. This will allow BHHO-EAS to exploit the *Accuracy* metric in the objective function values during the search for the best feature subset. But the data provided by the dataset still needs to be encoded to be usable by the CNN-IDS.

D. Data encoding

In order for our dataset to be understandable by the CNN-IDS, we applied the OHE and Min-Max methods to the 18 features. As shown in Table 3.7, the 18 features are divided into 12 categorical features and 6 numerical features. The OHE was applied to the 12 categorical features and Min-Max to the 6 numeric features.

Feature type	Index	Features selected
Categorical feature	1	*radiotap.present.tsft*
	2	*radiotap.channel.freq*
	3	*wlan.fc.type*
	4	*wlan.fc.subtype*
	5	*wlan.fc.ds*
	6	*wlan.fc.frag*
	7	*wlan.fc.retry*
	8	*wlan.fc.pwrmgt*
	9	*wlan.fc.moredata*
	10	*wlan.fc.protected*
	11	*wlan_radio.phy*
	12	*wlan.fc.order*
Numeric feature	13	*frame.len*
	14	*radiotap.length*
	15	*radiotap.dbm_antsignal*
	16	*wlan.duration*
	17	*wlan_radio.signal_dbm*
	18	*wlan_radio.duration*

Table 3.7. The 18 features in technical coherence with the specifications

The preprocessing is now complete. Fig. 3.14 summarizes the 4 stages of the preprocessing. The correlation heatmap of the 18 preselected AWID features will allow us to analyze in section 3.6 the correlation of features selected by BHHO-EAS.

We now have all the main bricks to apply the BHHO-EAS metaheuristic within our Wrapper feature selection method to select the most optimal AWID3 feature subset to maximize the performances of the CNN-IDS and to minimize the features subset size.

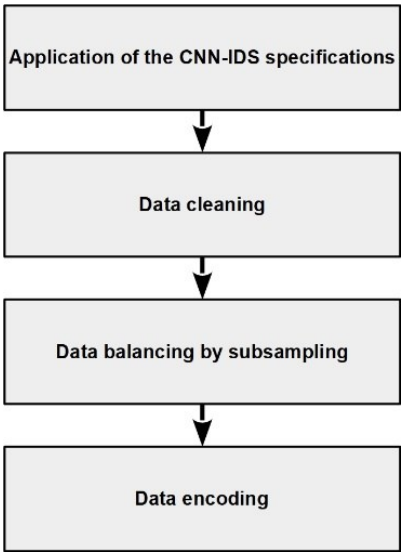


Fig. 3.14. The 4 steps of the AWID3 preprocessing

3.6. Experimental results and discussion

In a desire for clarity, reproducibility and scientific continuity for future works, we will provide in this section all the technical information necessary to reproduce our experiments.

This section provides the experimental results of the application of the metaheuristic BHHO-EAS for solving the Wrapper feature selection multi-objective optimization problem and the design of the CNN-IDS specific to 802.11 attacks. To do this, BHHO-EAS will provide an acceptable solution in a reasonable time to multi-objective optimization problem exposed in subsection 3.5.1. The results of the optimization are provided in Appendix 11. We will too specify in Appendix 11 AWID3 features selected by BHHO-EAS as well as their functions in a company Wi-Fi network.

As we detailed in section 3.3, there are very few works on the AWID3 dataset and only one in the same field and with the same purpose as our work. The works of Chatzoglou, E. et al [Chatzoglou E et al., 2022a], validated by the scientific community and researchers, constitutes very good technical references. In [Chatzoglou E et al., 2022a], the authors created IDSs through supervised training on Machine Learning algorithms combined with upstream feature selection methods. The IDS obtained at the end of the training phase make it possible to classify the test samples among the 3 classes: *Normal*,

Flooding and *Impersonation*. Our work goes beyond that of Chatzoglou E. et al. At the time of writing this manuscript, we are the first to have created and implemented, on the AWID3 dataset, an 802.11 specific IDS design method with a metaheuristic optimization algorithm driving the Wrapper feature selection with a Deep Learning model and exploiting GPU technology. We will, on the one hand, demonstrate the superiority of our Wrapper features selection method driven by BHHO-EAS to that of [Chatzoglou E et al., 2022a] based on expert knowledge and empirical techniques. On the other hand we will compare the performance of our CNN-IDS compared to the 4 IDS having obtained in [Chatzoglou E et al., 2022a] the best results on one of the 3 subsets of the AWID3 dataset: subset of 16 features, subset of 19 features and the subsets of 4 features Set4.

Appendix 12 details the numerical and graphical values of the CNN-IDS performances and the 4 models of [Chatzoglou E et al., 2022a], based on the 5 metrics *AUC*, *F1, Accuracy, Precision* and *Recall*. These metrics are used to comprehensively measure the prediction performances of a Machine Learning and Deep Learning classifier [Ramos-Michel A et al., 2021]. For the main metrics *AUC* and *F1*, the best results are in bold green and the worst in bold red.

We will also provide in Appendix 12 the *Number of features selected*, the *Confusion Matrix* of CNN-IDS as well as, in Fig. A12.3 and Fig. A12.4, its Best and Worst Loss and Accuracy curves obtained on the 10 folds of the stratified cross validation. The *Confusion Matrix* gives the results of the classification in number and in percentage of samples correctly or incorrectly classified. It thus measures the success and prediction error rates.

Finally, in order to prove the ability of our CNN-IDS to be embedded in an IoT with limited resources, we will demonstrate by a technical PoC the performance of the prototype E-CNN-IDS in a Raspberry Pi 4 Model B widely used in the literature and with very limited memory and computing resources.

Thus, we will first present the hardware and software experimental environment. Then we will give in detail, the spatial and temporal complexity parameters of BHHO-EAS, the parameters of the Wrapper feature selection optimization problem and the hyper parameters of the CNN-IDS. In order to better understand our Wrapper feature selection method, we will specify its five main steps illustrated in the architecture diagram in Appendix 9. Finally, we will analyze the metrics and the final solution of BHHO-EAS as well as the prediction performances of CNN-IDS compared to the 4 best IDSs of [Chatzoglou E

et al., 2022a]. Concerning E-CNN-IDS, in terms of prediction of 802.11 specific attacks, we will provide in Appendix 13 its *Confusion Matrix* (Fig. A13.2). To measure the preserving of memory and computing resources, Fig. A13.3 and Fig. A13.4 provide the minimum and maximum consumption of RAM and CPU resources of the Raspberry Pi 4 Model B. Finally, Fig. A13.5 demonstrates the ability of E-CNN-IDS to detect attack vectors belonging to the Flooding and Impersonation classes.

3.6.1. Analytical working station setup and IoT environment

We implemented BHHO-EAS, the CNN-IDS, the Wrapper feature selection and did the experimental tests on an analytical workstation with the following technical specifications:

- **Operating system**: UBUNTU 20.04 LTS 64 bits
- **Hardware environment**:
  - SSD 2,0 To;
  - Processor Intel Core i7, 2,5Ghz, 8 cores, 16 logical processors;
  - RAM 32 Go;
  - NVIDIA GeForce RTX 3060:
    - GPU with a frequency of 1320 MHz
    - 12 GB memory of GDDR6 type and 1750 MHz frequency
- **Software environment:** To design BHHO-EAS, CNN-IDS and to do the preprocessing of the AWID3 dataset we have used:
  - **BHHO-EAS**: *Python* programming language v.3.9.7;
  - **CNN-IDS**: *Scikit-learn* v.1.1.2, *Keras* framework v.2.10.0 and the *Tensorflow* v.2.10.0 platform;
  - **Pre-processing of AWID3 dataset**: *Pandas* v.1.3.4 and *Numpy* v.1.20.3.

The IoT environment represented by the Raspberry Pi 4 Model B has the technical specifications hereafter. Fig. A13.1 in Appendix 13 provides an image of the Raspberry Pi and a description of its electronic components for calculation and memory as well its ports.

- **Constructor:** Raspberry Pi in UK

- **Operating system**: Raspberry Pi OS Debian version 11 32 bits
- **Hardware environment**:
  - SSD 16Go;
  - Processor ARM v8 Quad core Cortex-A72 1.5 Ghz;
  - RAM 2 Go;

3.6.2. Experimental parameters and performance metrics

A. Wrapper feature selection process

To define the spatial and temporal complexity of BHHO-EAS and the Wrapper feature selection multi-objective optimization problem, we have set the following parameters:

- **Number of independent runs N_r**: 30 ;
- **BHHO-EAS population size *N***: 10 (spatial complexity);
- **Number of iteration *ItMax***: 100 (temporal complexity);
- **Values of the weights ω1 and ω2 :** To solve our multi-objective optimization problem with the weighted sum method (or aggregation method), based on the recent work in the literature [Faris H et al., 2018; Too J et al., 2019; Thaher T et al., 2020; Al-Wajih R et al., 2021], the a priori decision of the cybersecurity expert is to give preponderance to detection capability of CNN-IDS over to the size of the selected features subset. So we used $\omega = 0.01$ in order to obtain the couple of weights: $\{\omega 1 = 0.99\,, \omega 2 = 0.01\}$. This couple of weights is a good compromise between our two objectives. It promotes the performance of CNN-IDS without sacrificing its ability to be embedded in an IoT by reducing the number of features.
- **Transfer function and logic rule :** Based on the state of the art of the M1 and M2 modules and the best performance obtained, we have selected the quadratic transfer function Q4 with Rule R2 [Too J et al., 2019].

B. CNN-IDS Architecture

To design our CNN-IDS we used the architecture and supervised learning hyperparameters respectively explained in Table 3.8 and Table 3.9. We recall that the objective of the CNN-IDS is to be sufficiently efficient in the detection of intrusion but also light enough to be embedded in an IoT.

Table 3.10 communicates distribution of the 3 classes *Normal*, *Flooding* and *Impersonation* in the Training, Validation and Test data. As we can see in Table 3.10, although the dataset is balanced between *Normal* class samples and *Attack* samples with a ratio 1:1, we have a ratio 1:2 between *Impersonation* and *Flooding* samples. Consequently, we chose to use the method of Stratified cross validation with k=10 to train and validate the CNN model. This technique will help to better generalize CNN-IDS with our AWID3 dataset and to optimize its prediction performance. The number of fold k=10 is the most used and recommended in the state of the art.

Only Training and Validation samples interests us for the stratified cross validation which represent respectively 60% and 20% of the total samples. Test samples will only be used for the final evaluation of CNN-IDS prediction performance in subsection 3.6.4. During the stratified cross validation process, we merge the Training and Validation samples. These samples are randomly selected and split into 10 folds. Each fold has the same proportions in *Normal*, *Flooding* and *Impersonation* samples. At each iteration, 9 folds will be used for training and 1 fold for validation.

In order to avoid overfitting and to increase the generalization of the CNN-IDS to make it more resistant to the evolution of 802.11 specific attacks, we used the regularization techniques of *EarlyStopping* and *Dropout. EarlyStopping*, with a patience parameter=2, is monitored on the *val_loss* metric. Thus, as soon as the results provided by *val_loss* deteriorate during 2 successive epochs from the last "safe" epoch, the training process will be interrupted and back to the last "safe" epoch.

Dropout is a proven regularization technique for creating classifiers from Deep Learning models. We set an equiprobability of 0.5 to turn off a neuron at each training phase (Dropout rate). This technique thus allows the CNN model to learn to detect attacks by dispensing with half of the neurons randomly switched off in a layer. But this will not be the case during the validation and testing phase. Therefore, the *val_accuracy* metric will be larger than the *Accuracy* metric and the *val_loss* metric will be smaller than the *loss* metric.

Always in the desire to generalize the CNN-IDS as well as possible, we have chosen a large *Batch size* value of 200 samples. Finally, to save the best model during the training on the 10 folds we use the *ModelCheckpoint* technique monitored on the metric *val_accuracy*.

We provide, in Table 3.8, the architecture of the CNN-IDS layer by layer and its diagram in Appendix 10 as well as its 2D representation. The CNN-IDS architecture thus has 0.164M parameters which will be defined after the training phase and requires a computing power of 1.535M Flops. Based on the work of [Monteiro A et al., 2018; Liu J et al., 2020; Gonzalez-Huitron V et al., 2021; Bhosale YH et al., 2022; Joshila Grace LK et al., 2022], our CNN-IDS is able to be embedded and run in a IoT such as Mobile device, Jetson Nano or Raspberry Pi. This is what we prove in the experimental subsection 3.6.4 with a technical PoC.

Name	Type	Maps	Size	Kernel	Stride	Padding	Activation
In	Input	–	14x1	–	–	–	–
C1	Conv	64	12x1	3x1	1	Valid	ReLu
C2	Conv	64	10x1	3x1	1	Valid	ReLu
P3	MaxPool	64	5x1	2x1	2		–
C4	Conv	128	5x1	3x1	1	Same	ReLu
C5	Conv	128	5x1	3x1	1	Same	ReLu
C6	Conv	128	5x1	3x1	1	Same	ReLu
P7	MaxPool	128	2x1	2x1	2	–	–
F8	Flatten	–	256	–	–	–	–
L9	Dense	–	100	–	–	–	ReLu
D10	Droupout 0,5	–	100	–	–	–	–
F11	Dense	–	20	–	–	–	ReLu
Out	Dense	–	3	–	–	–	Softmax

Table 3.8. The CNN-IDS architecture from Input to Output layer

Training hyper parameters	Value
Training data ratio	80%
Final test data ratio	20%
Stratified cross validation	k=10
Batch size	200
Optimizer Adam	learning rate=0.001
	$\beta_1 = 0.9$
	$\beta_2 = 0.999$
EarlyStopping	monitor='loss'
	mode='min'
	patience=2
ModelCheckpoint	monitor='val_accuracy'
	mode='max'

Table 3.9. Hyperparameters for the training

Data category	Class	Value
Training data (90% of 748008)	Normal	336902
	Flooding	224089
	Impersonation	112216
Validation data ratio (10% of 748008)	Normal	37434
	Flooding	24899
	Impersonation	12468
Final Test data (187002)	Normal	93169
	Flooding	62390
	Impersonation	31443

Table 3.10. Distribution of the 3 classes

C. Performance metrics

The performance metrics for the evaluation of the metaheuristic BHHO-EAS, on the 30 independent runs, are ***AVG*** (Average result), ***MIN*** (Minimum or Best result), ***MAX*** (Maximum or Worst result) and ***STD*** (Standard deviation). The ***STD*** metric makes it possible to evaluate the stability and solidity of BHHO-EAS in the search for a good solution to the optimization problem. The ***AVG*** metrics are defined mathematically by equations (3.8) to (3.10). The functions that have been used are: $\boldsymbol{Fit}$ (Fitness), $\boldsymbol{Acc}$ (Accuracy) and $|\boldsymbol{V_b}|$ (Number of selected features).

$$\boldsymbol{AVG_{Fit}} = \frac{\mathbf{1}}{\boldsymbol{N_r}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{N_r}} \boldsymbol{Fit_k} \tag{3.8}$$

$$\boldsymbol{AVG_{Acc}} = \frac{\mathbf{1}}{\boldsymbol{N_r}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{N_r}} \boldsymbol{Acc_k} \tag{3.9}$$

$$\boldsymbol{AVG_{FS}} = \frac{\mathbf{1}}{\boldsymbol{N_r}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{N_r}} |\boldsymbol{V_b}|_{\boldsymbol{k}} \tag{3.10}$$

We provide below the definition and the formula, from (3.11) to (3.16), of the 6 metrics to evaluate the IDSs: *AUC*, *Precision, Recall, F1, Accuracy* and *Convolution matrix*. Knowing that the *AUC* and *F1* metrics will be preponderant. To calculate these metrics for CNN-IDS we use test data that has not been used for its training and validation phases. The values of the 6 metrics for the 4 best IDSs of [Chatzoglou E et al., 2022a] are detailed in their article.

Accuracy (3.11) is the ratio between the number of correctly predicted samples, *Normal* and *Attack*, over the total number of samples.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{FN} + \boldsymbol{TN} + \boldsymbol{FP}} \quad (3.11)$$

Precision (3.12) is the ratio between correctly predicted *Attack* samples and the total number of predicted *Attack* samples. This metric puts the cursor on the number of prediction errors in terms of false positives. The fewer false positives there are, the more the *Precision* tends towards 1.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \quad (3.12)$$

Recall (3.13) is the ratio between the number of correctly predicted *Attack* samples and the true number of *Attack* samples. Unlike *Precision*, *Recall* places the cursor on the number of prediction errors in terms of false negatives. Thus, the fewer false negatives there are, the more the Recall tends towards 1.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}} \quad (3.13)$$

As we can see, Precision and Recall constitute 2 important performance axes of a classifier because they both allow the prediction errors to be evaluated in false positives and false negatives. A performance metric that will combine fairly *Precision* and *Recall* will more measure the quality of a classifier. This is the role of *F1-score* metric (3.14). *F1-score* is a harmonic mean of *Recall* and *Precision*. This metric makes it possible to evaluate the performance of a classifier especially when the dataset is unbalanced between the *Normal* and *Attack* samples. This is the case in [Chatzoglou E et al., 2022a].

$$\boldsymbol{F1} = \frac{\mathbf{1}}{\frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\mathbf{1}}{\boldsymbol{Recall}} + \frac{\mathbf{1}}{\boldsymbol{Precision}}\right)} \quad (3.14)$$

Just like *F1-score*, *AUC* combines two performance metrics or rather two indicators: *Recall* and *Specificity*. In order to define the *AUC* metric, we must first define the *Specificity* (3.15). *Specificity* is the ratio between the *Normal* samples correctly predicted and the true number of *Normal* samples.

$$\boldsymbol{Specificity} = \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}} \tag{3.15}$$

Recall and *Specificity* metrics allow us to plot the *Receiver Operating Characteristic (ROC)* curve. The *ROC* curve is plotted with δ=*1-Speficity* (False positive rate) on the abscissa and θ=*Recall* (True positive rate) on the ordinate. The ROC curve thus makes it possible to graphically materialize the compromise between the False positive rate and the True positive rate. Via the minimum and maximum thresholds, the ROC curve makes it possible to measure the ability of a classifier to discriminate the negative and positive samples.

As *F1-Score*, the area under the *ROC* is an essential metric to evaluate the prediction performance of a classifier, especially when the dataset is unbalanced between the *Normal* and *Attack* samples. This area is called the *Area Under the Curve (AUC)* and is calculated by the formula (3.16). Consequently, the more the number of correct detection increases and the number of false detection decreases, the more the *AUC* will approach 1.

$$\boldsymbol{AUC} = \int_{\boldsymbol{0}}^{\boldsymbol{1}} \boldsymbol{\theta(\delta).d\delta} \tag{3.16}$$

Confusion matrix

The *Confusion matrix* crosses the real classification of the test dataset and the predictions of the IDS. The result of this crossing makes it possible to determine the quantity of correctly classified test samples and to calculate the 5 performance metrics of the IDS defined above.

		Predicted	
		Positif	*Negatif*
True	*Positif*	TP	FN
	Negatif	FP	TN

Among the set of metrics that we have just defined, the pair (*AUC*, *F1-Score*) stands out for its ability to fairly and accurately evaluate the prediction performance of a classifier. In subsection 3.6.4, this couple will be essential to demonstrate the detection qualities of CNN-IDS.

Before analyzing in subsection 3.6.4 the values of the performance metrics obtained following the resolution by BHHO-EAS of the Wrapper feature selection multi-objective optimization problem, we must first describe each of its 5 steps.

3.6.3. The 5 steps to implement Wrapper feature selection process driven by BHHO-EAS

The "black box" optimization problem seen by BHHO-EAS consists of the CNN model and the AWID3 dataset. The Appendix 9 details the architecture of the Wrapper feature selection. Our métaheuristique BHHO-EAS has a population size *N*. Each agent i of this population is positioned by a binary position $\boldsymbol{V_{bi}}$ of dimension d in one of the $2^d - 1$ edges of the hypercube. The null vector is excluded because it does not select any feature, which makes no sense in the context of our work. Our Wrapper feature selection process, driven by BHHO-EAS, is split into 5 steps:

1. V_{bi} creates a new dataset by selecting a feature subset from the AWID3 dataset as described in subsection 3.4.1;
2. The new dataset is divided into *Train*, *Validation* and *Test*;
3. The CNN model is trained with the *Train* data and validated with the *Validation* data. At this step we obtain the accuracy value $\boldsymbol{Acc_i}$;
4. The fitness value $\boldsymbol{Fit_i}$ of the agent i is calculated with $\boldsymbol{Acc_i}$, $|\boldsymbol{Vbi}|$, $\boldsymbol{d}$ and the weight $\boldsymbol{\omega}$;
5. BHHO-EAS uses the $\boldsymbol{N}$ values $\boldsymbol{Fit_i}$ to calculate the next $\boldsymbol{N}$ binary positions $\boldsymbol{V_{bi}}$ with the aim to minimize the objective function.

3.6.4. Results and discussion

A. Results of the Wrapper feature selection multi-objective optimization problem

We recall that the design strategy of the metaheuristic HHO-EAS allows it to have higher performance than HHO for medium and high-dimensional NP-Hard optimization problems. BHHO-EAS inherited the skills of HHO-EAS thanks to our high-level hybridization by means of the two modules M1 and M2 with a sequential execution order. BHHO-EAS therefore retains the mathematical and algorithmic strategy of HHO-EAS.

In this subsection we analyze the resolution by BHHO-EAS of the Wrapper feature selection multi-objective optimization problem. In this experimental phase, we provided BHHO-EAS with an objective function mathematically faithful to what was described in subsection 3.5.1, without first applying the stratified cross validation. Stratified cross validation was applied in the final design phase of the CNN-IDS in order to increase its performance and generalization capabilities for attack detection.

The parameters *Population size*, *Number of Iteration* but also the weights $\omega 1$ and $\omega 2$, are inspired by the work of Khurma RA et al. [Khurma et al., 2020], Jingwei, T. et al. [Too J et al., 2019] and Thaher, T et al. [Thaher T et al., 2020]. We performed 30 independent runs with the aim of obtaining statistical results representative of the performance of BHHO-EAS. From these 30 runs we get the ***AVG***, ***MIN***, ***MAX*** and ***STD*** metrics.

The values of these metrics as well as the convergence curve of BHHO-EAS are provided in Appendix 11 respectively in Table A11.1 to A11.3 and Fig. A11.1

In Table A11.1 BHHO-EAS obtains very good results in the optimization problem with an ***AVG*** of 8.40E-03. Moreover, BHHO-EAS testifies to a solid stability of the results during the 30 runs with a ***STD*** of 3.31E-04. The best result obtained during the 30 runs is represented by the ***MIN*** metric with the value 7.13E-03.

In Table A11.2 and Table A11.3, the metrics for the *Accuracy* and the *Number of selected features* are equally competitive with an ***AVG*** greater than 99% for *Accuracy* and less than 5 for the *Number of selected features*. However, the metric that will be essential for the final design of the CNN-IDS is the ***MIN*** of *Fitness* and the ***MAX*** of *Accuracy* equal to 99.58% because we promote in our work the detection

skills of the CNN-IDS. With the application of the stratified cross validation technique, the detection performance of CNN-IDS will increase.

The convergence curve of BHHO-EAS demonstrates, as for HHO-EAS, that the exploration and exploitation strategy of BHHO-EAS, implemented by the combination of the FIS-REE and the Encirclement and Attacks equations, attributes to it excellent skills in optimizing the NP-Hard Wrapper feature selection optimization problem. Indeed, this allows BHHO-EAS to have a smart distribution of the exploration and exploitation phases during the iterations, to adapt the search for a better solution to each iteration while maintaining a good balance between exploration and exploitation. In addition, it allows BHHO-EAS to avoid being blocked in a local optimum. The convergence rate and the absence of premature convergence until the end of the iterations are proof of this.

Finally, in the binary search space, BHHO-EAS manages to obtain a very good solution represented by the ***MIN*** value of *Fitness* and ***MAX*** of *Accuracy*. ***MIN*** is obtained with the binary position $\widehat{Vb} = \{1,1,1,0,0,0,0,0,0,1,0,0,1,0,1,0,1,1\}$. $\widehat{Vb}$ selects 8 features among the 18 preselected features in subsection 3.5.4: $\{1,2,3,0,0,0,0,0,0,10,0,0,13,0,15,0,17,18\}$. This solution is balanced between the categorical features and numeric features: the features $\{1,2,3,10\}$ are categorical and $\{13,15,17,18\}$ are numerical. Based on the literature [Chatzoglou E et al., 2022a; Chatzoglou E et al., 2022b; University of the Aegean,] and the specialized documentation at Wireshark, we have provided in Table A11.4 the role of the 8 features in a company Wi-Fi network and their interpretation in the detection of 3 classes *Normal*, *Flooding* and *Impersonation*.

Furthermore, if we refer to the correlation heatmap in Fig. 3.13, the 8 features selected by BHHO-EAS are very weakly correlated, apart the feature *radiotap.present.tsft* which has a correlation with *wlan.fc.type* greater than 0.8 in absolute value. This again attests to the quality of the solution obtained by BHHO-EAS.

To design the CNN-IDS we used the binary solution $\widehat{Vb}$, its architecture in Table 3.8 and the hyperparameters in Table 3.9 by applying stratified cross validation.

B. Experimental results of the CNN-IDS performances

In [Chatzoglou E et al., 2022a], due to the great imbalance of their datasets, the authors legitimately prioritized the metrics *AUC*, *F1* and *Confusion matrix* to evaluate the performance of their 4 best IDSs. Indeed, as specified in Table 3.6, the huge disproportion in [Chatzoglou E et al., 2022a] of the *Normal* samples to the detriment of the *Attack* samples has the effect of distorting the results of the 3 metrics *Precision*, *Recall* and *Accuracy*. This is not the case with our dataset, which we previously balanced between the *Normal* and *Attack* samples.

As a result, for a fair comparison between our CNN-IDS and the best 4 IDSs of [Chatzoglou E et al., 2022a], the performance measures will only be based on the metrics *AUC*, *F1* and *Confusion matrix*. However, as we have specified above, the *Precision*, *Recall* and *Accuracy* metrics have not been neglected and have also been treated and provided in Appendix 12 in Table A12.1 and in Fig. A12.1.

In [Chatzoglou E et al., 2022a], the 4 IDS having obtained at least a maximum value on one of the 2 metrics *AUC* and *F1* are:

- ET and LightGBM on the AWID3 subset of 16 features;
- ET on the AWID3 subset of 19 features;
- DT on the AWID3 subset of 4 features named Set4.

We named these 4 IDS ET_16f, LightGBM_16f, ET_19f and DT_4f_Set4 respectively.

We specify that, as in [Chatzoglou E et al., 2022a], all the performance values provided in Appendix 12 are the averages of the *AUC*, *F1* and *Confusion Matrix* values over the 10 folds of the stratified cross validation. The same is true for *Precision*, *Recall* and *Accuracy*. For completeness, we have also included in Appendix 12 the best and the worst *Accuracy* - *Loss* curves of CNN-IDS among the 10 fold respectively in Fig. A12.3 and Fig. A12.4.

We can see in Table A12.1 and in Fig. A12.1 that the CNN-IDS, with the 8 features selected by BHHO-EAS, demonstrates its skills detection of the 3 classes by obtaining the best values for the *AUC* and *F1* metrics. The *AUC* of CNN-IDS reaches an average value of 99.98 and the F1 the average value of 99.78. Without real surprise DT_4f_Set4 obtains the worst results with an average AUC of 97.16 and an average F1 of 95.19.

The Confusion Matrix in Fig. A12.2 confirms our conclusions above. The 4 IDS of [Chatzoglou E et al., 2022a] detect on average almost 100% of the *Normal* samples. On the other hand, the average detection of the classes *Flooding* and *Impersonation* is less good with 99.44% and 98.43% for ET_16f, 98.97% and 98.71% for LightGBM_16f, 98.60% and 98.37% for ET_19f and 97.94% and 89.25% for DT_4f_Set4. This detection imbalance between the 3 classes testifies to a lack of generalization and an overfitting in favor of the *Normal* class.

Our CNN-IDS, contrary to the 4 IDS of [Chatzoglou E et al., 2022a], demonstrates very competitive average detection rates, balanced and generalized on the 3 classes with values higher than 99.5%. Our CNN-IDS obtains detection rates of 99.56% for the *Normal* class, 99.97% for the *Flooding* class and 99.95% for the *Impersonation* class.

Finally, as we specified in subsection 3.4.1, another virtue of feature selection is the reduce of the computational load of the training, validation and testing phases as well as the number of Epochs during the training phase thanks to the reduction in the size of the dataset. It took an average of 8.6 Epoch out of the 10 folds to train the CNN-IDS. In Fig. A12.3 and Fig. A12.4 we notice that for the best *Accuracy - Loss* curves it took only 11 Epoch. And for the worst it took 6 Epoch, which is insufficient to have competitive performance.

We can deduct from the performances provided by *AUC*, *F1* and the *Confusion Matrix*, that our CNN-IDS has better generalization and detection capacities of the 3 classes than the 4 best IDS of [Chatzoglou E et al., 2022a], and this only with the 8 selected AWID3 features by BHHO-EAS. In addition, the very low number of 8 features to be processed by the CNN-IDS for the detection of 802.11 specific attacks greatly facilitates its ability to be embedded in an IoT and will increase its reaction capacity. Indeed, the number of parameters of the CNN-IDS and the amount of Flops provided in subsection 3.6.1 as well as the next experimental subsection C confirms this.

C. Experimental results of the E-CNN-IDS performances embedded in IoT

In order to assess the performance of the E-CNN-IDS in a real IoT environment and compared them to the *Confusion matrix* of CNN-IDS we proceeded as follows. We have configured E-CNN-IDS as a Linux service in the Raspberry Pi OS environment. This service is represented by the *e_cnn_ids.service*

file. Thus, E-CNN-IDS will be active as soon as the Raspberry Pi is started and will constantly search for 802.11 specific attacks. A laptop computer, connected to the same company Wi-Fi network, simulates the role of the hacker. Via this laptop we sent to the Raspberry Pi 4 the same quantity of attacks as during the test phase of the CNN-IDS detailed in Table 3.10.

The action of E-CNN-IDS following the detection of an attack is an alert dedicated to the administrator composed of 4 fields:

- The date,
- The time in the format *HH:MM:SS*,
- The class of attack,
- The source IP of the Hacker's STA.

Fig 44 illustrates an example of how E-CNN-IDS works in the face of 6 successive attacks: *Disas*, *Disas*, *Disas*, *Evil_Twin*, *(Re)Assoc* and *(Re)Assoc*. These 6 attacks were correctly detected and classified by E-CNN-IDS. We have blurred the Hacker's STA IP address for obvious computer security reasons.

At the end of the test phase we obtained the *Confusion matrix* of the E-CNN-IDS. But being in an IoT environment with very limited computing resources, we also evaluated the minimum and the maximum consumption of memory and CPU resources. All the experimental results of this PoC are provided in Appendix 13.

In terms of predictions of 802.11 specific attacks, we obtained very good performances materialized by the *Confusion matrix* of E-CNN-IDS in Fig. A13.2. Overall, these results are very close to those obtained for the CNN-IDS. Compared to the *Confusion matrix* of CNN-IDS, a slight performance decrease is observed for the *Flooding* and *Impersonation* attack classes. For the *Flooding* attack class, we note +0.11% false negative and -0.11% true positive. And for the *Impersonation* attack class, we have +0.2% false negative and -0.19% true positive. Despite this, E-CNN-IDS demonstrates, like CNN-IDS, better performance than the 4 best IDS of [Chatzoglou E et al., 2022a] as well as a much more generalized ability to predict.

In terms of hardware resource consumption, we can see in Fig 42 and Fig 43 that E-CNN-IDS consumes 1.4% of RAM and between 0.4% and 0.6% of CPU resources.

Our PoC experimentally demonstrates thus the excellent optimization performance of BHHO-EAS which is derived from the hybridization of HHO-EAS. Indeed, BHHO-EAS is able to select the most relevant features of the AWID3 dataset, thus maximizing the attack prediction performances of CNN-IDS and reducing its complexity so that it can be embedded in an IoT with limited computing and memory resources.

3.7. Conclusion

This chapter is the application of HHO-EAS to the field of IoT cybersecurity in a company information system. The main scientific contributions of our work are threefold.

The first contribution concerns the field of metaheuristic algorithms. We designed the new metaheuristic BHHO-EAS, hybrid of HHO-EAS, for solving the Wrapper feature selection multi-objective optimization problems in a binary search space. In order to make BHHO-EAS benefit from the mathematical and algorithmic strategy of HHO-EAS, we performed a high-level hybridization by integrating the *M1* and *M2* modules into HHO-EAS.

The second contribution is the creation of a new Wrapper feature selection method to create a CNN-IDS through a metaheuristic optimization method. The CNN-IDS obtained at the end will be an essential component of a company information system's cybersecurity strategy, efficient against 802.11 specific attacks and light enough to be embedded in an IoT with limited computing and memory resources. To do this, we have made significant changes to the classic Wrapper feature selection method by using not a classic Machine Learning algorithm as in the literature, but a Deep Learning algorithm and GPU technology for calculating the values of the objective function. Deep Learning algorithms have mathematical, algorithmic and technical specificities that make them more complex to implement in the Wrapper feature selection method. Consequently, we find in the literature Wrapper feature selection methods exploiting Machine Learning algorithms such as K-NN or SVM for their ease of implementation. Our Wrapper feature selection method is obviously compatible with classic Machine Learning algorithms, but above all it is able to integrate Deep Learning algorithms in order to benefit from their superior classification skills, as well as GPU technology for value calculation of the objective function of each agent in the population of BHHO-EAS. To design our IDS, we chose the Deep Learning

CNN algorithm for two main reasons. The first is motivated by the performance of CNN recognized in the literature in the detection of known and unknown attacks. The second is for its inherent properties in its architecture and operation allowing to embed a CNN in an IoT.

The third contribution, which is not the least, concerns the AWID3 dataset. Our work is the first, at the time of writing this manuscript, to have applied a feature selection process by metaheuristic optimization driven by BHHO-EAS on the AWID3 dataset, much more complex than its predecessor AWID2.

The resolution of the Wrapper feature selection multi-objective optimization problem by BHHO-EAS allowed us to obtain a binary solution $\widehat{Vb}$ representing a subset of 8 features among the 253 of the AWID3 dataset. We exploited this binary solution to design our CNN-IDS. In order to demonstrate the superiority of our method, we compared it to the work of Chatzoglou E. et al. [Chatzoglou E et al., 2022a] validated by the scientific community and having the same purposes as ours. Our CNN-IDS, that complies with the 5 requirements of the specifications formulated in section 3.2, obtains the best *AUC* and *F1* performance compared to the 4 best IDS of [Chatzoglou E et al., 2022a]. In addition, the generalization capabilities of CNN-IDS allow it to better classify the attack vectors with less than 0.05% error in the 3 classes *Normal*, *Flooding* and *Impersonation*.

In order to demonstrate the capabilities of CNN-IDS to be embedded in an IoT environment with very limited computing and memory resources, we embedded its prototype, E-CNN-IDS, in a Raspberry Pi 4 Model B. The technical PoC of our work provided a *Confusion matrix* as competitive as that obtained for the post training test phase of the CNN-IDS. In addition, CPU and RAM monitoring proves that the prototype E-CNN-IDS consumes very few CPU and RAM resources. These experimental results demonstrate the superiority and efficiency of our method compared to that of Chatzoglou E. et al.[Chatzoglou E et al., 2022a], with BHHO-EAS as the central actor.

General conclusion

The growing ubiquity of IoT systems exploiting Wi-Fi within companies' information systems and state administrations, increases the attack surface of hackers. Thanks to 802.11 specific vulnerabilities, hackers manage to use IoT systems as gateways to extend their reach to wireless and wired networks and then conclude their attack with an elevation of their privileges. Thus, thanks to its vulnerabilities, hackers are in a strategically advantageous position to maintain their access to the information system, exploit its resources as they please and execute disastrous attacks for the sustainability of the company. Faced with these risks of growing cyber threats, research for the design of Intrusion Detection Systems (IDS) using Artificial Intelligence techniques is a top priority for the research community. The work explained in this thesis manuscript makes contributions to three crucial research areas: the metaheuristic algorithms, the feature selection algorithms for the benefit of the design of IoT Intrusion Detection Systems in a company information system, and for the first time in the literature, a feature selection by metaheuristic optimization on the Wi-Fi attack dataset named AWID3, much more complex than its predecessor AWID2.

Our first contribution aims to design a metaheuristic able to deal with NP-Hard optimization problems by exploiting new bio-inspired models. Thus, we have significantly increased the performances of the HHO metaheuristic thanks to an unprecedented hybridization strategy entirely inspired by the win-win hunting synergy between the crows and the wolves. This made it possible to design the new metaheuristic HHO-EAS, which is much more efficient than HHO for high-dimensional and highly multimodal optimization problems. Then, we designed the new metaheuristic BHHO-EAS, hybrid of HHO-EAS, for solving the Wrapper feature selection NP-Hard multi-objective optimization problems in a binary search space. In order to make BHHO-EAS benefit from the mathematical and algorithmic strategy of HHO-EAS, we performed a high-level hybridization by integrating the M1 and M2 modules into HHO-EAS.

The second contribution, for the benefit of IoT cybersecurity, is the creation of a new Wrapper feature selection method via a metaheuristic optimization method. To accomplish this, we've made two important changes to the classic Wrapper feature selection method. We've built in the ability to leverage Deep Learning algorithms and GPU technology. Thus, our Wrapper feature selection method is able to

integrate all the Deep Learning algorithms in order to benefit from their predictive skills superior to traditional Machine Learning algorithms commonly observed in the literature. And in order to increase the computing power, our method can exploit the CPUs and GPUs according to the stage of the metaheuristic algorithm as illustrated by the flowchart of the metaheuristic BHHO-EAS in Appendix 8. This new method allowed us to create a CNN-IDS as a vital component of the cybersecurity strategy of a company information system integrating a Wi-Fi environment. The CNN-IDS is thus efficient in the prediction of 802.11 specific attacks and light enough, in a green computing logic, to be embedded in an IoT with limited computing and memory resources.

The third contribution, which is not the least, concerns the AWID3 dataset. Our work is the first, at the time of writing this thesis manuscript, to have applied a feature selection process on the AWID3 dataset by metaheuristic optimization. In our works, the feature selection process is driven by BHHO-EAS on the AWID3 dataset, much more complex than its predecessor AWID2.

In order to achieve these three major scientific contributions, our thesis has been structured in two phases.

In the first phase of this thesis, our ambition was to rely on a very recent metaheuristic, based on the population, with very few parameters to configure in order to reduce the curse of the parameters and enough leeway to integrate a sophisticated hybridization. The HHO metaheuristic satisfied all of its criteria. Then to design our hybridization strategy, we looked for a bio-inspired, intelligent and adaptive model, having proven its effectiveness in the complex search for good nutritious prey which offers a greater probability of survival to predators in a very hostile environment. Our research led us to the hostile regions of Eastern Europe and the state of Wyoming in the United States where an unexpected alliance between crows and wolves was observed in winter. To cope with the long and harsh winters, the extraordinary hunting synergy between crows and wolves allowed them to find very promising prey that was sufficiently consistent to feed these two predators. We modeled this hunting synergy by a Mamdani-type Fuzzy Inference System named FISREE and the Encirclement and Attack equations. Then we integrated them into HHO to create the new metaheuristic HHO-EAS. FISREE models the exploration strategy of the crows and significantly improves the distribution strategy of the exploration and exploitation phases compared to that of HHO. The Encirclement and Attack equations modeling the

attack technique of the wolves make it possible on the one hand to prevent premature convergence when agents gather too early near a local optimum, on the other hand to upgrade the Harris Hawk positions in a more promising area. The exploitation and the exploration performances as well as the balance between these two phases in HHO-EAS have been evaluated on a two benchmarks: a general and a specific. The general benchmark consisting of 19 functions allowed us to validate our bio-inspired hybridization strategy and to validate the exploration and exploitation capabilities of HHO-EAS overall more efficient than that of HHO as well as that of GWO and PSO. The performances of HHO-EAS are manifested all the more for multimodal and composite functions in search spaces of high dimensions 100 and 1000. In addition, HHO-EAS has a better convergence rate, a best scalability on all dimensions and generally gives the assurance of having a better quality solution faster than HHO. The specific benchmark allowed us to focus on the performance superiority of HHO-EAS compared to HHO in the 20 most complex environments and high-dimensional of the CEC2017 close to real-life optimization problems. Our bio-inspired hybridization strategy allowed HHO-EAS to demonstrate a success of 90% on hybrid functions and 90% on composite functions in a search space of dimension 100. As well as the results obtained on the general benchmark, the specific benchmark makes it possible once again to experimentally validate the exploration and exploitation strategy implemented in HHO-EAS inspired by the hunting synergy between the crows and the wolves.

The most important result that emerges from the first phase of this thesis and which served us in the second phase, is that HHO-EAS is better suited than HHO to deal with the real life optimization problems as evidenced by experimental results for dimensions 100 and 1000. Indeed, the real life optimization problems are generally highly multimodal, high-dimensional and NP-Hard such as feature selection optimization problems.

In the second phase of this thesis, our objectives are focused on the cybersecurity of IoT exploiting Wi-Fi within the companies' information systems and the state administrations. More specifically the design of an efficient, scalable and light enough CNN-IDS to be embedded in an IoT. In the context of Wrapper feature selection and supervised learning of a CNN, the feature selection of the AWID3 dataset and the design of the CNN-IDS is mathematically modeled by a multi-objective optimization problem. The HHO-EAS's performances are the keystone of the culmination of this second phase of our thesis.

The choice of the Deep Learning CNN algorithm to design the CNN-IDS is based on a tactical choice aimed at satisfying cybersecurity imperatives and technical constraints. The imperatives are satisfied by the performance of CNN recognized by the literature in the detection of known and unknown attacks. And the respect of the technical constraints is provided by the inherent properties of CNN's architecture and its operation allowing it to be embedded in an IoT. This demonstrates the relevance of our tactical choice to effectively satisfy the two main objectives of the second phase of our thesis: CNN-IDS is exploited for 802.11 specific attack prediction and to be embedded in an IoT with limited computing and memory resources.

With CNN, having the first component of our Wrapper feature selection process, we then proceeded to the analysis and processing of the second component, the new AWID3 dataset. The AWID3 dataset comes from the research work of the Greek Aegean University in Wi-Fi cybersecurity within a company information system. Much more complex and elaborate than its predecessor AWID2, AWID3 is made up of 253 features and a label to identify the attacks. Additionally, AWID3 encompasses all seven layers of the OSI stack by integrating 802.11 specific attacks as well as application attacks. In an attack Wi-Fi detection and blocking strategy at the Data Link Layer level, we first performed a preprocessing phase crucial to the prediction performance of the CNN-IDS. This preprocessing aims to select and clean the 802.11 features related to the Data Link Layer in accordance with our specifications based on the knowledge of information system experts. In order to have a balanced dataset between Attack and Normal records, we sub-sampled the records labeled Normal. We then applied an encoding of categorical features with One-Hot Encoding (OHE) and a normalization of numerical features with Min-Max. At this stage of the preprocessing the dataset is ready to be integrated into our Wrapper feature selection process.

The third and final major component of our Wrapper feature selection process is the HHO-EAS metaheuristic. In order to allow HHO-EAS to drive our Wrapper feature selection process, we hybridized it and designed the new Binary HHO-EAS (BHHO-EAS) metaheuristic which is able to exploit the binary discrete search space. BHHO-EAS resulted in the selection of the subset of the 8 most relevant features among the 253 features in the AWID3 dataset. BHHO-EAS thus made it possible to

maximize the prediction performance and the robustness of the CNN-IDS while minimizing its complexity in order to be embedded in an IoT.

To demonstrate the superiority of our method, we compared it to the works of Chatzoglou E. et al. in [Chatzoglou E et al., 2022a]. Their work are the only ones in the literature to be in the same dominants as ours. For a fair comparison between the best 4 IDS of [Chatzoglou E et al., 2022a] and our CNN-IDS, performance was assessed via *AUC* and *F1* metrics as well as the *Confusion Matrix* and the number of selected feature from AWID3. However, we have not neglected the *Accuracy*, *Precision* and *Recall* metrics in our work. Our CNN-IDS, that complies with the 5 requirements of the specifications formulated in the third chapter and with only 8 AWID3 features, obtains the best *AUC* and *F1* performance compared to the 4 best IDS of [Chatzoglou E et al., 2022a]. The *AUC* reaches an average value of 99.98 and the *F1* the average value of 99.78. In addition, the generalization capabilities of CNN-IDS allow it to better classify the attack vectors with less than 0.05% error on the 3 classes *Normal*, *Flooding* and *Impersonation*. Besides, in order to prove the ability of our CNN-IDS to be embedded in an IoT environment with limited computing and memory resources, we have integrated a prototype of CNN-IDS into a Raspberry Pi 4. This prototype, named E-CNN-IDS, demonstrated prediction performances very close to those of CNN-IDS with a satisfactory preservation of memory and calculation resources. These experimental results demonstrate the superiority and efficiency of our method with BHHO-EAS as the central actor.

The results of the work of our thesis inaugurate three very promising perspectives in the fields of metaheuristic algorithms, feature selection and cybersecurity.

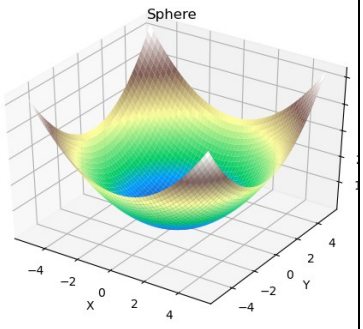
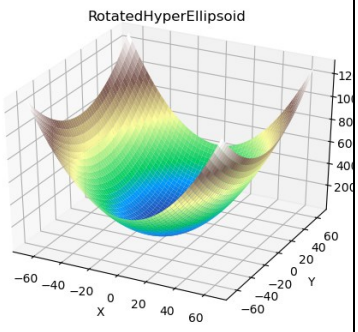
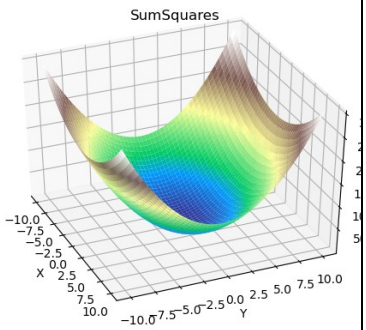
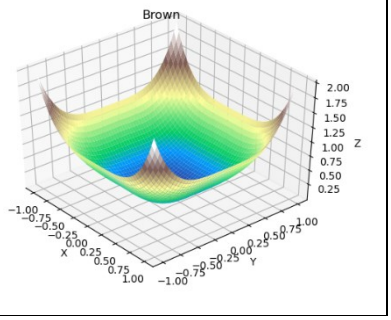
In the field of metaheuristic algorithms, we will conduct two works. In HHO-EAS, the amplitude range provided by FISREE is in [0.5,2.5]. As a result, the distribution of the exploration and exploitation phases is static whatever the optimization problem. Our first work will thus aim to enable HHO-EAS to dynamically adapt the management of the exploration and exploitation phases to the complexity of each optimization problem. To this end, since Reinforcement Learning algorithms provide the algorithms that employ them with the ability to learn and self-adapt to their environment [60], we will integrate into FISREE a Reinforcement Learning module in order to dynamically adapt the amplitude range to

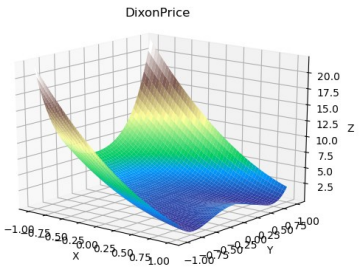
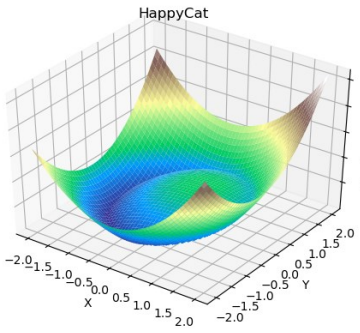
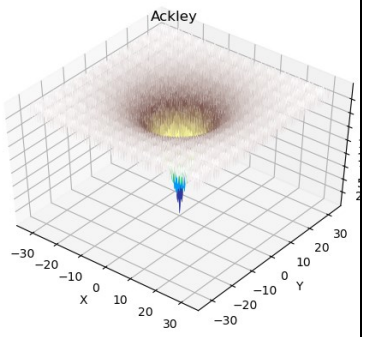
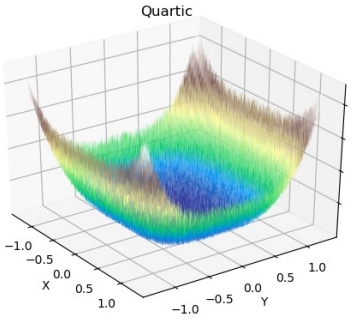
optimization problems. At the end of this work, we will compare the performance of this new metaheuristic named DHHO-EAS (Dynamic HHO-EAS) to HHO-EAS.

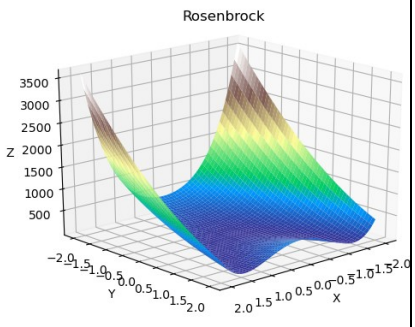
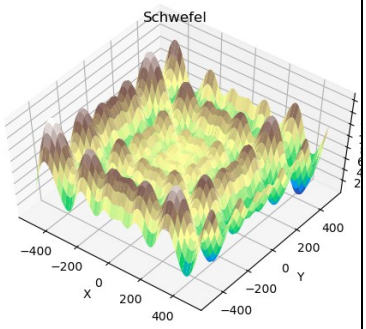
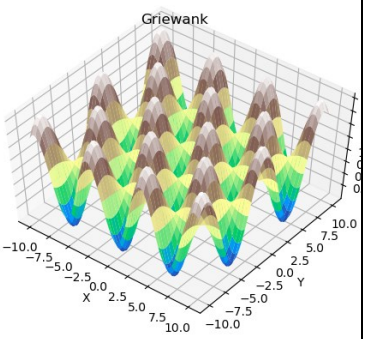
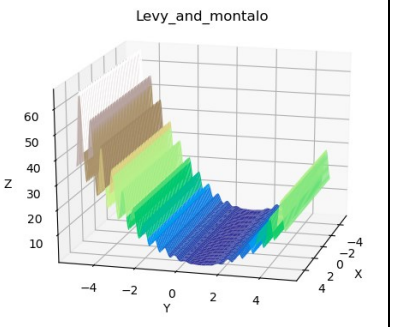
Our second work will adapt HHO-EAS to the context of multi-objective optimization by creating the metaheuristic MHHO-EAS (Multi-objective HHO-EAS). This new metaheuristic will allow us to exploit the solutions that have a relationship of dominance over the others and none between them, thus constituting the Pareto optimal Front. We will compare the results obtained by MHHO-EAS in an approach A Posteriori with those obtained by BHHO-EAS using an approach A Priori.

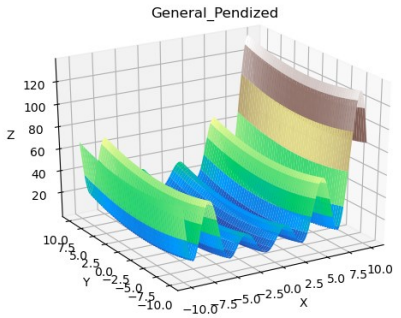
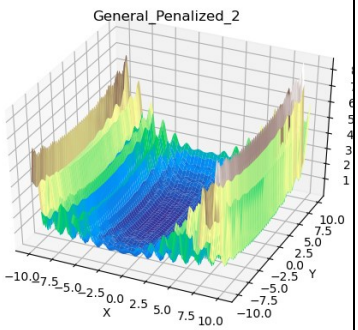
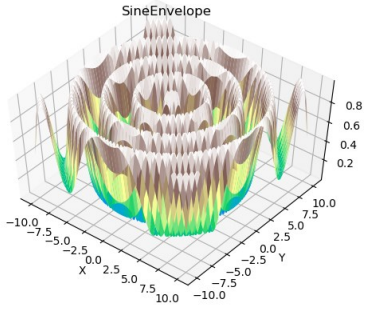
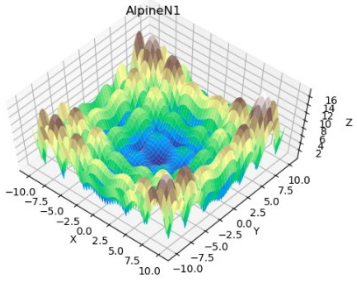
In the fields of feature selection and cybersecurity, our Wrapper feature selection method will allow us to extend the spectrum of our work to the benefit of UTMs (Unified Thread Management) in companies' information systems and state administrations. In this our works, we used the Deep Learning algorithm CNN and the AWID3 dataset. However, our Wrapper feature selection method is sufficiently modular to integrate metaheuristics, datasets and Deep Learning algorithms different from ours. Our method could therefore be applied to other fields than cybersecurity such as health, chemistry, electronics, etc. We will use the modularity of our Wrapper feature selection method in our third work to create an IDS dedicated to UTM. Our future IDS design method will therefore exploit a broader spectrum of artificial intelligence algorithms. We will implement metaheuristic optimization combined with Deep Learning algorithms RNN such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) and Reinforcement Learning algorithms to take into account the time dimension. This IDS will not only be able to protect wired networks from known and unknown attacks but also wireless networks such as Wi-Fi and 5G.

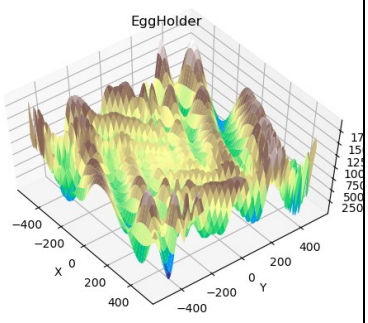
Appendix 1: General benchmark functions

Name	Function	3D plot
Sphere U1	$f(\mathrm{x}) = \sum_{i=1}^{n} x_i^2$	
Rotated Hyper Ellipsoid U2	$f(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	
Sum Squares U3	$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$	
Brown U4	$f(\mathbf{x}) = \sum_{i=1}^{n-1} (x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}$	

Dixon-Price M1	$f(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$	
Happy Cat M2	$f(\mathrm{x}) = [(\|\|\mathrm{x}\|\|^2 - n)^2]^{\alpha} + \frac{1}{n}(\frac{1}{2}\|\|\mathrm{x}\|\|^2 + \sum_{i=1}^{n} x_i) + \frac{1}{2}$	
Ackley M3	$f(\mathrm{x}) = -a.exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)\right) + a + exp(1)$ $a = 20, b = 0.2\ and\ c = 2\pi$.	
Quartic M4	$f(\mathbf{x}) = \sum_{i=1}^{n} ix_i^4 + \mathrm{random}[0,1]$	

Rosenbrock M5	$f(x) = \sum_{i=1}^{n-1} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$ a = 1 and b = 100	
Schwefel M6	$f(x) = -\sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$	
Griewank M7	$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	
Levy and montalo M8	$f(x) = \sin^2(\pi w_1 + \sum_{i=1}^{n-1} (w_i - 1)^2 (1 + 10\sin^2(\pi w_i + 1)) +$ $(w_n - 1)^2 (1 + \sin^2(2\pi w_n))$ $w_i = 1 + \frac{x_i - 1}{4}$	

General penalized 1 M9	$\frac{\pi}{D}\left[10\sin(\pi.y_1)^2+\sum_{i=1}^{n-1}\{(y_i-1)^2.(1+10\sin(\pi.y_{i+1})^2)\}+(y_n-1)^2\right]+\sum_{i=1}^{n}u_i(x_i,a,k,m)$ $\begin{cases} y_i=1+\frac{1}{4}(x_i+1) \\ u_i(x_i,a,k,m)=\begin{cases} k(x_i-a)^m & if\ x_i>a \\ 0 & if\ -a\le x_i\le a \\ k(-x_i-a)^m & if\ x_i<-a \end{cases} \\ a=10, m=4, k=100 \end{cases}$	
General penalized 2 M10	$\frac{1}{10}.\left[\sin(3\pi.y_1)^2.\sum_{i=1}^{n-1}\{(y_i-1)^2.(1+10\sin(3\pi.y_{i+1})^2)\}\right]+(y_n-1)^2.(1+\sin(2\pi.y_n)^2)+\sum_{i=1}^{n}u_i(x_i,a,k,m)$ $\begin{cases} y_i=1+\frac{1}{4}(x_i+1) \\ u_i(x_i,a,k,m)=\begin{cases} k(x_i-a)^m & if\ x_i>a \\ 0 & if\ -a\le x_i\le a \\ k(-x_i-a)^m & if\ x_i<-a \end{cases} \\ a=10, m=4, k=100 \end{cases}$	
Sine envelope M11	$f(\mathbf{x})=-\sum_{i=1}^{n-1}(0.5+\frac{\sin(\sqrt{x_{i+1}^2+x_i^2}-0.5)^2}{(0.001(x_{i+1}^2+x_i^2)+1)^2}$	
Alpine N1 M12	$f(\mathbf{x})=\sum_{i=1}^{n}\lvert x_i sin(x_i)+0.1x_i\rvert$	

EggHolder M13	$f(\mathbf{x}) = -\sum_1^{n-1}((x_{i+1}+47)sin\left(\sqrt{\lvert x_{i+1}+0.5x_i+47\rvert}\right)+$ $x_i sin\left(\sqrt{\lvert x_i-(x_{i+1}+47)\rvert}\right))$	

Appendix 2: Optimisation results of HHO-EAS against HHO, GWO and PSO on the general benchmark of 19 functions for dimension 2, 30,100 and 1000

Table A2.1: Test results of HHO-EAS against HHO, GWO and PSO on a benchmark of 19 functions for dimension 2

ID	*Functions*	*Metrics*	*HHO-EAS*	*HHO*	*GWO*	*PSO*
U1	***Sphere***	AVG	2.23E-277	2.02E-87	**2.85E-288**	3.95E-122
		STD	**0.00E+00**	1.30E-86	**0.00E+00**	1.97E-121
		Min	1.45E-307	6.70E-107	**0.00E+00**	1.31E-126
		Max	1.11E-275	9.23E-86	**1.42E-286**	1.41E-120
U2	***Rotated HyperEllipsoid***	AVG	3.47E-274	1.81E-84	**1.03E-287**	1.55E-121
		STD	**0.00E+00**	1.26E-83	**0.00E+00**	3.78E-121
		Min	6.12E-304	3.46E-106	**0.00E+00**	1.55E-125
		Max	1.74E-272	9.02E-83	**5.07E-286**	1.99E-120
U3	***Sum Squares***	AVG	**9.30E-282**	8.05E-89	1.85E-280	4.51E-121
		STD	**0.00E+00**	5.51E-88	**0.00E+00**	1.57E-120
		Min	2.63E-310	5.27E-106	**4.94E-324**	1.58E-126
		Max	3.73E-280	3.93E-87	**8.26E-279**	1.05E-119
U4	***Brown***	AVG	1.34E-274	5.61E-90	**1.66E-296**	1.15E-121
		STD	**0.00E+00**	3.91E-89	**0.00E+00**	6.67E-121
		Min	6.78E-307	6.21E-109	**0.00E+00**	1.48E-127
		Max	6.68E-273	2.79E-88	**5.45E-295**	4.75E-120
M1	***Dixon-Price***	AVG	1.07E-07	5.97E-08	1.29E-07	**3.70E-32**
		STD	4.35E-07	2.33E-07	1.51E-07	**0.00E+00**
		Min	2.36E-17	4.93E-32	1.24E-09	**3.70E-32**
		Max	2.72E-06	1.61E-06	7.70E-07	**3.70E-32**
M2	***Happy Cat***	AVG	3.30E-04	7.41E-04	3.43E-05	**2.39E-05**
		STD	5.06E-04	1.11E-03	**2.70E-05**	5.74E-05
		Min	1.00E-06	7.02E-08	1.82E-06	**1.46E-08**
		Max	2.99E-03	6.23E-03	**1.31E-04**	3.17E-04
M3	***Ackley***	AVG	**4.44E-16**	**4.44E-16**	**4.44E-16**	**4.44E-16**
		STD	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
		Min	**4.44E-16**	**4.44E-16**	**4.44E-16**	**4.44E-16**
		Max	**4.44E-16**	**4.44E-16**	**4.44E-16**	**4.44E-16**
M4	***Quartic***	AVG	1.26E-04	**1.03E-04**	1.58E-04	2.23E-04
		STD	1.16E-04	**1.09E-04**	1.16E-04	1.51E-04
		Min	**2.47E-06**	4.40E-06	5.68E-06	7.40E-06
		Max	4.32E-04	**4.28E-04**	5.02E-04	7.97E-04
M5	***Rosenbrock***	AVG	4.55E-09	5.13E-07	6.52E-07	**4.12E-20**
		STD	1.74E-08	9.69E-07	6.99E-07	**2.88E-19**
		Min	3.15E-13	**0.00E+00**	1.29E-08	**0.00E+00**
		Max	1.22E-07	4.00E-06	3.72E-06	**2.06E-18**
M6	***Schwefel***	AVG	**2.37E+00**	2.14E+01	4.74E+01	9.91E+01
		STD	**1.66E+01**	4.55E+01	6.70E+01	7.92E+01
		Min	**2.55E-05**	**2.55E-05**	3.69E-05	**2.55E-05**
		Max	**1.18E+02**	**1.18E+02**	2.37E+02	2.37E+02
M7	***Griewank***	AVG	**0.00E+00**	**0.00E+00**	3.06E-03	5.57E-03
		STD	**0.00E+00**	**0.00E+00**	3.77E-03	8.01E-03
		Min	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
		Max	**0.00E+00**	**0.00E+00**	9.87E-03	4.68E-02
M8	***Levy and montalo***	AVG	5.80E-10	7.13E-08	1.99E-06	**2.90E-30**

		STD	1.47E-09	2.13E-07	5.75E-06	**0.00E+00**
		Min	4.42E-18	**2.90E-30**	2.77E-14	**2.90E-30**
		Max	8.50E-09	1.38E-06	2.74E-05	**2.90E-30**
M9	***General penalized 1***	AVG	2.66E-10	3.87E-08	4.07E-08	**2.36E-31**
		STD	6.89E-10	6.26E-08	3.81E-08	**0.00E+00**
		Min	9.12E-16	2.06E-13	7.46E-11	**2.36E-31**
		Max	3.85E-09	2.60E-07	1.54E-07	**2.36E-31**
M10	***General penalized 2***	AVG	5.26E-10	3.25E-08	4.83E-08	**1.35E-32**
		STD	1.05E-09	7.62E-08	4.25E-08	**2.74E-48**
		Min	5.71E-15	2.62E-21	8.46E-10	**1.35E-32**
		Max	5.79E-09	4.21E-07	1.91E-07	**1.35E-32**
M11	***AlpineN1***	AVG	1.54E-08	2.18E-07	3.57E-06	**4.44E-18**
		STD	5.27E-08	8.06E-07	1.01E-05	**3.11E-17**
		Min	5.28E-154	3.53E-56	9.19E-179	**0.00E+00**
		Max	2.82E-07	4.49E-06	6.48E-05	**2.22E-16**
M12	***EggHolder***	AVG	**4.41E+00**	2.79E+01	5.05E+01	1.92E+02
		STD	**1.17E+01**	3.42E+01	6.59E+01	1.14E+02
		Min	**3.73E-05**	**3.73E-05**	**3.73E-05**	**3.73E-05**
		Max	**6.51E+01**	2.07E+02	2.43E+02	4.66E+02
M13	***Sine envelope***	AVG	**4.71E-06**	**4.71E-06**	**4.71E-06**	**4.71E-06**
		STD	4.58E-15	**6.66E-17**	1.94E-10	1.07E-16
		Min	**4.71E-06**	**4.71E-06**	**4.71E-06**	**4.71E-06**
		Max	**4.71E-06**	**4.71E-06**	**4.71E-06**	**4.71E-06**
C1	***Combination Griewank and Rosenbrock***	AVG	**8.88E-18**	3.95E-04	8.49E-03	1.58E-03
		STD	**3.74E-17**	2.76E-03	1.58E-02	4.98E-03
		Min	4.94E-324	**0.00E+00**	5.55E-16	**0.00E+00**
		Max	**2.22E-16**	1.97E-02	9.86E-02	1.97E-02
C2	***Combination Ackley and Rosenbrock***	AVG	**5.02E-09**	5.52E-05	3.31E+00	1.88E+00
		STD	**1.05E-08**	7.71E-05	3.59E+00	7.27E+00
		Min	3.55E-15	**0.00E+00**	1.77E-06	**0.00E+00**
		Max	**5.95E-08**	3.83E-04	7.20E+00	4.00E+01
Summary of results		AVG	+	2	4	8
			−	15	14	10
			≈	2	1	1
		STD	+	2	1	6
			−	15	14	12
			≈	2	4	1
		Min	+	3	4	9
			−	11	11	5
			≈	5	4	5
		Max	+	0	4	7
			−	16	13	10
			≈	3	2	2

Table A2.2: Test results of HHO-EAS against HHO, GWO and PSO on a benchmark of 19 functions for dimension 30

ID	*Functions*	*Metrics*	*HHO-EAS*	*HHO*	*GWO*	*PSO*
U1	***Sphere***	AVG	**6.61E-247**	8.43E-81	1.05E-40	1.05E+00
		STD	**0.00E+00**	5.61E-80	3.09E-40	5.14E+00
		Min	**2.99E-269**	8.34E-99	7.58E-43	1.71E-04
		Max	**3.09E-245**	4.01E-79	2.17E-39	2.62E+01
U2	***Rotated HyperEllipsoid***	AVG	**2.82E-246**	1.03E-76	9.26E-37	9.29E-01
		STD	**0.00E+00**	7.23E-76	4.70E-36	5.71E+00
		Min	**1.05E-272**	1.20E-93	1.91E-39	4.75E-03
		Max	**1.39E-244**	5.16E-75	3.37E-35	4.09E+01
U3	***Sum Squares***	AVG	**1.46E-243**	4.78E-80	4.82E-39	8.81E+01
		STD	**0.00E+00**	2.25E-79	1.04E-38	1.76E+02
		Min	**7.87E-272**	3.04E-99	4.36E-41	1.82E-03
		Max	**7.23E-242**	1.54E-78	6.19E-38	9.00E+02
U4	***Brown***	AVG	**1.64E-243**	1.12E-78	7.07E-42	1.28E+01
		STD	**0.00E+00**	7.86E-78	1.43E-41	5.95E+00
		Min	**2.20E-273**	5.96E-101	6.64E-44	7.00E+00
		Max	**8.19E-242**	5.61E-77	6.47E-41	3.00E+01
M1	***Dixon-Price***	AVG	**2.41E-01**	2.47E-01	6.67E-01	7.55E+01
		STD	3.20E-03	1.11E-02	**2.22E-04**	1.21E+02
		Min	2.28E-01	**1.84E-01**	6.67E-01	7.15E-01
		Max	**2.46E-01**	2.55E-01	6.68E-01	3.28E+02
M2	***Happy Cat***	AVG	**7.28E-03**	2.06E-02	3.10E-01	2.09E-01
		STD	**1.13E-02**	2.03E-02	5.54E-02	7.47E-02
		Min	**7.95E-05**	1.43E-04	1.99E-01	8.91E-02
		Max	**5.38E-02**	7.79E-02	4.55E-01	4.35E-01
M3	***Ackley***	AVG	**4.44E-16**	**4.44E-16**	3.77E-14	1.46E+00
		STD	**0.00E+00**	**0.00E+00**	3.64E-15	6.61E-01
		Min	**4.44E-16**	**4.44E-16**	2.89E-14	9.93E-03
		Max	**4.44E-16**	**4.44E-16**	4.31E-14	2.75E+00
M4	***Quartic***	AVG	1.43E-04	**1.40E-04**	1.97E-03	1.55E+00
		STD	2.03E-04	**1.58E-04**	8.43E-04	2.64E+00
		Min	**3.10E-07**	1.31E-05	7.01E-04	3.93E-02
		Max	**1.02E-03**	1.09E-03	4.51E-03	1.08E+01
M5	***Rosenbrock***	AVG	**9.11E-04**	4.82E-03	2.69E+01	5.94E+03
		STD	**2.10E-03**	8.27E-03	7.92E-01	1.65E+04
		Min	**3.45E-08**	7.60E-06	2.58E+01	2.62E+01
		Max	**1.41E-02**	3.58E-02	2.88E+01	5.66E+04
M6	***Schwefel***	AVG	**6.37E-01**	2.96E+01	6.68E+03	8.26E+03
		STD	**2.90E+00**	1.48E+02	1.14E+03	8.84E+02
		Min	**3.82E-04**	**3.82E-04**	5.12E+03	6.66E+03
		Max	**2.07E+01**	9.75E+02	9.66E+03	9.75E+03
M7	***Griewank***	AVG	**0.00E+00**	**0.00E+00**	4.07E-03	5.51E-02
		STD	**0.00E+00**	**0.00E+00**	9.20E-03	7.88E-02
		Min	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.37E-04
		Max	**0.00E+00**	**0.00E+00**	4.60E-02	5.17E-01
M8	***Levy and montalo***	AVG	**7.44E-06**	5.26E-05	1.72E+00	1.50E+00
		STD	**1.89E-05**	7.89E-05	1.45E+00	1.62E+00
		Min	**9.79E-10**	4.68E-08	1.25E-01	1.72E-02

		Max	**1.16E-04**	4.15E-04	5.78E+00	7.27E+00
M9	***General penalized 1***	AVG	**1.36E-06**	6.20E-06	6.32E-02	7.81E-02
		STD	**2.24E-06**	8.63E-06	3.80E-02	1.23E-01
		Min	**2.34E-10**	4.72E-09	1.53E-02	2.74E-06
		Max	**1.10E-05**	5.34E-05	2.45E-01	4.19E-01
M10	***General penalized 2***	AVG	**1.65E-05**	7.81E-05	1.02E+00	5.21E-01
		STD	**2.81E-05**	1.04E-04	3.95E-01	6.83E-01
		Min	**3.49E-08**	1.37E-07	1.99E-01	4.92E-03
		Max	**1.44E-04**	5.72E-04	1.91E+00	2.66E+00
M11	***AlpineN1***	AVG	**4.80E-07**	1.02E-06	9.62E-04	1.19E+00
		STD	**3.29E-06**	7.16E-06	1.14E-03	1.73E+00
		Min	**5.68E-139**	1.54E-52	5.48E-22	2.91E-02
		Max	**2.35E-05**	5.12E-05	5.28E-03	6.44E+00
M12	***EggHolder***	AVG	**4.71E+03**	5.35E+03	1.60E+04	2.03E+04
		STD	**6.77E+01**	1.76E+03	1.97E+03	1.36E+03
		Min	**4.59E+03**	4.91E+03	1.29E+04	1.67E+04
		Max	**4.86E+03**	1.76E+04	2.14E+04	2.27E+04
M13	***Sine envelope***	AVG	**2.06E-04**	4.20E-04	2.00E+00	9.88E+00
		STD	**1.28E-04**	4.85E-04	1.45E+00	1.93E+00
		Min	**1.37E-04**	**1.37E-04**	4.61E-01	3.91E+00
		Max	**9.54E-04**	2.06E-03	9.93E+00	1.36E+01
C1	***Combination Griewank and Rosenbrock***	AVG	**2.31E-08**	8.37E-07	9.69E+00	5.55E+04
		STD	**6.57E-08**	2.67E-06	3.23E+00	2.00E+05
		Min	**1.77E-14**	3.57E-12	4.32E+00	5.45E+00
		Max	**3.64E-07**	1.37E-05	1.73E+01	1.11E+06
C2	***Combination Ackley and Rosenbrock***	AVG	**2.58E-03**	1.25E-02	1.14E+02	4.67E+02
		STD	**3.93E-03**	1.73E-02	2.44E+01	2.54E+01
		Min	**4.44E-07**	1.98E-04	1.08E+02	4.09E+02
		Max	**1.59E-02**	9.54E-02	2.71E+02	5.16E+02
Summary of results		AVG	+	1	0	0
			−	16	19	19
			≈	2	0	0
		STD	+	1	1	0
			−	16	18	19
			≈	2	0	0
		Min	+	1	0	0
			−	15	18	19
			≈	3	1	0
		Max	+	0	0	0
			−	17	19	19
			≈	2	0	0

Table A2.3: Test results of HHO-EAS against HHO, GWO and PSO on a benchmark of 19 functions for dimension 100

ID	*Functions*	*Metrics*	*HHO-EAS*	*HHO*	*GWO*	*PSO*
U1	***Sphere***	AVG	**6.47E-245**	1.01E-77	2.91E-20	8.82E+01
		STD	**0.00E+00**	6.44E-77	3.42E-20	2.63E+01
		Min	**1.38E-271**	3.91E-101	2.85E-21	4.82E+01
		Max	**2.45E-243**	4.59E-76	1.94E-19	1.54E+02
U2	***Rotated HyperEllipsoid***	AVG	**7.38E-231**	2.17E-76	1.47E-16	6.69E+03
		STD	**0.00E+00**	1.51E-75	1.26E-16	2.43E+03
		Min	**1.17E-265**	3.30E-94	1.59E-17	2.76E+03
		Max	**3.69E-229**	1.08E-74	6.66E-16	1.35E+04
U3	***Sum Squares***	AVG	**1.90E-235**	9.23E-75	4.01E-18	7.67E+03
		STD	**0.00E+00**	6.46E-74	4.02E-18	2.62E+03
		Min	**2.13E-274**	6.65E-98	4.60E-19	2.57E+03
		Max	**8.70E-234**	4.62E-73	1.57E-17	1.53E+04
U4	***Brown***	AVG	**8.51E-244**	5.08E-79	4.44E-21	1.08E+02
		STD	**0.00E+00**	2.09E-78	5.86E-21	2.66E+01
		Min	**5.90E-269**	5.27E-99	3.37E-22	5.58E+01
		Max	**4.23E-242**	1.06E-77	3.79E-20	1.54E+02
M1	***Dixon-Price***	AVG	**2.50E-01**	2.52E-01	6.67E-01	4.29E+05
		STD	1.03E-03	3.11E-03	**3.60E-04**	3.71E+05
		Min	**2.50E-01**	**2.50E-01**	6.67E-01	3.02E+04
		Max	**2.54E-01**	2.67E-01	6.69E-01	1.46E+06
M2	***Happy Cat***	AVG	**2.26E-02**	1.14E-01	6.93E-01	7.31E-01
		STD	**3.89E-02**	1.13E-01	8.56E-02	7.06E-02
		Min	**2.43E-05**	8.64E-04	5.16E-01	6.16E-01
		Max	**2.50E-01**	4.24E-01	8.93E-01	8.82E-01
M3	***Ackley***	AVG	**4.44E-16**	**4.44E-16**	3.05E-10	6.06E+00
		STD	**0.00E+00**	**0.00E+00**	1.79E-10	6.03E-01
		Min	**4.44E-16**	**4.44E-16**	9.74E-11	4.57E+00
		Max	**4.44E-16**	**4.44E-16**	1.20E-09	7.42E+00
M4	***Quartic***	AVG	**1.51E-04**	1.75E-04	7.37E-03	1.90E+02
		STD	**1.37E-04**	1.74E-04	2.87E-03	9.65E+01
		Min	8.89E-06	**1.17E-06**	2.91E-03	5.09E+01
		Max	**7.52E-04**	8.27E-04	1.63E-02	4.37E+02
M5	***Rosenbrock***	AVG	**1.62E-03**	1.64E-02	9.75E+01	4.61E+05
		STD	**2.54E-03**	2.86E-02	7.28E-01	1.50E+05
		Min	**5.04E-09**	5.53E-05	9.59E+01	1.56E+05
		Max	**1.18E-02**	1.87E-01	9.85E+01	8.54E+05
M6	***Schwefel***	AVG	**4.68E-01**	2.16E+00	2.57E+04	3.35E+04
		STD	**7.85E-01**	6.02E+00	3.59E+03	2.67E+03
		Min	**1.27E-03**	**1.27E-03**	2.18E+04	2.56E+04
		Max	**3.31E+00**	3.94E+01	3.60E+04	3.71E+04
M7	***Griewank***	AVG	**0.00E+00**	**0.00E+00**	8.37E-16	4.63E+00
		STD	**0.00E+00**	**0.00E+00**	4.28E-16	1.70E+00
		Min	**0.00E+00**	**0.00E+00**	2.22E-16	2.22E+00
		Max	**0.00E+00**	**0.00E+00**	2.33E-15	1.12E+01
M8	***Levy and montalo***	AVG	**1.83E-05**	1.40E-04	2.25E+01	6.22E+01
		STD	**4.14E-05**	1.73E-04	7.88E+00	1.88E+01
		Min	**1.50E-09**	6.73E-07	1.39E+01	3.32E+01

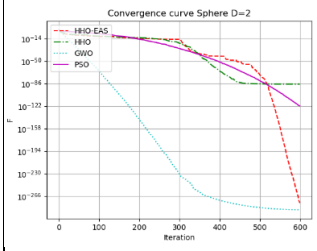
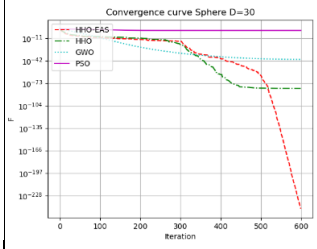
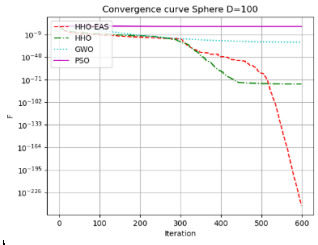
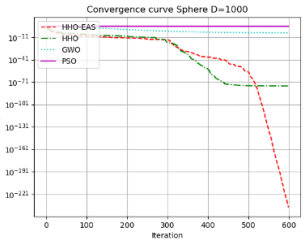
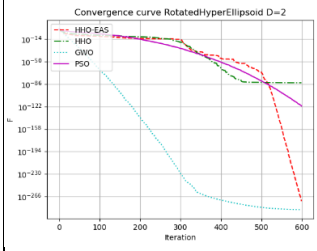
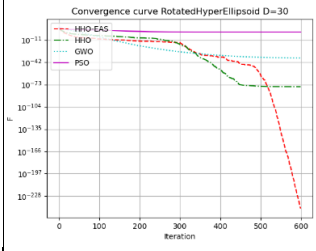
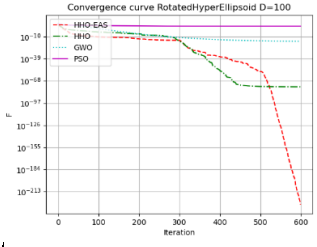
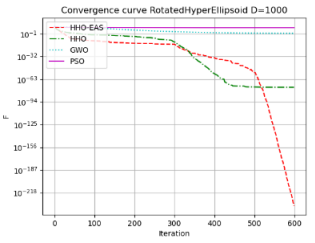
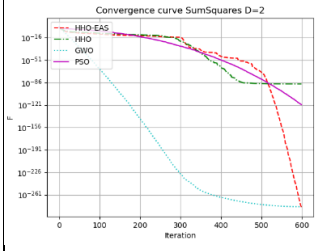
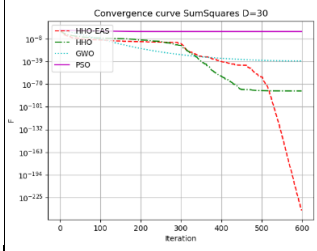
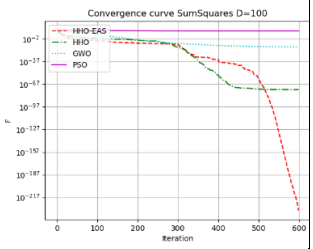
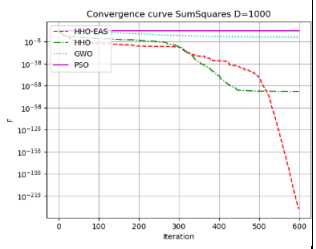
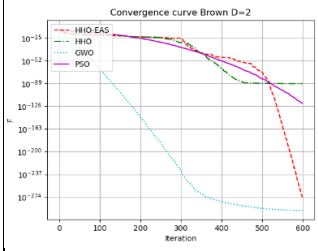
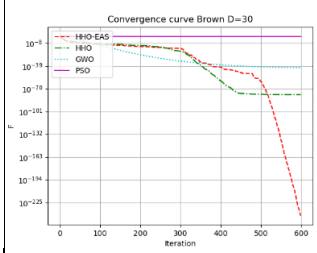
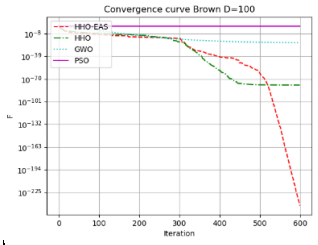
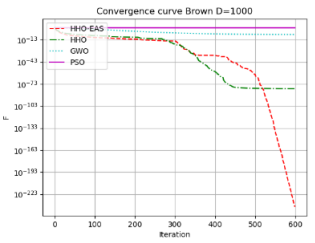
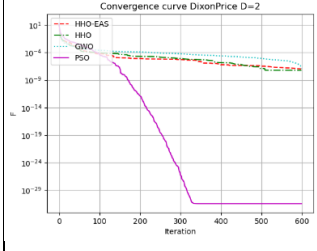
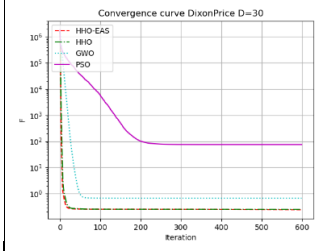
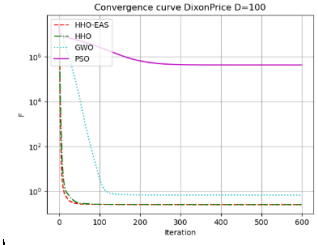
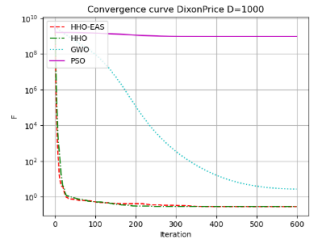
		Max	**2.73E-04**	8.33E-04	5.73E+01	1.09E+02
M9	***General penalized 1***	AVG	**5.97E-07**	3.27E-06	1.59E-01	3.06E+00
		STD	**9.64E-07**	4.93E-06	6.65E-02	2.34E+00
		Min	**1.06E-10**	2.52E-09	1.07E-01	1.25E+00
		Max	**3.99E-06**	2.97E-05	3.65E-01	1.68E+01
M10	***General penalized 2***	AVG	**1.70E-05**	1.22E-04	6.30E+00	1.89E+02
		STD	**2.54E-05**	1.47E-04	1.39E+00	5.30E+01
		Min	**5.73E-10**	8.15E-07	3.55E+00	7.25E+01
		Max	**1.13E-04**	7.86E-04	8.67E+00	3.15E+02
M11	***AlpineN1***	AVG	**2.94E-07**	9.28E-06	3.83E-03	4.81E+01
		STD	**1.72E-06**	6.50E-05	3.13E-03	9.30E+00
		Min	**2.05E-136**	2.15E-52	3.55E-11	2.59E+01
		Max	**1.20E-05**	4.64E-04	1.44E-02	7.48E+01
M12	***EggHolder***	AVG	**1.74E+04**	1.75E+04	6.61E+04	7.83E+04
		STD	3.22E+01	**2.73E+01**	6.91E+03	2.50E+03
		Min	**1.73E+04**	1.74E+04	5.75E+04	7.31E+04
		Max	**1.74E+04**	1.75E+04	8.04E+04	8.27E+04
M13	***Sine envelope***	AVG	**7.89E-04**	1.83E-03	1.99E+01	4.14E+01
		STD	**5.13E-04**	2.11E-03	5.57E+00	3.91E+00
		Min	**4.67E-04**	**4.67E-04**	1.14E+01	3.23E+01
		Max	**2.73E-03**	1.02E-02	4.86E+01	5.10E+01
C1	***Combination Griewank and Rosenbrock***	AVG	**4.97E-08**	1.08E-05	4.40E+01	5.73E+06
		STD	**1.02E-07**	4.55E-05	5.66E+00	2.61E+06
		Min	1.03E-14	**9.99E-15**	3.70E+01	1.48E+06
		Max	**5.06E-07**	3.22E-04	6.84E+01	1.14E+07
C2	***Combination Ackley and Rosenbrock***	AVG	**1.82E-02**	1.39E-01	3.95E+02	1.84E+03
		STD	**5.81E-02**	2.56E-01	2.34E+02	5.86E+01
		Min	**9.78E-06**	5.45E-04	3.61E+02	1.72E+03
		Max	**4.10E-01**	1.20E+00	2.03E+03	2.00E+03
Summary of results		AVG	+	0	0	0
			−	17	19	19
			≈	2	0	0
		STD	+	1	1	0
			−	16	18	19
			≈	2	0	0
		Min	+	2	0	0
			−	13	19	19
			≈	4	0	0
		Max	+	0	0	0
			−	17	19	19
			≈	2	0	0

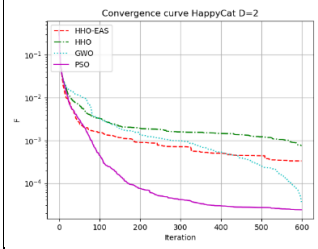
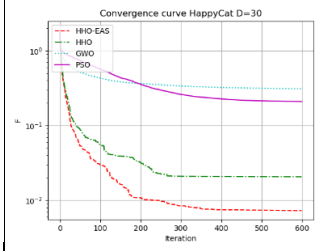
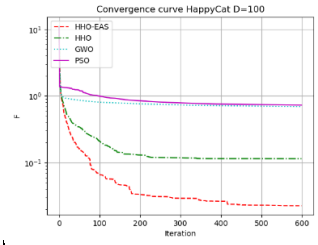
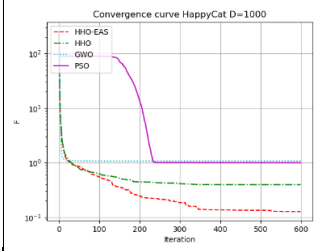
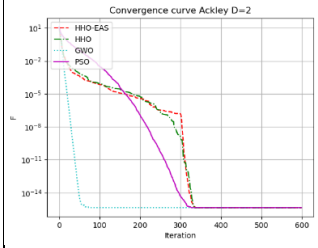
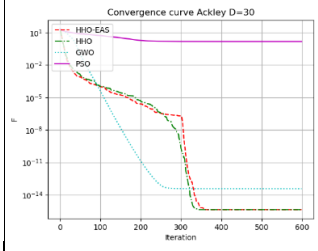
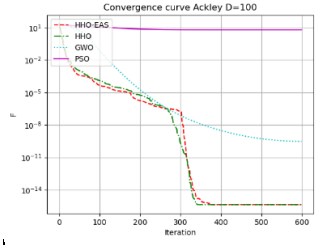
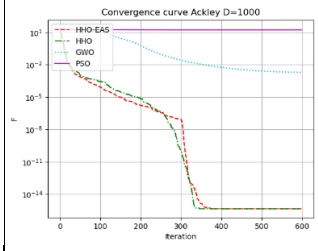
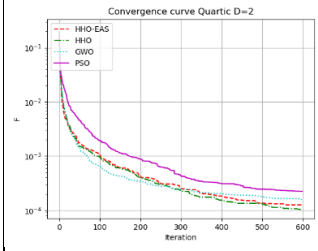
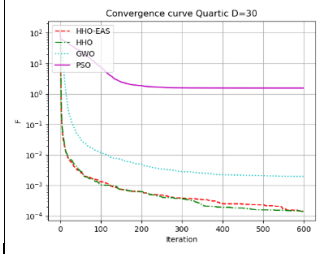
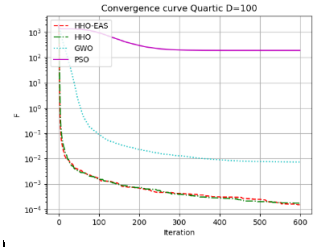
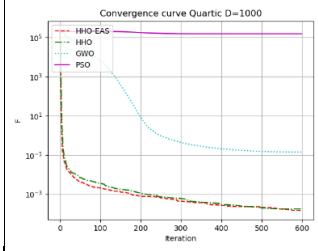
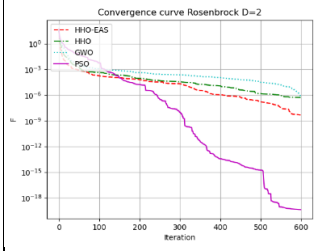
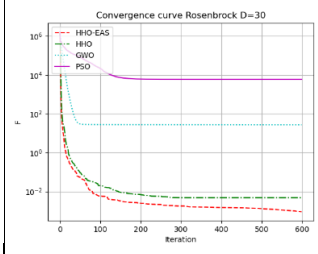
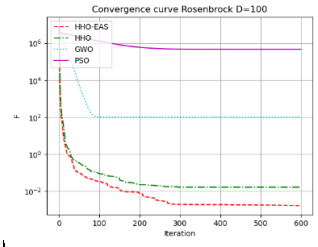
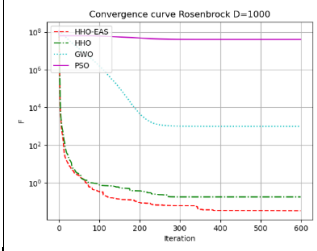
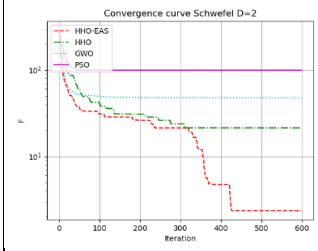
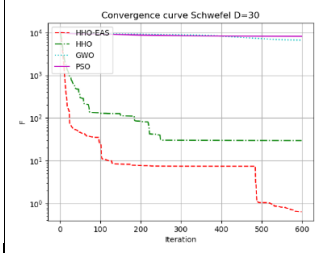
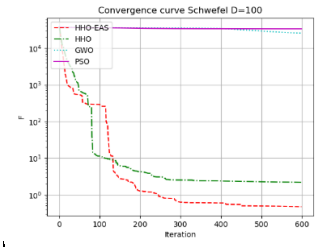
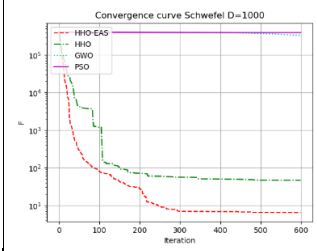
Table A2. 4: Test results of HHO-EAS against HHO, GWO and PSO on a benchmark of 19 functions for dimension 1000

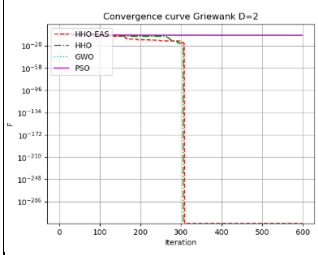
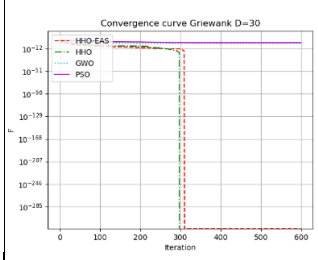
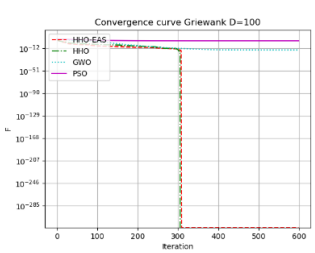
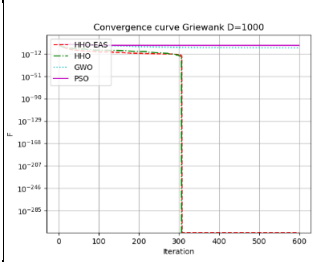
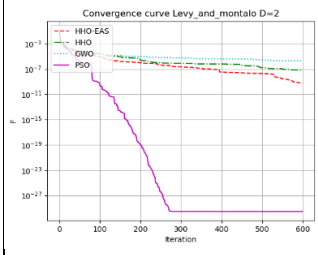
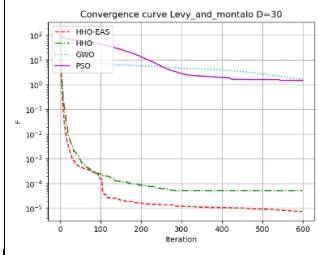
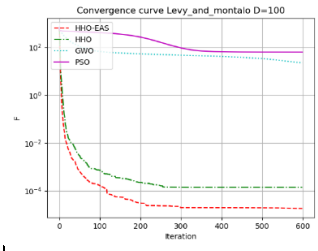
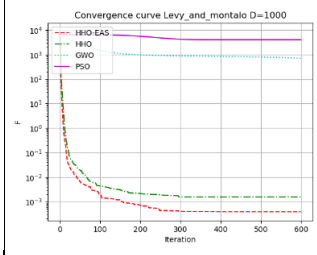
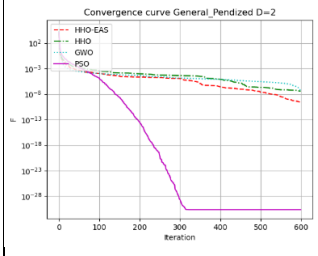
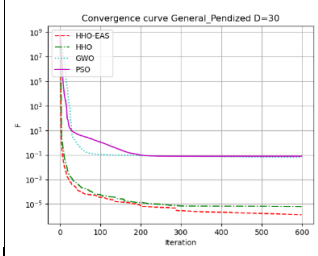
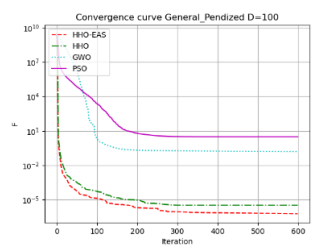
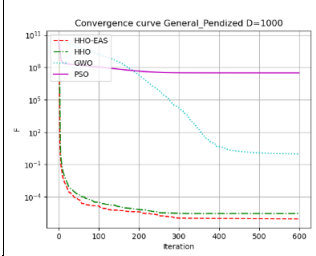
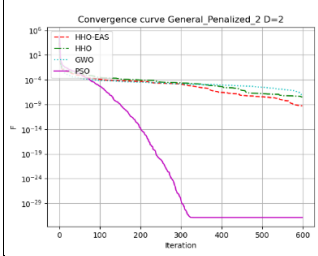
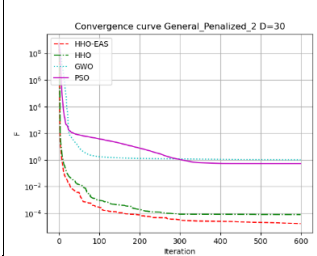
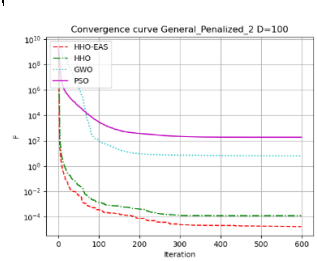
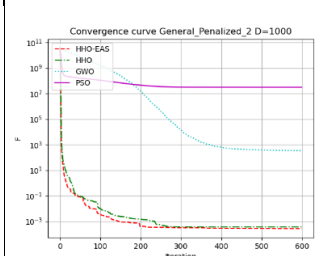
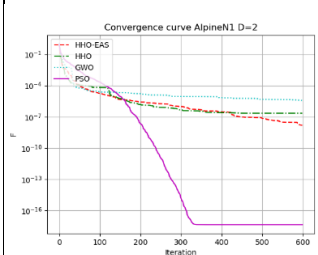
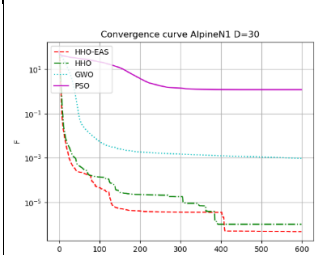
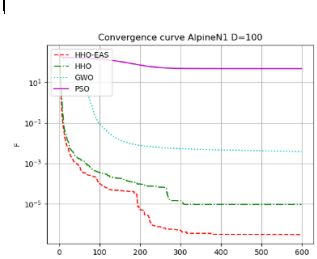
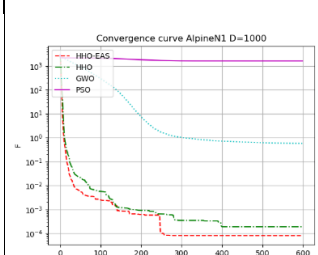
ID	*Functions*	*Metrics*	*HHO-EAS*	*HHO*	*GWO*	*PSO*
U1	***Sphere***	AVG	**1.46E-239**	6.64E-77	8.94E-06	4.94E+03
		STD	**0.00E+00**	4.59E-76	2.68E-06	1.33E+02
		Min	**1.86E-262**	2.19E-96	4.11E-06	4.56E+03
		Max	**6.89E-238**	3.28E-75	1.57E-05	5.14E+03
U2	***Rotated HyperEllipsoid***	AVG	**7.71E-237**	9.43E-75	5.99E-01	2.89E+07
		STD	**0.00E+00**	6.53E-74	1.55E-01	1.27E+06
		Min	**9.77E-261**	6.95E-94	3.32E-01	2.56E+07
		Max	**3.86E-235**	4.67E-73	1.07E+00	3.13E+07
U3	***Sum Squares***	AVG	**1.53E-236**	1.86E-76	1.44E-02	6.49E+06
		STD	**0.00E+00**	6.65E-76	4.76E-03	2.13E+05
		Min	**9.16E-264**	5.10E-97	6.10E-03	6.06E+06
		Max	**7.63E-235**	3.37E-75	2.61E-02	6.95E+06
U4	***Brown***	AVG	**4.58E-241**	3.93E-80	2.07E-06	2.98E+03
		STD	**0.00E+00**	1.47E-79	5.93E-07	1.23E+02
		Min	**8.04E-264**	5.15E-97	1.08E-06	2.60E+03
		Max	**1.10E-239**	8.17E-79	3.54E-06	3.25E+03
M1	***Dixon-Price***	AVG	**2.87E-01**	2.89E-01	2.75E+00	9.41E+08
		STD	**1.13E-01**	1.47E-01	9.82E-01	7.70E+07
		Min	**2.50E-01**	**2.50E-01**	1.18E+00	7.54E+08
		Max	**6.67E-01**	1.00E+00	6.96E+00	1.10E+09
M2	***Happy Cat***	AVG	**1.26E-01**	3.94E-01	1.07E+00	9.96E-01
		STD	2.65E-01	4.87E-01	7.69E-02	**3.16E-02**
		Min	**9.85E-05**	2.01E-03	9.50E-01	9.19E-01
		Max	1.68E+00	1.96E+00	1.37E+00	**1.07E+00**
M3	***Ackley***	AVG	**4.44E-16**	**4.44E-16**	2.00E-03	1.69E+01
		STD	**0.00E+00**	**0.00E+00**	2.57E-04	1.83E-01
		Min	**4.44E-16**	**4.44E-16**	1.50E-03	1.65E+01
		Max	**4.44E-16**	**4.44E-16**	2.58E-03	1.73E+01
M4	***Quartic***	AVG	**1.42E-04**	1.71E-04	1.39E-01	1.55E+05
		STD	**1.24E-04**	2.30E-04	3.32E-02	7.17E+03
		Min	**2.80E-06**	5.41E-06	6.55E-02	1.38E+05
		Max	**5.71E-04**	1.57E-03	2.20E-01	1.71E+05
M5	***Rosenbrock***	AVG	**3.33E-02**	1.80E-01	9.97E+02	4.01E+07
		STD	**6.88E-02**	2.51E-01	2.45E-01	2.45E+06
		Min	**1.09E-06**	1.34E-05	9.96E+02	3.63E+07
		Max	**3.42E-01**	1.35E+00	9.97E+02	4.59E+07
M6	***Schwefel***	AVG	**6.38E+00**	4.63E+01	3.33E+05	3.91E+05
		STD	**1.32E+01**	1.56E+02	2.92E+04	8.58E+03
		Min	**1.27E-02**	1.41E-02	3.10E+05	3.74E+05
		Max	**7.96E+01**	1.09E+03	4.01E+05	4.04E+05
M7	***Griewank***	AVG	**0.00E+00**	**0.00E+00**	1.34E-02	3.55E+02
		STD	**0.00E+00**	**0.00E+00**	3.72E-02	2.44E+01
		Min	**0.00E+00**	**0.00E+00**	1.05E-04	3.14E+02
		Max	**0.00E+00**	**0.00E+00**	1.45E-01	4.05E+02
M8	***Levy and montalo***	AVG	**3.85E-04**	1.54E-03	7.10E+02	4.07E+03
		STD	**8.42E-04**	2.24E-03	2.24E+01	2.01E+02
		Min	**7.72E-10**	3.42E-06	6.61E+02	3.69E+03

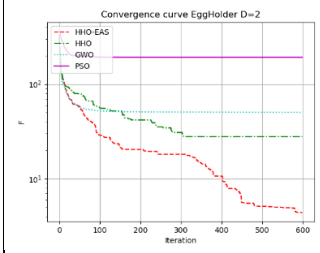
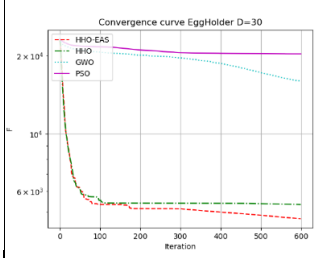
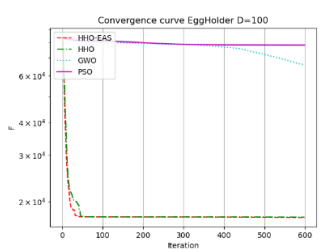
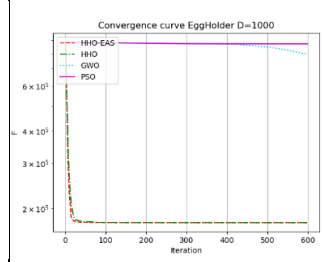
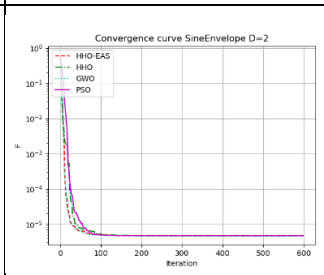
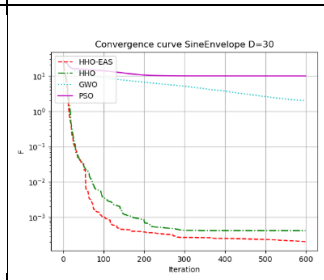
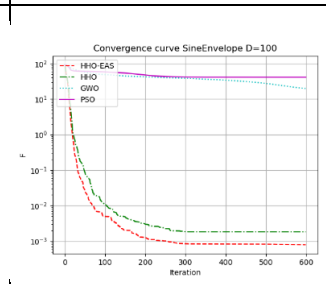
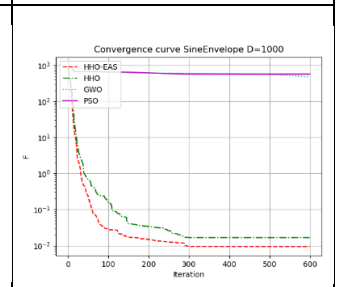
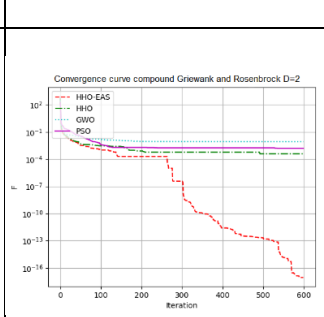
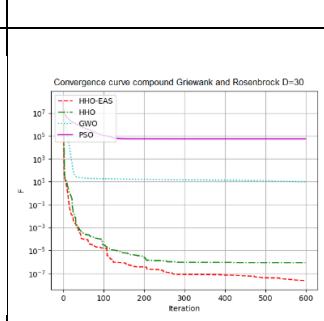
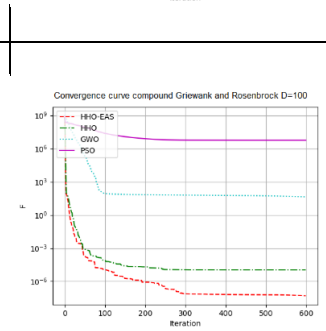
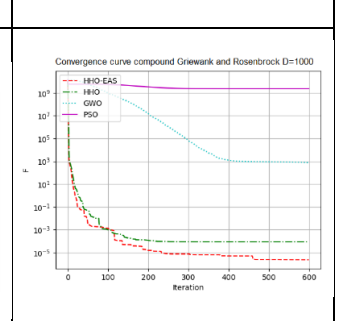
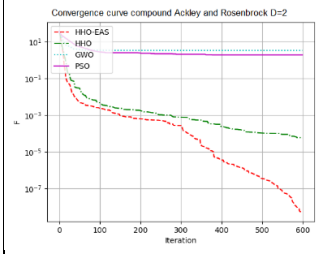
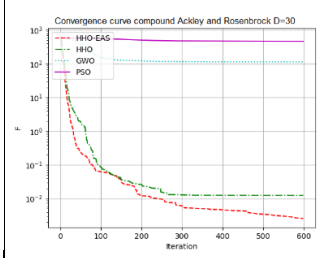
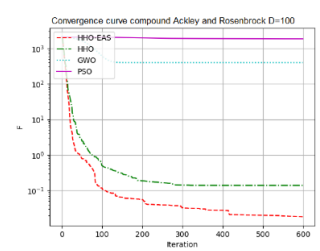
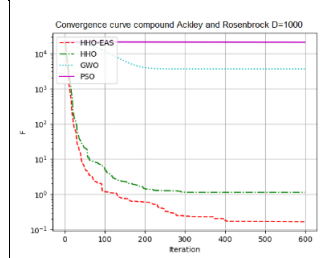
		Max	**5.36E-03**	1.07E-02	7.81E+02	4.59E+03
M9	***General penalized 1***	AVG	**9.18E-07**	2.86E-06	9.70E-01	3.11E+07
		STD	**1.73E-06**	4.12E-06	5.23E-01	6.60E+06
		Min	1.79E-10	**5.15E-11**	2.39E-01	1.89E+07
		Max	**1.02E-05**	1.96E-05	2.24E+00	4.29E+07
M10	***General penalized 2***	AVG	**2.61E-04**	3.55E-04	3.38E+02	3.10E+07
		STD	**3.74E-04**	4.75E-04	2.48E+02	6.91E+06
		Min	**4.19E-08**	3.61E-07	8.15E+01	1.63E+07
		Max	**1.56E-03**	2.42E-03	8.33E+02	4.66E+07
M11	***AlpineN1***	AVG	**8.14E-05**	1.90E-04	5.77E-01	1.62E+03
		STD	**5.19E-04**	9.36E-04	1.24E+00	4.58E+01
		Min	**1.07E-134**	5.94E-48	1.14E-01	1.49E+03
		Max	**3.70E-03**	6.50E-03	7.51E+00	1.68E+03
M12	***EggHolder***	AVG	**1.76E+05**	**1.76E+05**	7.97E+05	8.72E+05
		STD	**6.93E+00**	4.00E+01	4.09E+04	7.73E+03
		Min	**1.76E+05**	**1.76E+05**	7.58E+05	8.50E+05
		Max	**1.76E+05**	**1.76E+05**	8.80E+05	8.85E+05
M13	***Sine envelope***	AVG	**9.42E-03**	1.67E-02	4.72E+02	5.64E+02
		STD	**1.03E-02**	1.48E-02	2.88E+01	1.36E+01
		Min	**4.71E-03**	4.76E-03	4.23E+02	5.37E+02
		Max	**5.62E-02**	6.40E-02	6.31E+02	5.92E+02
C1	***Combination Griewank and Rosenbrock***	AVG	**2.36E-06**	8.76E-05	7.84E+02	2.36E+09
		STD	**6.22E-06**	2.84E-04	7.25E+01	2.73E+08
		Min	**4.94E-324**	4.39E-09	6.86E+02	1.86E+09
		Max	**2.75E-05**	1.93E-03	1.04E+03	2.96E+09
C2	***Combination Ackley and Rosenbrock***	AVG	**1.66E-01**	1.13E+00	3.62E+03	2.06E+04
		STD	**2.91E-01**	1.98E+00	9.89E+00	1.33E+02
		Min	**1.92E-07**	1.01E-04	3.62E+03	2.03E+04
		Max	**1.40E+00**	1.14E+01	3.66E+03	2.09E+04
Summary of results		AVG	+	0	0	0
			−	16	19	19
			≈	3	0	0
		STD	+	0	0	1
			−	17	19	18
			≈	2	0	0
		Min	+	1	0	0
			−	14	19	19
			≈	4	0	0
		Max	+	0	0	1
			−	16	19	18
			≈	3	0	0

Appendix 3: Graphical representation of the general test results

	D=2	*D=30*	*D=100*	*D=1000*
U1				
U2				
U3				
U4				
M1				

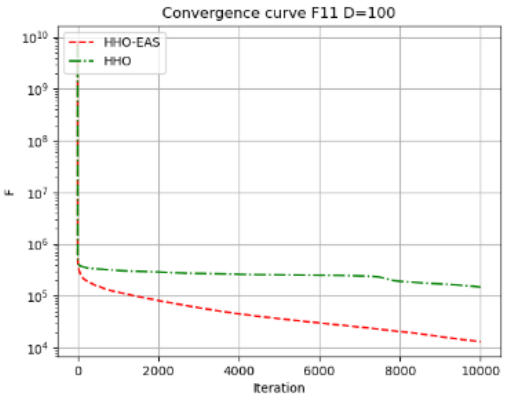
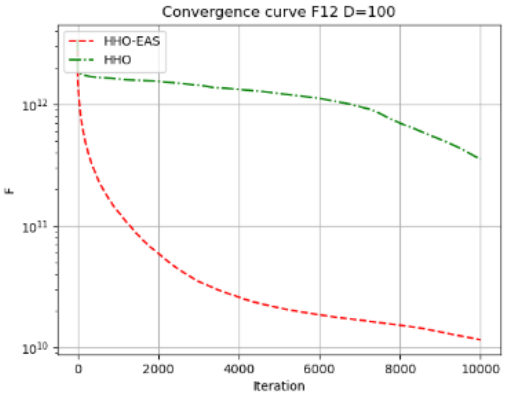
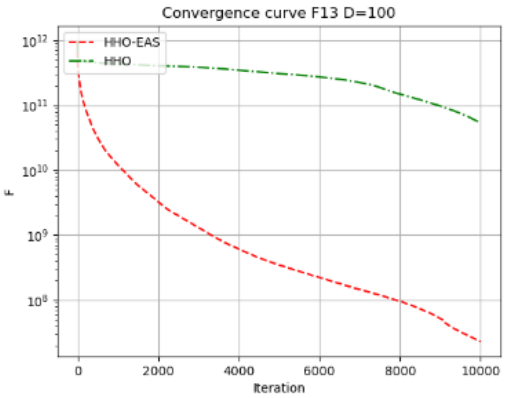
M2	 Convergence curve HappyCat D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve HappyCat D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve HappyCat D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve HappyCat D=1000 HHO-EAS HHO GWO PSO F Iteration
M3	 Convergence curve Ackley D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Ackley D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Ackley D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Ackley D=1000 HHO-EAS HHO GWO PSO F Iteration
M4	 Convergence curve Quartic D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Quartic D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Quartic D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Quartic D=1000 HHO-EAS HHO GWO PSO F Iteration
M5	 Convergence curve Rosenbrock D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Rosenbrock D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Rosenbrock D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Rosenbrock D=1000 HHO-EAS HHO GWO PSO F Iteration
M6	 Convergence curve Schwefel D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Schwefel D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Schwefel D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve Schwefel D=1000 HHO-EAS HHO GWO PSO F Iteration

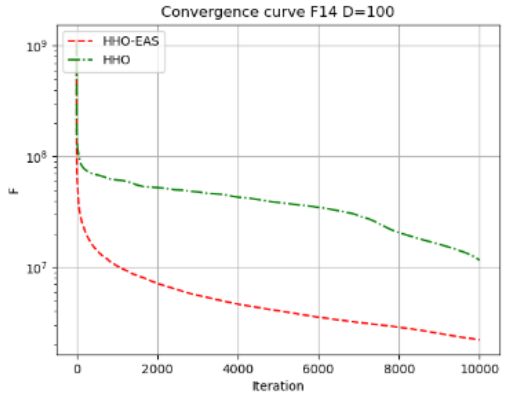
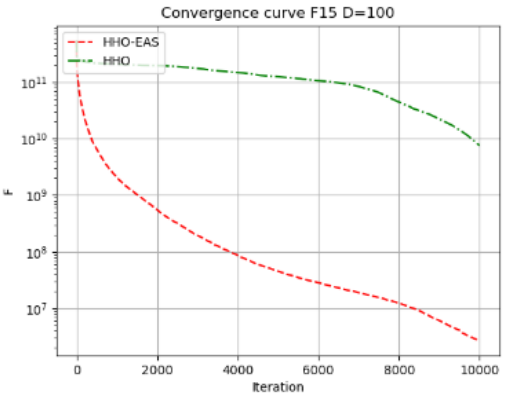
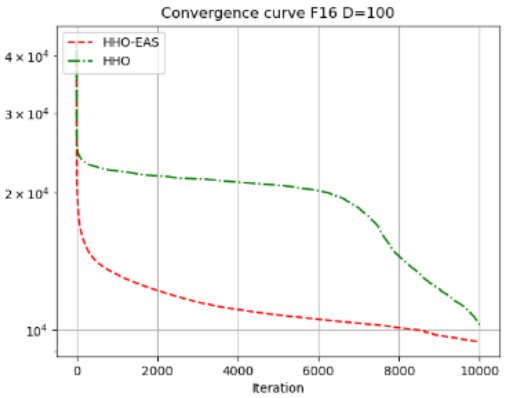
M7				
M8				
M9				
M10				
M11				

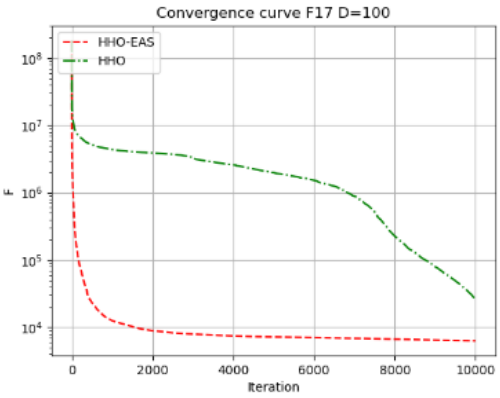
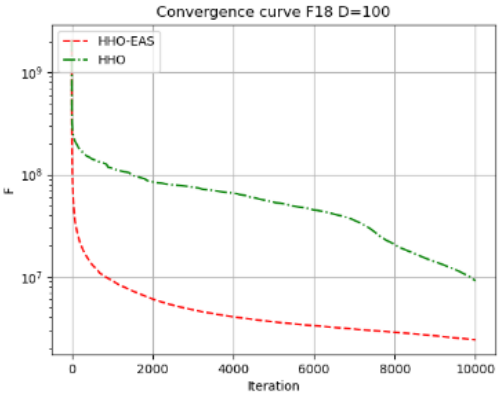
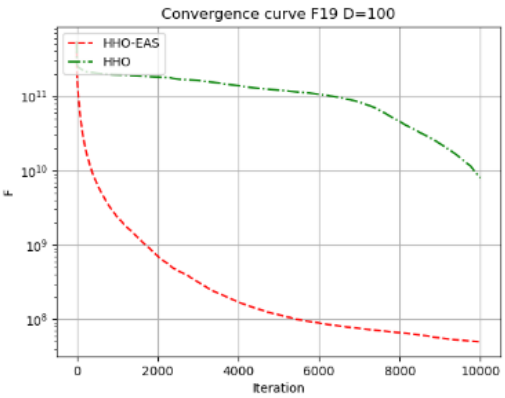
M12	 Convergence curve EggHolder D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve EggHolder D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve EggHolder D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve EggHolder D=1000 HHO-EAS HHO GWO PSO F Iteration
M13	 Convergence curve SineEnvelope D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve SineEnvelope D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve SineEnvelope D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve SineEnvelope D=1000 HHO-EAS HHO GWO PSO F Iteration
C1	 Convergence curve compound Griewank and Rosenbrock D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Griewank and Rosenbrock D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Griewank and Rosenbrock D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Griewank and Rosenbrock D=1000 HHO-EAS HHO GWO PSO F Iteration
C2	 Convergence curve compound Ackley and Rosenbrock D=2 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Ackley and Rosenbrock D=30 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Ackley and Rosenbrock D=100 HHO-EAS HHO GWO PSO F Iteration	 Convergence curve compound Ackley and Rosenbrock D=1000 HHO-EAS HHO GWO PSO F Iteration

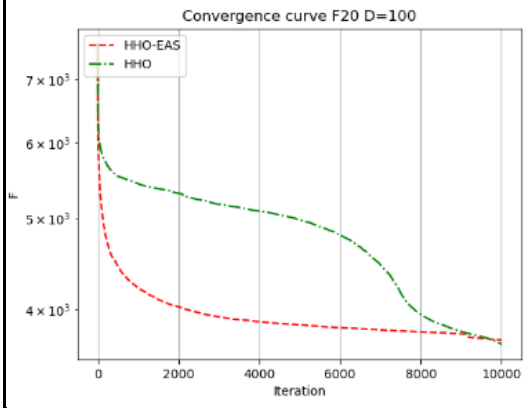
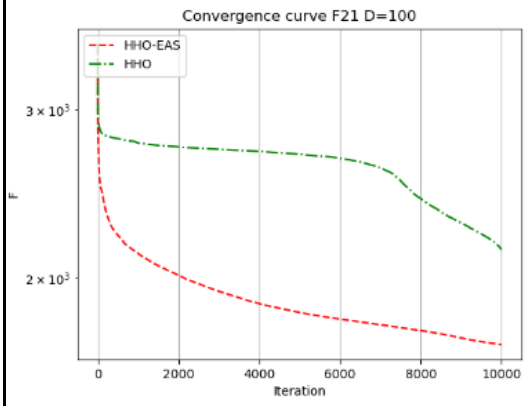
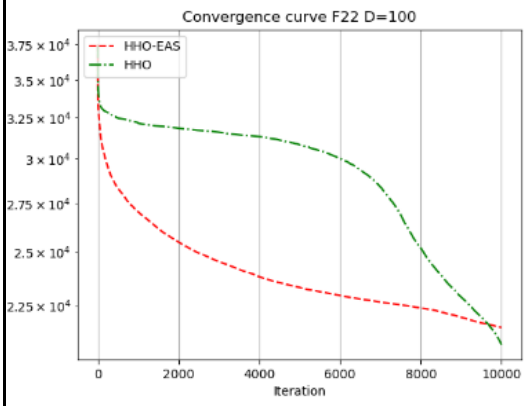
Appendix 4: Optimisation results of HHO-EAS against HHO on the specific benchmark of 20 functions from CEC 2017 for dimension 100

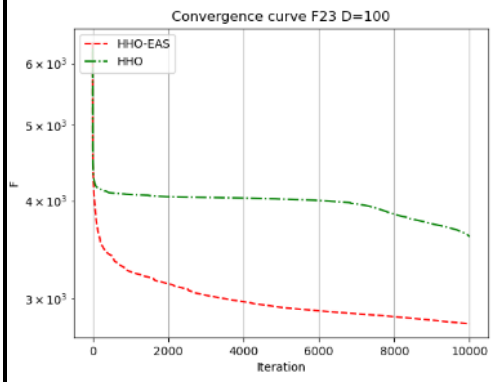
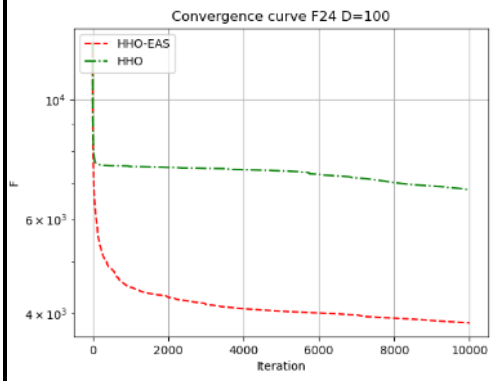
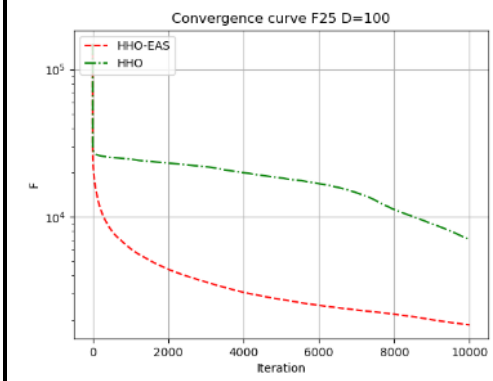
Table A4.1: Test results of HHO-EAS against HHO on a benchmark of 20 functions from CEC 2017 for dimension 100 [Wu G et al., 2017]

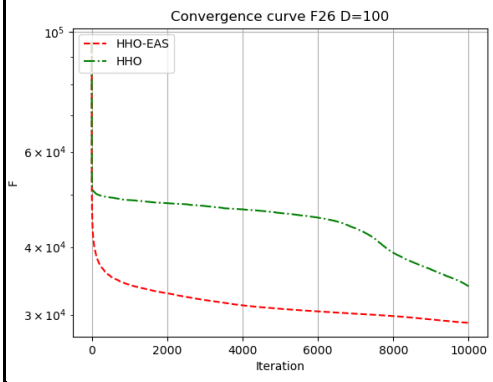
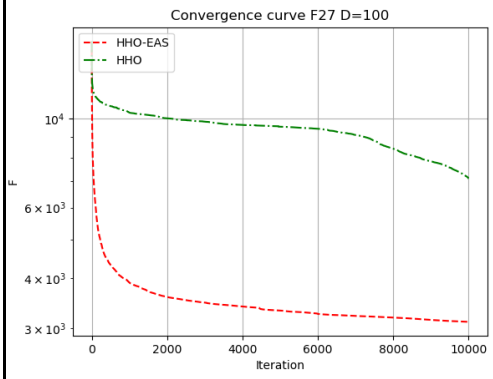
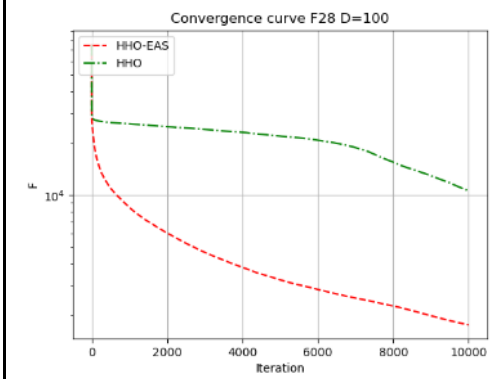
Function	*Description*	*Convergence curve*	*Metric*	*HHO-EAS*	*HHO*
F11	***Hybrid Function 1*** 1. Zakharov 2. Rosenbrock 3. Rastrigins Multimodal Non-separable subcomponents		***AVG***	**1.31E+04**	1.48E+05
			STD	**7.22E+03**	4.18E+04
			Min	**6.88E+03**	7.10E+04
			Max	**5.69E+04**	2.91E+05
F12	***Hybrid Function 2*** 1. High Conditione Elliptic 2. Modified Schwefels 3. Bent Cigar Multimodal Non-separable subcomponents		***AVG***	**1.16E+10**	3.55E+11
			STD	**4.48E+09**	9.11E+10
			Min	**4.36E+09**	1.37E+11
			Max	**2.65E+10**	5.14E+11
F13	***Hybrid Function 3*** 1. Bent Cigar 2. Rosenbrock 3. Lunache Bi-Rastrigin Multimodal Non-separable subcomponents		***AVG***	**2.28E+07**	5.43E+10
			STD	**1.66E+07**	1.95E+10
			Min	**2.73E+06**	1.71E+10
			Max	**7.29E+07**	1.02E+11

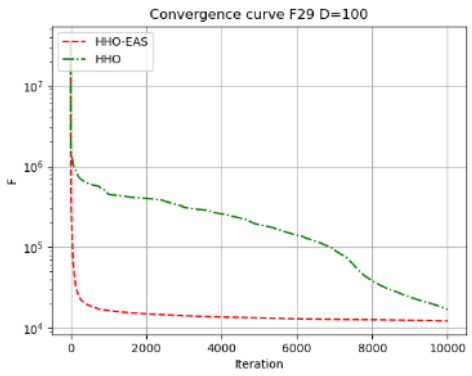
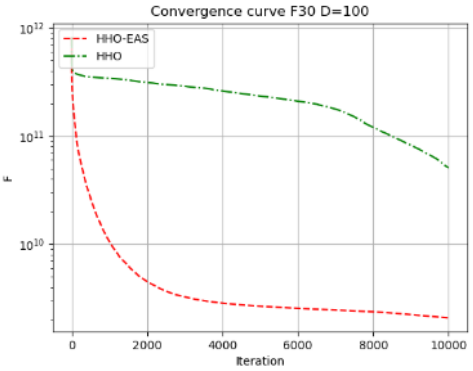
F14	***Hybrid Function 4*** 1. High Conditioned Elliptic 2. Ackley 3. Schaffer 4. Rastrigin Multimodal Non-separable subcomponents		***AVG***	**2.21E+06**	1.16E+07
			STD	**9.98E+05**	3.56E+06
			Min	**8.21E+05**	5.24E+06
			Max	**4.83E+06**	1.93E+07
F15	***Hybrid Function 5*** 1. Bent Cigar 2. HGBat 3. Rastrigin 4. Rosenbrock Multimodal Non-separable subcomponents		***AVG***	**2.67E+06**	7.60E+09
			STD	**2.89E+06**	5.97E+09
			Min	**2.31E+05**	2.78E+08
			Max	**1.49E+07**	2.66E+10
F16	***Hybrid Function 6*** 1. Expanded Schaffer 2. HGBat 3. Rosenbrock 5. Modified Schwefel Multimodal Non-separable subcomponents		***AVG***	**9.43E+03**	1.03E+04
			STD	2.11E+03	**1.63E+03**
			Min	**6.00E+03**	6.63E+03
			Max	**1.36E+04**	1.43E+04

F17	***Hybrid Function 7*** 1. Katsura 2. Ackley 3. Expanded Griewank + Rosenbrock 4. Modified Schwefel 5. Rastrigin Multimodal Non-separable subcomponents		***AVG***	**6.29E+03**	2.65E+04
			STD	**1.01E+03**	3.13E+04
			Min	**4.04E+03**	5.57E+03
			Max	**8.72E+03**	1.85E+05
F18	***Hybrid Function 8*** 1. High Conditioned Elliptic 2. Ackleys 3. Rastrigins 4. HGBat Function Multimodal Non-separable subcomponents		***AVG***	**2.42E+06**	9.13E+06
			STD	**1.02E+06**	6.91E+06
			Min	**3.77E+05**	9.78E+05
			Max	**5.33E+06**	3.72E+07
F19	***Hybrid Function 9*** 1. Bent Cigar 2. Rastrigin's 3. Expanded Griewanks + Rosenbrocks 4. Weierstrass 5. Expanded Schaffers Multimodal Non-separable subcomponents		***AVG***	**5.00E+07**	8.02E+09
			STD	**5.00E+07**	6.42E+09
			Min	**5.99E+06**	5.06E+07
			Max	**2.11E+08**	2.54E+10

F20	***Hybrid Function 10*** 1. Happycat 2. Katsura 3. Ackley 4. Rastrigin 5. Modified Schwefel 6. Schaffer Multimodal Non-separable subcomponents		***AVG***	3.72E+03	**3.67E+03**
			STD	5.00E+02	**4.64E+02**
			Min	2.47E+03	**1.95E+03**
			Max	**4.85E+03**	**4.85E+03**
F21	***Composition Function 1*** 1. Rosenbrock 2. High Conditioned Elliptic 3. Rastrigin Multimodal Non-separable Asymmetrical		***AVG***	**1.70E+03**	2.14E+03
			STD	2.01E+02	**1.92E+02**
			Min	**1.36E+03**	1.71E+03
			Max	**2.26E+03**	2.53E+03
F22	***Composition Function 2*** 1. Rastrigin 2. Griewank 3. Modifed Schwefel Multimodal Non-separable Asymmetrical		***AVG***	2.16E+04	**2.08E+04**
			STD	**2.02E+03**	2.27E+03
			Min	1.77E+04	**1.73E+04**
			Max	**2.61E+04**	**2.61E+04**

F23	***Composition Function 3*** 1. Rosenbrock 2. Ackley 3. Modifed Schwefel 4. Rastrigin Multimodal Non-separable Asymmetrical		***AVG***	**2.78E+03**	3.60E+03
			STD	**3.35E+02**	4.10E+02
			Min	**2.05E+03**	3.00E+03
			Max	**3.52E+03**	5.03E+03
F24	***Composition Function 4*** 1. Ackley 2. High Conditioned Elliptic 3. Girewank 4. Rastrigin Multimodal Non-separable Asymmetrical		***AVG***	**3.84E+03**	6.78E+03
			STD	**4.12E+02**	8.17E+02
			Min	**2.88E+03**	5.13E+03
			Max	**4.80E+03**	8.74E+03
F25	***Composition Function 5*** 1. Rastrigin 2. Happycat 3. Ackley 4. Discus 5. Rosenbrock Multimodal Non-separable Asymmetrical		***AVG***	**1.85E+03**	7.05E+03
			STD	**2.39E+02**	7.42E+02
			Min	**1.47E+03**	5.45E+03
			Max	**2.50E+03**	8.73E+03

F26	***Composition Function 6*** 1. Expanded Scaffer 2. Modified Schwefel 3. Griewank 4. Rosenbrock 5. Rastrigin Multimodal Non-separable Asymmetrical		***AVG***	**2.90E+04**	3.39E+04
			STD	2.07E+03	**1.96E+03**
			Min	**2.43E+04**	2.90E+04
			Max	**3.34E+04**	3.79E+04
F27	***Composition Function 7*** 1. HGBat 2. Rastrigin 3. Modifed Schwefel 4. Bent-Cigar 5. High Conditioned Elliptic 6. Expanded Scaffer Multimodal Non-separable Asymmetrical		***AVG***	**3.11E+03**	7.10E+03
			STD	**6.40E+02**	1.74E+03
			Min	**1.77E+03**	3.81E+03
			Max	**4.45E+03**	1.09E+04
F28	***Composition Function 8*** 1. Ackley's Function 2. Griewank's Function 3. Discus Function 4. Rosenbrock's Function 5. HappyCat Function 6. Expanded Scaffer's Function		***AVG***	**1.76E+03**	1.06E+04
			STD	**2.76E+02**	9.49E+02
			Min	**1.25E+03**	8.74E+03
			Max	**2.35E+03**	1.29E+04

F29	***Composition Function 9*** 1. Hybrid Function 5 2. Hybrid Function 6 3. Hybrid Function 7 Multimodal Non-separable Asymmetrical		***AVG***	**1.22E+04**	1.69E+04
			STD	**1.87E+03**	4.02E+03
			Min	**8.10E+03**	9.86E+03
			Max	**1.66E+04**	2.77E+04
F30	***Composition Function 10*** 1. Hybrid Function 5 2. Hybrid Function 8 3. Hybrid Function 9 Multimodal Non-separable Asymmetrical		***AVG***	**2.08E+09**	5.09E+10
			STD	**9.94E+08**	2.77E+10
			Min	**5.58E+08**	4.47E+09
			Max	**4.73E+09**	1.14E+11
Summary of results			***AVG***	+	2
				−	18
				≈	0
			STD	+	4
				−	16
				≈	0
			Min	+	2
				−	18
				≈	0
			Max	+	0
				−	20
				≈	0

Appendix 5 : Wilcoxon test, p-values results,Wilcoxon rank-sum test with 5% significance

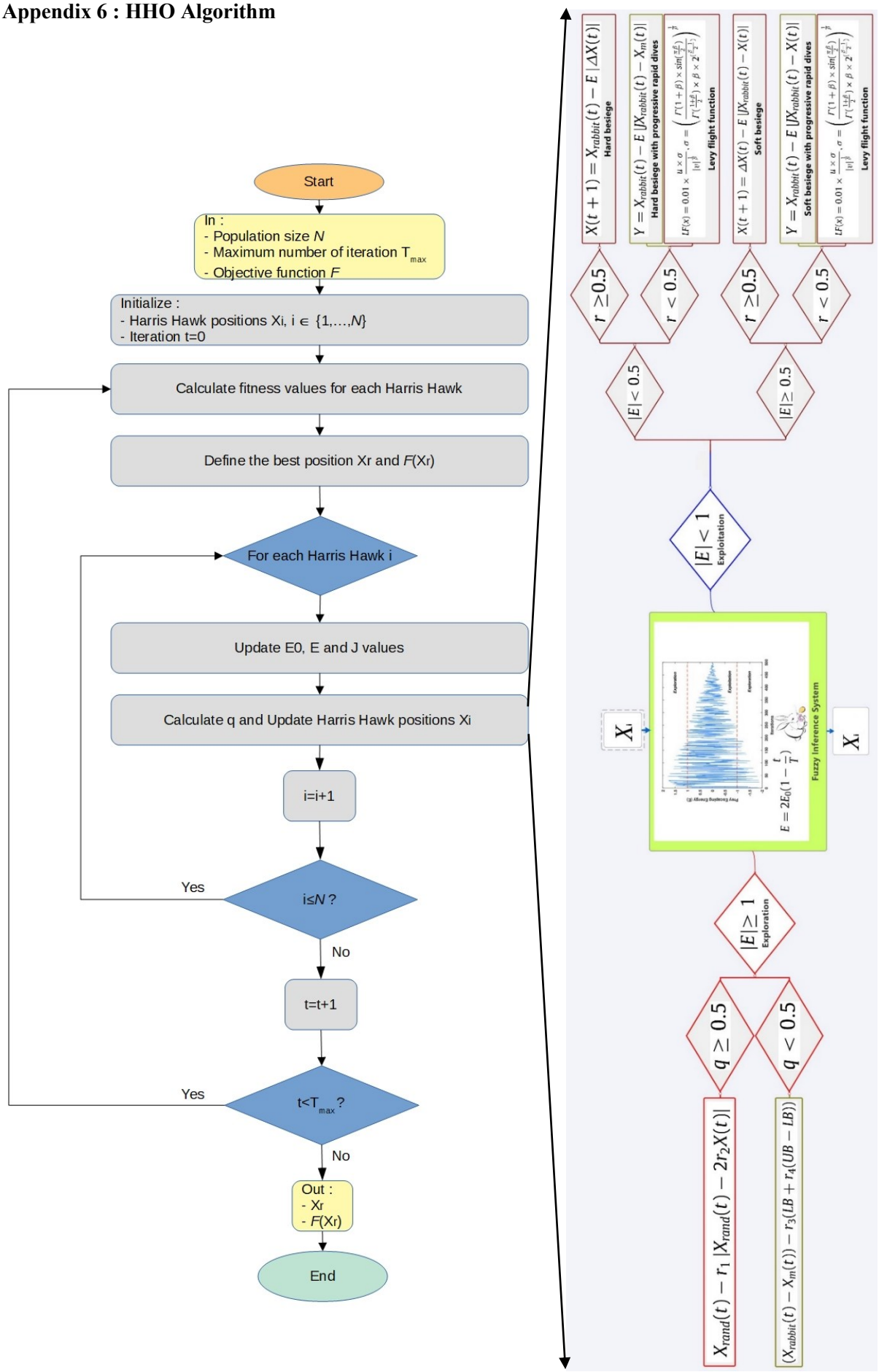
Dimension / Pair of Metaheuristic		2	30	100	1000
HHO\HHO-EAS	p-value	0.01048	0.00109	0.000321	0.0004824
	W+	118	146	153	136
	W-	18	7	0	0
GWO\HHO-EAS	p-value	0.009225	3.815E-06	3.815E-06	3.815E-06
	W+	132	190	190	190
	W-	21	0	0	0
PSO\HHO-EAS	p-value	0.3684	3.815E-06	3.815E-06	3.815E-06
	W+	96	190	190	190
	W-	57	0	0	0

Table A5.1: Wilcoxon tests for general tests

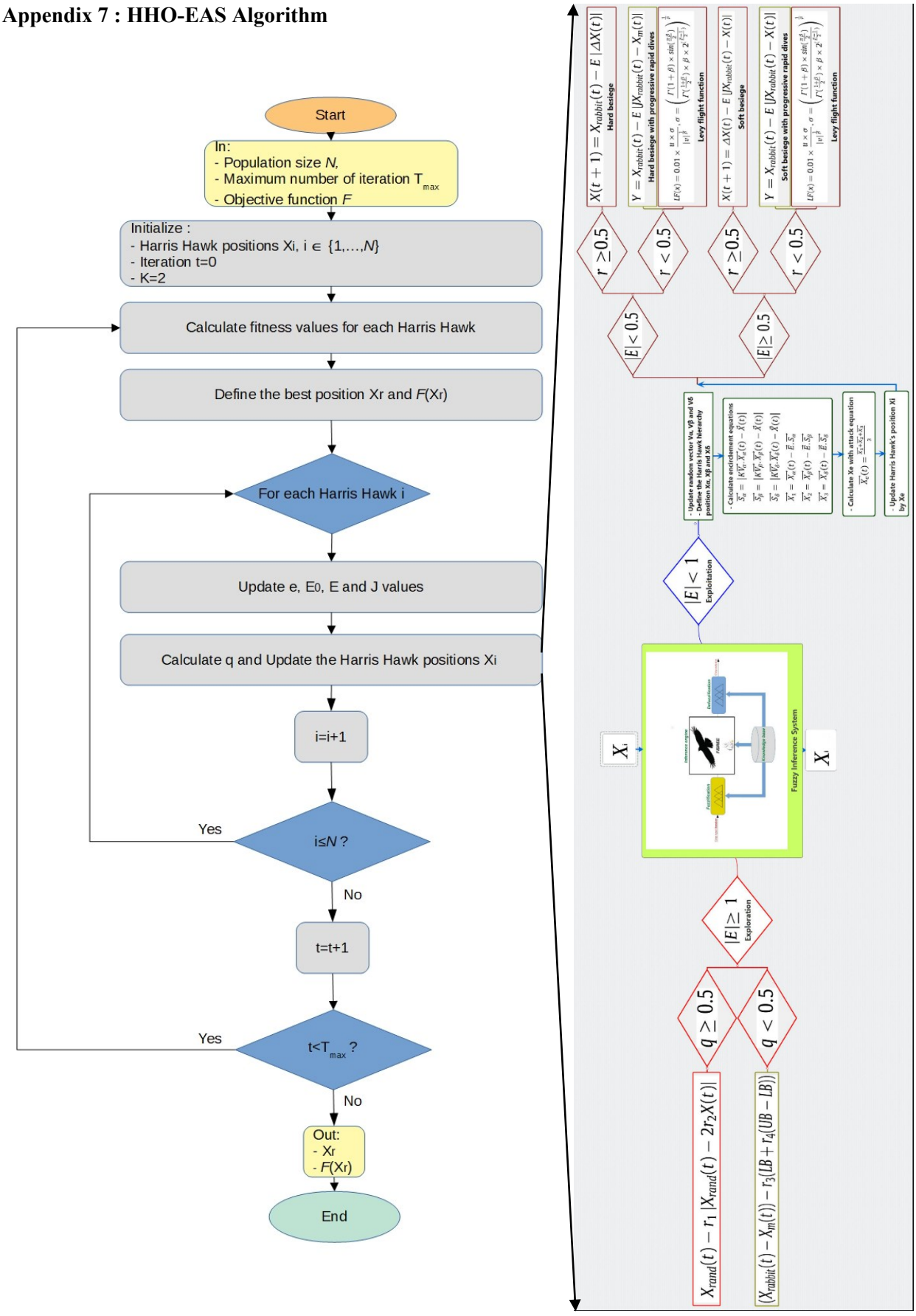
Dimension / Pair of Metaheuristic		100
HHO\HHO-EAS	p-value	1.752E-04
	W+	185
	W-	25

Table A5.2: Wilcoxon tests for specific tests

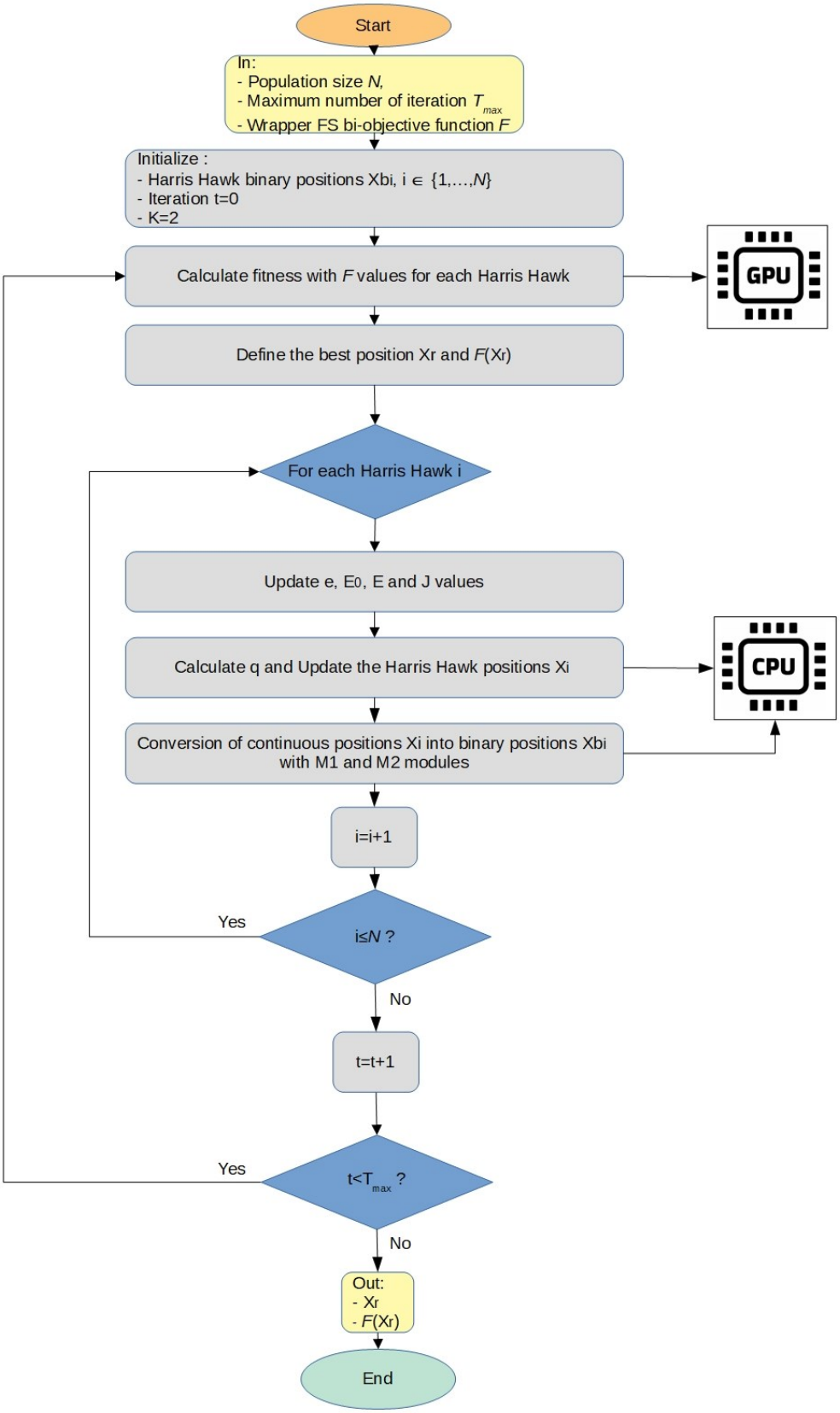
Appendix 6 : HHO Algorithm



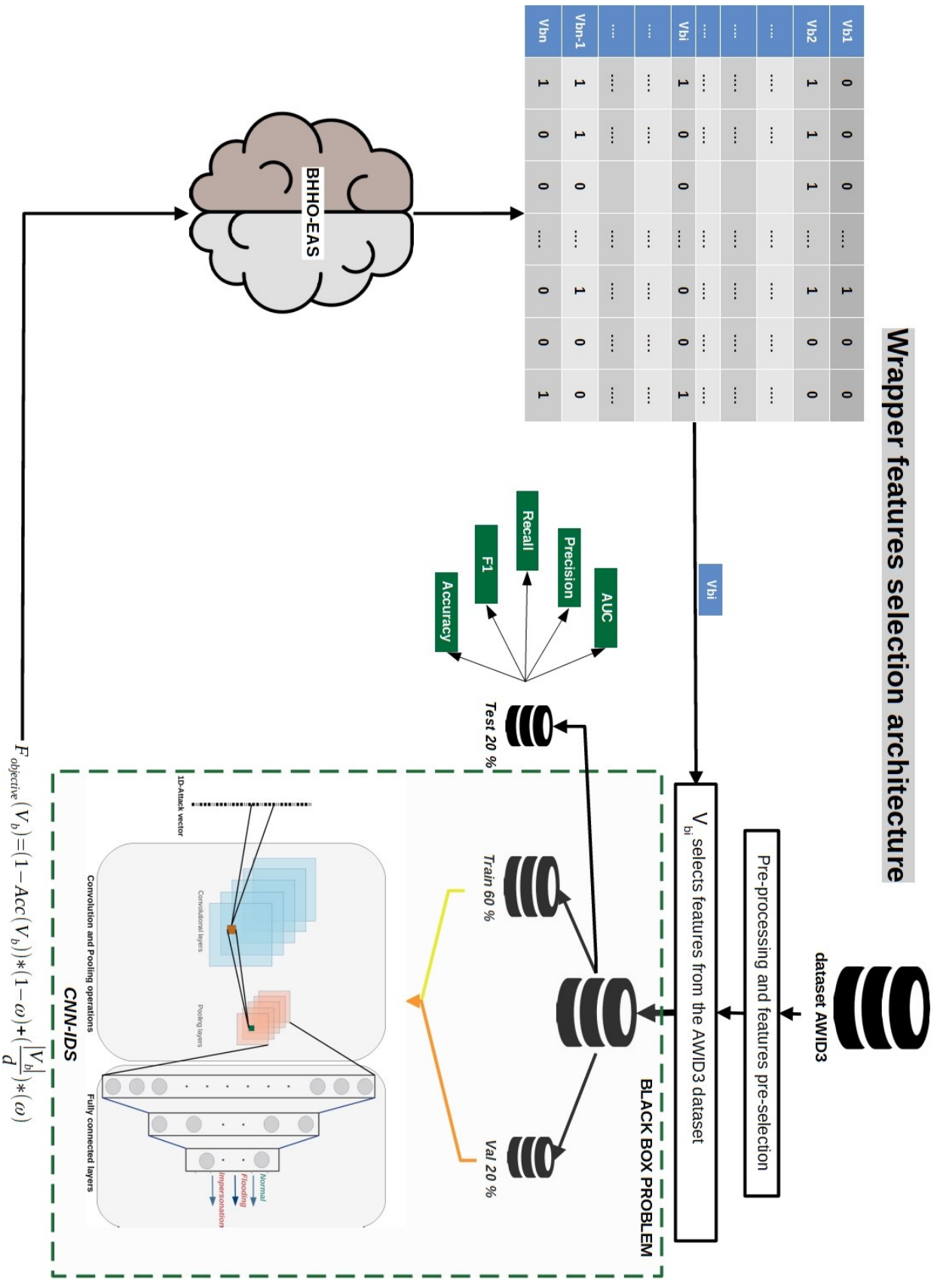
Appendix 7 : HHO-EAS Algorithm



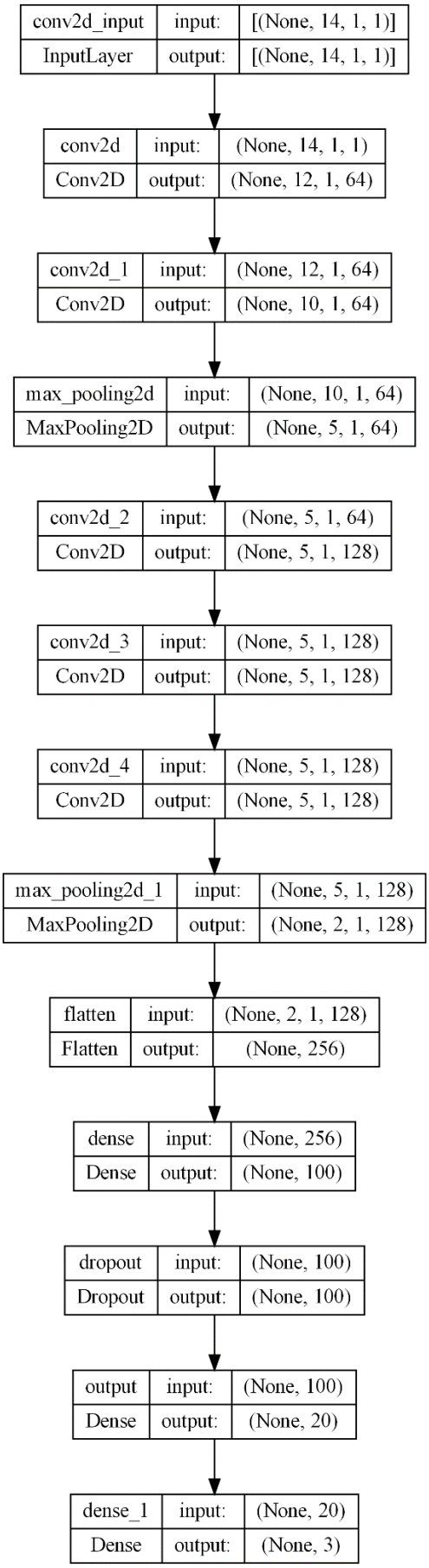
Appendix 8: BHHO-EAS Algorithm

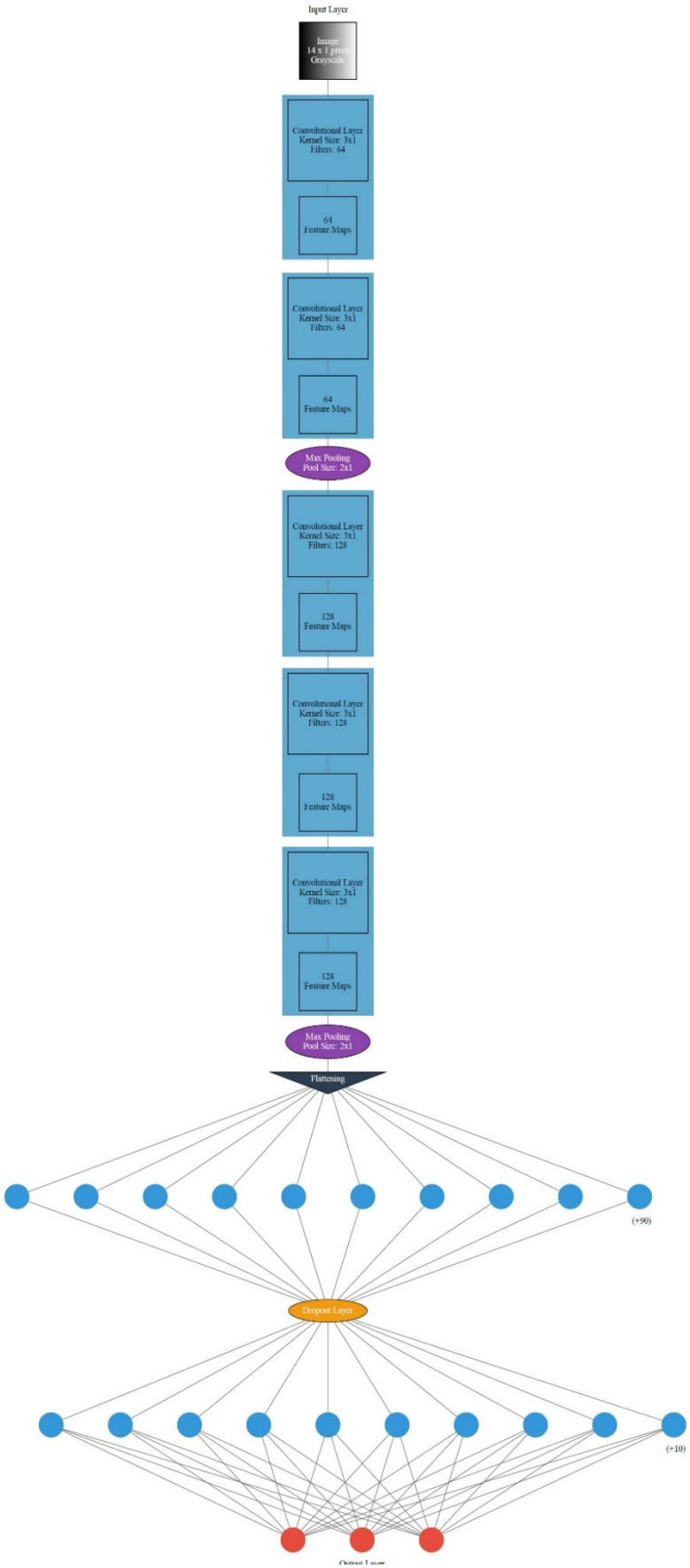


Appendix 9: Wrapper features selection architecture



Appendix 10: Diagramme and 2D architecture of CNN-IDS





Appendix 11: Experimental results of BHHO-EAS

Metaheuristic	*Metrics*	*Values*
BHHO-EAS	AVG	8.40E-03
	STD	3.31E-04
	Min	7.07E-03
	Max	8.86E-03

Table A11.1. Objective function values from the Wrapper feature selection optimization

Metaheuristic	*Metrics*	*Values*
BHHO-EAS	AVG	4,6
	STD	1.35
	Min	4
	Max	8

Table A11.2. Features number values

Metaheuristic	*Metrics*	*Values*
BHHO-EAS	AVG	99.07E-02
	STD	5.96E-03
	Min	98.08E-02
	Max	99.58E-02

Table A11. 3. Classification accuracy values

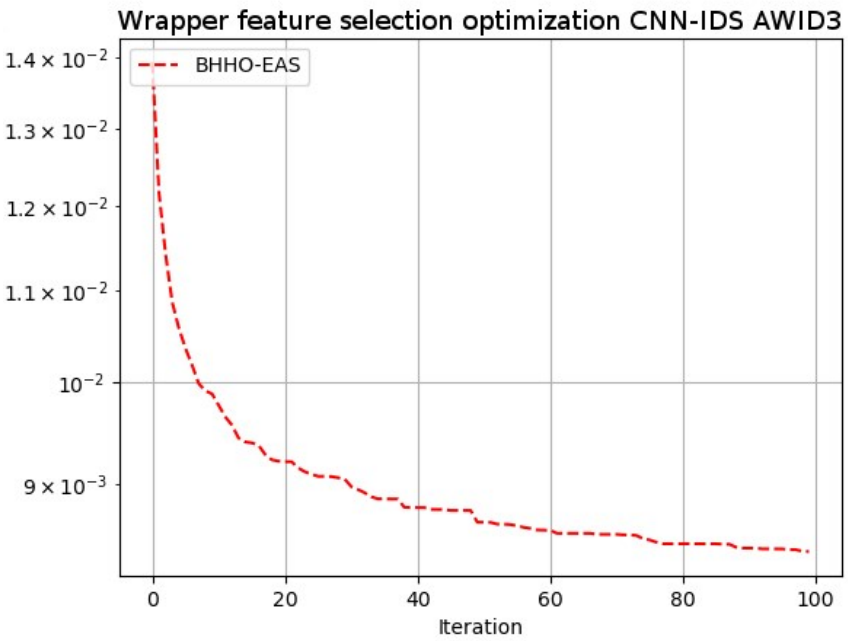


Fig. A11.1. Average convergence curve of the Wrapper feature selection multi-objective optimization problem

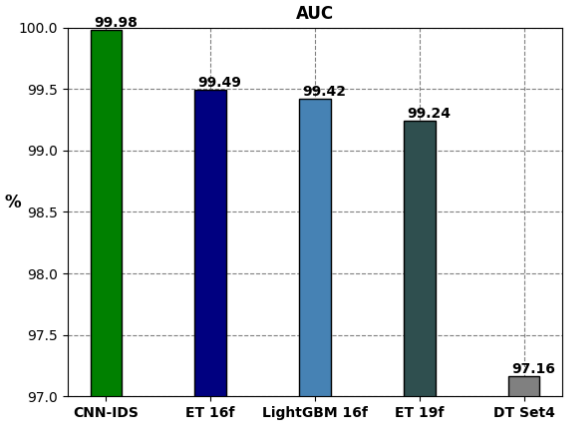
Index	Features selected	Description
1	***radiotap.present.tsft***	This flag signals that the AP will synchronize the MAC timestamp of the associated STAs. In a Wi-Fi network with a predefined flag for the synchronization of MAC timestamps, not taking this flag into account in the attack makes it possible to strengthen the detection of *Impersonation* or *Flooding* attacks with the support of the features below.
2	***radiotap.channel.freq,***	This feature provides the frequency value in Mhz of the channel with distinct frequencies from the 2.4Ghz and 5Ghz frequencies. These frequencies change according to the 802.11 standard and the country. The attacks exploit different channels with frequencies that may be inconsistent with that of the AP. This helps detect an *Impersonation* or *Flooding* attack.
3	***wlan.fc.type***	There are 3 types of frame: Management, Control and Data. Combined with the *wlan.fc.protected* feature, *wlan.fc.type* helps detect *Flooding* and *Impersonation* attacks.
10	***wlan.fc.protected***	Provided by the AP, this flag indicates to the STA that the frame is encrypted. With the 802.11w amendment, the MFP also protects management frames such as *Deauthentication* and *Disassociation* frames. The same is true for *Beacon* frames. This flag thus makes it possible to accompany the distinction between the 802.11 *Normal*, *Flooding* or *Impersonation* flow
13	***frame.len***	This is the total length of a frame. This length varies by frame type and subtype. But also depending on its protected state or not. Thus combined with *wlan.fc.type* and *wlan.fc.protected*, *frame.len* is an indicator of a *Flooding* or *Impersonation* attack.
15	***radiotap.length***	This feature, although specific to the wireless interface driver, is common to all Wi-Fi wireless devices. It provides the size of the Radiotap field. Radiotap is the field of the frame which provides physical data on the device such as the datarate, the frequency, the type of frequency modulation or the signal strength of the antennas. Combined with *frame.len*, this feature can detect *Impersonation* type attacks.
17	***wlan_radio.signal_dbm***	This is the transmit power of the device in dBm. The STA of an attacker with a position further away than a legitimate STA therefore has an ab normal transmission power compared to the latter. Combined with the *frame.len* feature, *wlan_radio.signal_dbm* helps detect Flooding and Impersonation attacks.
18	***wlan_radio.duration***	This feature provides the duration in ms of the frame taking into account the physical data and the 802.11 data, which the sender allocates for the benefit of the acknowledgment frame. This feature is independent of the frames that precede and succeed it. This feature is therefore consistent with our specifications. *wlan_radio.duration* scales based on frame type and subtype. Combin ed with *wlan_radio.signal_dbm*, it identifies *Flooding* and *Impersonat ion* attacks.

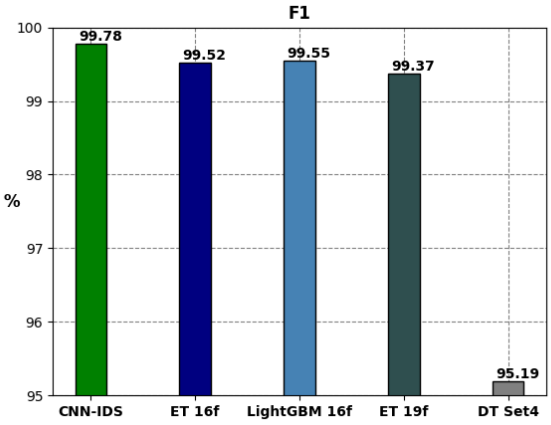
Table A11.4. Description of the 8 features selected by BHHO-EAS from AWID3 dataset

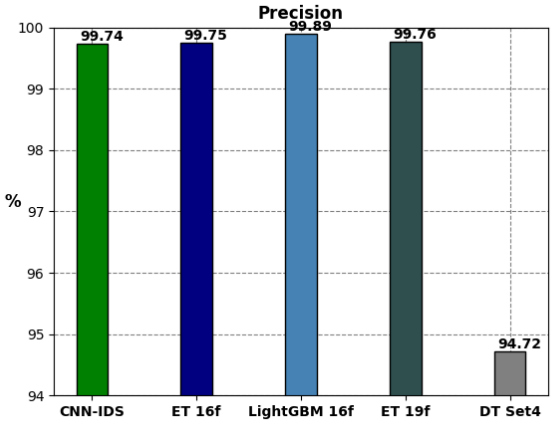
Appendix 12: Performances of the IDS trained with AWID3 dataset

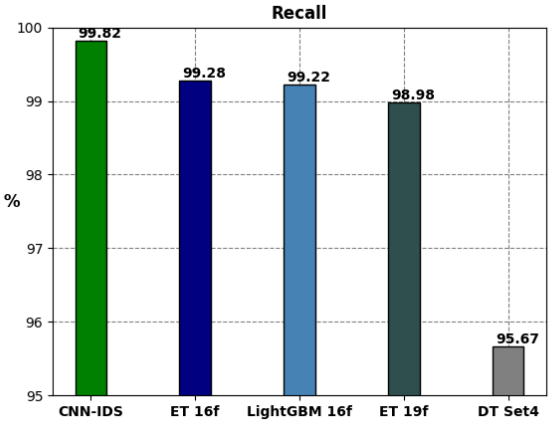
	AUC	Prec	Recall	F1-score	Acc	Epochs
ET 16f	99,49	99,75	99,28	99,52	99,96	N/A
LightGBM 16f	99,42	99,89	99,22	99,55	99,96	N/A
ET 19f	99,24	99,76	98,98	99,37	99,94	N/A
DT Set 4	**97,16**	94,72	95,67	**95,19**	99,69	N/A
CNN-IDS	**99,98**	99,74	99,82	**99,78**	99,84	7,6
BEST :	**99,98**	99,89	99,82	**99,78**	99,96	7,6
WORST :	**97,16**	92,40	87,70	**95,19**	99,46	N/A

Table A12.1. Performance comparison between the best performance results in [Chatzoglou E et al., 2022a] and the CNN-IDS performances









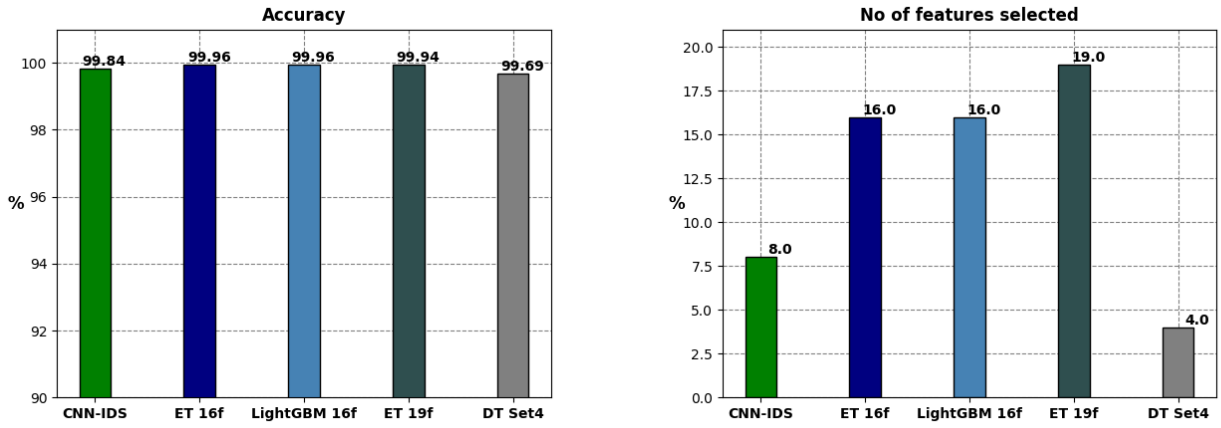
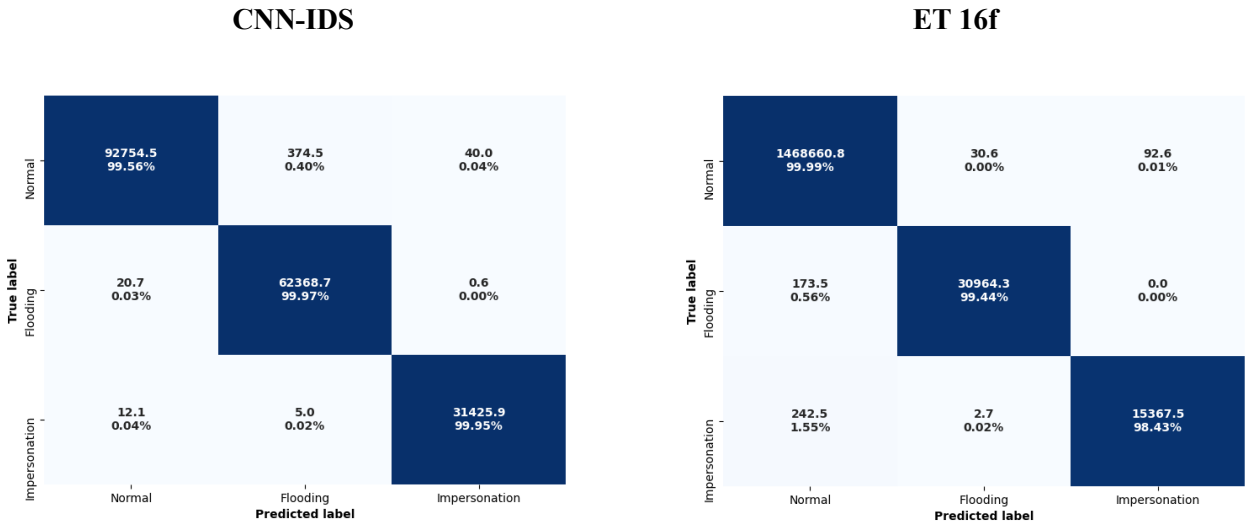
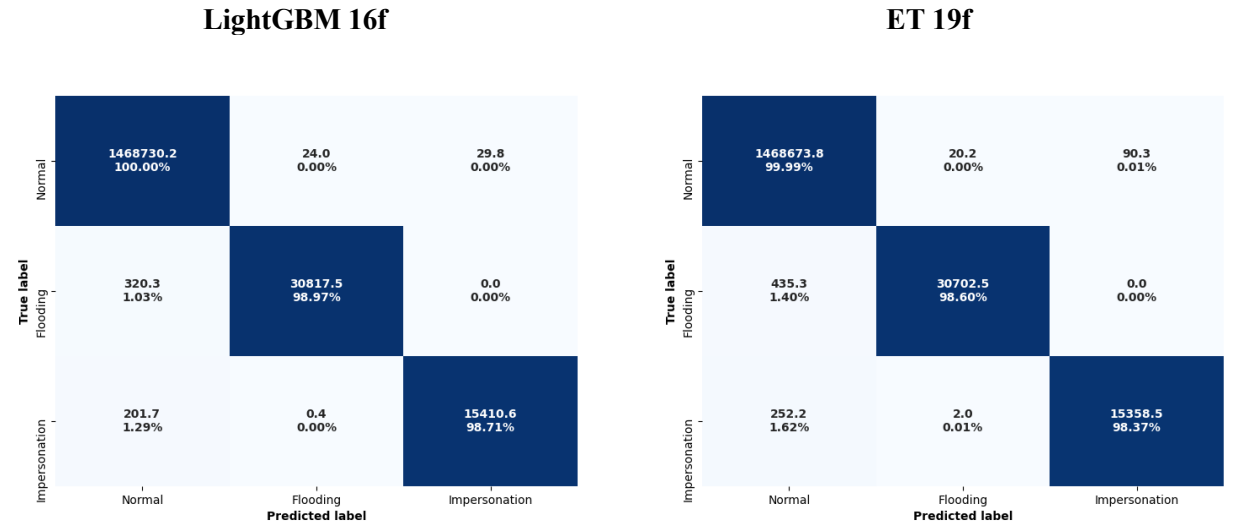


Fig. A12.1. Comparison between CNN-IDS and the 4 best Machine Learning IDS models of [Chatzoglou E et al., 2022a]





DT 4f Set4

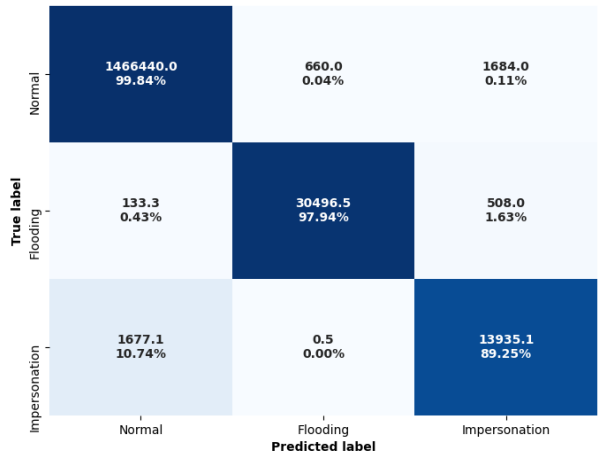


Fig. A12.2. Confusion matrix of the detection capabilities of the 3 classes of attacks by CNN-IDS and the 4 best Machine Learning IDS models of [Chatzoglou E et al., 2022a]

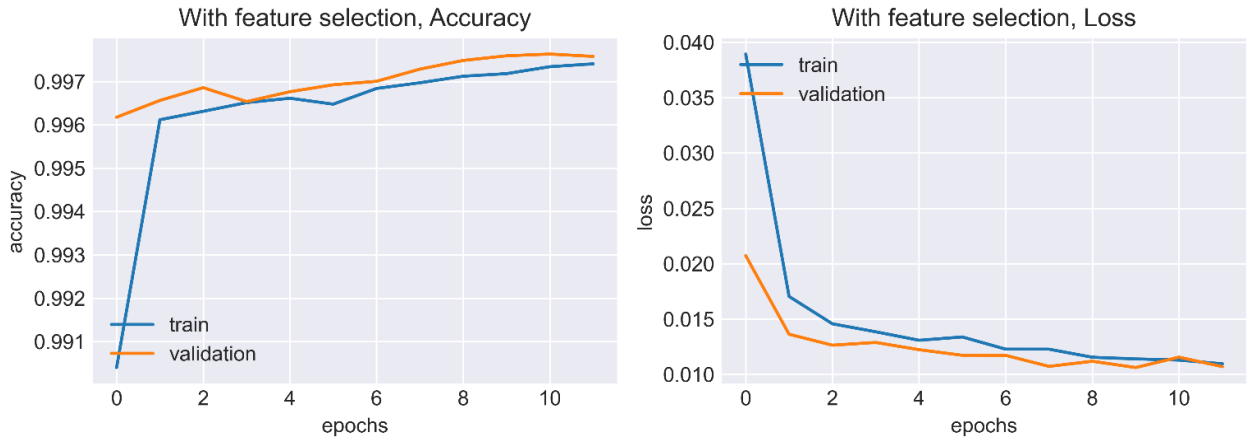


Fig. A12.3. Best Accuracy and Loss curve of CNN-IDS on the 10 folds

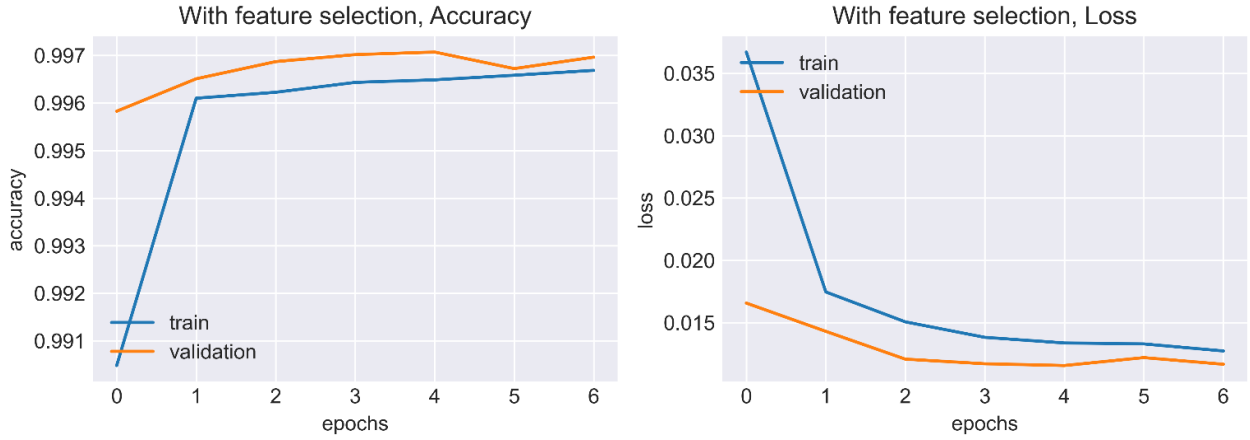


Fig. A12.4. Worst Accuracy and Loss curve of CNN-IDS on the 10 folds

Appendix 13: Performances of the E-CNN-IDS embedded on a Raspberry Pi 4

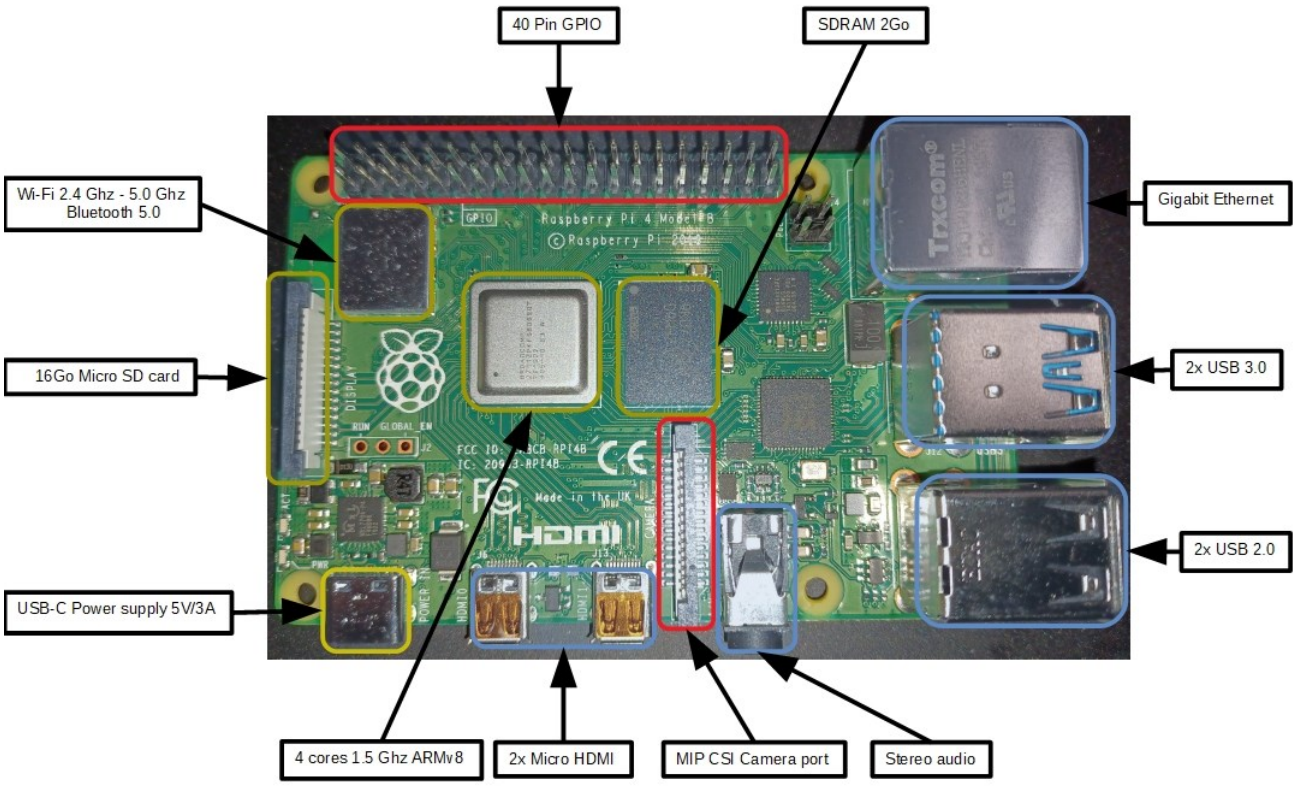


Fig. A13.1. IoT environment Raspberry Pi 4 Model B with E-CNN-IDS embedded

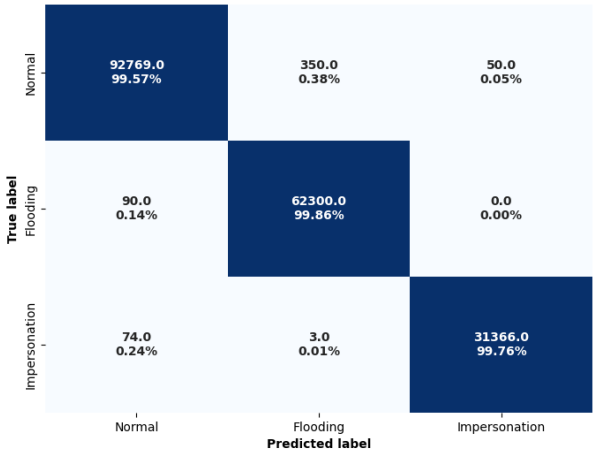


Fig. A13.2. Confusion matrix of the detection capabilities of the 3 classes of attacks by E-CNN-IDS

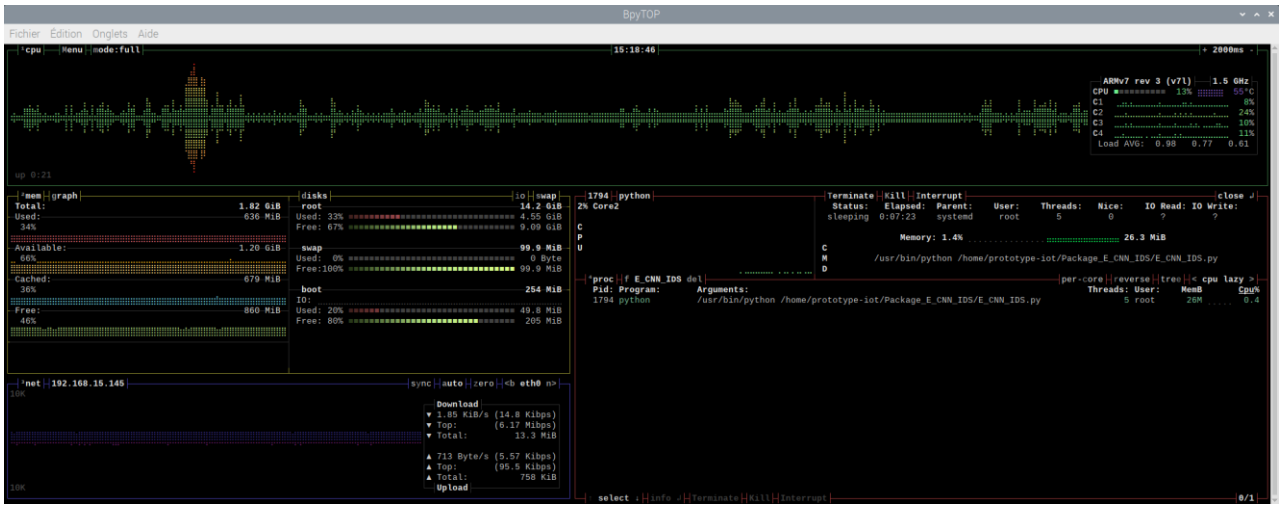


Fig. A13.3. Minimum consumption of memory and computing resources of the E-CNN-IDS

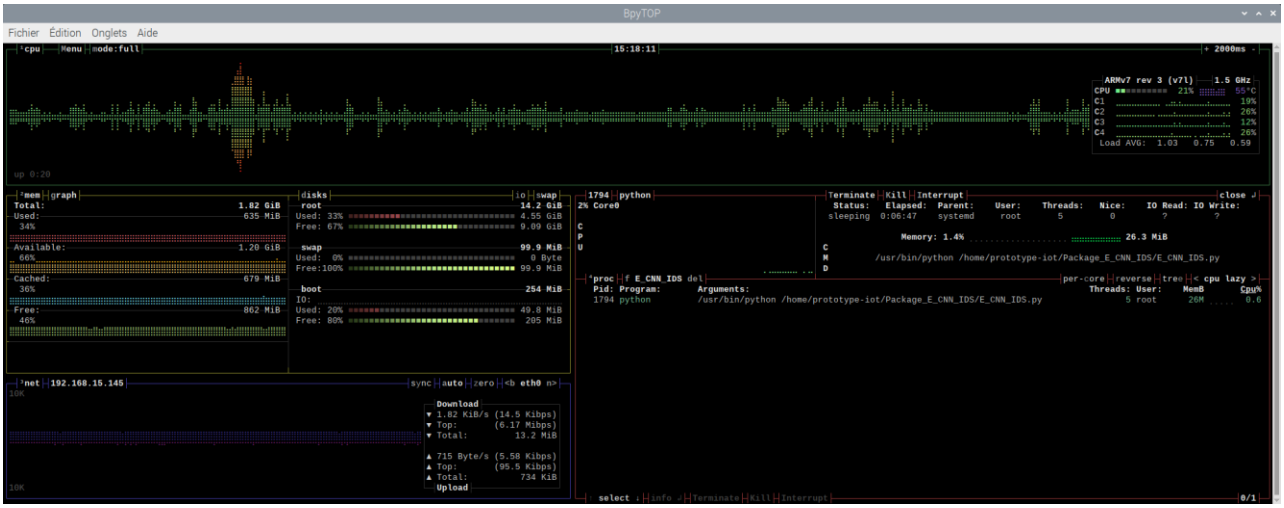


Fig. A13.4. Maximum consumption of memory and computing resources of the E-CNN-IDS

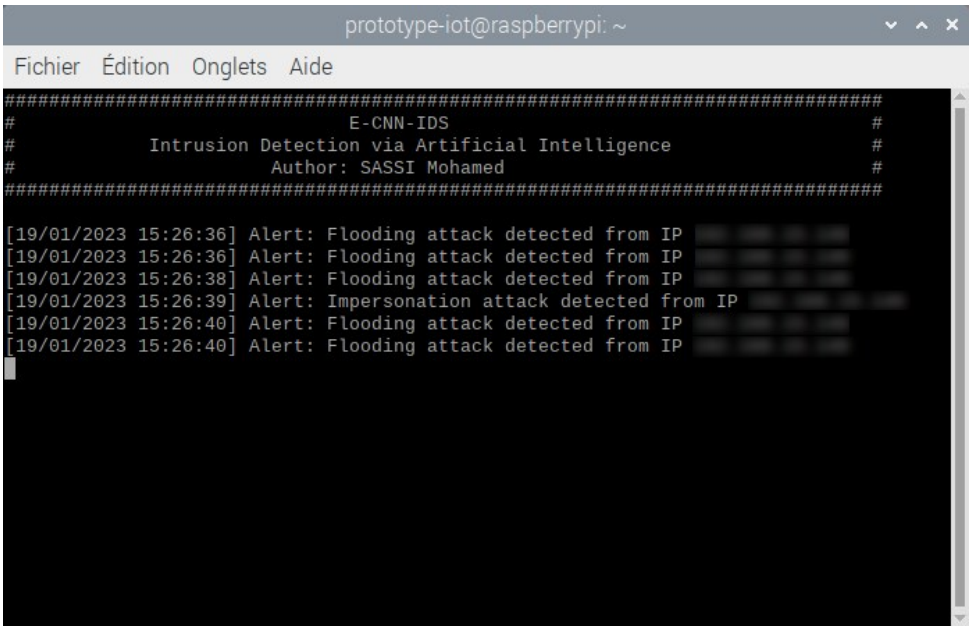


Fig. A13.5. Detection by E-CNN-IDS of six 802.11 specific attacks

References

[Abd EM et al., 2021] Abd EM, Dahou A, Abualigah, L. (2021) Advanced metaheuristic optimization techniques in applications of deep neural networks: a review. In: Neural Computing and Applications volume. pp 14079–14099

[Abdel-Basset M et al., 2018] Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Metaheuristic Algorithms: A Comprehensive Review. In: Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications. pp 185–231

[Abdollahzadeh B et al., 2022] Abdollahzadeh B, Gharehchopogh FS (2022) A multi-objective optimization algorithm for feature selection problems. In: Engineering with Computers. Springer, pp 1845–1863

[Abualigah L et al., 2021] Abualigah L, Abd Elaziz M, Shehab M, Alomari OA, Alshinwan M, Alabool H, Al-Arabiat DA (2021) Hybrid Harris Hawks Optimization with Differential Evolution for Data Clustering. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 267–299

[Adam SP et al., 2019] Adam SP, Alexandropoulos SAN, Pardalos PM, Vrahatis MN (2019) No Free Lunch Theorem: A Review. In: Approximation and Optimization. Springer, pp 57–82

[Agrawal P et al., 2021] Agrawal P, Abutarboush HF, Ganesh T, Mohamed AW (2021) Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019). In: IEEE Access. IEEE, pp 26766–26791

[Agushaka JO et al., 2022] Agushaka JO, Ezugwu AE (2022) Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review. In: Applied Sciences. mdpi

[Ahmed F et al., 2021] Ahmed F, Asghar MZ, Imran A (2021) Combinatorial Optimization for Artificial Intelligence Enabled Mobile Network Automation. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 663–690

[Alabas-Uslu C et al., 2020] Alabas-Uslu C, Dengiz B (2020) Parameter Tuning Problem in Metaheuristics: A Self-Adaptive Local Search Algorithm for Combinatorial Problems. In: Women in Industrial and Systems Engineering. Springer, pp 93–111

[Alabool HM et al., 2021] Alabool HM, Alarabiat D, Abualigah L (2021) Harris hawks optimization: a comprehensive review of recent variants and applications. In: Neural Computing and Applications. Springer, pp 8939–8980

[Alkebsi K et al., 2020] Alkebsi K, Du W (2020) A Fast Multi-Objective Particle Swarm Optimization Algorithm Based on a New Archive Updating Mechanism. IEEE, pp 124734–124754

[Almomani A et al., 2019] Almomani A, Alweshah M, Al Khalayleh S, Al-Refai M, Qashi R (2019) Metaheuristic Algorithms-based Feature Selection Approach for Intrusion Detection. In: Machine Learning for Computer and Cyber Security, 1st Edition. CRC Press, pp 184–208

[Al-Tashi Q et al., 2020] Al-Tashi Q, Abdulkadir SJ, Rais HMd, Mirjalili S, Alhussian H, Ragab MG, Alqushaibi A (2020) Binary Multi-Objective Grey Wolf Optimizer for Feature Selection in Classification. IEEE, pp 106247–106263

[Al-Wajih R et al., 2021] Al-Wajih R, Abdulkadir SJ, Aziz N, Al-Tashi Q, Talpur N (2021) Hybrid Binary Grey Wolf With Harris Hawks Optimizer for Feature Selection. In: IEEE Access. IEEE, pp 31662–31677

[Avjeet S et al., 2021] Avjeet S, Kumar A (2021) Applications of nature-inspired meta-heuristic algorithms: a survey. In: International Journal of Advanced Intelligence Paradigms. INDERSCIENCE, pp 388–417

[Azizjon M et al., 2020] Azizjon M, Jumabek A, Kim W (2020) 1D CNN based network intrusion detection with normalization on imbalanced data. International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp 218–224

[Banka H et al., 2017a] Banka H, Dara S (2017) High Dimensional Feature Selection: A Survey. In: Feature Selection in High Dimensional Data Using Metaheuristic. LAP LAMBERT Academic, Republic of Moldova, pp 27–35

[Banka H et al., 2017b] Banka H, Dara S (2017) Feature Selection. In: Feature Selection in High Dimensional Data Using Metaheuristic. LAP LAMBERT Academic, Republic of Moldova, pp 15–17

[Baron C et al., 2005] Baron C, Rochet S, Gutierrez C (2005) Proposition of a methodology for the management of innovative design projects

[Bhosale YH et al., 2022] Bhosale YH, Sridhar Patnaik K (2022) IoT Deployable Lightweight Deep Learning Application For COVID-19 Detection With Lung Diseases Using RaspberryPi. In: 2022 International Conference on IoT and Blockchain Technology (ICIBT). IEEE, pp 1–6

[Birogul S, 2019] Birogul S (2019) Hybrid Harrison Hawk Optimization Based on Differential Evolution (HHODE) Algorithm for Optimal Power Flow Problem. IEEE Access, pp 184468–184488

[Bonab MS et al., 2020] Bonab MS, Ghaffari A, Gharehchopogh FS, Alemi P (2020) A wrapper-based feature selection for improving performance of intrusion detection systems. In: International Journal of Communication Systems. WILEY

[Bouchaala A et al., 2022] Bouchaala A, Merroun O, Sakim A (2022) A MOPSO algorithm based on pareto dominance concept for comprehensive analysis of a conventional adsorption desiccant cooling system. In: Journal of Building Engineering. ELSEVIER, p 105189

[Burnwal S et al., 2013] Burnwal S, Deb, S. (2013) Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. In: The International Journal of Advanced Manufacturing Technology. Springer, pp 951–959

[Chatzoglou E et al., 2021] Chatzoglou E, Kambourakis G, Kolias C (2021) Empirical Evaluation of Attacks Against IEEE 802.11 Enterprise Networks: The AWID3 Dataset. IEEE Access, pp 34188–34205

[Chatzoglou E et al., 2022a] Chatzoglou E, Kambourakis G, Kolias C, Smiliotopoulos C (2022) Pick quality over quantity: Expert feature selection and data pre-processing for 802.11 Intrusion Detection Systems. IEEE Access

[Chatzoglou E et al., 2022b] Chatzoglou E, Kambourakis G, Smiliotopoulos C, Kolias C (2022) Best of Both Worlds: Detecting Application Layer Attacks through 802.11 and Non-802.11 Features. Sensors, p 5633

[Chelouah R, 2000] Chelouah R (2000) Adaptation aux problèmes à variables continues de plusieurs métaheuristiques d'optimisation combinatoire : la méthode tabou, les algorithmes génétiques et les méthodes hybrides. Application en contrôle non destructif

[Chelouah R et al., 2007] Chelouah R, Baron C (2007) Ant colony algorithm hybridized with tabu and greedy searches as applied to multi-objective optimization in project management. In: Journal of Heuristics. Springer

[Chelouah R et al., 2009] Chelouah R, Baron C, Zholghadri M, Gutierrez C (2009) Meta-heuristics for System Design Engineering. In: Foundations of Computational Intelligence Volume 3. Springer, Berlin, Heidelberg, pp 387–423

[Chen H et al., 2020] Chen H, Heidari AA, Chen H, Wang M, Pan Z, Gandomi AH (2020) Multi-population differential evolution-assisted Harris hawks optimization: Framework and case studies. In: Future Generation Computer Systems. ELSEVIER, pp 175–198

[Chiandussi G et al., 2012] Chiandussi G, Codegone M, Ferrero S, Varesio F.E. (2012) Comparison of multi-objective optimization methodologies for engineering applications. In: Computers & Mathematics with Applications. ELSEVIER, pp 912–942

[Congying L et al., 2011] Congying L, Huanping Z, Xinfeng Y (2011) Particle swarm optimization algorithm for quadratic assignment problem. In: Proceedings of 2011 International Conference on Computer Science and Network Technology. IEEE, pp 1728–1731

[Cortés-Toro EM et al., 2018] Cortés-Toro EM, Crawford B, Gómez-Pulido JA, Soto R, Lanza-Gutiérrez JM (2018) A New Metaheuristic Inspired by the Vapour-Liquid Equilibrium for Continuous Optimization. In: Applied Sciences. MDPI

[Coulson J et al., 2012] Coulson J, Coulson T (2012) The Harris's Hawk Revolution, 1st edn. Parabuteo Publishing, Kingman, Arizona

[Coulson JO, 2013] Coulson JO (2013) Reexamining cooperative hunting in Harris's Hawk (Parabuteo Unicinctus): large prey or challenging habitats ? In: The Auk. The American Ornithologists' Union, p 548−552

[Crawford B et al., 2013] Crawford B, Valenzuela C, Soto R, Monfroy E, Paredes F (2013) Parameter Tuning of Metaheuristics Using Metaheuristics. In: Advanced Science Letters. Science letters, pp 3556–3559

[Cuevas E et al., 2021a] Cuevas E, Diaz P, Camarena O (2021) Experimental Analysis Between Exploration and Exploitation. In: Metaheuristic Computation: A Performance Perspective. Springer, pp 249–269

[Cuevas E et al., 2021b] Cuevas E, Rodríguez A, Alejo-Reyes A, Del-Valle-Soto C (2021) Introductory Concepts of Metaheuristic Computation. In: Recent Metaheuristic Computation Schemes in Engineering. Springer, pp 1–9

[Cuevas E et al., 2021c] Cuevas E, Rodríguez A, Alejo-Reyes A, Del-Valle-Soto C (2021) Metaheuristic Algorithms for Wireless Sensor Networks. In: Recent Metaheuristic Computation Schemes in Engineering. Springer, pp 193–235

[Cuevas E et al., 2021d] Cuevas E, Rodríguez A, Alejo-Reyes A, Del-Valle-Soto C (2021) Metaheuristic Algorithms Applied to the Inventory Problem. In: Recent Metaheuristic Computation Schemes in Engineering. Springer, pp 237–277

[Cuevas E et al., 2021e] Cuevas E, Rodríguez A, Alejo-Reyes A, Del-Valle-Soto C (2021) Metaheuristic Algorithm Based on Hybridization of Invasive Weed Optimization asnd Estimation Distribution Methods. In: Recent Metaheuristic Computation Schemes in Engineering. Springer, pp 63–123

[Dara S et al., 2022] Dara S, Dhamercherla S, Jadav SS (2022) Machine Learning in Drug Discovery: A Review. In: Artificial Intelligence Review volume. Springer, pp 1947–1999

[Davahli A et al., 2020a] Davahli A, Shamsi M, Abaei, G. (2020) Hybridizing genetic algorithm and grey wolf optimizer to advance an intelligent and lightweight intrusion detection system for IoT wireless

networks. In: Journal of Ambient Intelligence and Humanized Computing volume. Springer, pp 5581–5609

[Davahli A et al., 2020b] Davahli A, Shamsi M, Abaei G (2020) A Lightweight Anomaly Detection Model using SVM for WSNs in IoT through a Hybrid Feature Selection Algorithm based on GA and GWO. In: Journal of Computing and Security. pp 63–79

[Dawson JW et al., 1991] Dawson JW, Mannan RW (1991) Dominance Hierarchies and Helper Contributions in Harris Hawks. In: The Auk. Central Ornithology Publication Office, pp 649–660

[Deb K et al., 2002] Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. In: IEEE Transactions on Evolutionary Computation. IEEE, pp 182–197

[Demyana IE et al., 2016] Demyana IE, Adil Y (2016) Scheduling Jobs on Cloud Computing using Firefly Algorithm. In: International Journal of Grid and Distributed Computing. pp 149–158

[Dhiman G et al., 2017] Dhiman G, Kumar V (2017) Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. In: Advances in Engineering Software. ELSEVIER, pp 48–70

[Digalakis JG et al., 2000] Digalakis JG, Margaritis KG (2000) On benchmarking functions for genetic algorithms. In: International Journal of Computer Mathematics. Taylor&Francis, pp 481–506

[Donoso Y et al., 2016] Donoso Y, Fabregat R (2016) Multi-Objective Optimization in Computer Networks Using Metaheuristics. Auerbach

[Drewek-Ossowicka A et al., 2021] Drewek-Ossowicka A, Pietrołaj M, Rumiński J (2021) A survey of neural networks usage for intrusion detection systems. In: Journal of Ambient Intelligence and Humanized Computing volume. Springer, pp 497–514

[Du D et al., 2014] Du D, Ko K (2014) Theory of Computational Complexity, 2nd edn. WILEY

[Eiben AE et al., 2011] Eiben AE, Smit SK (2011) Evolutionary Algorithm Parameters and Methods to Tune Them. In: Autonomous Search. Springer, pp 15–36

[Elkadeem MR et al., 2019] Elkadeem MR, Elaziz MA, Ullah Z, Wang S, Sharshir SW (2019) Optimal Planning of Renewable Energy-Integrated Distribution System Considering Uncertainties. In: IEEE Access. pp 164887–164907

[El-Shorbagy M, Hassanien AE, 2018] El-Shorbagy M, Hassanien AE (2018) Particle Swarm Optimization from Theory to Applications. In: International Journal of Rough Sets and Data Analysis. IGI, pp 1–24

[Emmerich MTM et al., 2018] Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. In: Natural Computing. Springer, pp 585–609

[Eschenauer H et al., 1990] Eschenauer H, Koski J, Osyczka A (1990) Multicriteria design optimization : procedures and applications. Springer-Verlag

[Fan Q et al., 2020] Fan Q, Chen Z, Xia Z (2020) A novel quasi-reflected Harris hawks optimization algorithm for global optimization problems. In: Soft Computing. Springer, pp 14825–14843

[Faris H et al., 2018] Faris H, Mafarja MM, Heidari AA, Aljarah I, Al-Zoubi AM, Mirjalili S, Fujita H (2018) An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. In: Knowledge-Based Systems. Science Direct, pp 43–67

[Fukushima K, 1980] Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In: Biological Cybernetics volume. pp 193–202

[Gabor T et al., 2017] Gabor T, Belzner L (2017) Genealogical distance as a diversity estimate in evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion GECCO '17. ACM, pp 1572–1577

[Gad AG et al., 2022] Gad AG, Sallam KM, Chakrabortty, R.K. (2022) An improved binary sparrow search algorithm for feature selection in data classification. In: Neural Computing and Applications

[Gao ZM et al., 2019] Gao ZM, Zhao J, Hu YR, Chen HF (2019) The improved Harris hawk optimization algorithm with the Tent map. In: 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE). IEEE, pp 336–339

[Gardeux V, 2011] Gardeux V (2011) Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à ADN. Université de Paris-Est Créteil

[Gardeux V et al., 2011] Gardeux V, Chelouah R, Siarry P (2011) EM323: a line search based algorithm for solving high-dimensional continuous non-linear optimization problems. In: Soft computing. Springer, pp 2275–2285

[Gardeux V et al., 2012] Gardeux V, Natowicz R, Chelouah R, Siarry P (2012) ABEUS : un algorithme d'optimisation discret appliqué à la sélection de variables sur des jeux de données de transcriptomique. In: Recherche Opérationnelle et Aide à la Décision (ROADEF)

[Gaspar A et al., 2021] Gaspar A, Oliva D, Cuevas E, Zaldívar D, Pérez M, Pajares G (2021) Hyperparameter Optimization in a Convolutional Neural Network Using Metaheuristic Algorithms. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 37–59

[Ghazali R et al., 2018] Ghazali R, Deris M, Nawi N, Abawajy J (2018) Exploration and Exploitation Measurement in Swarm-Based Metaheuristic Algorithms: An Empirical Analysis. In: Recent Advances on Soft Computing and Data Mining. Springer, pp 24–32

[Glover F et al., 1998] Glover F, Laguna M (1998) Tabu Search. In: Handbook of Combinatorial Optimization. Springer, pp 621–757

[Gonzalez-Huitron V et al., 2021] Gonzalez-Huitron V, León-Borges JA, Rodriguez-Mata AE, Amabilis-Sosa LE, Ramírez-Pereda B, Rodriguez H (2021) Disease detection in tomato leaves via CNN with lightweight architectures implemented in Raspberry Pi 4. In: Computers and Electronics in Agriculture. ELSEVIER

[Goodfellow I et al., 2016] Goodfellow I, Courville A, Bengio Y (2016) Convolutional Networks. In: Deep Learning. The MIT Press, Cambridge, Massachusetts, United States, pp 326–366

[Gupta S et al., 2020] Gupta S, Deep K, Heidari AA, Moayedi H, Wang M (2020) Opposition-based learning Harris hawks optimization with advanced transition rules: principles and analysis. Expert Systems with Applications 158:113510

[Hassan A, 2019] Hassan A (2019) Hybrid metaheuristics: An automated approach. In: Expert Systems With Applications. pp 132–144

[Hayashida T et al., 2017] Hayashida T, Nishizaki I, Sekizaki S, Koto S (2017) Cooperative Particle Swarm Optimization in Distance-Based Clustered Groups. In: Journal of Software Engineering and Applications. Scientific Research, pp 143–158

[Heidari AA et al., 2019] Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: Algorithm and applications. In: Future Generation Computer Systems. ELSEVIER, pp 849–872

[Hijazi NM et al., 2021] Hijazi NM, Faris H, Aljarah I (2021) A parallel metaheuristic approach for ensemble feature selection based on multi-core architectures. In: Expert Systems with Applications. ELSEVIER

[Hillier FS et al., 1995] Hillier FS, Lieberman GJ (1995) Introduction to Operations Research. McGraw-Hill International Editions

[Hodashinsky IA et al., 2019] Hodashinsky IA, Sarin KS (2019) Feature selection: Comparative Analysis of Binary Metaheuristics and Population Based Algorithm with Adaptive Memory. In: Programming and Computing Software. ACM, pp 221–227

[Houssein EH et al., 2021a] Houssein EH, Dirar M, Hussain K (2021) Artificial Neural Networks for Stock Market Prediction: A Comprehensive Review. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 409–444

[Houssein EH et al., 2022a] Houssein EH, Hassan HN, Al-Sayed MM, Nabil E (2022) Intelligent Computational Models for Cancer Diagnosis: A Comprehensive Review. In: Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems. Springer, pp 25–50

[Houssein EH et al., 2020] Houssein EH, Hosney ME, Elhoseny M (2020) Hybrid Harris hawks optimization with cuckoo search for drug design and discovery in chemoinformatics. In: Scientific Reports

[Houssein EH et al., 2022b] Houssein EH, Ibrahim IE, Hassaballah M, Wazery YM (2022) Integration of Machine Learning and Optimization Techniques for Cardiac Health Recognition. In: Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems. Springer, pp 121–148

[Houssein EH et al., 2021b] Houssein EH, Mahdy MA, Shebl D, Mohamed WM (2021) A Survey of Metaheuristic Algorithms for Solving Optimization Problems. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 515–543

[Houssein EH et al., 2021c] Houssein EH, Saad MR, Hussain K, Shaban H, Hassaballah M (2021) A Review of Metaheuristic Optimization Algorithms in Wireless Sensor Networks. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 193–217

[Houssein EH et al., 2022c] Houssein EH, Saber E, Wazery YM, Ali AA (2022) Swarm Intelligence Algorithms-Based Machine Learning Framework for Medical Diagnosis: A Comprehensive Review. In: Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems. Springer, pp 85–106

[Houssein EH et al., 2022d] Houssein EH, Zaki GN, Abualigah L, Younis EMG (2022) Metaheuristics for Parameter Estimation of Solar Photovoltaic Cells: A Comprehensive Review. In: Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems. Springer, pp 149–179

[Hu J et al., 2021] Hu J, Liu C, Cui Y (2021) An Improved CNN Approach for Network Intrusion Detection System. In: International Journal of Network Security. IJNS, pp 569–575

[Huang C et al., 2020] Huang C, Li and Y, Yao X (2020) A Survey of Automatic Parameter Tuning Methods for Metaheuristics. In: IEEE Transactions on Evolutionary Computation. IEEE, pp 201–216

[Huiqin C et al., 2011] Huiqin C, Sheng L, Zheng T (2011) Hybrid Gravitational Search Algorithm with Random-key Encoding Scheme Combined with Simulated Annealing. In: IJCSNS International Journal of Computer Science and Network Security. Computer Science

[Hussain F et al., 2020] Hussain F, Hussain R, Hassan SA, Hossain E (2020) Machine Learning in IoT Security: Current Solutions and Future Challenges. In: IEEE Communications Surveys & Tutorials. IEEE, pp 1686–1721

[Hussain K et al., 2019a] Hussain K, Mohd Salleh MN, Cheng S (2019) Metaheuristic research: a comprehensive survey. In: Springer. pp 2191–2233

[Hussain K et al., 2019b] Hussain K, Salleh MNM, Cheng S (2019) On the exploration and exploitation in popular swarm-based metaheuristic algorithms. In: Neural Computing and Applications. Springer, pp 7665–7683

[Hussain K et al., 2019c] Hussain K, Zhu W, Mohd Salleh MN (2019) Long-Term Memory Harris' Hawk Optimization for High Dimensional and Optimal Power Flow Problems. In: IEEE Access. pp 147596–147616

[Izadkhah H, 2022] Izadkhah H (2022) P, NP, NP-Complete, and NP-Hard Problems. In: Problems on Algorithms. Springer, pp 497–511

[Jerebic J et al., 2021] Jerebic J, Mernik M, Liu SH, Ravber M, Baketarić M, Mernik L, Črepinšek M (2021) A novel direct measure of exploration and exploitation based on attraction basins. In: Expert Systems with Applications. ELSEVIER

[Jia H et al., 2019] Jia H, Lang C, Oliva D, Song W, Peng P (2019) Dynamic Harris Hawks Optimization with Mutation Mechanism for Satellite Image Segmentation. Remote Sensing 11:1421.

[Jiyeon K et al., 2020] Jiyeon K, Jiwon K, Hyunjung K, Minsun S, Eunjung C (2020) CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. In: Electronics

[Joshila Grace LK et al., 2022] Joshila Grace LK, Asha P, Refonaa J, Jany Shabu SL, Viji Amutha Mary A (2022) Detect Fire in Uncertain Environment using Convolutional Neural Network. In: Advances in Intelligent Computing and Communication. Springer, pp 399–404

[Karaboga D et al., 2007] Karaboga D, Basturk B (2007) Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. In: Foundations of Fuzzy Logic and Soft Computing. Springer, pp 789–798

[Kazikova A et al., 2020] Kazikova A, Pluhacek M, Senkerik R (2020) Why Tuning the Control Parameters of Metaheuristic Algorithms Is So Important for Fair Comparison? In: Soft Computing. Mendel

[Khanduja N et al., 2021] Khanduja N, Bhushan B (2021) Recent Advances and Application of Metaheuristic Algorithms: A Survey (2014–2020). In: Metaheuristic and Evolutionary Computation: Algorithms and Applications. Springer, pp 207–228

[Khurma et al., 2020] Khurma RA, Castillo PA, Sharieh A, Aljarah I (2020) New Fitness Functions in Binary Harris Hawks Optimization for Gene Selection in Microarray Datasets. In: Proceedings of the 12th International Joint Conference on Computational Intelligence - ECTA. SCITEPRESS, pp 139–146

[Kim K et al., 2018a] Kim K, Aminanto ME, Tanuwidjaja HC (2018) Intrusion Detection Systems. In: Network Intrusion Detection using Deep Learning. Springer, pp 5–11

[Kim K et al., 2018b] Kim K, Aminanto ME, Tanuwidjaja, H.C. (2018) Classical Machine Learning and Its Applications to IDS. In: Network Intrusion Detection using Deep Learning. Springer, pp 13–26

[Kim K et al., 2018c] Kim K, Aminanto ME, Tanuwidjaja, H.C. (2018) Deep Learning-Based IDSs. In: Network Intrusion Detection using Deep Learning. pp 35–45

[Kirkpatrick S et al., 1983] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science, pp 671–680

[Klir G et al., 1995] Klir G, Yuan B (1995) Fuzzy Sets and Fuzzy Logic: Theory and Applications, 1st edn. Prentice Hall, U.S.A

[Krueger M, 1990] Krueger M (1990) Méthode d'analyse d'algorithmes d'optimisation stochastiques à l'aide d'algorithmes génétiques. Thèse de doctorat en informatique et réseaux de l'Ecole nationale supérieure des télécommunications

[Kumar V et al., 2021] Kumar V, Kumar D, Kaur M, Singh D, Idris SA, Alshazly, H. (2021) A Novel Binary Seagull Optimizer and its Application to Feature Selection Problem. In: IEEE Access. IEEE, pp 103481–103496

[Liu J et al., 2020] Liu J, Li Q, Cao R, Tang W, Qiu G (2020) MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation. In: ISPRS Journal of Photogrammetry and Remote Sensing. ELSEVIER, pp 255–267

[Loubiere P, 2016] Loubiere P (2016) Amélioration des métaheuristiques d'optimisation à l'aide de l'analyse de sensibilité. Université de Paris-Est Créteil

[Loubiere P et al., 2016] Loubiere P, Jourdan J, Siarry P, Chelouah R (2016) A sensitivity analysis method for driving the Artificial Bee Colony algorithm's search process. In: Applied Soft Computing. ELSEVIER, pp 515–531

[Mafarja MM et al., 2017] Mafarja MM, Mirjalili S (2017) Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. In: Neurocomputing. Science Direct, pp 302–312

[Marquez Casillas ES et al., 2021] Marquez Casillas ES, Osuna-Enciso V (2021) Architecture Optimization of Convolutional Neural Networks by Micro Genetic Algorithms. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 149–167

[Masadeh R et al., 2019] Masadeh R, Mahafzah BA, Sharieh A (2019) Sea Lion Optimization Algorithm. In: International Journal of Advanced Computer Science and Applications (IJACSA). The Science and Information Organization

[Milner R, 2020] Milner R (2020) The Unexpected Relationship Between Wolves And Ravens

[Mirjalili S et al., 2023] Mirjalili S, Gandomi AH (2023) Metaheuristic algorithms in network intrusion detection. In: Comprehensive Metaheuristics: Algorithms and Applications. Academic Press, pp 95–129

[Mirjalili S et al., 2016] Mirjalili S, Lewis A (2016) The Whale Optimization Algorithm. In: Advances in Engineering Software. ELSEVIER, pp 51–67

[Mirjalili S et al., 2014] Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. In: Advances in Engineering Software. ELSEVIER, pp 46–61

[Mitchell T, 1997] Mitchell T (1997) Machine Learning. McGraw-Hill 45.37, Burr Ridge, pp. 870–877

[Mitiku T et al., 2018] Mitiku T, Manshahia MS (2018) Neuro Fuzzy Inference Approach : A Survey. In: Engineering and Technology. pp 505–519

[Mohammadpour L et al., 2022] Mohammadpour L, Ling TC, Liew CS, Aryanfar AA (2022) A Survey of CNN-Based Network Intrusion Detection. MDPI

[Mohit J et al., 2019] Mohit J, Vijander S, Asha R (2019) A novel nature-inspired algorithm for optimization: Squirrel search algorithm. In: Swarm and Evolutionary Computation. ELSEVIER, pp 148–175

[Monicka JG et al., 2011] Monicka JG, Sekhar NOG, Kumar KR (2011) Performance Evaluation of Membership Functions on Fuzzy Logic Controlled AC Voltage Controller for Speed Control of Induction Motor Drive. In: International Journal of Computer Applications. pp 8–12

[Monteiro A et al., 2018] Monteiro A, De Oliveira M, De Oliveira R, Da Silva T (2018) Embedded application of convolutional neural networks on Raspberry Pi for SHM. In: Electronics Letters. IET, pp 680–682

[Monteiro ACB et al., 2021] Monteiro ACB, Iano Y, França RP, Arthur R (2021) A Metaheuristic Algorithm for Classification of White Blood Cells in Healthcare Informatics. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 219–238

[Morales-Castañeda B et al., 2020] Morales-Castañeda B, Zaldívar D, Cuevas E, Fausto F, Rodríguez A (2020) A better balance in metaheuristic algorithms: Does it exist ? In: Swarm and Evolutionary Computation. Science Direct, p 100671

[Moshtaghi HR et al., 2021] Moshtaghi HR, Eshlaghy AT, Motadel MR (2021) A Comprehensive Review on Meta-Heuristic Algorithms and their Classification with Novel Approach. In: Journal of Applied Research on Industrial Engineering. pp 63–89

[Muazu AA et al., 2022] Muazu AA, Hashim AS, Sarlan A (2022) Review of Nature Inspired Metaheuristic Algorithm Selection for Combinatorial t-Way Testing. IEEE Access, pp 27404–27431

[Nadimi-Shahraki, M.H. et al., 2021] Nadimi-Shahraki, M.H., Banaie-Dezfouli, M., Zamani, H., Taghian, S., Mirjalili, S. (2021) B-MFO: A Binary Moth-Flame Optimization for Feature Selection from Medical Datasets. Computers

[Nash JC, 2000] Nash JC (2000) The (Dantzig) simplex method for linear programming. In: Computing in Science & Engineering. IEEE, pp 29–31

[Neu DA et al., 2022] Neu DA, Lahann J, Fettke PA (2022) A systematic literature review on state-of-the-art deep learning methods for process prediction. In: Artificial Intelligence Review. Springer, pp 801–827

[Osman IH et al., 1996] Osman IH, Laporte G (1996) Metaheuristics: A bibliography. In: Annals of Operations Research volume. Springer, pp 511–623

[Pan QK et al., 2008] Pan QK, Tasgetiren MF, Liang YC (2008) A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling. In: Computers & Operations Research. pp 2807–2839

[Panagant N et al., 2021] Panagant N, Pholdee N, Bureerat S (2021) A Comparative Study of Recent Multi-objective Metaheuristics for Solving Constrained Truss Optimisation Problems. Springer, pp 4031–4047

[Park D et al., 2021] Park D, Kim S, Kwon H, Shin D (2021) Host-Based Intrusion Detection Model Using Siamese Network. IEEE Xplore, pp 76614–76623

[Pedroza M et al., 2021] Pedroza M, Ramírez-Bello A, Becerra AG, Martínez FAF (2021) Machine Reading Comprehension (LSTM) Review (State of Art). In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 491–514

[Pereira I et al., 2013] Pereira I, Madureira A, De Moura Oliveira PB, Abraham A (2013) Tuning Meta-Heuristics Using Multi-agent Learning in a Scheduling System. In: Transactions on Computational Science XXI. Springer, pp 190–210

[Pourjavad E et al., 2019] Pourjavad E, Mayorga RV (2019) A comparative study and measuring performance of manufacturing systems with Mamdani fuzzy inference system. In: Journal of Intelligent Manufacturing. Springer, pp 1085–1097

[Qian L et al., 2020] Qian L, San-yang L, Xin-She Y (2020) Influence of Initialization on the Performance of Metaheuristic Optimizers. In: Applied Soft Computing. Computer Science

[Rabbouch B et al., 2023] Rabbouch B, Rabbouch H, Saâdaoui F, Mraihi R (2023) Foundations of combinatorial optimization, heuristics, and metaheuristics. In: Comprehensive Metaheuristics: Algorithms and Applications. Academic Press, pp 407–438

[Raidl GR et al., 2019] Raidl GR, Puchinger J, Blum C (2019) Metaheuristic Hybrids. In: Handbook of Metaheuristics. pp 385–417

[Ramos ICO et al., 2005] Ramos ICO, Goldbarg MC, Goldbarg EG, Net ADD (2005) Logistic regression for parameter tuning on an evolutionary algorithm. In: 2005 IEEE Congress on Evolutionary Computation. IEEE, pp 1061–1068

[Ramos-Michel A et al., 2021] Ramos-Michel A, Pérez-Cisneros M, Cuevas E, Zaldivar, D. (2021) Image Classification with Convolutional Neural Networks. In: Metaheuristics in Machine Learning: Theory and Applications. Springer, pp 445–473

[Rasku J et al., 2016] Rasku J, Kärkkäinen T, Musliu N (2016) Feature Extractors for Describing Vehicle Routing Problem Instances. In: 5th Student Conference on Operational Research (SCOR 2016). OASICS, Dagstuhl, Germany, pp 1–13

[Reddy PVS, 2021] Reddy PVS (2021) Generalized Fuzzy Logic with twofold fuzzy set: Learning through Neural Net and Application to Business Intelligence. In: 2021 International Conference on Fuzzy Theory and Its Applications (iFUZZY). IEEE, pp 1–5

[Riyaz B et al., 2020] Riyaz B, Ganapathy S (2020) A deep learning approach for effective intrusion detection in wireless networks using CNN. In: Soft Computing volume. Springer, pp 17265–17278

[Rodríguez MA et al., 2020] Rodríguez MA, Mezura ME, Villarreal CMG, Aldape PM (2020) Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem. In: Applied Soft Computing. Science Direct, p 106342

[Ross TJ, 2016] Ross TJ (2016) Fuzzy Logic with Engineering Applications, 4th edn. WILEY

[Sabri N et al., 2013] Sabri N, Aljunid SA, Salim MS, Badlishah RB, Kamaruddin R, Abd Malek MF (2013) Fuzzy Inference System: Short Review and Design. In: International review of automatic control. pp 441–449

[Sakr MM et al., 2019] Sakr MM, Tawfeeq MA, El-Sisi AB (2019) Filter Versus Wrapper Feature Selection for Network Intrusion Detection System. In: Ninth International Conference on Intelligent Computing and Information Systems. IEEE, pp 209–214

[Sarker IH, 2021] Sarker IH (2021) Machine Learning: Algorithms, Real-World Applications and Research Directions. In: SN Computer Science. Springer

[Sassi M, 2022] Sassi M (2022) Solving Feature Selection Problems Built on Population-based Metaheuristic Algorithms. In: Optimization and Machine Learning Optimization for Machine Learning and Machine Learning for Optimization. WILEY, pp 55–88

[Sassi M et al., 2023] Sassi M, Chelouah R (2023) HHO-EAS: A new metaheuristic bio-inspired of the win-win hunting synergy between the two predators crow and wolf. In: Artificial Intelligence Review. Springer

[Schiezaro M et al., 2013] Schiezaro M, Pedrini H (2013) Data feature selection based on Artificial Bee Colony algorithm. In: EURASIP Journal on Image and Video Processing

[Segera D et al., 2023] Segera D, Mbuthia M, Nyete A (2023) Metaheuristics for optimal feature selection in high-dimensional datasets. In: Comprehensive Metaheuristics: Algorithms and Applications. Academic Press, pp 237–267

[Sengupta N et al., 2020] Sengupta N, Sil J (2020) Intrusion Detection: A Data Mining Approach. In: Cognitive Intelligence and Robotics. Springer, pp 1–25

[Sharma M et al., 2021] Sharma M, Kaur PA (2021) Comprehensive Analysis of Nature-Inspired Meta-Heuristic Techniques for Feature Selection Problem. In: Archives of Computational Methods in Engineering. pp 1103–1127

[Sharma S et al., 2022] Sharma S, Kumar V (2022) A Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future. In: Archives of Computational Methods in Engineering. Springer, pp 5605–5633

[Shu PW et al., 2020] Shu PW, Chu QX, Mai JY (2020) Harris Hawks Optimization Algorithm for Waveguide Filter Designs. In: IEEE Asia-Pacific Microwave Conference (APMC). IEEE, pp 406–408

[Siarry P, 2014a] Siarry P (2014) Avant propos. In: Métaheuristiques, 1er edn. Eyrolles, Paris 5e, France, pp 1–2

[Siarry P, 2014b] Siarry P (2014) Un sujet ouvert : le choix d'une métaheuristique. In: Métaheuristiques. Eyrolles, Paris 5e, France, p 17

[Siarry P, 2014c] Siarry P (2014) Techniques de modélisation et comparaisons de méthodes. In: Métaheuristiques. Eyrolles, Paris 5e, France, pp 337–359

[Siarry P, 2016] Siarry P (2016) Metaheuristics. Springer

[Siarry P, 2014d] Siarry P (2014) Exploitation et exploration. In: Métaheuristiques. Eyrolles, Paris 5e, France, p 215

[Siarry P et al., 2002a] Siarry P, Collette Y (2002) Optimisation multiobjectif. Eyrolles, Paris 5e, France

[Siarry P et al., 2002b] Siarry P, Collette Y (2002) Introduction : optimisation multiobjectif et dominance. In: Optimisation multiobjectif. Eyrolles, Paris 5e, France, pp 15–40

[Singh P et al., 2021] Singh P, Choudhary SK (2021) Introduction: Optimization and Metaheuristics Algorithms. In: Metaheuristic and Evolutionary Computation: Algorithms and Applications. Springer, pp 3–32

[Sörensen K et al., 2018] Sörensen K, Sevaux M, Glover F (2018) A History of Metaheuristics. In: Handbook of Heuristics. Springer, pp 791–808

[Stahler D et al., 2002] Stahler D, Heinrich B, Smith D (2002) Common ravens, Corvus corax, preferentially associate with grey wolves, Canis lupus, as a foraging strategy in winter. In: Animal behaviour. pp 283–290

[STeele S et al., 1988] STeele S et al. (1988) An Analysis of Injection Molding by Taguchi Methods

[Talbi EG, 2009] Talbi EG (2009) Parameter Tuning. In: Metaheuristics: From Design to Implementation. WILEY, pp 54–57

[Talbi EG, 2013] Talbi EG (2013) A Unified Taxonomy of Hybrid Metaheuristics with Mathematical Programming, Constraint Programming and Machine Learning. In: Hybrid Metaheuristics. Springer, pp 3–76

[Tansel D et al., 2022] Tansel D, Ayça D, Hakan EK (2022) A comprehensive survey on recent metaheuristics for feature selection. In: Neurocomputing. Science Direct, pp 269–296

[Teghem J, 2012] Teghem J (2012) Les heuristiques et les métaheuristiques. In: Recherche opérationnelle. Ellipse, pp 369–434

[Thaher T et al., 2020] Thaher T, Heidari AA, Mafarja MM, Dong JS, Mirjalili S (2020) Binary Harris Hawks Optimizer for High-Dimensional, Low Sample Size Feature Selection. In: Evolutionary Machine Learning Techniques. Springer, pp 251–272

[Tilahun SL, 2017] Tilahun SL (2017) Prey predator hyperheuristic. In: Applied Soft Computing. ELSEVIER, pp 104–114

[Ting TO et al., 2006] Ting TO, Wong KP, Chung CY (2006) A Hybrid Genetic Algorithm/Particle Swarm Approach for Evaluation of Power Flow in Electric Network. In: Advances in Machine Learning and Cybernetics. Springer, pp 908–917

[Ting TO et al., 2015] Ting TO, Yang XS, Cheng S, Huang K (2015) Hybrid Metaheuristic Algorithms: Past, Present, and Future. In: Recent Advances in Swarm Intelligence and Evolutionary Computation. Springer, pp 71–83

[Toloo M et al., 2022a] Toloo M, Talatahari S, Rahimi I (2022) Multi-Objective Combinatorial Optimization Problems and Solution Methods, 1st edn. ELSEVIER

[Toloo M et al., 2022b] Toloo M, Talatahari S, Rahimi I (2022) The fundamentals and potential of heuristics and metaheuristics for multiobjective combinatorial optimization problems and solution methods. In: Multi-Objective Combinatorial Optimization Problems and Solution Methods. ELSEVIER, pp 10–12

[Too J et al., 2019] Too J, Abdullah AR, Saad NM (2019) A New Quadratic Binary Harris Hawk Optimization for Feature Selection. In: Electronics. MDPI

[Tressa M, 2021] Tressa M (2021) CNN Intrusion Detection for Threat Analysis of a Network. In: Turkish Journal of Computer and Mathematics Education. TURCOMAT

[Tufan E et al., 2021] Tufan E, Tezcan C, Acartürk C (2021) Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. In: IEEE Access. IEEE, pp 50078–50092

[University of Aegean, 2014] University of Aegean (2014) AWID dataset - Wireless security project

[University of the Aegean,] University of the Aegean Aegean Wi-Fi Intrusion Dataset (AWID)

[Vanaret C et al., 2020] Vanaret C, Gotteland JB, Durand N, Alliot JM (2020) Certified Global Minima for a Benchmark of Difficult Optimization Problems. ArXiv

[Verma KK et al., 2022] Verma KK, Singh BM, Dixit A (2022) A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system. In: International Journal of Information Technology volume. Springer, pp 397–410

[Verma RS et al., 2012] Verma RS, Kumar S (2012) DSAPSO : DNA sequence assembly using continuous particle swarm optimization with smallest position value rule. In: 2012 1st International Conference on Recent Advances in Information Technology (RAIT). IEEE, pp 410–415

[Vinod Chandra SS et al., 2022] Vinod Chandra SS, Anand HS (2022) Nature inspired meta heuristic algorithms for optimization problems. In: Computing. pp 251–269

[Wang GG et al., 2018] Wang GG, Gao XZ, Zenger K, Coelho LS (2018) A Novel Metaheuristic Algorithm inspired by Rhino Herd Behavior. In: Proceedings of the 9th EUROSIM & the 57th SIMS. pp 1026–1033

[Wolpert DH, 2021] Wolpert DH (2021) What Is Important About the No Free Lunch Theorems? In: Black Box Optimization, Machine Learning, and No-Free Lunch Theorems. Springer, pp 373–388

[Wolpert DH et al., 1996] Wolpert DH, Macready WG (1996) No Free Lunch Theorems for Search

[Wolpert DH et al., 1997] Wolpert DH, William GM (1997) No Free Lunch Theorems for Optimization. In: IEEE Transactions on Evolutionary Computation. IEEE, pp 67–82

[Wu G et al., 2017] Wu G, Mallipeddi R, Suganthan PN (2017) Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization

[Yang XS, 2010] Yang XS (2010) A New Metaheuristic Bat-Inspired Algorithm. In: Nature Inspired Cooperative Strategies for Optimization. Springer, Berlin, Heidelberg, pp 65–74

[Yao X, Liu Y, Lin G, 1999] Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. In: IEEE Transactions on Evolutionary Computation. IEEE, pp 82–102

[Yassa S, 2014] Yassa S (2014) Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing. Sciences et Technologies de l'Information et de la Communication

[Yu FJ et al., 2009] Yu FJ, Chen HK (2009) Economic-Statistical Design of X-bar Control Charts Using Taguchi Loss Functions. In: IFAC Proceedings Volumes. ELSEVIER, pp 1719–1723

[Yu X et al., 2022] Yu X, Haokai Z, Ferrantec N (2022) A self-adaptive multi-objective feature selection approach for classification problems. In: Integrated Computer-Aided Engineering. IOS Press, pp 3 – 21

[Zadeh LA, 1965] Zadeh LA (1965) Fuzzy sets. In: Information and Control. pp 338–353

[Zhang H et al., 2020] Zhang H, Yu T (2020) Taxonomy of Reinforcement Learning Algorithms. In: Deep Reinforcement Learning. pp 125–133

[Zhang S et al., 2020] Zhang S, Zhang S, Wang B, Habetler TG (2020) Deep Learning Algorithms for Bearing Fault Diagnostics: A Comprehensive Review. In: IEEE Access. IEEE, pp 29857–29881

[Zhang X et al., 2020a] Zhang X, Wang X, Chen H (2020) Improved GWO for large-scale function optimization and MLP optimization in cancer identification. In: Neural Computing and Applications. Springer, pp 1305–1325

[Zhang X et al., 2020b] Zhang X, Zhao K, Wang L, Wang Y, Niu Y (2020) An Improved Squirrel Search Algorithm With Reproductive Behavior. In: IEEE Access. IEEE, pp 101118–101132

[Zhang Y et al., 2020] Zhang Y, Zhou X, Shih PC (2020) Modified Harris Hawks Optimization Algorithm for Global Optimization Problems. In: Arabian Journal for Science and Engineering volume. Springer, pp 10949–10974

[Zhao J et al., 2002] Zhao J, Bose BK (2002) Evaluation of membership functions for fuzzy logic controlled induction motor drive. In: IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IEEE, pp 229–234