Universiteit Leiden
The Netherlands

# Opleiding Informatica

Evaluating Whale Optimization Algorithm:
Mathematical Rigor, Reproducibility and Performance

Tijn van Son

Supervisors:
Hao Wang & Haoran Yin

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

21/06/2024

**Abstract**

The Whale Optimization Algorithm (WOA) is a novel, nature-inspired meta-heuristic optimization algorithm that takes its inspiration from the bubble-net hunting strategy of humpback whales. This thesis will focus on describing WOA in a rigorous way and on the reproducibility of the original experimental results. In additions to this, WOA and the two variants ILWOA and WOA-GA will be tested with the benchmarking tool Iterative Optimization Heuristics Profiler (IOHprofiler).

# Contents

# 1 Introduction

## 1.1 Meta-heuristic Algorithms

Nature inspired meta-heuristic optimization algorithms are known for their ability to tackle a wide range of real-world optimization problems. One of the first nature-inspired meta-heuristic algorithms is the Genetic Algorithm (GA) [Hol92], proposed by John Holland in the 1960s. This algorithm is inspired by the process of evolution in nature. GA uses mutation, selection and crossover to find the best solutions that will form the next generation of solutions. Other important metaheuristic algorithms are Ant Colony Optimization (ACO), Simulated Annealing (SA) and Particle Swarm Optimization (PSO) by J. Kennedy and R. Ederhart that was introduced in 1995 [KE95]. Today there exist many more different kinds of nature inspired meta-heuristic algorithms, each with their own strengths and weaknesses. Their popularity is due to several reasons. One is their applicability to a wide range of real-world problems. For example PSO has been used in the design of communication networks that are often NP-hard problems [PPP+08], but is also used in engineering like urban planning for optimizing road networks [FLT+11]. The other reason is that such algorithms are easy to implement since commonly they are based on relative simple concepts that make the algorithm easy to implement and they can work on optimization problems with a search space of multiple dimensions.

## 1.2 Introduction of Whale Optimization Algorithm

The Whale Optimization Algorithm (WOA) is a new nature inspired optimization algorithm and is proposed in 2016 by Seyedali Mirjalili and Andrew Lewis [ML16]. This algorithm is inspired by the bubble-net feeding maneuver performed by humpback whales. The bubble-net hunting method is performed by a group of whales that work together to catch large groups of small fish or krill. Through communication the humpback whales coordinate their positions. At least one or more of the whales will then dive deeper under the prey. From under the school of fish, they start ascending and create a rising spiral of air bubbles that act as a barrier with the intention to confuse and trap the prey. The whale manipulates the size of the bubbles by shrinking them and thus trapping the prey in a smaller space. When the prey is in a concentrated enough space, another or multiple other whales start attacking the prey by swimming through the bubble and swallowing the school of fish or krill all at once. WOA mathematically models this spiral bubble-net feeding bubble in order to perform optimization in a search space.
WOA is one of many recent examples of an animal inspired optimization algorithm and since it's introduction, many variants and hybrids have been introduced to optimize it's efficiency and exploration/exploitation balance.

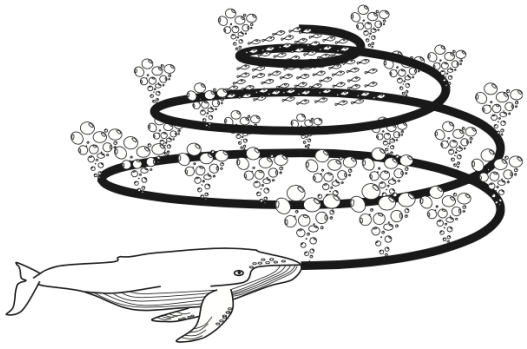


Figure 1: Schematic of Bubble-net feeding behavior of humpback whale as shown in original research paper [ML16]

## 1.3 Research Questions

In the past decades there has been a significant growth in the amount of presented nature-inspired metaheuristic algorithms. While more established models like PSO have been extensively studied and validated, there is concern that many newer metaheuristic algorithms lack this kind of validation [JYF+13]. Many of these newly inspired algorithms miss mathematical investigation of their key mechanics [SSS21]. It is also argued that it has become a trend to introduce new nature-inspired metaphors in metaheuristics [Sör15]. These metaphors can potentially hide the true underlying mathematical aspects of the algorithm. Many new introduced metaheuristics are given a new metaphor-inspired terminology, but in reality they are more or less a recombination of previously established algorithms. This could potentially mislead researchers about the nature of the techniques and cause for confusion within the academic community about the actual advancements of metaheuristics. The main purpose of this thesis is to study the mathematical aspect of WOA, to ensure that the findings of the original work are reproducible and to benchmark WOA as well as some of its variants. For this thesis two other variants will be benchmarked. These are Improved Lévy-flight Whale Optimization Algorithm (ILWOA) and the hybrid between WOA and GA, the WOA-GA algorithm. To conduct this research the following research questions are proposed:

1. How rigorous are the mathematical descriptions of WOA?
2. Is the work of the original research paper of WOA reproducible?
3. How do WOA, ILWOA and WOA-GA compare to other state-of-the-art metaheuristic algorithms and to each other?

WOA is inspired by the behavior of humpback whales. It is important to investigate the true nature of the algorithm. Even though the algorithm tries to mimic this hunting behavior, in the end the effectiveness of the optimization relies on the mathematical formulation of the algorithm. For researchers to understand how the algorithm works and to be able to reproduce the research, it is important that the formulation of the algorithm consists of rigorous math definitions. If the original paper lacks rigorous math definitions, other researchers may fail to recreate the same results accurately. Our first research question states: "How rigorous are the mathematical descriptions of WOA?"

To build upon this, we introduce the second research question: Is the work of the original research paper of WOA reproducible? In other words, are we able to repeat the experiment using their software and data and can we confirm their conclusion? Reproducibility is not only important for researchers trying to build upon the work of others, it also is a method of quality control. Being able to reproduce the research that is done, confirms that the algorithm performs as claimed in the paper and thus improves the reliability and credibility of the research.

For the final research question we will compare WOA to other state-of-the-art (SOTA) algorithms using IOHprofiler. IOHprofiler is a benchmarking tool provided by LIACS Natural Computing cluster [DWY+18]. In a recent study that systematically reviews WOA and its variant, it is stated there is a lack in study and comparison between WOA and its variants [NSZAVM23]. This makes it interesting to also benchmark other variants of WOA. For this reason our final research question will benchmark WOA, ILWOA and WOA-GA with he help of IOHprofiler.

### 1.4 Thesis overview

In Section 2, an overview will be given on some of the previous work on WOA. Since the original algorithm was proposed in 2016, many variants and hybrids have been introduced. The research behind ILWOA and WOA-GA will also be discussed here. In Section 3 the research questions will be explored. We will first define the mechanics of WOA, then investigate the mathematical rigor of WOA. After this, we will focus on the reproducibility of the research and benchmark WOA, ILWOA and WOA-GA with IOHprofiler. Finally in Section 4, conclusions will be drawn from exploring the research questions and some possible future research will be discussed.

## 2 Related Work

In this section, we will dive into research related to WOA. We will first focus on a research paper that made a big review across all variants of WOA using PRISMA methodology. Then will we continue with exploring two research papers that proposed ILWOA and WOA-GA.

### 2.1 A Systematic Review of the Whale Optimization Algorithm

In May 2023, a paper was published that systematically reviewed the development of WOA [NSZAVM23]. The paper argues that while WOA is an easy to implement algorithm, it still faces many issues like getting stuck at local optima too early. Since its introduction many variants and hybrids have been implemented. The authors selected eligible papers using the so called PRISMA methodology as a review method.

PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) is a methodology that provides a structured approach to performing systematic reviews. The methodology consists of three main stages: the identification stage, the evaluation stage and the reporting stage. In the identification stage, keywords and alternative synonyms of those keywords are defined as target search problems to extract documents from different databases. With their defined keywords, the

Google Scholar database was searched with a restriction from 2019 to the end of March 2023. Within this time, 2983 documents were identified. In the next stage, the evaluation stage, the documents were checked. Duplicates and documents, that were non-academic or disreputable journals, were removed. They continued with this selection process by first selecting documents based on abstract and title. After this, they selected the remaining articles by reading the full text. This ended in a total of 116 selected research papers that were classified as eligible papers. Finally, in the reporting stage, the selected papers were reviewed. In this review, all variants that were introduced in the selected papers, were briefly covered. They reviewed the variants in two categories: improved versions of WOA and WOA hybrids. They also classified these variants in continuous, single-objective and multi/many-objective categories. The paper quickly covers ILWOA and WOA-GA as well.

### 2.2 An improved Lévy based Whale Optimization algorithm

In January 2018, a new research paper was published that introduced ILWOA, a more advanced variant of WOA. This improved version was used to optimize the virtual machine (VM) placement in a cloud computing environment [ABAFS19]. The main objective of the VM placement problem is to minimize the number of running physical machines or hosts in cloud data centers. The improved version has Lévy-flight distribution incorporated to enhance the balance from exploration to exploitation of the algorithm. Their approach was to aim on optimizing the allocation of VMs based on bandwidth availability rather than focusing on processor or storage capabilities. The algorithm was tested with the Cloudsim toolkit across 25 different datasets. Here ILWOA showed the best performance compared to other algorithms like WOA, PSO and GA.

### 2.3 A Hybrid WOA-GA Algorithm

In 2022, the hybrid WOA-GA was introduced for time-jerk optimal trajectory planning of industrial robots[WWB22]. They focused on optimizing the time efficiency and the smoothness of a robot's movement. WOA-GA was proposed to solve this trajectory planning problem within kinematic limits such as jerk, acceleration and velocity. The authors argue that WOA was used for its strong exploration and GA for its effective exploitation. The results of their experiment showed that WOA-GA provided the best solutions in their test environment, that was focused on optimizing the time-jerk optimal trajectory problem.

## 3 Methodology

In this section, the research questions will be explored. We will first search for mathematical rigor in the original research paper of WOA. Secondly, we will focus on the reproducibility of the original experiment and try to reproduce some of these results. Finally, we will benchmark WOA, ILWOA and WOA-GA with IOHprofiler and compare their performance to other SOTA algorithms.

### 3.1 How to assess the rigor of the mathematical descriptions of WOA

Rigor is a term that is used to describe a strictness in how we approach mathematics. A traditional example of rigor in an argument is when the reasoning of that argument doesn't contain any

gaps[Kit81]. This means that for a finite sequence of statements, each statement that occurs in the argument either should support a final conclusion or should be build from previous statements. There exist many features that are fundamental for providing rigor in mathematics. Think for example about simplicity of statements and consistency in the use of symbols. It is also important that the algorithm can be understood based on the documentation provided by the research paper. This ensures that the work is reproducible. Mathematical rigor can also be about evaluating the meaning of the math. Does the math describe what it tries to do? In the case of WOA, is it indeed trying to mimic the bubble-net hunting behavior of the humpback whales? There are a lot of things that can be looked at in order to investigate whether the math in the original paper has rigorous math definitions. This is more complex than a simple yes or no question. We need to define a set of criteria in order to be able to answer this question. To answer our first research question, the following evaluation criteria are set up:

- **Completeness:** The math covers all necessary aspects of the algorithm. All parameters are explained and the math contains no gaps.
- **Consistency:** All statements and parameters are consistent within the paper. There are no contradictions or ambiguities present.
- **Meaning:** The mathematical descriptions of WOA are inspired by the bubble-net hunting behavior of the humpback whales.

We will first analyze how the paper describes the math and then how the paper proposes these definitions with the help of these evaluation criteria.

## 3.2 Mathematical model of the algorithm

The paper starts with explaining the mathematical model for the three main stages of the algorithm. These three phases are called the encircling prey phase, the spiral bubble-net feeding maneuver and the search for prey phase. After these phases are discussed, the paper continues with proposing the whole algorithm of WOA.

### 3.2.1 Encircling prey

In optimization problems, the exact position of the optimal solution in the search space is unknown to the algorithm. This is unlike in nature where the whales do know where the target prey is. Therefor, WOA assumes that the current best solution is the target prey or is close to the optimum. In the encircling prey phase, the search agents will update their position towards the best search agent. This is the search agent with the best current solution. This behavior is described in the paper by the following two equations:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}^*(t) - \vec{X}(t) \right| \tag{1}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D} \tag{2}$$

Where $t$ represents the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}^*$ is the position vector of the current best solution, $\vec{X}$ is the vector position of the $i$th search agent, the symbol $|\ |$ is the

absolute value and $\cdot$ in Equation 1 represents an element by element multiplication. The paper states that $X^*$ should be updated on each iteration, but this is not further defined mathematically. The coefficient vectors $\vec{A}$ and $\vec{C}$ are defined in the following two equations:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{4}$$

where $\vec{a}$ linearly decreases from 2 to 0 and $\vec{r}$ is a random value between 0 and 1. Adjusting the values of both vectors $\vec{A}$ and $\vec{C}$, makes it possible for the search agents to reach multiple places. Parameter $\vec{r}$ helps the search agent move into any direction and decreasing the value of $\vec{a}$ limits the reach of how far the search agent can update. This ensures a balance from exploration to exploitation.

### 3.2.2 Bubble-net attacking method

The bubble-net attacking method tries to mathematically model the bubble-net behavior of humpback whales. This phase acts together with the encircling prey phase as the exploitation phase. The bubble-net attacking method has two approaches:

1. **Shrinking encircling mechanism**: This behavior is simulated by the decreasing value of $\vec{a}$ in Equation 3 and happens in reality during the encircling prey phase. The value of $\vec{A}$ is randomly set by $0 \leq A \leq a$.

2. **Spiral updating position**: This approach between the search agent and the best current solution. Then a spiral equation is used between these two positions to mimic the helix-shaped movement of humpback whales.

The equation of this spiral updating position approach is as follows:

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{5}$$

where $\vec{D'} = |\vec{X}^*(t) - \vec{X}^*|$. This indicates the distance of the $i$th whale to the current best solution. $b$ is a constant for defining the shape of the logarithmic spiral, $l$ is a random number in the interval [-1, 1] and $\cdot$ again in this equation is an element-by-element multiplication.

The paper then explains that the humpback whales swim around the prey within a shrinking circle and along a spiral-shaped path simultaneously. In order to be able to model this, a probability of 50% is used to choose between the two approaches. This results is the following equation below:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \\ \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \tag{6}$$

### 3.2.3 Search for prey

The final phase is the search for prey phase. This phase makes it possibles for the whales to search for prey randomly within the search space and acts as the exploration phase of the algorithm. The absolute value of the coefficient vector $\vec{A}$ is a condition that determines whether the $i$th whale enters the encircling prey phase or the search for prey phase. When $|\vec{A}| > 1$ the whale enters the search for prey phase and updates its position according to a randomly chosen search agent, unlike to the current best search agent in the encircling prey phase. The mathematical model is as follows:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_{rand} - \vec{X} \right|, \tag{7}$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A} \cdot \vec{D}. \tag{8}$$

where $\overrightarrow{X_{rand}}$ is a random position vector that represent a random chosen whale from the population. This phase tries to prevent premature convergence on a local optima.

### 3.2.4 Pseudocode

The pseudocode of the algorithm as proposed in the papers is as follows:

**Algorithm 1** Whale Optimization Algorithm (WOA)

Initialize the whale population $\vec{X}_i$ $(i = 1, 2, ..., n)$
Initialize $a$, $A$, $C$, $l$, and $p$
Calculate the fitness of each search agent
$\vec{X}^*$ = the best search agent
**while** ($t <$ maximum number of iterations) **do**
    **for** each search agent **do**
        **if** $p < 0.5$ **then**
            **if** $|A| < 1$ **then**
                Update the position of the current search agent by Equation (2)
            **else**
                Select a random search agent $\vec{X}_{rand}$
                Update the position of the current search agent by Equation (8)
            **end if**
        **else**
            Update the position of the current search agent by Equation (5)
        **end if**
    **end for**
    Calculate the fitness of each search agent
    Update $\vec{X}^*$
    Update $a$, $A$, $C$, and $l$
    $t = t + 1$
**end while**
**return** $\vec{X}^*$

The algorithm starts with initializing a random set of solutions for the population of whales. Parameter $a$ is decreased from 2 to 0 to balance from being more exploration oriented to more exploitation oriented. Parameter $p$ is a value between 0 and 1. If $p \geq 0.5$, the search agent will perform the spiral updating position. When $p < 0.5$, then $|\vec{A}|$ determines if the encircling prey phase or the search for prey phase will take place. The algorithm is terminated once the maximum number of iterations is reached.

## 3.3 Evaluating mathematical rigor in WOA

The paper defines the math definitions as explained in Section 3.2. Based on these definitions and with the help of our evaluation criteria, we can determine if the paper consists of rigorous math definitions.

### 3.3.1 Completeness

For completeness we evaluate if the math covers all necessary aspects of the algorithm. This means for WOA that we will need to look at every detail in the algorithm. We will first analyze if all parameters are explained and mathematically defined. After this, the completeness in statements will be discussed.

**Completeness in parameters**: The pseudocode shows that in every iteration, the vector coefficients $\vec{A}$ and $\vec{C}$ and parameters $a$, $l$ and $p$ need to be updated. The vector coefficients are calculated with Equations 3 & 4. Here a new parameter $\vec{r}$ is defined as a random vector. Parameters $\vec{r}$, $a$, $l$ and $p$ are all explained within the paper.

The parameters used to describe the algorithm that don't show in the pseudocode, but are defined in the research paper are $\vec{X}$, $\vec{X}^*$, $\overrightarrow{X_{rand}}$, $\vec{D}$, $\vec{D}'$, $t$. Next to this also hyperparameters $b$, the number of iterations $t$ and the population size are discussed. For the encircling prey phase $\vec{D}$ is calculated by Equation 1 and for the search for prey phase this distance vector is calculated by Equation 7. The difference here is the use of the position vectors $\vec{X}^*$ (the current best search agent) and $\overrightarrow{X_{rand}}$ (a random search agent).

All parameters seem to be defined correctly. However, there is one missing parameter that is required to be able to write the algorithm and is not defined by the paper. This parameter is the so called *a_step* parameter. Parameter *a_step* is a constant value that is subtracted from $a$ on each iteration. It is mentioned that $a$ decreases on each iteration, but this constant is never defined in the mathematical formulation of the algorithm. This parameter is still crucial because adjusting the value of *a_step*, determines how quickly the algorithm will balance from being exploration oriented to exploitation oriented. Tweaking this parameter can change the performance of the algorithm. Overall, all coefficient vectors are mathematically defined by providing equations on how to calculate them. Except for *a_step*, all other parameters are defined with a symbol and are explained.

**Completeness in statements**: In the research paper, all three stages of the algorithm are defined with equations that show the behavior of the search agents. Equations 2, 5 and 8 represent these stages and show what possible paths can be taken for $\vec{X}(t+1)$

The encircling prey phase is described by Equation 2. Equations 1, 3 and 4 are statements that help build Equation 2 by showing how the distance vector $\vec{D}$ and coefficient vectors $\vec{A}$ and $\vec{C}$ are calculated.

The bubble-net attacking method is split into two approaches that each have a 50% probability of occurring. The shrinking encircling mechanism is the simulated decreasing behavior in Equation 3 and essentially a specific part of the encircling phase. The second approach, the spiral updating position is described by Equation 5. Together, the encircling phase and the bubble-net attacking method are summarized in Equation 6 and this acts as the exploitation part of the algorithm.

Finally, the search for prey phase is described by the 7 and 8, where the search agents update their position based on a randomly selected agent.

The paper tries to provide a clear structure by dividing the algorithm into three phases. Each phase is mathematically modelled by one equation that is being built from other equations. However, one improvement might be to formulate all three phases into one equation. The paper already provides an equation that includes the encircling prey phase and the bubble-net attacking method to summarize the exploitation phase, but this equation is incomplete. This equation makes it look like that, if $p < 0.5$, the encircling prey phase is always entered and the shrinking encircling mechanism is performed. By looking at the pseudocode, we can see that this is not the case. Only on the condition $|\vec{A}| < 1$, this is true. In order to provide more completeness in the mathematical formulation of the algorithm, the paper could have concluded the three phases by defining the following equation:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \text{ and } |\vec{A}| < 1 \\ \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \\ \vec{X}_{rand} - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \text{ and } |\vec{A}| \geq 1 \end{cases} \tag{9}$$

In conclusion, the completeness of the mathematical statements in the paper is strong but not without gaps. The provided equations cover all the necessary parts of the algorithm, but there isn't a mathematical model that summarizes all main parts of the algorithm. This also makes Equation 6 a bit more confusion, as this function is formulated as if it models the first two phases, but it fails to mention the condition $|A| < 1$.

### 3.3.2 Consistency

In order for WOA to be formulated in a consistent way, there shouldn't be any inconsistencies in the use of all parameters and statements within the paper. There also shouldn't be any contradictions or ambiguities present. Overall the paper seems to be consistent in the use of symbols and statements, but there are some small inconsistencies present.
First of all, the paper is sometimes inconsistent in using vector symbols. One example is when the authors explain the position vectors after presenting the first two equations as follows:

*"... $X^*$ is the position vector of the best solution obtained so far, $\vec{X}$ is the position vector ..."*

We can see here that that $\vec{X}^*$ is mentioned as $X^*$. The parameter $\vec{X}^*$ is not presented with the $\rightarrow$ symbol that is used to represent a vector. Parameter $a$ is also sometimes referred to as $\vec{a}$, while this is essentially the same parameter. It is unclear why both of these symbols are used to describe the same parameter. It seems that it has to do in what context parameter $a$ is used. It can be assumed that the choice for the use of symbol $\vec{a}$ is used in Equation 3, because this parameter here is part of calculating a coefficient vector. This does not change the fact that parameter $a$ itself, is a scalar value.

Parameter $\vec{D}$ is described by two different functions. These are Equations 1 and 7. If the paper had presented the mathematical model as proposed in Equation 9, the use of parameter $\vec{D}$ for both phases, would be ambiguous because they represent different calculations.

One mistake made by the authors in their pseudocode, was referring to Equation 1 instead of Equation 2 that correctly resembles the encircling prey phase. This mistake has been corrected in the pseudocode provided in this thesis to prevent confusion.

The inconsistencies in the math of the research paper are small mistakes that could cause for confusion, when reading the paper for the first time. Although these mistakes are not crucial, they take a part in understandability and reproducibility of the research.

### 3.3.3 Meaning

For nature-inspired algorithms it is not only important that the math that describes the algorithm, is complete and consistent. In context of metaheuristic algorithms, mathematical rigor should also ensure that the math is really inspired by the natural behavior of humpback whales.

The actual behavior of the bubble-net feeding maneuver that the whales perform is said to be mathematically modelled by the bubble-net attacking method. There are two different behaviors that the whales perform during the bubble-net feeding maneuver. The first behavior is the maneuver where the whales dive under the prey and create an upwards spiral movement. This brings a cylindrical bubble-net that traps and confuses the prey. This spiral path becomes smaller, so that the the trapped prey gets more concentrated. This is modelled by the spiral updating position approach, see Equation 5. The second behavior is performed by other whales, who reduce their distance to their prey by tightening their circular swimming pattern around it. This behavior is mathematically modelled by the shrinking encircling mechanism, see Equation 3.

On March 2022 a research paper was published that criticises multiple bestiary inspired metaheuristic algorithms for being constructed from components that are equivalent proposed from already well-established techniques [CVDS23]. It criticises WOA for being a recombination of the mathematical models of the Grey Wolf Optimizer algorithm (GWO) and the Moth-Flame Algorithm (MFA). Both of these algorithms are proposed by the same authors of WOA. The paper shows that the math for the encircling prey phase is directly taken over from the encircling prey phase in GWO. A part of the mathematical formulation of the spiral updating position has been directly taken over from MFA. This mathematical part is $e^{bl} \cdot \cos(2\pi l)$. The paper also argues that the mathematical model of GWO is a variant of SPSO-2011. MFA is stated to be inspired by ePSO. The research paper concludes that WOA is fundamentally based on principles from PSO

and that it therefor lacks in mathematical rigor, because it repackages existing algorithms under a new metaphor without introducing fundamental mathematically distinct mechanisms.

## 3.4 Reproducibility in WOA

Reproducibility is an elusive concept [Ple18]. This means that there doesn't exist one clear definition for this concept. For this thesis with reproducibility, we refer to the ability to repeat a study's research and obtain the same results. To reproduce a research is essentially to recreate the same results under slightly different conditions, done by a different team. To review the reproducibility of the original research paper, the following three criteria will be considered:

- **Implementation details:** Does the research paper include a detailed algorithmic description complete with mathematical definitions and pseudocode? Is the actual source code available?
- **Experimental setup:** Are the authors transparent in their experimental setup? Are the configurations of all parameters specified? Does the paper clearly mention the benchmark functions against which the algorithm is tested?
- **Recreating results:** Does the paper give enough information about their results? Can those results be reproduced?

### 3.4.1 Implementation Details

The mathematical formulation of the algorithm has already been reviewed in Section 3.2. In Section 3.3 is concluded that the research paper does provide a detailed algorithmic description and provides pseudo code as well. The source code is also available online and can be found at https://seyedalimirjalili.com/woa.

### 3.4.2 Experimental setup

In the original research paper, the algorithm was tested by solving 29 different mathematical optimization problems. In addition to this, WOA was also tested with six constrained engineering design problems. These design problems are optimizing a tension/compression spring, a welded beam, a pressure vessel, a 15-bar truss, a 25-bar truss, and a 52-bar truss.

Of the test functions, the first 23 problems are classical benchmark functions. The other 6 test problems are more complex test functions where the search space gets shifted, rotated and expanded in combined variations. Of these first 23 functions, F1 to F7 are unimodal functions. Unimodal functions have a single local optimum. The algorithm therefor automatically searches towards the global optimum. F8 to F13 are multimodal functions and F14 to F23 are multimodal functions of a fixed-dimension. Multimodal functions contain multiple local optima, making it more challenging to find the global optima. The fixed-dimension multimodal functions contain a subset of multimodal functions of a fixed dimension. The mathematical descriptions of all test functions are provided in the research paper. An example of how these are defined can be seen in Figure 3. The mathematical descriptions of the functions F1 to F23 are given here. The mathematical definitions of F24 to F26 can be found in the research paper [ML16].

Description of unimodal benchmark functions.

| Function | V_no | Range | $f_{min}$ |
|---|---|---|---|
| $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 30 | [−100,100] | 0 |
| $F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ | 30 | [−10,10] | 0 |
| $F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$ | 30 | [−100,100] | 0 |
| $F_4(x) = \max_i \{\lvert x_i \rvert,\ 1 \le i \le n\}$ | 30 | [−100,100] | 0 |
| $F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | 30 | [−30,30] | 0 |
| $F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$ | 30 | [−100,100] | 0 |
| $F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$ | 30 | [−1.28,1.28] | 0 |

Description of multimodal benchmark functions.

| Function | V_no | Range | $f_{min}$ |
|---|---|---|---|
| $F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$ | 30 | [−500,500] | $-418.9829 \times 5$ |
| $F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$ | 30 | [−5.12,5.12] | 0 |
| $F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$ | 30 | [−32,32] | 0 |
| $F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$ | 30 | [−600,600] | 0 |
| $F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ | 30 | [−50,50] | 0 |
| $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ | | | |
| $F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ | 30 | [−50,50] | 0 |

Description of fixed-dimenstion multimodal benchmark functions.

| Function | V_no | Range | $f_{min}$ |
|---|---|---|---|
| $F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}$ | 2 | [−65,65] | 1 |
| $F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$ | 4 | [−5,5] | 0.00030 |
| $F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ | 2 | [−5,5] | −1.0316 |
| $F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$ | 2 | [−5,5] | 0.398 |
| $F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$ | 2 | [−2,2] | 3 |
| $F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$ | 3 | [1,3] | −3.86 |
| $F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$ | 6 | [0,1] | −3.32 |
| $F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$ | 4 | [0,10] | −10.1532 |
| $F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$ | 4 | [0,10] | −10.4028 |
| $F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$ | 4 | [0,10] | −10.5363 |

Figure 2: Description of benchmark functions F1 to F23 used to test WOA in the original research paper

In these tables, V_no indicates the number of design variables. For the benchmark functions F1 to F13 there is tested with V_no = 30. This means that the search agents are operating in a 30-dimensional search space. The range refers to the constraints placed on the design variables. These constraints set the limits that each design variable must satisfy. Finally $f_{min}$ indicates the minimum value of the benchmark function. It resembles the fitness of the global optimum. The paper states that for testing, all the algorithms have been tested on a population size of 30 with a maximum amount of 500 iterations. In their mathematical formulation they already defined that parameter $\vec{a}$ should be decreasing from 2 to 0. Each function has been performed with a repetition of 30. The paper doesn't specify what value they have used for parameter $b$ and for $a$.

Each of the six classical engineering problems has their own objective, design variables and constraints. The authors mention that, because these engineering problems each have different constraints, a constraint handling method is needed. For testing on the classical engineering problems, they decided to implement the death penalty constraint method. This method simply removes solutions in the pool that have any of their design variables out of the constraints. For each engineering problem the design variables are given and the problem function is formulated. For example, the tension/compression spring design has the three design variables: wire diameter ($d$), mean coil diameter ($D$) and the number of active coils ($N$). The optimization problem is formulated as follows:

$$
\begin{aligned}
\text{Consider} \quad & \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N], \\
\text{Minimize} \quad & f(\vec{x}) = (x_3 + 2)x_2x_1^2 \\
\text{Subject to} \quad & g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0, \\
& g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} \leq 0, \\
& g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \\
& g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0, \\
\text{Variable range} \quad & 0.05 \leq x_1 \leq 2.00, \\
& 0.25 \leq x_2 \leq 1.30, \\
& 2.00 \leq x_3 \leq 15.0
\end{aligned}
\tag{5.1}
$$

Figure 3: Description of tension/compression spring engineering problem

The other five engineering problems are formulated in the same way in the research paper and therefor by following its math, the benchmark functions can be recreated.

### 3.4.3 Recreating results

The authors compared WOA to other metaheuristic algorithms. These are Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE) and Fast Evolutionary Programming (FEP). They presented the mean and standard deviation of the results with a repetition of 30 runs. The results for F1 to F23 are presented in Figure 4.

Comparison of optimization results obtained for the unimodal, multimodal, and fixed-dimension multimodal benchmark functions.

| F | WOA | | PSO | | GSA | | DE | | FEP | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *ave* | *std* | *ave* | *std* | *ave* | *std* | *ave* | *std* | *ave* | *std* |
| F1 | 1.41E−30 | 4.91E−30 | 0.000136 | 0.000202 | 2.53E−16 | 9.67E−17 | 8.2E−14 | 5.9E−14 | 0.00057 | 0.00013 |
| F2 | 1.06E−21 | 2.39E−21 | 0.042144 | 0.045421 | 0.055655 | 0.194074 | 1.5E−09 | 9.9E−10 | 0.0081 | 0.00077 |
| F3 | 5.39E−07 | 2.93E−06 | 70.12562 | 22.11924 | 896.5347 | 318.9559 | 6.8E−11 | 7.4E−11 | 0.016 | 0.014 |
| F4 | 0.072581 | 0.39747 | 1.086481 | 0.317039 | 7.35487 | 1.741452 | 0 | 0 | 0.3 | 0.5 |
| F5 | 27.86558 | 0.763626 | 96.71832 | 60.11559 | 67.54309 | 62.22534 | 0 | 0 | 5.06 | 5.87 |
| F6 | 3.116266 | 0.532429 | 0.000102 | 8.28E−05 | 2.5E−16 | 1.74E−16 | 0 | 0 | 0 | 0 |
| F7 | 0.001425 | 0.001149 | 0.122854 | 0.044957 | 0.089441 | 0.04339 | 0.00463 | 0.0012 | 0.1415 | 0.3522 |
| F8 | −5080.76 | 695.7968 | −4841.29 | 1152.814 | −2821.07 | 493.0375 | −11080.1 | 574.7 | −12554.5 | 52.6 |
| F9 | 0 | 0 | 46.70423 | 11.62938 | 25.96841 | 7.470068 | 69.2 | 38.8 | 0.046 | 0.012 |
| F10 | 7.4043 | 9.897572 | 0.276015 | 0.50901 | 0.062087 | 0.23628 | 9.7E−08 | 4.2E−08 | 0.018 | 0.0021 |
| F11 | 0.000289 | 0.001586 | 0.009215 | 0.007724 | 27.70154 | 5.040343 | 0 | 0 | 0.016 | 0.022 |
| F12 | 0.339676 | 0.214864 | 0.006917 | 0.026301 | 1.799617 | 0.95114 | 7.9E−15 | 8E−15 | 9.2E−06 | 3.6E−06 |
| F13 | 1.889015 | 0.266088 | 0.006675 | 0.008907 | 8.899084 | 7.126241 | 5.1E−14 | 4.8E−14 | 0.00016 | 0.000073 |
| F14 | 2.111973 | 2.498594 | 3.627168 | 2.560828 | 5.859838 | 3.831299 | 0.998004 | 3.3E−16 | 1.22 | 0.56 |
| F15 | 0.000572 | 0.000324 | 0.000577 | 0.000222 | 0.003673 | 0.001647 | 4.5E−14 | 0.00033 | 0.0005 | 0.00032 |
| F16 | −1.03163 | 4.2E−07 | −1.03163 | 6.25E−16 | −1.03163 | 4.88E−16 | −1.03163 | 3.1E−13 | −1.03 | 4.9E−07 |
| F17 | 0.397914 | 2.7E−05 | 0.397887 | 0 | 0.397887 | 0 | 0.397887 | 9.9E−09 | 0.398 | 1.5E−07 |
| F18 | 3 | 4.22E−15 | 3 | 1.33E−15 | 3 | 4.17E−15 | 3 | 2E−15 | 3.02 | 0.11 |
| F19 | −3.85616 | 0.002706 | −3.86278 | 2.58E−15 | −3.86278 | 2.29E−15 | N/A | N/A | −3.86 | 0.000014 |
| F20 | −2.98105 | 0.376653 | −3.26634 | 0.060516 | −3.31778 | 0.023081 | N/A | N/A | −3.27 | 0.059 |
| F21 | −7.04918 | 3.629551 | −6.8651 | 3.019644 | −5.95512 | 3.737079 | −10.1532 | 0.0000025 | −5.52 | 1.59 |
| F22 | −8.18178 | 3.829202 | −8.45653 | 3.087094 | −9.68447 | 2.014088 | −10.4029 | 3.9E−07 | −5.53 | 2.12 |
| F23 | −9.34238 | 2.414737 | −9.95291 | 1.782786 | −10.5364 | 2.6E−15 | −10.5364 | 1.9E−07 | −6.57 | 3.14 |

Figure 4: Results of WOA compared to other algorithms in the original research paper

To recreate the results, an experiment is setup to test the first 23 out of the 29 benchmark functions and to see if the same results can be recreated. As a starting point the source code was taken from https://seyedalimirjalili.com/woa. This is a website created by Seyedali Mirjalili, one of the two authors of the original research paper. The algorithm is provided in MATLAB, Python, C++, R and Java. The source code that is written in MATLAB, contains the experimental setup for the first 23 benchmark functions used in the experiment of the research paper. The source code written in Python will be used for Section 3.5. Here this version of the source code will be discussed. The MATLAB source code also contains the exact parameter setup as described in the paper. All 23 benchmark functions were tested on a 1,4 GHz Quad-Core Intel Core i5 processor with 16GB of RAM. Each function has been repeated 30 times. Parameter $b$ has been used with a value of 1. The results of this experiment are provided in the table below:

| Function | Results of original research | | Results of reproducing original research | |
|---|---|---|---|---|
| | Average (mean) | Standard Deviation | Average (mean) | Standard Deviation |
| F1 | $1.41 \times 10^{-30}$ | $4.91 \times 10^{-30}$ | $6.83 \times 10^{-74}$ | $3.15 \times 10^{-73}$ |
| F2 | $1.06 \times 10^{-21}$ | $2.39 \times 10^{-21}$ | $2.74 \times 10^{-51}$ | $9.33 \times 10^{-51}$ |
| F3 | $5.39 \times 10^{-07}$ | $2.93 \times 10^{-06}$ | $49,290$ | $12,884$ |
| F4 | 0.072581 | 0.39747 | 57.38 | 25.49 |
| F5 | 27.86558 | 0.763626 | 28.11 | 0.40 |
| F6 | 3.116266 | 0.532429 | 0.379 | 0.197 |
| F7 | 0.001425 | 0.001149 | 0.003644 | 0.003245 |
| F8 | $-5080.76$ | 695.7968 | $-9723.33$ | 1845.80 |
| F9 | 0 | 0 | 0 | 0 |
| F10 | 7.4043 | 9.897572 | $3.724 \times 10^{-15}$ | $2.593 \times 10^{-15}$ |
| F11 | 0.000289 | 0.001586 | 0 | 0 |
| F12 | 0.339676 | 0.214864 | 0.01674 | 0.008490 |
| F13 | 1.889015 | 0.266088 | 0.4903 | 0.3095 |
| F14 | 2.111973 | 2.498594 | 3.3273 | 3.5073 |
| F15 | 0.000572 | 0.000324 | 0.000813 | 0.0005692 |
| F16 | $-1.03163$ | $4.2 \times 10^{-7}$ | $-1.0316$ | 0 |
| F17 | 0.397914 | $2.7 \times 10^{-05}$ | 0.3978 | $2.039 \times 10^{-5}$ |
| F18 | 3 | $4.22 \times 10^{-15}$ | 3 | $1.33 \times 10^{-4}$ |
| F19 | $-3.85616$ | 0.002706 | $-3.8553$ | 0.01221 |
| F20 | $-2.98105$ | 0.376653 | $-3.2407$ | 0.09432 |
| F21 | $-7.04918$ | 3.629551 | $-8.6096$ | 2.568 |
| F22 | $-8.18178$ | 3.829202 | $-6.6344$ | 3.4449 |
| F23 | $-9.34238$ | 2.414737 | $-6.6758$ | 3.6408 |

Table 1: Comparison of Original and Recreated Results for WOA with Benchmark Functions F1 to F23

Table 1 shows on the left side the results of the original research and on the right side the results that have been obtained by trying to reproduce the same solutions. There are a few notable observations that can be made from looking at the results. First of all, the results for the unimodal functions F1 to F7 seem to have the biggest difference in performance compared to the original experiment. Functions F1 and F2 have a far better result, but F3 has a worse performance. For the multimodal benchmark functions F8 to F13 that are tested with a dimensionality of 30, the difference in performance is less significant. F9 does in both experiments always seem to find the global optimum. Only for F10 there is a big difference in performance. One remarkable observation is that F11 in the original experiment doesn't find the global optimum, but it does in the reproduced results. For the fixed-dimension multimodal functions F14 to F23, the difference in performance is least significant. The results still differ, but compared to the difference in the unimodal benchmark functions, this is almost insignificant. Finally, what is noticeable as well, is that even though the results of the unimodal functions have the biggest difference in performance, the scale of how the results deviate around the mean of the unimodal functions, is the same in both experiments.

Unimodal functions have a single global optimum. The path to the global optimum is more straight forward, because it misses the challenge of having to overcome a local optima. This causes the performance to primarily reflect the exploitation capabilities. The deviations show how closely the solutions cluster around the global optimum. For the unimodal functions, this deviation around the solutions is the same for both in experiments, but in terms of performance the results are different. For multimodal functions there exist multiple local optima. The challenge here is not only to find the global optimum, but also to escape local optima. It is interesting to see that once this algorithmic balance between exploration and exploitation is needed for the algorithm, that both experiments have a more similar result. For the fixed-dimension multimodal functions there are a subset of multimodal functions where the number of design variables cannot be changed. These constrains of the dimensionality causes the solutions of both experiments to be very similar.

An explanation for the difference in the unimodal functions could be explained by the fact that some post-research fine tuning has been done to the algorithm in terms of exploitation. It is not known what the value of parameter $b$ was in their test environment. For defining constant $l$, a value of $[-1, 1]$ is randomly generated on each iteration. This version of the source code uses an extra parameter called $a2$ that is used to calculate this constant, but linearly decreases from -1 to -2. Removing this feature of $a2$ and keeping true to the definition of $l$ in the paper almost doesn't affect the performance. The rest of the experimental setup follows the same parameter setup as described by the paper and the encircling prey and the bubble-net attacking method follow the same mathematical descriptions as well. In the comments the source code is presented as the first demo, making it look like this setup still contains the original setup. Another clarification could be the fact that the algorithm is being run on a different software. The paper never mentions that the experiment has been done in MATLAB and also doesn't specify the hardware setup. It is only assumed that the original experiment is done in MATLAB as they provide the same test functions in this version. This is for example not the case for the source code written in the other languages. Next to this, the authors present the MATLAB source code on their website as the primary source code.

Overall, it can be concluded that the experiments to some extent have been reproduced successfully. The biggest differences lie in the unimodal benchmark functions with test functions F1, F2 and F3 in particular. For the multimodal functions the results have been reproduced with more accuracy. The most accurate reproduced results are those of the fixed-dimension multi modal benchmark functions.

## 3.5 Benchmarking WOA using IOHprofiler

We will now focus on the third and final research question for this thesis. How does WOA compare to other SOTA algorithms and how does it compare to other variants of WOA? The variants that WOA will be compared to, are ILWOA and the hybrid WOA-GA. For benchmarking WOA, ILWOA and WOA-GA and comparing them to other SOTA algorithms, IOHprofiler will be used as the benchmark tool. All three algorithms will be written and tested in Python. Before presenting the results, the two variants of WOA and IOHprofiler will be discussed.

### 3.5.1 ILWOA

In ILWOA, Lévy-flight distribution is used to provide for the discovery of candidate solution, a more random way of moving through the search space [ABAFS19]. The movement of the whale is adjusted with Lévy-flight distribution. In ILWOA, the coefficient vector $\vec{C}$ is replaced by the following three equations:

$$Lévy \sim \frac{\lambda\Gamma(\lambda)\sin(\frac{\pi\lambda}{2})}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0) \tag{10}$$

$$s = \frac{U}{|V|^{\lambda-1}}, \quad U \sim N(0, \sigma_u^2), \quad V \sim N(0, \sigma_v^2) \tag{11}$$

$$\sigma_u^2 = \left[\frac{\Gamma(1+\lambda)\sin(\frac{\pi\lambda}{2})}{\lambda\Gamma(\frac{1+\lambda}{2})2^{\lambda-1}}\right]^{1/\lambda}, \quad \sigma_v^2 = 1 \tag{12}$$

In Equation 10, parameter $\lambda$ represents a shape parameter that controls the tail behavior of the distribution. The distribution's tail are the parts of a probability distribution, that represent extreme values far away from the mean of a distribution. $\Gamma(\lambda)$ is called the gamma function. Parameter $s$ is defined with Equation 11. This represents the step size $s > 0$ and is drawn according to Mantegna algorithm [ABAFS19] [Man94]. U an V are samples drawn from Gaussian normal distributions with means of zero and variances of $\sigma_u^2$ and $\sigma_v^2$ [ABAFS19] [YKH14]. These variances are defined by Equation 12. Equation 10 acts more as a conceptual idea of the distribution. $\vec{C}$ is replaced by $s$. The algorithm also makes use of chaotic maps.

WOA was originally tested on a continuous search space. For the researchers that proposed ILWOA, it was necessary that their algorithm could handle searching in both continuous and discrete search spaces. To implement this idea, the researchers made use of a largest order value (LOV) to map between the continuous and discrete solutions. The discrete solution is represented with a permutation order of items that is packed in bins. For benchmarking with IOHprofiler, there will only be tested in a continuous search space. Therefor, this LOV mapping is not necessary in our implementation. This leaves the pseudocode of ILWOA to be exactly the same except for coefficient vector $\vec{C}$, getting replaced by parameter $s$ with Equation 11.

### 3.5.2 WOA-GA

WOA-GA is the hybrid algorithm that combines WOA for its exploration ability and GA for its exploitation ability [WWB22]. GA tries to imitate evolution in nature. It consists of three main phases called selection, crossover and mutation. Selection is the process where the best solutions of a generation are selected. The higher the fitness, the better the chances of an individual to be selected for the next generation. This is inspired from natural selection in nature. Crossover is used to combine the genetic information of two parents to generate a new offspring. It represents the biological process of reproduction. Finally, mutation introduces random changes to the offspring's genetic code and resembles biological mutation. This acts more as the exploration phase that can prevent the algorithm from premature convergence to a local optima. There exist different kinds of methods for each of these three phases. Selection can for exmaple be done through roulette

wheel selection, rank selection and elitism selection [HMR16]. For crossover there are methods like single-point crossover, two-point crossover and uniform crossover. For mutation is bitwise mutation the most common and simple method, but here as well other methods exist. For WOA-GA, the researchers have chosen to implement WOA-GA with the roulette wheel selection method, single-point crossover and bitwise mutation. The mechanics of these three phases will quickly be discussed.

**Roulette wheel selection**: In roulette wheel selection, the best fitness gets the highest probability of being selected for the next generation. The probability of each solution is their fitness divided by the total sum of the fitnesses of all solutions. WOA-GA will be benchmarked on minimization processes. Therefor the fitness of the lowest value needs to have the highest probability of getting selected. To ensure this happens, the inverse of the finesses will be used to calculate the probability for the selection process.

**Single-point crossover**: Single-point crossover is for WOA-GA designed to operate on a list of binary-encoded solutions. It uses pair adjacent solutions as parents and combines their binary information to produce a new offspring based on the corssover probability $p_c$. For each iteration a random value between 0 and 1 is obtained and when this value is smaller than $p_c$, a random index within the binary string is calculated that acts as the split index of both parents to recombine into the new offspring.

**Bitwise mutation**: For bitwise mutation, every bit in the binary string has a probability of being mutated. When this happens the bit get shifted from 1 to 0 or from 0 to 1. The probability of mutation is defined by $p_m$.

In Figure 5, a flow chart is provided that describes how WOA-GA operates.

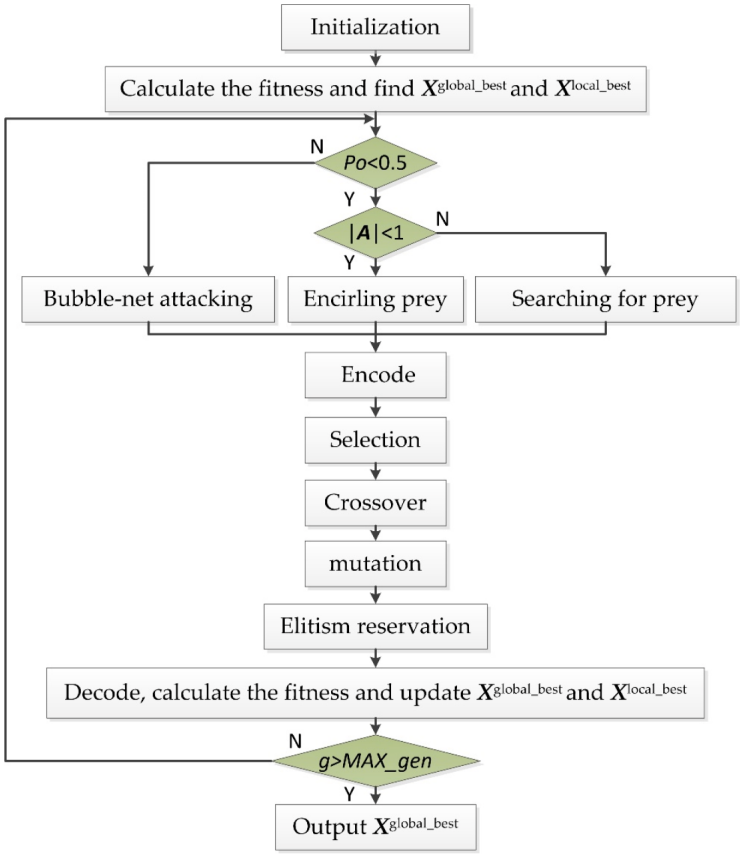



Figure 5: Flowchart of the WOA-GA optimization algorithm

The whole process of what WOA originally does, remains the same. After the whale has entered one of the three main stages of WOA, the solutions get encoded to a binary form. Then selection, crossover and mutation are applied. Elitism reservation is done to ensure that the best solutions are carried over to the next generation unchanged. The amount of unchanged solutions is preserved by the elitism rate $e_r$. If $e_r$ has a value of 0.05, the best top 5% of the solutions, will be ensured to make it to the next population. After elitism reservation has taken place, the new population will be decoded and the local best will be updated. If the local best has a better fitness than the global best, the global best will be updated to take over this better solution.

### 3.5.3 IOHprofiler

IOHprofiler is a benchmarking platform for evaluating the performance of iterative optimization heuristics (IOHs) [DWY+18]. IOHprofiler consists of multiple components that can be used to benchmark IOHs. The components that will be used for this thesis are IOHexperimenter to provide test functions, IOHdata that will be used for comparing to other algorithms that have been tested on the same data set and IOHanalyzer that can be used to visualize the obtained results of testing the algorithm.

All three algorithms will be benchmarked against 24 continuous Black-box Optimization Benchmarking (BOBB) problem functions. These 24 test functions can be divided into five categories. F1

to F5 are separable optimize functions. This means that each design variable is treated separately. This simplifies the problem. F6 to F9 are test functions with a low or moderate conditioning. Low and moderate conditioning allow for easier navigation through the search space. F10 to F14 are functions with high conditioning and are unimodal. As previously discussed, unimodal functions are function with one local optimum. Functions that have high conditioning, have steep areas in the search space, making it more challenging to be precise in convergence. F15 to F19 are multi-modal functions with a adequate global structure and F20 to F24 are multi-modal functions with weak global structure. Functions with a weaker global structure, are more challenging in finding the global optimum. The code for these 24 continuous BOBB problem sets can be found at https://github.com/IOHprofiler/IOHexperimenter/tree/master/include/ioh/problem/bbob. In Table 2, an overview is given of each of the 24 BBOB benchmark functions.

| Category | Function |
|---|---|
| **Separable Functions** | |
| F1 | Sphere Function |
| F2 | Separable Ellipsoidal Function |
| F3 | Rastrigin Function |
| F4 | Büche-Rastrigin Function |
| F5 | Linear Slope |
| **Functions with low or moderate conditioning** | |
| F6 | Attractive Sector Function |
| F7 | Step Ellipsoidal Function |
| F8 | Rosenbrock Function, original |
| F9 | Rosenbrock Function, rotated |
| **Functions with high conditioning and unimodal** | |
| F10 | Ellipsoidal Function |
| F11 | Discus Function |
| F12 | Bent Cigar Function |
| F13 | Sharp Ridge Function |
| F14 | Different Powers Function |
| **Multi-modal functions with adequate global structure** | |
| F15 | Rastrigin Function |
| F16 | Weierstrass Function |
| F17 | Schaffer's F7 Function |
| F18 | Schaffer's F7 Function, moderately ill-conditioned |
| F19 | Composite Griewank-Rosenbrock Function F8F2 |
| **Multi-modal functions with weak global structure** | |
| F20 | Schwefel Function |
| F21 | Gallagher's Gaussian 101-me Peaks Function |
| F22 | Gallagher's Gaussian 21-hi Peaks Function |
| F23 | Katsuura Function |
| F24 | Lunacek bi-Rastrigin Function |

Table 2: BOBB Functions

### 3.5.4 Implementation

To implement WOA in Python, the source code provided by the authors was obtained from https://seyedalimirjalili.com/woa as a starting point. This source code had five different default test functions, which where different from the original research paper and set to be able to handle only two dimensions. The source code had to be adjusted in order to meet the requirements for IOHprofiler and to be able to handle multiple settings of dimensions. When testing this version of WOA for the first time, the results were far worse compared to the benchmark functions it had been tested on in the experiment of the original research paper. It looked like this version of the source code was an incorrect implementation. On further analysis, two mistakes were found in the code. One mistake was the process how parameter $a$ decreased. The paper describes how this parameter should decrease from 2 to 0. In the code there was no condition set to stop decreasing once $a < 0$, eventually resulting in a negative value of $a$. An additional condition that $a$ is only allowed to decrease when $a > 0$ was added to the code. Another mistake was how coefficient vector $\vec{D}$ got calculated. In this version of the source code `np.linalg.norm()` was used on $\vec{D}$, making it a scalar parameter instead of a vector. After these corrections, the algorithm performed as expected.

For implementing ILWOA, only the way that $\vec{C}$ was calculated, had to be changed to the source code of the original WOA. This is replaced by the Lévy-flight distribution as shown in the math of Equation 10 to 12. For WOA-GA, roulette wheel selection, single-point crossover and bitwise mutation were implemented to handle the GA aspect of the algorithm. An encode and decode function were implemented as well to encode continuous data into binary strings and decode them back to their respective values.

For WOA-GA, an encode and decode function were written to switch between decimal and binary representations. Selection, crossover and mutation had to be implemented as well. Elitism reservation is applied to preserve the better solutions obtained by WOA itself. The source code for this thesis can be found at https://github.com/tvson2000/Evaluating-Whale-Optimization-Algorithm.

## 3.6 Results

To benchmark WOA, ILWOA and WOA-GA, an experiment is setup to evaluate the performance of these algorithms. All three algorithms have been tested on the BOBB functions with dimensions `D=[2,5,10,20,40]` with 50 repetitions and a maximum of 20000 iterations. For the parameter setup of WOA, the same setup as in the experiment of the original research has been used. For ILWOA and WOA-GA the same values of these parameters have been used as well as both research papers don't discuss these parameters. Both papers, however, do discuss the values of the new parameters that have been introduced to their variants. Table 3 showcases values of the parameter setup.

| Parameter | b | $a\_step$ | $\lambda$ | step size scale | $e_r$ | $p_c$ | $p_m$ |
|---|---|---|---|---|---|---|---|
| Value | 1.0 | 0.066667 | 1.5 | 0.01 | 0.05 | 0.8 | 0.05 |

Table 3: Parameter setup for WOA, ILWOA and WOA-GA

The original research paper only tested their results for the first 500 iterations. It is interesting to

see how WOA will perform on the first 500 iterations of these different optimization problems. The performance of an algorithm over 500 iterations can be important, but to only test the algorithm on a fixed-budget of this small amount is insufficient to investigate the potential of an algorithm. To expand on this, we will also take a look at a larger fixed-budget of 20000 iterations. We will also showcase the expected target value of the algorithms after a fixed-budget over a million iterations. The original research paper, only benchmarks WOA in a 30-dimensional search space and on problems with a fixed-dimension. We will evaluate the performance of the three algorithms across multiple dimensions. WOA will be compared with the well-established nature-inspired metaheuristics GA, PSO and DE, but also with one of the more competitive metaheuristic algorithms, a-CMA-ES.

### 3.6.1 Results over 500 iterations

PSO, GA and DE are compared in Figure 6 with WOA, ILWOA and WOA-GA over all 24 BBOB problems after 500 iterations in a 20-dimensional search space.

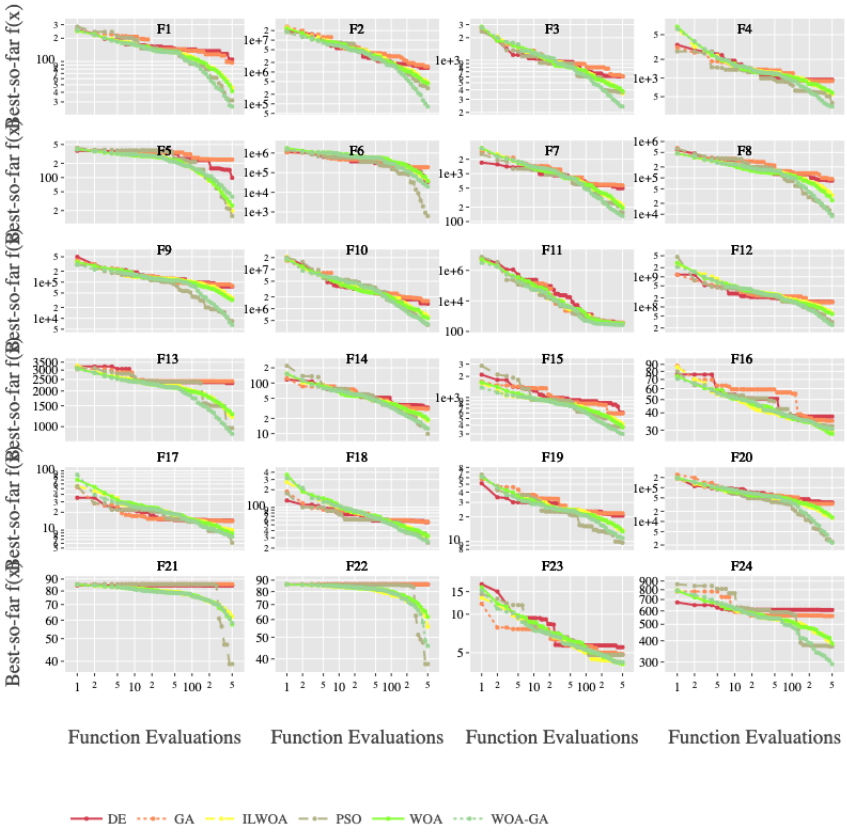



Figure 6: First 500 iterations over all 24 BBOB problems in a 20-dimensional search space

The results here show that in the first 500 iterations DE and GA overall have a slower convergence compared to the other algorithms. These two functions are struggling the most. PSO and WOA-GA seem to be the best performers. WOA-GA is the better performer out of the three variants. The difference between WOA and ILWOA is small. There are no individual benchmark functions where one of these two algorithm stands out on the other.

### 3.6.2 Results over 20000 iterations

In Figure 7 the same algorithms are shown in the same 20-dimensional search space, but after a fixed-budget of 20000 iterations. DE has compared to the other algorithms a much better performance over the 24 BOBB problems here than it had after 500 iterations, even in F21 being the strongest performer. This shows that testing on only 500 iterations can be insufficient to explore the full potential on algorithm. The difference between WOA and ILWOA has increased a bit, but this difference is still small. ILWOA is on most test functions slightly better than WOA with having the best performance on F22. GA still stands out as the weakest algorithm of them all, being the worse performer on almost all problems. PSO does overall best and WOA-GA is still the better performer out of the three WOA algorithms. One final interesting observation is that the behavior of WOA, ILWOA, WOA-GA and PSO is much more similar to each other compared to the behavior of GA and DE. This builds upon the critique that WOA has, for being fundamentally a PSO algorithm in terms of mathematical mechanics.

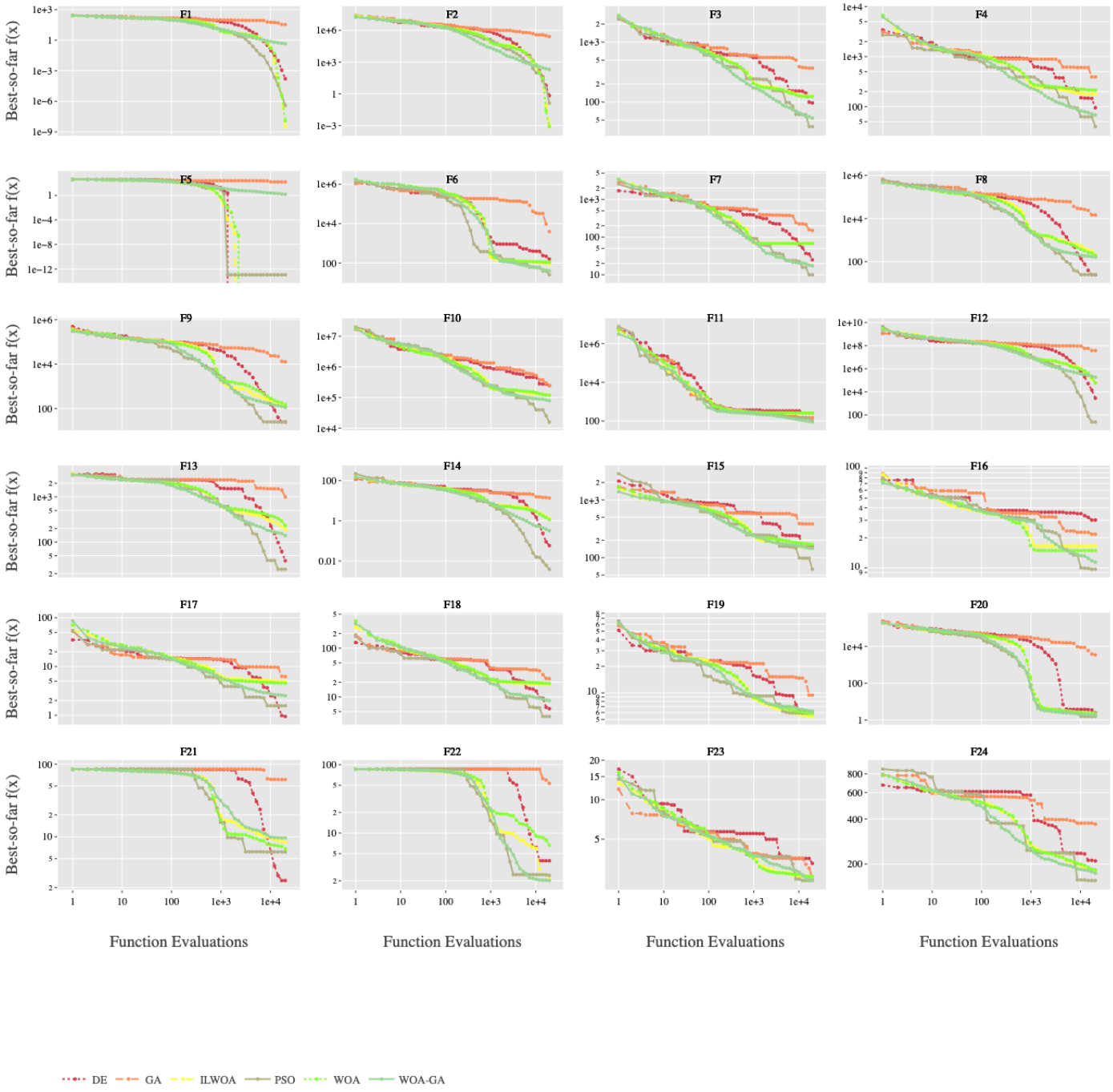



Figure 7: Results over 20000 iterations on all 24 BBOB problems in a 20-dimensional search space

### 3.6.3 Results compared to a-CMA-ES

The metaheuristic algorithm a-CMA-ES is one of the more competitive metaheuristics and acts as the standard algorithm in IOHprofiler for other IOHs to be compared with. We can see in Figure 8 that a-CMA-ES clearly stands out as the best performer, having only a few BOBB problems where other algorithms are competitive as well. It was chosen not to include a-CMA-ES in the previous

figures to give a better insight in scale of how the other algorithms compare to each other, because of the logarithmic scale the y-axis has.

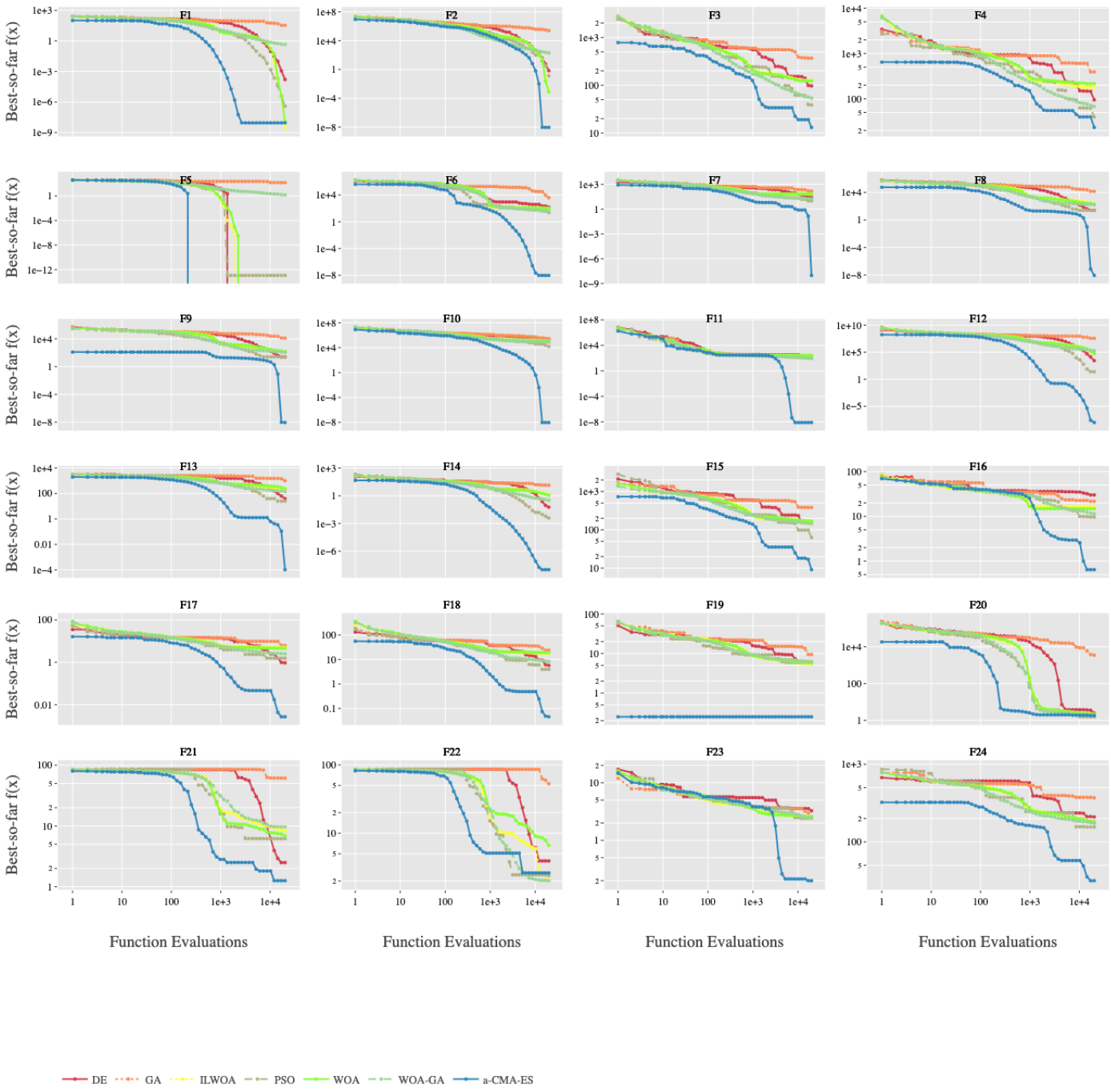



Figure 8: Results over 20000 iterations on all 24 BBOB problems in a 20-dimensional search space compared to a-CMA-ES

Even though the No Free Lunch theorem states that no metaheuristic is better on all optimization problems than the other, the inclusion of a-CMA-ES showcases how an optimization that is

mathematically fundamentally different to PSO, GA or DE, can have an outstanding performance overall on all different BOBB optimization problems. This appears not to be the case for WOA, ILWOA and WOA-GA, who are mathematically much more similar to PSO.

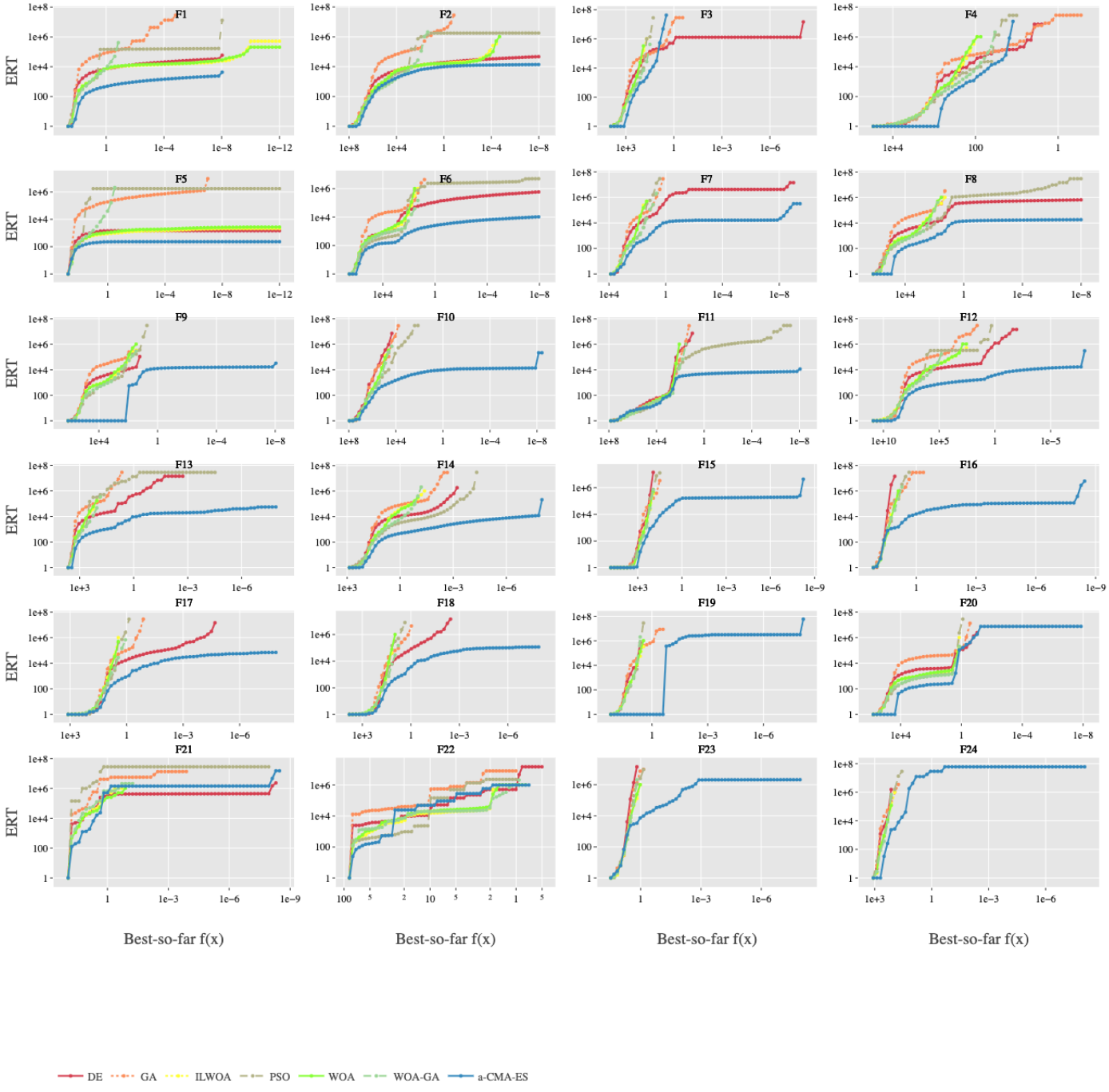


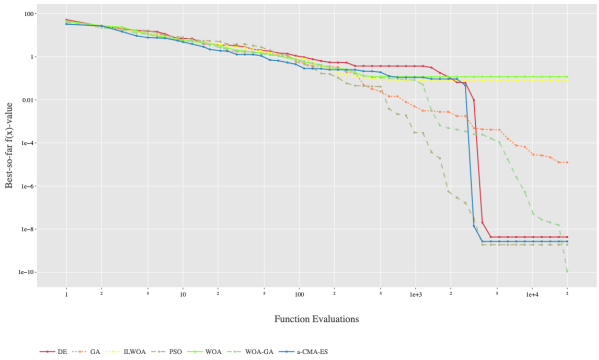

Figure 9: Expected fixed-target results over all 24 BBOB problems in a 20-dimensional search space

Figure 9 shows the expected fixed-target results of the algorithms. Again here a-CMA-ES proves to be the strongest algorithm overall. DE sometimes standout as the clear number two for example in

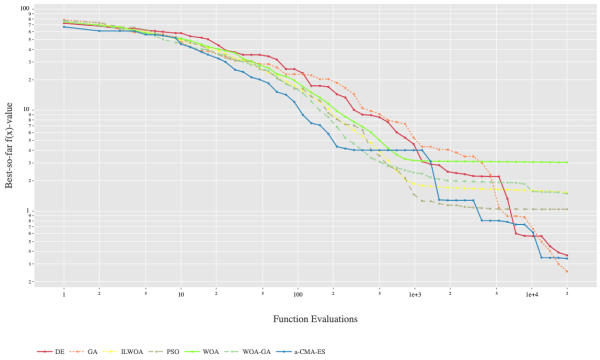
F3, F6, F7 and F21, even outperforming a-CMA-ES in F3 and F21. PSO is the clear number two in F11 and the clear number three in F8. Also here we can see that the difference in performance between ILWOA and WOA is minimal with WOA-GA often following the same results as well. On some BOBB problems, both of the three WOA algorithms are expected to find a premature convergence after a range between 10000 and 1000000 iterations. All algorithms seem to be the most competitive with each other for problem F22, one of the harder benchmark problems for finding the global optimum.

### 3.6.4 Results over multiple dimension

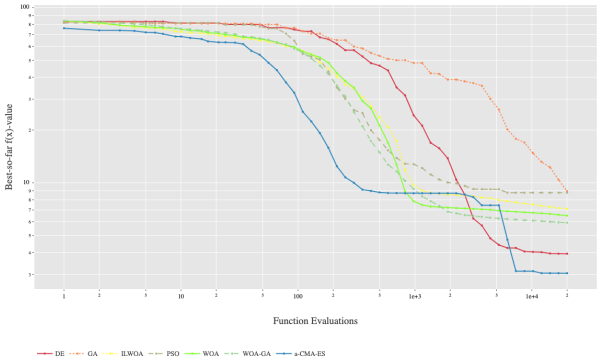
Until now the comparisons have been made only in a search space of 20 dimensions. The amount of dimensions can also affect the performance with some metaheuristics being better on a smaller set of dimensions and some on a higher set. Figure 10 shows the results of F22 over multiple dimensions with a fixed-budget of 20000 iterations. Here we see that for dimensionality 2, WOA-GA is surprisingly the most effective algorithm. DE and a-CMA-ES are here quite similar in performance, but eventually both converge prematurely where WOA-GA doesn't. WOA and ILWOA clearly struggle on lower dimensions and are also outperformed on a dimensionality of 20.



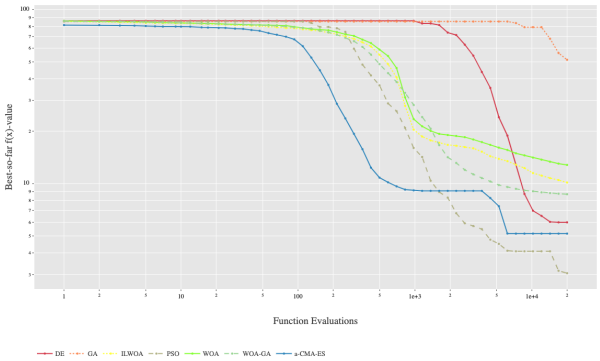

(a) Dimension 2



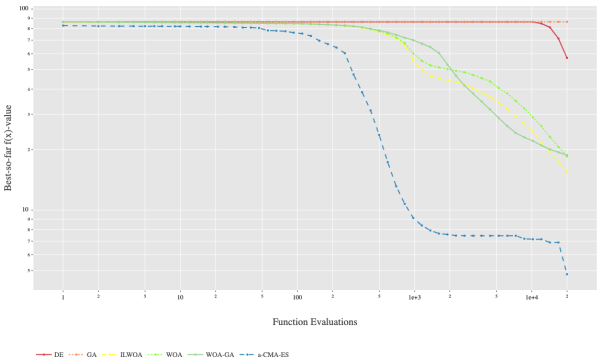

(b) Dimension 5




(c) Dimension 10




(d) Dimension 20




(e) Dimension 40

Figure 10: Results of F22 over multiple dimensions

The strength of WOA comes forward in the higher dimensional search spaces, especially with a dimensionality of 40. Here WOA and ILWOA both become competitive with WOA-GA. This increase in performance of WOA and ILWOA on a dimensionality of 40 seems to be consistent across al the benchmark functions. This can be observed in Figure 11. WOA-GA still remains the overall best performer of the three variants. Unfortunately the data for PSO with 40 dimensions is not available in IOHanalyzer. Therefor PSO is missing in Figure 11.

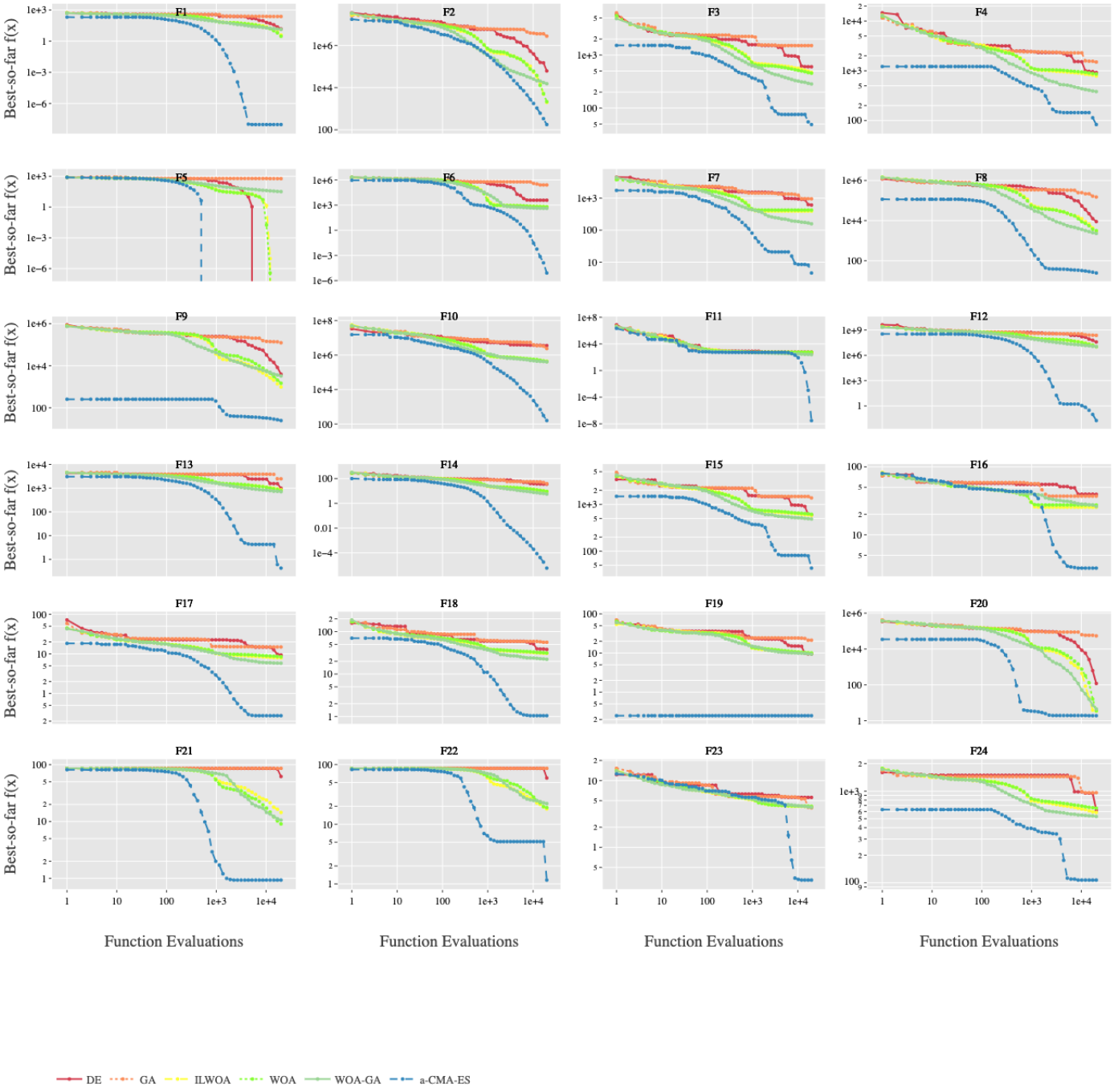



Figure 11: Results over 20000 iterations on all 24 BBOB problems with 40 dimensions

### 3.6.5 Computation time

Something that is also worth discussing, next to the actual performance of the algorithms, is the computational runtime. When two algorithms are very competitive with each other but one is significantly faster in terms of complexity, then this algorithm can be seen as the superior one. Figure 12 shows a heat map of the computational time, that shows which one is faster.

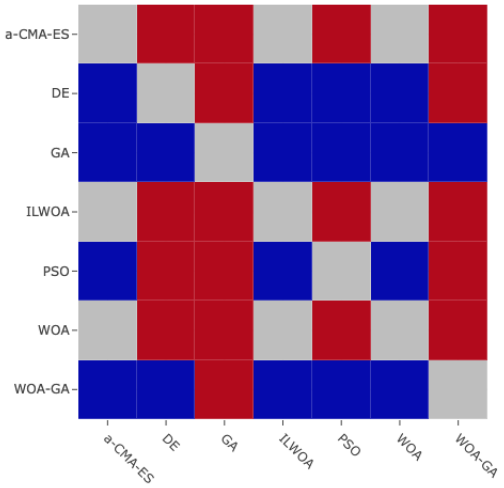



Figure 12: Heat map of run time on 20 dimensions

This heatmap should be interprated in the following way:

- Red: A row is better than the column
- Blue: A row is worse than the column
- Gray: No significant distinction between the row and column

It is clear that a-CMA-ES, WOA and ILWOA stand out as the faster algorithms, with having no significant distinction between them. GA is the slowest. After GA, WOA-GA is the slowest algorithm. It is interesting to see that WOA-GA appears to be faster than GA as next to the mechanics of WOA, it also includes selection, mutation and crossover. Next to this, WOA-GA has to follow the process of encoding to binary values and decoding the values back to their respective representations. This is also computationally expensive. Even though WOA-GA is in overall performance more competitive than WOA and ILWOA, it has a significant worse complexity. For a high dimensionality of for example 40 dimension where WOA and ILWOA are competitive with WOA-GA, these two algorithms actually might be preferred instead of WOA-GA. Especially when taken into account that higher dimensions, increase the computation time as well.

# 4 Discussion and Conclusion

For our first research question we investigated the mathematical rigor of WOA. We analyzed completeness, consistency and the meaning of the mathematical definitions. The authors overall did well in presenting the algorithm in such a way that it could be understood from a mathematical perspective. Despite the compelling narrative of WOA being inspired by the bubble-net hunting maneuver, the level of this being the case is limited. We discussed how WOA is a recombination of already existing mathematical concepts of previous defined methods and how it is more a PSO inspired algorithm than providing an actual new mathematical framework. WOA lacks mathematical rigor by being presented as the representation of the bubble-net hunting maneuver, but this is from a fundamental mathematical perspective not the case.

Secondly, the reproducibility of the original research done for WOA, was investigated. The authors were transparent in their research. They provided the actual source code for multiple programming languages. Their source code programmed in MATLAB presented the actual test functions that WOA was tested on in the original research paper. This made it easy to reproduce the results. The results where accurately reproduced to some extent. The results for the unimodals test functions in the original experiment were different from the ones we obtained, but this difference was leaning towards the performance of the replicated results being superior. This indicates that some post parameter tuning could have taken place. With more transparency towards what setup of parameters $b$ and $a\_step$ were and on what hardware and software WOA was tested, this may have been clarified. Ensuring that these results can be recreated, contributes to the robustness and credibility of the orignal research.

Finally, WOA, ILWOA and WOA-GA have been benchmarked. Out of the three variants, WOA-GA showed to be overall superior, but is the slowest in terms of computational time. The difference between WOA and ILWOA was small with ILWOA seeming to have a slight edge over WOA. For ILWOA to have the name "Improved" Lévy-flight Whale Optimization, seems from our results to be a bit of an overstatement. WOA and ILWOA were competitive on a dimensionality of 40, showing that WOA really is well performing on a high set of dimensions. Both WOA and ILWOA exhibit similar behaviors to PSO, reinforcing the critique that these algorithms are built upon mathematical concepts of PSO rather than being fundamentally new approaches.

Future research could focus on benchmarking WOA to more of its variants. It is interesting to see how there already exist so many variants of WOA, since its introduction in 2016. WOA-GA showed that a hybridization of two different algorithms, can have a big influence into how the algorithm improves and can be different in behavior from its predecessors. This shows that recombining algorithms can be beneficial. Research into the development of more distinct mathematical models would be encouraged even more. It is clear that there exists a lack in mathematical rigor for a lot of nature-inspired metaheuristics. Putting new nature-inspired metaphors on new introduced algorithms that are fundamentally not that different, causes for wild growth in the mathematical understanding of metaheuristics. More focus on the development of more distinct mathematical models is therefor needed.

I would like to give special thanks to Hao Wang for being the supervisor for this thesis. Secondly I would also like to give thanks to the second supervisor Haoran Yin and to the LIACS Natural Computing cluster, who provided the IOHprofiler benchmarking tool and helped me with my progress.

# References

[ABAFS19] Mohamed Abdel-Basset, Laila Abdle-Fatah, and Arun Kumar Sangaiah. An improved lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. *Cluster Computing*, 22(Suppl 4):8319–8334, 2019.

[CVDS23] Christian L Camacho-Villalón, Marco Dorigo, and Thomas Stützle. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. *International Transactions in Operational Research*, 30(6):2945–2971, 2023.

[DWY+18] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv e-prints:1810.05281*, oct 2018.

[FLT+11] Yongjiu Feng, Yan Liu, Xiaohua Tong, Miaolong Liu, and Susu Deng. Modeling dynamic urban growth using cellular automata and particle swarm optimization rules. *Landscape and Urban Planning*, 102(3):188–196, 2011.

[HMR16] Lingaraj Haldurai, T Madhubala, and R Rajalakshmi. A study on genetic algorithm and its applications. *International Journal of computer sciences and Engineering*, 4(10):139, 2016.

[Hol92] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[JYF+13] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, and Dusan Fister. A brief review of nature-inspired algorithms for optimization. *CoRR*, abs/1307.4186, 2013.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[Kit81] Philip Kitcher. Mathematical rigor–who needs it? *Noûs*, pages 469–493, 1981.

[Man94] Rosario Nunzio Mantegna. Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E*, 49(5):4677, 1994.

[ML16] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in engineering software*, 95:51–67, 2016.

[NSZAVM23] Mohammad H Nadimi-Shahraki, Hoda Zamani, Zahra Asghari Varzaneh, and Seyedali Mirjalili. A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. *Archives of Computational Methods in Engineering*, 30(7):4113–4159, 2023.

[Ple18] Hans E Plesser. Reproducibility vs. replicability: a brief history of a confused terminology. *Frontiers in neuroinformatics*, 11:76, 2018.

[PPP+08] C. Papagianni, K. Papadopoulos, C. Pappas, N. D. Tselikas, D. T. Kaklamani, and I. S Venieris. Communication network design using particle swarm optimization. In *2008 International Multiconference on Computer Science and Information Technology*, pages 915–920, 2008.

[Sör15] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[SSS21] Vishnu Soni, Abhay Sharma, and Vijander Singh. A critical review on nature inspired optimization algorithms. *IOP Conference Series: Materials Science and Engineering*, 1099(1), mar 2021.

[WWB22] Fang Wang, Zhijun Wu, and Tingting Bao. Time-jerk optimal trajectory planning of industrial robots based on a hybrid woa-ga algorithm. *Processes*, 10(5):1014, 2022.

[YKH14] Xin-She Yang, Mehmet Karamanoglu, and Xingshi He. Flower pollination algorithm: a novel approach for multiobjective optimization. *Engineering optimization*, 46(9):1222–1237, 2014.