People's Democratic Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي

جامعة قاصدي مرباح ورقلة
Kasdi Merbah University of Ouargla

**Academic Master Thesis**

to obtain a master's degree in Computer Science

**Major : Fundamental Computing**

# BHJO : A novel hybrid metaheuristic optimiser using Beluga Whale Optimisation, Honey Badger Algorithm and Artificial Jellyfish Search Optimiser for Optimisation Problems

*Realized by :*
Fatima Zohra KHECHIBA
Khadidja KHARCHOUCHE

*Supervised by :*
Dr. Farouq ZITOUNI
(UKMO)

*Presented in 18 June 2023, in front of the jury composed of :*

| | | |
|---|---|---|
| Dr.Abdelhakim CHERIET : | UKMO | - President |
| Dr.Meriem KHELIFA : | UKMO | - Examiner |

Promotion : 2022/2023

# Dedicace

“

بفرح عظيم وامتنان عميق، أتوجه بكل الشوق لأعبر عن تقديري الصادق وشكري العميق للأشخاص الأعزاء في حياتي.

إلى الذي وهبني كل ما يملك حتى أحقق له أماله، إلى من كان يدفعني قدمًا نحو النجاح والتفوق، إلى الشخص الذي أفتخر بحمل اسمه بكل فخر، إلى الذي سهر على تعليمي بتضحيات جسام،
أبي العزيز

إلى أغلى ما أملك في حياتي، إلى رمز الحب والعطف، إلى من يجسد معنى الرحمة والتضحية، إلى سر ابتسامتي وسر وجودي، إلى من دعاؤها يحقق لي النجاح ورحمتها تداوي جروحي، إلى أغلى حبيبة
أمي الغالية،

إلى إخوتي الأعزاء الذين كان لهم الأثر الكبير في تخطي العديد من العقبات والصعاب، دلال، حمزة، ولاء، عبد الحميد، رؤية ومحمد ياسين

إلى الجدتين العزيزتين اللتين رافقتني دائمًا بدعواتهما، أطال الله في عمرهما ورحم الله أرواح الجدين

إلى جميع أفراد عائلتي الكريمة، سواء كانوا قريبين أو بعيدين، أود أن أعبر عن امتناني وتقديري العميق لهم.

لأولئك الذين كانوا إلى جانبنا طوال هذا العام في إتمام هذا العمل، "خديجة" "سهلة" و"فردوس"، وتحية خاصة لزميلنا "بوعيشة عبد الحي" على نصائحه ومساعدته لنا. أنتم جميعًا كنز لا يقدر بثمن

أخيرًا ، أود أن أعرب عن خالص امتناني لكل من دعمني وساعدني طوال

رحلتي .أشكركم جزيل الشكر وأسأل الله أن يجعله في ميزان حسناتكم

وأخر دعوانا الحمد لله رب العالمي

”

***- Fatima.z Khechiba***

# Dedicace

“

أُهدي ثمرة جهدي "تخرجي" إلى ملاكي في الحياة وجنتي في الدنيا، الإنسانة العظيمة التي لطالما تمنيت أن تفرح برؤيتي في يوم كهذا. إلى روح جميلتي التي ارتقت إلى السماء، رحمها الله...

إلى والدي الغالي، أطال الله في عمره

إلى الشخص الذي ساندني وشجعني نحو النجاح، كان سندي ومسندي، الداعم الأول الذي لا يمكنني أن أكمل أفراحي بدونه... زوجي.

إلى أبنائي الأعزاء، لجين، هيثم، رنيم، رائد و ريتال أنتم ثمرة حياتي وفخري.

إلى أم أبنائي الغالية "أماني" وإلى أخوتي وعائلتي الصغيرة. وأيضًا إلى زميلتي في هذه المذكرة "فاطمة الزهراء" و كل من ساهم في هذا العمل المتواضع

أشكركم جزيل الشكر وأسأل الله أن يجعله في ميزان حسناتكم

وأخر دعوانا الحمد لله رب العالمين

”

*- Khadidja Kherchouche*

# Acknowledgment

First and foremost, we express our sincere gratitude to God Almighty for guiding us and granting us the strength and patience to navigate our academic journey and complete this work. We wholeheartedly acknowledge that this milestone has been achieved through his blessings and grace.

We would like to extend our heartfelt thanks to our parents, whose unwavering support, encouragement, and motivation have played a crucial role in our educational pursuits. Their boundless love and unwavering belief in our abilities have served as a constant source of inspiration and motivation. We are eternally grateful for their selflessness and dedication in shaping our lives and enabling us to reach this remarkable level of study.

We would like to express our sincere gratitude to our supervisor, **Dr. Farouq Zitouni**, for his unwavering support, guidance, and motivation throughout our research journey. are truly grateful for his continuous encouragement and belief in our abilities.

Likewise, we extend our respectful thanks to the members of the jury. The president **Dr.Cheriet Abdelhakim** and the examiner **Dr.Meriem Khelifa**, who have done the honor to participate in this jury and to examine this work.

We would like to take this opportunity to sincerely express our appreciation to all the professors in the Information Technology Department at the University of Kasdi Merbah Ouargla for their invaluable guidance throughout the past five years. Their dedication to teaching and commitment to fostering academic growth have played a pivotal role in shaping our knowledge and skills in the field.

Furthermore, we would like to extend you deepest gratitude and utmost respect to all our family members and friends who have supported us, both near and far, during our studies and in the preparation of this letter. Their contributions, whether through providing academic advice, engaging in research collaborations, or offering unwavering moral support, have been invaluable to our success.

# Abstract

Metaheuristic optimization algorithms often have limitations in terms of their exploration or exploitation capabilities. Therefore, relying on hybrid optimization algorithms has become essential to provide high-quality solutions. In this work, we propose a new hybrid optimization algorithm inspired by three recently developed algorithms, namely: Beluga Whale Optimization (BWO), Honey Badger Algorithm (HBA), and Artificial Jellyfish Search Optimizer (JS). The HBA and BWO algorithms have demonstrated promising exploitation capabilities and stable exploration phases, while JS has exhibited global exploration capacity but lacks sufficient exploitation during the exploitation phase. By leveraging these characteristics, we have combined the three algorithms in the exploration phase and integrate HBA and BWO in the exploitation phase. Therefore, we introduce a novel approach to achieve a balance between exploration and exploitation. To enhance population diversity and effectively guide the search process, we employ Compound Opposition-Based Learning (COBL) technology.We thoroughly investigate and analyze the performance of the proposed BHJO algorithm by comparing it to the base algorithms HBA, BWO, JS, and four other modern algorithms. Our evaluation includes 20 standard benchmark problems, 20 hybrid test suites, and composite problems for unconstrained optimization from IEEE CEC2017, CEC2020, and CEC2021. The performance and behavior analysis are evaluated using the Friedman test, followed by the Dunn's test to compare all the possible pairs of groups and identify the differences. The results demonstrate that BHJO outperforms other algorithms in terms of balancing exploration and exploitation. The computational complexity is also evaluated.

**Keywords :** Optimisation , Metaheuristic algorithms, Hybrid optimization algorithm, inspired nature, unconstrained optimization, Benchmark ($CEC2017, CEC2020$, and $CEC2021$) , Friedman and Dunn teste .

# Résumé

Les algorithmes d'optimisation métaheuristique présentent souvent des limitations en termes de leurs capacités d'exploration ou d'exploitation. Par conséquent, il est devenu essentiel de s'appuyer sur des algorithmes d'optimisation hybrides pour fournir des solutions de haute qualité. Dans ce travail, nous proposons un nouvel algorithme d'optimisation hybride inspiré de trois algorithmes récemment développés, à savoir : Beluga Whale Optimization (BWO), Honey Badger Algorithm (HBA) et Artificial Jellyfish Search Optimizer (JS). Les algorithmes HBA et BWO ont démontré des capacités d'exploitation prometteuses et des phases d'exploration stables, tandis que JS a montré une capacité d'exploration globale mais manque d'exploitation suffisante pendant la phase d'exploitation. En exploitant ces caractéristiques, nous avons combiné les trois algorithmes dans la phase d'exploration et intégré HBA et BWO dans la phase d'exploitation. Ainsi, nous introduisons une nouvelle approche pour atteindre un équilibre entre l'exploration et l'exploitation. Pour améliorer la diversité de la population et guider efficacement le processus de recherche, nous utilisons la technologie de l'apprentissage basé sur l'opposition composée (COBL). Nous étudions et analysons en détail les performances de l'algorithme BHJO proposé en le comparant aux algorithmes de base HBA, BWO, JS et quatre autres algorithmes modernes. Notre évaluation comprend 20 problèmes de référence standard, 20 ensembles de tests hybrides et des problèmes composites d'optimisation sans contrainte issus de IEEE CEC2017, CEC2020 et CEC2021. Les performances et le comportement sont évalués à l'aide du test de Friedman, suivi du test de Dunn pour comparer toutes les paires possibles de groupes et identifier les différences. Les résultats démontrent que BHJO surpasse les autres algorithmes en termes d'équilibre entre l'exploration et l'exploitation. La complexité de calcul est également évaluée.

**mot clé :** Optimisation, Algorithmes métaheuristiques, Algorithme d'optimisation hybride, nature inspirée, optimisation sans contrainte, Benchmark ($CEC2017, CEC2020$ et $CEC2021$), Friedman et Dunn teste.

# ملخص

تعاني خوارزميات التحسين ميتاهيوريستيك غالبًا من قيود فيما يتعلق بقدرات الاستكشاف أو استغلال الحلول. لذا، أصبح الاعتماد على خوارزميات التحسين الهجينة أمرًا ضروريًا لتوفير حلول عالية الجودة. في هذا العمل، نقترح خوارزمية جديدة للتحسين الهجينة مستوحاة من ثلاثة خوارزميات تم تطويرها مؤخرًا، وهي: خوارزمية الحوت الأبيض (OWB) غرير العسل (ABH) وخوارزمية البحث الاصطناعي للقناديل البحرية (SJ). لقد أظهرت خوارزميات ABH و OWB قدرات استغلال واعدة ومراحل استكشاف مستقرة، بينما أظهرت SJ قدرة استكشاف عالمية ولكنها تفتقر إلى استغلال كافٍ خلال مرحلة الاستغلال. من خلال الاستفادة من هذه الخصائص، قمنا بدمج الخوارزميات الثلاث في مرحلة الاستكشاف ودمج ABH وOWBفي مرحلة الاستغلال. بالتالي، نقدم نهجًا جديدًا لتحقيق توازن بين الاستكشاف والاستغلال. لتعزيز تنوع السكان وتوجيه العملية البحثية بشكل فعال، نستخدم تقنية التعلم القائم على الاعتراض المركبة (LBOC). نقوم بدراسة وتحليل أداء خوارزمية OJHB المقترحة بدقة عن طريق مقارنتها مع خوارزميات الأساس ABH وOWB وSJ وأربع خوارزميات حديثة أخرى. تشمل التقييم لدينا 02 مشكلة قياسية للمعايرة، و02 مجموعة اختبار هجينة، ومشكلات مركبة للتحسين غير المقيد من 7102CEC EEEI و 0202CEC و 1202CEC. يتم تقييم أداء وتحليل السلوك باستخدام اختبار فريدمان، تلاه اختبار دان لمقارنة جميع الأزواج الممكنة من المجموعات وتحديد الاختلافات. تظهر النتائج أن خوارزمية OJHB تفوق الخوارزميات الأخرى في تحقيق توازن بين الاستكشاف والاستغلال. يتم أيضًا تقييم التعقيد الحسابي.

---

**كلمات مفتاحية :** ,التحسينُ الخوارزميات هيوريستيكُ خوارزميات التحسين الهجينةُخوارزمبات المستوحاة من الطبيعة مشاكل التحسين غير المقيدةُاختبار فريدمان و دان وظائف $(CEC2017, CEC2020, CEC2021)$

---

# Contents

# List of Figures

# List of Tables

# Liste of algorithmes

# List of Abbreviations

| | |
|---|---|
| **SOO** | *Single Objective Optimization* |
| **MOO** | *Multiobjective Optimization* |
| **P** | *Polynomial Time* |
| **NP** | *Nondeterministic Polynomial Time* |
| **BHJO** | *hybrid metaheuristic optimiser using Beluga Whale Optimisation, Honey Badger Algorithm and Artificial Jellyfish Search Optimiser* |
| **BWO** | *Beluga Whale Optimisation* |
| **HBA** | *Honey Badger Algorithm* |
| **JS** | *Artificial Jellyfish Search Optimiser* |
| **COBL** | *Combined Opposition-Based Learning* |
| **PSO** | *Particle Swarm Algorithm* |
| **WOA** | *whale optimization algorithm* |
| **MFO** | *Moth-flame optimization algorithm* |
| **HHO** | *Harris hawks optimization* |

# General Introduction

## Context and problematic

On a daily basis, humanity confronts a multitude of intricate problems that necessitate effective solutions. To address these challenges, a systematic approach is employed to identify the problem and determine the most appropriate methodology for making decisions that optimize their interests. This process entails the application of methods aimed at enhancing the selected criteria, as improvement occupies a significant role in the research and engineering endeavors. To fully understand the problem of optimization, researchers and engineers follow a systematic process that involves first identifying the different objective functions that need improvement, along with the search space and variables associated with the problem.

To solve these problems, three primary research approaches are used in optimization: iterative, exact , and heuristic. The iterative approach involves mathematical calculations based on derivatives, Jacobians, and Gauss-Seidel to determine optimal solutions. These methods aim to converge towards an optimal solution by iteratively improving the initial solution.

On the other hand, the exact approach often refers to mathematical programming methods such as linear programming, integer programming, or nonlinear programming. It focuses on systematically exploring the search space using mathematical techniques to guarantee finding the optimal solution. In contrast, the heuristic approach, also known as exploratory search, follows guided random paths to find approximate solutions. Instead of exhaustively exploring all possibilities, heuristics use intelligent strategies or fundamental rules to quickly converge towards acceptable solutions. Researchers choose the most suitable approach based on the characteristics of the problem, available computational resources, and desired trade-off between solution quality and computational efficiency.

Optimization is a rapidly evolving field driven by advancements in technology, algorithms, and problem-solving techniques. Legacy algorithms may not be sufficient in today's landscape due to the availability of more efficient methods. Researchers continuously strive to improve optimization techniques to address complex problems across various domains. There is a high demand for innovative solutions as new challenges arise and existing problems evolve. This field offers opportunities for researchers to contribute by developing new approaches and enhancing algorithms to achieve better optimization outcomes.

## Contribution

Our thesis aims to make a valuable contribution by introducing a fresh hybrid metaheuristic optimizer called the Beluga Whale Optimisation, Artificial Jellyfish Search Optimizer, and Honey Badger Algorithm (BHJO). Initially, we focus on the design and implementation of this innovative algorithm and conduct tests using various benchmark problems. Subsequently, we compare the performance of our algorithm with that of established metaheuristic algorithms. The results demonstrate that our proposal is highly competitive and can effectively compete against the selected algorithms.

## Structure

This dissertation is organized into three chapters as follows:

In the first chapter“**Introduction Optimization**”, we provide an overview of the field of optimization. We begin by defining optimization problems and discussing their classification and characteristics. We explore the distinction between constrained and unconstrained optimization problems and delve into various complexity classes associated with optimization. we present a comprehensive classification of the methods commonly employed to solve optimization problems. For each category, we provide illustrative examples of algorithms.

In the second chapter“**Introduction to Metaheuristics**”, we cover technique used to generate initial solutions, and Search strategy, and Tuning and controlling of parameters, and Randomization techniques, and Classification of metaheuristic algorithms, and Hybridization methods, finally the “ no-free lunch theory”.

The third chapter “**BHJO:A novel hybrid metaheuristic algorithm for global optimization**”, represents the practical part of our thesis, we talk about the source biological inspiration for the algorithms used in our crossbreeding.Next, we introduce our mathematical implementation and pseudo code. Finally, we deliver the results of the algorithm and its comparison with the latest algorithms in metaheuristics.

# Chapter 1

# Introduction Optimization

## 1.1 Introduction

The desire to improve and optimize is a fundamental aspect of human nature that drives us to seek better solutions in various areas of life. Optimization, as a scientific and mathematical approach, provides a systematic framework for identifying and achieving these improvements. It involves finding the best possible solution to a problem[54], with the objective of maximizing or minimizing a specific measure or objective function.The applications of optimization are wide-ranging and diverse. It finds application in fields such as mathematics, engineering, economics, computer science, and many others. For example[97], in mathematics, optimization focuses on finding the maximum or minimum value of a function within certain constraints. This is done using various techniques and algorithms, such as gradient descent, Newton's method, or linear programming.

When it comes to optimization, there are two main categories: local optimization and global optimization. Local optimization aims to find the best solution within a specific region or neighborhood. It focuses on finding local minima or maxima, which are points where the objective function reaches the smallest or largest value in the immediate vicinity. Local optimization methods are often efficient in finding these local solutions.On the other hand, global optimization seeks to find the absolute best solution across the entire search space, considering all possible solutions. Unlike local optimization, which can easily find local minima or maxima, finding the global minimum or maximum is a more challenging task. It often requires more advanced algorithms and techniques, as well as careful consideration of the problem's constraints and properties[82].

This chapter is an introduction to the subject. It is organized as follows, starting by defining optimization problems and their taxonomy in section 1.2, after we will present classification of optimization problems in1.3 ,then the difference between constrained problems and unconstrained ones and how to deal with constraints in 1.4,1.5 then local search and global search is a hot topic, which is provided in section1.6 , moving to complexity classes and sub-classes in section1.7,then we present an overview of the deferent optimization methods in 1.8.

## 1.2 Optimization problem

In their daily activities, researchers, clients, companies, and government agencies often encounter various planning and optimization challenges. These challenges require making decisions from a range of options available. However, the selection of a particular option has implications for the user or organization. Evaluating the impact of different options is essential, and this evaluation is typically done using specific criteria.

To address these challenges, it becomes necessary to identify optimal or near-optimal solutions that align with the specific goals of the optimization problem. Mathematical equations and symbols can be employed to formulate optimization problems, taking into account the objective function, decision variables, and constraints. The formulation process is tailored to the particular problem and its unique requirements[61].

Optimization problems can be formulated using mathematical equations and symbols to represent the objective function, decision variables, and constraints. The formulation depends on the specific problem and its requirements. A general framework can be followed for formulating optimization problems, which encompasses the following key elements ,As show figure1.1: 1.1, 1.2, and 1.3 .

$$\min_{f\mathbf{x}\in R^d} f_m(\mathbf{x}) \ , m = \{1, 2, \ldots, M\} \tag{1.1}$$

under the constraints :

$$h_j(\mathbf{x}) = 0 \ , j = \{1, 2, \ldots, J\} \tag{1.2}$$

$$g_k(\mathbf{x}) \leq 0 \ , k = \{1, 2, \ldots, K\} \tag{1.3}$$

Or $f_m(\mathbf{x})$, $h_j(\mathbf{x})$, and $g_k(\mathbf{x})$ are functions of the vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)^T \tag{1.4}$$

Where

| | | |
|---|---|---|
| $f_m$ | : | $\Re \to \Re$ The objective function to be minimized over the n-variable vector $x$ . |
| $g_k(\mathbf{x})$ | : | Inequality constraints. |
| $h_j(\mathbf{x})$ | : | Equality constraints . |

1. Define the Decision Variables: Identify the variables that can be adjusted or optimized to achieve the desired objective. Assign symbols to represent these variables, such as $x_1, x_2, \ldots, x_n$.

2. Define the Objective Function: Specify the function that needs to be optimized. It can be a maximization or minimization objective. Represent the objective function using mathematical symbols and the decision variables, such as: $f(x_1, x_2, \ldots x_n)$.

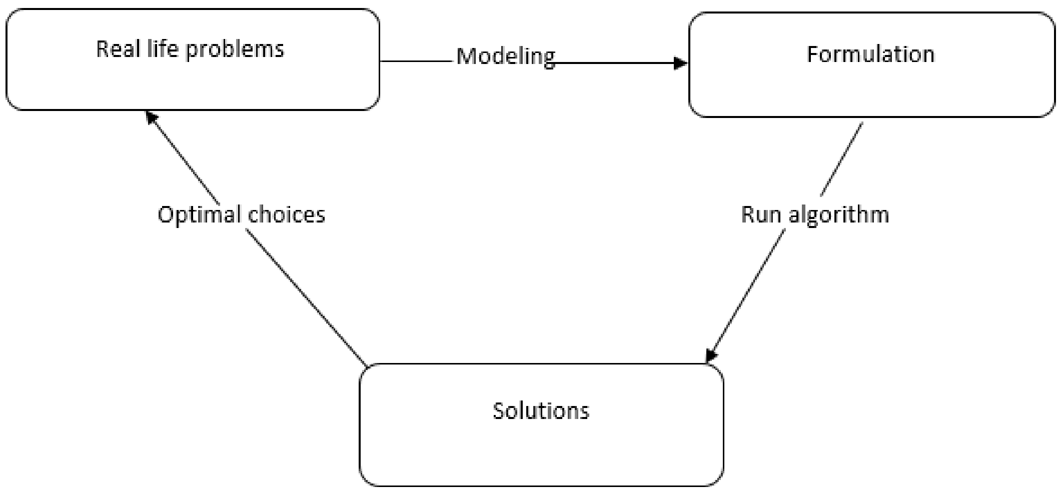
3. Identify Constraints: Determine any constraints or limitations that need to be satisfied in the problem. Constraints can be inequalities($\leq, \geq$)or equalities ($=$). Express the constraints mathematically using the decision variables, such as 1.2, and1.3.

4. Specify the Feasible Region: Define the feasible region, which represents the set of values that satisfy all the constraints. The feasible region is the region in which the decision variables can take values.

5. Formulate the Optimization Problem: Combine the decision variables, objective function, and constraints to formulate the optimization problem. The goal is to find the values of the decision variables that optimize the objective function while satisfying all the constraints.

## 1.3 Classification of optimization problems

Optimization problems can be classified into different categories based on their characteristics and mathematical properties. Here are some common classifications of optimization problems such as:



H

Figure 1.1: modeling problem

### 1.3.1 Single objective and Multiobjective optimization

In single objective optimization(SOO), there is a single objective function that needs to be optimized. The objective function represents a single measure of performance or a specific goal that the optimization seeks to achieve[124]. The task is to find the optimal values for the decision variables that maximize or minimize this objective function. Examples of (SOO) include maximizing profits, minimizing costs, or maximizing the efficiency of a system.

multiobjective optimization(MOO), the optimization problem involves multiple conflicting objectives that need to be simultaneously considered. Instead of a single objective function, there are multiple objective functions that represent different measures of performance or goals. These objectives may have competing interests, making it impossible to optimize they all simultaneously. The aim is to find a set of solutions that achieve a balance among these objectives, known as Pareto optimal solutions. A solution is Pareto optimal if there is no other feasible solution that improves one objective without sacrificing performance in another objective. (MOO) optimization is often used when decision-making involves trade-offs between different objectives, such as maximizing profit while minimizing environmental impact or maximizing customer satisfaction while minimizing production costs[39].

### 1.3.2 Continuous variables and Discrete variables

In continuous optimization, the variables can take on any real value within a defined range or domain. These variables are typically represented by real numbers and can be optimized using mathematical functions. Continuous optimization is well-suited for problems where variables can take on a wide range of values. Examples of continuous optimization problems include finding the optimal values of parameters in mathematical models or optimizing the dimensions of a physical system.

In discrete optimization, the variables can only take on a finite or countable set of values. These values are typically restricted to integers or specific discrete values. Discrete optimization problems involve finding the best solution from this restricted set of values. Examples of discrete optimization problems include combinatorial optimization, such as finding the optimal assignment of tasks to workers or selecting the best routing strategy in a network[76].

### 1.3.3 Linear Optimization and Nonlinear Optimization

Linear Optimization: In linear optimization, both the objective function and constraints are linear. This means the objective function is a linear combination of the decision variables, and the constraints can be expressed as linear equations or inequalities. Linear programming techniques, such as the simplex method, are commonly used to solve these problems efficiently. Linear optimization is used in resource allocation, production planning, and transportation logistics, among other areas.

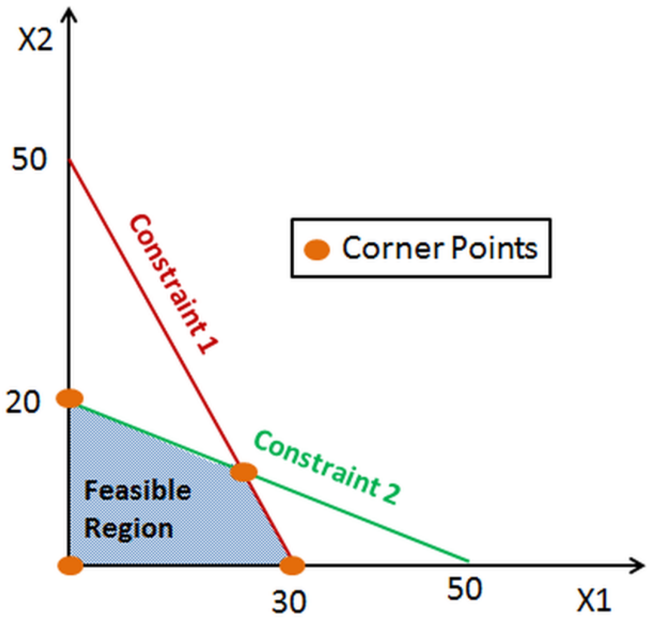
Nonlinear optimization deals with problems that involve nonlinear objective functions or constraints. In this case, the objective function or constraints contain nonlinear terms like products, powers, or trigonometric functions of the decision variables. Nonlinear optimization problems are generally more challenging to solve because they lack the nice mathematical properties of linearity. Advanced optimization algorithms, such as gradient-based methods or metaheuristic techniques, are often used to find optimal or near-optimal solutions. Nonlinear optimization is applied in engineering design, financial modeling, and data fitting, among other domains[53].

## 1.4 Unconstrained optimization

Unconstrained optimization refers to a category of optimization problems where the objective function is to be optimized without any constraints on the decision variables. In contrast to constrained optimization problems, there are no limitations or conditions imposed on the values that the decision variables can take[33]. The main objective in unconstrained optimization is to determine the optimal values for the decision variables that result in either the maximum or minimum value of the objective function. These decision variables can be either continuous or discrete, depending on the specific problem. To solve unconstrained optimization problems, it is typically necessary to identify the critical points of the objective function, such as local minimum or maximum points.

## 1.5 Constrained optimization

Constrained optimization refers to a class of optimization problems in which feasible solutions are subject to a set of constraints. These constraints impose restrictions or conditions on the decision variables that must be satisfied while optimizing the objective function.



h

Figure 1.2: Constraint problem optimization
https://www.datacamp.com/tutorial/linear-programming-with-spreadsheets

In a constrained optimization problem, the objective is to find values for the decision variables that optimize the objective function while simultaneously satisfying all specified constraints. Constraints can be categorized into two types:

**Equality constraints**, where the constraint equation must be exactly satisfied.

**Inequality constraints**, where the constraint equation must be satisfied within a certain range. The general formulation of a constrained optimization problem represented Equation1.6

$$\min_{f\mathbf{x}\in R^d} f_m(\mathbf{x}) \; , m = \{1, 2, \ldots, M\} \tag{1.5}$$

under the constraints

$$\begin{aligned} h_j(\mathbf{x}) &= 0 \quad , \quad j = \{1, 2, \ldots, J\} \\ g_k(\mathbf{x}) &\leq 0 \quad , \quad k = \{1, 2, \ldots, K\} \end{aligned} \tag{1.6}$$

Where

| | | |
|---|---|---|
| $f_m$ | : | $\Re \rightarrow \Re$ The objective function to be minimized over the n-variable vector $x$ . |
| $g_k(\mathbf{x})$ | : | Inequality constraints. |
| $h_j(\mathbf{x})$ | : | Equality constraints . |

To solve constrained optimization problems, specialized techniques are required to simultaneously handle objective function optimization and constraint satisfaction. Commonly used methods include linear programming, nonlinear programming, and constrained evolutionary algorithms.

The main objective in constrained optimization is to find optimal or near-optimal solutions that achieve the best possible objective value while satisfying all specified constraints. To address this, various techniques have been developed to combine the objective function and constraints into a unified equation. There are traditional and recent methods for handling constraints in optimization problems. Traditional methods include penalty methods, transformation methods, special representation techniques, and separation of objectives and constraints. Penalty methods introduce penalty terms into the objective function to incorporate constraints and transform constrained problems into unconstrained ones. Transformation methods use mathematical transformations to convert constrained problems into equivalent unconstrained problems. Special representation techniques explicitly represent constraints within the decision variables or problem structure. Separation of objectives and constraints involves treating objectives and constraints separately [40].

## 1.5.1 Penalty method

The penalty function is a technique used in constrained optimization problems to incorporate the constraints into the objective function. It achieves this by introducing a penalty term that quantifies the extent of constraint violation. The penalty term is added to the objective function, resulting in a modified objective function that penalizes solutions that do not satisfy the constraints[27]. By minimizing the modified objective function, the optimization algorithm seeks solutions that simultaneously optimize the objective function and satisfy the imposed constraints. The penalty function approach allows for the consideration of constraints during the optimization process and helps find feasible solutions within the given constraints.

$$\min F(x) = f(x) + P(x) \tag{1.8}$$

where $P(x)$ denotes the penalty term as given by 1.9.

$$P(x) = \sum_{j=1}^{J} \nu_j \max(0, h_j(x))^2 + \sum_{i=1}^{K} \mu_k |g_k(x)| \tag{1.9}$$

Where $\nu_j > 0, \mu_m > 0$ : Penalty coefficients.

## 1.5.2 Equality with tolerance

Equality with tolerance is a fundamental concept in optimization that addresses constraints involving strict equality. In real-world scenarios, achieving exact equality between variables and constants may not be practical due to factors such as measurement errors or uncertainties. To account for this, a tolerance value is introduced, allowing for a small deviation from the exact equality[119].In optimization problems, constraints of the form"$variable = constant$" are relaxed by introducing tolerance ranges. Instead of requiring an exact equality, the constraint is modified to allow for a difference between the variable and the constant within a specified tolerance. This relaxation means that the variable can deviate from the constant by an amount within the tolerance range, either as "$variable - constant \leq tolerance$" or "$constant - variable \leq tolerance$". This means that the difference between the variable and the constant should fall within the specified tolerance range. The tolerance represents an acceptable range within which the constraint is considered satisfied. The specific choice of tolerance depends on the nature of the problem and the desired level of precision.

One common representation of an equality constraint with tolerance in optimization is:

$$|h(x)| - \epsilon \leq 0$$

Where

| | | |
|---|---|---|
| $h(x)$ | : | Represents the constraint function. |
| $\epsilon$ | : | The tolerance value. |

### 1.5.3 Multi-objective approach to constraints

The multi-objective approach to handling constraints takes a different direction compared to weighted sum methods, which convert multi-objective optimization problems into single-objective ones. While this approach may initially raise questions about its effectiveness, studies have shown that it can be remarkably competitive.Instead of simplifying the problem into a single objective, the multi-objective approach considers conflicting objectives simultaneously while ensuring constraint satisfaction. Its goal is to find a set of solutions that optimizes multiple objectives, striking a balance between them and adhering to the constraints[81].

Although it may seem counterintuitive, the multi-objective approach to constraint handling has demonstrated its effectiveness in producing competitive solutions. It allows for a more comprehensive exploration of the trade-offs between conflicting objectives, providing decision-makers with a range of feasible options to consider.

## 1.6 Local search and Global search

### 1.6.1 Local search

Local search or local optimization is a method used to find optimal or near-optimal solutions within a limited neighborhood of the search space. Unlike global optimization, which aims to find the absolute best solution across the entire search space, local optimization focuses on improving the solution within a specific region[1].

In local optimization, an initial solution is chosen, and then iterative steps are taken to explore neighboring solutions and improve the objective function value[12]. The search is typically guided by a specific heuristic or algorithm that determines which neighboring solutions to explore and how to move towards better solutions[29].

The key characteristic of local optimization is that it operates based on local information, considering only the immediate neighborhood of the current solution. It does not guarantee finding the globally optimal solution, but rather seeks to improve the solution locally. Local optimization is often used when the search space is large and complex, making it computationally infeasible to explore the entire space[100]. Local optimization can be effective for solving problems where the objective function has multiple local optima, and finding the global optimum is not necessary or feasible. However, it may be limited by getting stuck in local optima and failing to find the best possible solution in the global sense[8].

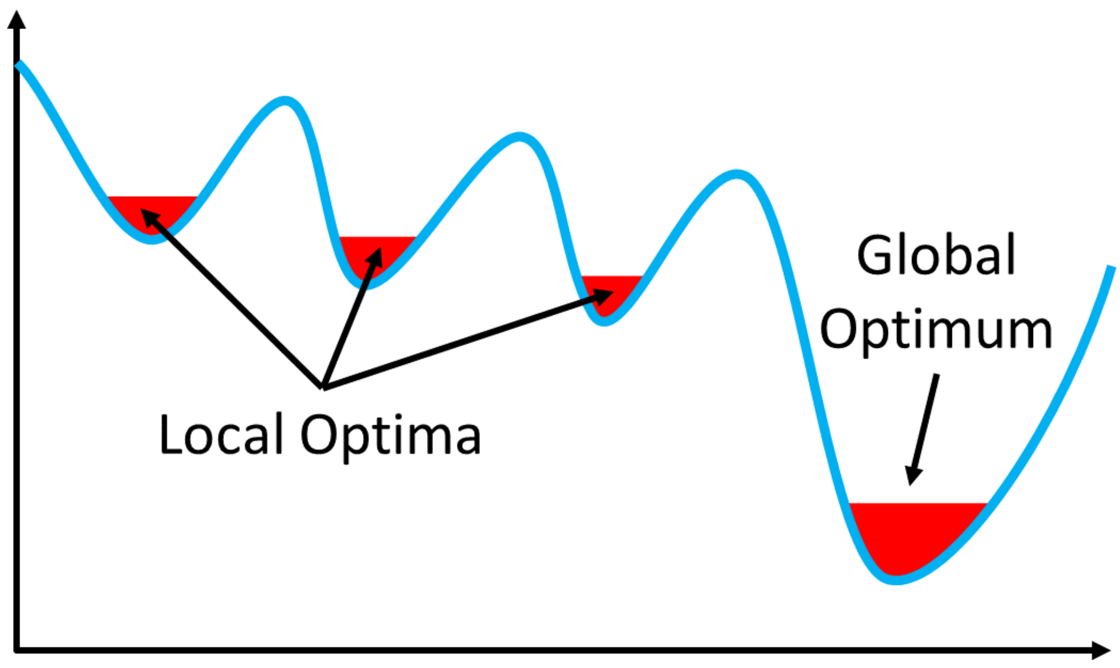


Figure 1.3: Local-Global optimum methods
https://www.allaboutlean.com/polca-pros-and-cons/local-global-optimum/

### 1.6.2 Globl search

Global search or global optimization, a method employed to discover the optimal solution across the complete range of potential solutions. Unlike local optimization, which focuses on improving solutions within a limited neighborhood[12], global optimization aims to find the globally optimal solution, considering all possible solutions and their corresponding objective function values[1].

In global optimization, the objective is to find the values of the decision variables that yield the optimal or near-optimal value of the objective function, regardless of where the optimal solution is located within the search space. This involves exploring the entire search space, which can be a challenging task, especially when the search space is large or complex[129].Due to the computational complexity involved in searching the entire space, global optimization methods often employ various techniques to efficiently explore the search space and improve the search process[29].

Global optimization methods strive to avoid getting trapped in local optima and aim to find the best possible solution in the entire search space. However, due to the complexity of the search, it is not always guaranteed to find the absolute global optimum, particularly in high-dimensional or non-convex problems.Despite their limitations, these methods can greatly enhance the search procedure and yield solutions that are very close to the optimal for a diverse array of optimization problems.

## 1.7 Complexity classes

A complexity class refers to a collection of functions that can be computed within a specified input[13]. While traditional complexity theory primarily focuses on decision problems solvable by Turing machines rather than optimization problems, it categorizes classes based on their resource requirements, such as time or memory. In the following, we will introduce the different complexity classes[82].

### 1.7.1 $\mathscr{P}$ class

The complexity class $\mathscr{P}$, known as the ”Polynomial Time” class, refers to the set of selection issues that may be solved with the aid of using a deterministic Turing gadget in polynomial time[95]. In other words, a problem belongs to class $\mathscr{P}$ if there exists an algorithm that can solve it with a time complexity bound by a polynomial function of the input size[54].

Being in class $\mathscr{P}$ means that there is an efficient algorithm that can find a solution to the problem within a reasonable amount of time, even for large inputs. Polynomial time complexity is generally considered manageable and desirable for problem-solving. Many fundamental problems in various domains, such as sorting, searching, and graph traversal, fall into the class of P problems. Algorithms like quicksort, binary search, and breadth-first search are examples of efficient algorithms that can solve these problems within class $\mathscr{P}$.

The class $\mathscr{P}$ encompasses problems that can be solved in polynomial time using deterministic algorithms, making it an important and widely studied complexity class in theoretical computer science.

### 1.7.2 $\mathscr{NP}$ class

The complexity class $\mathscr{NP}$ , known as the ”Nondeterministic Polynomial Time” ($\mathscr{NP}$) class, a set of decision problems is referred to as being in the class of $\mathscr{NP}$ if a solution to a given problem can be verified in polynomial time by a deterministic Turing machine. In other words, if a solution is proposed for an $\mathscr{NP}$ problem, it can be efficiently checked for correctness [54].

While the term ”Nondeterministic” may be misleading, $\mathscr{NP}$ does not imply that solutions can be found in polynomial time. Instead, it indicates that if a solution is given, its correctness can be efficiently verified. The term ”Polynomial Time” refers to the fact that the verification process can be performed in polynomial time.

Problems in $\mathscr{NP}$ are often characterized by the search for a solution among a large number of possibilities. This makes it challenging to find an optimal solution within a reasonable timeframe. Examples of $\mathscr{NP}$ problems include the traveling salesman problem, the knapsack problem, and the Boolean satisfiability problem. It is important to highlight that the issue of whether $\mathscr{P} = \mathscr{NP}$ or $\mathscr{P} \neq \mathscr{NP}$ remains one of the most significant unresolved problems in the field of computer science. If $\mathscr{P} = \mathscr{NP}$, it would imply that problems that can be verified in polynomial time could also be solved in polynomial time. However, if $\mathscr{P} \neq \mathscr{NP}$, it suggests that there are problems for which finding solutions is inherently more difficult than verifying them[60].

The class $\mathscr{NP}$ plays a fundamental role in complexity theory and has significant implications in various areas of computer science, including optimization, cryptography, and algorithm design.

## 1.8 Optimization methods

Optimization methods are developed with the goal of finding the best possible solution, or a solution that is very close to the optimal solution, while minimizing the computational effort needed. These methods involve searching through a set of potential solutions based on a given objective function and any constraints that are imposed. There are two main types of optimization methods: exact optimization methods and approximate optimization methods[44].

### 1.8.1 Exact optimization methods

Exact methods are optimization algorithms used to find at least one optimal solution to a problem [66], particularly when the problem size is small enough to be solved within a reasonable amount of time. These methods are capable of solving problems that fall into the category of $\mathscr{P}$, which refers to problems that can be solved in polynomial time. However, for larger or more complex problems, the computational cost and time required can become prohibitively high, as the search space grows exponentially with the problem size.

Despite the computational challenges, exact methods are valuable in scenarios where finding the absolute best solution is crucial and when the problem size allows for a manageable computation. They are widely used in various fields, including operations research, logistics, scheduling, and resource allocation. By exhaustively searching the solution space, exact methods provide rigorous and reliable solutions that can be used as benchmarks for evaluating the performance of approximate methods or as guarantees for achieving optimal solutions in specific problem instances.

#### Branch and Bound

Branch and Bound is an algorithmic technique used in optimization problems to systematically explore the search space and find the optimal solution efficiently[21]. It involves dividing the problem into smaller subproblems represented by nodes in a search tree. The algorithm applies two main strategies: branching and bounding. Branching involves partitioning the current subproblem into smaller subproblems by making decisions, while bounding estimates upper and lower bounds for the objective function value associated with each subproblem. These bounds guide the search by pruning unpromising branches. The algorithm terminates when all branches have been explored or when it determines that no better solution can be found. Branch and Bound is commonly used for combinatorial optimization problems and has applications in various domains.

### Dynamic programming

Dynamic programming is based on the principle of Bellman [Wolsey 1998]:

"If $C$ is a point that belongs to the optimal path between $A$ and $B$, then the portion of the same path from B to C is the optimal subpath between B and C."

This approach involves constructing optimal subpaths and recursively building the optimal path for the entire problem. In the following figure, the green path between A and B is optimal. This path passes through point C. The green subpath $[C, B]$ is therefore optimal, and the gray subpath $[C, B]$ cannot exist because it is shorter than the green subpath. In other words, if the gray subpath exists, the optimal solution becomes the sequence formed by the green subpath $[A, C]$ and the gray subpath $[C, B]$ instead of the green path $[A, B]$.

Dynamic programming relies on the principle of Bellman and involves extending the problem being studied to a more general problem with integer parameters. It typically requires determining recurrence relations that solve the problem of a given order based on its solutions for lower orders[67].

## 1.8.2 Approximate optimization methods

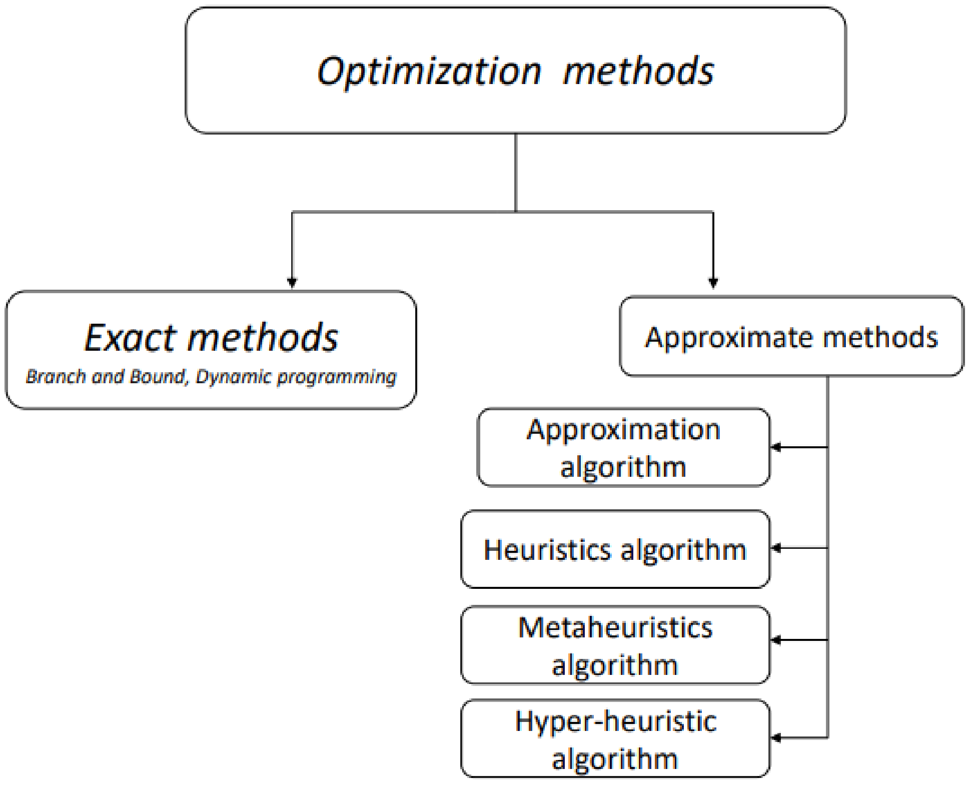
Approximate methode are optimization algorithms that aim to provide near-optimal solutions for computationally difficult problems[51]. These problems are often classified as $\mathscr{NP}$-hard, This implies that finding an exact optimal solution is either not feasible or demands a considerable amount of computational resources.

The objective of approximate methode is to find solutions that guarantee a certain level of closeness to the optimal solution. This closeness is measured by the approximation ratio, which quantifies the quality of the approximate solution in relation to the optimal solution. Generally, a smaller approximation ratio indicates a higher quality approximation[114].

Approximate methode are typically divided into three categories: approximation algorithms , heuristics ,metaheuristic .

### Approximation algorithm

An approximation algorithm is a specialized algorithm developed to discover solutions that are within a specified factor of the optimal solution. This factor, known as the approximation ratio, determines how close the approximate solution is to the optimal one. The smaller the approximation ratio, the better the quality of the approximation. Approximation algorithms are commonly used for $\mathscr{NP}$-hard problems where finding an exact optimal solution is impractical.



h

Figure 1.4: optimization methods

### Heuristics algorithm

Heuristics Are problem-solving techniques that rely on intuitive rules, strategies, or approximations to guide the search for solutions. In heuristic optimization, these heuristics are applied iteratively to explore the solution space, gradually improving the quality of the solutions. The key characteristic of heuristic optimization is its ability to trade off solution quality for computational efficiency. Heuristic algorithms may not guarantee finding the best solution, but they are designed to quickly find reasonably good solutions that satisfy the problem requirements[96].Heuristics can be classified into two categories[7]:

1. Constructive methods generate solutions starting from an initial solution and gradually adding elements until a complete solution is obtained.

2. Local search methods begin with an initially complete (potentially less desirable) solution and repetitively try to improve it by exploring its neighborhood.

### Metaheuristics algorithm

Metaheuristics are powerful optimization algorithms that efficiently explore large search spaces to find good solutions within a reasonable time frame. They provide a general framework for solving optimization problems by combining heuristics, local search methods, and problem-specific knowledge. Unlike traditional optimization methods[114], metaheuristics operate at a higher level of abstraction and are not tied to specific problem domains. They employ randomized or stochastic elements to balance exploration and exploitation, enabling them to overcome local optima and search for better solutions. Through iterative processes, solutions are refined and modified based on evaluation criteria or objective functions.In the upcoming chapter, you will discover more in-depth insights into various metaheuristics.

### Hyper-heuristic algorithm

Is a type of metaheuristic optimization that focuses on finding or generating effective heuristics for solving optimization problems. It operates at a higher level of abstraction by designing algorithms or frameworks that automatically generate or select heuristics, rather than directly optimizing the problem itself. the goal is to develop algorithms or techniques that can efficiently explore the solution space and adaptively select or generate heuristics to solve specific optimization problems. These heuristics can be considered as ”meta-heuristics” that guide the search process,it involves two key components: a set of low-level heuristics and a high-level framework or algorithm. The low-level heuristics represent different strategies or methods for exploring the solution space, while the high-level framework guides the selection or generation of these heuristics based on problem characteristics or performance feedback[36].The main advantage of hyper-heuristic optimization is its ability to adaptively search for and select suitable heuristics for different problem instances, without requiring explicit knowledge or customization for each problem. This makes hyper-heuristic approaches more flexible, scalable, and applicable to a wide range of optimization problems.

Hyper-heuristic optimization algorithms can be implemented using various techniques, such as genetic programming, machine learning, reinforcement learning, or evolutionary computation. These techniques allow the algorithm to learn and evolve over time, improving its ability to generate effective heuristics for different optimization problems[22].

## 1.9 Conclusion

In conclusion, the field of optimization is a powerful and versatile discipline that aims to find optimal solutions to complex problems. It utilizes various techniques, algorithms, and approaches to explore search spaces, improve efficiency, and achieve desired objectives. Throughout this chapter, we have discussed different types of optimization, including constraint and unconstrained optimization. We have also explored various methods and algorithms used in optimization, such as exact methods, approximate methods, heuristic methods, and metaheuristics.

The choice of the optimization method depends on factors such as problem complexity, available computational resources, time constraints, and the desired trade-off between solution quality and computational effort. Exact optimization methods are preferred when finding the absolute best solution is crucial, while approximate optimization methods offer efficient solutions that are acceptable in practice. By employing suitable optimization methods, researchers and practitioners can effectively tackle various optimization problems, finding optimal or near-optimal solutions with the least computational effort possible.

Finally, optimization is a dynamic and evolving field that continues to advance and contribute to various domains. By harnessing the power of optimization techniques, researchers, engineers, and decision-makers can find innovative solutions, make informed decisions, and effectively address complex challenges.

# Chapter 2

# Metaheuristics

## 2.1 Introduction

With the advancement of information technology, various fields such as engineering, bioinformatics, operations research, geophysics, etc..., and others have encountered numerous optimization problems Many of these problems are categorized as$\mathscr{N}\mathscr{P}$-hard problems, which implies that finding an efficient solution for them in polynomial time is only possible if$\mathscr{N}\mathscr{P}$ is equal to $\mathscr{P}$ [2]. As a result, finding exact solutions for larger instances of these problems becomes challenging. Consequently, engineers and researchers dealing with optimization problems must consider several factors when selecting an appropriate optimization algorithm [20].

When choosing an optimization algorithm, two common objectives are to reduce execution time and obtain an acceptable solution. To achieve these goals, Metaheuristic algorithms have become increasingly popular in the field of optimization. They belong to a class of algorithms specifically designed to tackle complex and challenging problems by finding good solutions [16]. They employ random operators, such as random initialization and perturbations, to explore the solution space and identify promising candidate solutions. Additionally, many metaheuristics incorporate local search algorithms to make incremental improvements to existing solutions. However, determining the optimal parameter settings for a specific problem significantly impacts the efficiency and effectiveness of the search process [108]. Despite not guaranteeing an optimal solution, metaheuristics have demonstrated strong performance across various problem domains. These algorithms are often inspired by natural phenomena, human intelligence, or animal behavior. Furthermore, metaheuristics are typically easy to implement and can be utilized by non-experts. Many software libraries and tools have been developed to provide practitioners with readily available metaheuristic implementations, minimizing the need for in-depth knowledge of underlying algorithms. Consequently, metaheuristics have become accessible to a broader audience, facilitating their adoption in diverse applications[108]. Many metaheuristics have been developed into software libraries and tools that can be readily used by practitioners without requiring extensive knowledge of the underlying algorithms. This makes metaheuristics accessible to a wider audience and enables their adoption in a variety of applications.

While heuristics are domain-specific and rely on experience and intuition to find solutions, metaheuristics offer a more generalized and systematic problem-solving approach. They are designed to tackle a wide range of optimization problems and often combine deterministic and stochastic elements. A concise quote highlighting the distinction between heuristics and metaheuristics is[34], ”A heuristic is a pretty good rule. A metaheuristic is a pretty good rule for finding pretty good rules.” This quote emphasizes the role of metaheuristics in developing and optimizing heuristics to efficiently explore solution spaces and identify near-optimal solutions.

This chapter will explore several crucial factors that contribute to the effectiveness of metaheuristics. These factors include techniques for generating initial solutions, search strategies, tuning and controlling of parameters, randomization techniques, classification of metaheuristic algorithms, hybridization methods, and the concept of the ”no-free lunch” theory.

## 2.2 Initialization methods

The initialization step in metaheuristics is a crucial component that sets the starting point for the search process. It involves generating an initial solution or a population of solutions for the optimization problem at hand. The quality and diversity of the initial solutions can significantly impact the performance and effectiveness of the metaheuristic algorithm. A well-designed initialization strategy in metaheuristics aims to generate an initial solution or population of solutions that can serve as a good starting point for the search process. The goal is to find an initial solution that is closer to the optimal solution, thereby reducing the time and effort required to converge towards the best possible solution[58].

There are various approaches to initialize solutions in metaheuristics, depending on the specific algorithm and problem domain. Theoretically, In theory, the available initialization methods can be broadly classified into two categories:

### 2.2.1 No Previous knowledge

In situations where there is little to no previous knowledge about the problem being solved, metaheuristic algorithms can overcome this limitation through simple initialization techniques[77]. We will examine two type of such techniques:

#### Random initialization

Random initialization refers to the process of generating initial solutions or starting points for an optimization or search algorithm using random values[102]. It is commonly used when there is no prior knowledge or information about the optimal solution or the search space, is particularly useful in population-based metaheuristics, such as genetic algorithms, particle swarm optimization, and ant colony optimization. These algorithms maintain a population of solutions and iteratively improve them over time. To start the process, a population of random solutions is generated, and the optimization algorithm iteratively refines these solutions to find better ones.

A typical approach to generate the random population is to use a uniform distribution as in Equation2.1:

$$X_i = l_b + rand(0,1)(u_b - l_b) \tag{2.1}$$

Where

| | | |
|---|---|---|
| $l_b$ and $u_b$ | : | the lower and upper bounds . |
| $rand$ | : | is a random number drawn from the range $[0,1]$. |

#### Chaotic initialization

Initialization with chaotic maps is a commonly employed technique in optimization algorithms, particularly in the field of metaheuristics. Chaotic maps are deterministic dynamic systems that exhibit chaotic behavior, characterized by sensitivity to initial conditions and a highly complex trajectory [102]. Various types of chaotic maps have been utilized, including logistic maps, pocket maps, and others. These maps have been specifically chosen to harness their chaotic properties for generating diverse initial solutions that are pseudo-random in nature. This approach aims to facilitate an extensive exploration of the solution space and improve the likelihood of discovering favorable solutions. In the literature[38], a generic formula has been proposed to represent this approach, as exemplified by the following Equation2.2:

$$X_{i,j}^{k} = f\ (X_{i,j}^{k}) \tag{2.2}$$

Where

| | | |
|---|---|---|
| $j$ | : | corresponds to the $j^{th}$variable of the $i^{th}$ individual of the population. |
| $f$ | : | epresents a chaotic mapping function. |
| $k = 1$ | : | is the iteration number, we are interested only in the first iteration. |

### 2.2.2 With Domain-Specific knowledge

some cases, domain-specific knowledge about the problem can be utilized to guide the initialization process. This knowledge can be incorporated into the initialization strategy to bias the initial solutions towards known or expected good regions in the search space. By incorporating problem-specific information, the algorithm can potentially converge faster towards optimal solutions[77].

When such domain-specific knowledge is available, it can be utilized to guide the initialization process in metaheuristic algorithms. The goal is to bias the generation of initial solutions towards regions in the search space that are known or expected to contain good solutions.

## 2.3 Search strategy

metaheuristics uses different techniques working together to converge toward an optimum, there is often a trade-off between different components, such as exploration and exploitation. achieving the best of both exploration and exploitation simultaneously in a single algorithm is a complex task. Researchers have indeed explored various approaches to address this trade-off, and the choice of methods depends on the problem at hand and the characteristics of the algorithm being used.

### 2.3.1 Exploration

In the exploration phase, the algorithm actively explores the entire solution space to uncover new and unexplored regions while moving away from existing solutions. This results in the generation of a diverse set of solutions. Global search strategies like random search or genetic algorithms are commonly employed to facilitate this exploration. These strategies generate candidate solutions across a wide range of the search space. The primary objective is to prevent the algorithm from becoming trapped in local optima. Diversification is typically achieved by promoting diversity among the members of the swarm. For instance, certain metaheuristic algorithms introduce randomness into the search process by incorporating random perturbations in parameter values or randomly selecting search directions. These techniques allow for the exploration of different areas of the solution space[63].

### 2.3.2 Exploitation

Exploitation can be described as the ability of an algorithm to use the current information available to it in order to search for a better position in the surrounding area of given candidates in the search space. Unlike the exploration technique, exploitation moves in smaller steps, generating outputs that are similar to its inputs but potentially better ones[28]. Exploitation is often viewed as an intensification phase rather than diversification because it focuses on improving the best solutions found so far, rather than generating new, diverse solutions. This stage typically employs local search strategies that fine-tune existing solutions to improve their quality.

Achieving a good balance between exploration and exploitation is indeed crucial for a metaheuristic to exhibit good performance, If a metaheuristic algorithm focuses excessively on exploitation and neglects exploration, it may converge quickly to a local optimum but at the risk of missing the global optimum. This means that the algorithm may settle for suboptimal solutions without sufficiently exploring other regions of the solution space. And if a metaheuristic algorithm emphasizes excessive exploration and underemphasizes exploitation, it may converge very slowly and require a significant computational effort. The algorithm might spend a considerable amount of time exploring unpromising regions of the solution space without effectively exploiting the known good solutions.

## 2.4 Tuning and Controlling of parameters

Achieving optimal performance in an algorithm often relies on the selection of appropriate algorithm-dependent parameters. Fine-tuning these parameters to their optimal values can significantly enhance the algorithm's effectiveness. The ideal scenario is for the algorithm to find the optimal solution with minimal iterations while maintaining high accuracy. However, parameter tuning poses a challenging optimization problem and can be considered a form of hyper-optimization, where the goal is to optimize the optimization process itself. Despite ongoing research efforts, determining the best parameter settings for an algorithm remains an open problem [122].

### 2.4.1 Tuning parameters

Parameter tuning is an important aspect of optimizing algorithm performance. Various tools and approaches exist for parameter tuning [118], but there is no universally established method that applies to all algorithms. One commonly used approach for parameter tuning is grid search, where a predefined set of parameter values is specified, and the algorithm's performance is evaluated for each combination of these values. This approach is straightforward but can be computationally expensive, especially when dealing with a large parameter space. Another popular approach is random search, where parameter values are randomly sampled from a predefined range or distribution. This method can be more efficient than grid search as it allows for exploration of a wider range of parameter value.

### 2.4.2 Controlling of parameters

that parameter control is another important aspect of optimization that should not be overlooked. While parameter tuning involves finding the optimal values of algorithm parameters [121], parameter control involves adjusting the values of these parameters over the course of the optimization process in order to improve the convergence rate and overall performance of the algorithm.the purpose of parameter control is to adaptively adjust the algorithm parameters based on the behavior of the optimization process. This can help to improve convergence rates, reduce the likelihood of getting stuck in local optima, and promote exploration of the search space. The goal of parameter control is to find a good balance between exploration and exploitation.

## 2.5 Random Walks

Randomization techniques, such as random walks, have become an essential component in the search process of stochastic algorithms. However, achieving effective randomization remains an ongoing challenge. To introduce randomization, pseudo-random numbers are commonly used, which are generated using deterministic algorithms that produce a sequence of numbers that exhibit properties similar to randomness. The distribution of these pseudo-random numbers depends on the specific algorithm and its parameters. Moreover, different probability density distributions, including uniform, Gaussian, Brownian, and Lévy distributions, can be employed to model random variables [121].

A random walk is a stochastic process that involves taking a sequence of consecutive random steps. Mathematically, we can represent a random walk by denoting the sum of each consecutive random step as $S_N$, where each step is represented by $X_i$. In this way, $S_N$ forms a random walk defined by Equation 2.3

$$S_N = \sum_{i=1}^{N} X_i = X_1 + \ldots + X_N \tag{2.3}$$

Where $X_i$ is a random step drawn from a random distribution. This relationship can also be written as a recursive formula.

$$S_N = \sum_{i=1}^{N} X_i + X_N = S_{N-1} + X_N \tag{2.4}$$

### 2.5.1 Gaussian Walk

A Gaussian walk, also referred to as a random walk with normally distributed steps, is a stochastic process in which a particle or system moves in discrete steps following a Gaussian or normal distribution. In other words, during each step of the random walk, the particle's displacement is determined by a random value drawn from a normal distribution[20]. This distribution is characterized by a mean of zero and a specified standard deviation, often denoted as $\sigma$, while $\mu$ represents the mean. The displacement at each step is independent of previous displacements as show in figure2.1 .

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X-\mu)^2}{2\sigma^2}} \tag{2.5}$$

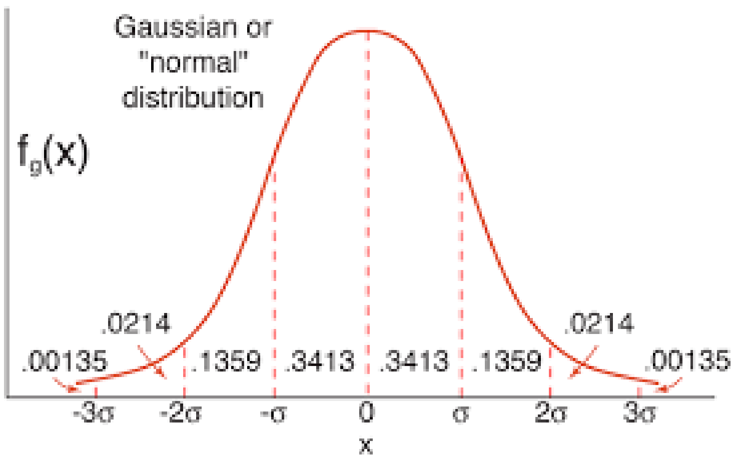


Figure 2.1: Function of Gaussian Distribution
hyperphysics.phy-astr.gsu.edu/hbase/Math/gaufcn.html

## 2.5.2 Lévy Walk

Lévy flights have been observed in various animal behaviors, including hunting and foraging. This is because Lévy flights enable animals to effectively explore their environment in unpredictable situations where food resources may be scattered or scarce[94].

To incorporate Lévy flights into an algorithm, two essential characteristics need to be defined: the step length of the walk, which follows the Lévy distribution, and the direction in which the walk should move towards the target position. The step length of the walk can be determined by generating random numbers from the Lévy distribution, which exhibits a heavy-tailed shape that allows for occasional large steps. The direction of the walk can be determined by generating random numbers from a uniform distribution. The variance of the uniform distribution is typically larger than that of a Gaussian distribution, making it more practical and efficient for simulating a wide range of scenarios[23].

## 2.5.3 Brownian Walk

Brownian motion, named after the botanist Robert Brown who observed the erratic motion of pollen particles in water, refers to the random movement of particles suspended in a fluid medium. It is a fundamental concept in physics and probability theory[64].

In a Brownian walk or Brownian motion, the random walker moves in a continuous manner, taking small and rapid steps in random directions[85]. The steps are typically assumed to be independent and identically distributed, following a Gaussian (normal) distribution. The cumulative effect of these small, random steps results in a trajectory that appears as a continuous, erratic, and unpredictable motion.

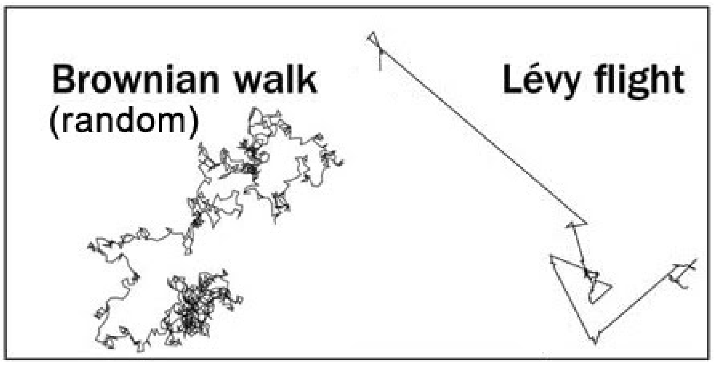


Figure 2.2: Brownian walk versus Lévy Flight walk adopted from[9]

## 2.6 Classification of meta-heuristic algorithms

Different methods have been proposed to classify metaheuristics based on selected characteristics. This section provides a brief summary of the most important categories, including nature-inspired versus non-nature-inspired, population-based versus single solution-based search, dynamic versus static objective function, single neighborhood versus various neighborhood structures, and memory usage versus memory-less methods. In our work, we focused on the category of population-based versus single solution-based search, examining the details within this classification:

### 2.6.1 Single solution-based

Single-solution algorithms, also referred to as local search algorithms, are designed to improve a single solution by exploring its immediate neighborhood. These algorithms begin with an initial solution and iteratively examine neighboring solutions until an improved solution is found or a termination condition is met. In the local search process of single-solution algorithms, the objective function of the current solution is evaluated, and small modifications are made to generate neighboring solutions. These modifications can involve altering a single element, swapping elements, or flipping bits within the solution. The algorithm then assesses the objective function of each neighboring solution and selects the best one as the new current solution. Here are some examples of single-solution algorithms:

### Simulated Annealing (SA)

Simulated annealing[18] is a metaheuristic that belongs to the family of local search methods. It is based on the analogy of the annealing process in metallurgy, where the metal is heated and then slowly cooled to achieve the desired microstructure. Simulated annealing works by randomly transitioning from the current solution to a new solution and evaluating the change in the objective function value. If the change is negative (i.e., the new solution is better), it is accepted. However, if the change is positive (i.e., the new solution is worse), it is accepted with a probability that depends on the temperature and the magnitude of the change. This probabilistic acceptance of worse solutions allows the algorithm to explore the solution space more extensively and avoid getting trapped in local optima. One of the main advantages of simulated annealing is its simplicity and ease of implementation.

### Tabu Search(TS)

Tabu Search (TS)[50] is a metaheuristic optimization algorithm designed to efficiently explore solution spaces and produce high-quality solutions. It is based on the idea that intelligent problem-solving requires two key components: adaptive memory and responsive exploration. The adaptive memory in TS is known as a ”tabu list,” which keeps track of previously visited solutions or actions that are deemed ”taboo” or forbidden for a certain period. By avoiding reconsideration of these forbidden solutions, TS promotes diversification in the search process and helps escape from local optima.

### Hill Climbing

The Hill Climbing algorithm[101] is a locally optimized method that uses feedback information to generate solutions. The algorithm simulates the process of climbing a mountain, moving in a higher direction at each step until it reaches the mountain's peak. Starting from the current node, the algorithm compares it with the values of neighboring nodes[83]. If the current node is the best, it is considered the maximum value (the highest point of the mountain). Otherwise, the highest neighbor node is selected to replace the current node, aiming to ascend the mountain. This iteration continues until the highest point is reached.

## 2.6.2 Population solution-based

Descriptive algorithms based on inference is a class one of the ways in which the population is updated Candidate solutions replace the existing population with a new, better one, and usually preserve the population. The community size is constant for each replicate Thus, the parallel exploration of many optima It can happen all at once, instead of the sequential exploration that the traditional one does Methods, whose point-to-point research is usually unable to overcome local ailments. Therefore, the possibility of being trapped in a bad locality is greatly diminished. Most (if not all) population-based methods are based on processes that occur in nature. thus, unlike the traditional methods, they use probability instead of determinism Transition rules, and the application of stochastic factors to the processes that direct the search, without Which means that the search performed is random. meta-heuristics algorithms can be categorized based on their inspiration[98]. There are four main categories of fanatical algorithms based on their inspiration Universe-based algorithms, Human-based algorithms, Evolutionary-based algorithms, Swarm-intelligence-based algorithms :

#### Universe-based algorithms

Universe-based algorithms are a class of metaheuristic optimization algorithms that draw inspiration from the principles of the universe, such as gravity and motion. These algorithms simulate the interactions between celestial bodies or particles to optimize a given objective function.Water cycle algorithm(WCA)[41], The proposed method is rooted in the fundamental concepts and ideas inspired by nature, specifically the observation of the water cycle process and the flow of rivers and streams towards the sea. Sine Cosine Algorithm (SCA)[88],which takes inspiration from sine and cosine functions and simulates their behavior to generate and iteratively update candidate solutions based on their fitness values. By simulating the properties of these mathematical functions. The Archimedes optimization algorithm(AOA)[55], is a metaheuristic optimization algorithm inspired by the principles and methods used by the ancient Greek mathematician Archimedes. The algorithm incorporates various mathematical and physical concepts to solve optimization problems. The atomic orbital search(AOS)[15],is an optimization algorithm inspired by the behavior of electrons in atomic orbitals. It draws analogies from quantum mechanics and utilizes the principles of wave functions and energy levels to solve optimization problems. Weighted mean of vectors (INFO)[6], INFO is a modified weighted average method that uses the weighted average concept to construct a fixed structure and update the position of vectors using three basic techniques: update rule, vector merging, and local search. The rule update phase uses the law based on averaging and convergence acceleration to generate new vectors. The vector combining step combines the resulting vectors with an update rule to produce a promising solution. The presented algorithm is inspired by the orbital dynamics observed in the solar system, including the motions of celestial bodies such as the sun, planets, moons, stars, and black holes. It adopts these orbital behaviors to address optimization problems.Solar system algorithm(SSA)[133], The presented algorithm is inspired by the orbital dynamics observed in the solar system, including the motions of celestial bodies such as the sun, planets, moons, stars, and black holes. It adopts these orbital behaviors to address optimization problems. Henry gas solubility optimization (HGSO)[56], is inspired by the Henry's law in chemistry, which describes the solubility of a gas in a liquid. The algorithm mimics the process of gas molecules dissolving in a liquid and aims to find the optimal solution .

#### Human-based algorithms

Human-based algorithms, also known as human-in-the-loop or human-guided optimization, involve incorporating human input or intervention into the optimization process. These algorithms aim to harness the strengths of human intuition and creativity to enhance the optimization process, particularly in situations where traditional methods may struggle to find optimal solutions. Let's comment on a few algorithms and mention others. Human mental search(HMS)[92], The poor and rich optimization (PRO)[91], inspired by the dynamics of two distinct groups: the poor and the rich, striving to improve their economic conditions and achieve wealth, applies the concept of poverty and wealth metaphorically to problems of gentrification. Poor individuals represent the initial solutions or search points with lower fitness values, while rich individuals correspond to solutions with higher fitness values. the Doctor and Patient Optimization (DPO)[31], Inspired by the doctor-patient relationship, the algorithm aims to simulate the diagnosis and treatment process in order to find optimal solutions for complex problems. Where the optimization problem is dealt with as a patient, the algorithm acts as a doctor trying to diagnose and provide the best treatment.Harmony Search(HS)[49], is a metaheuristic search algorithm inspired by the process of musicians improvising to find pleasing harmonies. In recent years, the HS algorithm has gained a lot of attention due to its many advantages. They are known for their ease of implementation and rapid convergence of optimal solutions. The Ebola Optimization Search Algorithm (EOSA)[93], The inspiration behind the Ebola Optimization Algorithm (EOSA) is derived from the Ebola virus and the way it spreads within a population. The algorithm takes inspiration from the random movement of individuals among different sub-populations, such as susceptible, infected, quarantined, hospitalized, recovered, and deceased individuals during an Ebola outbreak. By simulating the propagation of the disease, the algorithm aims to optimize solutions in a population-based manner. Bonobo Optimizer (BO)[52] ,is proposed It mimics several interesting reproductive strategies and social behaviour of Bonobos. Bonobos live in a fission-fusion type of social organization, where they form several groups (fission) of different sizes and compositions within the society and move throughout the territory. Afterward, they merge (fusion) again with their society members for conducting specific activities. Queuing search algorithm(QS)[127], inspired by human activities in queuing. It draws upon the principles and strategies observed in queuing systems to develop an efficient search algorithm.

#### Evolutionary-based algorithms

Evolutionary algorithms (EAs) are stochastic search methods that mimic the natural biological evolution and/or the social behavior of species. Such algorithms have been developed to arrive at near-optimum solutions to large-scale optimization problems, for which traditional mathematical techniques may fail[37]. The most popular evolutionary techniques are Genetic Algorithm (GA)[37] this algorithm based on theory of Darwin for evolution. GAs have some operators to evaluate its initial population generated randomly which are crossover, mutation and selection. the I Ching algorithm (ICA)[24], for optimization problem-solving. The algorithm incorporates unique operators inspired by the principles of the I Ching, an ancient Chinese cultural system. In addition, the algorithm utilizes transformation methods such as the penalty method and the multiplier method. The red deer algorithm(RDA)[43], takes inspiration from the behavior of male red deer during the mating season. Male red deer engage in competition to secure a large harem of females for mating purposes.The quantum-inspired evolutionary algorithm(QE)[109] is designed to handle continuous optimization problems while preserving the concept of superposition states. To achieve this, the algorithm incorporates a recursive sampling technique that progressively tightens the search space. By iteratively refining the search space .

### Swarm-intelligence-based algorithms

Swarm intelligence algorithms are nature-inspired algorithms developed based on organisms such as flocks of birds, ants, and fish[11]. These functions help algorithms in fitness functions in combination and numerical optimization problems from covering a wide range of search space, The Pity Beetle Algorithm (PBA)[69], developed by Kallioras is inspired by the gathering behavior and foraging strategies of Pityogenes chalcographus beetles, which have the ability to congregate on host trees and efficiently search for optimal nest sites and food sources using specific behaviors and communication patterns.The Sailfish Optimizer(SFO)[103], The algorithm you are referring to, which is inspired by a group of hunting sailfish, utilizes two types of populations: a sailfish population for intensification and a sardines population for diversification. is inspired by the behavior of sooty terns, a species of seabirds known for their remarkable navigation and foraging abilities.Sooty Tern Optimization Algorithm (STOA)[104],The algorithm mimics the foraging behavior of these birds to solve optimization problems.Chimp Optimization Algorithm (COA)khishe2020chimp a metaheuristic optimization algorithm inspired by the behavior and social structure of chimpanzees, one of the closest relatives of humans. The Archerfish Hunting Optimizer(AHO)[134], takes inspiration from the archerfish's ability to accurately target and shoot down insects by spitting water from its mouth. Dandelion Optimizer(DO)[130],is inspired by the characteristics and behavior of dandelion plants. Dandelions are known for their resilience, adaptability, and efficient dispersal of seeds. The DO algorithm aims to mimic these qualities in the optimization process. Mountain Gazelle Optimizer(MGO)[4], is inspired by the behavior and characteristics of mountain gazelles. Mountain gazelles are known for their agility, speed, and efficient navigation through complex terrains.Golden eagle optimizer(GEO)[90], is a nature-inspired metaheuristic algorithm inspired by the hunting and foraging behavior of golden eagles. Golden eagles are known for their powerful flight, keen vision, and efficient hunting strategies. Beluga whale optimization(BWO)[131] ,is a inspired by the social behavior and foraging strategies of beluga whales. Beluga whales are known for their cooperative hunting. The Coati Optimization algorithm (COA)[32], is a nature-inspired metaheuristic algorithm that mimics the foraging behavior and social interaction of coatis. Coatis, also known as coatimundis, are small mammals found in the Americas and are known for their efficient foraging strategies and group coordination.Water strider algorithm(WSA)[71], is a metaheuristic optimization algorithm inspired by the behavior of water striders, a type of insect that can walk on the surface of water. The algorithm mimics the movement and foraging strategies of water striders to solve optimization problems. Mouth Brooding Fish (MBF) [65], is a nature-inspired optimization algorithm that draws inspiration from the behavior of mouth brooding fish species. Mouth brooding fish, also known as parental fish, exhibit a unique reproductive strategy where the female fish carries and incubates the fertilized eggs in her mouth until they hatch. is takes inspiration from the foraging behavior of nutcracker birds. Nutcracker optimizer(NOA)[3], Nutcracker birds are known for their unique feeding strategy, where they gather and store food for future consumption.

## 2.7 Hybridization method

The concept of hybrid metaheuristics has gained acceptance in recent years, although the idea of combining different metaheuristic strategies and algorithms has been around since the 1980s [68].Hybrid solutions in metaheuristic algorithms involve combining two or more different algorithms or techniques to generate new and potentially better solutions. The objective of hybridization is to leverage the strengths of each algorithm while mitigating their weaknesses. There are various approaches to generating hybrid solutions[98].

One approach is the sequential method, where one algorithm is used to generate a set of candidate solutions, which are then refined using another algorithm. Another approach is the parallel method, where multiple algorithms work simultaneously to generate solutions, which are then combined to create new hybrid solutions. The choice of algorithms or techniques to combine depends on the specific problem being solved and the characteristics of the search space. It is important to carefully evaluate the performance of the hybrid algorithm and compare it to the performance of each individual algorithm or technique.

There are five important classes of hybrid metaheuristics[19]: constraint programming, tree search methods, problem relaxation, and dynamic programming. Hybridization with metaheuristics involves combining metaheuristics with other heuristics or optimization techniques, such as greedy algorithms, local search methods, or exact algorithms, to improve their performance. For example, a hybrid algorithm can combine a genetic algorithm with a local search method to efficiently explore the search space.Several articles have focused on the development of hybrid algorithms. Some examples include the hybridization of Grey Wolf Optimization and Particle Swarm Optimization (GWO-PSO)[106], a new hybrid GA/SA algorithm[126], an efficient hybrid DE-WOA algorithm[132], and improved hybrid AO and HHO algorithms[113].

Overall, hybrid algorithms provide a powerful approach to solve complex optimization problems by combining different optimization techniques to improve the performance of the algorithm. In this work, we propose A novel hybrid metaheuristic optimiser using Beluga Whale Optimisation, Artificial Jellyfish Search Optimiser, and Honey Badger Algorithm.

## 2.8 No-Free-Lunch Theorem

The ”No-Free-Lunch Theorem” states that if an algorithm performs well on a particular set of optimization problems, it may not perform well on another set of problems. This means that there is no universally superior algorithm that can outperform all other algorithms across all optimization problems[48]. However, this does not imply that all algorithms are equally effective for a given problem. Some algorithms may perform better than others for certain types of problems, and the choice of algorithm can have a significant impact on the quality of the solution obtained. Therefore, it is important to carefully consider the problem being solved and select an algorithm that is well-suited to that particular problem. This may involve comparing the performance of different algorithms on a set of benchmark problems or developing a new algorithm specifically tailored to the problem at hand.

## 2.9 Conclusion

In conclusion, scientists and researchers anticipate significant advancements in the field of metaheuristics in the future, given its wide range of applications across various domains.

The effectiveness of metaheuristics lies in its unique approach to generating solutions, which enables incremental convergence in search behavior. Achieving optimization relies heavily on finding the right balance between exploration and exploitation. Additionally, parameter tuning and control are critical aspects of metaheuristics, as these adjustable parameters have a significant impact on algorithm behavior and performance. Fine-tuning and controlling these parameters are essential for optimizing convergence speed, solution quality, and making informed decisions in random walks. Metaheuristic algorithms are classified into different categories based on their underlying principles and behaviors.

In this study, our exploration of the field of metaheuristics continues, and in the next chapter, we propose a novel hybrid metaheuristic optimizer that combines the strengths of the Beluga Whale Optimization, Honey Badger Algorithm, and Artificial Jellyfish Search Optimizer.

# Chapter 3

# BHJO:A novel hybrid metaheuristic algorithm for global optimization

## 3.1 Introduction

Optimization is a fundamental problem-solving approach that aims to find the best possible solution from a set of feasible alternatives. It plays a crucial role in various domains, including engineering[120], economics [62], logistics[10], and data analysis [111]. The complexity of real-world optimization problems often arises from large search spaces, non-linear relationships, and multiple conflicting objectives.

To tackle such challenging optimization problems, researchers have developed a diverse range of techniques, including metaheuristic algorithms. Metaheuristics are high-level problem-solving strategies that guide the search process by iteratively exploring and exploiting the search space to find optimal or near-optimal solutions. Unlike exact optimization methods that guarantee optimal solutions but are limited to small-scale problems, metaheuristics are capable of handling large-scale and complex optimization problems [128].

Metaheuristic algorithms draw inspiration from natural phenomena, social behaviors, and physical processes to create intelligent search strategies. These algorithms iteratively improve a population of candidate solutions by iteratively applying exploration and exploitation techniques. Exploration involves searching new regions of the search space to discover potential solutions, while exploitation focuses on refining and exploiting promising solutions to improve their quality. A good balance between exploration and exploitation is essential for the success of metaheuristic algorithms in solving optimization problems. Exploration allows the algorithm to search widely across the search space, uncovering diverse regions and potentially finding better solutions. It helps prevent the algorithm from getting stuck in local optima and promotes global exploration. On the other hand, exploitation focuses on intensively exploiting the promising regions of the search space, refining and improving the solutions to converge towards the optimal or near-optimal solutions. Balancing these two aspects ensures that the algorithm maintains a healthy exploration to discover new regions while exploiting the discovered promising solutions effectively. A well-balanced approach enables the algorithm to avoid premature convergence and thoroughly explore the search space to find high-quality solutions [78, 5].

Metaheuristic algorithms can be broadly categorized into two groups based on their search strategy: population-based algorithms and individual-based algorithms. Population-based algorithms maintain a population of candidate solutions and explore the search space collectively, exchanging information between individuals to guide the search process. Examples of population-based algorithms include Genetic Algorithms (GA) [75] and Particle Swarm Optimization (PSO) [112]. On the other hand, individual-based algorithms, also known as trajectory-based algorithms, focus on improving a single solution or trajectory by iteratively modifying it through exploration and exploitation. Examples of individual-based algorithms include Simulated Annealing (SA) [18] and Tabu Search (TS) [50]. These algorithms often rely on a memory mechanism to keep track of previously visited regions and avoid getting trapped in local optima.

Metaheuristic algorithms have demonstrated their effectiveness and versatility in solving a wide range of optimization problems in various fields. They have been successfully applied in areas such as engineering design [30], scheduling [17], resource allocation [99], data mining [79], and image processing[14] . The ability of metaheuristic algorithms to handle complex, non-linear, and multi-objective optimization problems makes them particularly useful in real-world applications where traditional optimization techniques may fail to provide satisfactory results.

In this paper, we focus on the combination of three well-known metaheuristic algorithms, Beluga Whale Optimization (BWO), Honey Badger Algorithm (HBA), and Artificial Jellyfish Search Optimizer (JS), to develop a hybrid metaheuristic algorithm. We aim to leverage the strengths of these individual algorithms and propose a novel approach that improves optimization performance by incorporating combined opposition-based learning and a new balancing mechanism. The algorithm is evaluated on a set of 40 test functions to assess its effectiveness and robustness.

## 3.2 Proposed hybrid algorithm

### 3.2.1 Source of inspiration

To show the working principle of the proposed hybrid algorithm, we explain the three metaheuristics used for its conception, which are the Beluga Whale Optimization (BWO) [131], the Honey Badger algorithm (HBA) [57], and the Jellyfish Search Optimizer (JS) [26]. In the following sections, we present an overview of each one.

#### Beluga Whale Optimization

The beluga whales (Delphinapterus leucas) [115] are members of whales living in the sea. They are medium-sized whales that live in the Arctic and subarctic oceans. Beluga whales are known for their social nature, sharp sense, and unique behavior, such as swimming with raised pectoral fins, diving and surfacing in a synchronized manner, and releasing bubbles from their blowholes. They are omnivorous and feed on a variety of prey, including fish and invertebrates, and can coordinate in groups to attack and feed on fish. However, they are also under threat from predators such as orcas and polar bears, as well as from humans. In addition, beluga whales die and fall to the ocean floor, providing a source of food for other deep-sea creatures. This phenomena is called whale fall. The mathematical formulation of exploration, exploitation, and whale fall (i.e., local optimum avoidance) concepts related to BWO are show in Figure 3.1 and are explained in the following sections.

**A- Exploration phase** The locations of search agents are determined by the pair swimming behavior, where two beluga whales swim together in a synchronized or mirrored manner[131]. This approach allows search agents to explore the search space more efficiently and effectively, and leads to discover new and possibly better solutions to the optimization problem being handled. The positions of the different agents are updated using Equation 3.1.

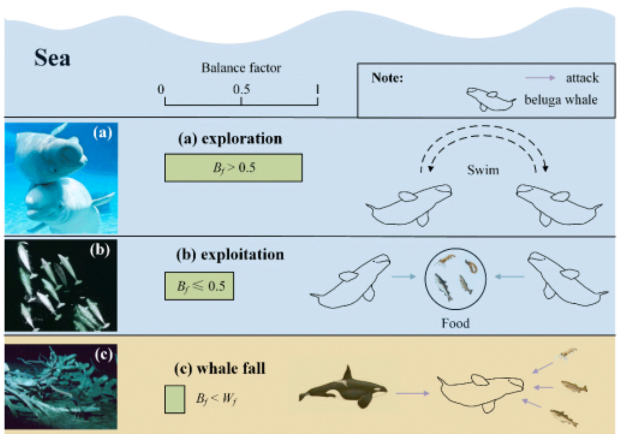


Figure 3.1: Behavior of beluga whales, (a) swimming (exploration phase), (b) foraging (exploitation phase), (c) whale fall (local optimum avoidance) [131].

$$\begin{cases} X_{i,j}^{t+1} = X_{i,p_j}^t + (X_{r,p_1}^t - X_{i,p_j}^t)(1+r_1)\sin(2\pi r_2) & , \quad j = 2k \\ X_{i,j}^{t+1} = X_{i,p_j}^t + (X_{r,p_1}^t - X_{i,p_j}^t)(1+r_1)\cos(2\pi r_2) & , \quad j = 2k+1 \end{cases} \tag{3.1}$$

Where

| | | |
|---|---|---|
| $t$ | : | The current iteration. |
| $X_{i,j}^{t+1}$ | : | The new value of agent $i$ on axis $j$. |
| $p_j$ | : | A random number drawn from the range $\{1, 2, \dots, d\}$. |
| $X_{i,p_j}^t$ | : | The position of agent $i$ on axis $p_j$. |
| $r$ | : | A random number drawn from the range $\{1, 2, \dots, N\}$. |
| $N$ | : | The population size. |
| $r_1$ and $r_2$ | : | Random numbers drawn from the range $[0, 1]$. |
| $\sin(2\pi r_2)$ and $\cos(2\pi r_2)$ | : | Designate the mirrored swimming behaviour. |

**B- Exploitation phase** The search agents share information about their current positions and consider the best solution as well as other nearby solutions when updating their locations. This mechanism helps the agents to efficiently move towards promising regions of the search space. The exploitation phase also includes the Lévy flight [84], which is a type of random walk characterized by long jumps in random directions interspersed with short local movements. This strategy helps enhancing search agents' convergence towards the global solution of the search space. The positions of the different agents are updated using Equations 3.2, 3.3, and 3.4.

$$X_i^{t+1} = r_3 X_{best}^t - r_4 X_i^t + 2r_4 \frac{1-t}{T_{\max}} L_F (X_r^t - X_i^t) \tag{3.2}$$

$$L_F = 0.05 \frac{\upsilon \times \sigma}{|\nu|^{\frac{1}{\beta}}} \tag{3.3}$$

$$\sigma = \left( \frac{\Gamma(1+\beta) \times \sin(\frac{\pi \times \beta}{2})}{\Gamma(\frac{(1+\beta)}{2}) \times \beta \times 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}} \tag{3.4}$$

Where

| | | |
|---|---|---|
| $X_{best}^t$ | : | The best solution in the current population. |
| $r_3$ and $r_4$ | : | Random numbers drawn from the range $[0, 1]$. |
| $T_{\max}$ | : | The maximum number of iterations. |
| $L_F$ | : | The lévy flight [84]. |
| $\upsilon$ and $\nu$ | : | Normally distributed random numbers. |
| $\beta$ | : | A constant number that equals 1.5. |
| $\Gamma$ | : | The Gamma function. |

**C- Balance between exploration and exploitation** During the search process, the balance between exploration and exploitation phases is granted based on a balance factor, named $B_f$, which is calculated using Equation 3.5. In addition to that, if the value of $B_f$ is greater than or equals a specific threshold (e.g. 0.5), then the search space is explored using Equation 3.1; otherwise, if the value of $B_f$ is less than the same threshold, then the search space is exploited using Equation 3.2.

$$B_f = B_0\left(1 - \frac{t}{2T_{max}}\right) \tag{3.5}$$

**D- Whale fall** The whale fall phase represents the fact that a beluga whale may die and become a food source for other creatures. In the BWO algorithm, the whale fall phase is used as a random operator to introduce diversity and prevent the search process from being trapped in local optima. The mathematical model of this step is expressed using Equations 3.6, 3.7, and 3.8.

$$X_i^{t+1} = r_5 X_i^t - r_6 X_r^t + r_7 X_{step} \tag{3.6}$$

$$X_{step} = (u_b - l_b)\exp(-(2W_f \times n)\frac{t}{T_{\max}}) \tag{3.7}$$

$$W_f = 0.1 - 0.05\frac{t}{T_{max}} \tag{3.8}$$

Where

| | | |
|---|---|---|
| $X_r$ | : | A random solution taken form the current population. |
| $r_5$, $r_6$, and $r_7$ | : | Random numbers drawn from the range $[0, 1]$. |
| $X_{step}$ | : | Represents the step size by which a dying whale falls. |
| $w_f$ | : | The probability of whale fall. |
| $u_b$ and $l_b$ | : | Upper and lower bounds of the search space. |

### Honey Badger Algorithm

Honey badgers are charming mammals known for their bold and stubborn nature. They are found in semi-deserts and rainforests of Africa, Southwest Asia, and Indian subcontinent. Honey badgers have distinctive fluffy black and white fur. Normally, they have

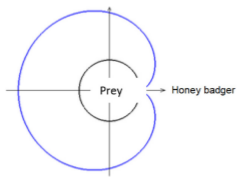


Figure 3.2: The black circle indicates the position of the prey, and the blue curve represents the smell intensity [57].

a body length of 60 to 77 centimeters and a body weight of 7 to 13 kilograms. Honey badgers rely on two modes for surviving: the digging mode and the honey mode [57]. In the digging mode, honey badgers use scent to locate prey, dig out and catch the prey. In the honey mode, honey badgers rely birds to locate beehives and get honey. The mathematical formulation and concepts related to HBA are explained in the following sections.

**A- Initialization phase** A population of initial candidate solutions is randomly generated between the lower and upper bounds $l_b$ and $u_b$ for the considered problem using Equation 3.9.

$$X_i = l_b + r_1(u_b - l_b) \tag{3.9}$$

where $r_1$ is a random number drawn from the range $[0, 1]$.

**B- Exploration phase** In nature, honey badgers dig a like-cardioid shape to surround and catch prey [86]. Figure 3.2 shows the form of a two-dimensional plane figure that has a heart-shaped curve. The exploration phase of HBA is formulated using Equation 3.10.

$$X_i^{t+1} = X_{best} + F\times\beta\times I\times X_{best} + F\times r_3\times\alpha X d_i\times \mid cos(2\pi r_4)\times[1 - cos(2\pi r_5)] \mid \tag{3.10}$$

$$F = \begin{cases} -1 & \text{if } r_6 \leq 0.5 \\ 1 & \text{else} \end{cases}$$

Where

| | | |
|---|---|---|
| $X_i^{t+1}$ | : | The new position of agent $i$ at iteration $t + 1$. |
| $X_{best}$ | : | The position of the best solution in the current population. |
| $I$ | : | The smell intensity, computed using Equation 3.12. |
| $\beta$ | : | Represents the ability of a honey badger to get food ($\beta > 1$). |
| $\alpha$ | : | The density factor, computed using Equation 3.13. |
| $d_i$ | : | The distance between prey and agent $i$. |
| $F$ | : | A term used to modify the search direction and avoid local optima. |
| $r_3$, $r_4$, $r_5$, and $r_6$ | : | Random number drawn from the range [0,1]. |

**C- Exploitation phase** The exploitation phase is depicted by honey badgers' behaviours as they approach beehives, when following honey guide birds. It it is mathematically represented by Equation 3.11.

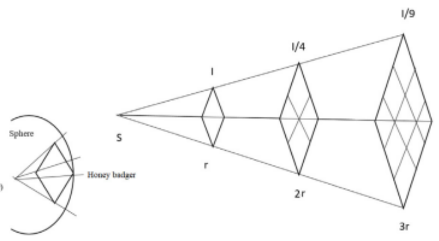


Figure 3.3: The Inverse Square Law [57].

$$X_i^{t+1} = X_{best} + F \times r_7 \times \alpha \times d_i \tag{3.11}$$

where the different terms of Equation 3.11 are the same as Equation 3.10, and $r_7$ is a random number drawn from the range $[0, 1]$.

**D- Smell intensity** The smell intensity $I_i$ relies on two factors: the prey energy and the distance between a honey badger and prey. If the smell concentration is strong, the motion of the honey badger will be fast and vice versa. This concept is modeled using the Inverse Square Law [70] and is shown in Figure 3.3. Finally, Equation 3.12 is utilized to compute the smell intensity used in the exploration phase.

$$I_i = r_2 \frac{(X_i^t - X_i^{t+1})^2}{4\pi d_i^2} \tag{3.12}$$

where $r_2$ is a random number drawn from the range $[0, 1]$, and $d_i$ is the distance between prey and agent $i$.

**E- Density factor** The density factor $\alpha$ controls the balancing between exploration and exploitation phases. It ensures the smooth transition form exploration to exploitation. Equation 3.13 is used for this purpose.

$$\alpha = C \times \exp\left(\frac{-t}{T_{\max}}\right) \tag{3.13}$$

where $C$ is a constant normally greater than or equals 1, and $T_{\max}$ is the maximum number of iterations.

### Artificial Jellyfish Search Optimizer

Jellyfish, also known as medusae, are amazing sea creatures that live in different depths and temperatures of water around the world. They come in a wide variety of shapes, sizes, and colours. They are made from a soft, translucent, and gelatinous substance. They usually have a bell-shaped body that gives them a distinctive appearance. The size and shape of the bell can vary greatly among different jellyfish species. Besides, they have a wide range of adaptation mechanisms to their oceanic environment. These mechanisms include specialized structures for hunting prey, such as long tentacles armed with stinging cells called nematocysts[26]. Jellyfish have two hunting strategies: passive swarming and

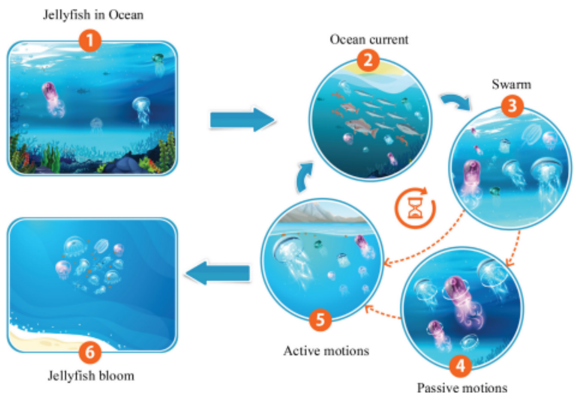


Figure 3.4: The behaviour of jellyfish in the ocean [26].

active swarming. In the former, a specific jellyfish follows the individuals of the school. In the later, a given jellyfish hunts alone. When jellyfish are not hunting, they are led by the oceanic current. The behaviour of jellyfish is shown in Figure 3.4. The mathematical formulation of exploration and exploitation concepts related to JS are explained in the following sections.

**A- Initialization phase** The initial population in the majority of metaheuristic algorithms is generally initialized in a random way. This method has two major drawbacks: slow convergence and tendency to get trapped in local optima due to reduced population diversity. In order to increase diversity from the initial population, the JS optimizer tested several chaotic maps [116, 25, 72], and concluded that the logistic map [47] gives a more diverse initial population compared to random initialization and reduces the premature convergence problem [25, 74]. Equation 3.14 is used to initialize the first population.

$$X_{i+1} = \eta X_i(1 - X_i) \quad , \quad 0 \leq X_0 \leq 1 \tag{3.14}$$

Where

| | | |
|---|---|---|
| $X_i$ | : | The logistic chaotic value of the $i^{th}$ jellyfish's location. |
| $X_0$ | : | Used to generate the initial population of jellyfish: $X_0 \in [0,1]$ and $X_0 \notin \{0, 0.25, 0.5, 0.75, 1\}$. |
| $\eta$ | : | Set to 4.0. |

**B- Exploration phase** Jellyfish follow the ocean currents to conserve energy and move quickly. In the JS optimizer, this refers to exploring the search space and generating new random candidate solutions. Hence, jellyfish updated their positions using Equations 3.15 and 3.16.

$$X_i^{t+1} = X_i^t + rand(0,1) \times \overrightarrow{trend} \tag{3.15}$$

$$\overrightarrow{trend} = X^* - \beta \times rand(0,1) \times \mu \tag{3.16}$$

Where

| | | |
|---|---|---|
| $X_i^{t+1}$ and $X_i$ | : | Locations of agent $i$ at iterations $t+1$ and $t$. |
| $X^*$ | : | The location of the current best solution in the swarm. |
| $\beta$ | : | A distribution coefficient . |
| $\mu$ | : | The mean location of all jellyfish |

**C- Exploitation phase** Jellyfish move inside the swarm to search for food. In the JS algorithm, this refers to exploiting the search space. There are two behaviours: either a jellyfish moves towards the best solution found so far, or it moves randomly looking for food on its own. These mechanisms are called passive motion and active motion, respectively [45, 125].

1. **Passive motion:** The different agents in a specific population update their locations using Equation 3.17.

$$X_i^{t+1} = X_i^t + \gamma \times rand(0,1) \times (u_b - l_b) \tag{3.17}$$

   where $l_b$ and $u_b$ are the lower and upper bounds of the search space, and $\gamma > 0$ is a motion coefficient which is related to the length of motion around jellyfish's locations.

2. **Active motion:** To simulate this kind of movement, we randomly select two locations $i$ and $j$. If the solution at the location $i$ is better than $j$, then $j$ moves towards $i$ and vice versa. Hence, the different agents in a specific population update their locations using Equation 3.18, and Equation 3.19. This exploitation mechanism has proven to be very efficient, according to the work reported in [46].

$$X_i^{t+1} = X_i^t + rand(0,1) \times \overrightarrow{D} \tag{3.18}$$

$$\overrightarrow{D} = \begin{cases} X_i^t - X_j^t & , \text{ if } f(X_i^t) < f(X_j^t) \\ X_j^t - X_i^t & , \text{ if } f(X_i^t) \geq f(X_j^t) \end{cases} \tag{3.19}$$

   where the variable $\overrightarrow{D}$ is used to determine the direction of jellyfish and $f(.)$ is the objective function to be optimized.

**D- Time control mechanism** A time control model is used in the JS algorithm to switch between exploration and exploitation, as well as between passive to active motions. The time control mechanism is represented by Equation 3.21 and depicted by Figure 3.5.

$$c(t) = \left| \left( 1 - \frac{t}{T_{\max}} \times (2 \times rand(0,1) - 1 \right) \right| \tag{3.20}$$

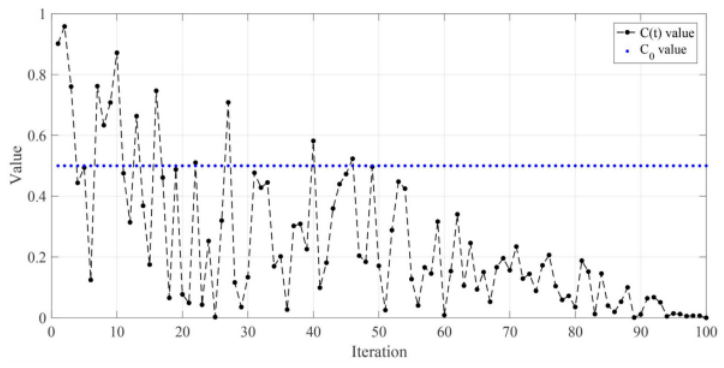


Figure 3.5: The graphical representation of the used time control mechanism.

### 3.2.2 Mathematical model of the proposed BHJO

By combining the previous algorithms, we aim at designing a more robust and efficient hybrid metaheuristic algorithm that can take the advantages and strengths of each one. Hybrid nature-inspired approaches involve combining different optimization algorithms in order to improve their performance. There are two manners of hybridization, namely high-level and low-level [107]. In the first type, the algorithms to be hybridized are put one after another; whereas, in the second type, the similar steps of each algorithm are merged.

It is worth mentioning out that the exploration and exploitation abilities of the above-mentioned algorithms – i.e., Beluga Whale Optimization (BWO) [131], the Honey Badger algorithm (HBA) [57], and the Jellyfish Search Optimizer (JS) [26] – are well balanced. However, according to the results reported in [131], [57], [26], it was observed that: i) the exploitation of BWO is better than its exploration; ii) the exploitation of HBA is better than its exploration; and the exploration of JS is better than its exploitation.

In our algorithm, we have adopted a low-level hybridization to design the BHJO. That is to say, on the one side, the exploration phases of BWO, HBA, and JS are combined to design the exploration of BHJO; and one the other side, the exploitation phases of BWO and HBA are used to design the exploitation of BHJO. The key to success is often in finding the right combination to create an algorithm that can effectively balance the trade-off between exploration and exploitation. The exploration consists in generating new solutions and discovering promising regions, and exploitation aims at improving existing solutions and avoiding local optima [105].

#### Initialization phase

To improve the diversity of the population, the logistic map has been used to generate chaotic sequences [80]. This can enhance the global search ability of BHJO by allowing it to explore the search space more effectively. Equation 3.14 is used to create the first population of our algorithm.

### Exploration phase

The exploration phase of our hybrid algorithm is composed of two strategies, namely first strategy and second strategy. In the first strategy, we combined the exploration of HBA and JS. In the second strategy, we used the exploration of BWO. To switch between the first and second strategies, we use the following technique. First, we assume a uniform random number drawn from the range $[0, 1]$. Next, if the generated number is less than 0.5, we apply the first strategy; otherwise, we apply the second strategy.

**First strategy** To perform the first strategy of exploration, we used Equations 3.10 and 3.16 to propose Equation 3.21.

$$X_i^{t+1} = X_{best}\times r_2 + F\times I\times\overrightarrow{trend}\times d_i\times B_f\times \mid (cos(2\times\pi\times r_4)\times(1 - cos(2\times\pi\times r_5))) \mid \quad (3.21)$$

**Second strategy** To perform the second strategy of exploration, we used Equation 3.1.

### Exploitation phase

Similarly, the exploitation phase of our hybrid algorithm is composed of two strategies, namely first strategy and second strategy. In the first strategy, we used the exploitation of BWO. In the second strategy, we used the exploitation of HBA. To switch between the first and second strategies, we use the following technique. First, we assume a uniform random number drawn from the range $[0, 1]$. Next, if the generated number is less than $1 - B_f$, we apply the first strategy; otherwise, we apply the second strategy, where $B_f$ is computed using Equation 3.22.

**First strategy** To perform the first strategy of exploitation, we used Equation 3.11.

**Second strategy** To perform the second strategy of exploitation, we used Equation 3.2.

### Balance between exploration and exploitation

In most metaheuristic algorithms, the optimizers explore and then exploit the search space. This trend has many disadvantages. For instance, during the exploration phase, the algorithm searches a wide range of solutions to identify promising areas in the search space, and it might not find optimal or near-optimal solutions. Consequently, the exploitation phase, as it focuses on intensifying the search in selected regions, can cause the algorithm to prematurely converge to a suboptimal solution, without exploring other regions of the search space that may contain better solutions. Therefore, we propose a mathematical technique that allows exploration in a wide range of solutions to identify promising areas in the field of research and intensify research in promising areas at the same time to ensure an effective balance between exploration and exploitation and finding optimal solutions.

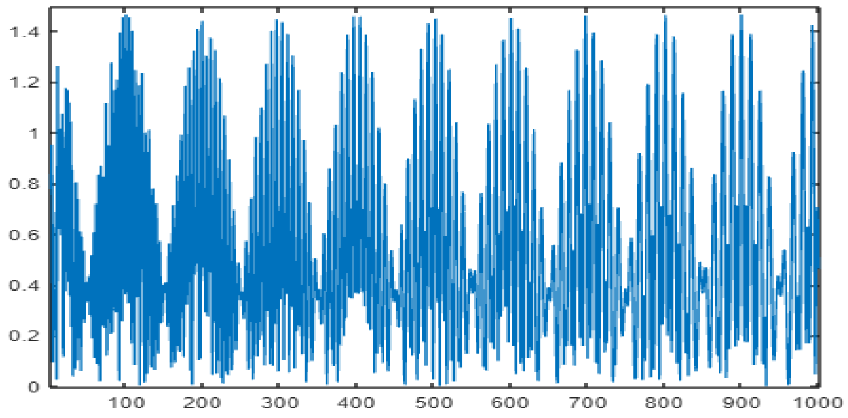


Figure 3.6: The curve of balancing between exploration and exploitation phases.

Figure 3.6 depicts the curve of balancing between exploration and exploitation, which is defined by Equation 3.22.

$$B_f = \mid ((C \times r_1 - 1) \times (1 + m \times cos(2 \times \pi \times f \times (\frac{t}{t_{max}})) \times cos(2 \times \pi \times f \times \sqrt{t}))) \mid \tag{3.22}$$

where $C$, $f$, and $m$ are constant numbers set to 2, 5, and -3, respectively, and $r_1$ is a uniform random number.

### Local optimum avoidance

To avoid getting stuck in local optimum we used Equation 3.6.

### Combined Opposition-Based Learning

The Combined Opposition-Based Learning (COBL) [117] is a novel research strategy that integrates two enhanced variants of opposition-based learning, namely the Lens Opposition-Based Learning [42] and Random Opposition-Based Learning [110]. The COBL is known for its ability to increase the convergence speed of metaheuristics [117]. It achieves this by utilizing the concept of lens opposition, which involves creating an opposite solution based on the lens formed between the current solution and the global best solution. By incorporating COBL in our hybrid algorithm, this would help escaping local optima and avoiding the premature convergence problem because the COBL focuses on promoting the population diversity. The COBL is given by Equation 3.23. It is worth emphasizing that Equation 3.23 is applied each time a candidate solution is created or updated.

$$\bar{X}_{COBL} = \begin{cases} lb_i + ub_i - rand \times X_i & , \quad if\ q < 0.5 \\ \frac{lb_i + ub_i}{2} + \frac{lb_i + ub_i}{2k} - \frac{X_i}{k} & , \quad otherwise \end{cases} \tag{3.23}$$

### 3.2.3 Pseudocode and time complexity of the proposed BHJO

Algorithm 1 describes the different steps of the proposed hybrid algorithm. Theoretically, the computational complexity of BHJO is an important metric to assess its performance. It includes three processes: fitness evaluation, initialization of the first population, and updating of search agents. First, the time complexity of the fitness evaluation is $O(D) \backsim O(n)$. Second, the time complexity of the initialisation of the first population is $O(\max\{N_{pop}, N_{pop}D\}) \backsim O(n^2)$. Finally, the time complexity of the swarming behaviour is $O(T_{max}N_{pop}D) \backsim O(n^3)$. Form the previous time complexities, we conclude that the time complexity of BHJO is $O(n^3)$.

## 3.3 Experimental results and discussion

The proposed BHJO algorithm is implemented using MATLAB version R2016a. The laptop's configuration is Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz, 2601MHz processor and 8.0 GB of RAM. The laptop's OS is Microsoft Windows 10 Enterprise 64–bit. The numerical efficiency of BHJO is evaluated on 40 challenging benchmark test functions of different complexity. To validate the performance of BHJO, the obtained results are compared to seven state-of-the-art optimization algorithms; namely, BWO [131], HBA [57], JS [26], WOA [89], MFO [87], PSO [73], and HHO [59]. These algorithms are relatively new, but they have shown outstanding performance when compared to many optimizers. The population size for all the optimizers is set to 30. In addition, each algorithm was run 30 times to minimize the variance, and each run was iterated 1000 iterations to ensure the validity of the law of large numbers. we discuss the performance of BHJO on the selected benchmark test functions.

To evaluate the effectiveness of BHJO, a collection of 40 well-established benchmark test functions from the existing literature is selected [35, 123]. This set covers four functions' types: unimodal, multimodal, hybrid, and composition functions. Tables 3.1, 3.2, 3.3, 3.4, and 3.5 give an overview on the details of each function: mathematical expression, dimension, optimum, etc. On the one hand, unimodal functions ($\{F_1, \ldots, F_7\}$) have only one global optimum, and they are used to assess the exploitation ability of optimization methods; while, on the other hand, multimodal functions $\{F_8, \ldots, F_{20}\}$) possess several local optimums, and they are employed to evaluate the exploration ability of optimization methods, i.e., the avoidance of local optimums and the prevention of the premature convergence. Finally, the hybrid and composition functions ($\{F_{21}, \ldots, F_{40}\}$) are know to be very challenging and hard functions. They have a large number of local optimums, and they are used to evaluate the well-balance between the exploration and the exploitation in metaheuristic algorithms.

In this paragraph, we present the parameter settings employed in our hybrid algorithm, as well as the parameter settings used for the comparative study involving other algorithms. Regarding the comparative study, we considered seven state-of-the-art metaheuristic algorithms: BWO [131], HBA [57], JS [26], WOA [89], MFO [87], PSO [73], and HHO [59]. In the comparative study, the parameter settings for the algorithms utilized were obtained from their respective papers. These settings were chosen based on the

**Algorithm 1:** The pseudo-code of the BHJO algorithm.

```
Input: Initialize the different parameters of BHJO.
Input: Define the objective function to be minimized $f(X)$.
Initialize the pseudo-time $t \leftarrow 0$
/* Initialization of the first population                              */
Initialize the first population $X_i^t$ where $i \in \{1, 2, \dots, N_{pop}\}$ using Equation 3.14
for $i \leftarrow 1$to $N_{pop}$ do
    Compute $\bar{X}_i^t$ the opposite of $X_i^t$ using Equation 3.23
    $X_i^t$ takes the location that minimizes $f(X)$ among the locations $\bar{X}_i^t$ and $X_i^t$
    Check the boundaries of the solution $X_i^t$
end
/* Swarming behaviour of BHJO                                          */
$t \leftarrow t + 1$
while $(t \leq T_{max})$ do
    Compute the factor $B_f$ using Equation 3.22
    for $i \leftarrow 1$ to $N_{pop}$ do
        /* Exploration of the search space                             */
        if $(B_f \geq 0.5)$ then
            /* First strategy                                          */
            if $(rand < 0.5)$ then
                Compute the position $X_i^t$ using Equation 3.21
            end
            /* Second strategy                                         */
            else
                Compute the position $X_i^t$ using Equation 3.1
            end
        end
        /* Exploitation of the search space                            */
        else
            /* First strategy                                          */
            if $Rand(0, 1) > (1 - B_f)$ then
                Compute the position $X_i^t$ using Equation 3.2
            end
            /* Second strategy                                         */
            else
                Compute the position $X_i^t$ using Equation 3.11
            end
        end
        Compute $\bar{X}_i^t$ the opposite of $X_i^t$ using Equation 3.23
        Compute $\bar{X}_i^{t-1}$ the opposite of $X_i^{t-1}$ using Equation 3.23
        $X_i^t$ takes the location that minimizes $f(X)$ among the locations $\bar{X}_i^t$, $X_i^t$, $\bar{X}_i^{t-1}$ and
          $X_i^{t-1}$
        Check the boundaries of the solution $X_i^t$
    end
    /* Local optimums avoidance                                        */
    Compute the parameter $W_f$ using Equation 3.8
    for $i \leftarrow 1$to $N_{pop}$ do
        if $B_f \leq W_f$ then
            Compute the position $X_i^t$ using Equation 3.6
            Compute $\bar{X}_i^t$ the opposite of $X_i^t$ using Equation 3.23
            $X_i^t$ takes the location that minimizes $f(X)$ among the locations $\bar{X}_i^t$ and $X_i^t$
            Check the boundaries of the solution $X_i^t$
        end
    end
    $t \leftarrow t + 1$
end
Output the best solution
```

Table 3.1: The description of unimodal functions.

| ID | Mathematical Expression | Dimension | Range | Global Optimum |
|---|---|---|---|---|
| $F_1$ | $f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |
| $F_2$ | $f(\mathbf{x}) = (\sum_{i=1}^{D} x_i^2)^2$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |
| $F_3$ | $f(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |
| $F_4$ | $f(\mathbf{x}) = \max_i\{\lvert x_i\rvert, 1 < i < n\}$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |
| $F_5$ | $f(\mathbf{x}) = \sum_{i=1}^{D} i x_i^2$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |
| $F_6$ | $f(\mathbf{x}) = \sum_{j=1}^{D-4}[(x_{i-1} - 10x_i)^2$ <br> $+5(x_{i+1} - x_{i+2})^2$ <br> $+(x_i - xi + 1)^4$ <br> $+10(x_{i-1} - x_{i+2})^4]$ | $\{30, 50, 100, 1000\}$ | $[-4, 5]$ | 0 |
| $F_7$ | $f(\mathbf{x}) = \sum_{i=1}^{D} x_i^{10}$ | $\{30, 50, 100, 1000\}$ | $[-100, 100]$ | 0 |

Table 3.2: The description of multimodal functions.

| ID | Mathematical Expression | Dimension | Range | Global Optimum |
|---|---|---|---|---|
| $F_8$ | $f(\mathbf{x}) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$ | $\{30, 50, 100, 1000\}$ | $[-500, 500]$ | 0 |
| $F_9$ | $f(\mathbf{x}) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$ | $\{30, 50, 100, 1000\}$ | $[-5.12, 5.12]$ | 0 |
| $F_{10}$ | $f(\mathbf{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$ <br> $- \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + \exp(1)$ | $\{30, 50, 100, 1000\}$ | $[-32, 32]$ | 0 |
| $F_{11}$ | $f(\mathbf{x}) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}})$ | $\{30, 50, 100, 1000\}$ | $[-600, 600]$ | 0 |
| $F_{12}$ | $f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$ | $\{30, 50, 100, 1000\}$ | $[-10, 10]$ | 0 |
| $F_{13}$ | $f(\mathbf{x}) = 0.1(\sin^{(} 3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2$ <br> $[1 + \sin^{(} 3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^{(} 2\pi x_i)]) +$ <br> $\sum_{i=1}^{D} u(x_i, 5, 100, 4)$ | $\{30, 50, 100, 1000\}$ | $[-50, 50]$ | 0 |
| $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & , \ if x_i > a \\ 0 & , \ if -a \leq x_i \leq a \\ k(-x_i - a)^m & , \ x_i < -a \end{cases}$ | | | | |

Table 3.3: The description of fixed-multimodal functions.

| ID | Mathematical Expression | Dimension | Range | Global Optimum |
|---|---|---|---|---|
| $F_{14}$ | $f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ | 2 | $[-10, 10]$ | 0 |
| $F_{15}$ | $f(\mathbf{x}) = -\sum_{j=1}^{m}(\sum_{i=1}^{4}(x_j - C_{ji})^2 + \beta_i)^{-1}$ | 4 | $[0, 10]$ | 0 |
| $F_{16}$ | $f(\mathbf{x}) = -\frac{0.001}{[0.001^2+(x_1-0.4x_2-0.1)^2]}$ $-\frac{0.001}{[0.001^2+(2x_1+x_2-1.5)^2]}$ | 2 | $[-500, 500]$ | $-2000$ |
| $F_{17}$ | $f(\mathbf{x}) = (2x_1^3x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)$ | 2 | $[-500, 500]$ | 0 |
| $F_{18}$ | $f(\mathbf{X}) = \sum_{i=1}^{11}[\alpha_i \frac{x_1(\beta_i^2+\beta_i x_2)}{\beta_i^2} + \beta_i x_3 + x_4]^2$ | 4 | $[-5, 5]$ | 3 |
| $F_{19}$ | $f(\mathbf{x}) = 100(x_1 - x_2^2)^2 + (1 - x_1)^2$ $+90(x_4 - x_3^2)^2 + (1 - x_3)^2$ $+10.1(x_2 - 1)^2 + (x_4 - 1)^2$ $+19.8(x_2 - 1)(x_4 - 1)$ | 4 | $[-10, 10]$ | 0 |
| $F_{20}$ | $f(\mathbf{x}) = 0.5x_1^2 + 0.5[1 - \cos 2x_1] + x_2^2$ | 2 | $[-500, 500]$ | 0 |

$m = 10$

$\beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T$

$$C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$$

Table 3.4: The description of hybrid functions.

| ID | ID in CEC | Dimension | Range | Global Optimum |
|---|---|---|---|---|
| $F_{21}$ | Hybrid Function $F_{17}$ (N=4) (CEC2017) | 10 | $[-100, 100]$ | 1700 |
| $F_{22}$ | Hybrid Function $F_{14}$ (N=4) (CEC2017) | 10 | $[-100, 100]$ | 1400 |
| $F_{23}$ | Hybrid Function $F_{20}$ (N=6) (CEC2017) | 10 | $[-100, 100]$ | 2000 |
| $F_{24}$ | Hybrid Function $F_{11}$ (N=3) (CEC2017) | 10 | $[-100, 100]$ | 1100 |
| $F_{25}$ | Hybrid Function $F_6$ (N = 3) (CEC 2020) | 10 | $[-100, 100]$ | 1700 |
| $F_{26}$ | Hybrid Function $F_8$ (N = 5) (CEC2021) | 10 | $[-100, 100]$ | 2100 |
| $F_{27}$ | Hybrid Function $F_5$ (N = 5) (CEC2021) | 10 | $[-100, 100]$ | 2100 |
| $F_{28}$ | Hybrid Function $F_6$ (N = 5) (CEC2021) | 10 | $[-100, 100]$ | 2200 |
| $F_{29}$ | Hybrid Function $F_7$ (N = 4) (CEC2020) | 10 | $[-100, 100]$ | 1600 |
| $F_{30}$ | Hybrid Function $F_{19}$ (N = 6) (CEC2017) | 10 | $[-100, 100]$ | 1900 |

Table 3.5: The description of composition functions.

| ID | ID in CEC | Dimension | Range | Global Optimum |
|---|---|---|---|---|
| $F_{31}$ | Composition Function $F_{20}$ (N=3) (CEC 2017) | 10 | $[-100, 100]$ | 2100 |
| $F_{32}$ | Composition Function $F_{28}$ (N=3) (CEC 2017) | 10 | $[-100, 100]$ | 2900 |
| $F_{33}$ | Composition Function $F_{23}$ (N=4) (CEC 2017) | 10 | $[-100, 100]$ | 2200 |
| $F_{34}$ | Composition Function $F_{24}$ (N=4) (CEC 2017) | 10 | $[-100, 100]$ | 2300 |
| $F_{35}$ | Composition Function $F_8$ (N = 3) (CEC 2020) | 10 | $[-100, 100]$ | 2300 |
| $F_{36}$ | Composition Function $F_9$ (N = 4) (CEC 2020) | 10 | $[-100, 100]$ | 2200 |
| $F_{37}$ | Composition Function $F_{10}$ (N = 5) (CEC 2020) | 10 | $[-100, 100]$ | 2500 |
| $F_{38}$ | Composition Function $F_{12}$ (N = 6) (CEC 2021) | 10 | $[-100, 100]$ | 2700 |
| $F_{39}$ | Composition Function $F_{11}$ (N = 5) (CEC 2021) | 10 | $[-100, 100]$ | 2600 |
| $F_{40}$ | Composition Function $F_{12}$ (N = 6) (CEC 2021) | 10 | $[-100, 100]$ | 2700 |

authors' recommendations and empirical evaluations, ensuring consistency and comparability with the existing literature. By adopting the parameter settings from the original papers, we aimed to establish a reliable basis for evaluating and comparing the performance of our algorithm against the established state-of-the-art methods. The values used in the BHJO algorithm are taken from the three algorithms utilized for its conception. By employing these specific parameter settings, we aim to ensure fair and comprehensive comparisons among the algorithms in our study, ultimately providing valuable insights into their performance and effectiveness. Table 3.6 provides a detailed overview of these parameters and their respective values. It is worth mentioning out that all the optimization algorithms share a population size of 30, and the number of iterations is set to 1000; moreover, each algorithm is run 30 times to ensure reliable and consistent results.

Tables 3.7, 3.8, 3.9, 3.10, 3.12, 3.13, and 3.14 present the statistical results of our comparative study, providing essential insights into the performance of the algorithms under evaluation. These tables encompass various statistical measures, including the average and standard deviation. The average values offer a representative measure of the algorithm's overall performance, while the standard deviation provides an indication of the degree of variability in the results. By reporting these comprehensive statistical measures, we aim to offer a comprehensive and informative evaluation of the comparative study, facilitating a deeper understanding of the algorithms' performance and enabling meaningful comparisons between the methods. Schemes are shown in Figure 3.8 Preliminary note Is that the BHJO algorithm can reveal great results compared to the other seven algorithms.

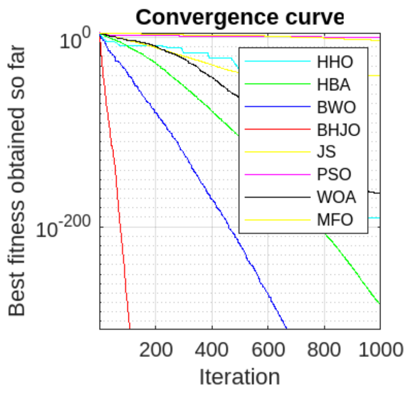
We perform the Friedman's test to analyze repeated measures data. If the Friedman's test reveals a significant difference among the groups, then post hoc tests can be performed to determine which groups differ from each other. One commonly used post hoc test for the Friedman's test is the Dunn's test. The Dunn's test is a pairwise comparison test that compares all possible pairs of groups and identifies significant differences be-

Table 3.6: Parameters' values of the algorithms used for the comparative study.

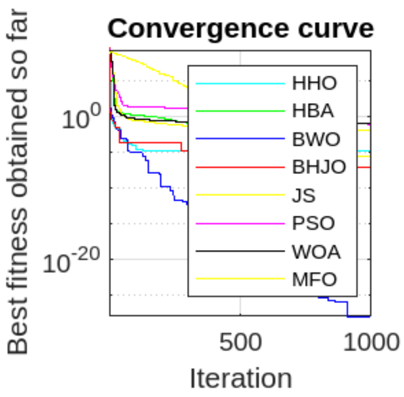
| **Algorithm** | **Parameter** | **Vaule** |
|---|---|---|
| BHJO | Probability of whale fall decreased at interval $W_f$ | $[0.1, 0.05]$ |
| | $m$ | $m = -3$ |
| | $f$ | $f = 5$ |
| | $C$ | $C = 2$ |
| | $\beta$ | $\beta = 3$ |
| BWO | Probability of whale fall decreased at interval $W_f$ | $[0.1, 0.05]$ |
| HBA | $\beta$ (the ability of a honey badger to get food) | $\beta = 6$ |
| | $C$ | $C = 2$ |
| JS | $\beta$ (Spatial distribution) | $\beta = 3$ |
| PSO | Inertia weight linearly decreased at interval | $[0.9, 0.2]$ |
| | Cognitive and social constant | $c1 = 2, c2 = 2$ |
| HHO | Probability thresholds of escaping, escaping energy | $0.5, 0.5$ |
| WOA | Probability of encircling mechanism, spiral factor | $0.5, 1$ |
| MFO | Convergence constant | $a = [-2, -1]$ |
| | Spiral factor | $b = 1$ |

Table 3.7: Statistical results obtained for the unimodal and multimodal functions with $D = 30$.

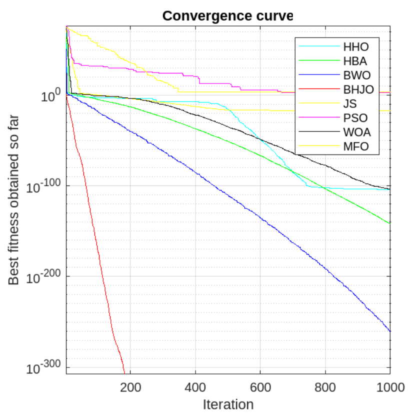
| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | AVG | **0.00E+00** | **0.00E+00** | 1.48E-277 | 2.64E-39 | 2.47E-01 | 1.64E-150 | 3.05E-183 | 1.00E+03 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 6.46E-39 | 1.52E-01 | 6.92E-150 | 0.00E+00 | 3.051E+03 |
| $F_2$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 9.26E-78 | 1.23E-01 | 4.45E-299 | 0.00E+00 | 1.66E+07 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 5.56E-78 | 2.60E-01 | 0.00E+00 | 0.00E+00 | 3.79E+07 |
| $F_3$ | AVG | **0.00E+00** | 3.94E-260 | 4.68E-145 | 1.67E-19 | 4.90E+00 | 1.90E-101 | 7.06E-91 | 4.40E+02 |
| | STD | **0.00E+00** | 0.00E+00 | 1.99E-144 | 3.00E-19 | 2.64E+01 | 1.00E-100 | 3.87E-90 | 2.35E+02 |
| $F_4$ | AVG | **0.00E+00** | 1.77E-252 | 1.83E-117 | 1.032E-15 | 1.49E+00 | 4.06E+01 | 1.00E-90 | 6.72E+01 |
| | STD | **0.00E+00** | 0.00E+00 | 6.94E-117 | 7.82E-16 | 2.74E-01 | 3.05E+01 | 2.31E-90 | 6.87E+00 |
| $F_5$ | AVG | **0.00E+00** | **0.00E+00** | 9.28E-274 | 2.91E-38 | 2.64E+00 | 6.50E-151 | 3.05E-185 | 6.866E+04 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 8.14E-38 | 1.22E+00 | 3.01E-150 | 0.00E+00 | 8.19E+04 |
| $F_6$ | AVG | 2.36E-223 | **0.00E+00** | 4.61E-174 | 1.06E-06 | 7.17E+01 | 4.57E-07 | 8.34E-187 | 1.057E+03 |
| | STD | 0.00E+00 | **0.00E+00** | 0.00E+00 | 2.62E-06 | 4.12E+01 | 1.64E-06 | 0.00E+00 | 1.38E+03 |
| $F_7$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.43E-163 | 2.56E+00 | **0.00E+00** | **0.00E+00** | 2.372E+04 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 3.75E+00 | **0.00E+00** | **0.00E+00** | 6.44E+04 |
| $F_8$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.12E+01 | 1.10E+01 | 6.48E-01 | **0.00E+00** | 6.896E+00 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 6.40E-01 | 6.56E-01 | 1.63E+00 | **0.00E+00** | 1.40E+00 |
| $F_9$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.37E+01 | 1.08E+02 | **0.00E+00** | **0.00E+00** | 1.57E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 6.57E+00 | 2.98E+01 | **0.00E+00** | **0.00E+00** | 3.90E+01 |
| $F_{10}$ | AVG | **4.44E-16** | **4.44E-16** | 1.99E+00 | 5.06E-15 | 1.03E+00 | 4.11E-15 | **4.44E-16** | 1.630E+02 |
| | STD | **0.00E+00** | **0.00E+00** | 6.07E+00 | 1.65E-15 | 5.65E-01 | 2.18E-15 | **0.00E+00** | 5.83E+00 |
| $F_{11}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 2.64E-02 | 2.06E-03 | **0.00E+00** | 1.50E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.53E-02 | 1.13E-02 | **0.00E+00** | 4.14E+01 |
| $F_{12}$ | AVG | **0.00E+00** | **0.00E+00** | 3.80E-279 | 1.75E-40 | 3.75E-01 | 2.02E-151 | 1.71E-180 | 1.66E+02 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 2.31E-40 | 2.42E-01 | 1.10E-150 | 0.00E+00 | 1.60E+02 |
| $F_{13}$ | AVG | 2.28E-08 | **8.98E-25** | 1.18E-01 | 9.57E-03 | 1.05E-01 | 1.52E-01 | 2.02E-05 | 2.73E+07 |
| | STD | 2.68E-08 | **2.72E-24** | 1.66E-01 | 1.27E-02 | 6.74E-02 | 1.18E-01 | 2.88E-05 | 1.04E+08 |



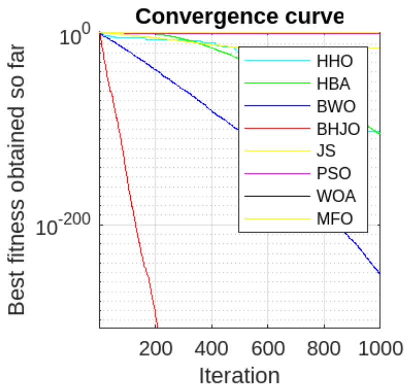
(a) Test function F1.



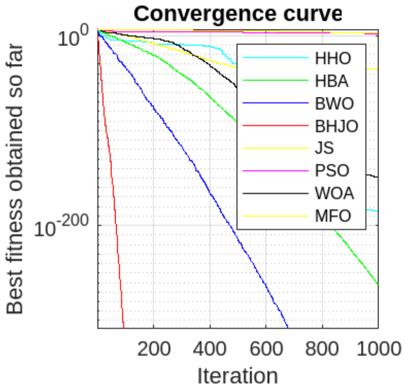
(b) Test function F2.



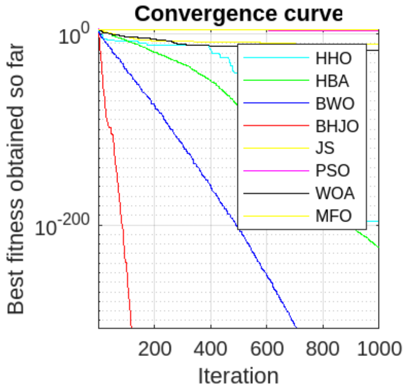
(c) Test function F3.



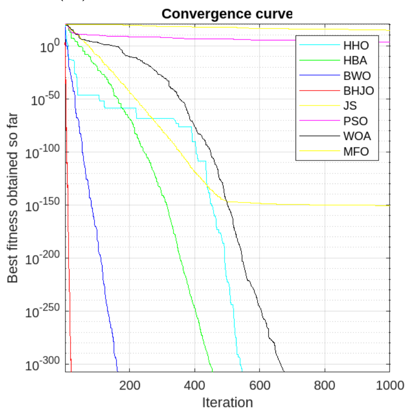
(d) Test function F4.



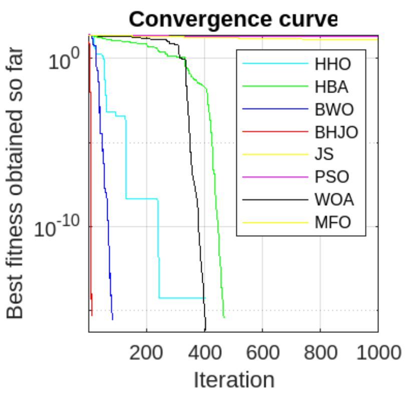
(e) Test function F5



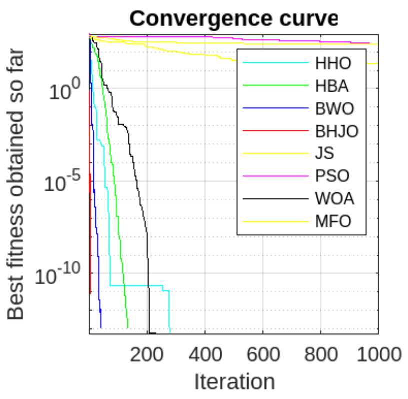
(f) Test function F6



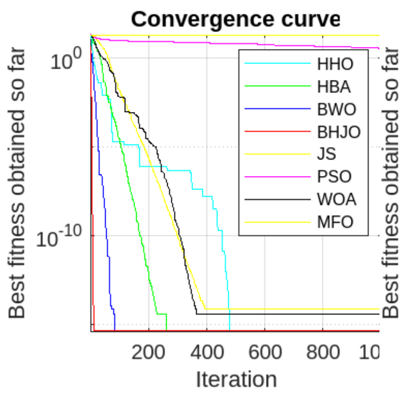
(g) Test function F7

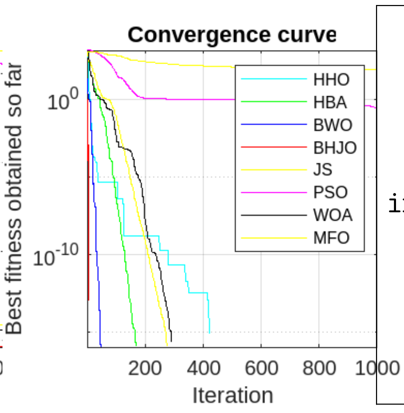


(h) Test function F8



(i) Test function F9





images/58.png

(j) Test function F10 (k) Test function F11 (l) Test function F12

Figure 3.7: Scalability analysis ($F_1 - F_{12}$) of the BHJO algorithm (1/2) .

Table 3.8: Statistical results obtained for the unimodal and multimodal functions with $D = 50$.

| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | AVG | **0.00E+00** | **0.00E+00** | 2.25E-263 | 2.43E-36 | 7.83E+00 | 2.46E-148 | 2.80E-189 | 4.03E+03 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 4.19E-36 | 2.71E+00 | 8.84E-148 | 0.00E+00 | 5.61E+03 |
| $F_2$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 2.551E-71 | 5.43E+01 | 1.65E-297 | **0.00E+00** | 1.09E+08 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.17E-70 | 3.15E+01 | **0.00E+00** | 0.00E+00 | 1.90E+08 |
| $F_3$ | AVG | **0.00E+00** | 8.52E-259 | 1.55E-137 | 3.32E-18 | 2.24E+02 | 9.69E-98 | 3.71E-94 | 6.61E+02 |
| | STD | **0.00E+00** | 0.00E+00 | 5.49E-137 | 2.23E-18 | 1.61E+02 | 5.30E-97 | 1.82E-93 | 2.85E+02 |
| $F_4$ | AVG | **0.00E+00** | 1.22E-245 | 4.31E-105 | 7.48E-15 | 3.16E+00 | 5.50E+01 | 1.09E-93 | 8.34E+01 |
| | STD | **0.00E+00** | 0.00E+00 | 6.94E-117 | 4.87E-15 | 3.67E-01 | 3.32E+01 | 5.29E-93 | 4.63E+00 |
| $F_5$ | AVG | **0.00E+00** | **0.00E+00** | 4.64E-259 | 2.74E-35 | 1.49E+02 | 8.35E-149 | 6.46E-179 | 2.49E+05 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 2.93E-35 | 5.62E+01 | 3.78E-148 | 0.00E+00 | 1.82E+05 |
| $F_6$ | AVG | 2.98E-261 | **0.00E+00** | 4.61E-174 | 3.81E-10 | 6.73E+02 | 2.76E-10 | 3.11E-179 | 4.23E+03 |
| | STD | 0.00E+00 | **0.00E+00** | 0.00E+00 | 1.82E-09 | 1.38E+02 | 1.51E-09 | 0.00E+00 | 2.71E+03 |
| $F_7$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 2.84E-153 | 6.03E+03 | **0.00E+00** | **0.00E+00** | 2.39E+14 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 8.00E-153 | 8.91E+03 | **0.00E+00** | **0.00E+00** | 6.26E+14 |
| $F_8$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 2.03E+01 | 1.94E+01 | 4.62E-01 | **0.00E+00** | 1.30E+01 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 7.16E-01 | 9.36E-01 | 1.86E+00 | **0.00E+00** | 2.04E+00 |
| $F_9$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.16E+01 | 3.10E+02 | 3.78E-15 | **0.00E+00** | 3.187E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.58E+01 | 5.98E+01 | 2.07E-14 | **0.00E+00** | 4.58E+01 |
| $F_{10}$ | AVG | **4.44E-16** | **4.44E-16** | 3.07E+00 | 6.48E-15 | 1.03E+00 | 2.69E-15 | **4.44E-16** | 1.90E+01 |
| | STD | **0.00E+00** | **0.00E+00** | 6.07E+00 | 1.65E-15 | 3.34E-01 | 2.37E-15 | **0.00E+00** | 1.33E+00 |
| $F_{11}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.73E-01 | 8.19E-03 | **0.00E+00** | 8.806E+01 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 4.03E-02 | 2.54E-02 | **0.00E+00** | 7.64E+01 |
| $F_{12}$ | AVG | **0.00E+00** | **0.00E+00** | 3.74E-264 | 2.72E-37 | 8.58E+00 | 8.13E-149 | 1.86E-175 | 1.890E+02 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 5.65E-37 | 3.88E+00 | 2.60E-148 | 0.00E+00 | 2.85E+02 |
| $F_{13}$ | AVG | 1.72E-07 | **7.02E-25** | 1.29E+00 | 1.62E-02 | 2.09E+00 | 6.74E-01 | 2.17E-05 | 9.56E+07 |
| | STD | 7.25E-07 | **2.78E-24** | 5.86E-01 | 1.88E-02 | 7.50E-01 | 3.44E-01 | 2.505E-05 | 3.35E+08 |

Table 3.9: Statistical results obtained for the unimodal and multimodal functions with $D = 100$.

| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | AVG | **0.00E+00** | **0.00E+00** | 2.68E-251 | 6.64E-34 | 1.03E+02 | 5.91E-143 | 2.80E-189 | 2.89E+04 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 8.56E-34 | 1.61E+01 | 3.23E-142 | 0.00E+00 | 1.20E+04 |
| $F_2$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 4.18E-67 | 1.03E+03 | 1.24E-292 | 0.00E+00 | 1.551E+09 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.05E-66 | 4.91E+03 | 0.00E+00 | 0.00E+00 | 1.41E+09 |
| $F_3$ | AVG | **0.00E+00** | 3.96E-258 | 2.27E-132 | 5.61E-17 | 3.74E+32 | 1.90E-99 | 5.49E-94 | 2.37E+03 |
| | STD | **0.00E+00** | 0.00E+00 | 4.89E-132 | 3.33E-17 | 2.05E+33 | 8.50E-99 | 2.94E-93 | 4.42E+02 |
| $F_4$ | AVG | **0.00E+00** | 2.85E-245 | 4.68E-82 | 5.00E-14 | 9.54E+00 | 7.37E+01 | 1.28E-92 | 9.35E+01 |
| | STD | **0.00E+00** | 0.00E+00 | 2.41E-81 | 2.67E-14 | 1.06E+00 | 2.43E+01 | 4.86E-92 | 2.05E+00 |
| $F_5$ | AVG | **0.00E+00** | **0.00E+00** | 9.84E-247 | 2.66E-32 | 5.08E+03 | 2.86E-147 | 6.46E-179 | 1.70E+06 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 3.16E-32 | 1.37E+03 | 1.21E-146 | 0.00E+00 | 8.16E+05 |
| $F_6$ | AVG | 3.63E-274 | **0.00E+00** | 1.31E-237 | 5.60E-14 | 1.10E+04 | 1.65E-28 | 3.11E-179 | 1.04E+04 |
| | STD | 0.00E+00 | **0.00E+00** | 0.00E+00 | 3.07E-13 | 3.37E+03 | 8.50E-28 | 0.00E+00 | 5.64E+03 |
| $F_7$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 8.31E-143 | 1.08E+07 | **0.00E+00** | **0.00E+00** | 8.40E+19 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 4.33E-142 | 5.91E+06 | **0.00E+00** | **0.00E+00** | 2.54E+19 |
| $F_8$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 4.37E+01 | 4.19E+01 | 1.04E-01 | **0.00E+00** | 3.27E+01 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 8.29E-01 | 1.07E-01 | 5.73E-01 | **0.00E+00** | 1.47E+00 |
| $F_9$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.06E+03 | **0.00E+00** | **0.00E+00** | 7.66E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.35E+02 | **0.00E+00** | **0.00E+00** | 7.02E+01 |
| $F_{10}$ | AVG | **4.44E-16** | **4.44E-16** | 6.62E-01 | 6.48E-15 | 5.38E+00 | 2.69E-15 | **4.44E-16** | 1.98E+01 |
| | STD | **0.00E+00** | **0.00E+00** | 3.63E+00 | 1.6559E-15 | 3.338E-01 | 2.3756E-15 | **0.00E+00** | 2.134E-01 |
| $F_{11}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 8.20E-01 | **0.00E+00** | **0.00E+00** | 2.606E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 9.11E-02 | **0.00E+00** | **0.00E+00** | 1.01E+02 |
| $F_{12}$ | AVG | **0.00E+00** | **0.00E+00** | 9.70E-252 | 4.04E-35 | 1.06E+02 | 5.39E-148 | 1.86E-175 | 1.65E+03 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 4.82E-35 | 1.93E+01 | 2.66E-147 | 0.00E+00 | 6.74E+02 |
| $F_{13}$ | AVG | 3.07E-07 | **4.30E-26** | 7.41E+00 | 1.86E-01 | 1.03E+02 | 1.59E+00 | 2.84E-05 | 3.70E+08 |
| | STD | 9.73E-07 | **1.46E-25** | 6.56E-01 | 7.84E-02 | 3.29E+01 | 8.06E-01 | 4.52E-05 | 3.45E+08 |

Table 3.10: Statistical results obtained for the unimodal and multimodal functions with $D = 1000$.

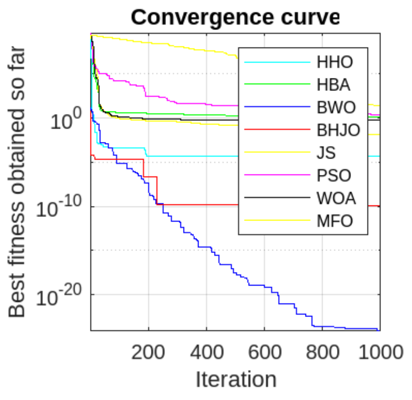
| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_1$ | AVG | **0.00E+00** | **0.00E+00** | 2.26E-229 | 1.83E-30 | 4.43E+04 | 5.77E-143 | 2.96E-182 | 2.62E+06 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 2.75E-30 | 1.68E+03 | 3.06E-142 | 0.00E+00 | 8.99E+04 |
| $F_2$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 5.17E-60 | 1.97E+09 | 1.51E-279 | **0.00E+00** | 2.66E+04 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.17E-59 | 1.82E+09 | 0.00E+00 | **0.00E+00** | 8.72E+04 |
| $F_3$ | AVG | **2.89E-256** | 7.16E-236 | INF | INF | 1.00E+300 | 6.27E-04 | 1.34E-90 | INF |
| | STD | **0.00E+00** | 0.00E+00 | NaN | NaN | 1.00E+300 | 3.57E-04 | 7.026E-90 | NaN |
| $F_4$ | AVG | **0.00E+00** | **0.00E+00** | 9.60E-228 | 3.01E-28 | 8.18E+06 | 4.51E-145 | 2.46E-177 | 5.01E+08 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 3.05E-28 | 2.44E+06 | 2.18E-143 | 0.00E+00 | 1.34E+07 |
| $F_5$ | AVG | **0.00E+00** | **0.00E+00** | 1.39E-222 | 5.94E-28 | 2.16E+07 | 6.88E-137 | 2.80E-171 | 1.17E+09 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 1.16E-27 | 1.09E+06 | 3.76E-136 | 0.00E+00 | 2.87E+07 |
| $F_6$ | AVG | **0.00E+00** | **0.00E+00** | 1.52E-228 | 6.31E-32 | 1.53E+06 | 3.79E-106 | 3.32E-184 | 6.70E+05 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 1.052E-31 | 1.84E+05 | 2.08E-105 | 0.00E+00 | 5.10E+04 |
| $F_7$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 6.05E-124 | 8.88E+14 | **0.00E+00** | **0.00E+00** | 5.29E+21 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 3.19E-123 | 1.76E+15 | **0.00E+00** | **0.00E+00** | 2.77E+20 |
| $F_8$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 3.20E+02 | 4.60E+02 | 9.44E-01 | **0.00E+00** | 4.67E+02 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.11E+02 | 3.97E+04 | 5.17E+00 | **0.00E+00** | 3.42E+00 |
| $F_9$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 3.20E+02 | 1.62E+04 | 6.06E-14 | **0.00E+00** | 1.47E+04 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 3.33E+02 | 3.32E-13 | **0.00E+00** | 1.89E+02 |
| $F_{10}$ | AVG | **4.44E-16** | **4.44E-16** | 2.65E+00 | 7.54E-15 | 1.61E+01 | 4.20E-15 | **4.44E-16** | 2.01E+01 |
| | STD | **0.00E+00** | **0.00E+00** | 6.88E+00 | 0.00E+00 | 3.51E+02 | 2.45E-15 | **0.00E+00** | 1.96E-01 |
| $F_{11}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 5.55E-17 | 3.044E+01 | **0.00E+00** | **0.00E+00** | 2.22E+04 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 5.64E-17 | 1.93E+00 | **0.00E+00** | **0.00E+00** | 5.33E+02 |
| $F_{12}$ | AVG | **0.00E+00** | **0.00E+00** | 1.60E-232 | 4.92E-32 | 2.38E+04 | 6.15E-146 | 1.25E-182 | 3.37E+04 |
| | STD | **0.00E+00** | **0.00E+00** | 0.00E+00 | 8.51E-32 | 1.58E+03 | 3.21E-145 | 0.00E+00 | 5.29E+03 |
| $F_{13}$ | AVG | 5.51E-04 | **7.97E-26** | 9.91E+01 | 6.66E+00 | 1.52E+08 | 2.09E+01 | 1.27E-04 | 4.94E+10 |
| | STD | 9.75E-04 | **2.18E-25** | 2.42E-01 | 1.97E+00 | 1.60E+07 | 5.10E+00 | 1.21E-04 | 1.52E+09 |

Table 3.11: Statistical results obtained for the fixed-dimension multimodal functions over different dimensions.

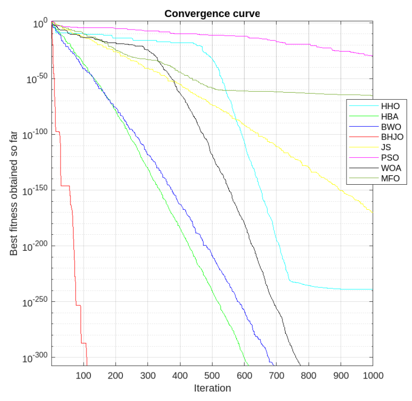
| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_{14}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 3.36E-165 | 3.53E-31 | **0.00E+00** | **0.00E+00** | 1.14E-48 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 1.35E-30 | **0.00E+00** | **0.00E+00** | 6.23E-48 |
| $F_{15}$ | AVG | -1.05E+01 | **-1.05E+01** | -8.72E+00 | -9.68E+00 | -1.05E+01 | -8.13E+00 | -5.09E+00 | -6.36E+00 |
| | STD | 5.12E-03 | **2.08E-06** | 3.07E+00 | 2.21E+00 | 1.14E-05 | 2.95E+00 | 1.96E-01 | 3.54E+00 |
| $F_{16}$ | AVG | -1.83E+03 | -1.55E+03 | -1.00E+03 | -9.98E+02 | -1.00E+03 | -1.99E+03 | -1.99E+03 | **-1.00E+03** |
| | STD | 2.33E+02 | 2.84E+02 | 1.15E-13 | 3.29E+00 | 3.58E-13 | 1.90E+01 | 1.07E-01 | **0.00E+00** |
| $F_{17}$ | AVG | **0.00E+00** | 5.83E-33 | 5.55E-15 | 2.51E-12 | 1.50E-10 | 6.44E-06 | 5.71E-11 | 5.93E-07 |
| | STD | **0.00E+00** | 3.08E-32 | 1.16E-14 | 4.92E-12 | 5.09E-10 | 1.23E-05 | 3.00E-10 | 2.11E-06 |
| $F_{18}$ | AVG | 8.44E-04 | **3.37E-04** | 5.41E-03 | 3.15E-04 | 9.59E-04 | 7.17E-04 | 3.26E-04 | 1.65E-03 |
| | STD | 3.5079E-04 | **4.40E-05** | 9.11E-03 | 4.17E-05 | 2.14E-04 | 4.01E-04 | 1.85E-05 | 3.56E-03 |
| $F_{19}$ | AVG | 3.24E-06 | 1.26E-03 | **1.23E-08** | 2.02E-06 | 4.03E-02 | 1.20E+00 | 8.67E-05 | 1.12E+00 |
| | STD | 9.44E-06 | 1.081E-03 | **3.54E-08** | 3.97E-06 | 4.07E-02 | 1.66E+00 | 1.17E-04 | 1.54E+00 |
| $F_{20}$ | AVG | **0.00E+00** | **0.00E+00** | **0.00E+00** | 5.13E-244 | 1.49E-38 | 1.58E-215 | 4.13E-215 | 8.58E-212 |
| | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 7.20E-38 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

Table 3.12: Statistical results obtained for the fixed-dimension multimodal functions over different dimensions.

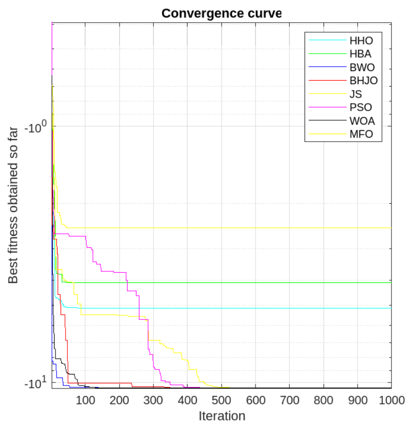
| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_{14}$ | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 1.35E-30 | **0.00E+00** | **0.00E+00** | 6.23E-48 |
| $F_{15}$ | STD | 5.12E-03 | **2.08E-06** | 3.07E+00 | 2.21E+00 | 1.14E-05 | 2.95E+00 | 1.96E-01 | 3.54E+00 |
| $F_{16}$ | STD | 2.33E+02 | 2.84E+02 | 1.15E-13 | 3.29E+00 | 3.58E-13 | 1.90E+01 | 1.07E-01 | **0.00E+00** |
| $F_{17}$ | STD | **0.00E+00** | 3.08E-32 | 1.16E-14 | 4.92E-12 | 5.09E-10 | 1.23E-05 | 3.00E-10 | 2.11E-06 |
| $F_{18}$ | STD | 3.5079E-04 | **4.40E-05** | 9.11E-03 | 4.17E-05 | 2.14E-04 | 4.01E-04 | 1.85E-05 | 3.56E-03 |
| $F_{19}$ | STD | 9.44E-06 | 1.081E-03 | **3.54E-08** | 3.97E-06 | 4.07E-02 | 1.66E+00 | 1.17E-04 | 1.54E+00 |
| $F_{20}$ | STD | **0.00E+00** | **0.00E+00** | **0.00E+00** | 0.00E+00 | 7.20E-38 | 0.00E+00 | 0.00E+00 | 0.00E+00 |



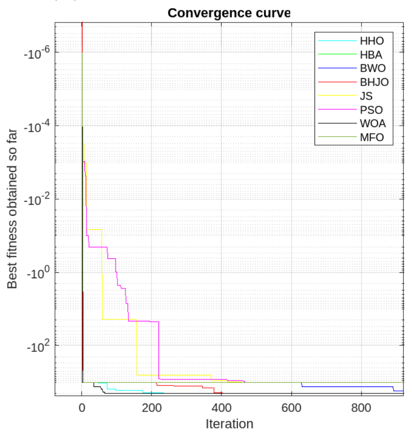
(a) Test function F13.



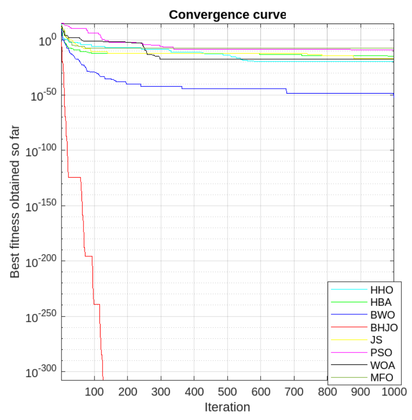
(b) Test function F14.



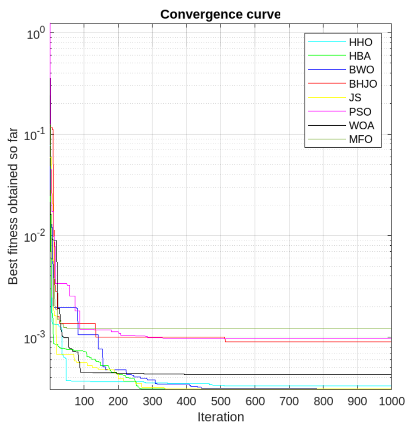
(c) Test function F15.



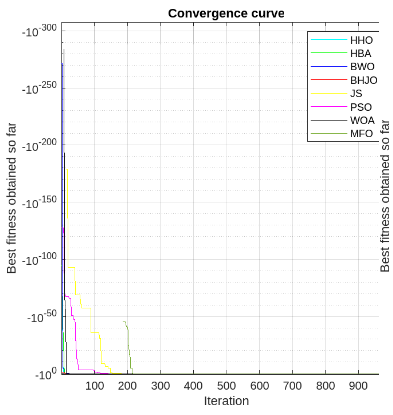
(d) Test function F16.



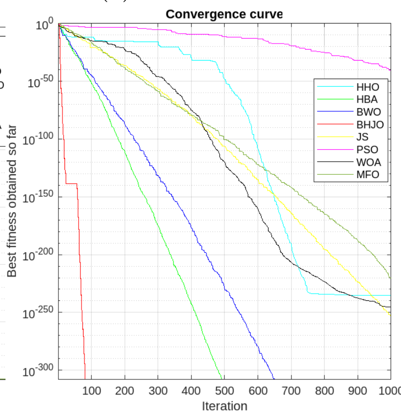
(e) Test function F17.



(f) Test function F18



(g) Test function F19.



(h) Test function F20.

Figure 3.8: Scalability analysis ($F_{13} - F_{20}$) of the BHJO algorithm (2/2).

Table 3.13: Statistical results obtained for the hybrid functions (CEC'17, CEC'2020, and CEC'2022).

| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_{21}$ | AVG | **1.77E+03** | 1.80E+03 | 1.74E+03 | 1.72E+03 | 1.77E+03 | 1.81E+03 | 1.77E+03 | 1.79E+03 |
| | STD | **7.70E+00** | 2.02E+01 | 2.61E+01 | 1.20E+01 | 4.74E+01 | 5.05E+01 | 3.48E+01 | 5.90E+01 |
| $F_{22}$ | AVG | 1.54E+03 | 1.64E+03 | 2.49E+03 | **1.45E+03** | 1.82E+03 | 2.38E+03 | 1.78E+03 | 5.60E+03 |
| | STD | 3.92E+01 | 6.49E+01 | 4.99E+03 | **2.65+01** | 7.76E+02 | 1.27E+03 | 5.89E+02 | 5.80E+03 |
| $F_{23}$ | AVG | 2.18E+03 | 2.20E+03 | 2.08E+03 | **2.02E+03** | 2.09E+03 | 2.16E+03 | 2.18E+03 | 2.08E+03 |
| | STD | 4.15E+01 | 4.72E+01 | 8.01E+01 | **1.05E+01** | 6.22E+01 | 7.01E+01 | 6.94E+01 | 7.51E+02 |
| $F_{24}$ | AVG | 1.19E+03 | 1.78E+03 | 1.11E+03 | **1.11E+03** | 1.13E+03 | 1.20E+03 | 1.15E+03 | 1.26E+03 |
| | STD | 4.79E+01 | 3.21E+02 | 1.18E+01 | **6.73E+00** | 2.54E+01 | 8.11E+01 | 3.44E+01 | 3.46E+02 |
| $F_{25}$ | AVG | **4.61E+01** | 5.35E+01 | 1.08E+02 | 1.13E+01 | 5.76E+01 | 1.36E+02 | 5.35E+01 | 1.28E+02 |
| | STD | **5.40E+00** | 8.89E+00 | 1.23E+02 | 9.60E+00 | 7.77E+01 | 1.08E+02 | 6.06E+01 | 1.47E+02 |
| $F_{26}$ | AVG | **2.230E+03** | 2.23E+03 | 2.22E+03 | 2.82E+03 | 2.21E+03 | 2.22E+03 | 2.23E+03 | 2.22E+03 |
| | STD | **1.59E+00** | 3.33E+00 | 2.82E+01 | 9.58E+00 | 1.68E+01 | 1.28E+01 | 1.76E+01 | 1.00E+01 |
| $F_{27}$ | AVG | 2.04E+03 | 2.09E+03 | 2.02E+03 | **2.02E+03** | 2.04E+03 | 2.05E+03 | 2.07E+03 | 2.04E+03 |
| | STD | 8.40E+00 | 1.90E+01 | 8.01E+00 | **6.43E+00** | 3.27E+01 | 2.33E+01 | 3.60E+01 | 2.51E+01 |
| $F_{28}$ | AVG | **2.04E+03** | 2.09E+03 | 2.02E+03 | 2.01E+03 | 2.04E+03 | 2.05E+03 | 2.11E+03 | 2.04E+03 |
| | STD | **6.82E+00** | 1.94E+01 | 1.09E+01 | 7.14E+00 | 3.21E+01 | 2.13E+01 | 5.68E+01 | 2.55E+01 |
| $F_{29}$ | AVG | 5.39E+03 | 1.13E+04 | **7.60E+02** | 1.26E+03 | 4.54E+03 | 2.98E+04 | 6.21E+03 | 5.71E+04 |
| | STD | 3.07E+03 | 7.81E+03 | **3.17E+02** | 1.29E+03 | 4.98E+03 | 8.53E+04 | 5.26E+03 | 2.58E+04 |
| $F_{30}$ | AVG | 5.43E+03 | 1.02E+04 | 3.02E+03 | **1.93E+03** | 3.16E+03 | 3.69E+04 | 8.83E+03 | 1.61E+04 |
| | STD | 3.91E+03 | 2.73E+03 | 5.67E+03 | **2.74E+01** | 1.80E+03 | 6.73E+03 | 6.72E+03 | 2.16E+04 |

Table 3.14: Statistical results obtained for the composition functions (CEC'17, CEC'2020, and CEC'2022).

| ID | | BHJO | BWO | HBA | JS | PSO | WOA | HHO | MFO |
|---|---|---|---|---|---|---|---|---|---|
| $F_{31}$ | AVG | **2.20E+03** | 2.25E+03 | 2.29E+03 | 2.25E+03 | 2.32E+03 | 2.32E+03 | 2.30E+03 | 2.32E+03 |
| | STD | **3.04E+00** | 1.63E+01 | 5.03E+01 | 5.36E+01 | 5.49E+01 | 5.11E+01 | 7.19E+01 | 4.29E+01 |
| $F_{32}$ | AVG | **3.27E+03** | 3.54E+03 | 3.39E+03 | 3.16E+03 | 3.17E+03 | 3.42E+03 | 3.36E+03 | 3.34E+03 |
| | STD | **3.36E+01** | 9.72E+01 | 2.24E+02 | 1.06E+02 | 5.40E+01 | 1.59E+02 | 1.11E+02 | 1.02E+03 |
| $F_{33}$ | AVG | 2.32E+03 | 2.56E+03 | **2.30E+03** | 2.29E+03 | 2.38E+03 | 2.35E+03 | 2.51E+03 | 2.31E+03 |
| | STD | 6.47E+00 | 1.98E+02 | **1.19E+00** | 1.59E+01 | 2.86E+02 | 2.29E+02 | 4.71E+02 | 2.08E+02 |
| $F_{34}$ | AVG | **2.53E+03** | 2.69E+03 | 2.75E+03 | 2.66E+03 | 2.78E+03 | 2.77E+03 | 2.82E+03 | 2.76E+03 |
| | STD | **1.12E+01** | 7.76E+01 | 2.67E+01 | 1.10E+02 | 9.60E+01 | 5.01E+01 | 4.39E+01 | 5.03E+01 |
| $F_{35}$ | AVG | 1.50E+02 | 1.50E+02 | **1.50E+02** | 1.20E+02 | 1.37E+02 | 1.50E+02 | 1.50E+02 | 1.24E+02 |
| | STD | 5.32E-09 | 5.88E-09 | **3.03E-14** | 5.55E+01 | 3.13E+01 | 7.91E-10 | 3.26E-08 | 4.82E+01 |
| $F_{36}$ | AVG | **2.00E+02** | **2.00E+02** | **2.00E+02** | 2.00E+02 | 2.00E+02 | 2.00E+02 | **2.00E+02** | 2.00E+02 |
| | STD | **2.89E-14** | **2.89E-14** | **2.89E-14** | 4.25E-14 | 2.82E-12 | 3.98E-14 | **2.89E-14** | 1.58E-14 |
| $F_{37}$ | AVG | 2.50E+03 | **2.50E+03** | **2.50E+03** | **2.50E+03** | 2.50E+03 | 2.50E+03 | 02.50E+03 | 2.50E+03 |
| | STD | 2.04E-14 | **0.00E+00** | **0.00E+00** | **0.00E+00** | 2.79E-13 | 2.89E-14 | 2.30E-14 | 1.49E-14 |
| $F_{38}$ | AVG | 2.70E+03 | 2.701E+03 | 2.70E+03 | 2.70E+03 | **2.70E+03** | 2.70E+03 | 2.70E+03 | 2.70E+03 |
| | STD | 6.35E-01 | 3.18E+01 | 5.46E-01 | 6.02E-00 | **6.22E-01** | 7.73E-01 | 7.12E-01 | 8.52E+00 |
| $F_{39}$ | AVG | **2.60E+03** | **2.60E+03** | 2.60E+03 | 2.60E+03 | 2.60E+03 | 2.60E+03 | 2.60E+03 | 2.60E+03 |
| | STD | **9.49E-05** | **9.49E-05** | 2.85E-04 | 2.30E-04 | 2.34E-04 | 1.45E-04 | 1.14E-04 | 2.02E-04 |
| $F_{40}$ | AVG | **1.78E+03** | 1.81E+03 | 1.74E+03 | **1.72E+03** | 1.75E+03 | 1.79E+03 | 1.78E+03 | 1.76E+03 |
| | STD | **1.07E+01** | 1.85E+01 | 3.21E+01 | **1.070E+01** | 3.30E+01 | 4.87E+01 | 3.84E+01 | 3.65E+02 |

tween them. By applying the Dunn's test after the Friedman's test, we can gain a deeper understanding of the specific group differences that contribute to the overall significant result. This approach helps to identify and highlight the particular groups that are driving the observed effects in the data, providing valuable insights for further analysis and interpretation.

We run the Friedman's test on the standard deviations values reported in Tables 3.7, 3.8, 3.9, 3.10, 3.12, 3.13, and 3.14. The obtained p-values are: $3.5107E-12$, $1.2533E-12$, $2.6161E-12$, $7.2087E-13$, 0.279761, $5.0485E-7$, and 0.000354, respectively. It is worth pointing out that if the p-value is greater than 5%, then it is concluded that there is no difference in performance among all the optimizers; which is the case for the fixed-dimension multimodal functions over different dimensions. For the rest of cases, we report the results of Dunn's test in Tables 3.15, 3.16, 3.17, 3.18, 3.19, and 3.15. The tables are read as follow: A cell found at the intersection of a row $i$ and a column $j$ might be empty, contain a normal typeface numerical value, or a bold typeface numerical value. An empty cell means that there now relationship between the optimizers found at the row $i$ and the column $j$, respectively. A normal typeface numerical value means that the optimizers found at the row $i$ and the column $j$ have the same performance. A bold typeface numerical value means that the optimizer found at the row $i$ has better performance when compared to the optimizer found at the column $j$.

In Tables 3.15, 3.16, 3.17, and 3.18, we observe that the performance of the BHJO algorithm is better than that of PSO and MFO optimizers; and it is the same as that of BWO, HBA, JS, WOA, and HHO optimizers. In Figures 3.9 (a)–(d), we notice that the BHJO optimizer has the second or third rank in term of performance. In Table 3.19, we observe that the performance of the BHJO algorithm is better than that of PSO, WOA, HHO, and MFO optimizers; and it is the same as that of BWO, HBA, and JS optimizers. In Figure 3.9 (e), we notice that the BHJO optimizer has the second rank in term of performance. Finally, in Table 3.20, we observe that the performance of the BHJO algorithm is better than that of PSO, WOA, HHO, and MFO optimizers; and it is the same as that of BWO, HBA, and JS optimizers. In Figure 3.9 (f), we notice that the BHJO optimizer has the first rank in term of performance. In conclusion, we can say that the proposed hybrid algorithm has greatly enhanced the ability of solving composition functions and this is due to the proposed balancing mechanism given by Equation 3.22.

## 3.4 Conclusion

This paper presents a novel hybrid metaheuristic algorithm that combines the strengths of three individual algorithms: Beluga Whale Optimization (BWO), Honey Badger Algorithm (HBA), and Artificial Jellyfish Search Optimizer (JS). The proposed hybrid algorithm is designed to overcome the limitations of each individual algorithm and leverage their complementary characteristics to enhance optimization performance. The algorithm's effectiveness and robustness were thoroughly evaluated through extensive experimentation on a diverse set of 40 test functions. To further enhance the algorithm's exploration and exploitation capabilities, the Combined Opposition-Based Learning (COBL) strategy was employed. COBL leverages the principles of opposition-based learning to

Table 3.15: Related-samples Friedman's two-way analysis of variance by ranks for the unimodal and multimodal functions with $D = 30$ (pairwise comparisons).

| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | - | 4.23E-1 | 3.78E-1 | **2.31E-4** | 1.28E-1 | 9.36E-1 | **7E-7** |
| **BWO** | 9.36E-1 | - | 3.78E-1 | 3.37E-1 | **1.68E-4** | 1.09E-1 | 8.73E-1 | **5E-7** |
| **HBA** | - | - | - | 9.36E-1 | **3.95E-3** | 4.71E-1 | - | **3.14E-5** |
| **JS** | - | - | - | - | **5.07E-3** | 5.22E-1 | - | **4.44E-5** |
| **PSO** | - | - | - | - | - | - | - | 2E-1 |
| **WOA** | - | - | - | - | **3.06E-2** | - | - | **5.76E-4** |
| **HHO** | - | - | 4.71E-1 | 4.23E-1 | **3.15E-4** | 1.5E-1 | - | **1E-6** |
| **MFO** | - | - | - | - | - | - | - | - |

Table 3.16: Related-samples Friedman's two-way analysis of variance by ranks for the unimodal and multimodal functions with $D = 50$ (pairwise comparisons).

| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | - | 4.71E-1 | 3.57E-1 | **7.4E-5** | 1.01E-1 | 9.36E-1 | **8E-7** |
| **BWO** | 9.36E-1 | - | 4.23E-1 | 3.17E-1 | **5.27E-5** | 8.52E-2 | 8.73E-1 | **6E-7** |
| **HBA** | - | - | - | 8.41E-1 | **1.18E-3** | 3.57E-1 | - | **2.63E-5** |
| **JS** | - | - | - | - | **2.35E-3** | 4.71E-1 | - | **6.25E-5** |
| **PSO** | - | - | - | - | - | - | - | 3.37E-1 |
| **WOA** | - | - | - | - | **2.02E-2** | - | - | **1.03E-3** |
| **HHO** | - | - | 5.22E-1 | 4.01E-1 | **1.03E-4** | 1.18E-1 | - | **1.3E-6** |
| **MFO** | - | - | - | - | - | - | - | - |

Table 3.17: Related-samples Friedman's two-way analysis of variance by ranks for the unimodal and multimodal functions with $D = 100$ (pairwise comparisons).

| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | - | 5.22E-1 | 3.37E-1 | **8.74E-5** | 2.98E-1 | 9.36E-1 | **3.4E-6** |
| **BWO** | 9.36E-1 | - | 4.71E-1 | 2.98E-1 | **6.25E-5** | 2.62E-1 | 8.73E-1 | **2.3E-6** |
| **HBA** | - | - | - | 7.49E-1 | **1.03E-3** | 6.89E-1 | - | **6.25E-5** |
| **JS** | - | - | - | - | **3.05E-3** | 9.36E-1 | - | **2.31E-4** |
| **PSO** | - | - | - | - | - | - | - | 4.71E-1 |
| **WOA** | - | - | - | - | **3.95E-3** | - | - | **3.15E-4** |
| **HHO** | - | - | 5.75E-1 | 3.78E-1 | **1.21E-4** | 3.37E-1 | - | **5E-6** |
| **MFO** | - | - | - | - | - | - | - | - |

Table 3.18: Related-samples Friedman's two-way analysis of variance by ranks for the unimodal and multimodal functions with $D = 1000$ (pairwise comparisons).

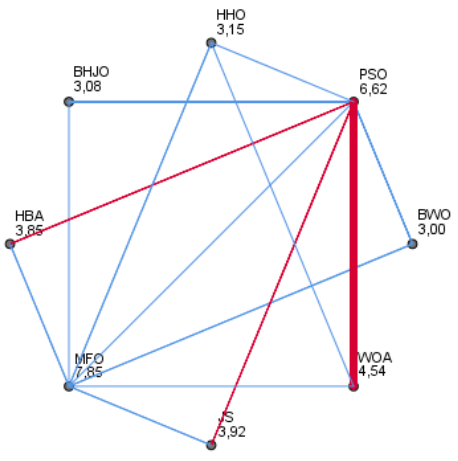
| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | - | 4.71E-1 | 4.01E-1 | **8.8E-6** | 3.57E-1 | - | **2.63E-5** |
| **BWO** | 8.73E-1 | - | 3.78E-1 | 3.17E-1 | **4.2E-6** | 2.8E-1 | 9.36E-1 | **1.28E-5** |
| **HBA** | - | - | - | 9.04E-1 | **1.97E-4** | 8.41E-1 | - | **4.96E-4** |
| **JS** | - | - | - | - | **3.15E-4** | 9.36E-1 | - | **7.72E-4** |
| **PSO** | - | - | - | - | - | - | - | - |
| **WOA** | - | - | - | - | **4.27E-4** | - | - | **1.03E-3** |
| **HHO** | 9.36E-1 | - | 4.23E-1 | 3.57E-1 | **6.1E-6** | 3.17E-1 | - | **1.84E-5** |
| **MFO** | - | - | - | - | 8.1E-1 | - | - | - |

Table 3.19: Related-samples Friedman's two-way analysis of variance by ranks for the hybrid functions (CEC'17, CEC'2020, and CEC'2022) (pairwise comparisons).

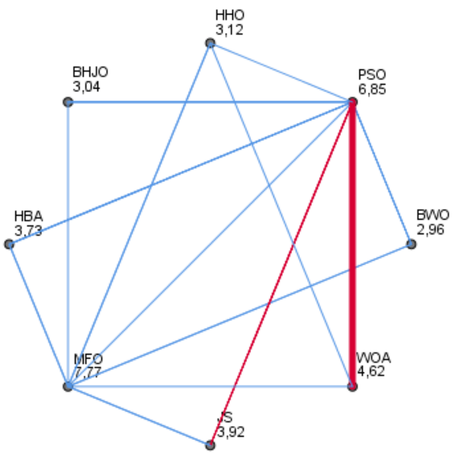
| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | 2.01E-1 | 5.52E-2 | - | **1.37E-2** | **3.71E-4** | **1.91E-3** | **1.18E-5** |
| **BWO** | - | - | 5.23E-1 | - | 2.35E-1 | **2.25E-2** | 6.79E-2 | **1.91E-3** |
| **HBA** | - | - | - | - | 5.84E-1 | 1E-1 | 2.35E-1 | **1.37E-2** |
| **JS** | 5.23E-1 | 5.52E-2 | **1.06E-2** | - | **1.91E-3** | **2.68E-5** | **1.82E-4** | **5E-7** |
| **PSO** | - | - | - | - | - | 2.73E-1 | 5.23E-1 | 5.52E-2 |
| **WOA** | - | - | - | - | - | - | - | 4.11E-1 |
| **HHO** | - | - | - | - | - | 6.48E-1 | - | 2.01E-1 |
| **MFO** | - | - | - | - | - | - | - | - |

Table 3.20: Related-samples Friedman's two-way analysis of variance by ranks for composition functions (CEC'17, CEC'2020, and CEC'2022) (pairwise comparisons).

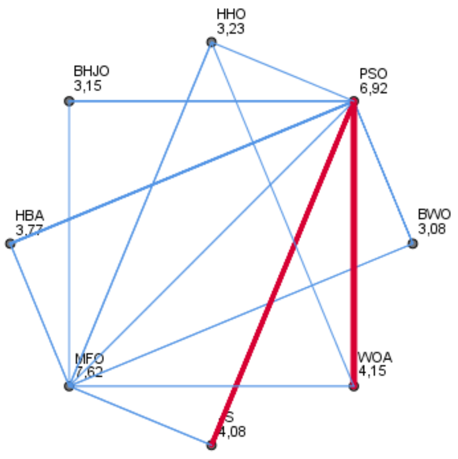
| | **BHJO** | **BWO** | **HBA** | **JS** | **PSO** | **WOA** | **HHO** | **MFO** |
|---|---|---|---|---|---|---|---|---|
| **BHJO** | - | 2.35E-1 | 6.16E-1 | 2.54E-1 | **9.28E-3** | **4.86E-5** | **7.08E-3** | **5.37E-3** |
| **BWO** | - | - | - | - | 1.57E-1 | **4.03E-3** | 1.32E-1 | 1.1E-1 |
| **HBA** | - | 4.94E-1 | - | 5.23E-1 | **3.58E-2** | **3.71E-4** | **2.85E-2** | **2.25E-2** |
| **JS** | - | 9.64E-1 | - | - | 1.44E-1 | **3.49E-3** | 1.21E-1 | 1E-1 |
| **PSO** | - | - | - | - | - | 1.44E-1 | 9.27E-1 | 8.55E-1 |
| **WOA** | - | - | - | - | - | - | - | - |
| **HHO** | - | - | - | - | - | 1.71E-1 | - | 9.27E-1 |
| **MFO** | - | - | - | - | - | 2.01E-1 | - | - |



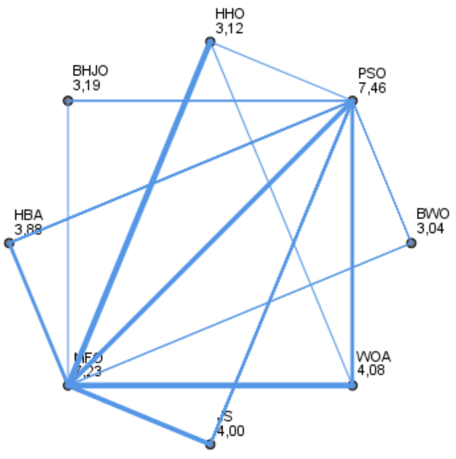
(a) Unimodal and multimodal functions with $D = 30$.



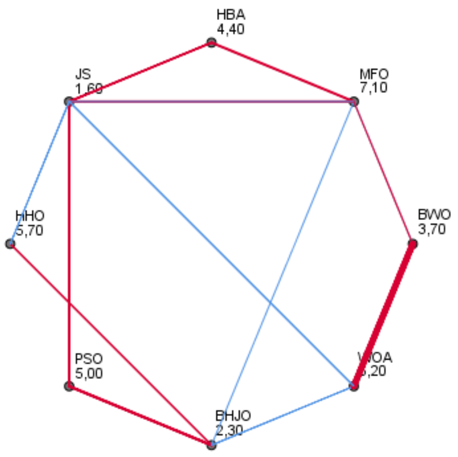
(b) Unimodal and multimodal functions with $D = 50$.



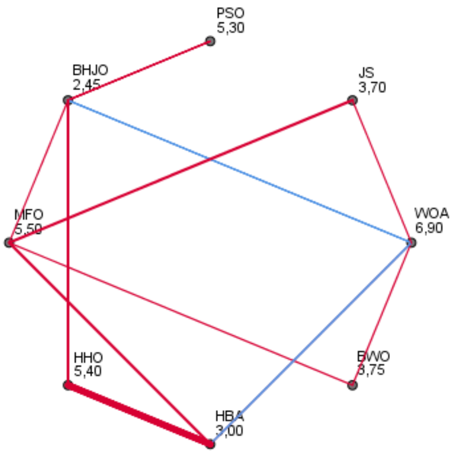
(c) Unimodal and multimodal functions with $D = 100$.



(d) Unimodal and multimodal functions with $D = 1000$.



(e) Hybrid functions (CEC'17, CEC'2020, and CEC'2022).



(f) Composition functions (CEC'17, CEC'2020, and CEC'2022).

Figure 3.9: Related-samples Friedman's two-way analysis of variance by ranks (pairwise comparisons).

enhance the search process by considering both the original solutions and their opposites. Moreover, the proposed hybrid algorithm introduces a new balancing mechanism to dynamically adjust the exploration and exploitation trade-off during the optimization process. This mechanism ensures a proper balance between exploration, to avoid premature convergence, and exploitation, to exploit promising regions in the search space effectively.

The experimental results demonstrate that the hybrid algorithm achieves superior/equal performance compared to the individual algorithms and other state-of-the-art metaheuristic approaches. It consistently outperforms the baseline algorithms in terms of convergence speed, solution quality, and robustness across various test functions. The findings of this research indicate that the combination of BWO, HBA, and JS, along with the incorporation of COBL and the novel balancing mechanism, yields a powerful hybrid metaheuristic algorithm capable of addressing complex optimization problems.

Future research directions may focus on further improving the hybrid algorithm by investigating alternative combinations of metaheuristic algorithms or exploring different balancing mechanisms. Additionally, conducting experiments on larger-scale problems and extending the algorithm's applicability to various real-world applications, that require efficient and effective optimization techniques, would provide valuable insights into its capabilities and limitations.

# Conclusion et perspectives

# General conclusion

In this thesis, we have explored the field of optimization and its fundamental concepts in Chapter  focused on a comprehensive study of metaheuristic algorithms in Chapter 1.9. Building upon this knowledge, in Chapter2.9 , we proposed a novel hybrid optimization algorithm called BHJO, which combines the strengths of BWO, HBA, and JS algorithms. We provided a thorough explanation of the algorithm's inspiration, a mathematical formulation, and its corresponding code implementation.

Furthermore, the BHJO algorithm was extensively evaluated by conducting experiments on a diverse set of 40 test functions, considering various problem definitions and dimensions ranging from 30 to 1000. The performance of the BHJO algorithm was rigorously analyzed using Friedman's test, providing statistical evidence of its efficacy and competitiveness in solving optimization problems.

# Perspectives

The findings of this research make significant contributions to the advancement of optimization techniques and open up promising avenues for further exploration. Future studies can build upon the BHJO algorithm and expand its capabilities by developing a multi-objective version that addresses problems involving the simultaneous optimization of multiple conflicting objectives. Additionally, an intriguing direction for research would involve applying the algorithm to real-world scenarios, such as optimizing a Deep Neuro Fuzzy network for cancer detection. By exploring these areas, researchers can further enhance the algorithm's effectiveness and broaden its application in solving complex optimization problems with practical implications.

# Bibliographie

[1] Hani Abdeen et al. “Multi-Objective Optimization in Rule-Based Design Space Exploration.” In: Sept. 2014. DOI: 10.1145/2642937.2643005.

[3] Mohamed Abdel-Basset et al. “Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems.” In: *Knowledge-Based Systems* (2023), p. 110248.

[4] Benyamin Abdollahzadeh et al. “Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems.” In: *Advances in Engineering Software* 174 (2022), p. 103282.

[5] Laith Abualigah et al. “The arithmetic optimization algorithm.” In: *Computer methods in applied mechanics and engineering* 376 (2021), p. 113609.

[6] Iman Ahmadianfar et al. “INFO: An efficient optimization algorithm based on weighted mean of vectors.” In: *Expert Systems with Applications* 195 (2022), p. 116516.

[7] Dabba Ali. *Méthodes de Résolution Exactes Heuristiques et Métaheuristiques*. 2023.

[8] M Montaz Ali, Charoenchai Khompatraporn, and Zelda B Zabinsky. “A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems.” In: *Journal of global optimization* 31.4 (2005), p. 635.

[9] Shima Amirsadri, Seyed Jalaleddin Mousavirad, and Hossein Ebrahimpour-Komleh. “A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training.” In: *Neural Computing and Applications* 30 (2018), pp. 3707–3720.

[10] Majsa Ammouriova et al. “A Heuristic-Based Simulation for an Education Process to Learn about Optimization Applications in Logistics and Transportation.” In: *Mathematics* 10.5 (2022), p. 830.

[11] Basavaraj M Angadi, Mahabaleshwar S Kakkasageri, and Sunilkumar S Manvi. “Computational intelligence techniques for localization and clustering in wireless sensor networks.” In: *Recent Trends in Computational Intelligence Enabled Research*. Elsevier, 2021, pp. 23–40.

[12] Mustapha Aouchiche et al. *Variable neighborhood search for extremal graphs 14: the AutoGraphiX 2 system*. Springer, 2006.

[13] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[14] Yasir Aslam and N Santhi. “A comprehensive survey on optimization techniques in image processing.” In: *Materials Today: Proceedings* 24 (2020), pp. 1758–1765.

[15] Mahdi Azizi. "Atomic orbital search: A novel metaheuristic algorithm." In: *Applied Mathematical Modelling* 93 (2021), pp. 657–683.

[16] Srinivasan Balan. "Metaheuristics in optimization: Algorithmic perspective." In: *INFORMS* ().

[17] Ozgur Baskan. *Optimization Algorithms*. Rijeka: IntechOpen, Sept. 2016. ISBN: 978-953-51-2593-8. DOI: 10.5772/61426.

[18] Dimitris Bertsimas and John Tsitsiklis. "Simulated annealing." In: *Statistical science* 8.1 (1993), pp. 10–15.

[19] Christian Blum et al. "Hybrid metaheuristics in combinatorial optimization: A survey." In: *Applied soft computing* 11.6 (2011), pp. 4135–4151.

[20] Mohamed Abdelhai Bouaicha and Abdellatif Houari Hocine. "The walking palm tree algorithm: A novel nature-inspired metaheuristic algorithm for global optimization." PhD thesis.

[21] Stephen Boyd and Jacob Mattingley. "Branch and bound methods." In: *Notes for EE364b, Stanford University* 2006 (2007), p. 07.

[22] Edmund K Burke et al. "Hyper-heuristics: A survey of the state of the art." In: *Journal of the Operational Research Society* 64 (2013), pp. 1695–1724.

[23] Alexei V Chechkin et al. "Introduction to the theory of Lévy flights." In: *Anomalous transport: Foundations and applications* (2008), pp. 129–162.

[24] C. L. Philip Chen, Tong Zhang, and Sik Chung Tam. "A novel evolutionary algorithm solving optimization problems." In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2014, pp. 557–561. DOI: 10.1109/SMC.2014.6973966.

[25] Jui-Sheng Chou and Ngoc-Tri Ngo. "Modified firefly algorithm for multidimensional optimization in structural design problems." In: *Structural and Multidisciplinary Optimization* 55 (2017), pp. 2013–2028.

[26] Jui-Sheng Chou and Dinh-Nhat Truong. "A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean." In: *Applied Mathematics and Computation* 389 (2021), p. 125535.

[27] Carlos A Coello Coello and Efrén Mezura Montes. "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection." In: *Advanced Engineering Informatics* 16.3 (2002), pp. 193–203.

[28] Erik Cuevas, Alonso Echavarría, and Marte A Ramírez-Ortegón. "An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation." In: *Applied intelligence* 40 (2014), pp. 256–272.

[29] Ana Luísa Custódio and JF Aguilar Madeira. "MultiGLODS: global and local multiobjective optimization using direct search." In: *Journal of Global Optimization* 72 (2018), pp. 323–345.

[30] Kalyanmoy Deb. *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[31] Mohammad Dehghani et al. "A new "Doctor and Patient" optimization algorithm: An application to energy commitment problem." In: *Applied Sciences* 10.17 (2020), p. 5791.

[32] Mohammad Dehghani et al. "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems." In: *Knowledge-Based Systems* 259 (2023), p. 110011.

[33] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* SIAM, 1996.

[34] Sachin Desale et al. "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey." In: *Int. J. Comput. Eng. Res. Trends* 351.5 (2015), pp. 2349–7084.

[35] Jason G Digalakis and Konstantinos G Margaritis. "On benchmarking functions for genetic algorithms." In: *International journal of computer mathematics* 77.4 (2001), pp. 481–506.

[36] John H Drake et al. "Recent advances in selection hyper-heuristics." In: *European Journal of Operational Research* 285.2 (2020), pp. 405–428.

[37] Emad Elbeltagi, Tarek Hegazy, and Donald Grierson. "Comparison among five evolutionary-based optimization algorithms." In: *Advanced engineering informatics* 19.1 (2005), pp. 43–53.

[38] Saber Elsayed, Ruhul Sarker, and Carlos A. Coello Coello. "Sequence-Based Deterministic Initialization for Evolutionary Algorithms." In: *IEEE Transactions on Cybernetics* 47.9 (2017), pp. 2911–2923. DOI: 10.1109/TCYB.2016.2630722.

[39] Michael TM Emmerich and André H Deutz. "A tutorial on multiobjective optimization: fundamentals and evolutionary methods." In: *Natural computing* 17 (2018), pp. 585–609.

[40] Andries P Engelbrecht. *Computational intelligence: an introduction.* John Wiley & Sons, 2007.

[41] Hadi Eskandar et al. "Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems." In: *Computers & Structures* 110 (2012), pp. 151–166.

[42] Qian Fan et al. "Engineering with Computers ESSAWOA: Enhanced Whale Optimization Algorithm integrated with Salp Swarm Algorithm for global optimization." In: *Engineering With Computers* 38 (Apr. 2022), p. 18. DOI: 10.1007/s00366-020-01189-3.

[43] Amir Mohammad Fathollahi-Fard, Mostafa Hajiaghaei-Keshteli, and Reza Tavakkoli-Moghaddam. "Red deer algorithm (RDA): a new nature-inspired meta-heuristic." In: *Soft Computing* 24 (2020), pp. 14637–14665.

[44] P. Festa. "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems." In: *2014 16th International Conference on Transparent Optical Networks (ICTON)*. 2014, pp. 1–20. DOI: 10.1109/ICTON.2014.6876285.

[45] Sabrina Fossette et al. “Current-oriented swimming by jellyfish and its role in bloom maintenance.” In: *Current Biology* 25.3 (2015), pp. 342–347.

[46] Prashant J Gaidhane and Madhav J Nigam. “A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems.” In: *Journal of computational science* 27 (2018), pp. 284–302.

[47] Amir H Gandomi et al. “Firefly algorithm with chaos.” In: *Communications in Nonlinear Science and Numerical Simulation* 18.1 (2013), pp. 89–98.

[48] Amir Hossein Gandomi et al. “ng and optimizatiMetaheuristic algorithms in modelion.” In: *Metaheuristic applications in structures and in frastructures* 1 (2013).

[49] Xiao Zhi Gao et al. “Harmony search method: theory and applications.” In: *Computational intelligence and neuroscience* 2015 (2015), pp. 39–39.

[50] Michel Gendreau. *An introduction to tabu search.* Springer, 2003.

[51] Carla P. Gomes and Ryan Williams. “Approximation Algorithms.” In: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.* Ed. by Edmund K. Burke and Graham Kendall. Boston, MA: Springer US, 2005, pp. 557–585. ISBN: 978-0-387-28356-2. DOI: 10.1007/0-387-28356-0_18.

[52] Vahid Goodarzimehr et al. “Bonobo optimizer algorithm for optimum design of truss structures with static constraints.” In: *Structures* 50 (2023), pp. 400–417. ISSN: 2352-0124. DOI: https://doi.org/10.1016/j.istruc.2023.02.023.

[53] Igor Griva, Stephen G Nash, and Ariela Sofer. *Linear and nonlinear optimization.* Vol. 108. Siam, 2009.

[54] Bertrand Guenin, Jochen Könemann, and Levent Tuncel. *A gentle introduction to optimization.* Cambridge University Press, 2014.

[55] Fatma A Hashim et al. “Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems.” In: *Applied Intelligence* 51 (2021), pp. 1531–1551.

[56] Fatma A Hashim et al. “Henry gas solubility optimization: A novel physics-based algorithm.” In: *Future Generation Computer Systems* 101 (2019), pp. 646–667.

[57] Fatma A Hashim et al. “Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems.” In: *Mathematics and Computers in Simulation* 192 (2022), pp. 84–110.

[58] Mohammad Reza Hassanzadeh and Farshid Keynia. “An Overview of the Concepts, Classifications, and Methods of Population Initialization in Metaheuristic Algorithms.” In: *Journal of Advances in Computer Engineering and Technology* 7.1 (2021), pp. 35–54.

[59] Ali Asghar Heidari et al. “Harris hawks optimization: Algorithm and applications.” In: *Future generation computer systems* 97 (2019), pp. 849–872.

[61] Reiner Horst and Panos M Pardalos. *Handbook of global optimization.* Vol. 2. Springer Science & Business Media, 2013.

[62] Patrik T Hultberg and David Santandreu Calonge. “Effective teaching of economics: A constrained optimization problem?” In: *The Journal of Economic Education* 48.4 (2017), pp. 265–275.

[63] Kashif Hussain et al. “On the exploration and exploitation in popular swarm-based metaheuristic algorithms.” In: *Neural Computing and Applications* 31 (2019), pp. 7665–7683.

[64] Satoshi Ide. “A Brownian walk model for slow earthquakes.” In: *Geophysical Research Letters* 35.17 (2008).

[65] Ehsan Jahani and Mohammad Chizari. “Tackling global optimization problems with a novel algorithm–Mouth Brooding Fish algorithm.” In: *Applied Soft Computing* 62 (2018), pp. 987–1002.

[66] Laetitia Jourdan, Matthieu Basseur, and E-G Talbi. “Hybridizing exact methods and metaheuristics: A taxonomy.” In: *European Journal of Operational Research* 199.3 (2009), pp. 620–629.

[67] Imed Kacem. *Méthodes exactes en optimisation combinatoire*. 2023.

[68] Janusz Kacprzyk. “Studies in Computational Intelligence, Volume 100.” In: (2008).

[69] Nikos Ath Kallioras, Nikos D Lagaros, and Dimitrios N Avtzis. “Pity beetle algorithm–A new metaheuristic inspired by the behavior of bark beetles.” In: *Advances in Engineering Software* 121 (2018), pp. 147–166.

[70] Daniel J Kapner et al. “Tests of the gravitational inverse-square law below the dark-energy length scale.” In: *Physical review letters* 98.2 (2007), p. 021101.

[71] A Kaveh and A Dadras Eslamlou. “Water strider algorithm: A new metaheuristic and applications.” In: *Structures*. Vol. 25. Elsevier. 2020, pp. 520–541.

[72] A Kaveh, R Mahdipour Moghanni, and SM Javadi. “Optimum design of large steel skeletal structures using chaotic firefly optimization algorithm based on the Gaussian map.” In: *Structural and Multidisciplinary Optimization* 60 (2019), pp. 879–894.

[73] J Kennedy and RC Everhart. “A new optimizer using particle swarm theory. In proceedings of the sixth international symposium on micro machine and human science. Nagoya Japón.” In: *IEEE service center Piscataway, NJ* (1995).

[74] Mustafa Servet Kıran and Oğuz Fındık. “A directed artificial bee colony algorithm.” In: *Applied Soft Computing* 26 (2015), pp. 454–462.

[75] Annu Lambora, Kunal Gupta, and Kriti Chopra. “Genetic algorithm-A literature review.” In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE. 2019, pp. 380–384.

[76] Jouni Lampinen and Ivan Zelinka. “Mixed integer-discrete-continuous optimization by differential evolution.” In: *Proceedings of the 5th international conference on soft computing*. Vol. 71. Citeseer. 1999, p. 76.

[77] Qian Li, Yiguang Bai, and Weifeng Gao. “Improved Initialization Method for Metaheuristic Algorithms: A Novel Search Space View.” In: *IEEE Access* 9 (2021), pp. 121366–121384. DOI: 10.1109/ACCESS.2021.3073480.

[78] Shimin Li et al. “Slime mould algorithm: A new method for stochastic optimization.” In: *Future Generation Computer Systems* 111 (2020), pp. 300–323.

[79] Xiaojian Li, Yijia Zhao, and Zhengxian Liu. “A novel global optimization algorithm and data-mining methods for turbomachinery design.” In: *Structural and Multidisciplinary Optimization* 60 (2019), pp. 581–612.

[80] Yilun Liu and Xiaoming Li. “A Hybrid Mobile Robot Path Planning Scheme Based on Modified Gray Wolf Optimization and Situation Assessment.” In: *Journal of Robotics* 2022 (2022).

[81] Zhi-Zhong Liu, Bing-Chuan Wang, and Ke Tang. “Handling Constrained Multiobjective Optimization Problems via Bidirectional Coevolution.” In: *IEEE Transactions on Cybernetics* 52.10 (2022), pp. 10163–10176. DOI: 10.1109/TCYB.2021.3056176.

[82] Marco Locatelli and Fabio Schoen. *Global optimization: theory, algorithms, and applications*. SIAM, 2013.

[83] Cuicui Lu, Hongyi Yuan, and Nianen Zhang. “Nanophotonic devices based on optimization algorithms.” In: *Intelligent Nanotechnology*. Elsevier, 2023, pp. 71–111.

[84] Rosario Nunzio Mantegna. “Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes.” In: *Physical Review E* 49.5 (1994), p. 4677.

[85] Boris Marcone et al. “Brownian non-Gaussian diffusion of self-avoiding walks.” In: *Journal of Physics A: Mathematical and Theoretical* 55.35 (2022), p. 354003.

[86] Robert M May. “Simple mathematical models with very complicated dynamics.” In: *Nature* 261 (1976), pp. 459–467.

[87] Seyedali Mirjalili. “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm.” In: *Knowledge-based systems* 89 (2015), pp. 228–249.

[88] Seyedali Mirjalili. “SCA: a sine cosine algorithm for solving optimization problems.” In: *Knowledge-based systems* 96 (2016), pp. 120–133.

[89] Seyedali Mirjalili and Andrew Lewis. “The whale optimization algorithm.” In: *Advances in engineering software* 95 (2016), pp. 51–67.

[90] Abdolkarim Mohammadi-Balani et al. “Golden eagle optimizer: A nature-inspired metaheuristic algorithm.” In: *Computers & Industrial Engineering* 152 (2021), p. 107050.

[91] Seyyed Hamid Samareh Moosavi and Vahid Khatibi Bardsiri. “Poor and rich optimization algorithm: A new human-based and multi populations algorithm.” In: *Engineering Applications of Artificial Intelligence* 86 (2019), pp. 165–181.

[92] Seyed Jalaleddin Mousavirad and Hossein Ebrahimpour-Komleh. “Human mental search: a new population-based metaheuristic optimization algorithm.” In: *Applied Intelligence* 47 (2017), pp. 850–887.

[93] Olaide Nathaniel Oyelade et al. “Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm.” In: *IEEE Access* 10 (2022), pp. 16150–16177.

[94] Vladimir V Palyulin et al. “First passage and first hitting times of Lévy flights and Lévy walks.” In: *New Journal of Physics* 21.10 (2019), p. 103028.

[95] Markus Pantsar. “Descriptive complexity, computational tractability, and the logical and cognitive foundations of mathematics.” In: *Minds and Machines* 31.1 (2021), pp. 75–98.

[96] Moushita Patnaik and Angelia Melani Adrian. “Chapter 4 - A perspective depiction of heuristics in virtual reality.” In: *Cognitive Big Data Intelligence with a Metaheuristic Approach*. Ed. by Sushruta Mishra et al. Cognitive Data Science in Sustainable Computing. Academic Press, 2022, pp. 101–116. ISBN: 978-0-323-85117-6. DOI: https://doi.org/10.1016/B978-0-323-85117-6.00006-6.

[97] Catherine PORTE. “Méthodes directes d’optimisation-Méthodes à une variable et simplex.” In: (2017).

[98] Hojatollah Rajabi Moshtaghi, Abbas Toloie Eshlaghy, and Mohammad Reza Motadel. “A comprehensive review on meta-heuristic algorithms and their classification with novel approach.” In: *Journal of Applied Research on Industrial Engineering* 8.1 (2021), pp. 63–89.

[99] Sohail Razzaq et al. “Efficient optimization techniques for resource allocation in UAVs mission framework.” In: *PLOS ONE* 18.4 (Apr. 2023), pp. 1–19. DOI: 10.1371/journal.pone.0283923.

[100] Franz Rothlauf. *Design of modern heuristics: principles and application*. Vol. 8. 9. Springer, 2011.

[101] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[102] Malek Sarhani, Stefan Voß, and Raka Jovanovic. “Initialization of metaheuristics: comprehensive review, critical analysis, and research directions.” In: *International Transactions in Operational Research* (2022).

[103] Soudeh Shadravan, Hamid Reza Naji, and Vahid Khatibi Bardsiri. “The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems.” In: *Engineering Applications of Artificial Intelligence* 80 (2019), pp. 20–34.

[104] Abha Singh et al. “Parameter extraction of solar module using the sooty tern optimization algorithm.” In: *Electronics* 11.4 (2022), p. 564.

[105] Narinder Singh and Hanaa Hachimi. “A new hybrid whale optimizer algorithm with mean strategy of grey wolf optimizer for global optimization.” In: *Mathematical and computational applications* 23.1 (2018), p. 14.

[106] Shanmugam Sundaramurthy and Preethi Jayavel. “A hybrid grey wolf optimization and particle swarm optimization with C4. 5 approach for prediction of rheumatoid arthritis.” In: *Applied Soft Computing* 94 (2020), p. 106500.

[107] E-G Talbi. “A taxonomy of hybrid metaheuristics.” In: *Journal of heuristics* 8 (2002), pp. 541–564.

[108] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[109] Hichem Talbi and Amer Draa. “A new real-coded quantum-inspired evolutionary algorithm for continuous optimization.” In: *Applied Soft Computing* 61 (2017), pp. 765–791. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2017.07.046.

[110] HR Tizhoosh. “Opposition-based learning: A new scheme for machine intelligence. Paper presented at the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06).” In: (2005).

[111] Jose L Villegas, Enrique Castro, José Gutiérrez, et al. “Representations in problem solving: A case study with optimization problems.” In: (2009).

[112] Dongshu Wang, Dapei Tan, and Lei Liu. “Particle swarm optimization algorithm: an overview.” In: *Soft computing* 22 (2018), pp. 387–408.

[113] Shuang Wang et al. “An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems.” In: *Processes* 9.9 (2021), p. 1551.

[114] HATTAB Wassila et al. “Algorithme d’optimisation inspiré de la poussée d’archiméde appliqué au dispatching économique.” In: ().

[115] Bernd Wursig, William F Perrin, et al. *Encyclopedia of marine mammals*. Academic Press, 2009.

[116] Tao Xiang, Xiaofeng Liao, and Kwok-wo Wong. “An improved particle swarm optimization algorithm combined with piecewise linear chaotic map.” In: *Applied Mathematics and Computation* 190.2 (2007), pp. 1637–1645.

[117] Yaning Xiao et al. “IHAOAVOA: An improved hybrid aquila optimizer and African vultures optimization algorithm for global optimization problems.” In: *Mathematical Biosciences and Engineering* 19.11 (2022), pp. 10963–11017.

[118] Xin-She Yang. “A new metaheuristic bat-inspired algorithm.” In: *Nature inspired cooperative strategies for optimization (NICSO 2010)* (2010), pp. 65–74.

[119] Xin-She Yang. “Chapter 13 - How to Deal with Constraints.” In: *Nature-Inspired Optimization Algorithms*. Ed. by Xin-She Yang. Oxford: Elsevier, 2014, pp. 183–196. ISBN: 978-0-12-416743-8. DOI: https://doi.org/10.1016/B978-0-12-416743-8.00013-0.

[120] Xin-She Yang. *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[121] Xin-She Yang. *Nature-inspired optimization algorithms*. Academic Press, 2020.

[122] Xin-She Yang et al. “A framework for self-tuning optimization algorithm.” In: *Neural Computing and Applications* 23 (2013), pp. 2051–2057.

[123] Xin Yao, Yong Liu, and Guangming Lin. “Evolutionary programming made faster.” In: *IEEE Transactions on Evolutionary computation* 3.2 (1999), pp. 82–102.

[124] Mohd Z Zakaria et al. “Comparison between multi-objective and single-objective optimization for the modeling of dynamic systems.” In: *Proceedings of the institution of mechanical engineers, part i: journal of systems and control engineering* 226.7 (2012), pp. 994–1005.

[125] DJMB Zavodnik. “Spatial aggregations of the swarming jellyfish Pelagia noctiluca (Scyphozoa).” In: *Marine Biology* 94.2 (1987), pp. 265–269.

[126] Chaoyong Zhang et al. “A new hybrid GA/SA algorithm for the job shop scheduling problem.” In: *Evolutionary Computation in Combinatorial Optimization: 5th European Conference, EvoCOP 2005, Lausanne, Switzerland, March 30-April 1, 2005. Proceedings 5*. Springer. 2005, pp. 246–259.

[127] Jinhao Zhang et al. “Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems.” In: *Applied Mathematical Modelling* 63 (2018), pp. 464–490.

[128] Yu-Jun Zhang et al. “AOAAO: The hybrid algorithm of arithmetic optimization algorithm with aquila optimizer.” In: *IEEE Access* 10 (2022), pp. 10907–10933.

[129] Kaiqing Zhang et al. “Global convergence of policy gradient methods: A nonconvex optimization perspective.” In: *SIAM Journal on control and Optimization (under review)* (2019).

[130] Shijie Zhao et al. “Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications.” In: *Engineering Applications of Artificial Intelligence* 114 (2022), p. 105075.

[131] Changting Zhong, Gang Li, and Zeng Meng. “Beluga whale optimization: A novel nature-inspired metaheuristic algorithm.” In: *Knowledge-Based Systems* 251 (2022), p. 109215. ISSN: 0950-7051.

[132] Wang Zhongyu, Li Yaru, and Tang Yingqi. “An efficient hybrid DE-WOA algorithm for numerical function optimization.” In: *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. IEEE. 2019, pp. 2629–2634.

[133] Farouq Zitouni, Saad Harous, and Ramdane Maamri. “The solar system algorithm: a novel metaheuristic method for global optimization.” In: *IEEE Access* 9 (2020), pp. 4542–4565.

[134] Farouq Zitouni et al. “The archerfish hunting optimizer: A novel metaheuristic algorithm for global optimization.” In: *Arabian Journal for Science and Engineering* 47.2 (2022), pp. 2513–2553.

# Webographie

[2] M Abdel-Basset, L Abdel-Fatah, and AK Sangaiah. *Chapter 10-Metaheuristic Algorithms: A Comprehensive Review In Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. 2018.

[60] B HongKong. *Approximation Algorithms*. https://www.ics.uci.edu/~vazirani/book.pdf. 2001.