Democratic And Popular Republic Of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry Of Higher Education And Scientific Research
وزارة التعليم العالي والبحث العلمي

University KASDI Merbah – Ouargla
جامعة قاصدي مرباح - ورقلة
Faculty Of New Information And Communication Technology
Department Of Computer Science And Information Technology

Academic Master Thesis

Speciality: Computer Science
Option: Fundamental Computing

Realized by: BAROUTCHI Baya Diyaa and SELLAT Fatoum

A Contrastive Analysis Between the Performance of Five Metaheuristic Algorithms for Parameter Estimation in Photovoltaic Models: (OFDA, SBOA, GRO, ZOA, HOA)

Publicly presented on: 24/06/2024

Jury members:

Dr. BEKKARI Fouad	Jury chair	University KASDI Merbah - Ouargla
Dr. ZITOUNI Farouq	Supervisor	University KASDI Merbah - Ouargla
Dr. TOUMI Chahrazed	Examiner	University KASDI Merbah - Ouargla

Academic Year: 2023/2024

Dedication

This thesis is dedicated to our parents, for their endless support and encouragement throughout our academic journey.

To our siblings, who have always believed in us and motivated us to persevere. To our family and friends, whose love and understanding have been a constant source of strength.

To all our teachers and mentors, from primary school to the completion of this thesis, thank you for your guidance and wisdom.

Finally, to everyone who contributed to this work, whether through advice, assistance or simply being there, we are deeply grateful.

Acknowledgments

First of all, we would like to thank **ALLAH** for giving us health, willpower, and patience to complete this humble work.

We wish to express our profound gratitude to our supervisor, **Dr ZITOUNI Farouq**, professor at the University of KASDI Merbah OUARGLA, for agreeing to guide and supervise our final project. His patience and invaluable advice have been of immense value throughout this period.

We would like to express our deep appreciation to **Ms LAKBICHI Rihab** for her invaluable help. Her competence, expertise, and willingness to assist have been an inspiration to us. Her generosity and dedication have had a significant impact on our journey.

We also wish to express our gratitude to the entire teaching staff of the Department of Computer Science and Information Technology at KASDI Merbah Ouargla University, who have been responsible for our training throughout our university education. Their commitment and dedication have ensured a quality educational experience.

Abstract

The optimization, control, and simulation of photovoltaic (PV) systems are essential to maximizing the effectiveness of solar energy utilization. An important aspect influencing the performance of PV systems is the variability and unavailability of model parameters, necessitating accurate identification and determination of these parameters. This thesis provides a comparative study of various metaheuristic algorithms utilized for extracting model parameters from specific (PV) panels. The primary objective is to evaluate and contrast the efficiency of five novel metaheuristic algorithms. The SDM mathematical model was employed for its remarkable accuracy and simplicity in achieving this objective. In contrast, the DDM mathematical model was ultimately chosen due to its superior ability to achieve a higher degree of accuracy, particularly under real-world operating conditions. Subsequently, simulations were carried out to identify the best model parameters based on the root mean square error (RMSE) values that each algorithm produced. The dataset obtained from these simulations, which consisted of P-V and I-V curves as well as comparisons of CPU times, was examined and contrasted to derive definitive conclusions. The Opposition Flow Directional Algorithm (OFDA) proved to be superior to the other algorithms in terms of minimizing the error between the experimental and simulated data. Even though some algorithms like ZOA and SBOA exhibited faster execution speeds, OFDA achieved competitive CPU times while demonstrating a higher level of resemblance between the simulated P-V and I-V curves compared to the experimental data. The main aim of this comparative study is to acquire valuable knowledge regarding the effectiveness of different emerging metaheuristic algorithms in identifying model parameters for photovoltaic modules.

Keywords: Photovoltaic (PV) systems, Extracting model parameters, Metaheuristic algorithms, Single diode model, Double diode model, Opposition Flow Directional Algorithm (OFDA).

Résumé

L'optimisation, le contrôle et la simulation des systèmes photovoltaïques (PV) sont essentiels pour maximiser l'efficacité de l'utilisation de l'énergie solaire. Un aspect important influençant les performances des systèmes photovoltaïques est la variabilité et l'indisponibilité des paramètres du modèle, nécessitant une identification et une détermination précises de ces paramètres. Cette thèse propose une étude comparative de divers algorithmes métaheuristiques utilisés pour extraire les paramètres de modèle de panneaux photovoltaïques spécifiques. L'objectif principal est d'évaluer et de comparer l'efficacité de cinq nouveaux algorithmes métaheuristiques. Le modèle mathématique SDM a été utilisé pour sa précision et sa simplicité remarquables dans la réalisation de cet objectif. En revanche, le modèle mathématique DDM a finalement été choisi en raison de sa capacité supérieure à atteindre un degré plus élevé de précision, en particulier dans des conditions de fonctionnement réelles. Par la suite, des simulations ont été effectuées pour identifier les meilleurs paramètres du modèle en fonction des valeurs d'erreur quadratique moyenne (RMSE) produites par chaque algorithme. L'ensemble de données obtenu à partir de ces simulations, composé de courbes P-V et I-V ainsi que de comparaisons de temps CPU, a été examiné et comparé pour tirer des conclusions définitives. L'algorithme directionnel de flux d'opposition (OFDA) s'est avéré supérieur aux autres algorithmes en termes de minimisation de l'erreur entre les données expérimentales et simulées. Même si certains algorithmes comme ZOA et SBOA présentaient des vitesses d'exécution plus rapides, OFDA a atteint des temps CPU compétitifs tout en démontrant un niveau de ressemblance plus élevé entre les courbes P-V et I-V simulées par rapport aux données expérimentales. L'objectif principal de cette étude comparative est d'acquérir des connaissances précieuses sur l'efficacité de différents algorithmes métaheuristiques émergents dans l'identification des paramètres de modèle pour les modules photovoltaïques.

Mots Clée: Systèmes photovoltaïques (PV), Extraire les paramètres du modèle, algorithmes métaheuristiques, modèle à simple diode, modèle à double diode, algorithme directionnel à flux d'opposition (OFDA).

ملخص

يعد تحسين الأنظمة الكهروضوئية (PV) والتحكم فيها ومحاكاتها أمرًا ضروريًا لتعظيم فعالية استخدام الطاقة الشمسية. أحد الجوانب المهمة التي تؤثر على أداء الأنظمة الكهروضوئية هو التباين وعدم توفر معلمات النموذج، مما يستلزم تحديدًا دقيقًا وتحديد هذه المعلمات. تقدم هذه الأطروحة دراسة مقارنة لمختلف خوارزميات الميتايورستك المستخدمة لاستخراج معلمات النموذج من ألواح (PV) محددة. الهدف الأساسي هو تقييم ومقارنة كفاءة خمس خوارزميات ميتايورستية جديدة. تم استخدام النموذج الرياضي (SDM) لدقته وبساطته الملحوظة في تحقيق هذا الهدف. في المقابل، تم اختيار النموذج الرياضي (DDM) في النهاية نظرًا لقدرته الفائقة على تحقيق درجة أعلى من الدقة، خاصة في ظل ظروف التشغيل الواقعية. بعد ذلك، تم إجراء عمليات المحاكاة لتحديد أفضل معلمات النموذج استنادًا إلى قيم جذر متوسط مربع الخطأ (RMSE) التي أنتجتها كل خوارزمية. تم فحص مجموعة البيانات التي تم الحصول عليها من عمليات المحاكاة هذه، والتي تتكون من منحنيات (P-V) و(I-V) بالإضافة إلى مقارنات أوقات وحدة المعالجة المركزية، ومقارنتها لاستخلاص استنتاجات نهائية. أثبتت خوارزمية اتجاه تدفق المعارضة (OFDA) تفوقها على الخوارزميات الأخرى من حيث تقليل الخطأ بين البيانات التجريبية والمحاكاة. على الرغم من أن بعض الخوارزميات مثل (ZOA) و(SBOA) أظهرت سرعات تنفيذ أسرع، إلا أن (OFDA) حقق أوقاتًا تنافسية لوحدة المعالجة المركزية مع إظهار مستوى أعلى من التشابه بين منحنيات(P-V) و(I-V) المحاكية مقارنة بالبيانات التجريبية. الهدف الرئيسي من هذه الدراسة المقارنة هو اكتساب معرفة قيمة فيما يتعلق بفعالية خوارزميات الميتايورستك الناشئة المختلفة في تحديد المعلمات النموذجية للوحدات الكهروضوئية.

كلمات مفتاحية: الأنظمة الكهروضوئية *(PV)*، استخراج معلمات النموذج، خوارزميات الميتايورستك، نموذج الصمام الثنائي الفردي، نموذج الصمام الثنائي المزدوج، خوارزمية اتجاه التدفق المعارض *(OFDA)*.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

RMSE	root mean square error
SDM	Single diode model
DDM	Double diode model
TDM	Triple diode model
PVMM	PV module model
MaxIt	Maximum of iterations
Pop	Population
PV	Photovoltaic
SC	Solar cell
I-V	current–voltage
P-V	power–voltage
GRO	Gold Rush Optimizer
ZOA	Zebra Optimization Algorithm
SBOA	Secretary Bird Optimization Algorithm
HOA	Hippopotamus optimization algorithm
OFDA	Opposition Flow Directional Algorithm
CPU	Central Processing Unit
RAM	Random Access Memory

General introduction

Context

Photovoltaic systems have attracted researchers' attention due to their capability to convert solar energy directly into electrical energy and their wide range of applications in various fields. Environmental factors like extreme temperatures, dust, pollution, rain, and snow impact PV operating in harsh conditions. These elements can lead to malfunctions, shorten the lifespan of the PV models, and decrease the overall system efficiency. Modelling PV systems in detail is essential for designing new systems and optimizing the efficiency of existing ones. The model should consider the impact of environmental factors for an optimal PV system design. The modelling of the PV system begins with the PV cells, the smallest unit of the system, followed by the handling of modules and arrays.

The main components of PV cells are diodes, which is why the circuit models of PV systems are named based on the number of diodes they contain. Photovoltaic models are categorized into four types: the single-diode model, the double-diode model, the triple-diode model, and the PV module model(PVMM). SDM, PVMM, DDM, and TDM are also known as the five-parameter model, the seven-parameter model, and the nine-parameter model, respectively. As the number of diodes increases, extracting more parameters becomes necessary, leading to increased computational complexity. Having accurate parameters is crucial for achieving high performance in PV models. Therefore, the search for these parameters can be seen as an optimization problem.

A review of valuable literature studies revealed that metaheuristic methods were used for PV parameter extraction. Many researchers focus on these methods due to their advantages, including lack of problem constraints, user-friendliness, applicability to multidimensional optimization problems, and ability to deliver quick and reliable results.

Contribution

This study conducts a detailed comparative analysis of five recent metaheuristic algorithms: Opposition Flow Directional Algorithm(OFDA), Secretary Bird Optimization Algorithm (SBOA), Gold Rush Optimizer (GRO), Zebra Optimization Algorithm (ZOA), and Hippopotamus Optimization Algorithm (HOA), focusing on parameter estimation for PV models. The performance of these algorithms is assessed according to their accuracy, speed of convergence, and computational efficiency. By performing a contrastive analysis, the study aims to identify the most effective algorithm for parameter estimation in PV systems. This research contributes to the optimization literature and practical management of PV systems.

Statement of the Problem

This study focuses on the main issue of accurately estimating the parameters of PV models through optimization techniques. Traditional methods frequently fail to provide accurate results because of the nonlinear and multimodal characteristics of the PV parameter estimation problem. A comprehensive evaluation of recent metaheuristic algorithms is required to identify which ones demonstrate superior performance in terms of accuracy, convergence rate, and computational resources. This contrastive analysis aims to fill the gap in the current literature by providing insights into the performance of Opposition Flow Directional Algorithm (OFDA), Secretary Bird Optimization Algorithm (SBOA), Gold Rush Optimizer (GRO), Zebra Optimization Algorithm (ZOA), and Hippopotamus Optimization Algorithm (HOA). This analysis aims to guide practitioners and researchers in choosing the most suitable algorithm for PV parameter estimation.

Structure

This thesis is organized into three chapters as follows:

In Chapter 1 “Optimization and Metaheuristic Algorithms”, we start by defining optimisation and its distinctive methods. Furthermore, we explain metaheuristic algorithms, discuss their strategies and characteristics, and present a comprehensive classification of these algorithms. Finally, we discuss the “ no-free lunch theory”.

In Chapter 2 “Metaheuristic Algorithms and Real-Life Application”, we provide a concise overview of real-world optimisation problems. We cover real-world optimisation problem definitions, some basic characteristics and the significance of metaheuristic algorithms in solving real-world optimisation problems. Next, we present a step-by-step framework for generic metaheuristic algorithms in addition to the main components of the metaheuristic process in solving any optimisation problem. Finally, we provide some real-life applications of metaheuristics.

Chapter 3 “Parameter Estimation in Photovoltaic Models Problem” focuses on the practical part of our thesis where we discuss the issue of parameter estimation in photovoltaic (PV) models. We also talk about the source of inspiration for the algorithms used in our thesis. Next, we introduce the mathematical implementation and pseudo-code. Finally, we present the results of the algorithm and a comparative study of five metaheuristic algorithms namely: Gold Rush Optimizer (GRO), Zebra Optimization Algorithm (ZOA), Secretary Bird Optimization Algorithm (SBOA), Hippopotamus optimization algorithm (HOA), and Opposition Flow Directional Algorithm (OFDA).

Chapter 1

Optimization and Metaheuristic Algorithms

1.1 Introduction

In computer science some problems that are difficult to solve by classic methods and why not sometimes impossible to solve, are known as NP problems. To solve those problems metaheuristic techniques have been invented. Metaheuristic algorithms have become increasingly popular in the last two decades, and they are now among the most widely used algorithms for optimization.

In this first chapter, we shed light on optimisation and metaheuristic algorithms. We start by explaining the notion of optimisation and its methods. We explain metaheuristic algorithms and their characteristics. Then, we examine and discuss the following subtopics: strategies of metaheuristic algorithms, classification of metaheuristic algorithms and well-known metaheuristic algorithms.

1.2 Optimization

Optimization is the process of finding the best solution to a problem or achieving the best outcome based on specific criteria [15]. In formal terms, an optimization problem seeks to find a vector of variables, also known as unknowns or parameters, that minimizes or maximizes a function within the set, as defined by constraints on its variables [16].

Optimization is a crucial field of applied mathematics because of its diverse range of applications and the existence of effective algorithms. It involves finding the best possible solution to a problem by minimizing or maximizing a specific objective function, which depends on several decision variables. Additionally, the solution must also satisfy functional constraints. Optimization can be used in various fields to help achieve the most desirable outcome given the available resources and limitations. The availability of efficient algorithms to quickly and effectively solve optimization problems further highlights the significance of optimization [17].
Studies on optimization are conducted using various analysis methods and algorithms.

1.3 Optimization Methods

Optimization methods are designed to find the best possible solution or a solution very close to the optimal one while minimizing the computational effort required. These methods entail searching through a set of potential solutions based on a specified objective function and any constraints. There are two main types of optimization methods: exact optimization methods and approximate optimization methods. Each method is classified into two subcategories as shown in Figure 1.1 [18].

1.3.1 Exact Optimization Methods

The term ”exact optimization method” refers to a category of optimization techniques that strive to identify the global optimal solution for a specific problem. These methods depend on mathematical models and algorithms that ensure convergence to the best possible solution within a finite number of steps. Exact methods offer mathematically proven guarantees about the optimality of the solution, unlike heuristic or approximation methods.

We can distinguish two distinct subcategories of the exact method: Direct Method and Indirect Method.

Direct Method

Direct methods solve optimization problems by formulating and solving them exactly as they are. Examples include the Simplex Method, used for linear programming problems, and Branch and Bound, used for integer programming problems [1].

Indirect Method

Indirect methods, also known as iterative methods, solve optimization problems by breaking them down into a series of simpler problems. They reach the optimal solution through a sequence of iterations. Examples include Gradient Descent, which uses the gradient of the objective function to find the minimum and Newton's Method, which uses second-order information (Hessian matrix) to find the optimal solution more efficiently than gradient descent [1].

1.3.2 Approximate Optimization Methods

Approximate optimization methods, also referred to as heuristic optimization methods, are algorithms designed to find near-optimal solutions for complex optimization problems. Approximate methods are classified into two different subcategories: the Heuristic Method and the metaheuristic Method.

Heuristic Method

Heuristic Method is a technique designed for solving a problem more quickly when classic methods are too slow or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed [19].

Metaheuristic Method

Metaheuristics is a set of techniques used to guide the search process to efficiently explore the search space and find optimal or near-optimal solutions. These techniques vary from simple local search methods to complex learning algorithms [20].

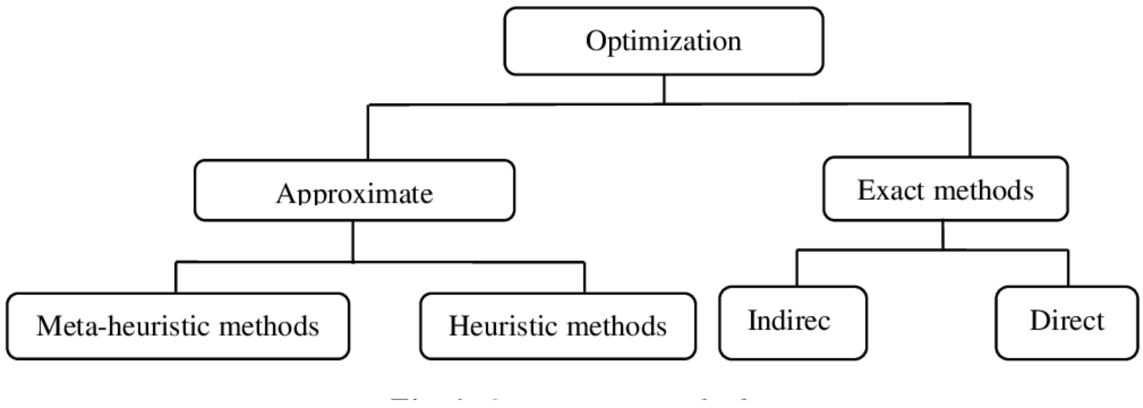


Figure 1.1: Optimization methods [1].

1.4 Optimization Algorithms

An optimization algorithm is a computational method used to find the best solution from a set of possible solutions by either maximizing or minimizing a specific objective function. These algorithms systematically adjust inputs, processes, or parameters to efficiently achieve the desired outcome. Algorithms used to solve optimization problems can also be referred to shortly as optimization algorithms.

Optimization algorithms are typically categorized into two groups: deterministic algorithms and stochastic algorithms. There is also a hybrid category that combines elements of both deterministic and stochastic algorithms.

1.4.1 Deterministic algorithms

Deterministic algorithms follow a strict and repeatable procedure. When given the same initial conditions, they consistently yield identical results each time they are executed. The subcategories of deterministic algorithms are listed below:

- **Gradient-Based Algorithms:** Use the gradient (derivatives) of the objective function to find the optimal solution.
- **Gradient-Free Algorithms:** Do not use derivatives but rely only on function values.
- **Classic Algorithms:** Generally well-established and follow a deterministic path [21].

1.4.2 Stochastic Algorithms

Stochastic algorithms introduce randomness into their search process, leading to unique paths in each run. They are beneficial for global optimization and tackling problems with numerous local optima. Below are the subcategories of stochastic algorithms:

- **Heuristic Algorithms:** Aim to find good enough solutions through trial and error within a reasonable time frame. They are not guaranteed to find the optimal solution.
- **Metaheuristic Algorithms:** Advanced heuristics generally perform better than simple heuristics. They use a balance of randomization and local search [21].

1.4.3 Hybrid Algorithms

Hybrid algorithms combine deterministic and stochastic methods to improve efficiency and effectiveness.

- **Deterministic-Stochastic Hybrids:** use deterministic methods with added random components to improve performance.

These categories help in selecting the suitable optimization algorithm depending on the problem's nature and the desired result. A classification of the optimization algorithms is shown in Figure 1.2 [21].

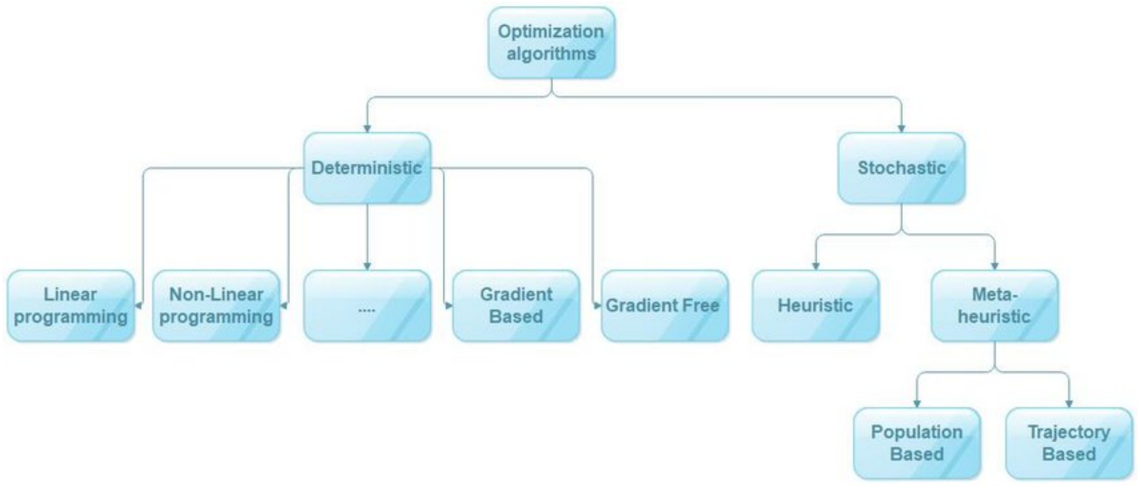


Figure 1.2: Classification of optimization algorithms.

1.5 Metaheuristic Algorithms

There are various definitions of metaheuristic algorithms. The term metaheuristic was first introduced by Glover in 1986. This term is derived from two Greek words 'meta' meaning "beyond in the upper level" and heuristic meaning "to find". Metaheuristic algorithms are stochastic search methods that combine local improvement with advanced strategies to avoid getting stuck in local optima.

In computer science, a metaheuristic refers to a computational method that optimizes a problem by iteratively attempting to enhance a candidate solution based on a specific quality measure. Metaheuristics do not rely on specific assumptions about the problem being optimized and can explore extensive solution spaces. However, they do not ensure finding an optimal solution. Many meta-heuristics use stochastic optimization. Other terms with similar meanings to meta-heuristic include derivative-free, direct search, black-box, or simply heuristic optimizer [22].

A metaheuristic is formally defined as an iterative generation process that guides a subordinate heuristic by intelligently combining different concepts to explore and exploit the search space. Learning strategies are employed to organize information effectively to find near-optimal solutions efficiently. Metaheuristic algorithms are approximate techniques that can be utilized to solve complex problems [23].

Metaheuristics are optimization algorithms capable of addressing complex, nonlinear problems and identifying satisfactory solutions without guaranteeing the global optimum. Metaheuristics use advanced strategies to search for solutions, unlike traditional optimization techniques that linearize the objective function or rely on derivatives and gradients. They are widely used in various industries and professions such as administration, planning, architecture, engineering, healthcare, and logistics. The effectiveness of metaheuristics in solving complex optimization problems has made them a popular choice in various applications. This group of optimization techniques, known as metaheuristics, guides the search process to deliver superior results. They are particularly useful in situations where it is not possible to create an explicit equation-based model [24].

Up to here, metaheuristic algorithms can handle large and complex search spaces, which traditional optimization methods cannot.

1.6 Characteristics of Metaheuristic Algorithms

Metaheuristic algorithms share several common characteristics that distinguish them from other classical methods. Here are some of the key characteristics.

1.6.1 Anytime Algorithms

All metaheuristics, as well as many other optimization and machine learning methods, are considered anytime algorithms. The concept behind anytime algorithms is that they begin with a (potentially poor) guess of what a good solution might be. Throughout the course, they aim to enhance the quality of their approximations by generating increasingly improved candidate solutions. At any given moment, we can determine the current best estimate of the optimum and choose to halt the optimization process if desired. This applies to optimization scenarios. It is often difficult to determine if the best solution we currently have is the globally optimal solution for the given problem instance. Therefore, we keep striving to enhance it until a termination criterion prompts us to stop or until the time runs out [25].

1.6.2 Return the Best-So-Far Candidate Solution

This concept is actually quite simple, but often overlooked and misunderstood by beginners in the field: No matter what the optimization algorithm does, it will never forget the best candidate solution found so far. In the formal definition of algorithms, there may not be explicit mention of it, but there is always a specific variable that stores the solution. This variable gets updated whenever a better solution is found and its value is returned when the algorithm finishes [25].

1.6.3 Randomization

Often, metaheuristics involve making random choices. When it's uncertain whether ”A” or ”B” is the better option, flipping a coin and choosing ”A” for heads or ”B” for tails is a practical approach. Random number generators are tools that offer functions capable of producing numbers within specific ranges, such as $[0, 1]$ or integer intervals. Each time we use such a function, it can yield any value within the specified range, but the exact value remains unknown until generated. The returned value should be independent of the values returned previously. In other words, based on past random numbers, we should not be able to predict the next one. By utilizing random number generators, algorithms can make random choices, select elements from a set randomly, or unpredictably modify a variable's value [25].

1.6.4 Black-Box Optimization

Metaheuristics are algorithms designed to solve various optimization problems using a single approach, commonly referred to as a black-box approach. These algorithms hide the details of the problems under the structural elements, focusing on rating candidate solutions and ensuring smaller ratings are better. There are various levels of general black-box metaheuristics, with many emphasizing the search operators and data structures employed, while others require a particular search space. These algorithms can be general as they make few assumptions about the objective function and the solution space.

A black-box metaheuristic is a versatile algorithm that can accommodate different representations, objective functions, and search operations required for a specific application. This concept is depicted in Figure 1.3 as the highest level of abstraction for solving complex problems [25].

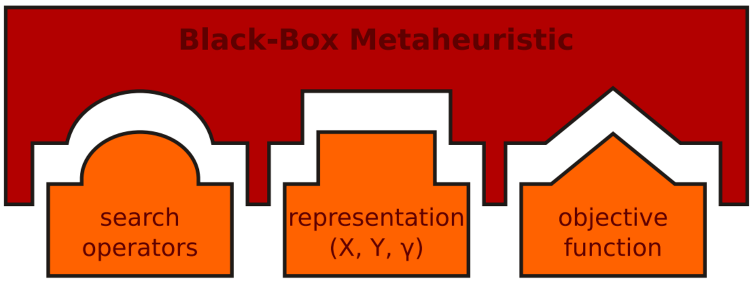


Figure 1.3: Black Box Metaheristic.

1.6.5 Heuristic Nature

Metaheuristics depend on problem-specific heuristics instead of complete problem information. They are designed to be used for a wide range of problems without needing in-depth problem-specific knowledge [25].

Other principal characteristics of Metaheuristic Algorithms include:

- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper-level strategy
- Today's more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

1.7 Strategies of Metaheuristic Algorithms

In every metaheuristic algorithm, exploration, exploitation, and avoiding local optima are crucial strategies for achieving success in solving optimization problems. It employs these strategies to find optimal solutions. However, researchers have not yet fully grasped how these strategies interact and affect each other.

1.7.1 Exploration

Exploration involves how effectively the operators diversify solutions within the search space, providing the metaheuristic with a global search behaviour. This aspect gives the metaheuristic a global search behaviour. Exploitation refers to how well the operators can utilize the information available from solutions from previous iterations to intensify the search. Such intensification gives the metaheuristic a local search characteristic. Some metaheuristics tend to be more explorative than exploitative, while others do the opposite [19].

1.7.2 Exploitation

Exploitation is the use of local knowledge of the search and solutions found so far so that new search moves can concentrate on the local regions or neighbourhood where the optimality may be close; however, this local optimum may not be the global optimality. The principal distinctions between metaheuristic algorithms are their attempt to balance exploration and exploitation [19].

1.7.3 Local Optima Avoidance

Local search algorithms generate potential configurations and modify them to reduce cost. They use a neighbourhood function and cost function to generate neighbouring solutions. However, they may become stuck in a local optimum after a limited number of iterations. Metaheuristic algorithms are designed to guide local search and prevent getting stuck in local optima.

To avoid local optima, it is necessary to introduce diversity and randomness into the algorithm. Diversity implies that a variety of solutions is generated and maintained, while randomness introduces chance or uncertainty into the algorithm to explore new regions of the search space. There are various ways to achieve diversity and randomness, depending on the type and structure of the algorithm [26].

In brief, a metaheuristic algorithm should be able to explore wide search spaces quickly, locate global solutions, and prevent falling into local optimum.

1.8 Classification of Metaheuristic Algorithms

Many classification criteria may be used for metaheuristics:

1.8.1 Nature inspired versus non-nature inspired

The most intuitive way to classify metaheuristics is based on the algorithm's origins. There are nature-inspired algorithms, such as Genetic Algorithms and Ant Algorithms, and non-nature-inspired ones like Tabu Search and Iterated Local Search [20].

1.8.2 Memory usage versus memoryless methods

A very important feature in classifying metaheuristics is how they use the search history ie whether they use memory or not. Memory-less algorithms perform a Markov process. Examples of such algorithms include local search and simulated annealing. While memory-based algorithms contain some information extracted online during the search. We usually differentiate between the use of short-term and long-term memory. Examples of such algorithms can be found in tabu search. The use of memory is nowadays recognized as one of the fundamental elements of a powerful metaheuristic [20].

1.8.3 Deterministic versus stochastic

A deterministic metaheuristic solves an optimization problem by making deterministic decisions for instance local search, and tabu search. In stochastic metaheuristics, some random rules are applied during the search for instance simulated annealing, and evolutionary algorithms. Deterministic algorithms obtain the final solution from the same initial solution, whereas stochastic metaheuristics can obtain different final solutions from the same initial solution. This characteristic must be taken into account in the performance evaluation of metaheuristic algorithms [20].

1.8.4 Population-based search versus single-solution based search

Metaheuristics can be classified based on the number of solutions they operate on simultaneously. Single-solution-based algorithms manipulate and transform a single solution during the search. These algorithms are known as trajectory methods and include local search and simulated annealing. Population-based algorithms, on the contrary, perform search processes in which a whole population of solutions is evolved. Examples of such algorithms include particle swarm and evolutionary algorithms [20].

1.8.5 Iterative versus greedy

In iterative algorithms, we start with a complete solution, or population of solutions, and transform it at each iteration using some search operators. Greedy algorithms start from an empty solution, and at each step, a decision variable of the problem is assigned until a complete solution is obtained. The majority of metaheuristics are iterative algorithms [20].

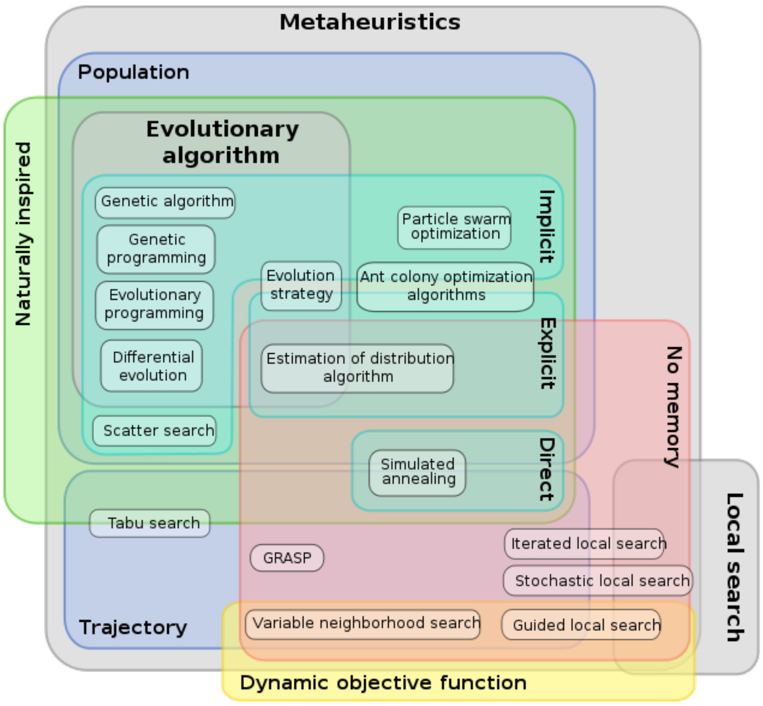


Figure 1.4: Classification of Metaheuristics

1.9 Well Known Metaheuristic Algorithms

Numerous metaheuristic algorithms have been developed over the years, each with its own strengths and weaknesses. We will briefly introduce some of the most well-known metaheuristic algorithms.

1.9.1 Genetic Algorithms (GA)

GA is inspired by Charles Darwin's theory of evolution, incorporating principles of genetics and natural selection. It was first used by J. Holland and his collaborators in the 1960s and 1970s. The process begins by randomly generating an initial population of solutions and then evolves the population by applying operators such as crossover and recombination, mutation, and selection. Each solution is represented by a string known as a chromosome. The crossover of two parent strings produces offsprings which are new solutions created by exchanging parts or genes of the chromosomes. On the other hand, Mutation involves flipping some digits of a string to create new solutions. These new solutions are then evaluated based on their fitness in each generation. The new solutions are chosen based on their fitness. Sometimes, to ensure that the best solutions remain in the population, they are passed on to the next generation with minimal changes. This is known as elitism. It has been successfully applied in fields like soft computing, health sciences and civil engineering.
GA has been widely used in optimization problems that require identifying the best combination of parameters. It has been successfully applied in fields like soft computing, health sciences and civil engineering [27].

1.9.2 Particle Swarm Optimization (PSO)

Particle swarm optimization was proposed by Kennedy and Eberhart in 1995. PSO is inspired by the collective behaviour of social organisms, such as birds flocking or fish schooling. PSO algorithm starts with a population (swarm) of random potential solutions (particles). Each particle moves in the search space using its position, velocity, and the best position found by the swarm so far. It is commonly used in optimization problems that require determining the optimal weight configuration in neural networks or other machine learning models. It is also used for route-finding issues, particularly in frequent changes, considering parameters like power, delay, and delivery rate, to construct optimal solutions and routing paths [27].

1.9.3 Simulated Annealing (SA)

Simulated annealing was developed by Kirkpatrick in 1983. It is inspired by the process of annealing in metallurgy. The basic idea of the simulated annealing algorithm is to employ random search through a Markov chain. This approach not only accepts changes that improve the objective function but also remains some changes that may not be optimal. It starts with generating an initial solution randomly, and then gradually reducing the temperature of the system. As the temperature decreases, the algorithm is more likely to accept suboptimal solutions to avoid getting stuck in local optima. SA has been Commonly used in optimization problems that require finding the best configuration of parameters in complex models or simulations [27].

1.9.4 Tabu Search (TS)

The Tabu Search algorithm was proposed by Glover in 1986. It is a metaheuristic algorithm inspired by the concept of memory in human decision-making. TS algorithm starts with randomly generating an initial solution, followed by exploring the solution's neighbourhood through operators like swapping or reversing. The algorithm utilizes a tabu list to store recently visited solutions and prevent revisiting them. Tabu Search is commonly applied in optimization problems requiring the identification of the optimal sequence of actions or decisions, like scheduling or routing [27].

1.9.5 Ant Colony Optimization (ACO)

In 1992, Dorigo developed a paradigm called Ant Colony Optimization, which is a cooperative search technique that imitates the foraging behaviour of real ant colonies.ACO is inspired by the behaviour of ants in finding the shortest path between their nest and a food source. Ants use pheromone as a chemical messenger and the pheromone concentration can also be considered as the indicator of quality solutions to a problem of interest. Ants start searching the area randomly surrounding their nest. Ants can construct the shortest path from their colony to the feed source and back using pheromone. When ants encounter an obstacle, initially, all ants have an equal chance of moving either right or left. Over time, more ants opt for the shorter path due to the higher concentration of pheromones on that path. So, the movement of an ant is influenced by the concentration of pheromones that gradually evaporate. Without this time-dependent evaporation, algorithms would prematurely converge to incorrect solutions. ACO has been widely used in optimization problems that involve finding the best routes such as in logistics and transportation [27].

1.10 No Free Lunch Theorems

The publication of the 'No Free Lunch Theorems for Optimization' in 1997 shocked the optimization community. Researchers have been always trying to find better algorithms for optimization, especially for tough NP-hard optimization problems. These theorems state that if algorithm A performs better than algorithm B for some optimization functions, then B will outperform A for other functions. That is to say, if averaged over all possible function space, both algorithms A and B will perform on average equally well.

Alternatively, there is no universally better algorithms exist. Then, people realized that we do not need the average overall possible functions for a given optimization problem. What we want is to find the best solutions, which have nothing to do with average overall possible function space. In addition, we can accept the fact that there is no universal or magical tool, but we do know from our experience that some algorithms indeed outperform others for given types of optimization problems. So, the research now focuses on finding the best and most efficient algorithms for a given problem. The objective is to design better algorithms for most types of problems, not for all the problems [23].

1.11 Conclusion

The first chapter comes to an end, we have discussed the importance of optimization which is crucial for solving complex problems. Various optimization methods, exact and approximate, are explored, highlighting the limitations of traditional approaches. We have also covered the basics as well as the definitions necessary for understanding metaheuristic algorithms as well as their characteristics strategies and classification. Well-known metaheuristic algorithms like Genetic Algorithms, Particle Swarm Optimization, Simulated Annealing, Tabu Search, and Ant Colony Optimization are reviewed, demonstrating their diverse applications and effectiveness in various fields. The No Free Lunch Theorems are briefly mentioned, emphasizing the need to choose appropriate algorithms based on specific problem contexts.

Chapter 2

Metaheuristic Algorithms and Real-Life Application

2.1 Introduction

Metaheuristics are a class of optimization algorithms that can handle complex, nonlinear problems and find a good solution without necessarily finding the global optimum. metaheuristics employ advanced strategies to search for a solution. They are extensively deployed in a wide range of domains, like education, robotics, sentiment analysis, finance, architecture, engineering, healthcare, and logistics. The efficiency of metaheuristics in solving difficult optimization problems has made them a popular choice in many applications.

This chapter presents a brief overview of real-world optimisation problems. The process of metaheuristic algorithms in solving any optimisation problem is described and their essential components are highlighted. Lastly, examples of real-world optimisation problems using metaheuristics are discussed.

2.2 Overview of Real-World Optimisation Problems

2.2.1 Definition of Real-World Optimisation Problems

An Optimization Problem is a problem that can be solved using optimization methods, and which seeks to find the best solution among a large set of possible solutions. In other words, an optimization problem involves finding the optimal solution that achieves the best possible outcome from a set of available options. These problems can arise in many different fields and can be addressed using various optimization techniques. Its goal is to find the solution that provides the most desirable results while satisfying any constraints or limitations that may be present.

The main fundamental concepts used in optimization problems are based on the following elements: The objective function: represents the primary goal of the model, which can be either minimized or maximized. It is also known as the optimization criterion, the cost function, or the fitness function. Variables (population or agents): they are a set of unknowns that influence the value of the objective function. They are adjusted iteratively throughout the optimization process to achieve optimal solutions. Constraints: they are a set of conditions that determine which values variables can take and which ones they must exclude. In essence, constraints outline the conditions that variables need to meet.

Optimization problems in the real world require identifying the optimal solution from a range of feasible options in fields like engineering, economics, logistics, and science. These problems usually involve optimizing one or more objectives within a set of constraints. They are considered ”real-world” as they arise in everyday practical situations and need to consider actual complexities and limitations [28].

2.2.2 Characteristics of Real-World Optimisation Problems

Real-world optimization problems are characterized by several features [29]:

Complexity

Complexity refers to the complex nature of real-world optimization problems, which involve high-dimensional search spaces, constraints, and challenges in accurately representing and solving these complex problems in optimization.

Presence of Constraints

Many real-world problems include constraints. Constraints in optimization problems are additional conditions or limitations that must be met in real-world scenarios. Understanding the types of constraints, evaluating their impact, and considering the quantity of constraints are crucial aspects of optimization problems.

High-Dimensionality

Several real-world problems have high-dimensional search spaces. These problems may involve numerous variables, resulting in a large search space.

Objective Types

Objective types are the various types of objectives found in real-world optimization problems. The objective types may include either Single Objective (SO) or Multi-Objective (MO). Single Objective (SO): Some real-world optimization problems involve a single objective function that requires optimization. Multi-Objective (MO): In other real-world optimization problems, there are multiple objective functions that need to be optimized simultaneously.

Non-linearity

Non-linearity refers to the relationship between the variables in the objective function or constraints is not linear. Non-linear optimization problems involve functions that do not satisfy the superposition principle, meaning that the output is not directly proportional to the input.

Dynamic and Stochastic Nature

The dynamic nature of a problem refers to situations where parameters, constraints, or objectives change over time, requiring algorithms to adjust to these variations. On the other hand, the stochastic nature involves uncertainty or randomness in problem components, which requires algorithms to make decisions based on probabilistic information.

Problem Domain

Real-world optimization problems can cover various domains, including engineering, computer science, mechanical engineering, medical engineering, robotics, engineering design, logistics, and scheduling.

2.2.3 The Significance of Metaheuristic Algorithms in Real-World Optimisation Problems

Traditional optimization algorithms are unsuitable for real-life applications due to practical reasons such as local search limitations and exhaustive algorithms being time-consuming and intractable. This has led to the development of heuristic optimization algorithms, such as evolutionary algorithms like differential evolution, and new features like hybridization and parallelization. The heuristic search involves making smart decisions based on minimal given information, obtaining relatively close results. However, it is crucial to determine if the ideal solution convergence will occur shortly or if there will always be a gap.

Metaheuristics are not specific to a particular problem. They are general strategies that help navigate the search process to explore the search space effectively. They aim to find satisfactory solutions within a reasonable time frame, rather than guaranteeing the discovery of the best solution. They aim to find an acceptable solution within a reasonable time frame rather than guaranteeing the best possible outcome. A prime example of modern metaheuristics is nature-inspired optimization methods, which consist of innovative algorithms inspired by nature. This belief stems from the common notion that nature offers the best solutions for a wide range of complex problems. I deas and concepts found in nature have been studied to develop algorithms that simulate them.

In recent decades, nature-inspired optimization methods have effectively addressed a wide range of real-world optimization problems, garnering significant interest from researchers across various fields. Nature-inspired optimization methods have successfully addressed a wide range of real-life optimization problems. Therefore, they have attracted significant attention from researchers across various domains, including evolutionary swarm intelligence methods. This is because they have advantages over conventional methods, as they require less domain-specific information. This feature is crucial when dealing with various optimization problems where obtaining domain-specific information is considered difficult. Their ideas were inspired by nature, as many natural systems excel at performing tasks. This is due to their common feature of seeking optimization. To find the best solution, they need to achieve objectives and meet constraints. This search for the best solution can be organized as an optimization problem, where the optimal solution is determined through an objective function assessment [30].

As a result, most real-world problems can be represented as optimization problems. metaheuristic optimization has become a crucial and widely applicable research field to solve these problems in areas like computer science, operations research and mathematics.

2.3 The Generic Process of Metaheuristic Algorithms

Metaheuristic algorithms operate within a fascinating framework that balances simplicity and complexity to solve optimization problems. The following is a general step-by-step optimisation process that may be used for any kind of metaheuristic algorithm [13].

2.3.1 Initialisation

Initial solutions can be generated randomly or through a heuristic that provides a good starting point. The choice between these methods can significantly influence the efficiency of the algorithm [13].

Initialize Iteration

The iteration counter is initialized to zero to start the algorithm. This counter will monitor the number of times the main loop has been executed.

Initialize Solutions

Initial solutions can be generated randomly or through a heuristic that provides a good starting point. The choice between these methods can significantly influence the efficiency of the algorithm.

Evaluate Each Member

The goal is to evaluate the quality of each initial solution by applying the objective function to determine its fitness. This step is crucial as it establishes a baseline for comparing future solutions.

Mark the Best Solution

Here, the aim is to identify and record the best solution found in the initial population. This requires comparing the fitness values of all solutions in the population and identifying the one with the highest (or lowest, depending on the problem) fitness.

2.3.2 Iterative improvement

The algorithm's main loop refines the population of solutions until the termination condition is met. This iterative process guarantees continuous progress towards a more optimal solution [13].

Choose Solutions Using SP

The purpose is to select a subset of solutions to serve as parents for generating new solutions. To do so, the Selection Plan (SP) is used to choose solutions from the current population. These selected solutions are typically those with higher fitness, although the specific method depends on the SP used.

Generate New Solutions Using GP

A Generation Plan (GP) is used to create new solutions to explore the search space. The solutions are selected to produce new offspring through strategies like crossover, mutation, or other methods to combine and alter the selected solutions.

Choose Solutions Using RP

The aim is to decide which solutions from the current population will be replaced. The Replacement Plan (RP) is utilized to select solutions from the current population that will be replaced by the new solutions generated in the previous step. This helps maintain diversity and allows for the exploration of new areas in the search space.

Update and Replacing Solutions with a Combination of Solutions from Using UP

The Update Plan is used to integrate new solutions into the population and ensure continuity in the search process by combining and replacing existing solutions with new ones. This step ensures the creation of a new generation of solutions by balancing the exploitation of known good areas with the exploration of new areas.

2.3.3 Solution Evaluation

Evaluate Each Member

Fitness functions evaluate the quality of each solution within the updated population, providing crucial feedback to guide the next iteration of the algorithm. Fitness functions establish a clear objective for the algorithm to pursue, enhancing the optimization process by making it more focused and effective [13].

Identify and Update the Best Solution

To track the best solution found so far, compare the fitness of all solutions in the updated population and update if a better solution is found. This will ensure that the algorithm maintains a record of the best solution discovered throughout the search process [13].

2.3.4 Termination Criteria

Once the termination condition is met, such as reaching the maximum number of iterations, convergence, or satisfactory fitness level, the algorithm stops. The best solution discovered during the search is then declared as the near-optimal solution [13].

By following these steps shown in algorithm 1, the algorithm refines the population of solutions iteratively. It balances the exploration of new areas with the exploitation of known good areas until it converges on a near-optimal solution for the problem.

Algorithm 1 A Generic Metaheuristic Framework [13]

```
Require: SP, GP, RP, UP, TP, $N(\geq 1)$, $\mu(\leq N)$, $\lambda(\leq N)$, $\rho(\leq N)$
 1: Set iteration counter $t = 0$
 2: Initialize $N$ solutions $S_t$ randomly
 3: Evaluate each member of $S_t$
 4: Mark the best solution of $S_t$ as $\mathbf{x}_t^*$
 5: repeat
 6:     Choose $\mu$ solutions (set $P_t$) from $S_t$ using a selection plan (SP)
 7:     Generate $\lambda$ new solutions (set $C_t$) from $P_t$ using a generation plan (GP)
 8:     Choose $\rho$ solutions (set $R_t$) from $S_t$ using a replacement plan (RP)
 9:     Update $S_t$ by replacing $R_t$ using $\rho$ solutions from a pool of any combination of at most three
    sets $P_t$, $C_t$, and $R_t$ using an updated plan (UP)
10:     Evaluate each member of $S_t$
11:     Identify the best solution of $S_t$ and update $\mathbf{x}_t^*$
12:     $t \leftarrow t + 1$
13: until a termination plan (TP) is satisfied
14: Declare the near-optimal solution as $\mathbf{x}_t^*$
```

2.4 The Components of Real-World Optimisation Problems' Process

An optimization algorithm, a simulation, and a realistic representation are fundamental and essential for addressing optimization problems in different domains. These components work cooperatively in the optimization process to facilitate the search for optimal solutions within the design space, taking into account constraints and objectives [31]. Here's a detailed explanation of these components:

2.4.1 Optimisation Algorithm

The optimization algorithm is the core of the optimization process, responsible for searching through possible solutions to find the optimal one. This method or technique defines how the search for the best solution is carried out. In many cases, we may not know which algorithm is most suitable for a specific problem until we try it. There could be a range of efficient algorithms available to solve these problems, and in some instances, new algorithms can still be developed.
Thus, the selection of algorithms, even existing ones, largely depends on the decision-maker's expertise, available resources, and the nature of the problem. It is ideal to use the most suitable algorithms and tools to solve a specific problem. The proper utilization of these tools may rely on the user's experience. Factors like computing costs, software availability, and solution production time also play a crucial role in determining which algorithms and methods to use. It is worth noting that having specific knowledge of a problem is always helpful in choosing the best and most efficient optimization methods [31].

In short, the algorithm iteratively explores the search space, evaluates potential solutions using the numerical simulator ie. simulation, and updates the search direction based on the evaluation results.

2.4.2 Simulation

Simulation is a widely used technique for studying the behaviour of Discrete Event Systems. It is commonly employed to analyze the intricate workings of manufacturing systems, logistics systems, healthcare systems, and more, to estimate their key performance indicators like throughput, flow times, resource utilization, etc. Simulation is particularly used in situations where it is impossible to define analytical mathematical expressions to describe system behaviour. This is due to the high complexity and various sources of randomness that are characteristic of most real-world systems. The main characteristic of simulation is the ability to predict system performance implicitly, without requiring the analyst to define complex mathematical equations to model the system.

A simulation model is usually a computer code used to predict the performance of a system with a specific configuration. The optimization model is an algorithm built on top of the simulation, which looks for the best configuration based on defined criteria (see Figure 2.1)

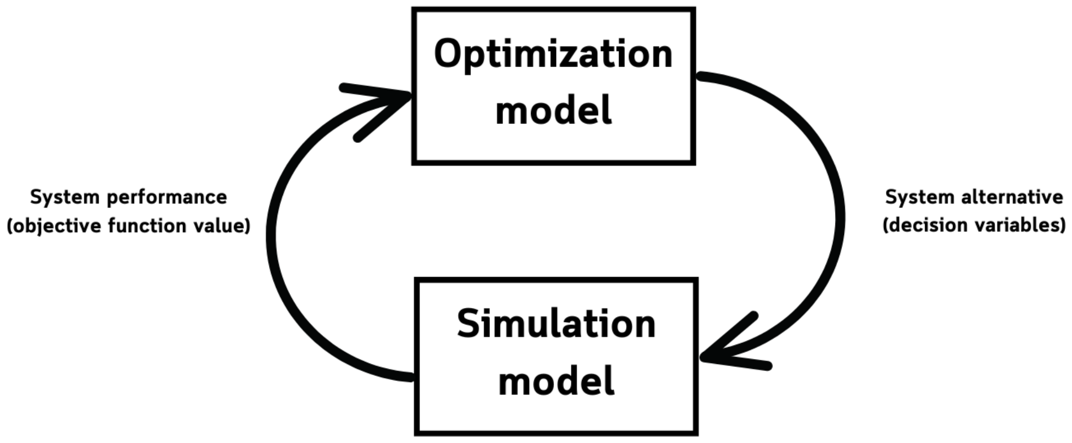

Optimization model
System performance (objective function value)
System alternative (decision variables)
Simulation model

Figure 2.1: Optimization for Simulation.

Simulation improves the optimization process by offering precise evaluations, modelling intricate systems, ensuring feasibility, conducting sensitivity analysis, allowing for iterative enhancements, developing surrogate models, and verifying solutions. This integration is crucial for addressing real-world problems that cannot be solved by analytical methods alone [32].

2.4.3 Realistic Representation

According to Xin-She Yang [31], realistic representation likely refers to accurately modelling the physical systems or processes being optimized. This involves creating mathematical models that closely reflect the behaviour and constraints of real-world systems in order to perform effective optimization. In optimization, realistic representation and mathematical modelling are used interchangeably. In other words, realistic representation in the context of optimization typically involves mathematical modelling of the physical processes or systems being studied. This modelling aims to represent the fundamental features, behaviours, and constraints of the real-world system in a mathematical form suitable for simulation and optimization. Mathematical modelling enables engineers and researchers to depict the relationships among various variables, parameters, and constraints that influence the system's behaviour. By formulating these relationships mathematically, it is possible to simulate the system's response to various inputs and conditions, assess performance metrics, and optimize design variables to meet specific objectives. Mathematical modelling is a powerful tool used in different disciplines like science and engineering to describe and analyze real-world problems. It involves representing the problem through mathematical equations, formulas, and algorithms. Mathematical modelling is a crucial tool in today's complex world, enhancing our understanding of systems and enabling efficient solutions for various problems [33].

By effectively integrating these components, engineers and researchers can tackle a variety of optimization problems, including engineering design, resource allocation, parameter tuning, and system optimization. The synergy among the optimization algorithm, simulation, and realistic representation is essential for achieving meaningful and practical solutions in optimization efforts.

2.5 Examples of Real-World Optimisation Problems Using Metaheuristics

Metaheuristic algorithms have been used in a variety of optimization problems across different fields like engineering, finance, logistics, and healthcare. We are going to explore some significant applications of metaheuristics.

2.5.1 Engineering

Metaheuristic algorithms are commonly utilized in engineering to solve a variety of optimization problems. Here are some notable applications of metaheuristic algorithms in engineering.

Parameter Estimation

Researchers use the Dynamic Model-Based Parameter Estimation (DMbPE) process to calibrate a dynamic system model for accurate experimental data reproduction by minimizing an objective function that evaluates the fit quality between predicted and observed values. The article "On Metaheuristics for Solving the Parameter Estimation Problem in Dynamic Systems: A Comparative Study" [34] evaluates the effectiveness of five metaheuristic algorithms for parameter estimation in dynamic systems. These algorithms, including the Firefly Algorithm, Harmony Search Algorithm, Differential Evolution Algorithm, Artificial Bee Colony Algorithm, and Directed Tabu Search, are widely used in real-world applications. The study compares the performance of algorithms on benchmark functions and real-world datasets. It concludes that no single algorithm is superior in all situations, and the selection of an algorithm should be based on specific problem characteristics.

Design Optimization

Optimization in engineering design involves solving complex problem-solving challenges within constraints, aiming to minimize effort and time, or achieve maximum efficiency, in fields like industry, automotive, construction, and machinery. Metaheuristic optimization approaches offer a fast and accurate solution to global optimization problems, eliminating the need for continuous cost functions and variables. They are commonly used in fields like structural design, and engineering. "A Comparative Study of Metaheuristic Optimization Algorithms for Solving Real-World Engineering Design Problems" [35] presents The Chimp Optimization Algorithm (ChOA) to solve constrained engineering design problems. The study compares 18 metaheuristic optimization algorithms, including The Chimp Optimization Algorithm (ChOA). It finds ChOA promising for complex problems, making it a valuable tool for engineers and researchers.

2.5.2 Finance

Metaheuristic algorithms are commonly utilized in finance to address a range of optimization problems. Below are some significant applications of metaheuristic algorithms in the field of finance.

Portfolio Optimization

Portfolio optimization is a crucial finance research area, involving allocating capital among assets. Harry Markowitz introduced the mean-variance model, which balances expected return and risk. This multi-objective nonlinear optimization problem requires discrete decision variables and multiple objectives, making exact algorithms unfeasible. The study ”Meta-heuristics for portfolio optimization” [36] explores the use of metaheuristics for portfolio optimization, utilizing various algorithms such as Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, Simulated Annealing, and Tabu Search. The study indicates that multi-objective metaheuristics are efficient for solving complex portfolio optimization problems. They offer a practical approach to determining the efficient frontier and handling intricate constraints and objectives.

Fraud Detection

Banks use rule-based and statistical analysis to prevent credit card fraud, but detecting statistical fraud is challenging due to skewed data, evolving techniques, and limited privacy exchanges. Fraud detection optimization is crucial in industries like banking and e-commerce to identify fraudulent transactions, minimize false positives, and minimize financial losses. It requires adaptive models, real-time detection, and data privacy constraints to achieve high accuracy. The study ”Solving Credit Card Fraud Detection Problem by the New Metaheuristics Migrating Birds Optimization” [37] applies the recently developed new metaheuristics algorithm namely the migrating birds optimization algorithm (MBO) to this problem in which offer significant contributions to fraud detection. Results show that the MBO algorithm performed better than traditional approaches and some other metaheuristics and it performed better than a previous implementation of a combination of genetic algorithms and scatter search. Overall, Metaheuristic algorithms such as MBO offer a robust and efficient method for optimizing fraud detection systems. They provide a balance of accuracy, adaptability, and computational efficiency.

2.5.3 Healthcare

Metaheuristic algorithms have been effectively used in different healthcare fields, including medical diagnosis, treatment planning, drug discovery, and healthcare management. Some specific applications of metaheuristic algorithms in healthcare are:

Healthcare Scheduling

Home Health Care Routing and Scheduling Problem (HHCRSP) involves the efficient routing and scheduling of home health care services to patients while considering various constraints and objectives such as minimizing costs, maximizing patient satisfaction, and optimizing staff assignments and travel times. It is a complex optimization problem requiring careful consideration of staff and vehicle routing and scheduling aspects to minimize costs and maximize patient satisfaction. The article ”Using a metaheuristic algorithm for solving a home health care routing and scheduling problem” [38] aims to offer insights on enhancing the efficiency and quality of home health care operations through effective optimization techniques. This article proposes using the Simulated Annealing (SA) metaheuristic algorithm to solve the Home Health Care Routing and Scheduling Problem (HHCRSP). The results show that using Simulated Annealing as a metaheuristic algorithm effectively addresses the complexity of the Home Health Care Routing and Scheduling Problem, providing reasonable solutions within a reasonable computational time.

Disease Diagnosis

The modern medical field greatly benefits from advanced computing facilities and the latest technologies. Computer-aided automated processes in body examinations and disease diagnosis result in enhanced performance and treatment outcomes. The increasing amount of healthcare data and the rapid expansion of database storage in clinics highlight the urgent need to access this vast data for efficient disease diagnosis and treatment. In this era, numerous researchers are focusing on automating disease diagnosis and prediction procedures. Metaheuristic algorithms are techniques capable of providing practical solutions to various issues. In " A Systematic Review on Metaheuristic Optimization Techniques for Feature Selections in Disease Diagnosis: Open Issues and Challenges" [39] article, various metaheuristic algorithms, such as Spider Monkey Optimization (SMO), Shuffled Frog Leaping Algorithm (SFLA), Cuckoo Search Optimization (CSO), Ant lion Optimization (ALO), Lion Optimization technique (LO), Moth Flame Optimization (MFO), Bat-inspired Algorithm (BA), Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), and Dragonfly Algorithm (DA) have been utilized to detect and classify different health-related diseases. According to this paper, researching the use of metaheuristic algorithms could provide a great opportunity for developing models to diagnose diseases effectively, benefiting both doctors and patients.

2.5.4 Machine Learning

Neural Network for Image Classification

The COVID-19 pandemic has highlighted the significance of wearing face masks to prevent the spread of the virus and protect millions of people. Deep learning and machine learning are two widely used subsets of artificial intelligence for face mask detection and classification. " Comparative Study of Metaheuristic Optimization of Convolutional Neural Networks Applied to Face Mask Classification" [40]is a research focuses on using Convolutional Neural Networks (CNNs) to classify face mask usage into three categories: no mask, incorrect mask, and proper mask. The research compares the effectiveness of four swarm intelligent metaheuristic optimization algorithms: Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Bat Algorithm (BA), and Whale Optimization Algorithm (WOA), in designing optimal CNN architectures for accurate face mask classification. The results showed that the Bat Algorithm (BA) performed better than other metaheuristics. These findings provide valuable insights into using metaheuristic optimization techniques to enhance the efficiency and accuracy of convolutional neural networks for face mask classification.

Feature Selection

Feature selection is a crucial and complex task in machine learning. It has various applications in different fields. Some examples include biomedical problems, text mining, and image analysis. Feature selection problem involves identifying and removing inappropriate, irrelevant, or unnecessary features from datasets, aiming to extract the most relevant ones. Hence, metaheuristic algorithms are receiving significant attention for addressing these conditions. The study: "Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019)" [41] presents a comprehensive literature review of metaheuristic algorithms developed between 2009 and 2019 and their applications in feature selection problems. This study introduced five algorithms: Binary Particle Swarm Optimization (BPSO), Binary Differential Evolution (BDE), Binary Ant Lion Optimizer (BALO), Binary Grey Wolf Optimizer (BGWO) and Binary gaining Sharing Knowledge Based Algorithm (FS-NBGSK) to assess the performance of metaheuristic algorithms on the datasets. The main purpose of feature selection is to maximize performance accuracy while minimizing the number of selected features. The results indicate that the FS-NBGSK algorithm converges to the minimum fitness value more effectively than other algorithms. Overall, the study emphasizes the potential of binary metaheuristic algorithms in solving feature selection problems. It highlights their advantages in improving classification performance by efficiently reducing feature dimensionality.

2.6 Conclusion

In this chapter, we explored the nature and characteristics of real-world optimization problems We have discussed the outlined metaheuristic process from initialization to termination, emphasizing important components like algorithms, simulation, and realistic representation. By showcasing examples like feature selection and parameter estimation, we show how metaheuristics are applied in different fields like machine learning and dynamic system modelling. For instance, feature selection and parameter estimation demonstrate how metaheuristic algorithms can effectively explore intricate search spaces, producing high-quality solutions that improve performance and accuracy in areas like machine learning and dynamic system modelling.

Based on this, we are interested in solving parameter estimation problems. We have decided to focus on studying parameter estimation in PV systems using various metaheuristic algorithms. This will be the subject of the next chapter.

Chapter 3

Parameter estimation in photovoltaic models problem

3.1 Introduction

In this chapter, we present a comparison of various metaheuristic algorithms used for extracting electrical parameters from specific photovoltaic (PV) panels. These algorithms include the Gold Rush Optimizer (GRO), Zebra Optimization Algorithm (ZOA), Secretary Bird Optimization Algorithm (SBOA), Hippopotamus Optimization Algorithm (HOA), and Opposition Flow Directional Algorithm (OFDA). This chapter aims to explore the effectiveness of these new algorithms in estimating PV model parameters. Renewable energy is increasingly being embraced as a sustainable substitute for fossil fuels across many sectors. Solar energy, in particular, has become a key component in global power systems because it is plentiful, environmentally friendly, and simple to install and maintain. Photovoltaic (PV) cells transform solar energy into electricity. To enhance the efficiency of PV cells, mathematical modelling is essential for simulating their performance and determining the best parameters. Finding the right parameters for PV models is an optimization challenge that can be effectively addressed with robust optimization techniques. Among the metaheuristic techniques employed to determine the parameters of photovoltaic cells, we can mention the walrus optimization algorithm (WaOA), Cheetah optimizer (CO), Enhanced prairie dog optimization algorithm (En-PDO), Coati Optimization Algorithm (COA), Aquila optimization (AO), Harris Hawks optimization (HHO), Wild Horse Optimizer (WHO), Honey Badger Algorithm (HBA) and African Vulture Optimization Algorithm (AVOA). Table 3.1 presents a comparative analysis of these algorithms based on their RMSE for (SDM) and (DDM), along with their maximum iterations (MaxIt) and population size (Pop).

Algorithm	**Year**	**RMSE**		**MaxIt**	**Pop**
		SDM	**DDM**		
WaOA [42]	2024	7.730062E-04	7.631566E-04	200	20
CO [42]	2024	7.7912E-04	7.757378E-04	200	20
En-PDO [43]	2024	7.7299E-04	7.4248E-04	1000	50
COA [3]	2024	7.73006E-04	7.41936E-04	Not reported	Not reported
AO [44]	2023	1.3481E-03	Not reported	50000	30
HHO [44]	2023	2.5000E-05	Not reported	50000	30
WHO [45]	2023	9.8602E-04	9.8248E-04	10000	50
HBA [46]	2022	7.7301E-04	7.60042E-4	1000	100
AVOA [46]	2022	7.85654E-04	7.74523E-04	1000	100
SEDE [10]	2020	9.8602188E-04	9.8248485E-04	50000	30

Table 3.1: Nature-Inspired Meta-heuristics for Solar Parameter Estimation.

The rest of the chapter is organized as follows:
Section 3.2 describes the problem and its significance. Then, Section 3.3 provides a detailed description of the proposed meta-heuristic algorithms, including GRO, ZOA, SBOA, HOA, and OFDA. Finally, Section 3.4 presents a comparative study of the proposed algorithms.

3.2 Description of the problem

The sun has around 173 trillion terawatts of solar energy, which is 10,000 times more than the world's population utilizes. Solar energy is created as both heat and light. Therefore, it can be harnessed into heat energy or electricity.

Solar photovoltaic (PV) power plants are a popular method for generating electricity through the photovoltaic effect. Solar cells, made from semiconductor materials, convert sunlight into electricity by absorbing light and releasing electrons, which then travel through circuits to the electricity grid. As shown in Figure 3.1, PV power plants consist of grids of PV modules, each containing a layer of thin-film material with internally connected solar cells. These modules protect the solar cells from the environment and produce higher voltage than a single cell. The efficiency of electricity production depends significantly on the performance of PV modules and cells, which must adapt to environmental and climatic changes [2].

To achieve higher conversion efficiency under varying weather and temperature conditions, it is vital to simulate, optimize, and control the PV model. Commonly used models include the single diode (SD) and double diode (DD) models. The goal is to find parameter values that closely match experimental data to maximize performance under specific conditions. Accurate parameters are essential for high performance, making parameter search an optimization problem [10].

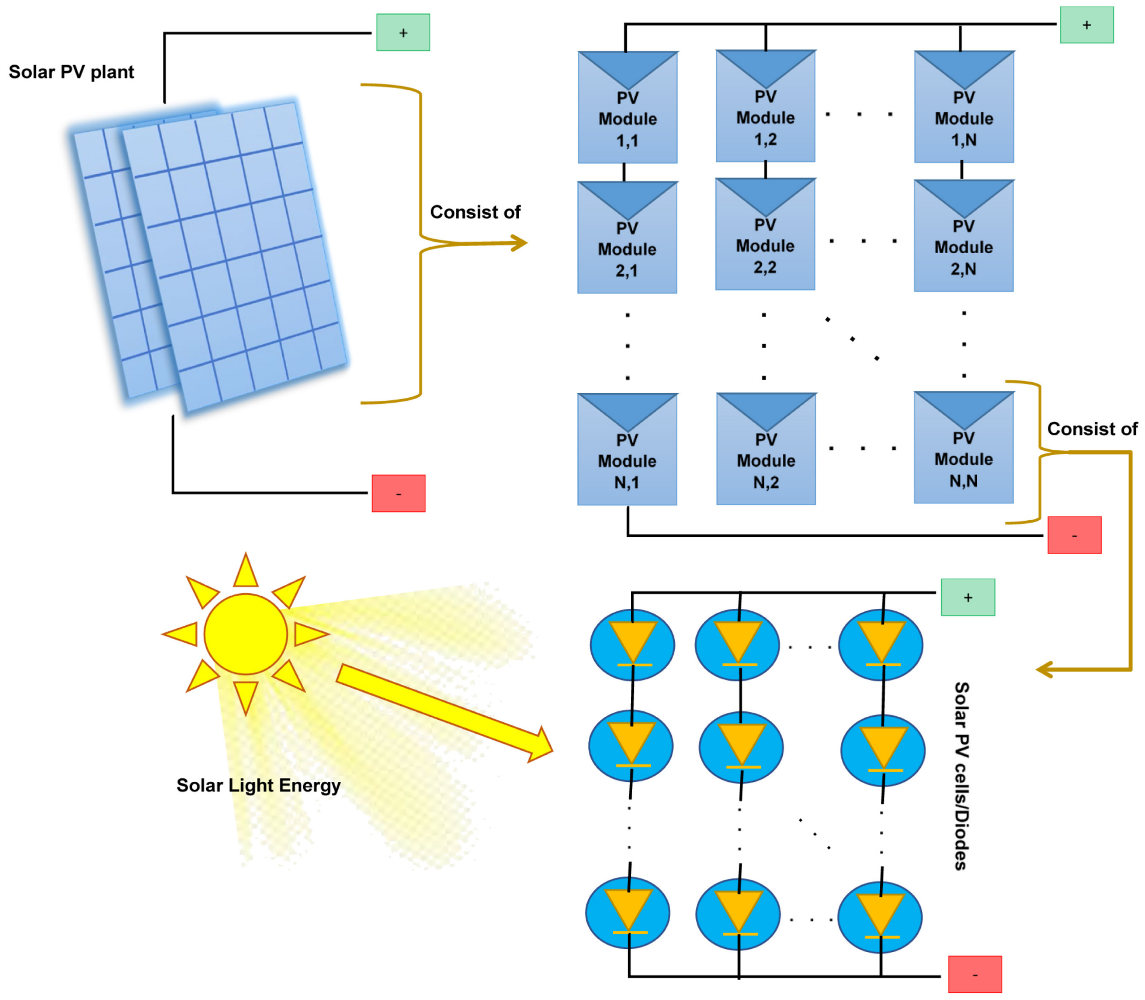


Figure 3.1: Solar PV Systems [2].

3.2.1 Photovoltaic Models

To design the SC and PV modules, a mathematical model is required to extract the SC parameters analytically. Based on this, electronic circuits using diodes are utilized for modelling the SC [47]. Photovoltaic models fall into four categories: single-diode model(SDM), double-diode model (DDM), triple-diode model (TDM) and PV module model(PVMM).

Single Diode Model

The SDM (Single Diode Model) represents a simple configuration for modelling a solar PV cell. It consists of a diode and two resistors, one in series and one in parallel Table 3.2, to account for various losses [48]. Due to its simplicity and minimal number of parameters (only five unknown parameters need to be estimated) Equation 3.1 [10], the SDM is the most widely used model. This makes it particularly suitable for cases where fast estimation is required with low manufacturing costs [49].

Double Diode Model

The DDM (Double Diode Model) is composed of two diodes and two resistors, one in series and one in parallel Table 3.3, to compensate for various losses [48]. This configuration makes the DDM more appropriate for the low irradiance level operation of PV cells. With seven unknown parameters to be estimated Equation 3.2 [5], the DDM is suitable for cases where accurate I-V estimation is needed. Additionally, it can easily consider the variations of environmental conditions during simulation [49].

Triple Diode Model

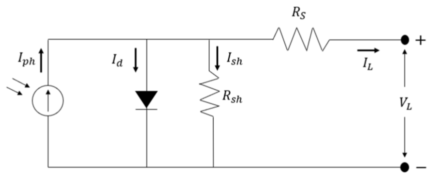
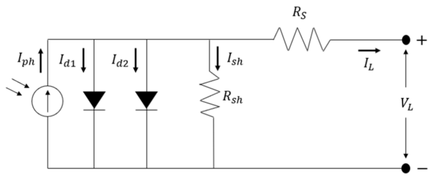
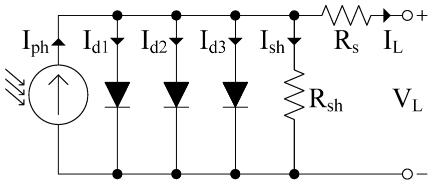
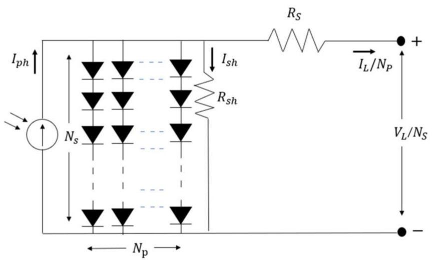
The third model, TDM (Triple Diode Model), is used for industrial applications. The TDM extends the DDM by adding an additional diode in parallel with the other two diodes Table 3.4. With nine unknown parameters to be estimated Equation 3.3 [5], the TDM is suitable for predicting values with high accuracy and can easily model the complex nature of different PV module types [49].

PV module Model

Different from the above three models, the PV module typically includes several solar cells connected in parallel or series. Based on the circuit presented in Table 3.5, the output current can be calculated by Equation 3.4. where N_p and N_s indicate the size of solar cells used in parallel and series, respectively. With five unknown parameters to be estimated in the PV module [10].

The parameter meanings, the equivalent circuits and the mathematical equations of the SDM, DDM, TDM and PVMM are represented in Table 3.2.

Table 3.2: Summary on four photovoltaic (PV) cell models.

Model	Electrical equivalent circuit	Mathematical model
Single diode model (SDM)	 R_S I_{ph} I_d I_{sh} R_{sh} I_L V_L + − **Figure 3.2:** SDM [3]	$I_L = I_{ph} - I_d - I_{sh}$ $I_L = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}}$ (3.1) 05 parameters to be estimated: I_{ph}, I_{sd}, R_s, R_{sh} and n [10]
Double diode model (DDM)	 R_S I_{ph} I_{d1} I_{d2} I_{sh} R_{sh} I_L V_L + − **Figure 3.3:** DDM [3]	$I_L = I_{ph} - I_{d1} - I_{d2} - I_{sh}$ $= I_{ph} - I_{sd1} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n_1 \cdot k \cdot T}\right) - 1\right]$ $- I_{sd2} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_{S} \cdot I_L)}{n_2 \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}}$ (3.2) 07 parameters to be estimated: I_{ph}, I_{sd1}, I_{sd2}, R_s, R_{sh}, $n1$ and $n2$ [10]
Triple diode model (TDM)	 Iph Id1 Id2 Id3 Ish Rs IL Rsh VL + − **Figure 3.4:** TDM [4]	$I = I_{ph} - \sum_{j=1\to3} I_{dj} - I_{sh}$ $= I_{ph} - \sum_{j=1\to3} I_{sdj} \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n_j \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}}$ (3.3) 09 parameters to be estimated: I_{ph}, I_{sd1}, I_{sd2}, I_{sd3}, R_s, R_{sh}, $n1$, $n2$ and $n3$ [50]
PV module model (PVMM)	 R_S I_{ph} I_{sh} R_{sh} N_s N_p I_L/N_p V_L/N_s + − **Figure 3.5:** PVMM [3]	$I_L/N_p = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L/N_S + R_S \cdot I_L/N_p)}{n \cdot k \cdot T}\right) - 1\right] - \frac{V_L/N_S + R_S \cdot I_L/N_p}{R_{sh}}$ (3.4) 05 parameters to be estimated: I_{ph}, I_{sd}, R_s, R_{sh} and n [10]

Note: $I_{ph}(A)$,photo-generated current;I_{sd},I_{sd1},I_{sd2},I_{sd3} (A),reverse saturation current;n,n1,n2,n3,ideality factor;$R_s(\Omega)$,series resistance; $R_{sh}(\Omega)$,shunt resistance; V_L,output voltage;I_L,PV current; k,boltzmann constant $k = 1.3806503 \times 10^{-23}$ J/K; q,electron charge $q = 1.60217646 \times 10^{-19}$C;T,cell temperature °K [3]

3.2.2 Problem Formulation

The Metaheuristic algorithm selection process will preserve exceptional individuals according to the values of their objective function, so the creation of an objective function is essential [10].

The primary objective of parameter extraction in both the SD and DD models is to identify model parameters that minimize the discrepancy between the computed and observed current. One popular technique for calculating the difference between two I-V curves is the root-mean-square error or RMSE [51].

The SD and DD model calculation methods are given by equations Equation 3.5 and Equation 3.6, respectively. The ultimate goal is then ascertained by the total error evaluated by using the root means square error (RMSE), as demonstrated by Equation 3.7 [10].

- SDM

$$\left\{\begin{array}{l} f_{SD}\left(V_L, I_L, \mathbf{X}\right) = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q\cdot(V_L+RS\cdot I_L)}{n\cdot k\cdot T}\right) - 1\right] - \frac{V_L+R_S\cdot I_L}{R_{sh}} - I_L \\ \mathbf{X} = \{I_{ph}, I_{sd}, R_S, R_{Sh}, n\} \end{array}\right. \tag{3.5}$$

- DDM

$$\left\{\begin{aligned} f_{DD}\left(V_L, I_L, \mathbf{X}\right) = & I_{ph} - I_{sd1} \cdot \left[\exp\left(\frac{q\cdot(V_L+R_S\cdot I_L)}{n_1\cdot k\cdot T}\right) - 1\right] \\ & - I_{sd2} \cdot \left[\exp\left(\frac{q\cdot(V_L+R_S\cdot I_L)}{n_2\cdot k\cdot T}\right) - 1\right] - \frac{V_L+R_{S\cdot}\cdot I_L}{R_{sh}} - I_L \\ & \mathbf{X} = \{I_{ph}, I_{sd1}, I_{sd2}, R_S, R_{Sh}, n_1, n_2\} \end{aligned}\right. \tag{3.6}$$

$$\mathrm{RMSE}(\mathbf{X}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} f_k\left(V_L, I_L, \mathbf{X}\right)^2} \tag{3.7}$$

Where V_L and I_L represent the experimental values of the solar cell's voltage and current Table 3.4, X indicates the solution consists of several unknown parameters and N shows the number of experimental data. However, the upper/lower boundaries that control the study search space of the PV model's unknown parameters are presented in Table 3.3 [10].

Parameters	$I_{ph}(A)$	$I_{sd}, I_{sd1}, I_{sd2}(A)$	$R_s(\Omega)$	$R_{sh}(\Omega)$	n, n_1, n_2
LB	0	0	0	0	1
UB	1	1	0.5	100	2

Table 3.3: Parameter ranges of SDM and DDM models [10].

Data	V_L(V) Measured	I_L(A) Measured	Data	V_L(V) Measured	I_L(A) Measured
1	-0.2057	0.7640	14	0.4137	0.7280
2	-0.1291	0.7620	15	0.4373	0.7065
3	-0.0588	0.7605	16	0.4590	0.6755
4	0.0057	0.7605	17	0.4784	0.6320
5	0.0646	0.7600	18	0.9460	0.5730
6	0.1185	0.7590	19	0.5119	0.4990
7	0.1678	0.7570	20	0.5265	0.4130
8	0.2132	0.7570	21	0.5398	0.3165
9	0.2545	0.7555	22	0.5521	0.2120
10	0.2924	0.7540	23	0.5633	0.1035
11	0.3269	0.7505	24	0.5736	-0.0100
12	0.3585	0.7465	25	0.5833	-0.1230
13	0.3873	0.7385	26	0.5900	-0.2100

Table 3.4: Terminal measurements of voltage and current [11].

3.3 Description of the Proposed Metaheuristic Algorithms

Accurate extraction of PV model parameters remains difficult to work due to the various characteristic types (non-linear, multi-variable, and multi-modal), as well as the limited information data provided by manufacturers.

As a result, in terms of non-linearity, multi-variability, and multi-modality difficulties in PV cell/module IV curves, metaheuristic algorithms appear to be a very efficient solution to overcome the constraints of analytical and deterministic methods. Metaheuristic algorithms have recently gained popularity in the PV energy sector for estimating a variety of parameters. However, applying optimization methods to this type of problem necessitates specifying the parameters to be extracted and the objective function to be decreased.

Figure 3.6 depicts the overall process of utilizing metaheuristic methods to improve the performance of a particular function (system to be optimized) [5].

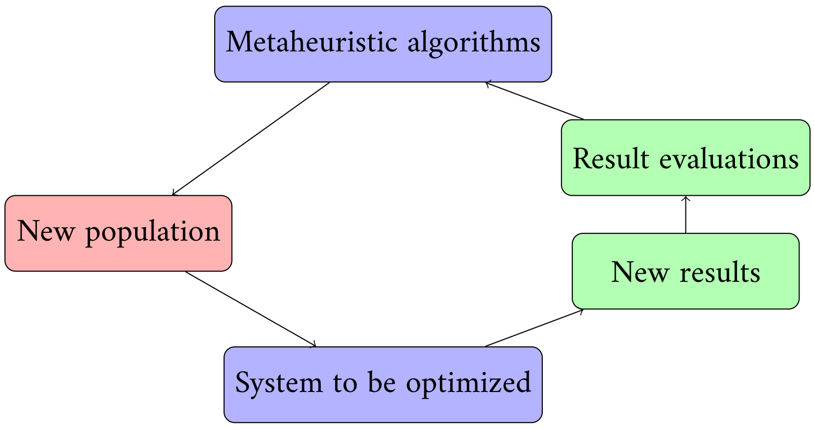


Figure 3.6: Interaction process between metaheuristic algorithms and system to be optimized [5].

This section discusses the algorithms and mathematical models for the five proposed optimization methods.

3.3.1 Gold Rush Optimizer (GRO) Algorithm

The Gold Rush Optimizer (GRO), created in 2023, is a new population-based metaheuristic algorithm inspired by the gold prospecting methods used during the Gold Rush Era. It incorporates concepts like migration, collaboration, and panning to efficiently search for optimal solutions to complex problems.

GRO's inspiration

Gold, symbolized as (Au), with an atomic number of 79, is a noble metal found in the 11th group of the periodic table, known for its shiny, yellow appearance and resistance to tarnishing. It has been highly valued throughout history for its durability and various applications, including in jewellery, coins, electronics, and dentistry.

A significant historical event related to gold is the gold rush, where people flocked to newly discovered gold deposits in the hope of striking it rich, leading to major events in countries like the United States, Australia, Canada, and South Africa during the 19th century [6].

Mathematical model and algorithm

This section introduces mathematical models for gold prospectors, migration, mining, and collaboration, followed by an explanation of the GRO metaheuristic algorithm.

- **Gold prospectors modeling**
  The (GRO) algorithm simulates the key aspects of the gold rush by using gold prospectors as the main entities in the algorithm. In this meta-heuristic algorithm, the location of each prospector is stored in a matrix called M_{GP} P that is expressed as Equation 3.8, where x_{ij} represents the position of prospector i in the jth dimension, d is the dimension size, and n is the number of gold prospectors [6].

$$M_{GP} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \tag{3.8}$$

  The algorithm aims to optimize the performance of gold prospectors by evaluating their positions using an objective function stored in an evaluation matrix M_F where x_{ij} represents the location of prospector i in the jth dimension, and f denotes the evaluation function used for this assessment. This process helps in analyzing and optimizing the performance of gold prospectors in the context of the (GRO) meta-heuristic algorithm [6].

$$M_F = \begin{bmatrix} f\,(x_{11} & x_{12} & \dots & x_{1d} \\ f\,(x_{21} & x_{22} & \dots & x_{2d}) \\ \vdots & \vdots & \ddots & \vdots \\ f\,(x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \tag{3.9}$$

- **Migration of prospectors**
  After finding a gold mine, prospectors move towards it to extract gold. The algorithm uses the position of the most successful prospector as an approximation for the location of the most

lucrative gold mine since the exact location of the richest mine is typically unknown. This migration process helps optimize the search for the best resources within the algorithm's search space Figure 3.7 and Figure 3.8 [6].

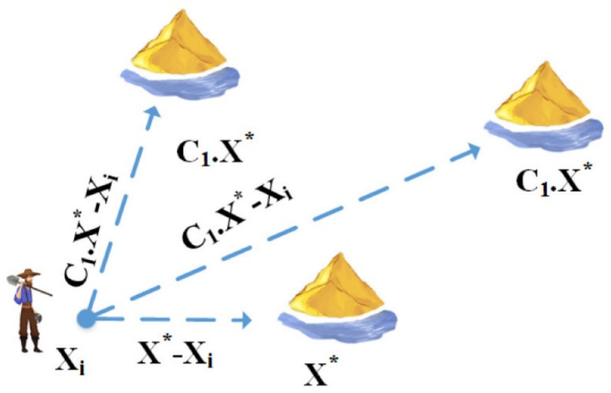


Figure 3.7: Schematic view of equation (3) in two dimensions [6].

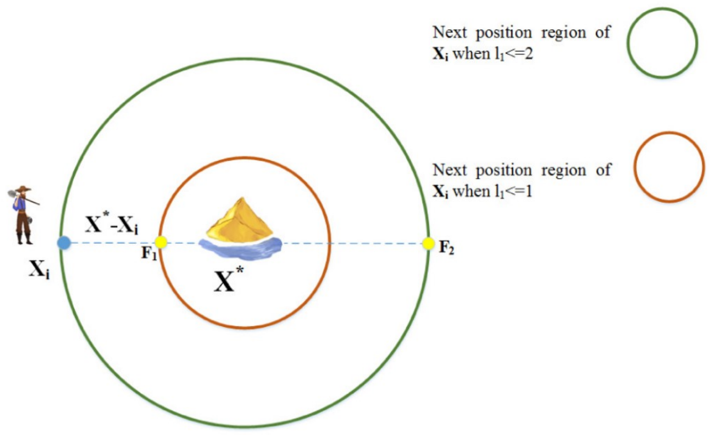

Next position region of X_i when l_1<=2
Next position region of X_i when l_1<=1
X^*-X_i
X_i
F_1
X^*
F_2

Figure 3.8: Schematic view of the operator of migration toward gold mine in two dimensions [6].

Equation 3.10 and Equation 3.11 are used to represent a gold prospector's migration to the mine [6].

$$\vec{D}_1 = \vec{C}_1 \cdot \vec{X}^*(t) - (t) \tag{3.10}$$

$$\overrightarrow{X \text{ new}}_i(t+1) = \vec{X}_i(t) + \vec{A}_1 \cdot \vec{D}_1 \tag{3.11}$$

The variables $\vec{X}^*$, $\vec{X}_i$, and t represent the location of the best gold mine, the location of a specific gold prospector i and the current iteration t, respectively.$\overrightarrow{X\text{new}}$ is the new location of the gold prospector$i$, and $\vec{A}_1$ and $\vec{C}_1$ are vector coefficients calculated by [6]:

$$\vec{A}_1 = 1 + l_1 \left(\vec{r}_1 - \frac{1}{2} \right) \tag{3.12}$$

$$\vec{C}_1 = 2\vec{r}_2 \tag{3.13}$$

where $\vec{r}_1$ and $\vec{r}_2$ are random vectors with value in the range $[0,1]$.The convergence component l_1 decreases linearly from 2 to $\frac{1}{\text{max}_{\text{iter}}}$ if e equals one, and for values greater than 1, it decreases non-linearly. Figures 3 and 4 illustrate the outcomes for power values 1 and 2, and the variations in A_1 over multiple iterations, respectively [6].

$$l_e = \left(\frac{\text{max}_{\text{iter}} - \text{iter}}{\text{max}_{\text{iter}} - 1}\right)^e \left(2 - \frac{1}{\text{max}_{\text{iter}}}\right) + \frac{1}{\text{max}_{\text{iter}}} \tag{3.14}$$

Gold-seekers migrate to nearby locations after reaching an assumed gold mine. The vector A_1 in Equation 3.12 is used to mathematically model this migration process, where a value of 1 indicates movement directly to the assumed mine, while other values represent migration to locations between the seeker's current position and the mine or beyond. Figure 4 shows that its value ranges from about 1. Seekers may explore areas close to known gold mines or consider similar regions to increase their chances of finding gold. Vector C_1 in Equation 3.10 is utilized to mathematically represent the migration behaviour of gold seekers. A value of 1 in Vector C_1 signifies that the seeker migrates directly to the assumed gold mine location, while other values indicate migration to locations with coordinates similar to the mine. Figure 3.7 provides a visual representation of an equation in two dimensions that can be extended to any number of dimensions (d). Figure 3.8 likely visually represents this migration process in a simplified two-dimensional illustration. When (A_1,C_1) is $(0.5, 1)$, the migration occurs at place F_1, and when it is (2, 1), migration happens at place F_2. This process can be extended to higher dimensions, where the migration region moves closer to the gold mine location based on the value of l_1 [6].

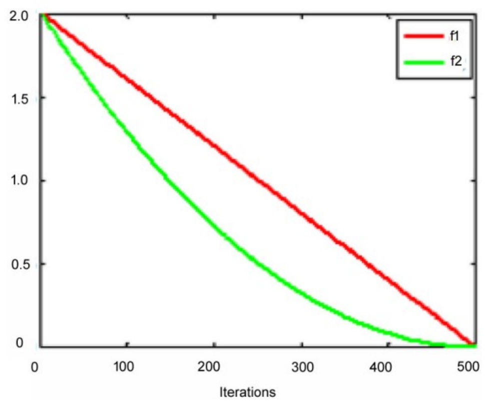


Figure 3.9: Graph for values of l_1 and l_2 during algorithm iteration [6] .

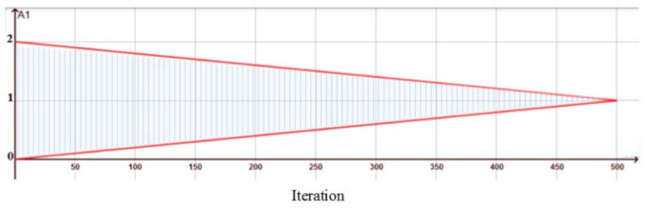


Figure 3.10: Range of changes in A_1 during algorithm iterations [6] .

- **Gold mining (gold panning)**

Each gold prospector mines gold regions in search of further gold. In mathematical modelling, each gold prospector represents the estimated location of a gold mine. The mathematical relationships of gold mining are as follows [6]:

$$\vec{D}_2 = \vec{X}_i(t) - \vec{X}_r(t) \tag{3.15}$$

$$\overrightarrow{X\ \text{new}}_i(t+1) = \vec{X}_r(t) + \vec{A}_2 \cdot \vec{D}_2 \tag{3.16}$$

where $\vec{X}_r$, $\vec{X}_i$, t, and $\overrightarrow{X\text{new}}$ denote the locations of a randomly picked gold prospector r, a gold prospector i, the current iteration t, and the new position of the gold prospector i, respectively. A_2 is the vector coefficient computed using Equation 3.17 [6].

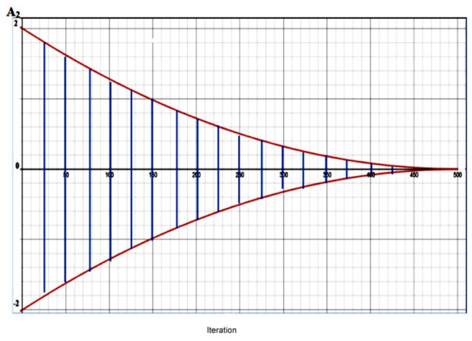


Figure 3.11: Permissible range of changes of A_2 during algorithm iteration [6] .

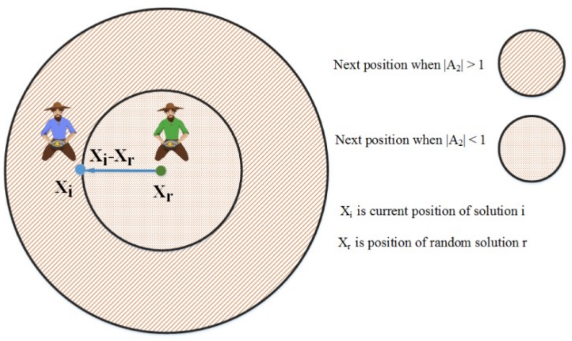

Xi-Xr
Xi
Xr
Next position when |A2| > 1
Next position when |A2| < 1
Xi is current position of solution i
Xr is position of random solution r

Figure 6. Schematic view of gold mining (panning) in 2D

Figure 3.12: Schematic view of gold mining (panning) in 2D [6].

In this equation, using parameter l_2 instead of l_1 improves the mining method's exploitation capabilities. Figure 3.11 shows the range of variations in the latter parameter [6].

$$\vec{A}_2 = 2l_2\vec{r_1} - l_2 \tag{3.17}$$

Figure 3.12 provides a schematic 2D picture of gold mining to help comprehend Equation 3.10 and Equation 3.16. This figure indicates whether a gold prospector should approach or avoid a specific mine based on its A_2 value. This approach can be used to a d-dimensional search space [6].

- **Collaboration between prospectors**

The Equation 3.15 and Equation 3.19 model the collaboration between gold prospectors working together in a team. In this scenario, two randomly selected prospectors, g_1 and g_2, collaborate with another prospector, i, forming a three-person team. The collaboration vector, represented as D_3, illustrates how these prospectors work together to search for gold in a coordinated manner, as shown in a schematic view in Figure 3.13 [6].

$$\vec{D}_3 = \vec{X}_{g_2}(t) - \vec{X}_{g_1}(t) \tag{3.18}$$

$$\overrightarrow{X\ \text{new}}_i(t+1) = \vec{X}_i(t) + \vec{r}_1 \cdot \vec{D}_3 \tag{3.19}$$

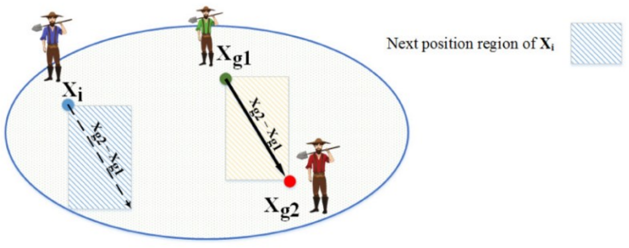


Figure 3.13: Schematic view of the collaboration between prospectors in two dimensions [6].

Seekers g_1 and g_2 specify the gold-seeking zone, but seeker i determines the actual site at random (see Figure 3.13). Thus, three-person cooperation is developed in this manner [6].

- **Prospectors relocation**

The decision-making process for a prospector involves constantly evaluating whether to stay in their current location or move to a new one based on maximizing the amount of gold they can obtain. This decision is guided by comparing the outcomes of an evaluation function, where the prospector updates their location if there is an improvement in the objective function value, otherwise, they remain in their current location. Equation 3.20 in the model represents this decision-making process in scenarios where the prospector aims to minimize certain criteria [6].

$$\vec{X}_i(t+1) = \overrightarrow{X\ \text{new}}_{ii}(t+1) \text{ if } f\left(\overrightarrow{X\ \text{new}}_{ii}(t+1)\right) < f\left(\vec{X}_i(t)\right) \tag{3.20}$$

- **Domain control**

If the location of Xnew $_i$, d at dimension d falls within the lower and upper bounds of dimension d, a new location is evaluated; otherwise, the old location of X_i, d is preserved.

Based on the preceding concepts, the Gold Rush Optimizer (GRO) algorithm, which starts with prospectors randomly placed in a search space to find the best solution (global optimum). Each prospector moves to a new location in each iteration using migration, gold mining, or collaboration methods based on the value of an objective function. If the new location offers more value than the current one, the prospector moves there until the iteration loop ends, with the best solution found considered as the final solution of the algorithm [6].

- **Exploitation and exploration of (GRO)**

  In the gold mining method of the Gold Rush Optimizer algorithm, prospectors search for gold mines to increase their findings. When $|A_2| < 1$, the distance between a prospector and the chosen mine decreases, leading to increased exploitation by the algorithm.
  In the Gold Rush Optimizer algorithm, teamwork among gold prospectors enhances the search for gold mines by combining individual efforts. Initially, teamwork focuses on exploration and searching, leading to increased gold findings through collective agreement. As prospectors move closer to the best gold mine, their collaboration method allows for high exploration ability at the start and increased exploitation ability as they converge towards optimal solutions [6].

  The pseudocode of the Gold Rush Optimizer (GRO) algorithm is presented in algorithm 2, outlining the steps for solving optimization problems theoretically [6].

  * In the Gold Rush Optimizer (GRO) algorithm, the best solution found by each search agent is saved or improved during the search process, making it part of the population.
  * Parameters l_1 and l_2 control the balance between exploring new areas and exploiting known solutions, with l_2 emphasizing exploitation for more focused gold mining efforts.
  * The (GRO) algorithm's parameters are versatile and do not need to be adjusted for different optimization problems.
  * Using the best solution's location in the migration process aids in discovering better solutions.
  * Parameter A_1 restricts the search area around the best solution, leading to improved exploitation of the search space.
  * On the other hand, parameter A_2 influences the algorithm to switch between exploitation $|A_2| < 1$ and exploration $|A_2| > 1$ phases
  * Utilizing the position of a random search agent within a mining operator enhances the algorithm's ability to avoid getting stuck in local optimal solutions.
  * Incorporating the parameter C_1 assists in ensuring that new solutions generated by migration operators are placed within a hypersphere with varying radii.
  * In the collaboration method, the distance between two solutions influences the search radius for prospecting. During early iterations with long distances, a larger search radius is used for exploring the global optimum. As the distance decreases, the search radius reduces, transitioning to a local search for exploitation.

Algorithm 2 Pseudo-code of the Proposed (GRO) Algorithm [6]

```
1: Initialize the gold prospectors' population $X_i,\ i = 1, 2, \ldots, N$
2: Initialize the gold prospectors' new positions $X_{new_i} = X_i,\ i = 1, 2, \ldots, N$
3: Initialize $t, l_1, l_2$
4: $X^*$ is the best search agent
5: while $t \leq$ maximum number of iterations do
6:     for each search agent $i$ do
7:         Calculate the fitness of the current search agent at new position $X_{new_i}$
8:         Update position of current search agent $X_i$ according to Equation 3.20
9:         Update best search agent $X^*$
10:    end for
11:    Update $l_1, l_2$ by Equation 3.14
12:    for each search agent $i$ do
13:        calculate the next position of current search agent $X_{new_i}$ with one of the migration, mining or collaboration methods
14:    end for
15:    $t \leftarrow t + 1$
16: end while
17: return $X^*$
```

3.3.2 Zebra Optimization Algorithm (ZOA)

The Zebra Optimization Algorithm (ZOA), created in 2022, is a bio-inspired metaheuristic algorithm based on the foraging behaviour and defence strategies of zebras. It effectively balances exploration and exploitation to solve optimization problems. This subsection introduces ZOA and presents its mathematical formulation and the pseudo-code.

ZOA's inspiration

Zebras are large equine animals native to eastern and southern Africa, known for their distinctive black-and-white striped coat that helps them camouflage and repel biting flies. They have a body length ranging from 210 to 300 cm, a tail length of 38 to 75 cm, shoulder height between 110 and 160 cm, and weigh between 175 and 450 kg as shown in the Figure 3.14.

Figure 3.14: Zebra.

Zebras are large animals with long, slender legs that enable them to run at high speeds when needed. Similar to other wild equines, zebras have a single toe on each foot, a long neck, and a head structure that facilitates grazing on grass. In their natural social behaviour, zebras exhibit significant behaviours related to foraging for food and employing defensive strategies against predators to ensure their survival in the wild.

In the foraging process, a pioneer zebra initiates the foraging process, allowing others to follow suit. As a result, the pioneer zebra leads the rest of the herd over the plains. The zebras' first strategy against predators is to move in a zigzag way. Sometimes they assemble to confuse or terrify predators.

The mathematical modelling of the intelligent behaviours exhibited by zebras serves as the foundational inspiration for the development of the ZOA [14].

Mathematical model and algorithm

This sub-subsection presents mathematical simulations of zebras' natural actions to model (ZOA).

- **Initialization**

The (ZOA) algorithm is inspired by the behaviour of zebras and is designed as a population-based optimizer. In this algorithm, each zebra represents a candidate solution to a problem, with their position in the search space determining the values for decision variables. The population of zebras is mathematically represented using a matrix, where each zebra is modelled as a vector with elements representing the values of problem variables, and their initial positions are randomly assigned in the search space. The ZOA population matrix is defined in Equation 3.21 [14].

$$X=\begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & . & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & . & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \tag{3.21}$$

Where:

X	: Represent the zebra population.
X_i	: Represent the ith zebra.
$x_{i,j}$	: Represent the value for the jth problem variable proposed by the ith zebra.
N	: Represent the number of population members (zebras).
m	: Represent the number of decision variables.

Each zebra symbolizes a potential solution to the optimization problem. By assessing the proposed values of each zebra for the variables involved, the algorithm can evaluate the objective function to determine the quality of the candidate solutions. The values for the objective function are supplied as a vector in Equation 3.22 [14].

$$F=\begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \tag{3.22}$$

Where:

F	: Represent the vector of objective function values.
X_i	: Represent the objective function value obtained for the ith zebra.

In optimization problems, comparing the objective function values helps evaluate the quality of candidate solutions. For minimization problems, the zebra with the lowest objective function value is considered the best solution, while for maximization problems, the zebra with the highest value is deemed the best solution. Continuously updating the positions of zebras and their objective function values in each iteration allows for the identification of the best candidate solution throughout the optimization process [14].

ZOA members were updated using two natural zebra behaviours seen in the wild. There are two sorts of behaviour:

(i) foraging and

(ii) defence strategies against predators.

During each iteration, the (ZOA) population is updated in two steps [14].

- **Phase 1: Foraging Behaviour**

The first phase involves updating population members based on simulations of zebra foraging behaviour. Zebras primarily feed on grasses and sedges, but they may consume other foods like buds, fruits, bark, roots, and leaves when their preferred options are scarce. Zebras often spend 60-80% of their time eating, depending on the quality and quantity of vegetation. The plains zebra is a pioneer grazer that consumes top, less nutritious grass, creating conditions for shorter, more nutritious grasses below.In ZOA, the best member of the population is referred to as the pioneer zebra, and it brings other population members to its position in the search space. Zebra location updates during foraging can be mathematically described using Equation 3.23 and Equation 3.24 [14].

$$x_{i,j}^{new,P1} = x_{i,j} + r \cdot (PZ_j - I \cdot x_{i,j}) \tag{3.23}$$

$$X_i = \begin{cases} X_i^{\text{new },P1}, & F_i^{\text{new },P1} < F_i \\ X_i, & \text{else }, \end{cases} \tag{3.24}$$

Where:

$X_i^{new\ ,P1}$	: Represent the new status of the ith zebra based on first phase.
$x_{i,j}^{new,P1}$	: Represent the jth dimension value.
$F_i^{new\ ,P1}$	: Represent the objective function value.
PZ	: Represent the pioneer zebra which is the best member.
PZ_i	: Represent the jth dimension of the pioneer zebra.
r	: Represent a random number in the interval $[0, 1]$.
I	: $I = \text{round}(1 + \text{rand})$ where *rand* is a random number in the interval $[0, 1]$.

Thus, $I \in \{1, 2\}$ and if parameter $I = 2$, then there are much more changes in population movement.

- **Phase 2: : Defense Strategies Against**

The second phase involves simulating the zebra's defence strategy against predator attacks to update the position of ZOA population members in the search space. Zebras are mostly preyed upon by lions but also face threats from cheetahs, leopards, wild dogs, brown and spotted hyenas.

Crocodiles are predators of zebras that approach water. Zebras' defence method differs according to the predator. Zebras defend themselves against lion assaults by moving in a zigzag manner and making random sideways turns. Zebras are more aggressive when attacked by smaller predators like hyenas and dogs, who confuse and frighten the hunter by grouping. The ZOA design assumes that one of the following two situations occurs with equal probability [14]:

(i) The lion attacks the zebra, prompting the latter to choose an escape option.

(ii) The zebra will select the attacking tactic when attacked by other predators.

The first strategy involves the zebras attacking the lions and running away from the attack in the area surrounding their current location. Therefore, the mode S1 in Equation 3.25 can be used to mathematically model this method. When a zebra is attacked by another predator, the other zebras in the herd move towards it in an attempt to create a defensive structure that would frighten and confuse the predator. This is the second method. The mode S2 in Equation 3.25 is used to mathematically model this zebra technique. When zebra positions are updated, a zebra's new position is allowed if it has a superior value for the objective function there. Modelling this update condition is done with Equation 3.26 [14].

$$x_{i,j}^{new,P2} = \begin{cases} S_1 : x_{i,j} + R \cdot (2r-1) & \\ \cdot \left(1 - \frac{t}{T}\right) \cdot x_{i,j}, & P_s \le 0.5; \\ S_2 : x_{i,j} + r \cdot (AZ_j - I \cdot x_{i,j}), & \text{else} \end{cases} \tag{3.25}$$

$$X_i = \begin{cases} X_i^{new,P2}, & F_i^{new,P2} < F_i \\ X_i, & \text{else} \end{cases} \tag{3.26}$$

Where:

$X_i^{new,P2}$	: Represent the new status of the ith zebra based on the second phase.
$x_{i,j}^{new,P2}$	: Represent the jth dimension value.
$F_i^{new,P2}$	: Represent the objective function value.
t	: Represent the iteration contour.
T	: Represent the maximum number of iterations.
R	: Represent a constant number equal to 0.01.
P_s	: Represent the probability of choosing one of two strategies that are randomly generated in the interval $[0, 1]$.
AZ	: Represent the the status of attacked zebra.
AZ_i	: Represent the jth dimension of the attacked zebra.

- **Repetitions Process and Pseudo-Code of (ZOA)**

The population members are updated based on the first and second phases at the end of each ZOA iteration. Up to the algorithm's complete implementation, the algorithm population is updated depending on steps Equation 3.23 through Figure 3.12. Through several cycles, the best candidate solution is updated and saved. ZOA presents the top contender as the best answer to the given problem after it has been fully developed. The ZOA phases are shown in Algorithm 3's pseudocode [14].

Algorithm 3 Pseudo-code of the Proposed (ZOA) Algorithm [14]

```
 1: Start ZOA.
 2: Input: The optimization problem information.
 3: Set the number of iterations $(T)$ and the number of zebras' population $(N)$.
 4: Initialization of the position of zebras and evaluation of the objective function.
 5: for $t = 1$ to $T$ do
 6:     Update pioneer zebra (PZ).
 7:     for $i = 1$ to $N$ do
 8:         Phase 1: Foraging behavior
 9:         Calculate new status of the $i$th zebra using Equation 3.23.
10:         Update the $i$th zebra using Equation 3.24.
11:         Phase 2: Defense strategies against predators
12:         if $P_s < 0.5$ then
13:             $P_s = \text{rand}$
14:             Strategy 1: against lion (exploitation phase)
15:             Calculate new status of the $i$th zebra using mode $S_1$ in Equation 3.25.
16:         else
17:             Strategy 2: against other predators (exploration phase)
18:             Calculate new status of the $i$th zebra using mode $S_2$ in Equation 3.25.
19:         end if
20:         Update the $i$th zebra using Equation 3.26.
21:     end for
22:     Save best candidate solution so far.
23: end for
24: Output: The best solution obtained by ZOA for a given optimization problem.
25: End ZOA.
```

3.3.3 Secretary Bird Optimization Algorithm (SBOA)

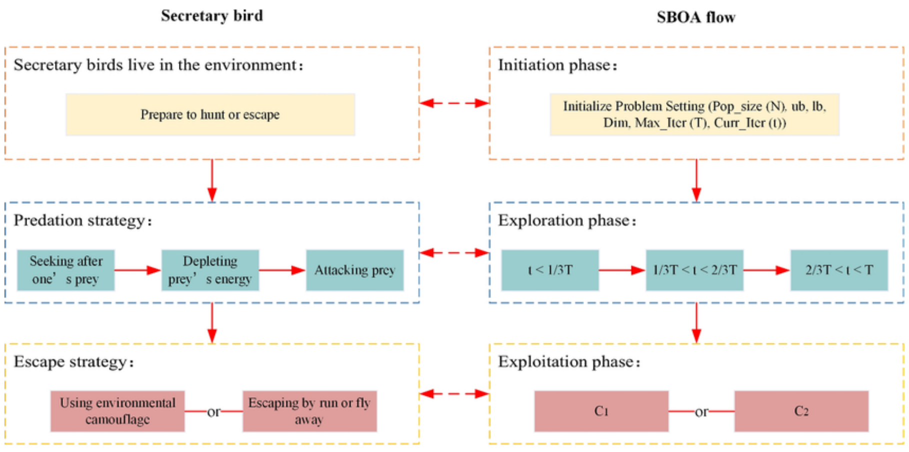
The Secretary Bird Optimization Algorithm (SBOA), created in 2024, is a metaheuristic inspired by the hunting and evasion strategies of secretary birds. This subsection provides a mathematical description of the bird's behaviour and the SBOA's pseudo-code.

SBOA's inspiration

The Secretary Bird, scientifically known as Sagittarius serpentarius, is a visually striking African bird known for its unique behaviours and appearance. It is commonly found in grasslands, savannas, and open riverine areas in Africa. In addition to semi-desert locations and wooded areas with wide clearings, secretary birds are commonly found in tropical open grasslands, savannas with few trees, and open places with long grass, showcasing a distinctive plumage with grey-brown feathers on their back and wings, white chest, and black belly as shown in Figure 3.15 [7].

Figure 3.15: Secretary bird [7].

The Secretary Bird's hunting behaviour involves using its long legs and talons to run and capture prey on the ground, resembling a secretary at work as it scans the grasslands for hidden prey. This bird feeds on insects, reptiles, small mammals, and other prey, swiftly charging towards its target, capturing it with its sharp talons, and then striking it against the ground to consume it [7].


Secretary bird
SBOA flow
Secretary birds live in the environment:
Prepare to hunt or escape
Initiation phase:
Initialize Problem Setting (Pop_size (N), ub, lb, Dim, Max_Iter (T), Curr_Iter (t))
Predation strategy:
Seeking after one' s prey
Depleting prey' s energy
Attacking prey
Exploration phase:
t < 1/3T
1/3T < t < 2/3T
2/3T < t < T
Escape strategy:
Using environmental camouflage
or
Escaping by run or fly away
Exploitation phase:
C1
or
C2
Figure 3.16: Secretary bird hunting and escape strategies correspond to (SBOA) [7].

Secretary Birds exhibit remarkable intelligence in hunting snakes by leveraging their height advantage to closely monitor and predict the snakes' movements. Their ability to anticipate the next move of the snake showcases their exceptional hunting skills and control over the situation, making them formidable adversaries to snakes in their natural habitat [7].

It leaps and provokes the snake while delicately hovering above it. It acts like a agile martial arts expert, while the snake, imprisoned inside its circle, writhes in terror. The snake becomes weaker as a result of the constant taunting. At this time, the Secretary Bird leaps behind the serpent to deal a fatal blow, avoiding its frontal attacks. It delivers a lethal blow to the snake by grasping its critical parts with its sharp talons. The Secretary Bird may find it difficult to handle larger snakes because of their strong constriction and crushing force. In these situations, the Secretary Bird has the ability to pick up the snake off the ground by either grasping it with its talons or carrying it in its beak. It then takes off into the sky and lets go of the snake, letting it hit the hard ground with a foreseeable end [7].

Furthermore, Secretary Bird's intelligence is obvious in its predator avoidance methods, which include two separate ways. The bird's first response is to conceal itself when detecting a nearby threat. The Secretary Bird uses camouflage to avoid potential predators. The bird uses the second approach when it learns its surroundings are unsuitable for camouflage. When faced with a predator, the animal will either fight or walk quickly to escape [7].

Figure 3.16 shows how the Secretary Bird's behaviour corresponds to the Secretary Bird Optimization Algorithm (SBOA). The secretary bird's prepared hunting behaviour correlates to the initialization step of the Secretary Bird Optimization Algorithm (SBOA). The secretary bird's hunting process follows the three exploratory stages of SBOA. The secretary bird uses C1 and C2 methods to avoid predators, which correlate to SBOA's exploitation stages [7].

Mathematical model and algorithm

This sub-subsection proposes a mathematical model of secretary bird behaviour to hunt snakes and avoid natural enemies, which will be presented to (SBOA).

- **Initialization**

  The Secretary Bird Optimization Algorithm (SBOA) is a population-based method inspired by the hunting behaviour of Secretary Birds. In this algorithm, each Secretary Bird represents a potential solution to a problem, with their positions in the search space determining the values of decision variables. The algorithm starts with random initialization of the Secretary Birds' positions using a mathematical Equation 3.27 to explore possible solutions efficiently [7].

$$X_{i,j} = lb_j + r \times (ub_j - lb_j), i = 1, 2, \ldots, N, j = 1, 2, \ldots, \text{ Dim} \tag{3.27}$$

Where:

X_i	: Represent the position of the i th secretary bird.
lb_j,ub_j	: Represent the lower and upper bounds, respectively.
r	: Represent a random number between 0 and 1.

The Secretary Bird Optimization Algorithm (SBOA) is a population-based method where optimization begins with a group of potential solutions as shown in Equation 3.28. These solutions X are randomly generated within specified upper and lower bounds, and the best solution found in each iteration is considered the optimal solution [7].

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,\text{Dim}} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,\text{Dim}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,\text{Dim}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,j} & \cdots & x_{N,\text{Dim}} \end{bmatrix}_{N \times \text{Dim}} \tag{3.28}$$

Where:

X	: Represent the secretary bird group.
X_i	: Represent the ith secretary bird.
$X_{i,j}$	: Represent the ith secretary bird jth question the value of a variable.
N	: Represent group members (the secretary) number.
Dim	: Represent problem of variable dimension.

Each secretary bird symbolizes a potential approach for optimizing the situation. The objective function can be assessed using the values suggested by each secretary bird for the problem variables. The objective function values are compiled into a vector by Equation 3.29 [7].

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_{\mathrm{i}} \\ \vdots \\ F_{\mathrm{N}} \end{bmatrix}_{N\times 1} = \begin{bmatrix} F\left(X_1\right) \\ \vdots \\ F\left(X_i\right) \\ \vdots \\ F\left(X_N\right) \end{bmatrix}_{N\times 1} \tag{3.29}$$

Where:

F	: Represent the vector of objective function values.
F_i	: Represent the objective function value obtained by the ith secretary bird.

Comparing objective function values effectively analyzes possible solutions to determine the best answer for a given problem. In minimization problems, the secretary bird with the lowest objective function value is the best candidate solution. In maximizing problems, the secretary bird with the greatest objective function value is the best candidate. Each iteration requires determining the best candidate solution, as the positions of the secretary birds and values of the objective function change [7].

The secretary bird's two natural actions were used to provide updates to SBOA members. These two categories of behaviours include:

(i) The secretary bird's hunting strategy;

(ii) The secretary bird's escape strategy.

Each iteration involves updating each member of the secretary bird colony in two steps [7].

- **Hunting strategy of secretary bird (exploration phase)**

Secretary birds hunt snakes in three stages: searching for, consuming, and attacking them. Figure 3.17 shows the hunting behaviour of the secretary bird.

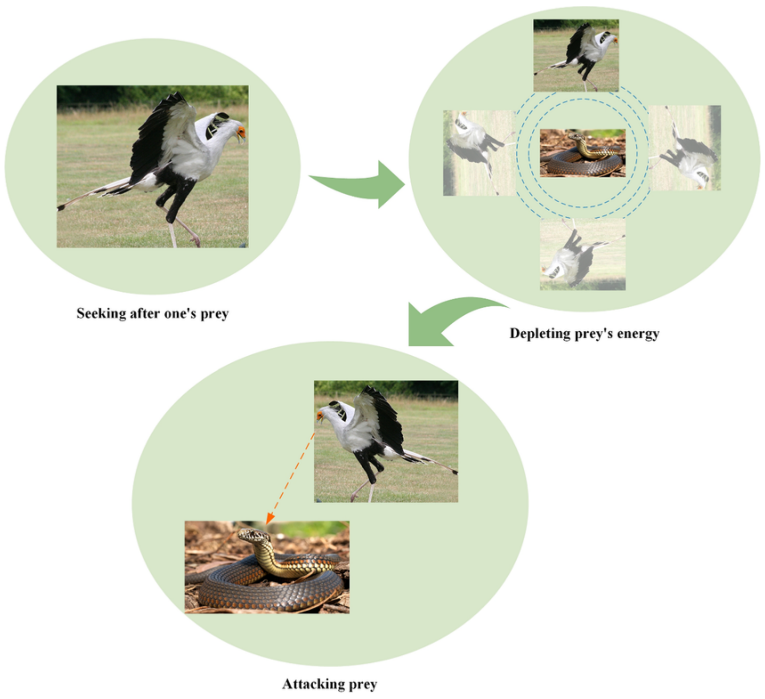


Figure 3.17: The hunting behaviour of secretary birds [7].

We divided the secretary bird's hunting process into three equal time intervals based on biological statistics and phase durations: $< \frac{1}{3}T, \frac{1}{3}T < t < \frac{2}{3}T$ and $\frac{2}{3}T < t < T$. These intervals correspond to the three phases of predation: searching for, consuming, and attacking prey. Thus, the modelling of each phase in SBOA is as follows:

Stage 1 (Searching for Prey): Secretary birds seek for snakes as their primary prey. Secretary birds have exceptional vision, enabling them to detect snakes in tall grass on the savannah. They sweep the ground carefully with their lengthy legs, keeping an eye out for snakes. Their large legs and necks allow them to maintain a safe distance from snake strikes. This occurs during the earliest iterations of optimization, where exploration is vital. This stage uses a differential evolution method. Differential evolution leverages individual differences to provide new solutions, increasing algorithm variety and global search capabilities. Diversity can help to prevent becoming trapped in local optima by introducing differentiable mutation processes. Exploring different parts of the solution space increases the probability of finding the global optimum. To represent the secretary bird's position during the Searching for Prey stage, use Equation 3.30 and Equation 3.31 [7].

$$\text{While } t < \frac{1}{3}T, x_{i,j}^{\text{new } P1} = x_{i,j} + (x_{\text{random } 1} - x_{\text{random } \_2}) \times R_1 \tag{3.30}$$

$$X_i = \begin{cases} X_i^{\text{new },P1}, & \text{if } F_i^{\text{new },P1} < F_i \\ X_i, & \text{else} \end{cases} \tag{3.31}$$

Where:

t	: Represent the current iteration number.
T	: Represent the maximum iteration number.
$X_i^{new\ ,P1}$	: Represent the new state of the ith secretary bird in the first stage.
X_{random_1}	: Represent the random candidate solutions in the first stage iteration.
X_{random_2}	: Represent the random candidate solutions in the first stage iteration.

R_1	: A randomly generated array of dimension$1 \times$ Dim from the interval $[0, 1]$.
Dim	: Represent the dimensionality of the solution space.
$x_i^{new\ ,P1}$	: Represent the value of the jth dimension of the ith secretary bird.
$F_i^{new\ ,P1}$	: The fitness value of the objective function for the ith secretary bird.

Stage 2 (Consuming Prey): When a secretary bird discovers a snake, it uses a unique hunting technique. Unlike other raptors, the secretary bird uses its agility to move around the snake. The secretary bird stands solid, watching every move of the snake from a high vantage point. The predator gradually wears down its opponent's stamina by hovering, jumping, and provoking it based on its behaviour. At this level, we use Brownian motion (RB) to imitate the random movement of the secretary bird. Brownian motion can be formally represented using Equation 3.32. The secretary bird's "peripheral combat" tactic offers them a significant physical advantage. The bird's long legs prevent snakes from entangling it, and its talons and leg surfaces are covered in thick keratin scales that act as protection against venomous snakes' fangs. During this step, the secretary bird may pause to focus on the snake's location using its keen vision. We employ "$xbest$" (individual historical best position) and Brownian motion. Using "$xbest$" allows for local searches based on previously discovered best placements, leading to a greater exploration of the solution space. This method not only prevents premature convergence to local optima but also accelerates the algorithm's convergence to optimal positions in the solution space. Individuals can look for the global optimum by combining global knowledge with their own previous best locations. Brownian motion's randomness allows for more effective solution exploration and avoidance of local optima, resulting in improved results for complicated problems. To represent the secretary bird's posture during the Consuming Prey stage, use equations Equation 3.33 and Equation 3.34 [7].

$$RB = \text{randn}(1,\ \text{Dim}\ ) \tag{3.32}$$

$$\text{While } \frac{1}{3}T < t < \frac{2}{3}T, \quad x_{i,j}^{\text{new }P1} = x_{\text{best}} + \exp((t/T) \wedge 4) \times (RB - 0.5) \times (x_{\text{best}} - x_{i,j}) \tag{3.33}$$

$$X_i = \begin{cases} X_i^{\text{new },P1}, & \text{if } F_i^{\text{new },P1} < F_i \\ X_i, & \text{else} \end{cases} \tag{3.34}$$

Where:

$randn(1, Dim)$: Represent a randomly generated array of dimension $1 \times$ Dim from a standard normal distribution (mean 0, standard deviation 1).
x_{best} : Represents the current best value.

Stage 3 (Attacking Prey): When the snake is tired, the secretary bird uses its strong leg muscles to attack. During this stage, the secretary bird uses its keen talons to kick the snake's head with speed and accuracy. The kicks aim to swiftly incapacitate or kill the snake, preventing further bites. The keen talons kill the snake by striking its vital areas. When a snake is too large to kill immediately, the secretary bird may release it into the sky, allowing it to fall to the ground and die. We use the Levy battle method in random search to improve global search capabilities, reduce SBOA stuckness in local solutions, and improve algorithm convergence accuracy. We introduce a nonlinear perturbation factor $\left(1 - \frac{t}{T}\right)^{(} 2 \times \frac{t}{T} \Big)$ to improve SBOA's dynamic, adaptive, and flexible optimization process. This factor balances exploration and exploitation, avoids premature convergence, accelerates convergence, and improves algorithm performance. Therefore, updating the secretary bird's position in the Attacking Prey stage can be analytically described using Equation 3.35 and Equation 3.36 [7].

$$\text{While } t > \frac{2}{3}T, x_{i,j}^{\text{new } P1} = x_{\text{best}} + \left(\left(1 - \frac{t}{T}\right) \wedge \left(2 \times \frac{t}{T}\right)\right) \times x_{i,j} \times RL \tag{3.35}$$

$$X_i = \begin{cases} X_i^{\text{new },P1}, & \text{if } F_i^{\text{new },P1} < F_i \\ X_i, & \text{else} \end{cases} \tag{3.36}$$

To improve the algorithm's accuracy, we apply the weighted Levy fight (RL).

$$\text{RL} = 0.5 \times \text{Levy}(\text{Dim}) \tag{3.37}$$

Here, Levy(Dim) denotes the Levy battle distribution function. The calculation goes as follows:

$$\text{Lev}(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{n}}} \tag{3.38}$$

The fixed constants s and η are 0.01 and 1.5, respectively. u and v are random numbers in the range $[0, 1]$. The formula for σ is given below:

$$\sigma = \left(\frac{\Gamma(1+\eta) \times \sin\left(\frac{\pi\eta}{2}\right)}{\Gamma\left(\frac{1+\eta}{2}\right) \times \eta \times 2\left(\frac{\eta-1}{2}\right)}\right)^{\frac{1}{\eta}} \tag{3.39}$$

Here, Γ represents the gamma function, and η has a value of 1.5.

- **Escape strategy of secretary bird (exploitation stage)**

Predators that may attack and steal food from secretary birds include eagles, hawks, foxes, and jackals. Secretive birds employ a variety of evasion techniques to shield their food and themselves from predators. These strategies can be roughly divided into two categories. The initial tactic is either sprinting or fighting fast. Secretary birds can sprint quickly because of their long legs. Because they may cover 20–30 kilometres in a day, they are referred to as "marching eagles". Secretary birds are adept combatants who can quickly escape peril and locate safe havens. The second tactic is disguising oneself. Secretary birds can blend in with their surroundings by using colours or structural features, which makes it more difficult for predators to find them. Figure 3.18 shows their evasive tendencies in response to perceived threats. One of the following two scenarios is thought to occur with equal likelihood in the SBOA's design [7]:

(i) C_1: Camouflage by environment;

(ii) C_2: Fly or run away.

In the first method, secretary birds first look for an appropriate camouflage environment when they sense the presence of a predator. They will choose to fight or run quickly away if there isn't a good and safe hiding place close by. We present a dynamic perturbation factor in this setting, represented as $\left(1 - \frac{t}{T}\right)^2$. The algorithm finds a balance between exploitation (using existing solutions) and exploration (looking for new ones) with the aid of this dynamic perturbation factor. It is feasible to improve exploitation at various phases or raise the degree of investigation by modifying these variables. In summary, Equation 3.40 can be used to mathematically model both of the evasion tactics used by secretary birds and Equation 3.41 may be used to define this updated condition [7].

$$x_{i,j}^{\text{new },P2} = \begin{cases} C_1 : x_{\text{best}} + (2 \times RB - 1) \times \left(1 - \frac{t}{T}\right)^2 \times x_{i,j}, & \text{if } r \text{ and } < r_i \\ C_2 : x_{i,j} + R_2 \times (x_{\text{random}} - K \times x_{i,j}), & \text{else} \end{cases} \tag{3.40}$$

$$X_i = \begin{cases} X_i^{\text{new },P2}, & \text{if } F_i^{\text{new },P2} < F_i \\ X_i, & \text{else} \end{cases} \tag{3.41}$$

Where:

$r = 0.5$
R_2 : Represents the random generation of an array of dimension ($1 \times$ Dim) from the normal distribution.
X_{random} : Represent the random candidate solutions of the current iteration.
K : Represents the random selection of integer 1 or 2, which can be calculated by Equation 3.42.

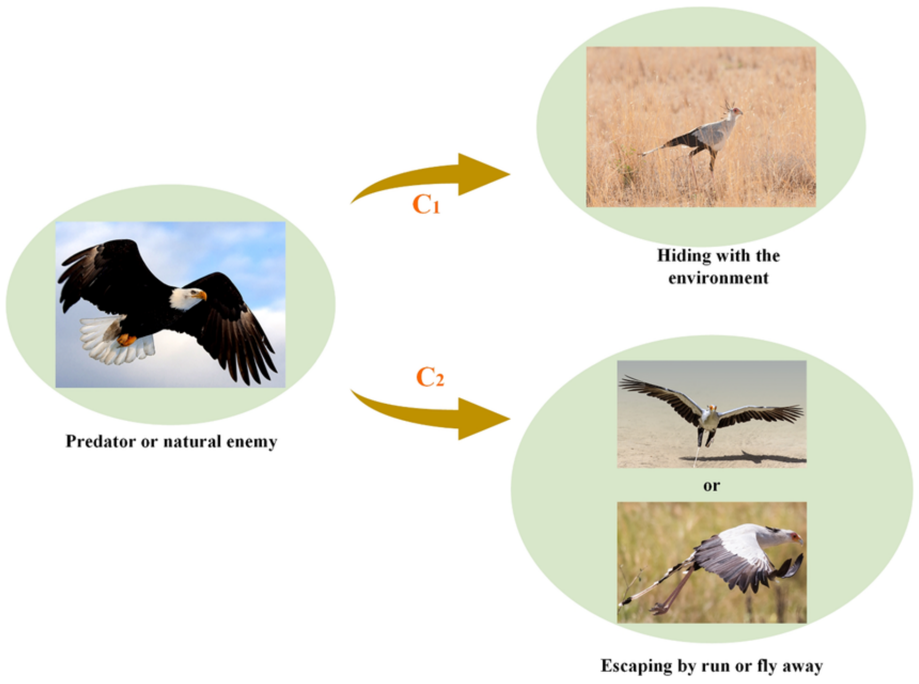

C1
Hiding with the environment
Predator or natural enemy
C2
or
Escaping by run or fly away

Figure 3.18: The escape behaviour of the secretary bird [7].

$$K = \text{round}(1 + \text{rand}(1, 1)) \tag{3.42}$$

`rand(1, 1)` in this context refers to selecting a random integer between $(0, 1)$. In summary, the pseudo-code appears in Algorithm 4 [7].

Algorithm 4 Pseudo-code of the Proposed (SBOA) Algorithm [7]

```
1: Initialize Problem Setting $(Dim, ub, lb, Pop\_size(N)), Max\_Iter(T), Curr\_Iter(t)$
2: Initialize the population randomly
3: for $t = 1$ to $T$ do
4:     Update Secretary Bird $x_{best}$.
5:     for $i = 1$ to $N$ do
6:         Exploration:
7:         if $t < \frac{1}{3}T$ then
8:             Calculate new status of the $i^{th}$ Secretary Bird using Equation 3.30
9:             Update the $i^{th}$ Secretary Bird using Equation 3.31
10:        else if $\frac{1}{3}T < t < \frac{2}{3}T$ then
11:            Calculate new status of the $i^{th}$ Secretary Bird using Equation 3.33
12:            Update the $i^{th}$ Secretary Bird using Equation 3.34
13:        else
14:            Calculate new status of the $i^{th}$ Secretary Bird using Equation 3.35
15:            Update the $i^{th}$ Secretary Bird using Equation 3.36
16:        end if
17:        Exploitation:
18:        if $r < 0.5$ then
19:            Calculate new status of the $i^{th}$ Secretary Bird using $C_1$ in Equation 3.40
20:        else
21:            Calculate new status of the $i^{th}$ Secretary Bird using $C_2$ in Equation 3.40
22:        end if
23:        Update the $i^{th}$ Secretary Bird using Equation 3.41
24:    end for
25:    Save best candidate solution so far.
26: end for
27: Output: The best solution obtained by SBOA for a given optimization problem.
28: Return best solution.
```

3.3.4 Hippopotamus Optimization Algorithm (HOA)

This section explains the inspiration and theoretical basis of the proposed HO Algorithm, which was created in 2024. The HO algorithm is a novel nature-inspired optimization technique that emulates the behaviours of hippopotamuses. It incorporates a trinary-phase model that reflects how hippos navigate rivers, defend against predators, and evade threats. The algorithm has demonstrated promising results in addressing various optimization problems, outperforming several well-known optimization algorithms in both efficiency and effectiveness [8].

Hippopotamus

The hippopotamus is one of the most interesting animals found in Africa. This animal is classified as a vertebrate, specifically a mammal. Hippopotamuses are semi-aquatic animals that live mostly in aquatic habitats, such as rivers and ponds. Hippopotamuses live in pods or bloats, consisting of 10 to 30 individuals. Identifying the gender of hippopotamuses is challenging due to their lack of obvious sexual organs and reliance on weight differences. Adult hippopotamuses can stay underwater for up to 5 minutes. This animal looks similar to harmful mammals like shrews, yet its nearest relatives are whales and dolphins, which shared a common ancestor approximately 55 million years ago.

Hippopotamuses, despite their herbivorous diet of grass, branches, leaves, reeds, flowers, leaves, and plant husks, are curious and actively seek out different food sources. Biologists believe that eating meat can create digestive problems in hippos. These animals have powerful jaws, an aggressive attitude, and territorial behaviour, making them one of the most harmful mammals in the world. Male hippopotamuses can weigh up to 9,920 pounds, while females average around 3,000 pounds. They consume around 75 pounds of food daily. Hippopotamuses frequently communicate with one another, and during these interactions, calves may receive injuries or even die. Adult hippopotamuses are typically not targeted by predators due to their size and power. Young hippopotamuses and weak adults are vulnerable to predators like Nile crocodiles, lions, and spotted hyenas.

Hippopotamuses defend themselves against predators by rotating and opening their formidable jaws. This is accompanied by a loud vocalization (about 115 decibels) that intimidates predators and stops them from chasing the dangerous prey. When a hippopotamus' defensive approach fails or it lacks sufficient strength, it retreats quickly at 30 km/h to avoid the threat. Typically, it travels to surrounding water bodies like ponds or rivers [8].

HOA's inspiration

The HO is inspired by three significant behavioural characteristics seen in the lives of hippos. Hippopotamus herds typically consist of females, calves, mature males, and a dominant male leader. Young and calves hippopotamuses are naturally curious and may wander away from their herd. As a result, they may become solitary and vulnerable to predators.

Hippopotamuses' secondary behaviour is protective, responding to predator attacks or outsiders in their territory. Hippopotamuses defend themselves by spinning toward predators and using powerful jaws and vocalizations to discourage and repel them Figure 3.19. Lions and spotted hyenas are aware of hippopotamus jaws and avoid close contact to prevent damage. The hippopotamus' last behavioural style involves instinctively fleeing predators and aggressively avoiding places of danger. The hippopotamus seeks out the nearest body of water, such as a river or pond, as lions and spotted hyenas often avoid watery habitats [8].

Mathematical model and algorithm

The HO algorithm is population-based and uses hippopotamuses as search agents. The HO algorithm uses hippopotamuses as candidate solutions for optimization problems. Each hippopotamus' position update in the search space represents values for decision variables. Tus represents each hippopotamus as a vector, whereas the population is formally defined by a matrix. The HO, like other optimization algorithms, generates randomized initial solutions during its initialization stage. In this stage, the vector of decision variables is created using the following formula [8]:

$$X_i : x_{ij} = lb_j + r \cdot (ub_j - lb_j), \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, m \tag{3.43}$$

Where:

X_i	: Represent the position of the i th candidate solution.
lb_j,ub_j	: Represent the lower and upper bounds of the jth decision variable, respectively.
r	: Represent a random number between 0 and 1.
N	: Represent the population size of hippopotamuses within the herd.
m	: Represent the number of decision variables in the problem.

And the population matrix is formed by Equation 3.44.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \cdots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \cdots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \tag{3.44}$$

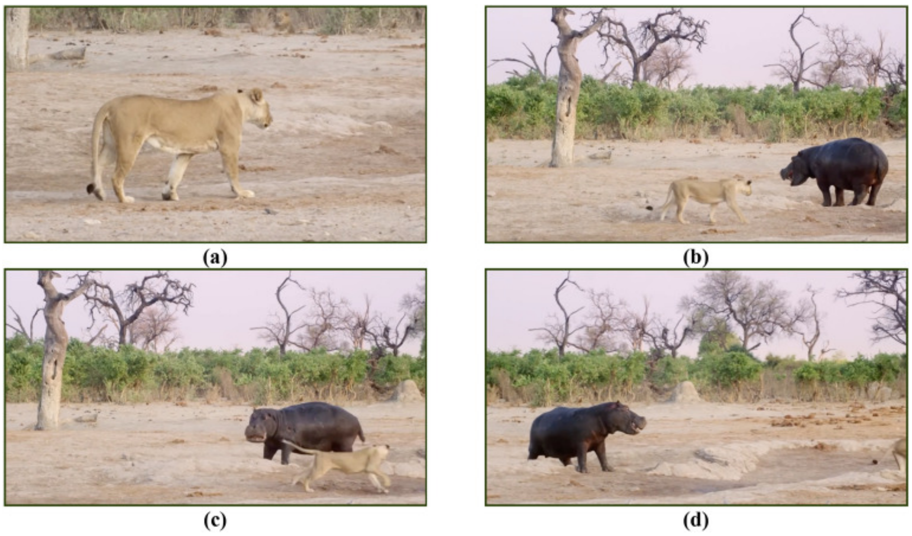


Figure 3.19: (a–d) shows the defensive behaviour of the hippopotamus against the predator [8].

- **Phase 1: The hippopotamuses position update in the river or pond (Exploration)**

Hippopotamus herds consist of adult females, calves, adult males, and the herd leader. The current hippopotamus is determined by iterating the objective function values (lowest for minimization, highest for maximization). Hippopotamuses typically cluster in close proximity to one another. Dominant male hippopotamuses defend their herd and territory from any dangers.

Female hippopotamuses surround the males. Male hippopotamuses are removed from the herd by the dominant male as they reach maturity. To establish dominance, removed males must either attract females or compete with established males in the herd. Equation 3.45 represents the mathematical position of male hippopotamus members of the herd in the lake or pond [8].

$$X_i^{M_{\text{hippo}}} : x_{ij}^{M_{\text{hippo}}} = x_{ij} + y_1 \cdot (D_{\text{hippo}} - I_1 x_{ij}) \tag{3.45}$$

$$\text{for }_i = 1, 2, \ldots, \left[\frac{N}{2}\right] \text{ and } j = 1, 2, \ldots, m$$

Where:

$X_i^{M_{hippo}}$	: Represents male hippopotamus position.
D_{hippo}	: Represent the dominant hippopotamus position (the hippopotamus that has the best cost in the current iteration).
y_1	: Represents a random number between 0 and 1.

$$h = \begin{cases} I_2 \times \vec{r}_1 + (\sim \varrho_1) \\ 2 \times \vec{r}_2 - 1 \\ \vec{r}_3 \\ I_1 \times \vec{r}_4 + (\sim \varrho_2) \\ r_5 \end{cases} \tag{3.46}$$

Where:

$\vec{r}_{1 \ldots 4}$	: Represents a random vector between 0 and 1.
r_5	: Represent a random number between 0 and 1.
I_1,I_2	: Represent an integer between 1 and 2.
$\sim \varrho_1$,$\sim \varrho_2$	: Represents an integer random number that can be one or zero.

$$T = \exp\left(-\frac{t}{T}\right) \tag{3.47}$$

$$X_i^{FB_{\text{hippo}}} : x_{ij}^{FB_{\text{hippo}}} = \begin{cases} x_{ij} + h_1 \cdot (D_{hippo} - I_2 MG_i)T & > 0.6 \\ \text{[I]} & \text{else} \end{cases} \tag{3.48}$$

Where:

MG_i	: Represents the mean values of some randomly selected hippopotamus with an equal probability of including the current considered hippopotamus (X_i) .
h_1	: Represent a number or vector randomly selected from the five scenarios in the h equation.

$$[\mathrm{I}] = \begin{cases} x_{ij} + h_2(MG_i - D_{hippo})r_6 & > 0.5 \\ lb_j + r_7 \cdot (ub_j - lb_j) & \text{else} \end{cases} \tag{3.49}$$

$$\text{for }_i = 1, 2, \ldots, \left[\frac{N}{2}\right] \text{ and } j = 1, 2, \ldots, m$$

Where:

h_2 : Represents a number or vector randomly selected from the five scenarios in the h equation.
r_7 : Represents a random number between zero and one.

Equation 3.48 and Equation 3.49 define the position of female or immature hippos (X_i) in the herd. Immature hippopotamuses often stay close to their mothers, however some may wander away from the herd owing to curiosity. If T is larger than 0.6, the young hippopotamus has moved away from its mother (Equation 3.47). If r_6, a number between 0 and 1 (Equation 3.49), exceeds 0.5, it indicates that the immature hippopotamus has separated from its mother but remains within or near the herd. Otherwise, it has left the herd. Equation 3.48 and Equation 3.49 model the behaviour of both immature and female hippopotamuses.

Equation 3.50 and Equation 3.51 describe male and female or immature hippopotamus position updates within the herd. F_i is an objective function value.

$$X_i = \begin{cases} X_i^{M_{\text{hippo}}} F_i^{M_{\text{hippo}}} & < F_i \\ X_i & \text{else} \end{cases} \tag{3.50}$$

$$X_i = \begin{cases} X_i^{FB_{\text{hippo}}} F_i^{FB_{\text{hippo}}} & < F_i \\ X_i & \text{else} \end{cases} \tag{3.51}$$

Using h vectors, I_1 and I_2 scenarios enhance the global search and improve exploration in the proposed algorithm. It leads to a better global search and enhances the exploration process in the proposed algorithm [8].

- **Phase 2: Hippopotamus defence against predators (Exploration)**

Hippos live in herds for safety and comfort. Large animal herds can dissuade predators from approaching too closely. Immature hippopotamuses may wander from the herd and become targets for predators such as Nile crocodiles, lions, and spotted hyenas due to their curiosity and lower strength compared to adults. Sick hippopotamuses, like immature ones, can be preyed upon by predators. Hippopotamuses deploy a protective strategy of quickly turning towards predators and emitting loud vocalizations to avoid close encounters Figure 3.20. During this phase, hippos may approach predators to induce their abandonment, thus warding off prospective threats. Equation 3.52 depicts the predator's position in search space [8].

$$\text{Predator} : \text{Predator}_j = \text{lb}_j + \vec{r}_8 \cdot (\text{ub}_j - \text{lb}_j), \quad j = 1, 2, \ldots, m. \tag{3.52}$$

where $\vec{r}_8$ represents a random vector ranging from zero to one.

$$\vec{D} = |\text{Predator}_j - x_{ij}| \tag{3.53}$$

Equation 3.53 indicates the distance of the ith hippopotamus to the predator. During this time, the hippopotamus adopts a defensive behaviour based on the factor $F_{Predator_j}$ to protect itself against the predator. If $F_{Predator_j}$ is less than F_i, indicating the predator is in very close proximity to the hippopotamus, in such a case, the hippopotamus swiftly turns towards the predator and moves towards it to make it retreat. If $F_{Predator_j}$ is greater, it indicates that the predator or intruding entity is at a greater distance from the hippopotamus's territory Equation 3.54. In this case, the hippopotamus turns towards the predator but with a more limited range of movement. The intention is to make the predator or intruder aware of its presence within its territory [8].

$$X_i^{HippoR} : x_{ij}^{HippoR} = \begin{cases} \overrightarrow{RL} \oplus Predator_j + \left(\frac{f}{(c-d\times\cos(2\pi g))}\right) \cdot \left(\frac{1}{\vec{D}}\right) F_{Predator_j} & < \mathcal{F}_j \\ \overrightarrow{RL} \oplus Predator_j + \left(\frac{f}{(c-d\times\cos(2\pi g))}\right) \cdot \left(\frac{1}{2\times\vec{D}+\vec{r}_9}\right) F_{Predator_j} & \geq \mathcal{F}_j \end{cases} \tag{3.54}$$

$$\text{for } i = \left\lceil \frac{N}{2} \right\rceil + 1, \left\lceil \frac{N}{2} \right\rceil + 2, \ldots, N \text{ and } j = 1, 2, \ldots, m$$

Where:

X_i^{HippoR}	: Represents a hippopotamus position which was faced to predator.
$\overrightarrow{RL}$	: Represents a random vector with a Levy distribution, utilized for sudden changes in the predator's position during an attack on the hippopotamus.
f	: Represents a uniform random number between 2 and 4.
c	: Represents a uniform random number between 1 and 1.5.
D	: Represents a uniform random number between 2 and 3.
g	: Represents a uniform random number between -1 and 1.
$\vec{r}_8$	: Represents a random vector with dimensions 1 × m.

The mathematical model for the random movement of Lévy movement is calculated as Equation 3.55.

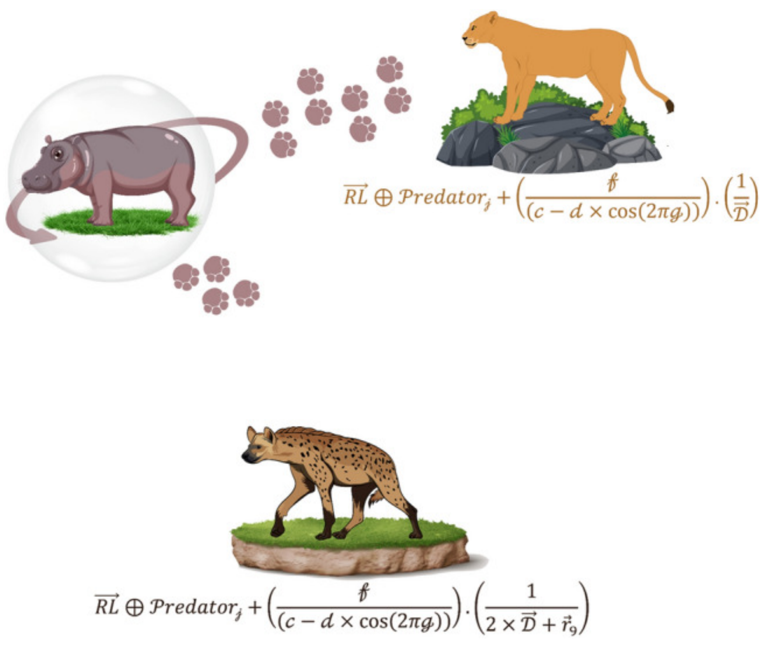

$\overrightarrow{RL} \oplus Predator_j + \left(\frac{f}{(c - d \times \cos(2\pi g))}\right) \cdot \left(\frac{1}{\overrightarrow{D}}\right)$
$\overrightarrow{RL} \oplus Predator_j + \left(\frac{f}{(c - d \times \cos(2\pi g))}\right) \cdot \left(\frac{1}{2 \times \overrightarrow{D} + \vec{r}_9}\right)$

Figure 3.20: Graphic representation of the phase 2 [8].

$$L_{evy}(V) = 0.05 \times \frac{w \times \sigma_w}{|v|^{\frac{1}{V}}} \tag{3.55}$$

Where:

w,v	: Represents random numbers in $[0, 1]$, respectively .
V	: Represents a constant (V = 1.5).
σ_w	: can be obtained by Equation 3.56.

$$\sigma_w = \left[\frac{\Gamma(1+v)\sin\left(\frac{\pi v}{2}\right)}{\Gamma\left(\frac{(1+v)}{2}\right) v 2^{\frac{(v-1)}{2}}} \right]^{\frac{1}{v}} \tag{3.56}$$

According to the Equation 3.57, if F_i^{HippoR} is greater than F, it means that the hippopotamus has been hunted and another hippopotamus will replace it in the herd, otherwise the hunter will escape and this hippopotamus will return to the herd. Significant enhancements were observed in the global search process during the second phase. The first and second phases complement each other and effectively mitigate the risk of getting trapped in local minima [8].

$$X_i = \begin{cases} X_i^{HippoR} F_i^{HippoR} & < F_i \\ X_i F_i^{HippoR} & \geq F_i \end{cases} \tag{3.57}$$

- **Phase 3: Hippopotamus Escaping from the Predator (Exploitation)**

Hippopotamus may exhibit defensive behaviour when confronted with a group of predators or unable to reject them. In this case, the hippopotamus attempts to leave the area Figure 3.21. To evade predators such as spotted lions and hyenas, hippopotamuses typically seek refuge in nearby lakes or ponds. This method allows the hippopotamus to find a safe spot near its current location. Modelling this behaviour in the Phase Tree of the HO improves the ability to exploit in local search. To imitate this behaviour, a random position is generated near the hippopotamuses' current location. The behaviour of hippopotamuses is represented using Equation 3.58 to Equation 3.61. When a newly formed position improves the cost function value, it signals that the hippopotamus has located a safer spot near its current location and moved accordingly. The current iteration is denoted as t, while the maximum iteration is represented by T [8].

$$lb_j^{local} = \frac{lb_j}{t}, \quad ub_j^{local} = \frac{ub_j}{t}, \quad t = 1, 2, \ldots, \mathcal{T} \tag{3.58}$$

$$X_i^{Hippo\mathcal{E}} : x_{ij}^{\mathcal{H}ippo\mathcal{E}} == x_{ij} + r_{10} \cdot \left(lb_j^{local} + s_1 \cdot \left(ub_j^{local} - lb_j^{local}\right)\right) \tag{3.59}$$

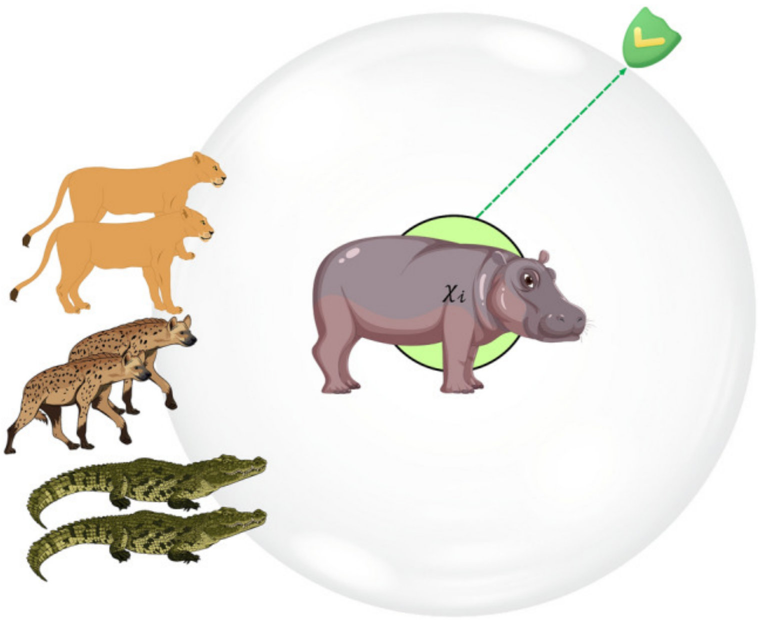


Figure 3.21: Drawing a Hippopotamus Escaping from the Predator [8].

$$i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, m$$

In Equation 3.59, $X_i^{\mathcal{H}ippo\mathcal{E}}$ represents the position of a hippopotamus that was searched for the closest safe place. Equation 3.60 selects s_1 as a random vector or number from three scenarios (s). The proposed approach improves its exploitation quality by resulting in a more appropriate local search scenario.

$$s = \begin{cases} 2 \times \vec{r}_{11} - 1 \\ r_{12} \\ r_{13} \end{cases} \tag{3.60}$$

In Equation 3.60 $\vec{r}_{11}$ represents a random vector between 0 and 1, while $r_1 0$ Equation 3.59 and r_{13} denote random numbers generated within the range of 0 and 1. Additionally, r_{12} is a normally distributed random number.

$$X_i = \begin{cases} X_i^{\mathcal{H}ippo\mathcal{E}} F_i^{\mathcal{H}ippo\mathcal{E}} & < F_i \\ X_i F_i^{\mathcal{H}ippo\mathcal{E}} & \geq F_i \end{cases} \tag{3.61}$$

To improve the performance of the HO algorithm, we did not categorize the population into immature, female, and male hippopotamuses, despite the fact that doing so would better reflect their nature [8].

- **Repetitions Process and Pseudo-Code of (HO)**

After each iteration of the HO algorithm, all population members are updated based on Phases 1–3 using Equation 3.55 to Equation 3.61. This procedure continues until the last iteration. The best possible answer is consistently recorded and stored as the algorithm is executed. After completing the procedure, the dominating hippopotamus solution emerges as the optimal answer to the problem. The pseudocode for Algorithm 5 displays the HO's procedural details [8].

Algorithm 5 Pseudo-code of the Proposed (HO) [8]

```
1:  Start HO
2:  Define an optimization problem
3:  Set the maximum number of iterations ($\mathcal{T}$) and number of hippopotamus ($\mathcal{N}$)
4:  Generate the initial position of all hippopotamus based on Figure 3.3.4 and objective function
    evaluation for this initial population
5:  for $t = 1 : \mathcal{T}$ do
6:      Update dominant hippopotamus position based on objective function value criterion
7:      Phase 1: The hippopotamus's position update in the river or pond (Exploration Phase)
8:      for $i = 1 : \mathcal{N}/2$ do
9:          Calculate the new position for $i$th hippopotamus using Equation 3.45 to Equation 3.48
10:         Update position of $i$th hippopotamus using Equation 3.50 to Equation 3.51
11:     end for
12:     Phase 2: Hippopotamus defence against predators (Exploration Phase)
13:     for $i = 1 + \mathcal{N}/2 : \mathcal{N}$ do
14:         Generate random position for predator using Equation 3.52
15:         Calculate the new position for $i$th hippopotamus using Equation 3.54
16:         Update the position of $i$th hippopotamus using Equation 3.57
17:     end for
18:     Phase 3: Hippopotamus Escaping from the Predator (Exploitation Phase)
19:     Calculate new bounds of variables decision using Equation 3.58
20:     for $i = 1 : \mathcal{N}$ do
21:         Calculate the new position for $i$th hippopotamus using Equation 3.59
22:         Update the position of $i$th hippopotamus using Equation 3.61
23:     end for
24:     Save the best candidate solution found so far
25: end for
26: Output the best solution of the objective function found by HO
27: End HO
```

3.3.5 Opposition Flow Directional Algorithm (OFDA)

The Proposed Opposition FDA (OFDA), created in 2022, is a technique that enhances the optimization process by introducing opposition-based learning (OBL) to prevent convergence to local optima. In OFDA, each flow in the optimization algorithm updates its solution based on an opposite flow in the search space, allowing for exploration in both directions. This method aims to improve the search process and decision-making by considering opposite directions in the optimization space. To understand OFDA, firstly, we will introduce FDA:

The Flow Directional Algorithm

The Flow Directional Algorithm (FDA) is detailed in [52]. Inspired by the behaviour of water flowing into a drainage basin, the FDA simulates the direction of flow towards the lowest height outlet points. The direction of flow is influenced by neighbouring flows and the slope, which is based on the D8 cell model (refer to Fig. 2 in [52]). Each flow position Flow_X and its height $\text{Flow}_{\text{fitness}}(f(\text{flow}_X))$ serves as a search agent for α flow, which is initialized within the drainage basin boundaries $[ub, lb]$. The new flow positions are estimated using two methods in the FDA.

Firstly, it is assumed that a flow generates its β neighbour flow Neighbor_X (refer to Eq. (3) in [52]) on its route to the drainage basin. Subsequently, the flow locations $\text{Flow}_{\text{new}X}$ (refer to Eq. (8) in [52]) are updated based on the best neighbour flow. The flow positions are further updated $\text{Flow}_{\text{new}X}$ (refer to Eq. (9) in [52]) in a second way by assuming that the present flow encounters a random flow and changes its path. Finally, the flow's position is updated if it is better than the old flow, which is expressed as:

$$\text{Flow}_X(i) = \begin{cases} \text{Flow}_{\text{new}X}(i) & \text{if } f(\text{Flow}_{\text{new}X}(i)) < f(\text{Flow}_X) \\ \text{Flow}_X(i) & \text{otherwise} \end{cases} \quad \forall i \in [1, \alpha] \tag{3.62}$$

where $f(\text{Flow}_{\text{new}X}(i))$ is the height of $\text{Flow}_{\text{new}X}(i)$. The flow position is updated iteratively until it reaches the optimal solution at the maximum iteration Max_Iter. The FDA demonstrated outstanding performance on benchmark functions and showed better results for real-world engineering design problems. More information regarding the FDA can be found in [52].

The Proposed Opposition FDA (OFDA)

As discussed earlier, each flow in the FDA updates its solution in the search space based on its random neighbour or some random flow. This may steer the flow to a local optimum solution, which may be a trap. This can be avoided by introducing opposition-based learning (OBL), which aids in the search process in both directions. Let us assume $\text{Flow}_X(i) = \{\text{Flow}_x(i,1), \text{Flow}_x(i,2), \ldots, \text{Flow}_x(i,d)\}$ be a flow in the d dimensional search space in the range $[LB, UB]$, where $LB = \{lb(1), lb(2), \ldots, lb(d)\}$ and $UB = \{ub(1), ub(2), \ldots, ub(d)\}$.
Then, the opposite flow $\text{OFlow}_X(i) = \{\text{OFlow}_x(i,1), \text{OFlow}_x(i,2), \ldots, \text{OFlow}_x(i,d)\}$ is determined as:

$$\text{OFlow}_x(i,j) = ub(j) + lb(j) - \text{Flow}_x(i,j), \quad \forall i \in [1, \alpha] \text{ and } \forall j \in [1, d] \tag{3.63}$$

Finally, the OFDA's selection-based updating rule is described as follows:

$$\text{Flow}_X(i) = \begin{cases} \text{OFlow}_X(i) & \text{if } f(\text{OFlow}_X(i)) < f(\text{Flow}_X(i)) \\ \text{Flow}_X(i) & \text{Otherwise} \end{cases}, \quad \forall i \in [1, \alpha] \tag{3.64}$$

where $f(\text{OFlow}_X(i))$ is the height of $\text{OFlow}_X(i)$. The OFDA implementation is illustrated in Figure 3.22 [9].

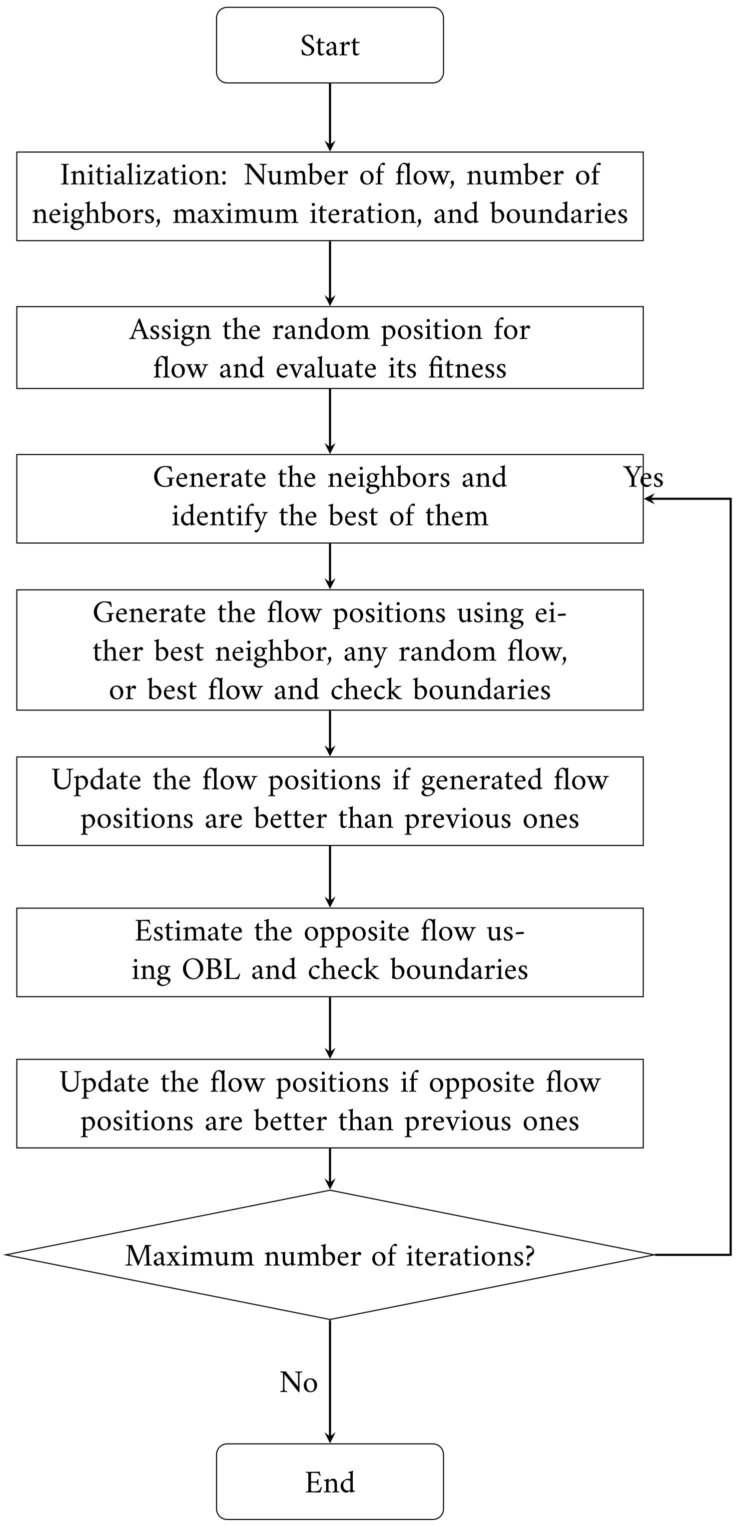

Start
Initialization: Number of flow, number of neighbors, maximum iteration, and boundaries
Assign the random position for flow and evaluate its fitness
Generate the neighbors and identify the best of them
Yes
Generate the flow positions using either best neighbor, any random flow, or best flow and check boundaries
Update the flow positions if generated flow positions are better than previous ones
Estimate the opposite flow using OBL and check boundaries
Update the flow positions if opposite flow positions are better than previous ones
Maximum number of iterations?
No
End

Figure 3.22: Flowchart of the optimization algorithm [9].

3.4 Results and discussion

This section employs the proposed algorithms: SBOA [7], GRO [6], ZOA [14], HOA [8] and OFDA [9] to estimate the unknown parameters of two PV models: SDM and DDM. These five algorithms, being novel and recently developed, offer fresh approaches to parameter estimation in PV models. For a fair comparison, GRO and SBOA algorithms are run 30 times using 30 search agents through 500 iterations, whereas the ZOA, HOA and OFDA algorithms are run 30 times using 30 search agents through 1000 iterations. The experimental measurements of current vs voltage using a 57 mm diameter commercial (R.T.C. France) silicon solar cell operating at 33 °C under 1000 W/m^2 irradiance and thoroughly contains 26 sets of current and voltage values taken from [11] are adopted in this study. Table 3.4 shows the terminal measuring of current and voltage. In addition, the Parameter ranges of each unknown parameter for SDM and DDM models is presented in Table 3.3 [10].

Table 3.5 displays the specifications of the device used in the experiments conducted in this work using Matlab R2018b.

	Name	Configuration
Hardware	CPU	Intel Core i7-5600U
	Frequency	2.6 Giga Hertz
	RAM	16 Giga Byte
	Hard Drive	238.47 Giga Byte
Software	Operating System	64 bit Windows 10 Pro
	Language	MATLAB 2018b

Table 3.5: The specification of hardware used in the simulation.

3.4.1 Comparison over single-diode model (SDM)

This section presents the results obtained by the proposed algorithms when solving the SDM over RTC France cells. Our analysis is divided into two parts:

- **Quantitative Analysis:** This examines the performance of the algorithms in terms of the best, average, worst, and standard deviation of RMSE values obtained over 30 independent runs, as described in Table 3.6. Additionally, it includes a Computational Efficiency Test, evaluating the algorithms based on their runtime.
- **Qualitative Analysis:** This visually compares the algorithms by analyzing their convergence curves, as well as depicting the I–V curves of both measured and estimated data and the difference between measured and estimated currents.

Metric	Description	Formula
Best	The smallest value estimated through the independent times is distinguished as the best value and used to compare the performance of the different algorithms	Best $= \min_i, i = 1, 2, 3, \ldots, \mathrm{NoR}(27)$ where f_i is the fitness value of the ith independent run, and NoR is the length of independent runs.
Average	The average of the fitness values obtained through the independent runs is computed and compared among the different algorithms to measure their stability	$\mathrm{Avg} = \frac{\sum_{i=0}^{\mathrm{NoR}} f_i}{\mathrm{NoR}}(28)$
Worst	The greatest fitness value obtained through the number of independent runs is another metric used to compare the efficacy of the algorithms and calculated as follows	Worst $= \max_i, i = 1, 2, 3, \ldots, \mathrm{NoR}(29)$
Standard deviation (SD)	Another statistical metric known as standard deviation (SD) is used to measure the amount of dispersion of the outcomes obtained by each algorithm	$\mathrm{SD} = \sqrt{\frac{\sum_{i=0}^{\mathrm{NoR}}(f_i - \mathrm{Avg})^2}{\mathrm{NoR}}}(30)$

Table 3.6: Performance metrics description [12].

Quantitative Analysis

This sub-subsection compares statistical information (best, average, worst, and standard deviation) of proposed algorithms for RTC France solar cells.

Table 3.7 demonstrates the comparison of results for SDM. This table provides the best RMSE and extracted parameters from each algorithm. There are only five parameters $\mathbf{X} = \{I_{ph}, I_{sd}, R_S, R_{Sh}, n\}$ which are required to be optimized in the case of SDM. Based on the output data in Table 3.7, the most successful algorithm for SDM was OFDA because this algorithm produced the smallest RMSE (**9.8623239349e-04**). While the second-best RMSE (**1.0137326785e-03**) was achieved by SBOA followed by ZOA (**3.1619917743e-03**), GRO (**2.7675043877e-03**) and HOA (**9.7512084076e-03**) respectively. The accuracy of the parameter values was measured by the RMSE.

Table 3.6 displays the performance metrics description. The minimum RMSE number indicates algorithm accuracy, the average RMSE represents mean accuracy, and the standard deviation (SD) represents system reliability. Based on the results presented in Table 3.8, it was observed that the proposed OFDA achieved the best accuracy and reliability in the SDM.

Figure 3.23 compares the average computational times of the five proposed metaheuristic algorithms OFDA, SBOA, ZOA, GRO, and HOA in estimating parameters for SDM (in purple) and DDM (in red) photovoltaic models. SBOA and GRO are the most efficient, consuming only 7.97 and 6.13 seconds, respectively. OFDA takes 24.52 seconds, which is considerably higher but still more efficient compared to ZOA and HOA, which require 17.46 and 40.86 seconds, respectively.

Qualitative Analysis

This sub-subsection compares the proposed methods based on the convergence curve, I-V characteristic, and P-V characteristic curves.

Figure 3.24 shows the convergence behaviour of the proposed methods when applied to the SD. Upon analyzing the convergence curves, we find that the suggested method, OFDA, outperformed all other competing algorithms in reaching the near-optimal solution, with SBOA coming in second. Nevertheless, it is clear that the suggested OFDA performs better because it routinely obtains the best result and the lowest RMSE value in previous iterations. OFDA's best performance is due to good exploration. However, ZOA performs the second best in terms of RMSE, followed by GRO. The worst performance comes from HOA, this is due to an issue with these algorithms known as premature convergence, which is brought about by unequal exploration and exploitation.

The P-V and I-V curves for the SDM were based on the estimated data from the five algorithms at the best RMSE, which is explained in Figure 3.24. These figures show an impressive agreement between the estimated data and the experimental data across the entire voltage range from the proposed OFDA. This close match validates the accuracy of the OFDA in capturing the behaviour of the SD model. While the rest of the algorithms show a random dispersion of their solutions, we can intuit that it is due to the nature of their operators which have different degrees of exploration and exploitation.

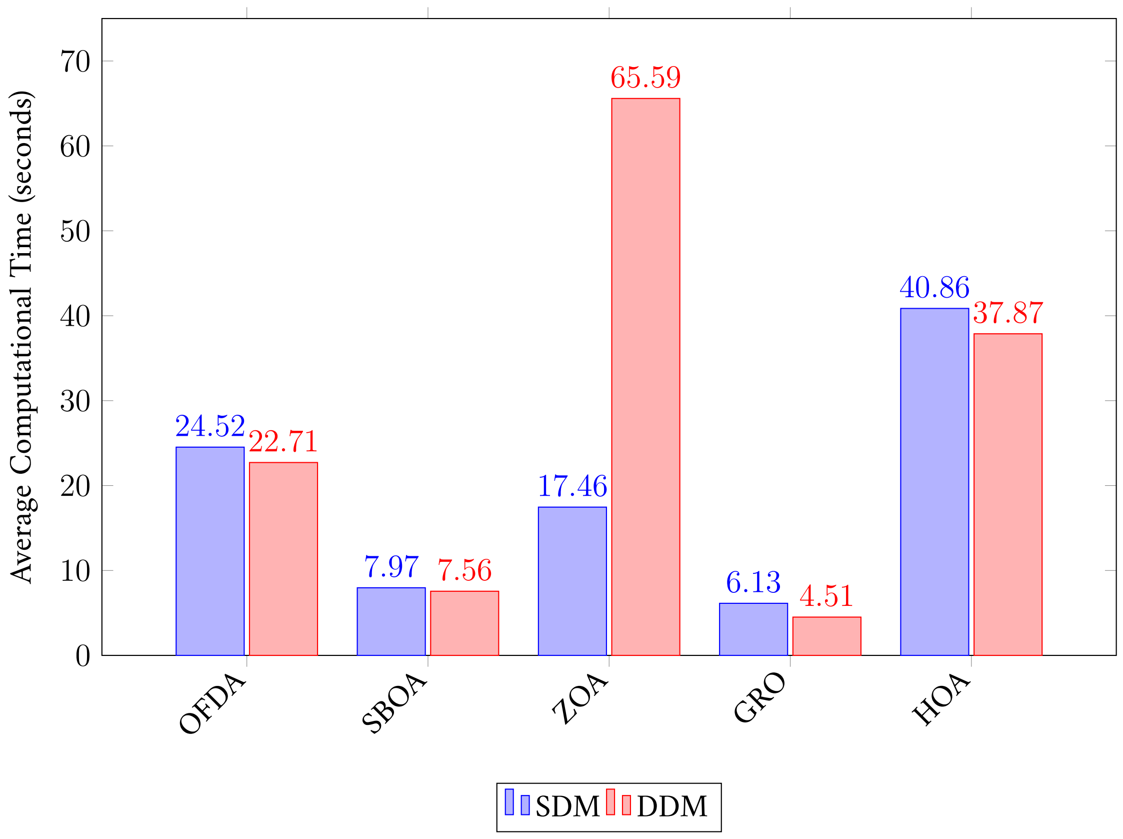
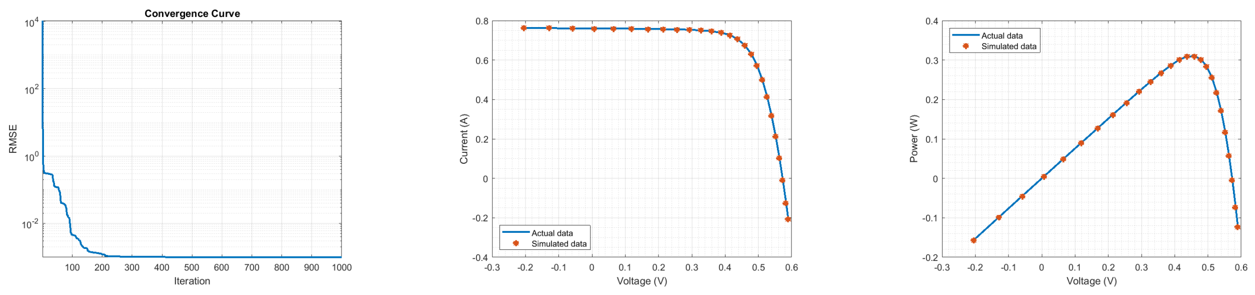
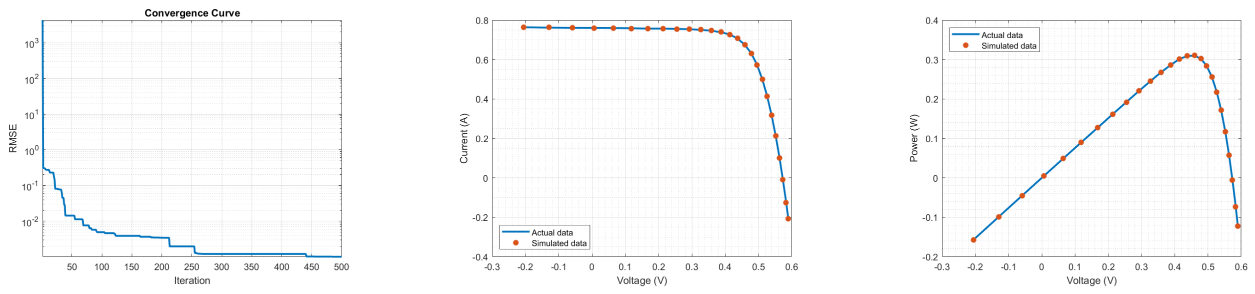


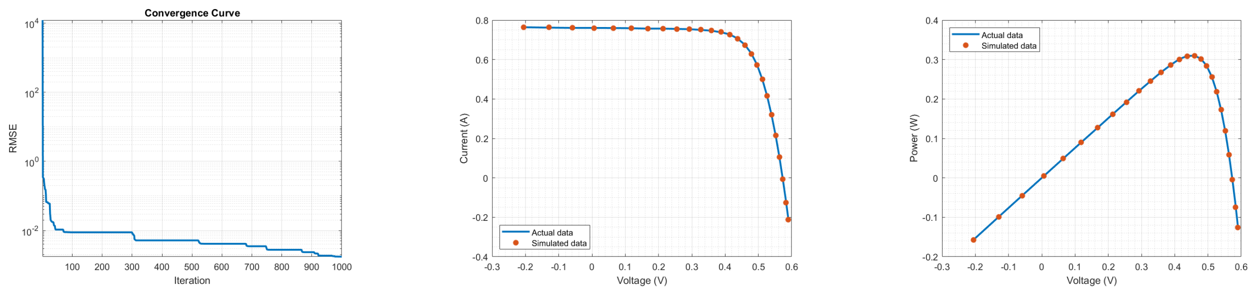
Figure 3.23: The average CPU time consumed by different algorithms on two models.



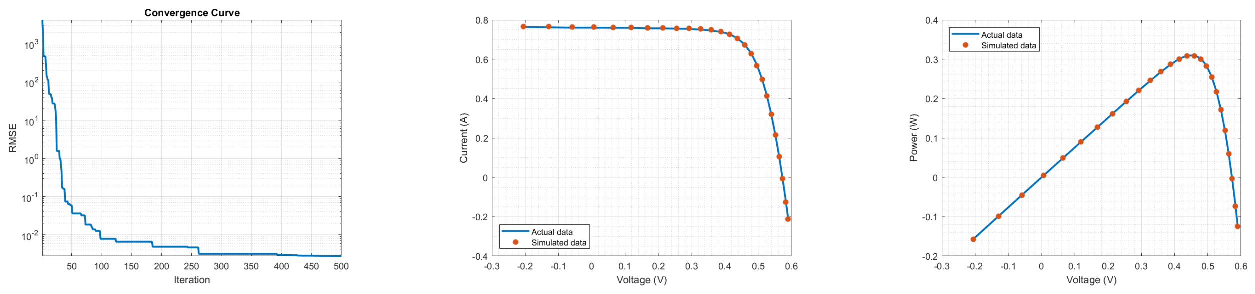
(a) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **OFDA** on the **SD** model.



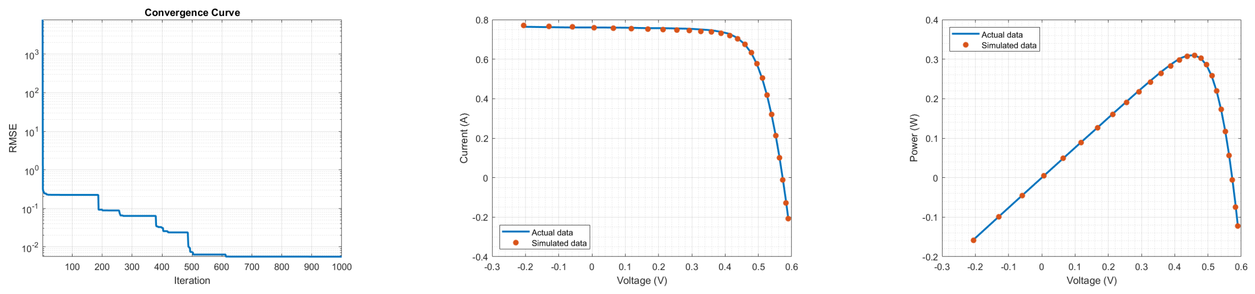
(b) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **SBOA** on the **SD** model.



(c) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **ZOA** on the **SD** model.



(d) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **GRO** on the **SD** model.



(e) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **HOA** on the **SD** model.

Figure 3.24: The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by the five algorithms on the **SD** model.

3.4.2 Comparison over double-diode model (DDM)

Like SDM discussed in the previous section, This section compares the performance of proposed and other algorithms in solving the DDM over RTC France cell, including best, average, worst, and standard deviation of RMSE values, as well as their visual performance.

Quantitative Analysis

This section analyzes statistical data (best, average, worst, and standard deviation) on proposed methods for RTC France solar cells.

Table 3.7 demonstrates the comparison of results for DDM. This table summarizes the optimal RMSE and extracted parameters for each approach.
There are seven parameters $\mathbf{X} = \{I_{ph}, I_{sd1}, I_{sd2}, R_S, R_{Sh}, n_1, n_2\}$ which are required to be optimized in the case of DDM. OFDA again shows the lowest RMSE (**9.8602169813e-04**), demonstrating superior accuracy. SBOA and GRO have notable RMSE values of (**1.7430803946e-03**) and (**2.1963134144e-03**), respectively. ZOA and HOA also have higher RMSE values of (**3.1619917743e-03**) and (**9.7512084076e-03**). Table 3.8 shows that the proposed OFDA achieved the highest accuracy and reliability in the DDM.

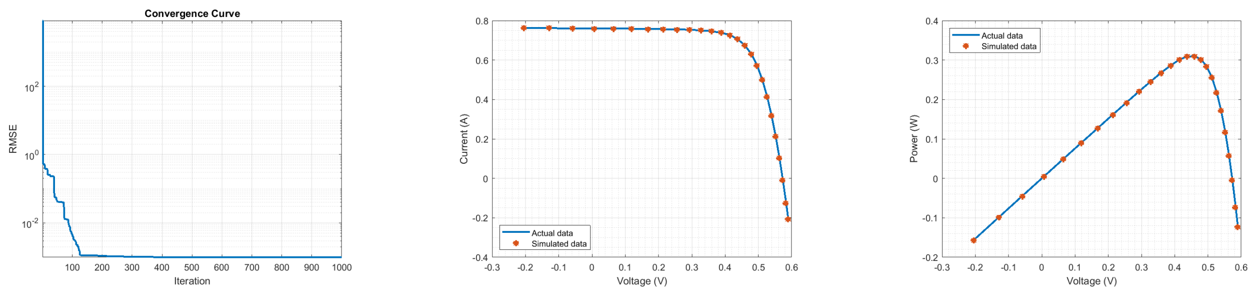
Regarding CPU time, OFDA takes 22.71 seconds, which is marginally lower than its time for the SDM model. SBOA remains efficient at 7.56 seconds. ZOA's CPU time is significantly higher at 65.59 seconds, while GRO and HOA are lower at 4.51 and 37.87 seconds, respectively as shown in Figure 3.23.

Qualitative Analysis

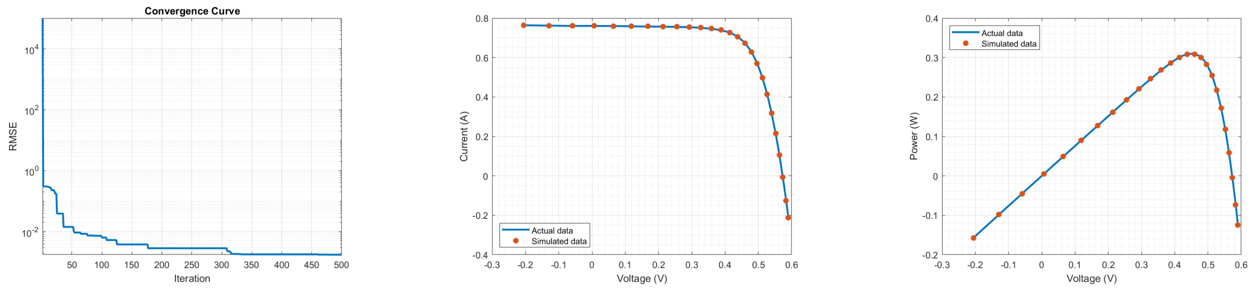
This sub-section compares proposed approaches using convergence, I-V, and P-V curves.

Figure 3.25 illustrates the convergence behaviour of the proposed approaches when applied to DD. The proposed method, OFDA, outscored all other competing algorithms in obtaining the near-optimal solution, with SBOA coming in second, according to our analysis of the convergence curves. However, since the recommended OFDA consistently yields the best outcome and the lowest RMSE value in prior rounds, it is evident that it performs better. Effective exploration is the reason behind OFDA's highest results. The worst RMSE values are found in GRO, ZOA, and HOA. This is because limited solution space exploration causes these algorithms to become trapped in local minima.

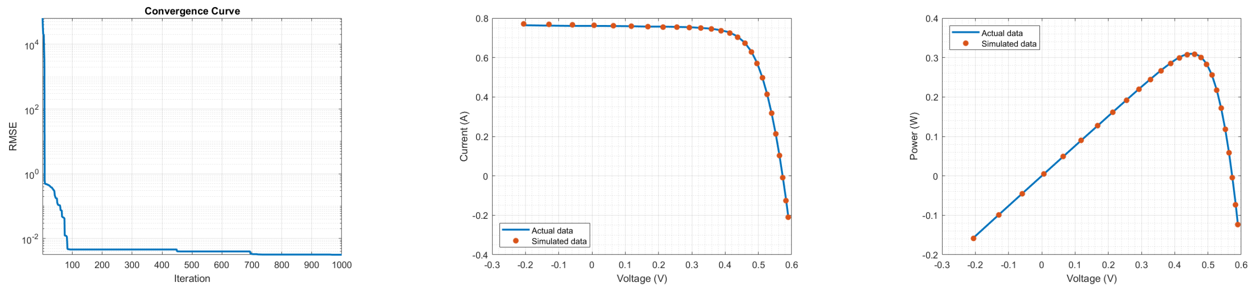
The P-V and I-V curves for the DDM were created using the estimated data from the five algorithms at the best RMSE, as shown in Figure 3.25. This figure showed that the measured and simulation data produced by the suggested OFDA for DDM aligned. As a result, the OFDA algorithm-based DDM performed more effectively. The rest of the algorithms have a random distribution of solutions.



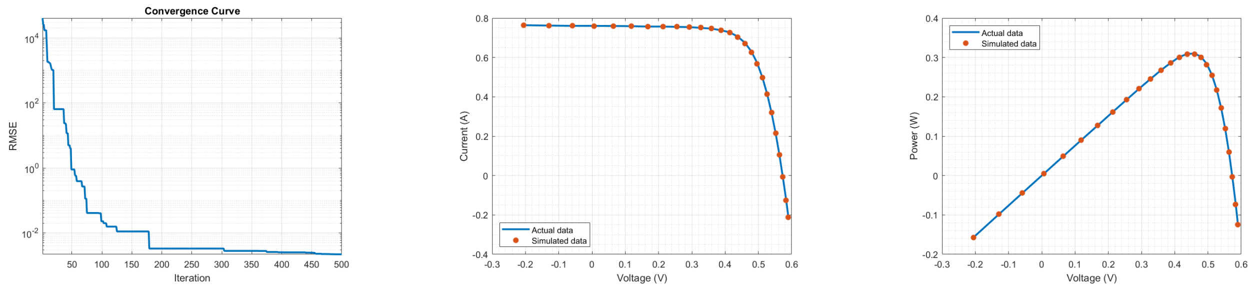
(a) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **OFDA** on the **DD** model.



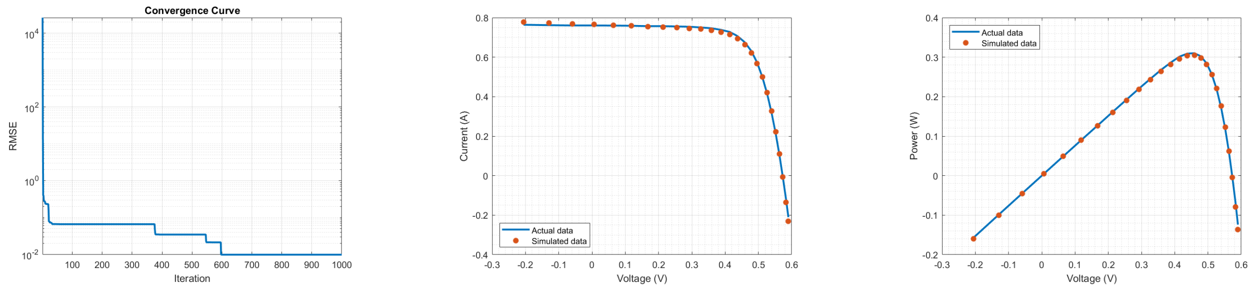
(b) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **SBOA** on the **DD** model.



(c) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **ZOA** on the **DD** model.



(d) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **GRO** on the **DD** model.



(e) The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by **HOA** on the **DD** model.

Figure 3.25: The curves of Convergence, I-V characteristic and P-V characteristic respectively based on the parameters identified by the five algorithms on the **DD** model.

Table 3.7: Estimated parameters with best RMSE for all compared algorithms for SD and DD model.

Model	Algorithms	Parameters							RMSE
		I_{ph} (A)	I_{sd} (μA)	I_{sd2} (μA)	R_s (Ω)	R_{sh} (Ω)	η	η_1	η_2	
SDM	**OFDA**	**0.7607744702**	**0.0000003197**	-	**0.0364204044**	**53.4848003330**	**1.4801417493**	-	-	**9.8623239349e-04**
	SBOA	0.7607584215	0.0000002865	-	0.0368703600	51.6437105061	1.4691956346	-	-	1.0137326785e-03
	ZOA	0.7608602483	0.0000004991	-	0.0338144737	54.5116463931	1.5266560048	-	-	1.7443100559e-03
	GRO	0.7637307382	0.0000008319	-	0.0320045168	53.6530156445	1.5832635967	-	-	2.7675043877e-03
	HOA	0.7619713152	0.0000000475	-	0.0413127913	18.6124493506	1.3117486238	-	-	5.5440469014e-03
DDM	**OFDA**	**0.7607760063**	**0.0000000000**	**0.0000003231**	**0.0363764780**	**53.7222351312**	-	**1.8217926025**	**1.4812001644**	**9.8602169813e-04**
	SBOA	0.7608838009	0.0000006651	0.0000000000	0.0332269118	72.7400021194	-	1.5577686748	1.9964278803	1.7430803946e-03
	ZOA	0.7649976154	0.0000003025	0.0000000000	0.0357875890	25.8645574342	-	1.4756686854	1.0448570198	3.1619917743e-03
	GRO	0.7603118761	0.0000005906	0.0000005902	0.0324489503	72.9450578860	-	1.9267876563	1.5517226016	2.1963134144e-03
	HOA	0.7667780932	0.0000007646	0.0000000000	0.0270413729	15.9261912417	-	1.5770449642	1.6219986881	9.7512084076e-03

Note: **Bold** values indicate the best results.

Table 3.8: Statistical outcomes of RMSE for all compared algorithms for SD and DD model.

Model	Algorithms	RMSE				MaxIt	Pop
		Minimum	Average	Maximum	SD		
SDM	**OFDA**	**9.8623239349e-04**	**2.6457675220e-03**	**1.2435918010e-02**	**3.0742022205e-03**	1000	30
	SBOA	1.0137326785e-03	2.0823032531e-01	2.2286139909e-01	5.5683848341e-02	500	30
	ZOA	1.7443100559e-03	2.0146388006e-01	2.2301020774e-01	6.5471006193e-02	1000	30
	GRO	2.7675043877e-03	5.8143666400e-03	2.7675043877e-03	1.1157125884e-03	500	30
	HOA	5.5440469014e-03	9.6307213754e-02	2.2287712391e-01	9.8742218282e-02	1000	30
DDM	**OFDA**	**9.8602169813e-04**	**1.8602666640e-03**	**7.9297704375e-03**	**1.3503866492e-03**	1000	30
	SBOA	1.7430803946e-03	1.6495376175e-01	2.2286139909e-01	9.7675690290e-02	500	30
	ZOA	3.1619917743e-03	1.8623892257e-01	2.2300026789e-01	7.5754966267e-02	1000	30
	GRO	2.1963134144e-03	6.2833720169e-03	7.7677012220e-03	1.2106233944e-03	500	30
	HOA	9.7512084076e-03	1.2183577144e-01	2.2563048072e-01	9.7271617605e-02	1000	30

Note: **Bold** values indicate the best results.

3.5 Conclusion

This chapter introduces a novel application of five algorithms to estimate the parameters of two photovoltaic (PV) models: the Single Diode Model (SDM) and the Double Diode Model (DDM). The SDM is represented by a nonlinear current and voltage equation involving five unknown parameters. Similarly, the DDM functions like the SDM but includes seven unknown parameters. The objective function for parameter extraction in both models aims to minimize the root mean square error between the simulated and experimental currents of the R.T.C. France solar cell. The proposed OFDA, SBOA, ZOA, GRO, and HOA are recent optimization techniques used to minimize this objective function for PV cells using SDM and DDM. Among these, the OFDA algorithm demonstrated superior performance in terms of solution accuracy and convergence speed. Consequently, the results obtained by OFDA were more precise than those from the four other algorithms, making OFDA a promising candidate for optimizing solar cell systems.

General Conclusion

Conclusion

This study compares five metaheuristic algorithms to optimize the parameters of two photovoltaic (PV) models: the Single Diode Model (SDM) and the Double Diode Model (DDM). The objective function is to minimize the root mean square error (RMSE) between simulated and experimental currents of the PV models.

The proposed algorithms are the Opposition Flow Directional Algorithm (OFDA), Secretary Bird Optimization Algorithm (SBOA), Zebra Optimization Algorithm (ZOA), Gold Rush Optimizer (GRO), and Hippopotamus Optimization Algorithm (HOA). These algorithms are recent metaheuristic techniques used for parameter estimation in PV cells. The analysis aims to determine which algorithm achieves the best balance between accuracy, convergence speed, and computational time.

Out of these algorithms, the Opposition Flow Directional Algorithm (OFDA) shows superior performance in terms of accuracy, balance, and convergence speed. As a result, OFDA algorithm produces more accurate results compared to the other four algorithms, establishing it as a promising choice for enhancing PV systems.

Perspectives

The comparative analysis offers a fundamental insight into the performance of these five metaheuristic algorithms for estimating PV parameters. Several potential research directions can improve the effectiveness and applicability of these methods:

Incorporating Chaotic Maps in Optimization: Chaotic maps can improve the exploration and exploitation abilities of algorithms when integrated into the optimization process. By using chaotic sequences, algorithms can avoid local optima and achieve superior global search results. This method is especially advantageous for intricate optimization challenges in PV parameter estimation, characterized by highly nonlinear and multi-modal solution spaces.

Opposition-Based Learning (OBL): Opposition-Based Learning can be further utilized to enhance the convergence speed and accuracy of optimization algorithms in future studies. It is beneficial for increasing population diversity and generating improved initial solutions, thereby speeding up the convergence process. Future research could investigate various strategies for integrating OBL with current algorithms to achieve enhanced performance.

Hybrid Approaches: By combining chaotic maps and Opposition-Based Learning (OBL) with traditional metaheuristic algorithms, it is possible to create hybrid approaches that capitalize on the strengths of each method. These hybrid algorithms have the potential to deliver better performance by effectively balancing exploration and exploitation, as well as offering mechanisms to avoid getting stuck in local optima. Research on these hybrid approaches could result in substantial enhancements in the accuracy and efficiency of estimating PV model parameters.

Advanced Algorithm Variants: Future research could concentrate on creating advanced versions of current algorithms that include adaptive mechanisms to dynamically adjust their parameters throughout the optimization process. This enhancement can improve the algorithms' capacity to handle the dynamic characteristics of the optimization landscape in PV parameter estimation, resulting in more precise and dependable outcomes.

Real-World Applications: It is important to further explore the practical use of these optimized models in real-world situations. By implementing these models in different geographical locations and varying environmental conditions, we can gain valuable insights into their reliability and flexibility. This process would also include validating the models using experimental data from different types of PV modules and various operating conditions.

On the whole, by focusing on these areas, future research can expand on the findings of this study to create more efficient optimization methods for estimating PV parameters. This will ultimately help advance renewable energy technologies and enhance the performance of PV system.

Bibliography

[1] P. Festa. A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *Journal of Applied Research on Industrial Engineering*, 1, 2021.

[2] Mohamed Abdel-Basset, Reda Mohamed, Marwa Sharawi, Laila Abdel-Fatah, Mohamed Abouhawwash, and Karam Sallam. A comparative study of optimization algorithms for parameter estimation of pv solar cells and modules: Analysis and case studies. *Energy Reports*, 8:13047–13065, 2022. URL: https://doi.org/10.1016/j.egyr.2022.09.193.

[3] Rafa Elshara, Aybaba Hançerlioğullari, Javad Rahebi, and Jose Manuel Lopez-Guede. Pv cells and modules parameter estimation using coati optimization algorithm. *Energies*, 17(7), 2024. URL: https://doi.org/10.3390/en17071716.

[4] Mehmet Yesilbudak. Parameter extraction of photovoltaic cells and modules using grey wolf optimizer with dimension learning-based hunting search strategy. *Energies*, 14(18), 2021. URL: https://doi.org/10.3390/en14185735.

[5] Sofiane Haddad, Badis Lekouaghet, Mohamed Benghanem, Ammar Soukkou, and Abdelhamid Rabhi. Parameter estimation of solar modules operating under outdoor operational conditions using artificial hummingbird algorithm. *IEEE Access*, 10:51299–51314, 2022. URL: 10.1109/ACCESS.2022.3174222.

[6] Kamran Zolfi. Gold rush optimizer: A new population-based metaheuristic algorithm. *Oper. Res. Decis.*, 33, 2023. URL: https://api.semanticscholar.org/CorpusID:258203735.

[7] Youfa Fu, Dan Liu, Jiadui Chen, and Ling He. Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems. *Artificial Intelligence Review*, 57, 04 2024. URL: https://doi.org/10.1007/s10462-024-10729-y.

[8] Mohammad Amiri, Nastaran Mehrabi Hashjin, Mohsen Montazeri, Seyedali Mirjalili, and Nima Khodadadi. Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm, 10 2023. URL: https://doi.org/10.21203/rs.3.rs-3503110/v1.

[9] Rutuparna Panda, Monorama Swain, Manoj Kumar Naik, Sanjay Agrawal, and Ajith Abraham. A novel practical decisive row-class entropy-based technique for multilevel threshold selection using opposition flow directional algorithm. *IEEE Access*, 10:110473–110484, 2022.

[10] Jing Liang, Kangjia Qiao, Kunjie Yu, Shilei Ge, Boyang Qu, Ruohao Xu, and Ke Li. Parameters estimation of solar photovoltaic models via a self-adaptive ensemble-based differential evolution. *Solar Energy*, 207:336–346, 09 2020.

[11] Diab Mokeddem. Parameter extraction of solar photovoltaic models using enhanced levy flight based grasshopper optimization algorithm. *Journal of Electrical Engineering & Technology*, 16:1–9, 11 2020. URL: https://doi.org/10.1007/s42835-020-00589-1.

[12] Mohamed Abdel-Basset, Reda Mohamed, Ripon K. Chakrabortty, Karam Sallam, and Michael J. Ryan. An efficient teaching-learning-based optimization algorithm for parameters identification of photovoltaic models: Analysis and validations. *Energy Conversion and Management*, 227:113614, 2021. URL: https://doi.org/10.1016/j.enconman.2020.113614.

[13] S. Bandaru. *Decision Sciences Metaheuristic Techniques*. Taylor & Francis Group, 2016.

[14] Eva Trojovská, Mohammad Dehghani, and Pavel Trojovský. Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm. *IEEE Access*, 10:49445–49473, 2022. URL: https://ieeexplore.ieee.org/document/9768820.

[15] K. Lange. *Optimization*. Springer Science & Business Media, 2013.

[16] S. J. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[17] G. Cornuejols. *Optimization Methods in Finance*. Pittsburgh: Cambridge University Press, 2006.

[18] P. Festa. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. *International Conference on Transparent Optical Networks (ICTON)*, 1-20, 2014.

[19] T. El-Ghazali. *Metaheuristics From Design to Implementation*. John Wiley & Sons, Inc. Publication, 2009.

[20] A. Blum, CH. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Research Gate*, 1:271–272, January 2001.

[21] Y. Xin-She. *Engineering Optimization*. JOHN WILEY & SONS, 2010.

[22] D. et al Sachin. Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *International Journal if Computer Engineering in Research Trends*, 2:229, MAY 2015.

[23] Y. Xin-She. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2010.

[24] T. et al Vinita. Metaheuristic algorithms for optimization: A brief review †. *Engineering Proceedings*, 1:3, March 2024.

[25] T. Weise. *An Introduction to Optimization Algorithms*. Institute of Applied Optimization (IAO), School of Artificial Intelligence and Big Data, Hefei University, 2020.

[26] H. et al Patrick. A survey of ai-based meta-heuristics for dealing with local optima in local search. *Research Gate*, 1:3–4, October 2004.

[27] Y. Xin-She. Optimization algorithms. *Research Gate*, 1, 2014.

[28] H. Schwefel. *Evolution and Optimum Seeking*. JOHN WILEY & SONS, 1995.

[29] K. Blom. Identifying properties of real-world optimisation problems through a questionnaire. *ResearchGate*, 2023.

[30] N. Abd-Alsabour. On tackling real-life optimization problems. *International Journal in Advanced Science Engineering Information Technology*, 2019.

[31] Y. Xin-She. Optimization and metaheuristic algorithms in engineering. *Elsevier*, 2013.

[32] A . Matta. Simulation optimization with mathematical programming representation of discrete event systems. *ResearchGate*, 2008.

[33] G. Huerta-Cuellar. Mathematical modeling and optimization for real life phenomena. *Research-Gate*, 2024.

[34] G . C. V. Ramadas. On metaheuristics for solving the parameter estimation problem in dynamic systems: A comparative study. *Journal of Optimization*, 2018.

[35] E . Varol Altay. A comparative study of metaheuristic optimization algorithms for solving real-world engineering design problems. *Tech Science Press*, 2023.

[36] K . Erwin. Meta-heuristics for portfolio optimization. *Springer*, 2023.

[37] A. Duman. Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. *Resaerch Gate*, 2013.

[38] N . Manavizadeh. Using a metaheuristic algorithm for solving a home healthcare routing and scheduling problem. *Journal of Project Management*, 2020.

[39] S . Kaur. A systematic review on metaheuristic optimization techniques for feature selections in disease diagnosis: Open issues and challenges. *National Library of Medicine*, 2022.

[40] P . Melin. Comparative study of metaheuristic optimization of convolutional neural networks applied to face mask classification. *MDPI*, 2020.

[41] P . AGRAWAL. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access*, 2021.

[42] Hamdy Sultan, Mahmoud Mossa, and Almoataz Abdelaziz. *Parameter Identification of Solar Cell Mathematical Models Using Metaheuristic Algorithms*. 01 2024.

[43] Davut Izci, Serdar Ekinci, and Abdelazim Hussien. Efficient parameter extraction of photovoltaic models with a novel enhanced prairie dog optimization algorithm. *Scientific Reports*, 14:7945, 04 2024. URL: https://doi.org/10.1038/s41598-024-58503-y.

[44] Abhishek Sharma, Abhinav Sharma, Moshe Averbukh, Vibhu Jately, Shailendra Rajput, Brian Azzopardi, and Wei Hong Lim. Performance investigation of state-of-the-art metaheuristic techniques for parameter extraction of solar cells/module. *Scientific Reports*, 13, 07 2023. URL: https://doi.org/10.1038/s41598-023-37824-4.

[45] Kezban Koç, Mehmet Demirtaş, and İpek Çetinbaş. Parameter extraction of photovoltaic models by honey badger algorithm and wild horse optimizer. *Politeknik Dergisi*, 26(4):1453–1465, 2023. URL: https://doi.org/10.2339/politeknik.1155696.

[46] Ahmed Diab, Mohamed Tolba, Ali El-Rifaie, and Kotin Denis. Photovoltaic parameter estimation using honey badger algorithm and african vulture optimization algorithm. *Energy Reports*, 8:384–393, 11 2022.

[47] Ahmed Diab, Hamdy Sultan, Ton Do, Omar Kamel, and Mahmoud Mossa. Coyote optimization algorithm for parameters estimation of various models of solar cells and pv modules. *IEEE Access*, 06 2020. URL: :https://www.researchgate.net/publication/342040306.

[48] Radhakrishnan Venkateswari and Natarajan Rajasekar. Review on parameter estimation techniques of solar photovoltaic systems. *International Transactions on Electrical Energy Systems*, 31, 09 2021. URL: https://doi.org/10.1002/2050-7038.13113.

[49] Samuel R. Fahim, Hany M. Hasanien, Rania A. Turky, Shady H. E. Abdel Aleem, and Martin Ćalasan. A comprehensive review of photovoltaic modules models and algorithms used in parameter extraction. *Energies*, 15(23), 2022. URL: https://doi.org/10.3390/en15238941.

[50] Zaiyu Gu, Guojiang Xiong, and Xiaofan Fu. Parameter extraction of solar photovoltaic cell and module models with metaheuristic algorithms: A review. *Sustainability*, 15(4), 2023. URL: https://doi.org/10.3390/su15043312.

[51] Ahmed Ginidi, Sherif M. Ghoneim, Abdallah Elsayed, Ragab El-Sehiemy, Abdullah Shaheen, and Attia El-Fergany. Gorilla troops optimizer for electrically based single and double-diode models of solar photovoltaic systems. *Sustainability*, 13(16), 2021. URL: https://doi.org/10.3390/su13169459.

[52] Hojat Karami, Mahdi Valikhan Anaraki, Saeed Farzin, and Seyedali Mirjalili. Flow direction algorithm (fda): a novel optimizer approach for solving optimization problems. *Computers & Industrial Engineering*, 156:107224, 03 2021. URL: https://doi.org/10.1016/j.cie.2021.107224.