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

جامعة قاصدي مرباح ورقلة
Kasdi Merbah University of Ouargla

Academic Master Thesis

To obtain a master's degree in Computer Science

Major : Fundamental Computing

A Contrastive Analysis Between the Performance of Four Metaheuristic Algorithms for Parameter Estimation in Photovoltaic Models

Realized by :
Djouhina Korichi
Dounia Hamdi

Supervised by :
Dr. Farouq Zitouni
(UKMO)

Presented in 22 *June* 2024, *in front of the jury composed of* :

Dr.Bekkari Fouad : UKMO - President
Dr.Meriem Khelifa : UKMO - Examiner

Promotion : 2023/2024

Dedication

الحمد لله رب العالمين, تبارك وتعالى, له الكمال وحده ,

والصلاة والسلام على نبينا محمد نبيه ورسوله الأمين

وعلى سائر الأنبياء والمرسلين

أحمد الله تعالى على الذي بارك لي في إتمام بحثي هذا

وأتقدم بجزيل الشكر وخالص الإمتنان

إلى هديّتي من الله، والنعمة الكبيرة التي أعيشها، أبي وأمي،

عسى أن يكون هذا العمل صدقة جارية عنّي وعنكم

إلى إخوتي، الذين كانوا دائمًا سندا ودعما في كل خطوة

أخصكم بشكري وامتناني الكبيرين

إلى زوجة أخي، شكراً على تشجيعِك الدائمِ لي وشكراً لِوجودِكِ كَأختٍ لي

إلى صديقاتي، شكرًا لكم على وجودكم في حياتي ودعمكم لي في رحلتي الأكاديمية

إلى الأقارب الذين وقفوا إلى جانبي، كما وقف أهلي،

لكم جزيل الشكر ووافر الاحترام على التمنيات بالنجاح والدعم والتشجيع

- Djouhina -

Dedication

أحمد الله عز وجل على منّه وعونه لإتمام هذا البحث

أتقدم بخالص الشكر والامتنان إلى من كان سبباً في وجودي وقوتي ودعمي، ومصدر طاقتي: أمي وأبي

وإلى الداعم الثابت والمحفز الكبير لي، إخوتي، وأخص بالذكر أخي الكبير
الذي شجعني ووفّر لي جميع المتطلبات. أشكركم جميعًا جزيل الشكر.

كما أشكر صديقاتي على دعمهن في مشواري الجامعي، وخاصة من رافقني في هذه الرحلة

وأشكر كل من ساهم وقدم لي المساعدة من الأهل والأصدقاء في مشواري الجامعي

- ***Dounia*** -

Acknowledgement

First and foremost, we would like to express our deepest gratitude to God Almighty for guiding us through this journey and granting us the strength and perseverance to see it through. This accomplishment would not have been possible without unwavering faith and the blessings received.

From the bottom of our hearts, we extend our sincerest thanks to our parents for their unwavering support, encouragement, and unwavering belief in us throughout our academic endeavors. Their endless love and faith in our abilities have been a constant source of inspiration and motivation.

We are immensely grateful to the esteemed supervisor, **Dr. Farouq Zitouni**, for their invaluable guidance, support, and mentorship throughout this research project. Dr. Zitouni has provided us with the necessary tools and knowledge to navigate the complexities of this research. We are truly fortunate to have had the opportunity to learn from such a dedicated and knowledgeable mentor.

Our deepest appreciation extends to the esteemed members of the jury. President of the Jury, and Examiner Thank you for taking the time to examine our work and providing valuable feedback. It is an honor to have our research assessed by such distinguished scholars.

Furthermore, we want to express our deepest gratitude to **Rihab Lakbichi**, Dr. Zitouni's co supervisor. Her contributions to our work have been invaluable. From the very beginning of this journey until the very end, Rihab has been a constant source of support and guidance. She has consistently offered us valuable advice, on research methods, overcoming challenges... Her willingness to go the extra mile, such as staying late to answer questions or helping us troubleshoot technical issues, has significantly contributed to the successful completion of this project. We are incredibly grateful for her dedication and unwavering support.

We would also like to express our sincere gratitude to all the professors in the Information Technology Department at the University of Kasdi Merbah Ouargla. Their dedication to teaching, fostering academic growth, and sharing their knowledge has significantly shaped our understanding and skills within the field. To each and every one of you, thank you for being a part of our academic journey and for the lasting impact you have made on us.

We extend our heartfelt appreciation to all the family members and friends, both near and far. Their unwavering support has been a source of strength and motivation throughout our academic journey. To each and every one of you, thank you for believing in us and offering your encouragement in whatever way you could.

This journey has been a testament to the power of collaboration, support, and perseverance. We are incredibly fortunate to have had such a remarkable network of individuals supporting us along the way. To each and every one of you, thank you.

Abstract

Solar energy continues to be the most promising renewable energy source for generating electricity due to its abundance on Earth's surface and non-polluting nature. Photovoltaic (PV) models used to generate electricity from solar energy are among the most popular renewable energy systems. To obtain maximum efficiency from PV models, their parameters should be estimated in an optimal way. Metaheuristic algorithms have emerged as promising alternatives, offering robustness and efficiency in solving complex optimization problems. However, the effectiveness of these algorithms can vary significantly, necessitating a comprehensive comparative analysis to identify the most suitable methods for PV parameter estimation. Therefore, we present a comparative study of metaheuristic algorithms for parameter estimation in two PV modules, namely the single diode model and the double diode model. Specifically, the analysis contrasts the performance of Red-tailed Hawk Optimizer (RTH), One-to-One based Optimizer (OOBO), Dung Beetle Optimizer (DBO), and Sand Cat Swarm Optimization (SCSO). The root mean square error (RMSE) value between the simulated data and the actual data is used as the objective function.Accordingly, it is observed that the best estimation accuracy is reached by the RTH algorithm with a value of 9.86E-04 for both models. Consequently, the present paper reports that RTH is an efficient and powerful method for parameter extraction in solar photovoltaic models.

Keywords: parameter estimation, optimization, metaheuristic algorithms, single diode mode, double diode mode, red-tailed hawk optimizer, one-to-one based optimizer, sand cat swarm optimization, and dung beetle optimizer.

Résumé

L'énergie solaire reste la source d'énergie renouvelable la plus prometteuse pour la production d'électricité en raison de son abondance à la surface de la Terre et de sa nature non polluante. Les modèles photovoltaïques (PV) utilisés pour produire de l'électricité à partir de l'énergie solaire sont parmi les systèmes d'énergie renouvelable les plus populaires. Pour obtenir une efficacité maximale des modèles photovoltaïques, leurs paramètres doivent être estimés de manière optimale. Les algorithmes métaheuristiques sont apparus comme des alternatives prometteuses, offrant robustesse et efficacité dans la résolution de problèmes d'optimisation complexes. Cependant, l'efficacité de ces algorithmes peut varier de manière significative, ce qui nécessite une analyse comparative complète afin d'identifier les méthodes les plus appropriées pour l'estimation des paramètres photovoltaïques. Nous présentons donc une étude comparative des algorithmes métaheuristiques pour l'estimation des paramètres de deux modules photovoltaïques, à savoir le modèle à diode unique et le modèle à double diode. Plus précisément, l'analyse compare les performances de l'optimiseur de la buse à queue rousse (RTH), de l'optimiseur basé sur la méthode One-to-One (OOBO), de l'optimiseur du bousier (DBO) et de l'optimisation de l'essaim de chats des sables (SCSO). La valeur de l'erreur quadratique moyenne (RMSE) entre les données simulées et les données réelles est utilisée comme fonction objective. En conséquence, il est observé que la meilleure précision d'estimation est atteinte par l'algorithme RTH avec une valeur de 9.86E-04 pour les deux modèles. Par conséquent, le présent document indique que l'algorithme RTH est une méthode efficace et puissante pour l'extraction des paramètres dans les modèles solaires photovoltaïques.

Mots clés: estimation des paramètres, optimisation, algorithmes métaheuristiques, mode à diode unique, mode à double diode, optimiseur de buse à queue rousse, optimiseur basé sur le principe du un pour un, optimisation de l'essaim de chats des sables et optimiseur de bousier.

ملخص

لا تزال الطاقة الشمسية هي أكثر مصادر الطاقة المتجددة الواعدة لتوليد الكهرباء بسبب وفرتها على سطح الأرض وطبيعتها غير الملوثة. وتعد النماذج الكهروضوئية المستخدمة لتوليد الكهرباء من الطاقة الشمسية من بين أكثر أنظمة الطاقة المتجددة شيوعًا. وللحصول على أقصى قدر من الكفاءة من النماذج الكهروضوئية، يجب تقدير معلماتها بالطريقة المثلى. وقد برزت خوارزميات الميتاهيوريستك كبدائل واعدة، حيث توفر المتانة والكفاءة في حل مشاكل التحسين المعقدة. ومع ذلك، يمكن أن تتفاوت فعالية هذه الخوارزميات بشكل كبير، مما يستلزم إجراء تحليل مقارن شامل لتحديد أنسب الطرق لتقدير المعلمات الكهروضوئية. ولذلك، نقدم دراسة مقارنة لخوارزميات الميتاهيوريستك لتقدير المعلمات في وحدتين كهروضوئيتين، وهما نموذج الصمام الأحادي ونموذج الصمام المزدوج. وعلى وجه التحديد، يقارن التحليل بين أداء مُحسِّن الصقر ذو الذيل الأحمر (RTH)، والمُحسِّن القائم على التوازي (OOBO)، ومُحسِّن خنفساء الروث (DBO)، ومُحسِّن سرب قطط الرمال (SCSO). يتم استخدام قيمة جذر متوسط الخطأ التربيعي (RMSE) بين البيانات المحاكاة والبيانات الفعلية كدالة هدف، وبناءً على ذلك، لوحظ أن أفضل دقة تقدير تم الوصول إليها بواسطة خوارزمية RTH بقيمة 9.86E-04 لكلا النموذجين. وبالتالي، تشير هذه الورقة البحثية إلى أن خوارزمية RTH هي طريقة فعالة وقوية لاستخراج المعلمات في النماذج الشمسية الكهروضوئية.

الكلمات المفتاحية: تقدير المعلمات، التحسين، خوارزميات الميتاهيوريستك، نموذج الصمام الأحادي، نموذج الصمام المزدوج، مُحسِّن الصقر ذو الذيل الأحمر، مُحسِّن قائم على التوازي، مُحسِّن خنفساء الروث، مُحسِّن سرب قطط الرمال.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

P	*Polynomial Time*
NP	*Nondeterministic Polynomial Time*
LP	*Linear Programming*
NLP	*Nonlinear Programming*
PV	*Photovoltaic*
OOBO	*One-to-One-Based Optimizer*
SCSO	*Sand Sat Swarm Optimization*
RTH	*Red-Tailed Hawk*
DBO	*Dung Beetle Optimizer*
SCs	*Solar Cells*
SDM	*Single-Diode Model*
DDM	*Double-Diode Model*
PVM	*Photovoltaic module Model*
TDM	*Triple-Diode Model*
I-V	*Current-Voltage*
P-V	*Power-Voltage*
RMSE	*Root Mean Squar Error*

General Introduction

context

The global shift to renewable energy has accelerated research and innovation in photovoltaic (PV) systems, which use solar energy to generate electricity. The necessity for accurate modeling and optimization to ensure maximum efficiency is growing along with the deployment of PV systems. PV system performance is largely dependent on accurate parameter estimation, which is critical for designing, managing, and predicting PV module behavior. The nonlinear and complex nature of PV models presents challenges for traditional parameter estimation methods, which has led to the exploration of modern computational methods.

Metaheuristic algorithms, inspired by natural processes and behaviors, offer flexibility and adaptability in solving optimization problems. They have been successfully applied in various fields, including engineering, economics, and artificial intelligence, due to their robustness and efficiency. In the context of PV systems, metaheuristic algorithms can significantly enhance the accuracy of parameter estimation, thereby improving the overall efficiency and reliability of PV installations. However, given the diversity of metaheuristic approaches, it is crucial to perform a comparative analysis to determine which algorithms are most effective for specific PV model parameter estimation tasks. This research aims to fill this gap by systematically evaluating the performance of four distinct metaheuristic algorithms on single-diode and double-diode PV models.

Problematic

Despite advances in PV technology, one of the most critical issues remains the accurate estimate of the parameters that characterize the electrical characteristics of PV cells. Traditional methods, such as curve fitting and analytical approaches, frequently have difficulty identifying the underlying nonlinearity and multiple local minima in the parameter space. This leads to suboptimal performance and reduced efficiency for PV systems. Metaheuristic algorithms have emerged as potential alternatives, providing both robustness and efficiency in tackling complex optimization problems. However, the effectiveness of these algorithms varies significantly, requiring a comprehensive comparative analysis to identify the best algorithms for PV parameter estimation.

Contribution

This thesis contributes to the field of PV parameter estimation by conducting a thorough comparative study of four metaheuristic algorithms: One-to-One based Optimizer (OOBO), Sand Cat swarm optimization (SCSO), Red-tailed hawk Optimizer (RTH), and Dung Beetle Optimizer (DBO). The research evaluates these algorithms in the context of single-diode (SD) and double-diode (DD) PV models, analyzing their performance in terms of accuracy, convergence speed, and robustness.

Structure

This thesis is structured as follows:

In the first chapter”**Introduction to Optimization**”, we provide an overview of the field of optimization. We define optimization problems and discuss their characteristics, with a particular emphasis on constraints handling. Additionally, we present a taxonomy of methods commonly employed to solve optimization problems. To illustrate these methods, we provide examples of algorithms belonging to each class.

In the second chapter ”**Introduction to Metaheuristics**”, we begin by providing the definition of metaheuristics. We then delve into various classifications of solution initializers when utilizing metaheuristics. Additionally, we explore the search behavior exhibited by metaheuristics. We discuss the ”No-Free-Lunch Theorem,” which sheds light on the limitations of optimization algorithms. The chapter also covers popular randomization walks employed in metaheuristics. Finally, we provide an overview of the state of the art in the field of metaheuristics.

The third chapter of the thesis focuses on the practical aspect ”**Parameter Estimation in Photovoltaic Systems.**” In this chapter, we provide a detailed background on the optimization algorithms used in the study: One-to-One-Based Optimizer (OOBO), Sand Cat Swarm Optimization (SCSO), Dung Beetle Optimizer (DBO), and Red-Tailed Hawk Optimizer (RTH). We discuss related work in the context of photovoltaic models and present the problem formulation. The chapter concludes with a comprehensive discussion of the results obtained from applying these algorithms to single-diode and double-diode PV models, highlighting their performance and effectiveness.

Chapter 1

Introduction to optimization

1.1 Introduction

In today's dynamic world,efficiency and resourcefulness are paramount. Across every field of human endeavors, from engineering marvels to commercial strategies,the continuous search for optimal solutions guides our daily actions.Optimization,a sophisticated mathematical framework,is a key component in this pursuit.It enables us to traverse complicated decision-making processes in an organized way,achieving the best results possible with limited resources.

Optimization is mainly concerned with functions,which are mathematical entities that connect certain inputs (variables) to related outputs.The essence of optimization is the deliberate modification of these variables to get the desired outcome.This typically entails either minimizing a function,such as reducing production costs,or maximizing it,such as raising profit margins.However,when constraints are present,the true potential of optimization emerges.These constraints represent real-world limitations, such as physical material restrictions or project financial constraints. Finding the optimum solution within these constraints is a mental challenge that necessitates a combination of analytical precision and creative problem-solving.

Optimization applications cover the entire spectrum of human activity. Optimization algorithms are the invisible hands that guide optimal outcomes,from designing strong yet cost-effective bridges to creating efficient logistics networks.In finance,optimization models drive investment strategies, whereas in healthcare,they guide treatment decisions that improve patient well-being.The widespread use of optimization emphasizes its basic function as a universal language for addressing complicated decision-making problems across multiple fields.

This chapter is divided into four sections.The first section recalls some fundamentals of optimization problems, including the definition and modeling of problems.The second section deals with the classification of optimization problems based on three criteria (objective function,decision variables,and constraints).The third section delves into the complexity classes,and the last section provides an overview of optimization algorithms (exact and approximate).

1.2 Optimization Problems

In today's world,where data has an ubiquitous influence,getting the optimal outcomes possible has become a top priority.Optimization problems,which are a fundamental component of applied mathematics and computer science,provide a strong framework for reaching this goal.They propose a systematic approach to navigating complex decision-making processes,allowing for the extraction of the best results from limited resources.

At their core,these problems imply a systematic attempt to minimize or maximize an objective function by carefully selecting values for decision variables from a permissible set[1].This goal is frequently constrained by a variety of problems,including resource limitations,technology constraints,and regulatory requirements,which add layers of complexity to the optimization effort.

The overall purpose of optimization is to find the best configuration or course of action for balancing competing objectives or constraints, maximizing utility, efficiency, or performance while minimizing costs, risks, or resource use.This concept emphasizes the multidisciplinary nature of optimization problems,which pervade various domains such as operations research, management science, and industrial engineering,and provides a structured way to address multifarious decision-making challenges.

An optimization problem have three major components: decision variables, objective function, and constraints.

1. **Decision variables** are physical quantities controlled by the decision maker and represented by mathematical symbols.For example,the decision variable x_j can represent the number of pounds of product j that a company will produce during some month.Decision variables take on any of a set of possible values.

2. **Objective function** defines the criterion for evaluating the solution.It is a mathematical function of the decision variables that converts a solution into a numerical evaluation of that solution.For example,the objective function may measure the profit or cost that occurs as a function of the amounts of various products produced.The objective function also specifies a direction of optimization,either to maximize or minimize.An optimal solution for the model is the best solution as measured by that criterion.

3. **Constraints(if the problem is constrained)** are a set of functional equalities or inequalities that represent physical, economic, technological, legal, ethical, or other restrictions on what numerical values can be assigned to the decision variables.For example, constraints might ensure that no more input is used than is available.Constraints can be definitional,defining the number of employees at the start of a period $t + 1$ as equal to the number of employees at the start of period t,plus those added during period t minus those leaving the organization during period t.In constrained optimization models we find values for the decision variables that maximize or minimize the objective function and satisfy all constraints.

- **Mathematical Formulation**

In mathematical terms,optimization refers to the process of minimizing or maximizing a function while taking into account constraints on its variables.We use this notation[2]:

- x is the vector of $variables$, also called unknowns or parameters.

- f is the $objective function$, a (scalar) function of x that we want to maximize or minimize.

- c_i are $constraint functions$, which are scalar functions of x that define certain equations and inequalities that the unknown vector x must satisfy.

Using this notation, the optimization problem can be written as follows:

$$min f(x)_{x \in R^n} \qquad \text{subject to} \qquad \begin{array}{ll} c_i(x) = 0, & i \in \epsilon, \\ c_i(x) \geq 0, & i \in \lambda. \end{array}$$

Here λ and ϵ are sets of indices for equality and inequality constraints, respectively.

Figure 1.1 shows the contours of the objective function, that is, the set of points for which $f(x)$ has a constant value. It also illustrates the *feasible region*, which is the set of points satisfying all the constraints (the area between the two constraint boundaries), and the point x^* which is the solution of the problem.Note that the “infeasible side” of the inequality constraints is shaded.

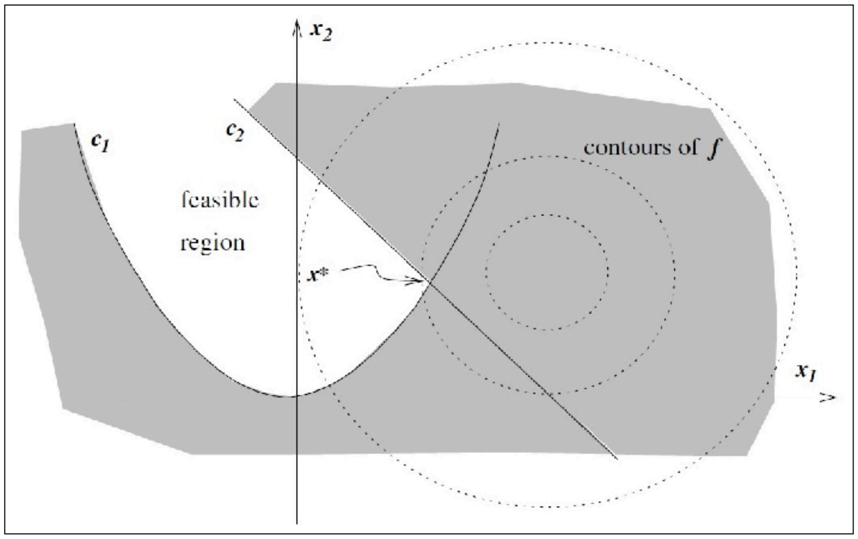


Figure 1.1: Geometrical representation of the problem

1.3 Classification of optimization problems

Some characteristics,including the form of the objective function, the type of decision variables, and the presence of constraints,may be used to classify optimizing problems.This is a classification according to these criteria:

1.3.1 Type of Objective Function

Linear

A linear program is an optimization problem in which the objective function is linear in the unknowns and the constraints consist of linear equalities and linear inequalities[3].In other words,a linear programming problem is concerned with determining the optimal value (maximum or minimum) of a linear function of numerous variables,provided that the variables are non-negative and satisfy a set of linear constraints.Variables,sometimes known as decision variables, are non-negative.

The standard mathematical formulation of a linear programming (LP) problem is as follows:

$$\text{Minimize } c^T x$$

Subject to:

$$Ax \leq b$$

$$x \geq 0$$

Where:

x	:	The vector of decision variables $(x_1, x_2, \cdots, x_n)$.
c	:	The vector of coefficients in the objective function $(c_1, c_2, \cdots, c_n)$.
A	:	Matrix $(m \times n)$ of coefficients representing the constraints.
b	:	An m-dimensional column vector $(b_1, b_2, \cdots, b_m)$.
$x \geq 0$	:	Denotes the non-negativity constraints on the decision variables.

In this formulation:

- The objective is to minimize the linear objective function $c^T x$, where $c^T x$ represents the dot product of the coefficient vector$c$and the decision variable vector$x$.
- The constraints are represented by linear inequalities of the form $Ax \leq b$, where Ax represents the matrix-vector product of A and x.
- The non-negativity constraints $x \geq 0$ ensure that the decision variables are non-negative.

Linear programs can be studied algebraically and geometrically.Both methods are equal,although one or the other may be more convenient for resolving a specific problem related to a linear program.For example,we might ask about the rank of the matrix or for a representation of its null space.It is this algebraic approach that is used in the simplex method[4].

Nonlinear

A nonlinear optimization model (also referred to as a “nonlinear program”) consists of the optimization of a function subject to constraints,where any of the functions may be nonlinear[4].Unlike linear objective functions,which adhere to additive and proportional relationships,nonlinear functions introduce complexities that necessitate specialized optimization techniques for their solution[5].

A general mathematical formulation of a nonlinear programming (NLP) problem can be expressed as follows:

$$\text{Minimize } f(x)$$

Subject to:

$$g_i(x) \leq 0, \qquad i = 1, 2, \cdots, m$$
$$h_j(x) = 0, \qquad j = 1, 2, \cdots, p$$

Where:

x	:	The vector of decision variables $(x_1, x_2, \cdots, x_n)$.
$f(x)$	:	The objective function that need to be minimized or maximized.
$g_i(x)$	:	Inequality constraints, defining m constraints that must be satisfied with $g_i(x) \leq 0$.
$h_j(x)$	:	equality constraints, defining p constraints that must be satisfied with $h_j(x) = 0$.

- The objective is to find the values of the decision variables x that minimize the objective function $f(x)$, subject to the constraints $g_i(x)$ and $h_j(x)$.

Some common methods for solving optimization problems with nonlinear objective functions include:(Gradient-Based Methods,Quadratic Programming,Global Optimization).

1.3.2 Type of Decision Variables

continuous

In continuous optimization,the variables in the model are nominally allowed to take on a continuous range of values, usually real numbers[6].

In mathematical terms, a continuous decision variable x_i can be defined within a continuous domain $[a_i, b_i]$, where a_i and b_i are the lower and upper bounds, respectively.The variable x_i can take on any value within this range,including fractional values.

Continuous optimization problems are normally easier to solve because the smoothness of the functions makes it possible to use objective and constraint information at a particular point x to deduce information about the function's behavior at all points close to x[2].Continuous optimization problems frequently solved using algorithms that generate a series of variable values,known as iterates,that converge to a solution. The algorithm uses knowledge gained from past iterates,as well as information about the model at the current iterate,maybe including information about its sensitivity to perturbations in the variables,to decide how to go from one iteration to the next.

Discrete

Discrete decision variables are variables that can only take on integer values from a predefined set or range.Unlike continuous variables,which can take on any real value within a specified range,discrete variables are restricted to specific,distinct values[7].

In mathematical terms,a discrete decision variable x_i can be defined within a discrete domain $\{a_i, a_i + 1, \cdots, b_i\}$, where a_i and b_i are the lower and upper bounds,respectively.The variable x_i can only take on integer values within this set.

Discrete problems,the behavior of the objective and constraints may change significantly as we move from one feasible point to another,even if the two points are "close" by some measure.The feasible sets for discrete optimization problems can be thought of as exhibiting an extreme form of non convexity,as a convex combination of two feasible points is in general not feasible[2].

Solving optimization problems with discrete decision variables often requires the use of specialized methods such as integer programming, constraint programming,or metaheuristic algorithms such as genetic algorithms or simulated annealing.These methods are intended to manage the combinatorial nature of discrete optimization problems while effectively searching the solution space for the best or near-optimal solution.

1.3.3 Presence of constraints

Many real-life optimization problems have certain constraints/rules that cannot be violated while finding an optimized solution.

Constrained Optimization Problem

A constrained optimization problem involves optimizing an objective function subject to one or more constraints.In mathematical terms,the goal is to minimize or maximize the objective function $f(x)$ while satisfying constraints $g_i(x)$ and $h_j(x)$(could be equality or inequality constraints).They are encountered in numerous applications,spanning various scientific fields such as structural optimization,engineering design,economics,and allocation and location problems[8][9].

Constrained optimization problems arise from models in which constraints play an essential role[2].These constraints establish the limits, restrictions,or conditions that the solution must satisfy.Additionally, Constraints can have a major impact on the optimization problem by restricting the feasible solution space.

The most common approach for solving CO problems is the use of a penalty function.The constrained problem is transformed to an unconstrained one,by penalizing the constraints and building a single objective function,which in turn is minimized using an unconstrained optimization algorithm[10].However,there are other methods can be used such as interior-point methods,sequential quadratic programming (SQP),augmented Lagrangian methods, and penalty methods.

Unconstrained Optimization Problem

An unconstrained optimization problem refers to the task of minimizing or maximizing a function without any restrictions or constraints on the decision variables[11].In mathematical terms,it involves finding the minimum or maximum of an objective function $f(x)$ over the entire feasible space of the decision variables x,without being subject to any constraints.

Unconstrained optimization problems occur directly in many real-world applications. Even for some problems with natural constraints on the variables,it may be safe to disregard them as they do not affect the solution and do not interfere with algorithms[2].

There exist two important classes of iterative methods—line search methods (such as Gradient Descent,Newton's Method, and quasi-Newton Methods) and trust-region methods—made with the aim of solving the unconstrained optimization problem[12].

Unconstrained optimization has a variety of applications, including machine learning and medical imaging, emphasizing its importance in today's optimization challenges. As computer resources develop, unconstrained optimization methods will become increasingly relevant in handling complex optimization problems spanning several domains.

1.4 Complexity Classes

Computational complexity theory divides decision problems into complexity classes based on the computational resources needed to solve them.These classes represent sets of functions that can be computed within specific resource constraints,such as time and memory[13]. In the following, we'll present the various complexity classes.

1.4.1 $\mathcal{P}$ C$lass$

The complexity class $\mathcal{P}$, which is also known as the "Polynomial Time" class,refers to the set of selection problems that can be solved with a deterministic in polynomial time[14].This means that there exists an algorithm capable of solving problems in P with a time complexity bound by a polynomial function relative to the size of the input.

Problems of class $\mathcal{P}$ are regarded manageable and desirable for problem-solving because effective algorithms exist to solve them in a reasonable amount of time, even with large inputs. Sorting, searching, and graph traversal are examples of class $\mathcal{P}$ problems that can be effectively solved by algorithms such as quicksort, binary search, and breadth-first search. The efficient solvability of problems in class $\mathcal{P}$ makes it an important and well-studied complexity class in theoretical computer science.

1.4.2 $\mathcal{NP}$ C$lass$

The complexity class $\mathcal{NP}$, which is also known as the "Nondeterministic Polynomial Time" class,refers to The set of all problems that can be solved by a nondeterministic algorithm in polynomial time[14].

In $\mathcal{NP}$, a given solution may be checked deterministically in polynomial time, implying that once provided, its correctness can be efficiently validated. However, the lack of a proven effective algorithm for finding solutions is what characterizes $\mathcal{NP}$. This class contains many important computing problems, including the traveling salesman problem and the Boolean satisfiability problem.There are also $\mathcal{NP}$-$Hard$ and $\mathcal{NP}$-$Complete$ sets, which we use to express more sophisticated problems. In the case of rating from easy to hard, we might label these as "easy", "medium","hard", and finally "hardest". And we can visualize their relationship, too:

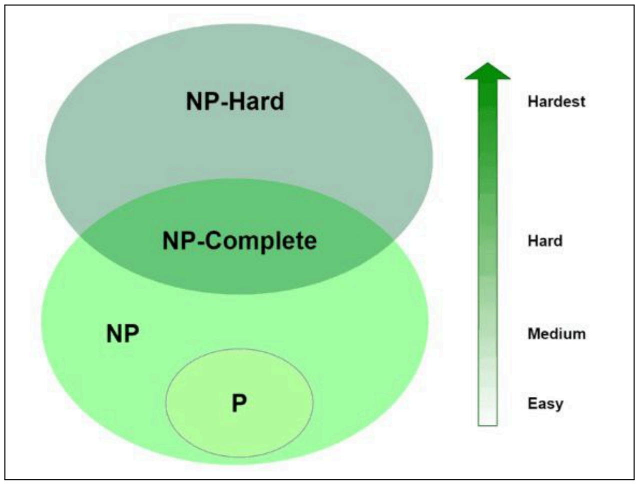


Figure 1.2: Complexity Classes.

The relationship between the complexity classes P and NP is a focal point in computational theory, with the unresolved question of whether P equals NP still one of the most critical open problems in computer science. At its core, this research tries to determine if problems that are efficiently verifiable by nondeterministic algorithms in polynomial time may also be effectively solved by deterministic algorithms in the same time frame.

1.5 Optimization Algorithms

Optimization algorithms are computational methodologies crafted to identify the optimal solution among a set of potential solutions for a given problem. Their primary objective is to minimize or maximize an objective function, all while ensuring adherence to specific constraints[15]. One way to classify optimization algorithms is based on their approach to finding the optimal solution. Two broad categories emerge from this classification:exact algorithms and approximate algorithms.

Mind that whatever the classification of an algorithm is, we have to make the right choice to use it correctly, and sometimes a proper combination of algorithms may achieve better results[16].

1.5.1 Exact Algorithms

The exact algorithm refers to an algorithm that can solve combinatorial optimization problems with a computational complexity that is bounded by a function growing polynomially in the size of the instance to be solved[17]. These algorithms systematically explore the entire feasible solution space to identify the globally optimal solution, ensuring that no better solution exists within the problem's constraints.

In practice,exact algorithms are commonly used in a variety of domains, including operations research, logistics, scheduling, and network design, where finding the globally optimal solution is critical. They give vital insights into the structure of optimization problems and help grasp the fundamental aspects of feasible remedies.

Enumeration

The enumeration technique is the most basic strategy to solve a pure integer optimization problem, involving the enumeration of all finitely various possibilities within a defined problem space. This technique carefully explores all possible solutions to the problem[18]. This technique is particularly useful for relatively small instances of combinatorial problems where it is feasible to enumerate all possible combinations or permutations. Enumeration guarantees that the optimal solution will be found because it exhaustively checks every possible solution[19].

Dynamic Programming

Dynamic programming is a fundamental technique in the field of optimization,it was proposed by Richard Bellman in 1953[20].

Dynamic programming solves a complex problem by dividing it into smaller, more manageable subproblems, solving each one just once, storing the solutions, and then combining them to find the optimal solution to the original problem. This method is frequently used in many fields to solve optimization problems since it computes optimal solutions efficiently.

Due to its ability to solve complex problems, it is widely used in versatile applications across various domains, including graph theory, game theory, AI and machine learning, economics and finance problems, bioinformatics,as well as calculating the shortest path, which is used in GPS. Dynamic programming is also used in laptops and phones to store data through caches so that it does not need to retrieve data from the servers when data is needed. In fact, you may be using dynamic programming without realizing that you are.

Branch and Bound

Branch-and-bound algorithms are an important tool for tackling discrete optimization problems because they provide a systematic technique for efficiently exploring the feasible region of a problem[21]. The algorithm uses a tree search strategy to implicitly enumerate all possible solutions to a given problem,applying pruning rules to eliminate regions of the search space that cannot lead to a better solution[22].Two tools are required to perform the branch-and-bound technique.

1.Branching:

The feasible region can be covered by multiple smaller feasible sub-regions. This can be done recursively for each sub-region, and this leads to sub-regions naturally forming a tree structure known as a branch-and-bound tree or search tree. Its nodes are the constructed sub-regions.

2. Bounding:

This technique quickly determines upper and lower bounds for the optimal solution within a feasible region.

There are three algorithmic components in B&B that can be specified by the user to fine-tune the behavior of the algorithm. These components are the search strategy, the branching strategy, and the pruning rules.each play a distinct role with respect to the search phase and the verification phase.

Overall, branch-and-bound algorithms have been effectively extended to a variety of optimization problems other than integer programming, demonstrating their adaptability and effectiveness in handling complicated combinatorial optimization tasks.

1.5.2 Approximate Algorithms

Approximation algorithms are an essential concepts in computer science and optimization theory.Their goal is to efficiently find near-optimal solutions to computationally challenging problems(commonly problems of class $NP - hard$)[23].They prioritize efficiency and practicality over the guarantee of finding the absolute best solution.

Approximation algorithms typically have two characteristics. First, they offer a feasible solution to a problem in polynomial time. In most circumstances, it is not difficult to devise a procedure that provides a feasible solution. The second characteristic that differentiates approximation algorithms is the need for some assurance of solution quality. The quality of an approximation method is the maximum ”distance” between its solutions and the optimal solutions,evaluated over all possible instances of the problem.

Greedy Algorithms

The greedy algorithms are an important class of computer algorithms with self-reducibility for solving combinatorial optimization problems. They use the greedy strategy in the construction of an optimal solution[20]. Employing a strategy of prioritizing locally optimal choices at each step,they aspire to construct an optimal solution incrementally[24].

It is important to note that while greedy algorithms are efficient and simple, they may not always produce the optimal solution. Despite this limitation, greedy algorithms are widely used in a variety of fields, including scheduling, network routing, and computational biology.Nonetheless,it is a useful tool for tackling many real-world problems due to their intuitive nature and efficiency, although it is necessary to carefully consider the applicability and correctness of these technologies in each specific context.

Among the various versions of greedy algorithms, the pure greedy, orthogonal greedy, relaxed greedy, and stepwise projection algorithms stand out as the most commonly utilized, denoted by the acronyms PGA, OGA, RGA, and SPA, respectively[25].

Heuristic algorithms

Heuristic algorithms are a type of problem-solving algorithm that finds efficient solutions to complex problems. They are frequently employed when there is no known algorithm capable of providing an exact solution or when using an exact approach is extremely computationally expensive[26].Typically, it operates in 5 phases, including problem formulation, solution representation, heuristic selection, search, and optimization, followed by solution evaluation and refinement.

Heuristic algorithms provide several advantages, including scalability, adaptation to real-world constraints, and the capacity to handle huge, complicated problem sets. They are typically faster than exact algorithms and can produce acceptable results in a reasonable amount of time. Furthermore, heuristic algorithms are adaptive and may be customized to specific problem domains, making them effective problem-solving tools.

Metaheuristic Algorithms

Metaheuristics are general-purpose algorithms that can be applied to solve almost any optimization problem. They may be viewed as upper-level general methodologies that can be used as a guiding strategy in designing underlying heuristics to solve specific optimization problems[27]. While computationally more difficult, they excel at solving complicated, nonlinear optimization problems with various constraints[28].Metaheuristics have grown in popularity over the last two decades due to their efficiency and efficacy in dealing with large-scale and complex problems in a variety of applications. This popularity emphasizes their importance as effective tools for navigating the intricacies of current optimization environments. The next chapter will provide further insights into various metaheuristics.

1.6 Conclusion

In conclusion,optimization is a fundamental concept with several applications. Optimizing everything is the objective of nearly every technical and industrial application, be it maximizing profit, production, performance, or efficiency, or minimizing cost and energy consumption. In this chapter,we have laid the groundwork for understanding optimization by discussing key concepts and classifications based on various characteristics.Additionally, we briefly discussed complexity classes, distinguishing between P and NP class problems. Finally, we introduced optimization algorithms and classified them as exact and approximate approaches.

Afterwards, exploring optimization algorithms,including both exact and approximate approaches, provides us with the tools we need to effectively navigate and solve optimization challenges. Overall,a comprehensive understanding of optimization problems is required for developing efficient solutions to solve a variety of real-world challenges and optimize outcomes.

Chapter 2

Introduction to Metaheuristics

2.1 Introduction

Real-world issues are often becoming more complicated, to the point that standard mathematical programming techniques are unable to adequately solve and optimize them. In real-life optimization, most problems are nonlinear, complex, multimodal, and have incompatible objective functions. Finding an optimal solution, or even one that is close to optimal, can be a very challenging task, even for simple, linear objective functions. In general, there is no guarantee of finding an optimal solution for real-life problems. The metaheuristic algorithms' usefulness is shown here. It is possible to address challenging optimization problems with a variety of metaheuristic optimization methods that are currently being researched.

This chapter focuses on metaheuristics, a flexible approach to problem-solving. It delves into the classification of metaheuristics, exploring their search strategies and introducing the ”No-Free-Lunch Theorem,” which underscores the constraints of individual metaheuristic methods. Additionally, it exposes typical randomization techniques utilized in metaheuristics and offers a comprehensive summary of the latest state of the art in the field. Finally, the chapter concludes by reviewing the key characteristics of metaheuristic algorithms.

2.2 Metaheuristics

In the realm of optimization, where the goal is to find the most favorable solution from a set of alternatives, traditional methods can become computationally intractable for complex problems with vast search spaces. Here, metaheuristics emerge as a powerful and versatile class of optimization algorithms that excel at tackling such challenges[29]. Unlike exact methods that guarantee finding the optimal solution, metaheuristics operate under the principle of ”good enough” solutions, aiming to identify high-quality solutions within a reasonable time frame. This characteristic makes them particularly well-suited for real-world optimization problems with practical constraints[30].

Metaheuristics are inspired by many natural phenomena and human problem-solving methodologies.Some well-known examples include genetic algorithms, which mimic the process of natural selection to evolve solutions; simulated annealing, which draws on the principles of physical annealing in materials science; and swarm intelligence algorithms, which are inspired by the collective behavior of social insects such as ant colonies or bird flocks[31].Metaheuristics iteratively explore the search space by incorporating bio-inspired or human-inspired heuristics, navigating to promising areas while avoiding getting trapped in local optima.This stochastic (randomized) nature enables them to escape from plateaus and potentially discover globally optimal solutions, even for extremely nonlinear and discontinuous problems.

While metaheuristics provide a compelling alternative to standard methods for hard optimization tasks, it is critical to recognize the inherent trade-offs. The quality of the solution is determined by factors such as the algorithm used, its parameter settings, and the specific problem being solved.Furthermore, unlike exact algorithms, metaheuristics rarely guarantee optimality. However, their flexibility, computational speed, and capacity to handle large and complex search spaces often make them the best choice for practical optimization problems when obtaining a ”good enough” solution in a reasonable time frame is critical.

2.3 Classification of metaheuristics

Metaheuristic algorithms can be classified in several ways based on different criteria. Here are some common classifications:

2.3.1 Nature-inspired vs. non-nature-inspired

Metaheuristics are classified based on their inspiration from natural processes or not. Nature-inspired metaheuristics include genetic algorithms, particle swarm optimization, and ant colony optimization. Non-nature-inspired metaheuristics include simulated annealing and tabu search.

2.3.2 Deterministic vs. stochastic

Deterministic metaheuristics produce and assess candidate solutions using a deterministic process that usually involves methodical solution space search. Hill climbing and tabu searching are two examples.Randomness is introduced into the search process via stochastic metaheuristics, which frequently employ probabilistic techniques for the generation, evaluation, or selection of solutions. Genetic algorithms and simulated annealing are two such examples.

2.3.3 Trajectory-based vs. population-based

Metaheuristics can be classified based on whether they aim to identify a single or several solutions. Trajectory-based metaheuristics aim to find a single solution and refine it iteratively. Examples include hill climbing and simulated annealing. Population-based metaheuristics utilize a set of solutions to explore the search space. Examples include genetic algorithms and particle-swarm optimization.

2.3.4 Local search-based vs. global search-based

Local search metaheuristics aim to improve a single solution by making incremental modifications to it while examining the nearby surroundings of the existing solution. Examples include hill climbing and gradient decrease.Global search metaheuristics search a larger area of the solution space for the global optimum, frequently foregoing efficiency in favor of the capacity to avoid local optimums. Examples include genetic algorithms and simulated annealing.

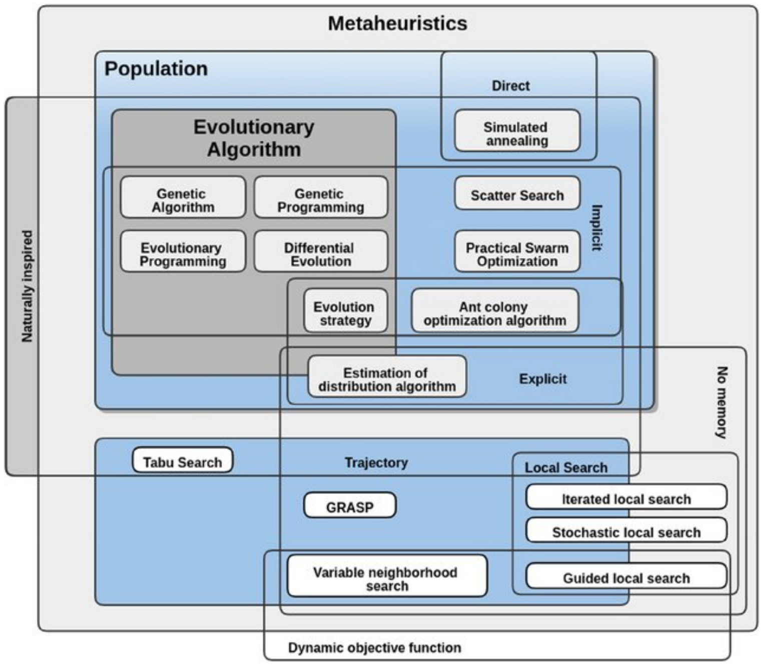


Figure 2.1: Generalized classifications of metaheuristics search techniques (De Castro and Von Zuben 2000).

2.4 Search strategy

The search strategy of metaheuristic algorithms involves iteratively navigating the trade-off between exploration and exploitation. This delicate balance allows metaheuristic algorithms to effectively explore complex solution spaces and find high-quality solutions to optimization problems. Let's delve into each of these components:

2.4.1 Exploration

Exploration is the process of looking for new and diverse solutions in the solution space, which is commonly accomplished by introducing randomness or perturbations to existing solutions. It entails exploring previously unknown areas in order to identify potentially better answers that may be situated far from the present search trajectory.

2.4.2 Exploitation

Exploitation is the use of local knowledge in the search.And solutions have been found so far so that new search moves concentrate on the local regions or neighborhoods where The optimality may be close; however, this local optimum may not be the global optimality[32]. Exploitation often employs significant local information, such as gradients, the shape of the mode, such as convexity, and the history of the search process. A typical strategy is ”hill-climbing,” which makes extensive use of local slopes or derivatives.

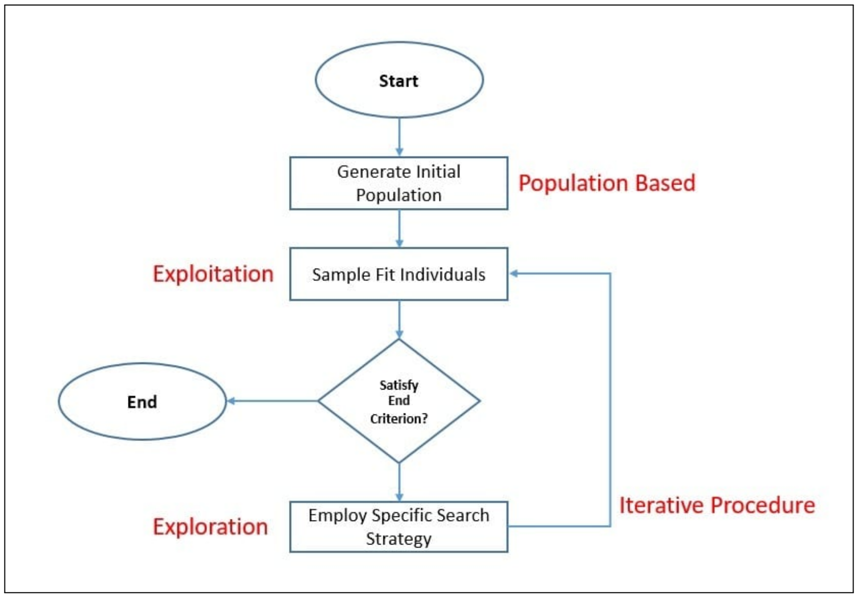


Figure 2.2: Exploitation and Exploration flowchart.

To identify the best solution effectively, an algorithm must be capable of both exploration and exploitation. If the exploration ability is weak, the algorithm may struggle to find the region holding the optimal solution, possibly becoming trapped in areas with local optima. As a result, the exploitation capabilities would be limited to the locations found during exploration. As a result, prioritizing a strong exploration capacity is critical for any algorithm before assessing its exploitation potential. Figure 2.2 illustrates the exploitation and exploration flowchart.

2.5 Random Walks

By analyzing the main characteristics of metaheuristic algorithms, we know that randomization plays an important role in both exploration and exploitation.

randomization prevents solutions from getting trapped in local optima while also increasing solution variety. A solid combination of best selection and randomization guarantees that global optimality is achieved. and one of the most effective methods of randomization is using a random walk (RW).

A random walk is a random process that consists of taking a series of consecutive random steps[33]. Let S_N denote the sum of each consecutive random step X_i, then S_N forms a random walk.

$$S_N = \sum_{i=1}^{N} X_i = X_1 + \cdots + X_N = \sum_{i=1}^{N-1} X_i + X_N = S_{N-1} + X_N \tag{2.1}$$

where X_i is a random step drawn from a random distribution. This relationship can also be considered as a recursive formula. That is, the next state SN will only depend on the current existing state S_{N-1} and the motion or transition X_N from the existing state to the next state.

As RWs are widely used for randomization and local search, a proper step size is very important.The step size in RW determines how much distance a random walker can travel for a given number of iterations[34].

2.5.1 Lévy flight

Lévy flights (LFs) are a class of non-Gaussian random processes whose stationary increments are distributed according to a Lévy stable distribution [33]. Additionally, various studies have shown that the flight behavior of many animals and insects has demonstrated the typical characteristics of Lévy flights. Subsequently, such behavior has been applied to optimization and optimal search, and preliminary results show its promising capability [35].

In general, the Lévy distribution is stable and can be defined in terms of a characteristic function or the following Fourier transform:

$$F(k) = \exp\left[-\alpha|k|^{\beta}\right], \qquad 0 < \beta \leq 2 \tag{2.2}$$

where α is a scale parameter.

2.5.2 Gaussian Walk

Gaussian random walks are a search method for optimization problems. They work by making small, random changes to a prospective solution, similar to taking an uncertain walk. These random steps allow the algorithm to explore different sections of the search space in the same way that a walker could discover new pathways. This randomness is critical because it helps the algorithm avoid dead ends (local optima) and maybe uncover even better solutions.

2.6 No-Free-Lunch Theorem

There are the so-called 'No free lunch theorems', which can have significant implications in the field of optimization (Wolpert and Macready, 1997). One of the theorems states that if algorithm A outperforms than algorithm B for some optimization functions, then B will be superior to A for other functions[36].In other words, no one-size-fits-all solution exists. An algorithm that works brilliantly for one sort of problem may not work as well for another. It's similar to having several tools in a toolbox; each one is valuable for a certain purpose, but none of them can do everything properly.

2.7 Applications of metaheuristics algorithms

Metaheuristic algorithms have been utilized to tackle a wide range of optimization problems across industries such as engineering, finance, logistics, and healthcare.Some widely used applications include:

- **Engineering**

Metaheuristics are extremely useful in optimizing complicated systems such as manufacturing processes and production schedules. They help find designs that fit numerous constraints while improving performance. For example, in structural design, methods such as genetic algorithms and particle swarm optimization are used to modify the shape and material of structures with the goal of reducing weight and increasing efficiency. In manufacturing, simulated annealing is used to enhance CNC machine cutting settings, while ant colony optimization is used to optimize production line architecture.

- **Data Mining and Machine Learning**

Data Mining and Machine Learning: Metaheuristics are employed for feature selection, parameter tuning, and model optimization in machine learning and data mining tasks. such as neural networks and support vector machines.For example, particle-swarm optimization can be used to optimize the weights and biases of a neural network for image classification. It also improves the performance of predictive models and enhances decision-making processes.

- **Logistics**

Metaheuristics are extremely useful in optimizing complex systems such as manufacturing processes and production schedules. They help to find designs that improve performance while satisfying different constraints. For example, in structural design, techniques such as particle swarm optimization and genetic algorithms are applied to optimize the material and shape of structures in order to reduce weight and maximize efficiency.

- **Healthcare**

Metaheuristic algorithms have shown effectiveness in healthcare domains including diagnosis, treatment planning, drug improvement, and management. Ant colony optimization has been used to optimize radiation therapy for cancer patients, reducing side effects and improving results.Genetic algorithms can detect diabetes using patient data, including age, weight, and blood glucose levels.

- **Energy Systems**

Metaheuristic algorithms play a crucial role in enhancing the efficiency of power generation systems, including thermal power plants, hydroelectric dams, wind farms, and solar plants. Their utilization aids in optimizing energy production by considering factors like fuel expenses, equipment limitations, and environmental compliance measures, ultimately maximizing output. In the next chapter, we'll delve deeper into a specific application of this concept—parameter estimation in photovoltaic (PV) systems—to illustrate how metaheuristics can be harnessed to optimize the performance of solar power plants.

2.8 Conclusion

In conclusion, the history of metaheuristic algorithms spans several decades and includes the development of numerous optimization algorithms inspired by natural systems. Metaheuristic algorithms have proven to be an effective tool in solving complex optimization problems in a variety of sectors, and they are expected to play an increasingly important role in the development of new technologies and applications.Metaheuristic algorithms have been used to solve a wide range of scientific, engineering, logistics, healthcare, and energy systems challenges, demonstrating their usefulness in identifying near-optimal solutions.

In this study, we present a contrastive analysis between the performance of four metaheuristic algorithms for parameter estimation in photovoltaic models. Algorithms are compared on the basis of the objective function and type of diode model used.

Chapter 3

Parameter Estimation in Photovoltaic Systems

3.1 Introduction

Solar energy has emerged as a viable renewable resource with the ability to meet global energy needs while transitioning to a more sustainable future. Photovoltaic (PV) systems, which convert sunlight directly into electricity, are crucial for utilizing this clean energy source. Accurate estimation of important parameters is critical for ensuring maximum performance and efficient power generation in PV systems. These parameters influence the system's electrical properties and its capacity to convert solar radiation into electricity.

Traditional parameter estimation methods for PV systems (Method of Moments (MoM), Ordinary Least Squares, Maximum Likelihood Estimation (MLE), ...) frequently use single- or double-diode models. Although these methods provide a fundamental framework, they are not without limits. In practice, issues such as non-ideal operating circumstances, environmental fluctuations, and aging effects might create complications that traditional methods fail to capture. This can lead to erroneous parameter estimates, limiting the ability to enhance PV system performance.

To solve these limitations and obtain a more robust parameter estimate, researchers explored the use of metaheuristic algorithms. These powerful optimization methods are based on natural phenomena or human problem-solving strategies. They excel at dealing with complex problems involving nonlinear relationships, making them well-suited to the challenges of estimating PV system parameters.

To assess the efficacy of various metaheuristic algorithms in estimating PV system parameters, an extensive literature review was undertaken. The findings will be presented in table (3.1),providing a structured analysis of existing research. This table will compare different algorithms, including: hybrid kepler optimization algorithm (HKOA), generalized normal distribution optimization (GNDO), hybrid gazelle-Nelder-Mead (GOANM), opposition-based exponential distribution optimizer (OBEDO), hyprid of single candidate optimizer and the chaotic sand cat optimizer (CSCSC), adaptive sine-cosine particle swarm optimization algorithm (ASCA-PSO), Improved Cheetah Optimizer (ICO), Growth Optimization (GO), enhanced sine-cosine algorithm (ESCA), and Mountain Gazelle Optimizer (MGO). For each algorithm, the table will detail the specific methods employed and, most importantly, the achieved Root Mean Square Error (RMSE) for both the single-diode (SD) and double-diode (DD) models. This emphasis on RMSE values will allow for a direct comparison of the accuracy achieved in estimating parameters for these two prevalent PV system models. This systematic organization will facilitate the identification of trends and strengths within various metaheuristic approaches for PV parameter estimation.

Building upon the comprehensive review of metaheuristic algorithms presented in the table, this chapter delves into a comparative analysis of four prominent algorithms for parameter estimation in photovoltaic (PV) systems. Our objective is to identify the most effective approach for achieving accurate and efficient parameter estimation within this context. Through an analytical contrastive study, we will evaluate the performance of each algorithm, considering factors like accuracy. This in-depth analysis will provide valuable insights into the strengths and weaknesses of each algorithm, ultimately guiding the selection of the most suitable approach for specific PV system parameter estimation tasks.

Table 3.1: Some metaheuristic algorithms proposed recently for the parameter estimation of PV models

Year	Algorithm	Used Method	RMSE (SDM)	RMSE (DDM)	References
2024	HKOA	Integrating the Kepler optimization algori thm (KOA) with the ranking-based updat-e and exploitation improvement mechan-isms.	$7.73006E-04$	$7.326E-04$	[37]
2024	GNDO	metaheuristic optimization algorithm that utilizes a simple structure and the traditi-onal generalized normal distribution for-mula to update the individual's locations based on population location information	$9.8602E-04$	$9.8248E-04$	[38]
2024	GOANM	combines the Gazelle Optimization Algo-rithm (GOA) with the Nelder-Mead (NM) algorithm.	$7.7299E-04$	$7.4339E-04$	[39]
2024	OBEDO	combines the principles of opposition-based learning (OBL) with the expone-ntial distribution optimizer	$9.8602E-04$	$9.8250E-04$	[40]
2024	CSCSC	hybrid approach that combines the Sin-gle Candidate Optimizer (SCO) with the Chaotic Sand Cat Optimizer (CSCO)	$9.86021E-04$	$9.82508E-04$	[41]
2024	ASCA-PSO	combines the strengths of the Sine–cosine algorithm and Particle swarm optimization algorithm in a two-tier process	$9.89E-04$	$9.97E-04$	[42]
2023	ICO	Based on improving the cheetah optimizer	$9.860218E-04$	$9.824860E-04$	[43]
2023	GO	developed and simulated from humans lea-rning and reflection capacities in social gr-owing activities	$9.8602E-04$	$9.8602E-04$	[44]
2023	ESCA	introduces the concept of population aver-age position to increase the population ex-ploration ability, and at the same time int-roduces the personal destination agent m-utation mechanism and competitive sele-ction mechanisminto SCA to provide more search directions for ESCA while ensuring the search accuracy and diver-sity maintenance	$9.860218E-04$	$9.824848E-04$	[45]
2023	MGO	The method used is inspired by the social life and hierarchy of mountain gazelles in the wild.	$2.042717E-03$	$1.387641E-03$	[46]

3.2 Background

3.2.1 One-to-One-Based Optimizer

The One-to-One-Based Optimizer (OOBO) is a new optimization technique for solving optimization problems in various scientific areas[47].Unlike other algorithms, OOBO does not rely on specific members of the population for updates but instead effectively utilizes the knowledge of all population members in the process of updating. This allows for a more accurate scanning of the problem's search space, preventing convergence towards local optimal areas. The algorithm operates in an iteration-based process, using the power of population search to find effective solutions to optimization problems. Additionally, OOBO does not have any control parameters, making it a convenient and efficient approach for solving optimization problems.The One-to-One Based Optimizer (OOBO) algorithm consists of the following steps:

Initialization

In OOBO, each population member is mathematically represented by a vector with the same number of elements as the number of decision variables. A population member can be represented using:

$$\vec{X}_i = [x_{i,1}, \ldots, x_{i,d}, \ldots, x_{i,m}], \quad i = 1, \ldots, N. \tag{3.1}$$

To generate the initial population of OOBO, population members are randomly positioned in the search space utilizing:

$$x_{i,d} = lb_d + \text{rand}() \cdot (ub_d - lb_d), \quad d = 1, \ldots, m, \tag{3.2}$$

Where

$\vec{X}_i$	:	The ith population member (that is, the proposed solution).
$x_{i,d}$	:	Position of the ith population member in the dth dimension.
rand	:	A function generating a random number from [0, 1].
N	:	The size of the population.
ub_d	:	The Upper bound for the position in the dth dimension.
lb_d	:	The Lower bound for the position in the dth dimension.

In OOBO, the algorithm population is represented using a matrix according to

$$\vec{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_i \\ \vdots \\ \vec{X}_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times M} \tag{3.3}$$

The optimization problem's objective function can be assessed based on each population member, which is a proposed solution. Thus, different values for the objective function are acquired in each iteration equal to the number of population members, which can be mathematically described by:

$$\vec{F} = \begin{bmatrix} f_1 \\ \vdots \\ f_i \\ \vdots \\ f_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} f(\vec{X}_1) \\ \vdots \\ f(\vec{X}_i) \\ \vdots \\ f(\vec{X}_N) \end{bmatrix}_{N\times 1} \tag{3.4}$$

where $\vec{F}$ is the objective function vector and f_i is the objective function value for the ith proposed solution.

Mathematical Modeling

OOBO utilizes a one-to-one correspondence model for updating population members' positions. Each member is selected only once and randomly to guide a different member in the search space.To model a one-to-one correspondence, the member position number in the population matrix is used. The random process of forming the set $\vec{k}$ as "the set of the positions of guiding members" is modeled by

$$\vec{K} = \{[k_1, \cdots, k_\updownarrow, \cdots, k_N] \in \mathcal{P}_{\overline{N}} \; ; \; \forall \updownarrow \; \in \overline{N} : k_\updownarrow \neq \updownarrow\} \tag{3.5}$$

where $\overline{N} = \{1, \cdots, N\}$, $\mathcal{P}_{\overline{N}}$ is the set of all permutations of the set $\overline{N}$, and $k_\updownarrow$ is the $\updownarrow$th element of the vector $\vec{K}$.

OOBO guide the population towards better solutions in the search space.It works by:

- Comparing members: Selecting two members (X_i and X_{ki}) and comparing their "fitness" based on the objective function.
- Moving towards better solutions: If X_{ki} is "better" (higher objective function value), X_i is moved closer to it in the search space. This encourages finding solutions similar to successful ones.
- Exploring the search space: If X_i is better, it moves away from X_{ki}, exploring different areas to potentially find even better solutions.

The process of calculating the new positions of population members and updating them in the search space is modeled by Equations (3.6) and (3.8).

$$x_{i,d}^{\text{new}} = \begin{cases} x_{i,d} + \text{rand}() \cdot (x_{k_i,d} - I x_{i,d}), f_{k_i} < f_i; \\ x_{i,d} + \text{rand}() \cdot (x_{i,d} - x_{k_i,d}), \text{ otherwise }, \end{cases} \tag{3.6}$$

$$I = \text{rand}(1 + \text{rand}()). \tag{3.7}$$

$$X_i = \begin{cases} X_i^{\text{new}}, f_i^{\text{new}} < f_i; \\ X_i, \text{ otherwise }, \end{cases} \tag{3.8}$$

Where

$x_{i,d}^{\text{new}}$	:	The new suggested status of the ith member in the dth dimension.
$x_{k_i,d}$	:	The dth dimension of the selected member to guide the ith member.
f_{k_i}	:	The objective function value obtained based on $X_{k,i}$.
I	:	Variable takes values from the set {1, 2}.
$X_{i,d}^{\text{new}}$	:	The new suggested status in the search space for the ith population member.
f_i^{new}	:	The value of the objective function of the ith population member.

Computational Complexity and Pseudocode of OOBO

The computational complexity of the One-to-One-Based Optimizer (OOBO) algorithm can be analyzed in terms of time complexity and space complexity.

- The total time complexity is $O(NTm)$, where N is the number of population members, T is the number of iterations, and m is the number of decision variables.

- The space complexity of the OOBO algorithm is $O(Nm)$, where N is the number of population members and m is the number of decision variables.

Algorithm 1 . Pseudocode of OOBO

```
Start OOBO
Input optimization problem information
Set $N$ and $T$
Create an initial population matrix
Evaluate the objective function
for $t \leftarrow 1$ To $T$ do
    Update $\vec{K}$ based on Equation (3.5)
    for $i \leftarrow 1$ To $N$ do
        Calculate $X_i^{\text{new}}$ based on Equations (3.6) and (3.7)
        Compute $f_i^{\text{new}}$ based on $X_i^{\text{new}}$
        Update $X_i$ using Equation (3.8)
    end for
    Save the best solution found so far
end for
Output the best quasi-optimal solution
End OOBO
```

3.2.2 Sand Cat swarm optimization

Sand cat swarm optimization (SCSO) is a new nature-inspired algorithm that mimics the sand cat behavior that tries to survive in nature[48]. The algorithm is designed to balance exploration and exploitation phases, and it has been shown to perform well in finding solutions with fewer parameters and operations. Inspired by the sand cat's ability to detect low frequencies below 2 kHz and its remarkable hunting skills, the SCSO algorithm consists of two main phases: search and attack. It is designed to control the transitions between these phases in a balanced manner to effectively find optimal solutions.

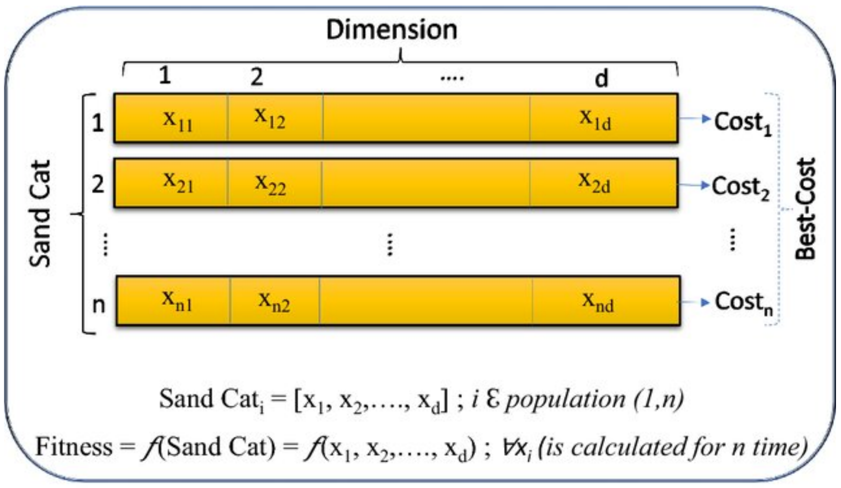

Dimension
1
2
....
d
Sand Cat
1
2
n
x11
x12
x1d
x21
x22
x2d
xn1
xn2
xnd
Cost1
Cost2
Costn
Best-Cost
Sand Cat_i = [x1, x2,…, xd] ; i Ɛ population (1,n)
Fitness = f(Sand Cat) = f(x1, x2,…, xd) ; ∀xi (is calculated for n time)

Figure 3.1: Working mechanism of SCSO in initial and definition phase

Initial Population

In the SCSO algorithm, the population is initialized as a matrix of sand cats, with each cat being represented as a $1{\times}d$ array, where d is the number of dimensions in the optimization problem, and is defined as shown in Figure (3.1).The initial population is created according to the size of the problem $(N_{\text{pop}} \times N_d)$, $(pop = 1, \cdots, n)$, with each sand cat's position being represented by a set of floating-point numbers $(x_1, x_2, \cdots, x_d)$ indicating the values of the problem variables. These values must fall within the specified lower and upper boundaries for each variable $(\forall x_i \in [\text{lower, upper}])$.
The fitness of each sand cat in the initial population is determined based on its position and its ability to provide a viable solution to the optimization problem.

Exploration

The exploration phase is a crucial part of this process, where the sand cats actively search for promising areas within the entire solution space. This phase utilizes several equations and parameters to guide the movement of these virtual predators. It begins with the creation of a ”candidate matrix” representing the current population of sand cat positions, each corresponding to a potential solution. Each sand cat's fitness is then evaluated using a specific function that assesses its effectiveness in solving the problem at hand.

A key aspect of the exploration phase is ensuring the sand cats don't get trapped in areas with sub-optimal solutions (local optima). The algorithm achieves this through an ”adaptive strategy” that utilizes two parameters: $\vec{r}_G$ and R. The $\vec{r}_G$ value (Equation (3.9)) represents the overall sensitivity range for the entire population and decreases linearly throughout the search. The R vector, derived from Equation (3.10), controls the transition between exploration and exploitation (refining current solutions). This balanced approach helps the sand cats explore new areas while also focusing on promising regions identified earlier. The sensitivity range for each sand cat is diferent, to avoid the local optimum trap and it is realized by Equation (3.11). Therefore, the $\vec{r_G}$ indicates the general sensitivity range that is decreased linearly from 2 to 0. Besides, $\vec{r}$ demonstrates sensitivity range of each cat.In addition, iter _c is current iteration and $\text{ iter }_{\text{Max}}$ is maximum iterations.

$$\overrightarrow{r_G} = s_M - \left(\frac{2 \times S_M \times \text{ iter }_c}{\text{iter }_{\text{Max}} + \text{ iter }_{\text{max}}} \right), \tag{3.9}$$

$$\vec{R} = 2 \times \overrightarrow{r_G} \times \text{rand}(0,1) - \overrightarrow{r_G}, \tag{3.10}$$

$$\vec{r} = \overrightarrow{r_G} \times \text{rand}(0,1). \tag{3.11}$$

During exploration, Each search agent (sand cat) updates its own position based on the best-candidate position ($\overrightarrow{\text{Pos}_{bc}}$) and its current position ($\overrightarrow{\text{Pos}_c}$) and its sensitivity range ($\vec{r}$). Therefore, the sand cats able to find the possible other best prey position (Equation (3.12)). This randomness allows the sand cats to investigate different parts of the search space and avoid settling for readily available but potentially sub-optimal solutions.

$$\overrightarrow{\text{Pos}}(t+1) = \vec{r} \cdot \left(\overrightarrow{\text{Pos}_{bc}}(t) - \text{rand}(0,1) \cdot \overrightarrow{\text{Pos}_c}(\text{t}) \right). \tag{3.12}$$

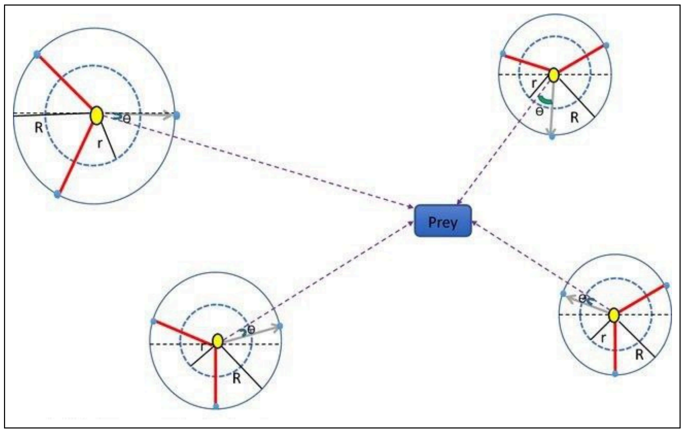
Exploitation

Following the exploration phase where sand cats search a vast area, the SCSO algorithm transitions to an exploitation phase focused on refining promising solutions. This phase employs Equation 3.13 to calculate the distance between the current best position (potential prey) and each sand cat's location. The sand cat's sensitivity range is modeled as a circle, and the Roulette Wheel selection algorithm picks a random angle within that circle to introduce beneficial randomness (between -1 and 1). This not only avoids local optima traps but also guides each sand cat towards the optimal solution from various circular directions within the search space.

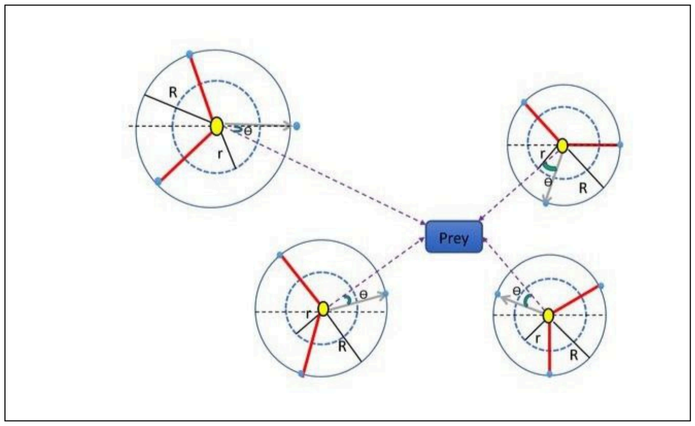
$$\begin{aligned} \overrightarrow{\text{Pos}_{\text{rnd}}} &= \left| \text{rand}(0,1) \cdot \overrightarrow{\text{Pos}_b}(t) - \overrightarrow{\text{Pos}_c}(\text{t}) \right| \\ \overrightarrow{\text{Pos}}(t+1) &= \overrightarrow{\text{Pos}_b}(t) - \vec{r} \cdot \overrightarrow{\text{Pos}_{\text{rnd}}} \cdot \cos(\theta) \end{aligned} \tag{3.13}$$

Equation 3.13 factors in the distance, random angle, and other parameters to update each sand cat's position, effectively steering them closer to the prey as iterations progress as seen in Figure (3.2).

In essence, the exploitation phase strategically refines solutions through calculated distances, random directional movement, and targeted position updates, ultimately leading the sand cats to converge on the optimal solution



(a) Position updating in iteration$_i$.



(b) Position updating in iteration$_{i+1}$.

Figure 3.2: Position updating mechanism in SCSO.

Balance between exploration and exploitation

The SCSO algorithm strikes a balance between exploration and exploitation through the dynamic duo of r_G and R parameters. r_G, mimicking a sand cat's narrowing focus, decreases linearly with iterations. R, derived from r_G, controls the switching between phases and falls within a range determined by r_G. The beauty lies in how a well-balanced r_G translates to a well-balanced R, ensuring smooth transitions between exploring new areas and exploiting promising solutions. Equation 3.14 plays a crucial role here. When the absolute value of $R$is ≤ 1, the equation steers the sand cats towards exploiting the current best area. Otherwise, it encourages exploration for new possibilities.

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{\text{Pos}_b}(t) - \overrightarrow{\text{Pos}_{rnd}} \cdot \cos(\theta) \cdot \vec{r} & |R| \leq 1;\ \text{exploitation} \\ \vec{r} \cdot \left(\overrightarrow{\text{Pos}_{bc}}(t) - \text{rand}(0,1) \cdot \overrightarrow{\text{Pos}_c}(\text{t})\right) & |R| > 1;\ \text{exploration} \end{cases} \tag{3.14}$$

Computational Complexity and The pseudocode of SCSO

The SCSO algorithm's computational complexity is $O(n^2)$, where n represents the population size. This signifies that the calculation time increases as the square of the population size. While initializing the algorithm and updating positions also contribute to the complexity, their impact is considered less significant, especially when n and m (problem size) have similar values. In simpler terms, the SCSO algorithm's efficiency scales with the square of the population size.

Algorithm 2 . Pseudocode of SCSO

```
Initialize the population
Calculate the fitness function based on the objective function
Initialize the $r$, $r_G$, $R$
while (t$\leq$ maximum iteration) do
    for each search agent do
        Get a random angle based on the Roulette Wheel Selection ($0^\circ \leq \theta \leq 360^\circ$ )
        if (abs(R)$\leq$1) then
            Update the search agent position based on the Eq (3.13)   ; $\overrightarrow{\text{Pos}}_b(t) - \overrightarrow{\text{Pos}}_{rnd} \cdot \cos(\theta) \cdot \vec{r}$
        else
            Update the search agent position based on the Eq (3.12)   ; $\vec{r} \cdot \left(\overrightarrow{\text{Pos}_{bc}}(t) - \text{rand}(0,1) \cdot \overrightarrow{\text{Pos}_c}\right)$
        end if
    end for
    $t = t++$
end while
```

3.2.3 Dung Beetle Optimizer

The dung beetle optimizer (DBO) is a novel population-based technique inspired by the behaviors of dung beetles in nature (ball-rolling, dancing, foraging, stealing, and reproduction)[49]. The algorithm uses behavior-inspired rules to update the positions of agents (dung beetles) in the search space, aiming to find a balance between global exploration and local exploitation. The DBO technique uses iterative optimization and dynamic parameters to effectively search for high-quality solutions in copmlex optimization problems.

The Dung Beetle Optimizer algorithm operates through a series of steps to efficiently search for optimal solutions in optimization problems.

Initialization

The algorithm starts by initializing a population of dung beetles within the search space. Each dung beetle represents a potential solution to the optimization problem. The position vector of the ith dung beetle at iteration t is denoted in Equation 3.15 where D is the dimension of the search space.

$$x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t)) \tag{3.15}$$

Movement of Dung Beetles

The movement of dung beetles is crucial because it mimics their natural activities. The positions of dung beetles are updated using dynamic equations based on their foraging, rolling, and reproductive behaviors.The specific equation for updating the position of the brood ball can be expressed as:

$$B_i(t+1) = X^* + b_1 \times (B_i(t) - Lb^*) + b_2 \times (B_i(t) - Ub^*) \tag{3.16}$$

Where $B_i(t+1)$ represents the updated position of the brood ball dung beetle i at the next iteration $t+1$, X^* is a calculated value that influences the movement of the brood ball, b_1 and b_2 are independent random vectors that introduce stochasticity into the movement, $B_i(t)$ is the current position of the brood ball dung beetle at iteration t, Lb^* and Ub^* are calculated values based on optimal foraging areas, which guide the movement of the brood ball towards promising regions in the search space.

Exploration and Exploitation

The DBO algorithm achieves a balance between exploration and exploitation by dynamically adjusting parameters. This adaptive strategy allows the algorithm to explore the search space thoroughly and exploit promising regions effectively, leading to improved optimization performance and the discovery of high-quality solutions.

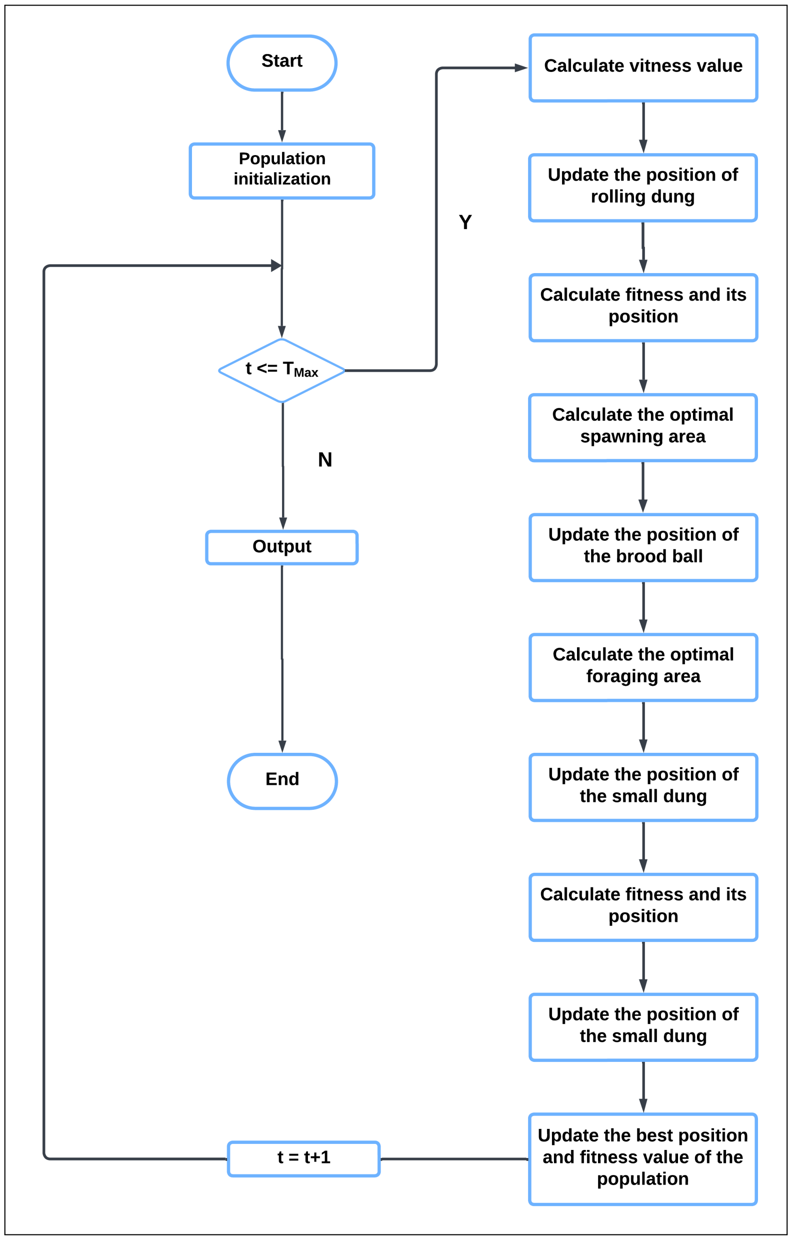

Start
Population initialization
t <= T_Max
Y
N
Output
End
t = t+1
Calculate vitness value
Update the position of rolling dung
Calculate fitness and its position
Calculate the optimal spawning area
Update the position of the brood ball
Calculate the optimal foraging area
Update the position of the small dung
Calculate fitness and its position
Update the position of the small dung
Update the best position and fitness value of the population

Figure 3.3: Flowchart of dung beetle optimization algorithm.

3.2.4 Red-tailed hawk Optimizer

The Red-tailed Hawk algorithm (RTH) is a nature-inspired metaheuristic optimization algorithm developed to address a variety of optimization problems[50]. This algorithm is based on the red-tailed hawk's hunting strategy; it mimics the stages of a hawk's hunt, such as high soaring, low soaring, stooping and swooping. These steps represent the hawk's exploration of the search space, selection of the prey position, and final attack on the target.

Mathematical Model

The RTH algorithm consists of three key stages mirroring the red-tailed hawk's hunting behavior.

1. High soaring

Red-tailed hawks soar far into the sky to find the best food source. Figure 3.4 shows the behavior of red-tailed hawks during the high soaring stage, whereas Eq.(3.17) represents the mathematical model for this stage.

$$X(t) = X_{\text{best}} + (X_{\text{mean}} - X(t-1)) \cdot \text{Levy}(\text{dim}) \cdot TF(t) \tag{3.17}$$

Where

$X(t)$	:	The red-tailed hawk position at the iteration t
X_{best}	:	The best-obtained position.
X_{mean}	:	The positions' mean.
$Levy$	:	The levy fight distribution function that is calculated in Eq.(3.18)
$TF(t)$	:	The transition factor function that is calculated in Eq.(3.19).

$$\text{Levy}(\text{dim}) = s\frac{\mu \cdot \sigma}{|v|^{\beta-1}} \quad ; \quad \sigma = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi\beta/2)}{\Gamma(1+\beta/2) \cdot \beta \cdot 2^{(1-\beta/2)}}\right) \tag{3.18}$$

$$TF(t) = 1 + \sin\left(2.5 + (t/T_{\text{max}})\right) \tag{3.19}$$

where s is a constant (0.01), dim is the problem dimension, β is a constant (1.5), u, and v are random numbers [0 to 1], and T_{max} represents the max number of iterations.

2. Low soaring

The hawk surrounds the prey by flying extremely close to the ground in a spiraling motion. Figure 3.5 illustrates this stage and its model, which can be expressed as:

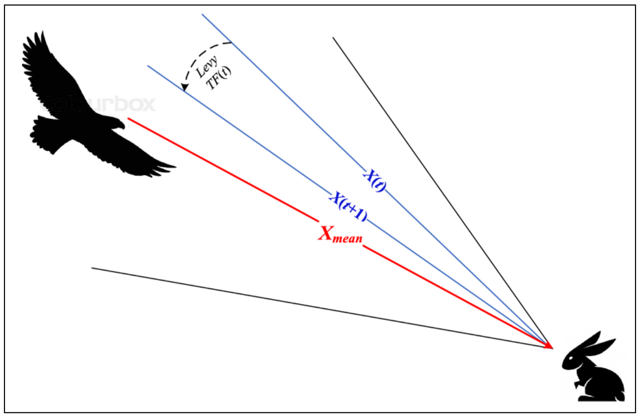


Figure 3.4: Behavior of red-tailed hawk during high soaring stage.

$$\begin{aligned} X(t) &= X_{\text{best}} + (x(t) + y(t)) \cdot \text{StepSize}(t) \\ \text{StepSize}\ (t) &= X(t) - X_{\text{mean}} \end{aligned} \tag{3.20}$$

where x and y are direction coordinates, which can be calculated as follows:

$$\begin{cases} x(t) = R(t) \cdot \sin(\theta(t)) \\ y(t) = R(t) \cdot \cos(\theta(t)) \end{cases} \begin{cases} R(t) = R_0 \cdot (r - t/T_{\max}) \cdot \ \text{rand} \\ \theta(t) = A \cdot (1 - t/T_{\max}) \cdot \ \text{rand} \end{cases} \begin{cases} x(t) = x(t)/ \max|x(t)| \\ y(t) = y(t)/ \max|y(t)| \end{cases} \tag{3.21}$$

where R_0 represents the initial value of the radius [0.5–3], A denotes the angel gain [5–15], rand is a random gain [0–1], and r is a control gain [1, 2]. These parameters help the hawk fly around the prey with spiral movements.

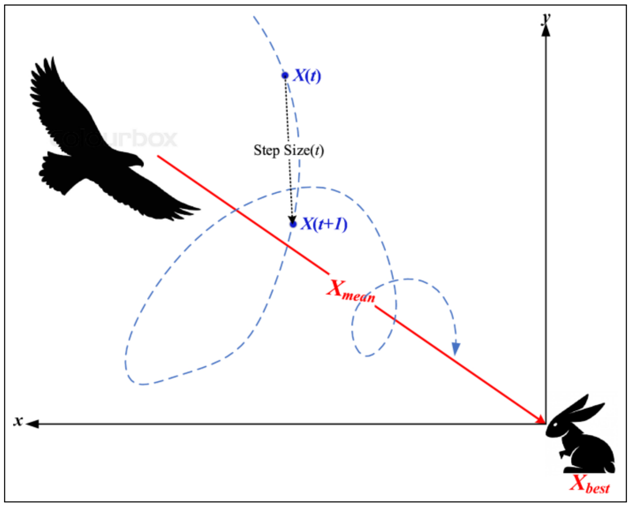


Figure 3.5: Behavior of red-tailed hawk during low soaring stage.

3. Stooping and Swooping

During this stage, the hawk stoops to attack the prey from its best low-soaring position. Figure 3.6 illustrates the behavior of red-tailed hawks at this stage . It can be represented as follows:

$$X(t) = \alpha(t) \cdot X_{\text{best}} + x(t) \cdot \text{StepSize}\,1(t) + y(t) \cdot \text{StepSize}\,2(t) \tag{3.22}$$

where each step size can be calculated as follows:

$$\begin{aligned} \text{StepSize1 } (t) &= X(t) - TF(t) \cdot X_{\text{mean}} \\ \text{StepSize2 } (t) &= G(t) \cdot X(t) - TF(t) \cdot X_{\text{best}} \end{aligned} \tag{3.23}$$

where α represents the hawk's acceleration that increases with the increase of t to enhance the convergence speed, and G is the gravity effect that decreases to reduce the exploitation diversity when the hawk is much near the prey.They can be defined as follows:

$$\begin{aligned} \alpha(t) &= \sin^2(2.5 - t/T_{\max}) \\ G(t) &= 2 \cdot (1 - t/T_{\max}) \end{aligned} \tag{3.24}$$

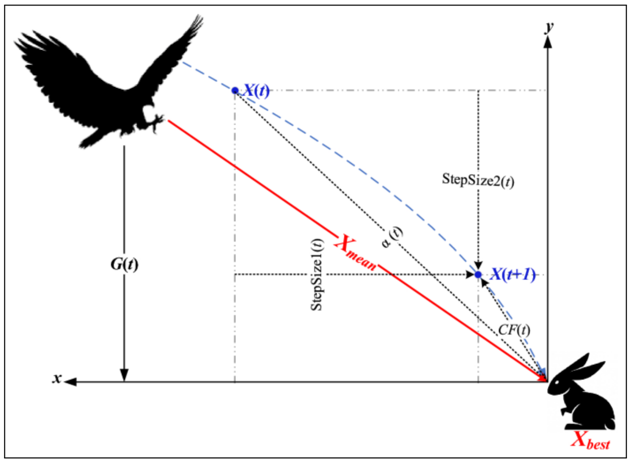


Figure 3.6: Behavior of red-tailed hawk during stooping and swooping stages.

Algorithm 3 . Pseudocode of RTH

```
Initialization: random generation within the search space.
while $t < T_{max}$ do
    High soaring stage:
    for $i = 1 : N_{pop}$ do
        Calculate Levy flight distribution Eq(3.18)
        Calculate the transition factor TF Eq(3.19)
        Update positions Eq(3.17)
    end for
    Low soaring stage:
    for $i = 1 : N_{pop}$ do
        Calculate direction coordinates Eq(3.21)
        Update positions Eq(3.20)
    end for
    Stooping and Swooping stage:
    for $i = 1 : N_{pop}$ do
        Calculate the acceleration and the gravity factors Eq(3.24)
        Calculate the step size Eq(3.23)
        Update positions Eq(3.22)
    end for
end while
```

3.3 Related Work

3.3.1 Photovoltaic models

Designing solar cells (SCs) and photovoltaic (PV) modules requires a precise mathematical model to accurately extract the solar cell's parameters[51]. This model is then used to design an electronic circuit with diodes that mimics the solar cell's behavior. A crucial step in achieving an accurate design is selecting an appropriate mathematical model. Subsequently, search strategies can be employed to find the optimal parameter values for the chosen model.

Commonly, electrical equivalent circuits are used to represent PV cells or modules.These circuits typically include a current source, a diode, and resistors. Three main types of models are used to create these equivalent circuits: the single-diode model (SDM), the double-diode model (DDM), the Photovoltaic module model (PVM), and the triple-diode model (TDM)[52]. Each model offers advantages and disadvantages, with the number of diodes directly impacting the accuracy of the predicted current-voltage (I-V) curve.

Single-Diode Model

The simplicity and accuracy make the single diode model as the simplest PV model[51].Its structure is given in Figure (3.7). The current output I_L can be calculated by Equation (3.25) .

$$I_L = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n \cdot k \cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}} \tag{3.25}$$

Where

I_{ph}	:	The photo-generated current.
I_{sd}	:	The reverse saturation current of diode.
V_L	:	The value the output voltage.
R_s	:	The value the series resistance.
R_{sh}	:	The shunt resistance.
T	:	The cell temperature $(°\mathrm{K})$.
n	:	The diode ideality factor .
k	:	The Boltzmann constant ($k = 1.3806503 \times 10^{-23}$ J/K).
q	:	The electron charge ($q = 1.60217646 \times 10^{-19}$ C).

It can be seen that there are five unknown parameters (I_{ph}, I_{sd}, R_s,R_{sh}, n) need to be estimated in the SD model.

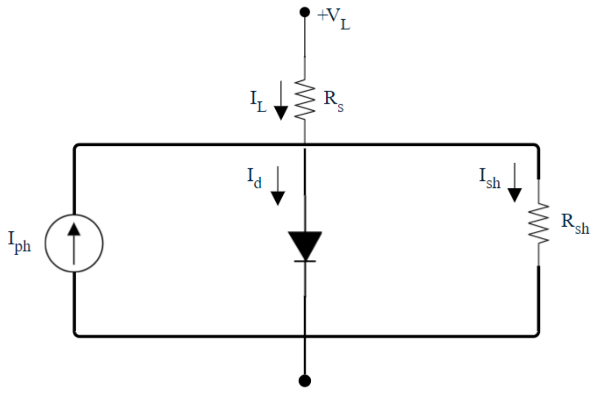


Figure 3.7: The schematic diagram of the SD model.

Double-Diode Model

The DD model is also commonly used in practice and literature, is more complex than the single-diode model, and includes two diodes to account for recombination currents in the solar cell[53]. As presented in Figure 3.8 , its output current is computed based on Equation (3.26).

$$\begin{aligned} I_L = &I_{ph} - I_{sd1} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n_1 \cdot k \cdot T}\right) - 1\right] - I_{sd2} \cdot \left[\exp\left(\frac{q \cdot (V_L + R_S \cdot I_L)}{n_2 \cdot k \cdot T}\right) - 1\right] \\ &- \frac{V_L + R_S \cdot I_L}{R_{sh}} \end{aligned} \tag{3.26}$$

where I_{sd1} and I_{sd2} are used to denote the diffusion and saturation currents, respectively. n_1 and n_2 are the ideal factors for the diffusion diode and the recombination diode, respectively.

Compared with the SD model, more parameters (I_{ph}, I_{sd1}, I_{sd2}, R_s, R_{sh}, n_1, n_2) should be identified accurately in the DD model.

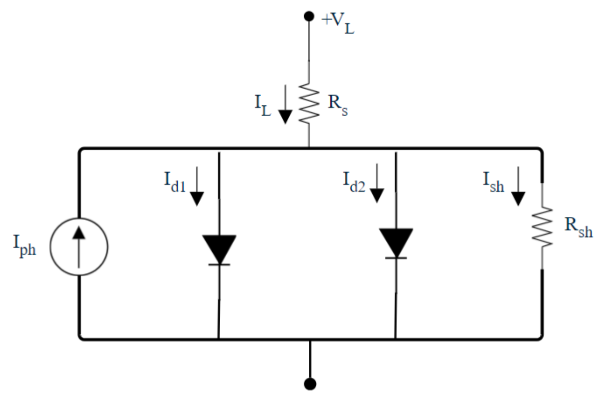


Figure 3.8: The schematic diagram of the DD model.

Photovoltaic module model

Different from the above two models, the PV module typically includes several solar cells connected in parallel or in series[53].Its structure is given in Figure 3.9. The output current can be calculated by Equation (3.27).

$$I_L/N_p = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q \cdot (V_L/N_S + R_S \cdot I_L/N_p)}{n \cdot k \cdot T}\right) - 1\right] - \frac{V_L/N_S + R_S \cdot I_L/N_p}{R_{sh}} \tag{3.27}$$

where N_p and N_S indicate the size of solar cells used in parallel and series, respectively. It is clear that there exist five unknown parameters(I_{ph}, I_{sd}, R_S, R_{sh}, n) in the PV module.

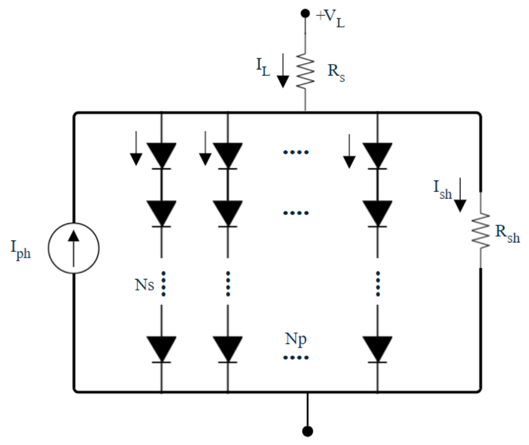


Figure 3.9: The schematic diagram of the PV module model.

Triple-diode model

This is the most complex and accurate PV model. It includes three diodes to account for different types of recombination currents in the solar cell and is more appropriate for industrial applications[51]. The equivalent circuit of the TD model after the parallel connection of the third diode is shown in Figure 3.10.The TDM's output current could be estimated by:

$$\begin{aligned} I_L =& I_{ph} - I_{sd1}\left[\exp\left(\frac{q\cdot(V_L+R_s\cdot I_L)}{n_1\cdot k\cdot T}\right)-1\right] - I_{sd2}\left[\exp\left(\frac{q\cdot(V_L+R_s\cdot I_L)}{n_2\cdot k\cdot T}\right)-1\right] \\ & - I_{sd3}\left[\exp\left(\frac{q\cdot(V_L+R_s\cdot I_L)}{n_3\cdot k\cdot T}\right)-1\right] - \frac{V_L+R_s\cdot I_L}{R_{sh}} \end{aligned} \tag{3.28}$$

where I_{sd3} is the current through the third diode, and n_3 represents the third diode's ideality factor.The TDM's mathematical model has nine unknown parameters that need to be accurately estimated to maximize the performance of the solar cell. These parameters are namely I_{ph},I_{sd1},I_{sd2},I_{sd3}, R_s, R_{sh}, n_1, n_2, and n_3.

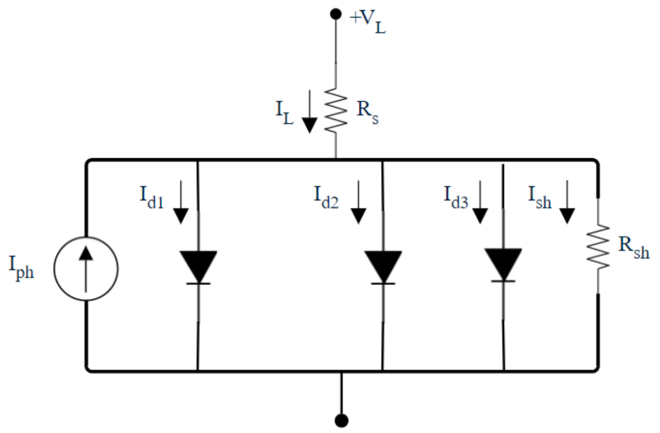


Figure 3.10: The schematic diagram of the TD model.

3.3.2 problem formulation

Accurate modeling of photovoltaic (PV) cells is critical for optimizing performance and predicting energy production. Parameter estimation is a critical phase in this process, in which mathematical equations are employed to determine values that represent the cell's characteristics. However, just knowing equations is not enough. We need a method to determine how accurate these estimated parameters are. This is where the objective function comes in. It is desired that the measured parameter values can approximate the experimental data as much as possible; that is, the error between them is as small as possible. Equations 3.29 and 3.30 show the calculation method for the SD and DD models[53], respectively. In Equation 3.31 [53], the root mean square error (RMSE) is used to calculate the overall error, used to measure the differences between values predicted by a model or an estimator and the actual values, leading to a more reliable and accurate representation of the PV cell's behavior, ultimately enabling optimal performance and energy prediction.

$$\begin{cases} f_k\left(V_L, I_L, \mathbf{x}\right) = I_{ph} - I_{sd} \cdot \left[\exp\left(\frac{q\cdot(V_L + R_S \cdot I_L)}{n\cdot k\cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}} - I_L \\ \mathbf{x} = \{I_{ph}, I_{sd}, R_S, R_{Sh}, n\} \end{cases} \tag{3.29}$$

$$\begin{cases} f_k\left(V_L, I_L, \mathbf{x}\right) = I_{ph} - I_{sd1} \cdot \left[\exp\left(\frac{q\cdot(V_L + R_S \cdot I_L)}{n_1\cdot k\cdot T}\right) - 1\right] \\ \qquad - I_{sd2} \cdot \left[\exp\left(\frac{q\cdot(V_L + R_S \cdot L_L)}{n_2\cdot k\cdot T}\right) - 1\right] - \frac{V_L + R_S \cdot I_L}{R_{sh}} - I_L \\ \mathbf{x} = \{I_{ph}, I_{sd1}, I_{sd2}, R_S, R_{Sh}, n_1, n_2\} \end{cases} \tag{3.30}$$

$$\text{RMSE}(\mathbf{x}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} f_k\left(V_L, I_L, \mathbf{x}\right)^2} \tag{3.31}$$

where N shows the number of experimental data.

3.4 Results and discussion

This section summarizes and discusses the optimization results of the solar PV cell parameter extraction problem. In the study, the electrical parameters of the single-diode cell model and the double-diode cell model were optimized with OOBO, SCSO, RTH, and DBO algorithms.

- The I-V data of single-diode and double-diode solar cells were taken from [53]. The data was measured on a 57-mm-diameter RTC France commercial silicon solar cell under 1000 W/m^2 at 33 $^\circ C$.
- For all algorithms, the population size and maximum iteration number were set to 30 and 1000, respectively, throughout the simulation studies. Furthermore, each algorithm was executed 30 times in MATLAB online to remedy its stochastic nature.The configuration of the lower (LB) and upper (UB) bounds for the SD and DD models are displayed in Table 3.2.

The outcomes are analyzed in terms of several performance metrics, such as standard deviation (std), best RMSE, average (Avg) RMSE, and worst RMSE. In addition, the I-V and P-V curves These curves help in determining the efficiency, power output, and behavior of the solar cells under different conditions. Also, a convergence curve is used in the comparison to disclose the convergence speed of each algorithm.

Table 3.2: Parameters ranges of PV models

Parameter	Single/Double diode	
	Lower bound	Upper bound
I_{ph} (A)	0	1
I_{sd}, I_{sd1}, I_{sd2}(μA)	0	1
R_S(Ω)	0	0.5
R_{sh}(Ω)	0	100
n, n_1, n_2	1	2

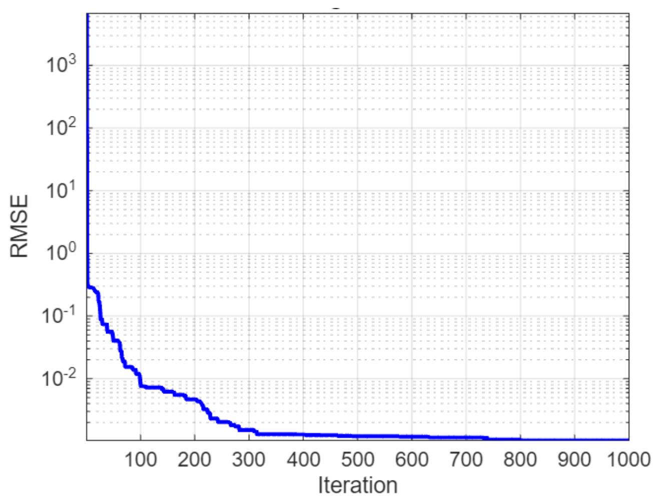
Result on single diode model

The OOBO, SCSO, RTH, and DBO algorithms are applied to extract the electrical parameters of the single-diode solar cell. Accordingly, the optimized solar cell parameters and the corresponding RMSE results are given in Table **??**. As can be seen from the table, the RTH algorithm obtained the minimum RMSE result with a value of 9.86E-04, followed by the OOBO (1.03E-03), SCSO (3.94E-03), and DBO (8.34E-03) algorithms. Figures 3.12 and 3.13 highlight that the simulated data (I-V and P-V) of three algorithms (RTH,OOBO,SCSO) are in good agreement with the experimental data points, thereby proving their effectiveness.For the DBO, its simulated data are not completely identical to the experimental data, which makes it less effective compared to the other algorithms.

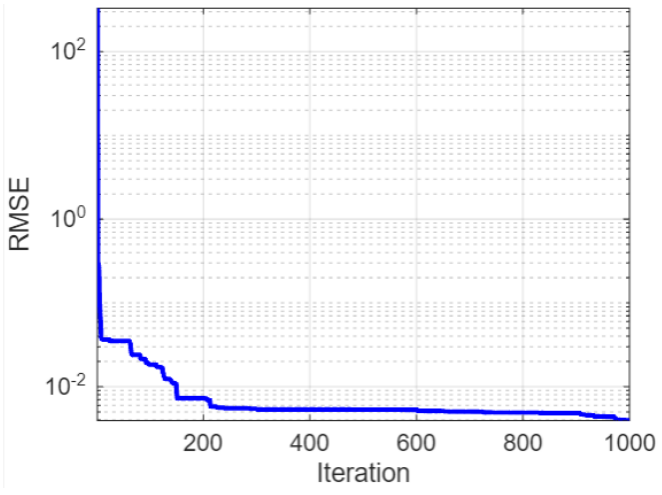
The parameter estimation algorithms OOBO, SCSO, and RTH show promising performance, as shown in Figure 3.11. Their convergence curves demonstrate a steady decrease in RMSE with increasing iterations, their ability to effectively minimize the difference between the simulated data and the actual data. This implies that these algorithms have the ability to find parameter values that result in a good fit, which might lead to reliable parameter estimate. In comparison, the DBO curve performs less efficiently, resulting in a greater RMSE value. This implies that DBO may not be as good at identifying optimal parameter values, perhaps resulting in less accurate parameter estimates.

Table 3.3: Comparison of experimental results for SD Model.

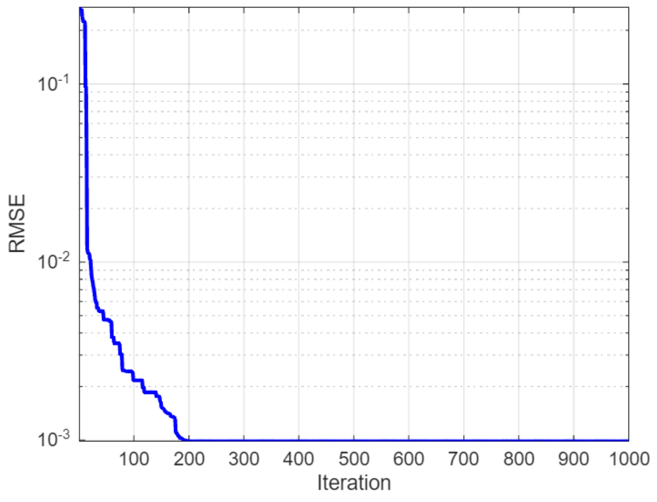
Parameters	OOBO	SCSO	RTH	DBO
I_{ph}	0.7606656152	0.7660362261	0.7607755310	0.7665235979
I_{sd}	2.966E-04	3724E-07	3230E-07	57623E-06
R_S	0.0366811638	0.0345825476	0.0363770918	0.0238527017
R_{sh}	50.0913102768	23.7943504934	53.7185209972	100.0000000000
n	1.4726864715	1.4970033356	1.4811836076	1.8434713087
$RMSE$	1.0363550777E-03	3.9488531675E-03	9.8602187789E-04	8.3468571895E-03



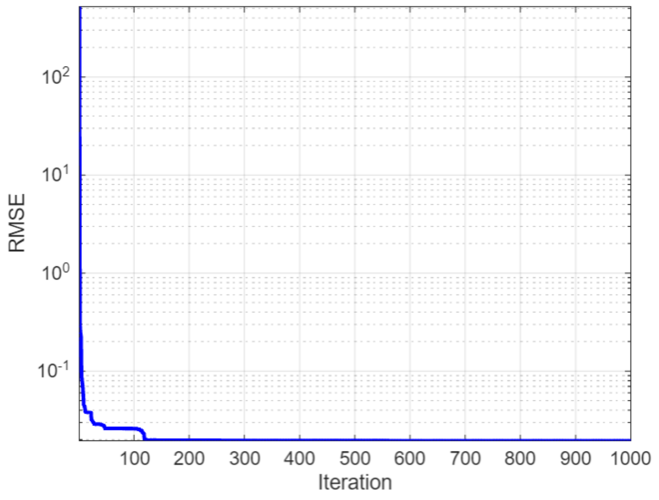
(a) Convergence curve of OOBO.



(b) Convergence curve of SCSO.

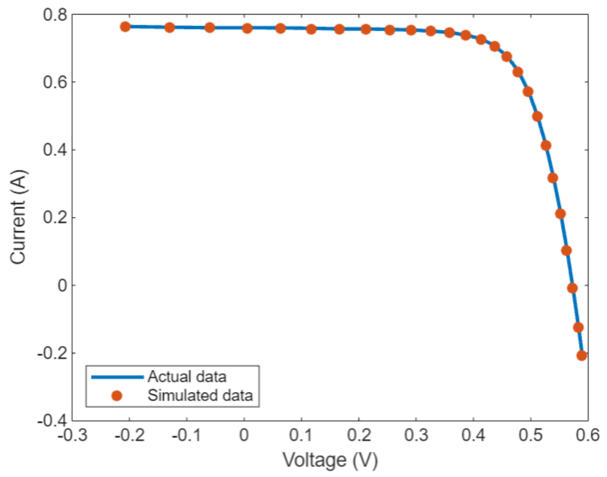


(c) Convergence curve of RTH.

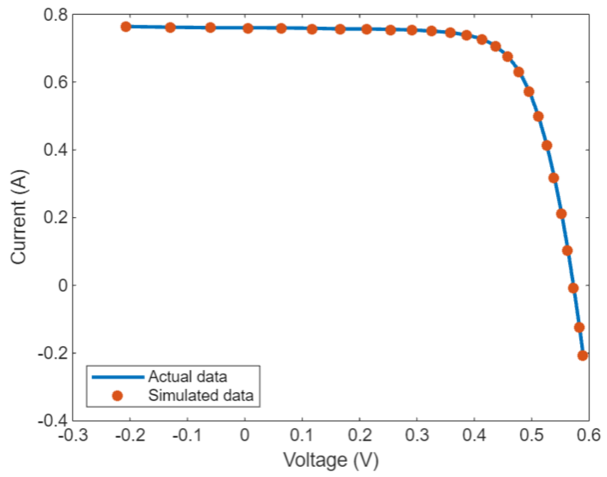


(d) Convergence curve of DBO.

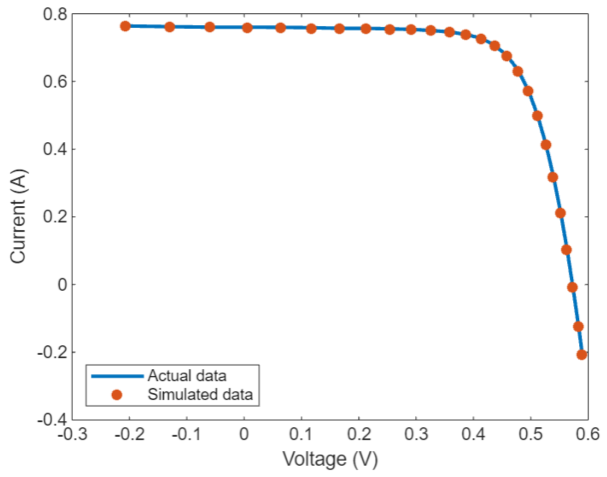
Figure 3.11: Convergence curves of the algorithms for the SD model.



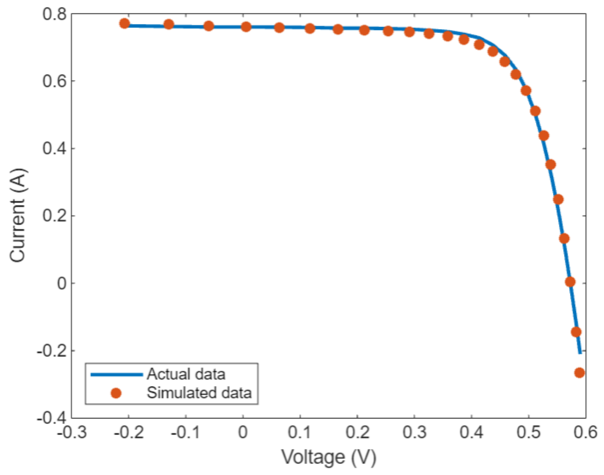
(a) I-V curve of OOBO.



(b) I-V curve of SCSO.

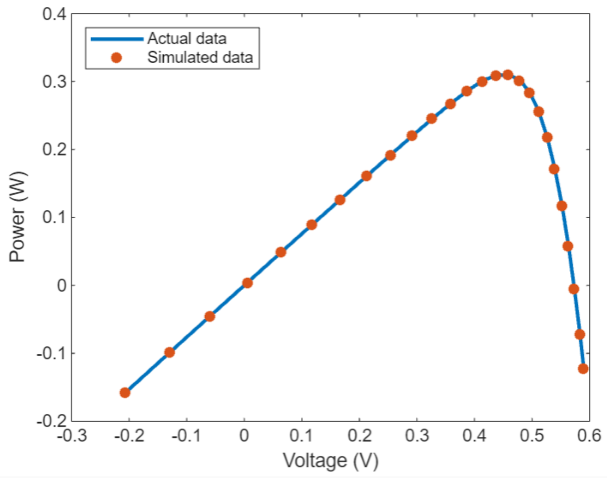


(c) I-V curve of RTH.

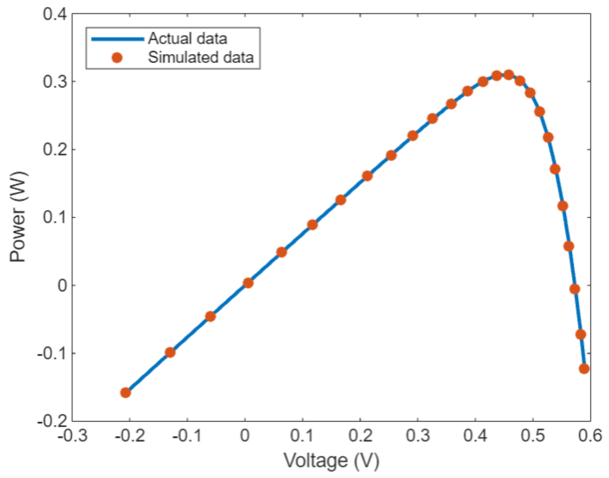


(d) I-V curve of DBO.

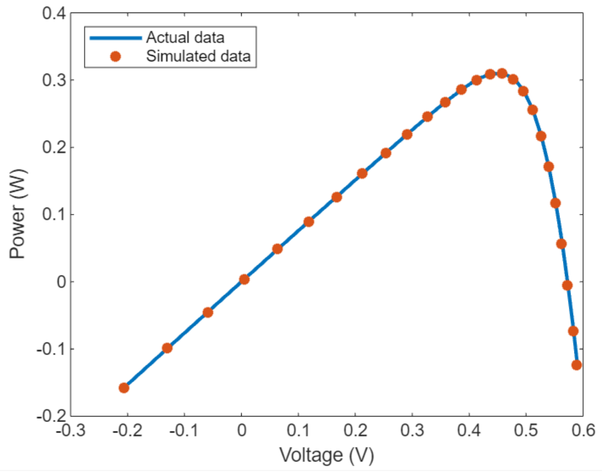
Figure 3.12: I-V curves of the algorithms for the SD model.



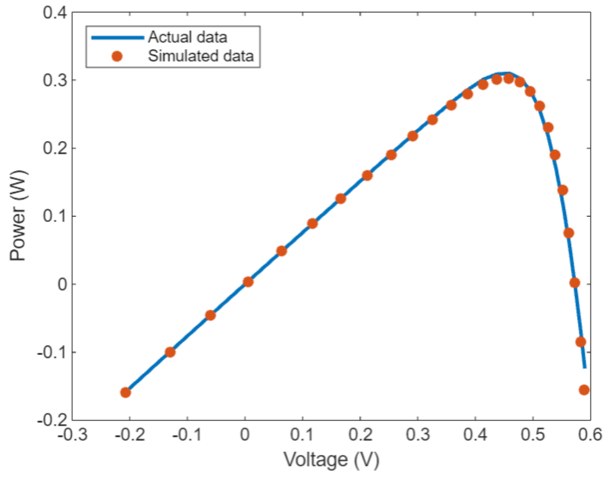
(a) P-V curve of OOBO.



(b) P-V curve of SCSO.



(c) P-V curve of RTH.



(d) P-V curve of DBO.

Figure 3.13: P-V curves of the algorithms for the SD model.

Result on double diode model

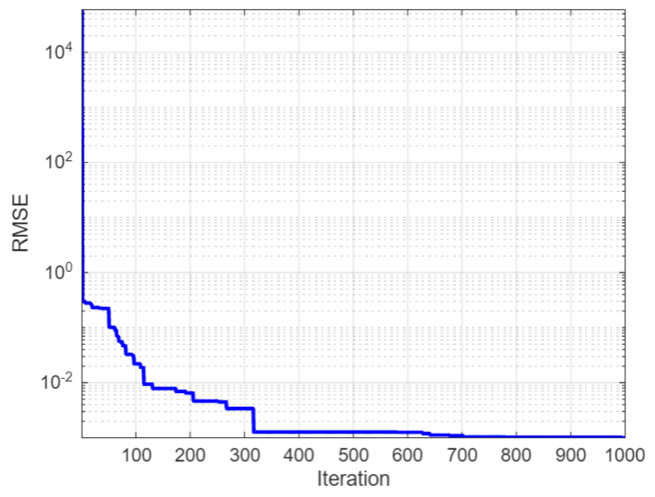
In this solar cell model, there are seven parameters that need to be estimated. Table **??** gives simulation results for the double-diode solar cell. Considering the results reported in the table, it is seen that the RTH algorithm exhibited the best prediction performance with an RMSE value of 9.86E-04, while DBO gave the worst performance. The performance of the OOBO and SCSO algorithms is remarkable.

Based on Figure 3.14 the RMSE curves for OOBO, SCSO, and RTH show a significant decrease as the iteration number increases. This shows they are efficient at estimating parameter values that lead to a strong model fit (possibly leading to faster convergence). When comparing the DBO curve to the others, it shows a slower RMSE drop and a higher ultimate value. This indicates that DBO might be unable to find optimal parameters, resulting in a less accurate model.

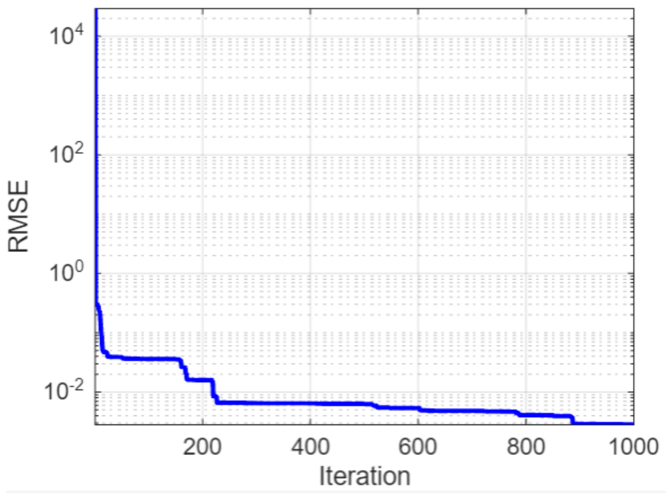
In Figures 3.15 and 3.16, the values of I and P obtained by RTH, OOBO, and SCSO are in high accordance with measured data for the DD model. These results clearly show that they are effective for parameter estimation for the DD model. DBO, on the other hand, may need additional exploration due to its lower accuracy in parameter estimation.

Table 3.4: Comparison of experimental results for DD Model.

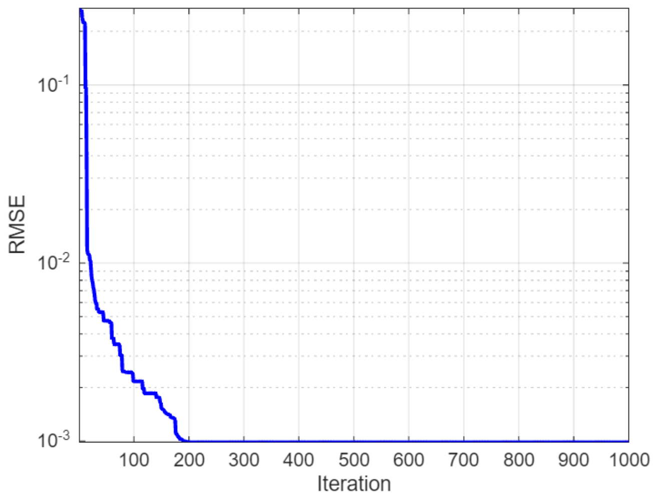
Parameters	OOBO	SCSO	RTH	DBO
I_{ph}	0.7605491031	0.7640579341	0.7607755298	0.7600669549
I_{sd1}	3.160E-07	4.650E-07	3.230E-07	9.196E-07
R_S	0.0361515074	0.0340217816	0.0363770931	0.0341941053
R_{sh}	58.3600164972	31.6347804509	53.7185209972	54.8165430770
n_1	1.4807402669	1.5197613887	1.4811835836	1.6174424750
I_{sd2}	2.512E-07	0.0000000000	0.4730620660	2.223E-07
n_2	1.9943475199	1.2299299218	1.0122550941	1.6414478431
$RMSE$	1.0057923010E-03	2.8076644646E-03	9.8602187789E-04	8.8522404440E-03



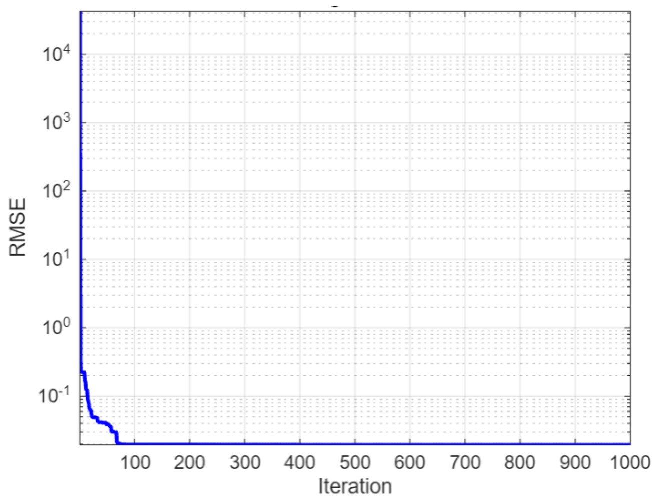
(a) Convergence curve of OOBO.



(b) Convergence curve of SCSO.

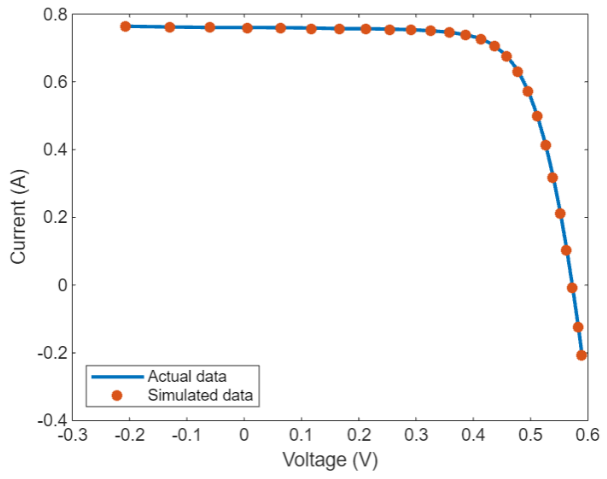


(c) Convergence curve of RTH.

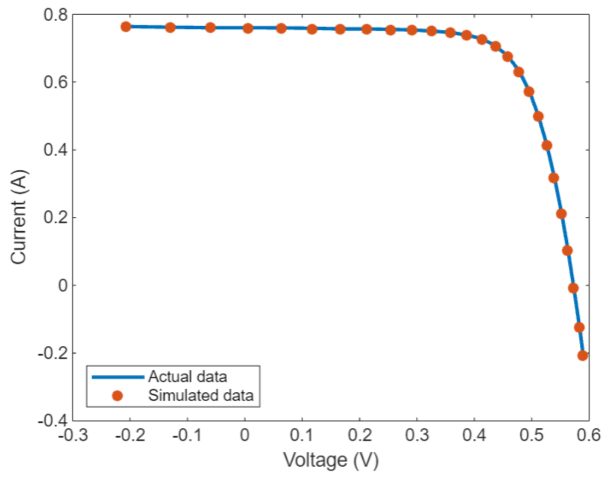


(d) Convergence curve of DBO.

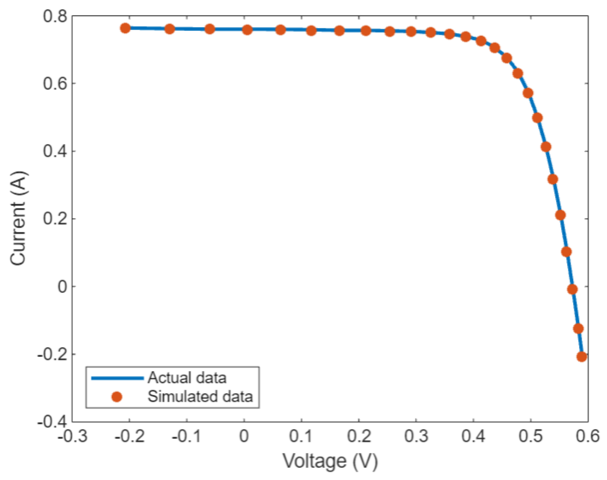
Figure 3.14: Convergence curves of the algorithms for the DD model.



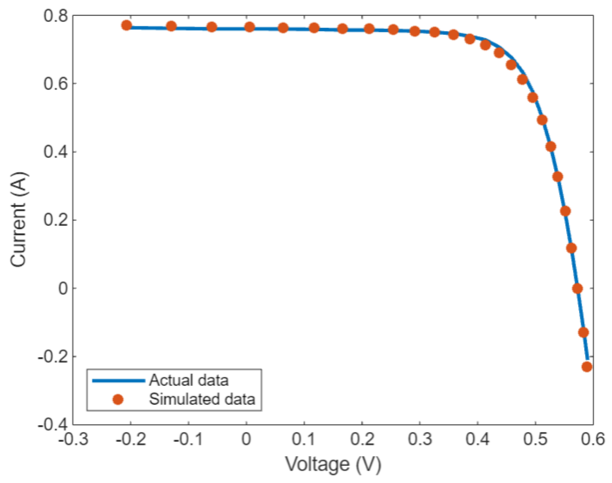
(a) I-V curve of OOBO.



(b) I-V curve of SCSO.

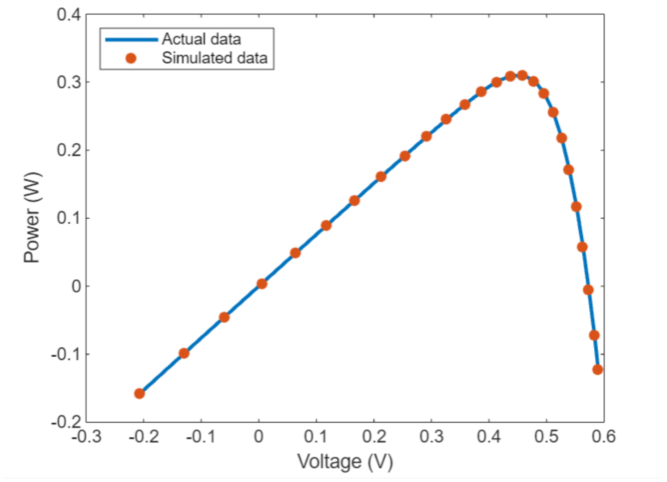


(c) I-V curve of RTH.

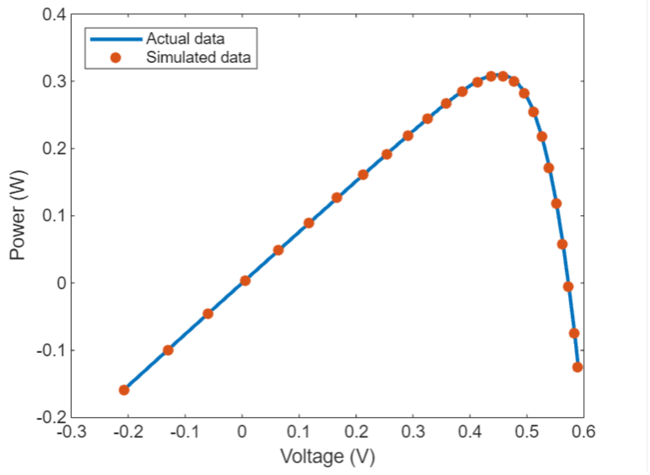


(d) I-V curve of DBO.

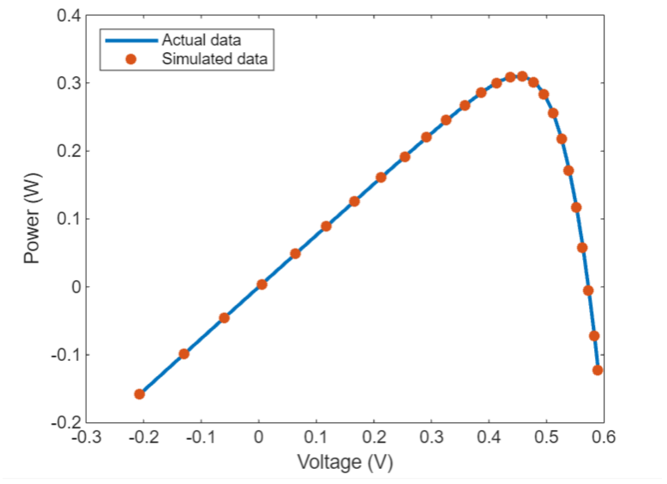
Figure 3.15: I-V curves of the algorithms for the DD model.



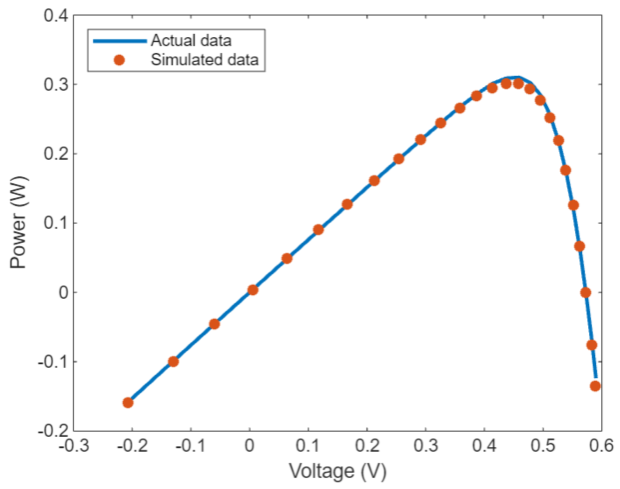
(a) P-V curve of OOBO.



(b) P-V curve of SCSO.



(c) P-V curve of RTH.



(d) P-V curve of DBO.

Figure 3.16: P-V curves of the algorithms for the DD model.

To compare the performance of the algorithms, we ran each one for 30 executions and gathered descriptive data for each test case, including SDM and DDM. Each of these test cases has the following information: the average of the best outcomes (Avg), the best result (Min), the worst result (Max), and the standard deviation (Std). The results of these two tests are reported in table 3.5. The statistics shown in this table show that the RTH algorithm produced exceptionally good results for both cases, followed by OOBO, SCSO, and DBO.

Table 3.5: Statistical results obtained by all algorithms on two PV models

Models	Algorithm	RMSE			
		Min	Max	Avg	Std
SD model	OOBO	1.0363550777E-03	2.2286139909E-01	2.0809294872E-01	5.6203193859E-02
	SCSO	3.9488531675E-03	2.2286139909E-01	1.7888224652E-01	9.8340374745E-02
	RTH	9.8602187789E-04	2.2286139909E-01	1.8588216956E-01	8.4101644692E-02
	DBO	8.3468571895E-03	3.0105827309E-01	1.5814760427E-01	8.4101644692E-02
DD model	OOBO	1.0057923010E-03	2.2286139909E-01	1.7143632750E-01	9.4812842612E-02
	SCSO	2.8076644646E-03	2.2286404236E-01	1.3637742992E-01	9.2113603702E-02
	RTH	9.8602187789E-04	2.2286139909E-01	1.7848632365E-01	9.0267357385E-02
	DBO	8.8522404440E-03	6.3074169621E-01	1.8902501009E-01	1.1742927870E-01

Discussion

The results demonstrated the distinct hierarchy in which the parameter estimation algorithms operate. RTH consistently had the lowest root mean square error (RMSE) in both single-diode (9.86E-04) and double-diode models, demonstrating its effectiveness in reducing the difference between simulated and measured data.

OOBO and SCSO performed similarly in both models, with a much lower RMSE than DBO. Their effectiveness in estimating the parameters for both models is demonstrated by the I-V and P-V curves in Figures 3.12, 3.13, 3.15, and 3.16, which closely match the simulated and experimental data. Their effective convergence is further demonstrated in Figure 3.14, where a strong model fit is established by a rapid decrease in RMSE as the number of iterations increases.

DBO, on the other hand, constantly had the highest RMSE of all the models, demonstrating a poor fit between the simulated and experimental data. Differences between simulated and experimental data appear to be visible, as seen in Figures 3.13 and 3.16 (P-V curves) ,3.12 and 3.15 (I-V curves). The slower decline in RMSE within the DBO convergence curve (Figures 3.14,3.11), which could result in less accurate models, suggests that finding the ideal parameter values may be difficult.

In conclusion, RTH is the best algorithm, with OOBO and SCSO ranking second and third. DBO's poor performance warrants additional study due to its lower parameter estimation accuracy.This shows that RTH may be the best strategy for determining the best parameter values.

3.5 Conclusion

This paper presents a comparative performance analysis of metaheuristic search algorithms for parameter extraction in photovoltaic solar cells. In the paper, the single-diode solar cell model and the double-diode solar cell model were used. The electrical parameters of solar PV models are extracted with OOBO, SCSO, RTH, and DBO algorithms.The analysis used Root Mean Squar Error (RMSE) as the key metric to evaluate each algorithm's efficacy in minimizing the difference between simulated and measured data. In addition, I-V and P-V curves were used to visually compare the agreement between simulated and experimental data.Given that all the results are together, it is evident that the RTH algorithm offered the lower RMSE value for all solar cell models. From the I-V and P-V curves, it is observed that the calculated data by the RTH algorithm is highly coincident with the measured data. As a result, this paper reports that RTH is an effective and powerful metaheuristic to deal with the parameter optimization of different solar cell models.

Conclusion and Perspectives

General Conclusion

This thesis has provided a thorough contrastive analysis of four metaheuristic search algorithms for parameter estimation in photovoltaic (PV) models: Red-tailed Hawk Optimizer (RTH), One-to-One Based Optimizer (OOBO), Sand Cat Swarm Optimization (SCSO), and Dung Beetle Optimizer (DBO). The single diode model and the double diode model, two common PV models, were the specific focus of the study.

The major goal was to determine the algorithms' efficacy and accuracy in estimating the parameters that optimize the efficiency of PV models. The root mean square error (RMSE) between simulated and actual data was used as the objective function to evaluate each algorithm's performance.

Our extensive experiments demonstrated that the Red-tailed Hawk Optimizer (RTH) consistently achieved superior performance, providing the best estimation accuracy with an RMSE value of 9.86E-04 for both PV models. This demonstrates that RTH successfully balances exploration and exploitation, making it a reliable and efficient approach for extracting parameters in PV systems.

Perspectives

The findings of this study make a significant contribution to the field of renewable energy, particularly in terms of improving the efficiency of solar energy systems through optimized parameter estimation. This work improves academic understanding by determining the most effective algorithm, while also providing practical insights for the development and deployment of more efficient photovoltaic systems.

Future research could involve extending this analysis to include more metaheuristic algorithms and testing their performance in different types of renewable energy systems. Furthermore, real-world applications and long-term performance assessments would be useful in determining the practical value of these findings.

Bibliography

[1] Wayne L Winston. Operations research: applications and algorithms, 4th edn [m]. *Publishing House of Tsinghua University, Beijing*, pages 406–452, 2006.

[2] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[3] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[4] Igor Griva, Stephen G Nash, and Ariela Sofer. *Linear and Nonlinear Optimization 2nd Edition*. SIAM, 2008.

[5] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[6] Stephen J Wright. Continuous optimization (nonlinear and linear programming). *Foundations of Computer-Aided Process Design*, 7, 1999.

[7] Frederick S Hillier and Gerald J Lieberman. *Introduction to operations research*. McGraw-Hill, 2015.

[8] Christodoulos A Floudas and Panos M Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.

[9] Jinn-Moon Yang, Ying-Ping Chen, Jorng-Tzong Horng, and Cheng-Yan Kao. Applying family competition to evolution strategies for constrained optimization. In *Evolutionary Programming VI: 6th International Conference, EP97 Indianapolis, Indiana, USA, April* 13–16, 1997 *Proceedings* 6, pages 201–211. Springer, 1997.

[10] Konstantinos E Parsopoulos, Michael N Vrahatis, et al. Particle swarm optimization method for constrained optimization problems. *Intelligent technologies–theory and application: New trends in intelligent technologies*, 76(1):214–220, 2002.

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

[12] Snezana S Djordjevic. Some unconstrained optimization methods. *Applied mathematics*, pages 1–29, 2019.

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

[14] Stephen Cook. The p versus np problem. *Clay Mathematics Institute*, 2:6, 2000.

[15] Xin-She Yang. Optimization and metaheuristic algorithms in engineering. *Metaheuristics in water, geotechnical and transport engineering*, 1:23, 2013.

[16] Slawomir Koziel and Xin-She Yang. *Computational optimization, methods and algorithms*, volume 356. Springer, 2011.

[17] Paola Festa. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In 2014 16*th International Conference on Transparent Optical Networks (ICTON)*, pages 1–20. IEEE, 2014.

[18] Karla L Hoffman and Ted K Ralphs. Integer and combinatorial optimization. *Encyclopedia of Operations Research and Management Science*, pages 771–783, 2013.

[19] Wang Juhui. Analysis and optimization of enumeration method in ancient classical problem.

[20] Dingzhu Du, Panos M Pardalos, Xiaodong Hu, and Weili Wu. *Introduction to Combinatorial Optimization*. Springer, 2022.

[21] Kiavash Kianfar. Branch-and-bound algorithms. *Wiley encyclopedia of operations research and management science*, 2010.

[22] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[23] Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.

[24] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[25] Andrew R Barron, Albert Cohen, Wolfgang Dahmen, and Ronald A DeVore. Approximation and learning by greedy algorithms. 2008.

[26] Éric D Taillard. *Design of heuristic algorithms for hard optimization: with python codes for the travelling salesman problem*. Springer Nature, 2023.

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

[28] Kanchan Rajwar, Kusum Deep, and Swagatam Das. An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artificial Intelligence Review*, 56(11):13187–13257, 2023.

[29] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

[30] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

[31] Michael A Lones. Metaheuristics in nature-inspired algorithms. In *Proceedings of the Companion Publication of the* 2014 *Annual Conference on Genetic and Evolutionary Computation*, pages 1419–1422, 2014.

[32] Xin-She Yang, Suash Deb, and Simon Fong. Metaheuristic algorithms: optimal balance of intensification and diversification. *Applied Mathematics & Information Sciences*, 8(3):977, 2014.

[33] Xin-She Yang, TO Ting, and Mehmet Karamanoglu. Random walks, lévy flights, markov chains and metaheuristic optimization. *Future Information Communication Technology and Applications: ICFICE 2013*, pages 1055–1064, 2013.

[34] Mridul Chawla and Manoj Duhan. Levy flights in metaheuristics optimization algorithms–a review. *Applied Artificial Intelligence*, 32(9-10):802–821, 2018.

[35] Xin-She Yang. Metaheuristic optimization: algorithm analysis and open problems. In *International symposium on experimental algorithms*, pages 21–32. Springer, 2011.

[36] Amir Hossein Gandomi, Xin-She Yang, Siamak Talatahari, and Amir Hossein Alavi. Metaheuristic algorithms in modeling and optimization. *Metaheuristic applications in structures and infrastructures*, 1:1–24, 2013.

[37] Reda Mohamed, Mohamed Abdel-Basset, Karam M Sallam, Ibrahim M Hezam, Ahmad M Alshamrani, and Ibrahim A Hameed. Novel hybrid kepler optimization algorithm for parameter estimation of photovoltaic modules. *Scientific Reports*, 14(1):3453, 2024.

[38] Mohamed Ghetas and Motasem Elshourbagy. Parameters extraction of photovoltaic models using enhanced generalized normal distribution optimization with neighborhood search. *Neural Computing and Applications*, pages 1–18, 2024.

[39] Serdar Ekinci, Davut Izci, and Abdelazim G Hussien. Comparative analysis of the hybrid gazelle-nelder–mead algorithm for parameter extraction and optimization of solar photovoltaic systems. *IET Renewable Power Generation*, 2024.

[40] Nandhini Kullampalayam Murugaiyan, Kumar Chandrasekaran, Premkumar Manoharan, and Bizuwork Derebew. Leveraging opposition-based learning for solar photovoltaic model parameter estimation with exponential distribution optimization algorithm. *Scientific Reports*, 14(1):528, 2024.

[41] Thira Jearsiripongkul, Pradya Prempraneerach, Mahdiyeh Eslami, and Mohammad Amin Moarrefi. A novel hybrid metaheuristic approach to parameter estimation of photovoltaic solar cells and modules. *Engineered Science*, 27:979, 2023.

[42] Mohamed Issa, Ahmed M Helmi, and Mohamed Ghetas. Estimation of solar cell parameters through utilization of adaptive sine–cosine particle swarm optimization algorithm. *Neural Computing and Applications*, pages 1–17, 2024.

[43] Zulfiqar Ali Memon, Mohammad Amin Akbari, and Mohsen Zare. An improved cheetah optimizer for accurate and reliable estimation of unknown parameters in photovoltaic cell and module models. *Applied Sciences*, 13(18):9997, 2023.

[44] Houssem Ben Aribia, Ali M El-Rifaie, Mohamed A Tolba, Abdullah Shaheen, Ghareeb Moustafa, Fahmi Elsayed, and Mostafa Elshahed. Growth optimizer for parameter identification of solar photovoltaic cells and modules. *Sustainability*, 15(10):7896, 2023.

[45] Ting-ting Zhou and Chao Shang. Parameter identification of solar photovoltaic models by multi strategy sine–cosine algorithm. *Energy Science & Engineering*, 2024.

[46] Rabeh Abbassi, Salem Saidi, Shabana Urooj, Bilal Naji Alhasnawi, Mohamad A Alawad, and Manoharan Premkumar. An accurate metaheuristic mountain gazelle optimizer for parameter estimation of single- and double-diode photovoltaic cell models. *Mathematics*, 11(22):4565, 2023.

[47] Mohammad Dehghani, Eva Trojovská, Pavel Trojovskỳ, and Om Parkash Malik. Oobo: A new meta-heuristic algorithm for solving optimization problems. *Biomimetics*, 8(6):468, 2023.

[48] Amir Seyyedabbasi and Farzad Kiani. Sand cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. *Engineering with Computers*, 39(4):2627–2651, 2023.

[49] Jiankai Xue and Bo Shen. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *The Journal of Supercomputing*, 79(7):7305–7336, 2023.

[50] Seydali Ferahtia, Azeddine Houari, Hegazy Rezk, Ali Djerioui, Mohamed Machmoum, Saad Motahhir, and Mourad Ait-Ahmed. Red-tailed hawk algorithm for numerical optimization and real-world problems. *Scientific Reports*, 13(1):12950, 2023.

[51] Ahmed A Zaki Diab, Hamdy M Sultan, Ton Duc Do, Omar Makram Kamel, and Mahmoud A Mossa. Coyote optimization algorithm for parameters estimation of various models of solar cells and pv modules. *Ieee Access*, 8:111102–111140, 2020.

[52] Abha Singh, Abhishek Sharma, Shailendra Rajput, Amarnath Bose, and Xinghao Hu. An investigation on hybrid particle swarm optimization algorithms for parameter optimization of pv cells. *Electronics*, 11(6):909, 2022.

[53] 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, 2020.