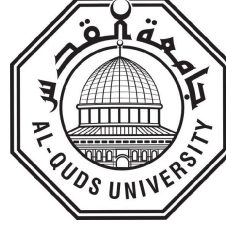


**ADVANCING METAHEURISTIC ALGORITHMS:  
INNOVATIVE OPERATORS AND ADAPTIVE  
COOPERATIVE ISLAND MODEL FOR EFFECTIVE  
OPTIMIZATION**



# **ADVANCING METAHEURISTIC ALGORITHMS: INNOVATIVE OPERATORS AND ADAPTIVE COOPERATIVE ISLAND MODEL FOR EFFECTIVE OPTIMIZATION**

By:

Thaer Ahmad Thaher

Supervisor:

Prof. Mohammed Awad

Co-Supervisors:

Prof. Alaa Sheta

Dr. Mohammed Aldasht

This Dissertation proposal was submitted in Partial Fulfillment of the  
Requirements for the PhD

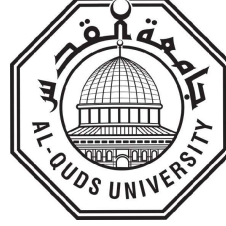
In Engineering of Information Technology

Joint PhD Program- AAUP, PPU and QU

Deanship of Scientific Research and Graduate Studies

Ramallah - Palestine

[July] [2024]



Joint PhD Program- AAUP, PPU and QU

[July] [2024]

**ADVANCING METAHEURISTIC ALGORITHMS:  
INNOVATIVE OPERATORS AND ADATIVE  
COOPERATIVE ISLAND MODEL FOR EFFECTIVE  
OPTIMIZATIO**

By:

Thaer Ahmad Thaher

Signature of Author

Committee Members

Signature and Date

Prof. Mohammed Awad (Chairman)

30/9/2024

Prof. Alaa Sheta (Co-Advisor)

1/10/2024

Dr. Mohammed Aldasht (Co-Advisor)

3/10/2024

Dr. Rashid Jayousi (Member)

30/9/2024

Prof. Ali Mohamed (External Examiner)

1/10/2024



Dr. Baker Abdalhaq (External Examiner)

19/10/2024

[July] [2024]

## تفويض

أنا الموقع أدناه، أتعهد بمنح الجامعات الشريكة في برنامج الدكتوراه في هندسة تكنولوجيا المعلومات حرية التصرف في نشر محتوى الأطروحة الجامعية، بحيث تعود حقوق الملكية الفكرية للأطروحة إلى الجامعات وفق القوانين والأنظمة والتعليمات المتعلقة بالملكية الفكرية وبراءة الاختراع.

المشرف الرئيس	الطالب
أ.د. محمد موسى عوض	ثائر أحمد ظاهر
التوقيع والتاريخ	الرقم الجامعي والتوقيع
.....  .....	.....20191295.....  .....
.....24/09/2024.....	

# **DEDICATION**

## **Declaration**

I declare that this dissertation is the result of my own independent research and has not been submitted, in whole or in part, for any other degree or academic qualification at any other institution. All sources of information and assistance have been duly acknowledged.

## **Dedication**

I dedicate this work to my beloved family and parents, whose endless love, guidance, and sacrifices have been my strongest support. Your wisdom, patience, and unwavering faith in my capabilities have not only shaped who I am but have also been the foundation of my every achievement. This accomplishment is a testament to the love and support you have all provided me.

## **ACKNOWLEDGMENTS**

My deepest gratitude goes to everyone who contributed to the journey and completion of this dissertation. First and foremost, I extend my deepest thanks to my supervisors, Prof. Mohammed Awad, Prof. Alaa Sheta, and Dr. Mohammed Aldasht, whose expertise, understanding, and patience added considerably to my graduate experience.

I would also like to express my gratitude to the professors of the PhD program, with special thanks to Dr. Badie Sartawi. His profound knowledge and dedicated mentorship profoundly influenced my scholarly development and inspired my research directions. The lessons learned under his tutelage have been invaluable and will continue to guide my future endeavors.

# TABLE OF CONTENTS

DEDICATION	i
ACKNOWLEDGMENTS	iii
LIST OF FIGURES	vi
LIST OF TABLES	x
LIST OF APPENDICES	xiv
LIST OF ABBREVIATIONS	xv
ABSTRACT	xvii
Chapter One: Introduction	1
1.1 Overview	1
1.2 Problem Statement	4
1.3 Research Objectives	7
1.4 Research Questions	8
1.5 Significance of the Study	9
1.6 Dissertation Organization	10
Chapter Two: Background and Literature Review	12
2.1 Metaheuristic Algorithms: An Overview	12
2.2 Parallel Approaches in MHs	19
2.3 Fundamentals of Cooperative Island-Based Model	25
2.4 Literature Review	31
Chapter Three: Methodology	63
3.1 Research Design	64
3.2 Data Collection	89
3.3 Data Analysis	90



Chapter Four: Results	93
4.1 Common Experimental Setup . . . . .	93
4.2 Experimental Results and Simulations on Mathematical Benchmarks . . . . .	96
4.3 Experimental Results of CSA . . . . .	100
4.4 Experimental Results of CapSA . . . . .	127
4.5 Impact of the Adaptive Island Migration Policy . . . . .	138
4.6 Experiments and Results of Real-World Applications . . . . .	142
Chapter Five: Discussion	187
5.1 Discussion of Results . . . . .	187
5.2 Implications . . . . .	190
5.3 Limitations . . . . .	191
Chapter Six: Conclusion and Future Work	194
6.1 Summary and Conclusion. . . . .	194
6.2 Recommendations and Future Work . . . . .	195
References	197
Appendices	228
الملخص	234

# LIST OF FIGURES

<b><u>Figure</u></b>	<b><u>Description</u></b>	<b><u>Page</u></b>
1.1	Visual roadmap of dissertation structure	11
2.1	A visualization of exploration and exploitation in MHs	14
2.2	Classification of MHs by the number of search agents, including representative examples for each category	15
2.3	Classification of MHs by the source of inspiration, including representative examples for each category	18
2.4	Representation of parallel design models of meta-heuristics	22
2.5	Migration process among algorithms using random ring topology	27
2.6	Example of population segmentation in an island model framework  ( $N = 12, s = 4, I_s = 3$ )	30
2.7	The general process diagram of the island-based parallel mechanistic	30
2.8	The schematic Pursuit behavior in CSA	34
2.9	Flowchart of the standard CSA	35
2.10	Number of island-based MHs publications per year	60
2.11	Number of publications per each algorithm combined with island based parallel model	60
2.12	Distribution of used migration strategies	61
3.1	Overview of research methodology steps	63
3.2	Trend of control parameter CP over 1000 iterations	74
3.3	The general process diagram of the proposed iECSA	80
4.1	Representative visualizations of mathematical benchmark functions	98

4.2	Comparison of convergence curves of CSA and proposed variants obtained in some of the benchmark problems	102
4.3	Comparison of diversity curves of CSA and proposed variants obtained in some of the benchmark problems	104
4.4	Box-plots of CSA, ECSA, IECSA on some of the standard benchmarks	107
4.5	Comparison of Convergence Curves for CSA, ECSA, and iECSA on some of the standard benchmark problems	108
4.6	Friedman mean rank based on results in Table 5.4	109
4.7	Scalability results of the CSA variants in dealing with F1–F13 functions with different dimensions	110
4.8	The convergence and diversity plots of iECSA using different values of population size	113
4.9	The convergence and diversity plots of iECSA using different values of migration frequency	115
4.10	The convergence and diversity plots of iECSA using different values of migration rate	117
4.11	Schematic view of the results of Friedman rank test (standard and CECE2014 suites) based on results in Tables 5.11 and 5.12	124
4.12	The convergence and diversity plots of top algorithms on some standard benchmarks	125
4.13	The convergence and diversity plots of top algorithms on some CEC2014 benchmarks	126

4.14	Convergence and diversity curves for the standard CapSA and its enhanced variants on standard test function F5 and sampled IEEE CEC2014 test functions (F3, F9, and F16)	131
4.15	Convergence and diversity curves for the standard CapSA and its enhanced variants on sampled IEEE CEC2014 test functions (F18, F21, F26, and F29)	132
4.16	Comparative analysis of convergence and diversity curves for CapSA, ECapSA, and iECapSA on standard test function F5 and sampled IEEE CEC2014 test functions (F3, F9, and F16)	136
4.17	Comparative analysis of convergence and diversity curves for CapSA, ECapSA, and iECapSA on sampled IEEE CEC2014 test functions (F18, F21, F26, and F29)	137
4.18	Mean rank comparison of iECSA and iECapSA with static and adaptive migration policies based on results in Tables 5.20 and 5.21	141
4.19	Simple FNN architecture with one hidden layer	143
4.20	Solution representation for MHs-Based training of FNN	145
4.21	Flowchart of optimizing FNN with the cooperative island-based model	146
4.22	MSE convergence curves for basic and enhanced CSA and CapSA models across various biomedical datasets	152
4.23	Flowchart of the multilevel image segmentation framework using cooperative island-based optimization model	159
4.24	Set of tested COVID-19 CT images and their histograms	162
4.25	Variation of cross entropy with increasing number of thresholds for iECapSA on COVID-CT1 and COVID-CT2 test images	166

4.26	Convergence trends for CSA and CapSA variants on COVID-CT	167
	Images	
4.27	Graphical representation of the selected SRGMs	172
4.28	Overall approach of SRGM optimization	173
4.29	MSE convergence trends for CSA and CapSA variants on Delayed S-	179
	Shaped SRGM	
4.30	Visualization of MSE and VAF across different datasets and SRGMs	181
	using CSA, ECSA, CapSA, ECapSA, iECSA, and iECapSA	
4.31	Comparison of actual vs. iECSA-estimated failures across selected	183
	datasets for three SRGMs	
4.32	Actual vs. predicted cumulative faults and correlation scatter plots for	184
	the Delayed S-shaped SRGM estimated by iECSA	

## LIST OF TABLES

<b><u>Table</u></b>	<b><u>Description</u></b>	<b><u>Page</u></b>
2.1	Overview of MHs based on the source of inspiration	20
2.2	Design models of parallel metaheuristics	22
2.3	Summary of recent advances in the CSA and their applications across various fields	46
2.4	Summary of recent studies utilizing CapSA for various applications	52
2.5	Summary of main previous works that incorporated island model with MHs for optimizing continuous benchmark functions	56
2.6	Summary of main island-based metaheuristics designed for specific real-world applications	59
3.1	Summary of enhanced variants of CapSA and their implications	75
3.2	Time Complexity Analysis of each step in the proposed iECSA	82
3.3	Summary of Datasets and Benchmarks Used	90
3.4	Summary of Evaluation Metrics Across Test Domains	91
4.1	Parameter settings of iECSA and other algorithms	95
4.2	Investigated variants of CSA	100
4.3	Results of CSA variants on the standard 30-dimensional unimodal and multimodal test problems	101
4.4	Comparison of CSA variants and cooperative island-based variants for the standard 30-dimensional functions	106

4.5	Results of benchmark functions F1-F13 with 100 dimensions	109
4.6	Results of benchmark functions F1-F13 with 500 dimensions	111
4.7	The experimental design for evaluating the sensitivity of iECSA N, MF, and Mr parameters	111
4.8	The impact of population size on the convergence behavior of iECSA	112
4.9	The impact of migration frequency on the convergence of iECSA	114
4.10	The impact of migration rate on the convergence of iECSA	116
4.11	Comparison between iECSA and other algorithms for the unimodal, multimodal, and fixed-dimension multimodal benchmark functions	119
4.12	Comparison between iECSA and other algorithms on IEEE CEC2014 functions	121
4.13	Average rankings of the algorithms calculated using Friedman's test	123
4.14	Comparison of iECSA Performance: Wilcoxon Test Results for Superior, Inferior, and Equal Outcomes	123
4.15	Summary of enhanced variants of CapSA and their implications	127
4.16	Comparative performance of CapSA and enhanced MCapSA variants on 30-dimensional standard test functions	128
4.17	Performance comparison of standard CapSA and enhanced variants on IEEE CEC2014 30-dimensional test functions	129
4.18	Performance comparison of standard CapSA, ECapSA, and their island-based counterparts iCapSA and iECapSA on 23 Standard mathematical functions	133

4.19	Performance comparison of standard of Standard CapSA, ECapSA, and their island-based models iCapSA and iECapSA on the IEEE CEC2014 benchmark functions	134
4.20	Comparative performance of iECSA and iECapSA with static and adaptive migration policies on 30-dimensional standard mathematical functions	138
4.21	Comparative analysis of iECSA and iECapSA performance on 30-dimensional CEC2014 benchmark functions with static and adaptive migration policies	140
4.22	Description of the biomedical datasets used for evaluating the training of FNN	149
4.23	Confusion matrix for binary classification	149
4.24	MSE results for CSA and CapSA variants across biomedical datasets	150
4.25	Testing classification accuracy for CSA and CapSA variants across biomedical datasets	151
4.26	Testing classification F1-score for CSA and CapSA variants across biomedical datasets	151
4.27	Classification performance comparison between stochastic gradient descent solvers and metaheuristic models across biomedical datasets	153
4.28	Average MSE results of the proposed iECSA and iECapSA against state-of-the-art Algorithms for neural network optimization	155
4.29	Accuracy results of the proposed iECSA and iECapSA against state-of-the-art Algorithms for neural network optimization	155



4.30	Comparative Mean Fitness Values (Cross Entropy) for Basic and Enhanced Variants of CSA and CapSA at Threshold Levels 4, 6, 8, and 10	165
4.31	Comparative evaluation of SSIM scores for CSA and CapSA variants on COVIDCT images	168
4.32	Performance Comparison of iECapSA with the Cooperative Island-Based Model (CPGH) from literature [4]	168
4.33	Parameter ranges for SRGM estimation	174
4.34	Summary of datasets used for SRGM Evaluation	175
4.35	Average MSE obtained by CSA and CapSA variants for SRGM parameter estimation across diverse datasets	178
4.36	VAF results of CSA and CapSA variants for SRGM parameter estimation across 10 datasets	180
4.37	Optimal SRGM parameters estimated by iECSA across datasets with corresponding MSE and VAF	182
4.38	Comparative MSE of iECSA and other Metaheuristic Algorithms for SRGM Parameter Estimation	185
4.39	Average VAF scores of iECSA and other Metaheuristic Algorithms for SRGM Parameter Estimation	186

## LIST OF APPENDICES

<u>Appendix</u>	<u>Description</u>	<u>Page</u>
A	Characteristic Tables of Real-Valued Mathematical Functions	228
B	Statistical Results	230

## LIST OF ABBREVIATIONS

<b><u>Abbreviation</u></b>	<b><u>Description</u></b>
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AI	Artificial Intelligence
ATCSA	Adaptive Tournament Selection Based Guided CSA
BFO	Bacterial Foraging Optimization
CapSA	Capuchin Search Algorithm
CEC	Congress on Evolutionary Computation
ChOA	Chimp Optimization Algorithm
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
CS	Cuckoo Search
CSA	Crow Search Algorithm
CT	Computed tomography
DE	Differential Evolution
EAs	Evolutionary Algorithms
ECapSA	Enhanced Capuchin Search Algorithm
ECSA	Enhanced Crow Search Algorithm
FNN	Feedforward Neural Network
FPA	Flower Pollination Algorithm
FSA	Fish Swarm Algorithm
GA	Genetic Algorithm
GLS	Guided Local Search
GPUs	Graphics Processing Units
GRASP	Greedy Randomized Adaptive Search Procedure
GWO	Grey Wolf Optimizer
HHO	Harris Hawks Optimization
HS	Harmony Search
ILS	Iterated Local Search
KHA	Krill Herd Algorithm

MFO	Moth-Flame Optimization
MHs	metaheuristics
MLT	Multi-level thresholding
MPA	Marine Predators Algorithm
MRCSA	Modified Random Movement CSA
MSE	Mean Squared Error
PMHs	Population-based Metaheuristics
PSO	Particle Swarm Optimization
PyGMO	Python Parallel Global Multiobjective
RAM	Random Access Memory
SA	Simulated Annealing
SI	Swarm Intelligence
SMA	Slime Mould Algorithm
SRGMs	Software Reliability Growth Models
SSA	Squirrel Search Algorithm
SSIM	Structural Similarity Index
TMHs	Trajectory-based Metaheuristics
TS	Tabu Search
UCI	University of California Irvin
VAF	Variance Accounted For
VNS	Variable Neighborhood Search
WHO	Wild Horse Optimizer
WOA	Whale Optimization Algorithm

# ABSTRACT

## **Advancing Metaheuristic Algorithms: Innovative Operators and Adaptive Cooperative Island Model for Effective Optimization**

By

**Thaer Ahmad Thaher**

Swarm intelligence algorithms are renowned for their ability to tackle complex global optimization problems by mimicking natural processes. However, these algorithms, including the Crow Search Algorithm (CSA) and Capuchin Search Algorithm (CapSA), often suffer from inherent limitations such as low search accuracy and a tendency to converge to local optima. This thesis aims to develop advanced variants of these algorithms that could effectively handle a diverse range of theoretical and practical optimization problems. One widely explored approach is the structured population mechanism, which maintains diversity during the search process to mitigate premature convergence. The island model, a common structured population method, divides the population into smaller independent sub-populations called islands, each running in parallel. Migration, the primary technique for promoting population diversity, facilitates the exchange of relevant and useful information between islands during iterations. Enhancements to CSA and CapSA include introducing adaptive strategies and novel operators, enhancing them into variants named ECSA and ECapSA, respectively. Furthermore, it integrates these enhancements within an island-based model -iECSA and iECapSA- equipped with an adaptive migration policy designed to dynamically adjust migration rates based on real-time evaluations of population diversity and fitness. This innovative approach aims to avoid the limitations of traditional island models and enhance global search capabilities. The performance of the proposed models is evaluated using 53 real-valued mathematical problems and three practical applications: neural network training, multilevel thresholding for image segmentation, and software reliability growth modeling. It is also validated by conducting an extensive evaluation against a comprehensive set of well-established and recently introduced meta-heuristic algorithms. Experimental results demonstrate that the enhanced variants of CSA and CapSA outperform their fundamental counterparts in the majority of test cases, providing more accurate and reliable outcomes. Furthermore, extensive experimentation consistently showcases that the iCapSA outperforms its comparable algorithms across a diverse set of practical applications.

# Chapter One: Introduction

---

## 1.1 Overview

Optimization is the process of finding the best (optimal) solution among different options while considering various constraints and conditions [1]. It is a crucial procedure that is extensively utilized in the scientific and engineering domains. Essentially, optimization addresses the challenge of planning, designing, and operating systems in ways that are both efficient and effective, thereby ensuring the attainment of superior outcomes. This involves identifying the optimal solutions that fulfill specific criteria to either maximize or minimize a defined objective function [2].

Optimization theory is a specialized branch of mathematics and computational sciences that focuses on identifying and analyzing the best possible solutions to specific problems by utilizing various techniques and algorithms [3]. It is widely applied across diverse fields such as engineering, physics, economics, and biomedicine [4]. Furthermore, in the realm of emerging technologies, it plays a critical role in the development and enhancement of Artificial Intelligence (AI) systems. It is extensively used in refining machine learning algorithms and enhancing AI's decision-making processes [5, 6, 7, 8, 9, 10]. Optimization methods are broadly classified into three categories: analytical, graphical, and numerical [3]. Numerical methods are the most widely used and effective, as they use iterative operations to refine solutions from an initial estimate until a predefined convergence criterion is met. This approach is particularly useful for handling intricate problems that cannot be solved using analytical or graphical methods [1].

Numerical optimization techniques are generally categorized into two main families: exact methods and approximate methods [11]. These methods are designed to address objective functions, which can be categorized as single-objective or multi-objective [12] or many-objective [13]. Exact methods are designed to find an optimal solution with certainty, systematically exploring the solution space based on predefined rules.

Examples of exact methods include linear programming, dynamic programming, and the branch and X family of algorithms (such as branch and bound, branch and cut, and branch and price) [14]. These methods may be viewed as tree search algorithms, where the search is carried out over the entire relevant search space, and the problem is solved by subdividing it into simpler problems. In contrast, approximate methods are used when finding an exact solution is impractical due to the complexity or size of the problem. These methods focus on finding a good enough solution within a reasonable timeframe and are particularly valuable for large-scale problems where an exhaustive search is not feasible. Approximate methods are commonly divided into three categories: specific-problem heuristics, metaheuristics, and hyper-heuristics. Specific-problem heuristics are tailored and designed to solve particular problems or specific instances. Metaheuristics are general-purpose algorithms that can be applied to a wide range of optimization problems. They are upper-level methodologies that can guide the design of underlying heuristics to solve specific optimization problems, including techniques like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [15]. Hyper-heuristics are search techniques for selecting, generating, and sequencing (meta)-heuristics to solve challenging optimization problems [16].

Combinatorial optimization problems, including scheduling, routing, and decision problems, are predominantly classified as NP-complete or NP-hard problems, where the time required to find an exact solution grows exponentially or even factorially with the size of the input [17, 18, 19]. Traditional mathematical programming techniques are inadequate for solving these problems [20, 21]. Conventional mathematical programming methodologies find themselves inadequately prepared to effectively handle the intricacy and non-linearity inherent in present-day optimization cases [22, 23]. The complex inter-dependencies and high-dimensional spaces encountered in real-world situations frequently result in sub-optimal solutions when employing traditional methods [24]. To surpass these constraints, innovative optimization methodologies, including evolutionary algorithms and metaheuristic methods, have gained prominence due to their capability to traverse intricate solution landscapes and uncover solutions that are nearly optimal [25]. These advanced techniques utilize stochastic processes and strategies based on population-based search principles, thus solidifying

their status as indispensable tools for addressing the diverse and intricate challenges posed by contemporary optimization problems [26, 27]. Stochastic optimization techniques, such as Metaheuristics (MHs), have emerged as powerful alternatives [28]. MHs are approximation algorithms that find near-optimal solutions within a reasonable computational time [29]. They are problem-independent and adaptable to various domains, including image segmentation, feature selection, machine learning optimization, fault diagnosis, economic local dispatch problems, and global optimization, which these works proposed by [30, 8, 31, 32, 33, 34, 35].

MHs follow a fundamental principle: they generate random solutions or populations and iteratively improve them using stochastic mathematical operators until a termination condition is met [36, 37, 38]. Among MHs, population-based approaches have gained popularity due to their effective performance [39]. These approaches strike a balance between global search (exploration) and local search (exploitation), reducing the effective search space by efficiently exploring promising regions [40]. Well-regarded MHs should possess this balance to avoid getting stuck in local optima [41, 42]. Recently, innovative Swarm Intelligence (SI) methods inspired by nature have emerged, incorporating well-known natural processes into numerical simulations. These algorithms effectively address problems by avoiding local minima, accelerating convergence, and providing accurate solutions [43, 44]. Examples of these algorithms include Grey Wolf Optimizer (GWO), Moth-Flame Optimization (MFO), Flower Pollination Algorithm (FPA), Krill Herd Algorithm (KHA), Squirrel Search Algorithm (SSA), Crow Search Algorithm (CSA), and Capuchin Search Algorithm (CapSA). These algorithms share common characteristics: the particles or agents are simple and non-sophisticated; they collaborate through indirect communication mediums<sup>1</sup>; and their movements in the decision space are strategically designed to mimic the natural behaviors of the species they are inspired by [11]. The CSA [45] and the CapSA [46] are recent, innovative contributions in the realm of nature-inspired algorithms, each drawing inspiration from the intelligent foraging behaviors of their respective animal counterparts. The CSA mimics the intelligent foraging behaviors of crows, while the CapSA imitates the search tactics used by

---

<sup>1</sup>Self-organization using indirect cooperation is an important issue in biological systems



capuchin monkeys in pursuit of food. Both methods demonstrate how successfully utilizing intelligent and adaptive behaviors derived from the natural world can lead to robust and effective solutions in the optimization field.

The use of MHs has progressed alongside parallelism and distributed computing. This has resulted in reduced search times and better outcomes [47]. The island model is a well-known cooperative algorithmic-level paradigm, originally designed to develop parallel Evolutionary Algorithms (EAs) [48, 49]. This model is developed based on the structured population concept where a larger population is divided into smaller, isolated sub-populations, commonly referred to as 'islands' [50]. Each of these islands evolves independently in a parallel process, typically using the same or different evolutionary or metaheuristic processes [51]. After certain intervals, the islands interact with each other through a process known as migration. This distinctive process in the cooperative island model involves exchanging selected individuals among islands. Migration introduces new variations to the populations on each island, aiding in maintaining diversity and preventing early convergence [52]. The frequency and policy of migration—such as which individuals migrate, how many migrate, and between which islands they move—are crucial parameters that can significantly impact the performance of the algorithm [53]. Because each island can be processed on a separate processor or core, the island model is particularly useful in systems that support parallel computing. The island model is a good tool for dealing with complicated optimization issues in many fields of science, engineering, and data processing. Works proposed by [54, 55, 56, 57, 53, 58, 51, 59, 60] show that it has particularly excelled at solving various optimization issues.

## **1.2 Problem Statement**

Consistently improving MHs performance across varied problem landscapes remains a fundamental challenge, despite their crucial role in tackling intractable optimization issues. These challenges include:

### 1.2.1 Challenge A: False Convergence in MHs

False convergence refers to a scenario where a metaheuristic algorithm fails to reach the global optimal solution within a predetermined maximum number of iterations or evaluations. This issue can arise when the population in a population-based search algorithm does not truly converge to the global optimum, often due to an imbalance between two crucial aspects: global exploration and local exploitation [61]. Effective optimization requires balancing exploration—the capacity to investigate new, unexplored regions of the search space—with exploitation, which focuses on refining solutions within promising regions.

False convergence can result from three primary situations [61]:

1. **Slow Convergence:** The algorithm gradually advances towards the solution, but the pace is too slow to reach the global optimum within the allowed time or iterations.
2. **Premature Convergence:** The algorithm converges to a local optimum too early. As iterations continue, MHs frequently produce suboptimal results because they lose the diversity required to thoroughly investigate potentially promising areas of the search space.
3. **Stagnation:** This occurs when the algorithm reaches the maximum number of generations without converging to a fixed point, and the population, due to a lack of exploitation tendency, is unable to produce solutions better than those already found.

Each metaheuristic algorithm has a unique set of exploration and exploitation operators, which may provide certain challenges. For instance, some algorithms may perform well in exploration but struggle to effectively focus on optimal solutions, leading to extended search times without significant improvements. Conversely, certain algorithms may converge quickly towards a solution but lack the explorative power to escape local optima, thus missing potentially better solutions. This differentiation in performance and challenges is particularly evident in the case of CSA and CapSA, which exemplify these challenges. Each has unique exploration and exploitation mechanisms that can result in an imbalance, either through excessive focus on exploration or premature convergence due to an overemphasis on exploitation.

### **1.2.2 Challenge B: Algorithmic Limitations in CSA and CapSA**

The fundamental CSA algorithm has three major flaws. Firstly, it employs fixed awareness probability and flight length values, limiting its adaptability [62]. Secondly, it adopts a single-mode searching mechanism, where each individual conducts a random search within a sector area defined by its current position, the historical ideal position (memory location) of other individuals, and their value differences. This approach restricts the crow's flying activity, reducing flexibility and mobility [63]. Thirdly, while the CSA performs well in exploration, it struggles in exploitation. The primary CSA relies on random individuals and probabilities to guide the search process, overlooking the significance of optimal solutions in population evolution. Consequently, the CSA shares flaws common to other swarm intelligence algorithms, such as premature convergence, low search accuracy, and susceptibility to local optima. These issues become particularly challenging in multi-dimensional optimization problems [63].

Similarly, the CapSA exhibits its unique limitations. Specifically, the alpha capuchin, which represents the best solution obtained so far, plays a pivotal role in generating new individuals within the basic framework of CapSA. This aspect of CapSA underscores its tendency towards a more exploitative approach. Furthermore, the followers, which represent half of the population, are repositioned based on a more exploitative operator. This mechanism ensures that a significant portion of the search effort is concentrated around the currently known best solution, intensifying the search in its vicinity. However, this approach may lead to a lack of diversity in the population, making it less practical and potentially resulting in rapid intensification and premature convergence. Consequently, to improve the performance of CapSA, it is necessary to incorporate additional operators that emphasize exploration and provide a better balance between diversification and intensification.

### **1.2.3 Challenge C: Design of Parallel MHs**

When designing parallel MHs, we should decide on the specific parallelism model and a well-established algorithm to solve the targeted problems. This choice has a significant impact on the effectiveness of the proposed model. Many MHs, with their different classifica-

tions in the literature and their continuous growth, make this choice an ongoing challenge. Despite the well-performing of recent parallel MHs, trying to design new parallel MHs and efficiently handling new classes of problems is a moving target. In general, cooperative island model techniques for MHs are distinguished depending on several factors, including the mechanism of communication and the nature of the information exchanged. These factors are problem-dependent; they should be appropriately tuned for each targeted problem. This is not applicable significantly when solving large-scale, time-consuming real problems. According to [64], the exchange of information should be timely and meaningful. Particularly, some design questions such as information exchange decision criteria (when?), the communication topology (where?), the exchanged information (what?), and the integration policy (how?) should be addressed when designing the model [47].

While existing basic island models in parallel MHs are beneficial for maintaining diversity and preventing premature convergence, they often lack the flexibility to adapt dynamically to the evolving needs of the optimization process, leading to a sub-optimal balance between exploration and exploitation. Therefore the dynamic adaptation of parameters in a reliable way is still limited. A key aspect of this dynamic adaptation is the migration policy, encompassing both the migration rate and the nature of information exchange. The migration rate, which determines the proportion of the population to be migrated between islands, is a critical parameter. For instance, a low rate may limit effective cooperation between neighboring islands, leading to a more independent operation. In contrast, a high rate could reduce diversity and raise the possibility of early convergence. The choice of whatever data to share during migration—such as problem-specific information, diversity measures, or solution quality—is equally crucial since it has a significant impact on the algorithm’s overall performance. The island model must successfully balance these two factors—the rate and content of migration—to perform well in challenging optimization tasks.

### **1.3 Research Objectives**

The main aim of this research is to develop an efficient parallel optimization model that effectively deals with optimization problems encountered in real-world applications. The primary

focus is on improving the performance of recent MHs, such as the CSA and the CapSA, by incorporating novel operators and embedding the cooperative island model principles. This integration aims to enhance the algorithms' ability to explore and exploit solutions, ensuring a more comprehensive search of the solution space. Moreover, the model will include an island framework that can dynamically adjust its strategies to handle different optimization challenges. The specific objectives are as follows:

1. Develop enhanced variants of CSA and CapSA with improved exploration and exploitation capabilities.
2. Design and introduce an adaptive island migration policy that dynamically adjusts migration rates based on real-time evaluations of population diversity and the fitness of individual islands.
3. Evaluate the effectiveness of the proposed models across a diverse set of optimization problems, including both theoretical real-valued benchmark functions and practical real-world applications.

#### **1.4 Research Questions**

Based on the objectives outlined earlier, this study is structured around several research questions. By addressing these questions, the research aims to bridge theoretical knowledge with practical efficacy, offering new perspectives and advancements in the optimization domain. The following questions have been carefully developed to guide this investigation:

1. How can the exploration and exploitation capabilities of CSA and CapSA be enhanced through the integration of novel operators?
2. What is the impact of implementing an adaptive migration policy on the performance of cooperative island models in metaheuristic optimization?
3. What are the practical implications and performance outcomes of applying the proposed models to real-world problems?

## 1.5 Significance of the Study

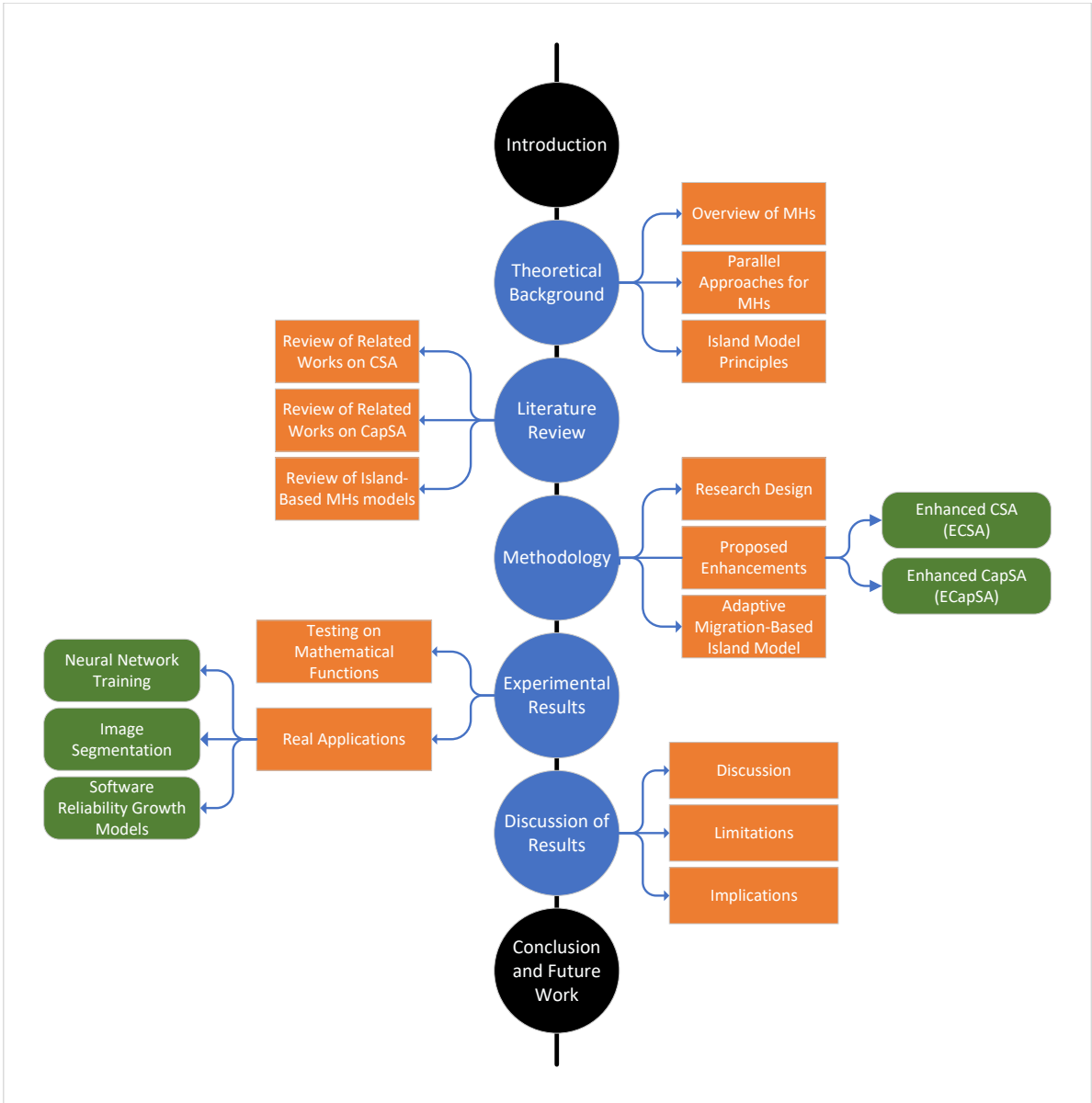
This study plays a pivotal role in advancing the field of optimization algorithms by significantly enhancing the capabilities of recent metaheuristic algorithms, particularly the CSA and the CapSA. To make these algorithms more effective, the researcher has incorporated novel operators that boost both exploitation and exploration potential, which are crucial for successful optimization. A notable innovation in this thesis is the integration of these enhanced algorithms within the cooperative island model framework. The significance of the research is further underscored by the introduction of a novel adaptive migration policy. This policy dynamically adjusts the migration rate during the optimization process by taking into account the population diversity and the fitness of each island. Such adaptability ensures a more efficient search process and better optimization outcomes.

The efficacy of these proposed models has been extensively tested through a diverse set of optimization problems, including real-valued mathematical benchmark functions with varying characteristics. Initial evaluations have demonstrated the robustness and versatility of the proposed enhancements. To highlight the practical applicability of the research more prominently, the researcher has included specific examples of how the models have been applied to real-world problems. These applications include the training of neural networks for classification, multilevel thresholding in image segmentation, and parameter optimization in Software Reliability Growth Models (SRGMs). The results of these applications have shown remarkable improvement and outperformed their traditional counterparts.

In conclusion, this thesis contributes significantly to the advancement of metaheuristic optimization algorithms. By incorporating novel operators, an adaptive migration policy, and a cooperative island model, we have developed models that are not only theoretically sound but also practically applicable. The results of this study have demonstrated the value of these models and their potential to outperform recent algorithms in most case studies.

## 1.6 Dissertation Organization

The dissertation is divided into six chapters, including the introduction. The remaining chapters are organized as follows: Chapter 2 provides a theoretical background on MHs and discusses parallel approaches for MHs, with a particular focus on the cooperative island model principles and their integration with MHs. In addition, it encompasses a thorough review of related works on CSA, CapSA, and island-based MHs models. Chapter 3 explains the methodologies used to enhance CSA, CapSA, and introduces the novel adaptive migration policy. Chapter 4 presents experimental results for the proposed algorithms, tested on mathematical real-valued functions and practical applications, including how to adapt the proposed optimization model for addressing these practical applications. Following Chapter 4, Chapter 5 is dedicated to the discussion of the results obtained. This chapter provides an in-depth discussion and interpretation of the experimental outcomes, acknowledges limitations, and offers insights into the implications of the research findings. Finally, Chapter 6 concludes the study by summarizing the findings, offering recommendations, and suggesting further exploration in the field. A visual roadmap in Figure 1.1 illustrates the structure of the thesis and outlines the flow and key components of each chapter.



**Figure 1.1:** Visual roadmap of dissertation structure



## Chapter Two: Background and Literature Review

---

To provide a solid basis for a thorough investigation in this thesis, the following structure has been formulated to clarify the techniques and models that support the suggested optimization strategies. An introduction to MHs is given in Section 2.1, emphasizing their significance in solving optimization problems and outlining their role in computational problem-solving. Section 2.2 presents the parallel approaches used in the development of parallel MHs. Finally, Section 2.3 explains the principles of the cooperative island model, outlining its structure, utility, and application in enhancing algorithmic performance.

### 2.1 Metaheuristic Algorithms: An Overview

The applications of optimization are vast and varied. In fact, many complex problems in real-life situations, such as those in science, industry, and engineering, can be formulated as optimization problems. However, most of these problems are intractable and cannot be solved efficiently using exact methods. Therefore, approximate algorithms have emerged as a promising alternative to handle these kinds of problems [11]. Approximate algorithms are generally categorized into two groups: specific heuristics and MHs. Specific heuristics are designed to tackle particular problems and are thus problem-dependent. In contrast, MHs are problem-independent (or general-purpose) approximate algorithms developed to handle a wide variety of optimization problems, especially those of a combinatorial nature [65].

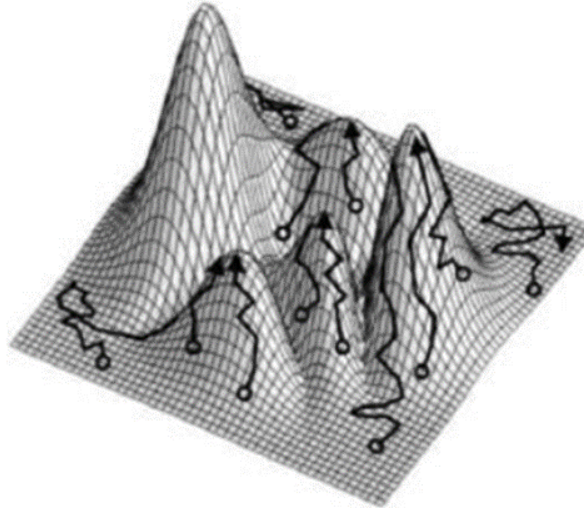
MHs are a class of stochastic optimization methods that employ a degree of randomness to find a high-quality, near-optimal solution within an acceptable timeframe. According to [66], MHs are search algorithms that coordinate the interaction between local improvement procedures and higher-level strategies to form an efficient search process. These algorithms can escape local optima and conduct a robust global search, thereby reducing the effective size of the search space by exploring promising regions effectively. As a result, MHs contribute to three main goals: handling problems more rapidly, managing large-scale problems, and developing powerful (robust) algorithms [11].

### 2.1.1 Exploration and Exploitation Mechanisms

Exploration and exploitation are two fundamental mechanisms in MHs that balance the process of discovering new solutions (exploration) and optimizing known solutions (exploitation). Effective MHs manage a trade-off between these two strategies to efficiently navigate the solution space and avoid premature convergence to suboptimal solutions [67].

Exploration refers to the algorithm's ability to investigate diverse areas of the search space, thereby reducing the risk of missing the global optimum. It's essential for identifying promising regions that haven't been visited before. For instance, a random walk introduces randomness in solution selection, allowing the algorithm to explore new areas of the solution space. This is evident in Simulated Annealing (SA), where the probability of accepting worse solutions decreases over time but initially allows for broad exploration [68]. On the other hand, exploitation involves intensively searching around the current best solutions to refine and improve them. It's crucial for fine-tuning solutions and converging to the optimum in a reasonable time. For instance, the crossover operator in GA combines parts of two good solutions (parents) to produce potentially better offspring. This operator focuses on areas of the search space where good solutions have already been found [69]. Figure 2.1 depicts the exploratory and exploitative phases that an algorithm undergoes when attempting to solve complex optimization problems with multiple local and global optima. The landscape depicted in the figure represents the solution space, and the algorithm explores it by sampling various points and avoiding local optima. Once the algorithm identifies the most promising areas, it intensively searches these regions to pinpoint the optimal solution, as indicated by the paths converging on high peaks.

It is essential to strike a balance between exploration and exploitation to ensure the effectiveness and efficiency of MHs across different optimization problems. An overemphasis on exploration can lead to an inefficient search process, while an excessive focus on exploitation can trap the algorithm in local optima. Empirical studies have established a strong correlation between the exploration-exploitation dynamics of a search method and its convergence rate [70]. Methods that prioritize exploitation can accelerate convergence towards the global optimum but may become stuck in local optima. In contrast, methods that prioritize ex-



**Figure 2.1:** A visualization of exploration and exploitation in MHs [15]

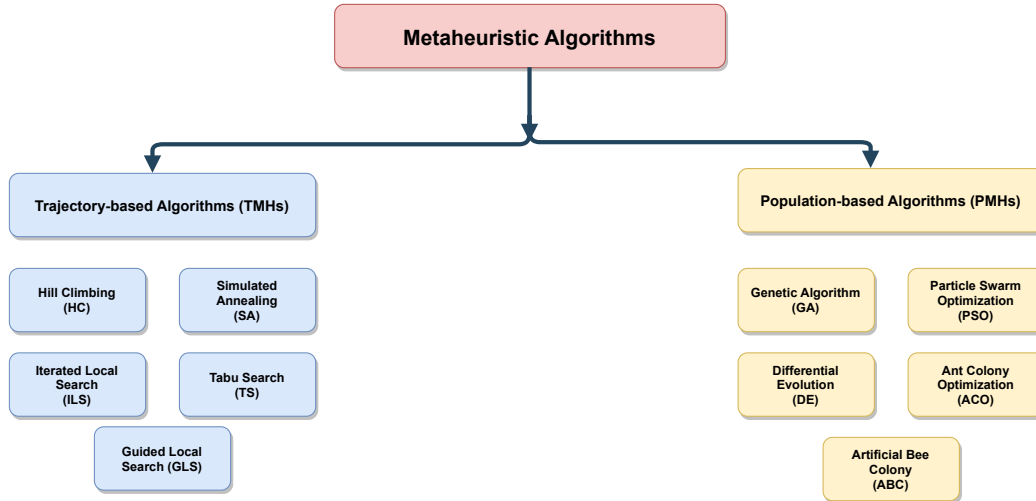
ploration are more likely to discover regions containing the global optimum, although at a slower convergence rate. Therefore, finding a balance between exploration and exploitation is critical to guide the search process and achieve a thorough exploration while focusing on exploitation to converge towards the global optimum [71]. However, addressing the exploration and exploitation potentials of MHs has been a subject of debate among researchers in recent years [70]. Determining the optimal rates of exploration and exploitation for an efficient search process is challenging due to the varied search strategies employed by various metaheuristic methods. Thus, designing an efficient search strategy requires a comprehensive understanding of the mechanisms utilized by these methods and how they contribute to the search process.

### 2.1.2 Classification of MHs

There are various classifications for MHs in the literature [44]. One way is to categorize them based on the number of search agents used in each iteration. Another way is to classify them based on their source of inspiration, which looks at the origins and conceptual foundations of each algorithm. These categories will be examined in detail in subsequent sections.

### 2.1.2.1 Classification by Number of Search Agents

Within this framework, MHs are primarily classified into two distinct types: trajectory-based and population-based algorithms [72], as illustrated in Figure 2.2. This classification depends on the number of temporary solutions used in each step of the iterative algorithm.



**Figure 2.2:** Classification of MHs by the number of search agents, including representative examples for each category

#### A) Trajectory-Based MHs

Trajectory-based Metaheuristics (TMHs), also known as single-solution-based metaheuristics, are commonly referred to as local search algorithms. This class of algorithms begins with an initial candidate solution and iteratively improves it while addressing optimization problems. They can be conceptualized as 'search trajectories' moving through neighborhoods in the problem's search area, employing iterative heuristic procedures [11]. Essentially, two main procedures are applied to the current solution: generation and selection. In the generation phase, a local search operator is applied to the current solution to generate a set of solutions ( $S$ ), while in the subsequent selection phase, the current solution is replaced by a better-quality solution chosen from the generated set  $S$ . The pseudocode for a general TMHs approach targeting a minimization problem is illustrated in Algorithm 1.

TMHs are prone to the challenge of premature convergence. This occurs when the algorithm settles for a suboptimal solution too early without exploring more promising regions

---

**Algorithm 1** Pseudocode of generic Trajectory-Based Metaheuristic

---

```
1: Input: ObjectiveFunction, MaxIterations, InitialSolution
2: Output: BestSolution
3: BestSolution  $\leftarrow$  InitialSolution
4: CurrentSolution  $\leftarrow$  InitialSolution
5: Iteration  $\leftarrow$  0
6: while Iteration < MaxIterations do
7:   NeighboringSolutions  $\leftarrow$  GenerateNeighbors(CurrentSolution)
8:   CandidateSolution  $\leftarrow$  SelectFrom(NeighboringSolutions)
9:   if AcceptanceCriteria(CandidateSolution, CurrentSolution) then
10:    CurrentSolution  $\leftarrow$  CandidateSolution
11:    if ObjectiveFunction(CurrentSolution) < ObjectiveFunction(BestSolution) then
12:     BestSolution  $\leftarrow$  CurrentSolution
13:   Iteration  $\leftarrow$  Iteration + 1
14: Return BestSolution
```

---

of the search space. This limitation often stems from the nature of TMHs relying on a single solution path, which can lead to a narrow search focus and make it difficult for these algorithms to escape local optima. Moreover, TMHs might have trouble navigating very complicated or multi-modal landscapes where there are multiple local optima. This could lead to less stable performance, especially when problems need a lot of exploration to find the global optimal. Additionally, the initial solution can significantly affect the efficiency of TMHs. A poor starting point might lead the algorithm towards less optimal regions, thereby impacting the overall solution quality.

Common examples of TMHs include Tabu Search (TS) [73], SA [68], Variable Neighborhood Search (VNS) [74], Greedy Randomized Adaptive Search Procedure (GRASP) [75], Iterated Local Search (ILS) [76], and Guided Local Search (GLS) [77]. Among these trajectory algorithms, SA and TS are the most widely used in the literature [72].

## B) Population-Based MHs

Population-based Metaheuristics (PMHs) are exploration-oriented algorithms, renowned for their superior exploratory potential compared to trajectory algorithms. These algorithms operate by generating and evolving a set of candidate solutions, known as the population. The process continues iteratively until satisfactory results are achieved. One of the distinguishing characteristics of PMHs is their ability to balance exploration and exploitation components. This balance is crucial when dealing with complex problems as it helps to discover new and

potentially promising areas of the search space while intensifying the search around already promising regions. The high-level procedure of PMHs is illustrated in Algorithm 2.

---

**Algorithm 2** Pseudocode of generic Population-Based Metaheuristic

---

```

1: Input: ObjectiveFunction, MaxGenerations, PopulationSize
2: Output: BestSolution
3: Initialize a population of solutions of size PopulationSize
4: Evaluate each solution in the initial population using ObjectiveFunction
5: BestSolution  $\leftarrow$  best solution from the initial evaluation
6: Generation  $\leftarrow$  0
7: while Generation < MaxGenerations do
8:   Generate New Solutions:
9:     Apply operators (e.g., modification, combination) to create new solutions
10:  Evaluate New Solutions:
11:    Evaluate each new solution using ObjectiveFunction
12:  Selection:
13:    Select solutions for the next generation based on fitness
14:  Update Best Solution:
15:    Update BestSolution if a new solution is better
16:  Generation  $\leftarrow$  Generation + 1
17: Return BestSolution

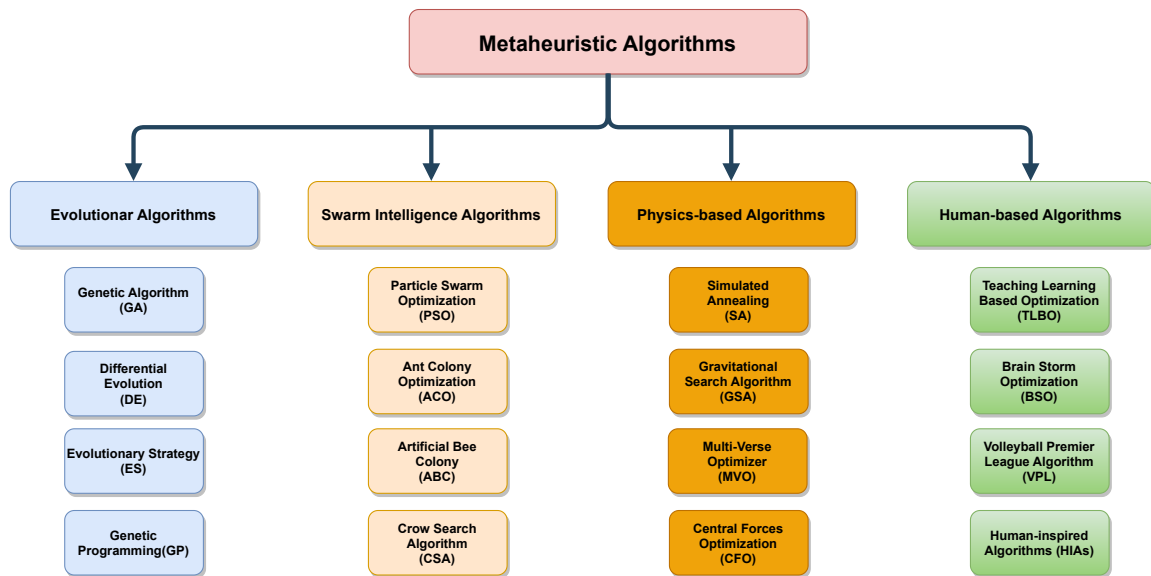
```

---

### 2.1.2.2 Classification by Source of Inspiration

The categorization of MHs based on their source of inspiration is one of the most beneficial and oldest classifications in the field of computational optimization [44]. This concept, which is primarily rooted in natural or biological phenomena, has led to the development of several nature-inspired algorithms. Various authors have categorized optimization algorithms based on their source of inspiration [78, 79]. One common approach divides algorithms into four categories: evolutionary-based, swarm intelligence-based, physics-based, and human-based [80]. Evolutionary-based algorithms draw inspiration from principles of genetics and natural evolution, while swarm intelligence algorithms are inspired by the group behavior of social organisms. Physics-based algorithms utilize concepts and laws from physics, and human-based algorithms are derived from human social and cognitive behaviors [81]. These diverse sources of inspiration demonstrate the adaptability and effectiveness of these algorithms in solving complex optimization problems. They also underscore the creativity and richness of the field of MHs. The various categories of algorithms have been extensively researched, each having its own distinct approach. However, the main focus of this thesis will be on

swarm intelligence-based algorithms that rely on the decision-making abilities of groups like flocks of crows or groups of capuchins. These algorithms are designed to efficiently solve optimization problems by emulating the collective intelligence and cooperative behaviors of these groups. Figure 2.3 provides a visual representation of the taxonomy of MHs by the source of inspiration, including representative examples for each category.



**Figure 2.3:** Classification of MHs by the source of inspiration, including representative examples for each category

### 2.1.2.3 Swarm Intelligence Algorithms

The term 'Swarm Intelligence' was coined by Beni and Wang in 1989 in the context of cellular robotic systems [82]. Since then, SI has become a widely discussed topic in many industries. According to *Bonabeau* [83], it can be defined as the collective intelligence of simple agents working in a group. In the context of MHs, SI methods appear as cooperative nature-inspired mechanisms that mimic the behavior of self-organized and decentralized systems found in species such as bees, wasps, ants, whales, and fish [84]. These methods are used to design optimization algorithms that leverage the exceptional features of swarm behavior in nature. The algorithms rely on how swarm individuals communicate with each other to reach a common goal. They perform searches based on their own cognition and experiences as well as the information available globally among all individuals. Information exchange occurs through various mechanisms such as pheromones in the case of ants,

sound waves in bats, and waggle dance in bees [85]. Researchers have also incorporated other intelligence mechanisms to develop optimization algorithms with better explorative and exploitative capabilities.

According to [11], SI-based algorithms are characterized by the simplicity and cooperation of agents in moving through the decision space. The most common algorithm among them is PSO, which was introduced by *Kennedy and Eberhart* in 1995 [86] and is inspired by the flocking behavior of birds. In this algorithm, candidate solutions are represented by a swarm of particles, each with a position ( $\vec{x}$ ) and velocity ( $\vec{v}$ ). The particles are modified towards the best solutions so far by considering both their own local best position and the global best particle position. The efficacy and simplicity of PSO have made it widely applicable in various fields and have served as the basis for developing many other SI-based algorithms [87]. Some of other widespread SI algorithms are Ant Colony Optimization (ACO) [88], Artificial Bee Colony (ABC) [89], Fish Swarm Algorithm (FSA) [90], and Bacterial Foraging Optimization (BFO) [91]. Recently, many excellent and well-regarded SI MHs have been proposed such as Harris Hawks Optimization (HHO) [92], Chimp Optimization Algorithm (ChOA) [93], Whale Optimization Algorithm (WOA) [80], GWO [94], MFO [95], Marine Predators Algorithm (MPA) [96], Slime Mould Algorithm (SMA) [97], CapSA [98], and CSA [45]. A brief review of well-known MHs, categorized by their source of inspiration, is presented in Table 2.1.

This study focuses on two recent and promising SI algorithms, the CSA and the CapSA. Section 2.4 will provide an overview of these algorithms, detailing their inspiration and mathematical foundations.

## **2.2 Parallel Approaches in MHs**

Parallel computing involves the simultaneous use of multiple processors or computing resources to handle large computational problems. To achieve this, a problem is divided into smaller tasks that can be executed concurrently [134]. Parallel computing lies at the heart of recent technological developments, especially in the design of distributed and multi-core systems [51]. Many recent research papers advocate for the use of parallel computing to



**Table 2.1:** Overview of MHs based on the source of inspiration

Category	Algorithm	Inspiration	Year
Evolutionary-Based	Evolutionary Strategy (ES) [99]	Biological Evolution Strategies	1960s
	Genetic Algorithm (GA) [69]	Natural Selection and Genetics	1975
	Genetic Programming (GP) [100]	Biological Evolution and Genetics	1990s
	Differential Evolution (DE) [101]	Evolutionary Computing	1997
	Memetic Algorithm (MA) [102]	Darwinian Evolution and Cultural Evolution	Late 1980s
	Evolutionary Programming (EP) [103]	Evolutionary Strategies of Finite State Machines	1960s
	Gene Expression Programming (GEP) [104]	Genetics and Natural Selection	2001
	Biogeography-Based Optimizer (BBO) [105]	Biogeography (Species Migration)	2008
Human-Based	Tabu Search (TS) [73]	Human Memory Processes	1986
	Cultural Algorithm [106]	Sociocultural Evolution	1994
	Teaching-Learning-Based Optimization (TLBO) [107]	Teaching-Learning Process	2011
	Brain Storm Optimization (BSO) [108]	Human Brainstorming Process	2011
	Human-Inspired Algorithm (HIA) [109]	Human Cognitive Processes	2009
	Volleyball Premier League Algorithm (VPLA) [110]	Strategies in Volleyball Game	2018
	Gaining Sharing Knowledge based Algorithm (GSK) [111]	gaining and sharing knowledge during the human life span	2020
	Human Evolutionary Optimization Algorithm (HEOA) [112]	The process of human evolution	2023
Physics-Based	Simulated Annealing (SA) [68]	Annealing Process in Physics	1983
	Central Forces Optimization (CFO) [113]	Central Forces in Physics	2008
	Gravitational Search Algorithm (GSA) [114]	Law of Gravity and Mass Interactions	2009
	Charged System Search (CSS) [115]	Electrodynamics and Charged Particles Interactions	2010
	Multi-Verse Optimizer (MVO) [116]	Concepts of White Hole, Black Hole and Wormhole in Universe	2015
	Henry Gas Solubility Optimization (HGSO) [117]	Henry's Law in Gas Solubility	2019
	Thermal Exchange Optimization (TEO) [118]	Thermal Exchange Processes	2017
	Ray Optimization (RO) [119]	Properties of Light Rays	2012
	RIME optimization algorithm [120]	physical phenomenon of rime-ice	2023
Swarm Intelligence-Based	Ant Colony Optimization (ACO) [121]	Foraging Behavior of Ants	1992
	Particle Swarm Optimization (PSO) [122]	Social Behavior of Birds and Fish	1995
	Bacterial Foraging Optimization (BFO) [91]	Foraging Strategy of E. coli Bacteria	2002
	Artificial Bee Colony (ABC) [123]	Food Foraging Behavior of Honey Bees	2005
	Firefly Algorithm (FA) [124]	Flashing Behavior of Fireflies	2008
	Cuckoo Search (CS) [125]	Brood Parasitism of Some Cuckoo Species	2009
	Bat Algorithm (BA) [126]	Echolocation Behavior of Bats	2010
	Grey Wolf Optimizer (GWO) [127]	Social Hierarchy and Hunting Mechanism of Grey Wolves	2014
	Moth-Flame Optimization Algorithm (MFO) [128]	Navigation Method of Moths	2015
	Whale Optimization Algorithm (WOA) [80]	Bubble-Net Hunting Strategy of Humpback Whales	2016
	Dragonfly Algorithm (DA) [129]	Swarming Behaviours of Dragonflies	2016
	Crow Search Algorithm (CSA) [45]	Food Searching Behavior of Crows	2016
	Grasshopper Optimization Algorithm (GOA) [130]	Swarming Behaviour of Grasshoppers	2017
	Harris Hawks Optimization (HHO) [39]	Cooperative Hunting Strategy of Harris' Hawks	2019
	Slime Mould Algorithm (SMA) [97]	Oscillation Mode of Slime Mould	2020
	Capuchin Search Algorithm (CapSA) [46]	Behavioral Traits of Capuchin Monkeys	2021
	Hunger Games Search [131]	hunger-driven activities and behavioral choices of animals	2021
	Moss Growth Optimization (MGO) [132]	The moss growth in the natural environment	2024
	Polar Fox Optimization Algorithm (PFA) [133]	Hunting method of polar foxes	2024

reduce execution time and enhance memory usage. Furthermore, numerous researchers have highlighted the importance of parallel computing in solving complex optimization problems using parallel metaheuristic algorithms [135].

Parallel MHs represent a promising class of optimization algorithms that combine the benefits of parallel computing techniques with MHs to address complex problems more efficiently, requiring less numerical effort and computational time. Essentially, the design of parallel MHs targets three crucial objectives [11] [134] [136]:

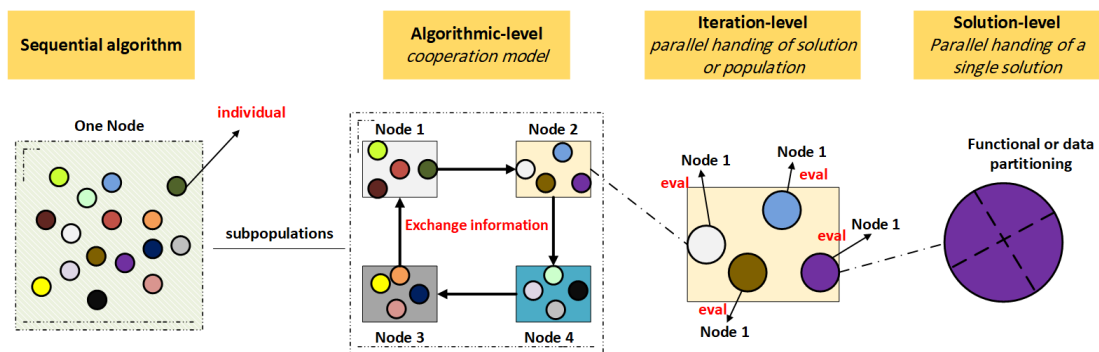
- The first objective, a common goal within the parallel computing domain, is to **solve large-scale problems more rapidly** than sequential methods can. This acceleration is particularly valuable for developing algorithms suited to real-time applications, such as critical control problems where time is a critical factor.
- **Improving algorithm robustness** is another significant advantage. Parallel MHs exhibit greater robustness compared to their sequential counterparts, both in their ability to efficiently tackle a wide array of optimization problems with diverse characteristics and in their reduced sensitivity to parameter settings, thereby lessening the need for extensive parameter-tuning efforts.
- Lastly, **providing high-quality solutions** is a key objective. Certain cooperative parallel models enable the collaborative use of multiple algorithms to solve a problem, promoting information exchange, adapting the search pattern, and ultimately facilitating the attainment of high-quality solutions within a reasonable time frame.

Up to date, many researchers continue to employ traditional MHs for solving real, complex optimization problems, attributed to factors such as the superior computational power of Central Processing Unit (CPU) and extensive random Access Memory (RAM) capacity. However, when addressing big data and real optimization challenges characterized by complex search spaces, the efficacy of traditional MHs becomes constrained. Despite these algorithms demonstrating commendable performance in terms of results, execution time and memory usage pose significant challenges, particularly for problems with nonlinear (i.e., complex) search spaces [11]. To address these issues, a significant number of researchers

are investigating the use of parallel MHs with two major goals in mind: to reduce execution time and to enhance the quality of solutions. The adoption of MHs in solving real-world optimization problems primarily aims to capitalize on these advantages, ensuring both efficient computation and superior solution outcomes.

### 2.2.1 Design Models of Parallel MHs

It is worth noting that this study focuses on the design aspects of parallel MHs using general-purpose parallel architectures rather than the implementation aspects of specific parallel architectures (language, environment, middleware). From a design perspective, parallel MHs can be conceptualized based on three parallel paradigms: the algorithmic level, iteration level, and solution level models [11] (Figure 2.4). These models are characterized by their granularity<sup>1</sup> (i.e., the level of parallelism). Table 2.2 summarizes the main characteristics of these three significant parallel designs for MHs. The following subsections provide a general overview of the design strategies that can be applied to most metaheuristic algorithms.



**Figure 2.4:** Representation of parallel design models of meta-heuristics

**Table 2.2:** Design models of parallel metaheuristics

model	Granularity	metaheuristic behavior	problem dependency	Objective	example
<b>solution-level</b>	solution	unaltered	dependent	efficiency	data partitioning
<b>iteration-level</b>	iteration	unaltered	independent	efficiency	evaluating neighbors
<b>Algorithmic-level</b>	metaheuristic	altered	independent	effectiveness	island model

<sup>1</sup>In parallel computing, granularity refers to the amount of computation performed by a task [137]

### **2.2.1.1 Iteration\_Level Parallel Model**

The iteration-level parallel model, known as functional parallelism, refers to executing the algorithm itself in parallel to reduce the execution time while exploring the search space. In this strategy, parallelization is leveraged to accelerate computing-intensive parts of the metaheuristic without altering the search space, sequence, or behavior of the original algorithm [11] [47]. This strategy is considered low-level and problem-independent. Typically, it can be adapted for both TMHs and PMHs, wherein the original sequential logic of the algorithm is maintained while the resource-consuming parts are decomposed and executed in parallel. Generally, MHs are iterative algorithms; hence, inner-loop components are the primary sources for achieving significant parallel computing gains. Examples include the generation and evaluation of neighborhoods for TMHs and the handling and evaluation of individuals for PMHs. Recently, high-performance Graphics Processing Units (GPUs) have been extensively exploited for the functional parallelism of MHs. This provides a suitable environment to effectively reduce computation time [134].

### **2.2.1.2 Solution-Level Parallel Model**

The solution-level parallel model refers to evaluating and building the solution in parallel (i.e., it is problem-dependent). In most real optimization problems, the problem-dependent search operations applied to solutions, such as the evaluation of objectives and constraints, are resource-consuming in terms of time, memory, and input/output operations [47]. To address this, two strategies can be applied: functional decomposition and data partitioning. The former strategy involves partitioning fitness function(s) and/or constraints into partial functions evaluated simultaneously. The partial results are then combined to produce the final result using a reduction operation [11]. In a data-oriented partitioning scheme, parallelism occurs at the level of data required by the objective function. For instance, evaluating candidate solutions may require accessing a vast database that cannot fit on a single machine.

### 2.2.1.3 Algorithmic-Level Parallel Model

An algorithmic-level model refers to executing a set of self-contained MHs, either independently or cooperatively, to tackle the optimization problem, also known as a multi-search model. Executing MHs independently is, in terms of solution quality, equivalent to running them sequentially. However, a cooperative model adapts the MHs' behavior to yield better-quality solutions. The independent algorithmic-level strategy, one of the earliest multi-search models proposed in the literature [64], is recognized for its simplicity and straightforward design, often enhancing the robustness of the algorithms considered [134]. This approach employs multiple classical optimizers, utilizing either the same or different search methods simultaneously across the entire search space. In this model, optimizers may have identical or varied populations and share common parameters (e.g., population size, stopping criteria) and internal parameters (e.g., selection, mutation, and crossover probabilities for EAs). Ultimately, the best result from all optimizers is selected. This model has demonstrated effectiveness across various combinatorial optimization problems. For example, *Kennedy* [138] presented a non-cooperative parallel variant of PSO, dividing the main swarm into sub-swarms using the K-means clustering technique, evolving independently without cooperation. Other independent multi-search strategies were introduced by [139] [140], with more examples in [64].

In contrast, the cooperative algorithmic-level model involves algorithms cooperating by exchanging search-based information, which, although more complex to design than the independent parallel system, shows superior performance. It has emerged as a highly effective methodology for complex optimization problems [141] [51] [142] [56] [47]. The cooperation mechanism during the search process benefits from new information to alter search behavior, explore the search space more effectively, and find better (or near-optimal) solutions. Cooperative model techniques for MHs vary based on several factors, including communication mechanisms and the nature of the exchanged information. Design considerations such as when and where information is exchanged, what information is exchanged, and how it is integrated into the search process are crucial in developing these models [47]. These aspects are further discussed in Section 2.3.

Among these models, the island model has proven remarkably effective and has become more reliable for use with MHs. In recent years, it has gained prominence as a preferred parallel design strategy for handling various types of combinatorial optimization problems [11]. Therefore, this study focuses on the island-based MHs and the modern studies that have dealt with it. Section 2.3 presents the fundamentals of the basic island model for MHs.

### **2.3 Fundamentals of Cooperative Island-Based Model**

In the cooperative MHs-based optimization paradigm, the utilized algorithms collaborate by exchanging search-based information. Cooperative algorithmic models perform better than independent-level models. It has become one of the best methodologies for tackling challenging optimization problems [141, 47]. The cooperation mechanism aims to gain from the shared information throughout the search process in order to change the search behavior, explore the search space more thoroughly, and ultimately discover better (or nearly optimal) solutions. In general, there are different ways to differentiate between cooperative model strategies for MHs based on the communication method and the type of information being shared [47]. The three main types of cooperative MHs models are the master-slave model, the cellular model (also known as fine-grained), and the island model (also known as coarse-grained) [51]. The island model has emerged as one of the most successful and trustworthy to employ with metaheuristic algorithms. It has gained prominence in recent years as a favored parallel design approach for resolving many kinds of combinatorial optimization problems [11].

The island model is regarded as one of the most successful structured population techniques for dividing the population into a number of subpopulations, each of which is termed an "island" [49]. This method is useful for empowering the capabilities of evolutionary algorithms by boosting aspects of diversity. In this paradigm, the evolutionary algorithm is iterated on each island, either synchronously or asynchronously [143]. The island model mainly comes in two variants: homogeneous and heterogeneous [144]. Each island in the heterogeneous model has its unique configurations, as well as its own search method. For instance, in an island-based Genetic Algorithm (iGA), each island may be set up with a dis-

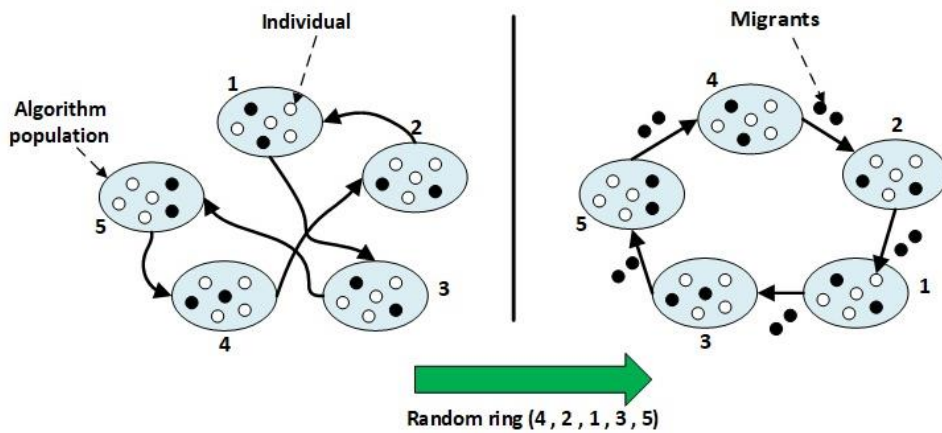
tinct crossover or selection strategy. In the homogeneous paradigm, the same configurations and search methods are embedded with each island.

The flow of information between the partitioned islands is managed by the periodically recurring migration process. In simple terms, after a certain number of iterations, some chosen solutions will migrate across some islands [59]. Such a technique might enhance the algorithm's performance as it facilitates exploring diverse areas of the search space. The distribution of individuals with a wide range of fitness values throughout the various islands may help the algorithm arrive at a globally optimal solution [143]. A number of migration factors should be given priority when incorporating the island model into the optimization framework of any population-based algorithm. These factors include the communication topology, the migration rate, the migration frequency, and the migration policy [145, 53, 59]. Details and the relevance of these factors will be discussed in the following parts.

### **2.3.1 Interconnection Topology**

The logical connections between the cooperating algorithms (or islands) construct a communication network. Within this network, each node represents an island, and the edges indicate the pairs of islands that might exchange their search information [11]. Therefore, the utilized topology defines how the islands are interconnected and determines the neighbor(s) for each island. The optimization quality is substantially impacted by the type of topology. Several migration topology structures have been presented in the literature, including star topology, grid topology, ring topology, and complete graph topology [146, 147, 148]. Nevertheless, the ring topology is becoming increasingly prevalent, and encouraging results have been gained through its utilization [57, 55]. The idea of migration topology may be classified into two subcategories: static and dynamic. In static topologies, the route of information exchange between neighboring islands and the related links is predefined and remains constant through the population evolution process. When a static topology is applied, there is a higher chance of diversity loss since the information exchange method follows a repeated pattern. In contrast, the destination island in dynamic topology changes arbitrarily with each algorithm run, thus preserving it to some extent. [143]. Figure 2.5 depicts the structure of

a directed random-ring topology where the viable route connecting the islands is randomly established



**Figure 2.5:** Migration process among algorithms using random ring topology [57]

### 2.3.2 Migration Rate

Another important component of the migration process is the migration rate ( $M_r$ ). It defines the percentage of individuals that will migrate from one island to another. The choice of  $M_r$  value is crucial since a small number can affect how well algorithms cooperate while a large value can reduce the diversity of the search process which leads to a high probability of early convergence (i.e., the majority of sub-populations may converge to the same solutions) [149].

### 2.3.3 Migration Frequency

It is necessary to precisely select when to exchange the search-based information. As a result, migration frequency ( $M_f$ ), a predetermined number of iterations, is frequently used to regulate the migration procedure. The number of rounds between each successful information swap between neighbor sub-populations is controlled using this value. The effectiveness of the cooperative model as a whole is significantly influenced by the value of the migration frequency [150]. Therefore, it's crucial to choose the right value for this parameter; for instance, when  $M_f$  is large, the islands prefer to evolve independently [11, 47].



### 2.3.4 Migration Policy

The migration policy specifies which individuals will be migrated from the source island (i.e., the selection strategy) and where they will be placed on the destination island (i.e., the replacement strategy) [145]. There are two widely used policies in the relevant literature: the best-worst policy and the random-random policy. The best-worst strategy involves selecting the fittest individuals from the source island to replace the worst individuals on the target island [58]. The random-random policy is implemented by choosing random members from the source island to be swapped with random members on the target island. However, the best-worst approach is the most prevalent migration policy in the literature [59, 57, 55, 56]. In contrast to the random-random policy, which has little impact, the best-worst policy has a significant impact on the optimization performance of island models [151].

It is worth mentioning that the performance of the island model is significantly impacted by the migration-based parameters that were discussed previously. As a result, they need to have the right tuning in order to provide high-quality solutions in a reasonable amount of time [59].

### 2.3.5 Formulating the Island Model for Population-Based MHs

This section outlines the general procedural steps for integrating the island model with a population-based metaheuristic algorithm (denoted as  $Z$ ). Initially, the generated population of size ( $N$ ) is divided into smaller sub-populations of number ( $s$ ), each comprising a set of size ( $I_s = N/s$ ) individuals. Assuming a homogeneous model where the same search algorithm is applied to each sub-population, each algorithm operates for several predefined maximum iterations. The migration process is initiated periodically after a predefined number of iterations, identified by the migration frequency ( $M_f$ ) parameter. When the migration process is triggered, several selected individuals (migrants) defined by the migration rate ( $M_r$ ) are transferred to neighboring islands. The process diagram is illustrated in Figure 2.7. The detailed steps are presented as follows:

- **Step 1:** Initialize the internal parameters of algorithm  $Z$  (such as mutation, crossover, and selection probabilities for GA), common controlling parameters (population size

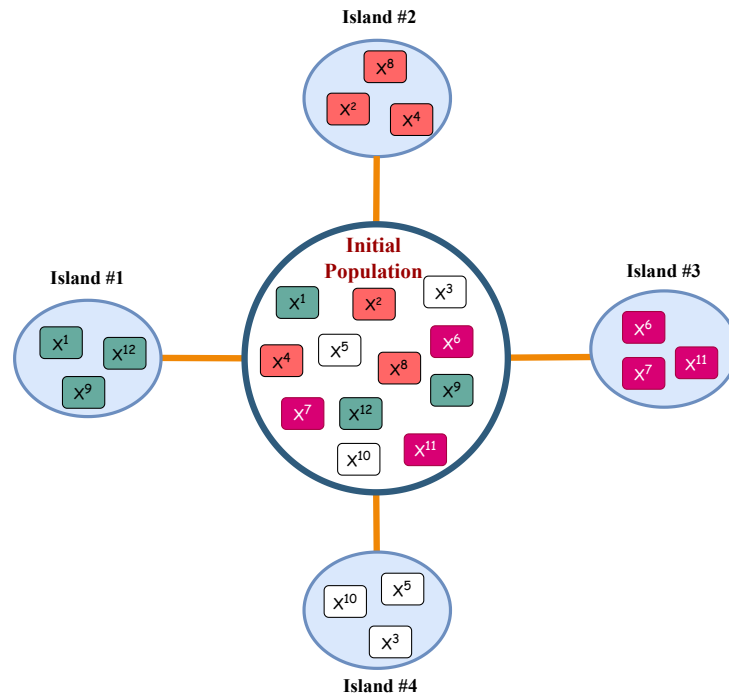
( $N$ ) and maximum number of iterations ( $itr$ ). In addition, the parameters of migration model should be initialized: number of islands ( $s \leq N$ ), island size  $I_s = N/s$ , migration frequency ( $M_f$ ), migration rate ( $M_r$ ) where  $M_r \cdot I_s \leq I_s$ , design of communication topology, selection and replacement strategies.

- **Step 2:** Generate the initial population randomly. In this step a population of candidate solutions  $X = (x_1, x_2, x_3, \dots, x_n)$  are usually generated randomly, where each candidate solution ( $x_j$ , where  $j \in (1, 2, 3, 4, \dots, N)$ ) is determined using Eq. (3.9).

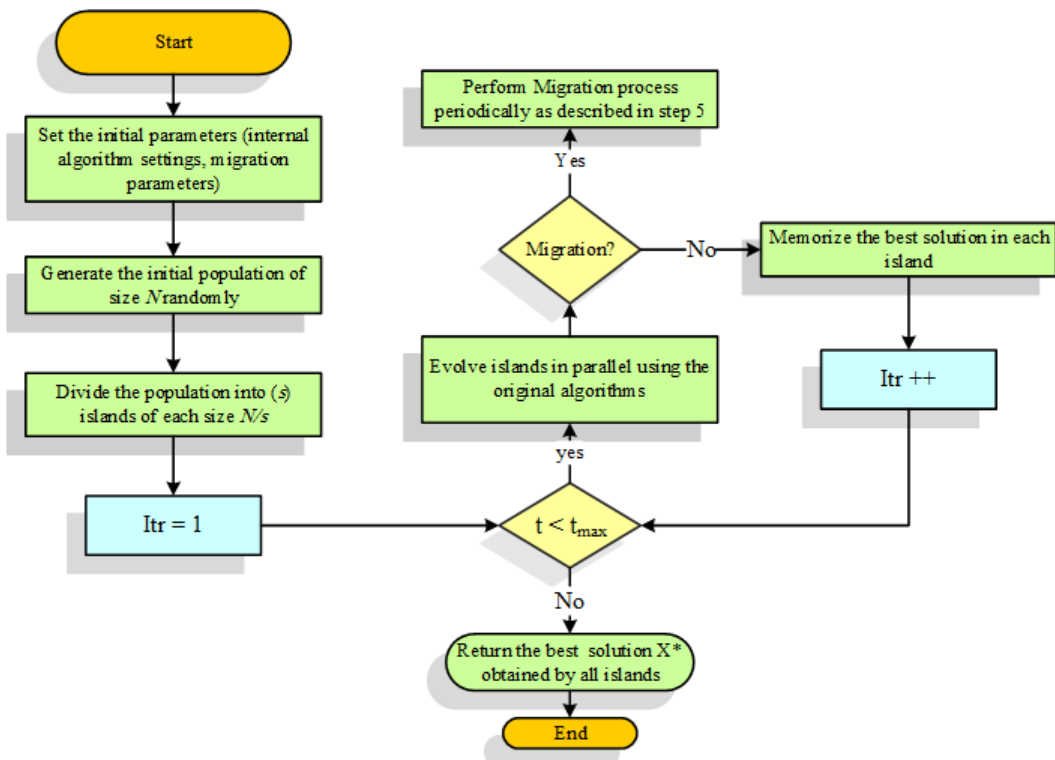
$$\vec{X}_j = \vec{X}_L + r(\vec{X}_U - \vec{X}_L) \quad (2.1)$$

where  $\vec{X}_L$  and  $\vec{X}_U$  are the lower and upper bound for the problem dimensions.

- **Step 3:** Divide the generated population into  $s$  islands each of size  $I_s$ . For instance, let  $N = 12$ ,  $s = 4$ , then  $I_s = 3$ . Assuming that  $X = (x_3, x_1, x_5, x_{12}, x_8, x_7, x_2, x_6, x_9, x_{10}, x_4, x_{11})$ , then  $island_1 = (x_1, x_{12}, x_9)$ ,  $island_2 = (x_4, x_8, x_2)$ ,  $island_3 = (x_7, x_{11}, x_6)$ , and  $island_4 = (x_{10}, x_5, x_3)$ . Note that each solution is assigned to a random island (see Figure 2.6).
- **Step 4:** Evolve the islands in parallel. In this step, each island is evolved using the operators of the original algorithm  $Z$ .
- **Step 5:** Migration process. This process is triggered periodically after a predefined number of iterations as specified by ( $M_f$ ) parameter. A commutation topology (e.g., bidirectional ring topology) is generated randomly to interconnect the islands (i.e., identify the neighbours for each island). Then, a percentage of selected individuals from each island, controlled by ( $M_r$ ) parameter, are transferred between the connected islands. The swapping process among every two neighbouring islands is executed based on the selection-replacement strategy (e.g., best-worst strategy).
- **Step 6:** Memorizing the best solution obtained so far in each island.
- **Step 7:** Repeat steps 4, 5, and 6 until the stop condition (e.g., maximum iterations is reached) and return the best solution found in all islands.



**Figure 2.6:** Example of population segmentation in an island model framework ( $N = 12$ ,  $s = 4$ ,  $I_s = 3$ )



**Figure 2.7:** The general process diagram of the island-based parallel mechanism

## 2.4 Literature Review

This section provides a thorough overview of the literature about two distinct MHs: CSA and CapSA. These algorithms have gained significant attention in the field of optimization due to their unique search methods inspired by the intelligent foraging behavior of crows and capuchin monkeys, respectively. The review focuses on the modifications, hybrid models, and innovative approaches proposed to enhance the original algorithms and overcome some of their limitations, such as early convergence and balancing exploration and exploitation. Additionally, the chapter introduces a systematic review of the island model and its application to MHs, which enhances their search capabilities by maintaining diversity and parallel exploration. Moreover, the chapter emphasizes the practical use of these improved algorithms and their effectiveness in solving real-world problems in various fields. The main objective of this comprehensive review is to analyze the current status of research in this field by highlighting advancements, applications, and unexplored directions. It also provides a foundation for further research that enables researchers to position their novel contributions within the existing landscape of MHs and island-based models.

### 2.4.1 Overview of the Standard Crow Search Algorithm

This section provides an overview of the original CSA; inspiration and the mathematical model. *Askarzadeh* [45] developed a unique metaheuristic algorithm called CSA that mimics crows' foraging behavior. The crow is considered to be one of the most clever birds due to its ability to recall where food has been concealed for an extended period of time. It often conceals the additional food and then retrieves it when it is required. In addition to this, it will imitate the behavior of other crows in order to steal food and will utilize the knowledge it has gained as a thief in order to deter other crows from committing theft. The primary source of inspiration for CSA was the search technique employed by crows to hide their extra food and then retrieve it at the appropriate moment. The four guidelines listed below, as proposed in [45], realize the fundamental CSA presumptions:

- Crows live in flocks in nature.
- Crows are able to remember the areas where they hide their food.
- The activities of crows tend to be cooperative since they follow each other to do thievery.
- Crows may self-manage to preserve their unseen food, where they can protect their hideouts from theft by a percentage.

In light of the aforementioned intelligent behaviors, the evolutionary process of CSA is mathematically represented as follows: It is presumed that there is a  $d$ -dimensional search space that contains a number of crows. The number of crows, also known as the population size, is denoted by  $N$ . The location of each individual crow  $i$  at a certain time (iteration)  $t$  represents a candidate solution to the problem of interest and can be defined by a vector  $X_{i,t}$  as shown in Eq (2.2).

$$X_{i,t} = [x_1^{i,t}, x_2^{i,t}, x_3^{i,t} \dots x_d^{i,t}] \quad (2.2)$$

where ( $i = 1, 2, \dots, N$ ), ( $t = 1, 2, \dots, t_{max}$ ),  $t_{max}$  is the maximum number of iterations,  $d$  is the problem dimension, and  $x_d^i$  denotes the  $d$ th dimension of the  $i$ th crow. Accordingly, the whole population of crows  $X$  (as shown in Eq. (2.3) ) is initially positioned randomly in the  $d$ -dimensional search space. These are the potential solutions to the problem at hand.

$$X = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & x_3^2 & \dots & x_d^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & x_3^N & \dots & x_d^N \end{bmatrix} \quad (2.3)$$

It is common knowledge that every individual crow possesses a memory that allows it to remember the specific location of its hiding place. Therefore, upon iteration  $t$ ,  $m_{i,t}$  (as shown in Eq. (2.4)) denotes the position of the hiding place of crow  $i$ . In alternative words, this is the best-visited position that crow  $i$  has attained so far. Each crow has, in fact, memorized

the location of its finest experience. Therefore, crows in CSA move and look for better food sources in their surroundings (i.e., hiding places). Since the crows are completely inexperienced at the beginning, it is presumed that they have concealed their food in their starting locations.

$$m_{i,t} = [m_1^{i,t}, m_2^{i,t}, m_3^{i,t} \dots m_d^{i,t}] \quad (2.4)$$

The exploratory and exploitative potentials of the CSA are achieved according to two behaviors: pursuit and evasion [152]. In specific, assume that during iteration  $t$ , the crow  $j$  wishes to go to its hiding location  $m_{j,t}$ . Crow  $i$  makes the decision to follow Crow  $j$  and get closer to its hiding location in this iteration. Two states might occur in this scenario:

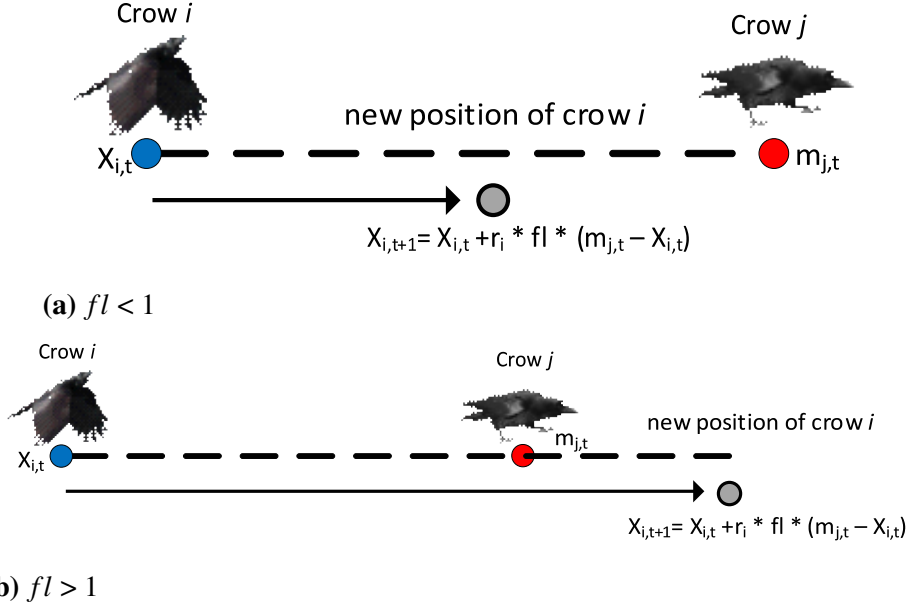
- **Pursuit behavior:** A crow  $i$  pursues crow  $j$  in an effort to approach to its hidden place. The goal of crow  $i$  is accomplished since the crow  $j$  is unaware of the other crow's existence. In this case, the new position of crow  $i$  is modeled using Eq. (2.5).

$$X_{i,t+1} = X_{i,t} + r_i \times fl_{i,t} \times (m_{j,t} - X_{i,t}) \quad (2.5)$$

where  $X_{i,t}$  and  $X_{i,t+1}$  represent the current and new positions of crow  $i$ , respectively,  $r_i$  is a uniformly distributed random number within  $[0,1]$ ,  $fl_{i,t}$  is the flight length of the crow  $i$  at iteration  $t$ , which has an impact on the exploitative and exploratory potentials of the CSA.  $m_{j,t}$  refers to the best position so far of the crow  $j$  (i.e., the memorized position of the hiding place of crow  $j$ ).

The schematic of this condition and the impact of flight length  $fl$  on search-ability are shown in Figure 2.8. Local searches (at the vicinity of  $X_{i,t}$ ) are facilitated by low values of  $fl$ , whereas global searches (far from  $X_{i,t}$ ) are facilitated by high values. The next location of the crow  $i$  is on the dashed line between  $X_{i,t}$  and  $m_{j,t}$  if the value of  $fl$  is chosen less than 1 (as shown in Fig. 2.8(a)). Whereas, the next location of the crow  $i$  is on the dashed line, which may surpass  $m_{j,t}$  if the value of  $fl$  is picked greater than 1 (as shown in Fig. 2.8(b)).

- **Evasion behavior:** Crow  $j$  is aware that crow  $i$  is following it and thus consciously



**Figure 2.8:** The schematic Pursuit behavior in CSA [153]

chooses a random trajectory to trick crow *i* and thereby protect its food. In CSA, this phenomenon is replicated by employing a random movement as given by Eq. (2.6).

$$X_{i,t+1} = LB + r_i \times (UB - LB) \quad (2.6)$$

where *LB* and *UB* represent the lower and upper bounds of the decision variables, respectively.

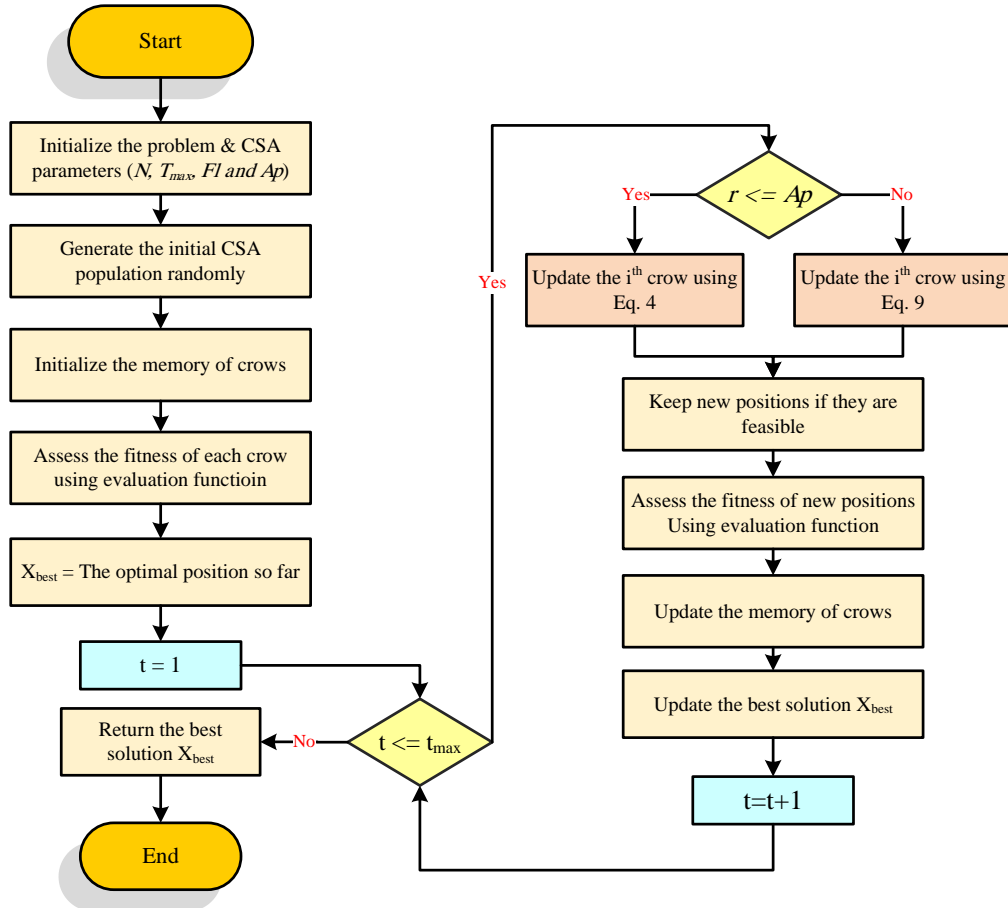
An awareness probability factor (*AP*) determines the kind of activity that each crow *i* will consider. A uniformly distributed random number *r* inside (0,1) is therefore sampled. If *r* is larger than or equal to *AP*, pursuit behavior is used; if not, an evasion scenario is selected. The following model neatly describes this process:

$$X_{i,t+1} = \begin{cases} X_{i,t} + r_i \times fl_{i,t} \times (m_{j,t} - X_{i,t}) & r_j \geq AP_{i,t} \\ random & otherwise \end{cases} \quad (2.7)$$

where  $AP_{i,t}$  refers to the awareness probability of crow *j* at iteration *t*. After the crow has been modified, its fitness is assessed, and the memory vector is updated as follows:

$$m_{i,t+1} = \begin{cases} X_{i,t+1} & F(X_{i,t+1}) \text{ is better than } F(m_{i,t}) \\ m_{i,t} & \text{otherwise} \end{cases} \quad (2.8)$$

where  $F(\cdot)$  represents the objective function to be optimized. Figure 2.9 depicts the flowchart of the standard CSA.



**Figure 2.9:** Flowchart of the standard CSA

The exploration (diversification) and exploitation (intensification) offered by any meta-heuristic algorithm ought to be well-balanced [11]. In CSA, the awareness probability value primarily dominates how CSA controls intensification and diversification. By lowering the AP value, CSA is more likely to focus its search on a local area when a good solution is already present there. As a consequence, intensity is increased by employing modest AP values. On the other side, when the AP value increases, it becomes less likely to search the vicinity of current good solutions, and CSA is more likely to broaden its search area



(randomization). Therefore, using high AP values enhances diversity [45].

Compared to its competitors, the CSA offers several advantages. It is relatively simple to implement, has a shorter run-time, involves fewer mathematical equations and control settings, and exhibits strong space exploration capabilities [63, 30]. Moreover, it incorporates a built-in control strategy that automatically switches between exploration and exploitation phases. Consequently, it has been successfully applied in various engineering applications, including DNA fragment assembly, image processing, feature selection, power distribution network optimization, and economic load dispatch [81, 154].

#### 2.4.2 Overview of Capuchin Search Algorithm (CapSA)

A novel metaheuristic called CapSA was developed by studying the natural behavior and daily routines of capuchin monkeys as they foraged for food in the wild. According to *Braik et al.*, [46], the most interesting fact regarding the social behavior of capuchin monkeys is that they employ three excellent maneuvers to move about when foraging on trees, riverbanks, and ground, namely jumping, swinging, and climbing. These facts form the basis of CapSA's fundamental assumptions. The CapSA algorithm, like other swarm intelligence-based algorithms, is classified under the umbrella of population-based algorithms. It starts by randomly initializing a preset number of individuals (i.e., capuchins). Each capuchin represents a possible solution to the problem being addressed. The mathematical representation of CapSA's evolutionary process is as follows: Several  $N$  capuchins are assumed to be distributed over a  $d$ -dimensional search space.  $N$  is often referred to as population size. A possible solution to the problem of interest may be determined by the location of each capuchin individual  $i$  at a specific moment ( $t$ ), which can be described as a vector:

$$X_i(t) = [x_i^1, x_i^2, x_i^3 \dots x_i^d]$$

Each individual is also characterized by its velocity:

$$V_i(t) = [v_i^1, v_i^2, v_i^3 \dots v_i^d]$$

where  $(i = 1, 2, \dots, N)$ ,  $(t = 1, 2, \dots, T)$ ,  $T$  is the maximum number of iterations,  $d$  is the number of variables of a test problem, and  $x_d^i$  denotes the  $d$ th dimension of the  $i$ th capuchin. The whole population of capuchins  $X$  and the corresponding velocities  $V$  are initially positioned randomly in the  $d$ -dimensional search space.

Capuchin swarms often consist of two categories of individuals: (1) leaders, also known as alphas, who are responsible for finding new food sources, and (2) followers, who are in charge of updating their positions by following the group's leaders.

#### 2.4.2.1 Leaders Updating Rules

The community's leaders employ five distinct mobility strategies to locate food during the evolutionary process. A random number named  $\varepsilon$  is generated to determine operation selection as follows:

- **Jumping on trees** ( $i < N/2; 0.1 < \varepsilon \leq 0.15$ ): In this situation, alpha capuchins can be positioned using the following updating rule:

$$X_i(t+1) = gbest + \left( \frac{P_{bf}(v_i)^2 \sin(2\theta)}{g} \right) \quad (2.9)$$

where the variable  $gbest$  denotes the current optimal position of the food source, while  $\varepsilon$  refers to a randomly generated number in the range of 0 to 1.  $P_{bf}$  represents the probability of the capuchin monkey's tail providing balance during the jumping process. The velocity of the  $i$ th capuchin, denoted as  $v_i$ , is computed using Equation (2.10), with  $g$  being the gravitational acceleration constant set to 9.81. The jumping angle, denoted as  $\theta$ , is calculated as  $1.5 \times r$ , where  $r$  is a uniformly distributed random number between 0 and 1.

$$\begin{aligned} V_i(t+1) &= \rho v_i(t) + a_1 \left( pbest_i - x_i(t) \right) r_1 \\ &+ a_2 \left( gbest - x_i(t) \right) r_2 \end{aligned} \quad (2.10)$$

where  $pbest$  refers to the best position so far of the  $i$ th capuchin. The factors  $a_1$  and  $a_2$  govern the influence of the individual best position ( $pbest$ ) and the global best position ( $gbest$ ) on the capuchin's velocity. The random variables  $r_1$  and  $r_2$  are uniformly distributed in the interval  $[0, 1]$ . The inertia coefficient  $\rho$  determines how much the previous velocity affects the current motion, and in this study, it is decreased during iterations using Equation (2.11) to regulate the search for either local or global solutions.

$$\rho = w_{max} - (w_{max} - w_{min})\left(\frac{t}{T}\right)^2 \quad (2.11)$$

where  $w_{max}$  and  $w_{min}$  take values of 0.8 and 0.1, respectively.

- **Jumping on the ground** ( $i < N/2; 0.15 < \varepsilon \leq 0.3$ ): Capuchins employ this behavior to travel long distances, especially when food is hard to come by on the trees. The new position of the leader and following capuchins in this instance may be determined as follows:

$$X_i(t+1) = gbest + \left(\frac{P_{ef}P_{bf}(v_i)^2 \sin(2\theta)}{g}\right) \quad (2.12)$$

where  $P_{ef}$  stands for the elasticity probability of the capuchin movement on the ground.

- **Normal walking on the ground** ( $i < N/2; 0.3 < \varepsilon \leq 0.9$ ): In this case, the leaders' position can be updated as follows:

$$X_i(t+1) = x_i(t) + v_i(t+1) \quad (2.13)$$

- **Swinging on the trees** ( $i < N/2; 0.9 < \varepsilon \leq 0.95$ ): While looking for food on tree branches, certain alpha capuchins and other accompanying capuchins may employ local search. The following rule was used to simulate this behavior:

$$X_i(t+1) = gbest + P_{bf} \times \sin(2\theta) \quad (2.14)$$

- **Climbing trees** ( $i < N/2; 0.95 < \varepsilon \leq 1.0$ ): Certain alpha capuchins and other following capuchins may repeatedly ascend and descend trees and their branches in a manner akin to local search. Specifically, the following rule can be used to update the

capuchins' location:

$$X_i(t+1) = gbest + P_{bf} \left( v_i(t+1) - v_i(t) \right) \quad (2.15)$$

- **Random migration of the capuchines** ( $i < N/2; \varepsilon < 0.1$ ): Capuchin monkeys engage in random migration during food foraging, wherein they search for food in various directions to more efficiently explore their surroundings in search of better food sources. The process of random migration is modeled using Eq. (2.16)

$$X_i(t+1) = \tau \times \left[ LB + r \times (UB - LB) \right] \quad (2.16)$$

where LB and UB represent the lower and upper bounds of the decision variables. Capuchin monkeys have a 0.1 probability of engaging in a random search. This strategy enhances CapSA's capacity to explore globally and reduces the likelihood of getting stuck in local optima. In CapSA, an exponential function with a lifetime parameter ( $\tau$ ) was introduced to balance exploration and exploitation during global and local search processes. This function is represented by Eq.

$$\tau = 2e^{-11\left(\frac{t}{T}\right)^2} \quad (2.17)$$

In summary, according to Eqs. (1) through (9), the capuchin monkeys adjust their positions based on the presence of food. This behavior is especially noticeable when  $r > 0.1$ . However, if  $r \leq 0.1$ , the capuchin monkeys tend to change their positions randomly to explore different areas for food. In such instances, the parameter  $\tau$  can expand the exploration space for searching.

#### 2.4.2.2 Followers Updating Rule

The positions of the followers (i.e.  $N/2 \leq i \leq N$ ) are updated according to the following formula, which is derived from Newton's law of motion:

$$x_i(t+1) = \frac{1}{2} (x_i(t) + x_{i-1}(t)) \quad (2.18)$$

where  $x_i(t)$  and  $x_{i-1}(t)$  are the positions of the  $i$ -th follower and the  $(i - 1)$ -th follower in the previous generation. The original paper describes the comprehensive steps and the complete mathematical model utilized to derive this formula.

The CapSA technique has shown to be effective in various fields such as optimizing renewable energy sources [155, 156], machine learning and data analysis [156, 157], computing and network systems [157, 158], industrial processes [159], and economic modeling [159]. This is because the method provides a good balance between exploration and exploitation, making use of both local and global search strategies. Its straightforward and intuitive design makes it easy to understand and use. Furthermore, the algorithm features a dynamic balance factor that helps to preserve population diversity during the search [160].

### 2.4.3 Recent Advancements in the CSA

The CSA has garnered significant research interest, as it has proven to be a versatile tool for optimizing a wide range of problems [161]. To further enhance its performance, researchers have made modifications and hybridization with various optimization techniques. In a comprehensive review by *Hussien et al.* [81], CSA variants were categorized into three classes: modified CSA, hybrid CSA, and multi-objective CSA. While there exist numerous effective applications of CSA, this section highlights a few notable state-of-the-art research endeavors. One remarkable contribution is the development of a binary CSA (BCSA) by *Al-Thanoon et al.* [162], aimed at improving classification performance through the selection of appropriate features. In BCSA, the concept of the opposition-based learning method is employed to determine the flight length parameter. Experimental results demonstrated the superior performance of this technique compared to alternative mathematical methods in terms of relevant feature selection for both datasets. Addressing the premature convergence issue in the original CSA and tackling the feature selection problem in packing mode, *Chaudhuri and Sahu* [163] proposed the binary CSA with time-varying flight length (BCSATVFL). This variant dynamically adjusts the flight length parameter over time, mitigating premature convergence. This approach proved effective in addressing the feature selection problem. In another study, *Eliguzel and Ozceylan* [164] effectively employed an enhanced

version of CSA, called I-CSA, to solve the P-median problem. The proposed approach utilized discretization, local search techniques, and an elitism strategy to improve the algorithm's performance. Through these enhancements, I-CSA demonstrated promising results in solving the P-median problem. These aforementioned research efforts exemplify the diverse applications and advancements in the field of CSA, showcasing its potential for solving complex optimization problems.

*Grisales-Norina et al.* [165] proposed an efficient master-slave methodology using the CSA to integrate photovoltaic generators into DC grids, minimizing operating costs. Numerical results demonstrate the approach's applicability, efficiency, and robustness compared to other methodologies. *Braik et al.*[166] propose a Memory-based Hybrid CSA (MHCSA) by combining CSA with PSO. MHCSA enhances CSA's memory representation and exploration-exploitation balance, outperforming CSA, PSO, and other methods in accuracy and stability across various benchmarks. However, the study lacks in-depth analysis of the hybridization impact and its adaptability to various problem domains, leaving room for further research. *Kumar et al.* [167] proposed a novel approach combining stacked autoencoders and the CSA for community detection in complex networks. Their method utilizes modularity matrix reduction and a modified k-means clustering algorithm with CSA-based optimization. The extensive experimental analysis demonstrates the effectiveness of the proposed method compared to traditional and contemporary community detection algorithms. *Bai et al.* [168] proposed a topology optimization protocol for the Internet of Things (IoT) perception layer, leveraging an improved CSA named Cauchy variation optimization CSA (CM-CSA). They also introduce an enhanced clustering approach using Cauchy mutation to address CSA's convergence speed limitations. The proposed CM-CSA algorithm significantly reduces energy consumption and enhances connectivity performance compared to PSO, AFSA, and basic CSA algorithms, offering promising advancements for wireless sensor networks. *He et al.* [169] introduced a Multi-Stage CSA (MSCSA) addressing the limitations of the original CSA in handling complex global optimization problems. MSCSA employs chaos and multiple opposition-based learning to enhance population quality, integrates local and global search stages, and incorporates large-scale migration for population diver-

sity. Experimental results showcase MSCSA's competitiveness and superiority over other algorithms on complex benchmarks. *Singh et al.* [170] introduced a novel load-shedding approach employing a modified CSA to ensure post-load-shedding voltage stability. The proposed method optimally selects load buses for shedding, using an objective function that considers voltage increments, PV-curve slope, transmission losses, and load shed amount. Comparison with other algorithms highlights the efficacy of the CSA-based strategy on IEEE 14 and 25-bus systems, enhancing voltage stability while minimizing load shedding.

*Rao et al.* [171] introduced a hybrid algorithm, the Probabilistic Simplified Sine Cosine CSA (PSCCSA), to enhance the CSA's location update process. PSCCSA combines the simplicity of CSA with a probabilistic simplified sine cosine approach, effectively improving its performance. Comparative experiments demonstrate PSCCSA's feasibility and effectiveness across standard test functions and classic engineering problems. *Liu et al.* [172] introduced a modified CSA called Group Strategy CSA (GCSA), which enhances optimization efficiency by dividing crows into competing groups with distinct roles and statuses. GCSA incorporates various search modes to enhance solution diversity and search efficiency, while an adaptive mechanism adjusts crow search ranges for balanced exploration and exploitation. Experimental results on benchmark functions and engineering design problems show GCSA's competitive performance against 11 other algorithms. *Osei-kwakye et al.* [173] proposed a diversity-enhanced hybrid algorithm called Hybrid PSO and CSA with a clustering initialization strategy (HPSOCSA-CIS) for feature selection. HPSOCSA-CIS addresses the premature convergence issue of the Binary PSO (BPSO) by incorporating crow intelligence and a clustering technique. Experimental results on 15 standard UCI datasets demonstrate that HPSOCSA-CIS outperforms other hybrid and standard optimization algorithms, achieving significant improvements in mean classification accuracy for low, medium, and high-dimensional datasets. *Andic et al.* [174] introduced a novel application of the CSA in power system state estimation using limited-channel Phasor Measurement Units (PMUs). The CSA is utilized to optimize PMU placement considering channel constraints and to estimate the system state based on the collected PMU data. Experimental results on IEEE test systems show that the proposed CSA-based state estimator outperforms GA, PSO, and Artifi-

cial Bee Swarm Optimization (ABSO) methods, demonstrating its effectiveness in achieving accurate and reliable power system state estimation. This study presents a valuable contribution to enhancing power system analysis, operation, and planning through improved state estimation techniques. *He et al.* [175] introduced a novel version of the CSA called Multi-Stage Search Integration CSA (MSCSA) to address the limitations of the original CSA in handling complex global optimization problems. MSCSA incorporates chaos and multiple opposition-based learning techniques to enhance population quality and ergodicity. It employs multiple search stages, including free foraging, following, and large-scale migration, to strike a balance between global exploration and local exploitation. Experimental comparisons with various algorithms on complex benchmark functions demonstrate the competitive performance and superiority of MSCSA in tackling large-scale complicated problems. *Zhao et al.* [30] introduced VMCSA, an improved CSA enhanced with variable neighborhood descent (VND) and information exchange mutation (IEM) strategies. VMCSA addresses local optimality issues in CSA and showcases superior optimization performance through experiments on CEC2014 and CEC'21 benchmarks. Furthermore, VMCSA demonstrates its effectiveness in multi-level thresholding image segmentation, particularly on COVID-19 X-ray images, exhibiting superior segmentation results and robustness compared to alternative algorithms.

In a notable contribution, *Gholami et al.* [176] introduced an enhanced version of the Crow Search Algorithm (CSA) known as Improved CSA (ICSA). This novel approach incorporates a sophisticated updating technique that leverages the advantages of the current global best position. By doing so, the algorithm's convergence is promoted, and its capability for local search is significantly enhanced. Experimental results demonstrate that the proposed ICSA method outperforms traditional CSA as well as other meta-heuristic algorithms, yielding promising and superior results. Furthermore, *Ke et al.* [177] proposed an improved version of CSA, also referred to as ICSA, for the optimal design of a typical commercial building in a selected Australian city. This method integrates value functions and reverse learning to improve the algorithm's performance. Comparative analysis using a number of benchmark methods shows that ICSA delivers more precise findings while significantly re-



ducing computing times, making it a highly efficient solution. To handle the limitation of the conventional CSA, *Necira et al.* [62] have introduced a new form of the CSA called the "dynamic crow search algorithm" (DCSA). The DCSA adds two modifications to the basic CSA. Firstly, the algorithm is dynamically adapted by changing the CSA parameters throughout the optimization process. Specifically, the flight length ( $FL$ ) is modified in accordance with the generalized Pareto probability density function, whilst the attraction parameter ( $AP$ ) is adjusted along a linear trajectory. Experimental findings show that DCSA outperforms its conventional counterpart. These CSA developments, such as the ICSA, ICSA with reverse learning and value functions, and the dynamic crow search algorithm (DCSA), demonstrate the ongoing work to enhance the algorithm's convergence, accuracy, and efficiency, further establishing CSA as a powerful tool for solving complex optimization problems.

To cope with the fundamental CSA flaws, *Braik et al.* [153] introduced a memory-based hybrid CSA (MHCSA) using Particle Swarm Optimization (PSO). This hybridization strategy was suggested to boost its diversity and balance its search capabilities. To take advantage of the most fruitful search regions, the memory component of MHCSA was started with the best solution (pbest) of PSO. The best CSA members are improved using the best PSO solutions discovered thus far (gbest) and (pbest). Furthermore, a better balance between exploration and exploitation throughout the course of iterations was achieved by utilizing two adaptive exponential functions for the  $AP$  and  $fl$  parameters. Recognizing the limitations of CSA, specifically those related to fixed predefined parameters such as  $fl$  and  $AP$ , the authors in [178] introduced adaptive parameter mechanisms to balance exploration and exploitation potentials. The study proposed three new versions of CSA: Exponential CSA (ECSA), Power CSA (PCSA), and S-shaped CSA (SCSA). These versions incorporate exponential, power, and S-shaped growth functions, respectively, to modulate  $fl$  and  $AP$  dynamically. Furthermore, a new dominant parameter was added to the positioning update process to further refine the algorithms' exploration and exploitation capabilities. To validate the effectiveness of these improved CSA versions, the researchers conducted extensive evaluations using 67 benchmark functions and applied the algorithms to four engineering design problems. Comparative analyses with standard CSA and other existing methods demonstrated that ECSA,

PCSA, and SCSA significantly enhanced performance.

*Cao et al.* [179] proposed an improved CSA to optimize the extreme learning machine neural network. The proposed improvements include incorporating the search strategy of the PSO algorithm and the Gaussian function into the primary CSA to boost its global search ability and convergence accuracy. *Khalilpourazari and Pasandideh* [180] developed a novel hybrid algorithm called the sine-cosine crow search algorithm (SCCSA) that combines the benefits of two recently created algorithms, CSA and sine-cosine algorithm (SCA). The suggested algorithm's capabilities for diversification and intensification have been significantly enhanced. Experimental results and analysis demonstrated that the recommended technique offered promising solutions compared to other state-of-the-art methods. *Rizk-Allah et al.* [181] incorporated chaos theory (CT) into the CSA (CCSA) for solving fractional optimization problems. The conventional CSA's parameters are tuned with CT to boost exploration/exploitation tendencies and improve the global convergence speed. In order to effectively tackle high-dimensional global optimization problems, an enhancement to CSA (ICSA) is presented by *Jain et al.* [182]. By including the adaptive adjustment operator, the experience factor, and the Levy flying distribution in the position update mechanism of the crows, the balance between the exploitation and exploration capacities of the CSA is enhanced. For a comprehensive review of other CSA variants and application scenarios, one can refer to [81]. Recently, *Awadallah et al.* [183] has introduced a cellular automata-based CSA (CCSA) where the framework of a cellular automata model is combined with the standard CSA to regulate its diversity throughout the search and hence increase its efficiency. The experimental findings over 23 standard benchmark functions proved the superiority of CCSA compared to other well-known algorithms. Overall, Table 2.3 presents a brief and informative summary of literature reviews that concentrate on the various variants and adaptations made to the CSA algorithm.

**Table 2.3:** Summary of recent advances in the CSA and their applications across various fields

Authors (Year)	Main Modification	Application	Remark
Althanoon et al. (2021) [162]	Opposition-based learning for flight length	Feature selection for classification	Superior feature selection performance
Chaudhuri et al. (2021) [163]	Time-varying flight length	Feature selection in packing mode	Prevented premature convergence effectively
Eliguzel (2021) [164]	Discretization, local search, and elitism strategy	P-median problem	Enhanced performance in solving the P-median problem
Grisales-Noreña et al. (2023) [165]	Master-slave methodology using CSA	Integration of photovoltaic generators into DC grids	Demonstrated applicability, efficiency, and robustness in minimizing operating costs compared to other methodologies.
Braik et al. (2023) [166]	Memory-based Hybrid CSA (MHCSA) combining CSA with PSO	Various benchmarks	Enhanced memory representation and balance between exploration-exploitation.
Kumar et al. (2023) [167]	Combining stacked autoencoders and CSA	Community detection in complex networks	Effectiveness in modularity matrix reduction and clustering optimization, surpassing traditional and contemporary algorithms.
Bai et al. (2023) [168]	Improved CSA named Cauchy variation optimization CSA (CM-CSA)	Topology optimization in IoT	Significantly reduces energy consumption and enhances connectivity, offering advancements for wireless sensor networks.
He et al. (2023) [169]	Multi-Stage CSA (MSCSA) with chaos and multiple opposition-based learning	Complex global optimization problems	Enhanced population quality by incorporating diverse search strategies.
Singh et al. (2023) [170]	Modified CSA	Voltage stability in power systems	Efficacy in selecting load buses for shedding, enhancing voltage stability while minimizing load shedding compared to other algorithms.
Rao et al. (2023) [171]	Probabilistic Simplified Sine Cosine CSA (PSCCSA)	Standard test functions and classic engineering problems	Demonstrates feasibility, effectiveness, and enhanced performance through a probabilistic simplified sine cosine approach.
Liu et al. (2023) [172]	Group Strategy CSA (GCSA)	Benchmark functions and engineering design problems	Enhanced solution diversity and search efficiency. Shows competitive performance against other algorithms.
Osei-kwakye et al. (2023) [173]	Hybrid PSO and CSA with clustering initialization strategy (HPSOCSA-CIS)	Feature selection on standard UCI datasets	Addresses premature convergence of BPSO, outperforming other hybrid and standard optimization algorithms in classification accuracy.
Andic et al. (2023) [174]	Application of CSA in power system state estimation	Power system state estimation with limited-channel PMUs	enhancing power system analysis and planning.

Continued on next page

**Table 2.3 continued from previous page**

<b>Authors (Year)</b>	<b>Main Modification</b>	<b>Application</b>	<b>Remark</b>
He et al. (2023) [175]	Multi-Stage Search Integration CSA (MSCSA)	Complex global optimization problems	Incorporates chaos and multiple opposition-based learning, showing competitive performance in tackling large-scale problems.
Zhao et al. (2023) [30]	VMCSA enhanced with VND and IEM strategies	Multi-level thresholding image segmentation, COVID-19 X-ray images	Addresses local optimality issues, showcasing superior performance in segmentation tasks.
Sheta et al. (2023) [178]	Integration of adaptive parameters ( <i>fI</i> and <i>AP</i> ) using three growth function variants	benchmark functions and engineering applications	superior performance in compared to the original CSA and other methods.
Gholami et al. (2021) [176]	Improved CSA (ICSA) with sophisticated updating technique	Not specified	Promotes convergence and enhances local search capabilities, outperforming traditional CSA and other meta-heuristics.
Ke et al. (2021) [177]	ICSA with value functions and reverse learning	Optimal design of commercial buildings	Delivers more precise findings and significantly reduces computing times, showcasing high efficiency.
Necira et al. (2021) [62]	Dynamic Crow Search Algorithm (DCSA) with dynamic adaptation of parameters	Not specified	Outperforms the conventional CSA by adapting flight length and attraction parameter, enhancing the algorithm's performance.
Cao et al. (2021) [179]	Improved CSA with PSO strategy and Gaussian function	Extreme learning machine neural network optimization	Boosted global search ability and convergence accuracy
Khalilpourazari and Pasandideh (2019) [180]	Sine-Cosine Crow Search Algorithm (SCCSA)	Not specified	Enhanced diversification and intensification capabilities, showing promising solutions compared to state-of-the-art methods.
Rizk-Allah et al. (2018) [181]	Chaos theory incorporated into CSA (CCSA)	Fractional optimization problems	Boosted exploration/exploitation and improved global convergence speed by tuning CSA's parameters with chaos theory.
Jain et al. (2017) [182]	Enhancement to CSA (ICSA) with adaptive adjustment, experience factor, and L'evy flying distribution	High-dimensional global optimization problems	Enhanced balance between exploitation and exploration capacities, improving performance in high-dimensional problems.
Awadallah et al. (2022) [183]	Cellular automata-based CSA (CCSA)	Standard benchmark functions	Increased efficiency and diversity regulation through a cellular automata model, proving superior to well-known algorithms.

#### 2.4.4 Applications and Developments in the CapSA

Recently, the CapSA has gained significant attention in the research community due to its exceptional capability in handling a wide range of optimization problems. Inspired by the intelligent foraging behavior of Capuchin monkeys, CapSA stands out with its unique searching mechanisms and adaptable approach. This section delves into the latest advancements in CapSA, shedding light on the adjustments, and hybrid variants that have been developed to enhance its capabilities in solving specific types of optimization challenges.

*Al-qaness et al.* [156] proposed a novel approach to predict wind power generation using an enhanced Artificial Neural Network (ANN) model. They employed a specific type of ANN called the Random Vector Functional Link (RVFL) network, which is well-suited for forecasting time-series data. To optimize the predictive capability of the RVFL network, they utilized the CapSA. The researchers tested their method using data from four wind turbines located in France. Their results showed that the RVFL model, when optimized using CapSA (CapSA-RVFL), performed significantly better than the original RVFL model. The results of this study show that CapSA can improve the performance of neural network-based forecasting models and could be used to solve real-world problems in renewable energy forecasting.

*Ali et al.* [155], in their 2022 study, put forward a new method for developing accurate electrical circuit models for photovoltaic (PV) power generation systems. This method tackles challenges like the unavailability of parameters and computational inefficiencies often faced in traditional methods. They used the CapSA technique to develop an optimization problem that minimizes the deviation between measured and simulated currents. CapSA is chosen for its simplicity and efficient exploration and exploitation balance. Testing on different PV models showed it outperformed traditional metaheuristic approaches with lower root mean squared error (RMSE) values. This study advances PV modeling methods and showcases CapSA's ability to solve complex engineering optimization problems.

In a significant advancement to the CapSA, *Abd Elaziz et al.* [184] proposed a wrapper-based feature selection (FS) technique. Their approach aims to improve the FS process by meticulously selecting the most relevant features. This approach addresses the challenges

associated with high data dimensions and the presence of features that are irrelevant or contain noise. The authors made several enhancements to CapSA. These enhancements entail dynamically adjusting the inertia weight, integrating sine-cosine accelerating factors, and employing a randomized learning technique. These advancements greatly enhanced CapSA's capacity to examine and utilize the solution landscape. The modified CapSA, termed ECapSA, was evaluated on several real-world datasets acquired from the UCI repository. The ECapSA outperformed conventional FS algorithms in these experiments, demonstrating its superiority as an FS tool. *Braik et al.* [185] suggested a modified Binary CapSA BCSA. To address the issues of local maxima and randomness, they incorporated Lévy flight and chaotic sequences into the foraging process. This resulted in two improved versions: Lévy-flight Capuchin Search Algorithm (LBCSA) which enhances search exploitation and exploration and Chaotic Binary Capuchin Search Algorithm (CBCSA) which amplifies search dynamics through chaotic mechanisms. Additionally, they combined these strategies into LCBCSA, leading to better diversity and an increased likelihood of finding optimal solutions. The proposed methods outperformed existing feature selection techniques in accuracy and fitness scores on a collection of twenty-six University of California Irvin (UCI) datasets. This represents a substantial improvement in the field of feature selection. In a different study, *Asgharzadeh et al.* [186] presented an improved version of the binary CapSA (BME-CapSA) for FS to enhance the accuracy of Intrusion Detection Systems (IDSs) in Internet of Things (IoT) devices. Aiming to tackle the rising security vulnerabilities in IoT infrastructures, the study focused on refining the FS process. The proposed BMECapSA algorithm, when coupled with a deep learning-based feature extraction process, proved effective in selecting features. The study demonstrated the potential of the enhanced binary CapSA as a promising approach for refining FS, thereby contributing to enhanced IoT security.

*Braik et al.* [159] have developed a novel approach to handle the optimization issues in Economic Load Dispatch (ELD) in power systems. ELD involves arranging the power generation from thermal units to meet the demand efficiently. To solve this, the authors proposed the Improved Hybrid CapSA (IHCSA), which initially introduces a memory component to enhance the algorithm's exploitation and adaptively adjust exploratory and exploitative ca-

pabilities during the search. The IHCSA aims to optimize the scheduling of thermal units to deliver desired power outputs while minimizing fuel costs. Additionally, the algorithm includes variable adjustments to maintain a balance between exploration and exploitation throughout the search process. Second, the algorithm combines the Gradient-Based Optimization (GBO) method and Local Escaping Operator (LEO) to improve its ability to conduct intensive searches, which allows for a more robust and effective search performance. The algorithm was tested in six scenarios with different generator counts and loading conditions and outperformed both the basic CapSA and other optimization methods in minimizing operational costs.

To optimize threshold levels in multi-level image segmentation, Zaki et al. proposed another notable hybrid approach using the CapSA [187]. The authors make an effort to address issues such as the uneven distribution of the initial population, limited global search capabilities, and premature convergence to local optima by introducing an Improved CapSA (ICAPSA). This improvement introduces a learning strategy based on chaos theory to set the initial positions of capuchins, which greatly improves the quality of the starting population. Moreover, ICAPSA incorporates a Lévy Flight disturbance strategy into the iterative update of positions, carefully balancing global and local search efficiencies. Using Kapur's entropy as the objective function, ICAPSA demonstrates its advanced segmentation abilities in multi-level thresholding for plant images. When compared against traditional CapSA, as well as other known methods, ICAPSA's performance stands out, showing superior visual segmentation effects and enhanced data metrics. (2023). In another hybrid approach, *Suba and Ahmad (2023)* [188] developed a new optimization method by combining the CapSA and the Wild Horse Optimizer (WHO) algorithms to enhance the performance of tandem perovskite solar cells (PSC). The combined method, known as CapSA-WHO, aimed to optimize solar cell efficiency by studying the composition of the perovskite solar cell and the impact of different materials on its performance. By implementing this technique, the researchers were able to identify optimal material compositions and structural configurations that resulted in improved solar cell efficiency with reduced computational effort. The findings from the application of the CapSA-WHO method demonstrated its effectiveness in achieving higher de-

vice performance compared to traditional optimization approaches. In a recent study, *Qtaish et al. (2023)* [189] investigated the improvement of the K-means clustering algorithm by developing a hybrid CapSA (HCSA) that combines features from the Chameleon Swarm algorithm to address issues related to local optima traps and initialization sensitivity. This novel approach aimed to leverage the adaptive movement behavior of capuchins, enhanced with a rotation mechanism, to improve exploration and exploitation across the search space. By combining these mechanisms, the algorithm's clustering efficiency was enhanced, providing expanded search capabilities and diversity. The HCSA has been proven effective in 16 diverse datasets, demonstrating its superior performance in clustering tasks when compared to traditional K-means and other meta-heuristic-based clustering methods. This research contributes to the continuous progress in CapSA applications, demonstrating its potential in addressing complex, real-world problems through enhanced meta-heuristic solutions. Table 2.4 provides an overview of recent studies focusing on various modifications made to CapSA and their applications.

The studies presented in Table 2.4 highlight the continuous progress and versatility of the CapSA. CapSA has been successfully applied to diverse areas, demonstrating its effectiveness in renewable energy forecasting, feature selection for IoT security, economic load dispatch in power systems, optimization of solar cell efficiency, and clustering tasks. Researchers have explored both basic applications of CapSA and innovative modifications, such as incorporating dynamic inertia weights and chaotic sequences, as well as hybridizing CapSA with other algorithms to enhance its performance. Despite advancements in the CapSA algorithm, there is a gap in research regarding its integration with an island model. This presents an opportunity to enhance CapSA's capabilities further. Our research introduces several novel improvements to CapSA, including a refined mechanism for updating followers and an enhanced local best perturbation strategy. Additionally, we integrate these enhancements with a dynamic migration-based island model, taking the algorithm beyond existing modifications. This approach aims to significantly improve the algorithm's efficiency, addressing the limitations of current models and setting a new benchmark for CapSA's application in intricate optimization problems.



**Table 2.4:** Summary of recent studies utilizing CapSA for various applications

Authors (Year)	Main Modification	Application	Remark
Al-qaness et al. (2022) [156]	Basic CapSA	Wind power prediction	Enhanced predictive accuracy of RVFL models
Ali et al. (2022) [155]	Basic CapSA	PV system modeling	Lower RMSE values, improved efficiency
Abd Elaziz et al. (2023) [184]	Dynamically adjusting inertia weight, sine-cosine acceleration coefficients, and incorporating a stochastic learning strategy	Feature selection	Improved handling of high data dimensionality and irrelevant features
Braik et al. (2023) [185]	Introduced binary CapSA (BCSA) with enhancements through Lévy flight and chaotic sequences	Feature selection	Higher accuracy and fitness scores
Asgharzadeh et al. (2022) [186]	Enhanced binary CapSA (BMECapSA)	IoT intrusion detection	Improved feature selection for IoT security
Braik et al. (2022) [159]	Incorporating a memory component, adaptive functions, and hybridizing with GBO and LEO for ELD	Economic Load Dispatch (ELD)	Outperformed CapSA and other methods in operational cost reduction
Zaki et al. (2023) [187]	Introduction of chaos theory-based learning strategy and Lévy Flight disturbance strategy for improved search capabilities	Multi-level image segmentation	Superior visual effects and enhanced data metrics
Suba and Ahmad (2023) [188]	Hybridization of CapSA with WHO algorithms for optimization of tandem perovskite solar cells	Optimization of perovskite solar cells	Improved solar cell efficiency with reduced computational effort
Qtaish et al. (2023) [189]	Combining CapSA with features from the Chameleon Swarm algorithm	K-means clustering tasks	Superior performance in clustering tasks compared to traditional methods

### 2.4.5 Review of Island-Based MHs

This section is dedicated to presenting a review of island-based parallel MHs and their applications. The island model has been applied in the literature with several MHs for various purposes. Some were tested on mathematical benchmark functions, while others were exploited to solve real-life optimization problems. Besides, multiple techniques have been used to construct and manage the migration process and its related factors. Moreover, different evaluation measures were adopted to prove the proposed models' effectiveness; some were concerned with improving the algorithm's speedup, while others were concerned with

obtaining a solution with good quality regardless of the computational time.

In the following subsections, we thoroughly review the main recent approaches proposed in the literature for the parallelization of MHs using the island model. We also identify some open challenges for future research directions.

#### **2.4.5.1 Mathematical Optimization Problems**

In the literature, a well-known set of mathematical functions is useful for assessing the behavior and characteristics of optimization algorithms such as robustness, convergence rate, and general performance aspects [103] [190]. These functions are modeled as a minimization problem where the global minimum (optimal solution) is known [142]. Based on their characteristics, they cover three major families: Uni-modal (UM), Multi-modal (MM), and composite MM functions. UM function has a unique global optimal solution and is usually useful for testing the exploitative capability of optimization algorithms. In contrast, MM functions have multiple local optima and are used to test the exploratory potential of algorithms. The third family is selected from the (IEEE CEC 2005, IEEE CEC 2009, IEEE CEC 2013) competition [191] [192] that covers hybrid rotated and shifted MM functions which are more challenging than traditional test functions. Details and formulas of these functions can be found in [92] [98] [93].

Some island-based MHs proposed in previous research have been benchmarked on the aforementioned functions. For instance, the island model concepts were integrated with the original Harmony Search (HS) to develop a new island-based version (iHS) with improved performance [193]. The random-ring migration topology and the best-worst selection replacement policy were utilized to exchange selected solutions among the islands. Extensive experiments based on simple factorial design were performed to investigate and analyze the effects of migration parameters (i.e.,  $M_f$ ,  $M_r$ ,  $I_n$ ) on iHS performance. Parameters that provided the optimal results were selected. Reported results revealed that the iHS is significantly sensitive to its parameters. The new variation iHS achieved good performance for global optimization functions. The same methodology was used to adapt the island model to build parallel variants of the Bat algorithm (called iBAT) [194], FPA (called IsFPA) [52],

and island-based ABC (iABC) [53].

Considering these works, the authors focused on applying the concepts of the island model without taking into consideration parallel computing (i.e., the authors did not implement their system on a parallel architecture). Hence, applying this approach to more challenging and time-consuming real-world problems will not be effective in terms of computational time, which is a critical factor in this kind of problem.

*Abed-alguni and Barhoush* [56] introduced a distributed version of the GWO (DGWO) by incorporating the basic island model concepts. The main objective is to enhance the diversity of the original GWO. The cooperative model was designed based on the random-ring topology and the best-worst migration scheme. The proposed DGWO is distinguished from the above mentioned works by utilizing a synchronous migration scheme that is actually applied on parallel platforms. The migration process is triggered periodically after a predefined number of iterations specified by  $M_f$ . Hence, after each  $M_f$  iteration, the islands are restructured randomly to form a ring topology and exchange solutions at the same time. Fifteen standard benchmarks were utilized for the sensitivity analysis, while 30 challenging CEC 2014 benchmarks were used to validate the performance of DGWO. Promising results in terms of solution quality and convergence behavior were achieved.

Following the same methodology, an island-based WOA (iWOA) parallel approach was introduced by *Abed-alguni et al.* [142]. The main aim was to maintain population diversity and reduce the required computational time to converge to high-quality solutions. Reported results emphasize the efficiency of incorporating the island model with WOA. Moreover, a modified variant of the Cuckoo Search (CS) algorithm utilizing disruptive polynomial mutation (CSPM) was incorporated with the island model to create a new method called iCSPM [195]. It was found that the island model enhances the ability of CS to preserve the diversity of its population.

Taken together, the works mentioned above are similar in that they design different experimental scenarios using a simple factorial design technique to investigate the sensitivity of the proposed models to migration parameters. However, this design does not consider the interaction between factors and may lead to incorrect conclusions [196]. Therefore, a full

or fractional factorial design augmented with appropriate statistical tests is recommended in this case.

Given the importance of the factors that affect the migration process, some studies have been conducted to carefully study how these factors affect the migration process. For instance, *Pais et al.* [51] and *Thaher and Sartawi* [59] provided interesting research to examine the effectiveness of island-based GA (PGA-I) and island-based ABC (iABC), respectively. To investigate the impact of migration parameters and their interactions, a statistically based full factorial design was used. To provide accurate speedup estimation that yields accurate and meaningful findings, different statistical concepts, such as the regression model and the Analysis of Variance (ANOVA), were utilized. These studies came to the conclusion that, depending on the problem at hand, different settings had different effects. Additionally, potential speedups for the PGA-I and iABC were observed.

The island model has been incorporated with different EAs for continuous optimization problems. Following a different approach, *Kushida et al.* [197] introduced an island-based Differential Evolution (DE). This approach is distinguished from the aforementioned papers by using heterogeneous island sizes (i.e., different-sized islands) and using different control parameters of DE for each island. Therefore, each island has varying convergence characteristics. In this approach, the size of each island is changed adaptively during the search process, utilizing an individual transfer policy.

Another interesting island-based approach has been recently proposed by *Turgut et al.* [198]. The authors embedded the concepts of the island model into the Crow Search Algorithm (CROW). They introduced four hierarchical migration topologies within the island model framework and evaluated their effectiveness using 45 optimization test functions, including classic benchmark optimization problems and CEC 2015 benchmark functions. They also applied these models to six multi-dimensional real-world optimal control problems: parallel reaction, continuous stirred tank reactor, batch reactor consecutive reaction, nonlinear constrained mathematical system, nonlinear continuous stirred tank reactor, and nonlinear crane container problems. The findings revealed that incorporating island model concepts significantly enhanced the optimization performance of CROW. The proposed island mod-

els either outperformed or matched the performance of six selected literature optimizers in 27-29 classic benchmark problems. Additionally, the inclusion of a master sub-population in the island model further improved optimization capabilities, consistently yielding better results in all optimal control problems compared to non-master sub-population models.

**Table 2.5:** Summary of main previous works that incorporated island model with MHs for optimizing continuous benchmark functions.

proposed approach	metaheuristic algorithm	benchmarks	migration policy		Evaluation measure	
			topology	selection-replacemnet	speedup	solution quality
iHs [193]	Harmony Search	25	random ring	best-worst		✓
island DE [197]	Differential evolution	9	random ring	best-worst		✓
iBAT [194]	Bat-inspired Algorithm	15	ring topology	best-worst		✓
IsFPA [52]	Flower Pollution Algorithms	23	random ring	best-worst		✓
DGWO [56]	Grey Wolf Optimizer	45	random ring	<i>best-worst</i>		✓
iWOA [142]	Whale Optimization Algorithm	18	random ring	<i>best-worst</i>		✓
iCSPM [195]	Cuckoo search with disruptive polynomial mutation	15	random ring	best-worst		✓
pGA-I [51]	genetic Algorithm	2	Single ring, All-to-all	random, best-worst	✓	✓
iABC [53]	Artificial Bee Colony	15	random ring	<i>best-worst</i>		✓
island CROW [198]	Crow search Algorithm	45	hierarchical	<i>best-worst</i>		✓
DM-LIMGA [199]	Genetic Algorithm	15	<i>master-slave</i>	DDMP policy		✓

In summary, as mentioned earlier, two possible evaluation criteria are used to evaluate the performance of parallel MHs: solution quality and computational time (e.g., speedup and efficiency). Most reviewed studies (as shown in Table 2.5) reported solution quality using a fixed number of iterations/evaluations. In practice, the parallelization scheme’s efficiency should be proven in terms of speedup and/or efficiency measures. Inspecting Table 2.5, it is clear that ring topology and the best-worst migration policy are extensively used in most studies. The ring topology is actually the simplest migration topology. However, the choice of topology has a significant impact on the quality of the obtained results [11]. Therefore, other structures such as star, grid, and array should be investigated. Moreover, to better prove the effectiveness of the proposed models, challenging real-world problems should be addressed.

### 2.4.5.2 Modern Applications of Island-Based MHs

Several island-based MHs approaches have been exploited to tackle a wide variety of challenging and real-world optimization problems in different domains. In this subsection, we briefly highlight the main applications, while Table 2.6 summarizes most of the approaches found in the literature.

## **Scheduling Problem Applications**

Some approaches have been applied to scheduling problems. *Doush et al.* [200] incorporated the island model concepts with a recently modified variant of HS (MHSNH) to propose a new parallel variant called iMHSNH. The proposed model was applied to handle the problem of flow shop scheduling with blocking. The authors utilized Taillard's benchmark, a *de facto* job scheduling dataset. The findings confirmed promising results, demonstrating the effectiveness of the proposed model in improving scheduling performance.

Another island-based model for scheduling problems was recently proposed by *Corcoran and Wainwright* [49]. They investigated the performance of an island-based GA on the problem of multiprocessor scheduling. The authors concluded that the parallel GA produced better-quality solutions than the sequential version. Recently, a variable-sized island model was incorporated with the DE algorithm without a migration process [201]. The proposed model was applied to solve the discrete-continuous scheduling problem with continuous resource discretization (DCSPwCRD).

For scheduling workflows in the cloud computing environment, *Alawad and Abed-alguni* [202] introduced a discrete variant of iCSPM proposed in [195], augmented with the opposition-based learning (OBL) method. The main contribution was the utilization of the OBL technique at the level of islands during the initialization steps. This novel approach had not been applied before. The proposed model, known as DiCSPM, demonstrated better performance compared to state-of-the-art methods.

## **Neural Network Training**

For training feed-forward neural network, *Thein* [203] applied an island-based DE model called Island DE Neural Network (IDENN). the proposed model was used to optimize the network parameters such as weights, learning rate, and momentum rate. Promising results in terms of error and convergence rate were achieved.

### **Shortest Common Supersequence (SCS)**

*Michel and Middendorf* [204] introduced an optimization model using island-based ACO for finding good parameters for the Shortest Common Supersequence (SCS) problem, which can be applied in mechanical engineering, molecular biology, and production planning.

### **Software Engineering**

In 2014, *Alshraideh et al.* [205] introduced a multi-population GA model for branch coverage test data generation used in the software testing phase. The model is based on a multiple path island GA. The proposed model was evaluated based on execution time, number of executions, time improvement, and search efficiency. The results demonstrated that the multi-population GA model outperformed traditional approaches, offering significant improvements in test data generation efficiency and effectiveness.

### **Industrial Manufacturing**

*Palomo-Romero et al.* [206] investigated the performance of a parallel GA based on the island model (IMGGA) for addressing the problem of unequal area facility layout (UA-FL). Well-known problems selected from the literature were used to validate the performance of the proposed IMGGA model. A full factorial design with 144 different configurations was applied to select suitable parameters for IMGGA, ensuring comprehensive evaluation and optimization of the algorithm's performance.

### **Other Applications**

Some proposed approaches were investigated using benchmarks and validated on challenging real-life problems. For instance, [194] applied the proposed iBA on three real-life cases of the economic load dispatch problem. Another parallel model was proposed by *Lardeux and Goëffon* [207] for solving the 0/1 knapsack and MAX-SAT problems. They introduced a dynamic island-based GA, with the main contribution being the allocation of adaptive migration probabilities to each edge based on the last migration effect. Recently, *Mohammed et al.* [208] investigated the island-based GA for solving the 3-SAT problem.

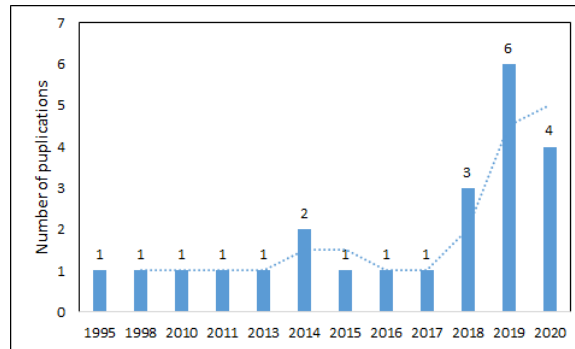
**Table 2.6:** Summary of main island-based metaheuristics designed for specific real-world applications

proposed approach	metaheuristic algorithm	problem	benchmarks	migration policy		Evaluation measure	
				topology	selection-replacement	speedup	solution quality
iMHSNH[200]	modified Harmony Search	flow shop scheduling with blocking	1 (Taillard)	random ring	best-worst		✓
island GA [49]	Genetic Algorithm	multiprocessor scheduling	4 cases	ring	fittest-worst	✓	✓
multi-size island DE [201]	Differential Evolution	discrete-continuous scheduling with continuous resource	-	-	-		✓
IDENN [203]	Differential Evolution	discretization (DCSPwCRD)					
island ACO [204]	Ant Colony Optimization	training neural networks	4 datasets	ring	best-worst		✓
island GA [205]	Genetic Algorithm	Shortest Common Supersequence	1	-	best-worst		✓
		branch coverage test	8 programs	complete graph	random-worst	✓	
		data generation					
IMGGA [206]	Genetic Algorithm	Unequal area facility layout	26	ring	best-worst		✓
DiCSPM [195]	Cuckoo Search	workflows scheduling in cloud computing	3	random ring	best-worst		✓
iBA [194]	Bat-inspired Algorithm	economic load dispatch	3 cases	ring	best-worst		✓
iCROW [198]	Crow search Algorithm	optimal control problems	6 problems	hierarchical	best-worst		✓
island GA [207]	Genetic Algorithm	0/1 knapsack and MAX-SAT	2	complete graph	best-worst		✓
island GA [208]	Genetic Algorithm	3-SAT problem	four suits	ring	best-worst		✓



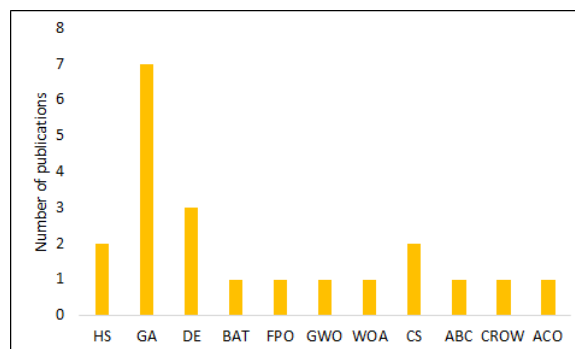
### 2.4.5.3 Summary of Reviewed Publications

This section discusses the expansion of island-based MHs models in the literature. Figure 2.10 depicts the number of publications per year that combine the island model with MHs. According to the findings, the number of publications remained almost constant (one publication per year) in the early years between 1995 and 2017. However, research in this domain has become very active in the last three years, reaching a peak in 2019 with six publications.



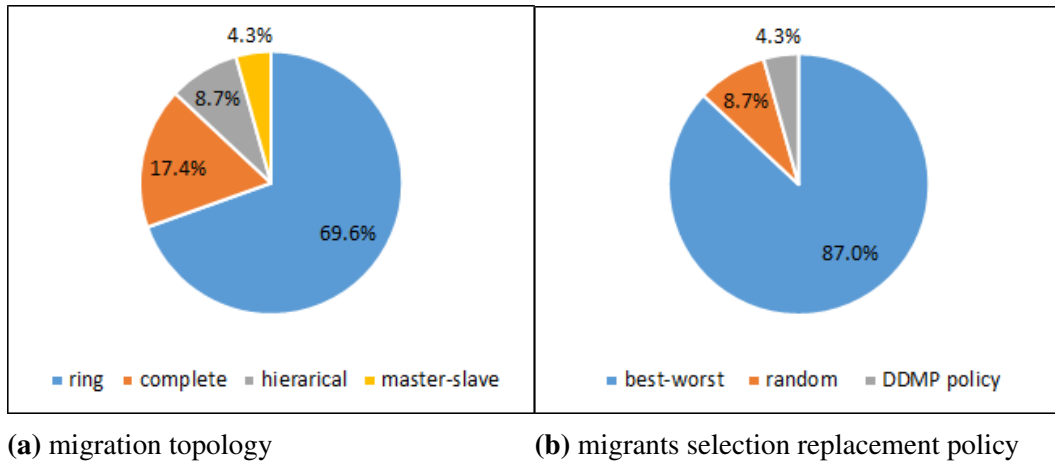
**Figure 2.10:** Number of island-based MHs publications per year

To understand the popularity of using MHs with the island model, Figure 2.11 shows the number of publications per algorithm combined with the island model. As can be observed, evolutionary algorithms (i.e., GA and DE) are used more frequently than swarm intelligence algorithms. GA is the most exploited algorithm, followed by DE, HS, and CS, respectively. In general, there is a relatively small number of MHs paralleled using the island model compared to the vast number of MHs in the literature. This leaves the research field open for implementation with many well-regarded MHs such as HHO, MFO, CSA, CapSA, and others.



**Figure 2.11:** Number of publications per each algorithm combined with island-based parallel model

Figure 2.12b summarizes the percentage of used migration policies (i.e., communication topology and selection-replacement strategy) in relation to the number of reviewed papers. These findings confirm the results presented in Tables 2.5 and 2.6. It is clear that ring topology and the best-worst migration policy are used extensively in most studies.



**Figure 2.12:** Distribution of used migration strategies

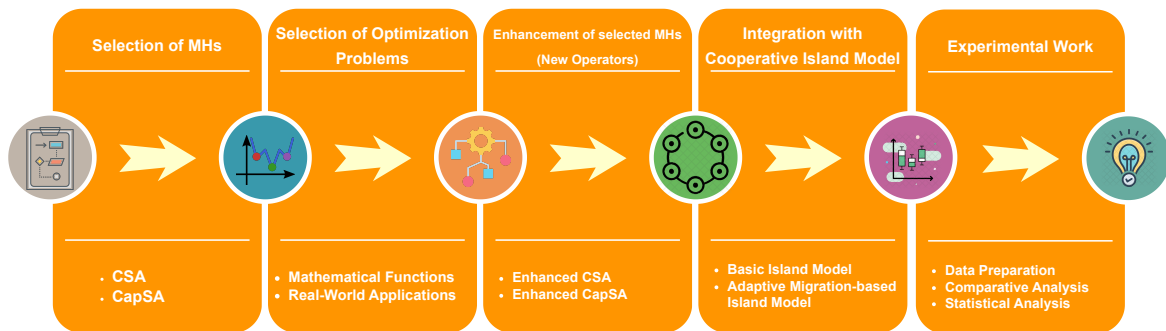
Despite the recent movement towards structured population approaches, specifically island-based models, most studies have incorporated the basic and fundamental versions of MHs with the island model without first focusing on enhancing the basic algorithm behavior. Additionally, most studies have used the basic island model with a fixed migration policy. The few studies that have switched to dynamic policies often do not consider comprehensive information from each island, such as population diversity and fitness values. In most studies, solutions are the primary information exchanged during the migration process. Only a minimal number of studies indicate the usefulness of context information (e.g., measures of population diversity). The exchange of information should be meaningful and timely; therefore, static or predefined migration frequency is not always practical. Introducing adaptive migration behavior based on the exchanged information would be an interesting development.

Furthermore, migration parameters are problem-dependent and should be appropriately tuned for each targeted problem. This tuning is not significantly applicable when solving large-scale, time-consuming real problems. Therefore, the dynamic adaptation of parameters in a reliable way is still limited. For instance, it would be interesting to propose a dynamic

model with migration rates. These challenges highlight potential areas for further research and improvement in the implementation of island-based models.

## Chapter Three: Methodology

This chapter outlines the methodology used to answer the research questions and achieve the study's objectives as illustrated in Figure 3.1. The primary objective of this study is to enhance the performance of two recent MHs, i.e., the CSA and CapSA, by incorporating new operators and the cooperative island model. One of the significant improvements is the development of an adaptive migration policy. This innovative approach aims to balance exploration and exploitation processes throughout the cooperative island framework, leading to more effective search and solution discovery. This will provide a parallel optimization model capable of handling a wide range of optimization issues in practical applications.



**Figure 3.1:** Overview of research methodology steps

The methodology is built upon three key components: research design, data collection, and data analysis. Each of these components is essential in verifying the efficacy of the suggested improvements and the newly proposed island migration strategy. In Section 3.1, the author discusses the foundation of this work. This section discusses the selection process and explains why CSA and CapSA should be improved. It details how new operators can enhance exploration and exploitation potential and how these algorithms can be combined into a cooperative island model. Additionally, it introduces the design of an innovative island migration policy that adaptively adjusts migration rates based on the dynamic assessment of population diversity and individual island fitness. Section 3.2 introduces a summary of the benchmark datasets used to validate the performance of the proposed models.

Lastly, the metrics and statistical analysis approach used to analyze the results from the experimental simulations are discussed in Section 3.3.

The approach utilized in this study is helpful since it establishes a connection between theoretical advancements and their actual application. Detailed explanations of the methods and assessments used to achieve the study objectives are provided in the subsequent sections.

## **3.1 Research Design**

### **3.1.1 Overview of the Research Approach**

This study employs a quantitative methodology, utilizing objective measures and statistical analysis of numerical data [209], to assess the efficacy of computational algorithms. By using this approach, a thorough assessment of algorithms' performance is conducted for a variety of optimization problems, which facilitates direct comparison between traditional and enhanced versions [131]. This approach is the perfect fit for this study as it emphasizes assessing the proposed improvements and their real-life impact. The use of quantitative methodology is extremely useful when seeking to confirm research hypotheses and establish a solid analytical foundation through techniques such as non-parametric tests [210]. Furthermore, it is necessary to create meaningful visualizations, such as diversity and convergence speed curves, to display the behavior of CSA and CapSA. Overall, to effectively answer the study questions and achieve the desired objectives, it is highly recommended to employ the quantitative research method.

### **3.1.2 Selection of Metaheuristic Algorithms**

This study focuses on enhancing CSA and CapSA among various types of MHs. Multiple factors contributed to the importance given to CSA and CapSA. These factors are listed below:

- Optimization problems that have been solved by CSA and CapSA have given positive results, but according to various studies and preliminary tests, integrating new operators can improve performance. It is an opportunity for exploring undiscovered

dynamics in exploration and exploitation, which is one of the main objectives of this study.

- Because of their well-known algorithmic flexibility, CSA and CapSA are readily adaptable for any type of optimization landscape. Moreover, they offer a convenient experimentation environment as well as creative innovation possibilities because they are simple and can be easily implemented. The novelty suggested in this study could be integrated into both CSA and CapSA through the incorporation of new operators and adaptive tactics due to its inherent flexibility.
- Utilizing the principles of the cooperative island model, in CSA and CapSA introduces a novel opportunity for research, enabling us to explore the impact of flexible migration rules on metaheuristic algorithms. This approach is expected to enhance the performance of these algorithms by leveraging the strengths of each algorithm within a collaborative and diverse island-based framework.
- The need for validation in optimization scenarios further justifies our selection. To conduct an extensive empirical evaluation, the study focuses on CSA and CapSA algorithms, comparing their enhanced versions with both state-of-the-art and traditional algorithms.

In summary, this study aims to advance CSA and CapSA as they have the potential to significantly improve performance, offer flexibility in tackling diverse optimization challenges, and employ a cooperative island model. These characteristics align with our objectives of developing optimization models that can effectively address the complexities encountered in real-world applications.

### **3.1.3 Proposed Enhancements in CSA: Introducing Novel Operators**

CSA drew the researcher's attention, so it has recently been enhanced through a number of approaches. As a particular class of MHs, CSA possesses impressive space exploration capabilities [30]. However, it has certain shortcomings [153, 30]. First, CSA is prone to stagnation in local optimum due to its constant awareness probability ( $A_p$ ) and flight length ( $Fl$ ).

Second, the crow search mode is somewhat limited (i.e., relatively single), which prevents it from efficiently locating the global optimal solution when tackling challenging problems [63]. In particular, the proposed operators of the original CSA let their search agents change their positions depending on random individuals and random probability to improve population variety. Thirdly, CSA ignores the importance of optimum solutions in population evolution, which results in difficulties like premature and sluggish convergence when attempting to solve complicated problems. As a result, CSA needs additional operators that will place an emphasis on exploitation and provide a better balance between diversification and intensification. Subsequent sections present proposed enhancements of the CSA.

### **3.1.3.1 Adaptive Tournament Selection Based Guided CSA (ATCSA)**

As previously explained in Section 2.4.1, in each iteration  $t$ , the crow  $i$  should select one of the flock crows (for example, the crow  $j$ ) and follow it to the location of its hidden food place. The guided crow  $j$  is chosen at random in the fundamental CSA. However, because random selection is used among the population, there is a significant chance of following a poor position (i.e. with a large fitness value considering the minimization problem). However, because it is performed randomly, many crows fail to advance because they choose the wrong target crow to follow. Actually, in this situation, many crows are flying to undesirable locations. Therefore, using random solutions to guide the optimization process results in difficulties while solving complex tasks, such as slow convergence and premature convergence [30].

There are different approaches that have been offered as ways to implement evolutionary selection mechanisms. The three most common approaches are a tournament, a roulette wheel, and a rank-based selection [6]. The idea behind the selection is "survival of the best," which states that the best solution has a more significant probability of being chosen and, as a result, produces a better population [11]. Nevertheless, the poorest possible solutions are not completely ignored, but they have a decreased probability of being chosen. Selective pressure, also known as the propensity to choose the fittest members of the existing population, is the main variable that has an impact on how effective the selection process is.

The harmony between intensification and diversification is impacted by the level of selection pressure. That is, too much pressure will lead to a bias toward the best-fit solutions, which will lead to a lack of diversity and premature convergence, whereas little pressure keeps diversity and slows convergence.

The tournament selection process proposed by *E. Goldberg et al* [211] is well-known for being straightforward, powerful, and the most prevalent selection strategy seen in EAs [212]. It is regarded as a two-step selection process. A group of  $k$  individuals is chosen at random from the existing population in the first phase, where  $k$  represents the size of the tournament. After that, the best-fit solution among those in the tournament is selected. By providing every individual an equal chance of getting chosen for the competition stage, this scheme's key advantage is that it preserves diversity. Algorithm 3 presents the pseudo-code of the standard tournament selection mechanism.

The tournament size ( $k$ ) is a crucial variable that is utilized to adjust the selection pressure and, as a result, the trade-off between exploitation and exploration. A bias toward the best solutions will result from greater values of  $k$  (more selection pressure), whereas a bias toward random behavior will result from lower values of  $k$  (lower selection pressure). However, figuring out the right value for the  $k$  parameter is difficult and depends on the kind of problem being handled [54].

---

**Algorithm 3** Pseudo-code of tournament selection [6]

---

```
//Tournament selection for one solution
Identify the tournament size  $k$ 
 $r$  = generate random index within [1, N]
set best =  $r$ 
set  $i$  = 2
while ( $i \leq k$ ) do
     $r$  = generate random index within [1, N]
    if ( $F(X_r) < F(X_{best})$ ) then
        best =  $r$ 
     $i = i + 1$ 
Return (best)
```

---

In this study, an adaptive tournament-based selection operator is utilized with the CSA to choose the guide solution. The proposed variant is called Adaptive Tournament Selection Based Guided CSA (ATCSA). This strategy selects  $K$  crows at random from the population



for each crow  $i$  at each iteration  $t$ . Then the fittest solution is picked to guide the crow  $i$ . One of the benefits of making this choice is that there is less of a chance of selecting an unsuitable target crow. In this scenario, the crows are able to improve their position by acting more effectively and following the crows that are better targeted, which ultimately leads to an enhancement in the convergence speed of the algorithm to some extent.

However,  $K$ 's value should be carefully considered. Intensification will result from larger values of  $k$ , whereas searches with smaller values of  $k$  will behave more randomly (i.e., with more diversification). To deal with this issue and strike a good balance between exploration and exploitation, we proposed a self-adaptive tournament-based selection scheme where the value of  $k$  is linearly increased over the course of iterations using Eq. (3.1). In this regard, to facilitate better solution space exploration, the value of  $K$  starts at a modest number at the beginning of the optimization process (i.e., in the early iterations). The  $K$  has a larger value as it increases in accordance with Eq. (3.1) in the final iterations, where precise searching around the best local optimums is critical. It is important to note that the original random selection process used in the basic *CSA* is a special case of the *ATCSA* when  $k = 1$ .

$$K = \text{round}(K_{min} + t * ((K_{max} - K_{min})/t_{max})) \quad (3.1)$$

### 3.1.3.2 Modified Random Movement CSA (MRCSA)

The basic *CSA* is designed to mimic two distinct actions shown by crows: pursuit and evasion. The behavior of evasion is modeled by implementing a random movement, which is then computed using a uniformly distributed random solution inside the search space. Although this updating mechanism improves the diversity of the population, it might result in difficulties like premature and slow convergence when attempting to solve complex problems because it ignores the importance of optimum solutions in population evolution [30].

To assist the basic *CSA* in avoiding these problems, the author introduced the Modified Random Movement *CSA* (MRCSA). In this regard, the rule given in Eq. (3.2) which is

derived from the HHO algorithm [92] is utilized to simulate the evasion behavior.

$$X_{i,t+1} = (X_{best,t} - X_{avg,t}) - r_1(LB + r_2(UB - LB)) \quad (3.2)$$

where  $X_{i,t+1}$  denotes crow' position vector in the next generation  $t + 1$ ,  $X_{best,t}$  refers to the global best solution so far,  $r_1, r_2$ , are randomly generated numbers within range (0, 1) in each generation, LB and UB denotes the lower and upper boundaries of decision variables, respectively,  $X_{avg,t}$  refers to the mean position of individuals in the current generation, which can be calculated using Eq. (3.3):

$$X_{avg,t} = \frac{1}{N} \sum_{i=1}^N X_{i,t} \quad (3.3)$$

where  $N$  indicates the population size, and  $X_{i,t}$  denotes the location of each crow at generation  $t$ .

This rule has the advantage of taking into account both the optimal position so far and the population's average position. Additionally, a randomly scaled component based on the range of decision variables is utilized. In order to investigate different areas of the search space and give additional diversification trends, two random coefficients,  $r_1$  and  $r_2$ , are considered for the component. Accordingly, the evasion rules in the basic CSA, as given in Eq. (2.7), are modified as shown in Eq. (3.4).

$$X_{i,t+1} = \begin{cases} X_{i,t} + r_i \times fl_{i,t} \times (m_{j,t} - X_{i,t}) & r_j \geq AP_{i,t} \\ (X_{best,t} - X_{avg,t}) - r_1(LB + r_2(UB - LB)) & otherwise \end{cases} \quad (3.4)$$

### 3.1.3.3 Enhanced Crow Search Algorithm (ECSA)

To improve the algorithm's ability to search globally and keep it from converging too quickly, both rules—adaptive tournament selection and modified random movement—are used to develop the Enhanced Crow Search Algorithm (ECSA), a better version of CSA. The pseudo-code of ECSA is presented in Algorithm 4.

---

**Algorithm 4** Pseudo-code of enhanced CSA

---

```
1: Input: define the adjustable parameters ( $N, d, t_{max}, K_{min}, K_{max}, fl$ , and  $AP$ )
2: Output: The best crow and its fitness value.
3: Randomly generate the initial positions of crows  $X_i(i = 1, 2, \dots, N)$ 
4: Initialize the memory of crows  $M_i = X_i(i = 1, 2, \dots, N)$ 
5: Compute the quality of each crow using the fitness function  $f(x)$ 
6:  $X_{best}$  = the optimal solution so far
7: while ( $t < t_{max}$ ) do
8:   Update tournament size  $K$  using Eq. (3.1)
9:   Compute the average position of all crows using Eq. (3.3)
10:  for  $i = 1$  to  $N$  do
11:    choose guided crow  $j$  using tournament selection
12:    if ( $r_j \leq AP_{j,t}$ ) then ▷ pursuit step
13:      Update the crow  $i$  based on Eq. (2.5)
14:    else if ( $r_j < AP_{j,t}$ ) then ▷ evasion step
15:      Update the crow  $i$  based on Eq. (3.2)
16:    Save the new positions if they are feasible
17:    Asses the quality of each new position  $f(x_{i,t+1})$ 
18:    Update the memory for each crow if  $f(x_{i,t+1})$  is superior to  $f(x_{i,t})$ 
19:    Update  $X_{best}$  if there is any better solution
20: Return  $X_{best}$ 
```

---

### 3.1.4 Advancements in CapSA: Novel Operators Integration

The CapSA is highly regarded by researchers and practitioners for its adaptability and effectiveness in handling various optimization problems. Since its inception in 2021, the CapSA has been widely applied in various research domains. The design takes inspiration from the clever foraging behavior of capuchin monkeys; utilizing social hierarchy and spatial awareness boosts the algorithms' capacity to explore and exploit the search space effectively. However, despite its numerous positive aspects, the algorithm still faces various issues [184, 159], especially under varying problem complexities. Similar to most MHs, CapSA is susceptible to the lack of diversity in its population. To be specific, CapSA has a tendency to become trapped in local optima, which could limit its effectiveness. These observations emphasize the importance of innovative enhancements to further improve its performance. To address these challenges and make the most of CapSA's capabilities, three significant enhancements have been introduced:

1. A refined follower update mechanism for improved exploration.

2. A novel followers' updating mechanism for improved local best perpetuation.
3. An adaptive dual update strategy incorporating an adaptive control parameter for dynamic strategy adaptation.

These enhancements focus on fine-tuning the algorithm's core mechanisms to achieve a better balance between exploration and exploitation, resulting in higher-quality solutions. The following sections provide an explanation of the theory behind the modifications, the mathematical formulation, and the anticipated impact on the algorithm's performance.

#### 3.1.4.1 A Refined Follower Update Mechanism (MCapSA1)

The follower updating mechanism is an important component of the search dynamics of the original CapSA. In particular, half of the population (for indices  $i > n/2$ ) are designated as followers, with their positions being updated by averaging with their adjacent neighbor's solution. Mathematically, this is represented as the position information of the  $i$ th follower is averaged with the position information of the preceding follower. This mechanism is inherently exploitative, with a strong emphasis on intensifying the search around the swarm's current positions. According to [213], employing this mechanism makes the followers' behavior pretty uniform, which might drive other followers to convergence to suboptimal solutions or entrapment in local optima when dealing with high-dimensional problems with lots of local optima. In addition, this might reduce population diversity, thereby hindering the algorithm's ability to investigate unexplored search regions.

To overcome these limitations, the first proposed modification of the CapSA, referred to as MCapSA1, is introduced. In this version, a refined follower update mechanism is proposed. The mathematical framework of the MCapSA1 remains consistent with the original CapSA in terms of leaders updating rules. However, the positions of the follower capuchins are refined using the following equation:

$$X_i(t+1) = \frac{(X_i(t) + X_{i-1}(t))}{2} + r(pb est_q - X_w(t)) \quad (3.5)$$

where  $X_i(t)$  represents the position of the  $i$ -th follower,  $X_{i-1}(t)$  indicates the position of the

preceding follower,  $pbest_q$  denotes the local best position of a randomly selected individual,  $X_w$  represents the position of a randomly selected individual within the swarm,  $q, w \in [1 \dots N]$ , and  $r$  signifies a random number uniformly generated within  $[0, 1]$ .

The addition of  $r(pbest_q - X_w(t))$  introduces a perturbation based on the DE concept, which is aimed at enhancing exploration by incorporating information from the local best positions and a randomly selected individual's position. In specific, the incorporation of the perturbation term, inspired by DE, greatly enhances the MCapSA1 by promoting a more dynamic exploration of the followers' neighborhood. This model combines traditional Newtonian motion with insights from DE, preserving the integrity of the followers' movement while enhancing the search by encouraging exploration in promising areas of the search space. As a result, this modification improves the balance between exploration and exploitation, maintains better diversity within the swarm, reduces the risk of premature convergence, and enhances the algorithm's ability to become more effective in navigating complex, multi-modal search spaces.

### 3.1.4.2 Enhanced Local Best Perturbation Strategy

As part of the ongoing development of CapSA, a new version named MCapSA2 has been introduced. This version features an innovative updating rule for followers, termed the enhanced local best perturbation strategy. While preserving the original leaders' position updating rules of CapSA, it completely transforms the followers' updating mechanism by incorporating a perturbation towards the local optimum, significantly improving the algorithm's exploration capabilities. The new updating rule is mathematically defined as:

$$X_i(t + 1) = pbest_i + r.(X_{rand} - X_i(t)) \quad (3.6)$$

where  $pbest_i$  denotes the local best position of the  $i$ th individual,  $X_{rand}$  is a randomly chosen position from a set of  $N$  solutions, and  $r$  is a random number uniformly distributed within the range  $[0, 1]$ .

The enhanced local best perturbation strategy employs a mathematical formula that introduces a dynamic exploration step around the local best positions, as shown in Eq. 3.6.

This change improves the exploration around the local best positions by adding a perturbation based on the idea of DE. Specifically, the additional term  $r \cdot (X_{rand} - X_i(t))$  dynamically modifies the step size for exploration, depending on how far the current position is from a randomly chosen solution. This approach enables a flexible step size (i.e., a flexible adjustment of the search radius) around the local best based on the landscape of the optimization problem and the current phase of the algorithm. This adaptability enhances the algorithm's ability to discover and exploit promising areas of the search space more efficiently than the original mechanism. In addition, this approach promotes greater diversity within the solution by enabling followers to explore their local best positions with an adaptive step size. The presence of a diverse population is essential to avoid getting stuck in suboptimal solutions.

In summary, the enhanced local best perturbation strategy achieves a better balance between exploration and exploitation. It uses the best local positions to guide the search process (exploitation) while also introducing variability and adaptability through the perturbation term to thoroughly explore the search space (exploration).

### 3.1.4.3 Adaptive Dual Update Strategy (MCApSA3)

Considering the strengths of the enhancements introduced in MCApSA1 and MCApSA2, the CapSA further evolves with the introduction of MCApSA3, which incorporates an adaptive dual update strategy. This strategy employs a dual mechanism for updating followers' positions and cleverly combines the refined follower update mechanism (Eq. 3.5) and the enhanced local best perturbation strategy (Eq. 3.6). By leveraging the unique advantages of both, MCApSA3 optimizes the search process. Utilizing multiple update mechanisms empowers the algorithm with a wider range of behaviors, enabling it to respond more flexibly to various challenges posed by optimization landscapes. The rule is defined as Eq. 3.7.

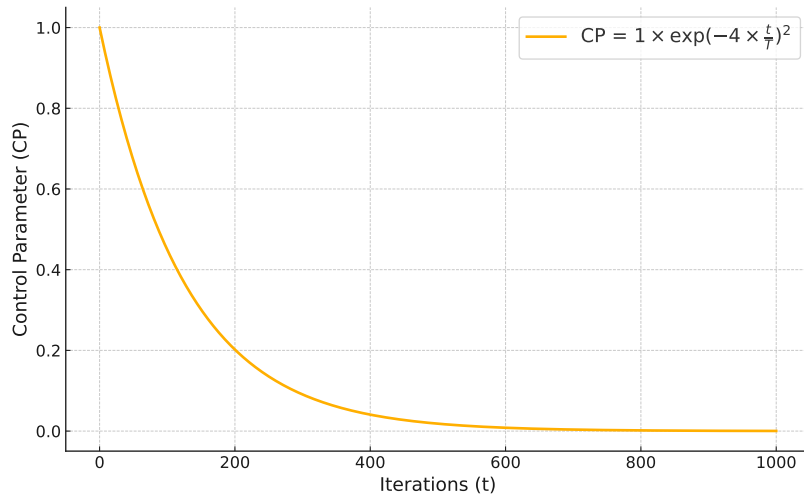
$$X_i(t+1) = \begin{cases} \left( \frac{X_i(t) + X_{i-1}(t)}{2} \right) + r \cdot (pbest_q - X_w(t)) & \text{if } r \leq CP, \\ pbest_i + r \cdot (X_{rand} - X_i(t)) & \text{otherwise.} \end{cases} \quad (3.7)$$

The updating rule for followers is adaptively guided by a control parameter (CP), which

is adaptively decreased over iterations according to the formula in Eq. 3.8.

$$CP = 1 \times \exp\left(-4 \times \frac{t}{T}\right)^2 \quad (3.8)$$

where  $t$  represents the current iteration, and  $T$  is the maximum number of iterations. This formula ensures that  $CP$  decreases exponentially over time as show in Figure 3.2.



**Figure 3.2:** Trend of control parameter  $CP$  over 1000 iterations.

The rule in Eq. 3.7 outlines how each follower's position is updated in the next iteration  $t + 1$ , depending on a crucial decision factor: the control parameter  $CP$  compared to a randomly generated number  $r$  within the range of 0 to 1. This formula is designed to balance two strategies for position updates, ensuring that the algorithm can effectively handle both exploration and exploitation during the optimization process.

1. When  $r \leq CP$ , the algorithm employs the specific refined follower update strategy given in Eq. 3.5 for updating the position of the  $i$ th follower. This strategy is designed to enhance the exploration of the search space by leveraging both the swarm's current position information and an additional exploration factor.
2. Conversely, when  $r > CP$ , a different strategy given in Eq. 3.6 is activated for the position update. This alternate approach focuses more on exploitation, aiming to refine the search around promising areas identified through the algorithm's exploration

activities.

The design of the adaptive control parameter, which decreases exponentially over time, ensures that the algorithm’s focus shifts from exploration to exploitation progressively, aligning with the evolving needs of the optimization process. Initially, a higher emphasis on exploration helps to discover diverse potential solutions across the search space. As *CP* diminishes, the algorithm increasingly concentrates on exploiting the most promising regions identified, fine-tuning the solutions to approach the optimal outcome.

### 3.1.4.4 Summary of the Introduced Modifications

In summary, to effectively overcome the limitations found in the original CapSA and fully utilize its capabilities in solving intricate optimization problems, this study presents three unique variations: MCapSA1, MCapSA2, and MCapSA3. Every variant includes a unique modification aimed at improving the algorithm’s performance. These modifications enhance exploration and exploitation capabilities and dynamically balance search strategies. In the rest of this thesis, the final modified version that combines all introduced enhancements will be referred to as Enhanced Capuchin Search Algorithm (ECapSA). Table 4.15 provides a clear summary of the modifications and their implications, showing how CapSA’s functionality and adaptability have been improved through an evolutionary approach. Algorithm 5 outlines the various steps involved in the proposed ECapSA.

**Table 3.1:** Summary of enhanced variants of CapSA and their implications

Variant	Refined Follower Update	Enhanced Local Best Perturbation	Adaptive Dual Update Strategy	Implications
CapSA	N	N	N	Baseline algorithm performance.
MCapSA1	Y	N	N	Improved exploration and convergence speed.
MCapSA2	N	Y	N	Enhanced exploitation and overcoming of local optima.
MCapSA3 (ECapSA)	Y	Y	Y	Better balance of exploration and exploitation, adaptively managed.

### 3.1.5 Integration with Cooperative Island Model

Following the strategic enhancements made to address specific limitations of the CSA and the CapSA, and to further amplify the performance and robustness of these algorithms, this study



---

**Algorithm 5** Pseudo-code of the enhanced CapSA (ECapSA)

---

```
1: Initialize the adjustable parameters of the basic CapSA
2: Generate the initial positions of the capuchins  $X_i (i = 1, 2, \dots, N)$  randomly.
3: Calculate the fitness of each individual capuchin
4: Initialize the velocity  $v_i$  and memory  $pbest_i$  of capuchins
5:  $gbest$  = the optimal solution so far
6: while ( $t < T$ ) do
7:   Update life time convergence  $\tau$  using Eq. (2.17)
8:   Update inertia weight  $\omega$  using Eq. (2.11)
9:   Update control parameter  $CP$  using Eq. (3.8)
10:  for  $i = 1$  to  $N$  do
11:    if ( $i < N/2$ ) then ▷ Leaders
12:      Update  $X_i$  following the leaders updating rules of the original
13:      CapSA through Eqs. (2.9) to (2.16).
14:    else ▷ Followers
15:      if ( $r \leq CP$ ) then ▷ Refined follower updating rule
16:        Update  $X_i$  based on Eq. (3.5)
17:      else ▷ Enhanced local best perputation rule
18:        Update  $X_i$  based on Eq. (3.6)
19:      Adjust  $X_i$  through the limits set for the variable's upper and lower boundaries.
20:      Evaluate the fitness of new position  $f(x_i)$ 
21:      if  $f(x_i) < f(pbest_i)$  then
22:         $pbest_i = x_i$ 
23:         $f(pbest_i) = f(x_i)$ 
24:      if  $f(x_i) < f(gbest)$  then
25:         $gbest = x_i$ 
26:         $f(gbest) = f(x_i)$ 
27:       $t = t + 1$ 
28:  Return  $gbest$ 
```

---

explores the integration of CSA, CapSA, and their enhanced variants within a cooperative island model framework. This integration seeks to enhance their exploration efficiency and the likelihood of discovering global optima.

CSA and CapSA, like other optimization methods, may initially find and converge to local optima, which is a limitation inherent to their search strategies. The cooperative island model provides a strong mechanism to address this challenge by promoting solution diversity through parallel evolution on separate islands. Each island evolves a population using either the base algorithm or one of its variations, enabling a combination of diverse exploration and exploitation strategies to be utilized across the search space. The use of the cooperative island model is highly beneficial for improving algorithmic performance for the following reasons:

1. **Diversity Management:** The island model's approach of dividing the population into smaller, isolated groups promotes independent exploration of various regions in the search space. This helps to minimize the risk of the entire population converging prematurely on local optima.
2. **Parallel Evolutionary Paths:** Enhancing the search process's efficiency and expediting it by exploring multiple promising areas concurrently.
3. **Migration Policies:** By incorporating migration policies, we can facilitate selective individual exchanges between islands, promoting the sharing of information and improving the global search capability.

### 3.1.5.1 Homogeneous Island Approache

Within the field of computational optimization, utilizing a homogeneous island approach in the cooperative island model framework uses a consistent algorithm variant to evolve the population on each island, ensuring a uniform evolutionary process across all islands while taking advantage of the specific strengths of the chosen algorithm variant.

For CSA and its modified versions (ATCSA, MRCSA, ECSA), as well as CapSA and its variants (MCapSA1, MCapSA2, ECapSA), employing the island model leads to the creation of island-based versions: *iCSA*, *iATCSA*, *iMRCSA*, *iECSA* for CSA variants, and *iCapSA*, *iMCapSA1*, *iMCapSA2*, and *iECapSA* for CapSA variants. This application follows a homogeneous approach. For instance, The flowchart of the proposed *iECSA* is depicted in Figure 3.3. In detail, the suggested *iECSA* follows the following steps:

- **Step 1: Initialize Problem and Adjustable Parameters**

This stage involves initializing the problem parameters, *ECSA* parameters, and the previously indicated migration parameters. First, the objective function  $f(x)$ , problem dimension ( $d$ ) and solution encoding ( $x = x_1, x_2, \dots, x_d$ ) are defined. The adjustable parameters of *ECSA* as described in Sections 2.4.1 and 3.1.3.1 must also setup in this step which include  $AP$ ,  $fl$ ,  $K_{min}$ , and  $K_{max}$ , and  $t_{max}$ . Additionally, it is necessary to set up the additional island model parameters described in Section 2.3 which comprise the

number of islands ( $s$ ), migration frequency ( $M_f$ ), migration rate ( $M_r$ ), interconnection topology, and migration policy.

- **Step 2: Generate the Initial Population of iECSA Randomly**

In this step, *iECSA* follows the standard *CSA*. Accordingly, a population of crows  $X = (x_1, x_2, x_3, \dots, x_N)$  are initially positioned randomly in the  $d$ -dimensional search space. These are the candidate solutions to the problem at hand. Each candidate solution ( $x_j$ , where  $j \in (1, 2, 3, 4, \dots, N)$ ) is determined using Eq. (3.9).

$$\vec{X}_j = \vec{X}_L + r(\vec{X}_U - \vec{X}_L) \quad (3.9)$$

where  $\vec{X}_L$  and  $\vec{X}_U$  are the lower and upper bound for the problem dimensions,  $r$  is a random number inside  $[0,1]$ .

- **Step 3: Split the Initial Population into Set of Islands**

The initial population is divided into  $s$  islands each of size  $I_s = N/s$ . For instance, let  $N = 9$ ,  $s = 3$ , then  $I_s = 3$ . Assuming that  $X = (x_3, x_1, x_5, x_8, x_7, x_2, x_6, x_9, x_4)$ , then *island*<sub>1</sub> =  $(x_8, x_6, x_1)$ , *island*<sub>2</sub> =  $(x_4, x_7, x_3)$ , and *island*<sub>3</sub> =  $(x_5, x_9, x_2)$ . Note that each solution is assigned to a random island.

- **Step 4: Optimization Process**

The optimization process starts in this phase by running the cooperative algorithms concurrently. Here, each island has its own population and search mechanism (i.e. *ECSA*). Therefore, each algorithm evolves its population using the mathematical operators of the *ECSA* as discussed in Section 3.1.3.3. It is important to note that the same search algorithm (e.g., *ECSA*) with the same settings is embedded with each island. Therefore, this study uses a homogeneous island model. In addition, the evolution process happens asynchronously based on the generalized island model [214, 215].

- **Step 5: Migration Process**

This process is triggered periodically after a predefined number of iterations as specified by the  $M_f$  parameter. A communication topology (e.g., bidirectional ring topol-

ogy) is generated randomly to interconnect the islands (i.e., identify the neighbors for each island). During migration, a percentage of selected individuals from each island, controlled by the  $M_r$  parameter, are transferred between the connected islands. The proposed model is implemented utilizing the Python Parallel Global Multiobjective (PyGMO) framework, which facilitates this information exchange through its parallel and distributed computing capabilities. Specifically, Pygmo employs message-passing and shared memory mechanisms to enable efficient communication and synchronization between parallel tasks. This approach ensures that relevant data and solutions are exchanged effectively, managing communication overhead and maintaining synchronization across different components of the model. The swapping process among every two neighboring islands is executed based on the best-worst policy.

- **Step 6: Check stop criterion:** Repeat steps 4-6 until the stop condition (e.g., maximum iterations) is not met.
- **Step 7: Return the best solution:** if the stop condition is met, return the best solution found in all islands.

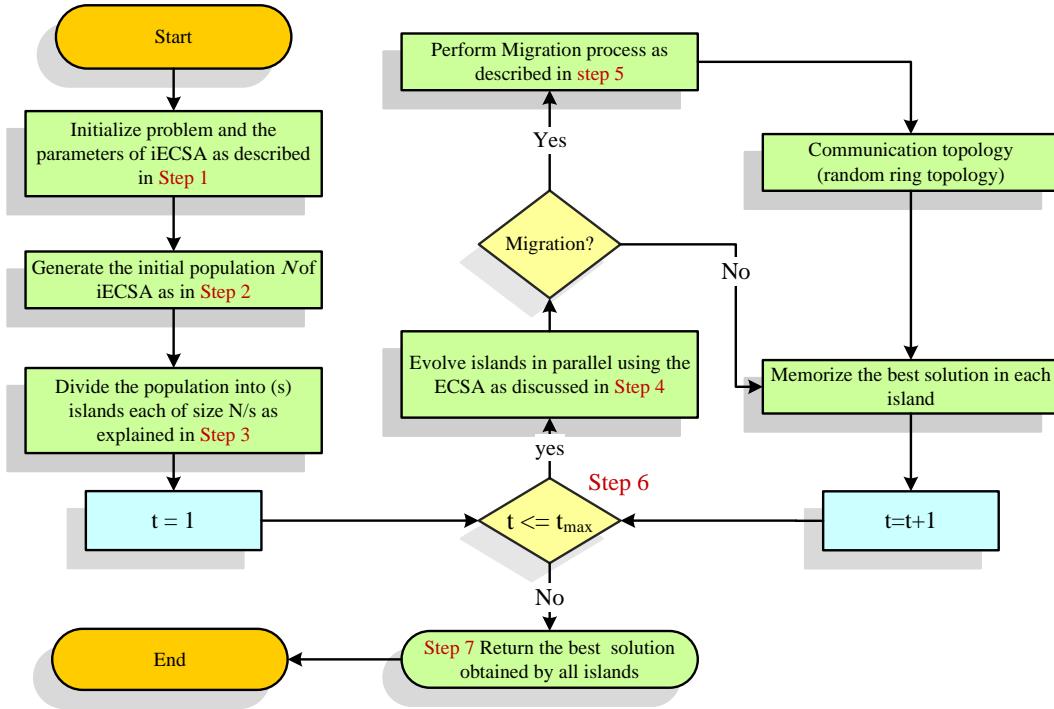
Within the framework of the homogeneous island approach, this strategy is replicated to develop island-based versions of CapSA, including *iCapSA*, *iMCapSA1*, *iMCapSA2*, and *iECapSA*. All of these variations follow the same basic steps, guaranteeing that the algorithm is applied consistently to all islands.

### 3.1.6 Mathematical Formulation of the Island-Based Model

The following mathematical formulations provide a detailed representation of the island-based model described earlier:

1. **Initialize Problem and Adjustable Parameters:** Let  $x_i$  represents a candidate solution in the search space  $X$ , where  $x_i \in \mathbb{R}^d$ . The initialization of the problem and ECSA parameters is defined as:

$$\text{Objective function: } f : \mathbb{R}^d \rightarrow \mathbb{R}$$



**Figure 3.3:** The general process diagram of the proposed *iECSA*

Problem dimension:  $d$

Solution encoding:  $x = (x_1, x_2, \dots, x_d)$

Adjustable parameters:  $\{AP, fl, K_{min}, K_{max}, t_{max}\}$

Island model parameters:  $\{s, M_f, M_r, T, Policy\}$

**2. Initial Population:** Generate initial population  $X = \{x_1, x_2, \dots, x_N\}$  with

$$x_j = \vec{X}_L + r \cdot (\vec{X}_U - \vec{X}_L)$$

**3. Population Division:** Divide the initial population into  $s$  islands, each of size  $I_s$ :

$$I_s = \frac{N}{s}$$

Assign solutions to islands randomly:

If  $X = \{x_1, x_2, \dots, x_N\}$ , then island  $I_k = \{x_{k_1}, x_{k_2}, \dots, x_{k_{I_s}}\}$  for  $k \in \{1, 2, \dots, s\}$ .

4. **Optimization Process:** Each island  $I_k$  applies the *ECSA* algorithm iteratively:

$$x_{k,j}^{t+1} = x_{k,j}^t + \Delta x_{k,j}$$

where  $\Delta x_{k,j}$  is updated based on *ECSA* operators:

$$\Delta x_{k,j} = \text{ECSA\_update}(x_{k,j}^t, \text{parameters})$$

5. **Migration Process:** Perform migration every  $M_f$  iterations. Define the communication topology  $T$  and perform migration:

$$T = \text{Randomly generated bidirectional ring topology}$$

Transfer a percentage  $M_r$  of individuals between connected islands:

$$\mathcal{P}_{k \rightarrow l} = \text{Top } M_r \text{ percentage of individuals from island } I_k \text{ to } I_l$$

Use the best-worst policy for swapping:

$$\text{Swapping: } x_{k,j}^{t+1} \text{ swaps with } x_{l,j}^{t+1}$$

6. **Stopping Criterion:** The optimization continues until:

$$t \geq t_{max} \text{ or convergence criteria are satisfied}$$

7. **Return the Best Solution:** After the stopping criterion is met, return the best solution

found:

$$x_{\text{best}} = \arg \min_{x \in \mathcal{P}} f(x)$$

where  $\mathcal{P}$  is the combined population of all islands.

### 3.1.7 Time complexity of the proposed iECSA model

The theoretical time complexity of the proposed *iECSA* model is calculated based on the steps outlined in the previous subsection. This analysis takes into account the initialization, population generation, optimization process, and migration operations across the parallel islands.

**Table 3.2:** Time Complexity Analysis of each step in the proposed *iECSA*

Step	Description	Time Complexity
1. Initialize Parameters	Setup problem parameters and iECSA parameters.	$O(1)$
2. Generate Initial Population	Create $N$ candidate solutions in $d$ -dimensional space.	$O(N \cdot d)$
3. Split Population	Divide the population into $s$ islands, each of size $I_s = \frac{N}{s}$ .	$O(N)$
4. Optimization Process	Evolve each island's population using ECSA.	$O(t_{\max} \cdot I_s \cdot (d + f(d)))$
5. Migration Process	Transfer individuals between islands periodically.	$O\left(\frac{t_{\max}}{M_f} \cdot s \cdot M_r \cdot \frac{N}{s}\right) = O\left(\frac{t_{\max}}{M_f} \cdot M_r \cdot N\right)$
6. Check Stop Criterion	Evaluate the stopping condition.	$O(1)$
7. Return Best Solution	Return the best solution found.	$O(1)$

The computational time of the island model using the *ECSA* is based on the steps detailed in Table 3.2. Since all islands run concurrently, the overall time complexity is equivalent to processing one island's population plus the migration overhead.

For each island, the *ECSA* is applied to a subpopulation of size  $I_s = \frac{N}{s}$ . Therefore, the time complexity for evolving the population in each island is:

$$T_{\text{island}} = O(t_{\max} \cdot I_s \cdot (d + f(d)))$$

where:

- $t_{\max}$  is the number of iterations.
- $I_s = \frac{N}{s}$  is the subpopulation size in each island.
- $d$  is the dimensionality of the search space.

- $f(d)$  represents the complexity of the fitness function evaluation (one evaluation) which depends on the problem being handled.

The migration process adds an additional overhead:

$$T_{migration} = O\left(\frac{t_{max}}{M_f} \cdot M_r \cdot N\right)$$

where:

- $\frac{t_{max}}{M_f}$  is the number of migration events.
- $M_r$  is the migration rate.
- $N$  is the total population size.

Combining these, the overall time complexity of the island model is:

$$T_{island} = O\left(t_{max} \cdot \frac{N}{s} \cdot (d + f(d))\right) + O\left(\frac{t_{max}}{M_f} \cdot M_r \cdot N\right)$$

This analysis captures the time required to handle the subpopulations within each island and the migration overhead across islands.

The island model is generally designed to improve solution quality and accelerate the search process to find better solutions in less time [51]. As noted in several studies [52, 55, 193, 194], communication overhead is often not considered, as the primary focus is on optimizing solution quality and search effectiveness. Our research primarily targets the first objective—achieving high-quality solutions. Furthermore, the real-world applications addressed in this study, such as neural network training, multilevel image segmentation, and software reliability growth model optimization, are not real-time applications. Consequently, the impact of communication overhead is relatively less critical in these contexts.

### 3.1.8 Development of Adaptive Island Migration Policy

Traditional approaches in the context of island-based models often employ a fixed migration rate. Although easy to implement, this one-size-fits-all approach does not adequately



address the varied and the dynamic nature of diverse optimization problems. Research underscores the critical impact of migration rate selection on island-based model performance [59]. Typically, most previous studies utilize hyperparameter tuning and factorial design to pinpoint optimal rates for varied scenarios [142, 52, 57, 194, 193, 216]. However, the process of tuning itself introduces a complex optimization challenge, especially for intricate and time-consuming problems. Specifically, fixed migration rates may lead to several key limitations:

1. **Inflexibility to problem complexity and dynamic environments:** Fixed rates do not adjust to the specific needs of varying problem landscapes, potentially leading to inefficient search strategies.
2. **Risk of premature convergence:** High fixed rates may cause rapid homogenization of the population, reducing diversity and increasing the risk of settling on suboptimal solutions early.
3. **Insufficient Exploration:** On the other hand, low fixed rates could prevent information from being shared between islands, which would prevent the exploration that is required to find global optima.

The limitations mentioned here emphasize the need for an effective adaptive migration strategy that takes into account the performance and characteristics of each island. Accordingly, this study introduces a novel adaptive migration policy that seeks to enhance the movement of individuals between islands. In this strategy, the migration rates are determined by considering two primary factors: the values of the objective function and measures of diversity. This allows islands with superior performance to distribute more individuals to other islands through higher migration rates, while islands with inferior performance experience lower migration rates. By assessing the performance of each island, the model ensures:

1. Islands with outstanding performance have a significant influence on the pool of migrants, encouraging the dissemination of promising solutions.
2. It is essential to maintain diversity across the cooperative islands to fully explore the search space and avoid premature convergence.

In the proposed strategy, when assessing the quality of each island, we consider both diversity and objective function values. Objective function values play a crucial role in assessing the proximity of a solution to the optimal one, providing insights into the solution's optimality and the convergence behavior of the island. In minimization problems, islands with lower objective function values tend to explore areas of the search space that lead to improved solutions. Furthermore, keeping track of these values over time proves valuable in evaluating the rate at which an island is making progress toward discovering effective solutions. A quick convergence is often a sign of an island that is doing exceptionally well. However, at times, the focus on objective function values can be deceptive, potentially causing the evolution process to become stuck in local optima. This happens when the search narrows down too quickly, missing out on possibly better solutions in unexplored areas of the search space.

Conversely, diversity quantifies the dispersion of solutions throughout an island. In order to guarantee a thorough exploration and to prevent being stuck in local optima, the island is actively investigating different parts of the search space. However, the overall quality of an island may not always be best measured by its high degree of diversity throughout time. An excessive amount of diversity in the search could be a sign of randomness, which could result in a less efficient investigation and a slower rate of discovering the optimal solutions. To maximize its success, an island must strike a delicate balance between two key factors: accurately assessing current performance through objective function values and maintaining enough diversity to facilitate future exploration. This equilibrium is crucial for avoiding the risks of both over-exploitation and the dangers of aimless exploration. As a result, under the optimization framework, islands that exhibit remarkable objective function values and diverse solutions are considered to be of high quality. However, achieving this ideal balance can prove to be a challenging task. This is due to the inherent conflict between two objectives, particularly when facing minimization problems, as explored in this study. In order to conquer this challenge, the proposed method utilizes a multiobjective optimization viewpoint to discover high-quality islands. The key to achieving optimal outcomes lies in striking a balance between considerable diversity and low objective values at the same time.

Using aggregation is a common approach to tackle problems that involve multiple objectives [217]. A popular method is to unify all objectives into a comprehensive function by assigning varying weights to each one and then summing them together. The objective with the greatest weight is deemed to be of utmost significance [8]. As such, our study utilizes the weighted aggregation process to assess the merit of an island in the cooperative optimization model. By employing this method, we can seamlessly incorporate competing objectives into one function, effectively ensuring both the solutions' quality and population diversity are given priority in minimization problems. Accordingly, in the proposed adaptable island migration technique, the migration rate can be adjusted using the formula shown in Eq. 3.10. This equation takes into account both the diversity and fitness of each island's population, enabling to score the overall quality of each island and its contribution to the migration process.

$$Migration\_Rate \uparrow = (1 - \alpha) \times (1 - Fitness) + \alpha \times Diversity. \quad (3.10)$$

where:

- $\alpha$  is a weight parameter that facilitates the control over the trade-off between diversity and fitness within the optimization process.
- Fitness and Diversity are normalized values representing the average fitness and diversity metrics of the island's population, respectively. To achieve this normalization, min-max normalization techniques, as shown in Eqs. 3.11 and 3.12

Normalization ensures that both fitness and diversity contribute meaningfully to the migration rate calculation. Fitness values are normalized using min-max normalization, where a lower raw fitness value (indicative of better performance) translates to a higher normalized value, closer to 1. The transformation  $1 - Fitness$  ensures that a higher normalized value emphasizes lower raw fitness values, aligning with the goal of maximizing the objective function (i.e. Migration\_Rate). On the other hand, diversity, being inherently positive and indicative of the population's spread within the search space, is used directly in its normalized form. It reflects the variety of solutions explored by an island, with higher values

suggesting a broader exploration.

$$\text{Normalized Fitness} = \frac{\text{Fitness} - \text{Fitness}_{\min}}{\text{Fitness}_{\max} - \text{Fitness}_{\min}} \quad (3.11)$$

$$\text{Normalized Diversity} = \frac{\text{Diversity} - \text{Diversity}_{\min}}{\text{Diversity}_{\max} - \text{Diversity}_{\min}} \quad (3.12)$$

In these equations, Fitness and Diversity represent the raw values of fitness and diversity, respectively.  $\text{Fitness}_{\min}$  and  $\text{Diversity}_{\min}$  are the minimum observed values for fitness and diversity across all islands, while  $\text{Fitness}_{\max}$  and  $\text{Diversity}_{\max}$  denote the maximum observed values. The resulting Normalized Fitness and Normalized Diversity are the values scaled between 0 and 1, ensuring that the metrics are on a comparable scale. This normalization facilitates their combined use in evaluating island quality and adjusting migration rates in the adaptive island model framework, providing a balanced consideration of both exploration and exploitation aspects. It is worth mentioning that fitness is calculated using the average fitness values of all individuals, which provides a measure of the overall performance of the entire population. Meanwhile, population diversity is calculated using an efficient measure, which will be detailed in Section 3.3.1.

The weight parameter  $\alpha$  plays a crucial role in balancing the emphasis between diversity and fitness. Values of  $\alpha$  closer to 1 give more importance to diversity, promoting exploration, while values closer to 0 favor fitness, encouraging exploitation and convergence towards optimal solutions. To address the challenge of tuning  $\alpha$  for different targeted optimization problems, and in line with the principle that exploration is favored in the initial iterations while exploitation becomes more pronounced as the process advances, an adaptive formula is employed to dynamically adjust  $\alpha$  over the course of the iterations. This adjustment ensures that  $\alpha$  decreases linearly, gradually shifting the focus from exploration towards exploitation. The formula for this adjustment is given by Eq. 3.13:

$$\alpha(t) = \alpha_{\max} - (\alpha_{\max} - \alpha_{\min}) \times \frac{t}{T} \quad (3.13)$$

where  $\alpha(t)$  denotes the value of  $\alpha$  at iteration  $t$ ,  $\alpha_{\max}$  refers to the initial (maximum) value of  $\alpha$  at the beginning of the optimization process,  $\alpha_{\min}$  is the minimum value of  $\alpha$  at the end

of the optimization process,  $t$  is the current iteration, and  $T$  is the total number of iterations.

Gradually reducing the value of  $\alpha$  from maximum to minimum value can help smoothly transition from the exploration phase to the exploitation phase. This adaptable modification of  $\alpha$  can be useful to meet the specific needs of different optimization landscapes. Once the calculation is completed, the result is set to a pre-defined range of 0.1 to 0.7 to carefully manage the migration rate within practical limits and maintain a balance between high and low values. This helps prevent loss of diversity due to excessively high rates and delay in convergence due to too low rates.

In summary, incorporating diversity and fitness into the migration rate calculation results in a formula that provides a comprehensive assessment of island quality. It tailors each island's contribution to the global research effort, ensuring a balanced and efficient exploration and exploitation strategy that is customized to the changing conditions of the optimization landscape.

### 3.1.9 Real-world Problems Selection

In addition to theoretical analysis, our proposed algorithms are extensively applied in real-world situations. These algorithms confidently tackle intricate optimization challenges in a multitude of domains:

- **Training of Feedforward Neural Networks:** This involves optimizing the weights of the network to minimize the difference between the predicted output and the actual output. Optimization in this context means adjusting the weights and biases to reduce errors, enhancing the network's ability to generalize from training data to unseen data.
- **Multilevel Thresholding Image Segmentation:** Segmentation divides an image into multiple regions or levels that represent different objects or features. Optimization here aims to select threshold values that maximize the accuracy of object detection and classification within an image. It seeks to improve the clarity and relevance of the segmented areas, facilitating easier analysis and interpretation.
- **Estimation of Parameters of Software Reliability Growth Models (SRGMs):** In

this application, optimization involves fine-tuning the parameters of reliability models to best fit historical failure data. This helps predict future behavior of software reliability more accurately. Optimization ensures that the model parameters provide the most reliable predictions about software failures, enhancing decision-making in software development and maintenance.

These specific problems were carefully chosen for several reasons. Firstly, they are highly relevant and essential, having a direct impact on the contemporary scientific and technological issues. Secondly, they provide the perfect opportunity for unbiased and comprehensive evaluation against existing methods. Moreover, they exhibit diversity in terms of application areas, effectively demonstrating the versatility of our algorithms. Lastly, the selected problems vary in computational complexity, allowing for a thorough examination of our algorithms at different levels of difficulty.

More details about the problem formulations and how the proposed optimization models can be adapted for these kinds of problems will be presented and explained in Chapter 4.

## **3.2 Data Collection**

For empirical studies, data collection is essential. It enables us to assess hypotheses and verify suggested approaches. The data collection phase of this study was carefully designed to evaluate enhanced MHs through a comprehensive methodology. This entails selecting from a wide range of applied optimization issues, including neural network training, image segmentation, and SRGMs, as well as theoretical challenges, such as real-valued mathematical functions. The datasets used include 53 benchmark functions—23 standard unimodal and multimodal functions, and 30 from the IEEE CEC2014 suite with unimodal, multimodal, hybrid, and composite functions [218, 219]. For real-world applications, the datasets encompass 10 biological datasets from the UCI repository for neural network training [220], 10 COVID-19 CT scan images for image segmentation [221, 222], and 10 historical faults datasets for optimizing SRGMs. Table 3.3 summarizes the datasets and benchmarks used in this study. More details about these datasets and how they are employed in the experiments will be presented in Chapter 4.

**Table 3.3:** Summary of Datasets and Benchmarks Used

Application Area	Dataset Description	Source	Purpose
Neural Network Training	10 biological datasets	UCI Repository [220]	Used for training and testing feedforward neural networks
Image Segmentation	10 COVID-19 CT scan images	sourced from [221, 222]	Employed for developing and evaluating multilevel thresholding techniques
Software Reliability Growth Models (SRGMs)	10 historical faults datasets	Various sources	Used for estimating parameters of SRGMs to predict software reliability
Benchmark Functions	23 standard unimodal and multimodal functions, 30 functions from IEEE CEC2014 suite (unimodal, multimodal, hybrid, composite functions)	IEEE CEC2014, Standard Benchmark Sources [218, 219]	Used to assess the performance of optimization algorithms under various complexity scenarios

### 3.3 Data Analysis

This section presents the methodology used to analyze the experimental data to assess the effectiveness and robustness of the enhanced optimization methods. To guarantee a diverse evaluation, we employed various quantitative and qualitative measures particular to each handled task. This section provides an overview of these measures, while the detailed examination, including mathematical formulas and comprehensive descriptions, is presented in Chapter 4.

#### 3.3.1 Performance Metrics

The evaluation metrics used in this study are crucial for assessing the performance of the optimization algorithms across various domains. These metrics have been carefully selected to reflect the specific characteristics and requirements of each domain. A summary of the evaluation metrics employed for each domain is provided in Table 3.4. For detailed explanations and mathematical formulations of these metrics, please refer to Chapter 4.

Table 3.4 summarizes the performance metrics used across different domains in this study.

#### 3.3.2 Statistical Analysis Methods

Due to the stochastic nature of the examined MHs, it is essential to adopt a robust approach for statistical analysis. Given that results from these algorithms may vary across multiple runs, our approach involves averaging findings from 30 independent runs for each algorithm.

**Table 3.4:** Summary of Evaluation Metrics Across Test Domains

Test Domain	Fitness Values	Quality Measures
Mathematical Benchmarks	Fitness Value (as obtained from benchmark function formula)	N/A
Neural Network Training	MSE (Eq. 4.29)	Accuracy (Eq. 4.13), F1-Score
Image Segmentation	Kpur Cross-Entropy (4.23)	SSIM (Eq. 4.24)
SRGM Optimization	MSE (Eq. 4.25)	VAF (4.30), correlation coefficient R (4.31)

Note: N/A indicates "Not Applicable"

Recently, the science of computational intelligence has been more interested in non-parametric statistical analysis methods, particularly in situations where the data may not satisfy the strict normal distribution assumptions [223]. Consistent with this approach, we utilized two well-known non-parametric tests, namely the **Friedman** test and the **Wilcoxon rank sum** test, both with a 5% level of significance, to calculate the overall rank of each examined algorithm and to determine specific pairwise comparisons.

### 3.3.2.1 Wilcoxon Rank Sum Test

The Wilcoxon Rank Sum Test, sometimes called the Mann-Whitney U test, is a common nonparametric test used to investigate if there is a significant difference in the population median ranks of two independent samples. This is done to judge whether the samples are likely to come from the same population. The null hypothesis ( $H_0$ ) and the alternative hypothesis ( $H_1$ ) for the test are defined as follows:

- $H_0$ : There is no difference in the median values of the two samples.
- $H_1$ : There is a difference in the median values of the two samples.

Mathematically, the test statistic  $U$  is calculated using the formula in Eq. 3.14:

$$U = \min(U_1, U_2) \quad (3.14)$$

where  $U_1$  and  $U_2$  are defined in Eqs. 3.15 and 3.16.

$$U_1 = n_1 \times n_2 + \frac{n_1(n_1 + 1)}{2} - R_1 \quad (3.15)$$



$$U_2 = n_1 \times n_2 + \frac{n_2(n_2 + 1)}{2} - R_2 \quad (3.16)$$

where  $R_1$  and  $R_2$  represent the sum of ranks in the first and second samples, respectively.  $n_1$  and  $n_2$  denote the number of observations in the two samples. Once  $U$  is calculated, the next step involves converting this statistic into a p-value to test the null hypothesis. This involves using the distribution of ( $U$ ) under the null hypothesis, which the normal distribution can roughly approximate for larger sample sizes. For a more detailed explanation of these calculations and the statistical theory behind them, readers can refer to [224].

Wilcoxon Rank Sum test is particularly useful when assessing the efficacy of algorithms on specific benchmarks or datasets where the samples are independent and may not adhere to normal distribution. In the realm of MHs, each sample comprises results from 30 independent runs, making this test crucial for highlighting significant differences in performance metrics between enhanced algorithms and their counterparts [131].

### 3.3.2.2 Friedman test

The Friedman test [225, 226] is employed when assessing the performance of different algorithms across multiple benchmarks or datasets. This test computes the ranks of each algorithm across all datasets and then analyzes the differences in these ranks. The test statistic  $\chi_F^2$  is computed as:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[ \sum_{j=1}^k R_j^2 - \frac{k(k+1)^2}{4} \right]$$

where  $N$  is the number of datasets,  $k$  is the number of algorithms, and  $R_j$  is the sum of the ranks for the  $j$ -th algorithm. This non-parametric test is invaluable for determining if there are significant differences in the algorithms' performance across various benchmarks. For instance, the test offers valuable insights into the effectiveness of popular algorithms such as CSA, CapSA, ECSA, and ECapSA when evaluated across various benchmarks. Employing the Friedman test allows us to calculate the average rank of each algorithm based on its performance on these benchmarks. This analysis provides a robust metric to identify which algorithm consistently outperforms the others.

## Chapter Four: Results

---

This chapter provides a detailed presentation of the results from extensive research focused on enhancing CSA and CapSA, as well as integrating these improvements within an island model framework. It is organized around three key areas of innovation: the enhancement of the CSA, advancements in the CapSA, and the introduction of an adaptive island-based migration policy. The discussion is structured into four well-defined sections, each dedicated to examining these contributions, their interdependencies, and their combined effect on optimization in the field.

Initially, Section 4.3 examines the enhancements made to the CSA, assessing how these improvements contribute to better algorithmic performance and the advantages of incorporating these enhancements within a basic island model. Section 4.4 delves into advancements in the CapSA and how they can be utilized in an island-based framework to achieve better optimization outcomes. Section 4.5 presents a comparative analysis where the enhanced island-integrated variants are tested against each other under different migration policies. This comparison is essential for understanding the impact of migration flexibility on optimization success. Finally, Section 4.6 evaluates the practical performance of these enhanced models in various real-world applications, including neural network training, image segmentation, and software reliability growth models.

### 4.1 Common Experimental Setup

All of the experiments in this thesis were conducted in a common computing environment to ensure that the comparisons were accurate and reliable. The following setup was used across various tests:

#### 4.1.1 Environment and Tools

All algorithms were developed and tested on a machine running Ubuntu 20.04 LTS, equipped with an Intel(R) Core(TM) i7-1165G7 CPU running at 2.80 GHz (8 CPUs), and 16 GB of

RAM. The algorithms were implemented in Python, utilizing the PyGMO library, which is a scientific library developed for massively parallel optimization [214]. PyGMO is built around the idea of providing a unified interface to optimization algorithms and problems, facilitating their deployment in massively parallel environments. For this research, the core of PyGMO was customized to develop the parallel island model based on the generalized island-model paradigm [215]. This customization involved adapting PyGMO's existing tools to better suit the specific needs of the enhanced CSA and CapSA variants and their integration into the adaptive island-based framework.

#### 4.1.2 Algorithm Configurations

Unless otherwise stated, all algorithms were evaluated utilizing identical standard configurations and settings customized for the specific problem types. For real-valued mathematical problems, the population size and the maximum number of iterations are initially set to 30 and 500, respectively. For more extensive testing, a configuration of 70 population size and 1000 iterations was employed. These parameter values are often used in the field of metaheuristic optimization. They were chosen because they have been widely used in previous studies that examined how metaheuristics perform on similar problems [39, 80, 131]. Such values are recommended widely in the literature to ensure a balance between computational efficiency and the quality of solutions. The detailed specific parameters of the suggested *iECSA*, *iCapSA*, and the competitor algorithms are reported in Table 4.1.

Due to the stochastic nature of the examined MHs, all findings are compiled from the results of 30 independent runs, where the average of the best solution (i.e., the minimum) attained so far in each iteration, as well as the standard deviation, are reported. Please note that in all tables presented, **AVG** denotes the average results of 30 independent runs, and **STD** represents the standard deviation, which provides an indication of the variability of the results across these runs. The most favorable findings are shown in **boldface**. Recently, the science of computational intelligence has been more interested in non-parametric statistical analysis [223]. In this regard, two non-parametric tests, namely the Friedman test [225] and the Wilcoxon rank sum test [210], both with a 5% level of significance, were utilized

**Table 4.1:** Parameter settings of iECSA and other algorithms

Common parameters		
Population size $N$		30, 50, 70
Maximum No. of iterations		500 for standard functions 1000 for CEC2014
No. of runs		30
significance level $\alpha$ (Friedman test)		0.05
Internal parameters		
Algorithm	parameter	value
iECSA	flight length $fl$	2
	Awariness probability $AP$	0.1
	Tournament size $K$	Increased linearly from 1 to $0.5 \cdot N$
	Number of islands	2, 4, 6
	Migration frequency $M_f$	25, 50, 100
	Migration rate $M_r$	0.1, 0.2, 0.3
	Migration policy	best-worst
PSO [31]	Comunication topology	random ring
	inertai weight ( $w$ )	decreased linearly [0.9 0.2]
	cognitive constant ( $c_1$ )	2
WOA [80]	social constant ( $c_2$ )	2
	convergence constant $a$	decreased linearly [2 0]
BAT [126]	Spiral factor $b$	1
	$Q_{min}$ Frequency minimum	0
DE [105]	$Q_{max}$ Frequency maximum	2
	loudness $A$	0.5
	Pulse rate $r$	0.5
	Mutation factor	0.5
SCA [227]	Crossover ratio	0.7
	$r1$	decreased linearly [2 0]
HGS [131]	$r2$	random values inside [0 $2\pi$ ]
	$r3$	random values inside [0 2]
	$r4$	random values inside [0 1]
	hunger threshold ( $LH$ )	10000
HHO [39]	probability of updating position $l$	0.08
	convergence constant ( $E$ )	decreased linearly [2 0]
GWO [127]	convergence constant ( $a$ )	decreased linearly [2 0]
AO [228]	$r_1$	10
	$U$	0.00565
	$w$	0.005
	$\alpha$ and $\delta$	0.1
	$G_2$	decreased linearly [2 0]
CapSA [46]	$a_1, a_2$	1.25, 1.5
	$P_{ef}, P_{bf}$	11, 0.7
	$w_{min}, w_{max}$	0.1, 0.8
EO [229]	$a_1, a_2$	2, 1
	$V$	1
	$GP$	0.5
AOA [230]	$\alpha$	5
	$\mu$	0.499
	$mop_{min}, mop_{max}$	0.2, 0.9
FPA [231]	$ps$	0.8
MFO [128]	$a$	[-2, -1]
	$b$	1
SHIO [232]	$a$	decreased linearly [1.5 0]

to calculate the overall rank of each examined algorithm and to determine specific pairwise comparisons. In the presented tables, please note that the symbols '+', '-', and '≈' have been utilized to indicate that the proposed enhanced variant has statistically significant superiority over certain alternative methods (+), while also displaying statistical inferiority to others (-). Additionally, instances where the performance does not notably differ from that of competing approaches are also highlighted as (≈). Moreover, instances marked with "NaN", which refers to "Not a Number", show that there is no significant difference between the compared approaches. In general, a 'NaN' p-value suggests from the Wilcoxon rank-sum test suggests that the data in both samples are nearly identical.

This thesis focused on the solution quality measure by using a fixed number of iterations for all tested algorithms [11, 51]. Accordingly, different quantitative and qualitative metrics such as fitness values, convergence curves, and diversity curves are used to assess the effectiveness and robustness of the proposed methods.

## **4.2 Experimental Results and Simulations on Mathematical Benchmarks**

### **4.2.1 Mathematical Benchmark Functions Set**

Evaluating the behavior and characteristics of optimization algorithms, including their robustness, convergence rate, and overall performance factors, is commonly done in the literature by employing various mathematical optimization problems. These problems are characterized as minimization problems, where the optimal solution is referred to as the global minimum. This study selects two extensively researched sets of diverse benchmark functions to evaluate the effectiveness of the suggested models: the standard benchmarks and the challenging IEEE Congress on Evolutionary Computation (CEC) functions introduced in 2014.

#### **4.2.1.1 Standard Benchmark Functions**

The first set contains 23 functions (F1-F23), which are standard benchmark functions used extensively in the literature of optimization [218, 103]. This benchmark collection includes two primary categories: unimodal ( $U$ ) and multimodal ( $M$ ). The  $U$  functions (F1-F7) have

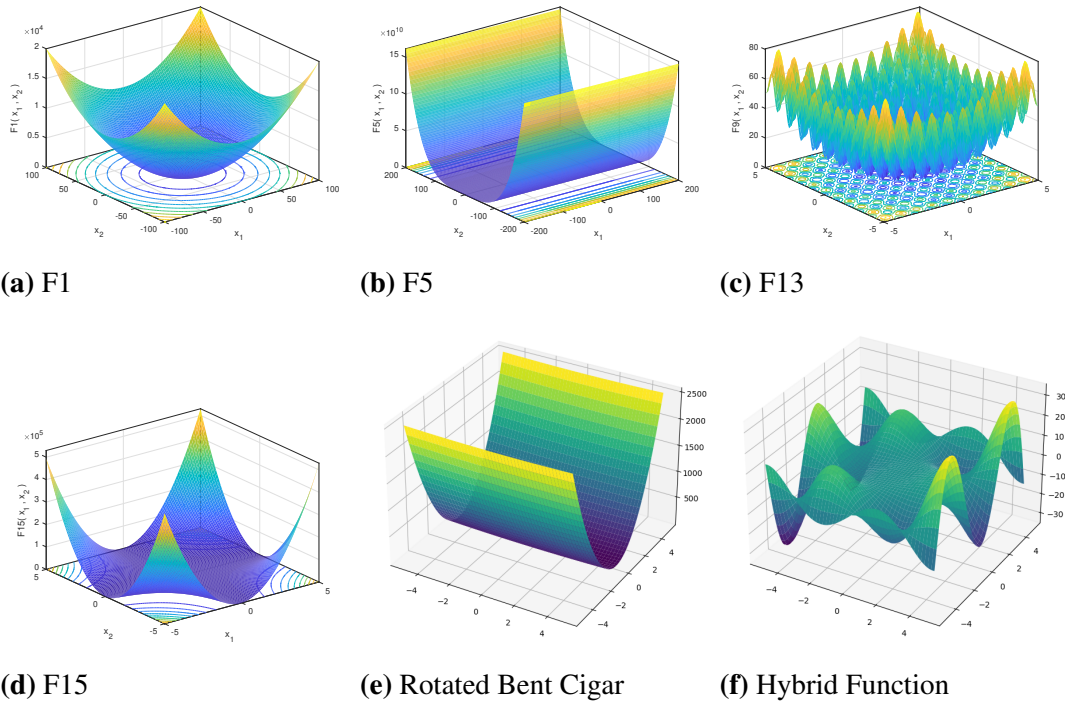
a unique global optimum solution and are typically helpful for examining the intensification potential of optimization techniques. In contrast, because they contain numerous local optima,  $M$  functions (F8-F23) are used to evaluate the diversification capabilities of algorithms. Tables A-1, A-2, and A-3 in *Appendix A* demonstrate the mathematical formulation and properties of the standard  $U$  and  $M$  problems.

#### **4.2.1.2 Challenging IEEE CEC 2014 Functions**

The second test suite of mathematical problems was chosen from the IEEE CEC 2014 competition [219], which includes 30 benchmark functions (F1–F30). These benchmark functions are modified versions of other challenging mathematical optimization problems that have been rotated, shifted, extended, and combined. Accordingly, they may be divided into four groups according to their characteristics: unimodal functions (F1-F3), multimodal functions (F4-F16), hybrid functions (F17-F22), and composition functions (F23-F30). These examples are also used in many other publications, and they may demonstrate how well the algorithms work at balancing exploration and exploitation tendencies and avoiding local optimums in difficult situations. The hybrid functions (F17-F22) are designed to test the algorithms' ability to handle optimization landscapes with a mix of different characteristics. These include varying levels of modality and separability within a single function. This set of functions pushes the algorithms to adjust their search strategies to the diverse characteristics of the landscapes, necessitating a smooth transition between modes of exploration and exploitation. Composition functions (F23-F30), however, combine multiple benchmark functions into a single composite problem, adding a greater level of complexity. These functions aim to replicate the complexities of real-world problems by incorporating a variety of optimization challenges into a single framework. The purpose of these composition functions is to assess the algorithms' ability to navigate through intricate and diverse landscapes, effectively handling various challenges and achieving optimal performance in optimization. The key features of IEEE CEC 2014 functions are outlined in Table A-4 in *Appendix A*. For further information on these functions, see [219].

The visualizations of selected benchmark functions, ranging from simple unimodal to

complex hybrid landscapes, are illustrated in Figure 4.1. These visualizations highlight the diverse optimization landscapes of the utilized benchmarks. These visualizations emphasize the importance of implementing advanced optimization strategies that can effectively navigate and leverage diverse problem spaces.



**Figure 4.1:** Representative visualizations of mathematical benchmark functions

## 4.2.2 Evaluation Metrics

*Fitness values* are utilized to assess the performance of algorithms on mathematical benchmark functions. This metric measures the accuracy of the algorithms in finding optimal or near-optimal solutions. As the functions used are minimization problems, a solution with a lower fitness value is deemed more 'fit' within the context of the problem.

## 4.2.3 Common Metrics and Visualization Techniques

In addition to domain-specific metrics, the study also covers diversity measures and visualization techniques for analyzing the performance of MHs. Diversity measures are used to quantify exploratory potential. Meanwhile, visual tools such as convergence and diversity curves are used to provide insights into how well the algorithms balance exploration and

exploitation.

- **Population diversity and balance measures**

In MHs, maintaining a balance or trade-off during the search process is crucial for achieving favorable outcomes. Search agents with the best solutions guide the process towards exploitation as they get closer, while increased distance enhances exploration. This study employs an efficient measure of population diversity, drawn from *Morrison and De Jong* [233], based on the moment of inertia concept. The definition of population diversity is demonstrated by Equation (4.1).

$$div = \sum_{j=1}^d \sum_{i=1}^N (X_{ji} - c_j)^2 \quad (4.1)$$

where the population size is denoted by  $N$ , while  $d$  indicates the problem dimension. The  $j$ th dimension of the  $i$ th solution is represented by  $X_{ji}$ . The centroid  $c_j$  for  $j = 1, 2, \dots, d$  is calculated utilizing Equation (4.2).

$$c_j = \frac{1}{N} \sum_{i=1}^N X_{ji} \quad (4.2)$$

This population diversity assessment was selected among other existing measures because it is closely connected to widely used measures of genotypic and phenotypic diversity, offering a universal diversity measuring technique. Moreover, it is a computationally efficient diversity measure [233].

Once the diversity is computed as described above, the exploration (XPL%) and exploitation (XPT%) proportions can be determined using the formula presented below [70].

$$XPL\% = (div/div_{max}) \times 100 \quad (4.3)$$

$$XPT\% = (|div - div_{max}| / div_{max}) \times 100 \quad (4.4)$$

where  $div_{max}$  represents the maximum value of population diversity that is calculated during the optimization process. The formula in Eq. (4.5) is used to quantify the trade-off ( $TO$ ) between exploration and exploitation [234]:



$$TO\% = \sqrt{XPL\% \times XPT\%} \quad (4.5)$$

- **Convergence and Diversity curves**

Visual tools, such as convergence and diversity curves, are employed to enhance analysis. Convergence curves are used to demonstrate the speed at which algorithms approach an optimal solution across iterations. Meanwhile, diversity curves illustrate the variations in solutions during the optimization process. Collectively, these visual tools provide a thorough perspective on the algorithms' performance, demonstrating their proficiency in identifying superior solutions and sustaining a wide range of potential solutions.

### 4.3 Experimental Results of CSA

In this section, a series of thorough experiments are carried out to demonstrate the effectiveness of the suggested enhancements on CSA. The findings are covered in more detail in the next subsections.

#### 4.3.1 Impact of Modifications on the Standard CSA

This subsection exhibits a thorough analysis to study how the suggested search strategies would affect the primary CSA. Therefore, a comprehensive analysis of the 3 developed variants of CSA (MRCSA, ATCSA, and ECSA) is carried out to identify the top variant in terms of solution quality, convergence trends, and diversity curves. Table 4.2 summarizes the CSA variants under investigation, where 'Y' indicates that the strategy is included, while 'N' means that the strategy is not included.

**Table 4.2:** Investigated variants of CSA

Algorithm	Adaptive tournament selection	Modified random solution generation
CSA	N	N
MRCSA	N	Y
ATCSA	Y	N
ECSA	Y	Y

The comprehensive results of the CSA and its proposed variants when applied to standard 30-dimensional test problems are presented in Table 4.3. The table enables a direct

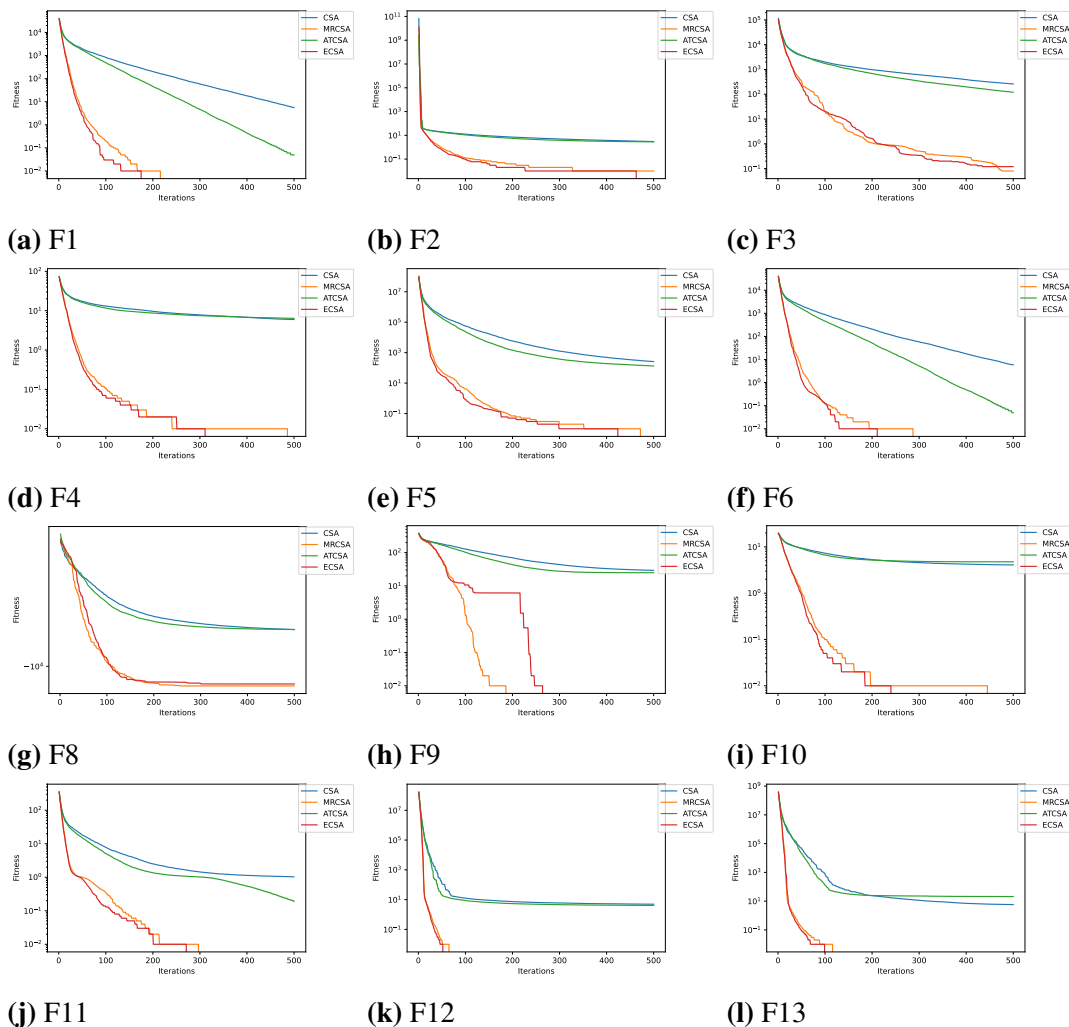
**Table 4.3:** Results of CSA variants on the standard 30-dimensional unimodal and multimodal test problems

Functions	CSA		MRCSA		ATCSA		ECSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	5.63E+00	2.69E+00	<b>1.75E-04</b>	2.79E-04	4.83E-02	3.22E-02	2.56E-04	5.25E-04
F2	3.01E+00	7.74E-01	6.75E-03	8.58E-03	2.83E+00	1.14E+00	<b>3.62E-03</b>	4.41E-03
F3	2.60E+02	8.36E+01	<b>7.68E-02</b>	1.53E-01	1.20E+02	4.28E+01	1.16E-01	4.40E-01
F4	5.96E+00	1.45E+00	4.59E-03	1.02E-02	6.39E+00	1.94E+00	<b>4.22E-03</b>	8.25E-03
F5	2.60E+02	1.17E+02	4.37E-03	1.03E-02	1.33E+02	1.04E+02	<b>2.53E-03</b>	7.14E-03
F6	5.88E+00	2.19E+00	<b>2.07E-04</b>	6.05E-04	5.11E-02	2.91E-02	2.28E-04	5.68E-04
F7	6.00E-02	2.24E-02	1.05E-03	8.90E-04	5.39E-02	2.18E-02	<b>8.13E-04</b>	6.34E-04
F8	-6.51E+03	7.05E+02	<b>-1.26E+04</b>	2.90E-01	-6.51E+03	8.93E+02	-1.23E+04	1.56E+03
F9	2.93E+01	1.04E+01	1.52E-04	2.83E-04	2.48E+01	9.10E+00	<b>7.30E-05</b>	1.98E-04
F10	4.08E+00	8.69E-01	3.49E-03	2.75E-03	4.76E+00	9.88E-01	<b>1.74E-03</b>	2.31E-03
F11	1.03E+00	4.56E-02	<b>5.17E-04</b>	1.32E-03	1.93E-01	7.35E-02	5.44E-04	1.26E-03
F12	4.92E+00	2.37E+00	3.00E-06	6.49E-06	4.07E+00	1.77E+00	<b>1.00E-06</b>	3.36E-06
F13	5.75E+00	1.09E+01	1.02E-04	2.98E-04	2.09E+01	1.75E+01	<b>1.40E-05</b>	3.62E-05
F14	1.02E+00	1.29E-01	<b>9.98E-01</b>	3.71E-16	1.11E+00	4.25E-01	<b>9.98E-01</b>	3.39E-16
F15	4.42E-04	3.54E-04	<b>3.95E-04</b>	3.26E-04	4.61E-04	3.10E-04	1.82E-03	5.16E-03
F16	<b>-1.03E+00</b>	6.78E-16	<b>-1.03E+00</b>	6.78E-16	<b>-1.03E+00</b>	6.78E-16	<b>-1.03E+00</b>	6.78E-16
F17	<b>3.98E-01</b>	1.13E-16	<b>3.98E-01</b>	1.13E-16	<b>3.98E-01</b>	1.13E-16	<b>3.98E-01</b>	1.13E-16
F18	<b>3.00E+00</b>	3.49E-15	<b>3.00E+00</b>	1.95E-15	<b>3.00E+00</b>	2.57E-15	3.90E+00	4.93E+00
F19	<b>-3.86E+00</b>	1.36E-15	<b>-3.86E+00</b>	1.36E-15	<b>-3.86E+00</b>	1.36E-15	<b>-3.86E+00</b>	1.36E-15
F20	<b>-3.30E+00</b>	4.52E-02	-3.15E+00	1.34E-01	-3.27E+00	6.07E-02	-3.08E+00	2.29E-01
F21	-8.66E+00	3.04E+00	<b>-8.98E+00</b>	2.61E+00	-8.57E+00	2.97E+00	-7.72E+00	3.40E+00
F22	-9.75E+00	2.02E+00	-9.13E+00	2.65E+00	<b>-1.04E+01</b>	9.03E-15	-9.60E+00	2.14E+00
F23	-1.00E+01	1.94E+00	-9.48E+00	2.73E+00	<b>-1.00E+01</b>	1.94E+00	-8.72E+00	3.01E+00

comparison of performance between the modified versions and the standard CSA. The results consistently demonstrate the superiority of the modified variants over the standard CSA in the majority of the test functions. Specifically, the MRCSA variant outperforms the standard CSA in 69.5% of the test functions (F1-F15 and F21), the ATCSA variant surpasses the standard CSA in 47.8% of the test functions (F1-F3, F5-F7, F9, F11-12, and F22-F23), and the ECSA variant exhibits superiority in 60.8% of the cases (F1-F14). Notably, the proposed variants excel in handling unimodal functions, showcasing their excellent exploitation capability compared to the standard CSA. Furthermore, they provide better or at least competitive results in the majority of multimodal cases, indicating their strong exploration capacity. These improved outcomes can be attributed to the integrated adaptive tournament selection mechanism and the modified random movement strategy employed by the variants. The adaptive tournament selection mechanism takes into account the global solution achieved thus far, while the modified random movement strategy incorporates a randomly scaled component. To demonstrate the statistical significance of these differences, the Wilcoxon rank sum test was applied, with the p-values presented in Table B-1 in *Appendix B*. Notably, most p-values are  $\leq 0.05$ , indicating that the enhanced variants statistically significantly outperform the standard CSA in the majority of test cases, thus confirming the superiority of the

modified versions.

To further analyze the convergence behavior of the algorithms, Figure 4.2 presents the convergence curves of CSA, MRCSA, ATCSA, and ECSA for selected test functions. These curves illustrate the evolution of the fitness value over the iterations. The term "Fitness" refers to the average of the best solution found up to that iteration over 30 independent runs. The graphical representation in Figure 4.2 allows for a visual comparison of the convergence rates and accuracy achieved by each algorithm. It becomes evident from these convergence curves that the proposed variants exhibit significantly improved convergence rates compared to the standard CSA. This further corroborates the effectiveness of the suggested techniques in accelerating the convergence process.



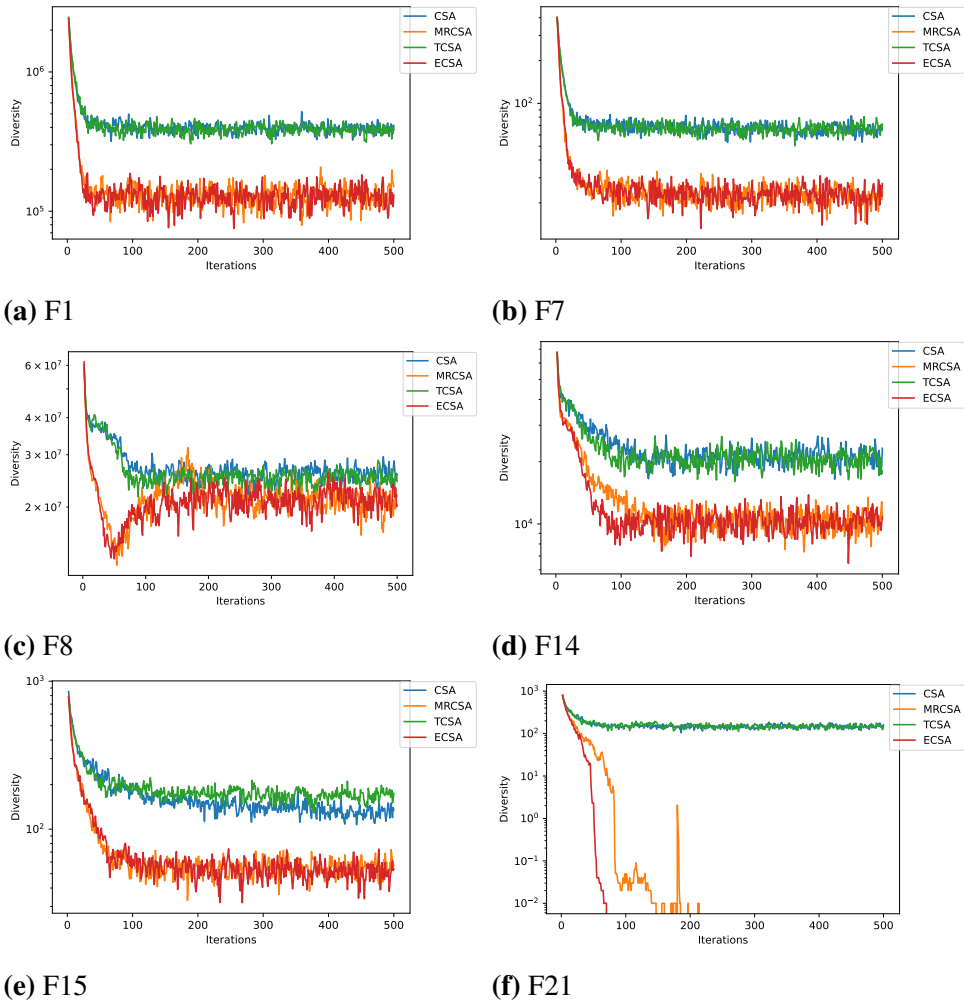
**Figure 4.2:** Comparison of convergence curves of CSA and proposed variants obtained in some of the benchmark problems.

For the diversity analysis, the figures in Figure 4.3 illustrate the diversity curves of the

standard CSA and its three enhanced variants: ATCSA, MRCSA, and ECSA across a selection of unimodal and multimodal benchmark functions (F1, F7, F8, F14, F15, and F21). It is evident that across all functions, both the basic CSA and ATCSA maintain relatively higher levels of diversity throughout the iterations compared to the other variants. This consistent maintenance of higher diversity indicates that these variants preserve exploratory behavior. Meanwhile, MRCSA and ECSA generally show a faster decline in diversity, which points to their stronger exploitation capabilities. Although the basic CSA maintains a higher level of diversity, this does not necessarily result in superior solutions compared to the enhanced variants, as indicated by the outcomes in Table 4.3 and Figure 4.2. The enhanced variants, particularly MRCSA and ECSA, demonstrate better overall performance by achieving a more balanced approach to exploration and exploitation. This balance is indicative of their ability to find better solutions across various test cases.

When comparing ATCSA to CSA, ATCSA utilizes an adaptive tournament selection approach, which not only preserves high diversity but also enhances the chances of guiding crows toward more suitable solutions. This approach facilitates a balance between exploration and intensive exploitation, which is essential for robust algorithm performance in diverse problem landscapes. Incorporating the modified random motion strategy into MRCSA and ECSA, which consider both the best solution obtained so far and the average position of all solutions, greatly enhances the exploitation capability of the algorithm. This strategy is crucial in reducing diversity at a controlled rate, ensuring that the algorithm does not miss potential regions in the search space while still focusing more effectively on the most promising regions.

The findings from the test functions suggest that extremely high diversity, as demonstrated by the basic CSA, may not always be advantageous, particularly for problems requiring more precise exploitation. The augmented versions, by employing adaptive tournament selection and modified random movement, maintain a beneficial level of diversity while enhancing exploitation capability, thereby leading to superior optimization results.



**Figure 4.3:** Comparison of diversity curves of CSA and proposed variants obtained in some of the benchmark problems.

To sum up, the standard CSA is recognized for its robust exploration capacity but lacks efficient exploitation mechanisms. It relies on random individuals and probabilities to guide the search process. Consequently, it often struggles to efficiently exploit the search space, despite its proficiency in exploration. The modifications proposed in the variants aim to strike a balance between exploration and exploitation, thereby reducing the risk of premature convergence to local optima. The findings of the experiments prove that these enhancements significantly improve the algorithm's performance, validating the proposed approach and confirming its effectiveness in balancing exploration and exploitation.

### 4.3.2 Analysis of Island-Based CSA Variants

In this section, the basic structure of the island model is integrated into each CSA variant to investigate its ability to probe the search domain. Accordingly, four different island-based variations (*iCSA*, *iMRCSA*, *iATCSA*, and *iECSA*) have been assessed using the 30-dimensional standard benchmark functions. Table 4.4 compares the outcomes of the island-based versions against the corresponding basic versions in terms of fitness values. Successively, in the majority of cases, the island-based versions declare superior results. In more detail, *iCSA* performs better than *CSA* in 82.6% of cases, and it achieves comparable outcomes in four cases (F16-F19). In 78.2% percent of problems, *iMRCSA* achieves lower fitness values than *MRCSA*, and it achieves similar results in five problems (F14, F16-F19). Besides, *iATCSA* and *iECSA* have yielded better outcomes than *ATCSA* and *ECSA* on 78.2% and 82.6% of problems, respectively. Generally, it is clear from table 4.4 that *iECSA* outperforms all other methods. The *iECSA* has perceived the lowest fitness values across 13 problems out of 23 problems and it produces comparable outcomes (i.e., a tie) for five cases.

Examining the boxplot findings in Figure 4.4, it can be observed that the suggested *iECSA* frequently offered the smallest median value. The outcomes demonstrate the superiority of *iECSA* in the majority of problems. The convergence trends of *CSA*, *ECSA*, and *iECSA* algorithms are displayed in Figure 4.5. It is clear from looking at this figure that the *iECSA* has substantially faster convergence rates than the traditional *CSA* and *ECSA* methods. According to the overall ranking rates (F-test) shown in Figure 4.6, we observe that the *iECSA* achieves the best rank of 2.26, followed by *iMRCSA*, *MRCSA*, *iATCSA*, *ECSA*, *iCSA*, *ATCSA*, and *CSA*, respectively.

Taken together, the testing and comparison findings confirmed the benefits of incorporating the structure of the island model. Frequent information exchange through migration mechanisms is essential to empowering the capability to navigate the search space more effectively. The method of segmenting the population into a number of sub-populations enables island-based MHs to retain several optimal solutions while simultaneously emphasizing different areas of the search space. The island model is particularly effective at enhancing global search, and it may be used to investigate promising areas of the search domain by uti-

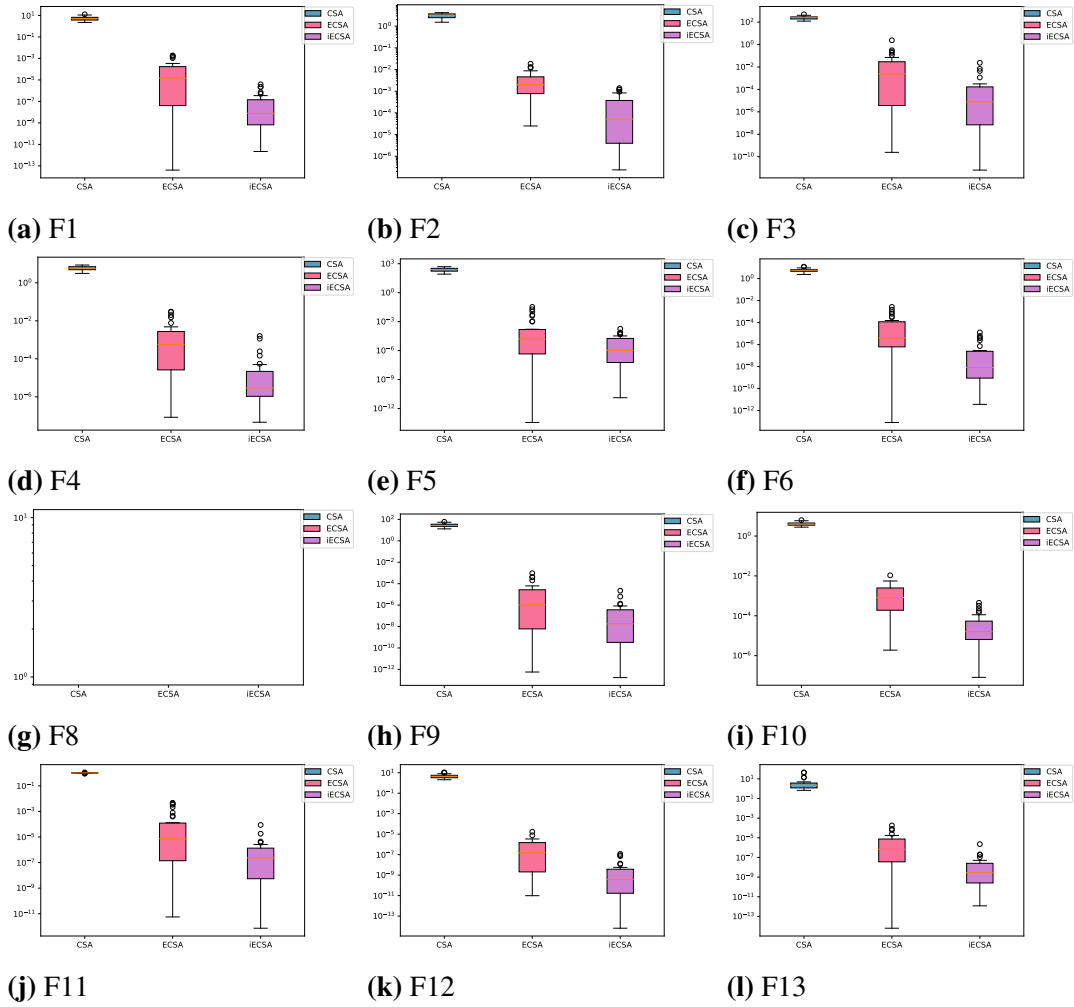
**Table 4.4:** Comparison of CSA variants and cooperative island-based variants for the standard 30-dimensional functions

Function	Measure	CSA	iCSA	MRCSA	iMRCSA	ATCSA	iATCSA	ECSA	iECSA
F1	AVG	5.63E+00	9.78E-01	1.75E-04	3.44E-05	4.83E-02	8.72E-03	2.56E-04	<b>3.52E-07</b>
	STD	2.69E+00	4.80E-01	2.79E-04	7.25E-05	3.22E-02	5.26E-03	5.25E-04	8.72E-07
F2	AVG	3.01E+00	1.31E+00	6.75E-03	2.13E-03	2.83E+00	7.25E-01	3.62E-03	<b>2.77E-04</b>
	STD	7.74E-01	3.99E-01	8.58E-03	2.10E-03	1.14E+00	2.26E-01	4.41E-03	4.27E-04
F3	AVG	2.60E+02	8.29E+01	7.68E-02	2.05E-02	1.20E+02	3.39E+01	1.16E-01	<b>1.25E-03</b>
	STD	8.36E+01	2.81E+01	1.53E-01	6.58E-02	4.28E+01	1.34E+01	4.40E-01	4.49E-03
F4	AVG	5.96E+00	2.73E+00	4.59E-03	1.38E-03	6.39E+00	2.90E+00	4.22E-03	<b>1.12E-04</b>
	STD	1.45E+00	5.23E-01	1.02E-02	2.14E-03	1.94E+00	6.98E-01	8.25E-03	3.51E-04
F5	AVG	2.60E+02	9.10E+01	4.37E-03	2.61E-04	1.33E+02	3.90E+01	2.53E-03	<b>1.78E-05</b>
	STD	1.17E+02	4.48E+01	1.03E-02	4.24E-04	1.04E+02	1.89E+01	7.14E-03	3.79E-05
F6	AVG	5.88E+00	1.15E+00	2.07E-04	4.60E-05	5.11E-02	8.34E-03	2.28E-04	<b>1.03E-06</b>
	STD	2.19E+00	5.57E-01	6.05E-04	1.01E-04	2.91E-02	3.95E-03	5.68E-04	2.65E-06
F7	AVG	6.00E-02	8.37E-03	1.05E-03	3.86E-04	5.39E-02	7.18E-03	8.13E-04	<b>3.48E-04</b>
	STD	2.24E-02	3.30E-03	8.90E-04	2.23E-04	2.18E-02	2.28E-03	6.34E-04	1.83E-04
F8	AVG	-6.51E+03	-7.31E+03	-1.26E+04	-1.26E+04	-6.51E+03	-7.41E+03	-1.23E+04	<b>-1.26E+04</b>
	STD	7.05E+02	4.55E+02	2.90E-01	4.83E-03	8.93E+02	4.54E+02	1.56E+03	1.22E-03
F9	AVG	2.93E+01	2.12E+01	1.52E-04	3.66E-05	2.48E+01	2.05E+01	7.30E-05	<b>1.13E-06</b>
	STD	1.04E+01	6.53E+00	2.83E-04	9.60E-05	9.10E+00	5.52E+00	1.98E-04	4.10E-06
F10	AVG	4.08E+00	2.26E+00	3.49E-03	9.73E-04	4.76E+00	2.91E+00	1.74E-03	<b>6.30E-05</b>
	STD	8.69E-01	5.26E-01	2.75E-03	7.72E-04	9.88E-01	6.56E-01	2.31E-03	1.06E-04
F11	AVG	1.03E+00	7.97E-01	5.17E-04	7.84E-05	1.93E-01	8.66E-02	5.44E-04	<b>4.22E-06</b>
	STD	4.56E-02	1.35E-01	1.32E-03	1.62E-04	7.35E-02	2.95E-02	1.26E-03	1.61E-05
F12	AVG	4.92E+00	1.82E+00	3.00E-06	3.94E-07	4.07E+00	3.42E+00	1.00E-06	<b>1.11E-08</b>
	STD	2.37E+00	1.21E+00	6.49E-06	5.36E-07	1.77E+00	1.16E+00	3.36E-06	2.87E-08
F13	AVG	5.75E+00	4.76E-01	1.02E-04	4.10E-06	2.09E+01	1.12E+00	1.40E-05	<b>1.05E-07</b>
	STD	1.09E+01	2.86E-01	2.98E-04	7.86E-06	1.75E+01	8.39E-01	3.62E-05	4.13E-07
F14	AVG	1.02E+00	<b>9.98E-01</b>	<b>9.98E-01</b>	<b>9.98E-01</b>	1.11E+00	<b>9.98E-01</b>	<b>9.98E-01</b>	<b>9.98E-01</b>
	STD	1.29E-01	3.55E-16	3.71E-16	3.39E-16	4.25E-01	3.39E-16	3.39E-16	3.39E-16
F15	AVG	4.42E-04	<b>3.07E-04</b>	3.95E-04	<b>3.07E-04</b>	4.61E-04	<b>3.07E-04</b>	1.82E-03	3.10E-04
	STD	3.54E-04	3.16E-13	3.26E-04	7.46E-15	3.10E-04	7.84E-10	5.16E-03	7.55E-06
F16	AVG	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>	<b>-1.03E+00</b>
	STD	6.78E-16	1.28E-14	6.78E-16	6.78E-16	6.78E-16	6.78E-16	6.78E-16	6.78E-16
F17	AVG	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>	<b>3.98E-01</b>
	STD	1.13E-16	6.22E-15	1.13E-16	1.13E-16	1.13E-16	1.13E-16	1.13E-16	1.13E-16
F18	AVG	<b>3.00E+00</b>	<b>3.00E+00</b>	<b>3.00E+00</b>	<b>3.00E+00</b>	<b>3.00E+00</b>	<b>3.00E+00</b>	3.90E+00	<b>3.00E+00</b>
	STD	3.49E-15	8.27E-15	1.95E-15	7.54E-15	2.57E-15	4.52E-16	4.93E+00	4.52E-16
F19	AVG	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>	<b>-3.86E+00</b>
	STD	1.36E-15	1.36E-15	1.36E-15	1.36E-15	1.36E-15	1.36E-15	1.36E-15	1.36E-15
F20	AVG	-3.30E+00	<b>-3.32E+00</b>	-3.15E+00	-3.31E+00	-3.27E+00	<b>-3.32E+00</b>	-3.08E+00	-3.29E+00
	STD	4.52E-02	2.88E-11	1.34E-01	8.42E-03	6.07E-02	4.40E-15	2.29E-01	4.90E-02
F21	AVG	-8.66E+00	<b>-1.02E+01</b>	-8.98E+00	<b>-1.02E+01</b>	-8.57E+00	<b>-1.02E+01</b>	-7.72E+00	-1.02E+01
	STD	3.04E+00	7.63E-12	2.61E+00	1.64E-13	2.97E+00	1.81E-15	3.40E+00	6.27E-03
F22	AVG	-9.75E+00	<b>-1.04E+01</b>	-9.13E+00	<b>-1.04E+01</b>	<b>-1.04E+01</b>	<b>-1.04E+01</b>	-9.60E+00	-1.04E+01
	STD	2.02E+00	1.74E-12	2.65E+00	6.69E-10	9.03E-15	9.03E-15	2.14E+00	2.98E-02
F23	AVG	-1.00E+01	<b>-1.05E+01</b>	-9.48E+00	<b>-1.05E+01</b>	-1.00E+01	<b>-1.05E+01</b>	-8.72E+00	-1.05E+01
	STD	1.94E+00	2.49E-12	2.73E+00	9.31E-13	1.94E+00	9.03E-15	3.01E+00	2.16E-02
Mean rank		6.63	4.76	4.30	2.61	6.33	4.41	4.70	<b>2.26</b>

lizing the advantages of the best individuals of the relevant sub-population.

### 4.3.3 Scalability Analysis

In this part, a scalability analysis is carried out to explore the influence of problem dimension on the outcomes of CSA variants. This test has been used in earlier studies and can show how dimensions affect the quality of solutions for the different optimizers to understand their efficacy for problems with lower and higher dimensions. In addition, it demonstrates how population-based MHs can maintain its search benefits in greater dimensions [39]. For this purpose, the scalable unimodal and multimodal test cases (F1-F13) with 30, 100, and 500 dimensions are handled using the different CSA variants. These functions are considered in

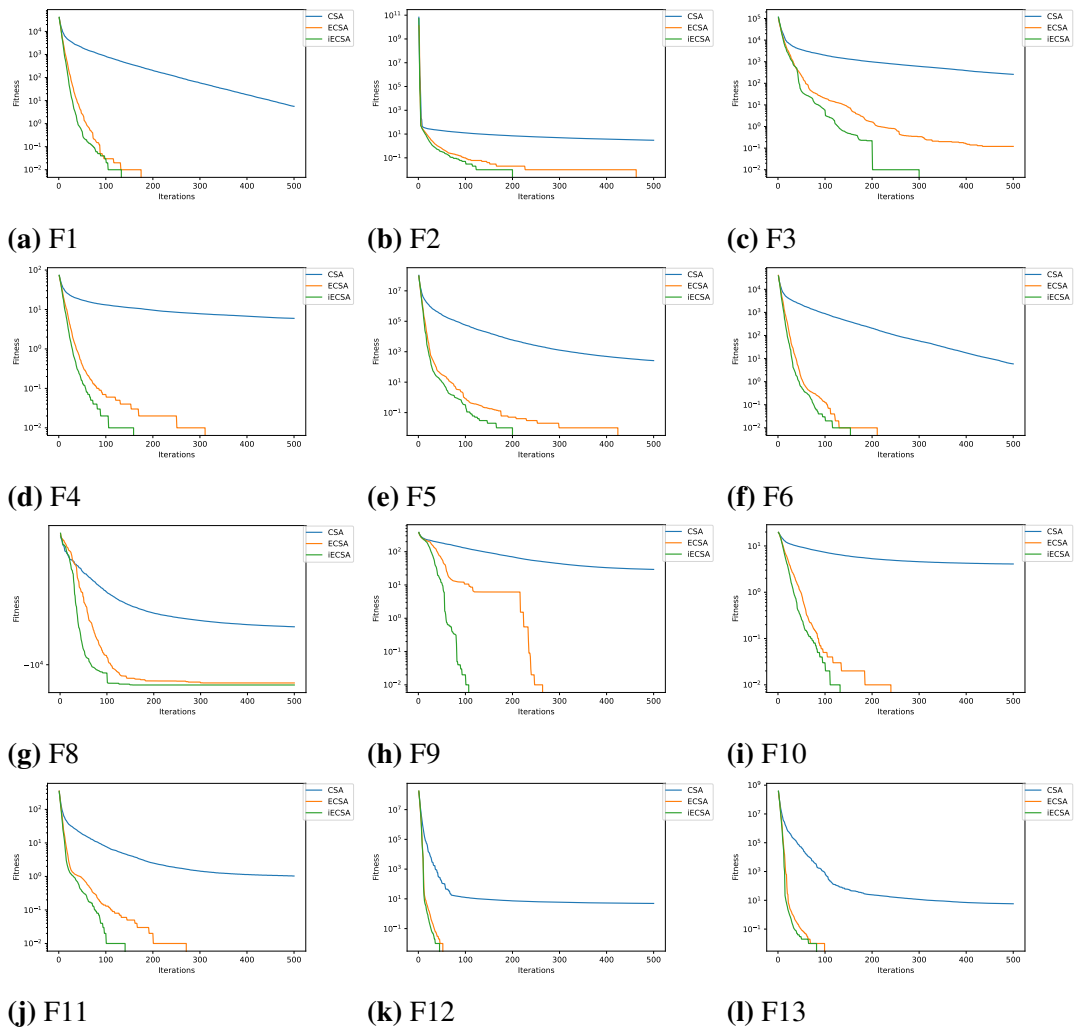


**Figure 4.4:** Box-plots of CSA, ECSA, IECSA on some of the standard benchmarks

this experiment because their mathematical formulations provide us the ability to adjust the number of design variables.

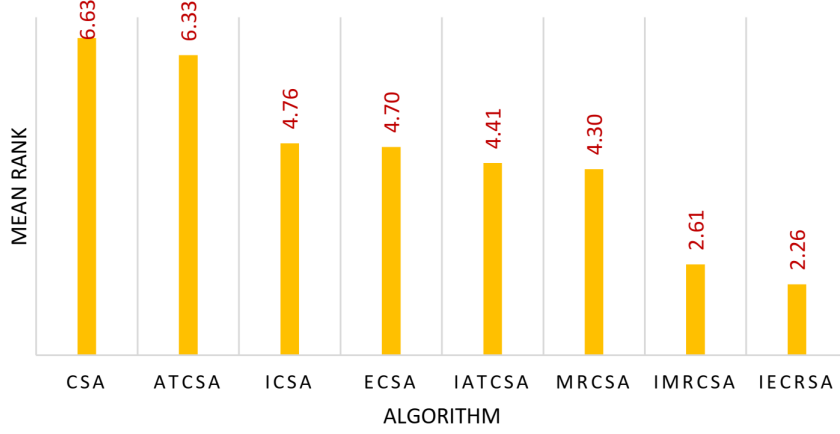
The results of the *iECSA* in comparison to other CSA variants while handling the standard scalable functions are outlined in Tables 4.5 and 4.6. It can be seen that the *iECSA* is capable of revealing outstanding findings across all dimensions, and its performance continues to be reliably superior even when dealing with problems that involve a large number of variables. The higher performance of this optimizer is demonstrated by the fact that the *iECSA* has the potential to significantly outperform other methods and provide the best results for 56.5% of 30-dimensional functions (as shown in Table 4.3). According to Tables 4.5 and 4.6, when we have a search space with 100, and 500 dimensions, the results obtained by *iECSA* are noticeably superior to those obtained by other methods when dealing with 100% of F1-F13 functions. As shown by the curves in Figure 4.7, it can be seen that as the dimen-





**Figure 4.5:** Comparison of Convergence Curves for CSA, ECSA, and iECSA on some of the standard benchmark problems.

sions increase, the outcomes of some other approaches are degraded. This demonstrates that *iECSA* can successfully maintain a balance between exploration and exploitation tendencies in problems with higher dimensions.



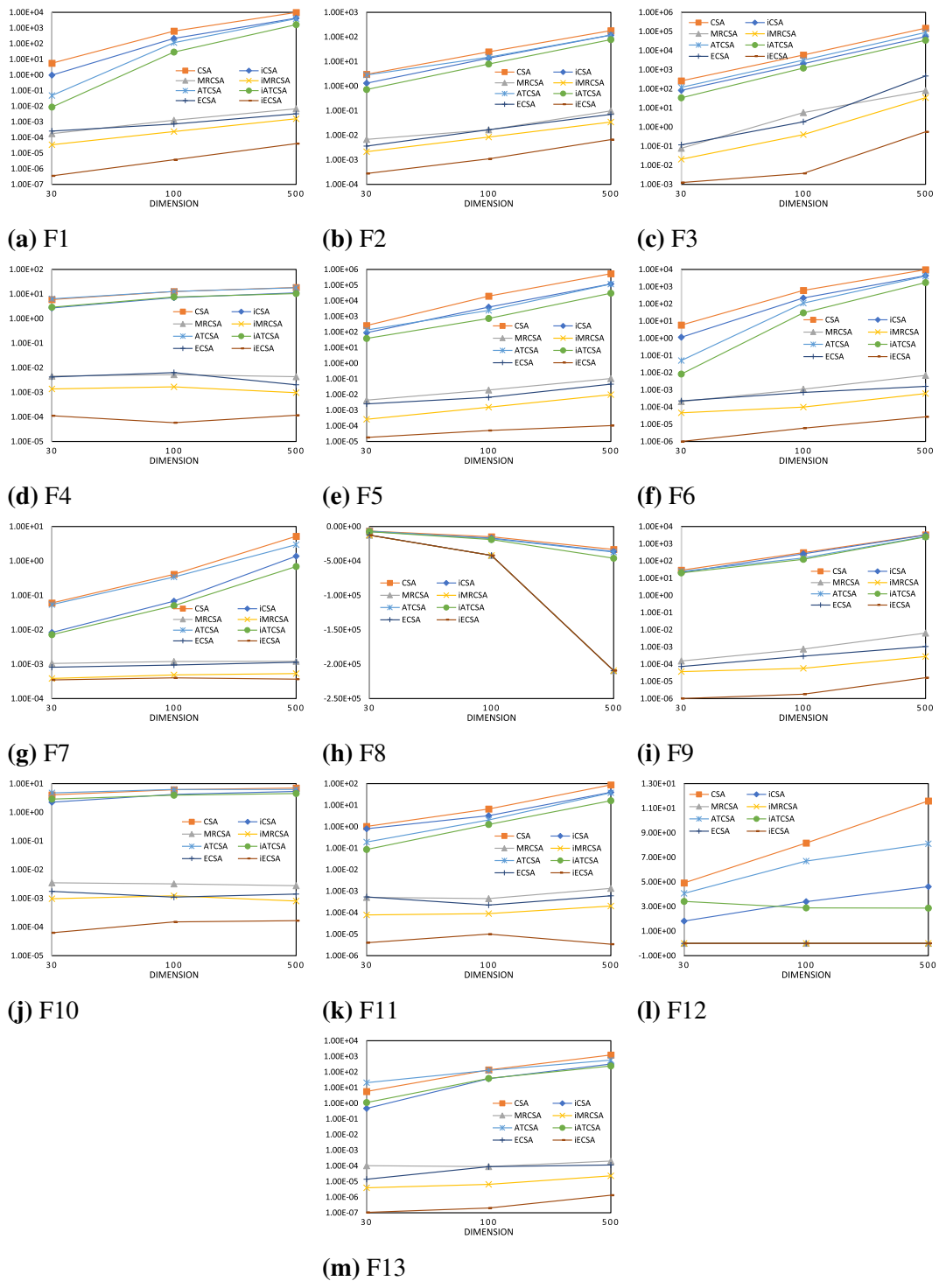
**Figure 4.6:** Friedman mean rank based on results in Table 4.4

**Table 4.5:** Results of benchmark functions F1-F13 with 100 dimensions

Function	Measure	CSA	iCSA	MRCSA	iMRCSA	ATCSA	iATCSA	ECSA	IECSA
F1	AVG	6.38E+02	2.17E+02	1.27E-03	2.36E-04	1.15E+02	2.90E+01	7.22E-04	<b>3.73E-06</b>
	STD	1.07E+02	3.77E+01	4.00E-03	4.64E-04	2.13E+01	6.31E+00	1.62E-03	8.20E-06
F2	AVG	2.49E+01	1.38E+01	1.65E-02	8.40E-03	1.49E+01	7.87E+00	1.67E-02	<b>1.08E-03</b>
	STD	2.24E+00	1.24E+00	1.83E-02	9.48E-03	1.87E+00	1.03E+00	1.86E-02	1.60E-03
F3	AVG	5.84E+03	2.06E+03	5.75E+00	4.04E-01	3.17E+03	1.23E+03	1.86E+00	<b>3.73E-03</b>
	STD	1.15E+03	3.05E+02	1.47E+01	5.12E-01	4.49E+02	2.04E+02	2.86E+00	5.27E-03
F4	AVG	1.28E+01	7.21E+00	5.24E-03	1.67E-03	1.26E+01	7.59E+00	6.33E-03	<b>5.79E-05</b>
	STD	1.55E+00	5.27E-01	1.16E-02	2.36E-03	1.56E+00	8.47E-01	1.01E-02	1.30E-04
F5	AVG	1.99E+04	3.93E+03	1.95E-02	1.56E-03	2.41E+03	7.43E+02	6.62E-03	<b>5.05E-05</b>
	STD	4.75E+03	8.93E+02	3.04E-02	2.84E-03	5.04E+02	1.51E+02	1.29E-02	1.36E-04
F6	AVG	6.09E+02	2.22E+02	1.11E-03	9.99E-05	1.12E+02	2.99E+01	7.12E-04	<b>5.93E-06</b>
	STD	9.32E+01	4.85E+01	3.85E-03	1.51E-04	2.20E+01	8.72E+00	1.18E-03	1.74E-05
F7	AVG	4.10E-01	6.77E-02	1.19E-03	4.87E-04	3.41E-01	5.02E-02	9.42E-04	<b>4.00E-04</b>
	STD	9.41E-02	1.57E-02	8.34E-04	2.63E-04	6.27E-02	1.16E-02	5.98E-04	2.74E-04
F8	AVG	-1.48E+04	-1.68E+04	-4.19E+04	-4.19E+04	-1.77E+04	-1.89E+04	-4.19E+04	<b>-4.19E+04</b>
	STD	1.52E+03	7.71E+02	3.29E-01	2.90E-02	2.14E+03	1.44E+03	2.48E-01	1.65E-03
F9	AVG	3.07E+02	2.61E+02	7.61E-04	5.74E-05	1.50E+02	1.24E+02	2.92E-04	<b>1.79E-06</b>
	STD	4.13E+01	4.40E+01	1.33E-03	1.06E-04	2.82E+01	2.40E+01	8.30E-04	3.94E-06
F10	AVG	6.12E+00	4.23E+00	3.21E-03	1.23E-03	6.28E+00	3.95E+00	1.10E-03	<b>1.50E-04</b>
	STD	6.15E-01	3.51E-01	3.40E-03	1.11E-03	7.40E-01	2.83E-01	1.71E-03	3.89E-04
F11	AVG	6.68E+00	3.22E+00	4.48E-04	9.02E-05	2.04E+00	1.27E+00	2.25E-04	<b>1.01E-05</b>
	STD	9.12E-01	3.55E-01	1.04E-03	1.44E-04	1.94E-01	5.19E-02	4.15E-04	3.42E-05
F12	AVG	8.17E+00	3.40E+00	1.71E-06	1.65E-07	6.70E+00	2.89E+00	6.82E-07	<b>1.21E-08</b>
	STD	1.30E+00	8.61E-01	3.05E-06	3.40E-07	2.21E+00	1.20E+00	1.92E-06	4.22E-08
F13	AVG	1.39E+02	3.81E+01	8.96E-05	6.51E-06	1.29E+02	3.94E+01	8.84E-05	<b>2.00E-07</b>
	STD	1.89E+01	6.82E+00	2.58E-04	1.53E-05	1.54E+01	5.12E+00	2.73E-04	5.89E-07

#### 4.3.4 Sensitivity Analysis of iECSA to its Migration Parameters

This part of the experiment aims to examine the impact of various parameters on the performance of the proposed *iECSA* algorithm. In this regard, different experimental scenarios were designed based on a simple design technique where we start with a common configuration and alter one parameter at a time to evaluate how that parameter impacts the performance [196]. These scenarios include three parameters: the population size ( $N$ ), the migration frequency ( $M_f$ ), and the migration rate ( $M_r$ ). These parameters are selected because their impact



**Figure 4.7:** Scalability results of the CSA variants in dealing with F1–F13 functions with different dimensions

**Table 4.6:** Results of benchmark functions F1-F13 with 500 dimensions

Function	Measure	CSA	iCSA	MRCSA	iMRCSA	ATCSA	iATCSA	ECSA	iECSA
F1	AVG	9.91E+03	4.22E+03	6.86E-03	1.57E-03	3.98E+03	1.64E+03	3.28E-03	<b>4.02E-05</b>
	STD	6.80E+02	3.58E+02	2.51E-02	4.85E-03	2.47E+02	1.44E+02	8.71E-03	1.10E-04
F2	AVG	1.81E+02	1.19E+02	9.80E-02	3.44E-02	1.20E+02	7.81E+01	7.07E-02	<b>6.62E-03</b>
	STD	7.26E+00	5.50E+00	1.87E-01	3.74E-02	5.14E+00	3.88E+00	9.49E-02	9.76E-03
F3	AVG	1.48E+05	5.42E+04	8.01E+01	3.45E+01	8.84E+04	3.46E+04	4.68E+02	<b>5.58E-01</b>
	STD	2.34E+04	7.77E+03	1.46E+02	7.00E+01	9.09E+03	4.48E+03	9.45E+02	1.08E+00
F4	AVG	1.85E+01	1.11E+01	4.31E-03	9.63E-04	1.82E+01	1.05E+01	2.04E-03	<b>1.17E-04</b>
	STD	1.30E+00	6.88E-01	9.55E-03	9.82E-04	1.56E+00	4.66E-01	4.70E-03	2.83E-04
F5	AVG	5.39E+05	1.19E+05	1.05E-01	9.94E-03	1.16E+05	3.05E+04	4.60E-02	<b>1.04E-04</b>
	STD	6.82E+04	1.68E+04	2.14E-01	1.98E-02	1.27E+04	4.45E+03	1.20E-01	2.06E-04
F6	AVG	9.77E+03	4.34E+03	6.91E-03	6.15E-04	4.10E+03	1.74E+03	1.59E-03	<b>2.69E-05</b>
	STD	6.25E+02	5.10E+02	1.37E-02	1.50E-03	2.55E+02	1.63E+02	2.96E-03	7.80E-05
F7	AVG	5.20E+00	1.38E+00	1.21E-03	5.33E-04	2.96E+00	6.91E-01	1.14E-03	<b>3.66E-04</b>
	STD	6.70E-01	2.59E-01	7.65E-04	2.81E-04	2.85E-01	1.31E-01	8.28E-04	2.06E-04
F8	AVG	-3.32E+04	-3.63E+04	-2.09E+05	-2.09E+05	-3.70E+04	-4.60E+04	-2.09E+05	<b>-2.09E+05</b>
	STD	4.92E+03	2.47E+03	1.73E+01	1.69E-01	7.51E+03	4.82E+03	2.19E+00	4.61E-02
F9	AVG	3.34E+03	3.21E+03	6.45E-03	2.79E-04	2.66E+03	2.51E+03	1.07E-03	<b>1.62E-05</b>
	STD	1.16E+02	8.03E+01	1.54E-02	4.08E-04	9.53E+01	8.39E+01	3.10E-03	3.47E-05
F10	AVG	7.04E+00	5.38E+00	2.75E-03	8.08E-04	6.29E+00	4.55E+00	1.41E-03	<b>1.66E-04</b>
	STD	2.03E-01	1.45E-01	3.25E-03	6.20E-04	2.71E-01	1.50E-01	2.44E-03	3.90E-04
F11	AVG	8.79E+01	3.98E+01	1.36E-03	2.03E-04	3.75E+01	1.62E+01	6.06E-04	<b>3.40E-06</b>
	STD	5.47E+00	3.52E+00	2.74E-03	3.65E-04	2.58E+00	1.47E+00	1.31E-03	8.03E-06
F12	AVG	1.16E+01	4.61E+00	2.86E-06	5.50E-08	8.11E+00	2.88E+00	4.78E-07	<b>6.65E-09</b>
	STD	1.34E+00	4.70E-01	1.07E-05	1.17E-07	1.20E+00	4.22E-01	9.49E-07	1.57E-08
F13	AVG	1.23E+03	3.24E+02	2.04E-04	2.29E-05	5.80E+02	2.44E+02	1.14E-04	<b>1.34E-06</b>
	STD	4.78E+02	3.64E+01	4.62E-04	5.70E-05	3.13E+01	2.15E+01	4.38E-04	5.52E-06

on several island models is statistically significant, as proved in previous studies [53]. The simple factorial design consisting of nine experiments is shown in Table 4.7. The first three experiments were employed to assess how  $N$  affected the functionality of iECSA. To assess how  $M_r$  affected the functionality of iECSA, the second three experimental situations were employed. The impact of  $M_r$  on the performance of iECSA was finally assessed using the final three experimental situations. Similar experimental designs were used in previous studies to assess the efficacy of iHS [193], iBAT [54], and iWOA [55]. For each configuration, the results are summarized over 30 trials and 500 iterations. It is important to note that the optimal settings in this stage are utilized for the following experiments.

**Table 4.7:** The experimental design for evaluating the sensitivity of *iECSA*  $N$ ,  $M_F$ , and  $M_r$  parameters

Experiment Number	N	Mf	Mr
1	30		
2	50	100	0.3
3	<b>70</b>		
4		25	
5	70	50	0.3
6		<b>100</b>	
7			<b>0.1</b>
8	70	100	0.2
9			0.3

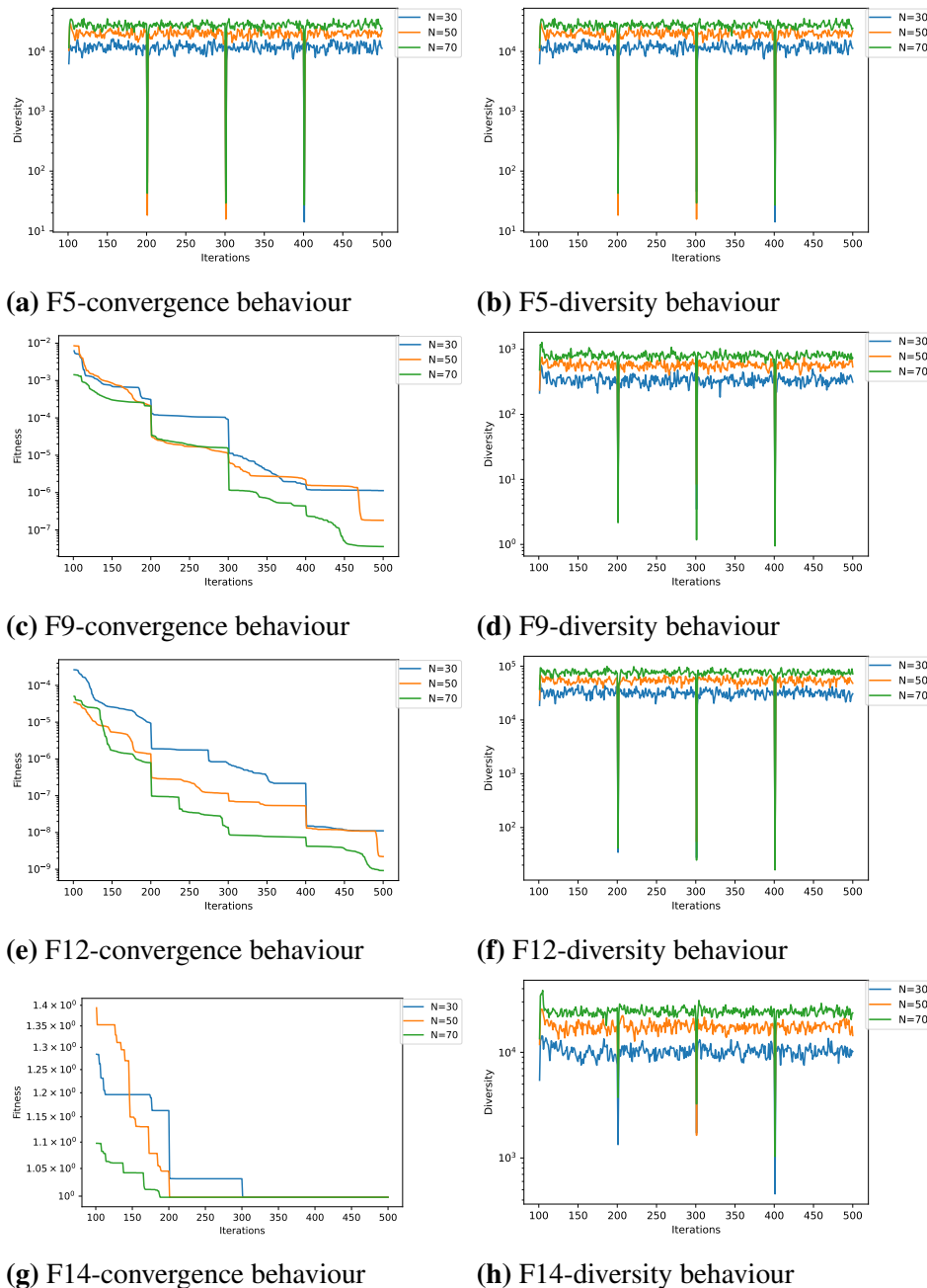
1. **Impact of population size** In this experiment, three alternative values  $N=30$ ,  $N=50$ , and  $N=70$  are used to examine the impact of the number of crows ( $N$ ) on the performance of the suggested *iECSA* algorithm. The other *iECSA* parameters are fixed in this experiment, thus, three scenarios have been developed, as shown in Table 4.7. It is important to notice that the results acquired by Sen1 are identical to those obtained in the earlier experiments. Table 4.8 presents the experimental results of the *iECSA* when dealing with the standard 30-dimensional test functions in terms of average fitness values. Evidently, Sen3 can provide the best solutions for 9 of the 23 test functions, and it scores comparable outcomes in 9 cases. As per mean rank, Sen3 provides the optimal rank of 1.61 followed by Sen2 (rank of 1.87) and Sen1 (rank of 2.52). This is because wider coverage of the search space regions results in higher  $N$  values. So, there are more options to find effective solutions. For the following experiments, the value of  $N$  for each incorporated *ECSA* algorithm in the *iECSA* model is set to 70 because this strategy produced the best results.

**Table 4.8:** The impact of population size on the convergence behavior of *iECSA*

Function	Sen1 N=30	Sen2 N=50	Sen3 N=70
F1	3.52E-07	4.54E-07	<b>1.78E-07</b>
F2	2.77E-04	1.15E-04	<b>1.05E-04</b>
F3	1.25E-03	1.45E-04	<b>1.12E-04</b>
F4	1.12E-04	1.09E-04	<b>9.49E-05</b>
F5	1.78E-05	6.48E-06	<b>2.40E-06</b>
F6	1.03E-06	<b>1.45E-07</b>	2.93E-07
F7	3.48E-04	<b>2.21E-04</b>	2.26E-04
F8	<b>-1.26E+04</b>	-1.26E+04	-1.26E+04
F9	1.13E-06	1.81E-07	<b>3.59E-08</b>
F10	6.26E-05	<b>4.91E-05</b>	7.09E-05
F11	4.22E-06	6.83E-07	<b>5.23E-07</b>
F12	1.11E-08	2.21E-09	<b>9.22E-10</b>
F13	1.05E-07	7.16E-08	<b>3.37E-08</b>
F14	<b>9.98E-01</b>	<b>9.98E-01</b>	<b>9.98E-01</b>
F15	3.10E-04	<b>3.07E-04</b>	<b>3.07E-04</b>
F16	-1.03E+00	<b>-1.03E+00</b>	<b>-1.03E+00</b>
F17	3.98E-01	<b>3.98E-01</b>	<b>3.98E-01</b>
F18	<b>3.00E+00</b>	<b>3.00E+00</b>	<b>3.00E+00</b>
F19	-3.86E+00	<b>-3.86E+00</b>	<b>-3.86E+00</b>
F20	-3.29E+00	-3.32E+00	<b>-3.32E+00</b>
F21	<b>-1.02E+01</b>	-1.02E+01	-1.02E+01
F22	-1.04E+01	<b>-1.04E+01</b>	<b>-1.04E+01</b>
F23	-1.05E+01	<b>-1.05E+01</b>	<b>-1.05E+01</b>
Mean Rank	2.52	1.87	<b>1.61</b>

The behavior of the proposed technique in terms of convergence and diversity after 500

iterations with different  $N$  values is depicted in Figure 4.8. In terms of convergence behavior, it can be demonstrated that the convergence of the iECSA members becomes better as  $N$  increases. In terms of diversity behavior, it is consistent with convergence behavior in that a bigger  $N$  value results in better diversity behavior. In other words, diversity is maintained throughout the search considerably more effectively when the value of  $N$  is high.



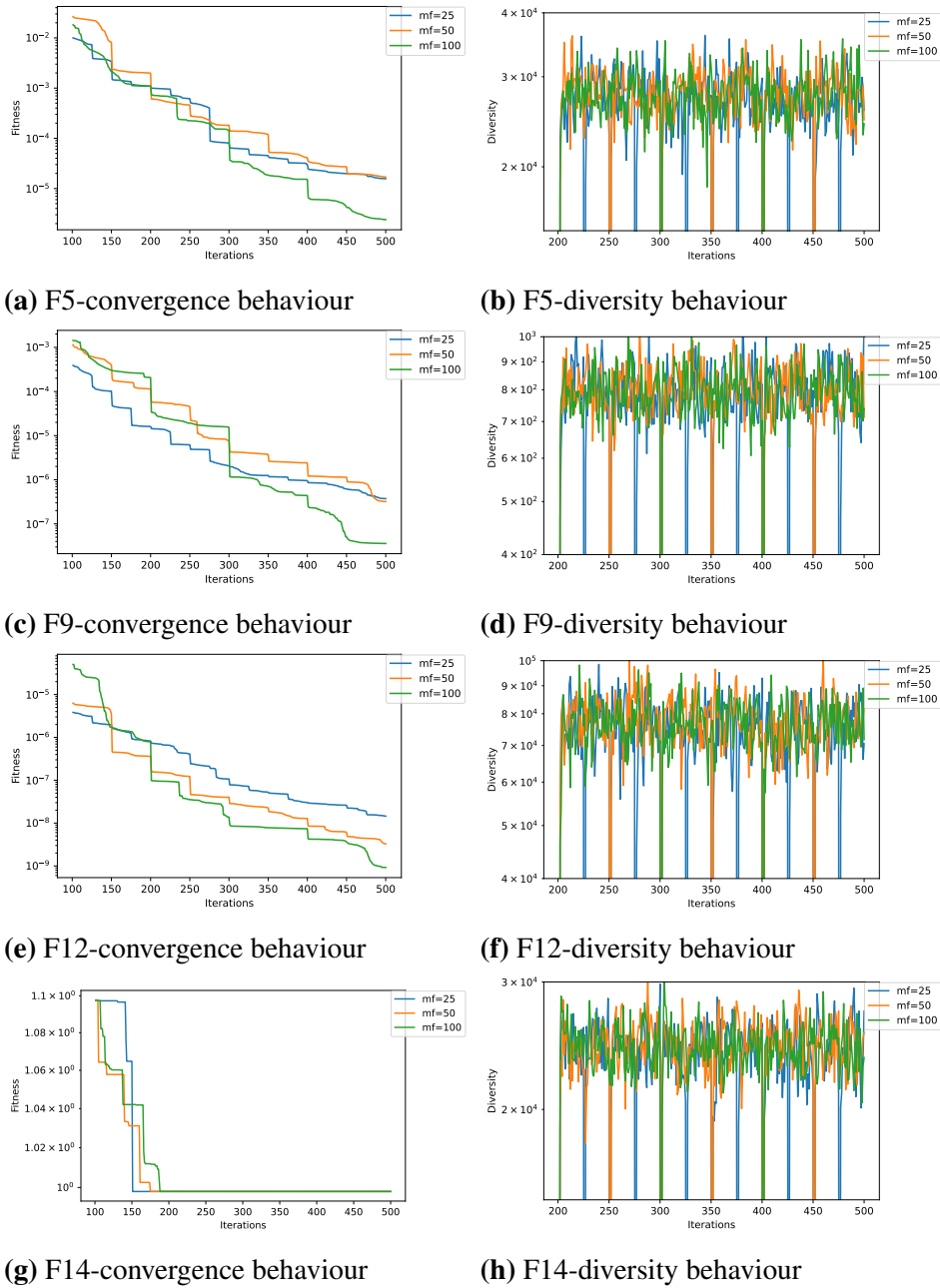
**Figure 4.8:** The convergence and diversity plots of iECSA using different values of population size

## 2 Impact of migration frequency: In this part, the influence of the migration frequency

( $M_f$ ) factor on the behavior of the proposed *iECSA* is investigated using three different experimental scenarios. These scenarios are constructed using different values of  $M_f$ , such as Sen4 ( $M_f = 25$ ), Sen5 ( $M_f = 50$ ), and Sen6 ( $M_f = 100$ ). When the value of  $M_f$ , for instance, is set to 50, this indicates that the migration process is activated after every 50 iterations (i.e., 10 times during the allowed 500 iterations). Please note that the remaining parameters of the suggested scenarios also stay the same as they are in Table 4.7. The experimental findings obtained utilizing the three aforementioned scenarios are reported in Table 4.9. In general, it is observed that the effectiveness of the *iECSA* is significantly influenced by the value of the  $M_f$ . In specific, Sen6 ( $M_f = 100$ ) clearly provides the best solutions for 15 of the 23 test functions (the optimal ranking of 1.52). In contrast, the *iECSA* method performs worse when the parameter  $M_f$  has a lower value. It is clear that frequent migration at close intervals reduces the effectiveness of the algorithm. As a result, the recommended value of the parameter  $M_f$ , which allows the *iECSA* algorithm to navigate the search space while balancing exploration and exploitation, is 100. Therefore, in the following experimental work, this value will be set.

**Table 4.9:** The impact of migration frequency on the convergence of *iECSA*

Function	Sen4 Mf=25	Sen5 Mf=50	Sen6 Mf=100
F1	8.778E-07	7.055E-07	<b>1.780E-07</b>
F2	3.727E-04	2.806E-04	<b>1.050E-04</b>
F3	2.071E-04	2.401E-04	<b>1.121E-04</b>
F4	2.050E-04	1.655E-04	<b>9.490E-05</b>
F5	1.567E-05	1.680E-05	<b>2.400E-06</b>
F6	4.410E-07	4.650E-07	<b>2.930E-07</b>
F7	2.413E-04	<b>2.247E-04</b>	2.256E-04
F8	<b>-1.257E+04</b>	-1.257E+04	-1.257E+04
F9	3.732E-07	3.205E-07	<b>3.590E-08</b>
F10	1.595E-04	9.603E-05	<b>7.090E-05</b>
F11	8.166E-07	9.723E-07	<b>5.230E-07</b>
F12	1.452E-08	3.262E-09	<b>9.220E-10</b>
F13	8.834E-08	6.529E-08	<b>3.370E-08</b>
F14	9.980E-01	9.980E-01	<b>9.980E-01</b>
F15	<b>3.075E-04</b>	<b>3.075E-04</b>	3.075E-04
F16	<b>-1.032E+00</b>	<b>-1.032E+00</b>	-1.032E+00
F17	<b>3.979E-01</b>	<b>3.979E-01</b>	3.979E-01
F18	<b>3.000E+00</b>	<b>3.000E+00</b>	3.000E+00
F19	-3.863E+00	-3.863E+00	<b>-3.863E+00</b>
F20	-3.322E+00	-3.322E+00	<b>-3.322E+00</b>
F21	<b>-1.015E+01</b>	<b>-1.015E+01</b>	-1.015E+01
F22	-1.040E+01	-1.040E+01	<b>-1.040E+01</b>
F23	<b>-1.054E+01</b>	<b>-1.054E+01</b>	-1.054E+01
Mean Rank	2.35	2.13	<b>1.52</b>



**Figure 4.9:** The convergence and diversity plots of iECSA using different values of migration frequency

The convergence curves confirm that a higher value of the  $M_f$  results in a better convergence rate. According to the diversity behavior, it is evident that the *iECSA* maintains high diversity before rapidly decreasing it for a short period of time when migration is triggered. This is due to the fittest individuals from the source islands replacing the worst individuals, which empowers the exploitation potential. However, when the value of  $M_f$  is 100 (i.e., when the migration process is triggered five times within the allowed 500 iterations), the



iECSA retains better diversity behavior.

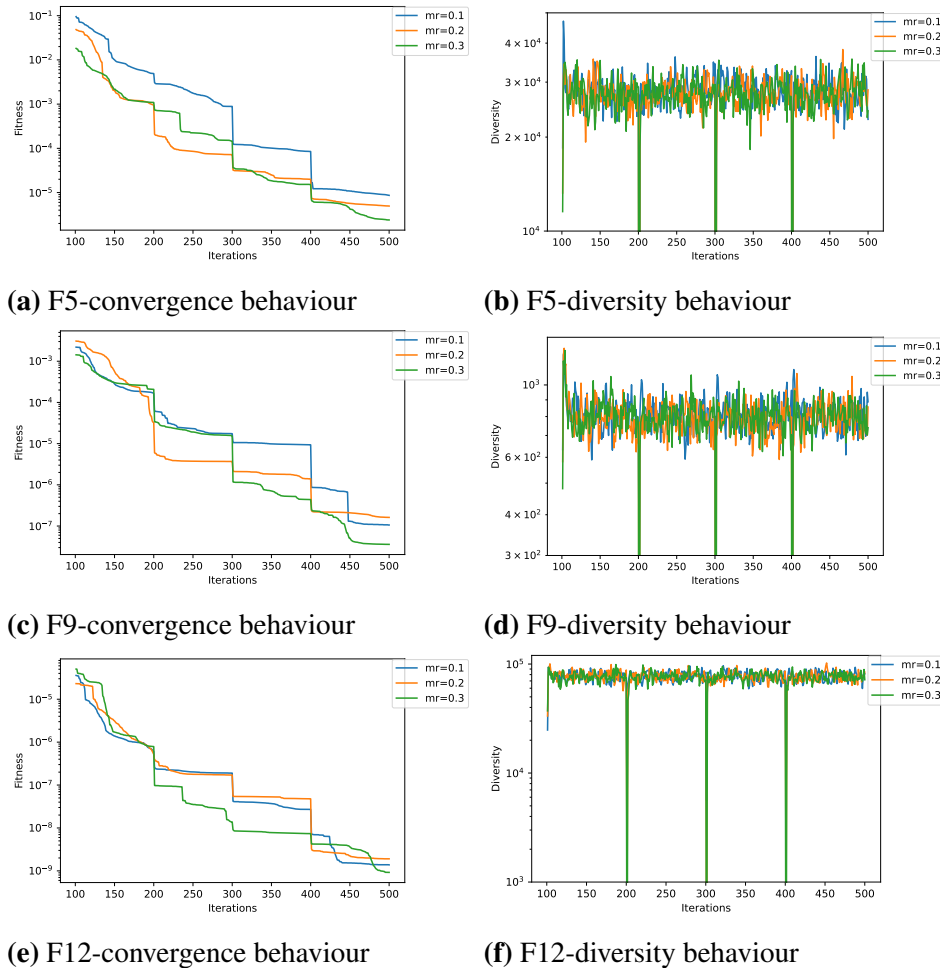
**3 Impact of migration rate:** Three experimental scenarios are set up in this part to investigate the impact of  $M_r$  on the performance of *iECSA*. In these cases, the parameter  $M_r$  is examined using three different values:  $M_r = 10\%$  (Sen7),  $M_r = 20\%$  (Sen8), and  $M_r = 30\%$  (Sen9). It should be remembered that  $M_r$  decides how many migrants are transferred between islands. When  $M_r = 30\%$ , for instance, this indicates that 30% of the source island individuals are selected to be exchanged. It should be noted that the other parameters are fixed to the best settings obtained in the previous experiments. Table 4.10 displays the experimental outcomes of the *iECSA* using different values of  $M_r$  for the standard benchmarks. The *iECSA* with  $M_r$  of 0.1 achieves the optimal rank of 1.74 with the best results on 12 out of 23 cases.

**Table 4.10:** The impact of migration rate on the convergence of *iECSA*

Function	Sen7 mr=0.1	Sen8 mr=0.2	Sen9 mr=0.3
F1	<b>7.67E-08</b>	1.92E-07	1.78E-07
F2	1.08E-04	1.71E-04	<b>1.05E-04</b>
F3	<b>4.19E-05</b>	1.53E-04	1.12E-04
F4	<b>4.96E-05</b>	7.77E-05	9.49E-05
F5	8.64E-06	4.95E-06	<b>2.40E-06</b>
F6	<b>2.92E-07</b>	3.58E-07	2.93E-07
F7	<b>2.15E-04</b>	2.16E-04	2.26E-04
F8	-1.26E+04	<b>-1.26E+04</b>	-1.26E+04
F9	1.06E-07	1.62E-07	<b>3.59E-08</b>
F10	<b>4.62E-05</b>	4.82E-05	7.09E-05
F11	<b>3.27E-07</b>	5.08E-07	5.23E-07
F12	1.38E-09	1.91E-09	<b>9.22E-10</b>
F13	2.62E-08	<b>1.07E-08</b>	3.37E-08
F14	9.98E-01	9.98E-01	<b>9.98E-01</b>
F15	3.07E-04	<b>3.07E-04</b>	3.07E-04
F16	<b>-1.03E+00</b>	<b>-1.03E+00</b>	-1.03E+00
F17	<b>3.98E-01</b>	<b>3.98E-01</b>	3.98E-01
F18	<b>3.00E+00</b>	<b>3.00E+00</b>	3.00E+00
F19	-3.86E+00	-3.86E+00	<b>-3.86E+00</b>
F20	-3.32E+00	-3.32E+00	<b>-3.32E+00</b>
F21	<b>-1.02E+01</b>	<b>-1.02E+01</b>	-1.02E+01
F22	-1.04E+01	-1.04E+01	<b>-1.04E+01</b>
F23	<b>-1.05E+01</b>	<b>-1.05E+01</b>	-1.05E+01
Mean Rank	<b>1.74</b>	2.22	2.04

A graphic representation of the convergence and diversity curves, utilizing various values of the migration rate, is provided in Figure 4.10. It is evident from the convergence curve that a lower  $M_r$  value yields a better convergence rate. In terms of diversity behavior, it is clear that *iECSA* with various maintains a high diversity. However, when the migration is initiated,

the diversity drops significantly when  $M_r = 30\%$  compared to smaller  $M_r$  values. These findings affirm that a large value of  $M_r$  reduces the diversity of the island members, which increases the likelihood of early convergence since if we interchange the fittest solutions, the majority of islands may converge to the same solutions.



**Figure 4.10:** The convergence and diversity plots of iECSA using different values of migration rate

### 4.3.5 Comparison of iECSA with Well-Established Methods

Following a thorough investigation of the suggested iECSA, the next step involves evaluating how well the proposed method performs in comparison to 17 well-established and recent optimizers. These comparison methods encompass a diverse set of optimization categories, including swarm-based, evolutionary-based, math-based, and physics-based algorithms [235]. The swarm-based category includes Aquila Optimization (AO) [228], Bat-inspired Algorithm (BAT) [126], Capuchin Search Algorithm (CapSA) [46], Grey Wolf Optimizer (GWO)

[127], Hunger Games Search (HGS) [131], Harris Hawks Optimization (HHO) [39], Jaya Algorithm [236], Moth-Flame Optimization (MFO) [128], Particle Swarm Optimization (PSO) [31], Slap Swarm Algorithm (SSA) [237], and Whale Optimization Algorithm (WOA) [80]. The evolutionary-based category comprises Differential Evolution (DE) [105] and Flower Pollination Algorithm (FPA) [231]. The math-based category encompasses Arithmetic Optimization Algorithm (AOA) [230], Sine Cosine Algorithm (SCA) [227], and Success History Intelligent Optimizer (SHIO) [232]. The physics-based category is represented by Equilibrium Optimizer (EO) [229]. The selection of these algorithms was based on their representation of distinct MH classes in terms of inspiration and mathematical formulation. Moreover, they incorporate both newly developed methods like HHO, HGS, AOA, SHIO, CapSA, and AO, along with widely adopted optimizers like DE, PSO, and GWO, ensuring a comprehensive evaluation across various optimization paradigms. The optimizers' parameters were adjusted in accordance with the settings suggested in previous related studies. Descriptions and specifics of the parameter settings applied in this work are detailed in Section 4.1.2. It is noteworthy that the algorithms under comparison were written in Python using the provided source codes for each individual algorithm as well as the fundamental ideas described in their original articles. Then, in order to permit a thorough examination, these implementations were incorporated into our framework.

The data presented in Table 4.11 provides the outcomes of a comparative analysis between iECSA and other MAs across F1-F23 functions. The average and standard deviation of the best-obtained fitness values for 30 independent runs are reported. In this context, the symbols "W", "L", and "T" (refers to Win, Loss, and Tie) are employed to signify that iECSA demonstrates superior, inferior, or equivalent performance to the corresponding algorithm. The "Mean rank" metric, which represents the algorithm's average ranking, is derived from the Friedman test. It is clear from Table 4.11 that the iECSA performs noticeably better than other approaches for the majority of the test cases with the lowest overall ranking of 4.89. Specifically, the iECSA outperforms its competitors by achieving the lowest fitness results across 10 functions. It becomes clear from pair-wise comparisons of iECSA and each of the other methods that iECSA consistently outperforms them over a notable range of cases,

**Table 4.11: Comparison between iECSA and other algorithms for the unimodal, multimodal, and fixed-dimension multimodal benchmark functions**

Function	Measure	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AO	CmPSA1	EO	HGS	AOA	FPA	MFO	SHHO	iECSA	
F1	AVG	1.04E+04	1.72E+02	1.55E+02	3.07E-05	1.50E-08	4.11E-91	1.77E+00	8.70E-139	2.69E-105	7.11E-05	9.61E-18	6.39E-42	<b>8.64E-184</b>	1.14E-24	6.71E+03	3.01E+03	1.09E-28	7.67E-08	
	STD	4.27E+03	7.21E-03	3.49E+01	3.63E-05	3.89E-09	2.19E-90	4.39E+00	2.63E-138	9.80E-105	1.46E-04	3.70E-17	1.94E-41	0.00E+00	9.93E-25	6.49E+02	6.51E+03	3.34E-28	1.41E-07	
F2	AVG	2.35E+04	9.37E+02	1.41E+01	2.00E+00	7.05E-01	9.81E-56	6.03E-03	1.05E-74	5.13E-54	4.96E-03	1.29E-09	1.93E-24	<b>9.67E-95</b>	2.16E-13	1.19E+02	3.18E+01	3.94E-18	1.08E-04	
	STD	1.28E+05	3.15E-02	7.58E+00	4.84E+00	8.58E-01	2.66E+00	9.09E-03	1.90E-74	2.08E-53	4.48E-03	1.46E-09	2.76E-24	5.29E-94	2.03E-13	3.62E+01	1.98E+01	3.57E-18	1.94E-04	
F3	AVG	3.48E+04	3.07E+04	3.94E+04	4.31E+01	2.74E+02	2.22E+04	4.31E+03	9.35E-25	2.33E-85	9.25E-03	4.97E-15	1.22E-14	<b>3.02E-148</b>	3.89E-23	9.38E+03	2.19E+04	3.47E-07	4.19E-05	
	STD	1.55E+04	4.45E+03	4.95E+03	1.47E+01	1.57E+02	1.03E+04	3.28E+03	2.58E-24	1.27E-84	2.05E-02	1.20E-14	4.78E-14	1.66E-147	4.55E-23	1.37E+03	1.51E+04	6.34E-07	8.98E-05	
F4	AVG	4.02E+01	1.23E+01	9.38E+01	3.67E+01	5.10E+00	3.63E+01	2.77E+01	4.81E-21	1.74E-51	7.94E-04	6.77E-10	7.41E-12	<b>1.67E-82</b>	1.03E-12	5.77E+01	6.57E+01	5.13E-08	4.96E-05	
	STD	8.75E+00	2.14E+00	6.90E+00	1.61E-01	2.48E+00	2.56E+01	9.67E+00	8.46E-21	1.15E-51	6.70E-04	9.77E-10	1.26E-11	9.16E-82	1.13E-12	2.99E+00	9.03E+00	6.24E-08	6.98E-05	
F5	AVG	4.18E+06	8.79E+01	2.68E+04	6.86E+01	9.00E+01	2.70E+01	5.72E+03	2.66E+01	2.28E-03	7.09E-04	4.49E-03	2.65E+01	2.66E+01	2.90E+01	4.56E+06	2.68E+06	2.84E+01	<b>8.64E-46</b>	
	STD	2.99E+06	5.43E+01	1.27E+04	6.34E+01	1.01E+02	2.90E-01	1.60E+04	5.07E-01	2.64E-03	1.60E-03	7.32E-03	4.57E-01	1.47E+00	1.86E-07	1.21E+06	1.46E+07	7.01E-01	2.95E-05	
F6	AVG	1.08E+04	1.76E-02	1.83E+02	1.77E-05	<b>1.63E-08</b>	2.03E-02	5.95E+00	2.40E-01	2.99E-05	1.36E-04	2.61E-04	4.30E-01	1.81E+00	7.50E+00	6.77E+03	9.97E+02	3.00E+00	2.92E-07	
	STD	3.03E+03	6.87E-03	3.44E+01	1.56E-05	3.57E-09	1.82E-02	1.71E+00	1.93E-01	3.79E-05	2.40E-04	6.78E-04	3.18E-01	4.03E-01	1.92E-07	7.34E+02	3.98E+03	6.25E-01	6.90E-07	
F7	AVG	1.60E+00	1.39E-02	8.58E-02	2.75E+00	3.11E-02	1.58E-03	4.58E-02	7.74E-04	6.94E-05	2.35E-03	3.41E-04	3.26E-04	6.81E-04	6.86E-05	8.12E-01	6.46E+00	3.47E-04	1.35E-04	
	STD	-3.51E+03	-5.71E+03	-6.45E+03	-6.45E+03	-7.47E+03	-1.19E+04	-3.98E+03	-6.49E+03	-1.25E+04	-8.60E+03	-1.26E+03	-6.36E+03	-8.53E+03	-2.28E+03	-7.52E+03	-8.45E+03	-2.64E+03	3.41E+02	<b>-1.26E-04</b>
F8	AVG	6.52E+02	3.24E+02	4.15E+02	1.03E+02	8.14E+02	1.09E+03	2.59E+02	7.67E+02	4.63E+02	2.38E+03	1.61E-02	9.22E+02	6.01E+02	4.67E+02	2.21E+02	7.48E+02	3.41E+02	2.08E-03	
	STD	2.18E+01	9.35E+00	1.81E+01	2.71E+01	1.30E+01	0.00E+00	3.64E+01	8.29E+00	0.00E+00	4.17E-04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.21E+01	3.48E+01	2.67E-13	1.06E-07
F9	AVG	7.34E+01	1.88E+02	2.53E+02	8.43E+01	3.89E+01	0.00E+00	3.23E+01	1.98E+00	0.00E+00	3.20E-04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.96E+02	1.65E+02	6.07E-13	1.06E-07	
	STD	2.17E+06	4.26E-02	8.01E+01	4.52E-16	7.76E-07	2.85E+00	2.64E+03	1.48E-02	1.17E-06	1.22E-06	4.801E-03	8.01E-03	2.88E-02	3.44E-02	5.25E+05	2.41E+02	1.68E-01	1.38E-09	
F10	AVG	1.44E+01	5.01E-02	1.71E+00	4.61E-03	1.92E+00	3.64E-15	1.45E+01	7.43E-15	4.44E-16	1.45E-03	4.93E-10	7.99E-16	4.44E-16	1.80E-13	1.78E+01	1.51E+01	4.83E-15	4.62E-05	
	STD	1.23E+02	1.23E-01	2.36E+00	6.57E-03	9.28E-03	6.85E-03	6.86E-01	2.89E-03	0.00E+00	7.92E-06	5.97E-17	0.00E+00	0.00E+00	0.00E+00	6.68E+01	1.30E+01	5.06E-08	3.27E-07	
F11	AVG	3.79E+01	1.24E-01	4.19E-01	9.89E-03	9.64E-03	2.64E-02	2.85E-01	5.05E-03	0.00E+00	1.33E-05	2.33E-16	0.00E+00	0.00E+00	0.00E+00	5.03E+00	3.90E+01	2.77E-07	7.42E-07	
	STD	1.56E+06	5.65E-02	4.82E+01	4.81E-07	4.43E+00	2.85E-03	1.53E+01	2.93E-02	1.17E-06	7.88E-07	4.35E-04	2.47E-02	7.13E-02	1.66E+00	9.18E+05	5.79E+01	3.37E-01	<b>1.38E-09</b>	
F12	AVG	2.27E+06	4.52E-16	9.04E-03	4.63E-01	4.52E-16	1.84E+00	6.86E-01	3.57E+00	1.50E+00	5.03E+00	1.05E+00	4.04E+00	2.50E+00	2.77E+00	3.99E-03	1.91E+00	4.51E+00	3.39E-16	
	STD	1.17E+07	1.13E-01	1.21E+04	2.59E-03	3.84E-02	9.07E-02	1.43E+04	2.53E-01	1.14E-05	7.96E-06	1.80E-05	4.65E-01	8.66E-01	2.99E+00	8.47E+06	9.52E+03	1.80E+00	<b>2.62E-08</b>	
F13	AVG	1.18E+07	4.25E-02	2.91E+04	4.71E-03	1.46E-01	8.40E-02	7.61E+04	1.54E-01	1.86E-05	2.25E-05	3.33E-05	2.10E-01	2.74E-01	1.92E-02	2.51E+06	4.71E+04	2.72E-01	7.58E-08	
	STD	9.46E+00	9.98E-01	1.00E+00	1.30E+00	9.98E-01	1.59E+00	1.26E+00	3.15E+00	1.56E+00	6.79E+00	1.33E+00	7.29E+00	1.91E+00	1.09E+01	1.00E+00	2.58E+00	8.49E+00	<b>9.98E-01</b>	
F14	AVG	6.86E-03	3.38E-04	4.87E-04	2.79E-03	1.40E-03	6.91E-04	1.05E-03	3.77E-03	3.09E-04	1.49E-03	6.69E-04	4.11E-03	1.98E-03	8.04E-02	7.85E-04	1.69E-03	2.79E-03	<b>3.07E-04</b>	
	STD	8.90E-03	1.67E-04	2.73E-04	5.97E-03	3.59E-03	4.93E-04	3.64E-04	7.55E-03	1.15E-06	4.40E-04	4.51E-04	6.76E-03	5.00E-03	4.93E-02	1.63E-04	3.55E-03	6.15E-03	3.93E-16	
F15	AVG	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	<b>-1.03E+00</b>
	STD	1.09E-09	0.00E+00	2.60E-05	0.00E+00	0.00E+00	0.00E+00	1.70E-05	5.15E-09	0.00E+00	9.29E-05	3.43E-08	4.23E-11	6.78E-16	3.72E-01	1.47E-08	6.78E-16	7.17E-08	6.78E-16	
F16	AVG	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	<b>3.98E-01</b>	
	STD	6.69E-10	1.13E-16	7.43E-04	1.13E-16	1.13E-16	8.80E-07	3.00E-03	5.05E-07	2.90E-07	5.96E-04	1.35E-06	1.90E-02	3.12E-16	6.56E-05	4.50E-07	1.13E-16	6.22E-06	1.13E-16	
F17	AVG	3.90E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	1.54E+01	6.60E+00	3.00E+00	<b>3.00E+00</b>	5.95E+01	3.00E+00	<b>3.00E+00</b>	5.70E+00	<b>3.00E+00</b>	
	STD	5.32E-08	0.00E+00	7.83E-04	0.00E+00	0.00E+00	3.09E-06	1.23E-05	4.59E-06	4.93E+00	1.44E+01	9.34E+00	2.91E-05	5.68E-15	8.24E+01	5.97E-06	4.52E-16	1.48E+01	4.52E-16	
F18	AVG	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.77E+00	-3.70E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	<b>-3.86E+00</b>
	STD	4.17E-08	2.26E-15	2.26E-15	2.26E-15	2.26E-15	2.26E-15	2.26E-15	2.26E-15	2.26E-15	1.00E-01	1.54E-01	3.00E-03	3.54E-03	4.08E-05	1.18E-06	1.36E-15	2.81E-03	1.36E-15	
F19	AVG	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-3.25E+00	-2.50E+00	-2.99E+00	-3.25E+00	-3.27E+00	-3.27E+00	-3.32E+00	-3.25E+00	-3.25E+00	-3.25E+00	<b>-3.32E+00</b>
	STD	5.99E-02	5.99E-02	5.99E-02	5.99E-02	5.99E-02	5.99E-02	5.99E-02	5.99E-02	5.99E-02	4.69E-01	1.78E-01	8.21E-02	7.94E-02	5.99E-02	2.55E-03	6.73E-02	2.59E-02	2.26E-04	
F20	AVG	-5.38E+00	-1.02E+01	-6.94E+00	-7.95E+00	-7.97E+00	-9.04E+00	-3.73E+00	-8.89E+00	-5.56E+00	-1.02E+01	-1.01E+01	-6.49E+00	-8.30E+00	-8.64E+00	-1.01E+01	-7.31E+00	-9.92E+00	-1.02E+01	-1.04E+01
	STD	2.89E+00	0.00E+00	1.89E+00	2.56E+00	3.00E+00	2.30E+00	1.88E+00	2.37E+00	1.56E+00	1.34E-03	2.98E-02	2.49E+00	2.96E+00	2.35E+00	8.56E-02	3.41E+00	9.85E-01	1.81E-15	
F21	AVG	-5.58E+00	-1.04E+01	-9.60E+00	-9.52E+00	-8.70E+00	-9.52E+00	-4.50E+00	-9.87E+00	-5.62E+00	-1.04E+01	-1.04E+01	-8.88E+00	-7.05E+00	-8.73E+00	-1.03E+01	-7.15E+00	-1.01E+01	-1.04E+01	-1.04E+01
	STD	3.17E+00	0.00E+00	1.12E+00	2.01E+00	3.17E+00	2.91E+00	1.56E+00	1.62E+00	1.62E+00	1.38E-03	3.40E-02	2.39E+00	3.53E+00	2.91E+00	1.25E-01	3.39E+00	9.81E-01	9.03E-15	
F22	AVG	-5.89E+00	-1.05E+01	-9.39E+00	-1.00E+01	-8.53E+00	-9.09E+00	-5.00E+00	-1.04E+01	-5.63E+00	-1.05E+01	-1.05E+01	-9.94E+00	-7.93E+00	-8.15E+00	-1.02E+01	-6.98E+00	-9.80E+00	-1.05E+01	-1.05E+01
	STD	3.66E+00	0.00E+00	1.19E+00	1.65E+00	3.20E+00	2.72E+00	1.52E+00	9.87E-01	1.68E+00	1.60E-03	3.80E-02	1.63E+00	3.36E+00	3.52E					

ranging from 13 to 23. For instance, iECSA shows remarkable superiority over BAT, SCA, AO, and FPA in all test cases. Additionally, when compared to HGS, iECSA performs better in 13 cases, loses in 7 cases, and achieves equivalent results in 3 cases. We can conclude that iECSA has a notable performance advantage over its counterparts. This demonstrates that the iECSA algorithm is able to solve considered problems by finding a better balance between local exploitation and global search, and as a consequence, it produces satisfactory outcomes.

To establish a solid statistical foundation for the results attained by iECSA, a thorough evaluation using a nonparametric Wilcoxon Ranksum test was conducted. This evaluation aimed to create a meaningful comparison between the suggested algorithm and alternative counterparts. The resulting p-values are outlined in Table B-2 in *Appendix B*. Referring to this table, it is worth highlighting that the majority (90.54%) of the p-values are less than 0.05. A p-value below 0.05 often denotes statistical significance of the findings. In general, iECSA exhibits superiority in approximately 76.98% of instances, is considered inferior in about 13.55% of cases, and performs equivalently in roughly 9.46% of cases. Specifically, among the top-performing algorithms (based on the mean rank), iECSA exhibits superiority over HHO, CapSA, HGS, GWO, and WOA in 14, 15, 11, 18, and 18 cases, respectively.

#### **4.3.6 Results on Challenging CEC2014 Benchmarks**

To further evaluate the effectiveness and applicability of the proposed method, we selected the challenging IEEE CEC2014 test suite [219], which comprises 30 test functions encompassing unimodal, multi-modal, hybrid, and composition scenarios. Detailed descriptions of these functions can be found in Section Section 4.2.1.2. The experimental setup for this section remained consistent with the previous section, with the exception that we increased the number of evaluations to  $70 \times 10^3$  for each tested method (the maximum number of iterations is 1000 and the size of the population is 70).

Table 4.12 presents the experimental outcomes of the proposed iECSA algorithm and other comparative methods when handling CEC2014 test functions, in terms of the means and standard deviations across 30 independent runs. The outcomes also demonstrate iECSA's

**Table 4.12: Comparison between iECSA and other algorithms on IEEE CEC2014 functions**

Function	Measure	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWG	HHO	AO	CapSAI	EO	HGS	AOA	FPA	MFO	SHHO	iECSA
F1-CEC214	AVG	1.15E+09	8.56E+07	1.14E+08	2.50E+06	1.42E+07	8.64E+07	3.26E+08	5.95E+07	5.05E+08	1.45E+09	2.95E+08	1.66E+07	1.72E+07	1.75E+09	6.64E+06	5.60E+07	1.84E+08	5.00E+06
F1-CEC214	STD	6.38E+08	1.98E+07	7.49E+05	9.39E+06	1.27E+08	4.31E+07	2.06E+08	1.47E+08	2.06E+08	1.50E+08	1.50E+08	8.00E+06	1.10E+07	2.54E+08	6.50E+07	6.50E+07	1.20E+08	1.98E+06
F2-CEC214	AVG	6.22E+10	7.66E+02	1.01E+10	7.94E+03	9.90E+08	3.90E+08	2.20E+10	1.03E+09	4.37E+10	8.08E+10	2.53E+08	7.37E+08	8.77E+08	7.88E+10	8.52E+08	1.04E+10	7.19E+09	4.07E+03
F2-CEC214	STD	1.29E+10	2.68E+02	2.17E+09	5.99E+03	9.91E+03	8.63E+07	3.98E+09	8.49E+08	7.89E+09	9.37E+09	1.54E+08	8.77E+08	9.93E+08	2.02E+08	7.09E+09	4.16E+09	4.16E+09	3.88E+03
F3-CEC214	AVG	2.28E+05	3.04E+02	1.17E+05	1.80E+03	4.00E+04	7.52E+04	5.16E+04	3.19E+04	7.09E+04	8.68E+04	6.49E+04	1.05E+04	1.16E+04	4.45E+06	1.59E+04	7.81E+04	7.36E+04	3.8E+03
F3-CEC214	STD	9.87E+04	1.10E+00	2.17E+04	1.22E+03	1.49E+04	5.29E+04	8.53E+03	7.73E+03	6.30E+03	5.55E+03	1.23E+04	2.95E+03	4.51E+03	9.04E+06	2.71E+03	4.78E+04	7.30E+03	1.06E+03
F4-CEC214	AVG	1.00E+04	4.84E+02	1.09E+03	4.92E+02	5.32E+02	7.04E+02	1.73E+03	3.33E+02	5.26E+02	1.73E+03	2.79E+03	5.30E+02	5.66E+02	1.35E+04	6.30E+02	1.18E+03	8.83E+02	4.94E+02
F4-CEC214	STD	3.07E+03	8.22E+00	1.56E+02	2.43E+01	4.00E+01	7.34E+01	3.10E+02	7.24E+01	1.45E+03	1.47E+03	9.13E+02	2.32E+01	5.44E+01	2.35E+03	2.75E+01	6.30E+02	3.15E+02	3.49E+01
F5-CEC214	AVG	5.20E+02	5.21E+02	5.21E+02	5.20E+02	5.21E+02	5.21E+02	5.21E+02	5.21E+02	5.20E+02	5.21E+02	5.21E+02	5.21E+02	5.20E+02	5.21E+02	5.20E+02	5.21E+02	5.20E+02	5.20E+02
F5-CEC214	STD	1.68E-01	8.46E-02	5.60E-02	9.52E-02	1.15E-01	9.82E-02	3.77E-02	2.99E-01	1.73E-01	7.15E-02	1.44E-01	1.30E-01	7.06E-02	7.55E-02	5.10E-02	1.30E-01	1.41E-01	5.62E-04
F6-CEC214	AVG	6.45E+00	6.33E+02	6.18E+02	6.19E+02	6.18E+02	6.19E+02	6.37E+02	6.15E+02	6.40E+02	6.35E+02	6.35E+02	6.08E+02	6.19E+02	6.45E+02	6.31E+02	6.22E+02	6.23E+02	6.27E+02
F6-CEC214	STD	2.27E+00	4.55E+00	1.53E+00	4.28E+00	5.46E+00	6.30E+00	3.83E+00	2.95E+00	2.36E+00	2.44E+00	3.07E+00	1.54E+00	2.56E+00	1.63E+00	1.25E+00	2.95E+00	2.40E+00	1.81E+00
F7-CEC214	AVG	1.38E+03	7.00E+02	7.32E+02	7.01E+02	7.00E+02	7.00E+02	8.85E+02	7.12E+02	7.99E+02	1.36E+03	8.42E+02	7.05E+02	7.05E+02	1.45E+03	7.10E+02	7.71E+02	7.69E+02	7.00E+02
F7-CEC214	STD	1.41E+02	5.29E+05	6.50E+00	1.96E+00	9.63E+03	1.13E+00	3.96E+01	1.11E+01	7.59E+01	1.04E+02	4.44E+01	1.75E+00	5.47E+00	5.05E+01	1.83E+00	4.55E+01	4.88E+01	5.53E-03
F8-CEC214	AVG	1.05E+03	9.52E+02	1.05E+03	9.12E+02	9.47E+02	9.83E+02	1.06E+03	8.90E+02	9.85E+02	1.14E+03	1.00E+03	8.68E+02	8.62E+02	1.25E+03	9.85E+02	9.17E+02	9.95E+02	9.03E+02
F8-CEC214	STD	2.77E+01	1.08E+01	1.58E+01	1.76E+01	2.81E+01	2.67E+01	2.81E+01	2.14E+01	2.11E+01	2.12E+01	2.83E+01	1.49E+01	1.49E+01	2.03E+01	2.21E+01	1.80E+01	2.64E+01	1.16E+01
F9-CEC214	AVG	1.18E+03	1.08E+03	1.18E+03	1.02E+03	1.06E+03	1.15E+03	1.19E+03	1.00E+03	1.12E+03	1.28E+03	1.17E+03	1.05E+03	1.05E+03	1.31E+03	1.19E+03	1.07E+03	1.11E+03	1.10E+03
F9-CEC214	STD	3.98E+01	1.22E+01	1.56E+01	2.23E+01	3.64E+01	6.71E+01	2.42E+01	2.96E+01	2.62E+01	3.05E+01	2.89E+01	1.84E+01	1.84E+01	2.88E+01	2.92E+01	3.94E+01	2.57E+01	1.35E+01
F10-CEC214	AVG	6.18E+03	5.55E+03	7.32E+03	3.91E+03	4.62E+03	5.71E+03	7.44E+03	3.60E+03	5.65E+03	8.59E+03	5.79E+03	2.27E+03	2.16E+03	9.97E+03	4.18E+03	4.55E+03	5.44E+03	3.88E+03
F10-CEC214	STD	8.70E+02	3.35E+02	4.04E+02	5.90E+02	8.04E+02	6.83E+02	4.33E+02	9.22E+02	6.33E+02	8.84E+02	1.04E+03	3.69E+02	3.16E+02	3.78E+02	1.43E+02	8.03E+02	6.25E+02	5.21E+02
F11-CEC214	AVG	6.52E+03	8.24E+03	8.32E+03	4.80E+03	5.08E+03	8.40E+03	8.40E+03	4.41E+03	6.35E+03	8.95E+03	6.92E+03	3.50E+03	3.88E+03	1.02E+04	5.22E+03	5.25E+03	6.22E+03	4.12E+03
F11-CEC214	STD	1.11E+03	3.75E+02	3.59E+02	8.01E+02	8.04E+02	8.36E+02	2.48E+02	8.12E+02	8.11E+02	4.60E+02	8.45E+02	4.50E+02	4.64E+02	2.20E+02	2.20E+02	8.13E+02	6.99E+02	3.85E+02
F12-CEC214	AVG	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03
F12-CEC214	STD	9.38E-01	4.00E-01	2.95E-01	1.26E-01	1.51E-01	6.01E-01	3.36E-01	1.06E+00	4.73E-01	4.46E-01	3.48E-01	8.15E-02	1.13E-01	6.77E-01	1.80E-01	1.40E-01	2.35E-01	1.84E-01
F13-CEC214	AVG	1.31E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03
F13-CEC214	STD	8.30E-01	6.10E-02	3.60E-01	1.18E-01	1.18E-01	1.18E-01	1.18E-01	8.30E-02	5.71E-01	9.82E-01	1.15E+00	6.69E-02	1.41E-01	5.17E-01	5.00E-02	1.05E+00	9.66E-01	7.29E-02
F14-CEC214	AVG	1.62E+03	1.40E+03	1.42E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03	1.40E+03
F14-CEC214	STD	4.73E+01	8.47E+02	3.84E+00	1.04E+01	1.91E-01	1.80E-01	2.98E+00	2.43E+01	3.26E+01	2.83E+01	1.53E+04	1.82E+00	6.70E-01	3.07E+01	4.08E-02	1.62E+03	1.42E+03	1.40E+03
F15-CEC214	AVG	3.81E+05	1.52E+03	1.63E+03	1.51E+03	1.51E+03	1.61E+03	1.70E+03	1.55E+03	1.68E+04	1.96E+05	1.13E+04	1.52E+04	1.52E+04	1.55E+03	1.58E+03	3.53E+04	2.91E+03	1.51E+03
F15-CEC214	STD	3.26E+05	1.42E+00	1.51E+01	3.34E+00	4.93E+00	5.35E+01	7.76E+01	1.11E+00	6.27E+04	1.16E+05	1.11E+04	1.81E+04	1.81E+04	1.68E+05	1.25E+01	6.42E+04	2.35E+03	3.00E+00
F16-CEC214	AVG	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03
F16-CEC214	STD	5.43E-01	1.82E-01	1.55E-01	5.52E-01	5.06E-01	4.55E-01	2.74E-01	1.74E-01	4.50E-01	2.44E-01	3.91E-01	5.71E-01	3.34E-01	2.11E-01	1.84E-01	5.89E-01	4.47E-01	4.23E-01
F17-CEC214	AVG	6.60E+07	9.96E+05	7.34E+06	3.29E+05	6.32E+05	7.02E+06	1.08E+07	1.67E+06	3.80E+07	1.43E+08	1.36E+07	6.65E+05	2.18E+06	1.81E+08	2.12E+04	2.60E+06	8.18E+06	6.65E+04
F17-CEC214	STD	4.75E+07	4.15E+05	2.98E+06	1.86E+05	3.69E+05	5.23E+06	4.30E+06	4.30E+06	2.99E+07	9.23E+07	1.57E+08	1.47E+06	1.47E+06	7.06E+07	4.15E+03	2.81E+06	8.09E+06	5.02E+04
F18-CEC214	AVG	1.50E+09	5.63E+03	5.11E+07	3.23E+03	1.35E+04	1.90E+05	2.47E+08	1.06E+07	2.93E+08	4.41E+09	1.57E+08	7.60E+04	1.20E+04	6.24E+09	2.82E+03	8.93E+06	5.84E+07	2.04E+03
F18-CEC214	STD	1.29E+09	2.88E+03	6.47E+07	2.51E+03	8.88E+03	4.95E+05	1.74E+08	1.82E+07	4.86E+08	2.79E+09	3.85E+08	3.11E+05	1.12E+04	1.55E+09	3.06E+02	4.71E+07	3.91E+07	2.93E+01
F19-CEC214	AVG	2.34E+03	1.91E+03	1.95E+03	1.93E+03	1.92E+03	1.92E+03	1.92E+03	1.94E+03	2.15E+03	2.49E+03	2.10E+03	1.91E+03	1.92E+03	2.35E+03	1.92E+03	1.92E+03	2.00E+03	1.92E+03
F19-CEC214	STD	1.37E+02	1.25E+00	3.16E+01	2.69E+01	1.48E+01	5.12E+01	2.69E+01	2.72E+01	9.37E+01	1.75E+02	9.19E+01	1.69E+00	3.11E+01	1.07E+02	2.19E+00	4.60E+01	2.57E+01	2.08E+00
F20-CEC214	AVG	1.64E+06	2.12E+03	3.70E+04	1.34E+04	1.41E+04	1.41E+04	1.41E+04	2.04E+04	1.19E+05	8.91E+05	5.72E+04	3.02E+04	1.31E+04	3.11E+04	2.97E+03	5.54E+04	9.14E+04	2.36E+03
F20-CEC214	STD	2.10E+06	1.50E+01	1.65E+04	6.98E+03	6.27E+03	4.26E+04	1.92E+04	7.91E+03	1.11E+05	1.04E+06	3.72E+04	1.21E+04	4.65E+03	3.32E+06	2.48E+02	2.75E+04	4.44E+04	1.20E+02
F21-CEC214	AVG	2.24E+07	1.34E+04	1.73E+06	9.28E+04	2.18E+05	3.02E+06	2.16E+06	1.22E+06	1.27E+07	4.62E+07	5.33E+06	4.30E+05	5.07E+05	8.02E+07	5.95E+03	1.03E+06	4.06E+06	2.47E+04
F21-CEC214	STD	2.22E+07	3.85E+03	8.20E+05	7.04E+04	1.34E+05	2.72E+06	1.43E+06	1.43E+06	1.22E+06	1.43E+06	5.27E+06	2.79E+05	4.69E+05	6.17E+07	6.17E+02	2.14E+06	3.62E+06	1.73E+04
F22-CEC214	AVG	3.85E+03	2.63E+03	3.00E+03	2.87E+03	2.77E+03	3.02E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03	3.18E+03
F22-CEC214	STD	7.15E+02	1.20E+02	1.50E+02	2.30														

superiority, inferiority, and equivalence to each of the test algorithms. Notably, the comparison reveals a number of patterns. It is clear that iECSA outperforms BAT in all test cases. Additionally, the constantly high superiority counts against algorithms like Jaya, WOA, SCA, SSA, AO, MFO, FPA, and SHIO highlight iECSA's ability to outperform its competitors across various problem scenarios. However, the findings also exhibit that iECSA may have certain limitations when dealing with specific types of functions. Comparing iECSA's performance to DE, EO, and HGS reveals a more balanced mix of superior and inferior results. This implies that while iECSA performs well in many cases, it might not be the best option for every optimization problem.

The Friedman test's mean rank gives a valuable overview of overall performance. Better performance is indicated by a lower mean rank, and iECSA's score of 5.02 indicates that it is competent across all test functions. It is important to note that in the case of composite functions, HHO, HGS, and AOA methods with mean ranks of 11.25, 5.23, and 13.26 respectively, exhibit a clear advantage over the examined methods. The p-values of the Wilcoxon rank-sum test are detailed in Table B-3 (Appendix B) to confirm the existence of significant differences. It is evident that 386 of the 510 examined samples—representing 92.75% of the whole dataset—yield values below 0.05. In these instances, iECSA outperforms its counterparts in 386 occurrences, comprising 75.69% of the whole dataset. When iECSA is compared to the controlled methods (the methods with the top ranks), iECSA outperforms HGS statistically in 16 scenarios, displays inferiority in 11 cases, and exhibits equivalency in 3 cases. In comparison with EO, iECSA surpasses it in 14 instances, displays inferiority in 12 instances, and maintains equivalence in 4 instances. This thorough analysis confirms that iECSA has statistical benefits over the other competitive methods.

To provide a comprehensive overview, we present the Friedman mean rank of the comparison methods for both sets of test functions in Table 4.13 and illustrate the results in Figure 4.11. These findings further reinforce the consistent and reliable behavior of iECSA when compared to other approaches across both sets of functions. This implies that iECSA exhibits a dependable and assured performance, making it a promising choice for various optimization problems. Additionally, Table 4.14 summarizes the comparisons based on Wilcoxon test

**Table 4.13:** Average rankings of the algorithms calculated using Friedman’s test

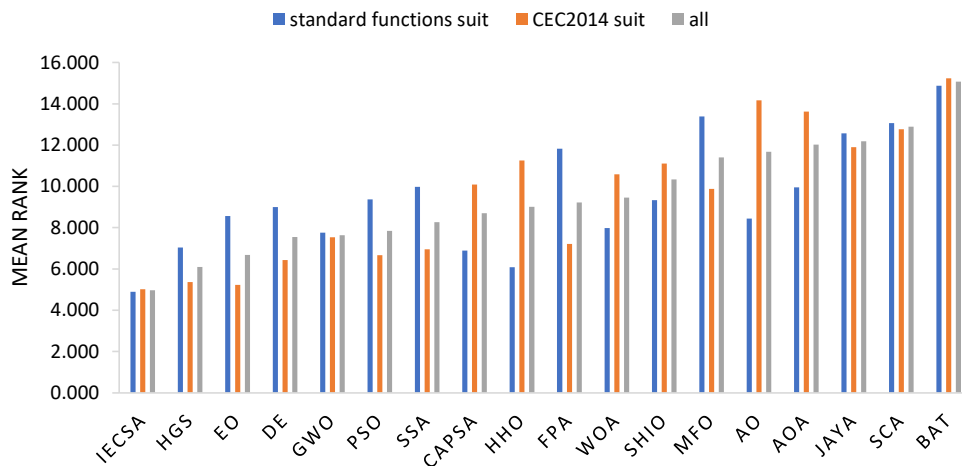
Algorithm	Mean Rank on F1-F23	Mean Rank on CEC2014	Mean Rank on all functions	Rank
iECSA	4.891	5.017	4.962	1
HGS	7.043	5.367	6.094	2
EO	8.565	5.233	6.679	3
DE	9.000	6.433	7.547	4
GWO	7.761	7.533	7.632	5
PSO	9.370	6.667	7.840	6
SSA	9.978	6.950	8.264	7
CapSA1	6.891	10.083	8.698	8
HHO	6.087	11.250	9.009	9
FPA	11.826	7.217	9.217	10
WOA	7.978	10.583	9.453	11
SHIO	9.326	11.100	10.330	12
MFO	13.391	9.883	11.406	13
AO	8.435	14.167	11.679	14
AOA	9.957	13.617	12.028	15
JAYA	12.565	11.900	12.189	16
SCA	13.065	12.767	12.896	17
BAT	14.870	15.233	15.075	18

results for superior, inferior, and equal outcomes. These findings validate the effectiveness of iECSA in tackling a wide range of optimization problems, showcasing its competitiveness and reliability. However, it is worth noting that further investigations and improvements can be pursued to enhance its performance and adaptability in addressing more complex and diverse real-world problems. The findings from this study contribute to the growing body of knowledge in the field of optimization algorithms and provide valuable insights for future research in this area.

**Table 4.14:** Comparison of iECSA Performance: Wilcoxon Test Results for Superior, Inferior, and Equal Outcomes

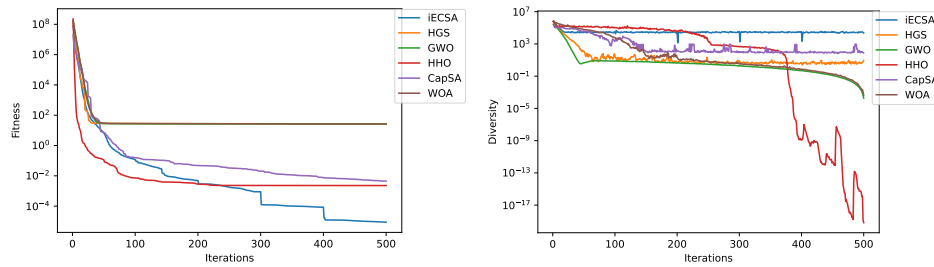
23 standard functions				30 CEC2014 functions				All functions			
Algorithm	Superior(+)	Inferior(-)	Equal(=)	Algorithm	Superior(+)	Inferior(-)	Equal(=)	Algorithm	Superior(+)	Inferior(-)	Equal(=)
BAT	23	0	0	BAT	29	0	1	BAT	52	0	1
DE	13	0	10	DE	13	9	8	DE	26	9	18
JAYA	22	0	1	JAYA	29	1	0	JAYA	51	1	1
PSO	18	0	5	PSO	17	7	6	PSO	35	7	11
SSA	16	0	7	SSA	22	2	6	SSA	38	2	13
WOA	18	4	1	WOA	25	1	4	WOA	43	5	5
SCA	23	0	0	SCA	28	2	0	SCA	51	2	0
GWO	18	5	0	GWO	22	6	2	GWO	40	11	2
HHO	14	8	1	HHO	24	6	0	HHO	38	14	1
AO	23	0	0	AO	26	4	0	AO	49	4	0
CapSA	15	7	1	CapSA	23	7	0	CapSA	38	14	1
EO	14	7	2	EO	14	12	4	EO	28	19	6
HGS	11	7	5	HGS	16	11	3	HGS	27	18	8
AOA	15	8	0	AOA	23	7	0	AOA	38	15	0
FPA	23	0	0	FPA	23	4	3	FPA	46	4	3
MFO	19	0	4	MFO	27	3	0	MFO	46	3	4
SHIO	16	7	0	SHIO	25	5	0	SHIO	41	12	0
Sum	301	53	37	Sum	386	87	37	Sum	687	140	74
%	76.98%	13.55%	9.46%	%	75.69%	17.06%	7.25%	%	76.25%	15.54%	8.21%





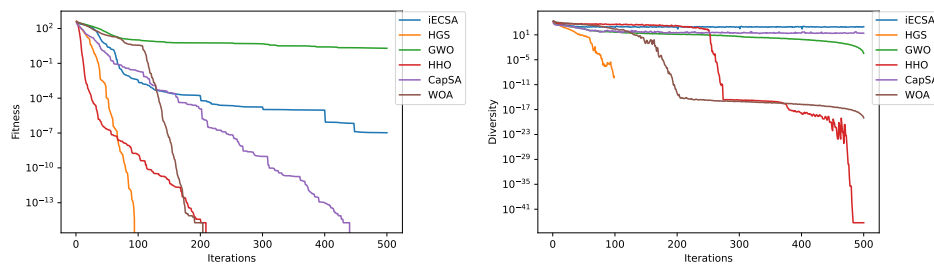
**Figure 4.11:** Schematic view of the results of Friedman rank test (standard and CECE2014 suites) based on results in Tables 4.11 and 4.12

The convergence curves and population diversity curves of comparison methods for chosen test functions are shown side by side in Figures 4.12 and 4.13. This makes it possible to comprehend how population diversity influences the optimization process. It is clear from a comparison of the population diversity with the convergence curves that the *iECSA* successfully maintained a high population diversity until the promising area was satisfied. This trend was repeated for the majority of the functions, demonstrating the algorithm’s efficacy in preserving population diversity. On the other side, the population diversity of other algorithms like SSA and WOA decreased while the convergence curve was too slow over the course of iterations. As a result, these algorithms became stuck early in a local optima.



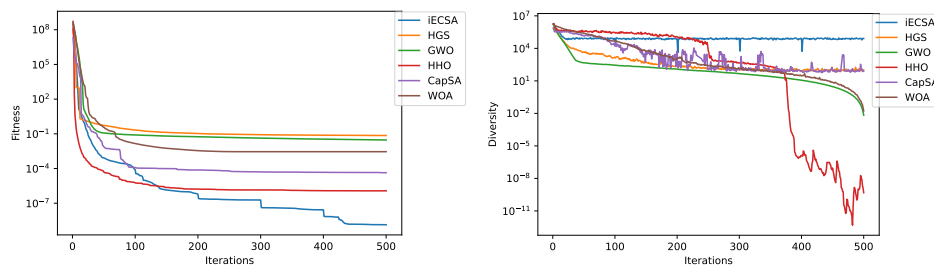
(a) F5-convergence behaviour

(b) F5-diversity behaviour



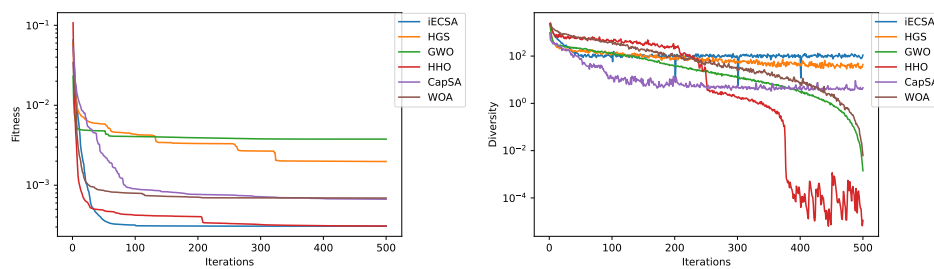
(c) F9-convergence behaviour

(d) F9-diversity behaviour



(e) F12-convergence behaviour

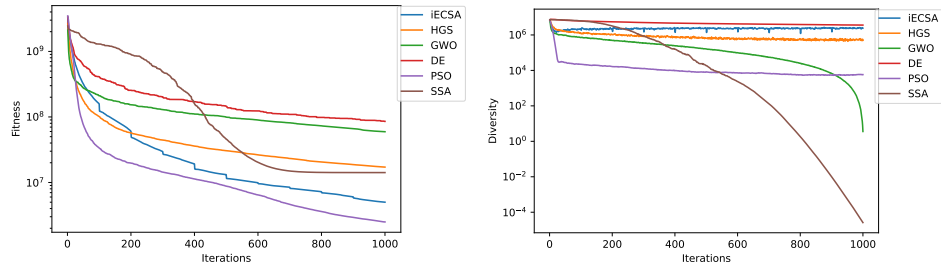
(f) F12-diversity behaviour



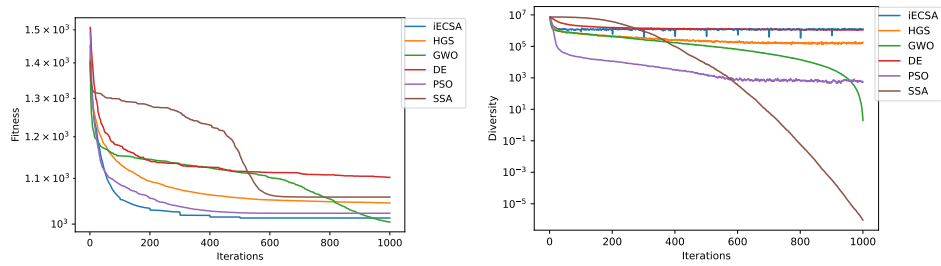
(g) F15-convergence behaviour

(h) F15-diversity behaviour

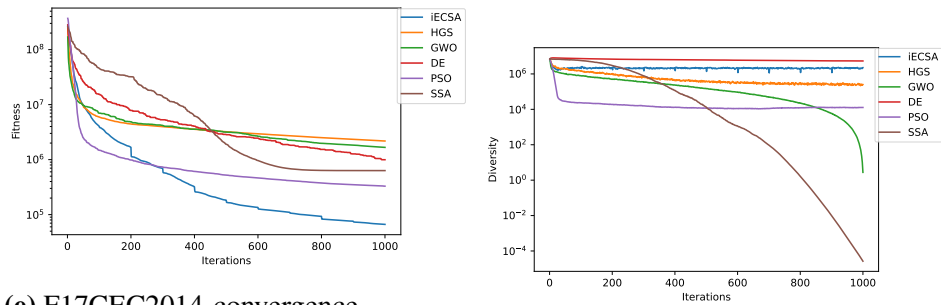
**Figure 4.12:** The convergence and diversity plots of top algorithms on some standard benchmarks



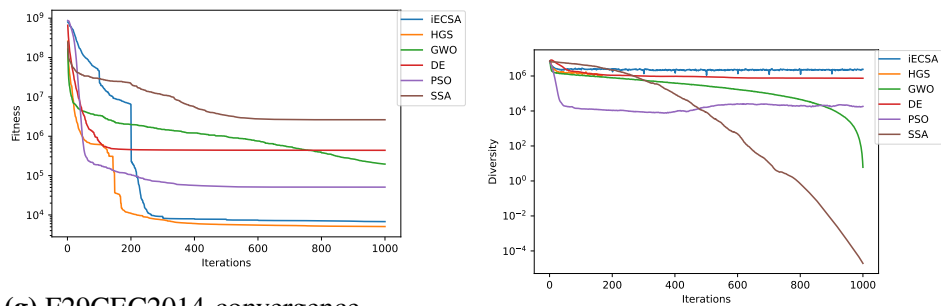
(a) F1CEC2014-convergence behaviour (b) F1CEC2014-diversity behaviour



(c) F9CEC2014-convergence behaviour (d) F9CEC2014-diversity behaviour



(e) F17CEC2014-convergence behaviour (f) F17CEC2014-diversity behaviour



(g) F29CEC2014-convergence behaviour (h) F29CEC2014-diversity behaviour

**Figure 4.13:** The convergence and diversity plots of top algorithms on some CEC2014 benchmarks

## 4.4 Experimental Results of CapSA

This section presents an empirical evaluation of the suggested enhancements to the CapSA. The experiments evaluate the effectiveness of new operators and their incorporation into the cooperative island model, using the same parameter settings presented in Table 4.1. The following subsections offer in-depth findings from these experiments. Initially, the effectiveness of traditional CapSA and its enhanced versions are compared across a variety of real-valued benchmark functions. Next, the investigation focuses on the influence of the adaptive island-based model on both CapSA’s original and improved versions. Lastly, a thorough comparison with other modern algorithms emphasizes the relative performance and advancements resulting from the implemented modifications. Each analysis step underscores the incremental improvements made and places the significance of this research within the wider context of optimization algorithm evolution.

### 4.4.1 Comparison of Traditional and Enhanced CapSA Variants

This subsection presents a comprehensive analysis aimed at examining the impact of the suggested search strategies on the primary CapSA. Hence, a thorough examination of the three established versions of CapSA (MCapSA1, MCapSA2, and MCapSA3) is conducted to determine the superior version based on solution quality, convergence patterns, and diversity curves. Table 4.15 provides a summary of the CapSA variants being examined. In the table, ‘Y’ indicates that the strategy is included, while ‘N’ indicates that the strategy is not included.

**Table 4.15:** Summary of enhanced variants of CapSA and their implications

Variant	Refined Follower Update	Enhanced Local Best Perturbation	Adaptive Dual Update Strategy	Implications
CapSA	N	N	N	Baseline algorithm performance.
MCapSA1	Y	N	N	Improved exploration and convergence speed.
MCapSA2	N	Y	N	Enhanced exploitation and overcoming of local optima.
MCapSA3 (ECapSA)	Y	Y	Y	Better balance of exploration and exploitation, adaptively managed.

The results of the CapSA and its suggested variations, when tested on 30-dimensional standard problems, are comprehensively presented in Table 4.16. The results showed that

MCapSA2 demonstrated the most promising outcomes, outperforming the baseline CapSA in 18 out of 23 cases, or approximately 78.3% of the test instances. MCapSA1 and MCapSA3 also proved to be strong contenders, surpassing the standard CapSA in 13 and 16 cases, respectively. The findings also demonstrated that the average solution quality consistently improved in the upgraded versions. Specifically, MCapSA2 and MCapSA3 had lower standard deviations, suggesting a more reliable and robust search capability. The overall mean rank metrics indicate a highly competitive optimization performance, with MCapSA2 and MCapSA3 sharing the top position with a mean rank of 2.33. They are followed by MCapSA1 with a mean rank of 2.63, and finally the basic CapSA with a mean rank of 2.72. These findings indicate that the proposed search strategies and algorithmic enhancements have contributed to the optimization efficiency of the standard CapSA.

**Table 4.16:** Comparative performance of CapSA and enhanced MCapSA variants on 30-dimensional standard test functions.

Functions	CapSA		MCapSA1		MCapSA2		MCapSA3	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	1.22E-17	3.44E-17	<b>8.56E-19</b>	2.54E-18	2.14E-18	5.54E-18	2.76E-18	8.63E-18
F2	<b>3.76E-10</b>	8.78E-10	4.47E-10	7.20E-10	8.54E-10	1.19E-09	7.66E-10	1.15E-09
F3	<b>1.88E-16</b>	9.54E-16	2.22E-16	5.66E-16	2.49E-15	1.09E-14	7.70E-16	2.36E-15
F4	1.18E-10	2.02E-10	1.45E-10	2.58E-10	<b>8.32E-11</b>	1.38E-10	1.47E-10	3.37E-10
F5	2.12E-03	2.52E-03	2.65E-03	5.51E-03	6.66E-04	1.22E-03	<b>5.97E-04</b>	1.54E-03
F6	1.91E-04	3.84E-04	2.29E-05	4.19E-05	<b>3.59E-06</b>	5.58E-06	8.70E-06	1.29E-05
F7	1.53E-04	1.24E-04	1.76E-04	2.00E-04	<b>1.39E-04</b>	1.52E-04	1.57E-04	1.13E-04
F8	-1.26E+04	1.68E-03	-1.26E+04	8.42E-04	<b>-1.26E+04</b>	3.48E-04	-1.26E+04	8.00E-04
F9	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00
F10	<b>1.21E-10</b>	2.62E-10	2.24E-10	3.68E-10	4.33E-10	6.10E-10	2.00E-10	3.48E-10
F11	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00
F12	1.03E-05	1.37E-05	7.24E-07	1.09E-06	<b>3.14E-07</b>	5.68E-07	4.43E-07	5.18E-07
F13	9.24E-06	1.24E-05	1.43E-05	4.62E-05	4.40E-06	5.84E-06	<b>2.13E-06</b>	3.68E-06
F14	1.16E+00	9.00E-01	9.98E-01	3.67E-11	<b>9.98E-01</b>	4.60E-12	9.98E-01	9.35E-12
F15	5.90E-04	3.30E-04	5.97E-04	4.24E-04	3.50E-04	1.66E-04	<b>3.22E-04</b>	2.74E-05
F16	-1.03E+00	2.87E-09	-1.03E+00	7.52E-15	-1.03E+00	1.99E-15	<b>-1.03E+00</b>	6.78E-16
F17	3.98E-01	7.26E-08	<b>3.98E-01</b>	7.13E-13	3.98E-01	1.92E-12	3.98E-01	2.42E-12
F18	3.00E+00	6.74E-09	3.00E+00	1.70E-13	3.00E+00	8.08E-13	<b>3.00E+00</b>	5.91E-14
F19	-3.81E+00	1.45E-01	-3.86E+00	5.38E-08	-3.86E+00	2.43E-08	<b>-3.86E+00</b>	1.15E-08
F20	-3.14E+00	1.29E-01	<b>-3.29E+00</b>	5.89E-02	-3.26E+00	7.51E-02	-3.28E+00	6.28E-02
F21	-1.02E+01	5.76E-03	-1.02E+01	1.39E-03	-1.02E+01	1.85E-04	<b>-1.02E+01</b>	8.17E-05
F22	-1.04E+01	3.92E-03	-1.04E+01	4.26E-03	<b>-1.04E+01</b>	4.61E-05	-1.04E+01	1.17E-04
F23	-1.05E+01	8.22E-03	-1.05E+01	8.13E-04	<b>-1.05E+01</b>	8.44E-05	-1.05E+01	1.29E-04
WITL	3 2 18		3 2 18		8 2 13		7 2 14	
Mean Rank	2.72		2.63		2.33		2.33	

The improvements introduced to the CapSA algorithm are thoroughly assessed using the challenging IEEE CEC2014 set of real-valued functions. The results presented in Table 4.17 indicate that the enhanced versions of CapSA (MCapSA1, MCapSA2, and MCapSA3) show significant improvements compared to the standard CapSA in the majority of test cases.

**Table 4.17:** Performance comparison of standard CapSA and enhanced variants on IEEE CEC2014 30-dimensional test functions

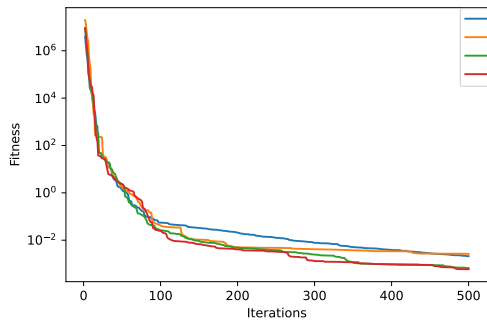
Functions	CapSA		MCApSA1		MCApSA2		MCApSA3	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1-CEC2014	2.258E+08	1.04E+08	1.027E+08	4.78E+07	6.755E+07	3.76E+07	<b>6.553E+07</b>	3.28E+07
F2-CEC2014	2.096E+10	8.99E+09	4.436E+09	3.66E+09	2.226E+09	1.46E+09	<b>2.175E+09</b>	1.10E+09
F3-CEC2014	6.724E+04	1.10E+04	4.529E+04	1.14E+04	<b>3.025E+04</b>	7.98E+03	3.050E+04	1.06E+04
F4-CEC2014	2.972E+03	1.37E+03	9.302E+02	2.45E+02	7.972E+02	1.87E+02	<b>7.387E+02</b>	1.28E+02
F5-CEC2014	5.207E+02	1.42E-01	5.207E+02	1.80E-01	<b>5.205E+02</b>	1.66E-01	5.205E+02	1.80E-01
F6-CEC2014	6.348E+02	1.98E+00	6.315E+02	3.57E+00	6.273E+02	3.19E+00	<b>6.273E+02</b>	3.18E+00
F7-CEC2014	8.580E+02	4.88E+01	7.582E+02	2.24E+01	<b>7.314E+02</b>	1.26E+01	7.320E+02	1.53E+01
F8-CEC2014	1.025E+03	2.86E+01	9.638E+02	2.83E+01	9.423E+02	2.79E+01	<b>9.231E+02</b>	2.34E+01
F9-CEC2014	1.174E+03	3.01E+01	1.105E+03	3.62E+01	1.077E+03	2.93E+01	<b>1.066E+03</b>	3.20E+01
F10-CEC2014	5.789E+03	1.03E+03	5.333E+03	1.04E+03	<b>4.600E+03</b>	9.42E+02	5.080E+03	8.00E+02
F11-CEC2014	7.058E+03	6.52E+02	6.323E+03	8.02E+02	<b>5.881E+03</b>	7.47E+02	6.232E+03	9.55E+02
F12-CEC2014	1.202E+03	6.02E-01	1.201E+03	4.51E-01	1.201E+03	3.52E-01	<b>1.201E+03</b>	4.00E-01
F13-CEC2014	1.304E+03	8.15E-01	1.301E+03	8.39E-01	1.301E+03	1.13E-01	<b>1.301E+03</b>	1.28E-01
F14-CEC2014	1.466E+03	2.53E+01	1.412E+03	1.33E+01	1.405E+03	6.09E+00	<b>1.404E+03</b>	5.33E+00
F15-CEC2014	9.224E+03	9.82E+03	2.440E+03	6.77E+02	1.820E+03	2.25E+02	<b>1.809E+03</b>	2.20E+02
F16-CEC2014	1.613E+03	4.01E-01	1.612E+03	6.10E-01	<b>1.612E+03</b>	5.21E-01	1.612E+03	4.69E-01
F17-CEC2014	2.087E+07	1.74E+07	1.373E+07	3.09E+07	<b>2.900E+06</b>	2.50E+06	3.062E+06	3.38E+06
F18-CEC2014	1.367E+08	3.17E+08	7.662E+07	3.02E+08	1.329E+05	3.29E+05	<b>1.038E+05</b>	2.11E+05
F19-CEC2014	2.054E+03	5.92E+01	1.978E+03	4.25E+01	<b>1.955E+03</b>	4.23E+01	1.963E+03	4.02E+01
F20-CEC2014	6.465E+04	6.28E+04	2.370E+04	1.42E+04	1.532E+04	1.14E+04	<b>1.037E+04</b>	6.14E+03
F21-CEC2014	7.614E+06	9.57E+06	1.561E+06	1.26E+06	9.761E+05	7.32E+05	<b>7.319E+05</b>	6.16E+05
F22-CEC2014	3.171E+03	3.20E+02	2.866E+03	2.33E+02	2.654E+03	1.49E+02	<b>2.636E+03</b>	1.80E+02
F23-CEC2014	2.500E+03	1.44E-08	2.500E+03	1.72E-08	2.500E+03	1.24E-08	<b>2.500E+03</b>	5.84E-09
F24-CEC2014	2.600E+03	2.28E-05	2.600E+03	3.54E-05	<b>2.600E+03</b>	1.94E-05	2.600E+03	3.58E-05
F25-CEC2014	<b>2.700E+03</b>	1.26E-10	2.700E+03	4.05E-10	2.700E+03	2.67E-10	2.700E+03	5.53E-10
F26-CEC2014	2.739E+03	4.71E+01	2.708E+03	2.53E+01	2.704E+03	1.82E+01	<b>2.700E+03</b>	1.10E+01
F27-CEC2014	<b>2.900E+03</b>	2.37E-09	2.900E+03	1.44E-09	2.900E+03	5.75E-09	2.900E+03	1.63E-09
F28-CEC2014	<b>3.000E+03</b>	4.80E-09	3.000E+03	1.17E-08	3.000E+03	8.74E-09	3.000E+03	5.80E-09
F29-CEC2014	3.100E+03	2.59E-02	3.100E+03	1.25E-02	3.100E+03	3.32E-02	<b>3.100E+03</b>	2.19E-02
F30-CEC2014	3.200E+03	3.22E-03	<b>3.200E+03</b>	1.22E-03	3.200E+03	2.72E-03	3.200E+03	3.19E-03
Mean Rank	3.63		2.80		1.88		<b>1.68</b>	

Specifically, MCApSA1 and MCApSA3 perform better than standard CapSA in 83.3% of the test cases, while MCApSA2 shows superior results in 80% of the cases. In addition, the performance on the composite functions (F23-F30) is quite impressive, as all versions demonstrate high-quality solutions close to the optimal. This implies that each version can handle complex optimization problems and produce results close to the most efficient solution. The average rank for each algorithm further supports these findings, with MCApSA3 ranked first with a mean rank of 1.68, MCApSA2 coming in second with a mean rank of 1.88, MCApSA1 coming in third with a mean rank of 2.80, and the standard CapSA coming in fourth with a mean rank of 3.63.

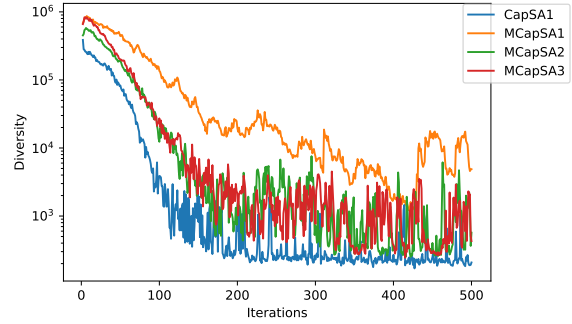
To judge the significance of the results presented in Tables 4.16 and 4.17, the Wilcoxon rank sum test p-values are provided in Tables B-4 and B-5 in the Appendix. These statistical results confirm that each enhanced variant shows either equivalent or superior performance in the majority of test cases compared to the original CapSA.

The convergence and diversity curves for the standard CapSA and its enhanced variants (MCApSA1, MCApSA2, and MCApSA3) are illustrated in Figures 4.14 and 4.15. These figures offer significant insights into the search behaviors of these variants on specific test functions that possess diverse characteristics. It is clear that in the early phases of the search process, algorithms exhibit a notable degree of diversity, which is indicative of the exploration phase wherein a comprehensive search is undertaken to investigate the extensive solution space. As the iterative process advances, there is a decline in diversity, and the trajectories exhibit variability based on the algorithm's characteristics. In the observed search dynamics, the enhanced MCApSA variants are observed to converge faster and maintain a higher level of diversity across the evaluated functions when compared to the standard CapSA. This behavior highlights the effectiveness of the novel strategies incorporated within the enhanced variants, which aim to enrich the search dynamics. Specifically, The standard CapSA exhibits the lowest diversity among the algorithms, indicating a more rapid shift to exploitation that may lead to premature convergence. On the other hand, MCApSA1 displays the highest diversity; while this enhances its convergence trajectory, the outcome is still suboptimal. MCApSA2 presents greater diversity than CapSA but less than MCApSA1, leading to an improved convergence profile. MCApSA3 finds a middle ground, presenting moderate diversity levels, which correlates with a more balanced exploration-exploitation compromise and yields superior convergence performance. The analysis suggests that neither the lowest nor the highest diversity is desirable in isolation, and an equilibrium between the two, as exemplified by MCApSA3, is conducive to the most efficient search performance.

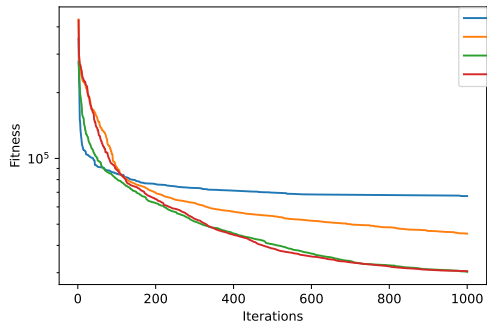
After conducting a comprehensive evaluation of CapSA and its enhanced versions on 53 real-valued test functions, it is evident that the proposed enhancements have significantly improved the optimization capabilities of the original algorithm. Among the investigated enhancements, MCApSA3 is the most remarkable variant. It incorporates an adaptive dual update strategy that combines the strengths of both MCApSA1 and MCApSA2. This combination allows MCApSA3 to provide a balance between exploration and exploitation and thus effectively adapt to various optimization scenarios. Overall, MCApSA3 is an efficient optimization framework that can tackle a wide range of optimization problems. The em-



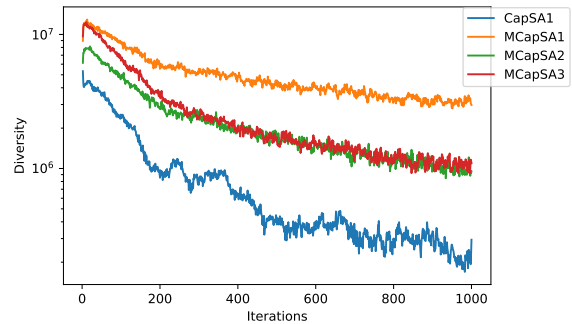
(a) F5-convergence



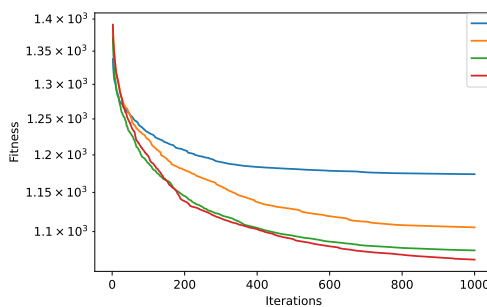
(b) F5-diversity



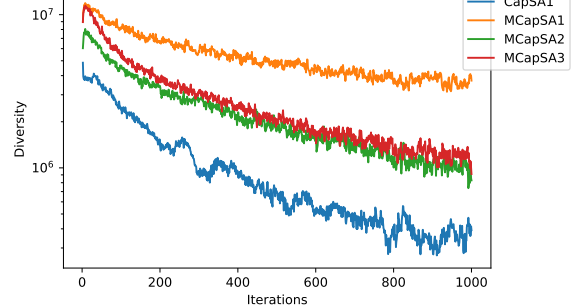
(c) F3CEC2014-convergence



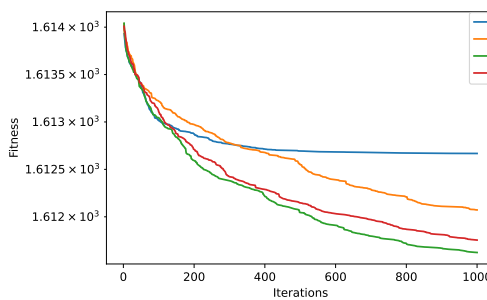
(d) F3CEC2014-diversity



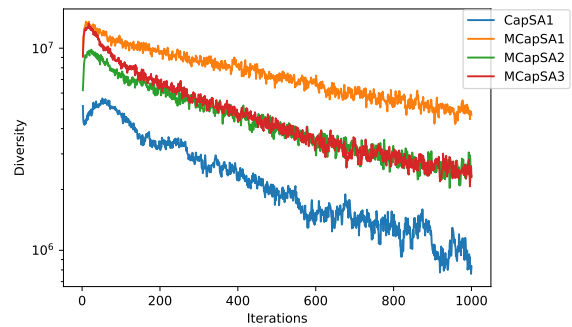
(e) F9CEC2014-convergence



(f) F9CEC2014-diversity



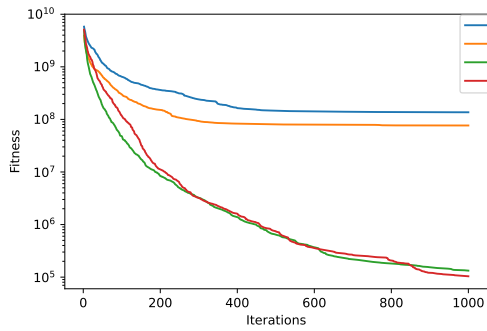
(g) F16CEC2014-convergence



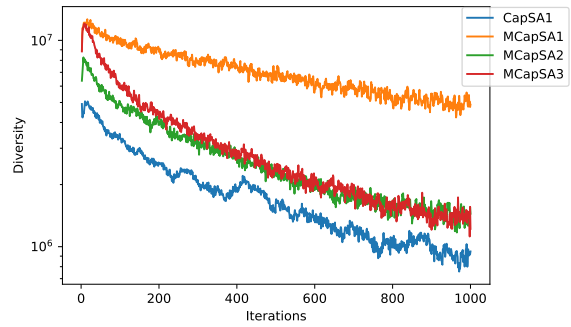
(h) F16CEC2014-diversity

**Figure 4.14:** Convergence and diversity curves for the standard CapSA and its enhanced variants on standard test function F5 and sampled IEEE CEC2014 test functions (F3, F9, and F16)

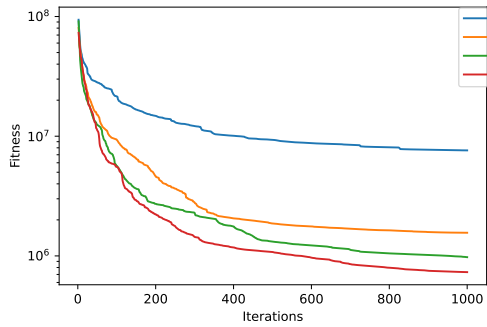




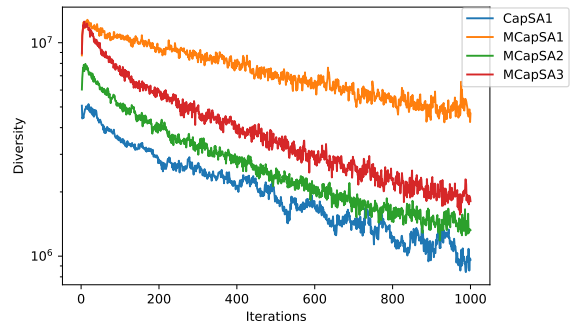
(a) F18CEC2014-convergence



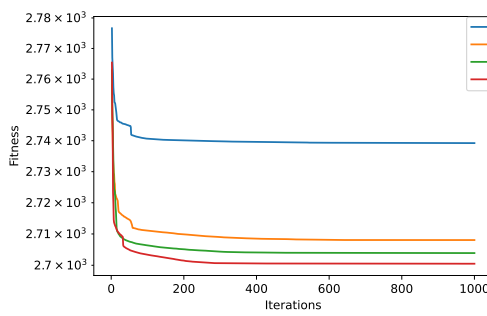
(b) F18CEC2014-diversity



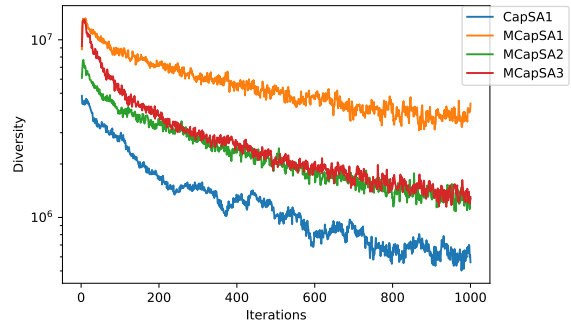
(c) F21CEC2014-convergence



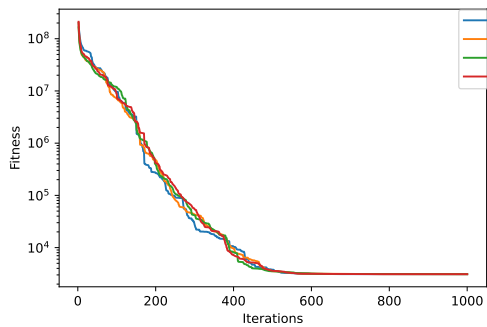
(d) F21CEC2014-diversity



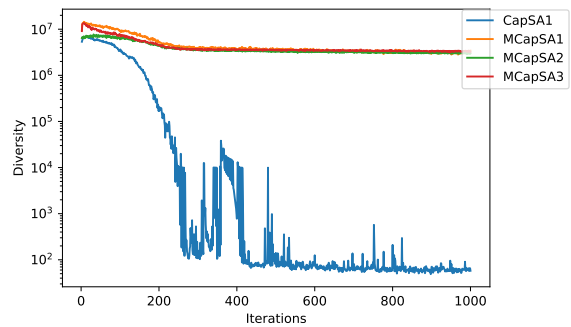
(e) F26CEC2014-convergence



(f) F26CEC2014-diversity



(g) F29CEC2014-convergence



(h) F29CEC2014-diversity

**Figure 4.15:** Convergence and diversity curves for the standard CapSA and its enhanced variants on sampled IEEE CEC2014 test functions (F18, F21, F26, and F29)

pirical results confirm the exceptional performance of MCapSA3, making it a significant advancement in the field of MHs.

#### 4.4.2 Impact of Island-Based Models on CapSA Variants

After verifying the effectiveness of the proposed enhancements to CapSA, this section explores the impact of island-based models on the standard CapSA and its best-performing enhanced variant, MCapSA3, which we will now refer to as ECapSA. These algorithms have been integrated into an island-based framework, resulting in the development of an island-based CapSA (iCapSA) and an equivalent island-based model for ECapSA (iECapSA). This comparison is crucial for understanding how the island model influences the optimization capabilities of both the original and enhanced versions of CapSA, providing insights into potential improvements in their search strategies and overall effectiveness.

**Table 4.18:** Performance comparison of standard CapSA, ECapSA, and their island-based counterparts iCapSA and iECapSA on 23 Standard mathematical functions

Function	CapSA		ECapSA		iCapSA		iECapSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	1.22E-17	3.44E-17	2.76E-18	8.63E-18	1.19E-22	5.54E-22	<b>2.08E-24</b>	8.18E-24
F2	3.76E-10	8.78E-10	7.66E-10	1.15E-09	3.61E-13	7.36E-13	<b>9.16E-14</b>	2.62E-13
F3	1.88E-16	9.54E-16	7.70E-16	2.36E-15	6.48E-19	3.55E-18	<b>1.57E-22</b>	7.58E-22
F4	1.18E-10	2.02E-10	1.47E-10	3.37E-10	3.07E-13	9.28E-13	<b>4.55E-15</b>	8.26E-15
F5	2.12E-03	2.52E-03	5.97E-04	1.54E-03	1.27E-04	2.32E-04	<b>1.86E-05</b>	2.34E-05
F6	1.91E-04	3.84E-04	8.70E-06	1.29E-05	2.92E-06	1.40E-05	<b>2.16E-07</b>	3.54E-07
F7	1.53E-04	1.24E-04	1.57E-04	1.13E-04	4.95E-05	4.63E-05	<b>2.61E-05</b>	1.93E-05
F8	-1.26E+04	1.68E-03	-1.26E+04	8.00E-04	-1.26E+04	2.57E-04	<b>-1.26E+04</b>	8.91E-05
F9	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00
F10	1.21E-10	2.62E-10	2.00E-10	3.48E-10	7.16E-13	3.12E-12	<b>2.35E-14</b>	8.48E-14
F11	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00	<b>0.00E+00</b>	0.00E+00
F12	1.03E-05	1.37E-05	4.43E-07	5.18E-07	2.94E-07	8.41E-07	<b>7.85E-09</b>	1.21E-08
F13	9.24E-06	1.24E-05	2.13E-06	3.68E-06	5.68E-07	1.31E-06	<b>9.65E-08</b>	9.85E-08
F14	1.16E+00	9.00E-01	9.98E-01	9.35E-12	9.98E-01	2.13E-12	<b>9.98E-01</b>	2.21E-15
F15	5.90E-04	3.30E-04	3.22E-04	2.74E-05	3.16E-04	1.48E-05	<b>3.08E-04</b>	7.04E-07
F16	-1.03E+00	2.87E-09	<b>-1.03E+00</b>	6.78E-16	-1.03E+00	7.77E-11	<b>-1.03E+00</b>	6.78E-16
F17	3.98E-01	7.26E-08	3.98E-01	2.42E-12	3.98E-01	3.13E-10	<b>3.98E-01</b>	9.36E-16
F18	3.00E+00	6.74E-09	3.00E+00	5.91E-14	3.00E+00	1.24E-10	<b>3.00E+00</b>	7.72E-15
F19	-3.81E+00	1.45E-01	-3.86E+00	1.15E-08	-3.86E+00	9.86E-08	<b>-3.86E+00</b>	1.19E-10
F20	-3.14E+00	1.29E-01	-3.28E+00	6.28E-02	-3.26E+00	6.04E-02	<b>-3.32E+00</b>	2.17E-02
F21	-1.02E+01	5.76E-03	-1.02E+01	8.17E-05	-1.02E+01	1.37E-04	<b>-1.02E+01</b>	3.34E-06
F22	-1.04E+01	3.92E-03	-1.04E+01	1.17E-04	-1.04E+01	3.23E-04	<b>-1.04E+01</b>	5.64E-06
F23	-1.05E+01	8.22E-03	-1.05E+01	1.29E-04	-1.05E+01	5.58E-04	<b>-1.05E+01</b>	4.17E-06
Mean Rank	3.08		2.82		2.28		<b>1.82</b>	

The comparative results of the standard CapSA and its enhanced variant ECapSA, alongside their island-based counterparts, iCapSA and iECapSA, on 23 standard mathematical functions are presented in Table 4.18. The outcomes reveal significant improvements when

comparing the standalone algorithms to their island-based counterparts. In comparing iCapSA to the original CapSA, it is evident that iCapSA performs better across most of the test functions (21 out of 23). The island-based approach improves solution quality and consistency in the search process, as seen in the lower average values and standard deviations. Furthermore, when the enhanced version ECapSA is combined with the island-based framework, iECapSA exhibits even greater improvements in performance. iECapSA outperforms all other variants in the mean ranks, with the lowest rank of 1.82, indicating its superior performance across the tested functions. Moreover, iECapSA typically exhibits stable and reliable performance in most scenarios, as indicated by the lowest standard deviations. This success highlights the effective collaboration between the enhancements of ECapSA and the strategic approach of the island-based model. It is worth noting that all variants, including the island-based models, have achieved the optimal solution for functions F9 and F11. This indicates the robustness of the algorithms for certain types of optimization problems, where the landscape allows for clear convergence to the global optimum.

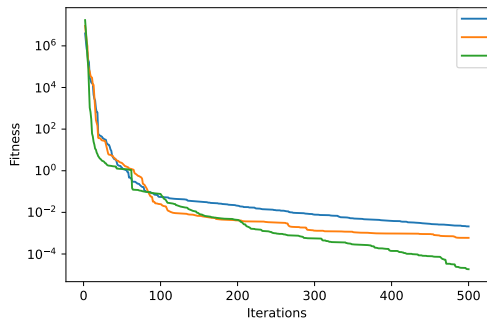
**Table 4.19:** Performance comparison of standard of Standard CapSA, ECapSA, and their island-based models iCapSA and iECapSA on the IEEE CEC2014 benchmark functions

Functions	CapSA		ECapSA		iCapSA		iECapSA	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1-CEC2014	2.258E+08	1.04E+08	1.027E+08	4.78E+07	6.572E+07	2.48E+07	<b>1.360E+07</b>	6.80E+06
F2-CEC2014	2.096E+10	8.99E+09	4.436E+09	3.66E+09	3.401E+09	1.11E+09	<b>1.981E+08</b>	1.16E+08
F3-CEC2014	6.724E+04	1.10E+04	4.529E+04	1.14E+04	3.986E+04	6.73E+03	<b>8.794E+03</b>	2.52E+03
F4-CEC2014	2.972E+03	1.37E+03	9.302E+02	2.45E+02	9.203E+02	1.51E+02	<b>5.681E+02</b>	2.29E+01
F5-CEC2014	5.207E+02	1.42E-01	5.207E+02	1.80E-01	5.206E+02	1.17E-01	<b>5.204E+02</b>	1.16E-01
F6-CEC2014	6.348E+02	1.98E+00	6.315E+02	3.57E+00	6.322E+02	2.38E+00	<b>6.250E+02</b>	2.15E+00
F7-CEC2014	8.580E+02	4.88E+01	7.582E+02	2.24E+01	7.418E+02	1.13E+01	<b>7.044E+02</b>	1.24E+00
F8-CEC2014	1.025E+03	2.86E+01	9.638E+02	2.83E+01	9.725E+02	2.07E+01	<b>9.016E+02</b>	1.95E+01
F9-CEC2014	1.174E+03	3.01E+01	1.105E+03	3.62E+01	1.138E+03	2.95E+01	<b>1.036E+03</b>	2.36E+01
F10-CEC2014	5.789E+03	1.03E+03	5.333E+03	1.04E+03	4.383E+03	6.22E+02	<b>3.548E+03</b>	5.98E+02
F11-CEC2014	7.058E+03	6.52E+02	6.323E+03	8.02E+02	6.103E+03	4.84E+02	<b>5.117E+03</b>	4.94E+02
F12-CEC2014	1.202E+03	6.02E-01	1.201E+03	4.51E-01	1.201E+03	2.28E-01	<b>1.201E+03</b>	1.95E-01
F13-CEC2014	1.304E+03	8.15E-01	1.301E+03	8.39E-01	1.300E+03	1.16E-01	<b>1.300E+03</b>	5.33E-02
F14-CEC2014	1.466E+03	2.53E+01	1.412E+03	1.33E+01	1.406E+03	4.67E+00	<b>1.400E+03</b>	3.63E-02
F15-CEC2014	9.224E+03	9.82E+03	2.440E+03	6.77E+02	1.990E+03	5.31E+02	<b>1.532E+03</b>	1.05E+01
F16-CEC2014	1.613E+03	4.01E-01	1.612E+03	6.10E-01	1.612E+03	3.33E-01	<b>1.611E+03</b>	4.25E-01
F17-CEC2014	2.087E+07	1.74E+07	1.373E+07	3.09E+07	2.442E+06	1.73E+06	<b>5.922E+05</b>	4.12E+05
F18-CEC2014	1.367E+08	3.17E+08	7.662E+07	3.02E+08	3.796E+05	6.78E+05	<b>3.386E+03</b>	1.26E+03
F19-CEC2014	2.054E+03	5.92E+01	1.978E+03	4.25E+01	1.958E+03	2.36E+01	<b>1.917E+03</b>	2.69E+00
F20-CEC2014	6.465E+04	6.28E+04	2.370E+04	1.42E+04	2.001E+04	7.07E+03	<b>3.137E+03</b>	5.48E+02
F21-CEC2014	7.614E+06	9.57E+06	1.561E+06	1.26E+06	9.363E+05	1.16E+06	<b>8.809E+04</b>	1.04E+05
F22-CEC2014	3.171E+03	3.20E+02	2.866E+03	2.33E+02	2.774E+03	1.71E+02	<b>2.478E+03</b>	1.28E+02
F23-CEC2014	2.500E+03	1.44E-08	2.500E+03	1.72E-08	2.500E+03	2.33E-11	<b>2.500E+03</b>	0.00E+00
F24-CEC2014	2.600E+03	2.28E-05	2.600E+03	3.54E-05	<b>2.600E+03</b>	4.95E-07	2.600E+03	4.72E-07
F25-CEC2014	2.700E+03	1.26E-10	2.700E+03	4.05E-10	<b>2.700E+03</b>	0.00E+00	2.700E+03	0.00E+00
F26-CEC2014	2.739E+03	4.71E+01	2.708E+03	2.53E+01	2.701E+03	1.24E+00	<b>2.700E+03</b>	7.84E-02
F27-CEC2014	2.900E+03	2.37E-09	2.900E+03	1.44E-09	2.900E+03	3.66E-12	<b>2.900E+03</b>	1.86E-12
F28-CEC2014	3.000E+03	4.80E-09	3.000E+03	1.17E-08	3.000E+03	2.25E-11	<b>3.000E+03</b>	4.62E-12
F29-CEC2014	3.100E+03	2.59E-02	3.100E+03	1.25E-02	3.100E+03	3.19E-05	<b>3.100E+03</b>	3.38E-06
F30-CEC2014	3.200E+03	3.22E-03	3.200E+03	1.22E-03	3.200E+03	6.90E-05	<b>3.200E+03</b>	3.71E-05
Mean Rank	3.63		2.75		2.22		<b>1.40</b>	

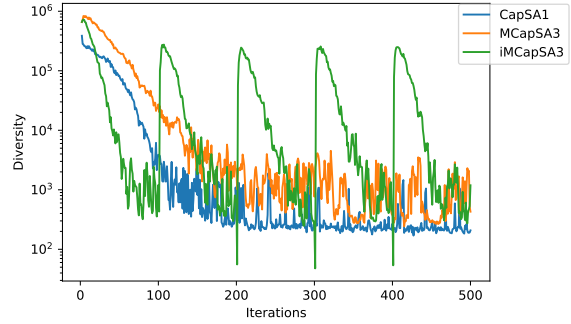
The results presented in Table 4.19 for the IEEE CEC2014 suite demonstrate a remark-

able improvement in optimization performance when incorporating island-based models. The comparison of iCapSA with CapSA indicates a clear trend of enhanced performance with iCapSA, which is further improved in the case of iECapSA when compared to ECapSA. This pattern is observed across the majority of the functions, indicating the beneficial impact of the island-based model in enhancing the algorithms' capabilities. In particular, the iECapSA variant shows the most pronounced improvements by achieving better average scores across test functions and lower standard deviations, indicating strong and consistent performance. Moreover, the average rank of 1.40 for iECapSA is much lower than other variants, strongly proving its superior optimization efficiency within this challenging set of functions. These findings highlight the effectiveness of island-based models and their significant role in improving search dynamics and performance, particularly when dealing with complex optimization challenges presented by the IEEE CEC 2014 suite.

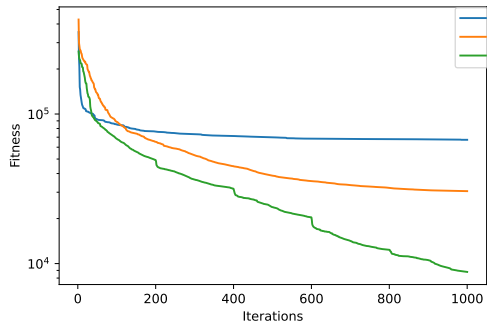
The qualitative outcomes of the original algorithm *CapSA* and its enhanced variants *ECapSA* and *iECapSA* are displayed in Figures 4.16 and 4.17. These figures depict the convergence and population diversity curves for selected test problems. According to the diversity curves illustrated in these figures, it is clear that as iterations progress, the diversity in the standard *CapSA* population gradually declines, showing smoother patterns that indicate a shift toward more exploitation-focused behavior. As illustrated by the convergence trends, this rapid decrease in diversity often leads to stagnation, a common issue in the conventional *CapSA* where it prematurely converges to local optima in the majority of test instances. On the other hand, the *iECapSA* variant exhibits highly oscillatory behavior, which highlights its better capacity to efficiently balance exploration and exploitation. These findings confirm that the migration approach in *iECapSA*, which involves exchanging information between different populations, introduces additional diversity in candidate solutions and causes more noticeable oscillations in diversity curves. In summary, it is clear that neither extremely high nor extremely low diversity constantly produces better results; rather, preserving an ideal balance of diversity is essential for achieving the best results in the majority of situations.



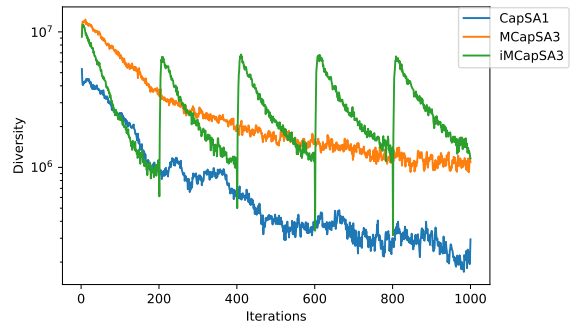
(a) F5-convergence



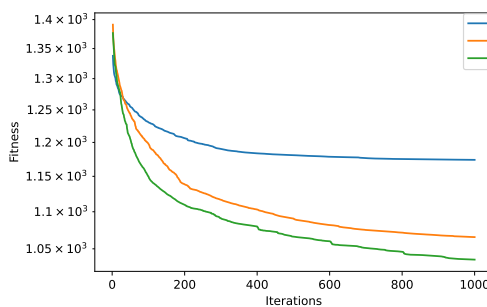
(b) F5-diversity



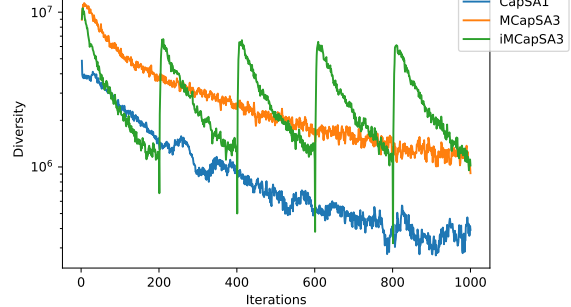
(c) F3CEC2014-convergence



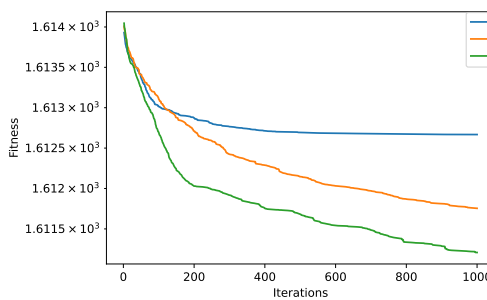
(d) F3CEC2014-diversity



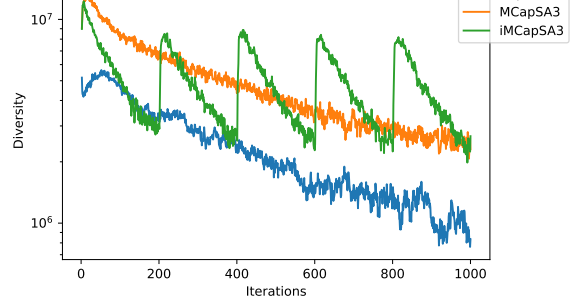
(e) F9CEC2014-convergence



(f) F9CEC2014-diversity

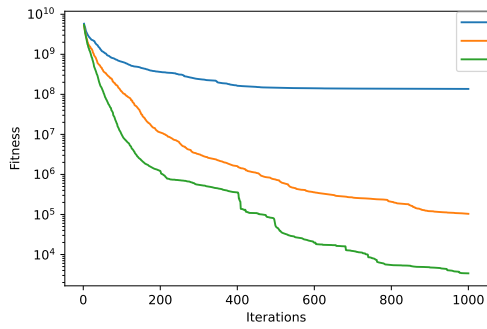


(g) F16CEC2014-convergence

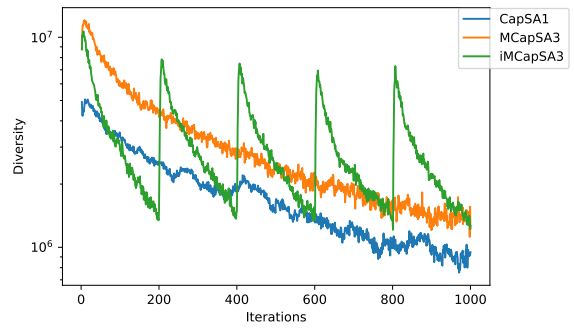


(h) F16CEC2014-diversity

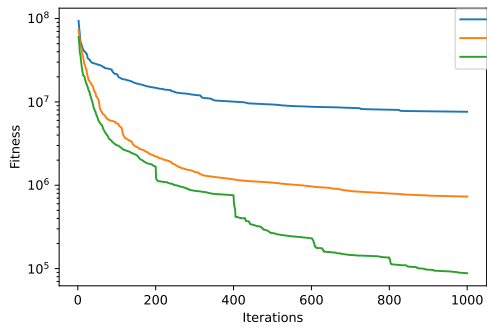
**Figure 4.16:** Comparative analysis of convergence and diversity curves for *CapSA*, *ECapSA*, and *iECapSA* on standard test function F5 and sampled IEEE CEC2014 test functions (F3, F9, and F16)



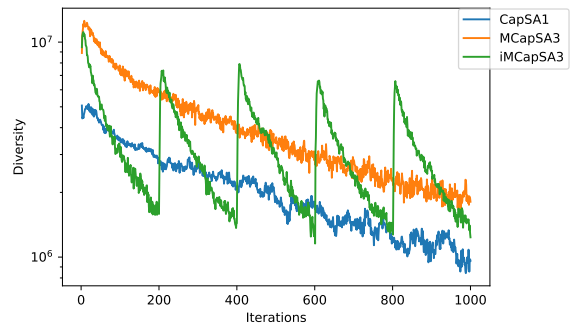
(a) F18CEC2014-convergence



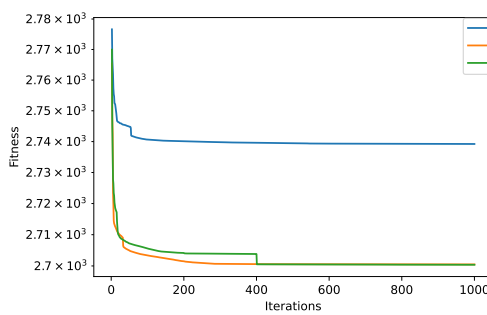
(b) F18CEC2014-diversity



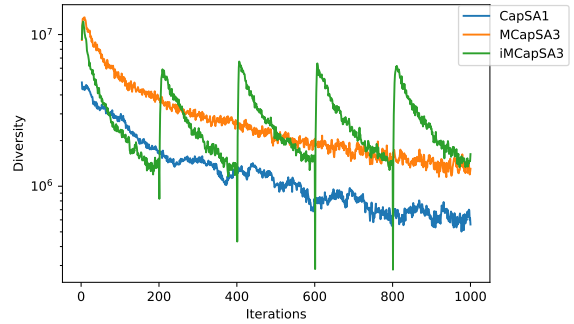
(c) F21CEC2014-convergence



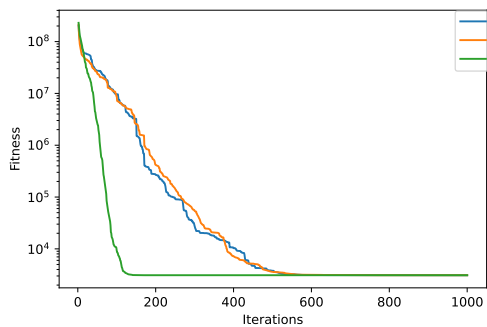
(d) F21CEC2014-diversity



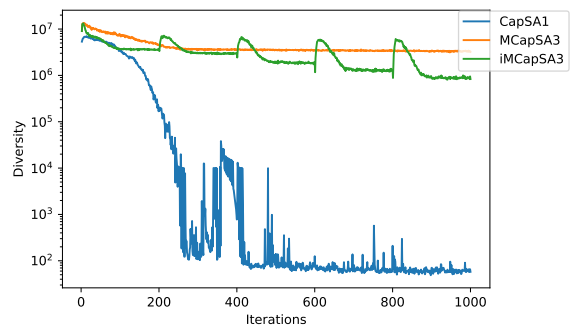
(e) F26CEC2014-convergence



(f) F26CEC2014-diversity



(g) F29CEC2014-convergence



(h) F29CEC2014-diversity

**Figure 4.17:** Comparative analysis of convergence and diversity curves for *CapSA*, *ECapSA*, and *iECapSA* on sampled IEEE CEC2014 test functions (F18, F21, F26, and F29)

## 4.5 Impact of the Adaptive Island Migration Policy

After conducting a thorough validation of the modifications made to the CSA and CapSA, and examining the benefits of integrating the island model's principles, this section shifts its focus to evaluating the impact of the newly introduced adaptive migration policy. This comparative analysis aims to uncover how an adaptive migration framework differs from its fixed counterpart in terms of improving the overall effectiveness of the search algorithms and the quality of solutions generated. To facilitate this exploration, the top-performing enhanced variants, iECSA and iECapSA, have been chosen for pairwise comparison under both migration paradigms. It is important to mention that the fixed migration paradigm was meticulously selected based on extensive tuning, as elaborated in the earlier sections.

**Table 4.20:** Comparative performance of iECSA and iECapSA with static and adaptive migration policies on 30-dimensional standard mathematical functions

Function	iECSA				iCapSA			
	static migration		Adaptive migration		static migration		Adaptive migration	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1	7.666E-08	1.41E-07	<b>7.528E-09</b>	2.46E-08	2.078E-24	8.18E-24	<b>4.260E-26</b>	1.30E-25
F2	1.076E-04	1.94E-04	<b>6.901E-06</b>	7.27E-06	9.164E-14	2.62E-13	<b>2.747E-14</b>	6.22E-14
F3	4.189E-05	8.98E-05	<b>2.802E-06</b>	9.69E-06	1.566E-22	7.58E-22	<b>6.768E-24</b>	1.89E-23
F4	4.959E-05	6.98E-05	<b>4.815E-07</b>	1.18E-06	<b>4.551E-15</b>	8.26E-15	8.239E-15	2.03E-14
F5	8.640E-06	2.95E-05	<b>1.637E-06</b>	6.77E-06	1.865E-05	2.34E-05	<b>1.208E-05</b>	1.08E-05
F6	2.923E-07	6.90E-07	<b>2.192E-08</b>	9.51E-08	2.161E-07	3.54E-07	<b>1.089E-07</b>	2.00E-07
F7	2.152E-04	1.35E-04	<b>1.959E-04</b>	1.07E-04	<b>2.613E-05</b>	1.93E-05	3.733E-05	2.91E-05
F8	-1.257E+04	2.08E-03	<b>-1.257E+04</b>	3.88E-05	-1.257E+04	8.91E-05	<b>-1.257E+04</b>	5.81E-05
F9	1.063E-07	2.90E-07	<b>4.896E-09</b>	1.43E-08	<b>0.000E+00</b>	0.00E+00	<b>0.000E+00</b>	0.00E+00
F10	4.625E-05	5.19E-05	<b>9.983E-06</b>	1.51E-05	2.354E-14	8.48E-14	<b>2.283E-14</b>	6.70E-14
F11	3.268E-07	7.42E-07	<b>3.143E-08</b>	7.15E-08	<b>0.000E+00</b>	0.00E+00	<b>0.000E+00</b>	0.00E+00
F12	1.385E-09	2.85E-09	<b>3.289E-11</b>	6.60E-11	7.848E-09	1.21E-08	<b>4.534E-09</b>	4.87E-09
F13	2.616E-08	7.58E-08	<b>7.440E-10</b>	2.45E-09	<b>9.653E-08</b>	9.85E-08	1.031E-07	1.52E-07
F14	<b>9.980E-01</b>	3.39E-16	<b>9.980E-01</b>	3.39E-16	<b>9.980E-01</b>	2.21E-15	9.980E-01	7.97E-15
F15	<b>3.075E-04</b>	3.93E-16	3.075E-04	2.72E-09	3.076E-04	7.04E-07	<b>3.076E-04</b>	1.92E-07
F16	<b>-1.032E+00</b>	6.78E-16	<b>-1.032E+00</b>	6.78E-16	-1.032E+00	6.78E-16	<b>-1.032E+00</b>	6.78E-16
F17	<b>3.979E-01</b>	1.13E-16	<b>3.979E-01</b>	1.13E-16	<b>3.979E-01</b>	9.36E-16	3.979E-01	5.83E-15
F18	<b>3.000E+00</b>	4.52E-16	<b>3.000E+00</b>	4.52E-16	<b>3.000E+00</b>	7.72E-15	3.000E+00	7.41E-15
F19	<b>-3.863E+00</b>	1.36E-15	<b>-3.863E+00</b>	1.36E-15	-3.863E+00	1.19E-10	<b>-3.863E+00</b>	6.38E-11
F20	<b>-3.322E+00</b>	2.26E-04	-3.322E+00	5.15E-04	<b>-3.318E+00</b>	2.17E-02	-3.314E+00	3.02E-02
F21	<b>-1.015E+01</b>	1.81E-15	<b>-1.015E+01</b>	1.81E-15	-1.015E+01	3.34E-06	<b>-1.015E+01</b>	2.82E-06
F22	<b>-1.040E+01</b>	9.03E-15	<b>-1.040E+01</b>	9.03E-15	<b>-1.040E+01</b>	5.64E-06	-1.040E+01	4.76E-06
F23	<b>-1.054E+01</b>	9.03E-15	<b>-1.054E+01</b>	9.03E-15	<b>-1.054E+01</b>	4.17E-06	-1.054E+01	7.41E-06
Mean Rank	1.76		<b>1.24</b>		1.57		1.43	

When analyzing the outcomes presented in Table 4.20, it becomes clear that the adaptive migration policy consistently outperforms the static policy in the context of the enhanced island-based variants, namely iECSA and iECapSA. This adaptive policy, which dynamically adjusts migration rates between 0.1 and 0.8, yields superior results compared to the static

policy's fixed rate of 0.3—a rate determined through extensive experimentation as detailed in previous sections.

The detailed results of iECSA underscore the superior performance of the adaptive migration policy, which surpasses its static counterpart in about 56.5% of the first 23 cases, specifically for functions F1 to F13. This notable majority of the benchmark suite showcases the adaptive migration policy's ability for real-time adjustments, thereby enhancing the algorithmic precision and effectiveness in a majority of tested scenarios. Moreover, in the remaining functions, iECSA with adaptive migration competes closely with, and at most times matches, the performance of the static policy. This demonstrates its broad applicability and reliability across a wide range of optimization challenges. The mean rank for iECSA improves from 1.76 under the static migration policy to 1.24 with adaptive migration. These improvements are evident across individual functions, where adaptive migration frequently achieves more favorable average values and lower standard deviations. This not only indicates better performance but also underscores greater reliability and consistency.

The improvements facilitated by the adaptive migration policy in iECapSA, although not as significant as in iECSA, are still worth noting. IECapSA with adaptive migration demonstrates superior performance in 10 instances, indicating a significant improvement in its search capabilities. The improvements highlight the importance of the adaptive policy in fine-tuning the algorithm's behavior. This approach allows for more focused and context-aware exploration and exploitation, especially in intricate optimization scenarios where fixed policies may fall short.

Table 4.21 presents the comparative results of iECSA and iECapSA under both static and adaptive migration policies in the context of 30-dimensional CEC2014 benchmark functions. It becomes clear how iECSA and iECapSA perform differently under static and adaptive migration policies. The performance of the adaptive migration policy in iECSA does not consistently surpass that of the static policy. While iECSA with adaptive migration shows superior performance in fewer cases compared to the standard benchmarks, it remains competitive, even equivalent, in most of the remaining functions. This indicates that the complexity of CEC2014 functions may necessitate a more refined approach to adjusting migration rates.



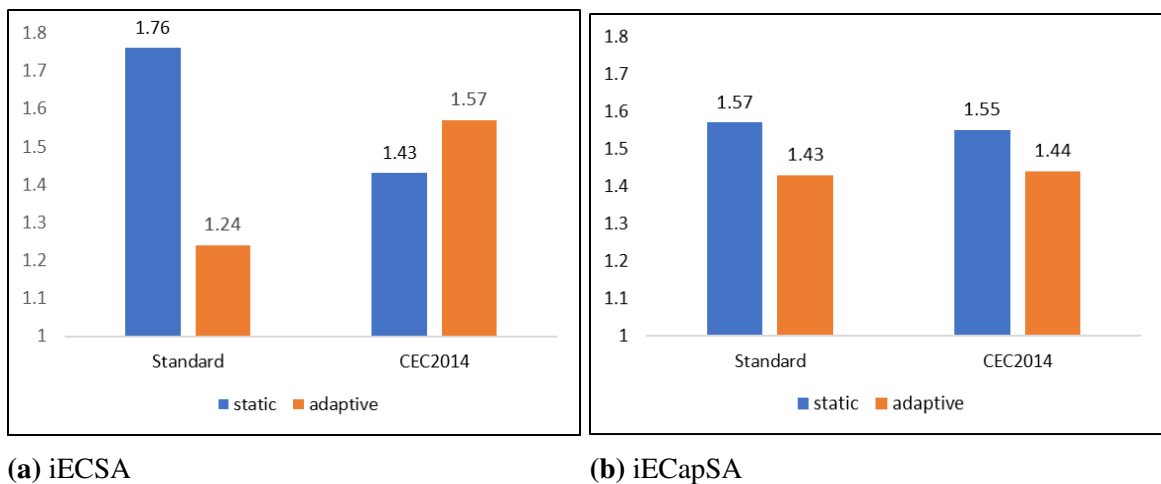
**Table 4.21:** Comparative analysis of iECSA and iECapSA performance on 30-dimensional CEC2014 benchmark functions with static and adaptive migration policies

Function	iECSA				iCapSA			
	static migration		Adaptive migration		static migration		Adaptive migration	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
F1-CEC2014	<b>5.001E+06</b>	1.98E+06	5.331E+06	2.13E+06	<b>1.360E+07</b>	6.80E+06	1.569E+07	1.16E+07
F2-CEC2014	4.071E+03	3.08E+03	<b>1.652E+03</b>	1.90E+03	<b>1.981E+08</b>	1.16E+08	2.301E+08	8.63E+07
F3-CEC2014	<b>3.825E+03</b>	1.06E+03	4.269E+03	1.45E+03	<b>8.794E+03</b>	2.52E+03	9.988E+03	2.95E+03
F4-CEC2014	4.936E+02	3.49E+01	<b>4.900E+02</b>	2.88E+01	<b>5.681E+02</b>	2.29E+01	5.824E+02	3.01E+01
F5-CEC2014	<b>5.200E+02</b>	5.62E-04	5.200E+02	4.96E-04	<b>5.204E+02</b>	1.16E-01	5.204E+02	1.24E-01
F6-CEC2014	6.272E+02	1.81E+00	<b>6.265E+02</b>	1.94E+00	6.250E+02	2.15E+00	<b>6.244E+02</b>	2.53E+00
F7-CEC2014	7.000E+02	5.53E-03	<b>7.000E+02</b>	5.08E-03	<b>7.044E+02</b>	1.24E+00	7.051E+02	1.90E+00
F8-CEC2014	<b>9.025E+02</b>	1.16E+01	9.034E+02	1.39E+01	9.016E+02	1.95E+01	<b>8.966E+02</b>	1.84E+01
F9-CEC2014	<b>1.013E+03</b>	1.35E+01	1.014E+03	1.28E+01	1.036E+03	2.36E+01	<b>1.036E+03</b>	2.10E+01
F10-CEC2014	3.877E+03	5.21E+02	<b>3.834E+03</b>	4.49E+02	3.548E+03	5.98E+02	<b>3.511E+03</b>	4.72E+02
F11-CEC2014	<b>4.122E+03</b>	3.85E+02	4.254E+03	4.49E+02	5.117E+03	4.94E+02	<b>5.053E+03</b>	4.49E+02
F12-CEC2014	1.201E+03	1.84E-01	<b>1.201E+03</b>	2.41E-01	1.201E+03	1.95E-01	<b>1.201E+03</b>	1.98E-01
F13-CEC2014	<b>1.300E+03</b>	7.29E-02	1.300E+03	6.31E-02	<b>1.300E+03</b>	5.33E-02	1.300E+03	7.85E-02
F14-CEC2014	1.400E+03	4.43E-02	<b>1.400E+03</b>	4.06E-02	1.400E+03	3.63E-02	<b>1.400E+03</b>	4.51E-02
F15-CEC2014	<b>1.511E+03</b>	3.00E+00	1.513E+03	4.21E+00	1.532E+03	1.05E+01	<b>1.529E+03</b>	1.04E+01
F16-CEC2014	1.612E+03	4.23E-01	<b>1.612E+03</b>	3.58E-01	1.611E+03	4.25E-01	<b>1.611E+03</b>	4.28E-01
F17-CEC2014	6.646E+04	5.52E+04	<b>6.593E+04</b>	4.78E+04	5.922E+05	4.12E+05	<b>4.873E+05</b>	4.54E+05
F18-CEC2014	<b>2.035E+03</b>	2.93E+01	2.048E+03	5.03E+01	<b>3.386E+03</b>	1.26E+03	4.572E+03	2.68E+03
F19-CEC2014	1.917E+03	2.08E+00	<b>1.916E+03</b>	1.95E+00	<b>1.917E+03</b>	2.69E+00	1.917E+03	2.62E+00
F20-CEC2014	<b>2.355E+03</b>	1.20E+02	2.390E+03	1.32E+02	3.137E+03	5.48E+02	<b>2.926E+03</b>	3.70E+02
F21-CEC2014	<b>2.470E+04</b>	1.73E+04	2.617E+04	1.34E+04	8.809E+04	1.04E+05	<b>8.474E+04</b>	8.74E+04
F22-CEC2014	<b>2.639E+03</b>	1.51E+02	2.650E+03	1.23E+02	2.478E+03	1.28E+02	<b>2.477E+03</b>	8.94E+01
F23-CEC2014	<b>2.617E+03</b>	5.92E-01	2.617E+03	6.92E-01	<b>2.500E+03</b>	0.00E+00	<b>2.500E+03</b>	0.00E+00
F24-CEC2014	<b>2.610E+03</b>	5.78E+00	2.615E+03	5.68E+00	2.600E+03	4.72E-07	<b>2.600E+03</b>	3.43E-07
F25-CEC2014	<b>2.702E+03</b>	3.19E+00	2.704E+03	3.21E+00	<b>2.700E+03</b>	0.00E+00	<b>2.700E+03</b>	0.00E+00
F26-CEC2014	<b>2.700E+03</b>	7.01E-02	2.700E+03	7.32E-02	<b>2.700E+03</b>	7.84E-02	2.700E+03	1.03E-01
F27-CEC2014	3.107E+03	4.41E+00	<b>3.106E+03</b>	2.65E+00	<b>2.900E+03</b>	1.86E-12	<b>2.900E+03</b>	2.55E-12
F28-CEC2014	5.794E+03	6.28E+02	<b>5.772E+03</b>	4.78E+02	3.000E+03	4.62E-12	<b>3.000E+03</b>	6.65E-12
F29-CEC2014	6.777E+03	1.92E+03	<b>6.645E+03</b>	1.95E+03	3.100E+03	3.38E-06	<b>3.100E+03</b>	2.81E-07
F30-CEC2014	<b>1.529E+04</b>	4.56E+03	1.543E+04	3.67E+03	3.200E+03	3.71E-05	<b>3.200E+03</b>	5.65E-05
Mean Rank	<b>1.43</b>		1.57		1.55		<b>1.44</b>	

The average rank is slightly lower with the static migration policy (1.43) in comparison to the adaptive migration policy (1.57). Conversely, iECapSA demonstrates a slight preference for the adaptive migration policy, which yields a slightly better mean rank (1.44) over the static policy (1.55). Charts in Figure 4.18 highlight the performance of iECSA and iECapSA using static and adaptive migration policies across standard and CEC2014 benchmark functions. It is important to mention that the static migration policy has been optimized through extensive experimentation to achieve the best-performing rate of 0.3. On the other hand, the adaptive migration policy calculates rates dynamically between 0.1 and 0.8, which allows for a customized approach to each specific problem without the need for time-consuming pre-tuning.

In summary, the results from both standard and CEC2014 benchmark functions clearly

highlight the effectiveness of the adaptive migration policy compared to the static approach. This adaptive policy, which dynamically fine-tunes the search process in real-time, is particularly beneficial in diverse and unpredictable optimization landscapes. In contrast to the static policy, which requires meticulous pre-tuning, the adaptive policy is robust and flexible. Its capacity to adapt to different problem demands makes it a practical and efficient choice. It streamlines the search process and provides acceptable outcomes without the need for the extensive time investment typically required to fine-tune migration rates. The findings across both benchmark suites underscore the importance of incorporating adaptability into the migration strategies of metaheuristic algorithms. This demonstrates how it can improve algorithmic performance and make them more applicable in real-world scenarios.



**Figure 4.18:** Mean rank comparison of iECSA and iECapSA with static and adaptive migration policies based on results in Tables 4.20 and 4.21

## 4.6 Experiments and Results of Real-World Applications

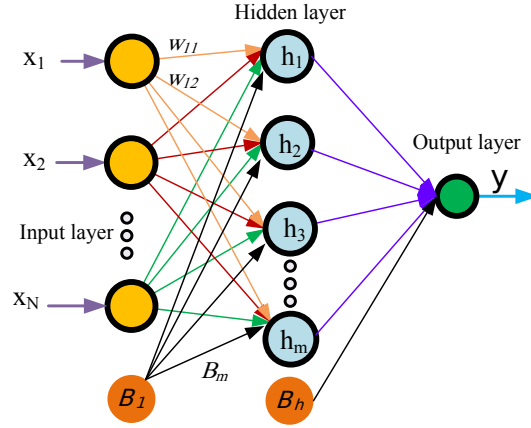
This section presents the practical applications of the iECSA and iECapSA models in real-world real-world scenarios. The efficacy of these models in addressing complex, practical problems, such as neural network training, image segmentation, and optimization of software reliability growth models will be analyzed. Initially, the performance of the models in neural network training with key metrics will be discussed in Section 4.6.1. Subsequently, Section 4.6.2 explores the application of these models in image segmentation and illustrates their impact on multilevel thresholding. Finally, in Section 4.6.3, the optimization of parameters within software reliability growth models in the domain of software engineering will be examined.

For each distinct problem, a clear problem formulation will be presented to help readers understand the challenges being addressed. This approach will make it easier for them to grasp the results and their implications. Additionally, this section will present and discuss in detail the experimental results, demonstrating the practical viability and effectiveness of the iECSA and iECapSA models in real-world applications.

### 4.6.1 Experimental Results: Neural Network Training

Feedforward Neural Network (FNN) is an artificial neural network where data flows in a unidirectional manner, moving from the input layer through the hidden layers to the output layer, without any feedback connections [238]. In this architecture, the input layer accepts the input data, the hidden layer(s) process the input data using a set of weights and activation functions, and the output layer generates the final output. Figure 4.19 provides an example of an FNN that has three input features, one hidden layer, and one output layer.

The mathematical model of the FNN relies on three primary components: input features, biases, and weights. The input layer receives a vector of features (i.e., input variables), while each neuron in the other layers performs a summation function and an activation function. The summation function calculates the weighted sum of inputs by multiplying them with their corresponding weights, adding a bias term, and applying an activation function, as



**Figure 4.19:** Simple FNN architecture with one hidden layer

shown in Eq. (4.6).

$$S_j = \sum_{i=1}^n w_{ij} \times X_i + B_j \quad j = 1, 2, \dots, m \quad (4.6)$$

where  $m$  represents the total number of hidden nodes. Meanwhile,  $n$  stands for the total number of input nodes. Each connection between the  $i$ th input node  $X_i$  and the  $j$ th hidden node has a connection weight  $w_{ij}$ , and each hidden neuron  $j$  has a bias term  $\beta_j$ .

Once the aggregation function defined in Eq. (4.6) is computed, an activation function is applied to activate the neurons' output. FNN networks can utilize several types of activation functions. In this research, the sigmoid function is used, which has been frequently applied in previous studies involving FNN networks [239, 240]. This function is used to propagate the weighted output of the hidden layer to the subsequent layer. Eq. (4.7) is used to determine the output of node  $j$  in the hidden layer.

$$h_j = \frac{1}{1 + e^{-S_j}} \quad j = 1, 2, \dots, m \quad (4.7)$$

where  $h_j$  represents the sigmoid activation function that is applied to the  $j$ th node in the middle layer, while  $S_j$  refers to the summation obtained from Eq. (4.6). Once the output for each neuron in the middle layer is computed, the next step is to determine the output of the FNN network. This can be done by applying Eq. (4.8).

$$\hat{y} = \text{sigmoid} \left( \sum_{j=1}^m w_j \times h_j + \beta \right) \quad (4.8)$$

After constructing the neural network, the weights linked with the network are adapted to approximate the desired outcomes. To accomplish this, a training algorithm is utilized to modify the weights in an iterative manner until a certain error criterion is met.

#### 4.6.1.1 Optimization Algorithms for Training FNN

During the training process of a FNN, the main objective is to assign appropriate weights and biases that can maximize the prediction performance or minimize the prediction error [240]. This process is considered as an optimization problem. Two primary families of optimization algorithms, deterministic and stochastic, can be used to train the FNN. While deterministic algorithms like backpropagation and traditional gradient descent are simple and fast, they often get stuck in local optima, known as the premature convergence problem. On the other hand, stochastic algorithms are relatively slower but have the ability to reduce the prematurity problem [241]. Recently, stochastic techniques such as metaheuristic algorithms have been widely adopted as alternatives to traditional methods for training FNN networks, and they have shown promising results in most FNN-related problems. The following subsection presents the problem formulation, including the objective function and solution representation.

#### 4.6.1.2 MHs-based Optimization of FNN: Problem Formulation

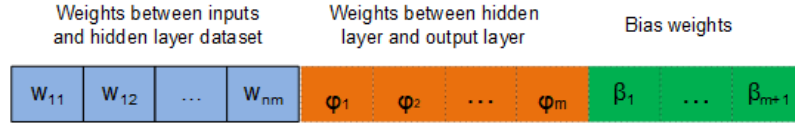
To adapt SI algorithms for optimization problems, two key steps are necessary: defining the solution representation and formulating the objective function [242]. In the realm of neural network training, the goal is to optimize the weights of a neural network to minimize the difference between the predicted output and the actual output for a given set of inputs. Particularly, in this study, the primary aim is to find the optimal values for weights and biases of a single hidden layer FNN that would yield the lowest prediction error. Consequently, the solution representation and the fitness function are formulated as follows:

- **Solution Representation:** A vector of real values was employed to encode the solution, as described by Eq. (4.9). The encoded solution is illustrated in Figure 4.20.

$$\vec{X} = \{\vec{W}, \vec{\beta}\} \quad (4.9)$$

where the weights and biases are denoted as  $\vec{W}$  and  $\vec{\beta}$  respectively, and their values are assumed to be within the range of  $[-1, 1]$  [240]. Although there is no established method in the literature for determining the optimal number of neurons in the hidden layer, in this study, the approach proposed in [239, 240] is utilized. This technique sets the number of hidden neurons ( $H$ ) as  $(2 \times F + 1)$ , where  $F$  represents the number of features in the dataset. Consequently, the dimension of each solution (i.e.,  $D$ ) is calculated as shown in Eq. (4.10).

$$D = (F \times H) + (2 \times H) + 1 \quad (4.10)$$



**Figure 4.20:** Solution representation for MHs-Based training of FNN

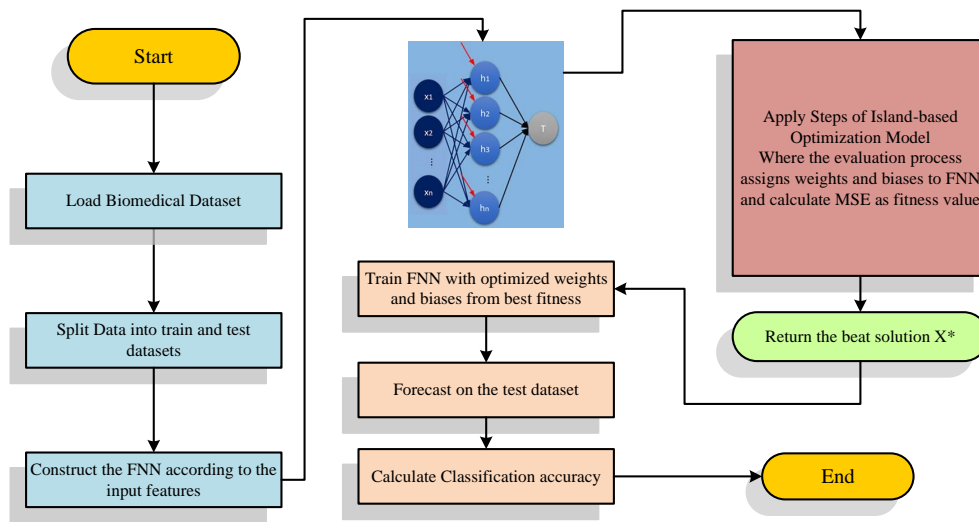
- **Objective Function:** Once the solution representation in FNN has been established, the subsequent step is to formulate an objective function to assess the quality of the produced solutions. In this study, we employed the Mean Squared Error (MSE) as our chosen metric for evaluating FNNs, as it is a widely used technique for this purpose [240]. The assessment of solutions involved inputting the generated weights and biases into the FNN and then computing the MSE, which is represented in Eq. (4.29).

$$MSE = \frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2 \quad (4.11)$$

where  $y$  and  $\hat{y}$  represent the actual and estimated values, respectively.  $n$  is the number of training samples. This formulation allows the metaheuristic algorithms to navigate the weight space in search of configurations that yield the highest accuracy or the lowest loss.

### 4.6.1.3 Optimizing FNN Using Cooperative Island-Based model

Figure 4.21 illustrates the application of the proposed optimization model to elevate the effectiveness of FNN in tackling binary classification tasks. This process incorporates a series of steps, starting with the careful selection of preprocessed biomedical datasets that have undergone thorough cleaning and normalization to serve as reliable benchmarks. To ensure a thorough and accurate evaluation of the model’s predictive abilities, the datasets are divided into two sets for training and testing.



**Figure 4.21:** Flowchart of optimizing FNN with the cooperative island-based model

In the next step, the FNN’s architecture is designed, taking into account the input attributes of the dataset (as explained in Section). Once established, the cooperative island model is activated, launching an optimization process. This process involves evaluating potential solutions, which are represented by different weight and bias configurations, through the network. fitness values are measured using the MSE to provide a clear idea of success. The optimization process continues until a specific termination condition is met, which could include reaching the maximum number of iterations. This continuous cycle ultimately leads to the discovery of the best solution, determined by the weights and biases that produce the lowest MSE. With these optimized parameters, the FNN is trained and then deployed to make predictions on the test data. To judge the success of the optimization in improving the FNN’s predictive capabilities, the classification accuracy metric is calculated as the final step

in the process.

#### 4.6.1.4 Experimental Setup

The objective of this experiment is to evaluate the performance of the iECSA and iECapSA models in optimizing the weights of FNN neural networks for biomedical classification tasks to improve the classification quality. For both iECSA and iECapSA, the population size was set to 50, with a maximum of 200 iterations (i.e., 10000 evaluations). To accommodate the stochastic nature of the metaheuristic algorithms, each model was executed 20 independent times on every dataset. Specific parameter settings for both iECSA and iECapSA were fine-tuned based on preliminary experiments to ensure optimal performance.

Experiments were conducted on seven biomedical datasets for classification. Data normalization was applied as a preprocessing step for each dataset to scale input features into a uniform range. In the experiments conducted, min-max normalization was performed, which is a linear transformation on the original data, as shown in Eq. 4.12. Here,  $X$  represents the original data,  $X_{\text{normalized}}$  denotes the normalized data, while  $X_{\text{min}}$  and  $X_{\text{max}}$  represent the minimum and maximum values of  $X$ , respectively.

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \quad (4.12)$$

All datasets are divided into 66% for training and 34% for testing. For a fair comparison, pre-split data is used across all experiments, ensuring that the same training and testing sets are used to evaluate each algorithm. This setup helps validate the models' performance in generalizing beyond the training data. Accuracy and F1-score served as the primary measures of classification performance. These metrics were chosen to provide a comprehensive view of the models' predictive capabilities, ensuring both the correctness and balance of the predictions made.

#### 4.6.1.5 Experimental Procedures

The effectiveness of the neural network training using the enhanced iECSA and iECapSA models will be comprehensively evaluated through a series of experiments. The experimental



design includes three main experiments.

- **Baseline Comparison:** The first experiment involves a comparative analysis between the basic variants of CSA and CapSA and their enhanced counterparts.
- **Comparison with Stochastic Gradient Descent Solvers:** Subsequent experiments compare the top-performing variants, iECSA and iECapSA, against traditional stochastic gradient descent solvers, including Adam and SGD (Stochastic Gradient Descent).
- **Benchmarking Against Established Metaheuristics:** The final set of experiments positions the proposed enhanced variants against a diverse set of well-established metaheuristic algorithms sourced from different categories. This comprehensive comparison is intended to validate the performance of the proposed models within a broader spectrum of optimization techniques.

#### **4.6.1.6 Tested Biomedical Datasets**

This study evaluates the effectiveness of the enhanced models in training FNN using seven real biomedical datasets sourced from the UCI machine learning repository [220]. These datasets were chosen based on their relevance to binary classification problems in biomedical informatics and machine learning. By focusing on biomedical datasets, this research emphasizes the significance of advanced algorithmic approaches in medical diagnostics. Furthermore, it contributes to the optimization field by improving predictive modeling.

The biomedical datasets cover a wide range of biomedical problems. These datasets differ in complexity, with varying numbers of features, instances, and patterns, which makes the optimization process more challenging. The diversity of these datasets ensures that the empirical analysis is thoroughly tested across different real-world scenarios. A detailed description of these datasets is presented in Table 4.22

#### **4.6.1.7 Evaluation Metrics**

In the field of optimizing neural network parameters, performance are measured using MSE as a fitness functions (see Eq.4.29) and classification quality metrics like accuracy, recall, and precision [243].

**Table 4.22:** Description of the biomedical datasets used for evaluating the training of FNN

Dataset	#Features	#samples	Hidden Layer	FNN structure
Blood	4	748	9	4-9-2
BreastCancer	8	699	17	8-17-2
Diabetes	8	768	17	8-17-2
diagnosis_II	6	120	13	6-13-2
Liver	6	345	13	6-13-2
Parkinsons	22	195	45	22-45-2
Vertebral	6	310	13	6-13-2

The classification quality is usually estimated using a special table named the confusion matrix. Table 4.23 presents the confusion matrix used to evaluate the binary classification model. In this system, instances are classified into two categories: positive or negative. The outcomes fall into one of four possible categories: True positives (TP) represent the instances correctly identified as positive. False positives (FP) are instances incorrectly identified as positive. True negatives (TN) are the instances correctly identified as negative, and false negatives (FN) are the instances incorrectly identified as negative. Various evaluation measures can be derived from the outcomes of the confusion matrix. In this study, *accuracy* and *F1-score* are key metrics for evaluating the performance of the proposed binary classification model. Accuracy, widely used in the classification of biomedical data [239, 160], measures the overall correctness of the model, while the F1-score provides a balance between precision and recall, particularly valuable when the cost of false positives and false negatives is high or when there is an imbalance in class distribution. Eq. 4.13 provides the accuracy formula, and Eq. 4.16 provides the formula for the F1-score.

**Table 4.23:** Confusion matrix for binary classification.

	Predicted positive	Predicted negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.13)$$

$$Precision = \frac{TP}{TP + FP} \quad (4.14)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.15)$$

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.16)$$

#### 4.6.1.8 Comparative Analysis of Basic and Enhanced Variants of CSA and CapSA

A direct comparison of the MSE for the original and improved CSA and CapSA models is presented in Table 4.24. The iECapSA model exhibits a clear advantage by achieving the lowest MSE values on the majority of datasets (6 out of 7). This is quantitatively evidenced by its leading mean rank of 1.14, which signifies its outstanding performance in minimizing error rates. Conversely, the original CSA and CapSA exhibit higher MSEs. Notably, the standard CSA and ECapSA models exhibit a competitive performance. However, the ECapSA model shows notable improvement over CapSA, especially on datasets like 'diagnosis\_II' and 'Vertebral'.

**Table 4.24:** MSE results for CSA and CapSA variants across biomedical datasets

Data	Measure	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
Blood	AVG	1.555E-01	1.574E-01	1.585E-01	1.551E-01	1.560E-01	<b>1.541E-01</b>
	STD	5.942E-04	1.558E-03	2.886E-03	8.996E-04	8.075E-04	9.570E-04
BreastCancer	AVG	3.977E-02	4.324E-02	4.987E-02	4.106E-02	4.123E-02	<b>3.897E-02</b>
	STD	5.476E-04	2.141E-03	7.012E-03	1.389E-03	8.330E-04	1.271E-03
Diabetes	AVG	1.622E-01	1.857E-01	1.851E-01	1.600E-01	1.765E-01	<b>1.563E-01</b>
	STD	2.720E-03	1.055E-02	1.102E-02	3.916E-03	6.044E-03	1.756E-03
diagnosis_II	AVG	8.396E-03	3.685E-02	4.531E-02	6.551E-03	1.855E-02	<b>4.511E-03</b>
	STD	3.313E-03	1.396E-02	2.596E-02	3.497E-03	5.274E-03	2.021E-03
Liver	AVG	2.144E-01	2.271E-01	2.264E-01	2.049E-01	2.225E-01	<b>2.008E-01</b>
	STD	3.341E-03	4.923E-03	7.042E-03	4.131E-03	3.422E-03	3.746E-03
Parkinsons	AVG	<b>1.086E-01</b>	1.225E-01	1.345E-01	1.151E-01	1.144E-01	1.122E-01
	STD	2.879E-03	6.269E-03	8.827E-03	6.036E-03	3.156E-03	2.275E-03
Vertebral	AVG	1.505E-01	1.608E-01	1.699E-01	1.419E-01	1.561E-01	<b>1.372E-01</b>
	STD	2.945E-03	5.269E-03	1.149E-02	5.167E-03	4.572E-03	3.628E-03
Mean Rank		2.57	5.29	5.71	2.43	3.86	<b>1.14</b>

In contrast to what was expected, the fundamental CSA model surpasses its enhanced counterpart, ECSA. This indicates that for certain datasets, the original algorithm's mechanisms might be more effective. Conversely, the ECapSA model produces superior MSE outcomes compared to the original CapSA model, indicating that the improvements incorporated into CapSA are beneficial. The overall performance ranking from best to least is

as follows: iECapSA, ECapSA, CSA, iECSA, ECSA, and CapSA. This ranking reflects the effectiveness of the iterative and adaptive improvements in iECapSA.

The convergence curves in Figure 4.22 depict the decrease in MSE of each algorithm over iterations and highlight their learning and optimization process. Overall, the convergence curves reinforce the quantitative findings previously discussed in Table 4.24. iECapSA algorithm exhibits the fastest convergence speed and has the lowest error rates for most datasets. The convergence behavior of iECapSA indicates that its enhancements successfully expedite the search towards optimal solutions.

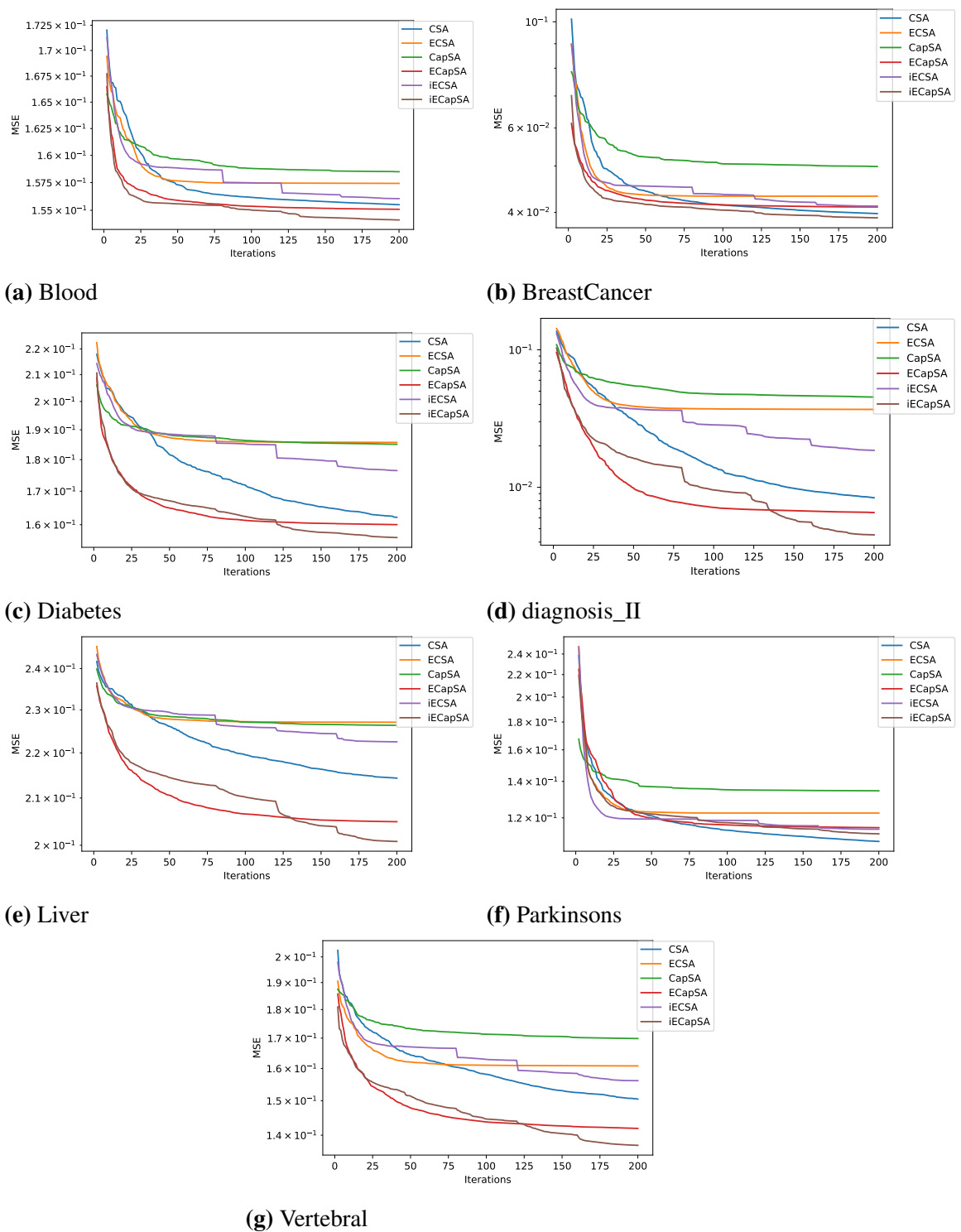
**Table 4.25:** Testing classification accuracy for CSA and CapSA variants across biomedical datasets

Dataset	Measure	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
Blood	AVG	0.7512	0.7561	<b>0.7563</b>	0.7500	0.7527	0.7539
	STD	0.0039	0.0064	0.0034	0.0044	0.0037	0.0065
BreastCancer	AVG	0.9725	0.9721	0.9689	0.9710	<b>0.9727</b>	0.9721
	STD	0.0029	0.0046	0.0090	0.0053	0.0048	0.0044
Diabetes	AVG	0.7263	0.6777	0.6966	0.7363	0.6990	<b>0.7462</b>
	STD	0.0143	0.0425	0.0291	0.0182	0.0242	0.0116
diagnosis_II	AVG	<b>1.0000</b>	0.9866	0.9573	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	STD	0.0000	0.0407	0.0826	0.0000	0.0000	0.0000
Liver	AVG	0.7000	0.6301	0.6339	0.7318	0.6614	<b>0.7326</b>
	STD	0.0413	0.0390	0.0446	0.0304	0.0302	0.0237
Parkinsons	AVG	<b>0.8485</b>	0.8194	0.8037	0.8351	0.8366	0.8418
	STD	0.0147	0.0205	0.0288	0.0272	0.0272	0.0267
Vertebral	AVG	0.8198	0.7910	0.7519	0.8401	0.8052	<b>0.8571</b>
	STD	0.0181	0.0340	0.0477	0.0273	0.0356	0.0318
Mean Rank		2.79	4.64	5.00	3.36	3.21	<b>2.00</b>

**Table 4.26:** Testing classification F1-score for CSA and CapSA variants across biomedical datasets

Data	Measure	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
Blood	AVG	0.8565	0.8599	<b>0.8600</b>	0.8554	0.8576	0.8574
	STD	0.0026	0.0042	0.0024	0.0026	0.0025	0.0034
BreastCancer	AVG	0.9790	0.9787	0.9763	0.9778	<b>0.9792</b>	0.9787
	STD	0.0022	0.0035	0.0069	0.0041	0.0037	0.0034
Diabetes	AVG	0.5469	0.4256	0.4633	0.5646	0.4861	<b>0.5866</b>
	STD	0.0263	0.0942	0.0609	0.0290	0.0463	0.0203
diagnosis_II	AVG	<b>1.0000</b>	0.9897	0.9584	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	STD	0.0000	0.0307	0.0845	0.0000	0.0000	0.0000
Liver	AVG	0.6053	0.4079	0.4341	0.6626	0.5044	<b>0.6674</b>
	STD	0.0587	0.1324	0.1375	0.0433	0.0792	0.0324
Parkinsons	AVG	<b>0.9083</b>	0.8906	0.8819	0.8997	0.9004	0.9038
	STD	0.0087	0.0126	0.0174	0.0164	0.0167	0.0159
Vertebral	AVG	0.8770	0.8578	0.8323	0.8894	0.8666	<b>0.9001</b>
	STD	0.0135	0.0247	0.0302	0.0191	0.0261	0.0226
Mean Rank		2.79	4.64	5.00	3.36	3.07	<b>2.14</b>

The testing classification results for various biomedical datasets are reported in Tables



**Figure 4.22:** MSE convergence curves for basic and enhanced CSA and CapSA models across various biomedical datasets

4.25 and 4.26. It is clear that the proposed iECapSA algorithm surpasses other algorithms in three datasets (Diabetes, Liver, and Vertebral). In most datasets, such as 'Blood', 'Breast-Cancer', and 'Parkinsons', iECapSA has shown competitive or superior accuracy and F1-score. Remarkably, for the dataset 'diagnosis\_II', all model variants have achieved perfect or near-perfect accuracy and F1 scores, indicating that these datasets might be less challenging or that the models are well-suited to their structure. Moreover, CSA again performs well, ranking second overall, which suggests that the foundational strategies of the algorithm are still effective for these tasks. Specifically, the enhanced variants, iECSA and ECSA, do not consistently outperform their basic counterpart, emphasizing the importance of problem-specific tuning. Nevertheless, the improvements in CapSA have yielded positive results, as ECapSA has outperformed the original CapSA in terms of performance rankings.

In summary, the findings collectively demonstrate that the iECapSA model is reliable and efficient for classifying biomedical data. It has consistent and robust optimization capabilities. However, the varying outcomes across different algorithm designs and datasets highlight the challenge of achieving consistent improvements. While enhancements in ECSA do not always result in better performance, strategic improvements, as seen in iECapSA, can effectively reduce prediction errors and enhance the model's accuracy.

#### 4.6.1.9 Comparison with Stochastic Gradient Descent Solvers

**Table 4.27:** Classification performance comparison between stochastic gradient descent solvers and metaheuristic models across biomedical datasets

		Accuracy				F1-score				
Dataset	Measure	Stochastic Gradient Descent		MHs		Measure	Stochastic Gradient Descent		MHs	
		Adam	SGD	CSA	iECapSA		Adam	SGD	CSA	iECapSA
Blood	AVG	0.7596	<b>0.7608</b>	0.7512	0.7539	AVG	0.8627	<b>0.8641</b>	0.8565	0.8574
	STD	0.0026	0.0000	0.0039	0.0065	STD	0.0021	0.0001	0.0026	0.0034
BreastCancer	AVG	0.9649	0.9429	<b>0.9725</b>	0.9721	AVG	0.9737	0.9582	<b>0.9790</b>	0.9787
	STD	0.0044	0.0116	0.0029	0.0044	STD	0.0032	0.0081	0.0022	0.0034
Diabetes	AVG	0.6947	0.6357	0.7263	<b>0.7462</b>	AVG	0.5178	N/A	0.5469	<b>0.5866</b>
	STD	0.0245	0.0071	0.0143	0.0116	STD	0.0596	N/A	0.0263	0.0203
diagnosis_II	AVG	0.8427	0.7183	<b>1.0000</b>	<b>1.0000</b>	AVG	0.8347	0.7481	<b>1.0000</b>	<b>1.0000</b>
	STD	0.1289	0.1729	0.0000	0.0000	STD	0.1492	0.1532	0.0000	0.0000
Liver	AVG	0.6055	0.5669	0.7000	<b>0.7326</b>	AVG	N/A	N/A	0.6053	<b>0.6674</b>
	STD	0.0342	0.0148	0.0413	0.0237	STD	N/A	N/A	0.0587	0.0324
Parkinsons	AVG	0.8216	0.7612	<b>0.8485</b>	0.8418	AVG	0.8917	0.8644	<b>0.9083</b>	0.9038
	STD	0.0197	0.0000	0.0147	0.0267	STD	0.0115	0.0000	0.0087	0.0159
Vertebral	AVG	0.5448	0.4481	0.8198	<b>0.8571</b>	AVG	N/A	N/A	0.8770	<b>0.9001</b>
	STD	0.1814	0.1830	0.0181	0.0318	STD	N/A	N/A	0.0135	0.0226

Table 4.27 compares the performance of traditional gradient descent solvers, Adam and

SGD, with the metaheuristic models CSA and iECapSA across biomedical datasets in terms of accuracy and F1-score. For certain datasets, the F1-score is reported as 'N/A' (not applicable) for Adam and SGD, typically due to zero precision or recall in the classification outcomes. This indicates a failure to correctly identify any positive cases. Consequently, this problem confirms the power of metaheuristic models, because they consistently provide valid F1 scores across all datasets.

Overall, metaheuristic models, especially iECapSA, often achieve comparable or superior classification outcomes compared to traditional gradient descent methods. These models are good at finding the best weights and biases that reduce errors in classification and thus produce better classification quality. Metaheuristic models can fine-tune the parameters of machine learning models, making them useful optimization tools in solving complex classification tasks.

#### **4.6.1.10 Performance Validation Against State-of-the-Art algorithms**

In this section, we evaluate the performance of iECSA and iECapSA against a diverse set of 11 widely recognized MHs, namely BAT, DE, JAYA, PSO, SSA, WOA, SCA, GWO, HHO, AOA, and AO. Table 4.28 presents a comparison of MSE values for these algorithms. Notably, GWO and PSO generally perform well, often recording the lowest MSE values in datasets like "BreastCancer," "Diabetes," and "Vertebral". Their effectiveness is underscored by their low mean ranks of 2.29 and 2.57, respectively. This indicates that these algorithms have superior performance in optimizing neural networks across the evaluated datasets. Meanwhile, iECapSA also show competitive performance, particularly in the "BreastCancer" and "Vertebral" datasets. Overall, each algorithm exhibits varying levels of effectiveness across different datasets. The algorithms GWO, PSO, BAT, SSA, and iCapSA are recognized for their exceptional performance, as seen by their top positions in the mean rank index.

The average accuracy results for 20 runs of the compared algorithms are presented in Table 4.29. Based on the results, iECapSA demonstrates superior performance compared to DE, JAYA, WOA, SCA, HHO, AOA, AO, and iECSA algorithms across most datasets.

**Table 4.28:** Average MSE results of the proposed iECSA and iECapSA against state-of-the-art Algorithms for neural network optimization

Dataset	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AOA	AO	iCapSA	iECSA
Blood	1.541E-01	1.571E-01	1.561E-01	<b>1.535E-01</b>	1.545E-01	1.573E-01	1.573E-01	1.545E-01	1.558E-01	1.697E-01	1.648E-01	1.541E-01	1.560E-01
BreastCancer	3.587E-02	5.011E-02	4.911E-02	3.584E-02	3.675E-02	4.652E-02	4.786E-02	<b>3.461E-02</b>	4.034E-02	7.703E-02	4.347E-02	3.897E-02	4.123E-02
Diabetes	1.535E-01	1.689E-01	1.672E-01	1.532E-01	1.531E-01	1.708E-01	1.750E-01	<b>1.531E-01</b>	1.593E-01	2.155E-01	2.000E-01	1.563E-01	1.765E-01
diagnosis_II	<b>1.696E-03</b>	2.730E-02	2.310E-02	3.274E-03	3.452E-03	3.410E-02	3.139E-02	2.862E-03	7.263E-03	1.066E-01	9.055E-02	4.511E-03	1.855E-02
Liver	2.023E-01	2.138E-01	2.106E-01	<b>1.916E-01</b>	2.010E-01	2.204E-01	2.184E-01	2.008E-01	2.126E-01	2.389E-01	2.359E-01	2.008E-01	2.225E-01
Parkinsons	9.281E-02	2.100E-01	2.070E-01	1.468E-01	1.026E-01	1.554E-01	1.675E-01	<b>9.085E-02</b>	1.095E-01	1.520E-01	1.612E-01	1.122E-01	1.144E-01
Vertebral	1.357E-01	1.501E-01	1.436E-01	<b>1.299E-01</b>	1.351E-01	1.544E-01	1.533E-01	1.357E-01	1.442E-01	1.893E-01	1.836E-01	1.372E-01	1.561E-01
Mean Rank	3.00	9.57	8.29	2.57	3.29	9.86	9.86	2.29	6.00	12.29	11.14	4.29	8.57

In addition, iECapSA demonstrates a perfect accuracy of 1.00 for Diagnosis-II, matching the results of BAT, JAYA, PSO, SSA, GWO, HHO, and iECSA. To summarize, iECapSA achieved a fourth-place ranking, demonstrating its competitive performance relative to the top algorithms.

**Table 4.29:** Accuracy results of the proposed iECSA and iECapSA against state-of-the-art Algorithms for neural network optimization

Dataset	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AOA	AO	iCapSA	iECSA
Blood	0.7529	0.7549	0.7551	0.7575	0.7502	0.7559	0.7512	0.7500	0.7529	<b>0.7594</b>	0.7578	0.7539	0.7527
BreastCancer	0.9761	0.9603	0.9599	0.9714	0.9725	0.9681	0.9624	0.9779	0.9727	0.9139	<b>0.9786</b>	0.9721	0.9727
Diabetes	0.7469	0.7275	0.7349	0.7506	<b>0.7525</b>	0.7135	0.7160	0.7504	0.7368	0.6441	0.6729	0.7462	0.6990
diagnosis_II	<b>1.0000</b>	0.9963	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>	0.9817	0.9829	<b>1.0000</b>	<b>1.0000</b>	0.7732	0.9073	<b>1.0000</b>	1.0000
Liver	0.7424	0.6911	0.6953	0.7508	<b>0.7538</b>	0.6547	0.6648	0.7534	0.7042	0.5780	0.5881	0.7326	0.6614
Parkinsons	0.8590	0.7224	0.7276	0.7843	0.8515	0.7940	0.7612	<b>0.8716</b>	0.8463	0.7784	0.7739	0.8418	0.8366
Vertebral	0.8627	0.8104	0.8283	0.8656	0.8679	0.8052	0.8127	<b>0.8698</b>	0.8387	0.7208	0.7458	0.8571	0.8052
Mean Rank	4.29	9.14	7.79	4.50	4.21	8.93	9.71	3.79	5.64	10.71	8.71	5.50	8.07

## 4.6.2 Experimental Results: Multilevel Image Segmentation

Image segmentation is the process of partitioning an image into multiple image segments, also known as image regions or objects, based on various criteria such as gray level values, color, shape, or textures [244, 245]. More precisely, image segmentation is assigning a label to each pixel in an image so that pixels with the same label share specific properties. In computer vision applications such as medical imaging, geographical imaging, autonomous recognition, robotic vision, and many others, image segmentation is considered the essential process for analyzing and interpreting the captured image [244, 246, 247]. For instance, medical imaging approaches are vital in diagnosing complex diseases and in the patient's healthcare. They help doctors diagnose, treat, and detect life-threatening diseases early. Much detailed information can be extracted from chest CT images; however, manual processing of these images is inaccurate. Therefore, image segmentation has become one of the



critical stages for processing medical images, and it has been widely employed in several medical applications [7].

Most image segmentation approaches are based on similarity and discontinuity, two critical features of intensity values. The similarity technique, which relies on similarity among image objects with pre-determined criteria for partitioning, is widely employed [244]. Among the various similarity-based techniques, the thresholding technique stands out as the most favorable choice due to its accuracy, simplicity, and robustness [248]. A threshold-based segmentation technique subdivides an image into smaller segments by determining their boundaries using at least one gray-level value. Hence, Multi-level thresholding (MLT) will find multiple gray-level threshold values to differentiate the objects of interest from the image's background.

Two widely regarded image thresholding methods are Otsu's approach, which utilizes the concept of between-class variance [249], and Kapur's approach, which is based on the principle of entropy [250]. These methods are considered among the top choices for image thresholding techniques. They are used to identify the best thresholds for dividing the region of gray-level values in an image, depending on some pre-defined criteria. However, both exploit an exhaustive search algorithmic paradigm to optimize the objective function, and thus the computational time grows exponentially with the number of threshold values. For this reason, they can not be used to resolve MLT problems [244]. Meta-heuristic algorithms have been leveraged in MLT, which is considered an NP-hard problem [251, 252].

#### **4.6.2.1 Problem Formulation**

The mathematical foundation of MLT is crucial for segmenting grayscale images into distinct classes. For an image  $I$ , the number of classes is  $k + 1$ , where  $k$  thresholds are needed to divide the image into multiple, non-overlapping segments. This problem can be mathematically formulated as in Eq. (4.17).

$$\begin{aligned}
C_0 &= \{I(x,y) \in X \mid 0 \leq I(x,y) \leq t_1 - 1\} \\
C_1 &= \{I(x,y) \in X \mid t_1 \leq I(x,y) \leq t_2 - 1\} \\
C_i &= \{I(x,y) \in X \mid t_i \leq I(x,y) \leq t_{i+1} - 1\} \\
C_m &= \{I(x,y) \in X \mid t_k \leq I(x,y) \leq N - 1\}
\end{aligned} \tag{4.17}$$

where  $C_i$  is the  $i^{th}$  class of the image  $I$ ,  $t_i$  denotes the threshold value for  $i \in 1, 2, 3, \dots$ ,  $k$ ,  $k$  is the total number of distinct threshold values,  $L$  the maximum intensity value, and  $X$  represents the set of all pixel coordinates  $(x,y)$  within the grayscale image  $I$  such that each pixel at location  $(x,y)$  in the image  $I$  has an intensity value.

To effectively handle the challenge of MLT in image segmentation using metaheuristic techniques, it is essential to redefine MLT as an optimization problem. This first step involves formulating the solution representation and the objective function. Ultimately, the main objective of this optimization task is to identify a set of  $k$  optimal threshold values that achieve the desired segmentation criteria.

- **Solution Representation:** To solve the MLT problem, we represent the solution with a vector of integer values, which serves as the threshold levels for segmenting the grayscale image. In particular, the solution vector is defined as:

$$\vec{T} = [t_1, t_2, \dots, t_k] \tag{4.18}$$

where each  $t_i$  is a threshold value within the range of  $[0, L-1]$ , and  $k$  is the total number of thresholds needed to segment the image into  $k + 1$  classes.

- **Objective Function (Kapur's Entropy):** Kapur's Entropy method is a well-established technique for image segmentation. It is particularly effective in determining the ideal threshold values for segmenting an image into multiple levels [253]. This technique, introduced by Kapur [253], leverages the concept of entropy from information theory to generate a statistical variety score based on the intensity levels of an input image. Entropy-based thresholding is reliable because the thresholds that are chosen do not depend on small changes in the image's pixel intensities. Instead, they are based on

the overall and objective quality of the image histogram [250]. The cross-entropy-based objective function has recently shown promising results for the segmentation tasks in different domains [254, 255, 256]. So, the entropy-based thresholding method is chosen to solve the COVID-19 CT image segmentation problem in this thesis. It is therefore used as an objective function to assess the fitness of the generated solutions (i.e., threshold values) in the proposed cooperative optimization model. The mathematical representation of this method is outlined as follows:

Given an image histogram  $h(i)$  for intensity levels  $i = 0, 1, \dots, L - 1$ .  $h(i)$  represents the observed grey-level frequencies. The probability of each intensity level  $i$  is given by:

$$p(i) = \frac{h(i)}{N} \quad (4.19)$$

where  $N$  is the total number of pixels in the image and is defined as:

$$N = \sum_{i=0}^{L-1} h(i) \quad (4.20)$$

For multilevel segmentation, we seek to find  $k$  thresholds  $t_1, t_2, \dots, t_k$  that divide the histogram into  $k + 1$  segments. The probability of each segment  $\omega_j$  defined by the thresholds can be calculated as follows:

$$\omega_j = \sum_{i=t_{j-1}+1}^{t_j} p(i) \quad (4.21)$$

where  $j = 1, 2, \dots, k + 1$ ,  $t_0 = -1$  and  $t_{k+1} = L - 1$ .

The entropy of each segment is calculated using the probabilities within that segment:

$$H_j = - \sum_{i=t_{j-1}+1}^{t_j} \frac{p(i)}{\omega_j} \ln \left( \frac{p(i)}{\omega_j} \right) \quad (4.22)$$

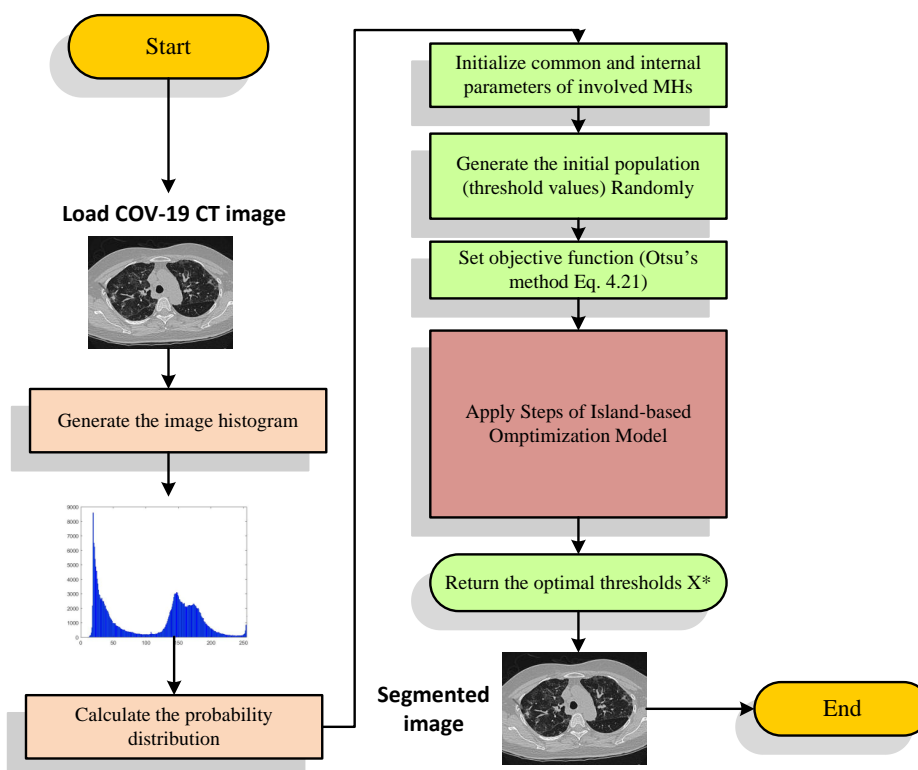
The total entropy for the given thresholds is the sum of the entropies of all segments:

$$H(T) \uparrow = \sum_{j=1}^{k+1} H_j \quad (4.23)$$

Accordingly, the formula given in Eq. (4.23) is employed as an objective function.

#### 4.6.2.2 Cooperative Island-based Optimization Model for MLT Segmentation

Figure 4.23 demonstrates the optimization flowchart, providing a clear outline of the exhaustive steps involved in successfully implementing the cooperative island-based optimization model for MLT image segmentation. This tailored approach is specifically designed for CT scan COVID-19 images and is thoroughly explained in the following steps, according to the flowchart.



**Figure 4.23:** Flowchart of the multilevel image segmentation framework using cooperative island-based optimization model

1. **Loading the CT scan image:** The first step involves loading the target CT scan COVID-19 image, which serves as the foundation for the subsequent segmentation steps.
2. **Calculating the histogram:** After loading the image, the histogram is calculated. This step is essential for gaining insight into the distribution of pixel intensities within the

image, serving as a valuable guide during the segmentation process.

3. **Calculating probability distribution:** Once the histogram has been calculated, the next step is to determine the probability distribution of the pixel intensities. This is crucial for Otsu's objective function, which uses probability distributions to determine the most effective thresholding values.
4. **Initializing parameters:** Before beginning the optimization process, multiple parameters must be set. These include standard parameters such as the number of iterations, population size  $N$ , the dimension of the problem (number of thresholds), and migration parameters. Additionally, internal parameters unique to cooperative metaheuristic algorithms, such as ECSA, are also established.
5. **Generation of the initial population:** Once the parameters are established, a random initial population is created. Each member of this population represents a vector of threshold values, which are viable candidates for image segmentation.
6. **Optimization Process:** Utilizing a collaborative island-based approach, the optimization process is initiated (as depicted in Section 3.1.5). This phase involves fine-tuning of threshold values through iterative refinement, resulting in improved segmentation performance.
7. **Implementation of optimal thresholds:** After completing the optimization process, the best threshold values are determined and implemented on the original image. This final action produces the segmented image, achieving the desired outcome.

#### 4.6.2.3 Experimental Setup

The experimental configuration for segmenting COVID-19 CT scan images was carefully developed to address the challenges inherent in real-world medical imaging. To assess segmentation quality and algorithm performance effectively, we implemented multiple thresholds: 4, 6, 8, and 10. These thresholds, recommended by prior studies, were selected to manage the complexity and variability of the scans while preventing under-segmentation and over-segmentation issues.

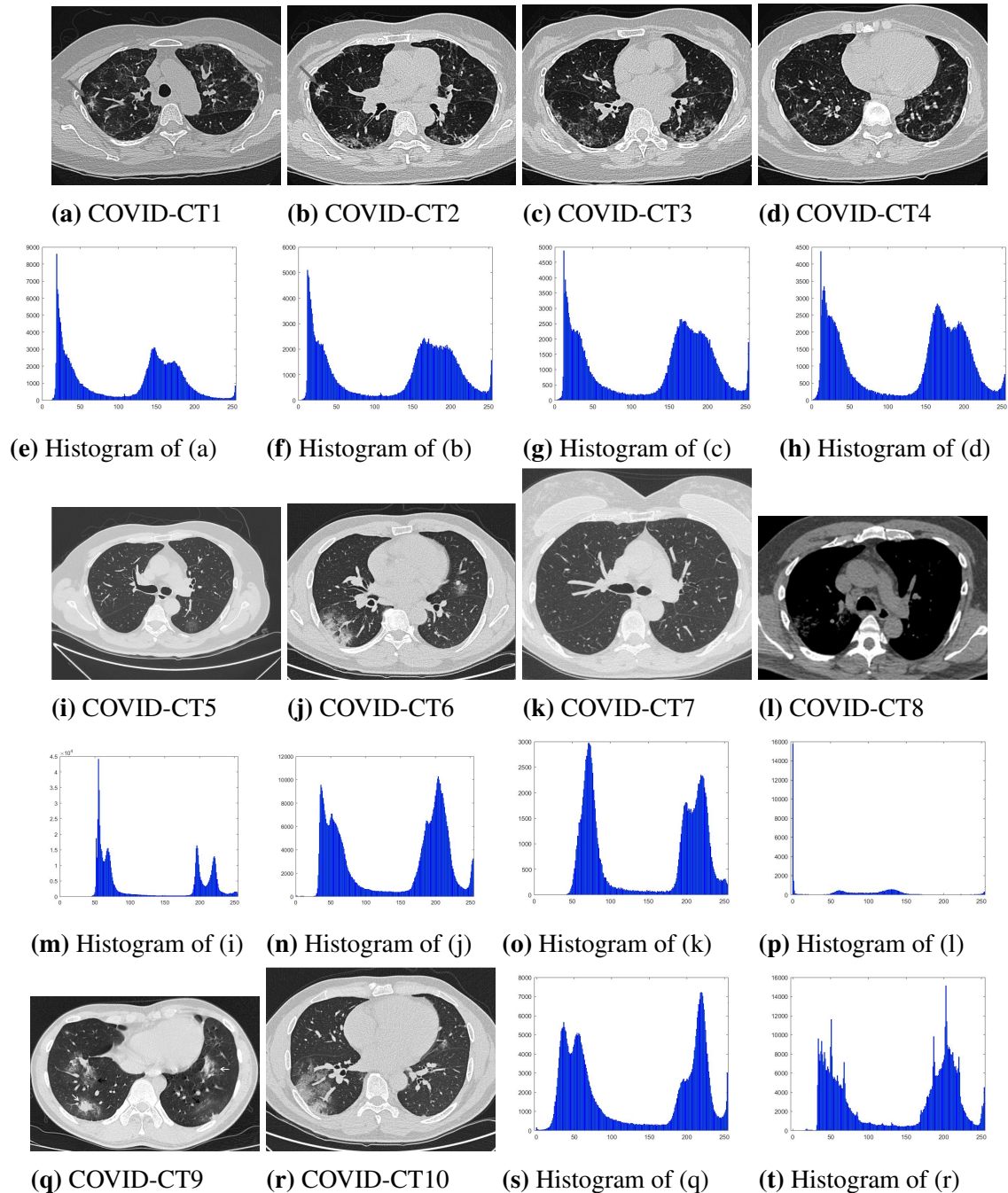
To guarantee fair and objective algorithm comparisons, all experiments were conducted under standardized conditions. Common parameters, including a population size of 30 and a maximum of 500 iterations (total of 15,000 evaluations), were uniformly applied across all optimizers for each image. This standardization enables a consistent and fair evaluation platform to analyze the effectiveness of each optimizer. The internal parameters for each algorithm were carefully chosen based on default values suggested in the literature, ensuring optimal operating conditions for a fair comparison. The search space for the segmentation problem was clearly defined with lower and upper bounds set at 1 and 256, respectively, to cover the full range of pixel intensity values typically observed in CT images.

Given the stochastic nature of the algorithms under review, each was tested 20 times on each image at every threshold level. Statistical tests, including the Wilcoxon rank-sum and Friedman mean rank tests, were employed to validate the results and confirm the superiority of the proposed methods. The outcomes reported reflect the average of all runs, with the best results highlighted in boldface.

#### **4.6.2.4 COVID-19 CT images dataset**

Computed tomography (CT) has emerged as a valuable tool for diagnosing Coronavirus Disease 2019 (COVID-19) patients, particularly during the COVID-19 outbreak. To evaluate the proposed methodology in this study, open-sourced datasets were employed from these references: [221, 222]. By utilizing this dataset, the study aims to test and prove the effectiveness of advanced MLT image segmentation techniques to enhance the diagnosis of COVID-19, thus improving medical diagnostics, especially during a global health crisis [257]. The CT COVID-19 dataset consists of 349 CT scans with clinical COVID-19 findings from 216 patients. An experienced radiologist who has been identifying and treating COVID-19 patients ever since the pandemic's emergence attests to the usefulness of this dataset. The COVID-19 images were collected from individuals of both sexes, aged 40 to 84. In this paper, ten images for ten distinct patients are selected from this dataset to assess the performance of the proposed methods. The chosen test images are named COVID-CT1, COVID-CT2, ..., and COVID-CT10. Figure 4.24 demonstrates these images and the corre-

sponding histograms. The histograms, which represent the visualization of the pixel intensity distribution across the images, provide further information about the textural and structural features of the COVID-19-affected lung areas.



**Figure 4.24:** Set of tested COVID-19 CT images and their histograms

#### 4.6.2.5 Image Segmentation Evaluation Metrics

For MLT segmentation, it is critical to examine the pixel classification's accuracy by using a quantitative measure of the quality of a segmented image. In this study, a popularly used

metric, namely Structural Similarity Index (SSIM) is employed to evaluate the quality of the segmented images. SSIM is defined as follows:

- **Structural Similarity Index (SSIM)** The SSIM [258] is a perceptual metric that quantifies image quality degradation caused by processing the reference image. This measure determines the similarity between the original and the segmented image. SSIM is calculated using Eq. (4.24)

$$SSIM = \frac{(2\mu_I\mu_{I_s} + c_1)(2\sigma_{I,I_s}) + c_2}{(\mu_I^2 + \mu_{I_s}^2 + c_1)(\sigma_I^2 + \sigma_{I_s}^2 + c_2)} \quad (4.24)$$

where  $\mu_I$ ,  $\mu_{I_s}$  denote the mean of intensity values of the original and the segmented images, respectively,  $\sigma_I$ ,  $\sigma_{I_s}$  represent the standard deviations of  $I$  and  $I_s$ ,  $\sigma_{I,I_s}$  is the covariance of  $I$  and  $I_s$ ,  $c_1$  and  $c_2$  are constants for the weak denominator stabilization.

#### 4.6.2.6 Experimental Design for Comparative Analysis

To evaluate and compare the segmentation algorithms under study. The analysis is organized into two experiments:

- Experiment 1: The first experiment assesses the basic and enhanced variants of the CSA and CapSA models using multilevel thresholds (4, 6, 8, and 10 thresholds).
- Experiment 2: The best-performing models from the first experiment are then compared against a selection of state-of-the-art algorithms and with similar studies from existing literature using the same objective functions and datasets.

The following parts will present the experimental findings, which comprise quantitative evaluations and visual illustrations.



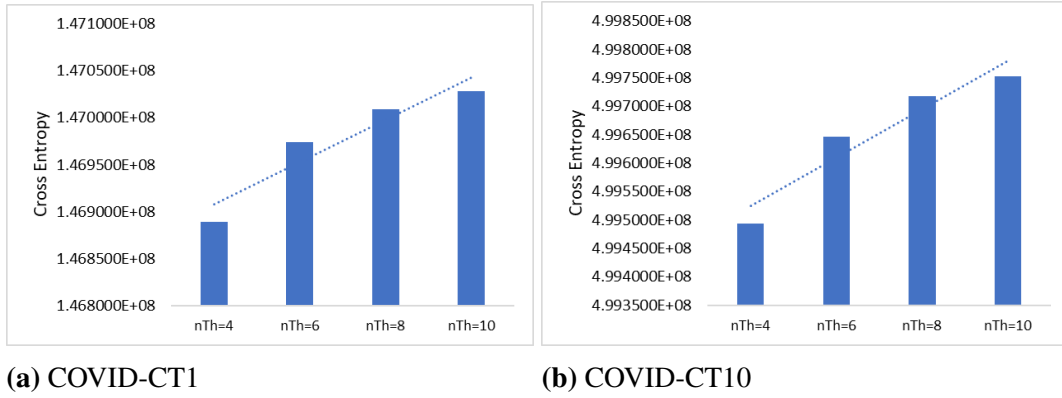
#### **4.6.2.7 Evaluation of Basic and Enhanced Variants of CSA and CapSA Across Multiple Thresholds**

The performance of CSA and CapSA algorithms and their enhanced versions across multiple threshold levels are presented in Table 4.30. A higher cross-entropy value indicates superior segmentation ability within this context. According to Table 4.30, it is clear that increasing the number of thresholds results in an overall improvement in cross-entropy values for all algorithmic variants. This improvement is expected since having more thresholds enables a more detailed segmentation. This allows algorithms to fine-tune the segmentation boundaries better. As a result, this leads to a higher cross-entropy value, which helps in achieving a more accurate segmentation of complex images like COVID-19 CT scans. The graphical representation in Figure 4.25 depicts how the cross entropy values change for iECapSA as the number of thresholds increases for two different test images. The graph indicates that the cross entropy generally increases with the increase in threshold levels for both images.

Moreover, it is observed that the algorithms perform similarly and produce competitive results when only four thresholds are used. This could be due to the lower complexity of segmentation tasks, which do not heavily strain the algorithms' capabilities. However, when the number of thresholds is increased, iECapSA stands out by showing consistently superior performance. This suggests that iECapSA is very good at handling more complex segmentation tasks and can use the additional thresholds to enhance the segmentation outcome. As per mean rank across all tested images and thresholds, iECapSA outperforms the other variants with the highest mean rank, followed by ECapSA and CSA, respectively.

**Table 4.30:** Comparative Mean Fitness Values (Cross Entropy) for Basic and Enhanced Variants of CSA and CapSA at Threshold Levels 4, 6, 8, and 10

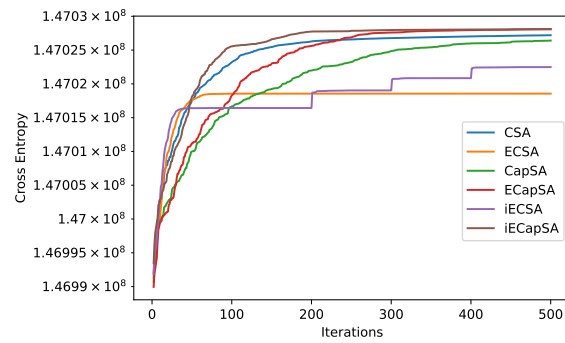
Test image	#Thresholds = 4										#Thresholds = 6									
	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA		
COVID-CT1	1.468881E+08	1.468876E+08	1.468895E+08	<b>1.468895E+08</b>	1.468888E+08	<b>1.468895E+08</b>	1.469742E+08	1.469681E+08	1.469740E+08	1.469742E+08	1.469720E+08	<b>1.469742E+08</b>	1.469742E+08	1.469720E+08	1.469742E+08	1.469720E+08	1.469742E+08	<b>1.469742E+08</b>		
COVID-CT2	<b>1.717946E+08</b>	1.717926E+08	1.717946E+08	<b>1.717946E+08</b>	1.717944E+08	<b>1.717946E+08</b>	1.718869E+08	1.718813E+08	1.718831E+08	1.718869E+08	1.718846E+08	<b>1.718869E+08</b>	1.718869E+08	1.718846E+08	1.718869E+08	1.718846E+08	1.718869E+08	<b>1.718869E+08</b>		
COVID-CT3	<b>1.778555E+08</b>	1.778537E+08	1.778555E+08	<b>1.778555E+08</b>	1.778554E+08	<b>1.778555E+08</b>	1.779492E+08	1.779418E+08	1.779492E+08	1.779492E+08	1.779457E+08	<b>1.779492E+08</b>	1.779492E+08	1.779457E+08	1.779492E+08	1.779457E+08	1.779492E+08	<b>1.779492E+08</b>		
COVID-CT4	<b>1.740768E+08</b>	1.740763E+08	1.740768E+08	<b>1.740768E+08</b>	1.740768E+08	<b>1.740768E+08</b>	1.741605E+08	1.741501E+08	1.741605E+08	1.741606E+08	1.741576E+08	<b>1.741606E+08</b>	1.741606E+08	1.741576E+08	1.741606E+08	1.741576E+08	1.741606E+08	<b>1.741606E+08</b>		
COVID-CT5	4.878392E+08	4.878380E+08	4.878392E+08	<b>4.878392E+08</b>	4.878392E+08	<b>4.878392E+08</b>	4.879085E+08	4.879009E+08	4.879085E+08	4.879088E+08	4.879052E+08	<b>4.879085E+08</b>	4.879088E+08	4.879052E+08	4.879088E+08	4.879052E+08	4.879088E+08	<b>4.879088E+08</b>		
COVID-CT6	<b>4.912576E+08</b>	4.912566E+08	4.912576E+08	<b>4.912576E+08</b>	4.912576E+08	<b>4.912576E+08</b>	4.914054E+08	4.913905E+08	4.914066E+08	4.914067E+08	4.914016E+08	<b>4.914066E+08</b>	4.914066E+08	4.914016E+08	4.914066E+08	4.914016E+08	4.914066E+08	<b>4.914066E+08</b>		
COVID-CT7	<b>1.21779E+08</b>	1.217777E+08	1.21779E+08	<b>1.21779E+08</b>	1.21779E+08	<b>1.21779E+08</b>	1.217940E+08	1.217929E+08	1.217945E+08	1.217944E+08	1.217938E+08	<b>1.217945E+08</b>	1.217944E+08	1.217938E+08	1.217944E+08	1.217938E+08	1.217944E+08	<b>1.217945E+08</b>		
COVID-CT8	<b>1.905565E+07</b>	1.905540E+07	1.905565E+07	<b>1.905565E+07</b>	1.905564E+07	<b>1.905565E+07</b>	1.907539E+07	1.907294E+07	1.907538E+07	1.907539E+07	1.907493E+07	<b>1.907539E+07</b>	1.907539E+07	1.907493E+07	1.907539E+07	1.907493E+07	1.907539E+07	<b>1.907539E+07</b>		
COVID-CT9	3.130331E+08	3.130316E+08	3.130344E+08	<b>3.130344E+08</b>	3.130343E+08	<b>3.130344E+08</b>	3.131500E+08	3.131443E+08	3.131526E+08	3.131526E+08	3.131496E+08	<b>3.131526E+08</b>	3.131526E+08	3.131496E+08	3.131526E+08	3.131496E+08	3.131526E+08	<b>3.131526E+08</b>		
COVID-CT10	<b>4.994937E+08</b>	4.994932E+08	4.994937E+08	<b>4.994937E+08</b>	4.994937E+08	<b>4.994937E+08</b>	4.996455E+08	4.996286E+08	4.996465E+08	4.996467E+08	4.996412E+08	<b>4.996467E+08</b>	4.996467E+08	4.996412E+08	4.996467E+08	4.996412E+08	4.996467E+08	<b>4.996467E+08</b>		
Mean Rank	4.4	1.1	3	4.95	2.3	<b>5.25</b>	3.6	1	3.5	4.8	2.1	<b>6</b>	3.6	1	3.5	4.8	2.1	<b>6</b>		
Test image	#Thresholds = 8										#Thresholds = 10									
	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA		
COVID-CT1	1.470087E+08	1.470023E+08	1.470076E+08	1.470090E+08	1.470045E+08	<b>1.470091E+08</b>	1.470272E+08	1.470186E+08	1.470264E+08	1.470281E+08	1.470225E+08	<b>1.470281E+08</b>	1.470272E+08	1.470186E+08	1.470264E+08	1.470281E+08	1.470225E+08	1.470281E+08		
COVID-CT2	1.719281E+08	1.719196E+08	1.719267E+08	1.719283E+08	1.719234E+08	<b>1.719284E+08</b>	1.719487E+08	1.719378E+08	1.719481E+08	1.719500E+08	1.719436E+08	<b>1.719501E+08</b>	1.719487E+08	1.719378E+08	1.719481E+08	1.719500E+08	1.719436E+08	1.719501E+08		
COVID-CT3	1.779923E+08	1.779810E+08	1.779916E+08	1.779926E+08	1.779873E+08	<b>1.779927E+08</b>	1.780134E+08	1.780019E+08	1.780119E+08	1.780151E+08	1.780062E+08	<b>1.780152E+08</b>	1.780134E+08	1.780019E+08	1.780119E+08	1.780151E+08	1.780062E+08	1.780152E+08		
COVID-CT4	1.742000E+08	1.741910E+08	1.741997E+08	1.742008E+08	1.741940E+08	<b>1.742009E+08</b>	1.742206E+08	1.742091E+08	1.742202E+08	1.742220E+08	1.742161E+08	<b>1.742221E+08</b>	1.742206E+08	1.742091E+08	1.742202E+08	1.742220E+08	1.742161E+08	1.742221E+08		
COVID-CT5	4.879411E+08	4.879348E+08	4.879444E+08	4.879473E+08	4.879394E+08	<b>4.879473E+08</b>	4.879620E+08	4.879473E+08	4.879581E+08	4.879633E+08	4.879563E+08	<b>4.879635E+08</b>	4.879620E+08	4.879473E+08	4.879581E+08	4.879633E+08	4.879563E+08	4.879635E+08		
COVID-CT6	4.914708E+08	4.914545E+08	4.914708E+08	4.914719E+08	4.914624E+08	<b>4.914727E+08</b>	4.915055E+08	4.914924E+08	4.915033E+08	4.915075E+08	4.914964E+08	<b>4.915090E+08</b>	4.915055E+08	4.914924E+08	4.915033E+08	4.915075E+08	4.914964E+08	4.915090E+08		
COVID-CT7	1.218030E+08	1.218011E+08	1.218026E+08	1.218030E+08	1.218018E+08	<b>1.218031E+08</b>	1.218075E+08	1.218053E+08	1.218068E+08	1.218078E+08	1.218060E+08	<b>1.218079E+08</b>	1.218075E+08	1.218053E+08	1.218068E+08	1.218078E+08	1.218060E+08	1.218079E+08		
COVID-CT8	1.908195E+07	1.908025E+07	1.908169E+07	1.908195E+07	1.908090E+07	<b>1.908201E+07</b>	1.908579E+07	1.908328E+07	1.908546E+07	1.908579E+07	1.908416E+07	<b>1.908590E+07</b>	1.908579E+07	1.908328E+07	1.908546E+07	1.908579E+07	1.908416E+07	1.908590E+07		
COVID-CT9	3.132039E+08	3.131917E+08	3.132052E+08	3.132068E+08	3.131987E+08	<b>3.132069E+08</b>	3.132353E+08	3.132196E+08	3.132352E+08	3.132351E+08	3.132277E+08	<b>3.132352E+08</b>	3.132353E+08	3.132196E+08	3.132351E+08	3.132352E+08	3.132277E+08	3.132352E+08		
COVID-CT10	4.997148E+08	4.997022E+08	4.997151E+08	4.997171E+08	4.997053E+08	<b>4.997175E+08</b>	4.997513E+08	4.997331E+08	4.997495E+08	4.997523E+08	4.997419E+08	<b>4.997525E+08</b>	4.997513E+08	4.997331E+08	4.997495E+08	4.997523E+08	4.997419E+08	4.997525E+08		
Mean Rank	3.7	1	3.3	5	2	<b>6</b>	4.1	1	3.1	4.8	2	<b>6</b>	4.1	1	3.1	4.8	2	<b>6</b>		



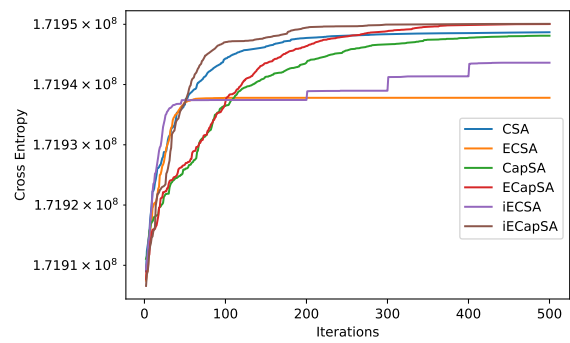
**Figure 4.25:** Variation of cross entropy with increasing number of thresholds for iECapSA on COVID-CT1 and COVID-CT2 test images

The convergence curves in Figure 4.26 reveal that all algorithms, including basic and enhanced variants, improve the quality of segmentation as the number of iterations increases. However, there are significant differences in their performance dynamics. Specifically, the iECapSA variant exhibits a faster convergence rate and achieves the segmentation task more efficiently than others. In contrast, algorithms such as ECSA show signs of stagnation, suggesting that further adjustments might be necessary to avoid premature convergence. These trends align with the fitness values reported in Table 4.30, which show that iECapSA has a superior mean rank and a robust optimization efficiency over the course of iterations. It is worth noting that each individual algorithm demonstrates similar convergence patterns across different datasets while exhibiting variations in cross-entropy values. This indicates that the algorithms exhibit stability and consistent performance regardless of the specific characteristics of the dataset. This is a desired property for MHs. The variation in cross-entropy values likely stems from the inherent differences in the datasets themselves, such as their complexity or underlying patterns.

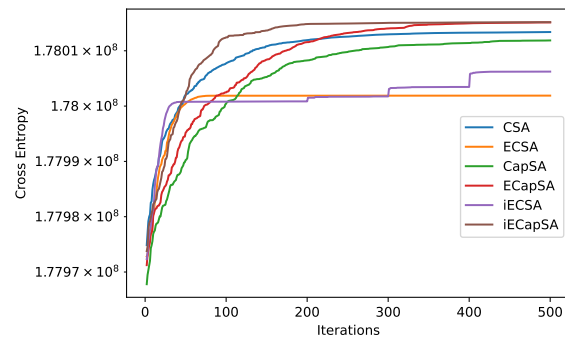
In image processing, the SSIM is a useful metric to evaluate the quality of image segmentation and is especially helpful for verifying the efficacy of image processing algorithms by calculating the similarity between the original and segmented images. It can be noticed from Table 4.31 that iECapSA exhibits strong performance with the highest SSIM scores in several test images. This implies its effectiveness in preserving the structure and details in segmented CT images. Furthermore, CSA and ECapSA, with mean ranks of 4.7 and 3.9, respectively underscores their capability to maintain structural details effectively. These



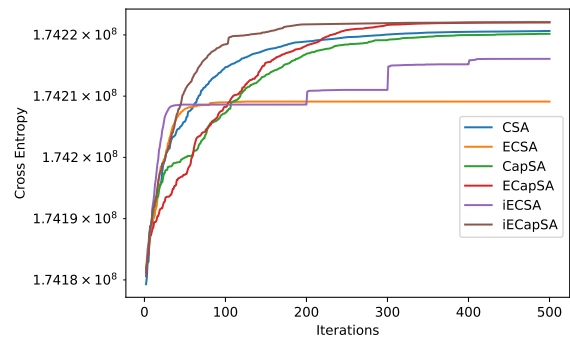
(a) COVID-CT1



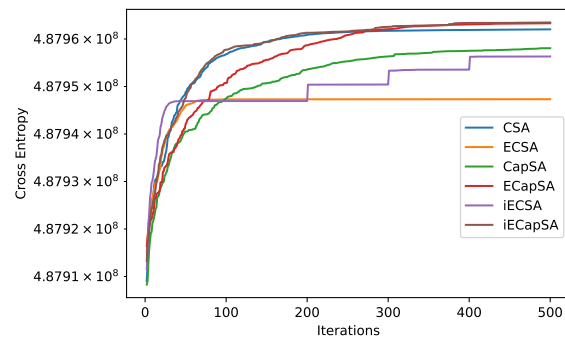
(b) COVID-CT2



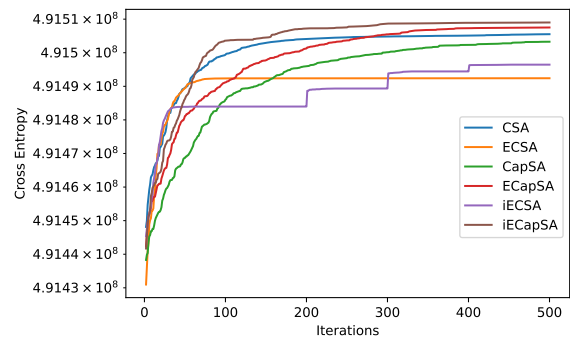
(c) COVID-CT3



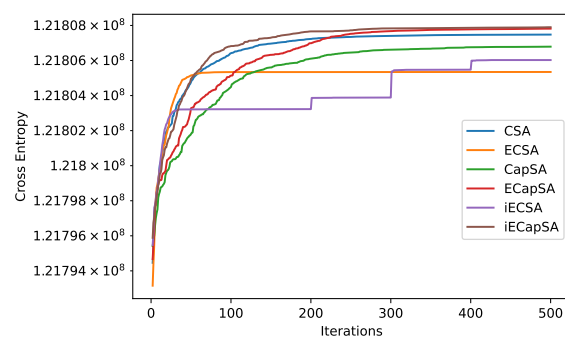
(d) COVID-CT4



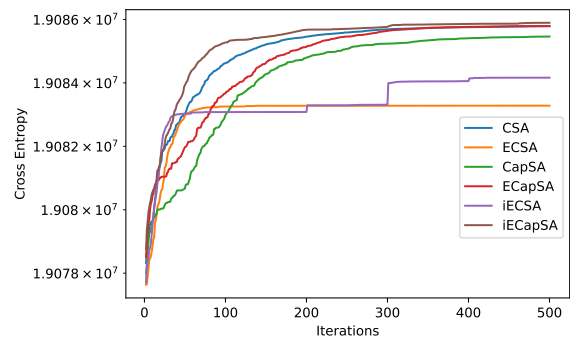
(e) COVID-CT5



(f) COVID-CT6



(g) COVID-CT7



(h) COVID-CT8

**Figure 4.26:** Convergence trends for CSA and CapSA variants on COVID-CT Images.

**Table 4.31:** Comparative evaluation of SSIM scores for CSA and CapSA variants on COVID-CT images

Test image	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
COVID-CT1	0.808734	0.786335	0.807495	0.812773	0.803975	<b>0.813083</b>
COVID-CT2	0.857784	0.842091	0.858374	0.861412	0.85496	<b>0.861432</b>
COVID-CT3	<b>0.876263</b>	0.863729	0.872113	0.875205	0.864948	0.875304
COVID-CT4	0.861498	0.843062	0.857002	0.86178	0.854459	<b>0.862086</b>
COVID-CT5	0.615431	<b>0.638566</b>	0.600875	0.60608	0.62522	0.59583
COVID-CT6	0.788994	0.782565	0.784851	0.810561	0.780604	<b>0.830072</b>
COVID-CT7	0.751017	0.750461	0.736967	0.742592	0.744764	<b>0.753414</b>
COVID-CT8	0.889179	<b>0.893008</b>	0.884482	0.875263	0.887375	0.879141
COVID-CT9	<b>0.789135</b>	0.769693	0.788014	0.787414	0.782185	0.786181
COVID-CT10	<b>0.781133</b>	0.76084	0.764067	0.780881	0.777346	0.780337
Mean Rank	<b>4.7</b>	2.4	2.9	3.9	2.6	4.5

findings indicate that both basic and enhanced variants of the algorithms have their merits in segmentation tasks, with enhanced versions like iECapSA providing notable improvements in several cases.

#### 4.6.2.8 Comparison with Similar Study in the Literature

**Table 4.32:** Performance Comparison of iECapSA with the Cooperative Island-Based Model (CPGH) from literature [7].

Test image	Cross Entropy		SSIM	
	CPGH [7]	iECapSA	CPGH [7]	iECapSA
COVID-CT1	1.47028101E+08	<b>1.47028118E+08</b>	0.812919	<b>0.813083</b>
COVID-CT2	1.71950071E+08	<b>1.71950075E+08</b>	0.860808	<b>0.861432</b>
COVID-CT3	1.78015154E+08	<b>1.78015175E+08</b>	0.875046	<b>0.875304</b>
COVID-CT4	1.74222051E+08	<b>1.74222079E+08</b>	0.861654	<b>0.862086</b>
COVID-CT5	4.87963312E+08	<b>4.87963483E+08</b>	<b>0.601746</b>	0.59583
COVID-CT6	4.91508567E+08	<b>4.91509011E+08</b>	0.821366	<b>0.830072</b>
COVID-CT7	<b>1.21807905E+08</b>	1.21807900E+08	<b>0.753667</b>	0.753414
COVID-CT8	1.90858934E+07	<b>1.90858955E+07</b>	<b>0.882086</b>	0.879141
COVID-CT9	3.13236302E+08	<b>3.13236531E+08</b>	<b>0.787793</b>	0.786181
COVID-CT10	4.99752447E+08	<b>4.99752482E+08</b>	0.779823	<b>0.780337</b>

In this part, the effectiveness of the proposed iECapSA model is compared with a contemporary island-based cooperative model known as CPGH, as reported in a recent study by Sabha et al. [7]. To ensure a fair and unbiased comparison, the researcher used the same parameters and settings for both models, including a population size and maximum iterations of 30 and 500, respectively, as well as the same objective function. This consistent experimental framework allows for an objective evaluation of the strengths of each approach.

Table 4.32 presents the comparative evaluation of cross-entropy and SSIM metrics for the

iECapSA and CPGH models on the COVID-CT Image Dataset. The results reveal that both methods perform well in multilevel thresholding tasks. However, iECapSA shows slightly better results in the majority of test cases. For cross-entropy results, iECapSA outperforms the CPGH model in 90% of the test cases. For the SSIM metric, which reflects the structural accuracy of segmentation, iECapSA surpasses CPGH in 60% of the cases. These results highlight the effectiveness of iECapSA in providing high-quality segmentation results that match, and often exceed, contemporary models in the literature.

#### **4.6.3 Experimental Results: Software Reliability Growth Models Estimation**

SRGMs are essential in software engineering because they offer a mathematical method for predicting reliability and estimating the number of defects or failures that may occur throughout its life cycle [259]. Because these models predict probable software failures, they are useful for controlling risk, scheduling maintenance tasks, and effectively allocating resources. Through the analysis of historical failure data, SRGMs contribute to the understanding and enhancement of the software development process. This makes reliability prediction a crucial component of software quality assurance and testing procedures [260, 261].

In the realm of complex software systems, reliability is of utmost significance. This is where the utilization of SRGMs becomes critical. These models play a vital role in decision-making regarding software release dates and also offer support even after the software is released by providing valuable insights into the debugging process and the overall failure of the system. In the world of research, a multitude of models have been introduced to forecast the reliability of software systems. Some renowned ones include the exponential Goel-Okumoto model [262], the power model known as the Non-Homogeneous Poisson Process [263], and the delayed S-shaped model [264].

The success of traditional models heavily relies on key parameters that depict various facets of a program's dependability, including the projected quantity of errors and the speed at which errors are identified. It is imperative to determine these parameters correctly, as they significantly impact the model's precision in predicting software reliability [265]. To this end, techniques such as numerical estimation, maximum likelihood estimation (MLE),

and least square estimation (LSE) are commonly utilized, as outlined by Lyu in 1996 [266]. However, these techniques come with their limitations. For example, they often assume that the model functions smoothly and its mathematical expressions can be differentiated [267]. However, when a model is complex or has many parameters, these methods struggle, especially with nonlinear functions or when trying to hone in on the best solution without getting stuck in less optimal ones. These challenges highlight the need for better methods to estimate the parameters of software reliability models. Currently, MHs show great potential in enhancing software reliability predictions [268, 269, 270]. By incorporating these algorithms, we can enhance the precision of reliability forecasts, enabling more informed decision-making in software development and maintenance.

#### 4.6.3.1 MHs-based Optimization of SRGMs: Problem Formulation

To achieve optimal results with SRGMs, it is crucial to fine-tune the model parameters to accurately capture patterns observed in past software failure data. This requires the formulation of an effective optimization problem, which can be addressed using MHs. This section will present the formulation of this optimization problem, paying close attention to the selection of solution representation, fitness function, and model selection methods.

- **Solution Representation:** When it comes to optimizing SRGM, a common approach is to represent the solution as a vector, where the model parameters are transformed into a suitable form that can be optimized using metaheuristic algorithms. For example, a vector such as  $[a, b, c]$  can effectively capture the three parameters  $a$ ,  $b$ , and  $c$  for a particular model. It's worth noting that these parameters are constrained by both theoretical limitations and practical considerations, and can vary within established ranges.
- **Fitness Function:** When it comes to optimizing SRGM, the fitness function plays a crucial role in evaluating the effectiveness of a set of model parameters in predicting historical software failures. Its main goal is to minimize the disparity between the failure rates predicted by the tested SRGM and the actual observed failure data. This objective can be quantified using statistical measures such as the Mean Squared Error

(MSE) or the Negative Log-Likelihood (NLL), where a lower value indicates a better fit of the model to the data. In this study, the author employed MSE in Eq. 4.25 as a fitness function.

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n (\hat{y}_t - y_t)^2 \quad (4.25)$$

where  $y_t$  represents the observed failures at time  $t$ , and  $\hat{y}_t$  denotes the failures predicted by the SRGM.

#### 4.6.3.2 Employed SRGMs

Choosing the appropriate SRGM is of extreme significance, as each model operates on distinct assumptions regarding the occurrence and resolution of software failures. The decision typically relies on specific characteristics of the software, the development process, and the existing data on failures. However, one of the main objectives of this study is to analyze the performance of well-developed MHs in optimizing SRGMs. For this purpose, the author utilized three models that are commonly used in the literature. These models include the exponential Goel-Okumoto model, power model, and the delayed S-shaped model (see Fig. 4.27)

1. **Exponential Goel-Okumoto Model:** This model is one of the earliest and most straightforward SRGMs, which assumes that the failure detection rate decreases exponentially with time. The cumulative number of detected failures  $\mu(t)$  at time  $t$  is given by:

$$\mu(t) = a(1 - e^{-bt}) \quad (4.26)$$

where  $a$  is the total number of faults detected by the model and  $b$  is the fault detection rate.

2. **Power Model:** Also known as the Non-Homogeneous Poisson Process (NHPP) model, the power model represents the cumulative number of failures as a function of time with a power-law distribution. The equation is as follows:

$$\mu(t) = at^b \quad (4.27)$$

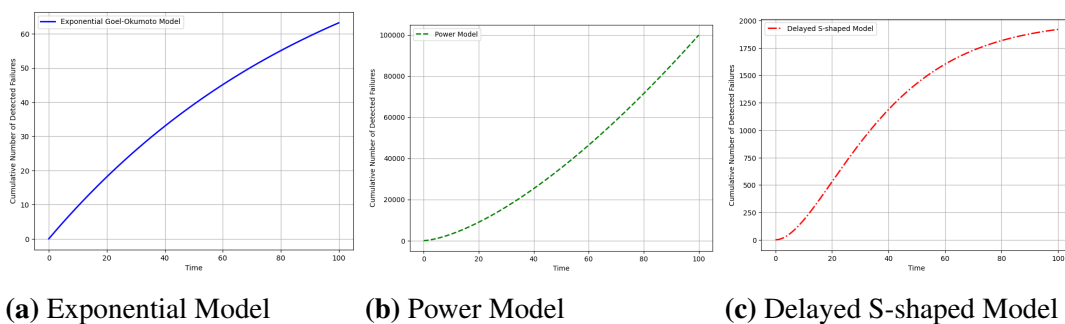


where  $a$  and  $b$  are parameters of the model, representing the initial fault detection rate and the growth parameter, respectively.

**3. Delayed S-shaped Model:** The Delayed S-shaped model assumes that the rate of failure detection follows a curve that is more delayed compared to other models. This description is appropriate for explaining the software failure process during testing phases, where the detection of failures begins slowly, then speeds up, and eventually slows down. The mathematical representation can be expressed as follows:

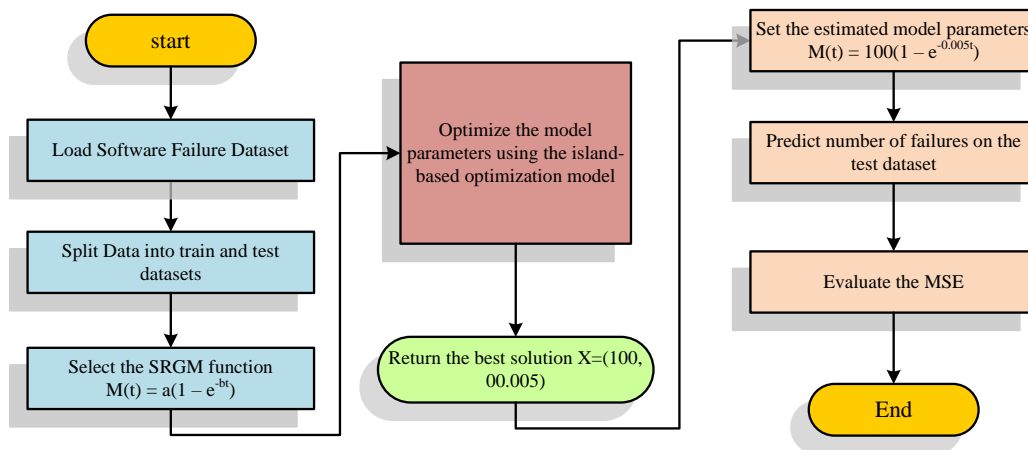
$$\mu(t) = a(1 - (1 + bt)e^{-bt}) \quad (4.28)$$

where  $a$  and  $b$  are model parameters similar to those in the Goel-Okumoto model but fitted to produce the S-shaped curve.



**Figure 4.27:** Graphical representation of the selected SRGMs

The models were chosen to thoroughly evaluate the optimization potential of the proposed algorithms in different SRGM formulations, considering their diverse assumptions about software failure detection. This aligns with the study's objectives by demonstrating that the developed models have the ability to effectively forecast software reliability by identifying the optimal values for parameters  $a$  and  $b$  for each model. This will provide us with valuable insights into the effectiveness of SRGMs in real-world scenarios.



**Figure 4.28:** Overall approach of SRGM optimization

### 4.6.3.3 Optimization Framework for SRGMs

#### 4.6.3.4 Experimental Setup

To ensure a fair comparison among the various algorithms investigated, this study uses standardized testing conditions. The population size for common parameters is fixed at 30, and the maximum number of iterations is limited to 50. This setup was determined after extensive preliminary testing, which revealed that 50 iterations were sufficient to achieve excellent outcomes. This balance between computational efficiency and thorough exploration of the solution space enables the identification of optimal or near-optimal solutions. Additionally, the internal parameters for all algorithms were carefully selected based on recommended settings and best practices outlined in prior research.

It is crucial to define the minimum and maximum values for parameters  $a$  (scaling factor) and  $b$  (rate parameter) when estimating statistical models. The selection of the parameter range can significantly impact the effectiveness of the search technique. By configuring parameters with a too-wide range, the search algorithms may inefficiently explore regions that are not feasible. Conversely, if the boundaries are excessively restrictive, there is a risk of missing the global optimum. In exponential and delayed S-shaped models, for instance, the parameter  $a$ , which reflects the total expected number of failures, should be flexible enough to cover the entire potential range of outcomes suggested by empirical data. Therefore, the

range for the parameter  $a$  (and other model parameters) can vary depending on the dataset used for modeling. Consequently, after extensive preliminary analysis, iterative refinement, and considering existing literature, the range for  $a$  is set between 0 and twice the total number of observed failures in each dataset. The range for  $b$  is limited from 0 to 1 to accommodate variability in failure detection rates. In the power model, the values of  $a$  and  $b$  are limited within the ranges of  $[0, 50]$  and  $[0, 1]$ , respectively. Table 4.33 summarizes the parameter ranges set for each SRGM.

**Table 4.33:** Parameter ranges for SRGM estimation

<b>Model Type</b>	<b>Parameter <math>a</math> Range</b>	<b>Parameter <math>b</math> Range</b>
Exponential & Delayed S-Shaped	$[0, 2 \times (\text{Total Failures})]$	$[0, 1]$
Power Model	$[0, 50]$	$[0, 1]$

Due to the stochastic nature of metaheuristic techniques, each experiment is repeated 30 independent times to ensure accurate and reliable results. Statistical tests, such as the Friedman and Wilcoxon rank-sum tests, are used to evaluate the significance of performance differences and provide a solid basis for comparative analysis.

#### 4.6.3.5 Real Datasets

Our study examines the effectiveness of proposed optimization methods for SRGMs using ten real datasets, labeled as DS1 through DS7. Notably, one of the datasets, DS2, includes four separate releases that illustrate how software development evolves and how software reliability is tested under different conditions. These datasets were carefully chosen from existing literature and are widely used in various studies. Each dataset has varying characteristics, such as time units and total numbers of failures, allowing for a robust evaluation framework to analyze the effectiveness of optimization techniques across different real-world conditions and challenges seen in software reliability modeling. Table 4.34 summarizes the details of the datasets used for evaluating SRGMs in this study. In this table, the "Measurements" column provides information on the time period during data collection, specified in days, weeks, or not applicable (N/A) when the unit is unspecified. The "Failures" column

displays the accumulative number of failures.

**Table 4.34:** Summary of datasets used for SRGM Evaluation

Dataset Name	Measurements	Time Unit	Failures	Description
DS1 [271]	109	Days	535	A real-time control application with 870 KLOC of FORTRAN.
DS2-R1 [272]	20	Weeks	100	First release testing phase.
DS2-R2 [272]	19	Weeks	120	Second release testing phase.
DS2-R3 [272]	12	Weeks	61	Third release testing phase.
DS2-R4 [272]	19	Weeks	42	Fourth release testing phase.
DS3 [273]	35	Weeks	1301	Radar system software testing data.
DS4 [274]	199	Days	55	The software consists of about 14.5 kilo lines of ASSEMBLER language and is developed for a railway interlocking system
DS5 [274]	111	Days	481	A real-time control application with daily collected data.
DS6 [275]	38	Weeks	231	Space shuttle flights application data.
DS7 [270]	46	Days	266	-

#### 4.6.3.6 Evaluation Metrics

To assess the performance of the proposed models in estimating SRGMs, three key metrics were utilized: MSE, Variance Accounted For (VAF), and the correlation coefficient (R). Each of these metrics offers insights into different aspects of estimation quality and predictive accuracy.

1. **Mean Squared Error (MSE):** MSE is a widely used measure of the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. It is mathematically defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (4.29)$$

where  $\hat{y}_i$  is the predicted value and  $y_i$  is the actual value. Lower values of MSE indicate better fit of the model to the data.

2. **Variance Accounted For (VAF):** VAF is a statistical measure used to assess the proportion of total variation in a dataset that is explained by the model. Higher VAF values indicate that the model explains a larger portion of the variance and is thus

more effective. It is calculated using the formula:

$$\text{VAF (\%)} = \left( 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)} \right) \times 100\% \quad (4.30)$$

where  $\text{Var}(y - \hat{y})$  is the variance of the model residuals, and  $\text{Var}(y)$  is the variance of the original data.

3. **Correlation Coefficient (R):** The correlation coefficient,  $R$ , measures the strength and direction of a linear relationship between the predicted values and actual values. It is a value between -1 and 1 where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship between the variables. It is especially useful in the context of SRGMs as it provides a clear indication of how well the model predictions conform to the actual trends in the data. The formula for  $R$  is:

$$R = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (4.31)$$

where  $x_i$  and  $y_i$  are the individual sample points indexed with  $i$ , while  $\bar{x}$  and  $\bar{y}$  are the means of the  $x$  and  $y$  samples, respectively.

#### 4.6.3.7 Experimental Procedures

In this study, a comprehensive comparative analysis design is used to evaluate the effectiveness of basic and augmented models in estimating SRGMs. The experimental design was structured into three distinct experiments, each designed to evaluate different aspects of the models' capabilities:

1. **Comparison of CSA and CapSA Variants:** In the initial experiment, the basic variants of CSA and CapSA, along with their enhanced versions and the cooperative island-based models (iECSA and iECapSA), are assessed. Each algorithm is applied to estimate the parameters of three different SRGMs: the Exponential Model, the Delayed S-shaped Model, and the Power Model. The effectiveness of each algorithm is

tested across ten distinct datasets.

2. **Detailed Parameter Estimation:** The second experiment focuses on the best-performing optimization model, as determined by the outcomes of the first experiment. For each selected SRGM, the parameters  $a$  (scale factor) and  $b$  (rate of failure detection) are analyzed. Various visualization techniques are employed to illustrate the quality of the estimations, including plots of the actual versus estimated failures and scatter plots that highlight the correlation between estimated and measured values.
3. **Benchmarking Against Established Metaheuristics:** The third experiment extends the validation by comparing the best-performing models from the initial experiment against a selection of ten well-known metaheuristic algorithms.

#### 4.6.3.8 Results of CSA and CapSA Variants for SRGMs

In this comprehensive evaluation, Table 4.35 presents the average MSE obtained by each variant of CSA and CapSA applied to Exponential, Delayed S-Shaped, and Power SRGMs across ten datasets. The results provide valuable benchmarks that reveal the precision of each algorithm in estimating parameters. Notably, ECSA demonstrates superior performance compared to the basic CSA in all test cases. Similarly, ECapSA outperforms CapSA in every scenario. A closer examination reveals very competitive outcomes between iECSA and ECSA, with identical results in the majority of cases. However, iECSA surpasses ECSA in 40% of the scenarios. Interestingly, iCapSA exhibits superiority over ECapSA in all test cases, highlighting the effectiveness of its island-based approach.

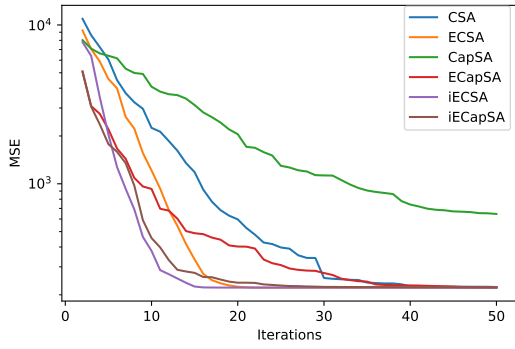
The overall mean rank, which represents overall performance, reveals that iECSA is the most effective (with a rank of 1.55), followed by ECSA (ranked 2.02), iECapSA (ranked 3.13), CSA (ranked 3.87), ECapSA (ranked 4.43), and CapSA (ranked 6.00). This ranking sheds light on the competitive landscape of SRGM estimation, highlighting the significant gains of enhanced variants and the optimization efficiency of cooperative models like iECSA. This ranking highlights the significant gains of enhanced variants and the optimization efficiency of cooperative models like iECSA.

The MSE convergence curves for the delayed S-shaped SRGM across selected datasets

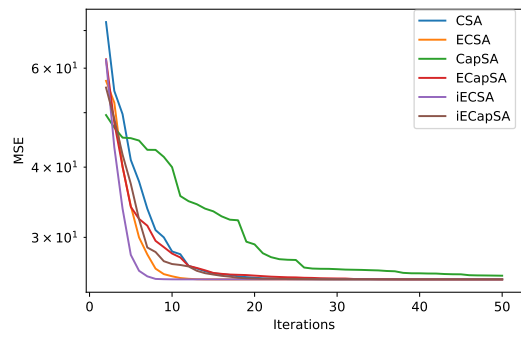
**Table 4.35:** Average MSE obtained by CSA and CapSA variants for SRGM parameter estimation across diverse datasets

Dataset	Model	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
DS-1	Exponential	832.9371	823.7756	1236.806	831.442	<b>823.7756</b>	824.1108
	Delayed S-Shaped	223.8423	<b>222.0614</b>	645.8898	223.4968	<b>222.0614</b>	222.1688
	Power	1708.484	<b>1708.477</b>	1846.375	1716.145	<b>1708.477</b>	1708.564
DS-2-R1	Exponential	11.61833	<b>11.61711</b>	14.68396	11.64696	<b>11.61711</b>	11.61842
	Delayed S-Shaped	25.25644	<b>25.25638</b>	25.63897	25.26045	<b>25.25638</b>	25.25674
	Power	23.35376	<b>23.35341</b>	29.54804	23.38944	<b>23.35341</b>	23.35412
DS-2-R2	Exponential	23.27168	23.26492	27.8051	23.35933	<b>23.26492</b>	23.26651
	Delayed S-Shaped	13.14209	<b>13.14193</b>	14.51285	13.16343	<b>13.14193</b>	13.14257
	Power	43.33847	<b>43.33782</b>	53.33324	44.22213	<b>43.33782</b>	43.3402
DS-2-R3	Exponential	22.07759	21.92471	25.42928	21.9199	21.92038	<b>21.9199</b>
	Delayed S-Shaped	8.018695	<b>8.018679</b>	8.344146	8.019457	<b>8.018679</b>	8.018848
	Power	22.87514	<b>22.8742</b>	30.29059	22.88428	<b>22.8742</b>	22.87453
DS-2-R4	Exponential	4.591566	4.506024	5.687898	4.543267	4.502008	<b>4.501976</b>
	Delayed S-Shaped	0.979913	<b>0.9799</b>	1.176811	0.980354	<b>0.9799</b>	0.979937
	Power	6.251995	<b>6.249236</b>	14.58894	6.292839	<b>6.249236</b>	6.249474
DS-3	Exponential	12562.24	11541.55	15599.16	11520.93	11521.63	<b>11520.93</b>
	Delayed S-Shaped	2713.447	2713.218	3668.76	2718.327	<b>2713.218</b>	2713.766
	Power	9070.404	<b>9070.385</b>	10636.88	9107.148	<b>9070.385</b>	9107.141
DS-4	Exponential	50.64331	42.0274	60.44406	36.3099	35.51735	<b>35.50973</b>
	Delayed S-Shaped	9.089848	7.886267	19.43244	7.255138	<b>7.064083</b>	7.064893
	Power	46.49216	18.76543	105.5807	35.15815	<b>11.96032</b>	20.16518
DS-5	Exponential	792.0958	791.519	1002.019	792.4542	<b>791.5172</b>	791.6149
	Delayed S-Shaped	323.86	<b>323.8196</b>	365.2136	324.1367	<b>323.8196</b>	323.8518
	Power	2275.824	<b>2275.823</b>	2292.901	2276.169	<b>2275.823</b>	2275.852
DS-6	Exponential	26.48546	23.2684	125.1229	22.12133	<b>20.19431</b>	20.19999
	Delayed S-Shaped	126.3304	<b>126.3272</b>	154.3746	126.5221	<b>126.3272</b>	126.3326
	Power	13.8483	<b>13.8436</b>	88.51212	17.20592	<b>13.8436</b>	13.84821
DS-7	Exponential	150.0178	147.8874	230.4005	148.7787	<b>147.8805</b>	147.8946
	Delayed S-Shaped	345.3016	<b>345.2994</b>	375.5375	345.4135	<b>345.2994</b>	345.306
	Power	150.7345	<b>150.7316</b>	210.0545	151.4052	<b>150.7316</b>	150.7518
Mean Rank		3.87	2.02	6.00	4.43	<b>1.55</b>	3.13

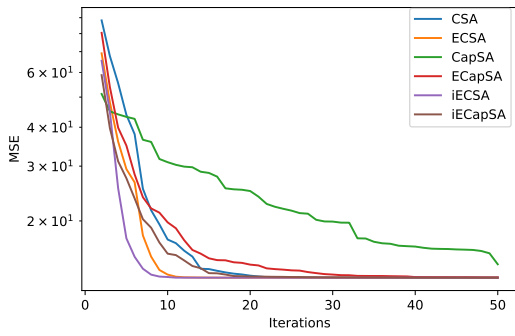
are shown in Figure 4.29. These curves show how the CSA and CapSA variants converged during the optimization process. Notably, while the final MSE values are comparable among the algorithms, island-based models such as iECSA and iECapSA showed faster convergence with fewer iterations and attained lower MSEs. This fast convergence proves their effectiveness in finding optimal parameters for SRGMs and indicates their potential for time-critical applications that require quick model tuning.



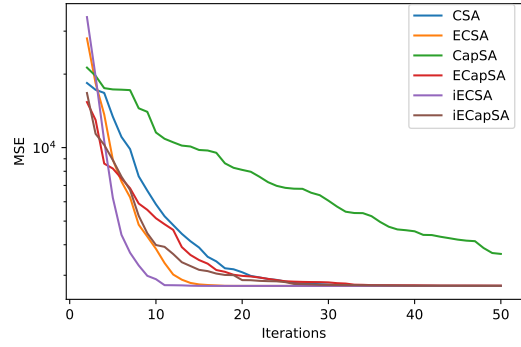
(a) DS-1



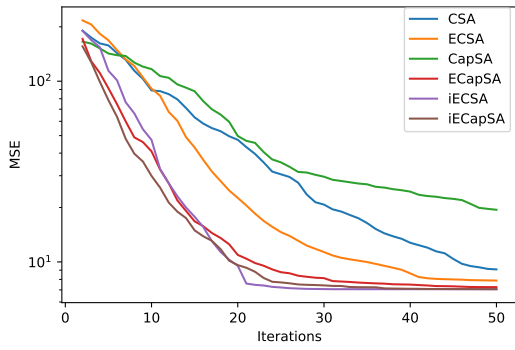
(b) DS-2-R1



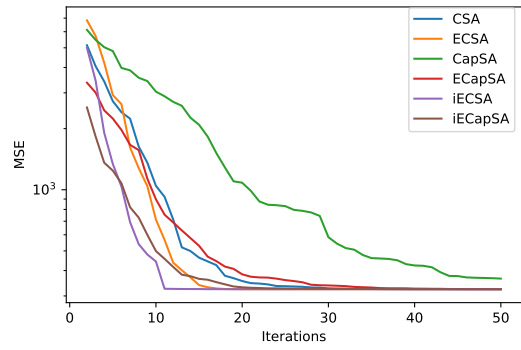
(c) DS-2-R2



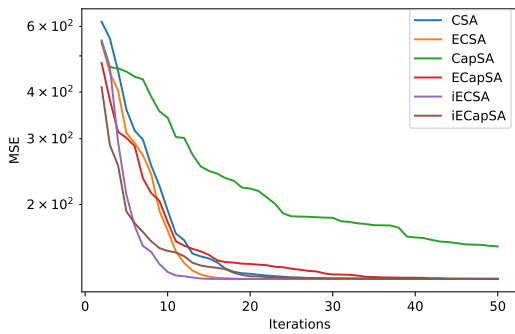
(d) DS-3



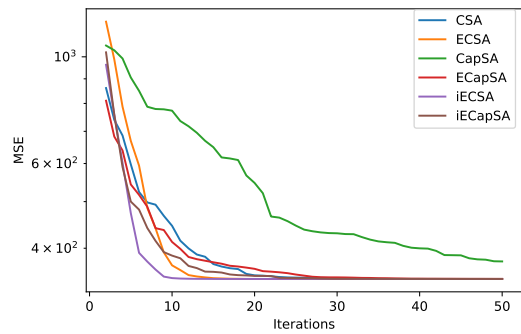
(e) DS-4



(f) DS-5



(g) DS-6



(h) DS-7

**Figure 4.29:** MSE convergence trends for CSA and CapSA variants on Delayed S-Shaped SRGM

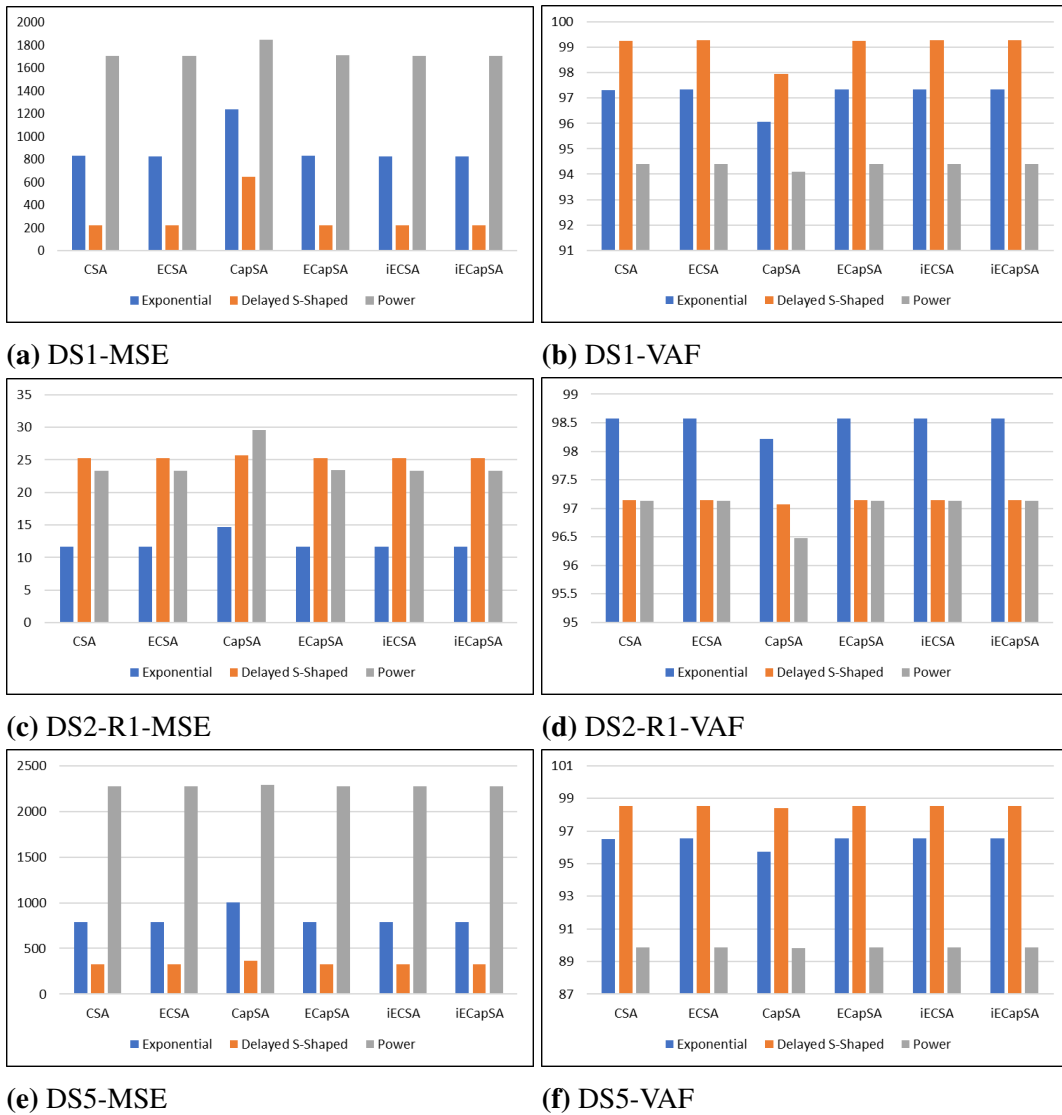


**Table 4.36:** VAF results of CSA and CapSA variants for SRGM parameter estimation across 10 datasets

Dataset	Model	CSA	ECSA	CapSA	ECapSA	iECSA	iECapSA
DS-1	Exponential	97.31927	97.34317	96.06008	97.32918	97.34317	<b>97.34551</b>
	Delayed S-Shaped	99.26438	<b>99.27067</b>	97.94214	99.26571	99.27067	99.27055
	Power	94.39411	94.39397	94.08971	94.38894	94.39397	<b>94.39484</b>
DS-2-R1	Exponential	98.57178	<b>98.572</b>	98.22076	98.56854	98.572	98.57178
	Delayed S-Shaped	<b>97.14106</b>	97.14081	97.07327	97.13765	97.14081	97.14015
	Power	97.13268	97.1328	96.47542	97.13065	97.1328	<b>97.13283</b>
DS-2-R2	Exponential	98.25795	98.25825	97.93939	98.2533	98.25825	<b>98.25855</b>
	Delayed S-Shaped	99.05915	99.05956	98.95323	99.05637	<b>99.05956</b>	99.05874
	Power	96.73398	96.73399	96.22333	96.68687	96.73399	<b>96.73429</b>
DS-2-R3	Exponential	95.00443	<b>95.04363</b>	94.30478	95.03912	95.03902	95.03913
	Delayed S-Shaped	98.10918	98.10918	98.03545	98.10906	<b>98.10918</b>	98.10916
	Power	94.64171	94.64228	93.47207	<b>94.64513</b>	94.64228	94.643
DS-2-R4	Exponential	97.65893	97.69828	97.14673	97.68618	97.70114	<b>97.70134</b>
	Delayed S-Shaped	99.47431	<b>99.47432</b>	99.38039	99.47409	99.47432	99.4743
	Power	96.69678	96.69647	93.07372	96.6913	96.69647	<b>96.69719</b>
DS-3	Exponential	94.52222	95.00277	93.2617	95.00691	<b>95.00953</b>	95.00691
	Delayed S-Shaped	98.74244	98.74223	98.33481	98.74145	98.74223	<b>98.74321</b>
	Power	95.80364	95.80342	95.41571	95.81513	95.80342	<b>95.81524</b>
DS-4	Exponential	87.74311	89.99245	85.2724	91.33633	91.50871	<b>91.51707</b>
	Delayed S-Shaped	97.69481	97.95439	95.10962	98.11884	<b>98.1596</b>	98.15948
	Power	89.22191	95.38236	75.70576	92.12177	<b>96.9852</b>	95.37573
DS-5	Exponential	96.5264	96.53205	95.73816	<b>96.53261</b>	96.53184	96.53226
	Delayed S-Shaped	98.55138	<b>98.5514</b>	98.3882	98.54982	98.5514	98.55117
	Power	89.86392	89.86389	89.81669	89.8628	89.86389	<b>89.86455</b>
DS-6	Exponential	99.31717	99.38836	96.78533	99.41894	<b>99.46998</b>	99.46932
	Delayed S-Shaped	<b>96.93198</b>	96.93179	96.31907	96.92446	96.93179	96.92988
	Power	99.62772	<b>99.62779</b>	97.94617	99.54795	99.62779	99.62761
DS-7	Exponential	97.57715	97.6099	96.35519	97.59595	97.60984	<b>97.60997</b>
	Delayed S-Shaped	94.68183	94.68262	94.18295	<b>94.68321</b>	94.68262	94.6818
	Power	97.56582	97.56587	96.8526	97.56051	97.56587	<b>97.56616</b>
Mean Rank		3.70	2.63	6.00	3.97	2.40	<b>2.30</b>

When inspecting the VAF outcomes in Table 4.36, it is evident that the enhanced variants of CSA and CapSA show significant improvement over their standard forms in SRGM parameter estimation. Specifically, the island-based approaches iECapSA and iECSA, along with the ECSA algorithm, consistently achieve high VAF scores across the majority of datasets. This indicates their precision in estimating SRGM parameters. Notably, the iECapSA variant has achieved remarkable results, often matching or surpassing its counterparts. Analysis of the mean ranks across all datasets—including the exponential, delayed S-shaped, and power models—indicates that iCapSA (with a mean rank of 2.3) and iECSA (with a mean rank of 2.4) are the top-performing algorithms for SRGM parameter optimization tasks. The results reported in Tables 4.35 and 4.36 are visually represented in Figure 4.30. The provided charts illustrate the MSE and VAF values for the exponential, delayed S-shaped, and

power models optimized using CSA, ECSA, CapSA, ECapSA, iECSA, and iECapSA across datasets DS-1, DS-2-R1, and DS-5. These visualizations provide a quick and clear understanding of how well the algorithms can accurately estimate software growth models in various settings.



**Figure 4.30:** Visualization of MSE and VAF across different datasets and SRGMs using CSA, ECSA, CapSA, ECapSA, iECSA, and iECapSA

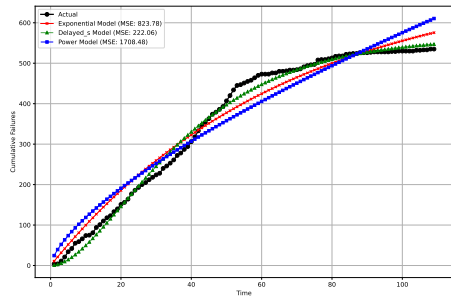
#### 4.6.3.9 Detailed Parameter Estimation Results of iECSA

This section focuses on highlighting the superior performance of the iECSA model, which was established in previous experiments. The results presented in Table 4.37 showcase the optimal parameters of the SRGMs that were determined by the iECSA algorithm across various datasets. It displays the optimal parameter values of  $a$  and  $b$  for each dataset, along with the model that had the lowest MSE and highest VAF. It is clear that the Delayed S-shaped model is the preferred choice for most datasets (7 out of 10), as it is known for capturing the iterative development and maturation of software systems. However, the Exponential model is optimal for DS-2-R1 and DS-7, which are typically suited for software processes with a consistent failure rate over time. On the other hand, the Power model is best for DS-6. These

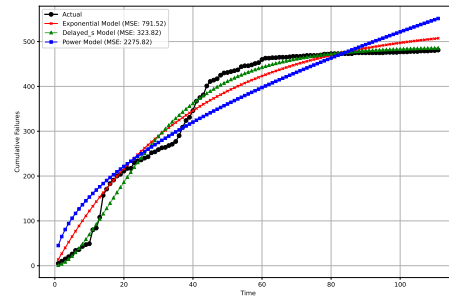
**Table 4.37:** Optimal SRGM parameters estimated by iECSA across datasets with corresponding MSE and VAF

Dataset	Best Model	a	b	Model Equation	MSE	VAF
DS-1	Delayed S-Shaped	563.761	0.049256	$\mu(t) = 563.761(1 - (1 + 0.049256t)e^{-0.049256t})$	222.06	99.27
DS-2-R1	Exponential	130.2015	0.083166	$\mu(t) = 130.2015(1 - e^{-0.083166t})$	11.62	98.57
DS-2-R2	Delayed S-Shaped	127.3989	0.241689	$\mu(t) = 127.3989(1 - (1 + 0.241689t)e^{-0.241689t})$	13.14	99.06
DS-2-R3	Delayed S-Shaped	74.69518	0.28847	$\mu(t) = 74.69518(1 - (1 + 0.28847t)e^{-0.28847t})$	8.02	98.11
DS-2-R4	Delayed S-Shaped	47.22906	0.207025	$\mu(t) = 47.22906(1 - (1 + 0.207025t)e^{-0.207025t})$	0.98	99.47
DS-3	Delayed S-Shaped	1689.371	0.089848	$\mu(t) = 1689.371(1 - (1 + 0.089848t)e^{-0.089848t})$	2713.22	98.74
DS-4	Delayed S-Shaped	101.3656	0.009991	$\mu(t) = 101.3656(1 - (1 + 0.009991t)e^{-0.009991t})$	7.06	98.16
DS-5	Delayed S-Shaped	488.2343	0.066212	$\mu(t) = 488.2343(1 - (1 + 0.066212t)e^{-0.066212t})$	323.82	98.55
DS-6	Power	10.80589	0.830345	$\mu(t) = 10.80589t^{0.830345}$	13.84	99.63
DS-7	Exponential	423.7194	0.023173	$\mu(t) = 423.7194(1 - e^{-0.023173t})$	147.88	97.61

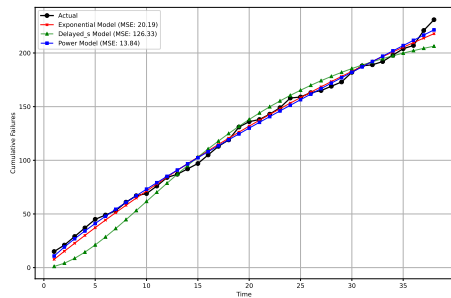
findings show how well the iECSA performs in precisely predicting the parameters of various SRGMs and adjusting to the underlying patterns in the data to get a good fit. The variability in the 'a' and 'b' parameters across models highlights the algorithm's flexibility in capturing the complex dynamics of software failure data. High VAF values across most datasets further confirm the reliability of the models in predicting the number of failures, which is crucial for planning and resource allocation in software maintenance and development.



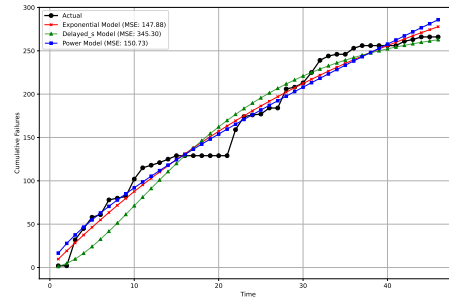
(a) DS1



(b) DS5



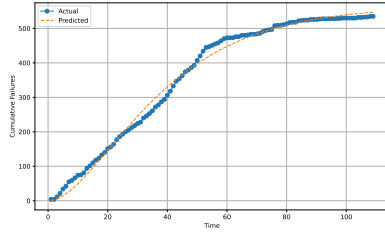
(c) DS6



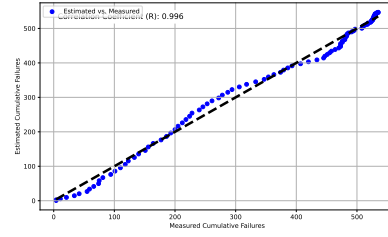
(d) DS7

**Figure 4.31:** Comparison of actual vs. iECSA-estimated failures across selected datasets for three SRGMs

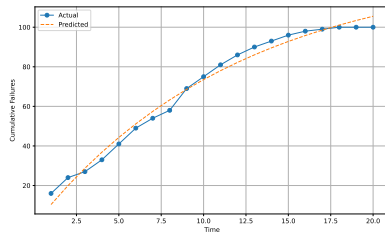
A visual comparison between the actual recorded number of failures and the predictions made by the iECSA-optimized SRGMs for selected datasets is illustrated in Figure 4.31. The proximity of the estimated fault curves to the actual data points highlights the predictive accuracy of the model. It is clear that the Delayed S-Shaped model shows a strong alignment with the real data in datasets DS-1 and DS-5, indicating a highly accurate fit that well reflects the underlying failure process. In contrast, the Exponential model displayed for DS-7 shows an initial overestimation, which stabilizes as time progresses, closely following the trend of the actual failures. Meanwhile, the Power model's fit to DS-6 indicates a good estimation throughout the observation period, with only minor deviations. Overall, the congruence of these curves validates the efficacy of iECSA in estimating parameters, demonstrating its potential usefulness in software reliability and fault prediction tasks. The actual vs. predicted cumulative faults and correlation scatter plots for the Delayed S-shaped SRGM estimated by iECSA are depicted in Figure 4.32. These plots provide a comprehensive investigation of the predicted accuracy of the iECSA model for the Delayed S-shaped SRGM using datasets



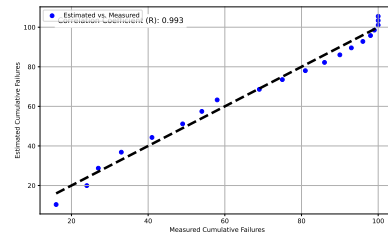
(a) DS1



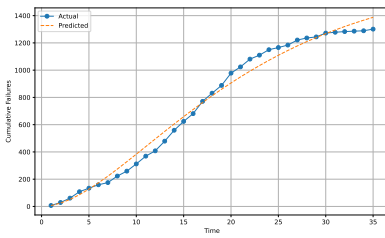
(b) DS1



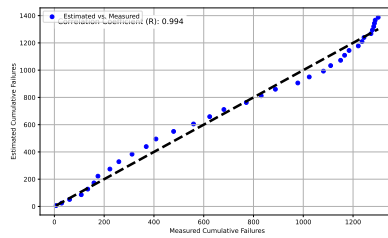
(c) DS\_2\_R1



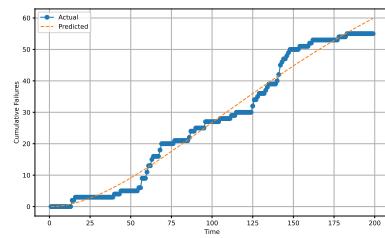
(d) DS\_2\_R1



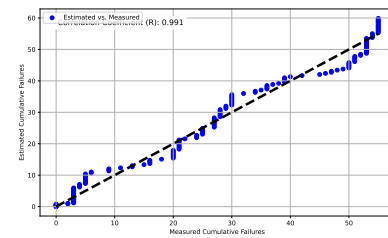
(e) DS\_3



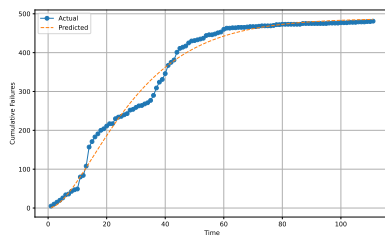
(f) DS\_3



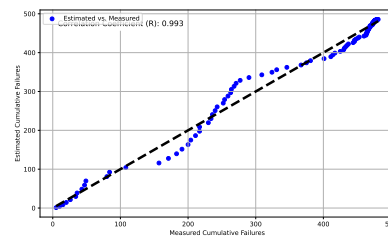
(g) DS\_4



(h) DS\_4



(i) DS\_5



(j) DS\_5

**Figure 4.32:** Actual vs. predicted cumulative faults and correlation scatter plots for the Delayed S-shaped SRGM estimated by iECSA

DS1, DS2\_R1, DS3, DS4, and DS5. The line graphs on the left demonstrate the alignment of estimated faults over time with the actual observed data. The close alignment of the two curves in each graph highlights the excellent precision of iECSA in accurately representing the total number of errors throughout the software lifecycle.

The scatter plots on the right depict the correlation between the measured and estimated cumulative faults. Each point represents an aggregated data point over time, with the dashed line indicating a perfect correlation. We can see that the proximity of the points to this line, together with the high correlation coefficients observed, indicate the robust prediction powers. The scatter plots demonstrate a significantly strong correlation, as seen by the correlation coefficients  $R$  of 0.996, 0.993, 0.994, 0.991, and 0.993 for DS1, DS2\_R1, DS3, DS4, and DS5, respectively. These findings again reinforce the strong predictive capabilities of the iECSA and its suitability for effective software reliability growth modeling.

#### 4.6.3.10 Benchmarking Results: Comparison with well-known algorithms

**Table 4.38:** Comparative MSE of iECSA and other Metaheuristic Algorithms for SRGM Parameter Estimation

Dataset	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AO	AOA	iECSA
DS-1	1793.76003	222.0619686	222.0750198	222.554258	441.475842	876.2813393	271.6464431	222.1482431	1107.731565	1795.937453	689.4712118	<b>222.0614259</b>
DS-2-R1	38.2703824	25.25638215	25.25648467	25.2563867	25.2563813	33.26444853	26.53313548	25.25984591	26.68642165	27.76591801	25.31468014	<b>25.25638133</b>
DS-2-R2	18.5708968	13.14193679	13.14220011	13.1419393	13.1419338	36.03961131	15.15197711	13.14828611	16.66267333	17.47519496	14.86028335	<b>13.14193376</b>
DS-2-R3	11.9502273	8.018679927	8.018885162	8.01868973	8.01867999	12.37108611	8.500845534	8.019613138	9.9423558	8.773072134	11.90277773	<b>8.018679082</b>
DS-2-R4	2.45973575	0.979900751	0.980005835	0.97990223	0.9799064	2.911764872	1.153821677	0.980515891	1.48571669	1.902874538	1.681959935	<b>0.979899526</b>
DS-3	5606.60569	2730.716004	2713.894292	2745.9164	2997.91746	5298.935145	2892.610232	2713.933269	3464.990595	5250.063533	7620.749634	<b>2713.218241</b>
DS-4	97.4479918	15.91199084	7.234672305	7.08019583	16.015029	28.60203195	7.194111755	7.064689205	34.49390423	87.94721833	11.79045859	<b>7.064082936</b>
DS-5	1427.52897	323.8196607	323.8209058	323.826085	323.82079	739.4149586	360.6362403	323.9117571	717.5924166	707.0692779	823.7600039	<b>323.8195631</b>
DS-6	180.343703	126.3272323	126.3362634	126.327402	130.399848	184.3556831	131.098337	126.3396429	133.2364133	180.391771	203.8756409	<b>126.3271687</b>
DS-7	473.953691	345.5127105	345.3033919	345.300451	365.491504	451.0891056	356.9659115	345.3176545	376.381344	429.2647964	458.6474753	<b>345.2993728</b>
Mean Rank	11.2	3.2	4.1	3.8	4.8	10.6	6.6	4.8	8.8	9.7	9.3	<b>1.1</b>

The MSE values achieved by different metaheuristic algorithms, including the proposed iECSA, are compared in Table 4.38. These algorithms were used to estimate the parameters of the Delayed S-Shaped model using various datasets. All algorithms were tested using population size of 30 and a maximum iterations of 50. It is observed that iECSA consistently outperforms other methods, as indicated by its lowest MSE values. This suggests that iECSA has a high level of precision in estimating parameters, which is further supported by its mean rank of 1.1. Additionally, there are other algorithms like DE and JAYA that show impressive results, with mean ranks of 3.2 and 4.1, respectively. In this context, it is worth mentioning that traditional algorithms like BAT and newer ones like AOA have higher mean

ranks, suggesting that they may be less accurate.

**Table 4.39:** Average VAF scores of iECSA and other Metaheuristic Algorithms for SRGM Parameter Estimation

Dataset	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AO	AOA	iECSA
DS-1	95.2762571	99.27067902	<b>99.27083511</b>	99.2690429	98.5834882	97.14087657	99.19921235	99.27049412	96.46108951	96.31810616	97.97641292	99.27066599
DS-2-R1	95.6911427	97.14077813	97.14088575	97.1407703	97.1408066	96.20651835	96.99596572	97.1403022	96.95863616	96.86649547	<b>97.1526829</b>	97.140806
DS-2-R2	98.6651021	99.05956334	<b>99.05974316</b>	99.0595799	99.0595564	97.41998209	98.95247313	99.05804179	98.80466127	98.83005861	98.94527839	99.05955577
DS-2-R3	97.2186126	<b>98.1091821</b>	98.10917794	98.1091815	98.1091817	97.1345105	98.04425617	98.10908802	97.66700037	98.00125038	97.24209404	98.10918194
DS-2-R4	98.7091081	99.47431565	99.47428646	99.4743146	99.4743122	98.45241132	99.4089447	99.47412947	99.20894505	99.15038257	99.11268929	<b>99.47431589</b>
DS-3	97.4015394	98.7355239	<b>98.74274149</b>	98.7300983	98.6062566	97.56198569	98.67831132	98.7426778	98.42194688	97.90521306	96.52189256	98.74223094
DS-4	78.3606385	95.93689058	98.13977741	98.1576912	95.8377981	92.63069747	98.13466222	<b>98.15993584</b>	91.32530193	80.06743237	97.40008643	98.15959569
DS-5	95.2954894	98.55139177	<b>98.55141318</b>	98.5513165	98.551411	96.78938096	98.4726046	98.55121411	96.79158514	97.89541113	96.57240567	98.55139889
DS-6	95.7081253	96.93143101	96.93197321	96.9315582	96.8417564	95.40987998	96.86072554	<b>96.9320472</b>	96.72346452	95.70403574	95.43415496	96.93178722
DS-7	92.9034718	94.67711919	<b>94.68269687</b>	94.6824703	94.4031559	93.12181563	94.55080446	94.68179713	94.1872969	93.7232105	93.05243797	94.6826203
Mean Rank	11.3	3.8	<b>2.3</b>	4	5.2	10.7	6.5	4.3	8.9	9.4	8.8	2.8

A comparison of the VAF scores achieved by several metaheuristic algorithms against iECSA is presented in Table 4.39. iECSA demonstrates satisfied performance, with an average rank of 2.8 (in the second place). The algorithms DE and JAYA are highly competitive, as seen by their mean ranks of 3.8 and 2.3, respectively. In summary, the VAF results indicate that iECSA, with its advanced adaptation mechanisms and integrated island-based methodology, offers a precise estimation of SRGM parameters. This confirms its potential as a dependable tool for software reliability analysis.

## Chapter Five: Discussion

---

This Chapter provides an in-depth discussion of the findings obtained from the extensive experiments conducted in Chapter 5. Moreover, this Chapter aims to establish a connection between the obtained results and the initial study objectives and consider their wider implications. Hence, the chapter is structured into three main sections. Section 5.1, "Discussion of Results," investigates the outcomes in detail and correlates the empirical data with the theoretical frameworks to reveal their significance. Section 5.2, "Implications," presents the potential academic and practical impacts of the research. Finally, Section 5.3, "Limitations," acknowledges the constraints and challenges encountered during the research. This reflective examination critiques the study's scope and methodologies, proposing areas for future research and highlighting opportunities for further inquiry.

### 5.1 Discussion of Results

This section provides an in-depth inspection of the results achieved in the experimentation with mathematical benchmarks and real-world applications. These discussions aim to link the experimental outcomes to the research objectives and questions outlined at the beginning of the study, thus providing a deeper understanding of how the enhancements and adaptations of the CSA and CapSA algorithms influence their performance in diverse optimization environments.

#### 5.1.1 Enhancements to CSA and CapSA

The experiments conducted on the 53 mathematical benchmark functions revealed significant improvements in the optimization capabilities of the enhanced variants of CSA and CapSA. These enhancements have proven especially effective in handling unimodal functions, demonstrating exceptional exploitation capabilities. Furthermore, the enhanced variants showed competitive performance in multimodal scenarios, indicating robust exploration abilities. This improvement is attributed to the integration of novel operators and



mechanisms that strategically enhance the algorithms' ability to balance exploration and exploitation, which is crucial in complex optimization landscapes.

### **5.1.2 Integration of Island-Based Model**

The integration of the island-based model into the basic and improved versions of CSA and CapSA has significantly strengthened their optimization capabilities. This approach segments the population into multiple sub-populations, facilitating more efficient exploration of the search space. The transfer of optimal solutions across different islands leads to a diverse set of solutions, which in turn improves the algorithms' global search capabilities and prevents premature convergence. The frequent information exchanges enabled by the migration mechanisms of the island model enhance its ability to maintain a balance between exploration and exploitation. It is crucial to maintain an optimal level of population diversity to prevent stagnation and boost overall search performance. The island-based models leverage this diversity to oscillate and maintain a high-quality population, which proves to be especially useful when dealing with complex optimization challenges.

### **5.1.3 Adaptive Island-Based Migration Policy**

It has been observed that island-based models are particularly sensitive to their migration parameters, such as migration frequency and rate. Finding the optimal settings for these parameters can be both challenging and time-consuming. After extensive experimentation, it was found that the migration rate significantly influences the performance of island-based models. To address these challenges, an adaptive migration policy was introduced. The use of an adaptive island-based migration policy has been a significant improvement over static policies. This approach allows for migration rate adjustments based on real-time optimization needs. The flexibility of this policy has proven to be beneficial in handling unpredictable optimization landscapes, as seen in the performance on the challenging CEC2014 benchmark functions. The ability to dynamically adjust migration rates without a lot of pre-tuning emphasizes its usefulness in real-world scenarios.

#### 5.1.4 Application to Real-World Problems

The effectiveness of the proposed models was further validated through their application to real-world problems such as neural network training, image segmentation, and software reliability growth modeling. It has been observed that in real-world applications, the proposed basic and enhanced variants, along with the island-based models, exhibit varied performance. The varying outcomes across different real applications highlight the challenge of achieving consistent improvements. For example, the enhanced variant of CSA does not demonstrate superior performance for neural network training and image segmentation compared to the original CSA, which shows good effectiveness. However, the adaptive-island-based models, particularly iECapSA and iECSA, provide promising results across all tested applications.

- **Neural Network Training:** Metaheuristic-based optimization, specifically iECapSA, has been shown to yield comparable or even better classification results than traditional gradient descent methods. These models excel at finding optimal weights and biases that minimize classification errors, leading to higher-quality classification outcomes. As a result, metaheuristic models can be useful tools for fine-tuning machine learning parameters and tackling complex classification tasks.
- **Image Segmentation:** The results showed that both basic and enhanced versions of the algorithms have their advantages in segmentation tasks, with enhanced versions such as iECapSA providing significant improvements in certain cases. In addition, applying the models to multilevel thresholding for image segmentation revealed that increasing the number of thresholds significantly enhances segmentation quality. iECapSA, in particular, used extra thresholds to improve segmentation details, which is vital for precise medical image analysis.
- **Software Reliability Growth Models:** In the application of SRGMs, it is evident that not all models are appropriate for all datasets. The nature of the data and its particular characteristics play a major role in choosing the right model. In most datasets, the delayed S-shaped model is the most popular choice among the tested models. This preference is supported by the model's consistent capacity to produce a better fit than

alternative models, as seen by its strong correlation coefficients, higher VAF, and MSE. These indicators show a strong correlation with the real data trends, which is essential for predicting software reliability effectively. Furthermore, the study results confirm the effectiveness of the island-based approaches, specifically the iECSA model, in accurately estimating SRGMs parameters. iECSA has demonstrated a superior ability to predict outcomes by efficiently benchmarking a set of SRGMs. The model's fast convergence and accurate parameter estimates closely match the actual data trends, confirming its exceptional performance.

To sum up, the experiments revealed that using the adaptive island model as a part of the metaheuristics framework leads to a high-performance search. This simply proves that static optimization should be replaced with responsive and dynamic techniques. This has been demonstrated by the upgraded CSA and CapSA as well as the newly introduced adaptive migration policy. Briefly, the results of the study help to develop theories about algorithmic design, but they also show the practical importance of the approach.

## **5.2 Implications**

### **5.2.1 Theoretical Implications**

The findings of this research contribute significantly to the theoretical foundations of optimization algorithms, especially in the field of MHs. Through the augmentation of the CSA and CapSA exploration and exploitation processes, this study not only advances our understanding of how these mechanisms influence algorithm performance, but also proves the practicality of incorporating adaptive strategies as part of these approaches. The successful implementation of adaptive island-based migration policies could offer a new perspective on managing population diversity, which might have an impact on the future theoretical developments in evolutionary computation and metaheuristic optimization.

### 5.2.2 Practical Implications

Regarding the practical implications, the enhancements and the novel adaptive migration policy proposed in this thesis would have direct impacts on the industries and fields where complex optimization problems are usually seen. For instance, one of the neural network training capabilities is the ability to optimize the process, resulting in more accurate and efficient predictive models useful in healthcare, finance, and other sectors that rely on data-driven decisions. In addition, the implementation of these enhanced algorithms in image segmentation, especially in medical imaging, for instance, the CT scans of COVID-19, is able to increase the accuracy and speed of diagnosis which in turn can directly impact patient care and treatment planning. The adaptation of these algorithms for SRGMs shows that better software development practices might be possible for developing more stable and reliable software systems.

### 5.3 Limitations

In this study, various enhancements and novel operators were integrated into CSA and CapSA, and an adaptive migration rate policy was introduced as a general framework aimed at enhancing both local and global search capabilities throughout the optimization process. Despite the extensive experimental work and promising results, it is crucial to acknowledge certain limitations:

1. **Scope of Tested Functions and Applications:** The effectiveness of the proposed models was demonstrated across specific datasets and mathematical functions. However, these results may not necessarily extend to all types of optimization problems, as the chosen functions and real-world applications may not cover the full spectrum of scenarios encountered in practice.
2. **Computational Complexity:** While the enhanced algorithms have shown improved performance, they also require more computational resources. This could be a limiting factor in scenarios where computational resources are constrained or in real-time applications where rapid response is crucial.

3. **Parameter Tuning and Migration Parameters:** Both the specific internal parameters of CSA and CapSA and the migration parameters are highly problem-dependent and require careful tuning for each targeted problem. This can be particularly challenging in large-scale or time-consuming real-world applications where dynamic adaptation of parameters is still limited. There is potential for further research in developing dynamic models that adjust island size, migration frequency, and migration policies more responsively. For instance, iECSA employs an adaptive migration rate but still relies on fixed settings for awareness probability, flight length, and the number of islands, which may not always yield optimal results in complex scenarios.
4. **Parallel Implementation Issues:** Achieving the goals of island-based metaheuristic models without excessive overhead is challenging. The primary aim of these models is to attain high-quality solutions efficiently, minimizing time overhead. This study focuses on achieving quality solutions, yet the implementation of island models in parallel and distributed environments introduces additional complexities that must be managed to optimize performance. To ensure the success of parallel implementations, it is important to consider various factors such as the mode of cooperation (direct vs. indirect), communication (synchronous vs. asynchronous), and the architecture of parallel environments. Additionally, aspects like speedup, efficiency, scalability, and the balance between computational load and communication overhead should be taken into account. Addressing these factors effectively can help maximize the benefits of parallelism while minimizing potential performance bottlenecks.
5. **Adaptation to Discrete and Binary Problems:** Most metaheuristics and island-based models are initially designed for continuous optimization problems. Adapting these models to handle binary problems, such as feature selection, represents a significant area for future research.
6. **Non-Deterministic Nature:** Metaheuristics by their nature are non-deterministic. This implies that the same algorithm might yield different outcomes under varying scenarios or initial configurations. That requires executing more testing in different contexts

in order to estimate the accuracy and reliability of the results.

These limitations indicate the need for continued research to address these challenges. We must improve the methods and extend the application of the proposed models so that they can provide reliable and robust results in different optimization scenarios.

## Chapter Six: Conclusion and Future Work

---

### 6.1 Summary and Conclusion

This thesis introduced noteworthy improvements in metaheuristic optimization algorithms. It focused on the improvement of CSA and CapSA algorithms, along with the integration of an adaptive island-based migration strategy. CSA algorithm has been enhanced with the introduction of adaptive tournament selection and a modified rule for generating random solutions, which makes it more efficient in exploiting the search space. CapSA has been reinforced with some refinements that include a modified follower update mechanism, an enhanced local best permutation strategy, and an adaptive dual update strategy. These improvements included capabilities to enhance the algorithms' exploitation and exploration potentials, increase their convergence rate, maintain diversity during the search, and increase the chances of finding the optimal solution.

The primary research question was to find out how these enhancements and the new migration policy could lead to the performance improvement of the algorithms across different optimization challenges. The proposed approaches were assessed using two comprehensive benchmark suites which had 53 functions in total. In addition to these theoretical tests, three practical applications were utilized: optimization of neural networks, thresholding for image segmentation, and the optimization of software reliability growth models. To further validate the efficacy of the proposed models, a comparative analysis was conducted against ten established metaheuristic algorithms from different categories, including BAT, DE, JAYA, PSO, SSA, WOA, SCA, GWO, HHO, and AOA. This comparison highlighted that the suggested iECSA and iECapSA variants provide extremely competitive results, often achieving the best outcomes across most cases.

The findings confirm that the proposed modifications substantially enhance the exploration and exploitation capabilities of the algorithms, enabling them to perform effectively on the involved wide range of optimization tasks. The research conducted provides new insights into the dynamics of metaheuristic algorithms in both theoretical and practical

contexts. By integrating adaptive elements and island-based modeling, the algorithms demonstrated improved performance metrics, suggesting a robust framework for tackling diverse and dynamic optimization problems. These enhancements contribute to the theoretical understanding of metaheuristic optimization and also offer practical implications by improving the efficiency and reliability of algorithms in real-world applications.

The findings ensure that the suggested amendments significantly improve the exploration and exploitation capabilities of both CSA and CapSA, which, in turn, allow them to perform effectively on the involved wide range of optimization tasks. The analysis conducted adds to the existing body of knowledge by examining the dynamics of metaheuristic algorithms from theoretical and practical perspectives. By incorporating adaptive elements and island-based modeling, the algorithms displayed better performance metrics, indicating a robust framework for handling diverse and dynamic optimization problems. These improvements contribute to the theoretical knowledge of metaheuristic optimization and also have practical implications by enhancing the efficiency and reliability of algorithms in real-world applications.

## **6.2 Recommendations and Future Work**

The successful application of adaptive strategies in island-based metaheuristic models has created a promising pathway for future investigation. Specifically, there is still scope for improvement by dynamically adapting model parameters. This can go beyond migration rates and include adjustments in variable island sizes, migration frequency, and communication topology. Such enhancements could greatly improve the models' efficiency, particularly in distributed computing environments, thus leading to reduced computational overhead. Exploring adaptive migratory behavior based on shared knowledge could further enhance exploration and exploitation capabilities, leading to better performance across various domains. Further exploration of parallelization techniques could also significantly decrease the time required for complex optimizations.

Building on these advancements, it is recommended that future research efforts generalize the findings by applying the enhanced algorithms to a broader range of problems, such



as those in emerging fields like deep learning, big data analytics, and scheduling techniques in cloud computing. This could refine the algorithms to enhance their applicability across diverse and evolving technological landscapes. Additionally, the development of discrete and binary variants of the models could enable new applications in feature selection and discrete optimization, broadening their utility. Moving forward, investigating the scalability of models like iECSA and iECapSA for large-scale optimization challenges and exploring hybridization approaches by combining it with other optimization techniques or problem-specific heuristics could leverage the strengths of different methodologies, creating synergistic effects that enhance overall optimization capabilities.

Based on the insights gained from this research, further recommendations can be made. For academic research, it is advised to continue exploring the island model and adaptation mechanisms within other metaheuristic frameworks and their influence on different types of problem sets. For industry practitioners, integrating such advanced algorithms into existing optimization tools can enhance their performance and reliability. Additionally, technology and engineering administrators might want to elevate research and development in advanced computational techniques, driven by the growing demand for problem-solving tools. Future research should also focus on the scalability of algorithms in larger, more complex systems to better understand their potential and limitations in real-world conditions.

## References

- [1] Jorge Nocedal and Stephen J. Wright. Numerical optimization. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.
- [2] E.K.P. Chong and S.H. Zak. An introduction to optimization. IEEE Antennas and Propagation Magazine, 38(2), 1996.
- [3] Andreas Antoniou and Wu-Sheng Lu. Practical Optimization: Algorithms and Engineering Applications. Springer Publishing Company, Incorporated, 1st edition, 2007.
- [4] Leandro dos Santos Coelho, Helon Vicente Hultmann Ayala, and Viviana Cocco Mariani. A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Applied Mathematics and Computation, 234:452–459, 2014.
- [5] Tobi Michael Alabi, Emmanuel I. Aghimien, Favour D. Agbajor, Zaiyue Yang, Lin Lu, Adebusola R. Adeoye, and Bhushan Gopaluni. A review on the integrated optimization techniques and machine learning approaches for modeling, prediction, and decision making on integrated energy systems. Renewable Energy, 194:822–849, 2022.
- [6] Majdi Mafarja, Ali Asghar Heidari, Maria Habib, Hossam Faris, Thaer Thaher, and Ibrahim Aljarah. Augmented whale feature selection for iot attacks: Structure, analysis and applications. Future Generation Computer Systems, 112:18–40, 2020.
- [7] Muath Sabha, Thaer Thaher, and Marwa M. Emam. Cooperative swarm intelligence algorithms for adaptive multilevel thresholding segmentation of covid-19 ct-scan images. JUCS - Journal of Universal Computer Science, 29(7):759–804, 2023.
- [8] Thaer Thaher, Mohammed Awad, Mohammed Aldasht, Alaa Sheta, Hamza Turabieh, and Hamouda Chantar. An enhanced evolutionary based feature selection approach

- using grey wolf optimizer for the classification of high-dimensional biological data. JUCS - Journal of Universal Computer Science, 28(5):499–539, 2022.
- [9] Thaer Thaher, Mahmoud Saheb, Hamza Turabieh, and Hamouda Chantar. Intelligent detection of false information in arabic tweets utilizing hybrid harris hawks based feature selection and machine learning models. Symmetry, 13(4), 2021.
- [10] Gui Zhou, Min jun Peng, and Hang Wang. Enhancing prediction accuracy for loca break sizes in nuclear power plants: A hybrid deep learning method with data augmentation and hyperparameter optimization. Annals of Nuclear Energy, 196:110208, 2024.
- [11] El-Ghazali Talbi. Metaheuristics: from design to implementation, volume 74. John Wiley & Sons, 2009.
- [12] Zhichao Lu, Ran Cheng, Yaochu Jin, Kay Chen Tan, and Kalyanmoy Deb. Neural architecture search as multiobjective optimization benchmarks: Problem formulation and performance assessment. IEEE transactions on evolutionary computation, 2023.
- [13] Bin Cao, Jianwei Zhao, Zhihan Lv, and Peng Yang. Diversified personalized recommendation optimization based on mobile data. IEEE Transactions on Intelligent Transportation Systems, 22(4):2133–2139, 2020.
- [14] H.A. Taha. Operations Research an Introduction. Pearson, 2017.
- [15] Sean Luke. Essentials of Metaheuristics. Lulu, second edition, 2013. Available for free at <http://cs.gmu.edu/~sean/book/metaheuristics/>.
- [16] Tansel Dokeroglu, Tayfun Kucukyilmaz, and El-Ghazali Talbi. Hyper-heuristics: A survey and taxonomy. Computers & Industrial Engineering, 187:109815, 2024.
- [17] Bin Cao, Jianwei Zhao, Yu Gu, Yingbiao Ling, and Xiaoliang Ma. Applying graph-based differential grouping for multiobjective large-scale optimization. Swarm and Evolutionary Computation, 53:100626, 2020.
- [18] Yuzhu Duan, Yiyi Zhao, and Jiangping Hu. An initialization-free distributed algorithm for dynamic economic dispatch problems in microgrid: Modeling, optimization

- and analysis. Sustainable Energy, Grids and Networks, 34:101004, 2023.
- [19] Zhihan Lv, Jingyi Wu, Yuxi Li, and Houbing Song. Cross-layer optimization for industrial internet of things in real scene digital twins. IEEE Internet of Things Journal, 9(17):15618–15629, 2022.
- [20] Guohua Wu. Across neighborhood search for numerical optimization. Information Sciences, 329:597–618, 2016. Special issue on Discovery Science.
- [21] Chao Lu, Jun Zheng, Lvjiang Yin, and Renyi Wang. An improved iterated greedy algorithm for the distributed hybrid flowshop scheduling problem. Engineering Optimization, pages 1–19, 2023.
- [22] Dongqing Luan, Along Liu, Xiaoli Wang, Yanxi Xie, and Zhong Wu. Robust two-stage location allocation for emergency temporary blood supply in postdisaster. Discrete dynamics in nature and society, 2022:1–20, 2022.
- [23] Yiyong Xiao and Abdullah Konak. The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion. Transportation Research Part E: Logistics and Transportation Review, 88:146–166, 2016.
- [24] Bin Cao, Jianwei Zhao, Yu Gu, Shanshan Fan, and Peng Yang. Security-aware industrial wireless sensor network deployment optimization. IEEE transactions on industrial informatics, 16(8):5309–5316, 2019.
- [25] Ali Asghar Heidari, Mehdi Akhoondzadeh, and Huiling Chen. A wavelet pm2. 5 prediction system using optimized kernel extreme learning with boruta-xgboost feature selection. Mathematics, 10(19):3566, 2022.
- [26] Jiazhao Zhang, Yijie Tang, He Wang, and Kai Xu. Asro-dio: Active subspace random optimization based depth inertial odometry. IEEE Transactions on Robotics, 39(2):1496–1508, 2022.
- [27] Jinchao Xu, Boyu Mu, Luwei Zhang, Rong Chai, Yanfu He, and Xiaoshuan Zhang. Fabrication and optimization of passive flexible ammonia sensor for aquatic supply

- chain monitoring based on adaptive parameter adjustment artificial neural network (apa-ann). Computers and Electronics in Agriculture, 212:108082, 2023.
- [28] Fevrier Valdez, Juan Carlos Vazquez, Patricia Melin, and Oscar Castillo. Comparative study of the use of fuzzy logic in improving particle swarm optimization variants for mathematical functions using co-evolution. Applied Soft Computing, 52:1070–1083, 2017.
- [29] Kai Zhang, Zhongzheng Wang, Guodong Chen, Liming Zhang, Yongfei Yang, Chuanjin Yao, Jian Wang, and Jun Yao. Training effective deep reinforcement learning agents for real-time life-cycle production optimization. Journal of Petroleum Science and Engineering, 208:109766, 2022.
- [30] Songwei Zhao, Pengjun Wang, Ali Asghar Heidari, Xuehua Zhao, and Huiling Chen. Boosted crow search algorithm for handling multi-threshold image problems with application to x-ray images of covid-19. Expert Systems with Applications, 213:119095, 2023.
- [31] Thaer Thaher, Hamouda Chantar, Jingwei Too, Majdi Mafarja, Hamza Turabieh, and Essam H. Houssein. Boolean particle swarm optimization with various evolutionary population dynamics approaches for feature selection problems. Expert Systems with Applications, 195:116550, 2022.
- [32] Hui ling Chen, Bo Yang, Su jing Wang, Gang Wang, Da you Liu, Huai zhong Li, and Wen bin Liu. Towards an optimal support vector machine classifier using a parallel particle swarm optimization strategy. Applied Mathematics and Computation, 239:180–197, 2014.
- [33] Helong Yu, Kang Yuan, Wenshu Li, Nannan Zhao, Weibin Chen, Changcheng Huang, Huiling Chen, and Mingjing Wang. Improved butterfly optimizer-configured extreme learning machine for fault diagnosis. Complexity, 2021:1–17, 02 2021.
- [34] Mohammed Azmi Al-Betar, Mohammed A. Awadallah, Sharif Naser Makhadmeh, Iyad Abu Doush, Raed Abu Zitar, Samah Alshathri, and Mohamed Abd

- Elaziz. A hybrid harris hawks optimizer for economic load dispatch problems. Alexandria Engineering Journal, 2022.
- [35] GuoChun Wang, Wenyong Gui, Guoxi Liang, Xuehua Zhao, Mingjing Wang, Majdi Mafarja, Hamza Turabieh, Junyi Xin, Huiling Chen, Xinsheng Ma, and Yanxia Sun. Spiral motion enhanced elite whale optimizer for global tasks. Complex., 2021, jan 2021.
- [36] Ali Asghar Heidari, Rahim Ali Abbaspour, and Ahmad Rezaee Jordehi. An efficient chaotic water cycle algorithm for optimization tasks. Neural Computing and Applications, 28(1):57–85, 2017.
- [37] Yuanzhou Zheng, Xuemeng Lv, Long Qian, and Xinyu Liu. An optimal bp neural network track prediction method based on a ga–aco hybrid algorithm. Journal of Marine Science and Engineering, 10(10):1399, Sep 2022.
- [38] Zhonglai Wang, Dongyu Zhao, and Yi Guan. Flexible-constrained time-variant hybrid reliability-based design optimization. Structural and Multidisciplinary Optimization, 66(4):89, 2023.
- [39] Ali Asghar Heidari, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 97:849 – 872, 2019.
- [40] Xizheng Zhang, Zeyu Wang, and Zhangyu Lu. Multi-objective load dispatch for microgrid with electric vehicles using modified gravitational search and particle swarm optimization algorithm. Applied Energy, 306:118018, 2022.
- [41] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. Information Sciences, 237:82 – 117, 2013. Prediction, Control and Diagnosis using Advanced Neural Computations.
- [42] Fevrier Valdez, Juan Carlos Vazquez, Patricia Melin, and Oscar Castillo. Comparative study of the use of fuzzy logic in improving particle swarm optimization variants for mathematical functions using co-evolution. Applied Soft Computing, 52:1070–1083,

- 2017.
- [43] Aminu Aminu Muazu, Ahmad Sobri Hashim, and Aliza Sarlan. Review of nature inspired metaheuristic algorithm selection for combinatorial t-way testing. IEEE Access, 10:27404–27431, 2022.
- [44] 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, 56:13187–13257, 04 2023.
- [45] Alireza Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures, 169:1–12, 2016.
- [46] Malik Braik, Alaa Sheta, and Heba Al-Hiary. A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural Computing and Applications, 33(7):2515–2547, 09 2021.
- [47] El-Ghazali Talbi. Parallel Combinatorial Optimization. 04 2006.
- [48] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel metaheuristics: Recent advances and new trends. International Transactions in Operational Research, 20:1–48, 01 2012.
- [49] Arthur Corcoran and Roger Wainwright. A parallel island model genetic algorithm for the multiprocessor scheduling problem. Selected Areas in Cryptography, 07 1995.
- [50] Ting Yee Lim. Structured population genetic algorithms: A literature survey. Artificial Intelligence Review, 41:385–399, 03 2014.
- [51] Mônica Pais, Igor Peretta, Keiji Yamanaka, and Edmilson Pinto. Factorial design analysis applied to the performance of parallel evolutionary algorithms. Journal of the Brazilian Computer Society, 20, 02 2014.
- [52] Mohammed Al-Betar, Mohammed Awadallah, Iyad Doush, Abdelaziz Hammouri, Majdi Mafarja, and Zaid Alyasseri. Island flower pollination algorithm for global optimization. The Journal of Supercomputing, 75, 08 2019.

- [53] Mohammed Awadallah, Mohammed Al-Betar, Asaju Bolaji, Iyad Doush, Abdelaziz Hammouri, and Majdi Mafarja. Island artificial bee colony for global optimization. Soft Computing, 24, 09 2020.
- [54] Mohammed Al-Betar, Mohammed Awadallah, Hossam Faris, Xin-She Yang, Ahamad Tajudin Khader, and Osama Alomari. Bat-inspired algorithms with natural selection mechanisms for global optimization. Neurocomputing, 273:448–465, 08 2017.
- [55] Bilal Abed-alguni, Ahmad Klaib, and Khalid Nahar. Island-based whale optimization algorithm for continuous optimization problems. International Journal of Reasoning-based Intelligent Systems, 11:319–329, 01 2019.
- [56] Bilal H. Abed-alguni and Malek Barhoush. Distributed grey wolf optimizer for numerical optimization problems. IJRIS, 4, 2018.
- [57] Mohammed Azmi Al-Betar and Mohammed A. Awadallah. Island bat algorithm for optimization. Expert Systems with Applications, 107:126–145, 2018.
- [58] Jun-ichi Kushida, Akira Hara, Tetsuyuki Takahama, and Ayumi Kido. Island-based differential evolution with varying subpopulation size. In 2013 IEEE 6th International Workshop on Computational Intelligence and Applications (IWCIA), pages 119–124, 2013.
- [59] Thaer Thaher and Badie Sartawi. An experimental design approach to analyse the performance of island-based parallel artificial bee colony algorithm. In 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), pages 1–7, 2020.
- [60] Alfian Akbar Gozali and Shigeru Fujimura. Localized island model genetic algorithm in population diversity preservation. In Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018), pages 122–128. Atlantis Press, 2019/03.



- [61] Ali Wagdy Mohamed, Anas A. Hadi, and Ali Khater Mohamed. Differential evolution mutations: Taxonomy, comparison and convergence analysis. IEEE Access, 9:68629–68662, 2021.
- [62] Abdelouahab Necira, Naimi Djemai, Salhi Ahmed, Souhail Salhi, and Smail Menani. Dynamic crow search algorithm based on adaptive parameters for large-scale global optimization. Evolutionary Intelligence, 15:2153–2169, 09 2022.
- [63] Chiwen Qu and Yanming Fu. Crow search algorithm based on neighborhood search of non-inferior solution set. IEEE Access, 7:52871–52895, 2019.
- [64] Teodor Gabriel Crainic. Parallel Metaheuristic Search, pages 809–847. Springer International Publishing, Cham, 2018.
- [65] Majdi M Mafarja and Seyedali Mirjalili. Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing, 260:302–312, 2017.
- [66] Fred Glover and Gary Kochenberger. Handbook of metaheuristics. Kluwer Academic Publishers, Boston, 2003, 57, 01 2003.
- [67] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. ACM Comput. Surv., 45(3), jul 2013.
- [68] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671–680, 1983.
- [69] John H. Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge, MA, USA, 1992.
- [70] Bernardo Morales-Castañeda, Daniel Zaldívar, Erik Cuevas, Fernando Fausto, and Alma Rodríguez. A better balance in metaheuristic algorithms: Does it exist? Swarm and Evolutionary Computation, 54:100671, 2020.
- [71] Xin-She Yang, Suash Deb, and Simon Fong. Metaheuristic algorithms: Optimal balance of intensification and diversification. Applied Mathematics & Information Sciences, 8:977–983, 08 2013.

- [72] Mohammed Eshtay, Hossam Faris, and Nadim Obeid. Metaheuristic-based extreme learning machines: a review of design formulations and applications. International Journal of Machine Learning and Cybernetics, 10:1543–1561, 06 2019.
- [73] Fred Glover. Tabu search - part i. INFORMS Journal on Computing, 2:4–32, 01 1990.
- [74] N. Mladenović and P. Hansen. Variable neighborhood search. Computers & Operations Research, 24(11):1097 – 1100, 1997.
- [75] Thomas Feo and Mauricio Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109–133, 03 1995.
- [76] Th. G. Stützle. Local search algorithms for combinatorial problems: analysis, improvements, and new applications. Dissertations in Artificial Intelligence-Infix, 220:203, 1999.
- [77] C. Voudouris. Guided Local search for combinatorial optimization problems. PhD thesis, 07 1997.
- [78] Daniel Molina, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain, and Francisco Herrera. Comprehensive taxonomies of nature- and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis and recommendations. Cognitive Computation, 12:897–939, 2020.
- [79] Nazmul Siddique and Hojjat Adeli. Nature inspired computing: An overview and some future directions. Cognitive Computation, 7:706–714, 12 2015.
- [80] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. Advances in Engineering Software, 95:51 – 67, 2016.
- [81] Abdelazim G. Hussien, Mohamed Amin, Mingjing Wang, Guoxi Liang, Ahmed Alsanad, Abdu Gumaei, and Huiling Chen. Crow search algorithm: Theory, recent advances, and applications. IEEE Access, 8:173548–173565, 2020.
- [82] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, Robots and Biological Systems:

- Towards a New Bionics?, pages 703–712, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
- [83] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 10 1999.
- [84] Mohd Nadhir Ab Wahab, Samia Nefti-meziani, and Adham Atyabi. A comprehensive review of swarm optimization algorithms. PloS one, 10:e0122827, 05 2015.
- [85] Kashif Hussain, Mohd Salleh, Shi Cheng, and Yuhui Shi. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Computing and Applications, 31:7665–7683, 11 2019.
- [86] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pages 39–43, 1995.
- [87] Jagdish Bansal. Particle Swarm Optimization, pages 11–23. 01 2019.
- [88] M. Dorigo, V. Maniezzo, and A. Coloni. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1):29–41, 1996.
- [89] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization, technical report - tr06. Technical Report, Erciyes University, 01 2005.
- [90] Li Xiao. An optimizing method based on autonomous animats: Fish-swarm algorithm. Systems Engineering - Theory & Practice, 2002.
- [91] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine, 22(3):52–67, 2002.
- [92] Ali Asghar Heidari, Seyedali Mirjalili, Hossam Faris, Ibrahim Aljarah, Majdi Mafarja, and Huiling Chen. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 97:849 – 872, 2019.

- [93] M. Khishe and M.R. Mosavi. Chimp optimization algorithm. Expert Systems with Applications, 149:113338, 2020.
- [94] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. Advances in Engineering Software, 69:46 – 61, 2014.
- [95] Seyedali Mirjalili. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 89:228 – 249, 2015.
- [96] Afshin Faramarzi, Mohammad Heidarinejad, Seyedali Mirjalili, and A. H. Gandomi. Marine predators algorithm: A nature-inspired metaheuristic. Expert Systems with Applications, 152:113377, 2020.
- [97] Shimin Li, Huiling Chen, Mingjing Wang, Ali Asghar Heidari, and Seyedali Mirjalili. Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems, 111:300 – 323, 2020.
- [98] Malik Braik, Alaa Sheta, and Heba Al-Hiary. A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural Computing and Applications, 07 2020.
- [99] Ingo Rechenberg. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart, Germany, 1973.
- [100] John Koza. Genetic programming: On the programming of computers by means of natural selection. Complex Adap. Syst., 1, 01 1992.
- [101] Rainer Storn and Kenneth V. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341–359, 1997.
- [102] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, Caltech Concurrent Computation Program (C3P Report 826), Pasadena, CA, 1989.

- [103] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 3(2):82–102, 1999.
- [104] Candida Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. Complex Systems, 13(2):87–129, 2001.
- [105] Dan Simon. Biogeography-based optimization. IEEE Transactions on Evolutionary Computation, 12(6):702–713, 2008.
- [106] Robert G. Reynolds. An introduction to cultural algorithms. In Proceedings of the 3rd Annual Conference on Evolutionary Programming, pages 131–139, 1994.
- [107] Ravipudi V Rao, Vimal J Savsani, and DP Vakharia. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design, 43(3):303–315, 2011.
- [108] Yuhui Shi. Brain storm optimization algorithm. In Advances in Swarm Intelligence: Second International Conference, ICSI 2011, Chongqing, China, June 12-15, 2011, Proceedings, Part I, pages 303–309. Springer, Berlin, Heidelberg, 2011.
- [109] Luna Mingyi Zhang, Cheyenne Dahlmann, and Yanqing Zhang. Human-inspired algorithms for continuous function optimization. In 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, volume 1, pages 318–321, 2009.
- [110] Reza Moghdani and Khodakaram Salimifard. Volleyball premier league algorithm. Applied Soft Computing, 64:161–185, 2018.
- [111] Ali Wagdy, Anas Hadi, and Ali Khater. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. International Journal of Machine Learning and Cybernetics, 11:1501–1529, 07 2020.
- [112] Junbo Lian and Guohua Hui. Human evolutionary optimization algorithm. Expert Systems with Applications, 241:122638, 2024.
- [113] Richard Formato. Central force optimization: A new metaheuristic with applications in applied electromagnetics. Progress In Electromagnetics Research, 77:425–491, 01

2007.

- [114] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. Gsa: a gravitational search algorithm. Information sciences, 179(13):2232–2248, 2009.
- [115] A. Kaveh and S. Talatahari. A novel heuristic optimization method: charged system search. Acta Mechanica, 213(3-4):267–289, 2010.
- [116] Seyedali Mirjalili, Seyed Mirjalili, and Abdolreza Hatamlou. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications, 27:495–513, 03 2015.
- [117] Fatma A. Hashim, Essam H. Houssein, Mai S. Mabrouk, Walid Al-Atabany, and Seyedali Mirjalili. Henry gas solubility optimization: A novel physics-based algorithm. Future Generation Computer Systems, 101:646–667, 2019.
- [118] A. Kaveh and A. Dadras. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Advances in Engineering Software, 110:69–84, 2017.
- [119] A. Kaveh and M. Khayatizad. A new meta-heuristic method: Ray optimization. Computers & Structures, 112-113:283–294, 2012.
- [120] Hang Su, Dong Zhao, Ali Asghar Heidari, Lei Liu, Xiaoqin Zhang, Majdi Mafarja, and Huiling Chen. Rime: A physics-based optimization. Neurocomputing, 532:183–214, 2023.
- [121] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), volume 2, pages 1470–1477. IEEE, 1999.
- [122] James Kennedy and Russell Eberhart. Particle swarm optimization. In Proceedings of ICNN’95-international conference on neural networks, volume 4, pages 1942–1948. IEEE, 1995.
- [123] Derviş Karaboğa and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization, 39:459–471, 2007.

- [124] Xin-She Yang. Firefly algorithms for multimodal optimization. In Stochastic algorithms: foundations and applications. SAGA 2009. Lecture Notes in Computer Science, volume 5792, pages 169–178. Springer, Berlin, Heidelberg, 2009.
- [125] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), pages 210–214. IEEE, 2009.
- [126] Xin-She Yang and Amir Gandomi. Bat algorithm: A novel approach for global engineering optimization. Engineering Computations, 29:464–483, 11 2012.
- [127] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. Advances in Engineering Software, 69:46–61, 2014.
- [128] Seyedali Mirjalili. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 89:228–249, 2015.
- [129] Seyedali Mirjalili. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4):1053–1073, 2016.
- [130] Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. Grasshopper optimisation algorithm: Theory and application. Advances in Engineering Software, 105:30–47, 2017.
- [131] Yutao Yang, Huiling Chen, Ali Asghar Heidari, and Amir H Gandomi. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. Expert Systems with Applications, 177:114864, 2021.
- [132] Boli Zheng, Yi Chen, Chaofan Wang, Ali Asghar Heidari, Lei Liu, and Huiling Chen. The moss growth optimization (MGO): concepts and performance. Journal of Computational Design and Engineering, 11(5):184–221, 09 2024.
- [133] Ahmad Ghiaskar, Amir Amiri, and Seyedali Mirjalili. Polar fox optimization algorithm: a novel meta-heuristic algorithm. Neural Computing and Applications, 36:20983–21022, 08 2024.

- [134] Teodor Crainic. Parallel Metaheuristics and Cooperative Search, pages 419–451. Springer International Publishing, Cham, 2019.
- [135] Neha Khanduja and Bharat Bhushan. Recent advances and application of metaheuristic algorithms: A survey (2014–2020). Metaheuristic and Evolutionary Computation: Algorithms and Applications, pages 207–228, 2021.
- [136] Fred W Glover and Gary A Kochenberger. Handbook of metaheuristics, volume 57. Springer Science & Business Media, 2006.
- [137] Kai Hwang. Advanced Computer Architecture: Parallelism, Scalability, Programmability. 01 1993.
- [138] J. Kennedy. Stereotyping: improving particle swarm performance with cluster analysis. In Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), volume 2, pages 1507–1512 vol.2, 2000.
- [139] Roberto Battiti and Giampietro Tecchiolli. Parallel biased search for combinatorial optimization: genetic algorithms and tabu. Microprocessors and Microsystems, 16(7):351 – 367, 1992.
- [140] Huub M. M. ten Eikelder, Bas J. M. Aarts, Marco G. A. Verhoeven, and Emile H. L. Aarts. Sequential and Parallel Local Search Algorithms for Job Shop Scheduling, pages 359–371. Springer US, Boston, MA, 1999.
- [141] Teodor Gabriel Crainic and Michel Toulouse. Explicit and emergent cooperation schemes for search algorithms. In Vittorio Maniezzo, Roberto Battiti, and Jean-Paul Watson, editors, Learning and Intelligent Optimization, pages 95–109, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [142] Bilal Abed-alguni, Ahmad Klaib, and Khalid Nahar. Island-based whale optimization algorithm for continuous optimization problems. International Journal of Reasoning-based Intelligent Systems, 11:319–329, 01 2019.



- [143] Marco Tomassini. Spatially Structured Evolutionary Algorithms Artificial Evolution in Space and Time. 01 2005.
- [144] Lucas A. Da Silveira, Jose L. Soncco-Alvarez, Thaynara A. De Lima, and Mauricio Ayala-Rincon. Heterogeneous parallel island models. In 2021 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1–8, 2021.
- [145] Lourdes Araujo and Juan Julián Merelo. Diversity through multiculturalism: Assessing migrant choice policies in an island model. IEEE Transactions on Evolutionary Computation, 15(4):456–469, 2011.
- [146] M. Ruciński, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. Parallel Computing, 36(10):555–571, 2010. Parallel Architectures and Bioinspired Algorithms.
- [147] Guo Sun, Yiqiao Cai, Tian Wang, Hui Tian, Cheng Wang, and Yonghong Chen. Differential evolution with individual-dependent topology adaptation. Information Sciences, 450:1–38, 2018.
- [148] Francisco Vega, Marco Tomassini, and Leonardo Vanneschi. An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines, 4:21–51, 03 2003.
- [149] Teodor Gabriel. Parallel Meta-heuristic Search, pages 1–39. Springer International Publishing, Cham, 2016.
- [150] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel metaheuristics: recent advances and new trends. International Transactions in Operational Research, 20(1):1–48, 2013.
- [151] Zbigniew Skolicki and Kenneth De Jong. The influence of migration sizes and intervals on island models. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05, page 1295–1302, New York, NY, USA, 2005. Association for Computing Machinery.
- [152] Erik Cuevas, Emilio Barocio, and Arturo Conde. A Modified Crow Search Algorithm

- with Applications to Power System Problems, pages 137–166. 01 2019.
- [153] Malik Braik, Hussein Al-Zoubi, Mohammad Ryalat, Alaa Sheta, and Omar Alzubi. Memory based hybrid crow search algorithm for solving numerical and constrained global optimization problems. Artificial Intelligence Review, pages 1–73, 03 2022.
- [154] Tingting Wang, Chengming Zhang, Ankun He, and Wei Dong. Overview of crow search algorithm. Journal of Physics: Conference Series, 2258(1):012017, apr 2022.
- [155] Hossam Hassan Ali, Ahmed Fathy, Mujahed Al-Dhaifallah, Almoataz Y. Abdelaziz, and Mohamed Ebeed. An efficient capuchin search algorithm for extracting the parameters of different pv cells/modules. Frontiers in Energy Research, 10, 2022.
- [156] Mohammed A.A. Al-qaness, Ahmed A. Ewees, Hong Fan, Laith Abualigah, Ammar H. Elsheikh, and Mohamed Abd Elaziz. Wind power prediction using random vector functional link network with capuchin search algorithm. Ain Shams Engineering Journal, 14(9):102095, 2022.
- [157] Sivakumar Ramu, Rameshkumar Ranganathan, and Ramakrishnan Ramamoorthy. Capuchin search algorithm based task scheduling in cloud computing environment. Yanbu Journal of Engineering and Science, 19(1):18–29, 3 2022.
- [158] Milad Mohseni, Fatemeh Amirghafouri, and Behrouz Pourghebleh. Cedar: A cluster-based energy-aware data aggregation routing protocol in the internet of things using capuchin search algorithm and fuzzy logic. Peer-to-Peer Networking and Applications, 16:189–209, 10 2022.
- [159] Malik Braik, Mohammed A. Awadallah, Mohammed Azmi Al-Betar, and Abdelaziz I. Hammouri. A hybrid capuchin search algorithm with gradient search algorithm for economic dispatch problem. Soft Comput., 27(22):16809–16841, August 2023.
- [160] Thaer Thaher, Mohammed Awad, Alaa Sheta, and Mohammed Aldasht. Enhanced capuchin search algorithm using cooperative island model with application of evolutionary feedforward neural networks. In 2023 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS), pages 237–245,

2023.

- [161] Ivette Miramontes, Patricia Melin, and German Prado-Arechiga. Fuzzy system for classification of nocturnal blood pressure profile and its optimization with the crow search algorithm. In Soft Computing Applications: Proceedings of the 8th International Workshop Soft Computing Applications (SOFA 2018), Vol. II 8, pages 23–34. Springer, 2021.
- [162] Niam Abdulmunim Al-Thanoon, Zakariya Yahya Algamal, and Omar Saber Qasim. Feature selection based on a crow search algorithm for big data classification. Chemometrics and Intelligent Laboratory Systems, 212:104288, 2021.
- [163] Abhilasha Chaudhuri and Tirath Prasad Sahu. Feature selection using binary crow search algorithm with time varying flight length. Expert Systems with Applications, 168:114288, 2021.
- [164] ibrahim Eliguzel and Eren ozceylan. Application of an improved discrete crow search algorithm with local search and elitism on a humanitarian relief case. Artificial Intelligence Review, 54:1–27, 08 2021.
- [165] Luis Fernando Grisales-Noreña, Brandon Cortés-Caicedo, Gerardo Alcalá, and Oscar Danilo Montoya. Applying the crow search algorithm for the optimal integration of pv generation units in dc networks. Mathematics, 11(2):387, 2023.
- [166] Malik Braik, Hussein Al-Zoubi, Mohammad Ryalat, Alaa Sheta, and Omar Alzubi. Memory based hybrid crow search algorithm for solving numerical and constrained global optimization problems. Artificial Intelligence Review, 56(1):27–99, 2023.
- [167] Sanjay Kumar, Abhishek Mallik, and Sandeep Singh Sengar. Community detection in complex networks using stacked autoencoders and crow search algorithm. The Journal of Supercomputing, 79(3):3329–3356, 2023.
- [168] Yang Bai, Li Cao, Binhe Chen, Yaodan Chen, and Yinggao Yue. A novel topology optimization protocol based on an improved crow search algorithm for the perception layer of the internet of things. Biomimetics, 8(2):165, 2023.

- [169] Jieguang He, Zhiping Peng, Lei Zhang, Liyun Zuo, Delong Cui, and Qirui Li. Enhanced crow search algorithm with multi-stage search integration for global optimization problems. Soft Computing, pages 1–31, 2023.
- [170] Pushpendra Singh, Rajesh Arya, LS Titare, Pradeep Purey, and LD Arya. Value aided optimal load shedding accounting voltage stability consideration employing crow search algorithm with modification based on lampinen’s criterion. Applied Soft Computing, 143:110391, 2023.
- [171] Yundi Rao, Dengxu He, and Liangdong Qu. A probabilistic simplified sine cosine crow search algorithm for global optimization problems. Engineering with Computers, 39(3):1823–1841, 2023.
- [172] Zhao Liu, Wenjie Wang, Guohong Shi, and Ping Zhu. A modified crow search algorithm based on group strategy and adaptive mechanism. Engineering Optimization, pages 1–19, 2023.
- [173] Jeremiah Osei-kwakye, Fei Han, Alfred Adutwum Amponsah, Qing-Hua Ling, and Timothy Apasiba Abeo. A diversity enhanced hybrid particle swarm optimization and crow search algorithm for feature selection. Applied Intelligence, pages 1–26, 2023.
- [174] Cenk Andic, Ali Ozturk, and Belgin Turkey. Power system state estimation using a robust crow search algorithm based on pmus with limited number of channels. Electric Power Systems Research, 217:109126, 2023.
- [175] Jieguang He, Zhiping Peng, Lei Zhang, Liyun Zuo, Delong Cui, and Qirui Li. Enhanced crow search algorithm with multi-stage search integration for global optimization problems. Soft Computing, pages 1–31, 06 2023.
- [176] Jafar Gholami, Farhad Mardukhi, and Hossam Zawbaa. An improved crow search algorithm for solving numerical optimization functions. Soft Computing, 25:1–14, 07 2021.
- [177] Yukai Ke, Jun Xie, and Somayeh Pouramini. Utilization of an improved crow

- search algorithm to solve building energy optimization problems: Cases of australia. Journal of Building Engineering, 38:102142, 2021.
- [178] Alaa Sheta, Malik Braik, Heba Al-Hiary, and Seyedali Mirjalili. Improved versions of crow search algorithm for solving global numerical optimization problems. Applied Intelligence, 53:1–45, 08 2023.
- [179] Li Cao, Yinggao Yue, Yong Zhang, and Yong Cai. Improved crow search algorithm optimized extreme learning machine based on classification algorithm and application. IEEE Access, 9:20051–20066, 2021.
- [180] Soheyl Khalilpourazari and Seyed Hamid Reza Pasandideh. Sine–cosine crow search algorithm: Theory and applications. Neural Comput. Appl., 32(12):7725–7742, jun 2020.
- [181] Rizk M. Rizk-Allah, Aboul Ella Hassanien, and Siddhartha Bhattacharyya. Chaotic crow search algorithm for fractional optimization problems. Applied Soft Computing, 71:1161–1175, 2018.
- [182] Mohit Jain, Asha Rani, and Vijander Singh. An improved crow search algorithm for high-dimensional problems. Journal of Intelligent & Fuzzy Systems, 33:3597–3614, 11 2017.
- [183] Mohammed A. Awadallah, Mohammed Azmi Al-Betar, Iyad Abu Doush, Sharif Naser Makhadmeh, Zaid Abdi Alkareem Alyasseri, Ammar Kamal Abasi, and Osama Ahmad Alomari. Ccsa: Cellular crow search algorithm with topological neighborhood shapes for optimization. Expert Systems with Applications, 194:116431, 2022.
- [184] Mohamed Elsayed Abd Elaziz, Souadfel Salima, and Rehab Ibrahim. Boosting capuchin search with stochastic learning strategy for feature selection. Neural Computing and Applications, pages 1–20, 03 2023.
- [185] Malik Braik, Abdelaziz Hammouri, Hussein Alzoubi, and Alaa Sheta. Feature selection based nature inspired capuchin search algorithm for solving classification prob-

- lems. Expert Systems with Applications, 235:121128, 2024.
- [186] Hossein Asgharzadeh, Ali Ghaffari, Mohammad Masdari, and Farhad Soleimani Gharehchopogh. Anomaly-based intrusion detection system in the internet of things using a convolutional neural network and multi-objective enhanced capuchin search algorithm. Journal of Parallel and Distributed Computing, 175:1–21, 2023.
- [187] Shujing Li, Zhangfei Li, Qinghe Li, Mingyu Zhang, and Linguo Li. Hybrid improved capuchin search algorithm for plant image thresholding. Frontiers in plant science, 14:1122788, 2023.
- [188] T.D. Subha and C. Arunachalaperumal. A hybrid capsa-who method used for performance enhancement of tandem perovskite solar cell. Expert Systems with Applications, 230:120554, 2023.
- [189] Amjad Qtaish, Malik Braik, Dheeb Albashish, Mohammad Alshammari, Abdulrahman Alreshidi, and Eissa Alreshidi. Optimization of k-means clustering method using hybrid capuchin search algorithm. The Journal of Supercomputing, 80:1–60, 07 2023.
- [190] J.G. Digalakis and K.G. Margaritis. On benchmarking functions for genetic algorithms. International Journal of Computer Mathematics, 77(4):481–506, 2001.
- [191] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization. J. Heuristics, 15:617–644, 12 2009.
- [192] Xiaodong Li, Andries Engelbrecht, and M. G. Epitropakis. Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, Melbourne, Australia, 2013.
- [193] Mohammed Azmi Al-Betar, Mohammed A. Awadallah, Ahamad Tajudin Khader, and Zahraa Adnan Abdalkareem. Island-based harmony search for optimization problems.

- Expert Systems with Applications, 42(4):2026 – 2035, 2015.
- [194] Mohammed Azmi Al-Betar and Mohammed A. Awadallah. Island bat algorithm for optimization. Expert Systems with Applications, 107:126 – 145, 2018.
- [195] Bilal H. Abed-alguni. Island-based cuckoo search with highly disruptive polynomial mutation. International journal of artificial intelligence, 17:57–82, 2019.
- [196] Raj Jain. The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling, NY: Wiley. 04 1991.
- [197] Jun-ichi Kushida, Akira Hara, Tetsuyuki Takahama, and Ayumi Kido. Island-based differential evolution with varying subpopulation size. In 2013 IEEE 6th International Workshop on Computational Intelligence and Applications (IWCIA), pages 119–124, 2013.
- [198] Mert Sinan Turgut, Oguz Emrah Turgut, and Deniz Türsel Eliiyi. Island-based crow search algorithm for solving optimal control problems. Applied Soft Computing, 90:106170, 2020.
- [199] Alfian Gozali. Dm-limga: Dual migration localized island model genetic algorithm—a better diversity preserver island model. Evolutionary Intelligence, 12:527–539, 12 2019.
- [200] Iyad Abu Doush, Mohammed Azmi Al-Betar, Mohammed A. Awadallah, Abdelaziz I. Hammouri, and Mohammed El-Abd. Island-based modified harmony search algorithm with neighboring heuristics methods for flow shop scheduling with blocking. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pages 976–982, 2020.
- [201] Aleksander Skakovski and Piotr Jędrzejowicz. An island-based differential evolution algorithm with the multi-size populations. Expert Systems with Applications, 126:308 – 320, 2019.
- [202] Noor Aldeen Alawad and Bilal Abed-alguni. Discrete island-based cuckoo search with highly disruptive polynomial mutation and opposition-based learn-

- ing strategy for scheduling of workflow applications in cloud environments. Arabian Journal for Science and Engineering, 11 2020.
- [203] Htet Thazin Tike Thein. Island model based differential evolution algorithm for neural network training. Advances in Computer Science : an International Journal, 3(1):67–73, 2014.
- [204] René Michel and Martin Middendorf. An island model based ant system with lookahead for the shortest supersequence problem. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, Parallel Problem Solving from Nature — PPSN V, pages 692–701, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [205] Mohammad Alshraideh, Basel Mahafzah, and Saleh al sharaeh. A multiple-population genetic algorithm for branch coverage test data generation. Software Quality Journal, 19:489–513, 09 2011.
- [206] Juan M. Palomo-Romero, Lorenzo Salas-Morera, and Laura García-Hernández. An island model genetic algorithm for unequal area facility layout problems. Expert Systems with Applications, 68:151 – 162, 2017.
- [207] Frédéric Lardeux and Adrien Goëffon. A dynamic island-based genetic algorithms framework. In Simulated Evolution and Learning, pages 156–165, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [208] Salmah Mousbah Zeed Mohammed, Ahamad Tajudin Khader, and Mohammed Azmi Al-Betar. 3-sat using island-based genetic algorithm. IEEJ Transactions on Electronics, Information and Systems, 136(12):1694–1698, 2016.
- [209] Douglas C. Montgomery. Design and Analysis of Experiments. Wiley, Hoboken, NJ, 9th edition, 2019.
- [210] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments



- in computational intelligence and data mining: Experimental analysis of power. Information Sciences, 180(10):2044–2064, 2010. Special Issue on Intelligent Distributed Information Systems.
- [211] David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems, 3:493–530, 01 1990.
- [212] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Foundations of genetic algorithms, volume 1, pages 69–93. Elsevier, 1991.
- [213] Chao Lin, Pengjun Wang, Ali Asghar Heidari, Xuehua Zhao, and Huiling Chen. A boosted communicational salp swarm algorithm: Performance optimization and comprehensive analysis. Journal of Bionic Engineering, 20:1296–1332, 2022.
- [214] Francesco Biscani and Dario Izzo. A parallel global multiobjective framework for optimization: pagmo. Journal of Open Source Software, 5(53):2338, 2020.
- [215] Dario Izzo, Marek Ruciński, and Francesco Biscani. The Generalized Island Model, pages 151–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [216] Thaer Thaher, Alaa Sheta, Mohammed Awad, and Mohammed Aldasht. Enhanced variants of crow search algorithm boosted with cooperative based island model for global optimization. Expert Systems with Applications, 238:121712, 2024.
- [217] Seyedali Mirjalili and Jin Dong. Multi-Objective Optimization using Artificial Intelligence Techniques. 01 2020.
- [218] J.G. Digalakis and K.G. Margaritis. On benchmarking functions for genetic algorithms. International Journal of Computer Mathematics, 77(4):481–506, 2001.
- [219] Jing Liang, B. Qu, and Ponnuthurai Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, November 2013.

- [220] Dheeru Dua and Casey Graff. UCI Machine Learning Repository. Technical report, University of California, Irvine, School of Information and Computer Sciences, 2019.
- [221] Jinyu Zhao, Yichen Zhang, Xuehai He, and Pengtao Xie. Covid-ct-dataset: a ct scan dataset about covid-19. arXiv preprint arXiv:2003.13865, 2020.
- [222] Joseph Paul Cohen, Paul Morrison, Lan Dao, Karsten Roth, Tim Q Duong, and Marzyeh Ghassemi. Covid-19 image data collection: Prospective predictions are the future. arXiv 2006.11988, 2020.
- [223] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1):3–18, 2011.
- [224] W.J. Conover. Practical Nonparametric Statistics. John Wiley & Sons, 3 edition, 1999.
- [225] Robert H. Riffenburgh. Chapter summaries. In Robert H. Riffenburgh, editor, Statistics in Medicine (Second Edition), pages 533 – 580. Academic Press, Burlington, second edition edition, 2006.
- [226] Myles Hollander and Douglas A. Wolfe. Nonparametric Statistical Methods. John Wiley & Sons, Hoboken, NJ, 3 edition, 2013.
- [227] Seyedali Mirjalili. Sca: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems, 96:120–133, 2016.
- [228] Laith Abualigah, Dalia Yousri, Mohamed Abd Elaziz, Ahmed A. Ewees, Mohammed A.A. Al-qaness, and Amir H. Gandomi. Aquila optimizer: A novel meta-heuristic optimization algorithm. Computers & Industrial Engineering, 157:107250, 2021.
- [229] Afshin Faramarzi, Mohammad Heidarinejad, Brent Stephens, and Seyedali Mirjalili. Equilibrium optimizer: A novel optimization algorithm. Knowledge-Based Systems, 191:105190, 2020.

- [230] Laith Abualigah, Ali Diabat, Seyedali Mirjalili, Mohamed Abd Elaziz, and Amir H. Gandomi. The arithmetic optimization algorithm. Computer Methods in Applied Mechanics and Engineering, 376:113609, 2021.
- [231] Xin-She Yang. Flower pollination algorithm for global optimization. In Jérôme Durand-Lose and Nataša Jonoska, editors, Unconventional Computation and Natural Computation, pages 240–249, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [232] Hussam Fakhouri, Faten Hamad, and Abedalsalam Alawamrah. Success history intelligent optimizer. The Journal of Supercomputing, 78:6461–6502, 04 2022.
- [233] Ronald W. Morrison and Kenneth A. De Jong. Measurement of population diversity. In Pierre Collet, Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, and Marc Schoenauer, editors, Artificial Evolution, pages 31–41, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [234] Tapas Si, Péricles B.C. Miranda, and Debolina Bhattacharya. Novel enhanced salp swarm algorithms using opposition-based learning schemes for global optimization problems. Expert Systems with Applications, 207:117961, 2022.
- [235] Nguyen Van Thieu and Seyedali Mirjalili. Mealpy: An open-source library for latest meta-heuristic algorithms in python. Journal of Systems Architecture, 2023.
- [236] R. Rao. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7:19–34, 2016.
- [237] Seyedali Mirjalili, Amir H. Gandomi, Seyedeh Zahra Mirjalili, Shahrzad Saremi, Hosam Faris, and Seyed Mohammad Mirjalili. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114:163–191, 2017.
- [238] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.
- [239] Mohammed A. Awadallah, Iyad Abu-Doush, Mohammed Azmi Al-Betar, and Ma-

- lik Shehadeh Braik. Chapter 19 - metaheuristics for optimizing weights in neural networks. In Seyedali Mirjalili and Amir H. Gandomi, editors, Comprehensive Metaheuristics, pages 359–377. Academic Press, 2023.
- [240] Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. Training feedforward neural networks using multi-verse optimizer for binary classification problems. Applied Intelligence, 45:322–332, 09 2016.
- [241] Seyed Mohammad Mirjalili. How effective is the grey wolf optimizer in training multi-layer perceptrons. Applied Intelligence, 43:150–161, 2014.
- [242] T. Thaher, M. Mafarja, B. Abdalhaq, and H. Chantar. Wrapper-based feature selection for imbalanced data using binary queuing search algorithm. In 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), pages 1–6, 2019.
- [243] Thaeer Thaher and Rashid Jayousi. Prediction of student’s academic performance using feedforward neural network augmented with stochastic trainers. In 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), pages 1–7, 2020.
- [244] K.P. Baby Resma and Madhu S. Nair. Multilevel thresholding for image segmentation using krill herd optimization algorithm. Journal of King Saud University - Computer and Information Sciences, 33(5):528–541, 2021.
- [245] Erick Rodríguez-Esparza, Laura A. Zanella-Calzada, Diego Oliva, Ali Asghar Heidari, Daniel Zaldivar, Marco Pérez-Cisneros, and Loke Kok Foong. An efficient Harris hawks-inspired image segmentation method. Expert Systems with Applications, 155:113428, 2020.
- [246] Essam H Houssein, Marwa M Emam, and Abdelmgeid A Ali. An optimized deep learning architecture for breast cancer diagnosis based on improved marine predators algorithm. Neural Computing and Applications, pages 1–19, 2022.
- [247] Essam H. Houssein, Kashif Hussain, Laith Abualigah, Mohamed Abd Elaziz, Waleed

- Alomoush, Gaurav Dhiman, Youcef Djenouri, and Erik Cuevas. An improved opposition-based marine predators algorithm for global optimization and multilevel thresholding image segmentation. Knowledge-Based Systems, 229:107348, 2021.
- [248] Mohamed H. Merzban and Mahmoud Elbayoumi. Efficient solution of otsu multi-level image thresholding: A comparative study. Expert Systems with Applications, 116:299–309, 2019.
- [249] Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1):62–66, 1979.
- [250] J.N. Kapur, P.K. Sahoo, and A.K.C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. Computer Vision, Graphics, and Image Processing, 29(3):273–285, 1985.
- [251] Laith Abualigah, Ali Diabat, Putra Sumari, and Amir H. Gandomi. A novel evolutionary arithmetic optimization algorithm for multilevel thresholding segmentation of covid-19 ct images. Processes, 9(7), 2021.
- [252] Mohamed Abd Elaziz, Ahmed A. Ewees, Dalia Yousri, Husein S. Naji Alwerfali, Qamar A. Awad, Songfeng Lu, and Mohammed A. A. Al-Qaness. An improved marine predators algorithm with fuzzy entropy for multi-level thresholding: Real world example of covid-19 ct image segmentation. IEEE Access, 8:125306–125330, 2020.
- [253] Jagat Narain Kapur, Prasanna K Sahoo, and Andrew KC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. Computer vision, graphics, and image processing, 29(3):273–285, 1985.
- [254] Rajarshi Bandyopadhyay, Rohit Kundu, Diego Oliva, and Ram Sarkar. Segmentation of brain mri using an altruistic harris hawks’ optimization algorithm. Knowledge-Based Systems, 232:107468, 2021.
- [255] Pankaj Upadhyay and Jitender Kumar Chhabra. Kapur’s entropy based optimal multilevel image segmentation using crow search algorithm. Applied Soft Computing, 97:105522, 2020.

- [256] Wenqi Ji and Xiaoguang He. Kapur's entropy for multilevel thresholding image segmentation based on moth-flame optimization. Mathematical Biosciences and Engineering, 18(6):7110–7142, 2021.
- [257] Essam H. Houssein, Bahaa El din Helmy, Diego Oliva, Pradeep Jangir, M. Premkumar, Ahmed A. Elngar, and Hassan Shaban. An efficient multi-thresholding based covid-19 ct images segmentation approach using an improved equilibrium optimizer. Biomedical Signal Processing and Control, 73:103401, 2022.
- [258] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004.
- [259] Cong Jin and Shu-Wei Jin. Parameter optimization of software reliability growth model with s-shaped testing-effort function using improved swarm intelligent optimization. Applied Soft Computing, 40:283–291, 2016.
- [260] Lokendra K. Sharma, R. K. Saket, and B. B. Sagar. Software reliability growth models and tools - a review. In 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pages 2057–2061, 2015.
- [261] Ankur Choudhary, Anurag Singh Baghel, and Om Prakash Sangwan. Efficient parameter estimation of software reliability growth models using harmony search. IET Software, 11(6):286–291, 2017.
- [262] John D. Musa. A theory of software reliability and its application. IEEE Transactions on Software Engineering, SE-1(3):312–327, 1975.
- [263] Larry H. Crow. Reliability analysis for complex, repairable systems. 1975.
- [264] Chin-Yu Huang, M.R. Lyu, and Sy-Yen Kuo. A unified scheme of some nonhomogenous poisson process models for software reliability estimation. IEEE Transactions on Software Engineering, 29(3):261–269, 2003.
- [265] Taehyoun Kim, Kwangkyu Lee, and Jongmoon Baik. An effective approach to estimating the parameters of software reliability growth models using a real-valued ge-

- netic algorithm. Journal of Systems and Software, 102:134–144, 2015.
- [266] Michael R. Lyu, editor. Handbook of software reliability engineering. McGraw-Hill, Inc., USA, 1996.
- [267] Chao-Jung Hsu and Chin-Yu Huang. A study on the applicability of modified genetic algorithms for the parameter estimation of software reliability modeling. In 2010 IEEE 34th Annual Computer Software and Applications Conference, pages 531–540, 2010.
- [268] Alaa F. Sheta and Amal Abdel-Raouf. Estimating the parameters of software reliability growth models using the grey wolf optimization algorithm. International Journal of Advanced Computer Science and Applications, 7(4), 2016.
- [269] Li Zhen, Yang Liu, Wang Dongsheng, and Zheng Wei. Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. IEEE Access, 8:29354–29369, 2020.
- [270] A. Sheta. Reliability growth modeling for software fault detection using particle swarm optimization. In 2006 IEEE International Conference on Evolutionary Computation, pages 3071–3078, 2006.
- [271] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata. Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution. IEEE Transactions on Software Engineering, 15(3):345–355, 1989.
- [272] A. Wood. Predicting software reliability. Computer, 29(11):69–77, 1996.
- [273] W.D. Brooks, R.W. Motley, and IBM FEDERAL SYSTEMS DIV GAITHERSBURG MD. Analysis of Discrete Software Reliability Models. Defense Technical Information Center, 1980.
- [274] T. Minohara and Y. Tohma. Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms. In Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95, pages 324–329, 1995.

[275] P. N. Misra. Software reliability analysis. IBM Systems Journal, 22(3):262–270, 1983.



## Appendices

### Appendix A: Characteristic Tables of Real-Valued Mathematical Functions

**Table A-1:** Description of the standard unimodal benchmark functions.

Function	Dimensions	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^n x_i^2$	30,100, 500	[-100,100]	0
$f_2(x) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	30,100, 500	[-10,10]	0
$f_3(x) = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2$	30,100, 500	[-100,100]	0
$f_4(x) = \max_i \{  x_i , 1 \leq i \leq n \}$	30,100, 500	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[ 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	30,100, 500	[-30,30]	0
$f_6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$	30,100, 500	[-100,100]	0
$f_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1)$	30,100, 500	[-128,128]	0

**Table A-2:** Description of the standard multimodal benchmark functions.

Function	Dimensions	Range	$f_{\min}$
$f_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30,100, 500	[-500,500]	$-418.9829 \times n$
$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30,100, 500	[-5.12,5.12]	0
$f_{10}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$	30,100, 500	[-32,32]	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30,100, 500	[-600,600]	0
$f_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$	30,100, 500	[-50,50]	0
$y_i = 1 + \frac{x_i+1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 - a & < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$			
$f_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30,100, 500	[-50,50]	0

**Table A-3:** Description of the standard fixed-dimension multimodal benchmark functions.

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

**Table A-4:** The characteristics of CEC2014 benchmark functions (U: Unimodal, M: Multimodal, H: Hybrid, C: Composition)

Function	Description	Category	Dimensions	$F_{min}$
F1	Rotated High Conditioned Elliptic Function	U	30	100
F2	Rotated Bent Cigar Function	U	30	200
F3	Rotated Discus Function	U	30	300
F4	Shifted and Rotated Rosenbrock's Function	M	30	400
F5	Shifted and Rotated Ackley's Function	M	30	500
F6	Shifted and Rotated Weierstrass Function	M	30	600
F7	Shifted and Rotated Griewank's Function	M	30	700
F8	Shifted Rastrigin's Function	M	30	800
F9	Shifted and Rotated Rastrigin's Function	M	30	900
F10	Shifted Schwefel's Function	M	30	1000
F11	Shifted and Rotated Schwefel's Function	M	30	1100
F12	Shifted and Rotated Katsuura Function	M	30	1200
F13	Shifted and Rotated HappyCat Function	M	30	1300
F14	Shifted and Rotated HGBat Function	M	30	1400
F15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	M	30	1500
F16	Shifted and Rotated Expanded Scaffer's F6Function	M	30	1600
F17	Hybrid Function 1 (N=3)	H	30	1700
F18	Hybrid Function 2 (N=3)	H	30	1800
F19	Hybrid Function 3 (N=4)	H	30	1900
F20	Hybrid Function 4 (N=4)	H	30	2000
F21	Hybrid Function 5 (N=5)	H	30	2100
F22	Hybrid Function 6 (N=5)	H	30	2200
F23	Composition Function 1 (N=5)	C	30	2300
F24	Composition Function 2 (N=3)	C	30	2400
F25	Composition Function 3 (N=3)	C	30	2500
F26	Composition Function 4 (N=5)	C	30	2600
F27	Composition Function 5 (N=5)	C	30	2700
F28	Composition Function 6 (N=5)	C	30	2800
F29	Composition Function 7 (N=3)	C	30	2900
F30	Composition Function 8 (N=3)	C	30	3000

Range  $[-100, 100]^D$

## Appendix B: Statistical Results

**Table B-1:** Statistical Comparison of Enhanced CSA Variants against Basic CSA Across Standard Functions F1-F23 Using Wilcoxon Rank Sum Test (+, -, = indicate superior, inferior, and equivalent performance, respectively)

Function	MRCSA vs CSA		ATCSA vs CSA		ECSA vs CSA	
	P-value	Sig.	P-value	Sig.	P-value	Sig.
F1	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+
F2	<b>3.02E-11</b>	+	2.77E-01	=	<b>3.02E-11</b>	+
F3	<b>3.02E-11</b>	+	<b>3.20E-09</b>	+	<b>3.02E-11</b>	+
F4	<b>3.02E-11</b>	+	4.20E-01	=	<b>3.02E-11</b>	+
F5	<b>3.02E-11</b>	+	<b>2.96E-05</b>	+	<b>3.02E-11</b>	+
F6	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+
F7	<b>3.02E-11</b>	+	3.63E-01	=	<b>3.02E-11</b>	+
F8	<b>3.02E-11</b>	+	9.47E-01	=	<b>5.56E-10</b>	+
F9	<b>3.02E-11</b>	+	1.09E-01	=	<b>3.02E-11</b>	+
F10	<b>3.02E-11</b>	+	<b>5.57E-03</b>	-	<b>3.02E-11</b>	+
F11	<b>3.02E-11</b>	+	<b>3.02E-11</b>		<b>3.02E-11</b>	+
F12	<b>3.02E-11</b>	+	1.96E-01	=	<b>3.02E-11</b>	+
F13	<b>3.02E-11</b>	+	<b>1.39E-06</b>	+	<b>3.02E-11</b>	+
F14	8.15E-02	=	7.23E-01	=	8.15E-02	=
F15	5.61E-01	=	<b>4.05E-03</b>	-	<b>7.63E-05</b>	-
F16	NaN	=	NaN	=	NaN	=
F17	NaN	=	NaN	=	NaN	=
F18	NaN	=	NaN	=	3.34E-01	=
F19	NaN	=	NaN	=	NaN	=
F20	<b>3.54E-09</b>	-	4.04E-01	=	<b>2.24E-09</b>	-
F21	2.08E-01	=	8.84E-01	=	<b>1.56E-06</b>	-
F22	<b>1.68E-03</b>	-	<b>4.19E-02</b>	+	<b>1.09E-07</b>	-
F23	<b>7.58E-04</b>	-	7.20E-01	=	<b>2.27E-09</b>	-

**Table B-2: Wilcoxon ranksum test results for iECSA vs. other algorithms across standard F1-F23 functions**

Function	BAT	DE	JAYA	PSO	SSA	WOA	SCA	GWO	HHO	AO	CapSAI	EO	HGS	AOA	FPA	MFO	SHIO
F1	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.42E-01	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.87E-10	3.02E-11	3.02E-11	3.02E-11	3.01E-11	3.02E-11	3.02E-11	3.02E-11
F2	3.02E-11	3.02E-11	3.69E-11	3.02E-11	3.02E-11	3.02E-11	7.38E-10	3.02E-11	3.02E-11	2.37E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F3	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.82E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.01E-07
F4	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	5.57E-10	3.02E-11	3.02E-11	3.02E-11	9.76E-10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F5	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.08E-11	2.19E-08	2.44E-09	3.02E-11	3.02E-11	1.44E-11	3.02E-11	3.02E-11	3.02E-11
F6	3.02E-11	3.02E-11	3.02E-11	5.22E-11	7.48E-02	3.02E-11	3.02E-11	3.02E-11	8.89E-10	2.87E-10	2.49E-06	3.02E-11	3.02E-11	3.01E-11	3.02E-11	3.02E-11	3.02E-11
F7	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	8.15E-05	3.02E-11	9.26E-11	2.01E-04	2.44E-09	0.482517	0.706171	0.251881	1.86E-06	3.02E-11	3.02E-11	0.006377
F8	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11	4.63E-09	2.69E-11	1.39E-06	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11	2.69E-11
F9	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.21E-12	3.02E-11	6.34E-07	1.21E-12	3.69E-11	1.21E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	1.44E-11
F10	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.83E-10	2.08E-11	3.02E-11	1.72E-12	1.21E-12	4.07E-11	3.02E-11	3.15E-12	1.21E-12	2.63E-11	3.02E-11	3.02E-11	7.57E-12
F11	3.02E-11	3.02E-11	3.02E-11	9.26E-09	3.47E-10	1.24E-09	3.02E-11	1.49E-03	1.21E-12	2.29E-08	2.36E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	3.15E-10
F12	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.08E-11	1.61E-10	3.50E-09	3.02E-11	3.02E-11	3.00E-11	3.02E-11	3.02E-11	3.02E-11
F13	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.73E-07	3.02E-11	3.02E-11	3.02E-11	3.33E-11	3.20E-09	1.87E-05	3.02E-11	3.02E-11	2.87E-11	3.02E-11	3.02E-11	3.02E-11
F14	5.69E-11	NaN	1.21E-12	1.31E-03	NaN	1.37E-03	4.56E-12	6.31E-05	2.15E-02	1.82E-10	1.26E-05	1.20E-12	0.01099	4.61E-13	1.21E-12	5.31E-06	9.54E-13
F15	1.21E-12	3.34E-01	1.21E-12	1.12E-12	1.21E-12	1.21E-12	1.21E-12	1.20E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.16E-12	1.21E-12	1.18E-12	1.21E-12
F16	3.09E-04	NaN	1.21E-12	NaN	NaN	NaN	1.21E-12	1.89E-09	NaN	1.21E-12	8.81E-07	NaN	NaN	1.21E-12	1.85E-10	NaN	1.65E-11
F17	0.002762	NaN	4.57E-12	NaN	NaN	1.21E-12	1.21E-12	1.21E-12	1.70E-08	1.21E-12	5.76E-11	0.081523	NaN	1.21E-12	4.57E-12	NaN	1.21E-12
F18	1.21E-12	NaN	1.21E-12	NaN	NaN	1.21E-12	1.21E-12	1.21E-12	0.002784	1.21E-12	1.65E-11	4.57E-12	NaN	1.21E-12	1.21E-12	NaN	4.57E-12
F19	1.21E-12	NaN	NaN	NaN	NaN	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	0.002698	1.21E-12	1.21E-12	NaN	1.21E-12
F20	2.31E-06	0.32041	0.005699	0.916703	5.52E-05	1.28E-09	3.00E-11	2.27E-05	1.31E-08	3.00E-11	3.00E-11	0.000183	0.09733	3.24E-07	3.00E-11	0.021386	4.97E-09
F21	1.21E-12	NaN	1.21E-12	6.34E-05	0.000309	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	4.36E-12	0.001354	1.21E-12	1.21E-12	6.47E-05	1.21E-12
F22	1.21E-12	NaN	5.85E-09	0.011015	0.005561	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.18E-12	1.25E-07	1.21E-12	1.21E-12	1.24E-05	1.21E-12
F23	1.21E-12	NaN	1.93E-10	0.041911	0.001364	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	6.44E-05	1.21E-12	1.21E-12	1.26E-05	1.21E-12

**Table B-3: Wilcoxon ranksum test results for iECOA vs. other algorithms on CEC2014 test suit**

Function	BAT	DE	JAVA	PSO	SSA	WOA	SCA	GWO	HHO	AO	CapSA	EO	HGS	AOA	PPA	MFO	SHO
F1-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.11E-06	1.60E-07	3.02E-11	3.02E-11	5.49E-11	3.02E-11	3.02E-11	3.02E-11	1.61E-10	1.87E-07	3.02E-11	1.22E-02	6.07E-11	3.02E-11
F2-CEC2014	3.02E-11	2.13E-05	3.02E-11	7.70E-04	8.31E-03	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F3-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.07E-07	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.55E-09	3.02E-11	3.02E-11	5.57E-10	3.02E-11
F4-CEC2014	3.02E-11	3.39E-02	3.02E-11	3.18E-02	1.06E-03	3.69E-11	3.02E-11	8.15E-11	3.02E-11	3.02E-11	3.02E-11	3.37E-05	2.20E-07	3.02E-11	3.02E-11	3.47E-10	3.02E-11
F5-CEC2014	4.61E-10	3.02E-11	3.02E-11	3.02E-11	4.20E-05	3.02E-11	3.02E-11	3.02E-11	3.02E-08	3.02E-11	3.02E-11	3.02E-05	3.02E-04	3.02E-11	3.02E-11	5.49E-11	3.02E-11
F6-CEC2014	3.02E-11	6.01E-08	3.02E-11	6.72E-08	3.35E-08	1.09E-10	4.50E-11	3.02E-11	3.02E-11	3.02E-11	8.15E-11	3.02E-11	3.34E-11	3.02E-11	1.29E-09	3.20E-11	1.20E-08
F7-CEC2014	3.02E-11	3.56E-01	3.02E-11	7.48E-02	2.83E-01	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F8-CEC2014	3.02E-11	9.49E-03	3.02E-11	1.60E-02	4.69E-08	3.34E-11	3.02E-11	5.08E-04	3.02E-11	3.02E-11	3.02E-11	1.86E-09	2.67E-10	3.02E-11	3.02E-11	1.38E-02	3.02E-11
F9-CEC2014	3.02E-11	3.02E-11	3.02E-11	7.90E-05	9.53E-07	3.02E-11	3.02E-11	2.97E-04	3.02E-11	3.02E-11	3.02E-11	2.28E-02	1.19E-06	3.02E-11	3.02E-11	1.69E-09	3.02E-11
F10-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.09E-01	1.17E-04	3.20E-09	3.02E-11	2.07E-06	3.69E-11	3.02E-11	4.20E-10	5.49E-11	4.08E-11	3.02E-11	3.02E-03	2.01E-04	1.33E-10
F11-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.78E-04	7.04E-07	3.02E-11	3.02E-11	1.45E-01	3.02E-11	3.02E-11	3.02E-11	4.74E-06	2.34E-10	3.02E-11	3.34E-11	1.43E-08	4.98E-11
F12-CEC2014	3.02E-11	4.13E-01	3.02E-04	5.60E-02	7.69E-02	8.99E-03	3.02E-04	1.56E-02	1.78E-10	3.02E-11	5.49E-05	3.02E-03	4.20E-01	3.02E-03	1.33E-08	6.52E-01	3.15E-02
F13-CEC2014	3.02E-11	3.25E-01	2.12E-03	5.99E-02	7.43E-02	9.79E-02	3.02E-05	4.86E-02	3.02E-11	3.02E-11	3.02E-06	6.57E-02	9.76E-01	3.02E-11	9.68E-02	6.07E-04	3.10E-02
F14-CEC2014	3.02E-11	9.66E-02	3.02E-11	8.11E-02	9.01E-02	5.06E-02	3.02E-11	5.46E-03	3.02E-11	3.02E-11	3.02E-11	7.24E-02	2.68E-02	3.02E-11	8.17E-02	3.02E-11	1.61E-10
F15-CEC2014	3.02E-11	5.46E-09	3.02E-11	1.52E-03	6.03E-02	3.02E-11	3.02E-11	2.38E-07	3.02E-11	3.02E-11	3.02E-11	2.78E-07	1.61E-06	3.02E-11	3.02E-11	5.57E-10	3.02E-11
F16-CEC2014	4.98E-11	4.50E-11	3.02E-11	8.88E-01	6.77E-03	2.39E-08	1.33E-08	6.64E-02	3.50E-03	3.02E-08	1.31E-03	5.49E-01	4.04E-01	3.02E-02	8.99E-03	3.83E-06	1.36E-03
F17-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.56E-08	5.57E-10	3.02E-11	3.02E-11	1.21E-10	3.02E-11	3.02E-11	3.02E-11	5.57E-10	3.02E-11	3.02E-11	1.39E-06	6.70E-11	3.02E-11
F18-CEC2014	3.02E-11	3.02E-11	3.02E-11	7.28E-06	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.67E-09	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F19-CEC2014	3.02E-11	3.34E-11	3.02E-11	2.25E-04	8.31E-03	3.69E-11	3.02E-11	4.42E-06	3.02E-11	3.02E-11	3.02E-11	9.92E-11	8.88E-06	3.02E-11	4.35E-05	4.36E-02	3.02E-11
F20-CEC2014	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	6.70E-11	3.02E-11	3.02E-11
F21-CEC2014	3.02E-11	1.75E-05	3.02E-11	1.86E-06	4.62E-10	3.02E-11	3.02E-11	1.46E-10	3.02E-11	3.02E-11	3.02E-11	4.08E-11	4.50E-11	3.02E-11	3.34E-11	3.34E-11	3.02E-11
F22-CEC2014	4.08E-11	9.71E-01	1.86E-09	2.43E-05	2.51E-02	6.72E-10	8.99E-11	4.64E-03	1.78E-10	3.02E-11	1.31E-08	1.52E-03	5.26E-04	3.02E-11	1.30E-08	1.03E-06	2.68E-06
F23-CEC2014	3.02E-11	7.85E-02	3.02E-11	9.04E-04	4.98E-11	3.02E-11	3.02E-11	3.02E-06	1.21E-12	3.02E-11	1.21E-12	2.52E-01	1.21E-12	1.21E-12	3.02E-04	3.02E-11	3.02E-11
F24-CEC2014	3.02E-11	4.23E-02	3.02E-11	1.56E-08	3.02E-11	5.11E-01	2.32E-06	3.02E-09	2.90E-11	3.02E-11	3.00E-11	2.99E-11	1.21E-12	1.21E-12	3.02E-11	3.02E-11	3.02E-11
F25-CEC2014	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.20E-10	7.85E-03	8.15E-11	3.02E-11	1.21E-12	1.14E-03	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.09E-10	4.50E-11	4.56E-04
F26-CEC2014	3.02E-11	6.55E-02	2.32E-03	5.46E-09	7.82E-02	6.57E-02	3.02E-08	6.12E-10	1.44E-11	2.69E-11	2.95E-11	6.00E-08	2.63E-11	4.11E-12	5.76E-02	3.69E-11	9.40E-12
F27-CEC2014	3.02E-11	5.49E-04	3.02E-11	1.22E-02	1.55E-09	3.02E-11	3.02E-11	3.34E-11	1.21E-12	1.21E-12	1.21E-12	1.29E-06	1.21E-12	1.21E-12	3.34E-11	4.62E-10	3.02E-11
F28-CEC2014	5.59E-01	3.02E-11	3.02E-11	2.27E-03	3.34E-11	3.18E-04	3.16E-05	3.69E-11	1.21E-12	3.02E-11	1.21E-12	3.02E-11	1.21E-12	1.21E-12	3.02E-11	3.02E-11	6.36E-05
F29-CEC2014	3.02E-11	3.02E-11	3.02E-11	1.02E-08	7.39E-11	3.01E-11	3.02E-11	6.70E-11	1.21E-12	5.57E-10	3.00E-11	3.69E-11	4.33E-04	1.21E-12	9.92E-11	1.20E-10	3.02E-11
F30-CEC2014	3.02E-11	1.29E-06	1.33E-10	1.73E-06	3.02E-11	3.02E-11	3.02E-11	3.47E-10	2.82E-11	1.11E-10	3.02E-11	1.02E-05	1.34E-05	1.21E-12	1.64E-11	9.07E-03	3.02E-11

**Table B-4:** Statistical Comparison of Enhanced CapSA Variants against Basic CapSA Across Standard Functions F1-F23 Using Wilcoxon Rank Sum Test

Function	MCapSA1 vs CapSA		MCapSA2 vs CapSA		MCapSA3 vs CapSA	
	P-value	Sig.	P-value	Sig.	P-value	Sig.
F1	9.94E-01	=	3.26E-01	=	4.73E-01	=
F2	1.45E-01	=	<b>5.83E-03</b>	-	1.02E-01	=
F3	<b>4.64E-03</b>	-	<b>1.52E-03</b>	-	<b>3.39E-02</b>	-
F4	8.77E-01	=	3.40E-01	=	7.96E-01	=
F5	1.86E-01	=	<b>9.07E-03</b>	+	<b>2.27E-03</b>	+
F6	7.28E-01	=	2.40E-01	=	5.11E-01	=
F7	7.06E-01	=	2.58E-01	=	7.96E-01	=
F8	9.47E-01	=	1.67E-01	=	7.06E-01	=
F9	NaN	=	NaN	=	NaN	=
F10	<b>3.45E-02</b>	-	<b>2.38E-03</b>	-	<b>3.51E-02</b>	-
F11	NaN	=	NaN	=	NaN	=
F12	<b>1.68E-03</b>	+	<b>2.68E-04</b>	+	<b>4.03E-03</b>	+
F13	6.52E-01	=	7.62E-01	=	2.12E-01	=
F14	9.23E-01	=	8.76E-02	=	6.67E-02	=
F15	1.02E-01	=	<b>1.85E-08</b>	+	<b>1.86E-09</b>	+
F16	<b>2.36E-11</b>	+	<b>2.74E-11</b>	+	<b>2.46E-11</b>	+
F17	<b>6.67E-11</b>	+	<b>1.71E-10</b>	+	<b>2.21E-10</b>	+
F18	<b>8.14E-11</b>	+	<b>8.14E-11</b>	+	<b>1.20E-10</b>	+
F19	<b>2.23E-09</b>	+	<b>8.10E-10</b>	+	<b>2.15E-10</b>	+
F20	<b>7.09E-08</b>	+	<b>7.66E-05</b>	+	<b>3.01E-07</b>	+
F21	<b>1.56E-02</b>	+	<b>2.19E-08</b>	+	<b>2.44E-09</b>	+
F22	2.12E-01	=	<b>1.86E-09</b>	+	<b>7.09E-08</b>	+
F23	8.24E-02	=	<b>2.02E-08</b>	+	<b>2.57E-07</b>	+

**Table B-5:** Statistical Comparison of Enhanced CapSA Variants against Basic CapSA Across CEC2014 Suit Using Wilcoxon Rank Sum Test

Function	MCapSA1 vs CapSA		MCapSA2 vs CapSA		MCapSA3 vs CapSA	
	P-value	Sig.	P-value	Sig.	P-value	Sig.
F1-CEC2014	<b>6.05E-07</b>	+	<b>1.41E-09</b>	+	<b>1.07E-09</b>	+
F2-CEC2014	<b>2.15E-10</b>	+	<b>3.34E-11</b>	+	<b>3.02E-11</b>	+
F3-CEC2014	<b>3.35E-08</b>	+	<b>3.02E-11</b>	+	<b>4.98E-11</b>	+
F4-CEC2014	<b>1.17E-09</b>	+	<b>1.33E-10</b>	+	<b>5.49E-11</b>	+
F5-CEC2014	1.67E-01	=	<b>7.66E-05</b>	+	<b>1.25E-04</b>	+
F6-CEC2014	<b>1.11E-04</b>	+	<b>1.78E-10</b>	+	<b>1.33E-10</b>	+
F7-CEC2014	<b>5.49E-11</b>	+	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+
F8-CEC2014	<b>9.26E-09</b>	+	<b>1.61E-10</b>	+	<b>4.98E-11</b>	+
F9-CEC2014	<b>6.52E-09</b>	+	<b>1.78E-10</b>	+	<b>8.99E-11</b>	+
F10-CEC2014	7.73E-02	=	<b>4.35E-05</b>	+	<b>1.50E-02</b>	+
F11-CEC2014	<b>4.71E-04</b>	+	<b>8.84E-07</b>	+	<b>9.03E-04</b>	+
F12-CEC2014	6.57E-02	+	<b>1.17E-03</b>	+	<b>4.71E-04</b>	+
F13-CEC2014	<b>7.39E-11</b>	+	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+
F14-CEC2014	<b>1.78E-10</b>	+	<b>3.02E-11</b>	+	<b>3.02E-11</b>	+
F15-CEC2014	<b>2.02E-08</b>	+	<b>3.34E-11</b>	+	<b>3.34E-11</b>	+
F16-CEC2014	<b>1.53E-05</b>	+	<b>2.03E-09</b>	+	<b>1.55E-09</b>	+
F17-CEC2014	<b>2.53E-04</b>	+	<b>5.00E-09</b>	+	<b>2.39E-08</b>	+
F18-CEC2014	<b>2.20E-07</b>	+	<b>4.08E-11</b>	+	<b>3.34E-11</b>	+
F19-CEC2014	<b>3.83E-06</b>	+	<b>4.31E-08</b>	+	<b>1.87E-07</b>	+
F20-CEC2014	<b>1.17E-04</b>	+	<b>1.25E-07</b>	+	<b>5.57E-10</b>	+
F21-CEC2014	<b>8.56E-04</b>	+	<b>7.22E-06</b>	+	<b>7.04E-07</b>	+
F22-CEC2014	<b>4.98E-04</b>	+	<b>9.26E-09</b>	+	<b>9.26E-09</b>	+
F23-CEC2014	<b>3.99E-02</b>	-	<b>1.66E-02</b>	-	1.12E-01	=
F24-CEC2014	6.52E-01	=	5.89E-01	=	3.04E-01	=
F25-CEC2014	<b>3.55E-05</b>	-	<b>6.77E-03</b>	-	<b>1.49E-02</b>	-
F26-CEC2014	<b>6.04E-08</b>	+	<b>6.73E-10</b>	+	<b>4.35E-11</b>	+
F27-CEC2014	<b>2.13E-05</b>	-	<b>1.83E-04</b>	-	<b>3.76E-04</b>	-
F28-CEC2014	5.01E-02	=	2.17E-01	=	6.25E-02	=
F29-CEC2014	<b>4.21E-02</b>	+	<b>2.38E-03</b>	-	1.09E-01	=
F30-CEC2014	6.79E-02	=	7.96E-01	=	4.83E-01	=

# تطوير الخوارزميات الاستدلالية: مشغلات مبتكرة ونموذج الجزيرة التعاوني التكيفي للتحسين الفعال

اعداد: ثائر أحمد درويش ظاهر

## الملخص

تُعدّ خوارزميات ذكاء السرب من الأدوات البارزة في حل مسائل التحسين الشامل المعقدة من خلال محاكاة العمليات الطبيعية. ومع ذلك، فإن هذه الخوارزميات، بما في ذلك خوارزمية بحث الغريبان (CSA) وخوارزمية بحث القردة الكبوشية (CapSA)، تعاني من قيود متأصلة مثل انخفاض دقة البحث والاتجاه نحو التقارب إلى الحلول المحلية. تهدف هذه الأطروحة إلى تطوير إصدارات محسنة من هذه الخوارزميات حيث يمكنها التعامل بفعالية مع مجموعة واسعة من مشكلات التحسين النظرية والتطبيقية. يُعدّ أحد الأساليب الشائعة لتحسين هذه الخوارزميات هو استخدام آلية التعداد الهيكلي، والتي تحافظ على التنوع أثناء عملية البحث للحد من التقارب المبكر. ومن بين هذه الأساليب، يُعتبر نموذج الجزر أحد أكثر الأساليب شيوعاً، حيث يتم تقسيم السكان إلى مجموعات فرعية صغيرة ومستقلة تُعرف بالجزر، تعمل كل منها بشكل متوازٍ. وتتمثل التقنية الأساسية لتعزيز التنوع السكاني في آلية الهجرة، التي تُسهّل تبادل المعلومات الهامة والمفيدة بين الجزر أثناء التكرارات. تشمل التحسينات المقترحة على CSA و CapSA تقديم استراتيجيات تكيفية وعوامل تشغيل جديدة، مما أدى إلى تطوير نسختين محسنتين منهما، أُطلق عليهما ECSA و ECapSA على التوالي. علاوة على ذلك، تم دمج هذه التحسينات ضمن نموذج قائم على الجزر، وهما iECSA و iECapSA، المزود بسياسة هجرة تكيفية تُعدّل معدلات الهجرة ديناميكياً بناءً على التقييمات الفورية لتنوع السكان وقيم اللياقة. يهدف هذا النهج المبتكر إلى تجاوز قيود النماذج التقليدية القائمة على الجزر وتحسين قدرات البحث الشامل. تم تقييم أداء النماذج المقترحة باستخدام 53 مسألة رياضية ذات قيم حقيقية، بالإضافة إلى ثلاثة تطبيقات عملية تشمل تدريب الشبكات العصبية، وتجزئة الصور باستخدام مستويات العتبة المتعددة، ونموذج نمو موثوقية البرمجيات. كما تم التحقق من صحة النماذج المقترحة من خلال مقارنتها مع مجموعة واسعة من خوارزميات التحسين الاستدلالية التقليدية والحديثة. وقد أظهرت النتائج التجريبية أن النسخ المحسنة من CSA و CapSA تتفوق على إصداراتها الأساسية في معظم حالات الاختبار، مما يوفر نتائج أكثر دقة وموثوقية. بالإضافة إلى ذلك، أظهرت التجارب الموسعة أن iECapSA تفوقت بشكل واضح على نظيراتها في مجموعة متنوعة من التطبيقات العملية.