COEVOLUTION FOR MANY-OBJECTIVE OPTIMIZATION

This thesis is submitted in the partial fulfilment of the requirements of the degree

MASTER OF TECHNOLOGY
in
INTELLIGENT AUTOMATION AND ROBOTICS
(Electronics and Telecommunication Engineering Department)

Submitted by

Shreya Pramanik

Examination Roll Number: M6IAR24004

Registration Number: 160221 of 2021-22

Under the Guidance of

Dr. Pratyusha Rakshit

Department of Electronics and Telecommunication Engineering

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY KOLKATA–700032

April, 2024

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

Kolkata-700032

Certificate of Recommendation

This is to certify that **Smt. Shreya Pramanik (002110704003)** has completed her dissertation entitled, "**Coevolution For Many-Objective Optimization**", under the direct supervision of **Dr. Pratyusha Rakshit**, Department of Electronics and Telecommunication Engineering, Jadavpur University. We are satisfied with her work, which is being presented for the partial fulfilment of the degree of **Master of Technology** in Intelligent Automation And Robotics of Jadavpur University, Kolkata-700032.

.....

Dr. Pratyusha Rakshit

Assistant Professor,
Electronics and Telecommunication Engineering Department,
Jadavpur University, Kolkata-700032

.....

Prof. Manotosh Biswas

Head of the Department,
Electronics and Telecommunication Engineering Department,
Jadavpur University, Kolkata-700032

.....

Prof. Dipak Laha

Dean,
Faculty Council of Engineering & Technology
Jadavpur University, Kolkata-700032

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

Kolkata-700032

Certificate of Approval

The foregoing thesis entitled "**Coevolution For Many-Objective Optimization**" is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree of Master of Technology in Intelligent Automation And Robotics for which it has been submitted. It is understood that, by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed, or conclusion therein but approve this thesis only for the purpose for which it is submitted.

Final Examination for Evaluation of the Thesis

……

Signature of the Examiners

……

Signature of the Supervisor

Declaration of Originality

I hereby declare that this thesis contains a literature survey and original research work by the undersigned candidate, as part of her Master of Technology in Intelligent Automation and Robotics curriculum. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Shreya Pramanik
Examination Roll No.:M6IAR24004
Thesis Title: **Coevolution For Many-Objective Optimization**

_____

Signature of the candidate with Date

ACKNOWLEDGEMENT

This thesis entitled "**Coevolution For Many-Objective Optimization**" is the result of the work whereby I have been accompanied and supported by many people, including my guide, my friends, and lab seniors. It is a pleasant aspect that now I can express my gratitude to all of them.

First and foremost, I would like to express my sincere gratitude to my thesis supervisor **Dr. Pratyusha Rakshit**, Assistant Professor of the Department of Electronics and Telecommunication Engineering at Jadavpur University for her valuable guidance, insightful suggestions, and support while conducting this thesis work as well as in the writing of this thesis. I have been very fortunate to have a guide like her.

I would like to acknowledge the help of Mrs. Selina Khoirom, who is a research scholar in this department for helping me with the idea, implementation, and analysis. I am also thankful to my friend Sayantani who have helped me directly or indirectly during the tenure of my thesis work.

I want to express my gratitude to my parents and my husband, as, without their sacrifices, I can't do anything. And their invaluable love, encouragement, and support make me, whatever I am today.

Shreya Pramanik
Jadavpur University,
Kolkata – 700032

LIST OF PUBLICATIONS BY THE AUTHOR

1. Selina Khoirom[1] , Shreya Pramanik[2] , Pratyusha Rakshit[3], Archana Chowdhury[4], "Co-evolution with Bee Colony: Paradigm Shift in Many Objective Optimization", OPJU International Technology Conference (OTCON 3.0) on Smart Computing for Innovation and Advancement in Industry 4.0,2024.

PREFACE

The research introduces a novel approach to many-objective optimization (MaOO), which is a challenging task involving the simultaneous optimization of multiple conflicting objectives. This new approach capitalizes on the concept of coevolution, where multiple populations evolve concurrently, and exploits the inherent parallelism of the evolutionary process to enhance optimization efficiency.

At the core of this approach is the utilization of the artificial bee colony (ABC) algorithm, a nature-inspired optimization technique inspired by the foraging behavior of honeybees. The ABC algorithm is employed to handle individual objectives in parallel, allowing for the simultaneous exploration of different facets of the optimization problem.

Once the ABC algorithm converges, a set of high-quality solutions is carefully selected from each population, with each set focusing on optimizing a specific objective. These solution sets, representing different aspects of the optimization problem, are then combined to form a unified set, capturing diverse trade-offs among the objectives.

To identify the top-ranked solutions within this unified set, a novel ranking method based on fuzzy membership induction is introduced. This method takes into account the degree of membership of each solution to the Pareto front, a set of optimal solutions where no solution can be improved in one objective without sacrificing performance in another. By considering the fuzzy nature of solution optimality, this ranking method provides a nuanced assessment of solution quality. The resulting MaOO algorithm, named fuzzy-bee colony (FBC), is rigorously evaluated against three state-of-the-art techniques commonly used in many-objective optimization scenarios. Experimental results demonstrate the superiority of FBC in terms of various performance metrics, indicating its effectiveness in generating high-quality solutions across multiple objectives.

In summary, the proposed approach offers a comprehensive framework for addressing many-objective optimization problems by leveraging coevolution, parallelism, and fuzzy ranking techniques to efficiently explore the solution space and identify Pareto optimal solutions.

CONTENTS

Table of Contents

CHAPTER 1: INTRODUCTION 1

1.1. OPTIMIZATION 1
1.2. NEED OF OPTIMIZATION 2
1.3. EVOLUTIONARY ALGORITHMS (EAS) 4
1.4. WHY EA? 6
1.5. CLASSIFICATION OF EA 7
1.6. SINGLE OBJECTIVE OPTIMIZATION 8
1.6.1. Differential Evolution 8
1.6.2. Particle Swarm Optimization (PSO) 11
1.7. MULTI OBJECTIVE EVOLUTIONARY ALGORITHM 13
1.7.1. Non-dominated Sorting Genetic Algorithm II 14
1.8. MANY OBJECTIVE OPTIMIZATION 16
1.9. WHY TRADITIONAL EMOO ALGORITHM CAN'T SOLVE MAOO PROBLEM 17
1.10. LITERATURE REVIEW 18
1.11. PROPOSED STRATEGY OF COEVOLUTION 20
1.12. OBJECTIVE OF THE THESIS 21
1.13. ORGANIZATION OF THESIS 21
REFERENCE: 22

CHAPTER 2: FUZZY BEE COLONY 27

2.1. INTRODUCTION 27
2.2. DEFINITIONS OF FUZZY SET 29
2.3. ABC ALGORITHM 29
2.4. FUZZY COEVOLUTION STRATEGY 31
2.4.1. Individual Parallel Optimization of N Objectives 32
2.4.2. Unifying the obtained candidate solution 34
2.4.3. Solution of MaOO Problem 35
2.5. RESULTS 37
2.5.1. Comparative Framework and Parameter Setting 37
2.5.2. Benchmark Functions and Termination Condition 38
2.5.3. Performance Metrics 38
2.5.4. Performance Analysis 38
REFERENCE: 41

CHAPTER 3: CONCLUSION & FUTURE RESEARCH 44

3.1. INTRODUCTION 44
3.2. APPLICATIONS 44
3.2.1. Applications and Benefits: 45
3.3. FUTURE RESEARCH 46
3.4. CONCLUSION 47

	List Of Abbreviation
EA	Evolutionary Algorithm
DE	Differential Evolution
FBC	Fuzzy Bee Colony
GA	Genetic Algorithm
ACO	Ant Colony Optimization
MOO	Multi Objective Optimization
MaOO	Many Objective Optimization
PSO	Particle Swarm Optimization
NSGA-II	Non-dominated Sorting Genetic Algorithm II
ABC	Artificial Bee Colony
HypE	hypervolume estimation algorithm
MOEA/D	the multi-objective evolutionary algorithm based on decomposition
GrEA	the grid-based evolutionary algorithm

LIST OF FIGURES

Fig. No.	Figure Title	Pg. No.
1.1	Maxima and minima of an optimization problem	3
1.2	Flowchart of EA.	6
1.3	Single objective optimization problem	8
1.4	Illustration of mutation (red) and crossover (blue) in differential evolution	9
1.5	Flowchart of Differential Evolution	10
1.6	Particle Swarm Optimization Architecture Illustration	12
1.7	Flowchart of non-dominated sorting genetic algorithm II (NSGA-II)	15
2.1.	Flowchart of ABC algorithm.	30
2.2.	Fuzzy Bee Colony based MaOO	31

List of Tables

Table No.	Title of Table	Pg. No.
TABLE-2-I	HV BASED PERFORMANCE EVALUATION FOR WFG	39
TABLE-2-II	IGD BASED PERFORMANCE EVALUATION FOR DTLZ1 AND DTLZ2	40

Chapter 1: Introduction

1.1. Optimization

Optimization involves finding the most efficient or effective solution to a problem, given certain constraints or objectives. It's about improving processes, systems, or outcomes by making adjustments and fine-tuning variables to achieve the best possible result. Whether it's in business, engineering, science, or daily life, optimization aims to maximize benefits, minimize costs, or strike a balance between competing goals. Essentially, it's about making things better by finding the optimal way to do them.

In computer science and artificial intelligence, optimization plays a crucial role in developing algorithms and systems that can solve complex problems efficiently. This could include optimizing algorithms for tasks like sorting data, searching databases, or training machine learning models to make predictions.

In recent times, there has been an increasing interest in biologically-inspired optimization algorithms. These algorithms draw their principles from natural selection and the survival of the fittest, concepts rooted in the biological world. Known collectively as Evolutionary Algorithms (EAs)[1], these techniques are capable of performing highly complex search and optimization tasks.

The initial application of evolutionary principles in optimization can be traced back to the work of Rechenborg and Schwefel[2],[3]. In their pioneering efforts, they

utilized evolution to find approximate solutions for the shape design of airplane wings. The specific method they employed was called the Evolutionary Strategy. This approach involves exploring new candidate solutions by applying distributed mutation—a process where random changes are introduced to the current solution to create variations. If a newly generated point demonstrates better fitness, it is selected as the new solution, and the search continues from this improved point. The simplicity of this technique, combined with its low computational demands, makes it particularly appealing.

Following the development of Evolutionary Strategy, a variety of other evolutionary algorithms emerged. Among these are Genetic Algorithms (GA)[4], which simulate the process of natural evolution by combining selection, crossover, and mutation to evolve solutions over successive generations. Genetic Programming (GP) extends this concept by evolving computer programs to solve problems. Another notable algorithm is Ant Colony Optimization (ACO), inspired by the foraging behavior of ants, which uses pheromone trails to find optimal paths. Differential Evolution (DE) [5]is another significant algorithm that optimizes problems by iteratively improving candidate solutions based on the differences between randomly selected pairs of solutions. Particle Swarm Optimization (PSO), inspired by the social behavior of birds flocking or fish schooling, optimizes a problem by having a population of candidate solutions, called particles, move around in the search-space according to simple mathematical formulas over the particle's position and velocity. Each algorithm brings unique mechanisms and strategies to the table, enhancing the toolbox available for tackling a wide range of optimization problems.

1.2. Need of Optimization

Optimization is needed because it helps us make the most out of our resources, time, and efforts. In a world where resources are often limited and goals are diverse and complex, optimization provides a structured approach to achieving the best possible outcomes.

A practical optimization problem-solving method must possess certain key characteristics to be effective. Primarily, it should have the capability to identify the true global minimum of the problem, avoiding the pitfall of getting trapped in local optima, which can lead to suboptimal solutions. Additionally, the method should demonstrate a rapid convergence to this global optimum, ensuring efficiency in terms of computational resources and time.

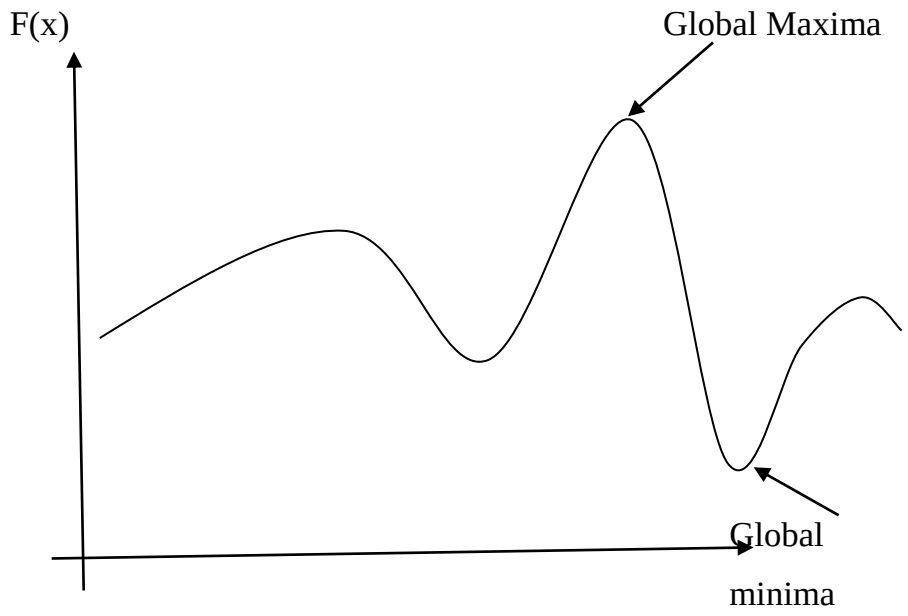


Fig.1.1 Maxima and minima of an optimization problem

One of the widely recognized algorithms that meet these criteria is Differential Evolution (DE[5]). Differential Evolution is a popular evolutionary algorithm (EA) known for its simplicity and robustness. It operates as a population-based stochastic direct search technique, making it well-suited for tackling numerical optimization problems, especially those situated in continuous search spaces.

Firstly, optimization helps us maximize efficiency. By finding the most effective ways to do things, whether it's manufacturing goods, organizing workflows, or allocating budgets, optimization ensures that we use our resources wisely and minimize waste.

Secondly, optimization allows us to navigate through constraints and limitations. Whether it's physical constraints like time, space, or materials, or abstract constraints

like budgetary restrictions or regulatory requirements, optimization helps us find solutions that work within these boundaries.

Thirdly, optimization helps us make informed decisions. By analyzing data, running simulations, or using mathematical models, optimization provides insights into complex systems and helps us understand the consequences of different choices. This enables us to make better decisions that are grounded in evidence and logic.

Finally, optimization fosters innovation and continuous improvement. By constantly seeking better solutions and refining our processes, optimization drives progress and pushes the boundaries of what's possible. It encourages creativity, experimentation, and learning from both successes and failures.

Overall, optimization is essential because it empowers us to do more with less, overcome challenges, and achieve our goals more effectively and efficiently. Whether it's in business, science, engineering, or everyday life, optimization helps us unlock our full potential and strive for excellence.

1.3. Evolutionary Algorithms (EAs)

Evolutionary algorithms (EAs) represent a group of optimization techniques that draw inspiration from the natural processes of biological evolution. These algorithms aim to discover the best possible solution to a given problem by emulating the mechanisms of natural selection, reproduction, mutation, and recombination.

To start, EAs create an initial population composed of a diverse set of potential solutions. Each individual in this population is evaluated for its "fitness," which measures how well it addresses or solves the problem at hand. The fitness evaluation is crucial as it determines the quality and effectiveness of each solution.

Once the initial fitness levels are assessed, the evolutionary process begins. EAs use selection methods to choose the fittest individuals from the current population to act

as parents for the next generation. These parents are then subjected to genetic operators such as crossover (or recombination) and mutation. Crossover involves combining parts of two or more parent solutions to produce offspring that may inherit the best traits of each parent. Mutation introduces small random changes to individual solutions, which helps to maintain genetic diversity within the population and allows the algorithm to explore a wider solution space.

This process of selection, crossover, and mutation is repeated over numerous generations. With each new generation, the algorithm evaluates the fitness of the solutions and continually selects the best-performing ones to propagate. Over time, this iterative cycle allows the population of solutions to evolve and improve, gradually converging towards an optimal or near-optimal solution for the problem being addressed.

The strength of evolutionary algorithms lies in their ability to search large and complex spaces efficiently, making them suitable for a wide range of optimization problems where traditional methods may struggle. By leveraging the power of simulated evolution, EAs can adapt to the specific characteristics of the problem, ensuring that the solutions evolve in a direction that enhances performance and effectiveness.

In summary, evolutionary algorithms harness the principles of biological evolution to solve optimization problems. By generating an initial population, evaluating fitness, and iteratively applying genetic operators such as mutation and crossover, EAs evolve solutions that are increasingly closer to the optimal solution. This approach allows for the exploration of diverse potential solutions and helps to achieve high-quality results in complex optimization tasks.

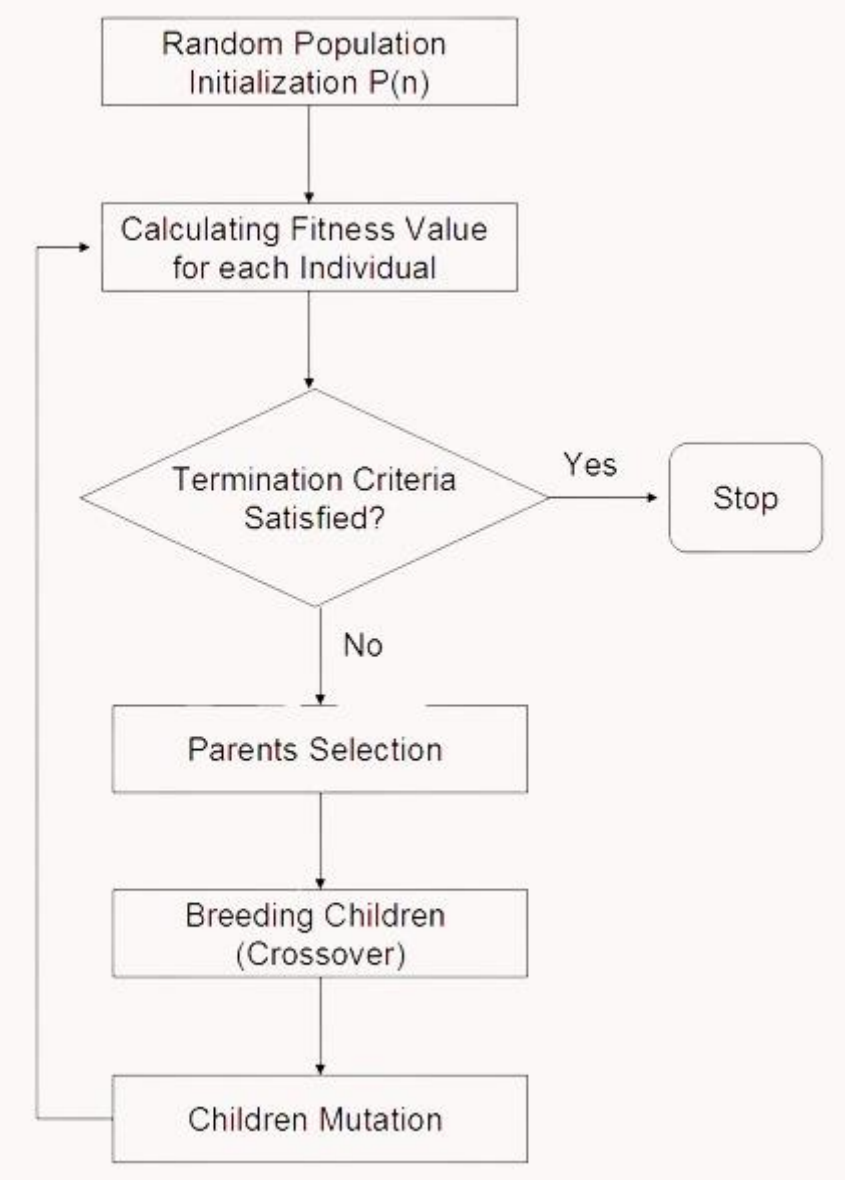


Fig.1.2.Flowchart of EA

1.4. Why EA?

We use EA because they offer a powerful and flexible approach to solve complex optimization problems across various domains. Here are a few reasons why EAs are commonly employed:

1. Versatility: EAs can handle a wide range of optimization problems, including those with non-linear, non-convex, or multimodal objective functions. They are not constrained by specific problem structures or assumptions, making them applicable to

diverse real world.

2. Global Search: EAs are effective in exploring large solutions spaces to find globally optimal or near-optimal solutions. By maintaining a diverse population and incorporating mechanisms like mutation, they can search extensively across the solution space, avoiding getting stuck in local optima.

3. **Robustness:** EAs are robust to noisy or uncertain environments and can handle problems with incomplete or noisy information. Their stochastic nature allows them to adapt and navigate through uncertain landscapes, making them suitable for real world optimization tasks where data may be imperfect or dynamic.

4. **Parallelism:** EAs are inherently parallelizable, allowing them to exploit the computational power of modern hardware and distributed computing environments. This parallelism accelerates the search process, enabling faster convergence towards optimal solutions, especially for computationally intensive problems.

5. **Easy Implementations:** EAs have a simple conceptual framework and are relatively easy to implement and customize for specific problem domains. They do not require extensive Mathematical modeling or domain specific knowledge, making them accessible to practitioners across various fields.

1.5. Classification of EA

Evolutionary algorithms (EAs) can be classified based on whether they are designed to solve single-objective or multi-objective optimization problems.

1. **Single-Objective Evolutionary Algorithms (SOEAs):**Single-objective evolutionary algorithms focus on optimizing a single objective function. They aim to find a single solution that maximizes or minimizes the objective function within the problem's constraints.

2. **Multi-Objective Evolutionary Algorithms (MOEAs):** Multi-objective evolutionary algorithms are designed to handle optimization problems with multiple conflicting objectives, where there is no single optimal solution but a set of trade-off solutions known as the Pareto frontier. MOEAs aim to find a diverse set of solutions

that represent the trade-offs between conflicting objectives.

MOEAs are widely used in various real-world applications where decision-makers need to consider multiple conflicting objectives simultaneously, such as engineering design, portfolio optimization, and resource allocation. They provide a valuable tool for generating a range of trade-off solutions that enable decision-makers to make informed decisions based on their preferences and constraints.

1.6. Single Objective Optimization

Single objective optimization refers to the process of optimizing a system or process by focusing on a single goal. This means identifying the most effective or efficient solution that maximizes or minimizes a specific criterion, such as maximizing profit, minimizing cost, or optimizing performance, while disregarding other potentially conflicting objectives. It involves mathematical modeling and algorithmic techniques to systematically search for the optimal solution within the constraints of the problem domain.

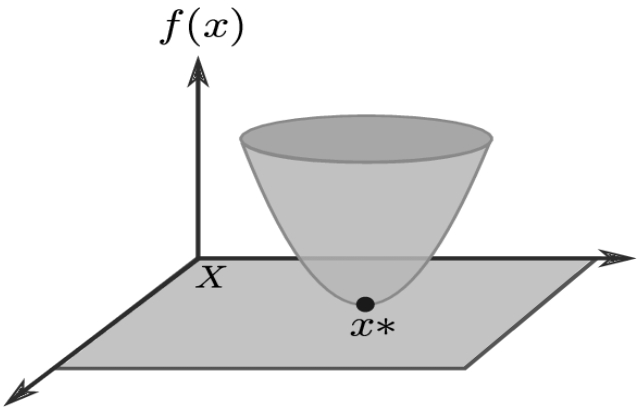


Fig. 1.3.Single objective optimization problem[6]

1.6.1. Differential Evolution

Differential Evolution (DE[5]) as a part of single-objective optimization focuses on finding the best solution to a single optimization problem. Here's a detailed

explanation of how DE operates within the context of single-objective optimization:

1. Initialization: DE[5] starts by generating an initial population of candidate solutions randomly within the search space. Each candidate solution is represented as a vector of real-valued variables.

2. Evaluation: The fitness of each candidate solution in the population is evaluated using the objective function of the optimization problem. This function quantifies how well each solution performs with respect to the single optimization objective.

3. Mutation: DE employs a mutation strategy to explore the search space. At each iteration, trial solutions are generated by perturbing the population using a mutation operator. This involves creating new candidate solutions by combining information from different individuals in the population. The mutation operator typically involves selecting three distinct individuals randomly and combining them to produce a mutant vector.

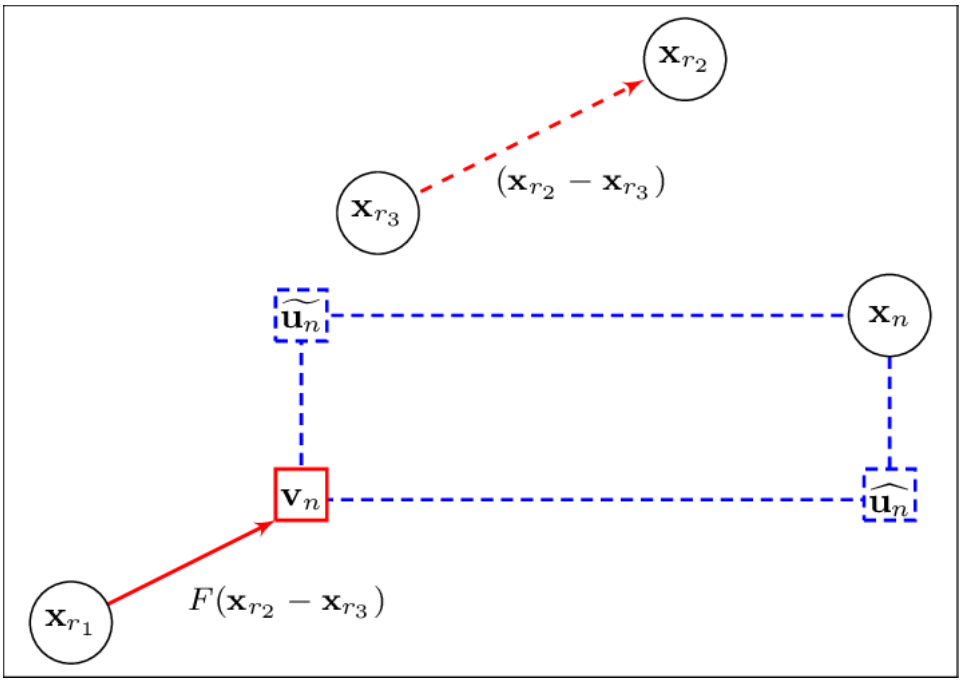


Fig. 1.4. Illustration of mutation (red) and crossover (blue) in differential evolution [7]

4. Crossover: After mutation, DE[5] blends the trial solutions with the original population through a crossover operation. This step helps to balance exploration (searching for new regions in the solution space) and exploitation (refining promising

solutions). The crossover operation combines information from the trial solution and the original solution to produce offspring solutions.

5. Selection: The selection process determines which individuals are selected for the next generation based on their fitness values. DE[5] selects individuals with higher fitness values, favoring solutions that perform better with respect to the optimization objective. The selected individuals become part of the next generation, replacing less fit individuals in the population.

6. Termination: DE[5] iteratively repeats the mutation, crossover, and selection steps until a termination condition is met. This condition could be a maximum number of iterations, reaching a specified level of convergence, or satisfying certain problem-specific criteria. Once the termination condition is met, DE[5] returns the best solution found during the optimization process as the final solution to the single-objective optimization problem.

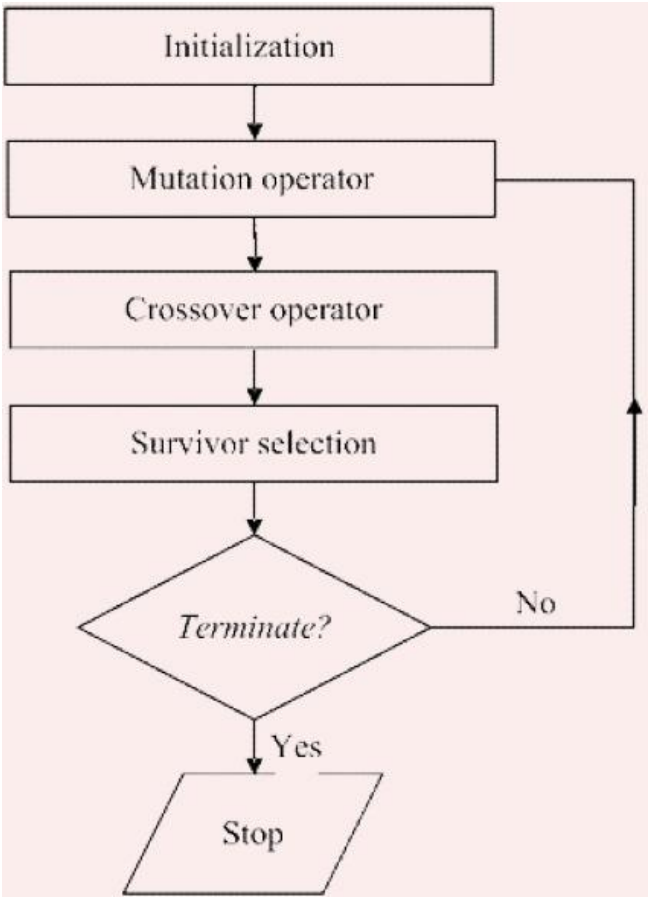


Fig.1.5. Flowchart of Differential Evolution

Overall, Differential Evolution is a powerful optimization algorithm for single-objective optimization problems, capable of efficiently exploring the search space to find high-quality solutions that optimize the single optimization objective.

1.6.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behavior of birds flocking or fish schooling. In the context of single-objective optimization problems, PSO aims to find the optimal solution within a search space by iteratively adjusting the positions of particles in the space. Here's how PSO works in the context of single-objective optimization:

1. Initialization: PSO begins by initializing a population of particles randomly within the search space. Each particle represents a potential solution to the optimization problem. Additionally, each particle has its own velocity, which determines how it moves through the search space.

2. Fitness Evaluation: The fitness of each particle is evaluated using the objective function of the optimization problem. This function quantifies how well each particle performs with respect to the single optimization objective.

3. Initialization of Particle's Best Position: Each particle maintains its own best position, known as its personal best. Initially, this is set to the particle's current position.

4. Initialization of Global Best Position: PSO also tracks the best position found by any particle in the population, known as the global best position. Initially, this is set to the position of one of the particles in the population.

5. Velocity Update: At each iteration, each particle adjusts its velocity based on its current velocity, its distance to its personal best position, and the distance to the global best position. This update equation encourages particles to move towards their personal best positions and the global best position.

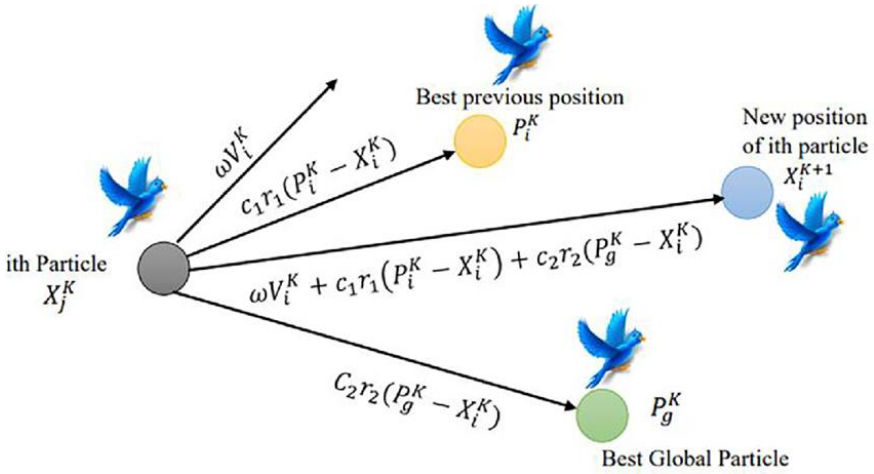


Fig 1.6. Particle Swarm Optimization Architecture Illustration.[8]

6. Position Update: After updating velocities, each particle updates its position by moving according to its new velocity. This movement allows particles to explore the search space in search of better solutions.

7. Update Personal and Global Bests: After updating positions, each particle compares its new position to its personal best position. If the new position is better, the particle updates its personal best position. Additionally, the global best position is updated if any particle finds a better solution than the current global best.

8. Termination: PSO iteratively repeats the velocity and position updates until a termination condition is met. This condition could be a maximum number of iterations, reaching a specified level of convergence, or satisfying certain problem-specific criteria.

9. Final Solution: Once the termination condition is met, PSO returns the best solution found during the optimization process as the final solution to the single-objective optimization problem.

Overall, PSO is a population-based optimization algorithm that effectively explores the search space by iteratively adjusting the positions of particles based on their own experience and the experience of the entire population, ultimately converging towards the optimal solution to the single-objective optimization problem.

1.7. Multi Objective Evolutionary Algorithm

Multi-objective evolutionary algorithms (MOEAs) are a class of optimization techniques used to solve problems with multiple conflicting objectives. In contrast to single-objective optimization, where there's only one criterion to optimize, multi-objective optimization involves optimizing multiple objectives simultaneously, often with conflicting goals.

Here's how MOEAs work:

1**. Initialization:** Similar to single-objective evolutionary algorithms, MOEAs start by generating an initial population of candidate solutions.

2**. Evaluation:** Each candidate solution is evaluated based on multiple objective functions, representing different criteria or goals. These objectives may conflict with each other, meaning improving one objective may degrade another.

3**. Selection:** MOEAs use specialized selection mechanisms to determine which solutions are better suited for advancing to the next generation. Unlike single-objective optimization, where solutions are ranked based on a single objective, in multi-objective optimization, solutions are ranked based on their performance across multiple objectives. Common selection mechanisms include Pareto dominance and fitness assignment based on non-dominated sorting.

4. **Variation:** New candidate solutions are generated through variation operators such as crossover and mutation, similar to single-objective evolutionary algorithms. However, in MOEAs, these operators need to preserve diversity among solutions to ensure a well-distributed set of solutions across the objective space.

5. **Replacement:** The new generation of solutions replaces the old one, maintaining diversity to ensure a representative set of solutions across the objective space.

6. **Termination:** The optimization process continues for a predetermined number of generations or until a termination condition is met. In MOEAs, termination may be based on convergence to a Pareto-optimal front, where no further improvement in any

objective is possible without worsening at least one other objective.

The goal of MOEAs is to find a set of solutions that represent a trade-off between conflicting objectives, known as the Pareto-optimal front. These solutions provide decision-makers with a range of trade-off options, allowing them to make informed decisions based on their preferences and priorities.

MOEAs find applications in various domains, including engineering design, finance, and scheduling, where decision-makers need to consider multiple conflicting objectives simultaneously. They offer a powerful approach for exploring the complex trade-offs inherent in many real-world optimization problems.

1.7.1. Non-dominated Sorting Genetic Algorithm II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a popular multi-objective optimization algorithm designed to find a diverse set of solutions that approximate the Pareto-optimal front. The Pareto-optimal front represents the trade-offs between conflicting objectives, where improving one objective comes at the expense of worsening another. Here's how NSGA-II works in the context of multi-objective optimization:

1. Initialization: NSGA-II begins by randomly generating an initial population of candidate solutions within the search space. Each candidate solution is represented as a vector of decision variables.

2. Fitness Evaluation: The fitness of each candidate solution is evaluated based on its dominance rank and crowding distance. Dominance rank determines whether one solution dominates another, while crowding distance encourages diversity in the population by measuring how crowded a solution is with respect to its neighbors in the objective space.

3. Non-dominated Sorting: NSGA-II performs a non-dominated sorting of the population to categorize solutions into different Pareto fronts. Solutions in the first Pareto front are non-dominated, meaning they are not worse than any other solution in the population with respect to all objectives. Solutions in subsequent fronts are

progressively dominated by solutions in earlier fronts.

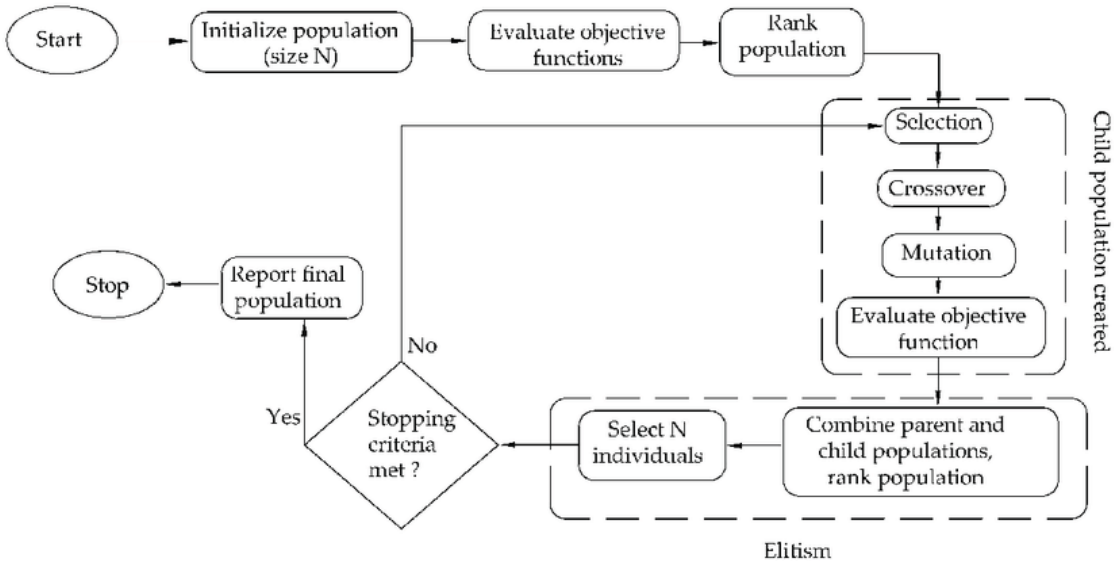


Fig.1.7.Flowchart of non-dominated sorting genetic algorithm II (NSGA-II[9]

4. Crowding Distance Assignment: Solutions within each Pareto front are ranked based on their crowding distance, which measures how closely they are packed together in the objective space. Solutions with higher crowding distance are preferred because they contribute to maintaining diversity in the population.

5. Selection: NSGA-II selects individuals for the next generation using a combination of non-dominated sorting and crowding distance. Solutions from less crowded regions of the Pareto front are preferred to maintain diversity while preserving solutions from the best Pareto fronts.

6. Crossover and Mutation: Selected individuals undergo genetic operators such as crossover and mutation to produce offspring solutions. These operators introduce diversity into the population and facilitate exploration of the search space.

7. Replacement: The offspring solutions are combined with the parent population to form the next generation. NSGA-II uses a combination of elitist selection and crowded comparison to ensure that the population size remains constant and the best solutions are preserved across generations.

8. Termination: NSGA-II iteratively repeats the selection, crossover, mutation, and

replacement steps until a termination condition is met. This condition could be a maximum number of generations, reaching a specified convergence criterion, or satisfying certain problem-specific criteria.

9. Final Solution: Once the termination condition is met, NSGA-II returns a diverse set of solutions that approximate the Pareto-optimal front, providing decision-makers with a range of trade-off solutions to choose from.

Overall, NSGA-II is a powerful multi-objective optimization algorithm that effectively balances convergence and diversity, allowing it to efficiently explore the search space and identify a diverse set of high-quality solutions that represent the trade-offs between conflicting objectives.

1.8. Many Objective Optimization

Multi-objective optimization and many-objective optimization are closely related concepts, but they differ in terms of the number of objectives involved and the complexity of the optimization problem.

In multi-objective optimization, we typically deal with a small to moderate number of conflicting objectives, often ranging from two to maybe five or six. The focus is on finding a set of solutions that represent a trade-off between these objectives, known as the Pareto frontier. Each solution in this frontier is considered optimal because it cannot be improved in any one objective without worsening another. Multi-objective optimization is commonly used in decision-making processes where there are multiple, but manageable, conflicting goals to consider.

On the other hand, many-objective optimization involves dealing with a large number of conflicting objectives, typically beyond what can be easily managed or visualized. The number of objectives can range from several to dozens or even hundreds. The challenge here is not only to find a trade-off between objectives but also to navigate through a vast solution space efficiently to identify the best possible solutions. Many-objective optimization often requires advanced algorithms and techniques to handle the increased complexity and dimensionality of the problem.

In summary, while both multi-objective optimization and many-objective optimization deal with multiple conflicting objectives, many-objective optimization tackles a higher level of complexity due to the sheer number of objectives involved.

1.9. Why traditional EMOO algorithm can’t solve MaOO problem

Traditional Evolutionary Multi-Objective Optimization (EMOO) algorithms may face challenges when applied to Many-Objective Optimization (MaOO) problems due to several reasons:

1. **Curse of Dimensionality:** As the number of objectives increases in MaOO problems, the size of the Pareto front grows exponentially. Traditional EMOO algorithms may struggle to efficiently explore and maintain diversity in high-dimensional objective spaces.

2. **Convergence and Diversity Trade-off:** EMOO algorithms typically aim to converge towards the true Pareto front while maintaining diversity among solutions. In MaOO problems, balancing convergence and diversity becomes increasingly challenging as the number of objectives increases. Traditional EMOO algorithms may prioritize convergence at the expense of diversity or vice versa, leading to suboptimal solutions.

3. **Limited Population Size:** Traditional EMOO algorithms often use fixed population sizes, which may not scale well to MaOO problems with a large number of objectives. Limited population sizes can restrict the ability of the algorithm to adequately sample the Pareto front and may lead to premature convergence or loss of diversity.

4.**Selection Pressure:** Selection mechanisms in EMOO algorithms, such as tournament selection or elitist selection, may exert strong pressure towards solutions with better fitness values. In MaOO problems, where the Pareto front may exhibit complex and irregular shapes, traditional selection mechanisms may struggle to maintain diversity and explore the entire front.

5. **Fitness Assignment:** Assigning fitness values to solutions in MaOO problems becomes increasingly challenging as the number of objectives grows. Traditional metrics for comparing solutions, such as hypervolume or crowding distance, may become less effective in MaOO problems with a large number of objectives, leading to inaccurate assessments of solution quality.

6. **Algorithm Scalability:** Traditional EMOO algorithms may lack scalability to handle MaOO problems efficiently. As the number of objectives increases, computational complexity grows, and traditional algorithms may become computationally prohibitive or impractical to use.

To address these challenges, specialized algorithms and techniques tailored for Many-Objective Optimization have been developed. These algorithms often incorporate strategies to enhance diversity maintenance, improve convergence performance, and efficiently handle high-dimensional objective spaces. Examples include indicator-based algorithms, decomposition-based algorithms, and evolutionary algorithms specifically designed for MaOO problems.

1.10. Literature Review

Evolutionary multi objective optimization (EMO) is a prominent research area within evolutionary computation. Numerous EMO algorithms have been developed and applied successfully across various real-world tasks[10-13]. These algorithms perform exceptionally well with two-objective problems, but their efficiency declines as the number of objectives increases. When a problem involves four or more objectives, it is often termed a many-objective problem. Using well-known Pareto dominance-based EMO algorithms for these problems can present several significant challenges:

1. **Reduced Search Ability:** Algorithms like SPEA[14] and NSGA-II[15], which rely on Pareto dominance, face a major issue when the number of objectives increases. In such cases, almost all solutions in a population become non-dominated, diminishing the selection pressure towards the Pareto front and impairing the convergence of these algorithms.

2. **Exponential Increase in Solutions:** The number of solutions required to approximate the entire Pareto front increases exponentially with the number of objectives. This is because the Pareto front becomes a hyper-surface in a high-dimensional objective space, necessitating thousands of non-dominated solutions for an accurate approximation.

3. **Visualization Challenges**: Selecting a final solution from a set of non-dominated solutions usually depends on a decision maker's preferences. With more objectives, visualizing these solutions becomes increasingly difficult, complicating the decision-making process.

The deterioration of search ability in EMO algorithms as the number of objectives increases has been well-documented in various studies [16-17]. This issue is often highlighted through comparisons with multiple runs of single-objective optimizers [18-20]. To address scalability issues in many-objective problems, several strategies have been proposed:

Increasing Selection Pressure: One method involves modifying Pareto dominance to reduce the number of non-dominated solutions [21] per population. Another method [22-26]assigns different ranks to non-dominated solutions.

Alternative Fitness Evaluation Mechanisms: Instead of Pareto dominance, indicator-based evolutionary algorithms use functions like hypervolume [27-28] for solution evaluation. Another approach employs various scalarizing[18],[19],[29-31] functions for fitness evaluation.

The exponential growth in the number of non-dominated solutions necessary for approximating the Pareto front has often been tackled by integrating preference information into EMO algorithms[32-34]. This approach focuses on a specific region of the Pareto front, allowing the algorithms to find multiple non-dominated solutions within that smaller area.

To address the difficulty in visualizing solutions, one approach is to reduce the number of objectives [35-38]. This reduction not only aids visualization but also mitigates other related challenges. Various visualization techniques [39-41] have been proposed, including mapping high-dimensional objective vectors to a lower-dimensional space,

making it easier to visualize many-objective solutions. These methods have been extensively studied in multiple criteria decision making (MCDM)[42].

Traditional multi-objective particle swarm optimization (PSO) research primarily addresses problems with a limited number of variables and fewer than four objectives. Conventional multi-objective PSO algorithms struggle with efficiency when dealing with these more complex problems. To address these multi-objective large-scale optimization problems (MOLSOPs) and many-objective large-scale optimization problems (MaOLSOPs), it is essential to deeply explore the parallel properties [43] of the particle swarm and create innovative PSO algorithms that leverage distributed parallel computation.

1.11. Proposed strategy of Coevolution

Optimization problems involving four or more conflicting objectives are referred to as many-objective optimization (MaOO) problems. These types of problems are becoming increasingly important due to their applications in real-time optimization challenges, such as brain-computer interfacing, bioinformatics, and aerospace engineering. Traditionally, evolutionary multi-objective optimization (EMOO) methods were used to address MaOO problems. However, as the number of objectives increases, the Pareto dominance criteria used in EMOO fail to provide sufficient selection pressure, resulting in almost all members of the population becoming non-dominated.

Before this study, researchers explored several non-Pareto techniques to tackle MaOO problems. However, these methods did not leverage the inherent parallelism of population-based metaheuristic algorithms to speed up convergence. The novel strategy proposed here is parallelism through coevolution techniques.

Coevolution, inspired by biological processes, involves the exchange of evolutionary information between two interacting species. Unlike traditional EMOO, which evolves a single globally distributed population, the proposed approach coevolves multiple populations, each focusing on specific components of multi-modal objective surfaces. This coevolutionary search ensures precise localization of global optima in MaOO

problems.

Building on this concept, the study aims to optimize each of the N objectives of the given MaOO problem in parallel, using N evolutionary algorithms. This approach effectively harnesses the built-in parallelism of population-based search. The main idea is that it is unrealistic to expect each population member to optimize all objectives equally well. Instead, it is more reasonable to allow each member to evolve with respect to a specific objective. The artificial bee colony (ABC) algorithm, known for its effectiveness in global optimization, is chosen as the evolutionary framework for the proposed MaOO algorithm.

1.12. Objective of the thesis

The objective of the thesis is to develop and validate a novel many-objective optimization (MaOO) approach that leverages the inherent parallelism of the evolutionary process through a coevolution strategy. This new approach, called the fuzzy-bee colony (FBC) algorithm, employs the artificial bee colony (ABC) algorithm to handle each objective in parallel. After convergence, high-quality solutions from each ABC population are selected and combined into a union set. A fuzzy membership-induced rank measure is then used to identify the top-ranked members of this set, representing approximate Pareto optimal solutions. The FBC algorithm is compared against three state-of-the-art methods, with experimental results demonstrating its superior performance across various metrics.

1.13. Organization of Thesis

Chapter 2 includes the the proposed strategy to solve MaOO problems. This chapter offers a unique coevolution based MaOO approach. A brief introduction to Fuzzy Coevolution strategy along with Mathematical formulation and result is presented in chapter 2. Chapter 3 includes Conclusion and future research scopes. Bibliography is given at the end of each chapter.

Reference:

[1] . L. j. Fogel, A. J. Owens, M. J. Walsh, “Artificial Intelligence through Simulated Evolution”, John Wiley & Sons, New York, 1966.

[2]. I. Rechenberg, “Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution”, Formman-Holzboog, Stuttgart, 1973.

[3]. H. P. Schefel, “Numerical Optimization of Computer Models”, John Wiley & Sons Ltd., Chichester, 1981.

[4]. M. Mitchell, “An Introduction to Genetic Algorithms”, MIT Press, 1996.

[5]. R. Storn and K. V. Price, “Differential evolution-A simple and efficient heuristic for global optimization over continuous Spaces,” Journal of Global Optimization, vol.11, pp.341-359. 1997.

[6]. Jes´us-Adolfo Mej´ıa-de-Dios and Efr´en Mezura-Montes. A Physics-Inspired Algorithm for Bilevel Optimization

[7] Daniel J. Poole, C.B.Allen, Thomas Rendall,”Identifying Multiple Optima in Aerodynamic Design Spaces”, Conference: AIAA Aviation and Aeronautics Forum and Exposition,2018.

[8] Yazan Al-Smadi a , Mohammed Eshtay b , Ahmad Al-Qerem a , Shadi Nashwan c, , Osama Ouda c , A.A. Abd El-Aziz d,e, “Reliable prediction of software defects using Shapley interpretable machine learning models”, Egyptian Informatics Journal 2023.

[9] Dhakal, Resham & Zhou, Jianxu & Palikhe, Sunit & Bhattarai, Khem. (2020). Hydraulic Optimization of Double Chamber Surge Tank Using NSGA-II. Water. 12. 455. 10.3390/w12020455.

[10] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Chichester, 2001.

[11] C. A. C. Coello, D. A. van Veldhuizen, and G. B. Lamont, Evolutionary Algorithms *f*or Solving Multi-Objective Problems, Kluwer, Boston, 2002.

[12] C. A. C. Coello and G. B. Lamont, Applications o*f* Multi-Objective Evolutionary Algorithms, World Scientific, Singapore, 2004.

[13] A. Abraham, L. C. Jain, and R. Goldberg (eds.), Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Springer, Berlin, 2005.

[14] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," IEEE Trans. on Evolutionary Computation 3 (4), pp. 257-271, November 1999.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. on Evolutionary Computation 6 (2), pp. 182-197, April 2002.

[16] V. Khara, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," Lecture Notes in Computer Science 2632: Evolutionary Multi-Criterion Optimization - EMO 2003, pp. 367-390, Springer, Berlin, April 2003.

[17] R. C. Purshouse and P. J. Fleming, "Evolutionary many-objective optimization: An exploratory analysis," Proc. o*f* 2003 IEEE Congress on Evolutionary Computation, pp. 2066-2073, Canberra, December 8-12, 2003.

[18] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics: The knapsack problem case study," European Journal o*f* Operational Research 158 (2), pp. 418-433, October 2004.

[19] E. J. Hughes, "Evolutionary many-objective optimization: Many once or one many?," Proc. o*f* 2005 IEEE Congress on Evolutionary Computation, pp. 222-227, Edinburgh, September 2-5, 2005.

[20] H. Ishibuchi, Y. Nojima, and T. Doi, "Comparison between single- objective

and multi-objective genetic algorithms: Performance comparison and performance measures," Proc. of 2006 IEEE Congress on Evolutionary Computation, pp. 3959-3966, Vancouver, July 16-21, 2006.

[21] H. Sato, H. E. Aguirre, and K. Tanaka, "Controlling dominance area of solutions and its impact on the performance of MOEAs," Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization-EMO 2007, pp. 5-20, Springer, Berlin, March 2007.

[22] N. Drechsler, R. Drechsler, and B. Becker, "Multi-objective optimization based on relation favour," Lecture Notes in Computer Science 1993: Evolutionary Multi-Criterion Optimization - EMO 2001,pp. 154-166, Springer, Berlin, March 2001.

[23]A. Sülflow, N. Drechsler, and R. Drechsler, "Robust multi-objective optimization in high dimensional spaces," Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007,pp. 715-726, Springer, Berlin, March 2007.

[24] M. Köppen and K. Yoshida, "Substitute distance assignments in NSGA-II for handling many-objective optimization problems," Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 727-741, Springer, Berlin, March 2007.

[25] D. Corne and J. Knowles, "Techniques for highly multiobjective optimization: Some non-dominated points are better than others," Proc. of 2007 Genetic and Evolutionary Computation Conference, pp. 773-780, London, July 7-11, 2007.

[26] S. Kukkonen and J. Lampinen, "Ranking-dominance and many- objective optimization," Proc. of 2007 IEEE Congress on Evolutionary Computation, pp. 3983-3990, Singapore, September 25-28, 2007.

[27] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 742-756,

Springer, Berlin, March 2007.

[28] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, “Iterative approach to indicator-based multiobjective optimization,” Proc. of 2007 IEEE Congress on Evolutionary Computation, pp. 3697-3704, Singapore, September 25-28, 2007.

[29] H. Ishibuchi, T. Doi, and Y. Nojima, “Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms,” Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX, pp. 493-502, Springer, Berlin, September 2006.

[30] H. Ishibuchi and Y. Nojima, “Optimization of scalarizing functions through evolutionary multiobjective optimization,” Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 51-65, Springer, Berlin, March 2007.

[31] E. J. Hughes, “MSOPS-II: A general-purpose many-objective optimiser,” Proc. of 2007 IEEE Congress on Evolutionary Computation, pp. 3944-3951, Singapore, September 25-28, 2007.

[32] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, “Many-objective optimization: An engineering design perspective,” Lecture Notes in Computer Science 3410: Evolutionary Multi-Criterion Optimization - EMO 2005, pp. 14-32, Springer, Berlin, March 2005.

[33] K. Deb and J. Sundar, “Preference point based multi-objective optimization using evolutionary algorithms,” Proc. of 2006 Genetic and Evolutionary Computation Conference, pp. 635-642, Seattle, July 8-12, 2006.

[34] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina, “A preference- based interactive evolutionary algorithm for multiobjective optimization,” Helsinki School of Economics, Working Paper, W-412, January 2007.

[35] K. Deb and D. K. Saxena, “On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization

problems," KanGAL Report, No. 2005011, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology Kanpur, 2005.

[36] K. Deb and K. Saxena, "Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems," Proc. of 2006 IEEE Congress on Evolutionary Computation, pp. 3353-3360, Vancouver, July 16-21, 2006.

[37] D. Brockhoff and E. Zitzler, "Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization," Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX, pp. 533-542, Springer, Berlin, September 2006.

[38] D. Brockhoff and E. Zitzler, "Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods," Proc. of 2007 IEEE Congress on Evolutionary Computation, pp.2086-2093, Singapore, September 25-28, 2007.

[39] S. Obayashi and D. Sasaki, "Visualization and data mining of Pareto solutions using self-organizing map," Lecture Notes in Computer Science 2632: Evolutionary Multi-Criterion Optimization - EMO 2003,pp. 796-809, Springer, Berlin, April 2003.

[40] T. Yoshikawa and T. Furuhashi, "Visualization techniques for mining of solutions," Proc. of 8th International Symposium on Advanced Intelligent Systems, pp. 68-71, Sokcho, Korea, September 5-8, 2007.

[41] M. Köppen and K. Yoshida, "Visualization of Pareto-sets in evolutionary multi-objective optimization," Proc. of 7th International Conference on Hybrid Intelligent Systems, Kaiserslautern, Germany, pp. 156-161, September 17-19, 2007.

[42] K. Miettinen, Nonlinear Multiobjective Optimization, Kluwer, Boston, 1998.

[43] B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, and K. Kang, "Distributed parallel particle swarm optimization for multiobjective and many-objective large-scale optimization," IEEE Access, vol. 5, 2017, pp. 8214-8221.

Chapter 2: Fuzzy Bee Colony

2.1. Introduction

Many-objective optimization (MaOO) [1] refers to optimization problems that involve four or more conflicting objectives. These MaOO approaches are gaining prominence due to their application in various real-time optimization challenges, such as brain-computer interfacing[2], bioinformatics[3], and aerospace engineering[4]. Traditionally, evolutionary multi-objective optimization (EMOO) methods have been employed to tackle MaOO problems. However, as the number of objectives increases, the Pareto dominance criteria used in EMOO lose their effectiveness in generating sufficient selection pressure[5],[6], resulting in nearly all population members becoming non-dominated.

To address this challenge, researchers have explored several non-Pareto techniques, including subspace dominance comparison[10], and fuzzy Pareto dominance[11]. Despite these efforts, these methods have not fully exploited the inherent parallelism of population-based metaheuristic algorithms to accelerate convergence.

This thesis introduces a novel strategy that leverages this parallelism through coevolution techniques to solve MaOO problems. Coevolution, inspired by biological processes, involves the exchange of evolutionary knowledge between interacting species. Instead of evolving a single, globally distributed population as in traditional EMOO, the proposed approach coevolves multiple populations, each targeting specific components of multi-modal objective surfaces. This co-evolutionary search ensures the

precise localization of global optima in MaOO problems.

The main objective of this research is to optimize individual objectives of a MaOO problem in parallel by employing multiple evolutionary algorithms. This approach effectively utilizes the built-in parallelism of population-based searches. It is unrealistic to assume that each population member in an evolutionary search can optimize all objectives equally well. Therefore, it is more reasonable for each member to evolve concerning a specific objective. The artificial bee colony (ABC) algorithm [12],[13]is selected as the evolutionary framework for the proposed MaOO algorithm due to its proven effectiveness in global optimization. The merit of this research is evident in several key aspects:

1. **Parallel Objective Optimization:** Existing MaOO techniques typically use a single population, which may not effectively optimize all objectives in high-dimensional spaces. By optimizing individual objectives in parallel, the proposed approach enhances diversity and convergence through thorough exploration of each fitness landscape.

2. **Fuzzy Rule-Based Ranking:** A novel fuzzy rule-based ranking procedure is developed to rank members of each population, forming a pool of equally good solutions for the MaOO problem. Fuzzy membership is assigned to each solution to measure its optimization extent for each objective. This approach eliminates the need for Pareto ranking, addressing the reduction in selection pressure in MaOO problems.

3. **Enhanced ABC Algorithm:** To address diversity concerns, the traditional ABC algorithm is improved through two extensions. First, the mutation policy is adjusted to facilitate communication between high-quality members of different populations, promoting efficient coevolution and guiding the search towards promising regions. Second, parental selection is replaced by a niching mechanism to maintain population diversity and improve overall solution quality.

The proposed MaOO algorithm, referred to as the fuzzy-bee colony (FBC), is compared with three state-of-the-art[15],[16],[17] techniques in optimizing the DTLZ[18] and WFG[19] test suites, which include a distinct number of objectives. Experiments reveal that the proposed coevolution-based MaOO algorithm outperforms

its competitors regarding inverted generational distance [20] and hypervolume metrics.[21]

2.2. Definitions of fuzzy set

Here we have definitions of a fuzzy set and the α-cut of a fuzzy set:

Definition 1: A fuzzy set A[14] can be described as a collection of ordered pairs (z, $\mu_A(z)$), where z belongs to the universe of discourse P. The function $\mu_A(z)$, which takes values between 0 and 1, indicates the degree to which z is a member of the set A .

$$A = \{Z, \mu_A(Z) | Z \in P\} \quad (2.1)$$

Definition 2: The α-cut of a fuzzy set A[14] is a standard (crisp) set consisting of all elements z for which the membership function $\mu_A(z)$ is greater than or equal to a specified threshold α, where α is a chosen value between 0 and 1.

$$A_\alpha = \{Z | \mu_A(Z) \geq \alpha\} \quad (2.2)$$

2.3. ABC algorithm

Imagine a colony of bees searching for the best flowers to collect nectar from. Each bee explores the area, evaluates the nectar quantity of the flowers it visits, and communicates this information with its fellow bees. The entire colony gradually converges towards the best flower patches.

In the Artificial Bee Colony (ABC) [12] algorithm, we simulate this process in optimization problems. Initially, we have a population of artificial bees (solutions), each representing a potential solution to the optimization problem. These artificial bees explore the solution space by adjusting their positions (solution parameters) iteratively.

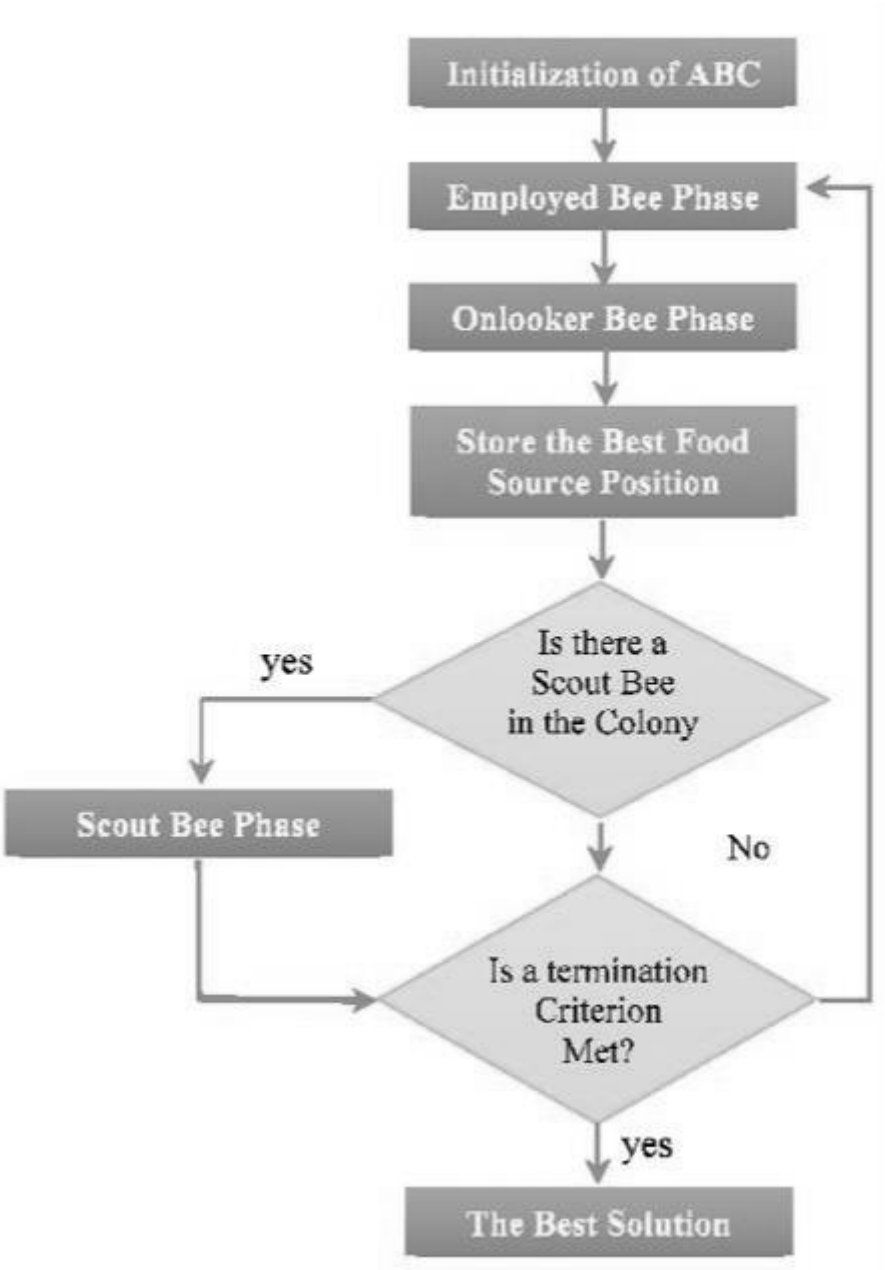


Fig.2.1. Flowchart of ABC algorithm.

During each iteration, artificial bees perform three main actions:

1. Employed Bees(EB): These bees explore the neighborhood of their current solution by slightly modifying it. They evaluate the quality of the new solutions and compare them with the current ones.

2. Onlooker Bees(OB): These bees make decisions based on the information shared by employed bees. They select solutions to explore based on the quality of the solutions found by employed bees.

3. Scout Bees(SB): If a solution has not been improved for a certain number of iterations, it becomes abandoned. Scout bees then generate a new random solution to replace the abandoned one.

Through this process of exploration and exploitation, the ABC algorithm aims to iteratively improve solutions and converge towards the optimal solution of the optimization problem.Overall, the ABC algorithm is a nature-inspired metaheuristic optimization technique that mimics the foraging behavior of honeybees to efficiently search for optimal solutions in complex search spaces.

2.4. Fuzzy coevolution strategy

Here, a three-step approach is used to optimize N objectives in a Multi-objective Optimization (MaOO) problem. First, N distinct populations are evolved simultaneously, with each population concentrating on a different objective function. In the second step, the top solutions from each of the N populations are selected to achieve the best possible outcomes for all objectives. Finally, a fuzzy membership metric is applied to ascertain the optimal solutions.

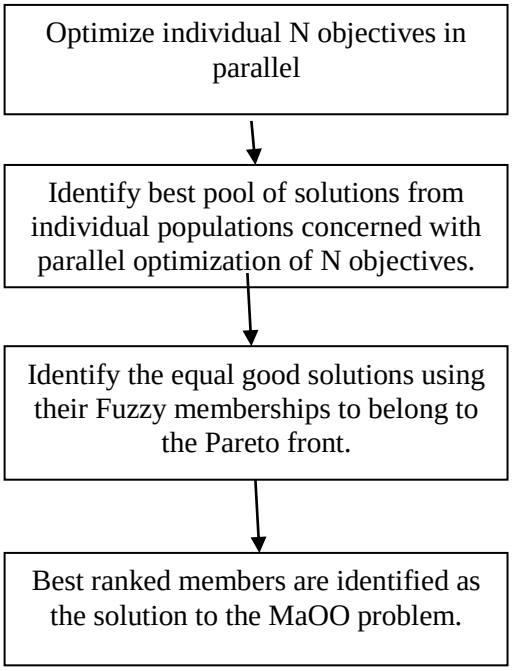


Fig. 2.2. Fuzzy Bee Colony based MaOO

2.4.1. Individual Parallel Optimization of N Objectives

The Artificial Bee Colony (ABC) algorithm focused on optimizing a specific i-th objective, where i is one of (1, 2, ..., N) is referred to as i-ABC. The population and solutions are represented as P^i(g) and Z_k^i(g) for k = 1, 2, ..., S, respectively. The traditional ABC algorithm, described before, is enhanced in two key ways.

First, the mutation strategy, is modified to enable an individual member Z_k^i(g) of the population P^i(g) (with i in {1, 2, ..., N}) to interact with the best members of all populations P^l(g) for l = 1, 2, ..., N. This interaction is achieved by the following equation:

$$Z_{k,j}(g) = Z_{k,j}^i(g) + \sum_{l=1}^{N} rand(-1,1) \times (Z_j^{l-best}(g) - Z_{k,j}^i(g)) \quad (2.3)$$

Here, $Z_j^{l-best}(g)$ denotes the j-th component of the best member of the population P^l(g). This communication among populations across N objectives supports the effective coevolution of solutions in a Multi-objective Optimization (MaOO) problem, thereby allowing $Z_k^i(g)$ to be influenced by the best members of all populations P^l(g).

Second, the traditional ABC's greedy selection mechanism is modified. In the standard approach, $Z_k^i(g)$ is replaced by $Z_k^i(g)$' if $f_i(Z_k^i(g)') < f_i(Z_k^i(g))$ for a minimization problem. Otherwise, $Z_k^i(g)$ remains in the population. However, this selection process does not account for the spatial location of competing members within the search space. Promoting individuals based solely on fitness values, while ignoring their spatial distribution, can lead to reduced population diversity, especially when solutions from sparsely populated regions are discarded. This reduction in diversity can negatively impact optimization performance in multimodal fitness landscapes.

To address this, a niching mechanism is introduced to maintain population diversity and enhance solution quality. The crowding policy of the niching mechanism promotes competition among geographically proximate members within the ecosystem. Specifically, the offspring solution $Z_k^i(g)$' is paired with its nearest

neighbor $Z_j^i(g)$ [22] within the population P^l(g) for the selection phase.

The pseudo-code of the i-ABC based optimization of the i-th objective $f_i(.)$ is given below.

Procedure i-ABC

Input:Population size S, search space dimension D and boundary, optimizing objective function fl(.).

Output:Final population P l after convergence.

Begin

I. Initialization:

(a) Randomly initialize the solutions as Z_k^i(g) of population P^i(g) for k = 1, 2, …, S within the prescribed bound of the D-dimensional search space at generation g = 0.

(b) Evaluate the value of the i-th objective function $f_i(Z_k^i(g))$ for k= 1, 2, …, S.

(c) Set $Z^{i-best}(g)$ ← arg($min_{k=1}^{s}(f_i(Z_k^i(g))$)

II. While stopping criterion is not reached, do begin:

(a) Employed bee phase:

(i) Generate a new solution $Z_k^i(g)$' for each solution $Z_k^i(g)$ using (2.3) for k= 1, 2, …, S

(ii) Evaluate the value of the i-th objective function $f_i(Z_k^i(g))$ for k= 1, 2, …, S.

(iii) Set $Z_k^i(g)$ ← $Z_k^i(g)$' if $f_i(Z_k^i(g)') < f_i(Z_k^i(g))$ for k= 1, 2, …, S.

(b)Calculation of selection probabilities: Compute the probability prob($Z_k^i(g)$) of each solution $Z_k^i(g)$ by the onlooker bee for k= 1, 2, …, S.

(c) Onlooker bee phase:

For i = 1 to S do

Begin

(i) The i-th onlooker selects a solution Z_k^i(g) ∈ P^i(g) based on prob($Z_k^i(g)$).

(ii) Generate a new solution $Z_k^i(g)$' for each solution $Z_k^i(g)$ using (2.3) for k= 1, 2, …, S.

(iii) Evaluate the value of the i-th objective function $f_i(Z_k^i(g)')$ for i= 1, 2, …, S.

(iv) Identify the nearest member Z_k^i(g) ∈ P^i(g) of $Z_k^i(g)'$.

(v) Set $Z_j^i(g+1) \leftarrow Z_k^i(g)'$ if $f_i(Z_k^i(g)') \leq f_i(Z_k^i(g)$

(vi) Set $Z^{i-best}(g) \leftarrow Z_k^i(g)'$ if $f_i(Z_k^i(g)') \leq f_i(Z^{i-best}(g))$.

End For.

(d) Set g←g+1.

End While.

III. Return P^i(g)

2.4.2. Unifying the obtained candidate solution

This step involves selecting a set of high-quality solutions from N populations. After these populations have converged, let f^{i-best} represent the objective function value of the best solution Z^{i-best} in the i-th population, where i = 1, 2,…., N . However, the best solution for one population might not be the best for others that are optimizing different objectives. To address this, instead of choosing only the best solution, a group of high-quality solutions is selected from each population. This ensures that solutions slightly worse than the best can still be strong candidates for other populations. The process for forming these groups is detailed below. The generation index g is omitted from now on since we are dealing with the final population p^i of each i-ABC after convergence.

1. Calculate the grade of membership for each solution in p^i to belong to group A which needs to be formed after convergence using the formula:

$$\mu_A(Z_k^i) = [1 - \frac{f(Z_k^i) - f^{i-min}}{f^{i-max} - f^{i-min}}] \qquad (2.4)$$

for k = 1, 2,……, S . Here, f^{i-max} and f^{i-min} denote the maximum and minimum values of the i-th objective function obtained from p^i after convergence. Note that $f^{i-min} = f^{i-best}$ for minimization problems, so Z^{i-best} is assigned the highest membership value of 1. As the quality of a solution in p^i reduces with increasing objective function value, its grade of membership and thus its likelihood of being in group A decreases.

2. Define the final group of high-quality solutions related to the optimization of the i-th objective function as:

$$A_\alpha^i = \{Z_k^i | \mu_A(Z_k^i) \geq \alpha\} \qquad (2.5)$$

Here, $0 < \alpha < 1$ is the cut-off parameter. The α-cut of the fuzzy set ensures that 100α% of the fit solutions in a population are included in the group. As α approaches 0 (or 1), all members (or only the best member) of the population will be included.

3. Repeat the formation of group A_α^i for all objectives i = 1, 2,….., N of the MaOO problem.

4. A union set Ω is formed by including the top 100α% best-fi solutions A_α^i for all objectives i = 1, 2, …., N . Symbolically,

$$\Omega = \bigcup_{\{l=1\}}^{\{N\}} A_{\{\alpha\}}^l \qquad (2.6)$$

The pseudo-code to form Ω is given below.

Procedure unify i-ABCs

Begin

I. Evaluate $\mu_A(Z_k^i)$ using (2.4) for k = 1, 2, …, S and l = 1, 2, …, N.

II. Evaluate A_α^i using (2.5) for l = 1, 2, …, N.

III. Set $\Omega \leftarrow \bigcup_{\{l=1\}}^{\{N\}} A_{\{\alpha\}}^l$.

IV. Return Ω.

End

2.4.3. Solution of MaOO Problem

The following strategy is used to rank members of the union set Ω in order to efficiently identify the solution to the MaOO problem:

1. **Evaluation of Objectives:** First, evaluate all N objectives for all members of Ω. Let f^{i-max} and f^{i-min} be the maximum and minimum values of the i-th objective function in Ω for i = 1, 2, ..., N. The value of the i-th objective function for a member Z in Ω is denoted as $f^i(Z)$.

2. **Membership Grade Calculation:** Next, calculate the membership grade of each member Z in Ω for each objective i= 1, 2, ..., N using the formula:

$$\mu_\Omega^i(Z) = \left[1 - \frac{f^i(Z) - f^{i-min}}{f^{i-max} - f^{i-min}}\right], \forall Z \in \Omega \qquad (2.7)$$

3. **Assigning Ranks Based on Membership Grade:** Assign ranks to members based on their membership grades for each objective \(l \):

- If $\left(\alpha \leq \mu_{\Omega}^{i}(Z) \leq 1 \right)$, assign a rank $r^{i}(Z)$= 1, indicating $Z \in \Omega$ satisfies the i-th objective by at least 100α% and at most 100%.

- If $(2\alpha - 1) \leq \mu_{\Omega}^{i}(Z) \leq \alpha$, assign a rank $r^{i}(Z)$= 2.

- Similarly, for membership grades $(3\alpha - 2) \leq \mu_{\Omega}^{i}(Z) \leq (2\alpha - 1)$, assign a rank of 3. Repeat this step for all members of Ω to determine their ranks for the i-th objective.

4. **Summing Ranks Across Objectives:** Repeat the ranking process for all objectives i = 1, 2, ..., N for each member $Z \in \Omega$. Calculate the Sum of Ranks (SoR) for each member:

$$SOR(Z) = \sum_{i=1}^{N} r^{i}(Z) \qquad (2.8)$$

Note that a higher SoR indicates a lower quality solution.

5. **Iterative Evaluation:** Repeat steps 2-4 for all members of Ω.

6. **Identifying Optimal Solutions:** Members of Ω with a rank of 1 for all N objectives (SoR = N) are considered approximate global optima for the MaOO problem. These members form the approximate Pareto set Ω^{app} and are declared as the solution to the MaOO problem.

7. **Handling Non-Optimal Cases:** If no such solutions are found (indicating no member satisfies all N objectives by at least 100α%), identify members with rank 1 for N-1 objectives and rank 2 for the remaining objective. The SoR for these members will be $(N - 1) \times 1 + 2 = N + 1$. These members are added to Ω^{app}.

8. **Continuing the Process:** If the previous steps fail to yield a non-empty Ω^{app}, search for candidates with rank 1 for N-2 objectives and rank 2 for the remaining two objectives. The SoR for these solutions will be $(N - 2) \times 1 + 2 \times 2 = N + 2$.

9. **Finalizing the Pareto Set:** Continue this process until a non-empty Ω^{app} is obtained. The members of Ω^{app}.are declared as the approximate global optima for the MaOO problem with N objectives.

The pseudo-code to identify Ω^{app} is given below.

Procedure FBC

Input: MaOO problem with N objective functions.
Output: Approxmate global optima Ωapp of the given problem

Begin

1. Run N, i-ABC(S, D, $f_k(.)$) in parallel to obtain final population P^i for i = 1, 2, …, N.
2. Call unify i-ABCs(P^i for i= 1, 2, …, N) to obtain the union set Ω.
3. Evaluate $f_i(.)$ for i = 1, 2, …, N for all members of Ω.
4. Evaluate $\mu_{\Omega}^{i}(Z)$ using (2.7) and thus determine $r_i(Z)$ for i = 1, 2, …, N using principle given in Section 2.4.3.
5. Evaluate SoR (Z) using (2.8) for all Z ∈ Ω.
6. Set Ωapp← NULL and k←1.
7. **While** Ωapp is empty
   (a) Identify the solutions of Ω, with SoR=N+k-1 and include them in Ωapp .
   (b) Set k←k+1.
   End While.
8. Return Ωapp .

End

2.5. Results

2.5.1. Comparative Framework and Parameter Setting

The proposed algorithm, FBC, is compared with three leading multi-objective optimization (MaOO) techniques, the hypervolume estimation algorithm (HypE) [15], the multi-objective evolutionary algorithm based on decomposition (MOEA/D) [16], and the grid-based evolutionary algorithm (GrEA) [17]. The parameter settings for these competing algorithms were sourced from their original descriptions. For FBC, extensive experiments with α values ranging from 0 to 1 (in increments of 0.001) determined that the optimal balance between performance quality and

diversity occurs at $\alpha = 0.95$. This means that 95% of the best-fit members of each population (for a specific objective) are used for forming the groups.

2.5.2. Benchmark Functions and Termination Condition

The performance of the algorithms is evaluated using the DTLZ [18] and WFG [19] benchmark suites. The DTLZ includes nine functions, while the WFG includes seven. For each function, the number of objectives, N, is set to 6 or 8, or 10. The maximum number of generations, G^{max}, is set to 700 for DTLZ1 and WFG2, 1000 for DTLZ3 and WFG1, and 250 for the remaining functions.

2.5.3. Performance Metrics

The effectiveness of the algorithms is measured using hypervolume (HV) [20] and inverted generational distance (IGD) [21] metrics. Higher HV and lower IGD values indicate better performance of the MaOO algorithms.

2.5.4. Performance Analysis

The performance metrics (HV and IGD) for the algorithms (FBC, HypE, MOEA/D, and GrEA) are analyzed based on the median and interquartile range (IQR) obtained from 50 independent runs. The median and IQR of the HV and IGD metrics for these runs are reported in Tables I and II, respectively.

According to the results in Table I, FBC generally outperforms its competitors regarding the HV metric. However, GrEA surpasses FBC for WFG4 and WFG6 when N = 6, and WFG8 and WFG9 when N = 10. Similar trends are observed for the IGD metric in Table II. Except for three cases, FBC is the top-performing algorithm. HypE outperforms FBC for DTLZ5 with N = 6, and MOEA/D outperforms FBC for DTLZ6 and DTLZ7, both with N = 6.

Function	*N*	HypE	MOEA/D	GrEA	FBC
WFG1	6	9.327e-01 (9.113e-03)	9.786e-01 (2.711e-02)	9.413e-01 (3.911e-02)	**9.893e-01 (2.996e-02)**
	8	9.271e-01 (4.598e-02)	9.154e-01 (4.858e-02)	9.150e-01 (5.812e-02)	**9.891e-01 (3.843e-02)**
	10	9.217e-01 (1.257e-01)	9.597e-01 (1.677e-02)	8.592e-01 (3.778e-02)	**9.858e-01 (2.792e-02)**
WFG2	6	6.787e-01 (9.780e-02)	8.594e-01 (8.626e-02)	9.447e-01 (7.328e-02)	**9.991e-01 (2.777e-02)**
	8	9.930e-01 (2.620e-03)	9.720e-01 (2.187e-02)	9.423e-01 (9.993e-02)	**9.991e-01 (2.925e-03)**
	10	9.977e-01 (2.992e-03)	9.833e-01 (3.981e-02)	9.671e-01 (1.790e-02)	**9.991e-01 (1.993e-03)**
WFG3	6	5.430e-01 (7.146e-03)	4.836e-01 (6.545e-03)	5.891e-01 (8.391e-03)	**6.265e-01 (7.982e-03)**
	8	5.942e-01 (1.748e-02)	5.947e-01 (5.284e-03)	5.740e-01 (1.325e-02)	**6.438e-01 (2.698e-02)**
	10	5.547e-01 (4.840e-03)	6.027e-01 (2.160e-03)	4.975e-01 (1.625e-02)	**6.371e-01 (2.955e-03)**
WFG4	6	7.594e-01 (2.397e-02)	7.615e-01 (1.446e-02)	**8.125e-01 (1.171e-02)**	7.972e-01 (1.686e-02)
	8	7.395e-01 (3.130e-02)	6.634e-01 (1.975e-02)	7.978e-01 (8.102e-03)	**8.952e-01 (6.936e-03)**
	10	5.997e-01 (5.557e-02)	4.796e-01 (1.728e-02)	8.123e-01 (8.971e-03)	**8.426e-01 (2.538e-02)**
WFG5	6	6.597e-01 (1.767e-02)	5.414e-01 (1.286e-02)	7.134e-01 (5.933e-03)	**7.474e-01 (6.261e-03)**
	8	5.756e-01 (1.416e-02)	4.224e-01 (1.284e-02)	6.671e-01 (5.913e-03)	7.113e-01 (5.771e-03)
	10	3.860e-01 (1.634e-02)	3.432e-01 (8.560e-03)	5.942e-01 (5.883e-03)	**6.782e-01 (5.366e-03)**
WFG6	6	4.684e-01 (7.546e-03)	6.656e-01 (3.900e-02)	**6.944e-01 (2.480e-02)**	6.626e-01 (2.712e-02)
	8	2.720e-01 (1.932e-02)	4.920e-01 (2.381e-02)	6.256e-01 (1.690e-02)	**7.714e-01 (1.991e-02)**
	10	3.968e-01 (1.167e-02)	4.692e-01 (3.418e-02)	7.803e-01 (1.930e-02)	**8.475e-01 (2.697e-02)**
WFG7	6	3.755e-01 (1.756e-02)	5.923e-01 (3.211e-02)	6.621e-01 (2.761e-02)	**7.367e-01 (6.724e-03)**
	8	6.341e-01 (4.510e-02)	5.595e-01 (3.574e-02)	7.466e-01 (8.654e-03)	**8.416e-01 (7.241e-03)**
	10	6.400e-01 (3.648e-02)	3.168e-01 (8.193e-03)	8.724e-01 (7.547e-03)	**9.136e-01 (6.961e-03)**
WFG8	6	3.876e-01 (1.618e-02)	2.164e-01 (1.977e-02)	4.581e-01 (3.151e-02)	**4.946e-01 (2.931e-02)**
	8	2.845e-01 (1.807e-02)	3.228e-01 (2.348e-02)	4.421e-01 (1.750e-02)	**6.155e-01 (1.671e-02)**
	10	3.101e-01 (2.137e-02)	5.398e-01 (2.720e-02)	**7.936e-01 (3.701e-02)**	6.922e-01 (1.712e-02)
WFG9	6	3.535e-01 (3.611e-02)	3.934e-01 (4.175e-02)	7.196e-01 (4.451e-02)	**7.962e-01 (4.752e-02)**
	8	4.225e-01 (6.515e-02)	4.980e-01 (2.791e-02)	5.750e-01 (1.604e-02)	**6.914e-01 (1.891e-02)**
	10	4.447e-01 (3.806e-02)	5.663e-01 (3.725e-02)	**7.740e-01 (6.215e-03)**	7.661e-01 (6.223e-03)

TABLE-2-I HV BASED PERFORMANCE EVALUATION FOR WFG

Function	*N*	HypE	MOEA/D	GrEA	FBC
DTLZ1	6	2.391e-01 (4.456e-02)	1.184e-01 (5.955e-02)	1.472e-01 (6.844e-03)	**9.231e-02 (3.465e-03)**
	8	3.247e-01 (2.633e-02)	1.965e-01 (1.476e-02)	2.611e-01 (3.764e-02)	**1.241e-01 (1.955e-02)**
	10	2.477e-01 (3.766e-02)	2.317e-01 (3.432e-02)	1.494e-01 (1.885e-02)	**1.325e-01 (2.484e-02)**
DTLZ2	6	4.796e-01 (5.460e-02)	4.165e-01 (4.801e-02)	2.673e-01 (1.993e-03)	**2.111e-01 (1.823e-03)**
	8	7.116e-01 (4.593e-02)	6.743e-01 (5.800e-02)	3.768e-01 (3.482e-03)	**3.256e-01 (3.912e-03)**
	10	7.428e-01 (4.546e-02)	7.596e-01 (6.952e-02)	3.950e-01 (2.700e-02)	**1.261e-01 (2.647e-02)**
	N	HypE	MOEA/D	GrEA	FBC
	6	4.586e-01 (7.753e-02)	4.936e-01 (1.910e-01)	2.266e-01 (6.331e-02)	2.691e-01 (7.724e-02)
	8	6.443e-01 (1.326e-01)	7.853e-01 (1.386e-01)	3.877e-01 (1.981e-01)	3.772e-01 (4.824e-02)
	10	7.468e-01 (5.950e-02)	6.973e-01 (2.816e-01)	4.958e-01 (1.616e-01)	3.486e-01 (7.681e-02)
	6	6.482e-01 (1.308e-01)	5.367e-01 (1.778e-02)	2.757e-01 (2.310e-03)	2.484e-01 (1.621e-03)
	8	7.132e-01 (7.602e-02)	7.695e-01 (6.615e-02)	3.882e-01 (2.981e-03)	3.245e-01 (2.636e-03)
	10	8.357e-01 (1.868e-02)	7.910e-01 (2.122e-02)	4.140e-01 (9.472e-03)	3.315e-01 (1.737e-02)
	6	1.630e-02 (9.816e-03)	2.311e-02 (1.638e-02)	8.920e-02 (1.685e-02)	5.698e-02 (1.845e-02)
	8	5.120e-02 (1.121e-02)	8.541e-02 (1.921e-02)	1.802e-01 (4.857e-02)	3.141e-02 (3.848e-02)
	10	2.933e-02 (2.916e-03)	7.262e-02 (4.267e-02)	4.254e-01 (7.976e-02)	2.713e-01 (5.681e-02)
	6	3.656e-01 (5.787e-02)	1.686e-01 (3.253e-01)	2.620e-01 (6.277e-02)	1.751e-01 (1.838e-01)
	8	4.621e-01 (3.502e-01)	1.757e-01 (2.917e-02)	5.690e-01 (1.834e-01)	1.621e-01 (4.568e-02)
	10	2.420e-01 (4.820e-02)	4.723e-01 (5.633e-02)	4.215e-01 (2.816e-01)	2.272e-01 (4.727e-02)
	6	1.654e+00 (1.375e-01)	4.123e-01 (1.751e-02)	5.701e-01 (2.601e-02)	5.242e-01 (4.733e-03)
	8	2.931e+00 (2.321e-01)	1.441e+00 (2.522e-01)	1.101e+00 (2.433e-02)	9.151e-01 (1.567e-02)
	10	2.960e+00 (1.782e-01)	1.642e+00 (4.653e-01)	1.421e+00 (9.538e-02)	1.271e+00 (8.628e-02)

TABLE-2-II IGD BASED PERFORMANCE EVALUATION FOR DTLZ1 AND DTLZ2

Reference:

[1] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, “Evolutionary manyobjective optimization: A short review,” in IEEE Congress on Evolutionary Computation, 2008, pp. 2419-2426.

[2] M. Pal and S. Bandyopadhyay, “Many-objective feature selection for motor imagery EEG signals using differential evolution and support vector machine,” in IEEE International Conference on Microelectronics, Computing and Communication, MicroCom, 2016, pp. 1-6.

[3] A. Chowdhury, P. Rakshit, and A. Konar, “Protein-protein interaction network prediction using stochastic learning automata induced differential evolution,” Applied Soft Computing, vol. 49, 2016, pp. 699-724.

[4] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, “Many-objective optimization: an engineering design perspective,” in International Conference, EMO, 2005, pp. 14-32.

[5] Y. Fang, Q. Liu, M. Li, Y. Laili, and D. T. Pham, “Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations,” European Journal of Operational Research, vol. 276, no. 1, 2019, pp. 160-174.

[6] B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, and K. Kang, “Distributed parallel particle swarm optimization for multiobjective and many-objective large-scale optimization,” IEEE Access, vol. 5, 2017, pp. 8214-8221.

[7] W. Hu, G. G. Yen, and G. Luo, “Many-objective particle swarm optimization using two-stage strategy and parallel cell coordinate system,” IEEE transactions on cybernetics, vol. 47, no. 6, 2016, pp. 1446-1459.

[8] D. Brockhoff and E. Zitzler, “Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization,” in Parallel Problem Solving from Nature, 2006, pp. 533-542.

[9] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining convergence and

diversity in evolutionary multiobjective optimization," Evolutionary Computation, vol. 10, no. 3, 2002, pp. 263–282.

[10] A. L. Jaimes, C. A. Coello Coello, H. Aguirre, and K. Tanaka, "Adaptive objective space partitioning using conflict information for many objective optimization," in Evolutionary Multi-Criterion Optimization, 2011, pp. 151–165.

[11] M. K¨oppen and K. Yoshida, "Substitute distance assignments in NSGA-II for handling many-objective optimization problems," in Evolutionary Multi-Criterion Optimization, 2007, pp. 727–741.

[12] B. Basturk, D. Karaboga, "An artificial bee colony (ABC) algorithm for numeric function optimization," in Proceedings of the IEEE Swarm Intelligence Symposium, 2006.

[13] P. Rakshit, and A. Konar, "Realization of learning induced selfadaptive sampling in noisy optimization," Applied Soft Computing, vol. 69, 2018, pp. 288-315.

[14] A. Konar, Computational intelligence: principles, techniques and applications, Springer Science & Business Media, 2006.

[15] J. Bader and E. Zitzler, "HypE: an algorithm for fast hypervolume based many-objective optimization," Evolutionary Computation, vol. 19, no. 1, 2011, pp. 45–76.

[16] Q. Zhang and H. Li, "MOEA/D: a multi-objective evolutionary algorithm based on decomposition," IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, 2007, pp. 712–731.

[17] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," IEEE Transactions on Evolutionary Computation, vol. 17, no. 5, 2013, pp. 721–736.

[18] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, A. Abraham, L. Jain, and R. Goldberg, Eds., Springer, 2005, pp. 105–145.

[19] S. Huband, P. Hingston, L. Barone, and L. While, “A review of multiobjective test problems and a scalable test problem toolkit,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 5, 2006, pp. 477–506.

[20] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, New York: Wiley, 2001.

[21] M. Li and J. Zheng, “Spread assessment for evolutionary multiobjective optimization,” in 5th International Conference on Evolutionary Multi-Criterion Optimization, 2009, pp. 216–230.

[22] X. Li, “Niching without niching parameters: particle swarm optimization using a ring topology,” IEEE Transactions on Evolutionary Computation, vol. 14, no. 1, 2009, pp. 150-169.

[23] J. Derrac, S. García, D. Molina, and F. 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, vol. 1, no. 1, 2011, pp. 3-18

Chapter 3: Conclusion & Future Research

3.1. Introduction

The paper presents an innovative method to tackle many-objective optimization (MaOO) by creating a coevolution policy based on fuzzy membership. This policy leverages the artificial bee colony (ABC) optimization technique to simultaneously target different objectives within MaOO problems. The approach recognizes that a single solution may perform well for one objective but not for others, thus it develops a system to optimize each objective separately. The traditional ABC mutation policy is modified to allow communication between populations focusing on various objectives. A new fuzzy membership measure is used to selectively identify top-quality solutions within each ABC population based on their closeness to the best-performing solutions. These high-quality solutions are grouped for each objective and then combined into a union set. A fuzzy membership-based ranking system is used to identify the top solutions, approximating the Pareto front of the MaOO algorithm.

Experimental tests using the WFG and DTLZ test suites show that the proposed fuzzy membership-induced coevolution (FBC) algorithm outperforms existing methods.

3.2. Applications

The primary application of this research lies in its ability to handle complex optimization problems involving many objectives, which are often encountered in real-world scenarios such as engineering design, financial portfolio optimization, and logistics

planning. By processing each objective in parallel, the FBC algorithm maximizes the use of computational resources and speeds up the convergence towards optimal solutions.

Here's how the process works in detail:

1. **Parallel Handling of Objectives:** The FBC algorithm employs the ABC algorithm to handle each objective of the MaOO problem separately but in parallel. This means that instead of trying to optimize all objectives simultaneously in a single population, each objective has its own dedicated population of solutions evolving independently.

2. **Selection of High-Quality Solutions:** After each population has evolved and converged towards high-quality solutions, a subset of the best solutions is selected from each population. These solutions are chosen based on their performance concerning the specific objective they were optimizing.

3. **Formation of Union Set:** The selected high-quality solutions from all populations are then combined into a single union set. This union set represents a diverse pool of solutions that have been optimized for different objectives.

4. **Ranking and Identification of Optimal Solutions:** A fuzzy membership-induced ranking measure is applied to this union set to identify the top-ranked solutions. This ranking system evaluates how well each solution balances the various objectives, aiming to approximate the Pareto optimal front, which represents the set of solutions that offer the best trade-offs among the objectives.

5. **Comparison and Validation:** The FBC algorithm is benchmarked against three state-of-the-art techniques to validate its performance. Experimental results indicate that the FBC algorithm consistently outperforms these other methods according to various performance metrics.

3.2.1. Applications and Benefits:

Engineering Design: In engineering, optimizing designs often involves balancing multiple criteria such as cost, performance, durability, and environmental impact. The FBC algorithm can help engineers find designs that offer the best trade-offs among these

competing objectives.

Financial Portfolio Optimization: Investors seek to maximize returns while minimizing risk. The FBC algorithm can assist in finding portfolios that optimally balance these objectives, providing more efficient frontiers of investment options.

Logistics and Supply Chain Management: Logistics problems often require optimizing factors like cost, delivery time, and reliability. The FBC algorithm can help develop strategies that best balance these factors, leading to more efficient and effective logistics solutions.

Healthcare: In healthcare, many optimization problems involve balancing treatment effectiveness, cost, and patient quality of life. The FBC algorithm can aid in developing treatment plans that offer the best possible outcomes across these dimensions.

Overall, the fuzzy-bee colony algorithm's ability to efficiently handle and optimize many-objective problems makes it a valuable tool across various fields where complex decision-making and trade-offs are essential. The approach not only enhances computational efficiency but also improves the quality of solutions, offering significant advantages over existing methods.

3.3. Future research

Future research directions encompass several intriguing possibilities. One key area of focus is the automation of parameter tuning. This involves developing advanced algorithms or techniques that can automatically adjust the parameters of the optimization process, thereby reducing the need for manual intervention and enhancing the efficiency and effectiveness of the optimization.

Another promising avenue is the optimization of only the most significant objectives. In Many-objective optimization problems, not all objectives carry the same weight or importance. By identifying and concentrating on the most crucial objectives, the optimization process can become more targeted and efficient, leading to better overall performance.

Additionally, there is a suggestion to devise strategies that can guide the search towards the most promising regions of non-convex Pareto fronts. Non-convex Pareto fronts are complex and challenging to navigate due to their irregular shapes and multiple peaks and valleys. Developing methods to effectively explore these areas can significantly improve the quality of the solutions found.

Incorporating expert decision-making into this process could further enhance its efficacy. Experts can provide valuable insights and heuristics that can help steer the optimization process more effectively. By integrating these expert inputs, the optimization algorithm can potentially make more informed decisions, leading to superior outcomes.

Overall, these research directions aim to refine and advance the capabilities of multi-objective optimization algorithms, making them more robust, efficient, and capable of tackling complex real-world problems.

3.4. Conclusion

This chapter briefly highlights the contributions and findings of the thesis. The idea of FBC Algorithm and the steps involved to find the solution of a Many objective optimization problem is briefly provided here. This chapter includes the applications of this research and also introduces some of the future research scopes.