Fitness Dependent Optimizer: A Novel Swarm Intelligence Algorithm for Single and Multi-Objective Optimization Problems

A Thesis Submitted to the Council of the College of Science at the University of Sulaimani in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science (Artificial Intelligence)

By

Jaza Mahmood Abdullah

MSc (2012) Software Systems and Internet Technology, University of Sheffield, UK.

Supervised by

Dr. Tarik Ahmed Rashid

Professor

May-2020 Gelawêj-2720

Supervisor Certification

I certify that the preparation of thesis titled " Fitness Dependent Optimizer: A Novel Swarm Intelligence Algorithm for Single and Multi-Objective Optimization Problems" accomplished by (Jaza Mahmood Abdullah), was prepared under my supervision in the college of Science, at the University of Sulaimani, as partial fulfillment of the requirements for the degree of doctor of philosophy in computer science (artificial intelligence).

Signature:

Name: Tarik Ahmed Rashid

Title: Professor

Date: 10/02/ 2020

In view of the available recommendation, I forward this thesis for debate by the examining committee.

Signature:

Name: Mustafa Ibrahim Khaleel

Title: Lecturer

Date: 24/02/ 2020

Examining Committee Certification

We certify that we have read this thesis entitled " Fitness Dependent Optimizer: A Novel Swarm Intelligence Algorithm for Single and Multi-Objective Optimization Problems" prepared by (Jaza Mahmood Abdullah), and as Examining Committee, examined the student in its content and in what is connected with it, and in our opinion, it meets the basic requirements toward the degree of doctor of philosophy in Computer Science (Artificial Intelligence).

Signature:
Name: **Dr. Aree Ali Mohammed**
Title: **Professor**
Date: 24/05/2020
(Chairman)

Signature:
Name: **Dr. Soran Abubakr Muhamad**
Title: **Professor**
Date: 24/05/2020
(Member)

Signature:
Name: **Dr. Omar Younis Abdulhammed**
Title: **Assistant Professor**
Date: 24/05/2020
(Member)

Signature:
Name: **Dr. Laith Razuqi Falih Hassan**
Title: **Assistant Professor**
Date: 24/05/2020
(Member)

Signature:
Name: **Dr. Nawzad Kamaran Salayi**
Title: **Assistant Professor**
Date: 24/05/2020
(Member)

Signature:
Name: **Dr. Tarik Ahmed Rashid**
Title: **Professor**
Date: 24/05/2020
(Supervisor- Member)

Approved by the Dean of the College of Science.

Signature:
Name: **Dr. Soran Mohammed Mhamand**
Title: **Assistant Professor**
Date: 24/05/2020

Acknowledgments

First and most thanks and gratefulness for the ALLAH who made things easy for me, then I would like to thank Prof. Dr. Tarik A. Rashid for the extraordinary guiding support he showed, also the University of Sulaimani for their financial supports.

Abstract

Many swarm optimization algorithms have been introduced since the early 60’s, evolutionary programming (AP) to the most recent, whale optimization algorithm (WOA). All of these algorithms have proved their potential to solve many optimization problems. The aim of this work, is to enhance or introduce nature-based optimization algorithms. Thus, in this thesis, two novel swarm intelligent algorithms are proposed, single-objective fitness dependent optimizer (SOFDO) and multi-objective fitness dependent optimizer (MOFDO). The bee swarming reproductive process and their collective decision-making have inspired these two algorithms; they have no algorithmic connection with the honey bee algorithm or the artificial bee colony algorithm. It is worth mentioning that both SOFDO and MOFDO are considered a particle swarm optimization (PSO)-based algorithm, that they update the search agents' position by adding velocity (pace). However, they calculate velocity differently; they use the problem fitness function value to produce certain weights, and these weights guide the search agents during both the explorations and exploitations phases. Moreover, MOFDO is well-thought as a cultural algorithm, therefore, all five types of knowledge (situational, normative, topographical, domain, and historical knowledge) are considered during the implementation. Throughout this work, both SOFDO and MOFDO algorithms are presented, and the motivation behind the idea is explained.

For the performance-proof purpose, the first algorithm SOFDO is tested on a group of 19 classical benchmark test functions, and the results are compared with three well-known algorithms: PSO, the genetic algorithm (GA), and the dragonfly algorithm (DA), additionally, SOFDO is tested on IEEE Congress of Evolutionary Computation benchmark test functions (CEC-C06, 2019

Competition) [1]. The results are compared with three modern algorithms: (DA), the WOA, and the salp swarm algorithm (SSA). The SOFDO results show better performance in most cases and comparative results in other cases. Furthermore, the results are statistically tested with the Wilcoxon rank-sum test to show the significance of the results. Likewise, SOFDO stability in both the exploration and exploitation phases is verified and performance-proofed using different standard measurements. Finally, SOFDO is applied to real-world applications as evidence of its feasibility, for instances, aperiodic antenna array designs, and frequency-modulated sound waves.

The Second algorithm MOFDO is tested on two standard benchmarks: classical ZDT test functions and CEC 2019 multimodal multi-objective test functions. MOFDO results are compared to the latest variant of multi-objective particle swarm optimization (MOPSO), non-dominated sorting genetic algorithm third improvement (NSGA-III), and multi-objective dragonfly algorithm (MODA). The overall comparison shows the superiority of MOFDO in the majority of the cases and comparative results in other cases. Moreover, MOFDO is used for optimizing real-world engineering problems (e.g. welded beam design problem); the proposed algorithm successfully tackles the problem and provides a wide variety of well-distributed feasible solutions, which enable the decision-makers to have more applicable-comfort choices to consider.

Table of content

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Descriptions
ABC	Artificial Bee Colony
AMPS	Adaptive Method For The Population Size
ARABC	Ranking-Based Adaptive Artificial Bee Colony
CEC	Congress Of Evolutionary Computation
CF	Cost Function – Fitness Value
CLONAL	Clonal Selection Theory Of Acquired Immunity
CS	Cuckoo Search
DA	Dragonfly Algorithm
DE	Differential Evolution
DFnABC	Distance-Fitness-Based Neighbor Artificial Bee Colony
DPF	Dual-Population Framework
EOA	Evolutionary Optimization Algorithm
FA	Firefly Algorithm
FM	Frequency-Modulated
GA	Genetic Algorithm
HS	Harmony Search
IGD	Inverse Generational Distance
MMF	Multimodal Functions
MOEAs	Multi-Objective Evolutionary Algorithms
MOFDO	Multi-Objective Fitness Dependent Optimizer
MOOPs	Many-Objective Optimization Problems
MOPs	Multi-Objective Optimization Problems
MOPSO	Multi-Objectives Particle Swarm Optimization
NSGA-II	Non-Dominated Sorting Genetic Algorithm Version 2
NSGA-III	Non-Dominated Sorting Genetic Algorithm Version 3
SA	Simulated Annealing
SBX	Simulated Binary Crossover
SLL	Side Lobe Level

SOFDO	Single-Objective Fitness Dependent Optimizer
SOPs	Single-Objective Optimization Problems
SPEA2	Strength Pareto Evolutionary Algorithm 2
SSA	Salp Swarm Algorithm
VLE	Vapor-Liquid Equilibrium
WOA	Whale Optimization Algorithm
ZDT	Zitzler Deb Thiele

Chapter One

Introduction

Chapter One

1. Introduction

From the time when computers were invented, searching for the unknown and looking for the best solution were points of focus. As early as 1945, Alan Turing used a type of search algorithm for breaking German Enigma ciphers during World War II [1]. To date, hundreds of types of algorithms have been developed for various purposes, including optimization problems. Optimization algorithms are used to find suitable solutions for a problem. There might be many different solutions for a single problem, but the optimum solution is preferable. Usually, optimization problems are nonlinear with a complex landscape. Generally, optimization algorithms can be classified into traditional and evolutionary algorithms. Traditional algorithms include gradient-based algorithms and quadratic programming. Evolutionary algorithms include heuristic or metaheuristic algorithms and many hybrid techniques.

1.1. Traditional Algorithms

Traditional algorithms are efficient in their work; however, several facts can be discussed about them. They are mostly deterministic; for example, a given input will always obtain the same output (except the hill-climbing algorithm when using random restart). Moreover, they perform local searches, which is why there is no guarantee that global optimality will be reached for most of the optimization problems. Consequently, they have limited diversity in the obtained solutions. Additionally, they use some information about the problems, and therefore, they tend to be problem-specific. Furthermore, these traditional algorithms cannot effectively solve multimodal problems because they do not work on highly nonlinear problems.

1.2. Evolutionary Algorithms

Evolutionary algorithms could be the correct answer to previous limitations as they have stochastic behaviors. They come in two forms: heuristic and meta-heuristic algorithms. Heuristic algorithms search for a solution by trial and error; they hope that a quality solution will be found in a reasonable amount of time. Similarly, they tend to use specific randomization mechanisms and local searches in various ways. More studies and developments have been conducted on heuristic algorithms to make what is known as metaheuristic algorithms. Metaheuristic algorithms have better performance than heuristics algorithms, which is why the “meta” prefix was added, which means “higher” or “beyond”. However, researchers currently use these two terms (heuristic and metaheuristic) interchangeably, as there is little difference in their definitions [2] [3] [4].

1.3. Problem Statement

The complexity of real-world problems that exist around us makes it impossible to search every possible solution simply because of time, space, and cost considerations. As a result, low cost, fast, and more intelligent mechanisms are required. Therefore, researchers have studied the behaviors of animals and natural phenomena to understand how they solve their problems. For example, how ants find their path, how a group of fish, birds or flies avoid the enemy or hunt their prey, and how gravity works. Thus, these algorithms, which are inspired by nature, are known as nature-inspired algorithms.

Regarding problem complexity, there are three major types of’ optimization problems, single-objective optimization problems (SOPs), multi-objective optimization problems (MOPs) and many-objective optimization problems (MOOPs).

1.3.1. Single-Objective Optimization Problems

SOPs are usually simple and has only one objective to be discovered by the evolutionary algorithm. The goal of SOPs is to discover the best solution for a specific criterion or metric, such as execution time or performance and/or a combination of this metric with energy consumption or power dissipation metrics. We can further combine multiple criteria into a single-objective optimization problem by defining the single-objective cost function as a weighted sum of the normalized costs associated with each of the metrics

1.3.2. Multi-Objectives and Many-Objective Optimization Problems

MOPs are more complex because they have more than one objective, usually two or three. Additionally, MOOPs is an optimization problem with four or more objective, which is considered to be even more complex than MOPs.

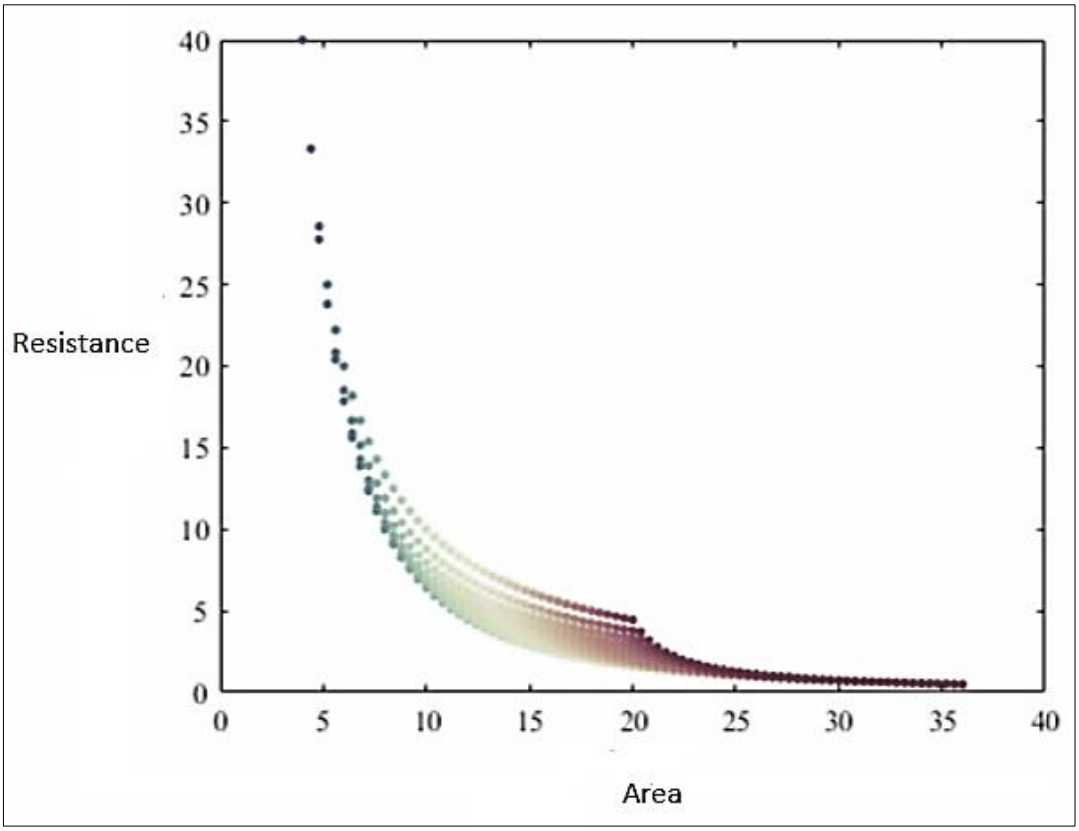

Figure 1.1 Wooden boat example [75]

Having said that, MOPs are problems with more than one objective, and usually, these objectives are in conflict, meaning that increasing the value of

one objective results in decreasing another objective value. However, MOPs with none conflicting objectives can be easily solved by using single-objective optimization, which means running single-objective techniques for each objective separately, then aggregate the results. An example of MOPs with none conflicting objectives is a wooden boat optimization problem as shown in Figure (1.1), in this example, there are both boat area and floating force objectives (buoyancy), by maximizing the area of the boat the floating force also increases and vice versa. However, there is another objective called resistance (resistance of wooden plats), the resistance of wood plates decreases by increasing the boat area, so that, the area and resistance are two objectives in conflict, maximizing one of them results in minimizing the other. The answer to such kind of problem is Pareto optimal solutions set.

1.4. Work Motivations

Various research has been conducted in the field of nature-inspired metaheuristic algorithms; additionally, many efficient algorithms have been proposed in the literature. Alternatively, there is always room for new algorithms, as long as the proposed algorithm provides better or comparative performances, as explained by David H. Wolpert and William G. Macready in their work titled “No Free Lunch Theorems for Optimization” in 1997. Thus, there is no single global algorithm that can provide the optimum solution for every optimization problem. For example, if algorithm “A” works better than algorithm “B” on optimization problem X, then there is a high chance that there is an optimization problem Y that works better on algorithm “B” than on algorithm “A” [5]. For these reasons, two new algorithms called SOFDO and MOFDO are proposed in this work. This algorithm is inspired by the swarming behavior of bees during the reproductive process when they search for new hives. The proposed algorithms have nothing in common with the ABC

algorithm (except both algorithms are inspired by bee behavior, and both are nature-inspired meta-heuristic algorithms).

1.5. Work Contributions

The major contributions of this work are summarized as follows:

1. Two new novel swarm intelligent algorithms are proposed for both SOPs and MOPs in this work: SOFDO is used for solving SOPs and real-world applications for instances, aperiodic antenna array designs, and frequency-modulated sound waves, and MOFDO is used for tackling MOPs, also used for multi-objective real-world application such as welded beam design engineering problem.
2. The proposed algorithms are using certain characteristics of the honey bee swarms. For example, it uses a fitness function for generating suitable weights that help the algorithm in both exploration and exploitation phases, as it provides fast convergence towards global optimality with respect to fair coverage of the search space.
3. One unique feature of SOFDO and MOFDO algorithm is that they store the past search agent paces (velocity) for potential reuse in future steps, to the best of our knowledge this feature is unique and not presented before in the literature.
4. Both proposed SOFDO and MOFDO can be considered a PSO-based algorithm since the use a similar mechanism for updating agents' positions; however, our proposed algorithms do it in a very different way, and it is statistically proven in this work that our proposed algorithms outperform PSO, DA, GA, WOA, SSA, MOPSO, MODA and (NSGA-III) in many benchmark test functions and has comparative results on others.

1.6. Thesis Map

The rest of the thesis is organized as follows:

a) The related previous works are comprehensively discussed as a review of background literature in chapter two. Here, the first versions of the search algorithm are discussed through the history of modern computing, then afterward, the most state-of-art search algorithms are presented.

b) Then in chapter three, the proposed algorithms (SOFDO and MOFDO) are described, the details of the algorithms are presented as a methodology of this work. The opening of the section explains the inspiration of SOFDO and MOFDO algorithms from the honey bee. Afterward, each of SOFDO and MOFDO is mathematically presented, then each of them is programmatically illustrated using detailed pseudocode and flow charts.

c) In chapter four, results and discussions, both algorithms are tested on 19 classical standard benchmarks, 10 CEC-06 2019 modern test function, 5 ZDT multi-objective standard test functions, and 12 multi-modal multi-objective functions, the results of our algorithms are compared to PSO, DA, GA, WOA, SSA, MOPSO, MODA and (NSGA-III) algorithms, the results are discussed directly afterword in the same section, we intentionally avoided creating new discussion section for readability purposes. Finally, both of the proposed algorithms are applied to real-world applications such as aperiodic antenna array designs, frequency-modulated sound waves, and welded beam design engineering problems.

d) Finally, in chapter five, the final notes and further potentials of this work are revealed as a conclusion. Some of the produced data are located in the appendix section in the form of the tables due to their long and complex nature.

Chapter Two

Literature Review

Chapter Two

2. Literature Review

Through this chapter, history and background overview of the related previous works are comprehensively discussed as a review of the literature. Firstly, the history of traditional single-objective algorithms is presented, then, multi-objective algorithms are discussed in detail. Cultural algorithm phenomena are presented as a basic foundation of this work, also Pareto solution set optimality is explained briefly, finally, honey bee lifestyle is reviewed as a source of inspiration of this work.

2.1. Single-Objective Optimization Algorithms

Development in nature-inspired metaheuristic algorithms began in the 1960s at the University of Michigan. John Holland and his colleagues published their genetic algorithm (GA) book in 1960 and republished it in 1970 and 1983 [6]. An algorithm that is inspired by the annealing process of metal, known as simulated annealing (SA), was developed by S. Kirkpatrick, C. D. Gellar, and M. P. Vecchi, [7]. Nevertheless, in the past two decades, this field has witnessed many major signs of progress. For instance, particle swarm optimization, which was proposed by James Kennedy and Russel C. Eberhart, has been used for many real-world applications [8]. PSO was inspired by the swarm intelligence of fish and birds while the authors were studying a flock of birds. They found that they could apply these behaviors to optimization problems; later, PSO became a base algorithm for other algorithms, including our algorithm. R. Storn and K. Price developed differential evolution (DE) in 1997. It is a vector-based algorithm that outperforms GA in many applications [9]. After that, in 2001, Zong WooGeem et al. developed the harmony search (HS), which was applied in many optimization problems such as transport modeling and water distribution [10]. Then, in 2004, C. Tovey and S. Nakrani developed the honey bee algorithm. They used it for Internet hosting center optimization [11]. This

was followed by the development of a novel bee algorithm proposed by D. T. Pham et al. [12], and one year later, D. Karaboga et al. created the artificial bee colony (ABC) algorithm in 2005. In 2009, Xin-She Yang developed the firefly algorithm (FA) [13]; and then, the cuckoo search (CS) algorithm was proposed by the same author [14]. Additionally, Xin-She Yang proposed a bat-inspired algorithm in 2010 [15]. Then, in 2015, Mirjalili A. S. proposed the dragonfly algorithm (DA) [16], which is a PSO-based algorithm inspired by the dragonfly swarm behavior of attraction to food and distraction by the enemy, then the whale optimization algorithm (WOA) in 2016 [17], and the salp swarm algorithm (SSA) in 2017 were proposed by the same author [18]. Two new variants of the ABC are proposed by Laizhong et al, the authors showed that they managed to enhance the exploitations of the novel ABC algorithm, as it is well known that the novel ABC has a good exploration ability, however, it suffers from slow exploitations. In their first work, they employed an adaptive method for the population size (AMPS) [19]. In the second paper, they proposed a ranking-based adaptive ABC algorithm (ARABC) [20], the attention on both works was to improve exploitations ability of the novel ABC. Nonetheless, two more improvements were suggested on the novel ABC in 2018, firstly, by proposing the distance-fitness-based neighbor search mechanism (DFnABC), which is a new variant of the ABC [21], and secondly, by proposing the dual-population framework (DPF), again to enhance ABC convergence speed [22]. Additionally, in 2018, a new algorithm which inspired by vapor-liquid equilibrium (VLE) was proposed by Enrique M. Cortés-Toro and his colleagues, the authors claim that their algorithm can solve highly nonlinear optimization problems in continuous domains [23].

2.2. Multi-Objective Optimization Algorithms Overview

MOPs are in an area of multiple criteria decision-making, the area is also known as multi-objective programming, multi-criteria optimization or Pareto

optimization. Precisely, this area deals with mathematical optimization problems. Researches on MOPs became widely popular since 2002 [24]. It is worth mentioning, that various real-world applications are falling under this area, such as engineering design, energy, economics, logistics, and health science. Since many real-world problems are MOPs, thus, multi-objective evolutionary algorithms (MOEAs) are used to solve them. Generally, MOEAs includes two branches: dominance-based and decomposition-based [25].

The first branch of MOEAs includes a non-dominated sorting genetic algorithm (NSGA-II), which was improved by [26], multi-objective particle swarm optimization (MOPSO) [27] , and strength Pareto evolutionary algorithm 2 (SPEA2) [28]. On the other hand, MOEAs based on decomposition comes in the second branch [29].

For instance, the NSGA-II starts with a randomly generated population, as per fitness, it uses a fast-nondominated sorting technique to sort the overall population, and then the children generation is produced by crossover and mutation. To enhance the level of diversity among the solutions, a special crowding distance operator is also applied in later improvement by [26]. Moreover, NSGA-II was improved to solve many-objective problems (having four or more objectives) known as NSGA-III [30].

Another popular algorithm, which has lower computational complexity than NSGA-III is a MOPSO [27], which uses an archive grid-based approach to keep diversity among the solutions. In the last decade, many new MOEAs have been reported in the literature, such as multi-objective cat swarm optimization [31], multi-objective CLONAL algorithm, which is inspired by the clonal selection theory of acquired immunity [32], multi-objective moth flame optimization [33], multi-objective ant lion optimizer [34], multi-objective grey-wolf optimization [35], multi-objective dragonfly algorithm [36], and multi-objective whale algorithm [37]. Having said that, Some MOEAs have been

used for production scheduling [38], optimal truss design [39], and resource allocation in cognitive radio networks [40]. Furthermore, some other MOEAs employed to classify normal and aggressive behavior of 3D human beings [32].

To solve MOPs correctly, two factors need to be considered. Firstly, the determination of personal and global guides (best solutions). Secondly, it is to sustain the diversity of the population as much as possible. Regarding the local and global guide selections, many efforts were made, such as using the adaptive grid for selecting the global guide and introducing the extra repository to store the non-dominated particles in MOPSO [27]. In another attempt, the global guide is selected using crowding distance in crowding distance MOPSO [41]. Moreover, Mostaghim and Teich selected the local guide based on the sigma method [42], while Pulido and Coello used a clustering technique for the same purpose [43]. On the other hand, in terms of population diversity, a genetic operator and special domination principle have been employed to improve the diversity of the Pareto front. Zitzler proposed the elitism mechanism that employs crossover and mutation on individuals that have been selected from the combination of population and repository [44], while Laumanns proposed Ɛ-box dominance to combine diversity and convergence [45]. Tang and then Yu used a simulated binary crossover (SBX) in their work [46] [47]. Additionally, in Pareto entropy MOPSO, a cell distance-based individual density is used to select the global guide [48].

2.3. Cultural Algorithm Phenomena

According to [49], a cultural algorithm formed of five types of knowledge: situational, normative, topographical, domain, and historical knowledge, see Figure (2.1).

1. Situational knowledge is a set of objects, which are useful for the experience interpretation of all individuals in a certain population. In other

words, Situational knowledge guides the individuals to move toward exemplars (best local or best global guides).

2. Normative knowledge: includes a set of promising ranges of decision variables. It offers strategies for individual adjustments. More precisely, it leads individuals to dive into a good range.
3. Topographical knowledge: it splits the completely feasible search landscape into cells, which each of these cells represents a different spatial characteristic; also, each cell selects the best individual in its specific ranges. Keeping the idea simple, the topographical knowledge leads individuals toward the best cell.
4. Domain knowledge: it records information about the problem domain to guide the whole search, it is considered useful during the search process.
5. Historical knowledge: it records the key events in the search process by keeping track of the history of significant individuals. Key events might be considerable move in the search space, or sometimes it comes in the form of notable changes in the search landscape. Individuals are using historical knowledge in order to select a preferable direction.

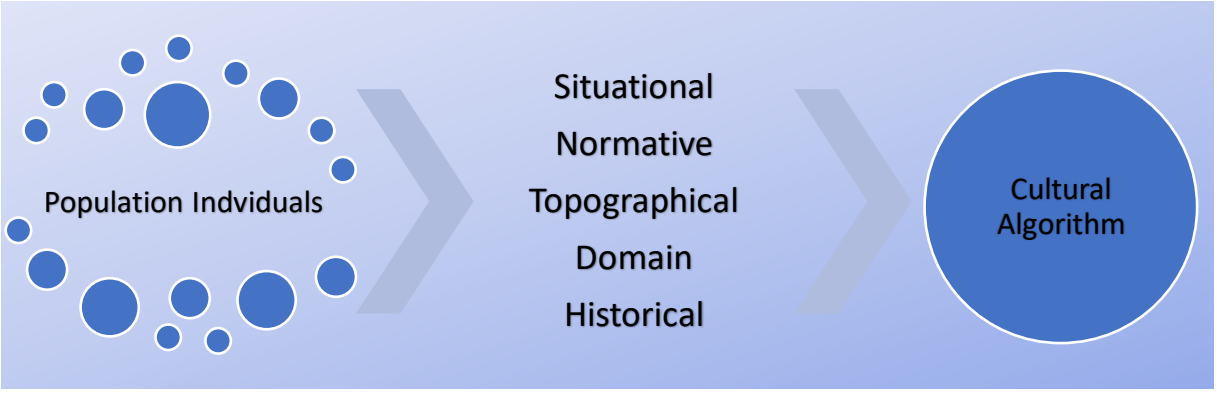

Figure 2.1 Explains how the combination of separated uninformed individuals and knowledge can form an intelligent cultural algorithm.

2.4. Pareto Optimal Solutions Set

Mathematically speaking, MOPs can be represented as follows with no loss of generality:

Minimize: $\vec{F}(\vec{x}) = \{f_1(\vec{x}),\ f_2(\vec{x}), \dots, f_n(\vec{x})\}$ (2.1)

Subject to:

$$g_{i(\vec{x})} \leq 0, \qquad i = 1,2, \dots, m$$

$$h_{i(\vec{x})} = 0, \qquad i = 1,2, \dots, p$$

Where: n is a number of objectives, g and h are constraints, m is a number of inequality constraints and p is a number of equality constraints [50].

Having said that, this type of problems cannot be optimized normally with traditional single-objective algorithm, not just because of multi-objective nature, but also because of existence of conflicting objectives in the same optimization problem, which means, there is no single optimum solution, instead, there is a set of optimum solutions known as Pareto optimality solutions set, which represents the best trade-offs between objectives. For readability purposes, Pareto optimality will be discussed briefly. Pareto optimality solutions can be explained using the following definitions [51]:

Def. #1: For vectors (solution) $\vec{a}$ and $\vec{b}$ in optimization problem K^t. For $i = 1, 2, \dots, m$, $\vec{a} \leq \vec{b}$ if objectives of vector $\vec{a}$ smaller or equal than objectives of vector $\vec{b}$ and at least there is $\overrightarrow{a_i} < \overrightarrow{b_i}$.

Def. #2: If $\vec{a} \leq \vec{b}$ then: $\vec{a}$ dominates $\vec{b}$, and denoted by $\vec{a} \prec \vec{b}$.

Def. #3: Two solutions might not dominate each other if Def. #1 is not applied, in this case, solutions $\vec{a}$ and $\vec{b}$ are nondominated with respect to each other, and denoted as $\vec{a} \nprec \vec{b}$ is the set of all nondominated known as Pareto optimal solution set P_s, and defined as Equation (2.2):

$$P_s \coloneqq \{a\ , b\ \in K \mid \exists F(a) \succ F(b)\} \quad (2.2)$$

Def #4: The set is holding equivalent object values of Pareto optimal solutions in P_s, is known as Pareto optimal front P_f (see Figure 2.2), and defined as Equation (2.3):

$$P_f \coloneqq \{F(\vec{a}\ ) \mid \vec{a}\ \in |P_s\} \quad (2.3)$$

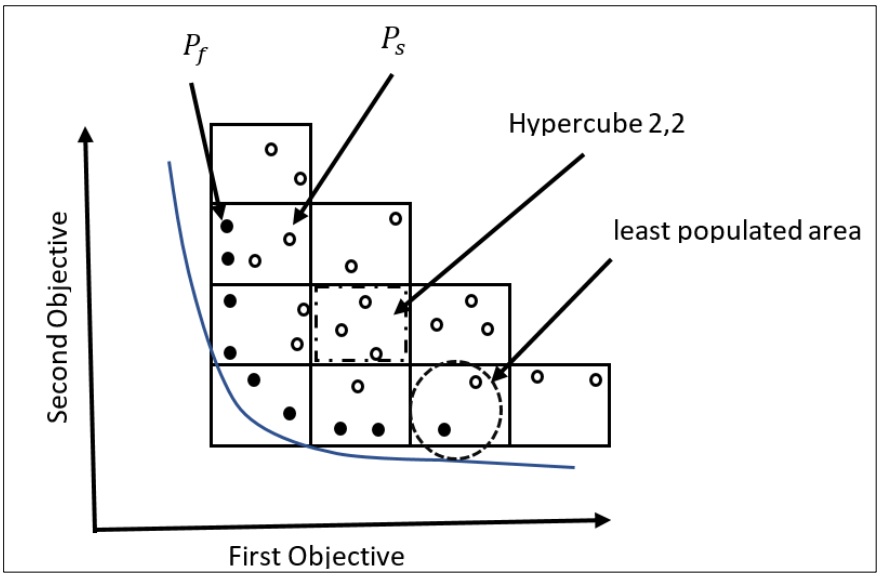

Figure 2.2 Shows Pareto solution, Pareto front, and hypercube grids which is provide helps in selecting global and local guide.

2.5. Honey Bee

Since ancient times, this remarkable social insect has been one of the most famous creatures on the planet. Honeybees have been the subject of scientific observations. Likewise, considerable research and many books have been published about them, for example, “Behavior and the Social Life of Honeybees” by Ribbands in 1953. Snodgrass wrote “Anatomy of the Honey Bee in 1956, also “the wisdom of hive” by Thomas D. Seeley was written in 1995, and many other great works. Figure (2.3) shows the anatomy of a bee. Presenting the colony structure of bees and their biological details are beyond the scope of this paper. However, the swarming behavior of the bee life cycle will be discussed shortly since it is related to SOFDO.

As widely known, bees live and work in groups inside a colony called a hive (nest site). In brief, there are several types of bees: queen bees, worker bees, and scout bees. As their names suggest, the queen bee is responsible for making decisions and producing the next generation of bees. Worker bees work under the command of queen bees; they also create queen cells throughout the year. Finally, scout bees explore the environment and exploit the preferable targets, which is the most important feature of this work. Usually, when the number of bees in the hive increases and the inside colony conditions and outside weather conditions are suitable, then the queen lays eggs into the queen cells, and the bee colony starts the reproductive processes by swarming [52] [53].

Swarming is mostly a late spring phenomenon; it is the process by which a new honeybee colony is formed. The queen bee leaves the old colony with a group of worker bees and some scout bees; Figure (2.4) shows the swarming cycle. A swarm typically consists of thousands to tens of thousands of bees. They settle 20–30 meters away from the natal hive temporarily for a few hours to a few days. They may gather in a tree or on a branch where they cluster around their queen, and then, they send 20 -50 scout bees out to find suitable new hives, usually after several tries, which might take several hours or up to three days. Eventually, with the guidance from the scouts, the rest of the bees flying overhead in the proper direction.

A swarm may fly a kilometer or more to the scouted location. Through direct observation, it can be said that the scout bee has several criteria for a suitable hive. For instance, a suitable hive has to be large enough to accommodate the whole swarm (minimum of 15 liters, preferably 40 liters in volume). It should have a small entrance (approximately 12.5 cm2), as well as being located at the lowest point of the hive, and obtain a certain amount of warmth from sunlight [53], [54].

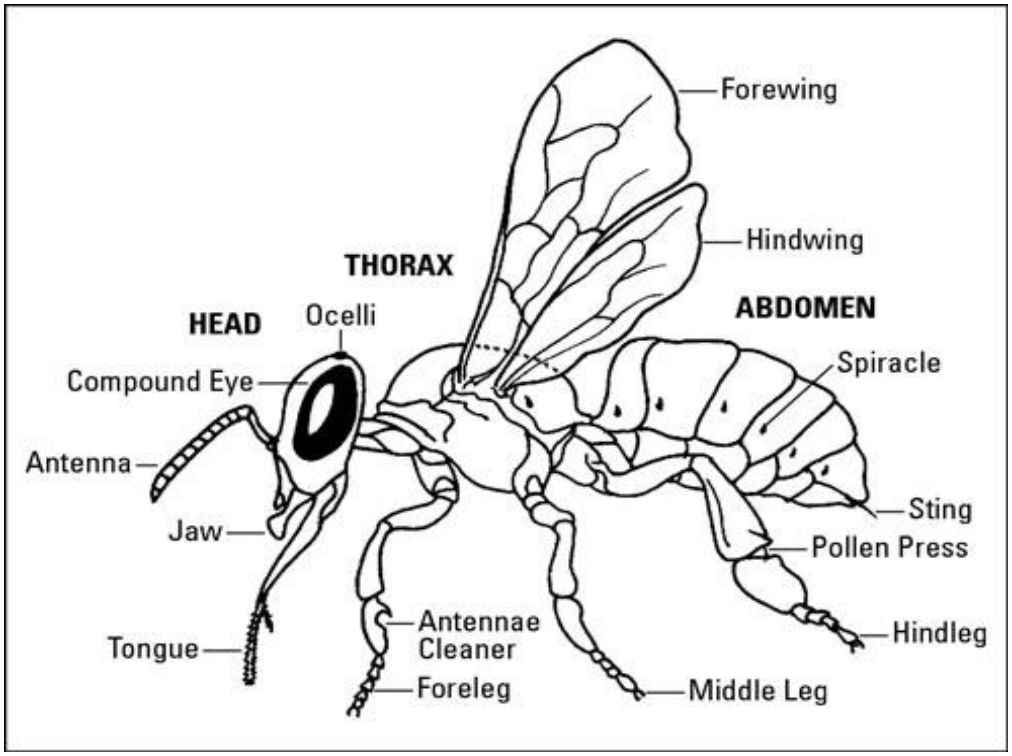

Figure. 2.3 Honey bee anatomy [55]

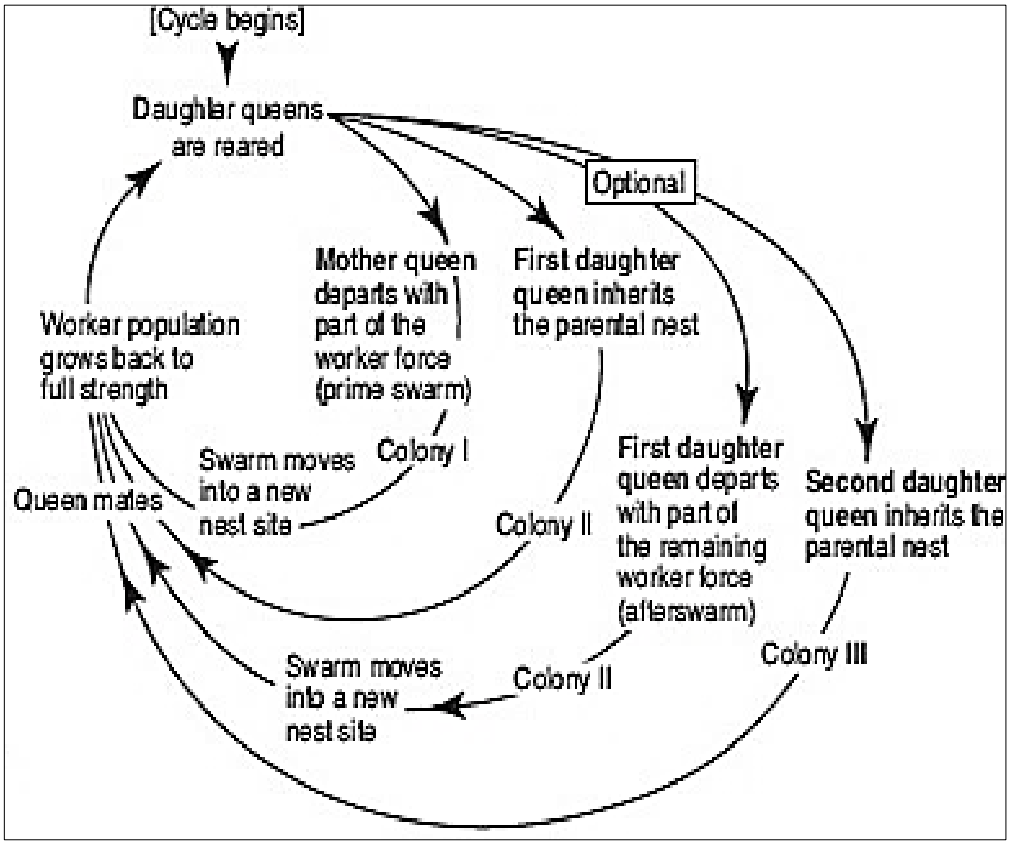

Figure. 2.4 Bee swarming process cycle [56]

Chapter Three

Methodology

Chapter Three

3. Methodology

In this chapter, in the first section, the source of inspiration (biological aspects of the honey bee) of this work is connected to the actual algorithm structure (algorithm aspects). In the second section, SOFDO is presented, mathematically explained through a sequence of related equations, and programmatically is illustrated using pseudocode and flowchart with detailed descriptions. In section three, MOFDO is presented, again, the algorithm is mathematically and programmatically presented in details using pseudocode and flowchart, the algorithms are constructed in a way that other researchers can easily duplicate our work.

3.1. Inspiration

What inspired us were the scouts' collective decision-making processes. When a number of scout bees discover some suitable hives, they will choose the most suitable hive, and they keep the swarm intact. Typically, scout bees communicate through moving their legs and wings, which is known as a bee dance. Usually, a decision will be made when approximately 80% of the scouts have agreed upon a certain hive location or when there is a quorum of 20-30 scout bees present at a potential hive [53], [54].

Table 3.1 SOFDO-related bee biological entities

Nature	Algorithm
Scout bee	Search agent
Hive	Solution discovered
Hive specification	Fitness function
Scout collective decision	Fitness Weight
Selected hive	Optimum Solution

Algorithmically speaking, each hive that a scout bee exploit, represents a possible solution exploited by an artificial search agent, and the best hive represents the global optimum solution, as shown in Table (3.1). The hive specifications, such as its volume, entrance size, entrance location, and amount of sunlight, can also be considered as fitness functions of the solution. The scout's collective decision-making process, represented by fitness weight (fw) in the algorithm, fw is discussed further in the next section.

3.2. Single-Objective Fitness Dependent Optimizer Algorithm

This algorithm replicates what a swarm of bees is doing during the reproduction process. The main part of this algorithm is taken from the process of scout bees searching for a new suitable hive among many potential hives. Every scout bee that searches for new hives represents a potential solution in this algorithm; furthermore, selecting the best hive among several good hives is considered as converging to the optimality.

3.2.1. Mathematical Definitions of Single-Objective Fitness Dependent Optimizer

The algorithm begins by randomly initializing an artificial scout population in the search space $X_i(i = 1,2, \dots n)$; where X_i represent a scout bee, each scout bee position represents a newly discovered hive (solution). Scout bees try to find better hives by randomly searching more positions; each time a better hive is found, the previously discovered hive is ignored; thus, each time the algorithm identifies a new, better solution, then the previously discovered solution will be ignored. In addition, if the current move is not leading the artificial scout bee to a better solution (hive), it will continue in its previous direction, hoping that the previous direction takes the scout to a better solution. However, if the previous direction does not lead to a better solution, it will then continue to the current solution, which is the best solution that has been found to that point.

In nature, scout bees search for hives randomly. In this algorithm, artificial scouts initially search the landscape randomly using a combination of a random walk and fitness weight mechanism. Accordingly, every time an artificial scout bee moves by adding pace to the current position, the scout hopes to explore a better solution. The movement of artificial scout bees is expressed as follows:

$$X_{i,t+1} = X_{i,t} + pace_{i,t} \tag{3.1}$$

Where *i* represents the current search agent, *t* represents the current iteration, *x* represents an artificial scout bee (search agent), and $pace_{i,t}$, is the movement rate and direction of the artificial scout bee. $pace_{i,t+1}$ is mostly dependent on the fitness weight fw. However, the direction of $pace_{i,t}$ is completely dependent on a random mechanism. Thus, the fw for minimization problems can be calculated as:

$$fw = \left| \frac{x^*_{i,t\ fitness}}{x_{i,t\ fitnees}} \right| - wf \tag{3.2}$$

The $x^*_{i,t\ fitness}$, is a fitness function value of the best global solution that has been discovered thus far. $x_{i,t\ fitness}$ is a value of the fitness function of the current solution, wf is a weight factor, and its value is either 0 or 1, which is used for controlling the fw. If it is equal to 1, then it represents a high level of convergence and a low chance of coverage. Nonetheless, if $wf = 0$, then it is not affecting the Equation (3.2), thus it can be neglected, setting $wf = 0$ provides a more stable search. However, this is not always the case; sometimes, the opposite occurs because the fitness function value is completely optimization problem-dependent. Nevertheless, the fw value should be in the [0, 1] range; however, there are some cases where fw = 1, for example, when the current solution is the global best, or when the current and global best solutions are identical or have the same fitness value. Additionally, there is a chance that fw = 0, which occurs when $x^*_{i,t\ fitness} = 0$. Finally, division by

zero should be avoided when $x_{i,t\ fitness} = 0$, therefore, the following rules should be used:

$$\begin{cases} fw = 1 \ or \ fw = 0 \text{ or } x_{i,t\ fitness} = 0, \quad pace_{i,t} = x_{i,t} * r & (3.3) \\ fw > 0 \ and \ fw < 1 \begin{cases} r < 0, pace_{i,t} = \left(x_{i,t} - x^*_{i,t}\right) * fw * -1 & (3.4) \\ r \geq 0, \quad pace_{i,t} = \left(x_{i,t} - x^*_{i,t}\right) * fw & (3.5) \end{cases} \end{cases}$$

Here, r is a random number in the [-1, 1] range. There are different implementations of the random walk; however, Levy flight has been chosen since it provides more stable movements because of its good distribution curve [2].

3.2.2. Single-Objective Fitness Dependent Optimizer Algorithm Time and Space Complexity

Regarding SOFDO mathematical complexity: For each iteration, it has an *O(p*n + p*CF)* linear time complexity, where *p* is the population size, *n* is the dimension of the problem, and *CF* is the cost of the objective function. Whereas, for all iterations, it has an *O(p*CF +p*pace)* space complexity, where the *pace* is the best previous paces stored. From here, SOFDO time complexity is proportional to the number of iterations. However, its space complexity will be the same during the course of iterations.

Nevertheless, SOFDO has a simple calculation mechanism in terms of objective value calculations, it has only (fitness weight and one random number) to be calculated for each agent, whereas, in PSO for calculating each solution, there are global best, agent best, search factors *C1* and *C2*, and random numbers (*R1* and *R2* parameters) to be calculated [8]. Also, in the DA, there are five different parameter weights to be calculated (*separation, alignment, cohesion, attraction, distraction*, and some random values), and most of these parameters have accumulative nature (summation and multiplication), and their values depend on all other agents' value, resulting in even more complex calculations [16].

3.2.3. Single-Objective Fitness Dependent Optimizer Algorithm Working Mechanism

The SOFDO with single-objective optimization problems begins by initializing artificial scouts at random locations on the search landscape, using upper and lower boundaries. For every iteration, the global best solution is selected; then, for every artificial scout bee, the fw is calculated according to Equation (3.2). After that, the fw value is checked to determine if $fw = 1$ or 0, also whether $x_{i,t\ fitness} = 0$. Then Equation (3.3) is used for generating the $pace_{i,t}$. However, if $fw > 0\ and\ fw < 1$, then a random number r will be generated in the [-1, 1] range. If $r < 0$ then Equation (3.4) is used to calculate the $pace_{i,t}$, in this case, fw gets a negative sign, but if $r \geq 0$ then Equation (3.5) is used to calculate the $pace_{i,t}$, accordingly, fw gets a positive sign. Randomly selecting a negative or positive sign for a fw will guarantee that the artificial bee will search randomly in every direction.

In SOFDO randomization mechanism controls the $pace_{i,t}$ size and direction, whereas in most cases, the randomization mechanism only controls the $pace_{i,t}$ direction; in these cases, the $pace_{i,t}$ size depends on the fw. Moreover, each time the artificial scout bee finds a new solution, it checks whether the new solution is better than the current solution, depending on the fitness function as shown in Figure (3.1) the pseudocode of the SOFDO. If the new solution is better, then it is accepted, and the old solution is ignored. Additionally, one of the special features of SOFDO is that if the new solution is not better, then the artificial scout bee continues using the previous direction (using the previous $pace_{i+t}$ value if available, here the *t* is the latest iteration that the solution is ever accepted in), but only if it takes the scout bee to a better solution. Moreover, the algorithm is explained using the flowchart diagram as shown in Figure (3.2).

```
Initialize scout bee population X_{i,t} (i = 1, 2, ..., n)
while iteration (t) limit not reached
    for each artificial scout bee X_{i,t}
        find best artificial scout bee x*_{i,t}
        generate random-walk r in [-1, 1] range
        if( X_{i,t} fitness == 0) (avoid divide by zero)
            fitness weight = 0
        else
            calculate fitness weight. equation (3.2)
        end if
        if (fitness weight = 1 or fitness weight = 0)
            calculate pace_{i,t} using equation (3.3)
        else
            if (random number >= 0)
                calculate pace_{i,t} using equation (3.5)
            else
                calculate pace_{i,t} using equation (3.4)
            end if
        end if
        calculate X_{i,t+1} equation (3.1)
        if( X_{i,t+1} fitness < X_{i,t} fitness)
            move accepted and pace_{i,t} saved
        else
            calculate X_{i,t+1} equation (3.1) with previous pace_{i,t}
            if (X_{i,t+1} fitness < X_{i,t} fitness)
                move accepted and pace_{i,t} saved
            else
                maintain current position (don't move)
            end if
        end if
    end for
end while
```

Figure. 3.1 Pseudocode of SOFDO

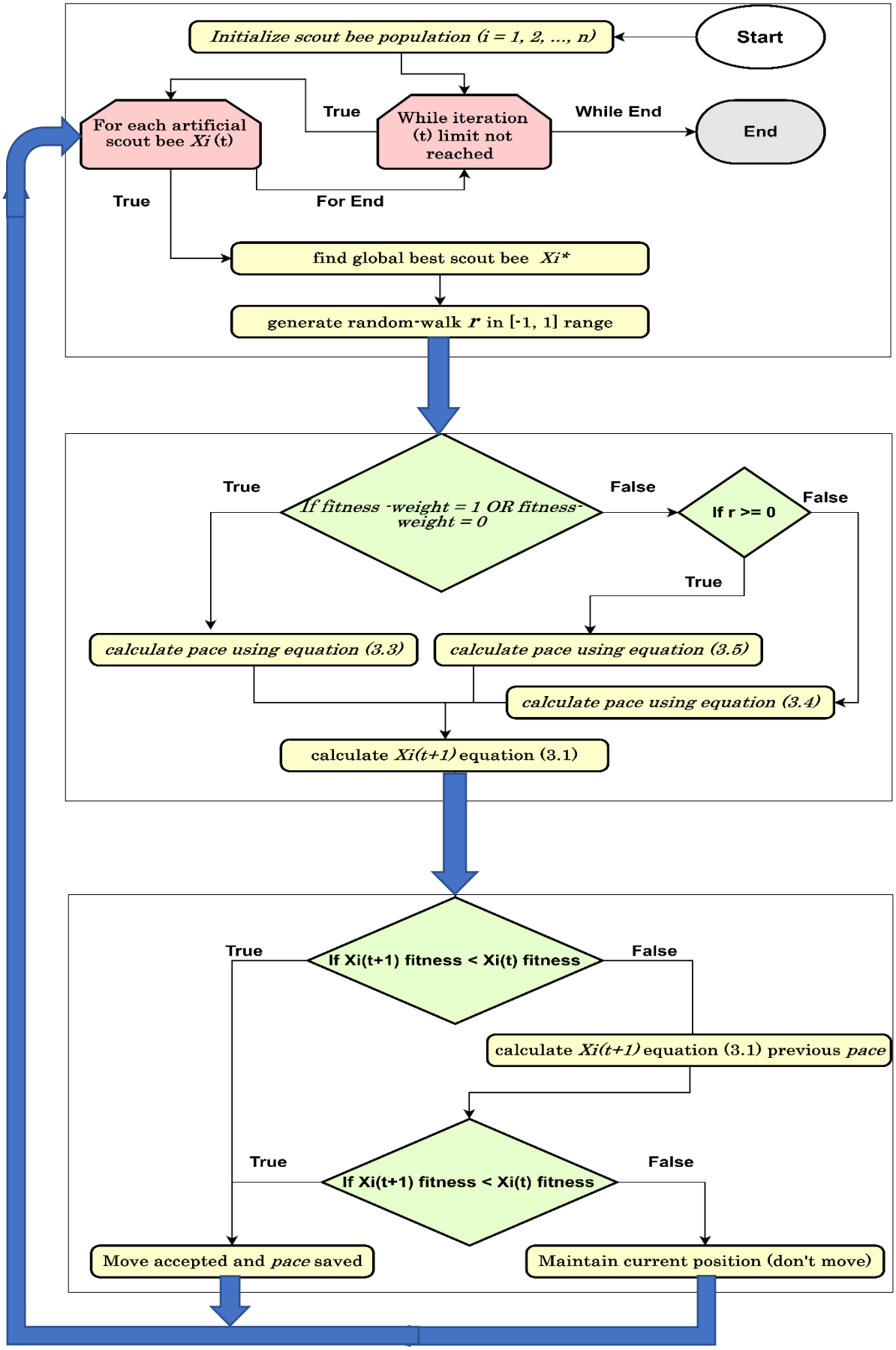

Figure. 3.2 Flowchart diagram of SOFDO

In addition, if using the previous $pace_{i,t}$ is not leading the scout bee to a better solution, then SOFDO maintains the current solution until the next iteration. In this algorithm, every time the solution is accepted, its $pace$ value is saved for potential reuse in the next iteration.

When implementing SOFDO for maximization problems, two minor changes are needed. First, Equation (3.2) must be replaced by Equation (3.6), as Equation (3.6) is simply an inverse version of Equation (3.2).

$$fw = \left| \frac{x_{i,t\ fitness}}{x^*_{i,t\ fitness}} \right| - wf \tag{3.6}$$

Second, the condition for selecting a better solution should be changed. The line: "if ($X_{i,t+1} fitness < X_{i,t}\ fitness$)" must be replaced with the line: "if ($X_{i,t+1}\ fitness > X_{i,t}\ fitness$)" in both occurrences in the pseudocode shown in Figure (3.1).

3.3. Multi-Objective Fitness Dependent Optimizer

Multi-objective fitness dependent optimizer (MOFDO) is based on SOFDO. Having said that, the SOFDO is a metaheuristic algorithm, the bee swarming reproductive process and their collective decision-making has inspired this algorithm. Algorithmically speaking, SOFDO updates the individual position by adding *pace* to the current position as shown in Equation (3.1); the same mechanism is also applied in MOFDO. However, in order to calculate the $pace$ we need to consider the conditions that presented in Equations (3.7, 3.8, and 3.9), and these conditions depend on the fitness weight (fw) value, summation of objective functions or a randomization mechanism. fw can be calculated using the problem cost function values according to Equation (3.10). It is worth mentioning that, the $pace$ represents both domain and historical knowledge in MOFDO.

3.3.1. Mathematical Definitions Multi-Objective Fitness Dependent Optimizer

As mentioned before, the algorithm moves its individual search agent by adding the $pace$ to the current position, as shown in Equation (3.1).

$$\begin{cases} fw = 1 \; or \; fw = 0 \; or \; \sum_{o=1}^{n} x_{i,t\,fitnees_o} = 0, \quad pace_{i,t} = x_{i,t} * r & (3.7) \\ fw > 0 \; and \; fw < 1 \; \begin{cases} r < 0, pace_{i,t} = \left(x_{i,t} - x_{i,t}^{*}\right) * fw * -1 & (3.8) \\ r \geq 0, \quad pace_{i,t} = \left(x_{i,t} - x_{i,t}^{*}\right) * fw & (3.9) \end{cases} \end{cases}$$

Where i represents the current individual number, t represents the current iteration, x is the individual itself, and $pace$ is the movement rate and direction. Recalling that the $pace$ value mostly relies on the fw. However, the direction of $pace$ (value sign) entirely depends on a random mechanism. The $\sum_{o=1}^{n} x_{i,t\,fitnees_o}$ is the sum of the cost function of the current individual, and n is the number of objectives, and $o = \{1, 2, \ldots, n\}$. Equations (3.7, 3.8, and 3.9) contain two different conditions. Firstly, if fw is equal to zero, or if fw is equal to one, or if $\sum_{o=1}^{n} x_{i,t\,fitnees_o} = 0$, then the $pace_{i,t}$ should be calculated as Equation (3.7). Secondly, if fw value comes in between zero and one, then random number r is generated in [-1, 1] range if r is a negative number, then Equation (3.8) will be used, otherwise, Equation (3.9) will be used for calculating the $pace_{i,t}$. fw Can be calculated using Equation (3.10):

$$fw = \left| \frac{\sum_{o=1}^{n} x_{i,t\,fitness_o}^{*}}{\sum_{o=1}^{n} x_{i,t\,fitnees_o}} \right| - wf \qquad (3.10)$$

Where the $\sum_{o=1}^{n} x_{i,t\,fitness_o}^{*}$ is a sum of the cost function of the global best individual, again n is the number of objectives, and $o = \{1, 2, \ldots, n\}$. Finally, wf in Equation (3.10) is a weight factor, and its value is either 0 or 1. One may notice that, when wf = 0, it does not affect the equation and can be ignored.

3.3.2. Multi-Objective Fitness Dependent Optimizer Extra Features

Although, the algorithm structure is the same as SOFDO to some extends, however, there are several additional improvements in MOFDO:

A. An archive (repository) is used for holding Pareto front solutions during optimization, as it has been widely used in the literature for this purpose.

B. Before adding the non-dominated solution to the archive, a polynomial mutation is applied to each solution of Pareto front solutions. The polynomial mutation has been employed in EOAs as a variation operator [57], and it is defined as Equation (3.11) [58]:

$$S_i = (x_1, x_2, \dots, x_n)$$

$$S_{Ni}(x_j) = S_i(x_j) + \alpha.\beta_{max}(x_j), \qquad i = 1,2,\dots,NP, j = 1,2,\dots n$$

$$\alpha = \begin{cases} (2v)^{\frac{1}{(q+1)}} - 1, & v < 0.5 \\ 1 - \left(2(1-v)\right)^{\frac{1}{(q+1)}}, & otherwise \end{cases} \tag{3.11}$$

$$\beta_{max}(x_j) = Max\left[S_i(x_j) - l_j, u_j - S_i(x_j)\right], i = 1,2,\dots,NP, j = 1,2,\dots n$$

Where S_{Ni} is a new solution, $S_i(x_j)$ is a current solution, $\beta_{max}(x_j)$ is the maximum perturbation acceptable between original and mutated solution, NP is a population size, q is a positive real number, v is a uniformly distributed random number in [0, 1] range, l is a lower boundary of decision variable x, u is an upper boundary of decision variable x, and n number of decision variables (problem dimensions).

C. In MOPs, we cannot simply choose the fittest solution as a global guide (normative knowledge), as this might be the case in single-objective optimization, since there is more than one objective, and usually these objectives are in conflict with each other. Therefore, selecting a global guide needs a more careful decision. In this work, the global guide individual is denoted by x^*, which is a global-best non-dominated solution, it is selected from the least populated region by artificial scout

bees the same as to [59] work on MOPSO. For this purpose, a mechanism called archive controller is used to divide the archive into multiple equally sized grid (sub-hyper-spheres in multi-dimension problems) [36], in our case, we call them hypercube grids, which represent a topographical knowledge usage in MOFDO. The hypercube grid mechanism allows the algorithm to determine the least populated area, simply by counting the number of solutions in each grid, then the global best solution will be selected from the least populated area, see Figure (2.2) chapter two. The reason behind selecting the global guide from the least populated area is to maintain a good diversity in the obtained Pareto front solutions. As a result, decision-makers will have more diverse choices (solutions) to consider. Nevertheless, the archive has a limited size, when a new non-dominated solution is found and the archive is already reached its maximum capacity, the archive controller removes the worst solution form the most populated grid, so that, the newly discovered solution can fit in, as long as the new solution is better than the worst solution in the archive.

D. Regarding selecting the personal guide (situation knowledge), the same hypercube grid mechanism has been used for dividing the search landscape into equally sized cells, then inside each cell, the best personal solution is selected as a local guide.

3.3.3. Multi-Objective Fitness Dependent Optimizer Working Mechanism

The MOFDO starts by randomly distributing the search individuals over the search space as presented in the pseudocode Figure (3.3), and more explanations are given in the flowchart Figure (3.4). Then after, an archive with a specific size is created, and hypercube grids are generated.

From here, the main algorithm loop will start in Line (4), which mainly depends on a specific number of iterations or until a certain condition is met.

In the Line (5), for each search individual (artificial scout bee) operations from Lines (6 to 27) will be repeated according to the number of individuals. The mentioned operations include: finding the global best search agent, finding fw using Equation (3.10), Lines (10 to 12) applying conditions from Equations (3.7, 3.8, and 3.9) in order to calculate $pace_{i,t}$, then afterword, Line (13) calculating a new search agent position using Equation (3.1). When the new search agent is discovered, we always check whether the new result (cost function) is dominating the old result or not, see Line (14). If it is, then the new position will be accepted and the $pace_{i,t}$ will be stored for potential reuse in the future as shown in Line (15). However, if it is not, then if previously saved $pace_{i,t}$ available, then it will be used instead of the new one hoping to generate a better result unless the search agent will maintain current position, see Lines (17 to 22). The polynomial mutation will apply in order to get more variant solutions in Line (24), then check whether the solution can fit inside the archive or not Lines (25 and 26). Hypercube grid indices always update according to the search landscape changes in Line (27).

```
Initialize the artificial scout bee population X_{i,t} i = 1, 2, ..., n, and t=1,2, ..., m
Creating archive for nondominated solutions
Generate Hypercube grid
While (t) iteration limit not reached (m) (or solution good enough)
    for each artificial scout bee X_{i,t}
        find best artificial scout bee X*_{i,t}
        generating random walk r in [-1, 1] rang
        calculating fitness weight value. Equation (3.10)
        //checking Equation (3.7,3.8, and 3.9) conditions
        if (fitness_weight ≥ 1 or fitness_weight ≤ -1 or ∑_{o=1}^{n} x_{i,t fitnees_o} = 0)
            fitness_weight = r
        end if
        calculate X_{i,t+1}    equation (3.1)
        if( X_{i,t+1} fitnesses dominate X_{i,t} fitnesses)
            move accepted and pace saved
        else
            calculate X_{i,t+1} equation (3.1) with previous pace
            if (X_{i,t+1} fitnesses dominate X_{i,t} fitnesses)
                move accepted and pace saved
            else
                maintain current position (don't move)
            end if
        end if
        Apply polynomial mutation
        Add non-dominated Bees (solutions) to archive
        Keep only non-dominated members in the archive
        Update Hypercube Grid Indices
    end for
    end while
```

Figure 3.3 MOFDO pseudocode

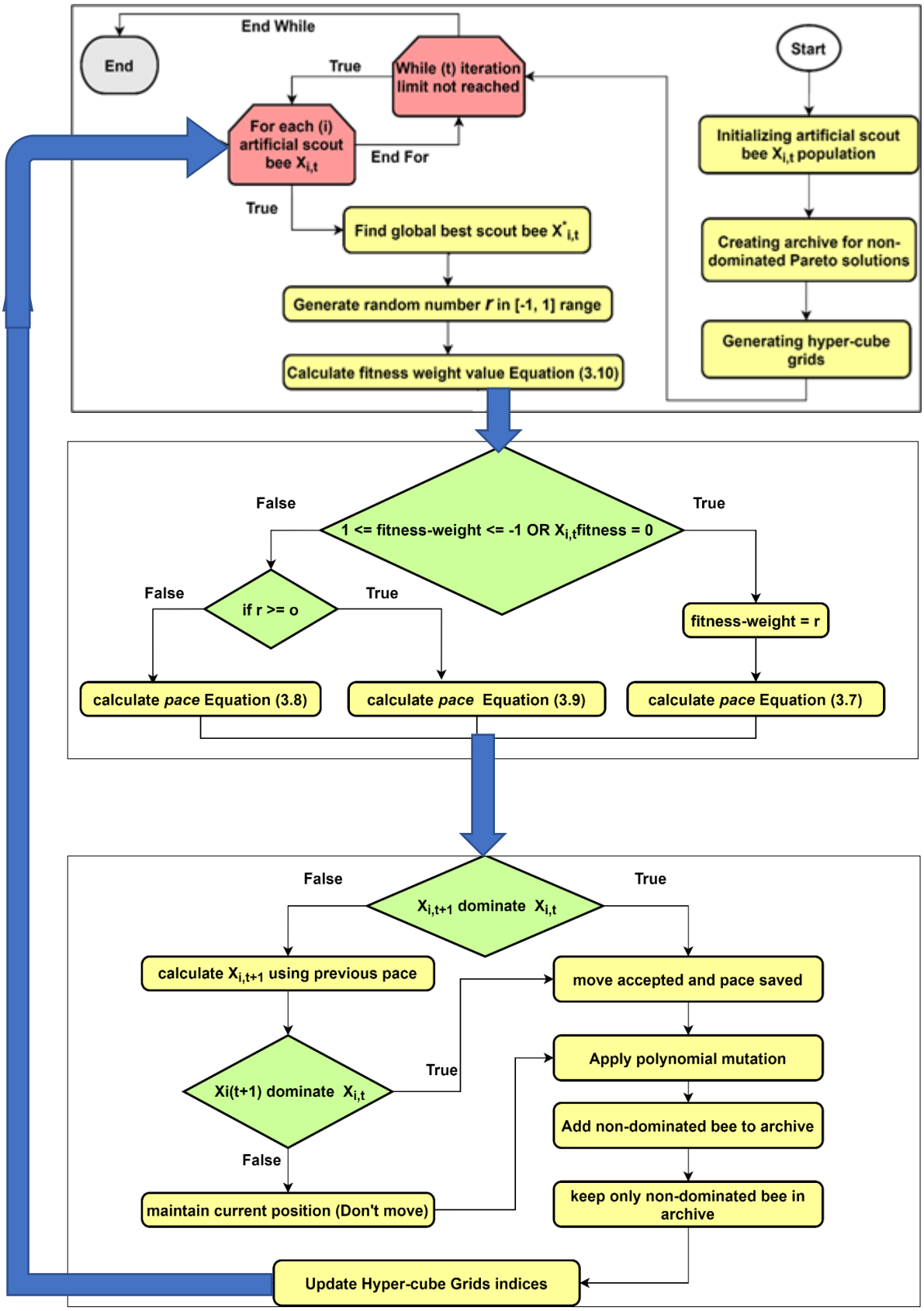

Figure 3.4 Flowchart of MOFDO shows how it works programmatically

Chapter Four

Results & Discussions

Chapter Four

4. Results and Discussions

In this chapter, the results are presented, then directly analyzed and discussed for readability purposes. Thus, the result section is relatively long, so that, having results analysis is discussed in every section will provide a more straightforward mechanism.

Having said that, both algorithms are tested on 19 classical standard benchmarks, 10 CEC-06 2019 modern test function, 5 ZDT multi-objective standard test functions, and 12 multi-modal multi-objective functions, the results of our algorithm compared to PSO, DA, GA, WOA, SSA, MOPSO, MODA and (NSGA-III) algorithms. Finally, both of the proposed algorithms applied to real-world application such as aperiodic antenna array designs, frequency-modulated sound waves, and welded beam design engineering problem

4.1. Results and Discussions of Single-Objective Fitness Dependent Optimizer

To test the performance of the SOFDO algorithm, a number of standard benchmark test functions exist in the literature is used. Additionally, our results are compared to five other well-known algorithms in the literature: PSO, GA, DA, WOA, and SSA. It is worth mentioning that results of (19 classical benchmark test functions) PSO, GA, and DA is taken from this work [16], however, we conducted the CEC-C06 tests. Also, all test results were compared using the Wilcoxon rank-sum test to prove their statistical significance. Moreover, four measurement metrics were used for further observation. Finally, the SOFDO was used for optimizing two real-world applications.

4.1.1. Classical Benchmark Test Functions

Three sets of test functions are selected to test the performance of the SOFDO algorithm [16]. The test functions have different characteristics, for instance, unimodal test functions, multimodal test functions, and composite test functions. Each set of these test functions is used to benchmark certain perspectives of the algorithm. Unimodal benchmark functions, for example, are used for testing the exploitation level and convergence of the algorithm, as their name might imply that they have a single optimum. However, multimodal benchmark functions have multi optimal solutions, and they are used for testing the local optima avoidance and exploration levels. As in multimodal algorithms, there are many optimum solutions; one of them is a global optimum solution and most local optimum solutions. An algorithm must avoid local optimum solutions and converge to a globally optimum solution. Furthermore, the composite benchmark functions are mostly combined, shifted, rotated, and biased versions of other test functions. Composite benchmark functions provide diverse shapes for different regions of the search landscape; they also have a very large number of local optima. This type of benchmark function demonstrates that complications exist in real-world search spaces, see Appendix Tables (7.1, 7.2 and 7.3) [16].

Each algorithm in Table (4.1) has been tested 30 times by using 30 search agents each with 10 dimensions; in each test, the algorithm was allowed to look for the best optimum solution in 500 iterations, and then, the average and standard deviation were calculated. Regarding parameter sets, GA, PSO, and DA parameter set described in this paper [16]. But for SOFDO parameters, there is only wf to be tuned. In Table (4.1), for all test functions wf was equal to 0 except test function (2 and 6) where wf equal to 1. Every test function was minimized towards 0.0 except TF8, which was minimized towards -418.9829

Also, some test functions were shifted by some degrees from the origin point to prove that the algorithms were not biased towards the origin.

In Table (4.1), the results of SOFDO, DA, PSO, and GA are presented. The TF1 to TF6 results showed that SOFDO generally provided better results than the other algorithms; however, the TF7 results showed the other algorithms were better. SOFDO in TF8 showed poor performance even though it had better results than PSO. In contrast, TF9 SOFDO provided a better result than both GA and DA, and comparative results were produced by PSO. In TF10 to TF13 and TF18, SOFDO provided relatively comparative results to the other algorithms. However, the results of TF14 to TF17 and TF19 confirm that the SOFDO algorithm outperformed DA, PSO, and GA in all cases.

4.1.2. CEC-C06 2019 Benchmark Test Functions

A group of 10 modern CEC benchmark test functions is used as an extra evaluation on SOFDO, these test functions were improved by professor Suganthan and his colleges for a single-objective optimization problem [60], the test functions are known as “The 100-Digit Challenge”, which are intended to be used in annual optimization competition. See Appendix Table (7.4).

Functions CEC04 to CEC10 are shifted and rotated, whereas functions CEC01 to CEC03 are not. However, CEC04 to CEC10 test functions are scalable. The parameter set was defined by the CEC benchmark developer, as functions CEC04 to CEC10 were set as a 10-dimensional minimization problem in [-100, 100] boundary range, however, CEC01 to CEC03 have different dimensions as shown in Appendix Table (7.4). For more convenient, all CEC global optimum where unified toward point 1. SOFDO is competed with three modern optimization algorithms: DA, WOA, and SSA. The reasons behind selecting these algorithms are:

1. They are all PSO-based algorithms the same as SOFDO.
2. All of them are well cited in the literature.

Table 4.1 Classical benchmark results of selected algorithms with SOFDO [16]

Test Function	SOFDO		DA		PSO		GA	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std
TF1	7.47E-21	7.26E-19	2.85E-18	7.16E-18	4.2E-18	1.31E-17	748.5972	324.9262
TF2	9.388E-6	6.90696E-6	1.49E-05	3.76E-05	0.003154	0.009811	5.971358	1.533102
TF3	8.5522E-7	4.39552E-6	1.29E-06	2.1E-06	0.001891	0.003311	1949.003	994.2733
TF4	6.688E-4	0.0024887	0.000988	0.002776	0.001748	0.002515	21.16304	2.605406
TF5	23.50100	59.7883701	7.600558	6.786473	63.45331	80.12726	133307.1	85,007.62
TF6	1.422E-18	4.7460E-18	4.17E-16	1.32E-15	4.36E-17	1.38E-16	563.8889	229.6997
TF7	0.544401	0.3151575	0.010293	0.004691	0.005973	0.003583	0.166872	0.072571
TF8	-2285207	206684.91	-2857.58	383.6466	-7.1E+11	1.2E+12	-3407.25	164.4776
TF9	14.56544	5.202232	16.01883	9.479113	10.44724	7.879807	25.51886	6.66936
TF10	3.996E-15	6.3773E-16	0.23103	0.487053	0.280137	0.601817	9.498785	1.271393
TF11	0.568776	0.1042672	0.193354	0.073495	0.083463	0.035067	7.719959	3.62607
TF12	19.83835	26.374228	0.031101	0.098349	8.57E-11	2.71E-10	1858.502	5820.215
TF13	10.2783	7.42028	0.002197	0.004633	0.002197	0.004633	68,047.23	87,736.76
TF14	3.7870E-7	6.3193E-7	103.742	91.24364	150	135.4006	130.0991	21.32037
TF15	0.001502	0.0012431	193.0171	80.6332	188.1951	157.2834	116.0554	19.19351
TF16	0.006375	0.0105688	458.2962	165.3724	263.0948	187.1352	383.9184	36.60532
TF17	23.82013	0.2149425	596.6629	171.0631	466.5429	180.9493	503.0485	35.79406

TF18	222.9682	9.9625E-6	229.9515	184.6095	136.1759	160.0187	118.438	51.00183
TF19	22.7801	0.0103584	679.588	199.4014	741.6341	206.7296	544.1018	13.30161

Table 4.2 IEEE CEC 2019 benchmark results

Test Function	**SOFDO**		**DA**		**WOA**		**SSA**	
	Average	**STD**	**Average**	**STD**	**Average**	**STD**	**Average**	**STD**
CEC01	4585.27	20707.627	543×10^{8}	669×10^{8}	411×108	542×108	605×107	475×107
CEC02	4.0	3.22414E-9	78.0368	87.7888	17.3495	0.0045	18.3434	0.0005
CEC03	13.7024	1.6490E-11	13.7026	0.0007	13.7024	0.0	13.7025	0.0003
CEC04	34.0837	16.528865	344.3561	414.0982	394.6754	248.5627	41.6936	22.2191
CEC05	2.13924	0.085751	2.5572	0.3245	2.7342	0.2917	2.2084	0.1064
CEC06	12.1332	0.600237	9.8955	1.6404	10.7085	1.0325	6.0798	1.4873
CEC07	120.4858	13.59369	578.9531	329.3983	490.6843	194.8318	410.3964	290.5562
CEC08	6.1021	0.756997	6.8734	0.5015	6.909	0.4269	6.3723	0.5862
CEC09	2.0	1.5916E-10	6.0467	2.871	5.9371	1.6566	3.6704	0.2362
CEC10	2.7182	8.8817E-16	21.2604	0.1715	21.2761	0.1111	21.04	0.078

1. They are proven to have an outstanding performance both on benchmark test functions and real-world problems.
2. These algorithm implementations are publicly provided by their authors.

Regarding algorithms parameter settings, their default parameter settings were not modified during the tests, all competitors are set the same as the settings used in their original papers [16] [17] [18]. Interested readers can find these algorithms MATLAB implementations and their parameter setting specification here [61]. Additionally, SOFDO default parameter set $wf = 0$ is used for all test functions.

Each algorithm was allowed to search the landscape for 500 iterations using 30 agents, the detailed results for each algorithm can be found in the appendix section Tables (7.7 to 7.17). As shown in Table (4.2), SOFDO outperforms other algorithms except in CEC06. Even though other algorithms have a comparative result in CEC03, CEC05, and CEC09 benchmarks, for example, WOA has the same result as SOFDO in CEC03, but the WOA standard deviation is equal to 0, this shows that WOA has the same result every time it uses with no chance for further improvements.

4.1.3. Statistical Tests

To show that the results presented in Table (4.1) and Table (4.2) are statistically significant, the p-values of the Wilcoxon rank-sum test are found for all test functions, and the results of a statistical comparison are shown in Table (4.3) and Table (4.4). In Table (4.3), the comparison is conducted only between the SOFDO and DA algorithms, because the DA algorithm was already tested against both PSO and GA in this paper [16]. According to the mentioned work, it has been proven that the DA results are statistically significant compared with PSO and GA.

Table 4.3 The Wilcoxon rank-sum test for classical benchmarks

Function	SOFDO vs. DA (P-value)
TF1	0.000513631
TF2	0.7111046
TF3	8.2371E-05
TF4	1.0750E-08
TF5	0.0023817
TF6	1.7620E-04
TF7	6.6643E-12
TF8	1.4805E-34
TF9	5.8820E-08
TF10	5.6002E-17
TF11	0.012793
TF12	9.4381E-05
TF13	1.0426E-37
TF14	1.1142E-21
TF15	0.000094477
TF16	0.0000E+00
TF17	1.0668E-120
TF18	0.0000E+00
TF19	5.5373E-166

Again, as shown in Table (4.3), the SOFDO results are considered significant in all statistical tests (unimodal, multimodal and composite test functions), except in TF2, that is because the results are more than 0.05. There are two unusual results in the composite test functions in both TF16 and TF18 because the DA algorithm provided the same fitness function value for each of the 30 different individual tests.

Table (4.4) shows the Wilcoxon rank-sum test of SOFDO against DA, WOA, and SSA for 10 CEC benchmark test functions, the results show that SOFDO performances are statistically significant in all cases, except in test function CEC03 for DA and WOA algorithms, and test function CEC04 and CEC08 for WOA algorithm. The results of Table (4.3) and Table (4.4) prove that SOFDO results are statistically significant, consequently, the existence of the SOFDO algorithm is statistically feasible.

Table 4.4 The Wilcoxon rank-sum test (p-value) for CEC 2019

Function	SOFDO Vs. DA	SOFDO Vs. WOA	SOFDO Vs. SSA
CEC01	4.03455E-05	0.000108312	3.1831E-09
CEC02	1.81428E-05	3.17E-252	1.1E-196
CEC03	0.244847	0.363647	1E-306
CEC04	0.000124	0.095911	7.4365E-11
CEC05	6.05468E-09	0.007884	2.3682E-15
CEC06	2.90314E-09	2.196E-28	1.9679E-08
CEC07	2.69474E-10	1.0424E-06	6.2691E-15
CEC08	2.3638E-05	0.131704	5.2331E-06
CEC09	1.80487E-10	3.7992E-43	7.4029E-19
CEC10	2.2248E-111	6.397E-131	2.459E-122

4.1.4. Quantitative Measurement Metrics

For more detailed analyses and in-depth observation of the SOFDO algorithm, four more quantitative metrics were used, as shown in Figures (4.1, 4.2, 4.3 and 4.4). In each experiment, the first test function is selected from the unimodal benchmark functions (FT1 to FT7), the second test function selected is from the multimodal test functions (TF8 to TF13), and the last test function

is selected from the composite benchmark functions (TF14 to TF19). The experiment was conducted using 10 search agents, each allowed to search the two-dimensional landscape through 150 iterations.

The first metric measures the convergence and illustrates how well the artificial scout covers the search landscape. This is merely a search history of artificial scout movements because the position of the artificial scouts is recorded from the beginning to the end of the test. As presented in Figure (4.1), the scout quickly explores the overall area first and then gradually moves towards optimality.

The second metric measures the value of the search agent (fitness function value), as shown in Figure (4.2). The values start with large values and then steadily decrease. This behavior guarantees that SOFDO will eventually reach optimality [62].

The third test metric is shown in Figure (4.3) and shows that the average fitness value of all SOFDO agents decreased dramatically over the course of the iterations, which verifies that the algorithm not only improves the global best agent (x_i^*) but also improves the overall agent fitness values.

The fourth metric measures the convergence of the global best agent through the course of the iteration. This proves that x_i^* becomes more accurate as the number of iterations increases again, clear abrupt changes can be seen due to the emphasis on the local search and exploitation, see Figure (4.4).

Overall, in this section, measurement metrics showed that SOFDO is capable of effectively exploring the search space, improving the overall solution, avoiding local optimum and fairly converging towards optimality.

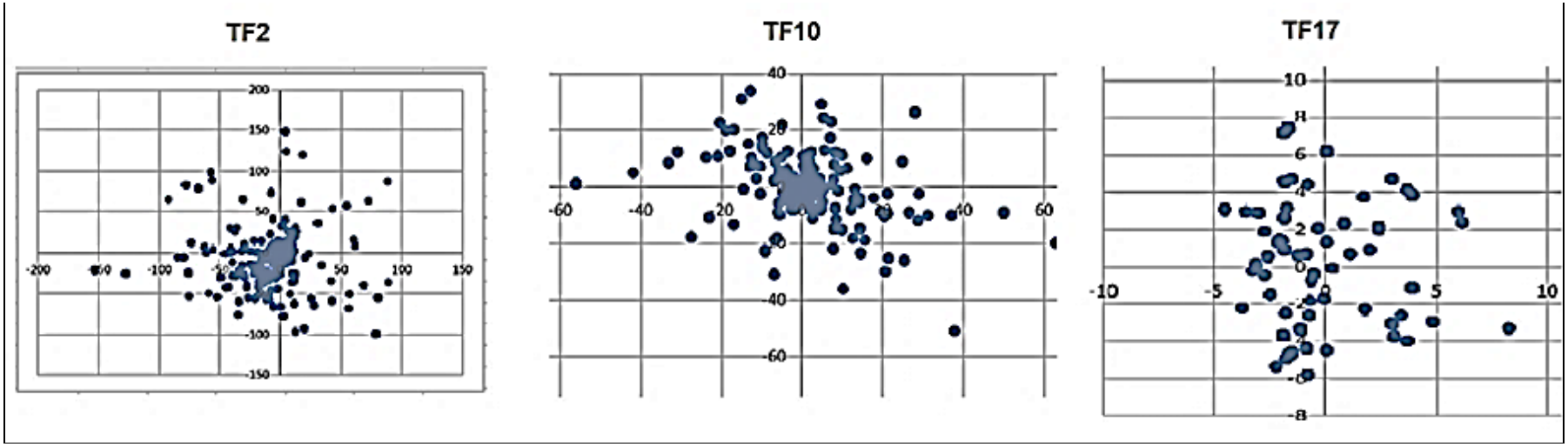

Figure 4.1 Search history of the SOFDO algorithms on unimodal, multimodal, and composite test functions

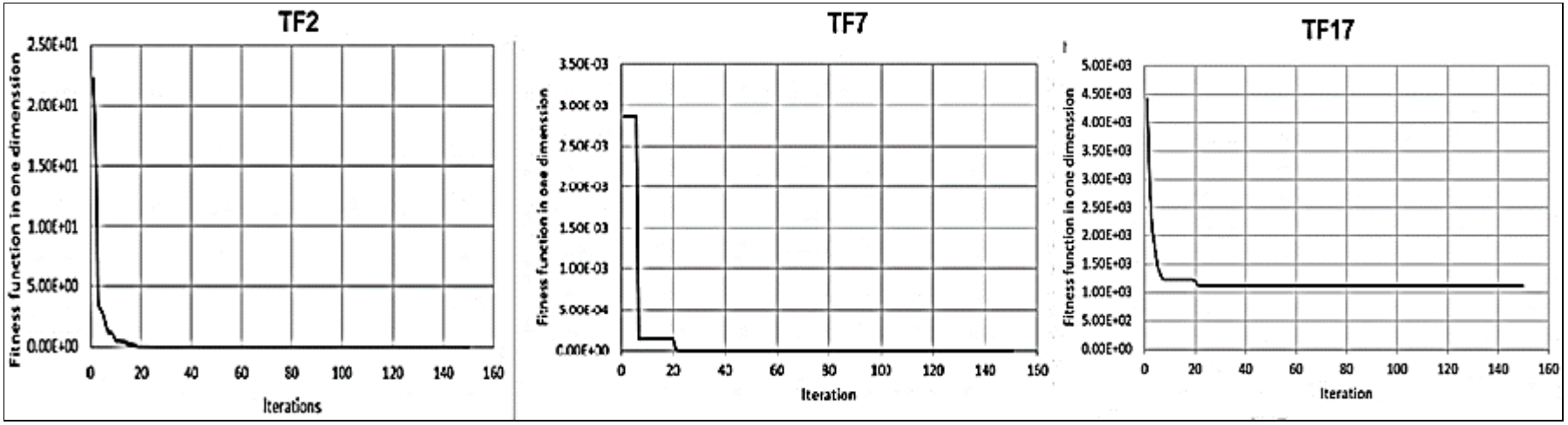

Figure 4.2 The trajectory of SOFDO's search agents on unimodal, multimodal, and composite test functions

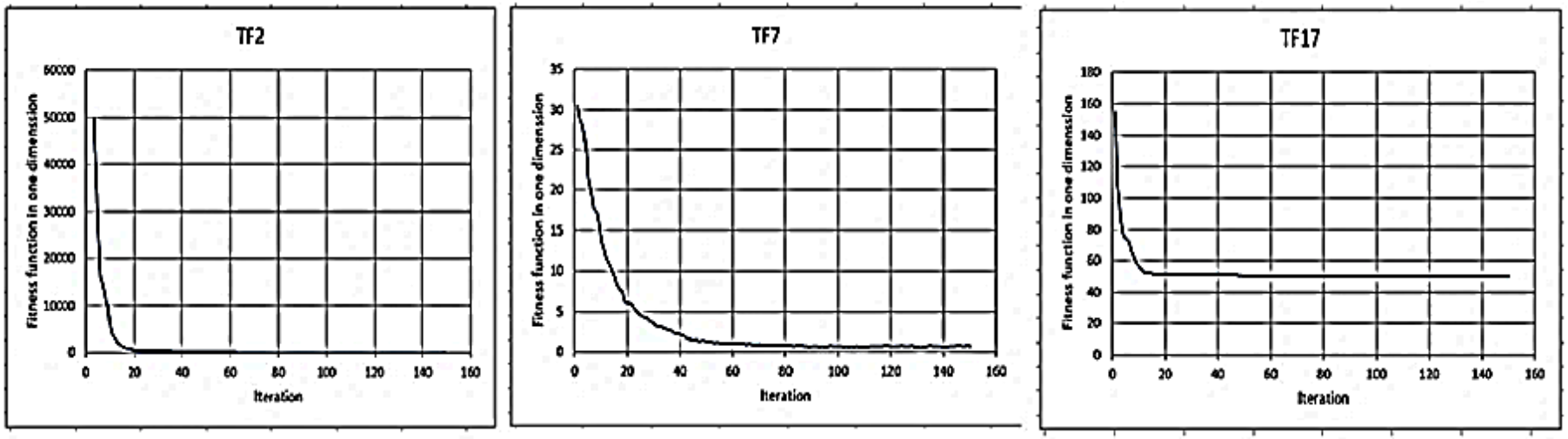

Figure 4.3 The average fitness of SOFDO's search agents on unimodal, multimodal, and composite test functions

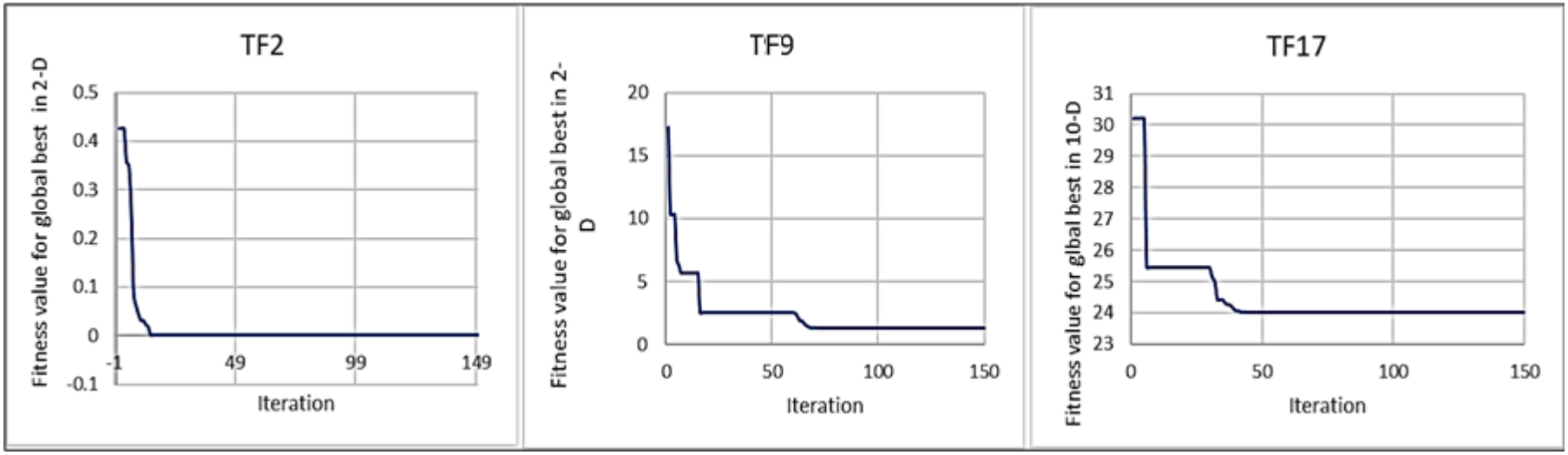

Figure 4.4 Convergence curve of the SOFDO's algorithms on unimodal, multimodal, and composite test functions

4.1.5. SOFDO Real-World Application

Similar to any other metaheuristic algorithm, SOFDO can be used to solve real-world application problems. In this section, SOFDO is applied to two different applications:

A. SOFDO Usage on Aperiodic Antenna Array Designs.

Since the 1960s, with the advances in both radar techniques and radio astronomy, aperiodic antenna arrays have received considerable attention, as shown in Figure (4.5); there are two types of aperiodic antenna arrays: non-uniform antenna arrays and thin antenna arrays.

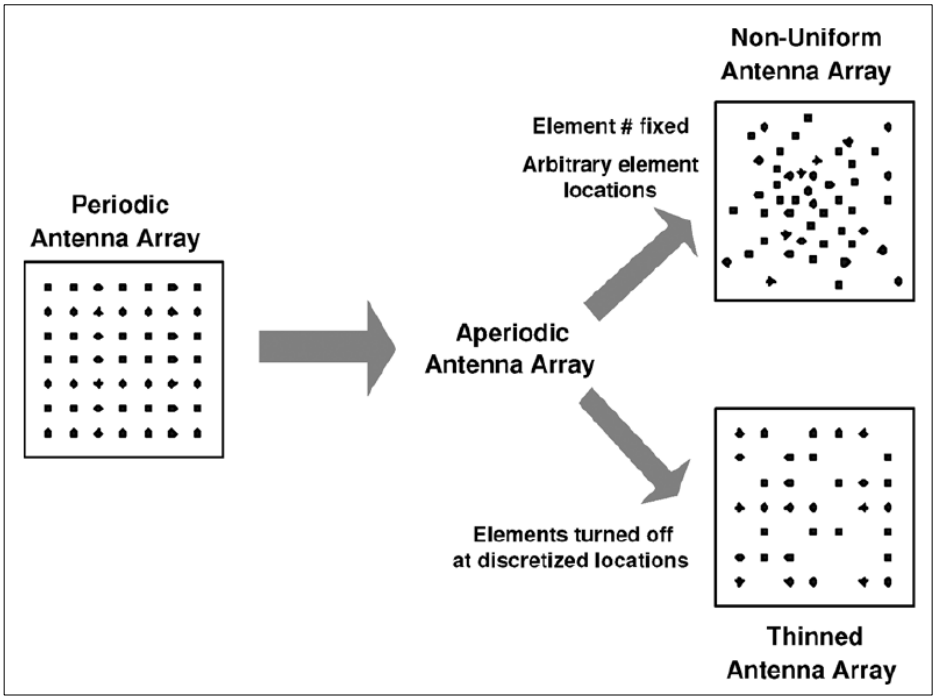

Figure 4.5 Non-uniform antenna array and a thinned antenna array [22].

In particular, to obtain the peak sidelobe level (SLL) in non-uniform arrays, the element position should be optimized in terms of a real number vector, as shown in Figure (4.6). Moreover, to avoid grating lobes, a certain element spacing limit exists for conventional periodic arrays (see constraints in Equation (4.1)). Interested readers can review [63] for more details on this problem.

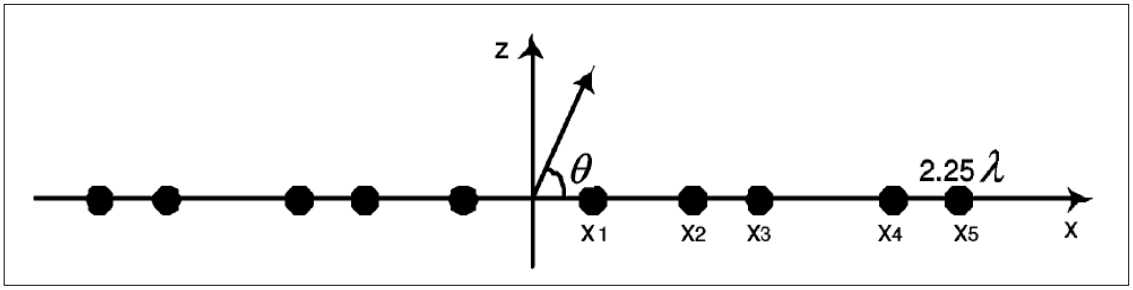

Figure 4.6 Array configurations for 10-elements [63].

Again, as shown in Figure (4.6), there are 10 elements of a non-uniform isotropic array, and only four element locations need to be optimized on each side. Since the outermost element is fixed at location $2.25\lambda_0$ with an average element spacing of $d_{avg} = 0.5\lambda_0$, this is a four-dimensional optimization problem with the following constraints:

$$x_i \in (0, 2.25)\left|x_i - x_j\right| > 0.25\lambda_0 \qquad (4.1)$$

$$min\{x_i\} > 0.125\lambda_0.\ i = 1,2,3,4.\ i \neq j.$$

The constraints show that there is a boundary between 0 and 2.25 for every element. However, each element cannot be smaller than $0.125\lambda_0$ or larger than $2.0\lambda_0$; that is because of $2.25\lambda_0$ is a fixed element and two adjacent elements cannot get closer than $0.25\lambda_0$. The fitness function problem is described as:

$$f = \ max\{20\log|AF(\theta)|\} \qquad (4.2)$$

where

$$AF(\theta) = \sum_{i=1}^{4} \cos[2\pi x_i(\cos\theta - \cos\theta_s)] + \cos[2.25 \times 2\pi(\cos\theta - \cos\theta_s)] \qquad (4.3)$$

Consider that $\theta_s = 90^{\circ}$ in this work is defined in Figure (4.6) [63].

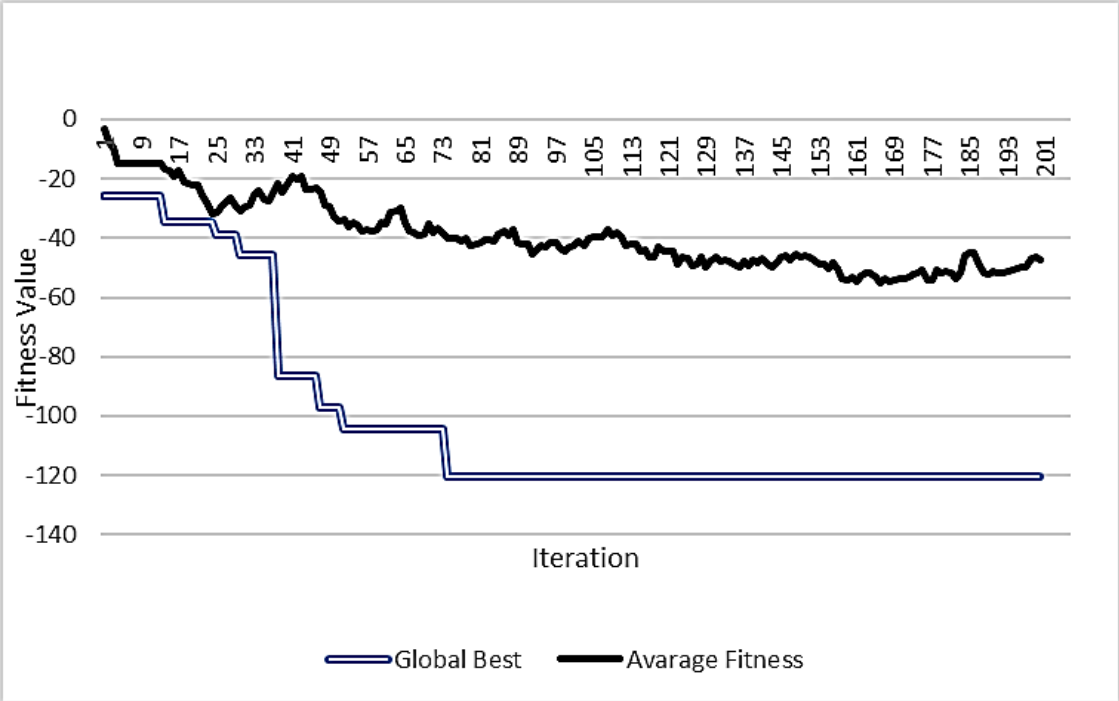

Figure 4.7 Global best with average fitness results for 200 Iteration with 20 artificial scout bees on aperiodic antenna array designs.

The DFO algorithm is used to optimize this problem, considering the constraints mentioned in Equation (4.1). Twenty artificial scout search agents are used for 200 iterations, and the presented result in Figure (4.7) includes the global best fitness in each iteration and the average fitness value according to Equation (4.2 and 4.3). The result shows that the global best solution reached its optimum solution in iteration 78 with element positions $=\{0.713, 1.595, 0.433, 0.130\}$.

B. SOFDO Usage on Frequency Modulated Sound Waves

SOFDO is used on frequency-modulated sound waves (FM) to optimize the parameter of an FM synthesizer, which has an essential role in several modern music systems; this problem has six parameters to be optimized as indicated in Equation (4.4).

$$X = \{a_1, w_1, a_2, w_2, a_3, w_3\} \quad (4.4)$$

The objective of this problem is to generate a sound, as in Equation (4.5), that is similar to the target sound, as in Equation (4.6).

$$y(t) = a_1.\sin(w_1.t.+a_2.\sin(w_2.t.\theta + a_3.\sin(w_3.\mathrm{t}.\theta))) \quad (4.5)$$

$$y_o(t) = (1.0).\sin((5.0).t.+(1.5).\sin((4.8).t.\theta + (2.0).\sin((4.9).t.\theta))) \quad (4.6)$$

$$f(\vec{x}) = \sum_{t=0}^{100}(y(t) - y_o(t))^2 \quad (4.7)$$

where the parameters should be in the range [−6.4, 6.35] and $\theta = 2\pi/100$, the fitness function can be calculated using Equation (4.7), which is simply the summation of the square root between the result of Equation (4.5) and Equation (4.6), while t = 100 turns. Interested readers can find more details on this problem in [64].

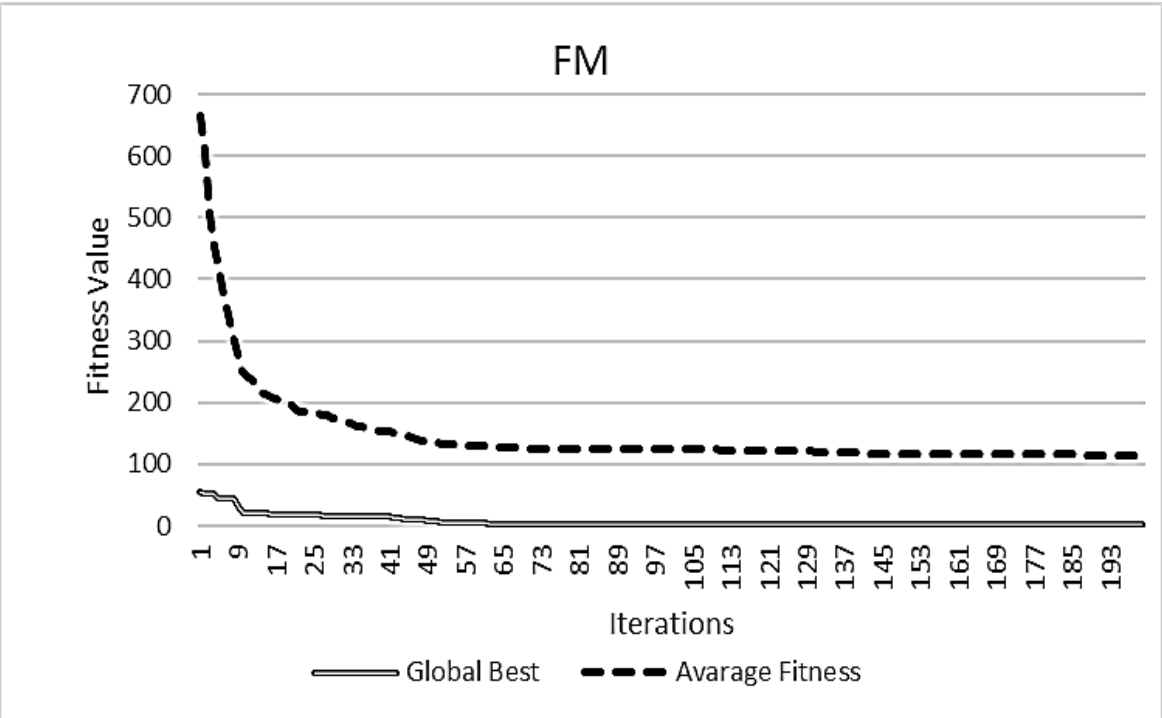

Figure 4.8 Global best with average fitness results for 200 Iteration with 30 artificial scout bees on the FM synthesis problem.

SOFDO is applied to the problem with 30 agents for 200 iterations, and records of the global best solutions and average fineness values can be seen in Figure (4.8). Parameter-set $X = \{a_1$ =0.974, w_1 = -0.241, a_2 = -4.3160, w_2 =

-0.0193, a_3 = -0.5701, w_3 = 4.937} were also generated at iteration 200. The global best value converges to the near-global optimal value from iteration 64.

4.2. Results and Discussions of Multi-Objective Fitness Dependent Optimizer

For testing MOFDO algorithm performance, two different types of multi-objective test functions were selected: Classical ZDT benchmarks [65] and 2019 CEC Multi modal multi-objective benchmarks [66]. The MOFDO results compared to the results of the latest state of the art of MOPSO, NSGA-III, and modern multi-objective dragonfly algorithm (MODA) [36].

4.2.1. Classical ZDT Benchmark Results.

This benchmark includes five well-known challenging test functions from ZDT1 to ZDT5 respectively, their mathematical definition is presented in Appendix Table (7.5), the MOFDO results compared to three well-known algorithms: MOPSO, MODA, and NSGA-III. The results are shown in Table (4.5), each algorithm is allowed to run for 500 iterations, each equipped with initial 100 search individuals and archive size of 100, the parameter settings for each algorithm are as described in their original papers [16] [66] [26]. However, the parameter settings for MOFDO are:

Polynomial Mutation Rate = 5.

Number of Grids per Dimension = 7.

Best Bee Selection Factor = 2.

Delete Factor =2.

Inflation Rate =1.

The inverse generational distance (IGD) as shown in Equation (4.8) is used for calculating quantitative results as described by [67].

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad (4.8)$$

Where d_i is the Euclidean distance between the closest obtained Pareto optimal solutions and the i^{th} true Pareto optimal solution in the reference set, and n the number of true Pareto optimal solutions. The IGD results of 30 independent runs are collected for each algorithm, then the average (mean), standard deviation STD, best, and worst IGD are calculated, see Table (4.5).

Table 4.5 Classical ZDT Benchmark results

Functions	Algorithms	IGD AVG.	IGD STD.	IGD Best	IGD Worst
ZDT 1	MOFDO	0.06758	0.030911	0.0018	2.61533
	MODA	0.07653	0.012071	0.0420	0.59398
	MOPSO	0.07843	0.008848	0.0446	1.14508
	NSGA-III	0.52599	0.509184	0.0134	3.69236
ZDT 2	MOFDO	0.03511	0.00404	0.0207	0.0515
	MODA	0.00292	0.00026	0.0002	0.0116
	MOPSO	0.03243	0.00093	0.0212	0.0682
	NSGA-III	0.13972	0.02626	0.1148	0.1834
ZDT 3	MOFDO	0.06676	0.023913	0.0014	2.2206
	MODA	0.07653	0.014411	0.0401	0.8267
	MOPSO	0.07758	0.005755	0.0427	1.0355
	NSGA-III	0.19474	0.080043	0.1935	0.1962
ZDT4	MOFDO	0.68020	0.352945	0.2679	1.6776
	MODA	64.9628	2.847807	51.742	500.93
	MOPSO	0.46175	0.047785	0.2515	5.1602
	NSGA-III	0.73731	0.307518	0.7360	0.7387
ZDT5	MOFDO	0.35853	0.161795	0.1221	1.9125

	MODA	0.11349	0.018270	0.0142	2.3938
	MOPSO	0.26862	0.136598	0.0468	2.1894
	NSGA-III	0.66397	0.235754	0.66273	0.66545

As can be seen in the ranking-table Table (4.6), MOFDO outperforms MOPSO and NSGA-III in most cases; however, it has a comparative result to MODA, as the MOFDO and MODA achieve the total ranking of 10, the detailed results for each algorithm can be found in appendix section Tables (7.29 to 7.33) respectively.

Table 4.6 The ranking table shows algorithms performances in Table (4.5)

Functions	**MOFDO Rank**	**MODA Rank**	**MOPSO Rank**	**NSGA-III Rank**
ZDT1	1	2	3	4
ZDT2	3	1	2	4
ZDT3	1	2	3	4
ZDT4	2	4	1	3
ZDT5	3	1	2	4
Total Ranking	10	10	11	19

Figure (4.9) shows the ZDT3 solution landscape as an example of MOFDO performance, Figure 4.9.1 shows how initially randomly distributed solutions which only 23 P_{fs} is located, then over the courses of iterations, MOFDO successfully increased the number of well-distributed P_f solutions as shown in Figures (4.9.2, 4.9.3 and 4.9.4).

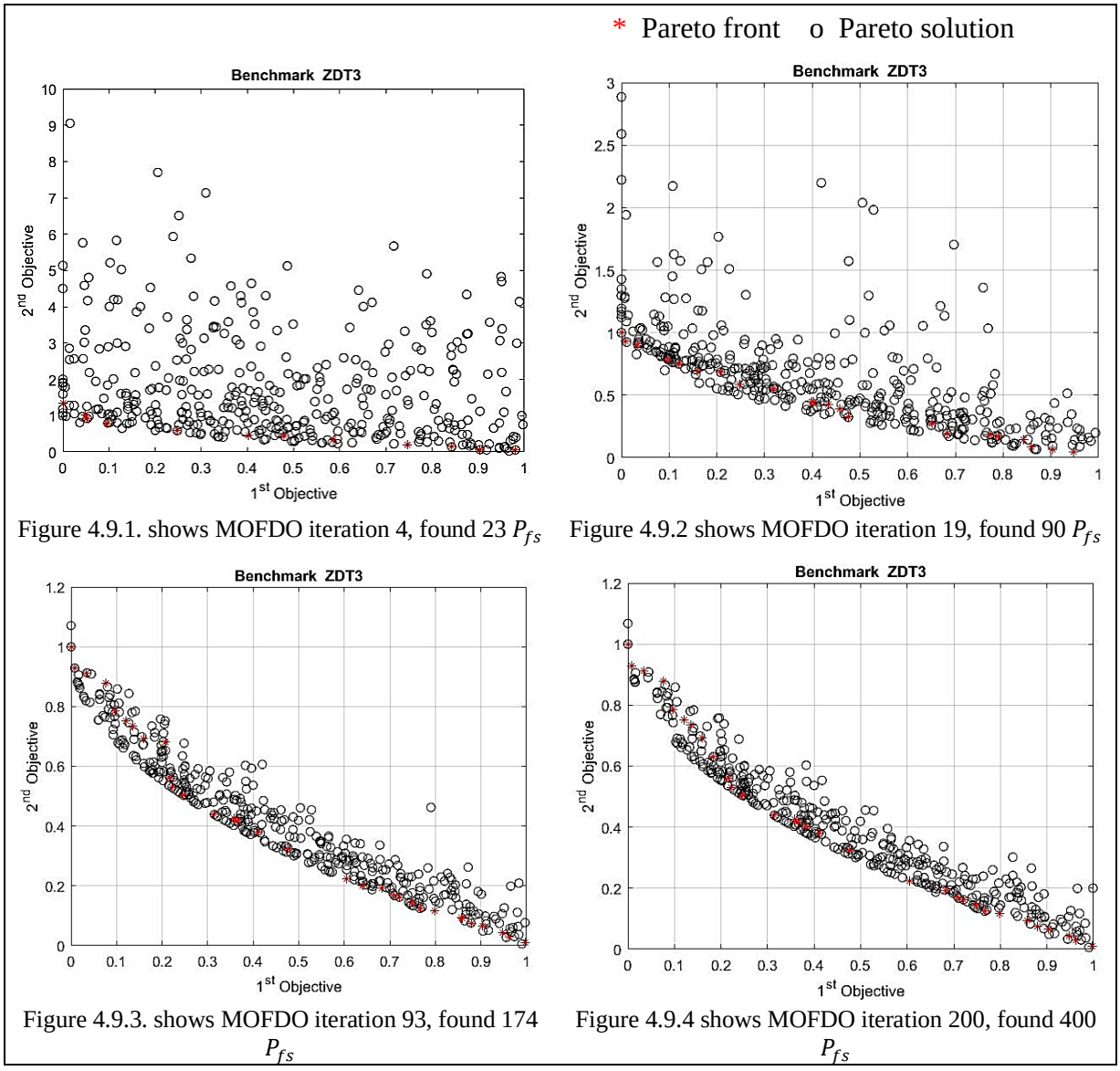

Figure 4.9.1. shows MOFDO iteration 4, found 23 P_{fs}

Figure 4.9.2 shows MOFDO iteration 19, found 90 P_{fs}

Figure 4.9.3. shows MOFDO iteration 93, found 174 P_{fs}

Figure 4.9.4 shows MOFDO iteration 200, found 400 P_{fs}

Figure 4.9 Shows how MOFDO solved the ZDT3 test function from initially random solution toward Pareto front optimality.

4.2.2. CEC 2019 Multi Modal Multi-Objective Benchmarks.

For more performance proof, a set of 12 CEC-2019 multi-modal multi-Objective (MMO) 2019 benchmarks are selected as described by [68], and their mathematical definition is shown in Appendix Table (7.6). The reason behind selecting this benchmark is because, these test functions are represented more difficult challenge than ZDT benchmark for MOFDO, as they have different characters, such as problems with different shape of PSs and PFs, with

coexistence of local and global PSs, also having scalable number of PSs, decision variables, and objectives.

MOFDO results compared to MOPSO, MODA and NSGA-III as shown in Table (4.7). The results are explained by the ranking system in Table (4.8), which shows the MOFDO ranked in first place with superior results in most cases; MODA comes in second place, then MOPSO and NSGA-III. The detailed results for each algorithm can be found in appendix section Tables (7.19 to 7.28) respectively.

Table 4.7 CEC 2019 MMF Benchmark results

Functions	**Algorithms**	**IGD AVG.**	**IGD STD.**	**IGD Best**	**IGD Worst**
MMF1	MOFDO	0.18401921	0.0454458	0.0882267	2.2406685
	MODA	0.87703300	0.5302916	0.3618665	9.6322659
	MOPSO	0.32173518	0.1108645	0.1419164	0.5425832
	NSGA-III	0.00351527	0.0005796	0.0022760	0.0049877
MMF2	MOFDO	0.09108902	0.0237087	0.0377645	0.9732300
	MODA	0.41152959	0.3041183	0.0883137	16.525584
	MOPSO	0.27264430	0.0606373	0.1906970	1.1093294
	NSGA-III	N/A	N/A	N/A	N/A
MMF3	MOFDO	0.09121177	0.0184429	0.0412612	1.0219979
	MODA	0.38999723	0.3195349	0.0720422	20.350064
	MOPSO	0.46594619	0.1566197	0.3377674	1.7063454
	NSGA-III	6.22408148	2.4773195	0.0523665	17.564674
MMF4	MOFDO	0.08195533	0.0340485	0.0453016	0.1879044
	MODA	0.00781723	0.0038766	0.0003086	0.0319178
	MOPSO	0.06806595	0.0056268	0.0437656	0.1621893
	NSGA-III	0.04784677	0.0102452	0.0054074	0.1314727
MMF5	MOFDO	0.08166825	0.0203041	0.0410373	0.2872334
	MODA	0.20697935	0.1068910	0.1177007	0.2665746
	MOPSO	0.36516458	0.0818221	0.1431733	0.6443415
	NSGA-III	0.23876644	0.1195411	0.2370972	0.2405229
MMF6	MOFDO	0.06319825	0.0052717	0.0435369	0.3023473
	MODA	6.20722767	7.0529532	3.8844812	18.481523
	MOPSO	0.57534658	0.2975283	0.2154710	1.0872217

	NSGA-III	0.70090140	0.2650748	0.6990628	0.7024793
MMF7	MOFDO	0.14853951	0.0219769	0.0870897	0.2362694
	MODA	0.36139133	0.1036987	0.1322042	1.0331912
	MOPSO	0.33104321	0.0493751	0.1186243	0.4153237
	NSGA-III	0.40264339	0.1635305	0.4014061	0.404124
MMF8	MOFDO	0.15550447	0.0869989	0.0343290	1.8046810
	MODA	0.08058735	0.2652865	0.0056401	10.189273
	MOPSO	0.14156195	0.1003047	0.0450518	2.6718013
	NSGA-III	0.01038634	0.0032172	0.0041821	0.0276770
MMF9	MOFDO	0.47321267	0.1219659	0.3404164	1.7427477
	MODA	0.05060995	0.0274896	0.0051946	0.3219936
	MOPSO	1.33275589	0.1427530	0.7792982	2.0541908
	NSGA-III	0.96369603	0.3011030	0.0027674	0.2438216
MMF10	MOFDO	0.44207841	0.1277887	0.3104489	1.1022388
	MODA	0.09017605	0.0385574	0.0039308	0.3893014
	MOPSO	1.00054897	0.1542964	0.7005662	1.4956293
	NSGA-III	3.89641261	4.6634273	0.0028740	4.4067242
MMF11	MOFDO	0.09260275	0.0209854	0.0635536	0.2692325
	MODA	0.09291338	0.0515551	0.0042148	0.4962967
	MOPSO	1.30789085	0.1622864	0.6847497	2.2372910
	NSGA-III	1.18058557	0.7034533	0.0034136	4.2312541
MMF12	MOFDO	0.08314653	0.0217281	0.0554114	0.2901663
	MODA	0.03661122	0.0119014	0.0035018	0.1841556
	MOPSO	0.13651933	0.0237385	0.0667090	0.3416214
	NSGA-III	0.35064339	0.1613096	0.3494061	0.352124

Table 4.8 The ranking table shows algorithms performances in table (4.7)

Functions	**MOFDO RANK**	**MODA RANK**	**MOPSO RANK**	**NSGA-III RANK**
MMF1	2	4	3	1
MMF2	1	3	2	4
MMF3	1	2	3	4
MMF4	4	1	2	3
MMF5	1	2	3	4
MMF6	1	4	3	2

MMF7	1	2	3	4
MMF8	4	2	3	1
MMF9	2	1	4	3
MMF10	2	1	3	4
MMF11	1	2	4	3
MMF12	2	1	3	4
TOTAL RANKING	22	25	36	37

Again, as a performance prove, Figure (4.10) shows MMF4 benchmark solution landscape as an example, Figure 4.10.1 shows the initial random distribution of only 31 P_{fs} solutions, then over the courses of iterations, the number of well-distributed P_f solutions are increased in Figures (4.10.2, 4.10.3 and 4.10.4).

In order to prove whether the results of Table (4.5) and (4.7) are statistically significant or not, the Wilcoxon rank-sum test has been conducted for finding the p-value between the MOFDO and other algorithms. As presented in Table (4.9), the high majority of the results in Table (4.5) (ZDT benchmarks results) are statistically significant, since the p-value is smaller than 0.05.

Nonetheless, Table (4.10) also shows the level of significance of the results in Table (4.7) (CEC 2019 benchmarks results), as clearly it can be seen, that the results are statistically significant in almost all cases, except three cases in MMF8 and MMF11.

Table 4.9 The Wilcoxon rank-sum test (p-value) for ZDT benchmarks

Functions	**MOFDO Vs. MOPSO**	**MOFDO Vs. MODA**	**MOFDO Vs. NSGA-III**
ZDT1	0.069585315	0.144884804	1.06115E-05
ZDT2	0.000828462	5.73525E-46	0.002956318
ZDT3	0.01911876	0.06013187	1.35925E-11

ZDT4	0.001385852	9.07401E-72	0.506664868
ZDT5	0.023547289	2.40076E-11	2.39778E-07

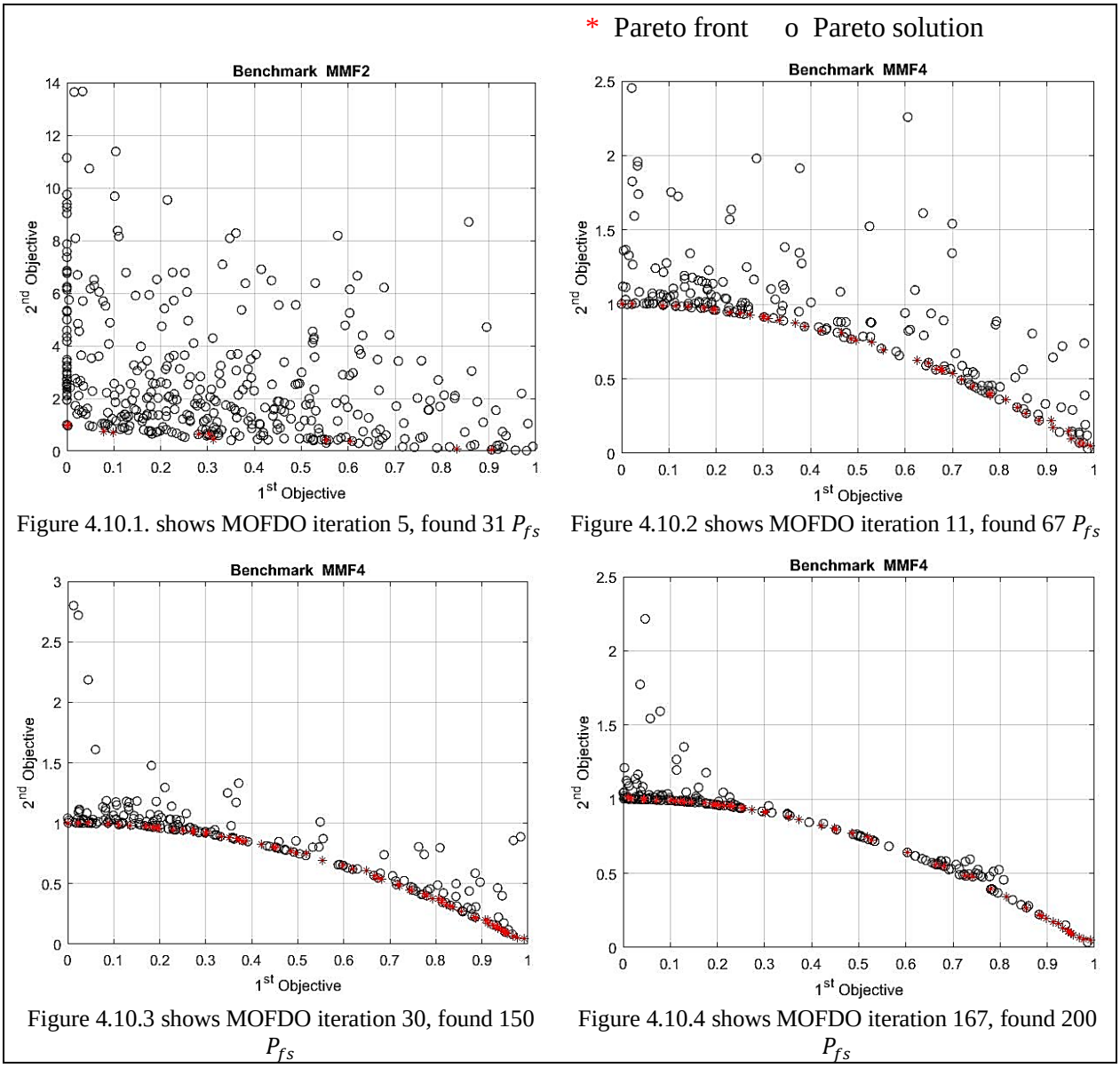

Figure 4.10.1. shows MOFDO iteration 5, found 31 P_{fs}

Figure 4.10.2 shows MOFDO iteration 11, found 67 P_{fs}

Figure 4.10.3 shows MOFDO iteration 30, found 150 P_{fs}

Figure 4.10.4 shows MOFDO iteration 167, found 200 P_{fs}

Figure 4.10 Shows how MOFDO solved the MMF4 test function from initially random solution toward Pareto front optimality.

Table 4.10 The Wilcoxon rank-sum test (p-value) for CEC 2019 benchmarks

Functions	**MOFDO Vs. MOPSO**	**MOFDO Vs. MODA**	**MOFDO Vs. NSGA-III**
MMF1	4.41881E-08	1.76049E-09	1.41173E-29

MMF2	5.48551E-22	3.45868E-07	N/A
MMF3	7.46013E-19	3.72574E-06	1.23675E-19
MMF4	0.031476101	3.98328E-17	2.22201E-06
MMF5	6.57481E-26	4.20625E-08	2.01881E-09
MMF6	2.64131E-13	1.27723E-05	4.39284E-19
MMF7	5.34254E-26	8.11656E-16	1.14766E-11
MMF8	0.567417814	0.147031583	3.58707E-12
MMF9	8.12705E-33	5.09201E-26	5.83029E-07
MMF10	5.57562E-22	7.29596E-21	5.74382E-06
MMF11	2.44568E-44	0.975720942	1.36419E-10
MMF12	9.63988E-13	1.07115E-14	1.31857E-12

4.2.3. Real-World Application

MOFDO is implemented in MATLAB codes; the algorithm is well structured, which allows for easy modifications and integration. It also enables researchers to easily understand how it works and how real-world applications can be solved with less effort. To demonstrate this, we optimized the welded beam design problem with MOFDO, the problem definition and the results are discussed below.

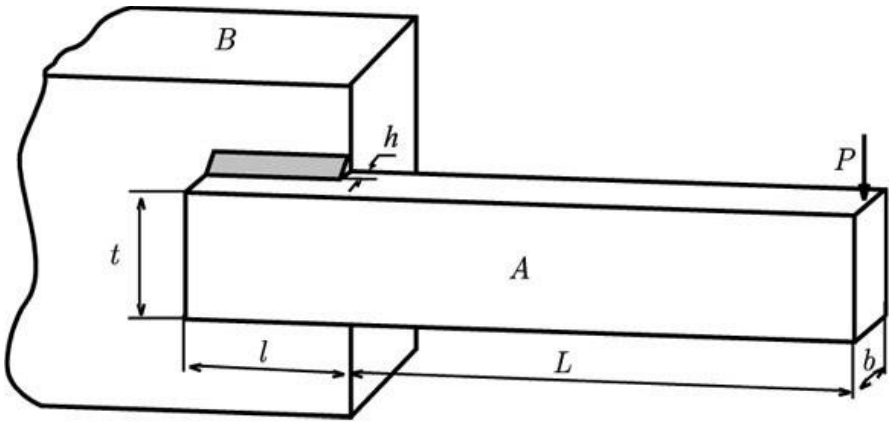

Figure 4.11 Welded beam design problem [69]

The welded beam design problem is a very well-known real-world engineering design problem, it has been considered by other researchers previously as a test problem for various multi-objective algorithms, such as [70]. As shown in Figure (4.11), this problem has four real-parameter variables *x = (h, l, t, b)*, where *h* is the thickness of the welds, *l* is the length of the welds, *t* is the height of the beam, and *b* is the width of the beam. This design problem has bi-objective to be minimized Equation (4.9), knowing that the objectives are conflicting in nature, the first objective is to minimize the cost of fabrication and the second objective is to minimize the end deflection of the welded beam as follows:

$$
\begin{aligned}
&Minimize\ \ f1(\vec{x}) = 1.10471h^2l + 0.04811tb(14.0 + l),\\
&Minimize\ f2(\vec{x}) = \frac{2.1952}{t^3b},\\
&Subject\ to:\\
&g1(\vec{x}) \equiv 13{,}600 - T(\vec{x}) \geq 0,\\
&g2(\vec{x}) \equiv 13{,}600 - \sigma(\vec{x}) \geq 0,\\
&g3(\vec{x}) \equiv b - h \geq 0,\\
&g4(\vec{x}) \equiv Pc(\vec{x}) - 6000 \geq 0,\\
&0.125 \leq h, \qquad b \leq 5.0\\
&0.1 \leq l, \qquad t \leq 10.0
\end{aligned}
\tag{4.9}
$$

As presented in Equation (4.9), this problem has four constraints to be considered, a violation of any of these constraints will make the design unacceptable.

1. The first constraint is to make sure that the shear stress produced at the support location of the beam is less than the acceptable shear strength of the material, which is equal to 13,600 psi.

2. The second constraint is to guarantee that standard stress produced at the support location of the beam is less than the acceptable yield strength of the material, which is equal to 30,000 psi.
3. The third constraint is to ensure that the breadth of the beam is not less than the weld width from a practical perspective.
4. The constraint number four is to ensure that the acceptable buckling load of the beam is larger than the applied load F = 6; 000 lbs. The shear stress $T(\vec{x})$ and the buckling load $\sigma(\vec{x})$ can be calculated as Equation (4.10 and 4.11) respectively:

$$T(\vec{x}) = \sqrt{(T')^2 + (T'')^2 + (lT'T'')/\sqrt{0.25(l^2 + (h+t)^2)}},$$

$$T' = \frac{6000}{\sqrt{2}\,hl} \qquad (4.10)$$

$$T'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\left\{0.707hl\left(\frac{l^2}{12} + 0.25\,(h+t)^2\right\}\right.}$$

$$\sigma(\vec{x}) = \frac{504000}{t^2 b}, \qquad (4.11)$$

$$Pc(\vec{x}) = 64746.022(1 - 0.0282346t)\,tb^3$$

The welded beam design problem is optimized using MOFDO, the algorithm is applied to solve this engineering design problem for 100 iterations, using 100 search agents, the P_{fs} is stored in 100 sized archives. Figure (4.12) shows that the obtained *PFs* are smoothly distributed between these two objectives (Cost and Deflection), also MOFDO provides a wide variety of feasible solutions for the decision-makers to choose among them, this wide variety and smooth distribution of the obtained PFs are prove the maturity of the MOFDO in terms

of the algorithm capability of tackling the real-world engineering design problem effectively.

Regarding MOFDO's P_{fs} discovering rate, MOFDO starts from 17 P_{fs} for the first iteration, and then dramatically reaches $100P_{fs}$ in iteration 36 as presented in Figure (4.13). This shows how the algorithm efficiently improves multiple initial solutions toward the optimality, then occasionally some P_{fs} becomes nondominated solution, so that, they are deleted from archive, this can be seen from iteration 36 to 100 in Figure (4.13), this shows that MOFDO is constantly improving all solutions (dominated and non-dominated) throughout all iterations, this feature guarantees that MOFDO avoids local solution and eventually reaches optimality [71].

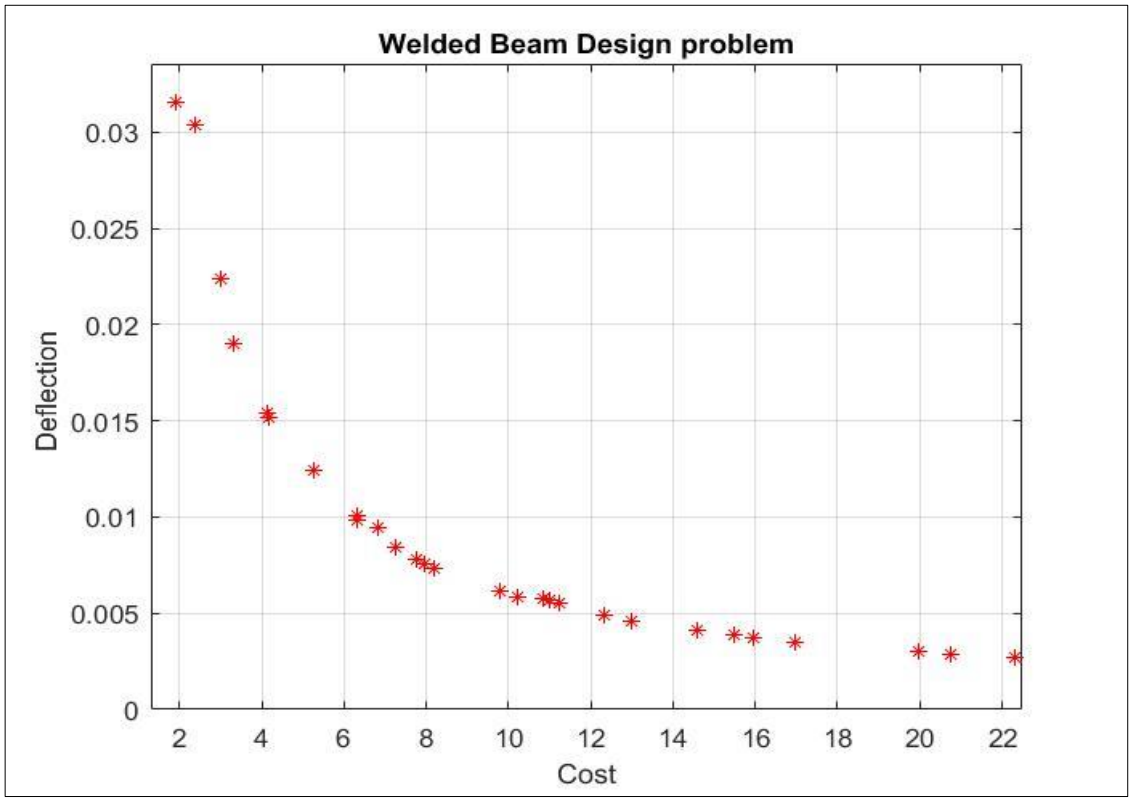

Figure 4.12 MOFDO results on Welded Beam Design problems

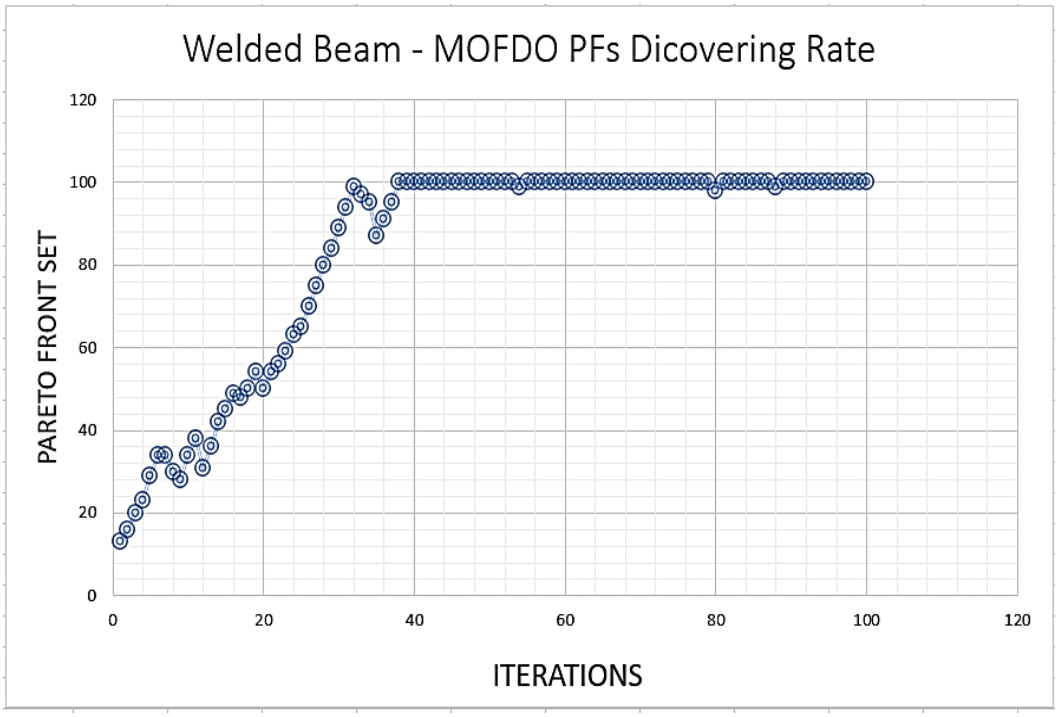

Figure 4.13. Shows MOFDO Pfs discovering rate

Chapter Five

Conclusions

Chapter Five

5. Conclusions

A new swarm intelligent algorithm was proposed called the fitness dependent optimizer; it is inspired by the bee reproductive swarming process, where scout bees search for a new nest site. Additionally, the algorithm is inspired by their collective decision-making. It has no algorithmic connection with the ABC algorithm. SOFDO employs fitness function values to generate weights that drive the search agents towards optimality. Additionally, SOFDO depends on the randomization mechanism in the initialization, exploration and exploitation phases. A group of 19 single-objective benchmark testing functions was used to test the performance of the SOFDO. The benchmark testing functions were divided into three subgroups (unimodal, multimodal and composite test functions). Additionally, SOFDO tested on 10 modern CEC-C06 benchmarks. The SOFDO results compared to two well-known algorithms (PSO and GA) and three modern algorithms (DA, WOA, and SSA), SOFDO outperformed the competing algorithms in the majority (20/30) of cases as shown in Tables(4.1 and 4.2) and produced a comparative result on the others. The test results were compared using the Wilcoxon rank-sum test to prove their statistical significance. Four additional experiments were conducted on the SOFDO algorithm to measure, prove and verify the performance and credibility. Furthermore, SOFDO was practically applied to two real-world examples as evidence that the algorithm can address real-life applications.

Generally, we found that the number of search agents was related somehow to SOFDO performance after testing on many standard test functions and real-world applications. Thus, using a small number of agents (below five) would notably decrease the accuracy of the algorithm, and a large number of search agents would improve the accuracy and cost more time and space, partially because the algorithm depends on the fitness weight on the significant part of

its searching mechanism; in view of this, it is known as the fitness dependent optimizer.

A multi-objective approach for a novel SOFDO is proposed, known as a multi-objective fitness dependent optimizer, which is inspired by the bee reproductive swarming process. During the implementation, MOFDO is being treated as a typical cultural algorithm, for this purpose: situational, normative, topographical, domain, and historical knowledge were employed. MOFDO is tested on two different sets of test functions, the first set is ZTD test functions, which are a classical benchmark used by many other researchers for testing MOEAs. The second set is a modern CEC 2019 multimodal multi-objective, which is considered to be a more challenging benchmark. The MOFDO results were compared to three other algorithms: the latest state of the art of MOPSO, NSGA-III, and MODA. The comparison showed that MOFDO outperformed other algorithms in most cases, and provided comparative results in other cases. Table (4.8) shows that MOFDO out performed MOPSO and NSGA-III, also Table (4.8) shows that MOFDO out performed MODA, MOPSO and NSGA-III in (6/10) cases. MOFDO was easily used for solving real-world problems, as an example, the welded beam design problem was solved using MOFDO, and it provided well-distributed solutions, which possibly enables decision-makers to have more variant options to consider.

Regarding CEC 2019 benchmark complexity, it represents a real challenge for MOEAs comparing to the classical ZDT benchmark, because it contains both local and global Pareto front. The algorithm must try to avoid trapping in the local Pareto front. Although, arguably in some applications, the local optima are more preferable, as global optima solutions may not always be applicable in real-world problems [68]. Moreover, having different decision variable space boundary for each dimension make the CEC 2019 MMF even more difficult. In fact, the NSGA-III couldn’t produce a correct result for

MMF2 test function as presented in Table (2), also it has difficulties in some other test functions. On the other hand, MOFDO constructed in a way that makes it easy to deal with these difficulties.

Future works will adopt, implement and test both multi-objective and binary objective optimization problems on SOFDO. Finally, integrating evolutionary operators into SOFDO and hybridizing it with other algorithms can be considered as potential future research. Moreover, researchers might try to improve algorithm performance by adapting new parameters, which might lead to enhancing the learning rate and communication range between individuals, or possibly integrating or hybridizing MOFOD with other MOEAs.

6. Publications

The first proposed algorithm SOFDO has already published in the journal of IEEE Access entitled “Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process”, which has an impact factor of 4.098, also indexed by Clarivate Analytics, Scopus and others. The second proposed algorithm MOFDO is under consideration for reviewing purpose.

7. References

[1] Copeland, B. J., Alan Turing's Automatic Computing Engine: The Master Codebreaker's Struggle to build the Modern Computer, London: Oxford University Press, (2005).

[2] Yang, X. S., Nature-Inspired Metaheuristic Algorithms, United Kigdom: Luniver Press, (2010).

[3] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, Dušan Fister, "A Brief Review of Nature-Inspired Algorithms for Optimization," *ELEKTROTEHNISKI VESTNIK,* vol. 80, no. 3, pp. 1-7, (2013).

[4] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J., "A survey on metaheuristics for stochastic combinatorial optimization," *Springer Netherlands,* vol. 8, no. 1, p. 239–287, 2008.

[5] Wolpert, D. H., Macready, W. G., "No free lunch theorems for optimization," *IEEE Transaction on Evolutionary Computation,* vol. 1, no. 1, pp. 67-82, 1997.

[6] Melanie, M., An Introduction to Genetic Algorithms, London, England: The MIT Press, 1999.

[7] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. , "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680, 1983.

[8] Kennedy, J., Eberhart, R., "Particle swarm optimization," in *IEEE International Conference*, 1995.

[9] Storn, R.; Price, K., "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization,* vol. 11, p. 341–359, 1997.

[10] Zong Woo Geem, Joong Hoon Kim, G.V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *SIMULATION,* vol. 76, no. 2, pp. 60-68, 2001.

[11] Nakrani, S., and Tovey, C., "On honey bees and dynamic server allocation in Internet hostubg centers,," *Adaptive Behavior,,* vol. 12, pp. 223-240, 2044.

[12] Pham, D & Ghanbarzadeh, Afshin & Koç, Ebubekir & Otri, Sameh & Rahim, Sahra & Zaidi, Mb., "The Bees Algorithm Technical Note," Cardiff University, Cardiff , 2005.

[13] Yang, Xin-She & Xingshi, He, “Firefly Algorithm: Recent Advances and Applications,” *International Journal of Swarm Intelligence,* vol. 1, no. 1, pp. 36--50, 2013.

[14] Yang, X. S., and Deb S., “Cuckoo search via L¶levy °ights,,” in *World Congress on Nature & Biologically Inspired Computing IEEE Publlication*, USA, 2009.

[15] Yang, X. S.,, “A new metaheuristic bat-inspired algorithm,,” in *Nature Inspierd Cooperative Strategies for Optimization (NICSO 2010), Springer, SCI 284, 65-74*, 2010.

[16] Mirjalili, Seyedali, “Dragonfly algorithm: a new meta-heuristic optimization technique,” *Neural Comput & Applic,* vol. 27, no. 4, p. 1053–1073, 2015.

[17] Seyedali Mirjaliliab, Andrew Lewisa, “The Whale Optimization Algorithm,” vol. 95, pp. 51-67, 2016.

[18] Seyedali Mirjalilia, Amir H.Gandomibf, Seyedeh Zahra Mirjalili, cShahrzad Saremia, Hossam Farisd, Seyed Mohammad Mirjalilie, “Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems,” *Advances in Engineering Software,* vol. 114, pp. 163-191, 2017.

[19] Laizhong Cui, Genghui Li, Zexuan Zhu, Qiuzhen Lin, Zhenkun Wen, Nan Lu, Ka-Chun Wong, Jianyong Chen, “A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization,” *Information Sciences,* 2017.

[20] Laizhong Cui , Genghui Li, Xizhao Wang , Qiuzhen Lin , Jianyong Chen , Nan Lu , Jian Lu , “A ranking-based adaptive artificial bee colony algorithm for global numerical optimization,” *Information Sciences,* vol. 417, p. 169–185, 2017.

[21] Laizhong Cui, Kai Zhang, Genghui Li, Xizhao Wang, Shu Yang, Zhong Ming, Joshua Zhexue Huang, Nan Lu, “A smart artificial bee colony algorithm with distance-fitness-based neighbor search and its application.,” *Future Generation Computer Systems,* vol. 89, pp. 478-493 , 2018.

[22] Laizhong Cui; Genghui Li; Yanli Luo; Fei Chen; Zhong Ming; Nan Lu; Jian Lu, “An enhanced artificial bee colony algorithm with dual-population framework,” *Swarm and Evolutionary Computation,* vol. 43, pp. 184-206 , 2018.

[23] Enrique M. Cortés-Toro, Broderick Crawford, Juan A. Gómez-Pulido , Ricardo Soto, “A New Metaheuristic Inspired by the Vapour-Liquid,” *Applied Sciences,* vol. 11, October 2018.

[24] Deb, K., Pratap, A., Agarwal S., & Meyarivan T., “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE transactions on evolutionary computation,* pp. 182-197, 2002.

[25] Liu, T., Jiao, L., Ma, W., Ma, J., & Shang, R., "A new quantum-Behaved particle swarm optimization based on cultural evolution mechanism for multiobjective problems," *Appl. Soft Comput,* vol. 46, p. 267–283., 2016.

[26] Deb, K., & Jain, H. , "An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, PartI: Solving problems with box constraints," *IEEE Transaction on Evolutionary Computation,* vol. 18, no. 4, pp. 577-601, 2014.

[27] Coello, C. A. C., Pulido, G. T., & Lechuga, M. s., "Handlling multiple objectives with particle swarm optimization," *Transaction on evolutionary computation,* pp. 256-279, 2004.

[28] Zitzler, E., Laumanns, M., & Thiele, L., "SPEA2: Improving the Strength ParetoEvolutionary Algorithm," Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland, 2001.

[29] Zhang, Q., & Li, H., "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation,* vol. 11, no. 6, pp. 712 - 731, 2011.

[30] Jain, H., & Deb, K. , "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part II: Handling constraints and extending to an adaptive approach," *IEEE Transaction on Evolutionary Computation,* pp. 602-622, 2014.

[31] Pradhan, P. M., & Panda, G. , "Solving multi-objective problems using cat swarm optimization," *Expert Systems with Applications,* pp. 2956-2964, 2012.

[32] Nanda, S. J., & Panda, G. , "Automatic clustering using MOCLONAL for classifying actions of 3D human models," *IEEE Humanities, Symposium on Science and Engineering Research,* pp. 945-950, 2012.

[33] Vikas & Nanda S. J. , "Multi-objective Moth Flame Optimization," *IEEE Int. Conf. on In Advances Comm. and Informatics in Computing (ICACCI),* pp. 2470-2476, 2016.

[34] Mirjalili A. S., Jangir, P., & Saremi, S. , "Multi-objective ant lion optimizer:a multi-objective optimization algorithm for solving engineering problems," *Applied Intelligence,* pp. 79-95, 2017.

[35] Mirjalili S., Saremi, S., Mirjalili, S. M., & Leandro, D. S. C. , “Multiobjective grey wolf optimizer: a novel algorithm for multi-criterion optimization,” *Expert Systems with Applications,* pp. 106-119, 2016.

[36] Mirjalili, S., “Dragonfly algorithm: a new meta-heuristic optimization technique,” *Neural Comput & Applic,* vol. 27, no. 4, p. 1053–1073, 2015.

[37] Kumawat, I. R., Nanda, S. J., & Maddila, R. K., “Multi-objective whale optimization,” in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Penang, 2017.

[38] Lei, D., “Multi-objective production scheduling: a survey,” *The Int. Jou. Adv. Manufacturing Tech.,* pp. 926-938, 2009.

[39] Panda A., & Pani, S. , “A Symbiotic Organisms Search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems,” *Applied Soft Computing,* vol. 46, pp. 344-360, 2016.

[40] Ng, D. W., Lo, E. S., & Schober, R. , “Multiobjective resource allocation for secure communication in cognitive radio networks with wireless information and power transfer,” *IEEE Trans. on Vehicular Technology,* pp. 3166-3184, 2016.

[41] Raquel, C.R., & Naval, P.C., “An effective use of crowding distance in multiobjective particle swarm optimization,” in *the Conference on Genetic and Evolutionary Computation*, Washington, DC, USA, 2005.

[42] Mostaghim, S., & Teich, J., “Strategies for finding good local guides in multi-Objective particle swarm optimization (MOPSO),” in *IEEE Swarm Intelligence Symposium (SIS 2003)*, Indianapolis, IN, USA, 2003.

[43] Pulido, G.T., & Coello, C. .A. C., “Using clustering techniques to improve the performance of a particle swarm optimizer,” in *the Genetic and Evolutionary Computation Conference (GECCO)*, Seattle, WA, USA, 2004.

[44] Zitzler, E., Deb, K., & Thiele, L., “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,” *Evolutionary Computation,* vol. 8, no. 2, pp. 173-195, 2000.

[45] Laumanns, M., Thiele, L., Deb, K., & Zitzler, E., “Combining Convergence and Diversity in Evolutionary Multiobjective Optimization,” *Evolutionary Computation,* vol. 10, no. 3, pp. 263-282, 2002.

[46] Tang L., & Wang X., "A Hybrid Multiobjective Evolutionary Algorithm for Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation,* vol. 17, no. 1, pp. 20-45, 2013.

[47] Yu, G., "Multi-Objective estimation of Estimation of Distribution Algorithm based on the Simulated binary crossover," *Convergence Information Technology ,* vol. 7, no. 13, p. 110–116, 2012.

[48] Hu, W., Yen, G.G., & Zhang, X., "Multiobjective particle swarm optimization based on Pareto entropy," *Software,* vol. 25, no. 5, pp. 1025-1050, 2014.

[49] Peng, B., Reynolds, R. G., "Cultural algorithms: Knowledge learning in dynamic environments," in *the 2004 Congress on Evolutionary Computation*, Portland, OR, USA, 2004.

[50] Coello, C. A. C., "Evolutionary multiobjective optimization a historical view of the field," *IEEE Computational Intelligence Magazine,* vol. 1, no. 1, p. 28–36, 2006.

[51] Coello, C. A. C., "Evolutionary multi-objective optimization:some current research trends and topics that remain to be explored," *Front Comput Sci China,* vol. 3, p. 18–30, 2009.

[52] Villa, J. D., "Swarming Behavior of Honey Bees (Hymenoptera: Apidae) in Southeastern Louisiana," *Annals of the Entomological Society of America,* vol. 1, no. 97, p. 111–116, 2004.

[53] AVITABILE, A., MORSE, R. A., BOCH R., "Swarming Honey Bees Guided by Pheromones," *Annals of the Entomological Society of America,* vol. 68, no. 6, p. 1079–1082, 1975.

[54] Schultz, K. M., Passino, K. M. and Seeley, T. D, "The mechanism of flight guidance in honeybee swarms: subtle guides or streaker bees?," *The Journal of Experimental Biology,* vol. 211, pp. 3287-3295, 2008.

[55] Blackiston, Howland, Beekeeping For Dummies, 2nd Edition, A.I. Root Company, 2009.

[56] Seeley, Thomas Dyer, Honeybee Democracy, USA, New jersey: Princeton University Press, 2010.

[57] Liagkouras, K., & Metaxiotis, K. , "An Elitist Polynomial Mutation Operator for Improved Performance of MOEAs in Computer Networks," in *Computer Communication and Networks (ICCCN)*, Nassau, 2013.

[58] Li, L. Lu, K. Zeng, G. Wu, L. Chen, M., "A novel real-coded population-based extremal optimization algorithm with polynomial mutation: A non-parametric statistical study on continuous optimization problems," *Neurocomputing,* vol. 174, pp. 577-587, 2016.

[59] Coello, C. A. C., & Lechuga, M. S., "MOPSO: A proposal for multiple objective particle swarm optimization," in *Evolutionary computation, 2002. CEC'02. Proceedings of the 2002 congresson,*, Honolulu, HI, USA., 2002.

[60] Price, K. V., Awad, N. H., Ali, M. Z., Suganthan, P. N., "The 100-Digit Challenge: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization," Nanyang Technological University,, Singapore, November 2018.

[61] MirJalili, "Ali MIrjalili," Seyedali Mirjalili, 2015. [Online]. Available: http://www.alimirjalili.com/Projects.html. [Accessed 01 01 2019].

[62] Frans van den Bergh and Andries Petrus Engelbrecht, "A study of particle swarm," *Inf. Sci.,* vol. 176, pp. 937-971, 2006.

[63] Nanbo Jin, Student Member, IEEE, and Yahya Rahmat-Samii, Fellow, IEEE, "Advances in Particle Swarm Optimization for Antenna Designs: Real-Number, Binary, Single-Objective and Multiobjective Implementations," *IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION,,* vol. 55, no. 3, pp. 560-562, 2007.

[64] Das, Swagatam and Suganthan, Ponnuthurai, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," ResearGate, 2018.

[65] Zitzler E., Deb K., & Thiele L., "Comparison of multiobjective," *Evol Comput,* vol. 8, p. 173–195, 2000.

[66] Yue, C., Qu, B., & Liang J., "A Multiobjective Particle Swarm Optimizer Using Ring Topology for Solving Multimodal Multiobjective Problems," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,,* vol. 22, no. 5, pp. 805-827, 2018.

[67] Sierra M. R., & Coello, C. A. C., "Improving PSO-based Multi-Objective Optimization using Crowding, Mutation and E-Dominance," CINVESTAV-IPN (Evolutionary Computation Group) Electrical Eng. Department, Computer Science Dept, Col. San Pedro Zacatenco, Mexico, 2005.

[68] Liang, J. J., Qu, B. Y., Gong, D. W., & Yue, C. T. , “Problem Definitions and Evaluation Criteria for the CEC Special Session on Multimodal Multiobjective Optimization,” Zhengzhou University,, Zhengzhou, China, 2019.

[69] Radovan R. Bulatović, G. B., Savković, M. M., & Gašić M. M. , “Improved Cuckoo Search (ICS) algorthm for constrained optimization problems,” *Latin American Journal of Solids and Structures,* vol. 11, pp. 1349-1362, 2014.

[70] Ray, T., & Liew K. M. , “A Swarm Metaphor for Multiobjective Design Optimization,” *Engineering Optimization,* vol. 34, no. 2, p. 141–153, 2002.

[71] van den Bergh F., & Engelbrecht A. P., “A study of particle swarm,” *Inf. Sci,* vol. 176, no. 8, pp. 937-971, 22 April 2006.

[72] Abdullah, J. M., & Rashid, T. A., “Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process,” *IEEE Access,* vol. 7, pp. 43473-43486, 2019.

[73] Kirkpatrick, L., Gelatt C. D., Vecchi, Jr. M. P., “Optimization by Simulated Annealing,” *Science,* vol. Vol. 220, no. 4598, pp. 671-680, 13 May 1983.

[74] Melanie, Mitchell, An Introduction to Genetic Algorithms, Cambridge, Massachusetts • London, England: The MIT Press, 1999.

[75] Mirjalili, “Udemy,” Udemy, 01 06 2018. [Online]. Available: https://www.udemy.com/course/optimisation/. [Accessed 01 08 2018].

8. Appendix

Table 7.1 Unimodal Benchmark Functions [16]

Dimension = 10 and $f_{min} = 0$		
Functions	**Range**	**Shift position**
$\boldsymbol{TF1(x)} = \sum_{i=1}^{n} x_i^2$	[-100, 100]	[-30, -30, … -30]
$\boldsymbol{TF2(x)} = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	[-3, -3, … -3]
$\boldsymbol{TF3(x)} = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	[-100, 100]	[-30, -30, … -30]
$\boldsymbol{TF4(x)} = \max_i \{\lvert x \rvert, 1 \le i \le n\}$	[-100, 100]	[-30, -30, … -30]
$\boldsymbol{TF5(x)} = \sum_{i=1}^{n-1} [100(x_{i+1} - x_1^2)^2 + (x_i - 1)^2]$	[-30, 30]	[-15, -15, … -15]
$\boldsymbol{TF6(x)} = \sum_{i=1}^{n} ([x_i + 0.5])^2$	[-100, 100]	[-750, … -750]
$\boldsymbol{TF7(x)} = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1]$	[-1.28,1.28]	[-0.25, …-0.25]

Table 7.2 Multimodal Benchmark Functions (10 Dimensional) [16]

Functions	**Range**	**Shift position**	$\boldsymbol{f_{min}}$
$\boldsymbol{TF8(x)} = \sum_{i=1}^{n} -x_i^2 \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	[-300, … -300]	-418.9829

$\boldsymbol{TF9(x)} = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12,5.12]	[-2, -2, …-2]	0
$\boldsymbol{TF10(x)} = -20exp\left(-0.2\sqrt{\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-32, 32]		0
$\boldsymbol{TF11(x)} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]	[-400, … -400]	0
$\boldsymbol{TF12(x)} = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\ \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4).$ $y_i = 1 + \frac{x+1}{4}.$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m\ x_i > a \\ 0\ -a < x_i < a \\ k(-x_i - a)^m\ x_i < -a \end{cases}$	[-50,50]	[-30, 30, … 30]	0
$\mathbf{TF13(x)} = \mathbf{0.1}\left\{\mathbf{sin^2(3\pi x1)} + \sum_{i=1}^{n}(\boldsymbol{x_i} - \mathbf{1})^2[\mathbf{1} + \mathbf{sin^2(3\pi x_i + 1)}] + (\boldsymbol{x_n} - \mathbf{1})^2[\mathbf{1} + \mathbf{sin^2(2\pi x_n)}]\right\} + \sum_{i=1}^{n}\boldsymbol{u(x_i, 5, 100, 4)}.$	[-50,50]	[-100, … -100]	0

Table 7.3 Composite Benchmark Functions [16]

Note: Dimension=10, Range = [-5, 5], and $f_{min} = 0$

Functions
TF14 (CF1) $f1, f2, f3 \dots f10 =$ Sphere function $\delta 1, \delta 2, \delta 3 \dots \delta 10 = [1,1,1, \dots .1]$ $\lambda 1, \lambda 2, \lambda 3 \dots \lambda 10 = \left[\frac{5}{100}, \frac{5}{100,}, \frac{5}{100}, \dots \frac{5}{100}\right]$
TF15 (CF2) $f1, f2, f3 \dots f10 =$ Griewank's function $\delta 1, \delta 2, \delta 3 \dots \delta 10 = [1,1,1, \dots .1]$ $\lambda 1, \lambda 2, \lambda 3 \dots \lambda 10 = \left[\frac{5}{100}, \frac{5}{100,}, \frac{5}{100}, \dots \frac{5}{100}\right]$
TF16 (CF3) $f1, f2, f3 \dots f10 =$ Griewank's function $\delta 1, \delta 2, \delta 3 \dots \delta 10 = [1,1,1, \dots .1]$ $\lambda 1, \lambda 2, \lambda 3 \dots \lambda 10 = [1,1,1, \dots .1]$
TF17 (CF4) $f1, f2 =$ Ackley's function $f3, f4 =$ Rastrigin's function $f5, f6 =$ Weierstrass function $f7, f8 =$ Griewank's function $f9, f10 =$ Sphere function $\delta 1, \delta 2, \delta 3 \dots \delta 10 = [1,1,1, \dots .1]$ $\lambda 1, \lambda 2, \lambda 3 \dots \lambda 10 = \left[\frac{5}{32}, \frac{5}{32,}, 1,1, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}\right]$
TF18 (CF5) $f1, f2 =$ Rastrigin's function $f3, f4 =$ Weierstrass function $f5, f6 =$ Griewank's function $f7, f8 =$ Ackley's function $f9, f10 =$ Sphere function $\delta 1, \delta 2, \delta 3 \dots \delta 10 = [1,1,1, \dots .1]$ $\lambda 1, \lambda 2, \lambda 3 \dots \lambda 10 = \left[\frac{1}{5}, \frac{1}{5,}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}\right]$

TF19 (CF6)

$f1, f2 =$ Rastrigin's function

$f3, f4 =$ Weierstrass function

$f5, f6 =$ Griewank's function

$f7, f8 =$ Ackley's function

$f9, f10 =$ Sphere function

$\delta 1, \delta 2, \delta 3 \ldots \delta 10 = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$

$\lambda 1, \lambda 2, \lambda 3 \ldots \lambda 10$

$$= \left[0.1 * \frac{1}{5}, 0.2 * \frac{1}{5}, 0.3 * \frac{5}{0.5}, 0.4 * \frac{5}{0.5}, 0.5 * \frac{5}{100}, 0.6 * \frac{5}{100}, 0.7 * \frac{5}{32}, 0.8 * \frac{5}{32}, 0.9 * \frac{5}{100}, 1 * 5/100\right]$$

Table 7.4 CEC-C06 2019 Benchmarks "The 100-Digit Challenge": [60].

No.	Functions	Dimension	Range
1	STORN'S CHEBYSHEV POLYNOMIAL FITTING PROBLEM	9	[-8192, 8192]
2	INVERSE HILBERT MATRIX PROBLEM	16	[-16384, 16384]
3	LENNARD-JONES MINIMUM ENERGY CLUSTER	18	[-4,4]
4	RASTRIGIN'S FUNCTION	10	[-100, 100]
5	GRIEWANGK'S FUNCTION	10	[-100, 100]
6	WEIERSTRASS FUNCTION	10	[-100, 100]
7	MODIFIED SCHWEFEL'S FUNCTION	10	[-100, 100]
8	EXPANDED SCHAFFER'S F6 FUNCTION	10	[-100, 100]
9	HAPPY CAT FUNCTION	10	[-100, 100]
10	ACKLEY FUNCTION	10	[-100, 100]

"NOTE: Interested reader can see this technical paper [60] for more information about the CEC benchmarks."

Table 7.5 ZDT benchmark Mathematical definition

Functions	Mathematical definition
ZDT1	$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) / (n-1)$ $F_1(x) = x_1$ $F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right]\ x \in [0,1]\,.$
ZDT2	$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) / (n-1)$ $F_1(x) = x_1$ $F_2(x) = g(x)\left[1 - (x_1/g(x))^2\right]\ x \in [0,1]\,.$
ZDT3	$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right) / (n-1)$ $F_1(x) = x_1$ $F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)} - x_1/g(x)\sin(10\pi x_1)\right]\ x \in [0,1]\,.$
ZDT4	$g(x) = 91 + \sum_{i=2}^{n}\left[x_i^2 - 10\cos(4\pi x_i)\right]$ $F_1(x) = x_1$ $F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right]\ x_1 \in [0,1], x_i \in [-5,5]\ i = 2,\cdots,10.$
ZDT5	$g(x) = 1 + 9[(\sum_{i=2}^{n} x_i) / (n-1)]^{0.25}$ $F_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi\, x_1)$ $F_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]\ x \in [0,1]\,.$

Table 7.6 CEC 2019 multimodal multi-objective benchmark Mathematical definition [68]

Function	Mathematical definition	Range
MMF 1	$\begin{cases} f_1 = \lvert x_1 - 2\rvert \\ f_2 = 1 - \sqrt{\lvert x_1 - 2\rvert} + 2(x_2 - \sin(6\pi\lvert x_1 - 2\rvert + \pi))^2 \end{cases}$	$x_1 \in [1,3]\,,\quad x_2 \in [-1,1]\,.$

MMF 2	$\begin{cases} f_1 = x_1 \\ f_2 = \begin{cases} 1-\sqrt{x_1}+2(4(x_2-\sqrt{x_1})^2 \\ -2\cos(\frac{20(x_2-\sqrt{x_1})\pi}{\sqrt{2}})+2), \quad 0 \le x_2 \le 1 \\ 1-\sqrt{x_1}+2(4(x_2-1-\sqrt{x_1})^2 \\ -\cos(\frac{20(x_2-1-\sqrt{x_1})\pi}{\sqrt{2}})+2), 1 < x_2 \le 2 \end{cases} \end{cases}$	$x_1 \in [0,1]$, $x_2 \in [0,2]$.
MMF 3	$\begin{cases} f_1 = x_1 \\ f_2 = \begin{cases} 1-\sqrt{x_1}+2(4(x_2-\sqrt{x_1})^2- \\ 2\cos(\frac{20(x_2-\sqrt{x_1})\pi}{\sqrt{2}})+2) \\ 0 \le x_2 \le 0.5, 0.5 < x_2 < 1 \& 0.25 < x_1 \le 1 \\ 1-\sqrt{x_1}+2(4(x_2-0.5-\sqrt{x_1})^2 \\ -\cos(\frac{20(x_2-0.5-\sqrt{x_1})\pi}{\sqrt{2}})+2) \\ 1 \le x_2 \le 1.5, 0 \le x_1 < 0.25 \& 0.5 < x_2 < 1 \end{cases} \end{cases}$	$x_1 \in [0,1]$, $x_2 \in [0,1.5]$.
MMF 4	$\begin{cases} f_1 = \lvert x_1 \rvert \\ f_2 = \begin{cases} 1-x_1^2+2(x_2-\sin(\pi\lvert x_1 \rvert))^2 & 0 \le x_2 < 1 \\ 1-x_1^2+2(x_2-1-\sin(\pi\lvert x_1 \rvert))^2 & 1 \le x_2 \le 2 \end{cases} \end{cases}$	$x_1 \in [-1,1]$, $x_2 \in [0,2]$.
MMF 5	$\begin{cases} f_1 = \lvert x_1-2 \rvert \\ f_2 = \begin{cases} 1-\sqrt{\lvert x_1-2 \rvert}+2(x_2-\sin(6\pi\lvert x_1-2 \rvert+\pi))^2 & -1 \le x_2 \le 1 \\ 1-\sqrt{\lvert x_1-2 \rvert}+2(x_2-2-\sin(6\pi\lvert x_1-2 \rvert+\pi))^2 & 1 < x_2 \le 3 \end{cases} \end{cases}$	$x_1 \in [-1,3]$, $x_2 \in [1,3]$.
MMF 6	$\begin{cases} f_1 = \lvert x_1-2 \rvert \\ f_2 = \begin{cases} 1-\sqrt{\lvert x_1-2 \rvert}+2(x_2-\sin(6\pi\lvert x_1-2 \rvert+\pi))^2 & -1 \le x_2 \le 1 \\ 1-\sqrt{\lvert x_1-2 \rvert}+2(x_2-1-\sin(6\pi\lvert x_1-2 \rvert+\pi))^2 & 1 < x_2 \le 3 \end{cases} \end{cases}$	$x_1 \in [-1,3]$, $x_2 \in [1,2]$.
MMF 7	$\begin{cases} f_1 = \lvert x_1-2 \rvert \\ f_2 = 1-\sqrt{\lvert x_1-2 \rvert}+\{x_2-[0.3\lvert x_1-2 \rvert^2 \cdot \cos(24\pi\lvert x_1-2 \rvert \\ \quad +4\pi)+0.6\lvert x_1-2 \rvert]\cdot\sin(6\pi\lvert x_1-2 \rvert+\pi)\}^2 \end{cases}$	$x_1 \in [1,3]$, $x_2 \in [-1,1]$.
MMF 8	$\begin{cases} f_1 = \sin\lvert x_1 \rvert \\ f_2 = \begin{cases} \sqrt{1-(\sin\lvert x_1 \rvert)^2}+2(x_2-\sin\lvert x_1 \rvert-\lvert x_1 \rvert)^2 & 0 \le x_2 \le 4 \\ \sqrt{1-(\sin\lvert x_1 \rvert)^2}+2(x_2-4-\sin\lvert x_1 \rvert-\lvert x_1 \rvert)^2 & 4 < x_2 \le 9 \end{cases} \end{cases}$	$x_1 \in [-\pi,\pi]$, $x_2 \in [0,9]$.

MMF 9	$\begin{cases} f_1 = x_1 \\ f_2 = \dfrac{g(x_2)}{x_1} \end{cases}$ $g(x) = 2 - \sin^6\left(n_p \pi x\right)$, n_p is the number of global PSs.	$x_1 \in [0.1, 1.1]$, $x_2 \in [0.1, 1.1]$
MMF 10	$\begin{cases} f_1 = x_1 \\ f_2 = \dfrac{g(x_2)}{x_1} \end{cases}$ $g(x) = 2 - \exp\left[-\left(\dfrac{x-0.2}{0.004}\right)^2\right] - 0.8\exp\left[-\left(\dfrac{x-0.6}{0.4}\right)^2\right]$.	$x_1 \in [0.1, 1.1]$, $x_2 \in [0.1, 1.1]$.
MMF 11	$\begin{cases} f_1 = x_1 \\ f_2 = \dfrac{g(x_2)}{x_1} \end{cases}$ $g(x) = 2 - \exp\left[-2\log(2)\cdot\left(\dfrac{x-0.1}{0.8}\right)^2\right]\cdot\sin^6\left(n_p \pi x\right)$, n_p is the total number of global and local PSs.	$x_1 \in [0.1, 1.1]$, $x_2 \in [0.1, 1.1]$.
MMF 12	$\begin{cases} f_1 = x_1 \\ f_2 = g(x_2)\cdot h(f_1, g) \end{cases}$ $g(x) = 2 - \exp\left[-2\log(2)\cdot\left(\dfrac{x-0.1}{0.8}\right)^2\right]\cdot\sin^6\left(n_p \pi x\right)$, n_p is the total number of global and local PSs. $h(f_1, g) = 1 - \left(\dfrac{f_1}{g}\right)^2 - \dfrac{f_1}{g}\sin(2\pi q f_1)$, q is the number of discontinuous pieces in each PF (PS).	$x_1 \in [0, 1]$, $x_2 \in [0, 1]$.

Table 7.7 SOFDO, DA, SSA, and WOA (CEC-1 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC01	CEC01	CEC01	CEC01
Turn-1	47.31038969	28037475792	617169438.7	228926244.8
Turn-2	18.06222239	43822626775	14071704894	7219551658
Turn-3	18699.60882	1314544837	2692769097	57720846.9
Turn-4	1.187890105	19271598315	2228631095	44841246342
Turn-5	9.118739773	22041147121	3842295043	17902732455
Turn-6	7.706902555	15727363198	2008602913	86142933072
Turn-7	174.4159264	7294097614	5184442741	43598257.48
Turn-8	97.97266216	11074650461	11021134427	1.37875E+11
Turn-9	121.7120581	21054635722	468313625.2	64909013519
Turn-10	19.7708307	39132941986	5731614640	18266679217
Turn-11	0.151134525	2.04406E+11	7377251234	1641851.132
Turn-12	4128.8366	2.94636E+11	7970125218	11947308.41
Turn-13	12.67273906	1.63448E+11	2036108830	1.46956E+11
Turn-14	5.454668019	16211657313	10090009171	8751046501
Turn-15	203.811196	73658160256	9929018808	1.21463E+11
Turn-16	2.317801886	1.0673E+11	5321685518	1773200602
Turn-17	1.076157528	1.03833E+11	1258235849	74823454.68
Turn-18	112685.0764	28789516867	21686615089	73237792862
Turn-19	1.782977559	5374692459	9856054633	1110649718
Turn-20	1.536477195	1.09584E+11	2157778145	71733749.77
Turn-21	42.48541374	37654944096	9946802070	14178377809
Turn-22	10.04810942	8678354294	10060791838	462723261.7
Turn-23	1.173608023	74035957270	7760903337	1.41724E+11
Turn-24	1164.180315	12477452348	6442907408	1716750.865
Turn-25	3.419110333	23783481085	873477278	4001823299
Turn-26	75.60996475	8964772906	2389397987	98241735079
Turn-27	1.178233026	31393348840	3863455458	57807207059
Turn-28	13.4028015	12312886389	2139540064	21962756048
Turn-29	2.883288321	86163208948	8671629166	1592984.94
Turn-30	4.373834994	19511334385	3880024370	1.6438E+11
Average	4585.277909	54347246803	6052616313	41123390194
Standard Deviation	20707.62706	66981083351	4750406982	54208979580
WILCOXON RANK-SUM		4.03455E-05	3.18312E-09	0.000108312

Table 7.8 SOFDO, DA, SSA, and WOA (CEC-2 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC02	CEC02	CEC02	CEC02
Turn-1	4.00000000000213	18.9492	18.3432	18.3594
Turn-2	4.00000000000058	69.3545	18.3435	18.3552
Turn-3	4.00000000005982	18.9714	18.3432	18.3491
Turn-4	4.00000000079331	19.7754	18.3439	18.3536
Turn-5	4.00000000000003	259.0589	18.3431	18.3564
Turn-6	4.00000000001410	18.3456	18.3434	18.3524
Turn-7	4.00000000029621	18.3447	18.3433	18.3445
Turn-8	4.00000000000001	20.5691	18.3433	18.3497
Turn-9	4.00000001776344	195.2578	18.3443	18.3544
Turn-10	4.00000000000012	20.8476	18.343	18.3447
Turn-11	4.00000000000007	265.4919	18.3434	18.3452
Turn-12	4.00000000000002	18.7243	18.3431	18.3601
Turn-13	4.00000000006464	18.5931	18.3433	18.3469
Turn-14	4.00000000000282	39.558	18.3435	18.3512
Turn-15	4.00000000000000	54.7404	18.3437	18.3464
Turn-16	4.00000000000109	20.8476	18.3441	18.3453
Turn-17	4.00000000001411	34.8106	18.3431	18.3484
Turn-18	4.00000000134220	18.3487	18.3432	18.3458
Turn-19	4.00000000148891	141.2985	18.3432	18.346
Turn-20	4.00000000080540	52.9786	18.3435	18.3475
Turn-21	4.00000000000129	20.8476	18.3433	18.3454
Turn-22	4.00000000381405	19.1417	18.343	18.3506
Turn-23	4.00000000000000	103.8407	18.3432	18.3478
Turn-24	4.00000000001066	283.6467	18.3433	18.3445
Turn-25	4.00000000000028	20.8476	18.3432	18.3483
Turn-26	4.00000000009768	162.3411	18.3432	18.3494
Turn-27	4.00000000000035	18.592	18.3433	18.3467
Turn-28	4.00000000005117	18.6164	18.3433	18.3441
Turn-29	4.00000000000651	201.1435	18.3455	18.35
Turn-30	4.00000000000000	167.2206	18.3431	18.3545
Average	4.000000001	78.03679333	18.3434233	18.34945
Standard Deviation	3.27925E-09	86.78883455	0.00049458	0.004487588
WILCOXON RANK-SUM		1.81428E-05	3.1703E-252	1.1004E-196

Table 7.9 SOFDO, DA, SSA, and WOA (CEC-3 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC03	CEC03	CEC03	CEC03
Turn-1	13.7024042179575	13.7024	13.7024	13.7024
Turn-2	13.7024042179603	13.7024	13.7024	13.7024
Turn-3	13.7024042179566	13.7024	13.7024	13.7024
Turn-4	13.7024042179563	13.7025	13.7024	13.7024
Turn-5	13.7024042179564	13.7024	13.7024	13.7024
Turn-6	13.7024042179568	13.7024	13.7024	13.7024
Turn-7	13.7024042179569	13.7024	13.7024	13.7024
Turn-8	13.7024042179569	13.7024	13.7024	13.7024
Turn-9	13.7024042179563	13.7024	13.7024	13.7024
Turn-10	13.7024042179569	13.7024	13.7024	13.7024
Turn-11	13.7024042179958	13.7024	13.7024	13.7024
Turn-12	13.7024042179584	13.7025	13.7024	13.7024
Turn-13	13.7024042179563	13.7024	13.7039	13.7024
Turn-14	13.7024042179611	13.7024	13.7024	13.7024
Turn-15	13.7024042179590	13.7024	13.7024	13.7024
Turn-16	13.7024042179563	13.7024	13.7024	13.7024
Turn-17	13.7024042179582	13.7024	13.7024	13.7024
Turn-18	13.7024042179564	13.7024	13.7024	13.7024
Turn-19	13.7024042179939	13.7024	13.7024	13.7024
Turn-20	13.7024042179571	13.7024	13.7024	13.7024
Turn-21	13.7024042179563	13.7024	13.7024	13.7024
Turn-22	13.7024042179580	13.7025	13.7024	13.7024
Turn-23	13.7024042180319	13.7024	13.7024	13.7024
Turn-24	13.7024042179564	13.7024	13.7024	13.7024
Turn-25	13.7024042179563	13.7024	13.7024	13.7024
Turn-26	13.7024042179628	13.7024	13.7024	13.7024
Turn-27	13.7024042179824	13.7062	13.7024	13.7024
Turn-28	13.7024042179779	13.7029	13.7024	13.7024
Turn-29	13.7024042179564	13.7024	13.7024	13.7024
Turn-30	13.7024042179598	13.7024	13.7024	13.7024
Average	13.70240422	13.70255	13.70245	13.7024
Standard Deviation	1.67631E-11	0.000695	0.000274	7.23E-15
WILCOXON RANK-SUM		0.244847	0.363647	1.2E-306

Table 7.10 SOFDO, DA, SSA, and WOA (CEC-4 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC04	CEC04	CEC04	CEC05
Turn-1	42.78813606	89.6286	15.9244	164.5268
Turn-2	14.78013468	167.0365	10.9496	296.0315
Turn-3	28.85880335	432.4384	45.773	183.8552
Turn-4	33.889548	535.1613	53.7325	240.5655
Turn-5	25.87437127	34.0404	16.9193	496.6138
Turn-6	29.86515906	446.6602	44.7779	307.6974
Turn-7	25.99999354	138.6272	39.8033	210.571
Turn-8	69.65174971	110.1406	42.788	376.6262
Turn-9	24.87916538	71.5877	77.6112	156.8946
Turn-10	80.59622351	52.7565	34.8285	995.9347
Turn-11	22.88905891	72.1789	49.7528	357.3357
Turn-12	10.98302357	728.4271	36.8184	332.3681
Turn-13	5.993071905	586.8426	50.7478	666.2306
Turn-14	45.77378602	124.1159	33.8336	232.6558
Turn-15	18.35127492	120.446	29.8538	464.7928
Turn-16	25.87948706	479.5643	24.879	486.0267
Turn-17	38.52292218	94.2813	39.8031	351.4897
Turn-18	14.08001799	157.269	45.7726	536.7224
Turn-19	17.68165395	101.4658	36.8183	229.4065
Turn-20	43.81735389	346.493	47.7629	514.4632
Turn-21	42.7879921	197.8004	72.6367	444.6604
Turn-22	59.70223506	1675.78	20.8991	299.6801
Turn-23	20.00627968	167.5702	125.3686	288.8738
Turn-24	36.25108137	508.5567	52.7373	522.5866
Turn-25	39.80329451	107.5156	20.8991	499.5005
Turn-26	26.3504424	173.7315	33.8335	261.7723
Turn-27	30.95242982	581.1883	37.8134	1298.5194
Turn-28	38.80828254	1705.7	46.7678	146.5564
Turn-29	25.94865361	189.7548	13.9345	156.6548
Turn-30	50.74776509	133.9233	46.7678	320.6484
Average	33.0837797	344.3561	41.69359	394.6753633
Standard Deviation	16.81143076	414.0982	22.21909	248.562718
WILCOXON RANK-SUM		0.000124	0.095911	7.43657E-11

Table 7.11 SOFDO, DA, SSA, and WOA (CEC-5 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC05	CEC05	CEC05	CEC05
Turn-1	2.063899381	2.8755	2.3885	2.5468
Turn-2	2.147672574	2.7173	2.0787	3.7271
Turn-3	2.027093396	2.8811	2.1157	2.2438
Turn-4	2.049207798	2.7866	2.1106	2.7735
Turn-5	2.211490813	2.8681	2.2853	3.0202
Turn-6	2.041838794	2.165	2.1354	3.0368
Turn-7	2.123045461	3.0034	2.165	2.5722
Turn-8	2.297786812	2.1296	2.1525	2.5281
Turn-9	2.142647078	2.5433	2.1305	2.9799
Turn-10	2.226491485	2.5589	2.2561	2.552
Turn-11	2.130355627	2.7431	2.2164	2.6909
Turn-12	2.137832728	2.4013	2.1451	2.3443
Turn-13	2.120714245	3.1426	2.1649	2.5361
Turn-14	2.13539351	2.4085	2.1672	2.5784
Turn-15	2.066404738	2.3946	2.3496	2.4513
Turn-16	2.427576298	2.3469	2.2534	2.9412
Turn-17	2.086021183	2.6721	2.1183	2.7177
Turn-18	2.132887844	3.2787	2.3173	2.5785
Turn-19	2.218857232	2.2813	2.3691	2.9724
Turn-20	2.078706066	2.2754	2.1697	2.6988
Turn-21	2.056523037	2.5324	2.0983	2.8538
Turn-22	2.036960532	2.4483	2.1206	2.8501
Turn-23	2.164835378	2.4502	2.2263	3.0072
Turn-24	2.223834127	2.3975	2.337	3.1599
Turn-25	2.117997749	3.15	2.4403	2.6397
Turn-26	2.054154862	2.1768	2.241	2.5319
Turn-27	2.246130146	2.4426	2.0911	2.6997
Turn-28	2.152484698	2.2157	2.0764	2.7464
Turn-29	2.135265722	2.0693	2.3887	2.6321
Turn-30	2.123097114	2.3584	2.1426	2.4151
Average	2.139240214	2.55715	2.208387	2.734196667
Standard Deviation	0.087217722	0.324493801	0.106429	0.291747891
WILCOXON RANK-SUM		6.05468E-09	0.007884	2.36828E-15

Table 7.12 SOFDO, DA, SSA, and WOA (CEC-6 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC06	CEC06	CEC06	CEC06
Turn-1	10.39637831	5.5487	4.9295	10.3069
Turn-2	11.63071162	10.5822	9.8578	10.7929
Turn-3	12.58023279	8.6436	5.1733	11.1581
Turn-4	12.30460484	9.904	6.4655	10.5958
Turn-5	11.69124008	12.7974	4.703	10.7253
Turn-6	11.38021785	9.3111	5.906	11.4092
Turn-7	13.2108508	9.5077	8.466	11.0506
Turn-8	12.52072973	10.3758	7.6821	11.016
Turn-9	12.28822755	12.2179	4.0309	10.8705
Turn-10	11.50422739	12.017	5.4241	11.9412
Turn-11	11.65260248	10.975	4.6514	13.2563
Turn-12	12.49558646	12.1569	5.3138	10.1435
Turn-13	12.60617414	7.9069	5.5037	10.4676
Turn-14	12.29600911	6.1943	5.2774	10.0224
Turn-15	12.59789955	11.0248	5.9707	11.1143
Turn-16	12.42025224	10.3275	5.981	11.7086
Turn-17	12.79266496	8.7483	5.5106	10.7182
Turn-18	12.19912322	10.1463	8.1922	9.5097
Turn-19	12.00312658	9.3353	4.6209	12.9841
Turn-20	12.06655808	10.2246	7.3945	11.3831
Turn-21	12.91716509	10.3802	6.0266	10.2766
Turn-22	11.90377152	9.4517	8.1371	10.0168
Turn-23	12.27636218	11.8455	5.9562	10.857
Turn-24	12.38062877	10.3299	5.0643	10.7565
Turn-25	10.80314081	9.8607	6.7059	9.5419
Turn-26	12.56325726	8.9004	4.2675	9.6577
Turn-27	11.48026586	9.6158	3.9714	11.4028
Turn-28	12.55390535	10.9282	6.921	9.6763
Turn-29	12.48168732	7.8221	8.5798	7.9987
Turn-30	12.00027882	9.7856	5.7095	9.8954
Average	12.13326269	9.895513333	6.07979	10.70846667
Standard Deviation	0.61049913	1.640410821	1.487278233	1.032452946
WILCOXON RANK-SUM		2.90314E-09	2.19607E-28	1.96797E-08

Table 7.13 SOFDO, DA, SSA, and WOA (CEC-7 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC07	CEC07	CEC07	CEC07
Turn-1	152.64419	379.6903	414.0824	193.1639
Turn-2	145.96787	673.3759	761.598	364.2248
Turn-3	112.9891	280.7092	213.5215	545.2148
Turn-4	130.27749	892.8387	597.4322	392.624
Turn-5	128.60217	264.5389	659.443	690.3121
Turn-6	116.15789	306.8299	255.0727	776.3766
Turn-7	103.95957	989.5615	1244.641	781.2573
Turn-8	133.48626	366.6919	339.221	613.5876
Turn-9	103.59088	762.7382	954.437	499.3797
Turn-10	110.11986	296.0694	221.9694	276.0807
Turn-11	128.86368	872.5925	501.5119	472.434
Turn-12	110.64458	784.3011	427.9813	382.1168
Turn-13	110.61542	71.2341	404.3251	560.0888
Turn-14	127.34166	409.69	55.8464	331.5762
Turn-15	136.56458	255.9608	234.8383	412.9315
Turn-16	120.58763	-48.244	80.4828	516.7854
Turn-17	115.48106	912.1153	452.1595	538.4653
Turn-18	104.55616	944.6172	708.5561	134.1435
Turn-19	105.07145	972.7928	345.2185	496.9047
Turn-20	121.47754	335.6231	-0.7572	623.2101
Turn-21	128.61796	368.5244	206.0207	188.3266
Turn-22	117.41311	740.5508	238.3186	537.4072
Turn-23	128.54729	155.9691	175.9931	743.3001
Turn-24	113.9323	213.6375	271.6066	347.0528
Turn-25	151.02416	1041.3708	261.6816	1004.7824
Turn-26	112.77487	553.1951	425.4884	297.1115
Turn-27	105.31476	727.2664	826.7202	402.9513
Turn-28	116.31113	1021.7706	722.4315	613.5173
Turn-29	104.07245	945.5743	40.8896	434.638
Turn-30	117.56743	877.0076	271.1605	550.5653
Average	120.48582	578.9531133	410.39639	490.6843433
Standard Deviation	13.82608	329.3983039	290.556184	193.831761
WILCOXON RANK-SUM		2.69474E-10	1.0424E-06	6.26919E-15

Table 7.14 SOFDO, DA, SSA, and WOA (CEC-8 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC08	CEC08	CEC08	CEC08
Turn-1	6.859511388	7.4482	6.9561	5.7971
Turn-2	6.505207612	6.998	6.4551	7.3622
Turn-3	6.066890951	6.2305	6.1743	6.77
Turn-4	5.74620445	6.6045	6.4865	6.5837
Turn-5	5.491266762	7.6277	6.2141	7.0908
Turn-6	6.130261133	6.7965	7.0755	7.0245
Turn-7	5.259087655	6.9041	6.6906	6.1614
Turn-8	5.528476595	7.425	7.2673	6.9177
Turn-9	5.785685511	6.8713	6.5516	6.949
Turn-10	7.463143827	6.9199	6.0854	7.1525
Turn-11	6.224493406	6.9575	6.9877	6.612
Turn-12	5.367837302	7.0771	4.9909	7.3004
Turn-13	5.78039937	6.7057	6.3324	6.4327
Turn-14	7.21044196	7.2308	6.1481	6.9063
Turn-15	6.688923818	6.3027	5.8683	7.5205
Turn-16	6.971349336	7.1246	5.781	7.6415
Turn-17	5.531485242	6.6583	5.6015	7.1216
Turn-18	6.218795742	6.0661	5.7275	7.3328
Turn-19	5.105802423	6.4066	6.8119	7.2649
Turn-20	5.132493271	6.6279	6.513	6.5111
Turn-21	6.865575083	7.0172	6.9309	6.6222
Turn-22	5.146094042	6.8715	6.7335	6.7024
Turn-23	6.793155497	7.4661	6.5345	6.2238
Turn-24	4.715090381	6.8776	5.7262	7.1486
Turn-25	5.491497602	7.3181	5.3343	6.477
Turn-26	5.346112738	7.314	6.8832	6.9553
Turn-27	6.865229356	5.4949	7.0898	6.996
Turn-28	6.770146559	7.3213	6.14	7.2686
Turn-29	7.327894441	6.0196	5.8268	7.1748
Turn-30	6.676007224	7.5189	7.2504	7.2471
Average	6.102152023	6.873406667	6.37228	6.90895
Standard Deviation	0.769938406	0.501501218	0.586156	0.426862168
WILCOXON RANK-SUM		2.36381E-05	0.131704	5.23311E-06

Table 7.15 SOFDO, DA, SSA, and WOA (CEC-9 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC09	CEC09	CEC09	CEC09
Turn-1	2	6.3329	3.5083	10.017
Turn-2	2	6.1564	3.4818	5.1904
Turn-3	2	6.6349	3.7236	6.0222
Turn-4	2	7.4622	3.6735	4.8435
Turn-5	2	4.5705	4.1222	5.9159
Turn-6	2	4.7252	3.4654	6.9429
Turn-7	2	7.0999	3.6386	5.9272
Turn-8	2	6.005	3.478	6.3543
Turn-9	2	4.8411	3.7195	4.7623
Turn-10	2	5.5265	3.5966	4.8368
Turn-11	2	3.9522	3.6604	5.2188
Turn-12	2	4.0579	3.5143	5.6591
Turn-13	2	6.3484	4.3611	5.8536
Turn-14	2	4.0179	3.638	12.6776
Turn-15	2	4.4408	3.6408	6.0921
Turn-16	2	5.3724	3.5187	4.4763
Turn-17	2	7.796	4.0819	4.8945
Turn-18	2	6.0583	4.0786	5.9504
Turn-19	2	5.6386	3.5009	5.4161
Turn-20	2	5.904	3.4759	4.9949
Turn-21	2	20.1226	3.5705	4.8122
Turn-22	2	5.1588	3.4783	5.5228
Turn-23	2	4.7246	4.0846	6.0424
Turn-24	2	4.6094	3.5097	4.6486
Turn-25	2	4.7184	3.5959	6.9274
Turn-26	2	8.0226	3.4581	5.7938
Turn-27	2	5.4979	3.7002	4.7358
Turn-28	2	4.8912	3.5725	6.5317
Turn-29	2	5.4353	3.5222	6.1261
Turn-30	2	5.2798	3.7416	4.927
Average	2	6.046723333	3.67039	5.937123333
Standard Deviation	2E-10	2.870985139	0.236213552	1.656591579
WILCOXON RANK-SUM		1.80487E-10	3.79927E-43	7.40292E-19

Table 7.16 SOFDO, DA, SSA, and WOA (CEC-10 results) for 3o turns.

Algorithms	SOFDO	DA	SSA	WOA
CEC Benchmarks	CEC10	CEC10	CEC10	CEC10
Turn-1	2.718281828	21.0277	21.0022	21.5536
Turn-2	2.718281828	21.0177	20.9997	21.2558
Turn-3	2.718281828	21.1826	20.9999	21.4091
Turn-4	2.718281828	21.5789	21	21.4027
Turn-5	2.718281828	21.1221	21.14	21.2813
Turn-6	2.718281828	21.0769	21	21.1476
Turn-7	2.718281828	21.1345	21	21.0703
Turn-8	2.718281828	21.4297	21.0459	21.2292
Turn-9	2.718281828	21.183	20.9994	21.4089
Turn-10	2.718281828	21.2166	21	21.2474
Turn-11	2.718281828	21.3773	21	21.2318
Turn-12	2.718281828	21.0121	21	21.2386
Turn-13	2.718281828	21.3871	21.1716	21.1999
Turn-14	2.718281828	21.2647	21	21.2161
Turn-15	2.718281828	21.108	21	21.2767
Turn-16	2.718281828	21.1077	21	21.2294
Turn-17	2.718281828	21.4606	20.9999	21.3178
Turn-18	2.718281828	21.5398	21	21.2702
Turn-19	2.718281828	21.0045	21	21.5316
Turn-20	2.718281828	21.3781	21	21.1789
Turn-21	2.718281828	21.5106	20.9999	21.2532
Turn-22	2.718281828	21.5302	21.139	21.1993
Turn-23	2.718281828	21.3126	21	21.2273
Turn-24	2.718281828	21.2775	21.1069	21.1659
Turn-25	2.718281828	21.2375	21	21.2659
Turn-26	2.718281828	21.2465	21.1227	21.4229
Turn-27	2.718281828	21.1716	21	21.2942
Turn-28	2.718281828	21.3572	21.325	21.1109
Turn-29	2.718281828	21.4105	21.1475	21.3028
Turn-30	2.718281828	21.1488	21	21.3446
Average	2.718281828	21.26042	21.03998667	21.27613
Standard Deviation	4.51681E-16	0.171459448	0.0779867	0.111071795
WILCOXON RANK-SUM		2.2248E-111	6.3971E-131	2.4593E-122

Table 7.17 MOFDO, MODA, MOPSO, and NSGA-III (MMF-1 results)

Benchmarks	MMF1											
Algorithms	MOFDO			MOPSO			MODA			NSGA-III		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.220494064	0.105646	3.543953	0.424244817	0.160283	0.529915	0.337568	0.132947	1.61469	0.00352	0.002671	0.005414
2	0.195809057	0.103026	1.274311	0.542845346	0.169664	0.717629	2.089593	1.485137	7.815159	0.003334	0.002037	0.004482
3	0.316828801	0.091739	0.963267	0.338186944	0.135344	0.479175	0.801761	0.290654	1.00689	0.004438	0.003235	0.005871
4	0.176909381	0.06517	0.346557	0.315942451	0.151968	0.564066	1.076244	0.410441	2.417011	0.003552	0.002644	0.004566
5	0.126211628	0.063471	0.378836	0.306009085	0.183058	0.723522	0.91446	0.236259	1.090711	0.003916	0.002936	0.005424
6	0.166741545	0.089358	1.115436	0.238792271	0.092079	0.371919	0.27665	0.096082	0.875318	0.003583	0.002019	0.004363
7	0.176807596	0.093818	2.40853	0.261299755	0.102986	0.386088	0.673809	0.316312	0.897928	0.001293	0.000939	0.002751
8	0.134154594	0.08636	4.123824	0.210556945	0.153264	0.452644	0.777896	0.40848	18.05439	0.003777	0.003026	0.005205
9	0.177695266	0.093219	1.530539	0.19740199	0.101545	0.464027	0.33168	0.154481	0.519446	0.003785	0.002615	0.005655
10	0.163145725	0.093712	2.053542	0.406721692	0.171911	0.624904	0.281537	0.117568	0.418536	0.003271	0.002412	0.005624
11	0.137263541	0.094979	0.602591	0.303852085	0.143397	0.529131	0.614164	0.073823	0.79877	0.003722	0.002493	0.004625
12	0.197366791	0.091362	1.934955	0.331447783	0.148963	0.559782	0.393085	0.04303	1.575941	0.0037	0.001967	0.004849
13	0.147129978	0.076518	0.603724	0.268167141	0.105232	0.501241	0.265113	0.081689	0.403481	0.003919	0.002613	0.007045
14	0.122992008	0.076453	1.227907	0.206372955	0.10116	0.363955	0.587872	0.260985	0.679939	0.002948	0.002171	0.004627
15	0.208331008	0.084784	6.482355	0.210298383	0.109477	0.424654	0.888002	0.195393	1.26068	0.003119	0.001804	0.004241
16	0.201950099	0.080456	3.919223	0.222346959	0.167351	0.604766	1.953375	1.335854	18.54926	0.003482	0.001918	0.004584
17	0.250899609	0.090904	6.090059	0.252647398	0.111147	0.381351	0.415541	0.265941	0.574782	0.003609	0.002383	0.00594
18	0.167669972	0.108522	0.926954	0.532653643	0.201104	0.673186	1.553651	0.699707	3.594658	0.00302	0.001836	0.004591
19	0.140991095	0.08052	1.325201	0.364247578	0.092753	1.145611	1.096948	0.537195	17.33246	0.00362	0.002731	0.006124
20	0.223233145	0.075041	2.449195	0.231918517	0.153665	0.442817	0.830049	0.125561	0.9486	0.002985	0.001647	0.004
21	0.141337645	0.087594	0.884974	0.356875758	0.135695	0.533776	0.497367	0.077906	2.278821	0.00304	0.001925	0.004726
22	0.227415454	0.094905	1.844236	0.246186326	0.106983	0.491178	1.294946	0.571189	7.598447	0.003526	0.001882	0.004868
23	0.208431094	0.101706	0.664315	0.396322095	0.221279	0.735476	1.539361	0.253792	118.9634	0.00384	0.002359	0.004841
24	0.114229087	0.078615	0.769521	0.327865071	0.142835	0.444101	2.106776	1.000186	2.268363	0.00362	0.002744	0.005072
25	0.186932075	0.089298	0.70043	0.393579708	0.137895	0.476681	0.862589	0.109066	3.34267	0.004628	0.001825	0.005816
26	0.1761374	0.094824	2.491466	0.260574216	0.133813	0.493611	0.610698	0.184262	6.082971	0.004192	0.0027	0.005435
27	0.16743679	0.082952	2.415965	0.146594632	0.091222	0.275672	1.000146	0.759731	1.215569	0.003298	0.002082	0.004012
28	0.233542962	0.111682	1.352158	0.314714226	0.181354	0.443771	1.139315	0.22268	56.2344	0.003614	0.00236	0.005111
29	0.158116342	0.072613	2.685899	0.419825784	0.118677	0.520223	0.594026	0.264271	0.852636	0.003246	0.002046	0.004233
30	0.254372623	0.087552	10.11013	0.623563961	0.23139	0.92263	0.506 768	0.145374	9.702033	0.003861	0.002262	0.005537
Mean	0.184019212	0.088227	2.240669	0.321735184	0.141916	0.542583	0.877033	0.361867	9.632266	0.003515	0.002276	0.004987787
STD	0.045445864			0.110864516			0.530292			0.00058		
P-value				4.41881E-08			1.76E-09			1.41E-29		

Table 7.18 MOFDO, MODA, MOPSO, AND NSGA-III (MMF-2 results).

Benchmarks	MMF2											
Algorithms	**MOFDO**			**MOPSO**			**MODA**				**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.10415	0.042603	0.392329	0.263710593	0.176003	1.74974	0.087489068	0.042178	0.393981	NaN	0	6.236517
2	0.106212	0.03444	1.065737	0.246677629	0.165642	0.466803	0.6158273	0.119171	1.828538	NaN	0	6.540795
3	0.090791	0.045086	2.174423	0.295205901	0.204537	2.135928	0.334609636	0.007784	1.059702	NaN	0	15.81373
4	0.035204	0.0173	0.639249	0.232954883	0.177173	2.294571	0.566678687	0.12937	1.463504	NaN	0	24.08798
5	0.108128	0.034024	1.835712	0.232733594	0.160934	2.056999	0.031972538	0.001517	0.190776	NaN	0	55.1582
6	0.071242	0.021635	3.710047	0.314410012	0.203257	0.451427	0.199802675	0.031325	0.597394	NaN	0	12.81478
7	0.114308	0.029593	0.276629	0.253534167	0.184623	0.659288	0.359632308	0.043368	1.721268	NaN	0	7.372137
8	0.062128	0.039781	1.553245	0.193790277	0.143242	0.445122	0.226837261	0.006009	13.04409	NaN	0	2.107909
9	0.118296	0.034765	1.262518	0.230113466	0.163671	0.541107	1.38415706	0.144502	183.39	NaN	0	17.7101
10	0.106689	0.036885	1.190002	0.323794653	0.227966	1.230936	0.460119053	0.057221	1.035446	NaN	0	80.14186
11	0.058979	0.025723	0.174941	0.293605691	0.193893	3.038245	0.311188482	0.051438	2.336916	NaN	0	9.772417
12	0.086572	0.042766	0.220413	0.282508721	0.203821	0.394914	0.349381833	0.086618	3.216488	NaN	0	13.33163
13	0.103236	0.048239	0.37964	0.330166498	0.238674	0.424499	0.250097111	0.140858	1.64872	NaN	0	15.85728
14	0.084614	0.038253	0.899843	0.253647916	0.166715	0.729319	0.176827505	0.050716	0.401444	NaN	0	20.36475
15	0.065809	0.034435	0.395134	0.177928964	0.130982	1.080938	0.29488042	0.050932	1.701687	NaN	0	9.949882
16	0.135908	0.081027	0.273623	0.212503615	0.143909	0.264839	0.31413053	0.178527	1.578534	NaN	0	22.24762
17	0.134386	0.058794	1.59672	0.249669482	0.177441	0.323944	0.45973869	0.039749	2.015196	NaN	0	23.22014
18	0.086044	0.047371	2.108948	0.28352746	0.218902	0.951643	0.62779877	0.254576	2.296881	NaN	0	49.80878
19	0.106344	0.026542	0.408287	0.287889116	0.208182	0.998099	0.699021778	0.267098	4.59711	NaN	0	5.272455
20	0.11416	0.053951	1.41322	0.278338016	0.204195	0.622576	0.32899374	0.199078	1.490572	NaN	0	15.90178
21	0.068455	0.034021	0.691523	0.246493113	0.180476	0.519913	0.425886254	0.113213	1.268957	NaN	0	17.03255
22	0.068829	0.042191	1.348942	0.311670449	0.233087	1.164496	0.149269447	0.040871	0.775143	NaN	0	82.12944
23	0.079205	0.034607	0.811516	0.288797613	0.186276	3.088039	0.437357432	0.018322	70.92937	NaN	0	7.987966
24	0.084862	0.043348	0.519503	2.08E-01	0.16612	0.315008	0.296694611	0.026573	3.496328	NaN	0	2.64033
25	0.094307	0.028221	0.812387	0.279125193	0.205041	1.016997	0.32808839	0.109329	1.479076	NaN	0	37.62585
26	0.073687	0.035417	0.688562	3.01E-01	0.20602	3.33098	0.099063604	0.01608	0.274982	NaN	0	11.45158
27	0.072514	0.030863	0.438493	0.241720414	0.178619	0.714755	0.565125384	0.282557	1.626026	NaN	0	11.06373
28	0.126716	0.020418	0.536318	0.214801214	0.158256	0.282627	1.297628207	0.06576	183.8025	NaN	0	20.75984
29	0.085281	0.033765	0.367786	0.506546237	0.289103	1.243836	0.151659917	0.037318	1.288801	NaN	0	9.383675
30	0.085612	0.036871	1.011212	0.344455903	0.22415	0.742294	0.515930241	0.037353	4.818113	NaN	0	75.20503
Mean	0.091089	0.037765	0.97323	0.272644304	0.190697	1.109329	0.411529598	0.088314	16.52558	NaN	0	22.96636
STD	0.023709			0.060637357			0.304118383					
P value				5.48551E-22			3.45868E-07					

Table 7.19 MOFDO, MODA, MOPSO, and NSGA-III (MMF-3 results).

Benchmarks	MMF3											
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.084779	0.038524	1.957016	0.547167586	0.419846	1.94971	0.096878	0.004635	1.841466	3.638797	0.003799	8.750559
2	0.077457	0.036256	0.450205	0.580508034	0.425026	1.165872	0.154699	0.03414	0.758218	5.246135	0.406897	17.38011
3	0.059605	0.039026	0.384787	0.513246506	0.347297	1.334269	0.253807	0.074584	0.767995	4.63844	0.032774	14.34921
4	0.100117	0.04231	1.180474	0.724875427	0.536558	3.227663	0.537981	0.15662	4.267671	5.742251	0.006092	16.56828
5	0.077422	0.03228	1.497297	0.468543813	0.360964	0.811757	0.149141	0.025203	1.632161	5.207974	0.006622	14.86276
6	0.108114	0.027324	2.150871	0.663014437	0.479454	1.635842	1.089914	0.153097	155.2578	4.618232	0.002799	16.22396
7	0.122631	0.050723	1.372592	0.649143454	0.489605	0.99914	0.305566	0.145035	0.669544	4.03329	0.008358	11.17838
8	0.102469	0.034689	0.934443	0.338144107	0.245327	0.48875	0.0593	0.013408	0.37941	7.636254	0.004527	19.20256
9	0.086411	0.03991	0.621877	0.439863293	0.311505	0.976723	0.280917	0.010344	22.67627	8.004299	0.034491	22.0218
10	0.091497	0.044113	0.506175	0.679137732	0.566539	1.713874	0.240371	0.040408	11.19409	6.092482	0.00466	18.10488
11	0.05672	0.040027	0.315149	0.302876335	0.192525	3.005671	0.033393	0.002469	0.231078	5.748422	0.309348	16.4163
12	0.102575	0.046794	1.14245	0.324891601	0.237682	0.44584	0.147509	0.017725	0.35382	5.388575	0.129474	16.74938
13	0.093386	0.032412	1.116024	0.308959509	0.224388	0.655155	0.377447	0.018815	26.78053	4.869932	0.004555	13.77996
14	0.08618	0.028388	1.361184	0.209087774	0.157153	0.510439	0.130558	0.006005	1.311184	4.145873	0.053366	10.63949
15	0.08113	0.042276	0.253492	0.57601224	0.420379	3.020181	0.285434	0.062675	0.816336	7.782087	0.007865	20.19212
16	0.125902	0.04339	1.065393	0.448790909	0.331241	4.119746	0.378291	0.152816	1.126928	5.938083	0.004213	15.41965
17	0.056652	0.039104	0.471591	0.762887165	0.573795	1.243596	0.288032	0.03458	2.268135	4.920791	0	14.0417
18	0.06697	0.035667	0.439513	0.276281643	0.219368	0.429528	0.389604	0.164753	1.013315	7.024639	0.003727	17.63984
19	0.085261	0.037882	0.671729	0.30446839	0.211904	1.655647	0.402818	0.181919	4.496757	6.013155	0.006989	15.73729
20	0.081699	0.036301	0.523186	0.343571357	0.282308	0.588884	0.546145	0.100805	2.203008	9.906555	0.014056	32.12804
21	0.091481	0.059609	0.770977	0.447794856	0.298153	1.641	0.175069	0.060701	1.031299	4.008853	0.170831	11.75522
22	0.097887	0.055834	1.262064	0.517031317	0.384079	1.66454	0.746734	0.365027	1.835173	4.648376	0.163239	14.67967
23	0.08111	0.036888	1.886861	0.635047408	0.454253	3.890423	0.29651	0.036991	1.954739	3.688879	0.03245	10.93256
24	0.113764	0.051096	0.983182	2.40E-01	0.148547	1.283179	0.363439	0.018324	16.00553	10.27648	0.007567	28.26593
25	0.109754	0.044486	0.30913	0.621841151	0.449345	2.990084	0.362901	0.019277	4.759344	7.2782	0.067008	17.98147
26	0.077232	0.034394	1.202733	0.310414347	0.173037	3.049095	0.225872	0.04898	1.042223	4.431331	0.007733	11.45199
27	0.095688	0.044539	1.032611	0.506119436	0.338137	2.040027	1.354225	0.060352	214.3836	8.803401	0.011057	25.84451
28	0.107553	0.045184	2.576098	0.271030135	0.177205	1.461569	0.625668	0.122618	2.440401	8.686266	0.043304	28.75343
29	0.121079	0.05101	0.966368	0.457029211	0.334077	1.100815	1.143579	0.003657	102.4654	3.493568	0.005534	9.802979
30	0.093829	0.047402	1.254463	0.510637273	0.343327	2.091346	0.258115	0.025304	24.53846	14.81083	0.017658	36.08623
Mean	0.091212	0.041261	1.021998	0.465946198	0.337767	1.706345	0.389997	0.072042	20.35006	6.224081	0.052367	0.004987787
STD	0.018443			0.156619762			0.319535			2.47732		
P-value				7.46013E-19			3.73E-06			1.24E-19		

Table 7.20 MOFDO, MODA, MOPSO, and NSGA-III (MMF-4 results).

Benchmarks	MMF4											
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.068065	0.046315	0.130608	0.067045	0.042134	0.143518	0.010122	0.000389	0.025655	0.059008	0.007177	0.167588
2	0.065766	0.047384	0.148289	0.071828	0.043766	0.152347	0.005962	0.000151	0.043754	0.043363	0.005964	0.113527
3	0.065584	0.045099	0.110111	0.053603	0.046381	0.157928	0.008495	2.54E-04	0.018616	0.043856	0.006202	0.11794
4	0.178983	0.057099	0.217984	0.072729	0.043341	0.147485	0.006378	3.66E-05	0.014455	0.035355	0.004623	0.103444
5	0.110828	0.04302	0.214691	0.074663	0.044058	0.167086	0.00958	0.000223	0.019506	0.04264	0.005415	0.123774
6	0.083989	0.043268	0.093854	0.071724	0.045488	0.174811	0.005304	5.87E-07	0.0153	0.047679	0.006242	0.127337
7	0.081525	0.044402	0.154857	0.066893	0.045905	0.139216	0.005254	6.21E-04	0.029298	0.042262	0.005701	0.111129
8	0.055768	0.044203	0.151517	0.066086	0.041683	0.130992	0.005689	1.41E-05	0.042552	0.044423	0.005386	0.143678
9	0.082976	0.044794	0.188902	0.072517	0.044418	0.137409	0.009262	0.001133	0.018952	0.059103	0.00449	0.161737
10	0.056352	0.04491	0.10986	0.061854	0.04371	0.100654	0.006949	1.08E-05	0.028618	0.047844	0.004232	0.13236
11	0.076258	0.044185	0.201057	0.082545	0.044463	0.427929	0.005881	0.00025	0.016479	0.054144	0.005986	0.1345
12	0.063004	0.044592	0.215516	0.074738	0.053392	0.141161	0.004909	2.14E-04	0.047565	0.044498	0.004995	0.135164
13	0.066291	0.043072	0.222162	0.071066	0.043688	0.108736	0.005056	6.69E-05	0.031078	0.068511	0.00572	0.176275
14	0.097672	0.049542	0.178349	0.058655	0.036883	0.152073	0.005604	2.80E-04	0.047251	0.040237	0.004109	0.117105
15	0.091103	0.044784	0.392682	0.065032	0.043527	0.145113	0.02346	0.001175	0.030838	0.043802	0.005982	0.120184
16	0.192679	0.050915	0.211903	0.065732	0.044462	0.121093	0.017611	0.000145	0.041181	0.052845	0.005845	0.167444
17	0.055011	0.042316	0.184496	0.075193	0.047631	0.136596	0.006019	0.000217	0.04762	0.05801	0.005871	0.161729
18	0.088726	0.048712	0.212478	0.062729	0.04454	0.168035	0.008161	4.19E-04	0.038433	0.065515	0.007376	0.190736
19	0.066082	0.041212	0.098108	0.068282	0.043007	0.170301	0.00581	0.000198	0.048241	0.043441	0.005232	0.114747
20	0.128383	0.047949	0.284853	0.063483	0.041159	0.241931	0.006516	0.00022	0.020621	0.057841	0.005035	0.15564
21	0.061288	0.045276	0.126442	0.072446	0.043235	0.161144	0.007492	4.53E-04	0.036959	0.036296	0.005853	0.107767
22	0.11747	0.044314	0.242057	0.068513	0.044579	0.159872	0.006268	0.000476	0.022548	0.041223	0.004617	0.088688
23	0.057525	0.042742	0.238205	0.067276	0.044055	0.163149	0.006534	1.21E-04	0.034639	0.024153	0.004366	0.066276
24	0.069109	0.046101	0.234371	0.065342	0.044129	0.153521	0.008707	4.98E-06	0.048865	0.064478	0.005299	0.163947
25	0.057872	0.046198	0.182977	0.063014	0.043077	0.149111	0.006691	2.07E-04	0.037803	0.056728	0.005787	0.156067
26	0.056579	0.041707	0.118318	0.067497	0.042761	0.142883	0.00689	5.97E-05	0.042373	0.032804	0.005314	0.088825
27	0.069669	0.042695	0.168499	0.067473	0.042721	0.192267	0.004454	1.06E-03	0.012713	0.044181	0.005874	0.126569
28	0.075159	0.045315	0.209132	0.071676	0.038395	0.147847	0.009751	0.000206	0.056809	0.040175	0.003993	0.110906
29	0.062017	0.043485	0.198365	0.063833	0.04305	0.15427	0.009374	0.000538	0.021272	0.048051	0.005121	0.122411
30	0.056929	0.043446	0.196493	0.06851	0.043329	0.177203	0.006335	0.000117	0.017541	0.052937	0.004416	0.136688
Mean	0.081955	0.045302	0.187904	0.068066	0.043766	0.162189	0.007817	0.000309	0.031918	0.047847	0.005407	0.131473
STD	0.034049			0.005627			0.003877			0.010245		
P-value				0.031476			3.98E-17			2.22E-06		

Table 7.21 MOFDO, MODA, MOPSO, and NSGA-III (MMF-5 results).

Benchmarks	MMF5											
Algorithms	**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.058761	0.042854	0.266174	0.342665195	0.154994	0.577333	0.095127	0.034214	0.256507	0.184658	0.183084	0.185805
2	0.086339	0.043441	0.318752	0.254168582	0.11714	0.515963	0.263604	0.173663	0.326297	0.274538	0.272962	0.275806
3	0.061651	0.044167	0.157686	0.499170484	0.130816	0.63566	0.144129	0.09057	0.217503	0.094742	0.092775	0.096358
4	0.131904	0.042135	0.426281	0.330967362	0.11589	0.610961	0.077782	0.036762	0.113996	0.275856	0.273984	0.278548
5	0.070698	0.042954	0.276834	0.414153221	0.13925	0.654167	0.333817	0.178625	0.360566	0.455223	0.453092	0.457472
6	0.088216	0.040173	0.262326	0.295305827	0.124832	0.578512	0.13613	0.076685	0.192922	0.274763	0.273706	0.276276
7	0.068484	0.040065	0.306295	0.406784923	0.127121	0.567568	0.308921	0.23712	0.360597	0.185129	0.183688	0.187171
8	0.078508	0.044696	0.188402	0.309161957	0.134701	0.646964	0.249053	0.11679	0.287337	0.364239	0.363015	0.36528
9	0.07616	0.038362	0.278852	0.305869197	0.140587	0.634894	0.103315	0.051704	0.164701	0.274429	0.272822	0.275574
10	0.064971	0.042678	0.250401	0.358801254	0.126277	0.615132	0.097469	0.030599	0.160319	0.094455	0.093114	0.09595
11	0.093017	0.043814	0.313482	0.388912402	0.173199	0.574522	0.364099	0.245704	0.441107	0.184929	0.182925	0.186924
12	0.093082	0.039381	0.370593	0.296034848	0.112002	0.584467	0.309708	1.93E-01	0.359882	0.094439	0.092827	0.096245
13	0.121075	0.038609	0.487966	0.306677128	0.133427	0.706275	0.139453	0.074148	0.215529	0.18498	0.183105	0.187122
14	0.062287	0.03965	0.339246	0.291737053	0.132149	0.600693	0.338092	0.281924	0.379939	0.095989	0.093649	0.098882
15	0.104171	0.034676	0.267668	0.483651979	0.178373	0.651472	0.104	0.061414	0.155917	0.18509	0.183174	0.18659
16	0.075308	0.043184	0.264185	0.476073695	0.144073	0.694433	0.332599	0.215738	0.379982	0.094944	0.094041	0.096599
17	0.071016	0.038267	0.171571	0.426275075	0.110187	0.708735	0.122754	0.080754	0.174022	0.094962	0.09303	0.096498
18	0.068888	0.038849	0.262467	0.554533474	0.229398	1.254119	0.225924	0.149533	0.279771	0.184788	0.182939	0.187986
19	0.059978	0.040828	0.652225	0.252974873	0.092978	0.564398	0.122619	0.052407	0.178968	0.274026	0.272574	0.275515
20	0.082846	0.04442	0.288648	0.411192577	0.16042	0.720554	0.261088	0.138601	0.32102	0.364418	0.362625	0.366075
21	0.067433	0.037242	0.261327	0.376384672	0.157083	0.651709	0.091254	0.019521	0.147151	0.454569	0.452899	0.456095
22	0.085861	0.039768	0.196903	0.292150056	0.156221	0.457231	0.093497	0.029476	0.188755	0.364539	0.362935	0.365858
23	0.074119	0.040656	0.257869	0.462054289	0.228262	0.598162	0.363137	0.282083	0.408341	0.183844	0.182632	0.185045
24	0.11985	0.040002	0.239178	0.376544971	0.193593	0.742632	0.081727	0.030801	0.145194	0.094451	0.092729	0.096709
25	0.125667	0.041455	0.422414	0.488656436	0.135407	0.644192	0.321992	0.054635	0.367882	0.454731	0.452825	0.456468
26	0.073101	0.042642	0.220827	0.255850019	0.104231	0.583478	0.243095	0.204341	0.2697	0.274856	0.272794	0.277152
27	0.077393	0.04345	0.157248	0.312484605	0.115062	0.668592	0.09931	0.043391	0.190081	0.185067	0.183245	0.187019
28	0.071463	0.040843	0.202918	0.292852459	0.144876	0.649004	0.239066	0.157913	0.295083	0.184488	0.183127	0.185851
29	0.060165	0.038751	0.180081	0.366625361	0.128299	0.612893	0.407113	9.67E-02	0.464189	0.274975	0.27312	0.276677
30	0.077638	0.043109	0.328188	0.326223595	0.154354	0.625531	0.139507	0.092103	0.193982	0.454878	0.453482	0.456139
Mean	0.081668	0.041037	0.287233	0.365164586	0.143173	0.644342	0.206979	0.117701	0.266575	0.238766	0.237097	0.24052293
STD	0.020304			0.0818221			0.106891			0.119541		
P-value				6.57481E-26			4.21E-08			2.02E-09		

Table 7.22 MOFDO, MODA, MOPSO, and NSGA-III (MMF-6 results).

Benchmarks	MMF6											
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.061719	0.041446	0.186138	0.211677942	0.115977	0.589646	0.791503	0.352781	29.69857	0.20329	0.201724	0.204627
2	0.061891	0.045025	0.121301	0.409147066	0.131847	0.617439	12.87504	5.981841	13.3063	0.402303	0.400053	0.403813
3	0.058107	0.043749	0.137796	0.720307236	0.418557	1.337886	0.217459	0.077238	0.391947	0.601866	0.600108	0.604107
4	0.057401	0.0434	0.179202	0.704779087	0.198937	1.294991	0.83255	0.442638	1.761646	0.800414	0.798052	0.801798
5	0.05143	0.043358	0.114705	0.745277677	0.364153	1.450068	8.786474	3.766783	9.090508	0.799902	0.798561	0.80174
6	0.062477	0.044049	0.657506	0.224987714	0.112095	0.604905	0.61281	0.215423	0.814224	0.601331	0.59978	0.602862
7	0.069646	0.048162	0.141105	0.788831987	0.481106	1.268132	0.811991	0.356405	25.16252	0.79955	0.798415	0.800808
8	0.070074	0.03751	0.338529	0.352129598	0.137487	0.583561	1.208224	0.464601	41.23893	0.998992	0.997444	1.000717
9	0.065925	0.042528	0.105407	0.623082605	0.123966	1.279473	17.01873	12.24286	17.41893	0.999844	0.998292	1.002232
10	0.061536	0.04335	0.112416	0.257059264	0.129659	0.622848	0.552397	0.311971	0.865906	0.801761	0.800494	0.803534
11	0.063328	0.0418	0.112428	0.284495325	0.117068	0.621197	30.64494	20.19228	31.51784	0.602065	0.599098	0.603699
12	0.069486	0.043441	0.134558	0.282326009	0.126139	0.584288	3.772073	2.738782	4.026601	0.601538	0.599323	0.60273
13	0.064672	0.044613	0.12663	0.282547482	0.123516	0.57591	2.428836	1.442226	2.706455	0.999696	0.997117	1.001102
14	0.063433	0.044907	0.134274	0.629334868	0.17436	1.147905	4.18307	0.817082	4.713362	0.600537	0.599209	0.601707
15	0.05945	0.042832	0.138099	0.91607027	0.533465	1.643228	8.027372	5.367901	8.30296	0.799948	0.797039	0.801812
16	0.063659	0.042584	0.144464	0.86026168	0.197047	2.275823	16.83357	8.743896	17.59981	0.203306	0.201755	0.20491
17	0.055986	0.047117	0.162126	0.843637383	0.101479	1.479815	0.771846	0.400083	1.499057	0.799455	0.797563	0.800997
18	0.064585	0.04483	0.43336	0.239448791	0.096497	0.46103	0.499996	0.230723	0.79208	0.799938	0.797876	0.800857
19	0.068262	0.044564	1.364531	0.370701898	0.142039	0.592956	0.657861	0.322864	1.32208	0.402651	0.401007	0.40417
20	0.05891	0.043222	0.170584	0.382515164	0.120984	0.590596	4.786278	3.385939	60.80077	0.999065	0.997877	1.000835
21	0.06977	0.039186	0.917843	0.58445248	0.10415	1.269762	0.905559	0.489265	2.52698	0.602784	0.600464	0.604936
22	0.059752	0.0445	0.972598	0.398403927	0.205273	0.69911	3.844299	1.178836	4.131045	0.801481	0.799338	0.802933
23	0.072331	0.04543	0.195327	0.358309854	0.162596	0.517463	6.079134	5.030467	6.471165	0.998884	0.996633	1.000254
24	0.056392	0.044181	0.175074	0.701036796	0.4651	1.241912	4.669282	3.06695	31.03677	0.998813	0.997437	1.000094
25	0.064593	0.047843	0.119481	1.060530908	0.173945	1.809226	5.685612	3.299648	82.27643	0.203201	0.202232	0.204104
26	0.063388	0.045243	0.121893	1.162394093	0.627163	1.94929	3.632143	1.054373	4.092462	0.204313	0.20216	0.206696
27	0.066177	0.043433	1.08732	1.185392696	0.315548	2.431437	13.69369	10.44628	83.32127	0.999714	0.997706	1.001184
28	0.05464	0.04437	0.12788	0.239213624	0.096399	0.549224	5.00293	2.523308	5.375199	0.999248	0.997722	1.001319
29	0.06485	0.041853	0.091746	0.501063604	0.233823	0.708389	9.177092	7.433983	29.84103	0.601407	0.599803	0.602812
30	0.072077	0.037581	0.246099	0.940980475	0.13376	1.819145	17.21408	14.15701	32.34285	0.799745	0.797601	0.80099
Mean	0.063198	0.043537	0.302347	0.575346584	0.215471	1.087222	6.207228	3.884481	18.48152	0.700901	0.699063	0.702479
STD	0.005272			0.297528396			7.052953			0.265075		
P-value				2.64131E-13			1.28E-05			4.39E-19		

Table 7.23 MOFDO, MODA, MOPSO, and NSGA-III (MMF-7 results).

Benchmarks	MMF7											
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.133808	0.098323	0.192533	0.364220298	0.122128	0.431426	0.365338	0.131032	1.08539	0.264092	0.262732	0.265971
2	0.204403	0.09867	0.316447	0.328806164	0.088251	0.3843	0.384818	0.233493	1.859289	0.133536	0.132508	0.136179
3	0.152217	0.087591	0.310538	0.313982772	0.090091	0.399679	0.431478	0.123611	4.202138	0.524079	0.522824	0.525107
4	0.151322	0.091344	0.307859	0.242563531	0.090388	0.329969	0.184616	0.027966	0.344224	0.394161	0.391929	0.395171
5	0.169606	0.104195	0.278306	0.24819494	0.102042	0.336104	0.288463	0.137162	0.561741	0.263479	0.262468	0.264913
6	0.13455	0.083562	0.186136	0.319281247	0.136041	0.397335	0.472573	0.164611	0.937506	0.263702	0.262688	0.265135
7	0.127488	0.063146	0.281021	0.375195935	0.097099	0.442276	0.432481	0.173554	0.680845	0.524341	0.522945	0.525652
8	0.151414	0.083527	0.198764	0.341579073	0.113551	0.545394	0.404453	0.097405	0.630326	0.653979	0.652746	0.654794
9	0.13556	0.070911	0.198283	0.376968441	0.080033	0.415212	0.414012	0.14614	0.662442	0.524038	0.52281	0.525893
10	0.12726	0.077415	0.165052	0.251938004	0.126092	0.3208	0.291259	0.14381	0.398492	0.394017	0.392876	0.395125
11	0.178995	0.090592	0.257357	0.305557735	0.149827	0.347519	0.279403	0.101495	0.448688	0.523739	0.522735	0.524334
12	0.16461	0.080213	0.214004	0.284083932	0.147723	0.328508	0.297098	0.159913	0.466743	0.524049	0.52305	0.525495
13	0.176257	0.09567	0.298141	0.387325863	0.186205	0.613806	0.539703	0.15094	2.34792	0.13389	0.133035	0.135415
14	0.144432	0.078715	0.17526	0.290710299	0.139258	0.414848	0.477517	0.077103	0.709601	0.134114	0.132845	0.135186
15	0.129102	0.078764	0.286285	0.290118855	0.155626	0.348069	0.199559	0.042627	0.322566	0.654164	0.652449	0.655405
16	0.137834	0.093483	0.209884	0.380844563	0.105317	0.419731	0.206152	0.105015	0.333725	0.39345	0.392417	0.395262
17	0.116016	0.083435	0.184425	0.343469849	0.107328	0.413472	0.353907	0.113759	0.491962	0.394143	0.393255	0.397056
18	0.156725	0.083746	0.208571	0.296174986	0.071974	0.38763	0.307921	0.080539	0.449399	0.263721	0.262869	0.265086
19	0.128374	0.085021	0.177069	0.291524983	0.15067	0.448651	0.441219	0.298309	1.271128	0.524004	0.522787	0.527088
20	0.133902	0.079055	0.176588	0.449648124	0.132493	0.491965	0.416028	0.204969	0.596444	0.264101	0.263105	0.266219
21	0.154342	0.097643	0.244127	0.347269179	0.105374	0.404192	0.391153	0.125674	0.559614	0.264134	0.262871	0.265293
22	0.16441	0.098256	0.277289	0.345142191	0.101199	0.42566	0.206447	0.092041	0.334527	0.264384	0.263202	0.265357
23	0.161709	0.11106	0.272839	0.327483163	0.156429	0.408769	0.558678	0.102895	0.781256	0.52412	0.522636	0.526076
24	0.156768	0.083111	0.249155	0.329559665	0.124762	0.384105	0.344567	0.082891	0.64907	0.394505	0.393128	0.395914
25	0.196252	0.119912	0.357595	0.439384137	0.104513	0.495381	0.247317	0.026974	0.385195	0.264077	0.262377	0.265003
26	0.157791	0.096372	0.222808	0.378282666	0.123467	0.528854	0.352179	0.115613	0.541867	0.653791	0.65286	0.654667
27	0.120045	0.050052	0.246939	0.309182084	0.102406	0.416038	0.495896	0.31235	5.522353	0.523958	0.523116	0.525614
28	0.129504	0.078853	0.162853	0.320048618	0.08677	0.333877	0.366847	0.160405	0.497246	0.263659	0.262734	0.264464
29	0.134603	0.101708	0.191336	0.317643028	0.109758	0.376447	0.226339	0.080135	2.241369	0.524113	0.521643	0.525416
30	0.126885	0.068348	0.240619	0.335111988	0.151917	0.469699	0.464317	0.153697	0.682671	0.653761	0.652543	0.655429
Mean	0.14854	0.08709	0.236269	0.33104321	0.118624	0.415324	0.361391	0.132204	1.033191	0.402643	0.401406	0.404124
STD	0.021977			0.049375125			0.103699			0.163531		
P-value				5.34254E-26			8.12E-16			1.15E-11		

Table 7.24 MOFDO, MODA, MOPSO, and NSGA-III (MMF-8 results).

Benchmarks		MMF8										
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.256761	0.069952	0.999079	0.11998	0.046088	2.001654	0.032736	0.000217	0.518596	0.009825	0.004902	0.030312
2	0.063414	0	3.79104	0.120133	0.042998	2.04711	0.013714	0.002529	0.235209	0.010459	0.007177	0.022051
3	0.209949	0.107351	0.451057	0.136118	0.04582	5.122521	0.027413	0.005582	0.469995	0.008936	1.63E-05	0.013755
4	0.156378	0	4.378301	0.119577	0.061766	0.217931	0.047004	0.025953	0.460587	0.010432	0.006007	0.030546
5	0.251901	0.053052	0.872509	0.088139	0.04582	2.001351	0.010111	0.005027	0.061631	0.010997	0.007243	0.031802
6	0.012783	0	1.768225	0.120201	0.038749	1.998705	0.026759	4.87E-05	0.100412	0.009451	0.00597	0.015485
7	0.24665	0.056461	0.692334	0.120363	0.046229	2.018883	0.04834	0.000383	2.586445	0.010576	0.004776	0.036378
8	0.205224	0	3.240014	0.654201	0.04358	1.91393	0.042379	0.001395	0.265586	0.007834	0.00055	0.015426
9	0.270119	0.084097	3.621273	0.132047	0.045678	2.00856	0.013225	0.001711	0.041528	0.01058	0.004136	0.031561
10	0.037532	0	0.910817	0.087705	0.042833	0.349952	0.041157	0.000695	0.316319	0.00955	0.006381	0.020848
11	0.265063	0.065034	2.361753	0.145964	0.062814	11.15846	0.029985	0.003798	0.22922	0.010915	0.006202	0.024883
12	0.195461	0.058826	1.34531	0.107649	0.038619	2.017579	1.482583	0.002401	293.7408	NaN	0.010796	0.03013
13	0.05018	0	2.692286	0.145465	0.045115	5.392312	0.034793	0.000486	0.447438	0.009947	0.001597	0.022353
14	0.217414	0	1.737457	0.091528	0.046418	1.063855	0.075572	0.035477	0.418178	0.010538	0.008064	0.019642
15	0.140011	0.06458	1.416442	0.181511	0.038814	4.479774	0.019239	0.003408	0.502448	0.002132	8.23E-08	0.01526
16	0.229326	0.075649	0.704402	0.156965	0.045749	2.005064	0.025308	0.000437	0.190967	0.008858	0.001219	0.014523
17	0.219207	0.08357	0.862595	0.083022	0.046097	0.60878	0.018727	0.002349	0.280299	0.010058	0.002272	0.027307
18	0.06555	0	1.986759	0.118271	0.044619	2.015825	0.023785	0.016981	0.156484	0.011023	0.006623	0.033441
19	0.224005	0.086426	0.581109	0.113113	0.046044	3.00721	0.036512	0.00079	0.433535	0.009493	0.000926	0.013731
20	0.011202	0	1.782114	0.144591	0.038381	3.926212	0.056197	0.034468	0.223679	0.009827	0.004605	0.023892
21	0.258727	0.000211	0.744853	0.194541	0.043216	5.027089	0.011216	0.000829	0.055789	0.009919	0.007737	0.025192
22	0.063408	0	3.011561	0.112298	0.040497	2.618328	0.0349	0.003003	0.39859	0.010329	0.005134	0.021663
23	0.068767	0	1.34238	0.112368	0.042337	1.267595	0.024748	0.001813	1.055577	0.011533	0.005928	0.042556
24	0.081291	0	1.999493	0.127759	0.045867	2.474589	0.027247	0.002674	0.349331	0.009222	0.001783	0.01823
25	0.186683	0.087727	1.543292	0.155693	0.045783	5.008834	0.018514	0.000112	0.373072	0.018707	0.003779	0.056694
26	0.095966	0	1.998684	0.108287	0.050196	0.28392	0.019245	0.000142	0.412662	0.021287	0	0.116222
27	0.028385	0	1.969312	0.097261	0.045174	2.531623	0.068612	0.010051	0.252935	0.009779	0.004985	0.018608
28	0.229482	0.04996	0.654455	0.124774	0.046184	2.010187	0.051172	0.003363	0.511225	NaN	0.005385	0.009623
29	0.164993	0.054935	4.211855	0.095497	0.0421	1.578575	0.021187	0.002974	0.117821	0.008244	0.000101	0.013364
30	0.159301	0.032041	0.46967	0.131837	0.037972	1.99763	0.035 238	0.000108	0.471892	0.010365	0.001169	0.03483
Mean	0.155504	0.034329	1.804681	0.141562	0.045052	2.671801	0.080587	0.00564	10.18927	0.010386	0.004182	0.027677
STD	0.086999			0.100305			0.265287			0.003217		
P-value				0.567418			0.147032			3.59E-12		

Table 7.25 MOFDO, MODA, MOPSO, and NSGA-III (MMF-9 results).

Benchmarks	MMF9											
Algorithms	**MOFDO**				**MOPSO**		**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.4111	0.338753	1.527488	1.37838964	0.960585	1.720277	0.051557	0.001534	0.154504	NaN	0.003235	0.003495
2	0.415428	0.365296	1.761888	1.348849835	0.576741	2.28032	0.039018	0.002082	0.341168	NaN	0.004193	0.004502
3	0.429423	0.299619	0.804695	1.269031995	0.740967	1.9881	0.036812	0.003442	0.156217	NaN	0.003378	0.003386
4	0.434613	0.332124	1.861649	1.271834817	0.890767	2.24864	0.046917	0.001555	0.171651	NaN	0.003171	0.00382
5	0.473714	0.319339	2.846362	1.25613177	0.679745	1.793378	0.040971	0.000464	0.14717	NaN	0.004296	0.005378
6	0.441396	0.336329	1.995687	1.259712252	0.938811	1.688139	0.040142	0.003812	0.394685	NaN	0.002223	0.002615
7	0.423083	0.324146	1.187994	1.602358803	1.191858	1.983915	0.065733	0.001798	0.236487	NaN	0.004721	0.007085
8	0.381137	0.334123	1.463316	1.182892498	0.589616	2.212253	0.035395	0.001161	0.298707	NaN	0.002357	0.003297
9	0.362528	0.278039	1.882721	1.384526177	0.797484	2.225364	0.03523	9.63E-05	0.275358	NaN	0.003887	0.003887
10	0.389446	0.274704	1.111939	1.096748431	0.690815	1.975258	0.03952	0.003157	0.191937	NaN	NaN	NaN
11	0.509932	0.389881	2.800362	1.648207002	1.076774	2.41306	0.033569	0.004718	0.353711	1.251885	0	1.671873
12	0.294365	0.229237	1.193053	1.398831218	0.953363	2.064759	0.066068	0.003107	0.34443	NaN	NaN	NaN
13	0.539985	0.368595	2.701657	1.36892283	0.6555	2.560592	0.040128	0.004323	0.726787	0.925744	0	1.228788
14	0.443836	0.259045	3.267564	1.238349943	0.880946	1.819613	0.033704	0.002722	0.177715	NaN	0.002514	0.003042
15	0.522454	0.200433	2.065635	1.376774354	0.592099	2.382176	0.037558	0.004151	0.482708	NaN	0.003089	0.003773
16	0.359737	0.31622	1.115081	1.264295494	0.64797	2.260717	0.089715	0.00486	0.322682	NaN	0.003429	0.00384
17	0.314988	0.243888	1.492403	1.232524066	0.727256	1.941673	0.063497	0.008585	0.445918	NaN	NaN	NaN
18	0.670327	0.461876	2.221537	1.272816529	0.623218	1.728147	0.036478	0.001998	0.109311	1.118029	0	1.831661
19	0.594567	0.490677	1.37475	1.228233513	0.649261	1.654772	0.040328	0.001297	0.553845	NaN	NaN	NaN
20	0.462744	0.391495	1.093302	1.240510593	0.638363	2.294667	0.043578	0.003172	0.295183	NaN	NaN	NaN
21	0.282593	0.181035	1.070801	1.102848726	0.862488	1.688865	0.032693	0.000205	0.190976	NaN	NaN	NaN
22	0.665793	0.502496	1.391053	1.596572337	1.342856	1.857835	0.042656	0.004645	0.352363	NaN	NaN	NaN
23	0.56703	0.324771	2.450823	1.470503263	1.022883	2.120174	0.039196	0.001295	0.207795	NaN	0.003179	0.003179
24	0.461532	0.240645	1.349616	1.299268425	0.683282	2.453327	0.049447	8.67E-05	0.387086	NaN	0.002229	0.0026
25	0.378211	0.316749	0.599627	1.515688342	0.674426	2.14423	0.071434	0.004329	0.726065	NaN	0.003365	0.004518
26	0.512082	0.317171	2.80193	1.433426836	0.832502	1.756562	0.17379	0.074978	0.6518	0.559126	0	0.799146
27	0.808148	0.487434	1.666529	1.521907646	0.638626	2.014889	0.043871	0.002545	0.337229	NaN	0.004598	0.004826
28	0.486577	0.425009	2.405601	1.341004855	0.579045	2.104603	0.038867	0.003404	0.202279	NaN	0.003968	0.003968
29	0.699918	0.477401	1.330119	1.188347962	0.639461	2.057927	0.078225	0.003391	0.150576	NaN	0.001939	0.003576
30	0.459694	0.38596	1.44725	1.193166617	0.601243	2.191492	0.032201	0.002927	0.273468	NaN	0.00388	0.005644
Mean	0.473213	0.340416	1.742748	1.332755892	0.779298	2.054191	0.05061	0.005195	0.321994	0.963696	0.002767	0.243822
STD	0.121966			0.142753053			0.02749			0.301103		
P-value				8.12705E-33			5.09E-26			5.83E-07		

Table 7.26 MOFDO, MODA, MOPSO, and NSGA-III (MMF-10 results).

Benchmarks	MMF10											
Algorithms	**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.618851	0.305382	1.209258	1.282466539	0.709195	1.602153	0.056622	0.00368	0.302445	NaN	0.001977	0.002519
2	0.487591	0.321791	1.053768	0.832315286	0.740362	1.424847	0.098738	0.003657	0.43841	7.193954	0	10.6601
3	0.319867	0.208944	0.865587	1.094142406	0.685271	1.405454	0.068218	0.001718	0.373974	NaN	0.003252	0.003843
4	0.565169	0.491312	1.464831	1.277681059	0.728985	1.487211	0.065729	0.003452	0.498293	NaN	0.002175	0.002574
5	0.323806	0.242983	0.879524	1.067443879	0.804283	1.511369	0.049923	0.003823	0.376988	NaN	0.00325	0.00325
6	0.55497	0.346104	1.037213	0.942882102	0.698973	1.444199	0.107634	0.001845	0.594766	0.598872	0	0.714941
7	0.442292	0.314469	1.05364	0.870338438	0.678265	1.497459	0.081421	0.003772	0.500905	NaN	0.003232	0.003232
8	0.466287	0.33186	1.150756	0.973150274	0.687746	1.461834	0.103403	0.003005	0.200422	NaN	0.002635	0.002877
9	0.255233	0.177556	0.667876	0.947158906	0.696797	1.458209	0.15985	0.008131	0.392424	NaN	0.003973	0.003973
10	0.25177	0.194186	1.008037	1.002488177	0.783166	1.417179	0.104594	0.008203	0.369059	NaN	0.00322	0.004395
11	0.373143	0.250061	0.906406	0.830282153	0.719324	1.437384	0.113303	0.002611	0.306373	NaN	0.002431	0.002585
12	0.444275	0.287291	1.363992	0.989264838	0.665376	1.52615	0.143665	0.007704	0.528136	NaN	0.002186	0.002944
13	0.434638	0.42003	0.983239	1.116187193	0.692962	1.454358	0.138874	0.004772	0.386394	NaN	0.004096	0.004096
14	0.344719	0.268354	0.495839	0.807575372	0.71968	1.433654	0.090778	0.002326	0.389693	NaN	NaN	NaN
15	0.415382	0.292125	1.423748	1.052150083	0.614803	1.753753	0.175985	0.002372	0.622802	NaN	0.00267	0.00337
16	0.516017	0.247435	1.587995	1.13466267	0.73082	1.64102	0.1385	0.005971	0.422472	NaN	0.003599	0.003599
17	0.7651	0.426241	1.37658	1.078556955	0.628474	1.730958	0.120239	0.003555	0.508095	NaN	NaN	NaN
18	0.364528	0.270977	0.609703	0.866625657	0.654699	1.497805	0.084149	0.003903	0.176513	NaN	0.002366	0.003127
19	0.560456	0.308841	1.159543	0.825774889	0.621046	1.507994	0.058618	0.003071	0.498436	NaN	0.002793	0.002793
20	0.614157	0.496819	1.471247	0.795106526	0.731676	1.471101	0.058102	0.003502	0.329018	NaN	0.003419	0.003419
21	0.320528	0.268243	0.845071	1.025141579	0.722042	1.441835	0.035525	0.002812	0.177415	NaN	NaN	NaN
22	0.589331	0.475753	1.257925	1.285031982	0.769486	1.439002	0.071549	0.006486	0.680747	NaN	0.002499	0.002499
23	0.522394	0.409689	1.008844	0.825064986	0.73503	1.379532	0.108981	0.002274	0.234099	NaN	0.003485	0.003485
24	0.609984	0.380296	1.430632	0.872747781	0.71644	1.394758	0.046456	0.003927	0.343982	NaN	0.003952	0.003952
25	0.400151	0.358158	0.974749	0.862046856	0.722851	1.404882	0.032766	0.003038	0.180131	NaN	0.002944	0.002944
26	0.399128	0.211734	1.103546	0.873186141	0.681142	1.468642	0.077603	0.004023	0.523442	NaN	0.002528	0.002528
27	0.399352	0.276924	1.12088	1.061081373	0.574966	1.528795	0.125445	0.003983	0.399904	NaN	NaN	NaN
28	0.289697	0.258498	1.014888	1.080672835	0.718244	1.469521	0.040324	0.004069	0.232033	NaN	0.002958	0.002958
29	0.327926	0.268528	1.456195	1.034432863	0.648346	1.487937	0.043576	0.00328	0.370919	NaN	0.005543	0.007938
30	0.285609	0.202884	1.085655	1.310809486	0.73654	1.689883	0.104709	0.002961	0.320753	NaN	0.003543	0.003543
Mean	0.442078	0.310449	1.102239	1.000548976	0.700566	1.495629	0.090176	0.003931	0.389301	3.896413	0.002874	0.440672
STD	0.127789			0.154296479			0.038557			4.663427		
P-value				5.57562E-22			7.3E-21			5.74E-06		

Table 7.27 MOFDO, MODA, MOPSO, and NSGA-III (MMF-11 results).

Benchmarks	MMF11											
Algorithms	**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.059602	0.031635	0.338	1.254312396	0.599874	2.25025	0.096379	0.003413	0.524971	NaN	0.003551	0.003642
2	0.107179	0.064576	0.290001	1.056786916	0.702326	1.720209	0.169852	0.013981	0.565111	NaN	0.004718	0.004718
3	0.073419	0.050057	0.230972	0.985383786	0.628495	2.126213	0.048799	0.004429	0.259977	NaN	0.00213	0.0042
4	0.080857	0.070625	0.219346	1.312654164	0.56738	2.409353	0.098291	0.004327	0.741973	NaN	0.002259	0.004961
5	0.073047	0.045769	0.235828	1.118695623	0.624748	2.029237	0.079824	0.004763	0.623726	NaN	0.004181	0.004286
6	0.092138	0.063506	0.287841	1.504312032	0.963869	2.102595	0.057683	0.004312	0.419128	NaN	0.003514	0.004382
7	0.118914	0.075636	0.213125	1.311073033	0.643857	1.696592	0.055088	0.003665	0.887333	0.772799	0	1.002184
8	0.077763	0.059651	0.242102	1.395408721	0.782576	2.361028	0.06515	0.001563	0.262334	NaN	0.003044	0.004234
9	0.091232	0.054297	0.379903	1.627949215	0.809694	2.811092	0.158469	0.00442	0.484819	NaN	0.003141	0.003141
10	0.075754	0.053699	0.232302	1.503586376	0.581816	1.743087	0.108614	0.002661	0.425002	1.91995	0.003627	3.582788
11	0.096209	0.061463	0.253463	1.429653025	0.994763	2.462224	0.131484	0.004504	0.570466	NaN	NaN	NaN
12	0.133065	0.056347	0.278398	1.294992032	0.630715	2.392906	0.075978	0.004403	0.293995	NaN	0.004262	0.004262
13	0.106016	0.087676	0.273579	1.325197655	0.55696	2.10815	0.064638	0.003665	0.300246	0.42102	0.003935	0.653119
14	0.057793	0.041542	0.236582	1.185627153	0.567032	2.450962	0.057333	0.004336	0.238862	NaN	0.003868	0.004893
15	0.107432	0.070917	0.246849	1.174917039	0.598677	1.803766	0.058907	0.001873	0.507086	NaN	0.003682	0.005198
16	0.086704	0.078623	0.197928	1.327922676	0.722575	2.58331	0.053672	0.003268	0.199384	NaN	0.002981	0.003536
17	0.105115	0.059548	0.389692	1.243822335	0.631838	2.261931	0.079123	0.00481	0.252678	NaN	0.003459	0.003979
18	0.126726	0.105001	0.349186	1.19409982	0.582535	2.354158	0.058475	0.004039	0.408949	NaN	0.004119	0.004119
19	0.094428	0.05458	0.263377	1.149284086	0.611635	2.240894	0.159135	0.00425	0.622378	NaN	NaN	NaN
20	0.080393	0.071527	0.204152	1.41035396	0.619574	2.14358	0.253974	0.007604	0.842439	NaN	0.004599	0.004897
21	0.052122	0.033036	0.169058	1.298353057	0.658861	2.620545	0.155238	0.003661	0.714036	0.847812	0	3.193843
22	0.093935	0.048356	0.258794	1.476103571	0.979187	2.406818	0.063507	0.003182	0.527941	NaN	0.00439	0.00439
23	0.110667	0.067949	0.381803	1.449054314	0.585112	2.37246	0.072206	0.003649	0.311601	NaN	NaN	NaN
24	0.095355	0.066607	0.226959	1.180471289	0.567304	2.185119	0.048759	0.004254	0.306995	NaN	0.004479	0.005915
25	0.066524	0.058619	0.344592	1.532896233	1.108946	2.182825	0.067342	0.004663	0.43252	NaN	0.002933	0.004487
26	0.091189	0.072954	0.193247	1.336453511	0.631029	2.018952	0.100047	0.00262	0.375853	NaN	0.006017	0.006017
27	0.089503	0.060775	0.200757	1.201124925	0.510034	2.628413	0.049645	0.004145	0.371306	NaN	0.002876	0.004433
28	0.087771	0.057744	0.356398	1.549789939	0.813811	2.247439	0.048785	0.004088	0.713417	1.941347	0	2.472647
29	0.123408	0.076256	0.322225	1.379551922	0.640831	2.328841	0.192589	0.003302	0.705061	NaN	0.006992	0.006992
30	0.12382	0.107637	0.260519	1.026894705	0.626438	2.075783	0.058417	0.002595	0.999315	NaN	NaN	NaN
Mean	0.092603	0.063554	0.269233	1.30789085	0.68475	2.237291	0.092913	0.004215	0.496297	1.180586	0.003414	0.423125
STD	0.020985			0.162286416			0.051555			0.703453		
P-value				2.44568E-44			0.975721			1.36E-10		

Table 7.28 MOFDO, MODA, MOPSO, and NSGA-III (MMF-12 results).

Benchmarks	MMF12											
Algorithms		**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.094034	0.072932	0.283411	0.177184642	0.04918	0.29023	0.033673	0.001752	0.261168	0.394092	0.392732	0.395971
2	0.076136	0.046601	0.187694	0.178265296	0.073143	0.473134	0.022613	0.001437	0.174182	0.133536	0.132508	0.136179
3	0.120569	0.082271	0.284328	0.129550688	0.071434	0.302951	0.046371	0.001123	0.269776	0.264079	0.262824	0.265107
4	0.0784	0.047299	0.287229	0.124929016	0.095559	0.350331	0.038241	0.001798	0.179259	0.134161	0.131929	0.135171
5	0.077388	0.045773	0.219324	0.099253275	0.033209	0.245602	0.03335	0.006019	0.130941	0.523479	0.522468	0.524913
6	0.075586	0.062679	0.200487	0.197062361	0.091518	0.442456	0.0501	0.009072	0.222005	0.393702	0.392688	0.395135
7	0.116129	0.055935	0.262551	0.161447887	0.104063	0.463231	0.022757	0.002591	0.097472	0.394341	0.392945	0.395652
8	0.030482	0.015498	0.411329	0.115018337	0.050533	0.311217	0.039823	0.00518	0.127493	0.263979	0.262746	0.264794
9	0.104052	0.056944	0.238145	0.143105552	0.079706	0.397347	0.03	0.000696	0.198793	0.654038	0.65281	0.655893
10	0.107935	0.059977	0.215872	0.109540409	0.053583	0.27232	0.040288	0.005734	0.108708	0.394017	0.392876	0.395125
11	0.07859	0.068541	0.416901	0.124838226	0.066093	0.258645	0.035019	0.005806	0.176225	0.393739	0.392735	0.394334
12	0.130521	0.053913	0.376515	0.135534275	0.041048	0.295861	0.02525	0.004864	0.120652	0.134049	0.13305	0.135495
13	0.076668	0.061044	0.346289	0.130177758	0.067829	0.259768	0.041207	0.004022	0.151965	0.26389	0.263035	0.265415
14	0.088337	0.078889	0.217308	0.156362174	0.065275	0.549595	0.039381	0.003926	0.186498	0.264114	0.262845	0.265186
15	0.070353	0.048717	0.360934	0.156735583	0.08047	0.29343	0.057203	0.009095	0.219385	0.394164	0.392449	0.395405
16	0.099676	0.054747	0.31209	0.152562031	0.056602	0.367776	0.024952	0.003744	0.148678	0.26345	0.262417	0.265262
17	0.073429	0.0486	0.390026	0.134881786	0.09178	0.417213	0.034198	0.000263	0.156604	0.394143	0.393255	0.397056
18	0.065239	0.040206	0.190702	0.142023949	0.056481	0.354752	0.026605	0.000302	0.105913	0.393721	0.392869	0.395086
19	0.090947	0.057634	0.339478	0.12038323	0.061567	0.219134	0.024955	0.002464	0.128545	0.134004	0.132787	0.137088
20	0.05793	0.051355	0.216924	0.119646071	0.040523	0.28409	0.044293	0.00599	0.24599	0.394101	0.393105	0.396219
21	0.086032	0.051452	0.355322	0.133474694	0.058488	0.295362	0.059517	0.00554	0.292704	0.654134	0.652871	0.655293
22	0.092846	0.065137	0.310191	0.123385101	0.074917	0.266787	0.033666	0.002695	0.198234	0.134384	0.133202	0.135357
23	0.060309	0.037635	0.163657	0.133494384	0.093135	0.368435	0.038701	0.000342	0.262557	0.13412	0.132636	0.136076
24	0.075674	0.059132	0.257473	0.094482676	0.065064	0.164125	0.068022	0.009908	0.220289	0.264505	0.263128	0.265914
25	0.070957	0.049263	0.56661	0.137206524	0.053773	0.411543	0.037474	0.00249	0.18729	0.524077	0.522377	0.525003
26	0.065734	0.054825	0.25985	0.142456675	0.04056	0.419692	0.01815	9.65E-05	0.194494	0.263791	0.26286	0.264667
27	0.075982	0.048729	0.19559	0.162938244	0.097038	0.530787	0.023093	0.000774	0.109691	0.523958	0.523116	0.525614
28	0.099096	0.088185	0.258773	0.128837398	0.037161	0.431913	0.027312	0.003135	0.095122	0.653659	0.652734	0.654464
29	0.048377	0.039149	0.26883	0.101339597	0.064683	0.230783	0.032196	0.001163	0.342089	0.524113	0.521643	0.525416
30	0.106989	0.059281	0.311158	0.12946231	0.086856	0.280133	0.049927	0.003037	0.211948	0.263761	0.262543	0.265429
Mean	0.083147	0.055411	0.290166	0.136519338	0.066709	0.341621	0.036611	0.003502	0.184156	0.350643	0.349406	352124
STD	0.021728			0.023738586			0.011901			0.16131		
P-value				9.63988E-13			1.07E-14			1.32E-12		

Table 7.29 MOFDO, MODA, MOPSO, and NSGA-III (ZDT-1 results).

Benchmarks					ZDT1							
Algorithms		**MOFDO**			**MOPSO**			**NSGA-III**			**MODA**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.095906	0.000305	2.332651	0.071801	0.044633	0.454482	0.125935	0.007972	1.716367	0.086496	0.02929	0.72583
2	0.150976	0.001932	1.896392	0.090403	0.044618	1.168066	0.147654	0.005232	1.008278	0.064075	0.042738	0.510061
3	0.112966	0.005483	9.020892	0.067558	0.044612	0.326032	0.188652	0.011178	1.488623	0.089566	0.035293	0.283382
4	0.064871	3.70E-05	5.817909	0.075969	0.044987	0.845379	NaN	0.107994	10.63249	0.05981	0.045328	0.747263
5	0.076199	0.003378	3.320255	0.084101	0.044601	1.524927	NaN	0	5.365275	0.090505	0.047708	1.113317
6	0.076848	0.000321	4.063877	0.081037	0.043986	1.893228	NaN	0.005483	0.146394	0.077057	0.043958	0.408467
7	0.049821	0.00016	4.831265	0.074641	0.044666	0.954364	0.078234	0.014952	0.491901	0.065238	0.04156	0.850783
8	0.061922	0.002912	6.26773	0.111134	0.045022	2.067354	NaN	0	3.762881	0.064923	0.043081	0.389522
9	0.030973	0.001159	1.023093	0.069507	0.044577	0.59074	0.53198	0.005315	5.250496	0.060266	0.038642	0.377993
10	0.12629	9.51E-05	5.49958	0.078496	0.044471	1.238959	1.255884	0.010664	7.795809	0.094819	0.058475	0.266339
11	0.046288	0.00071	1.223747	0.081194	0.044805	1.382584	0.296814	0.016343	3.481644	0.059168	0.038421	0.253582
12	0.063513	0.008419	0.913488	0.078018	0.044635	1.325632	NaN	0.005698	1.342221	0.068923	0.043162	0.251063
13	0.059233	0.001949	0.594096	0.070071	0.044375	0.825328	1.290865	0.00432	4.608374	0.093885	0.031484	2.794284
14	0.092026	0.003685	2.69787	0.085997	0.044522	2.614217	NaN	0.00444	0.703095	0.086784	0.036175	0.186439
15	0.068561	0.000408	3.066398	0.073943	0.04365	0.609401	NaN	0	8.221919	0.08031	0.045459	0.163294
16	0.022768	0.001731	1.184329	0.070866	0.04488	0.999956	0.208869	0.016748	0.739149	0.073511	0.034865	1.54402
17	0.041898	3.53E-05	0.455089	0.074317	0.044741	1.044855	NaN	0	5.381639	0.095419	0.038975	0.214632
18	0.055883	0.002756	0.718342	0.068519	0.044556	0.521133	NaN	0	2.891242	0.081922	0.044517	0.457868
19	0.00976	0.000101	0.253659	0.077176	0.044856	1.629952	1.537413	0.017998	7.504958	0.059562	0.036238	0.209232
20	0.065845	0.000581	0.671035	0.083594	0.044893	1.417402	0.235266	0.043783	2.768782	0.055525	0.043146	0.282797
21	0.09016	0.003892	2.7548	0.072895	0.044463	0.572484	NaN	0.003918	2.453508	0.07075	0.043135	0.344303
22	0.058417	0.000225	0.995189	0.079486	0.045074	1.32006	NaN	0	12.98886	0.069574	0.040521	0.612493
23	0.06267	9.28E-05	0.654582	0.082817	0.044914	0.704814	0.473829	0.018586	3.317437	0.075923	0.042492	0.570781
24	0.062653	0.000428	5.215003	0.080058	0.044726	1.410369	0.441546	0.008306	3.28092	0.085008	0.046063	1.720382
25	0.090314	0.001165	1.165902	0.077587	0.045118	0.827646	0.122661	0.007846	0.383681	0.077705	0.04069	0.528092
26	0.087982	4.45E-05	4.854654	0.068981	0.044679	0.596138	0.435321	0.023729	1.620827	0.073477	0.057296	0.663251
27	0.018243	0.002306	0.22163	0.073862	0.043947	0.423517	NaN	0.004306	0.732666	0.072767	0.043685	0.29726
28	0.087896	0.011213	1.041551	0.092138	0.044167	1.613036	1.432437	0.013658	7.774643	0.088803	0.035271	0.221892
29	0.056473	0.000134	4.426559	0.08334	0.044324	2.467209	NaN	0.020463	2.038046	0.08901	0.038894	0.21126
30	0.040068	6.15E-05	1.278431	0.073511	0.044789	0.983175	0.13864	0.024418	0.878968	0.085242	0.056189	0.619615
Mean	0.067581	0.001857	2.615333	0.078434	0.04461	1.145081	0.526	0.013445	3.69237	0.076534	0.042092	0.593983
STD	0.030912			0.008848			0.509184			0.012072		
P-value				0.069585			1.06E-05			0.144885		

Table 7.30 MOFDO, MODA, MOPSO, and NSGA-III (ZDT-2 results).

Benchmarks	ZDT2											
Algorithms	**MOFDO**			**MOPSO**			**MODA**			**NSGA-III**		
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.033001	0.019219	0.041957	0.032342	0.021137	0.060411	0.003166	6.91E-05	0.014258	0.155678	0.098433	0.095336
2	0.03874	0.020394	0.044328	0.030932	0.021395	0.072878	0.002821	0.000112	0.018962	0.533972	0.463916	0.461206
3	0.035094	0.019822	0.049258	0.032884	0.020885	0.064323	0.002859	0.000425	0.012101	0.12373	0.062266	0.059831
4	0.027452	0.018329	0.041983	0.032064	0.019625	0.073347	0.002713	0.000141	0.008126	0.215951	0.145316	0.142744
5	0.032822	0.021326	0.047888	0.032924	0.021424	0.061667	0.002964	8.49E-05	0.014985	0.126019	0.067316	0.064437
6	0.029899	0.021921	0.054927	0.034001	0.019954	0.06915	0.002544	0.000564	0.006819	0.120794	0.054188	0.052207
7	0.037473	0.020897	0.052816	0.031557	0.021523	0.049681	0.00285	8.45E-05	0.008382	0.102659	0.055827	0.053062
8	0.039427	0.01933	0.049686	0.033401	0.021098	0.068928	0.002994	0.00034	0.009266	1.134137	1.068203	1.065549
9	0.037528	0.021745	0.072632	0.032365	0.020979	0.060105	0.003422	0.000222	0.011881	0.090752	0.03407	0.030869
10	0.030391	0.019632	0.042778	0.032899	0.021128	0.061636	0.002586	0.000687	0.017691	0.08633	0.02728	0.025381
11	0.037504	0.024494	0.04934	0.033853	0.021291	0.064845	0.003033	7.99E-05	0.016146	0.111973	0.050161	0.047207
12	0.043167	0.026227	0.060016	0.032007	0.019578	0.06444	0.00257	0.000707	0.006514	0.15742	0.095551	0.093688
13	0.033336	0.019408	0.057537	0.032948	0.020716	0.07391	0.003389	0.000254	0.010456	0.097715	0.027194	0.024059
14	0.035647	0.019791	0.043864	0.030753	0.022147	0.071175	0.002801	0.000152	0.007484	0.111291	0.042918	0.040269
15	0.029719	0.019366	0.040186	0.032429	0.021675	0.063516	0.002746	0.000623	0.016966	0.137896	0.077127	0.075004
16	0.036384	0.027513	0.065836	0.031976	0.02098	0.071421	0.003482	0.000138	0.013174	1.165827	1.097888	1.094544
17	0.038633	0.024547	0.042021	0.033391	0.021605	0.062317	0.002841	0.000166	0.006626	0.096229	0.036753	0.034078
18	0.035078	0.020959	0.045873	0.032547	0.020829	0.052128	0.002744	0.000174	0.009272	0.264213	0.214829	0.212259
19	0.030438	0.018358	0.053865	0.031508	0.020921	0.0684	0.002629	0.000918	0.005001	1.091015	1.025245	1.023533
20	0.039939	0.01976	0.064585	0.030111	0.021561	0.07217	0.003216	7.63E-05	0.012937	0.128137	0.059183	0.056044
21	0.035261	0.021004	0.04544	0.03247	0.021229	0.070549	0.002611	0.000412	0.022713	0.205	0.137062	0.134863
22	0.034703	0.021778	0.057045	0.03236	0.022647	0.066693	0.002733	0.000125	0.010321	0.515002	0.451041	0.448433
23	0.036676	0.020868	0.049758	0.032903	0.021205	0.082562	0.00278	8.34E-05	0.009552	7.101404	7.021622	7.018925
24	0.035474	0.020001	0.061188	0.032202	0.021478	0.082969	0.003076	5.41E-05	0.009422	0.216802	0.13691	0.133888
25	0.028154	0.020662	0.043307	0.032804	0.02136	0.073525	0.003025	7.76E-05	0.010176	0.270743	0.200243	0.197295
26	0.036825	0.018876	0.045834	0.032352	0.021202	0.068492	0.003204	0.000113	0.01489	0.427387	0.362099	0.359009
27	0.033602	0.017292	0.046355	0.032222	0.020832	0.075538	0.002709	0.000345	0.010984	0.147257	0.074427	0.072063
28	0.037918	0.020269	0.063828	0.032379	0.021959	0.083193	0.003032	0.000112	0.007913	3.401023	3.320862	3.317941
29	0.042966	0.020801	0.058205	0.034573	0.021763	0.075649	0.003142	3.25E-05	0.013918	0.125954	0.053448	0.050338
30	0.030081	0.019049	0.055548	0.03198	0.021908	0.060392	0.00311	0.000129	0.011774	0.632812	0.57553	0.572549
Mean	0.035111	0.020788	0.051596	0.032438	0.021201	0.0682	0.002926	0.00025	0.011624	0.636504	0.57123	0.568554
STD	0.004045			0.000931			0.00026			1.38126		
P-value				0.000828			5.74E-46			0.020381		

Table 7.31 MOFDO, MODA, MOPSO, and NSGA-III (ZDT-3 results).

Benchmarks						ZDT3					
Algorithms		**MOFDO**			**MOPSO**			**NSGA-III**			**MODA**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.062276	0.005397	1.638356	0.085318	0.045134	2.357094	0.122092	0.120732	0.123971	0.054882	0.045501	0.171901
2	0.042995	0.000532	0.707233	0.067135	0.044257	0.361286	0.239536	0.238508	0.242179	0.055465	0.043476	0.31941
3	0.082922	0.003137	1.041326	0.077204	0.041186	0.871293	0.299079	0.297824	0.300107	0.078674	0.041709	0.265703
4	0.080187	0.000927	0.901428	0.071086	0.04429	0.917109	0.299161	0.296929	0.300171	0.07959	0.047118	0.327203
5	0.084004	0.001264	2.182243	0.077358	0.04453	0.664552	0.239479	0.238468	0.240913	0.074442	0.044708	0.839044
6	0.087514	0.012478	2.574528	0.068958	0.044622	0.518438	0.298702	0.297688	0.300135	0.064073	0.037728	2.801742
7	0.033804	0.001044	0.489138	0.074615	0.044206	0.810917	0.299341	0.297945	0.300652	0.096792	0.040271	1.214681
8	0.047873	3.05E-05	0.609472	0.082693	0.044119	0.857704	0.180979	0.179746	0.181794	0.099995	0.036202	3.119884
9	0.075602	6.59E-05	6.830801	0.076954	0.044206	0.749217	0.122038	0.12081	0.123893	0.054245	0.039564	0.214655
10	0.086877	0.001962	2.14927	0.076421	0.044414	0.653783	0.181017	0.179876	0.182125	0.073252	0.04088	0.370338
11	0.07515	7.79E-05	4.643909	0.090015	0.044676	1.015118	0.239739	0.238735	0.240334	0.06996	0.042678	0.560836
12	0.069194	0.001464	3.068494	0.088406	0.04526	2.867498	0.063049	0.06205	0.064495	0.079971	0.025527	1.183478
13	0.02444	0.000166	0.823825	0.080483	0.044927	1.6188	0.23989	0.239035	0.241415	0.093702	0.041853	0.28572
14	0.028535	0.001213	1.083881	0.077132	0.045018	1.217994	0.181114	0.179845	0.182186	0.051946	0.035095	0.422097
15	0.074316	0.0007	1.47028	0.073251	0.04485	0.464276	0.240164	0.238449	0.241405	0.071451	0.035712	0.186248
16	0.105943	0.000526	5.081081	0.074951	0.044144	0.442124	0.23945	0.238417	0.241262	0.074204	0.046066	0.820638
17	0.072504	0.000391	6.67632	0.073677	0.041744	0.817704	0.063143	0.062255	0.066056	0.069258	0.045247	0.241254
18	0.059607	0.000419	2.027879	0.072099	0.04384	1.024658	0.121721	0.120869	0.123086	0.105552	0.03808	2.894187
19	0.057569	0.000661	0.14492	0.078266	0.045584	0.639641	0.181004	0.179787	0.184088	0.086095	0.046741	2.021603
20	0.095927	0.003441	3.908949	0.074714	0.045086	0.555842	0.240101	0.239105	0.242219	0.057035	0.042088	0.148421
21	0.086174	9.72E-05	0.366441	0.070516	0.044553	0.39429	0.299134	0.297871	0.300293	0.06682	0.033098	0.724456
22	0.051627	5.79E-05	0.208771	0.081296	0.042891	0.875057	0.299384	0.298202	0.300357	0.072158	0.040466	0.542876
23	0.088098	0.000412	0.746823	0.078904	0.045251	2.09277	0.06312	0.061636	0.065076	0.077414	0.037243	0.188142
24	0.033568	0.001909	0.468259	0.088158	0.044509	1.559797	0.240505	0.239128	0.241914	0.092328	0.037597	1.073395
25	0.048982	6.05E-05	1.021408	0.078389	0	1.509653	0.122077	0.120377	0.123003	0.089554	0.041122	0.496605
26	0.099393	0.00192	2.301644	0.073405	0.043275	0.984219	0.121791	0.12086	0.122667	0.073132	0.039082	0.215847
27	0.035959	7.80E-05	1.039951	0.073121	0.043701	0.973632	0.062958	0.062116	0.064614	0.070867	0.03952	0.247191
28	0.110529	0.001823	7.889509	0.078651	0.044762	0.682168	0.239659	0.238734	0.240464	0.093977	0.042904	0.67443
29	0.046344	0.000115	3.568416	0.085264	0.044215	1.191244	0.181113	0.178643	0.182416	0.077429	0.034621	0.912832
30	0.054982	0.001136	0.954705	0.079206	0.044565	1.377525	0.121761	0.120543	0.123429	0.091838	0.043635	1.316867
Mean	0.066763	0.00145	2.220642	0.077588	0.042794	1.035513	0.194743	0.193506	0.196224	0.076537	0.040184	0.826723
STD	0.023914			0.005755			0.080044			0.014412		
P-value				0.019119			1.36E-11			0.060132		

Table 7.32 MOFDO, MODA, MOPSO, and NSGA-III (ZDT-4 results).

Benchmarks	ZDT4											
Algorithms		**MOFDO**			**MOPSO**			**NSGA-III**			**MODA**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.501608	0.263502	1.193981	0.425684	0.256286	2.993125	0.444092	0.442732	0.445971	69.56207	51.81152	730.2351
2	0.57568	0.236016	1.134353	0.444503	0.253738	4.807117	0.443536	0.442508	0.446179	62.88423	51.17447	429.2433
3	0.719838	0.347964	1.579042	0.442059	0.254349	3.970729	1.104079	1.102824	1.105107	67.07025	51.18764	867.5366
4	0.561183	0.251296	1.063522	0.559633	0.24948	10.82849	0.664161	0.661929	0.665171	65.5532	51.30849	505.4544
5	0.562553	0.225969	1.200403	0.454693	0.250487	2.836659	0.663479	0.662468	0.664913	64.33402	51.27346	581.0313
6	0.462736	0.234685	1.139921	0.52844	0.249087	7.161605	0.663702	0.662688	0.665135	63.34068	51.80189	832.598
7	3.82E-01	0.255803	1.124358	0.448792	0.25067	10.23314	0.664341	0.662945	0.665652	67.11351	51.15887	329.8522
8	0.371716	0.255869	0.704552	0.474109	0.248197	4.068897	1.103979	1.102746	1.104794	65.87016	51.17335	447.4578
9	0.628182	0.223411	1.006202	0.407769	0.251594	3.97219	0.884038	0.88281	0.885893	58.62504	51.44762	472.5677
10	0.799466	0.253323	1.616551	0.373616	0.254281	1.389945	0.444017	0.442876	0.445125	61.88936	51.23744	361.3001
11	1.679996	0.374775	2.499869	0.415407	0.252731	2.391366	0.883739	0.882735	0.884334	66.58221	51.18319	485.125
12	0.649062	0.265265	2.155388	0.41576	0.255985	4.344767	0.664049	0.66305	0.665495	68.89694	57.16953	455.3356
13	3.96E-01	0.248385	1.48691	0.49651	0.219284	8.617647	0.88389	0.883035	0.885415	63.30141	51.82974	492.2761
14	0.464708	0.205478	1.022592	0.482048	0.250662	5.062121	0.884114	0.882845	0.885186	62.24437	51.43052	370.2326
15	1.390486	0.294857	2.836018	0.437069	0.255783	5.697825	0.224164	0.222449	0.225405	61.06805	51.55192	510.0294
16	0.398058	0.195501	0.918248	0.529921	0.254687	8.537481	1.10345	1.102417	1.105262	67.09614	51.12355	538.885
17	0.763934	0.226435	3.20023	0.472243	0.255312	2.344645	0.884143	0.883255	0.887056	66.05687	51.88529	341.8472
18	0.431924	0.25242	3.680553	0.396768	0.247864	1.995582	0.223721	0.222869	0.225086	68.66719	51.22492	382.9895
19	3.94E-01	0.241733	1.193421	0.480982	0.257917	4.503863	1.104004	1.102787	1.107088	62.93178	51.15103	639.0773
20	0.505436	0.21529	0.708872	0.449051	0.253311	4.090232	1.104101	1.103105	1.106219	61.24712	51.18291	438.7137
21	1.180564	0.414899	1.767307	0.453047	0.254782	3.143242	0.444134	0.442871	0.445293	67.16612	51.16149	547.431
22	0.427927	0.247604	3.977259	0.485769	0.251091	6.489652	1.104384	1.103202	1.105357	61.52138	51.19177	559.2781
23	0.602405	0.248651	1.70872	0.444641	0.253524	7.884768	0.44412	0.442636	0.446076	67.25835	51.85532	464.4187
24	0.562379	0.270772	1.303488	0.445717	0.233212	6.94077	0.884505	0.883128	0.885914	62.6597	51.22099	594.6946
25	0.958548	0.314042	1.95301	0.399203	0.257333	1.526915	0.664077	0.662377	0.665003	68.78317	51.26081	612.1739
26	0.596267	0.251145	1.031952	0.485757	0.25877	4.094758	1.103791	1.10286	1.104667	63.43928	51.24429	415.8264
27	1.64E+00	0.389742	3.539831	0.589466	0.25511	13.38893	0.223958	0.223116	0.225614	67.9857	51.30341	488.885
28	0.733705	0.367987	1.633678	0.495321	0.256305	3.418142	1.103659	1.102734	1.104464	62.36858	51.13545	478.0794
29	0.47481	0.244438	0.992227	0.461552	0.254022	6.288041	0.224113	0.221643	0.225416	66.54766	51.81577	302.73
30	0.587943	0.222729	0.956471	0.457132	0.249232	1.784249	0.883761	0.882543	0.885429	66.81991	56.7717	352.8466
Mean	0.680202	0.267999	1.677631	0.461755	0.251503	5.16023	0.73731	0.736073	0.738791	64.96281	51.74228	500.9384
STD	0.352945			0.047786			0.307518			2.847807		
P-value				0.001386			0.506665			9.07E-72		

Table 7.33 MOFDO, MODA, MOPSO, and NSGA-III (ZDT-5 results).

Benchmarks	ZDT5											
Algorithms		**MOFDO**			**MOPSO**			**NSGA-III**			**MODA**	
Turns	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
1	0.480168	0.190596	1.427909	0.499046	0.036241	2.576891	0.604092	0.602732	0.605971	0.112033	0.007839	2.100606
2	0.57938	0.1279	2.457694	0.145476	0.036549	2.552476	0.603536	0.602508	0.606179	0.108315	0.020551	1.99456
3	0.205416	0.038228	1.104134	0.322771	0.054003	2.278078	0.604079	0.602824	0.605107	0.109166	0.009391	2.835237
4	0.393651	0	1.536738	0.134451	0.036701	2.520962	1.004161	1.001929	1.005171	0.103016	0.006414	1.692453
5	0.40683	0.113574	0.876059	0.235293	0.036516	2.994643	0.603479	0.602468	0.604913	0.130759	0.011025	3.523593
6	0.200343	0	4.591367	0.221375	0.050156	3.11805	1.003702	1.002688	1.005135	0.104512	0.017099	1.626087
7	0.336076	0.190053	1.327961	0.330377	0.043771	1.98616	0.404341	0.402945	0.405652	0.170398	0.012688	9.714147
8	0.387506	0	2.567574	0.590118	0.10737	2.399428	0.603979	0.602746	0.604794	0.103808	0.008103	1.297161
9	0.475818	0	1.959065	0.259405	0.04547	1.245033	1.004038	1.00281	1.005893	0.098503	0.0071	2.108973
10	0.109184	0	1.457939	0.433312	0.058863	1.689852	0.804017	0.802876	0.805125	0.111479	0.014514	2.660849
11	0.5144	0.354545	1.8109	0.164872	0.036504	1.355011	0.403739	0.402735	0.404334	0.127771	0.014863	2.785708
12	0.249614	0.139415	1.02249	0.108415	0.036539	2.449236	0.204049	0.20305	0.205495	0.099884	0.009512	1.244873
13	0.305578	0.135119	1.365795	0.185479	0.03653	1.722728	0.60389	0.603035	0.605415	0.112987	0.002301	4.247754
14	0.626523	0.269135	1.272013	0.135493	0.036319	2.694618	0.404114	0.402845	0.405186	0.101936	0.007343	1.305113
15	0.223152	0.029184	1.439839	0.157949	0.03688	2.167788	0.804164	0.802449	0.805405	0.123335	0.026862	1.873972
16	0.407833	0.203217	1.919709	0.25792	0.036739	2.373885	0.60345	0.602417	0.605262	0.099074	0.014627	1.151438
17	0.246713	0.105355	1.913232	0.204248	0.036864	2.021944	1.004143	1.003255	1.007056	0.119419	0.038112	2.983925
18	0.215622	0.06061	1.491514	0.334494	0.037042	2.130911	0.403721	0.402869	0.405086	0.094862	0.034177	2.061569
19	0.617568	0.225525	2.538779	0.51264	0.046499	2.270001	1.004004	1.002787	1.007088	0.100043	0.009788	2.214823
20	0.105095	0	1.210132	0.113914	0.03665	2.258108	0.804101	0.803105	0.806219	0.116883	0.022804	3.037678
21	0.320182	0.209767	1.63322	0.170936	0.037043	2.356589	0.404134	0.402871	0.405293	0.153999	0.006732	2.194439
22	0.38528	0.168285	3.730247	0.33345	0.064163	2.010273	0.604384	0.603202	0.605357	0.10001	0.035675	1.532359
23	0.638474	0.370005	2.471775	0.193567	0.03675	3.181097	0.80412	0.802636	0.806076	0.133204	0.007273	1.942145
24	0.553181	0.094761	2.132204	0.176554	0.038675	1.817086	0.404505	0.403128	0.405914	0.09677	0.007051	1.487705
25	0.510341	0.278164	1.240521	0.544334	0.118778	2.139262	0.804077	0.802377	0.805003	0.124899	0.048183	2.352479
26	0.392351	0.095336	1.288716	0.301917	0.047013	2.43419	0.403791	0.40286	0.404667	0.096677	0.000551	1.332485
27	0.104897	0.018205	5.893423	0.140048	0.036724	2.030647	1.003958	1.003116	1.005614	0.141929	0.011144	3.221339
28	0.272818	0.136391	1.37289	0.208877	0.036599	1.807486	0.803659	0.802734	0.804464	0.094169	0.000809	2.150296
29	0.370307	0.093242	1.188437	0.225486	0.056177	1.335736	0.404113	0.401643	0.405416	0.110967	0.016264	1.829411
30	0.121823	0.018277	1.134968	0.416484	0.052168	1.766127	0.803761	0.802543	0.805429	0.104036	0.000134	1.312063
Mean	0.358538	0.122163	1.912575	0.268623	0.046876	2.189477	0.663977	0.662739	0.665457	0.113495	0.014298	2.393841
STD	0.161795			0.136599			0.235755			0.018271		
P-value				0.023547			2.4E-07			2.4E-11		

باشکەری پاڵپشتیکار بە نرخی هاوکێشە: ئەلگۆریسمێکی نوێ زیرەکی دەسکردی پورەیە بۆ کێشە تاک وە فرە ئامانجەکان

نامەیەکە پێشکەش کراوە بە ئەنجومەنی کۆلێجی زانست لە زانکۆی سلێمانی وەک بەشێك لە پێداویستیەکانی بەدەست هێنانی بڕوانامەی دکتۆرا لە زانستی کۆمپیوتەر (زیرەکی دەستکرد)

لە لایەن

جزا محمود عبدالله

ماستەر لە سیستەمی سۆفتوێر و تەکنەلۆجیایی ئینتەرنێت

٢٠١٢ زانکۆی شەفێلد، بەریتانیا

بە سەرپەرشتی

د.طاریق احمد رشید

پرۆفیسۆر

2020-May Gelawêj-2720

پوختە

لەم تێزەدا، هەستاوین بە پێشکەش کردنی دوو ئەلگۆریسمی نوێ لە بواری زیرەکی دەستکرد، ئەوانیش SOFDO یەک ئامانج و MOFDO فرە ئامانج، کە بەکاردێن بۆ دۆزینەوەی باشترین چارەسەر لەنێوان کۆمەڵێک چارەسەردا، ئەگەر ئەوکێشەیە بتواندرێت بەشێوەیەکی بیرکاریانە پێناسە بکرێن. ئەلگۆریسمە پێشکەشکراوەکان لاسایی زیرەکی پوورەی هەنگیان کردۆتەوە بۆ دۆزینەوەی چارەسەر بۆ کێشەکان، ئەمەش بە ووردبوونەوە لە مێش هەنگ کاتێک کە ئەگڕێن بە دوای هێلانەیەکی تردا (کۆڵەس) بۆ ژیان لە وەرزی زاوزێیاندا، لێرەدا سود لەو زیرەکیە وەرگیراوە کە هەنگەکان پیشانی ئەدەن کاتێک بە کۆمەڵ بریارێك ئەدەن. لێرەدا ئەمانەوێت ئاماژە بەوەبکەین کە هیچ جۆرە پەیوەندیەکی ئەلگۆریسماتیکی لە نێوان ئەلگۆریسمە پێشکەشکراوەکان و ئەلگۆریسمی ABC دا نیە. شایەنی باسە، کە SOFDO و MOFDO لە سەر بنەمای ئەلگۆریسمی بەناوبانگ PSO درووستکراونەتەوە لە ڕووی جوڵاندن و ئاڕاستەکردنی یەکەکانی گەڕان، بەڵام شێوازی جێبەجێکردنی ئەم ئەرکانە بەتەواوەتی جیاوازە. لە ئەلگۆریسمە پێشکەشکراوەکاندا هەندێ گۆڕاوی تایبەت کە لە ئەنجامی نرخی هاوکێشەکانی یەکەکانی گەڕان، لەگەڵ هەندێک نەگۆڕی تر کە ئەلگۆریسمەکە ڕێگە بەبەکارهێنەرانی ئەدات ئارەزومەندانە نرخی پێبدەن درووست ئەکرێن. ئەم گۆڕاو و نرخە جێگیرانە یارمەتی یەکەکانی گەڕان ئەدەن بۆ دۆزینەوەی چارەسەر لە هەردوو بەشی دیاریکردن و نزیک بوونەوە لە ئامانجەکان. لەوەش زیاتر، لە ئەلگۆریسمی MOFDO فرە ئامانجدا، هەموو زانیاریەکانی کە لە ئەلگۆریسمێکی کۆمەڵایەتیدا هەیە جێگیر کراوە، وەک (زانیاری باری، پێوەری، تۆپۆگرافیک، بوار و مێژووی گەڕان). بەدرێژای ئەم تێزە، هەردوو ئەلگۆریسمی پێشنیار کراو باسدەکرێن بە ووردری لیان دەکۆڵدرێتەوە. بۆ ئەم مەبەستەش ئەلگۆریسمە پێشکەشکراوەکان بەم شێوەی خراونەتە ڕوو لە درێژەی تێزەکەدا.

یەکەم، ئەلگۆریسمی SOFDO یەک ئامانج تاقی ئەکرێتەوە لەسەر ١٩ هاوکێشیە کلاسیکی کە ستاندارکردن لە جیهاندا بۆ بابەتی تاقیکردنەوە و دەرخستنی توانستی ئەلگۆریسمەکان، لەمەش زیاتر SOFDO یەک ئامانج تاقی ئەکرێتەوە لەسەر ١٠ هاوکێشیەی ستانداردی مۆدێرن کە ناسراون بە CEC-01 2019، ئەنجامەکانی SOFDO بەراورد کران بە ئەنجامەکانی پێنج ئەلگۆریسمی تری ستانداردی بەتوانا کە ئەوانیش : PSO ، GA ، DA WOA، SSA ن. ئەنجامی بەراورد کردنەکە دەری خست کە SOFDO ئەنجامی باشتری هەیە لە زۆربەی تاقیکردنەوەکاندا، هەروەها ئەنجامەکانی تری هاونرخ یان نزیکی ئەنجامەکانی ئەلگۆریسمە کانی تر بوون. جگە لەمەش، بەشێوازێکی ئاماریانە سەلمێنراوە کە ئەنجامەکانی SOFDO گرنگ لە چاو ئەنجامەکانی ئەلگۆریسمەکانی تر، ئەمەش بە

بەکار هێنانی Wilcoxon rank-sum وە دۆزینەوەی P-value. لەمانەش زیاتر، هەستاوین بە پیشاندانی چۆنێتی کارکردنی SOFDO بە چەندین هێلکاری وێنەی جۆراو جۆر، کە پێشاندەدەن چۆن ئەلگۆریسمە پێشنیار کراوەکە کرداری باڵو بوونەوەو جوڵانی یەکەکانی گەڕان ئەنجام ئەدات. هەروەها بۆ سەلاماندنی ئەوەی کە SOFDO ئەتوانرێت بەکار بهێندرێت لە کێشەکانی ژیانی ڕاستەقینەدا. هەستاوین بە بەکارهێنانی ئەلگۆریسمی SOFDO لە بواری ڕێخستن و دیزاینی ئەنتێنای پەیوەندی کردن (aperiodic antenna array designs) و بواری شەپۆلە ڕادیۆیەکانی ئێف ئێم (frequency-modulated sound waves).

دووەم، ئەلگۆریسمی پێسنیارکراو MOFDO فرە ئامانجمان تاقیکردۆتەوە لەسەر کۆمەڵێک هاوکێشەی ستانداردی جیهانی کە بۆ بابەتی تاقیکردنەوە و دەرخستنی توانستی ئەلگۆریسمەکان ئامادەکراون لە وانەش کۆمەڵێکی ناسراو بە ZDT کلاسیک وە ١٢ هاوکێشەی ستانداردی مۆدێرن کە ناسراون بە CEC-2019 MMF، ئەنجامەکانی ئەم تاقیکردنەوانە بەراورد کراوە بە ئەنجامی تاقیکردنەوە کۆمەڵێ ئەلگۆریسمی بەناوبانگی وە ک MOPSO, NSGA-III, MODA، ئەنجامی بەراوردکردنەکە دەریخست کە MOFDO ئەنجامی باشتری هەیە لە زۆربەی تاقیکردنەوەکاندا، هەروەها ئەنجامەکانی تری هاونرخ یان نزیکی ئەنجامەکانی ئەلگۆریسمە کانی تر بوون. جگە لەمەش، بەشێوازێکی ئاماریانە سەلمێنراوە کە ئەنجامەکانی MOFDO گرنگ لە چاو ئەنجامەکانی ئەلگۆریسمەکانی تر، ئەمەش بە بەکار هێنانی Wilcoxon rank-sum وە دۆزینەوەی P-value. لەمانەش زیاتر، هەستاوین بە پیشاندانی چۆنێتی کارکردنی MOFDO بە چەندین هێلکاری وێنەی جۆراو جۆر، کە پێشاندەدەن چۆن ئەلگۆریسمە پێشنیار کراوەکە ئامانجەکان دەدۆزێتەوە و ڕێکیان دەخات، هەروەها بۆ سەلاماندنی ئەوەی کە MOFDO ئەتوانرێت بەکار بهێندرێت لە کێشەکانی ژیانی ڕاستەقینەدا. هەستاوین بە بەکارهێنانی ئەلگۆریسمیکە لە بواری ئەندازەی دروستکردنی شێلمانی پۆڵاین (welded beam design problem)، ئەنجامەکانی ئەم تاقیکردنەوەیە دەیسەلمێنێت کە ئەلگۆریسمە پێشنیارکراوەکەمان ئەتاوانرێت بەکار بهێندرێت بۆ دۆزینەوەی چارەسەر بۆ گرفتەکانی ژیانی مرۆڤ.

محسن المعتمد على قيمة الدالة: خوارزمية الجديدة لزكاء السرب لحل المشاكل ذات هدف واحد و اهداف المتعديدة

أطروحة مقدمة لمجلس

كلية العلوم في جامعة السليمانية كجزء من متطلبات لنيل الشهادة الدكتوراه في علوم الحاسب الآلي (الذكاء الاصطناعي)

من قبل

جزا محمود عبد الله

ماجستير (2012) أنظمة البرمجيات وتكنولوجيا الإنترنت ، جامعة شيفيلد ، المملكة المتحدة.

باشراف

د. طارق احمد راشد

أستاذ

2020-May Gelawêj-2720

الخلاصة

في هذا العمل ، تم اقتراح خوارزميتين الجديدتين لسرب الذكى (المحاكات من الزكاء سرب من نحلة العسل)، الأول خوارزمية وحيدة الهدف SOFDO, والثانى خوارزمية متعديدة الأهداف MOFDO. الخوارزميات المذكورة ألهمت من العمليات البحث لنحل واتخاذ القرارات الجماعية بينهم. ليس لدى خوارزمياتينا أي اتصال حسابي مع خوارزمية نحل العسل أو خوارزمية مستعمرة النحل الاصطناعي (ABC). تجدر الإشارة إلى أن كلاً من SOFDO و MOFDO يعتبران خوارزميات تعتمد على تحسين الجسيمات السرب كما هن موجودات في الخوارزمية (PSO) الشهيرة، حيث يقومان بتحديث موضع وكلاء البحث عن طريق إضافة متغيرات السرعة. ومع ذلك ، فإنها تحتسب معادلة السرعة بشكل مختلف تماماً. خوارزميات المقترحة من قبلينا يستخدمون قيمة دالية و أوزان المعينة لإنتاج المعادلة المذكورة، وهذه الأوزان توجه وكلاء البحث خلال كل من مراحل الاستكشاف والاستغلال. علاوة على ذلك، تعتبر MOFDO كخوارزمية الجتماعية ، وبالتالي ، تم النظر في جميع أنواع المعرفة الخمسة (المعرفة الظرفية والمعيارية والطوبوغرافية و نطاق البحث والتاريخ البحث) أثناء التنفيذ. خلال هذا العمل ، يتم تقديم خوارزميات SOFDO و MOFDO ، ويتم توضيح الدافع وراء الفكرة.

أولاً, يتم اختبار الخوارزمية SOFDO على مجموعة من 19 معاير اختبار الكلاسيكية ، اضافتا الى ذلك, تتم مقارنة النتائج مع ثلاث خوارزميات معروفة: PSO ، الخوارزمية الجينية (GA) ، وخوارزمية اليعسوب (DA) ، بالإضافة إلى ذلك ، يتم اختبار SOFDO على 10 معاير اختبار CEC-C06 2019 الجديدة. في هذا الاختبار الأخيرة, تتم مقارنة النتائج SOFDO مع ثلاث خوارزميات حديثة : (DA)، وخوارزمية تحسين الحوت (WOA) ، وخوارزمية سرب سالب(SSA) . تظهر نتائج SOFDO أداء أفضل في معظم الحالات ونتائج متقاربية في حالات أخرى. علاوة على ذلك ، يتم اختبار النتائج إحصائيًا باستخدام اختبار مجموع تصنيف ويلكوكسون لإظهار أهمية النتائج. وبالمثل، يتم التحقق من ثبات SOFDO في كل من مرحلتي الاستكشاف والاستغلال ومقاومة الأداء باستخدام ةعاير القياسية مختلفة. أخيرًا ، يتم تطبيق SOFDO على تطبيقات العالم الحقيقي كدليل على جدواه.

ثانياً, الخوارزمية الثانية, يتم اختبارMOFDO على معيارين القياسيين: المعيار ZDT الكلاسيكية و المعيار متعدد الأهداف CEC 2019 الجديدة. تم مقارنة نتائج MOFDO مع أحدث نسخة من الخوارزمية جسيمات السرب متعدد الأهداف (MOPSO) ، الخوارزمية الجينية لفرز الخواص الوراثية غير المهيمنة (NSGA-III) ، وخوارزمية اليعسوب متعدد الأهداف(MODA) . MOFDO تتفوق

على متنافيساتيها في غالبية الحالات والنتائج المتقاربة في الحالات الأخرى. علاوة على ذلك ، يتم استخدام MOFDO لتحسين المشاكل الهندسية في العالم الحقيقي (مثل مشكلة تصميم الشعاع الملحوم) ؛ الخوارزمية المقترحة تعالج المشكلة بنجاح وتوفر مجموعة واسعة من الحلول الممكنة الموزعة بشكل جيد ، والتي تمكن صناع القرار من الحصول على مزيد من خيارات القابلة للتطبيق.