CHEMOTAXIS DIFFERENTIAL EVOLUTION OPTIMIZATION TECHNIQUES FOR GLOBAL OPTIMIZATION

A THESIS SUBMITTED TO THE FACULTY OF ACHITECTURE AND ENGINEERING
OF EPOKA UNIVERSITY

BY

YUNUS EMRE YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

· JUNE, 2016 ·

Approval of the thesis:

CHEMOTAXIS DIFFERENTIAL EVOLUTION OPTIMIZATION TECHNIQUES FOR GLOBAL OPTIMIZATION

submitted by **YUNUS EMRE YILDIZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Computer Engineering, Epoka University** by,

Assoc. Prof. Dr. ....
Dean, Faculty of Architecture and Engineering

Assist. Prof. Dr. ....
Head of Department, **Computer Engineering, EPOKA University**

Assist. Prof. Dr. ....
Supervisor, **Computer Engineering, EPOKA University**

Examining Committee Members:

Prof. Dr. ....
.... Dept. .... University

Prof. Dr. ....
.... Dept. .... University

Prof. Dr. ....
.... Dept. .... University

Prof. Dr. ....
.... Dept. .... University

Assist. Prof. Dr. ....
.... Dept. .... University

Date : 11.06.2016

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: ....

Signature: ....

ABSTRACT

CHEMOTAXIS DIFFERENTIAL EVOLUTION OPTIMIZATION TECHNIQUES FOR GLOBAL OPTIMIZATION

Yildiz, Yunus Emre
Ph.D., Department of Computer Engineering
Supervisor: Dr. Oguz Altun

June 2016

Nature inspired and bio-inspired algorithms have been recently used for solving low and high dimensional search and optimization problems. In this context, Bacterial Foraging Optimization Algorithm (BFOA) and Differential Evolution (DE) have been widely employed as global optimization techniques inspired from social foraging behavior of *Escheria coli* bacteria and evolutionary ideas such as mutation, crossover, and selection, respectively.

BFOA employs chemotaxis (tumble and run steps of a bacterium in its lifetime) activity for local search whereas the global search is performed by elimination-dispersal operator. Elimination-dispersal operator kills or disperses some bacteria and replaces others randomly in the search space. This operator mimics bacterium's death or dispersal in case of high temperature or sudden water flow in the environment.

DE employs the mutation and crossover operators to make a local and a global search that explore the search space. Exploration and exploitation balance of DE is performed by two different parameters: mutation scaling factor and crossover rate. These two parameters along with the number of population have an enormous impact on optimization performance.

In this thesis, two novel hybrid techniques called Chemotaxis Differential Evolution Optimization Algorithm (CDEOA) for low dimensions and micro CDEOA (µCDEOA) for high dimensional problems are proposed. In these techniques, we incorporate the principles of DE into BFOA with two conditions. What makes our techniques different from its counterparts is that it is based on two optimization strategies: exploration of a bacterium in case of its failure to explore its vicinity for food source and exploitation of a bacterium in case of its achievement to exploit more food source. By means of these evolutionary ideas, we manage to establish an efficient balance between exploration of new areas in the search space and exploitation of search space gradients. Statistics of the computer simulations indicate that µCDEOA outperforms, or is comparable to, its competitors in terms of its convergence rates and quality of final solution for complex high dimensional problems.

ABSTRAKT

TEKNIKAT E OPTIMIZMIT TË EVOLUCIONIT DIFERENCIAL TË KEMOTAKSES PËR OPTIMIZIM GLOBAL

Yildiz, Yunus Emre
Doktoraturë, Departamenti i Inxhinierise Kompjuterike
Udhëheqësi: Dr. Oguz Altun

Qershor 2016

Algoritmat e frymëzuar nga natyra dhe biologjia kohët e fundit po përdoren për zgjidhjen e problemave të optimizmit me dimension kërkimi të ulët dhe të lartë. Në këtë kontekst, Algoritmi i Optimizmit të Sjelljeve Ushqyese të Baktereve (BFOA) dhe Evolucioni Diferencial (DE) janë përdorur gjerësisht si teknika globale të optimizmit, e para e frymëzuar nga sjelljet ushqyese të bakterit Escherichia Coli dhe e dyta nga proçeset evolucionare të tilla si mutacioni, kryq këmbimi dhe seleksionimi natyror.

BFOA-ja përdor aktivitetin Kemotaksik (lëvizjen e një bakteri gjatë gjithë jetës së tij) për kërkim lokal, ndërsa kërkimi global kryhet duke përdorur operatorin e eliminim-shpërndarjes. Operatori i eliminim-shpërndarjes vret ose shpërndan disa nga bakteret në hapësirën e kërkimit dhe po në ketë hapësire, disa baktere të tjera i zëvendëson në mënyre rastësore. Ky operator imiton vdekjen ose shpërndarjen e baktereve në temperatura të larta apo në rast të një vërshimi të papritur të ujit në ambient.

DE përdor operatoret e mutacionit dhe kryq këmbimit për kërkime lokale dhe globale në zbulimin e hapësirës se kërkimit. Balanca e zbulimit dhe shfrytëzimit të DE realizohet nga 2 parametra të ndryshëm: faktori i shkallëzimit të mutacionit dhe shpeshtia e kryq këmbimit. Këto 2 parametra së bashku me numrin e popullatës kanë një ndikim të madh në performancën e optimizmit.
Në këtë tezë propozohen 2 teknika të reja hibride, njëra për problemat me dimension të ulët e quajtur Algoritmi i Optimizmit te Evolucionit Diferencial të Kemotakses (CDEOA) dhe njëra për problemat me dimension të lartë quajtur mikro CDEOA (CDEOA). Në këto teknika ne përdorim disa parime të DE në BFOA me 2 kushte. Ajo që e bën këtë teknike të ndryshme nga simotrat e saj është fakti se ajo bazohet në 2 strategji optimizmi: zbulimi i një bakteri në rast kur ai dështon në zbulimin e ushqimit dhe shfrytëzimin e një bakteri kur ai arrin të shfrytëzoje më shumë burime ushqimi. Me anë të këtyre ideve evolucionare ne arrijmë të vendosim një balancë ndërmjet zbulimit të hapësirave të reja në hapësirën e kërkimit dhe shfrytëzimin e gradientit të hapësirës së kërkimit. Statistikat e simulimeve kompjuterikë të CDEOA tregojnë se ajo i tejkalon ose është e krahasueshme me konkurentet e vet në terma të kursit të konvergjencës dhe cilësisë së zgjidhjes përfundimtare të problemave të komplikuara dhe me dimension të lartë.

To My Family

ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Dr. Oguz Altun for his continuous guidance, encouragement, motivation, patience, support, and enormous knowledge during all the stages of my thesis. I sincerely appreciate the time and effort he has spent to improve my experience during my graduate years.
I would like to thank to my thesis progress committee members, Dr. Arban Uka, Dr. Ilir Capuni, Dr. Elton Domnori, Dr. Endri Stoja, and Dr. Albana Halili for their comments and suggestions throughout the entire thesis.
I am also thankful to Ph.D. candidate Ali Osman Topal for sharing his ideas and his friendly support regarding this thesis study.
Last but not the least, I would like to thank my family: my parents and to my wife and children for supporting me morally throughout writing this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACBSFO_DES	Adaptive Chemotaxis Bacterial Swarm Foraging Optimization Differential Evolution Strategies
ACO	Ant Colony Optimization
BA	Bat Algorithm
BCV	Best cost value
BFOA	Bacterial Foraging Optimization Algorithm
CDEOA	Chemotaxis Differential Evolution Optimization Algorithm
CDE	Chemotaxis Differential Evolution
CEC	Congress of Evolutionary Computation
DE	Differential Evolution
EA	Evolutionary Algorithms
ES	Evolutionary Strategies
FA	Firefly Algorithm
FE	Function Evaluation

FERDE	Fitness Euclidean - distance Ratio Differential Evolution
FWA-DE	Fireworks Algorithm with Differential Mutation
GA	Genetic Algorithm
GP	Genetic Programming
ICDEOA	Improved Chemotaxis Differential Evolution Optimization Algorithm
IEEE	Institute of Electrical and Electronics Engineers
mCDEOA	Micro Chemotaxis Differential Evolution Optimization Algorithm
MC	Market Cost
NP	Non-deterministic Polynomial-time
PC	Production Cost
POBL-ADE	Partial Opposition-Based Adaptive Differential Evolution
PSO	Particle Swarm Optimization
RS	Random Search
SCND	Supply Chain Network Design
SCRM	Supply Cost of Raw Material
TS	Tabu Search
WCV	Worst Cost Value

CHAPTER 1
INTRODUCTION

1.1 Overview

In the past few decades, nature and natural processes have been studied by several researchers in order to get inspiration for tackling complex real-world problems. Natural selection tends to eliminate species with poor foraging strategies through methods for locating, handling, and ingesting food and favors the propagation of genes of species with successful foraging behavior since they are more likely to obtain enough food to enable them to reproduce. After many generations, poor foraging strategies are either eliminated or shaped into good ones. In this context, many nature inspired techniques such as Ant Colony Optimization (ACO) [17], Artificial Bee Colony (ABC) [28] have been employed. Particularly, such evolutionary principles have led Kevin M. Passino to develop a new nature-inspired technique known as Bacterial Foraging Optimization Algorithm (BFOA) which maximizes foraging organism or animal's energy intake per unit time spent, considering all the constraints presented by its own physiology such as sensing and cognitive capabilities, environment (e.g., density of prey, risks from predators, physical characteristics of the search space) for distributed search and optimization ([50]; [37]).

In order to enhance BFOA performance, a considerable number of ameliorations have been performed, including hybridization with evolutionary algorithms (EA) ([30]; [4]), and improvements based on analysis of the BFOA operators ([15]; [2]; [3] ; [13]). Up to now, BFOA has been applied in applications in transmission loss reduction ([63]), optimal control design ([50]), estimation of harmonic components ([41]), active power filter ([42]), learning of artificial neural networks ([31]), and PID controller tuning ([32]).

In hybrid algorithms, Kim et al. [30] have proposed a hybrid algorithm BFOA-Genetic Algorithm (GA) which manipulates on mutation, crossover, different step sizes, chemotaxis steps, and lifetime of the bacteria. Biswas et al. [4] have proposed a BFOA-Differential Evolution (DE) hybrid which employs mutation and crossover operators. Jarraya et al. [26] have proposed Adaptive Chemotactic Bacterial Swarm Foraging Optimization with Differential Evolution Strategy (ACBSFO_DES) which integrates Particle Swarm Optimization and DE operators into BFOA to cope with the premature convergence and slowness of the standard and variants of BFOA. As opposed to the aforementioned improvements of BFOA and the state-of-the-art optimization algorithms in the literature, BFOA still needs to be optimized in high dimensional unimodal and multimodal problems in terms of the convergence speed and the quality of final solution. In order to overcome the deficiencies of BFOA, three novel optimization algorithms are proposed in this thesis.

1.2 Originality and Motivation

Each classical algorithm has been improved with different contributions by the researchers since its inception in the literature. In this respect, classical BFOA has been hybridized with evolutionary and nature-inspired algorithms, too. Although there exists some hybridization studies of BFOA with DE ([4], [26]), they only employ DE operators (mutation and crossover) explicitly. The proposed techniques employ DE operators implicitly in case that some conditions are met. In the literature of BFOA, there is no enough study as opposed to some bio inspired optimization algorithms such as DE and Particle Swarm Optimization (PSO). On the other hand, the fact that BFOA does not possess the evolutionary operators such as mutation and crossover has led us to hybridize with DE operators. Furthermore, DE participated in the First International IEEE Competition on Evolutionary Optimization and became the fastest algorithm ([61]). These aforementioned milestones encouraged us to integrate DE operators into BFOA.

1.3 Objective of Thesis

The objectives of the work proposed in this thesis are as follows:

1. Present a brief literature review of metaheuristics algorithms.
2. Develop a novel search algorithm (CDEOA) hybridizing the bacterium's chemotaxis operator with Differential Evolution (DE) evolutionary operators.
3. Optimize the performance of CDEOA by eliminating the premature convergence effect of standard BFOA reproduction operator.
4. Validate the performance of the proposed algorithms (CDEOA, iCDEOA, and µCDEOA) by comparing with its canonical and state-of-the-art counterparts.
5. Analyze the performance of the proposed algorithm (CDEOA) on a real life problem.
6. Improve the performance of the micro BFOA in terms of quality of final solution by introducing µCDEOA for high dimensional problems.

1.4 Organization of Thesis

Organization of this thesis is as follows:

Chapter 1 presents a brief introduction and overview of the study. It also focuses on the objectives to achieve the desired goals.

Chapter 2 presents the introduction to optimization techniques in the metaheuristics field. It also presents a literature review of the related algorithms.

Chapter 3 proposes two hybrid optimization techniques and discusses the numerical results obtained in benchmark tests in detail. It also gives the optimization of Supply Chain Problem using the proposed algorithm.

Chapter 4 includes the proposed micro bio inspired optimization technique and discusses the numerical results obtained in benchmark tests. It also presents the hybrid micro bio inspired techniques in the literature briefly.

Chapter 5 discusses the effects of Differential Evolution (DE) optimization algorithm mutation strategies and DE parameters on CDEOA with the numerical results.

Chapter 6 concludes the thesis. It also presents the future aspects of three hybrid optimization techniques.

CHAPTER 2

BIO INSPIRED OPTIMIZATION TECHNIQUES

2.1 Introduction

Optimization is considered to be mathematical procedures in all engineering fields. It literally means a kind of process or technique in order to make a system or a decision as excellent or effective as possible. In computational intelligent field, it is finding the best optimum solution out of a number of candidate solutions. Optimization algorithms can be categorized as deterministic or stochastic. The stochastic algorithms depend on the random variables generated at the beginning of search. Every time the algorithm is launched, it will end up with different points since there is a randomness in the algorithm. Some examples of stochastic algorithms are Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Differential Evolution (DE), and Bacterial Foraging Optimization Algorithm (BFOA). On other hand, deterministic techniques work in a deterministic way without randomness. The same initial points will yield the same final solution even if the algorithm is launched several times. The examples of deterministic approaches are linear programming, non-linear programming, mixed integer non-linear programming. In comparison with stochastic techniques, deterministic methods require huge computational time, thereby tending to fail in converging the optimal solution when the search space range and the number of dimensions of the problem increase.

Algorithms based on stochastic approaches were recently called metaheuristics which means higher level process or heuristic (to find or to discover by trial and error) to find near optimal solutions in an optimization domain. The definition of Glover and Laguna ([21]) describes the best what the metaheuristics is: ”master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality”. The characteristics of metaheuristics are as follows:

1. Metaheuristics algorithms need to employ local search and global search operators to make a search.
2. A reasonable amount of time is reserved in order to find the optimal solution in a difficult optimization problem. However, there is no guarantee that the optimal solution is achieved.
3. Metaheuristic algorithms yield approximate solutions and are non-deterministic.
4. Metaheuristics are not problem specific.

One major impact that makes the metaheuristic algorithm superior in solving hard and complex optimization problem is whether the algorithm possesses exploration and exploitation balance or intensification and diversification balance [6]. Diversification helps in generating the diverse candidate solutions in global scale to explore the whole search space whereas intensification makes the search process more focused on the local area as long as the current candidate solution is better than previous candidate solution. There must be a good trade-off between intensification (exploitation) and diversification (exploration) in order to improve the convergence rate and the quality of final solution within a period of time. Once the candidate solution gets trapped or gets stagnated in a local optima, diversification helps the candidate solution escape it and also increases diversity of candidate solutions.

As in Fig. 2.1, the most employed metaheuristic optimization algorithms are depicted in the field. Bio inspired techniques are divided into two categories namely, evolutionary and nature inspired techniques. Evolutionary algorithms are GA (Genetic Algorithm), GP (Genetic Programming), ES (Evolutionary Strategies), and DE (Differential Evolution). Nature inspired techniques are BFOA (Bacterial Foraging Optimization Algorithm), PSO (Particle Swarm

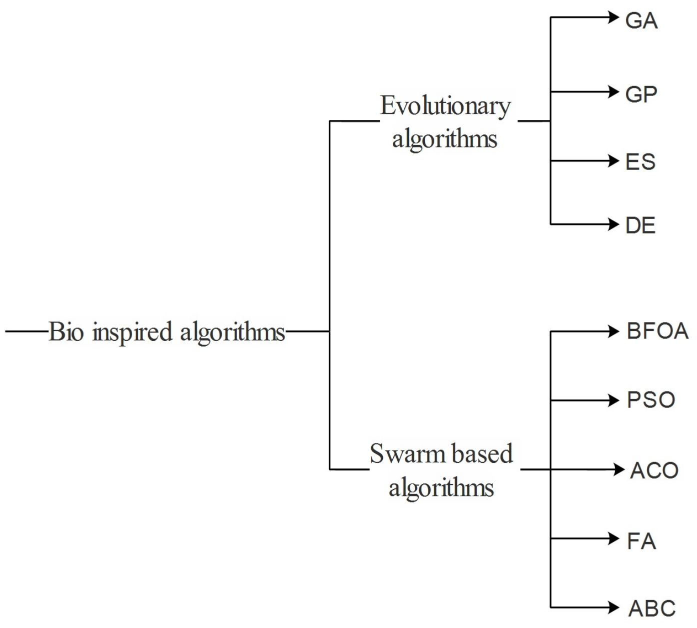


Figure 2.1: Taxonomy of Bio inspired algorithms

Optimization), ACO (Ant Colony Optimization), FA (Firefly Algorithm), and ABC (Artificial Bee Colony).

2.2 Genetic Algorithm

GA is an evolutionary based stochastic optimization algorithm which mimics the natural selection ([23]). It is subclass of evolutionary algorithms (EA) which generate solutions for the optimization problems. With the power of natural evolution techniques such as mutation, crossover, and selection, it has a potential global search ability. To generate a good solution, a population of randomly generated candidate solutions undergoes an iterative optimization process which the population is called generation in each iteration. In other words, the candidate solutions (individuals) in the population which have a set of chromosomes are initialized on the search space. The representation of the candidate solutions are in binary as strings of 0's and 1's. In each generation, every individual's fitness (cost in case of minimization problem) is evaluated according to objective function. Assessment of an individual is performed according to objective function used in the optimization problem. The best individuals in terms of objective function value are selected out of current generation which will be used for the next generation. The general flowchart of Genetic Algorithm is shown in Fig. 2.2.

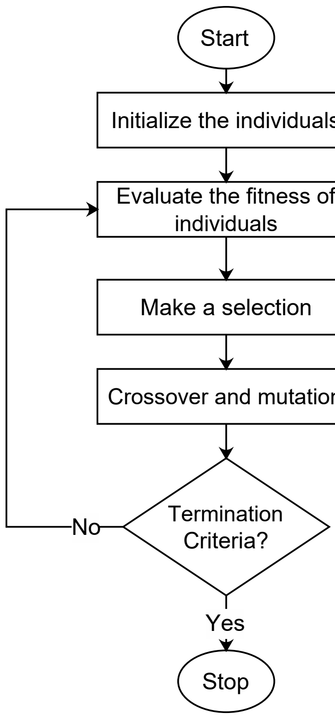


Figure 2.2: General flowchart of Genetic optimization

2.3 Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a population based optimization algorithm that optimizes a problem iteratively trying to help the candidate solution reach the global optimum according to a termination criteria. The origin of the algorithm belongs to ([18]) and the algorithm was employed for the simulation of social behavior: bird flock and fish school. With the population called particles, PSO optimizes a problem by moving the particles randomly according to objective function value on the search space.

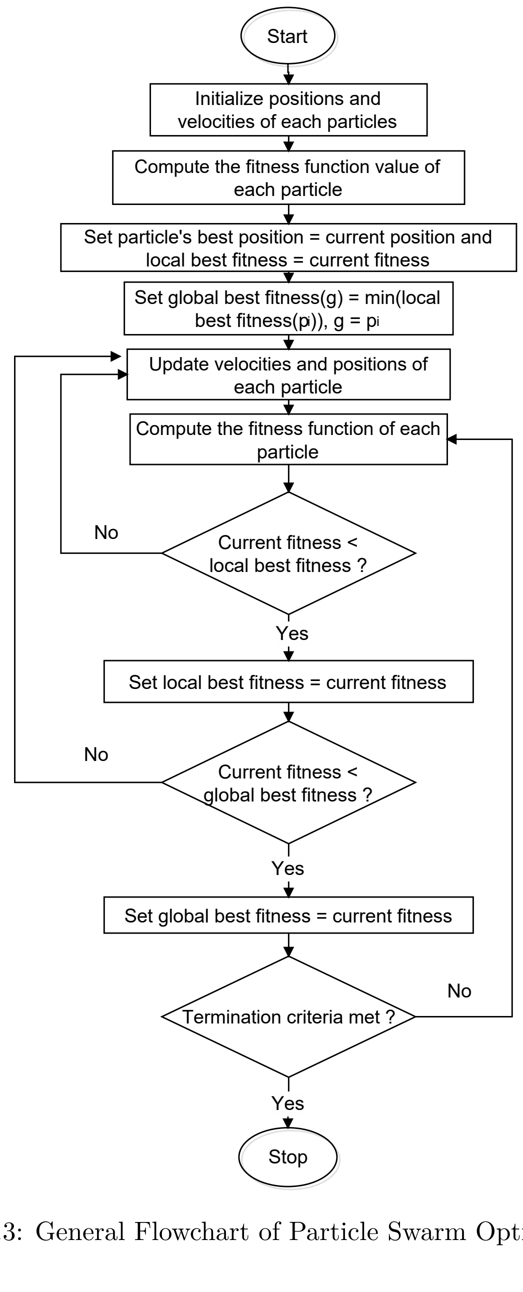

Start
Initialize positions and velocities of each particles
Compute the fitness function value of each particle
Set particle's best position = current position and local best fitness = current fitness
Set global best fitness(g) = min(local best fitness(pi)), g = pi
Update velocities and positions of each particle
Compute the fitness function of each particle
Current fitness < local best fitness ?
No
Yes
Set local best fitness = current fitness
Current fitness < global best fitness ?
No
Yes
Set global best fitness = current fitness
Termination criteria met ?
No
Yes
Stop

Figure 2.3: General Flowchart of Particle Swarm Optimization

In order for a particle to update its position, its current position and its velocity are taken into consideration by adding these two parameters each other with a formula 2.1. According to the formula, there are two parameters that will influence the performance of the algorithm: particle's best known position ($p_{i,d}$) and particle's best known position of the whole individuals (g_d). The general flowchart of Particle Swarm Optimization is shown in Fig. 2.3.

$$v_{i,d} = \omega * v_{i,d} + \varphi_p * r_p * (p_{i,d} - x_{i,d}) + \varphi_g * r_g * (g_d - x_{i,d}) \tag{2.1}$$

where r_p and $r_g \sim U(0,1)$, the parameters ω, φ_p, and φ_g are other parameters that are chosen by the practitioners. These parameters control the behavior and efficiency of optimization process of PSO.

2.4 Bacterial Foraging Optimization Algorithm

The bacterial foraging system consists of four principal mechanisms, namely chemotaxis, swarming, reproduction, and elimination-dispersal ([50]). Below we briefly describe each of these processes.

2.4.1 Chemotaxis

This process is the motion of an *E.coli* bacterium using consecutive *tumble* and *swim (run)* steps via flagella.[1] While *tumble* is a unit walk in random direction, *swim* is the consecutive movement in the same direction. *E.coli* alternates between these two modes of operation throughout its entire life-time. Suppose $\theta(i,j,k,l)$ represents the position of the ith bacterium at jth chemotactic, kth reproductive and lth elimination-dispersal step. The position of the bacterium in the next step may be represented by Eq. (2.2) and Eq. (2.3),

$$\vec{t}\,(j) = \frac{\Delta(i)}{\sqrt{\Delta^T\,(i)\,\Delta\,(i)}} \tag{2.2}$$

[1]Note that swim and run can be used interchangeably in the literature of BFOA.

$$\theta(i, j+1, k, l) = \theta(i, j, k, l) + C(i)\,\vec{t}(j) \tag{2.3}$$

where $C(i)$ is the size of the unit walk (run-length unit) which is pre-defined as a constant, $\vec{t}(j)$ (Eq. 2.2) is the direction angle of the step, and $\Delta(i)$ is the random vector whose elements lie in $[-1, 1]$. In a *run* step, $\vec{t}(j)$ remains the same as $\vec{t}(j-1)$; in a *tumble* step, $\vec{t}(j)$ is generated randomly from uniform distribution in the range of $[0, 2pi]$.

2.4.2 Reproduction

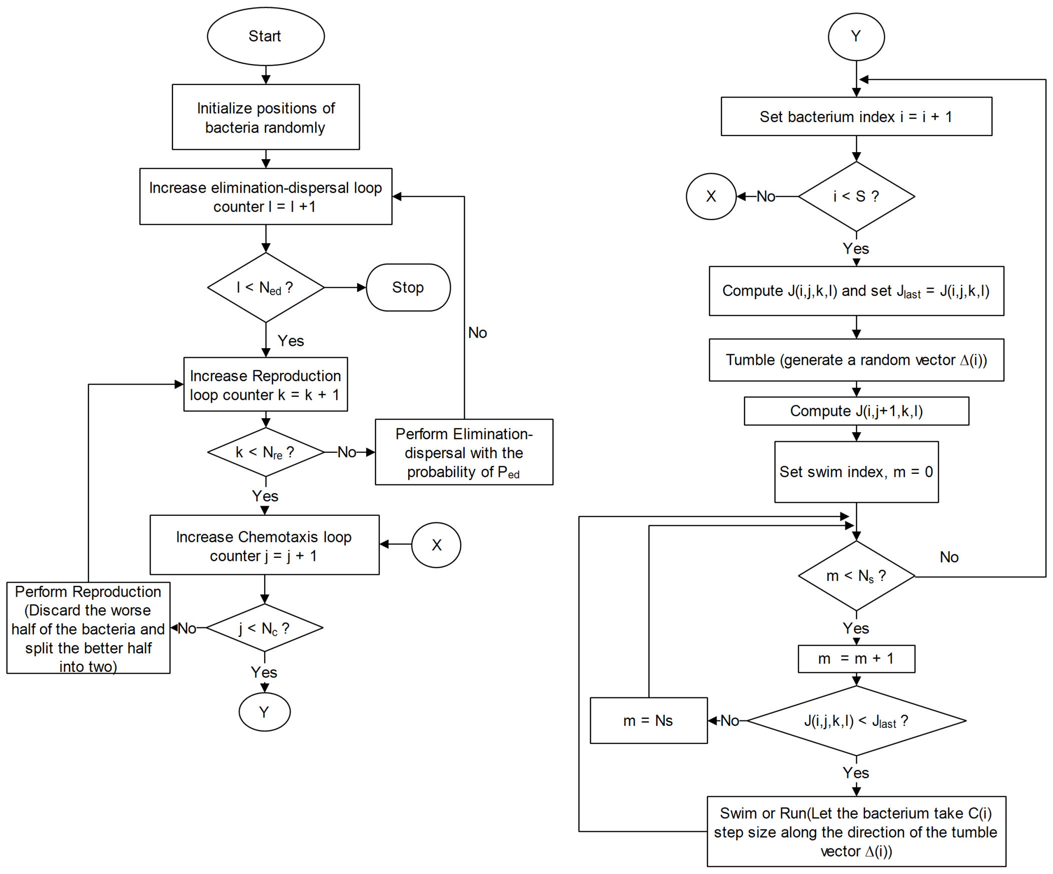
The health of the each bacterium is computed as the sum of the objective function values calculated during its life-time and the population is sorted according to their health. The healthiest 50% bacteria asexually split into two which are then placed at the same locations. The remaining 50% with poor health is discarded to keep the population size constant.

2.4.3 Elimination and Dispersal

Elimination and dispersal events may occur in the local environment when the bacteria are exposed to gradual or sudden changes such as significant rise of temperature or sudden flow of water. In order to simulate these events in BFOA, some bacteria are liquidated at random with a pre-determined probability (P_{ed}) while the new replacements are initialized randomly in the search space.

2.4.4 Swarming

A group of *E.coli* bacteria arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemoeffector. The bacteria, when stimulated by a high level of succinate, release an attractant aspartate which helps them to aggregate into groups and thus move as concentric patterns of swarms with high



Start
Initialize positions of bacteria randomly
Increase elimination-dispersal loop counter l = l +1
l < N_ed ?
Stop
Yes
No
Increase Reproduction loop counter k = k + 1
k < N_re ?
No
Perform Elimination-dispersal with the probability of P_ed
Yes
Increase Chemotaxis loop counter j = j + 1
X
Perform Reproduction (Discard the worse half of the bacteria and split the better half into two)
No
j < N_c ?
Yes
Y
Y
Set bacterium index i = i + 1
X
No
i < S ?
Yes
Compute J(i,j,k,l) and set J_last = J(i,j,k,l)
Tumble (generate a random vector Δ(i))
Compute J(i,j+1,k,l)
Set swim index, m = 0
m < N_s ?
No
Yes
m = m + 1
m = Ns
No
J(i,j,k,l) < J_last ?
Yes
Swim or Run(Let the bacterium take C(i) step size along the direction of the tumble vector Δ(i))

Figure 2.4: Flowchart of the Classical BFOA

bacterial density. The bacterium-to-bacterium signaling in *E. coli* swarm may be represented by Eq. (2.4),

$$
\begin{aligned}
&J_{cc}\left(\theta, P\left(j,k,l\right)\right) = \sum_{i=1}^{S} J_{cc}\left(\theta, \theta\left(i,j,k,l\right)\right) = \\
&\sum_{i=1}^{S}\left[-d_{attractant}exp\left(-w_{attractant}\sum_{m=1}^{p}\left(\theta_m-\theta_m^i\right)^2\right)\right] \\
&+\sum_{i=1}^{S}\left[h_{repellent}exp\left(-w_{repellent}\sum_{m=1}^{p}\left(\theta_m-\theta_m^i\right)^2\right)\right]
\end{aligned}
\tag{2.4}
$$

where $P(j,k,l) = \{\theta(j,k,l) | i = 1,2,...,S\}$ represents the positions of each member in the population, S is the total number of bacteria, p is the number of variables to be optimized that are present in each bacterium, $J_{cc}(\theta, P(j,k,l))$ is the objective function value to be added to the actual objective function (to be minimized) to present a time-varying objective function, $\theta = [\theta_1, \theta_2, ..., \theta_p]^T$ is a point in the search domain, and θ_m^i is the mth elements of the ith bacterium position θ^i . $d_{attractant}$, $w_{attractant}$, $h_{repellent}$, $w_{repellent}$ are distinct coefficients from each other.

A flowchart of the classical BFOA is given in Fig. 2.4 which is adapted from the study of [15].

2.5 Differential Evolution

Differential evolution (DE) is a population based bio-inspired technique which utilizes mutation, crossover, and selection operators to minimize an objective function. For each generation G, a new population is created from the current population members,

$$x_{i,G}, i = 1,2,...,N \tag{2.5}$$

where N is the population size. The initial population (Eq. 2.5) is randomly generated in the search domain with N vectors according to a uniform probability distribution. After initialization, DE enters mutation, crossover, and selection processes. Basically, DE chooses three candidate vectors randomly from the population and new solution vectors are created by adding the scaled difference between two population vectors to a third population ([61]).

2.5.1 Mutation

At each generation G, a mutant vector $v_{i,G}$ is generated for each target vector $x_{i,G}, i = 1,2,...N$ in the current population. The most used mutation strategies[2] in the literature are as follows:

[2]http://www.icsi.berkeley.edu/~storn/code.html

- “DE/rand/1”

$$v_{i,G} = x_{r_0,G} + F\left(x_{r_1,G} - x_{r_2,G}\right) \tag{2.6}$$

- “DE/rand/2”

$$v_{i,G} = x_{r_0,G} + F\left(x_{r_1,G} - x_{r_2,G}\right) + F\left(x_{r_3,G} - x_{r_4,G}\right) \tag{2.7}$$

- “DE/current-to-best/1”

$$v_{i,G} = x_{i,G} + F(x_{best,G} - x_{i,G}) + F(x_{r_0,G} - x_{r_1,G}) \tag{2.8}$$

- “DE/best/1”

$$v_{i,G} = x_{best,G} + F\left(x_{r_0,G} - x_{r_1,G}\right) \tag{2.9}$$

- “DE/best/2”

$$v_{i,G} = x_{best,G} + F(x_{r_0,G} - x_{r_1,G}) +F(x_{r_2,G} - x_{r_3,G}) \tag{2.10}$$

- “DE/rand-to-best/1”

$$v_{i,G} = x_{r_0,G} + F(x_{r_1,G} - x_{r_2,G}) +F(x_{r_{best},G} - x_{r_1,G}) \tag{2.11}$$

where r_0, r_1, r_2, r_3, and r_4 are distinct integers randomly chosen from the current population and are different from i. $x_{best,G}$ is the best individual vector in the current generation G, and F is the mutation factor generally within the range of $[0, 2]$.

2.5.2 Crossover

After mutation, a binomial crossover operation is carried out on $v_{i,G}$ and $x_{i,G}$ to generate a new trial vector $u_{i,G} = (u_{i,1,G}, u_{i,2,G}, ..., u_{i,D,G})$:

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & if\ R_j\left(0,1\right) \leq C_r\ or\ j = j_{rand} \\ x_{i,j,G}, & if\ R_j\left(0,1\right) > C_r \end{cases} \tag{2.12}$$

where $j = 1, 2, ..., D$, D is the dimension of the search space, j_{rand} is a randomly chosen integer in $[1, D]$, $R_j\left(0,1\right)$ is uniformly generated random number between 0 and 1 for each j, and $C_r \in [0,1]$ is the crossover rate parameter.

2.5.3 Selection

Selection process is carried out to choose the better of the parent vector $x_{i,G}$ and the trial vector $u_{i,G}$. In case of a minimization problem, the selected parent vector in the next generation is given by Eq. (2.13),

$$x_{i,G+1} = \begin{cases} u_{i,G}, & if\ \ f\left(u_{i,G}\right) < f\left(x_{i,G}\right) \\ x_{i,G}, & otherwise \end{cases} \tag{2.13}$$

where $f\left(\cdot\right)$ is the function for minimization. If trial vector $u_{i,G}$ produces a better fitness value, it replaces its parent in the next generation; otherwise the parent is kept in the population.

2.6 Artificial Bee colony

Artificial Bee Colony (ABC) [28] is another nature inspired optimization algorithm which is based on behaviors of honey bees. In the population, there are three types of roles: employed bees which go to the food source previously visited by itself; onlooker bees which wait on the dance area to decide whether or not the food source is worth being visited; scout bees which makes a random search.

In ABC algorithm, each search process cycle is made up of three steps carried out by the aforementioned bee groups: employed bees are sent onto food sources to measure their nectar amounts; the food sources are chosen by the onlooker bees depending on the information of the nectar amount which employed bees share. The scout bees are determined and sent onto possible food sources [28]. The main steps of ABC algorithm are as follows:

- Initialize the food source positions.
- REPEAT.
    1. Each employed bee goes to a food source that visited previously in her memory then assesses the amount of nectar and makes a special dance for onlooker bees.
    2. Depending on the employed bees dances, each onlooker selects one of their food sources and assesses the quality of nectar.
    3. Scouts discover new food sources and these are replaced by the abandoned food sources.
    4. The position of the best food source is kept in the memory.
- UNTIL (all the requirements are satisfied).

2.7 Benchmark Test Functions

In this section, in order to give an idea regarding the different situations which the algorithms experience, the most frequently used single-objective optimization test problems and artificial landscapes are presented to assess the characteristics of the optimization algorithms, such as: convergence rate, quality of the final solution, robustness, and general performance. [43], [1]. Eq. 2.14, Eq. 2.15, Eq. 2.16, Eq. 2.17, Eq. 2.18, and Eq. 2.19 shows each problem's function definition. In addition to the artificial landscape and function definition, the search range and the global minimum of problems are denoted, as well.

1. Sphere Function

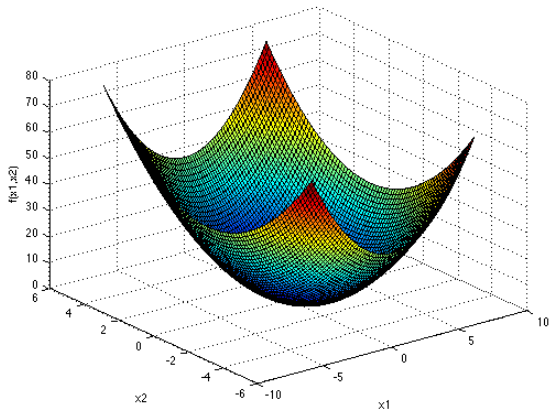


Figure 2.5: 2 dimensional Sphere function

$$f_1(x) \quad = \sum_{i=1}^{D} x_i^2 \tag{2.14}$$

Search range : $x_i \in [-5.12, 5.12], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (0, , , , 0)$

2. Rosenbrock's function

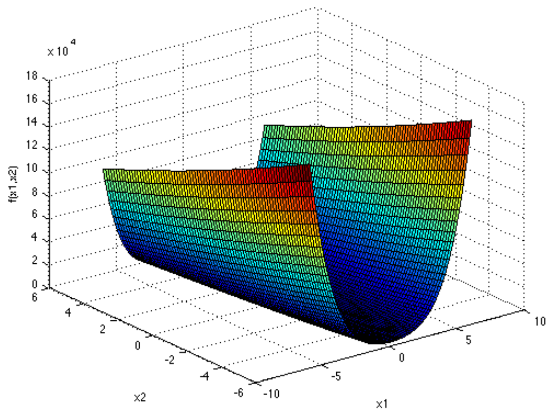


Figure 2.6: 2 dimensional Rosenbrock function

$$f_2(x) \quad = \textstyle\sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{2.15}$$

Search range : $x_i \in [-5, 10], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (0, , , , 0)$

3. Ackley's function

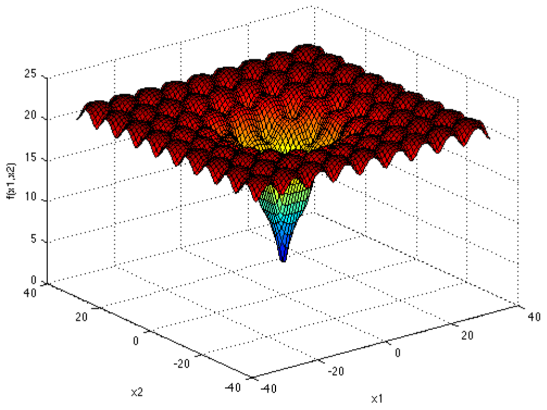


Figure 2.7: 2 dimensional Ackley function

$$f_3(x) \quad = -20exp(-0.2\sqrt{\tfrac{1}{D}\textstyle\sum_{i=1}^{D} x_i^2}) \quad -exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)) + 20 + e, \tag{2.16}$$

Search range : $x_i \in [-32.768, 32.768], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (0, , , , 0)$

4. Griewank's function

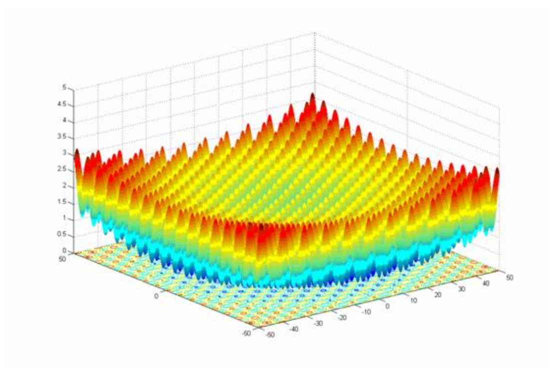


Figure 2.8: 2 dimensional Griewank function

$$f_4(x) \quad = \sum_{i=1}^{D} \frac{x_i^2}{4000} \tag{2.17}$$

Search range : $x_i \in [-600, 600], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (0, , , , 0)$

5. Rastrigin's function

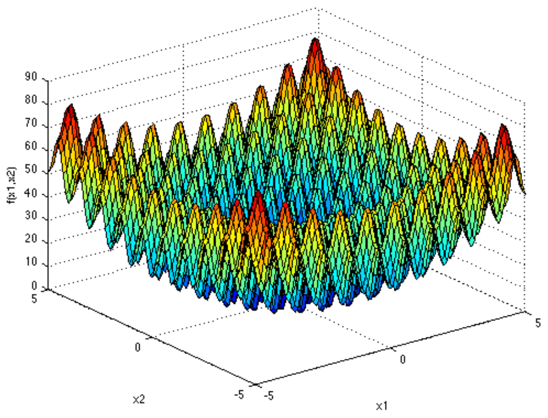


Figure 2.9: 2 dimensional Rastrigin function

$$f_5(x) \quad = \textstyle\sum_{i=1}^{D}(x_i^2 - 10cos(2\pi x_i) + 10)polghkh \tag{2.18}$$

Search range : $x_i \in [-5.12, 5.12], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (0, , , , 0)$

6. Schwefel's function

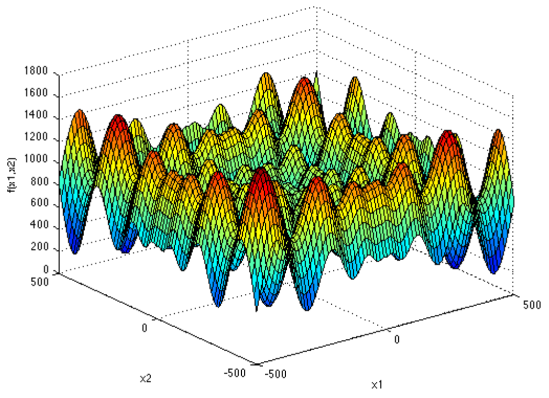


Figure 2.10: 2 dimensional Schwefel function

$$f_{16}(x) \quad = 418.9829 * D - \textstyle\sum_{i=1}^{D} x_i sin(|x_i|^{1/2}) \tag{2.19}$$

Search range : $x_i \in [-500, 500], i = 1, 2, ..., D$

Global minimum : $f(x^*) = 0, x^* = (420.9687, , , , 420.9687)$

CHAPTER 3
BFOA BASED HYBRID OPTIMIZATION ALGORITHMS

This chapter introduces a novel optimization approach named Chemotaxis Differential Evolution Optimization Algorithm (CDEOA) and its improved variant (ICDEOA). This new approach is based on the integration of two new strategies into the chemotaxis step of BFOA: weak bacterium's search and strong bacterium's foraging. The performance of weak bacteria is enhanced by randomly moving to new positions whereas the strong bacteria is enhanced by integrating the ideas of differential evolution (DE) operators. With these strategies, we establish an effective distribution of the responsibility of exploring new areas, and responsibility of exploiting the search space gradients. The simulation results reveal that CDEOA has shown superior performance in multi-model and high-dimensional functions in terms of convergence speed and quality of final solution. The proposed algorithm has been compared with classical BFOA, DE, and BFOA variants, and the state-of-the-art DE variants over a test suit of 30 CEC 2014 benchmark functions ([35]).

3.1 CDEOA

Empirical studies ([5] ; [15]) report that BFOA possesses a poor convergence behavior on several multi-modal benchmark functions that have rough fitness landscapes when compared with other naturally inspired optimization techniques like the Genetic Algorithm (GA) ([65]), Particle Swarm Optimization (PSO) ([51], [29]), and DE ([61]). In addition, the BFOA elimination-dispersal step (the death and then random position assignment of a bacterium) is not a complete biologically valid model ([37]). Additionally, the performance of BFOA needs to be improved on complex optimization problems with high dimensionality due to fact that classical BFOA yields poor convergence behavior on high dimensional problems ([4]).

Based on above inadequacies of classical BFOA, a novel optimization method, called CDEOA (Algorithm 1), which hybridizes BFOA with DE is proposed. CDEOA adopts the same mutation, crossover, and selection operations of DE as described in Eq. (2.10), Eq. (2.12), and Eq. (2.13) in Section 2.5, respectively.

3.1.1 Distribution of the exploration and exploitation responsibilities

The basic idea behind the proposed CDEOA algorithm is centered on two different strategies: a) making "weak" bacteria more explorative, where "weak" bacteria are the ones on positions with low nutrient concentrations, and b) making "strong" bacteria more exploitative, where "strong" bacteria are the ones on positions with high nutrient concentrations. Črepinšek et al. [76] wrote: "Exploration is the process of visiting entirely new regions of a search space while exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points". Exploration and exploitation are opposing forces that need to be balanced ([19]). In this respect, in order to establish a good ratio between exploration and exploitation, selection operator and search operators (mutation and crossover) of DE are employed in this study. The selection operators we use favor a search toward the regions of the best individuals. In addition, Bäck and Schwefel [7] reported that selection pressure has a great control over level of exploration and exploitation. While high selection pressure forces the search to be more exploitative, low selection pressure encourages the search to be more explorative. From this perspective, CDEOA tends to have an exploitative search due to behaviors of two selection operators, the reproduction operator of BFOA and selection operator of DE. The differential mutation operator randomly generates different bacteria and thus increases the diversity of the population. From this point of view, the differential mutation operator tends to be more exploration operator. However, the mutation strategy, “DE/best/2”, which CDEOA employs makes the search more exploitative by guiding the search with the best solution so far discovered. De Jong and Spears [16] wrote: “A potential number of ways in which a genetic operator can effect a change has been called its exploratory power.” In this context, the crossover operator which CDEOA employs urges the search toward more explorative due to its high crossover rate. By means of random search (RS), CDEOA carries out an explorative search on the fitness landscape. On the other hand, it is not easy to predict if individuals produced for the next generation by a crossover and/or mutation operator will fall into the exploration or exploitation zones ([76]).

If the bacterium discovers a new, promising area and keeps running for a predefined number M_r of successive generations, then this bacterium undergoes exploitation state (line 60 in Algorithm 1). By “Discovering promising area” we mean the case when bacterium records a fitness improvement from last generation to the current. If the bacterium’s current fitness remain unchanged for a predefined number M_t of successive generations, then this bacterium undergoes exploration state (line 50 in Algorithm 1). Through the means of these strategies, we effectively distribute the responsibilities of exploration and exploitation of fitness landscape amongst the bacteria. The proposed algorithm performs local search through the chemotaxis movement operation of BFOA and the global search over the search space through RS and DE operators Eq. (2.10), Eq. (2.12), and Eq. (2.13).

3.1.2 Making Weak Bacteria Explorative

During the chemotaxis process of BFOA, bacterium in the vicinity of noxious substance will try to move to a position with better nutrient concentration by taking larger steps. This approach is the same as the adaptive step strategy of Dasgupta et al. [15] to make the bacterium more explorative. In the proposed approach, the bacterium performs an elimination process which leads it to change its position randomly in fitness landscape after a number of unsuccessful

Algorithm 1 Detailed pseudo-code of CDEOA. Comments start with "//". The code we discuss in the text is in boldface.

```
1: Parameters:
2:   $p \leftarrow$ dimensions of the search space
3:   $S \leftarrow$ total number of bacteria in the population
4:   $N_c \leftarrow$ number of chemotaxis steps
5:   $N_s \leftarrow$ swimming steps
6:   $N_{re} \leftarrow$ number of reproduction steps
7:   $C(i) \leftarrow$ the run-length unit
8:   $M_t \leftarrow$ maximum number of tumble steps
9:   $M_r \leftarrow$ maximum number of run steps
10:  $f \leftarrow$ objective function to be minimized
11: //Initialize some local variables
12:  $E_t \leftarrow 0$ //bacterium's unsuccessful tumble step
13:  $E_r \leftarrow 0$ //bacterium's unsuccessful run step
14:  $\theta_{best} \leftarrow$random position in the search space
15:  $f_{best} \leftarrow f(\theta_{best})$
16:  $M_{fes} \leftarrow$maximum number of FEs allowed
17:  $N_{fes} \leftarrow 0$ //current number of function evaluations
18: // Define a helper function $J$ that will call the actual objective function $f$. This helper
    function also updates the $N_{fes}$, $\theta_{best}$, and $f_{best}$ variables. This approach makes the rest
    of the algorithm cleaner. Depending on the programming language and programming
    paradigm that will be used, this helper function may be moved outside the CDEOA block,
    or may be a method of a class.
19: **function** $J(\theta)$:
20:     $v \leftarrow f(\theta)$
21:     $N_{fes} \leftarrow N_{fes} + 1$//update number of FEs
22:     **if** $v < f_{best}$ **then**
23:         $\theta_{best} \leftarrow \theta$ //update global best location
24:         $f_{best} \leftarrow v$ //update global best function value
25:     end//if
26:     **return** $v$
27: end// function
28: **while** $N_{fes} < M_{fes}$ **do**//FEs control loop
29:     **for** $k$ from 1 to $N_{re}$ **do**// Reproduction loop
30:     **for** $j$ from 1 to $N_c$ **do**// Chemotaxis loop
31:     **for** $i$ from 1 to $S$ **do**// Tumble-Swim loop
32:         $J_{last} \leftarrow J(\theta(i,j,k))$//$J(\cdot)$ computes the health (fitness) of a bacterium.
33:         $\Delta(i)$ random vector within $[-1,1]$//tumble
34:         $\theta(i,j+1,k) = \theta(i,j,k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
35:         **if** $J(\theta(i,j+1,k)) < J(\theta(i,j,k))$ **then**
36:             $\boldsymbol{E_t} \leftarrow \boldsymbol{E_t} + 1$
```

[1]

```
37:         //Swim:
38:         for $m$ from 1 to $N_s$ do// Swim loop
39:             if $J\left(\theta\left(i,j+1,k\right)\right) < J_{last}$ then
40:                 $J_{last} = J\left(\theta\left(i,j+1,k\right)\right)$
41:                 $\theta\left(i,j+1,k\right) = \theta\left(i,j,k\right) + C\left(i\right)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$
42:                 $\boldsymbol{E_r} \leftarrow \boldsymbol{E_r} + 1$
43:             else
44:                 $m = N_s$//Break from switch loop
45:             end//if
46:         end//Swim loop
47:     end//Tumble-Swim loop
48:     //Exploration Loop
49:     for $\boldsymbol{i}$ from 1 to $\boldsymbol{S}$ do
50:         //Take an exploration step for bacterium i
51:         if $\boldsymbol{E_t} = \boldsymbol{M_t}$ then
52:             $\boldsymbol{\theta(i,j+1,k)} \leftarrow$ random position
53:             $\boldsymbol{J_{last}} = \boldsymbol{J(\theta(i,j+1,k))}$
54:             if $\boldsymbol{J_{last}} < \boldsymbol{J(\theta(i,j,k))}$ then
55:                 $\boldsymbol{J(\theta(i,j+1,l))} \leftarrow \boldsymbol{J_{last}}$
56:             end//if
57:             $\boldsymbol{E_t} = 0$
58:         end//if
59:     end//Exploration Loop
60:     //Exploitation Loop
61:     for $\boldsymbol{i}$ from 1 to $\boldsymbol{S}$ do
62:         if $\boldsymbol{E_r} = \boldsymbol{M_r}$ let bacterium undergo : then
63:             DE mutation as in Eq. (2.10)
64:             DE crossover as in Eq. (2.12)
65:             DE selection as in Eq. (2.13)
66:         end//if
67:     end//ExploitationLoop
68:     end //Chemotaxis loop
69:     //Reproduction
70:     $J^i_{health} = \sum_{j=1}^{N_c+1} J(\theta(i,j,k))$ // Compute the health of each bacterium
71:     Sort bacteria cost $J_{health}$ in ascending order. Let bacteria with the highest $J_{health}$ values
      die and the remaining bacteria with the best values reproduce
72:     end // Reproduction loop
73: end // FEs control loop
74: return $\theta_{best}$
```

Table 3.1: Summary of the CEC 2014 Test Functions
Search Range: $[-100, 100]^D$

	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal Functions	1	Rotated High Conditioned Elliptic Function	100
	2	Rotated Bent Cigar Function	200
	3	Rotated Discus Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Ackley's Function	500
	6	Shifted and Rotated Weierstrass Function	600
	7	Shifted and Rotated Griewank's Function	700
	8	Shifted Rastrigin's Function	800
	9	Shifted and Rotated Rastrigin's Function	900
	10	Shifted Schwefel's Function	1000
	11	Shifted and Rotated Schwefel's Function	1100
	12	Shifted and Rotated Katsuura Function	1200
	13	Shifted and Rotated HappyCat Function	1300
	14	Shifted and Rotated HGBat Function	1400
	15	Shifted and Rotated Expanded Griewank's + Rosenbrock's	1500
	16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Functions	17	Hybrid Functions (N=3)	1700
	18	Hybrid Function 2 (N=3)	1800
	19	Hybrid Function 3 (N=4)	1900
	20	Hybrid Function 4 (N=4)	2000
	21	Hybrid Function 5 (N=5)	2100
	22	Hybrid Function 6 (N=5)	2200
Composition Functions	23	Composition Function 1 (N=5)	2300
	24	Composition Function 2 (N=3)	2400
	25	Composition Function 3 (N=3)	2500
	26	Composition Function 4 (N=5)	2600
	27	Composition Function 5 (N=5)	2700
	28	Composition Function 6 (N=5)	2800
	29	Composition Function 7 (N=3)	2900
	30	Composition Function 8 (N=3)	3000

N:Number of functions used, D: Dimensions.

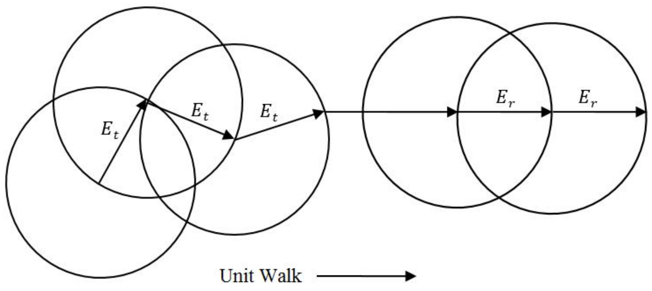


Figure 3.1: Exploration Scheme of a Bacterium

movement attempts M_t. In other words, if the bacterium's number of unsuccessful attempts of taking tumble steps E_t reaches the predefined maximum M_t, the bacterium is liquidated at random (line 50-51 in Algorithm 1). By "Bacterium's unsuccessful attempt to take step" we mean that the bacterium does not take a step while by "bacterium's successful step" we mean that the bacterium takes step to a promising area. In this respect, the unsuccessful step attempt occurs when the current bacterium's objective function value $J(\theta(i, j))$ is worse than bacterium's objective function value $J(\theta(i, j + 1))$ in the next chemotaxis step. Notice that this strategy randomly assigns the locations of several bacteria which make the algorithm more explorative. This process is performed by the elimination-dispersal process in the classical BFOA. In the proposed method, the elimination-dispersal process of BFOA is replaced with "making weak bacteria explorative" strategy.

Fig. 3.1 illustrates the steps of a bacterium in exploration state. If a bacterium fails to discover a promising area in its vicinity, it is randomly dispersed to another location.

3.1.3 Making Strong Bacteria Exploitative

As mentioned in Section 2.4.1, the bacterium runs for a period of time in the same direction as long as it discovers better nutrient-rich concentration and retains its position. In order to make the bacterium more exploitative, the bacterium is supposed to exploit the gradient of the promising area in the vicinity of nutrient-rich substance. Accordingly, each bacterium takes the mutation, crossover, and selection operators of DE (line 62-64 in Algorithm 1) only if the bacterium's successful number of run steps E_r reaches the maximum number of run steps M_r (line 61 in Algorithm 1).

Consequently, these two strategies, making weak bacteria explorative and making strong bacteria exploitative, which are denoted with boldface in the Algorithm 1 leads to a new global hybrid optimization algorithm named CDEOA.

3.1.4 Experimental Study

The CDEOA algorithm was tested using a set of 30 standard benchmark functions (see Table 3.1) of IEEE CEC 2014 single objective optimization competition. These functions include some novel basic problems, graded level of linkages, and rotated trap problems. The suite also has composition test problems obtained by extracting features dimension-wise from several problems. The descriptions of these functions are given in [35]. Functions 1-3 are unimodal, functions 4-16 are simple multimodal, functions 17-22 are hybrid, and functions 23-30 are composition functions.

Parametric Setup

In the experimental studies, parameter settings of the methods are the same as in their original papers. The population size S for the proposed method has been kept to 50 regardless of the dimension size of the problem. M_t and M_r parameters were both set to optimum value 3 after a series of fine-tuning.

The control parameters F (scaling factor) and C_r (crossover rate) of DE need to be tuned properly by the practitioner. When C_r is close to 1.0, the mutation operator can produce the trial vector $u_{i,G}$ different from the target vector $x_{i,G}$ with a high probability. Hence trial vector receives huge data from the mutant vector $v_{i,G}$. Therefore, choosing C_r=0.9 or 1.0 not only speeds up convergence but also diversifies the population by means of one of the best solution dependent DE mutation strategies, "DE/best/2" (Eq. 2.10 in Chapter 2). In this context, C_r

parameter of DE was set to be 0.9. Generally, F is selected within the range of $[0.5 - 1.0]$. It is reported that a smaller F value (e.g., 0.5) can lead to the statistically better performance than the other parameter values ([60], [54]). In this context, F parameter of DE was set to be 0.5. Notice that CDE method employs "DE/rand/1" strategy just as in its original paper. On the other hand, standard DE, ACBSFO_DES employ "DE/best/1" (Eq. 2.9 in Chapter 2); CDEOA employs "DE/best/2" strategy. For the proposed method, classical BFOA, and BFOA variants, following parameter values were chosen: N_c=100, N_s=12, N_{re}=16, $C(i)$=0.1.

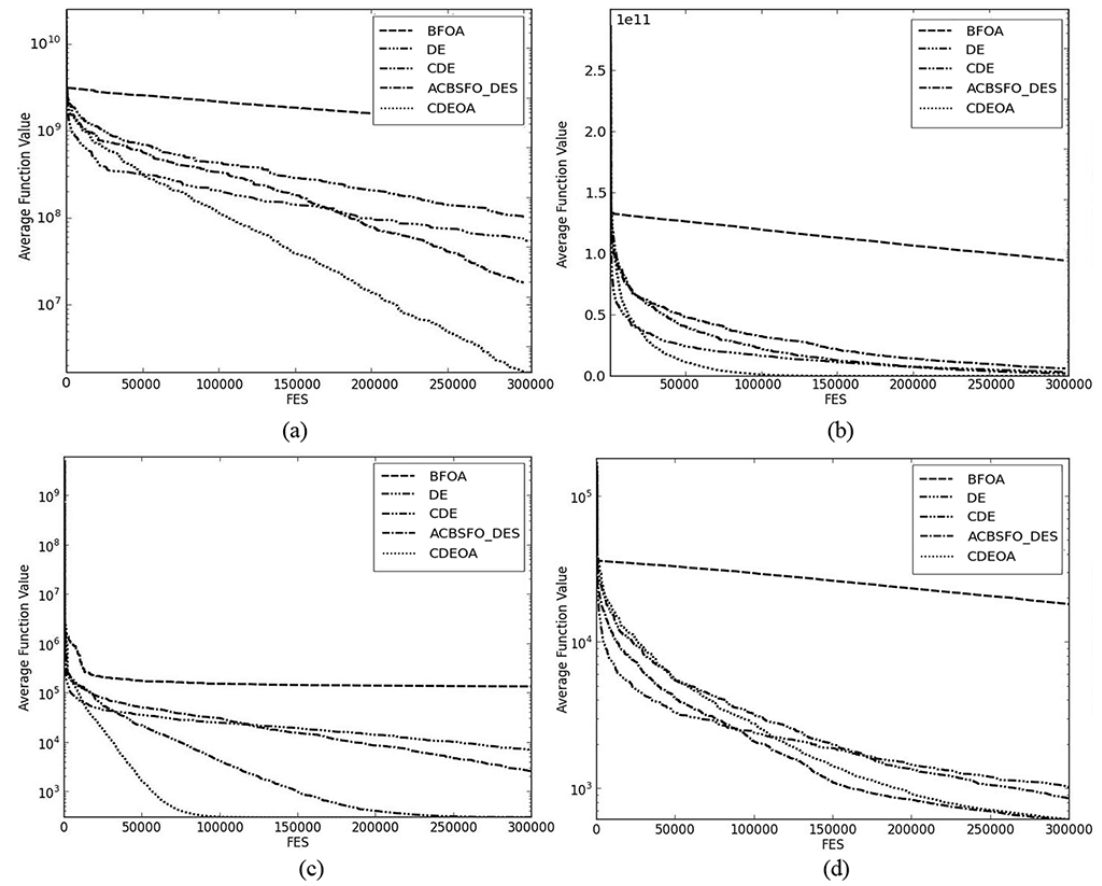


Figure 3.2: (a) F_1: Rotated High Conditioned Elliptic; (b) F_2: Rotated Bent Cigar; (c) F_3: Rotated Discus; (d) F_4: Shifted and Rotated Rosenbrock.

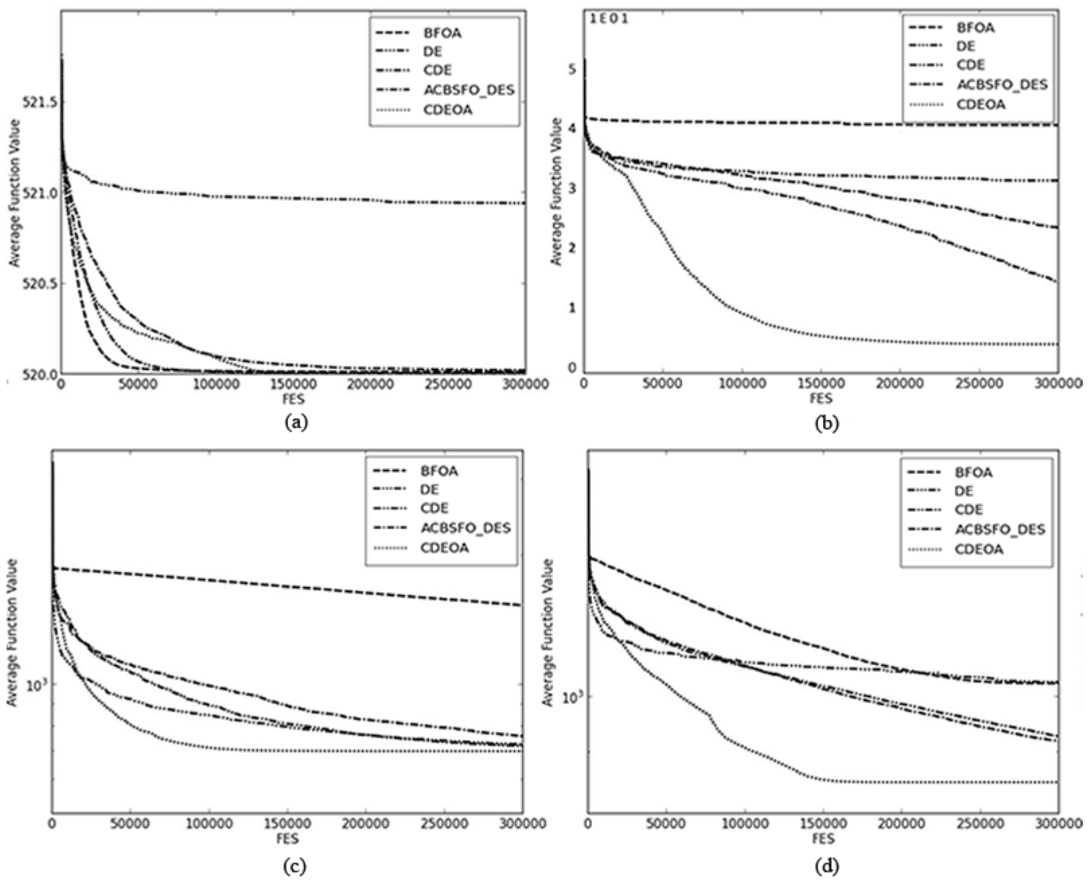


Figure 3.3: (a) F_5: Shifted and Rotated Ackley; (b) F_6: Shifted and Rotated Weierstrass; (c) F_7: Shifted and Rotated Griewank; (d) F_8: Shifted Rastrigin.

Simulation

The study introduced in this section aims to test the quality of the final solution and the convergence speed at the end of a fixed number of function evaluations (FEs). The maximum number of FEs was set to 3×10^5 for 30 dimensions in accordance with the instructions in CEC 2014 special session. For illustrations, median convergence graphs of BFOA, DE, CDE, and ACBSFO_DES, CDEOA for test functions was plotted for unimodal, simple multimodal problems, hybrid problems, and composition problems in Fig. 3.2, Fig. 3.3, Fig. 3.4, Fig. 3.5, Fig. 3.6, Fig. 3.7, and Fig. 3.8. The horizontal axis of these graphs is the number of objective

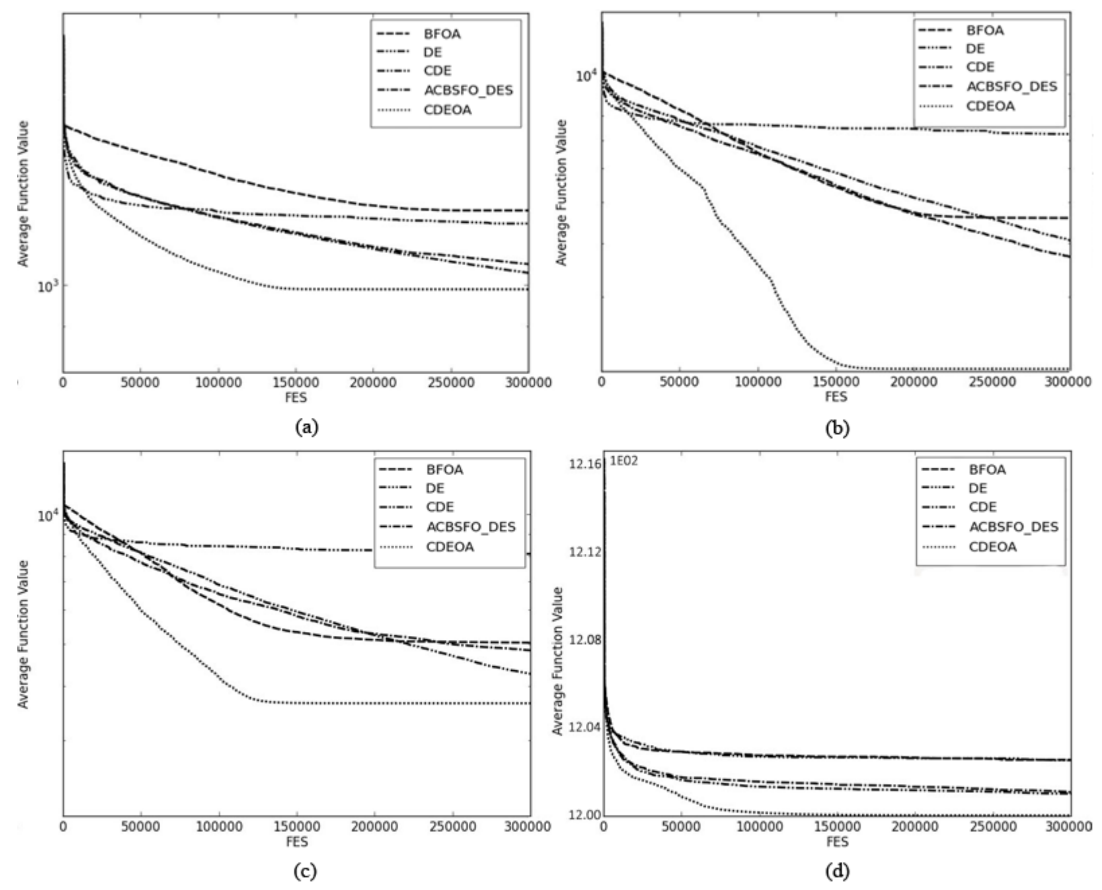


Figure 3.4: (a) F_9: Shifted and Rotated Rastrigin'; (b) F_{10}: Shifted Schwefel; (c) F_{11}: Shifted and Rotated Schwefel; (d) F_{12}: Shifted and Rotated Katsuura.

function evaluations, and the vertical axis is the mean of objective function values. Notice that all the experimental results on these graphs are mean of the function values, not the mean error values. The error values were given in Appendix A.1 and A.3 according to $(F(\vec{x}) - F(\vec{x}^*))$ for evaluating the success of five algorithms, where $\vec{x}$ is the best value of the bacterium in a run and $\vec{x}^*$ is the global best of the test function (Table 3.1). The error values of the function which is less than 1×10^{-8} are considered as zero since such a small error is sufficient for an acceptable convergence to a correct solution and substituted by zeros in Appendix A.1, A.2, A.3, and A.4. CDEOA was compared with classical DE, BFOA, and two other BFOA variants

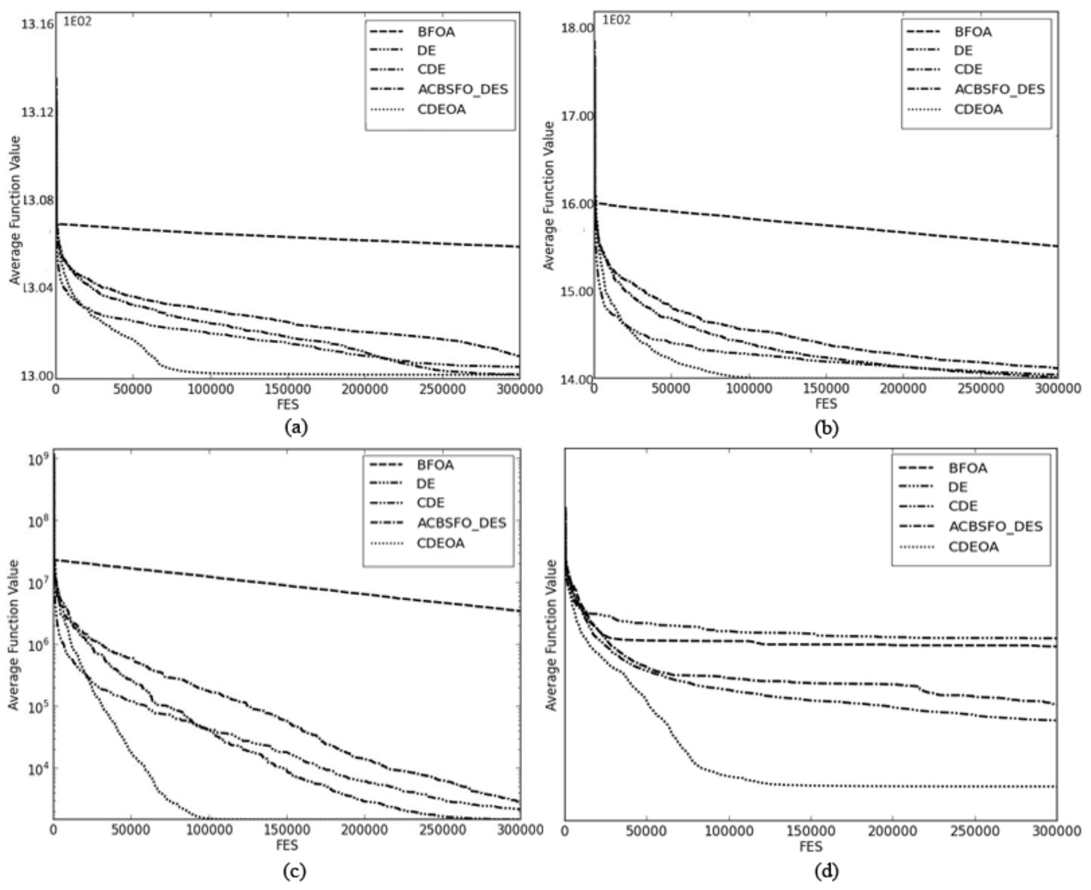
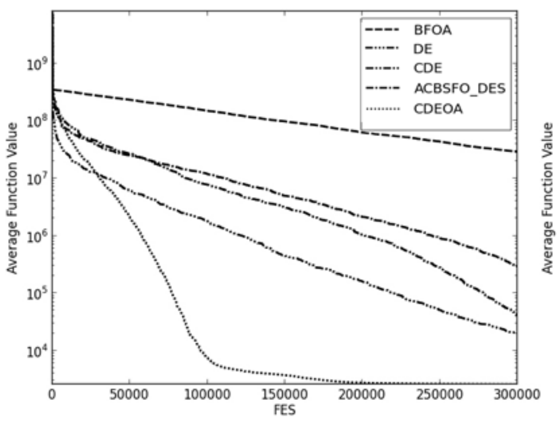


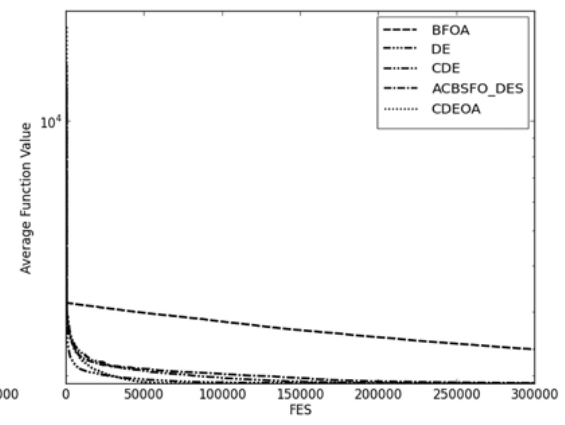
Figure 3.5: (a) F_{13}: Shifted and Rotated HappyCat; (b) F_{14}: Shifted and Rotated HGBat; (c) F_{15}: Shifted and Rotated Expanded Griewank's plus Rosenbrock; (d) F_{16}: Shifted and Rotated Expanded Scaffer's F6.

and four state-of-the-art DE approaches.

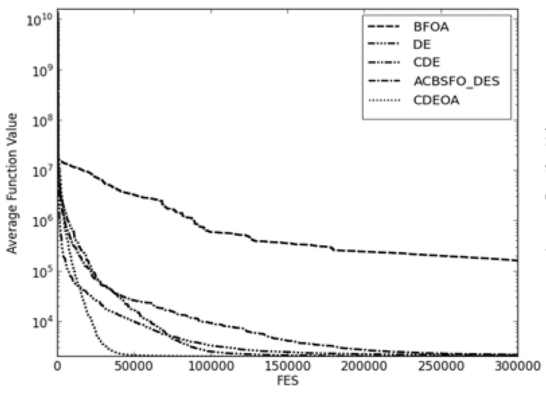
A brief summary of the results at the last three rows of Appendix A.1 and A.3 was given. In these last three rows of Appendix A.1 and A.3, the signs "-", "+", and "" indicate the performance of the corresponding competitors as opposed to CDEOA method. "-" means that the corresponding method in the column performed worse than CDEOA. "+" implies that the corresponding method in the column performed better than CDEOA. Finally, "$\approx$ " denotes that the corresponding algorithm in the column performed comparable to CDEOA. The best final function values (BFV), the worst final function values (WFV), and the median of the final



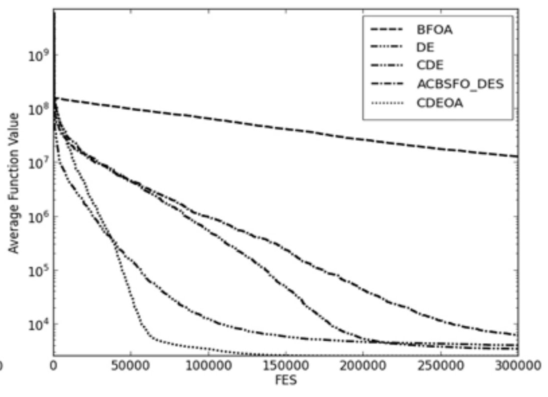
(a)



(b)



(c)



(d)

Figure 3.6: (a) F_{17}: Hybrid ;(N=3); F_{18}: Hybrid 2[1](b) F_{19}: Hybrid 4 (N=4); (c) F_{20}: Hybrid 4 (N=4); (d) F_{21}: Hybrid 5 (N=5)

values were given in Appendix A.2 and A.4.

Comparison of CDEOA with Four Soft Computing Methods

The performance of CDEOA algorithm was compared with classical BFO algorithm ([50]), classical DE algorithm ([38], [61]), two classical BFOA variants, Chemotaxis Differential Evolution (CDE) ([4]), and Adaptive Chemotactic Bacterial Swarm Foraging Optimization with Differential Evolution Strategy (ACBSFO_DES) ([26]). Compared algorithms were

[1]The graph of F_{18} was not printed out due to the huge gap in objective function values of algorithms.

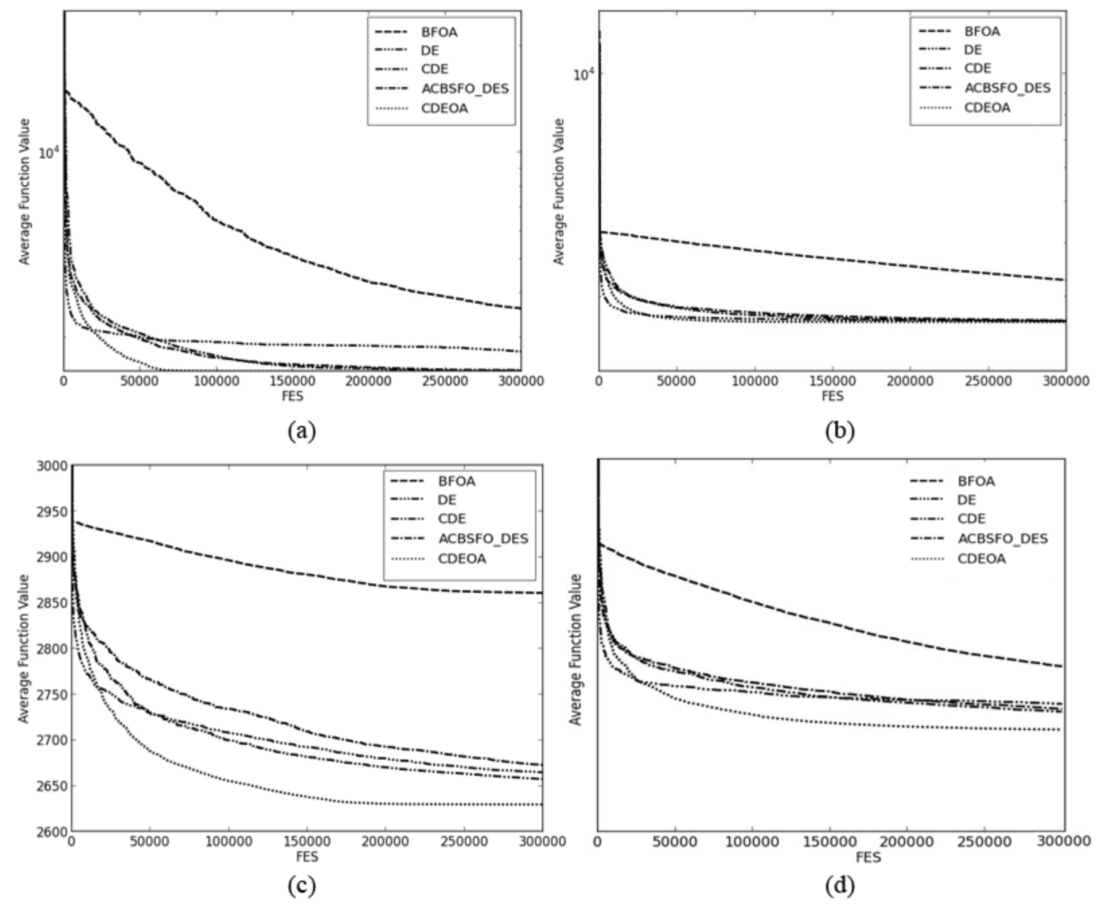
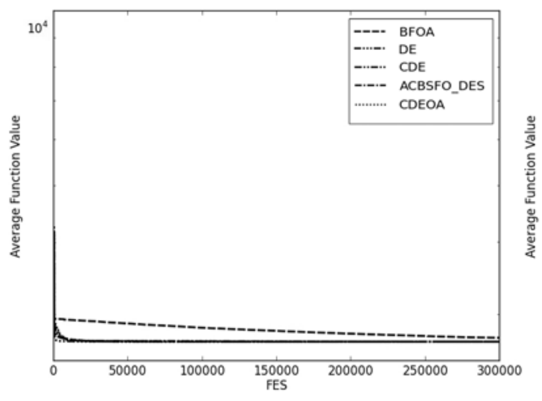


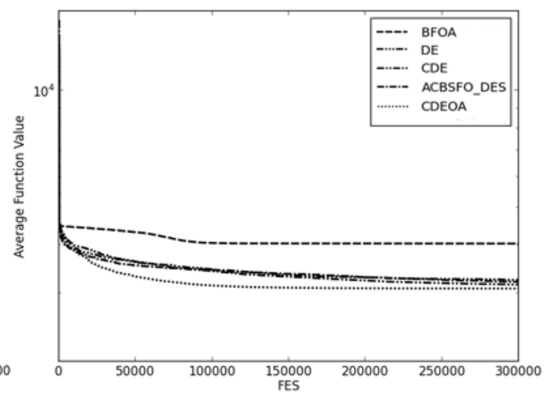
Figure 3.7: (a) F_{22}: Hybrid 6 (N=5); (b) F_{23}: Composition 1 (N=5); (c) F_{24}: Composition 2 (N=3); (d) F_{25}: Composition 3 (N=3).

chosen in accordance with operators that they employ in common. Classical BFOA employs elimination-dispersal, reproduction and chemotaxis; classical DE employs mutation, crossover and selection; CDE employs chemotaxis, mutation, crossover, and selection; ACBSFO_DES employs reproduction, elimination-dispersal, chemotaxis, mutation, crossover, and selection.

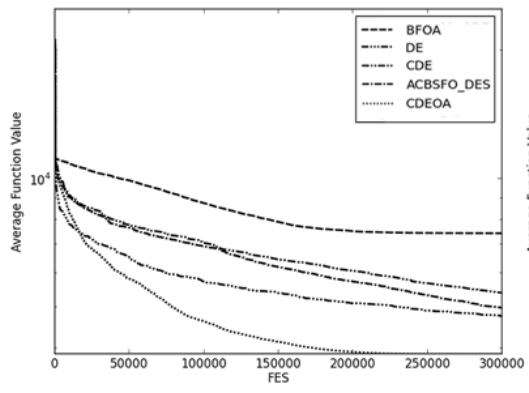
The experiment was performed in 30 dimensions with 25 runs for each algorithm-problem pair. The statistics were given in Appendix A.1 and A.2.



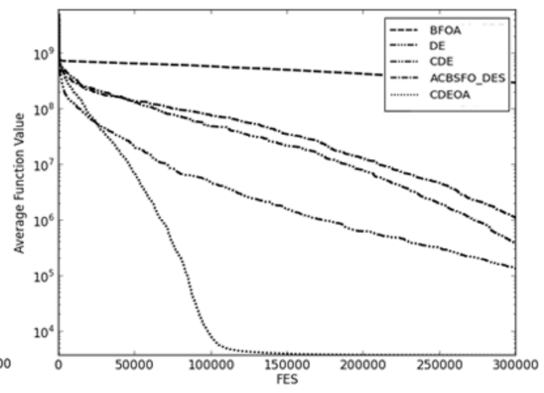
(a)



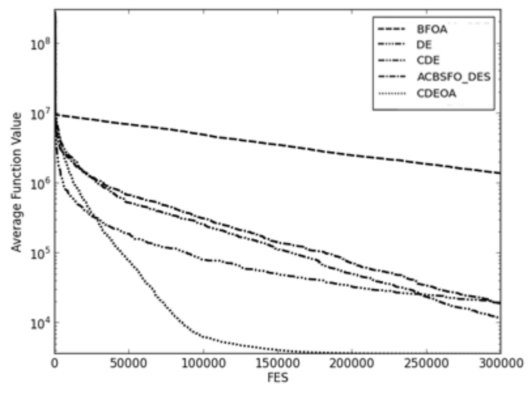
(b)



(c)



(d)



(e)

Figure 3.8: (a) F_{26}: Composition 4 (N=5);(b) F_{27}: Composition 5 (N=5); (b) F_{28}: Composition 6 (N=5); (c) F_{29}: Composition 7 (N=3); (e) F_{30}: Composition 8 (N=3).

Unimodal Functions F_1 - F_3

ACBSFO_DES outperforms CDEOA on the two unimodal functions F_1 and F_2. CDEOA and ACBSFO_DES exhibit similar performance on one function and outperform the others.

Simple Multimodal Functions F_4 - F_{16}

CDEOA is remarkably better than BFOA, DE, CDE, and ACBSFO_DES on these 12 test functions. However, ACBSFO_DES performs better than CDEOA on F_4, F_{11}, F_{12}, and F_{16} test functions. It is interesting to note that, the five methods show similar performance on F_5 test function. We can clearly observe that BFOA, DE, and CDE fail on most functions, as well.

Hybrid Functions F_{17} - F_{22}

To obtain these hybrid functions, the variables are randomly divided into some subcomponents and then different basic multimodal and unimodal functions are used for different subcomponents. On these six functions, CDEOA exhibits better performance than four other methods. While CDE exhibits similar performance with CDEOA on F_{22} test function, ACBSFO_DES outperforms CDEOA on F_{17} and F_{18} test functions .

Composition Functions F_{23} - F_{30}

The composition functions merge the properties of the sub-functions better and maintain continuity around the global/local optima. We can observe that the performance of CDEOA method is superior overall to that of four competitors except on F_{23}, F_{24}, F_{25}, and F_{26} test functions which ACBSFO_DES, CDE, and DE perform comparable to CDEOA. In addition, the classical BFOA method catches up with CDEOA on F_{25} and F_{26} test functions.

In Fig. 3.2, Fig. 3.3, Fig. 3.4, Fig. 3.5, Fig. 3.6, Fig. 3.7, and Fig. 3.8 the convergence map of BFOA, DE, CDE, ACBSFO_DES, and CDEOA shows that CDEOA converges faster than others on F_6, F_8, F_9, and F_{10} problems while similar convergence performance on the rest of the problems. Although ACBSFO_DES shows better convergence than CDEOA on some functions, it gets trapped in local minima. We can observe that CDEOA is superior overall to that of four competitors in terms of quality of the final solution at the end of the fixed function evaluations.

In short, CDEOA is superior to the four methods as compared to simple multimodal functions, hybrid functions, and composition functions and it is the second best in unimodal functions.

Comparison of CDEOA with Four State-of-the-art DE

CDEOA was also compared with Differential Evolution Strategy based on the Constraint of Fitness Values Classification (FCDE) ([34]), Partial Opposition-Based Adaptive Differential Evolution (POBL-ADE) ([24]), Fireworks Algorithm with Differential Mutation (FWA-DE) ([71]), and Differential Evolution Algorithm based on Fitness Euclidean-distance ratio (FERDE) ([55]). Compared algorithms were chosen because they are all the variants of DE that competed in CEC 2014 on Single Objective Real Parameter Numerical Optimization Competition. For each algorithm-problem pair, the experiment was performed in 30 dimensions with 51 runs. The statistics were given in Appendix A.3 and A.4. The maximum number of FEs was set to 3 $\times 10^5$ in accordance with instructions in CEC 2014 special session. The numerical benchmark results were taken from the aforementioned studies.

Unimodal Functions F_1 - F_3

As depicted in Appendix A.3, CDEOA is the best among the five algorithms on these three unimodal functions. It outperforms FCDE and FERDE on two test functions and exhibits similar performance with FWA-DE and POBL-ADE.

Simple Multimodal Functions F_4 - F_{16}

FERDE is the best among the five methods on these functions. It outperforms CDEOA on six test functions. In contrast, CDEOA performs better than FERDE on two test functions.

Overall, CDEOA exhibits similar performance with FWA-DE. FCDE can not even outperform on any test function while POBL-ADE outperforms on one function.

Hybrid Functions F_{17} - F_{22}

On these six test functions, CDEOA is significantly better than that of the four methods. However, FERDE and FWA-DE outperform CDEOA on F_{22} test function. FCDE and POBL-ADE cannot be significantly better than CDEOA on any test function.

Composition Functions F_{23} - F_{30}

FWA-DE is the best among the five methods on these eight composition functions. It outperforms CDEOA on three test functions (i.e., F_{28}-F_{30}). CDEOA and FERDE exhibit similar performance and outperform FCDE and POBL-ADE.

In short, overall, CDEOA performs better than FCDE, POBL-ADE, and FWA-DE while it exhibits similar performance with FERDE.

3.2 Improved CDEOA

This section presents a novel variant of CDEOA, Improved Chemotaxis Differential Evolution Optimization Algorithm (ICDEOA) [74], to cope with premature convergence of reproduction process. In ICDEOA, reproduction operator of BFOA is replaced with probabilistic reposition operator to enhance the intensification and the diversification of the search space. ICDEOA was compared with state-of-the-art DE and non-DE variants on 7 numerical functions of the 2014 Congress on Evolutionary Computation (CEC 2014). Simulation results of CEC 2014 benchmark functions reveal that ICDEOA performs better than that of competitors in terms of the quality of the final solution in unimodal and multimodal for high dimensional problems.

3.2.1 Concept of ICDEOA

The concept of ICDEOA depends on two approaches in CDEOA ([73]): a) making "weak" bacteria more diversified, where "weak" bacteria are the ones in positions with nutrient-poor medium, and b) making "strong" bacteria more intensified, where "strong" bacteria are the ones in positions with nutrient-rich medium. Based on the aforementioned approaches, a new operator, probabilistic repositioning, which balances the exploration and the exploitation trade-off was introduced. The reproduction process of classical BFOA is replaced with probabilistic repositioning operator. Unlike the reproduction process, probabilistic repositioning operator retains the strong bacteria in the vicinity of the best bacterium, whereas the weak bacteria are dispersed to the random positions in the search space.

ICDEOA tends to improve the optimization performance of CDEOA. In this contribution, in place of reproduction operator of BFOA, the probabilistic repositioning operator which acts based on the bacterium's fitness (Algorithm 2, line 67) was employed. If the function value (cost) of a bacterium is high, the bacterium most likely will change its position (Algorithm 2, line 70). If the function value is low, the bacterium is moved to the vicinity of the best bacterium (Algorithm 2, line 73). Reproduction operator of CDEOA possesses intensive exploitation capability which may result in premature convergence since it chooses the best of the population and kills the rest for the next generation. The probabilistic repositioning operator may tend to prevent not only the premature convergence problem of reproduction operator, but also diversify the half of the population. The pseudo code of ICDEOA is shown in Algorithm 2.

3.2.2 Experimental Study

The study introduced in this section aims to test the quality of the final solution at the end of a fixed number of function evaluations (FEs). The maximum number of FEs was set to 3×10^5 for 30 dimensional functions with the population size S=50. The error function values were given in Appendix A.5. "Mean Error" and "Std Dev" in Appendix A.5 indicate the average and the standard deviation of the error values obtained in 25 runs. The CEC 2014 test functions

Algorithm 2 Detailed pseudo-code of ICDEOA. Comments start with "//". The code we discuss in the text is in boldface.

```
1: Parameters:
2:   $p \leftarrow$ dimensions of the search space
3:   $S \leftarrow$ total number of bacteria in the population
4:   $N_c \leftarrow$ number of chemotaxis steps
5:   $N_s \leftarrow$ swimming steps
6:   $N_r \leftarrow$ re-positioning steps
7:   $C(i) \leftarrow$ the run-length unit
8:   $M_t \leftarrow$ maximum number of tumble steps
9:   $M_r \leftarrow$ maximum number of run steps
10:  $f \leftarrow$ objective function to be minimized
11: //Initialize some local variables
12:  $E_t \leftarrow 0$ //bacterium's unsuccessful tumble step
13:  $E_r \leftarrow 0$ //bacterium's unsuccessful run step
14:  $\theta_{best} \leftarrow$random position in the search space
15:  $f_{best} \leftarrow f(\theta_{best})$
16:  $M_{fes} \leftarrow$maximum number of FEs allowed
17:  $N_{fes} \leftarrow 0$ //current number of function evaluations
18: // Define a helper function $J$ that will call the actual objective function $f$. This helper
    function also updates the $N_{fes}$, $\theta_{best}$, and $f_{best}$ variables.
19: **function** $J(\theta)$:
20:     $v \leftarrow f(\theta)$
21:     $N_{fes} \leftarrow N_{fes} + 1$//update number of FEs
22:     **if** $v < f_{best}$ **then**
23:         $\theta_{best} \leftarrow \theta$ //update global best location
24:         $f_{best} \leftarrow v$ //update global best function value
25:     end//if
26:     **return** $v$
27: end// function
28: **while** $N_{fes} < M_{fes}$ **do**//FEs control loop
29:     **for** $k$ from 1 to $N_{re}$ **do**// Re-position loop
30:     **for** $j$ from 1 to $N_c$ **do**// Chemotaxis loop
31:     **for** $i$ from 1 to $S$ **do**// Tumble-Swim loop
32:         $J_{last}$ $J(\theta(i,j,k))$//$J(\cdot)$ computes the fitness
33:         $\Delta(i)$ random vector within $[-1,1]$//tumble
34:         $\theta(i,j+1,k) = \theta(i,j,k) + C(i) * \frac{\Delta(i)}{\sqrt{\Delta^T(i)*\Delta(i)}}$
35:         **if** $J(\theta(i,j+1,k)) < J(\theta(i,j,k))$ **then**
36:             $\boldsymbol{E_t} \leftarrow \boldsymbol{E_t} + 1$
```

[1]

```
37:         //Swim:
38:         for $m$ from 1 to $N_s$ do// Swim loop
39:             if $J\left(\theta\left(i, j+1, k\right)\right) < J_{last}$ then
40:                 $J_{last} = J\left(\theta\left(i, j+1, k\right)\right)$
41:                 $\theta\left(i, j+1, k\right) = \theta\left(i, j, k\right) + C\left(i\right) * \frac{\Delta(i)}{\sqrt{\Delta^T(i) * \Delta(i)}}$
42:                 $\boldsymbol{E_r} \leftarrow \boldsymbol{E_r} + 1$
43:             else
44:                 $m = N_s$//Break from switch loop
45:             end//if
46:         end//Swim loop
47:     end//Tumble-Swim loop
48:     //Exploration Loop
49:     for $\boldsymbol{i}$ from 1 to $\boldsymbol{S}$ do
50:         //Take an exploration step for bacterium i
51:         if $\boldsymbol{E_t} = \boldsymbol{M_t}$ then
52:             $\boldsymbol{\theta}(\boldsymbol{i}, \boldsymbol{j}+1, \boldsymbol{k})$ random position
53:             $\boldsymbol{J_{last}} = \boldsymbol{J}(\boldsymbol{\theta}(\boldsymbol{i}, \boldsymbol{j}+1, \boldsymbol{k}))$
54:             if $\boldsymbol{J_{last}} < \boldsymbol{J}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})$ then
55:                 $\boldsymbol{J}(\boldsymbol{i}, \boldsymbol{j}+1, \boldsymbol{l}) \leftarrow \boldsymbol{J_{last}}$
56:             end//if
57:             $\boldsymbol{E_t} = 0$
58:         end//if
59:     end//Exploration Loop
60:     //Exploitation Loop
61:     for $\boldsymbol{i}$ from 1 to $\boldsymbol{S}$ do
62:         if $\boldsymbol{E_r} = \boldsymbol{M_r}$ let bacterium undergo : then
63:             DE mutation, crossover, and selection
64:         end//if
65:     end //Exploitation Loop
66:     end //Chemotaxis loop
67:     Re − positioning of all bacteria
68:     Prob ← Assign probabilities of each bacterium
69:     for $\boldsymbol{i}$, $\boldsymbol{e}$ in enumerate(Prob) do
70:         if $\boldsymbol{e} >$ random number within $[0, 1]$ then
71:             $\boldsymbol{\theta}\left(\boldsymbol{i}, \boldsymbol{j}+1, \boldsymbol{k}\right) \leftarrow$ to a random location
72:         else
73:             $\boldsymbol{\theta}\left(\boldsymbol{i}, \boldsymbol{j}+1, \boldsymbol{k}\right) \leftarrow$ to the vicinity of the best bacterium
74:     end//Re − positioning
75: end // FEs control loop
76: return $\theta_{best}$
```

are as follows: F_1=Rotated high conditioned Elliptic, F_2=Rotated Bent Cigar, F_3=Rotated discus, F_4= Shifted and rotated Rosenbrock, F_5=Shifted and rotated Ackley, F_6=Shifted and rotated Weierstrass, and F_7=Shifted and rotated Griewank ([35]).

3.2.3 Comparison with Three State-of-the-art DE and One Non-DE

The performance of the ICDEOA algorithm was compared with OptBees which is inspired by the bee colonies ([39]), Memetic Differential Evolution based on Fitness-Euclidean distance Ratio (FERDE) ([55]), Differential Evolution with Replacement Strategy for Real-Parameter Numerical Optimization (RSDE) ([66]), and CDEOA ([73]).

Unimodal Functions F_1- F_3

As presented in Appendix A.5, overall, ICDEOA is better than that of four methods on these three unimodal functions. It outperforms OptBees on 2, FERDE on 2, RSDE on 1, and CDEOA on 2 test functions. In contrast, FERDE and RSDE perform better than ICDEOA on test function F_1. ICDEOA also exhibits similar performance with OptBees, RSDE, and CDEOA on test function F_2.

Multimodal Functions F_4- F_7

On these four multimodal test functions, ICDEOA outperforms OptBees on 3, FERDE on 2, RSDE on 2, and CDEOA on 2 test functions. FERDE, RSDE, and CDEOA exhibit better performance than ICDEOA on test function F_7. Overall, ICDEOA performs better than OptBees, FERDE, RSDE, and CDEOA.

3.3 Supply Chain Cost Problem

In this section, CDEOA has been employed to optimize the Supply Chain Cost problem [72] by comparing the performance with other well-known algorithms; Particle Swarm Optimization

(PSO), Bacterial Foraging Optimization Algorithm (BFOA), Tabu Search (TS) [21], Bat Algorithm (BA) [67], and Genetic Algorithm (GA) .

In order to maximize the profits and minimize the cost, there has been always a research in the business world. This requirement unveiled a new optimization problem known as Supply Chain Cost Problem. When large numbers of decision variables and alternatives exist, these kinds of problems are identified as non-deterministic polynomial-time hard (NP-hard) problems and they need more complex optimization algorithms to guide the search for optimum or near-optimum solutions [47]. In this context, random search techniques have been popular in solving computationally complex (NP-hard) problems due to their ability to find effective solutions in a short amount of time. In this field, Castillo [11] has proposed a novel capacitated Supply Chain Network Design (SCND) model which evaluates the overall economic profit of the supply chain with a metaheuristic-based approach. Castillo [10] has also presented an extensive study by analyzing the application of metaheuristics to solve bio-energy supply chain models.

A supply chain is a dynamic supply and demand network of globally distributed organizations, activities, people, and resources that provide the materials; transform these materials into products, and distribute these products to retailers or customers. The architecture of a supply chain as illustrated in Fig. 3.9 is: suppliers, producers, warehouses, retailers, and customers. Suppliers provide the unprocessed materials to the producers; producers convert the unprocessed materials to end products. By means of the warehouses the products are transferred from producers to retailers, and retailers sell these products to the end customers [40].

Minimizing the total cost, maximizing the profit and fulfilling customers' needs while ensuring satisfaction has been studied by researchers [22] in terms of designing, analyzing, and managing of supply chain. Many companies are concerned about analyzing their supply chain as a whole system to improve their business. However, the process of analyzing and managing the supply chain has been performed based on experience and intuition. This implies that finding the best supply chain strategies for a particular firm is a significant issue for industry. In this context, bio inspired and nature inspired metaheuristics algorithm may play an important role

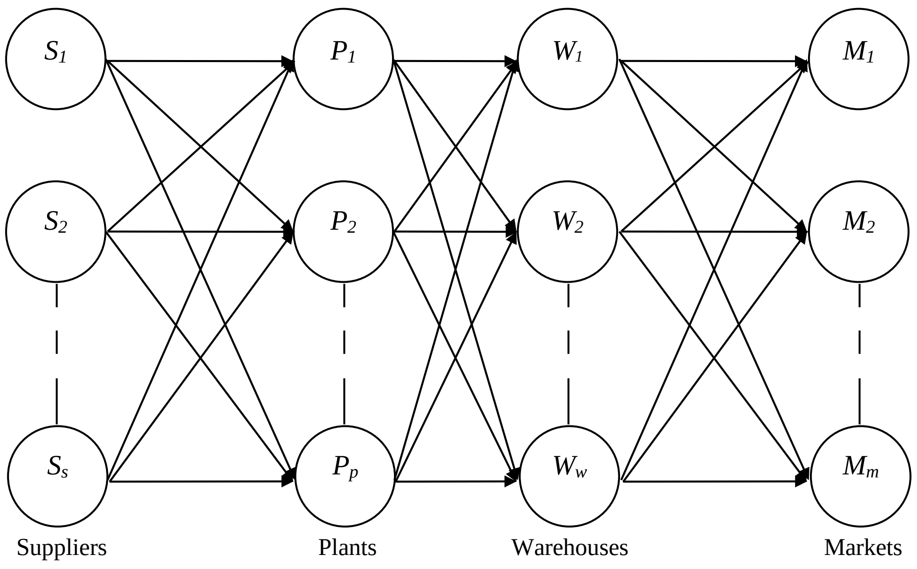


Figure 3.9: Architecture of a supply chain

in helping managers and consultants in the decision-making process. Recently, metaheuristic algorithms have been broadly employed for optimizing NP-hard since they are simple, easy to implement, robust, and have been proven highly effective to solve complex problems [57]. The total cost of Eq. 3.1 of a globally distributed supply chain [22] is composed of supply cost of raw material (SCRM), cost of production (PC), cost associated with warehouses (WAC), and cost of markets (MC).

$$Total \quad Cost(TC) = SCRM + PC + WAC + MC \tag{3.1}$$

The mathematical programming formulation that minimizes the total supply chain total cost (TC) is presented in Eq. 3.1 and considers all supplier, plant, warehouse, and market costs.

3.3.1 Experimental Study

Five different supply chain scenarios were employed to test CDEOA. In each scenario, production capacity of suppliers and plants, capacity of warehouses, and demand of each market were created randomly and their complexities were increased. The scenarios are presented in Table 3.2. In scenario 2: There are 3 suppliers with production capacities, 500, 500, and 1000 units; 3 plants with production capacities, 600, 400, and 400 units; 5 warehouses with storage capacities, 400 300, 250, 250, and 200 units; and 5 markets with demands, 100, 100, 200, 70, and 30 units respectively.

Table 3.2: Supply chain problem scenarios

	1	2	3	4	5
Suppliers	1000;1000	500;500;1000	500;500;1000	500;500 750;250	500;500;250 250;250;250
Plants	600;400;400	600;400;400	600;200 200;400	300;300;200 200;100;300	300;300;200;200 100;150;150
Warehouses	400;300;500;200	400;300;250 250;200	200;200;300 250;250;200	200;200;300 250;250;200	200;200;300 250;250;200
Markets	100;100;200 70;30	100;100;200 70;30	100;50;50 200;70;30	100;50;50 200;70;30	100;50;50 200;70;30

3.3.2 Compared Algorithms and Parametric Setup

The performance of the CDEOA was compared with BFOA, PSO, TS, BA, and GA. The study aims to test the quality of final solution and convergence speed at the end of a fixed number of function evaluations (FEs). The objective is to minimize the total cost of the supply chain operation which includes supplier cost, production cost, warehouse associated cost, and market cost. Each scenario has a distinct number of dimensions depending on the complexity of the supply chain. In this context, scenarios have 38, 54, 72, 96, and 120 dimensions, respectively. The maximum number of FEs was set to 1000. The population size of each algorithm was set to 20. Each algorithm was run 30 times. For PSO, we employed the standard PSO and set the inertia weight w = 1 and acceleration coefficients c_1= c_2 = 2 according to [18]. As for the BA, the algorithmic constants $\alpha = \gamma = 0.5$, frequency is in the range [0, 2], and loudness $A_0 = 0.5$. For GA, mutation probability = 00.5 and crossover probability = 0.95. BFOA and CDEOA employ the same parameters as follows: N_c= 100, N_s= 16, N_{re}= 8, $C(i)$= 0.1.

Table 3.3: Comparison of PSO, BFOA, TS, GA, BA, and CDEOA on Supply Chain Cost problem.

Scenarios	**Dimensions**	**Algorithms**	**BCV**	**WCV**	**Mean**	**StDev**	**Median**	**Time(sec)**
1	38D	PSO	171774.07	406389.87	271898.87	61422.80	262674.67	2.75
		BFOA	158073.95	328604.65	256788.83	45965.09	245733.44	2.64
		TS	167997.11	448938.90	265207.46	60890.62	253209.65	3.26
		BA	149057.04	374563.73	264678.66	60600.65	267249.05	10.16
		GA	138873.38	319263.21	247192.37	46310.93	252379.22	3.30
		CDEOA	131436.66	483223.62	263664.37	66805.99	262662.79	3.65
2	54D	PSO	221549.15	424156.34	316044.20	50554.57	319207.67	16.89
		BFOA	177070.27	562895.49	339305.96	91896.03	323970.03	16.55
		TS	150616.39	468185.32	318433.36	73805.63	313026.23	21.98
		BA	173366.76	486141.78	319110.49	78046.28	308508.69	21.43
		GA	209731.19	498573.87	342671.21	65858.21	340691.22	22.19
		CDEOA	178180.85	536455.05	301219.47	79367.57	303121.86	16.79
3	72D	PSO	314819.83	698583.76	501751.33	101797.08	504006.41	5.59
		BFOA	301504.10	715808.25	500714.51	108195.15	508918.69	2.45
		TS	325904.98	657693.39	498861.93	85473.23	510978.49	3.05
		BA	289262.97	743135.80	488318.85	91721.61	490993.25	16.36
		GA	323051.57	725538.68	536753.61	82662.90	528767.20	6.24
		CDEOA	281094.41	739537.04	493928.04	115114.72	483599.93	3.07
4	96D	PSO	685639.86	870112.42	789663.58	63665.56	809801.15	4.83
		BFOA	530568.81	944208.98	741662.90	136583.12	742537.79	5.00
		TS	497928.05	981825.53	740919.58	149498.71	733425.54	5.55
		BA	491848.56	978094.90	710897.41	123887.43	683134.40	26.38
		GA	578853.52	1061163.49	760919.39	140865.37	700789.34	6.24
		CDEOA	477030.32	1019036.83	709229.15	159874.33	669061.34	5.01
5	120D	PSO	1007800.99	1619948.08	781593.63	164448.82	414210.04	28.54
		BFOA	1023073.99	1642535.82	777055.16	165842.10	392991.58	29.42
		TS	1038346.98	1665123.56	772516.68	167235.38	371773.11	30.29
		BA	1053619.97	1687711.30	767978.21	168628.66	350554.65	31.16
		GA	1068892.96	1710299.05	763439.74	170021.94	329336.19	32.04
		CDEOA	1084165.96	1732886.79	758901.27	171415.22	308117.73	32.91

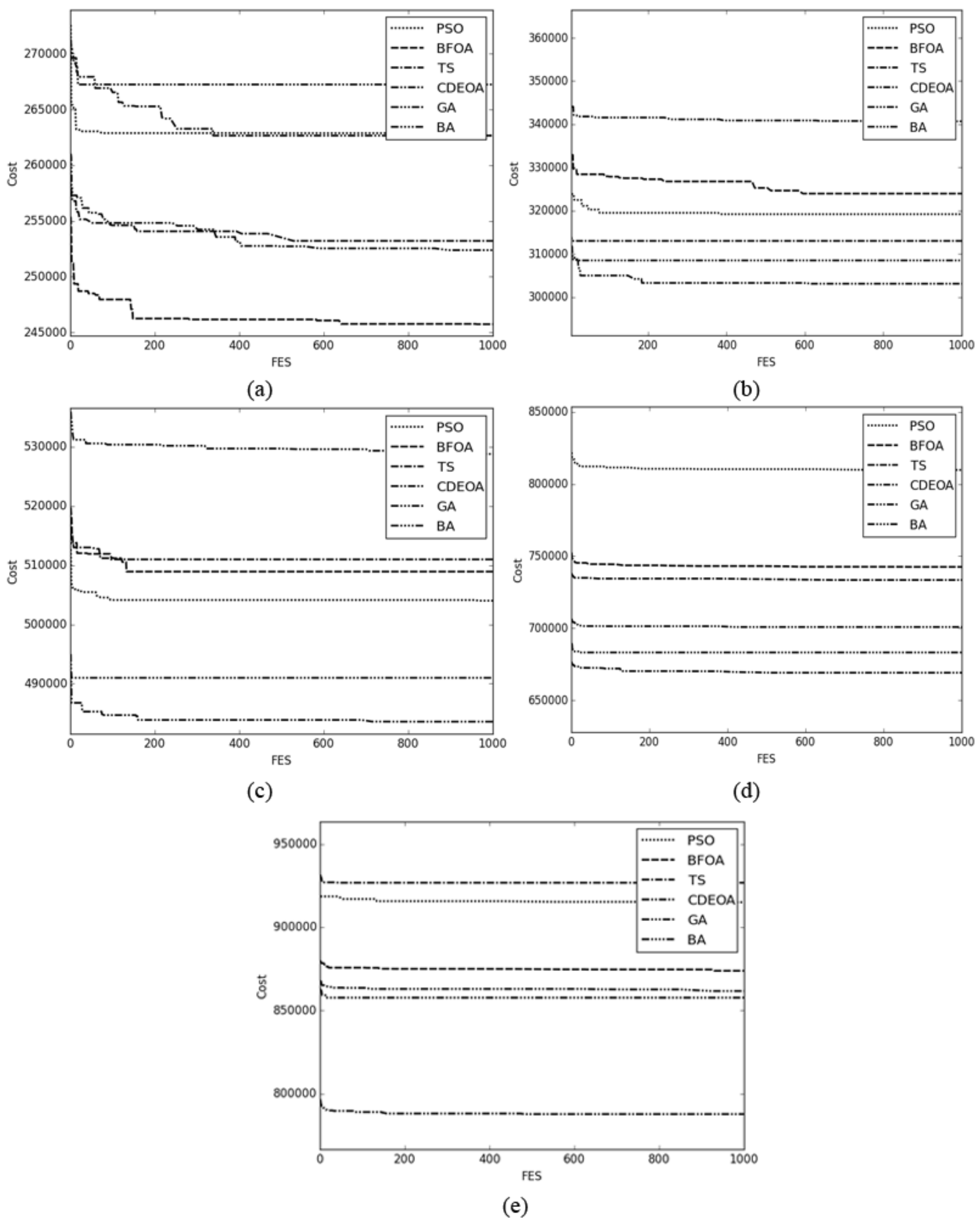


Figure 3.10: The convergence map of PSO, BFOA, TS, GA, BA, and CDEOA (a) Scenario 1; (b) Scenario 2; (c) Scenario 3; (d) Scenario 4; (e) Scenario 5.

3.3.3 Discussions

Table 3.3 reports the best cost value (BCV), the worst cost value (WCV), mean of the final best function values, the standard deviation of the final best function values (STDEV), median of the final best function values, and the mean time spent per trial in seconds. In Table 3.3, we can observe that CDEOA is superior overall to five algorithms in five different scenarios. We can infer the success of CDEOA in these scenarios may be due to its capability of balancing the exploration and exploitation with the aforementioned two CDEOA strategies. CDEOA outperforms its five competitors except GA in the first scenario. Worst cost value (WCV) of PSO shows better performance than CDEOA in all scenarios.

In Fig. 3.10, the convergence map of PSO, BFOA, TS, GA, BA, and CDEOA shows that the CDEOA overall led faster convergence than its competitors in all scenarios. It did fail against BFOA, GA, and TS in the first scenario; however, these three algorithms were unable to maintain the same performance in the rest of the scenarios. In addition, we can also observe from scenario 1 of Table 3.3, the aforementioned algorithms slightly outperformed CDEOA.

CHAPTER 4
MICRO BIO AND NATURE INSPIRED OPTIMIZATION ALGORITHMS

In the past few decades, the micro bio and nature inspired algorithms have been studied by several researchers in order to solve high dimensional optimization problems. High dimensionality makes the problems hard and computational time consuming due to the fact that it increases the number of parameters to be optimized. In this context, in case that the population size remains large as in its original algorithm, it would not be that easy for the parameters to converge to the optimal values. As a remedy to this challenge, Krishnakumar [33] has proposed to use micro genetic algorithm (µGA) based on a very small population approach. It is clear that although an algorithm with small population size such as (e.g. 2, 3, 5, or 6) is good at exploiting the promising areas of the search space, it is not able to preserve the diversity of population. However, when the diversity of population fails, the population can be reinitialized and the best individuals are kept on the search space. This not only leads to prevent the premature convergence but also makes the individuals explorative [44].

Micro algorithms have proved to be an efficient tool in solving optimization problems for high dimensional (e.g. 500, 750, and 1000) problems that standard nature-inspired and bio-inspired techniques fail. Recently, several studies have been conducted regarding the micro bio and nature inspired algorithms to solve the high dimensional optimization problems. Caraffino et al. [9] have proposed micro Differential Evolution (µDE) that incorporates an extra search move into DE to improve the best solution. Chu et al. [12] proposed

Fast Bacterial Swarming Algorithm that hybridizes BFOA and PSO. Parsopoulos [48] has proposed a cooperative micro technique, Cooperative Micro Differential Evolution, to solve high dimensional problems. Parsopoulos et al. [49] have also introduced a parallel master-slave model for cooperative micro-particle swarm optimization approach. Olorunda [46] has presented cooperative differential evolution that divides the high dimensional problem space into smaller parts and have each part optimized by a separate population. Sotelo-Figueroa [59] have proposed a novel approach called Micro Differential Algorithm that evolves an indirect representation of bin packing problem. Fuentes et al. [8] have presented a particle swarm optimizer that solves constrained optimization problems. Also, Rahnamayan et al. [56] have proposed micro Opposition based DE that deals with minimization of dissimilarity between the input grey-level image and the bi-level (thresholded) image in image processing field. Olguin-Carbajal et al. [45] have proposed the micro DE Local Search that incorporates local search technique into micro DE.

In this chapter, a micro Chemotaxis Differential Evolution Optimization Algorithm (µCDEOA) [69] which hybridizes BFOA and DE was proposed. The inspiration was taken from the ideas of micro Bacterial Foraging Optimization Algorithm (µBFOA) [14] which is successfully used to solve high-dimensional optimization problems. µBFOA does not use the reproduction operator to avoid premature convergence whereas the chemotaxis operator is employed for updating the position of a bacterium. In µBFOA, which uses three bacteria, the best bacterium retains its position in the swarm; the second best bacterium is re-positioned in the vicinity of the best bacterium; and the third bacterium is dispersed to a random location. This approach aims to avoid premature convergence and helps to maintain the search diversity. In this study, in order to increase the convergence performance and quality of the final solution, after re-initializing the population, the bacteria are ranked according to their cost function values. The best bacterium's position is preserved in the population. The second best bacterium is reinitialized in the neighborhood of the best bacterium based on the ideas of DE technique, whereas the rest of the bacteria (4 bacteria) are dispersed at random on the search space.

4.1 Micro CDEOA

In µCDEOA, a population of 6 bacteria which make consecutive *tumble* and *run* steps (chemotaxis) throughout their lifetime. After a chemotaxis loop, all the bacteria are sorted according to their objective function values. A bacterium which is close to the global optimum is called the best bacterium (rank 1). The second best bacterium (rank 2) attempts to approach the neighborhood of the best bacterium through the means of DE operators (mutation, crossover, and selection). The rest of the population (4 bacteria) are dispersed to the random positions in the search space. Unlike the population size of µBFOA, the population size of µCDEOA is increased to an appropriate value, 6, due to the number of the individuals chosen in mutation strategies (Eq. 2.6, Eq. 2.8, Eq. 2.9, and Eq. 2.10). The second best bacterium (rank 2) is positioned in the vicinity of the best bacterium. This is carried out through the means of DE mutation strategies. In this study, we have employed DE/best/1 (Eq. 2.9 in Chapter 2) mutation strategy which yields a best solution based trial vector.

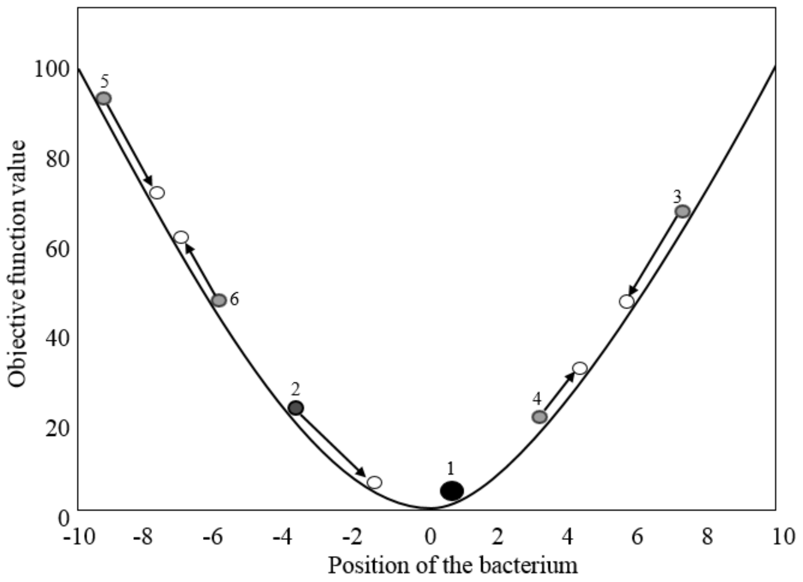


Figure 4.1: Behavior of the bacteria on one dimension

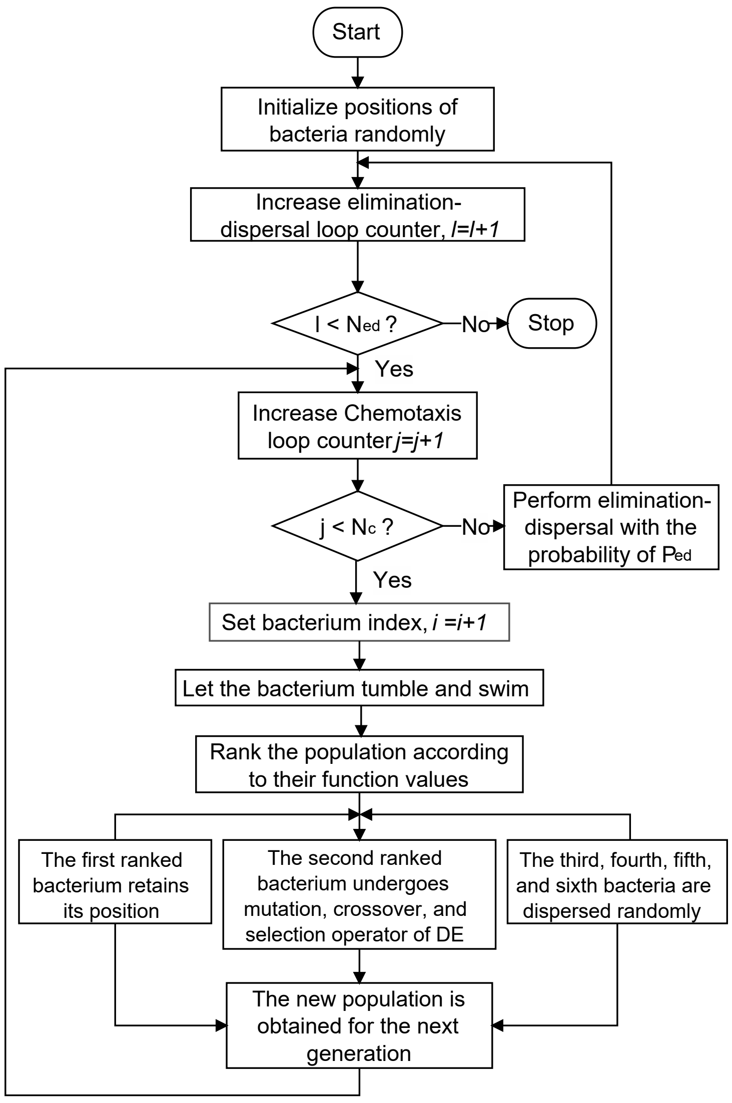

Start
Initialize positions of bacteria randomly
Increase elimination-dispersal loop counter, l=l+1
l < Ned ?
No
Stop
Yes
Increase Chemotaxis loop counter j=j+1
j < Nc ?
No
Perform elimination-dispersal with the probability of Ped
Yes
Set bacterium index, i =i+1
Let the bacterium tumble and swim
Rank the population according to their function values
The first ranked bacterium retains its position
The second ranked bacterium undergoes mutation, crossover, and selection operator of DE
The third, fourth, fifth, and sixth bacteria are dispersed randomly
The new population is obtained for the next generation

Figure 4.2: Flowchart of the μCDEOA

In order to figure out the behavior of the virtual bacteria in BFOA, we illustrated the six bacteria in a one dimensional search space in Fig. (4.1). The objective is the minimization of 1-dimensional sphere function Eq. (4.1) which is a widely employed unimodal function with a minimum equal to 0.

$$F\left(x\right) = x^2 \tag{4.1}$$

In Eq. (4.1), the parameter x is the position of a bacterium, and $f(x)$ is the objective function value. The roles of six bacteria may change after a chemotaxis process. The closest position to the global optimum of the search space is retained by the best bacterium (rank 1). The dispersal of the second best bacterium (rank 2) to a position which is close to the best one (rank 1) will ease local search for the next chemotaxis process. Maintaining the population diversity and avoiding premature convergence are performed by the worst bacteria (rank 3-6). A flowchart of the micro BFOA adapted from Dasgupta [14] is given in Fig. 4.2.

4.2 Experimental Study

The µCDEOA was tested using a set of 16 unimodal and multimodal benchmark functions (see Section 4.2.1) taken from IEEE CEC special sessions and competitions on single objective real parameter numerical optimization [62], [35]. Unlike standard benchmark functions, the shifted functions shift the global optimum to a random position, i.e., $F(x) = f(x - o_{new})$, where $F(x)$ is the new function, $f(x)$ is the old function, and o_{new} is the new global optimum with different values for different dimensions. Its global optimum is not situated at the center of the search space. The rotated functions rotate the function $F(x) = f(Mx)$, where M is an orthogonal rotation matrix [36]. The descriptions of these functions are given in Table 4.1. Functions 1-6 are unimodal and functions 7-16 are simple multimodal functions. $\boldsymbol{O}$ is the shifted vector; C is the characteristics of test functions; U is unimodal; M is multimodal; S is separable; and N is non-separable functions.

Table 4.1: Global optimum, the search ranges and the global best (f(x*)) of 500 dimensional test functions.

f	Global Optimum x*	f(x*)	C	Search Range
F_1	$\boldsymbol{0}$	0	US	(-2,2)
F_2	(0,0,...,0)	0	UN	(-500,500)
F_3	$\boldsymbol{0}$	0	UN	(-500,500)
F_4	$\boldsymbol{0}$	0	UN	(-500,500)
F_5	(0,0,...,0)	0	UN	(-500,500)
F_6	$\boldsymbol{0}$	0	UN	(-100,100)
F_7	$\boldsymbol{0}$	0	MN	(-2,2)
F_8	$\boldsymbol{0}$	0	MN	(-2,2)
F_9	$\boldsymbol{0}$	0	MN	(-2,2)
F_{10}	$\boldsymbol{0}$	0	MN	(-2,2)
F_{11}	$\boldsymbol{0}$	0	MN	(-10,10)
F_{12}	$\boldsymbol{0}$	0	MN	(-10,10)
F_{13}	$\boldsymbol{0}$	0	MS	(-2,2)
F_{14}	$\boldsymbol{0}$	0	MN	(-2,2)
F_{15}	(420.96,...,420.96)	0	MN	(-2,2)
F_{16}	(420.96,...,420.96)	0	M	(-500,500)

4.2.1 Shifted and Rotated Test Functions

Eq. 4.2, Eq. 4.3, Eq. 4.4, Eq. 4.5, Eq. 4.6, Eq. 4.7, Eq. 4.8, Eq. 4.9, Eq. 4.10, Eq. 4.11, Eq. 4.12, Eq. 4.13, Eq. 4.14, Eq. 4.15, Eq. 4.16, and Eq. 4.17 are the function definitions of the problems, respectively.

1. Shifted Sphere Function

$$F_1(x) = \sum_{i=1}^{D} z_i^2 \tag{4.2}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

2. Schwefel problem 1.2

$$F_2(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2 \tag{4.3}$$

3. Shifted Schwefel problem 1.2

$$F_3(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2 \tag{4.4}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

4. Shifted Schwefel problem 1.2 with noise in fitness

$$F_4(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2 * (1 + 0.4|N(0,1)|) \tag{4.5}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

5. Schwefel problem 2.21

$$F_5(x) = max\left\{|x_i, 1 \leq i \leq D|\right\} \tag{4.6}$$

6. Shifted and rotated high conditioned elliptic function

$$F_6 = \sum (10^6)^{\frac{i-1}{D-1}} z_i^2 \tag{4.7}$$

$z = M(x - o)$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

7. Shifted Rosenbrock's function

$$F_7(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{4.8}$$

$z = x - o + 1$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

8. Shifted and rotated Rosenbrock's function

$$F_8(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{4.9}$$

$z = M(x - o)$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

9. Shifted Ackley's function

$$F_9(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2)} - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i)) + 20 + e, \tag{4.10}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

10. Shifted rotated Ackley's function

$$F_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i)) + 20 + e, \tag{4.11}$$

$z = M(x - o)$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

11. Shifted Griewank's function

$$F_{11}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} \tag{4.12}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

12. Shifted and rotated Griewank's function

$$F_{12}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} \tag{4.13}$$

$z = M(x - o)$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

13. Shifted Rastrigin's function

$$F_{13}(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) \tag{4.14}$$

$z = x - o$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

14. Shifted and rotated Rastrigin's function

$$F_{14}(x) = \sum_{i=1}^{D}(z_i^2 - 10cos(2\pi z_i) + 10) \tag{4.15}$$

$z = M(x - o)$

$o = [o_1, o_2, ...o_D]$: shifted global optimum

15. Shifted noncontinuous Rastrigin's function

$$F_{15}(x) = \sum_{i=1}^{D}(y_i^2 - 10cos(2\pi y_i) + 10) \tag{4.16}$$

$$y_i = \begin{cases} round(2z_i)/2, & |z_i| >= 1/2 \\ z_i, & |z_i| < 1/2 \end{cases}$$

`for i  = 1,2,...,D`

$o = [o_1, o_2, ...o_D]$: shifted global optimum

16. Shwefel's function

$$F_{16}(x) = 418.9829 * D - \sum_{i=1}^{D} x_i sin(|x_i|^{1/2}) \tag{4.17}$$

4.2.2 Parametric Setup

The same parameter values have been used as in the original papers in each technique. DE is sensitive to the mutation scaling factor F and crossover rate C_r. Choosing C_r=0.9 or 1.0 not only speeds up convergence but also diversifies the population by means of one of the best solution dependent DE mutation strategies, “DE/best/1”. In this context, C_r parameter of DE was set to be 0.9. F is selected within the range of $[0 - 2.0]$. It is reported that a smaller F value (e.g., 0.5) can lead to a statistically better performance than the other parameter values ([60], [54]). Therefore, F of µ-CDEOA was set to be 0.5. For the proposed technique and the classical BFOA, the following parameter values were set: N_s=12, N_{re}=16, $C(i)$=0.1.

4.2.3 Simulation

The study introduced in this section aims to test the quality of the final solution and the convergence speed at the end of a fixed number of function evaluations (FEs). The maximum number of FEs was set to 5×10^3. All simulations were done on 500 dimensional problems. Each algorithm and the objective function pair were run 50 times. The convergence graph was plotted in Fig. 4.3, Fig. 4.4, Fig. 4.6, and Fig. 4.5. The horizontal axis of these graphs is the number of function evaluations and the vertical axis is the mean of function values. Table 4.2 and Table 4.3 report the best final function value (BFV), the worst final function value (WFV), the mean of the final best function value (Mean), the standard deviation of the final best function value (StdDev), and the median of the final best function value (Median). These values comply $(F(x) - F(x^*))$ for evaluating the success of five algorithms, where $\vec{x}$ is the best value of the bacterium in a run and $\vec{x}^*$ is the global best of the test function (Table 4.1). The standard deviation of the final best function value and the mean time spent per trial in seconds are also reported.

Table 4.2: Comparison of BFOA-6, BFOA-30, µBFOA, and µ-CDEOA in F_1 through F_8.

Functions	Algorithm	BFV	WFV	Mean	StdDev	Med	Time
F_1	BFOA-6	9.17E+02	1.05E+03	9.95E+02	3.22E+01	9.96E+02	0.47
	BFOA-30	1.07E+03	1.16E+03	1.12E+03	2.3E+01	1.12E+03	0.56
	µ-BFOA	6.77E+02	9.05E+02	7.91E+02	5.97E+01	7.96E+02	0.65
	µ-CDEOA	7.16E+02	8.68E+02	7.82E+02	3.93E+01	7.80E+02	0.55
F_2	BFOA-6	2.33E+09	2.37E+09	2.35E+09	7.48E+06	2.35E+09	27.61
	BFOA-30	1.69E+09	1.69E+09	1.69E+09	9.80E+05	1.69E+09	27.77
	µ-BFOA	2.26E+08	6.11E+08	4.24E+08	7.93E+07	4.27E+08	28.08
	µ-CDEOA	1.17E+08	5.28E+08	2.92E+08	8.65E+07	2.89E+08	27.32
F_3	BFOA-6	6.27E+12	6.27E+12	6.27E+12	5.04E+08	6.27E+12	27.74
	BFOA-30	6.27E+12	6.27E+12	6.27E+12	2.58E+08	6.27E+12	28.35
	µ-BFOA	4.47E+12	5.70E+12	5.07E+12	2.74E+11	5.04E+12	28.42
	µ-CDEOA	5.23E+12	6.18E+12	5.60E+12	2.19E+11	5.63E+12	28.43
F_4	BFOA-6	7.14E+12	7.17E+12	7.14E+12	6.51E+09	7.14E+12	28.33
	BFOA-30	7.14E+12	7.22E+12	7.16E+12	1.82E+10	7.15E+12	28.27
	µ-BFOA	6.91E+12	7.18E+12	7.08E+12	5.85E+10	7.09E+12	28.23
	µ-CDEOA	6.95E+12	7.21E+12	7.11E+12	4.86E+10	7.11E+12	28.52
F_5	BFOA-6	9.81E+01	9.87E+01	9.84E+01	1.24E-01	9.85E+01	0.91
	BFOA-30	9.85E+01	9.90E+01	9.87E+01	1.25E-01	9.88E+01	0.86
	µ-BFOA	9.51E+01	9.80E+01	9.73E+01	5.85E-01	9.74E+01	0.96
	µ-CDEOA	7.30E+01	8.94E+01	8.21E+01	3.55E+00	8.17E+01	0.90
F_6	BFOA-6	1.46E+11	1.51E+11	1.51E+11	9.73E+08	1.51E+11	1.14
	BFOA-30	1.14E+11	1.17E+11	1.17E+11	3.68E+08	1.17E+11	1.12
	µ-BFOA	8.33E+10	1.11E+11	1.02E+11	6.92E+09	1.03E+11	1.27
	µ-CDEOA	7.21E+10	1.08E+11	8.84E+10	7.77E+09	8.86E+10	1.15
F_7	BFOA-6	1.30E+06	1.62E+06	1.47E+06	6.13E+04	1.47E+06	0.55
	BFOA-30	1.60E+06	1.82E+06	1.73E+06	4.47E+04	1.73E+06	0.55
	µ-BFOA	3.57E+05	5.79E+05	4.90E+05	5.58E+04	4.97E+05	0.59
	µ-CDEOA	8.13E+05	1.25E+06	1.09E+06	7.43E+04	1.10E+06	0.52
F_8	BFOA-6	5.16E+06	6.87E+06	5.91E+06	3.41E+05	5.87E+06	1.27
	BFOA-30	6.43E+06	7.52E+06	7.07E+06	2.52E+05	7.08E+06	1.28
	µ-BFOA	1.32E+06	2.59E+06	2.13E+06	2.80E+05	2.20E+06	1.42
	µ-CDEOA	3.68E+06	4.91E+06	4.16E+06	2.91E+05	4.14E+06	1.11

Table 4.3: Comparison of BFOA-6, BFOA-30, µBFOA, and µ-CDEOA in F_9 through F_{16}.

Functions	Algorithm	BFV	WFV	Mean	StdDev	Med	Time
F_9	BFOA-6	6.29E+00	6.69E+00	6.49E+00	8.88E-02	6.50E+00	0.56
	BFOA-30	6.69E+00	6.91E+00	6.83E+00	4.79E-02	6.83E+00	0.54
	µ-BFOA	5.90E+00	6.46E+00	6.22E+00	1.20E-01	6.22E+00	0.63
	µ-CDEOA	5.91E+00	6.32E+00	6.09E+00	8.63E-02	6.06E+00	0.78
F_{10}	BFOA-6	8.52E+00	8.97E+00	8.80E+00	1.23E-01	8.83E+00	1.39
	BFOA-30	8.76E+00	9.06E+00	8.92E+00	5.31E-02	8.92E+00	1.37
	µ-BFOA	8.01E+00	8.67E+00	8.41E+00	1.34E-01	8.44E+00	1.83
	µ-CDEOA	7.65E+00	8.22E+00	7.88E+00	1.28E-01	7.87E+00	1.18
F_{11}	BFOA-6	8.13E+00	8.56E+00	8.39E+00	8.78E-02	8.40E+00	1.24
	BFOA-30	8.50E+00	8.74E+00	8.63E+00	6.03E-02	8.64E+00	1.11
	µ-BFOA	5.65E+00	7.19E+00	6.36E+00	3.74E-01	6.29E+00	1.39
	µ-CDEOA	5.81E+00	6.84E+00	6.23E+00	2.06E-01	6.22E+00	1.07
F_{12}	BFOA-6	1.66E+01	1.75E+01	1.71E+01	1.98E-01	1.71E+01	1.75
	BFOA-30	1.72E+01	1.79E+01	1.77E+01	1.78E-01	1.76E+01	1.78
	µ-BFOA	1.15E+01	1.50E+01	1.29E+01	8.26E-01	1.27E+01	2.19
	µ-CDEOA	1.15E+01	1.34E+01	1.25E+01	4.14E-01	1.26E+01	2.09
F_{13}	BFOA-6	1.02E+04	1.16E+04	1.10E+04	2.85E+02	1.11E+04	0.54
	BFOA-30	1.18E+04	1.23E+04	1.20E+04	1.02E+02	1.20E+04	0.51
	µ-BFOA	9.52E+03	1.09E+04	1.03E+04	3.39E+02	1.03E+04	0.65
	µ-CDEOA	8.94E+03	1.01E+04	9.53E+03	2.80E+02	9.49E+03	0.63
F_{14}	BFOA-6	1.95E+04	2.12E+04	2.05E+04	4.52E+02	2.06E+04	1.41
	BFOA-30	2.06E+04	2.15E+04	2.11E+04	2.09E+02	2.11E+04	1.31
	µ-BFOA	1.66E+04	1.99E+04	1.84E+04	6.64E+02	1.84E+04	1.52
	µ-CDEOA	1.46E+04	1.70E+04	1.60E+04	5.41E+02	1.60E+04	1.12
F_{15}	BFOA-6	1.23E+04	1.27E+04	1.25E+04	1.08E+02	1.25E+04	0.73
	BFOA-30	1.20E+04	1.24E+04	1.22E+04	9.18E+01	1.22E+04	0.95
	µ-BFOA	9.90E+03	1.15E+04	1.09E+04	2.69E+02	1.09E+04	0.74
	µ-CDEOA	9.39E+03	1.04E+04	9.87E+03	2.28E+02	9.86E+03	0.76
F_{16}	BFOA-6	2.04E+05	2.04E+05	2.04E+05	1.24E+02	2.04E+05	0.77
	BFOA-30	1.97E+05	1.98E+05	1.98E+05	4.98E+01	1.98E+05	0.45
	µ-BFOA	1.87E+05	1.98E+05	1.94E+05	2.25E+03	1.94E+05	0.50
	µ-CDEOA	1.92E+05	2.02E+05	1.98E+05	2.21E+03	1.98E+05	0.48

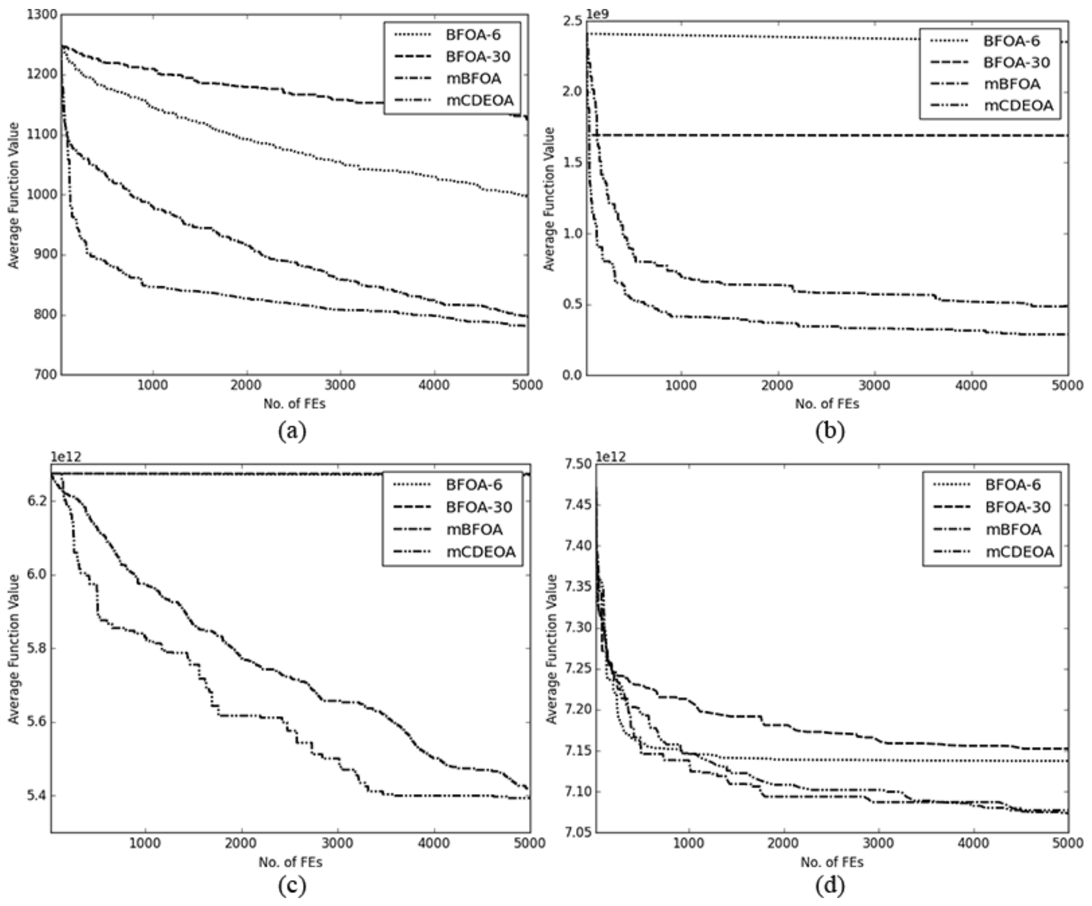


Figure 4.3: (a) F_1: Shifted Sphere; (b) F_2: Schwefel problem 1.2; (c) F_3: Shifted Schwefel problem 1.2; (d) F_4: Shifted Schwefel problem 1.2 with noise in fitness;

4.2.4 Comparison of µCDEOA with three Nature-inspired Techniques

The performance of µCDEOA technique was compared with classical BFOA with 6 and 30 population sizes, BFOA-6, BFOA-30 [50] and µBFOA [14]. The compared algorithm (µBFOA) was chosen due to fact that it possesses very small population size (3) as in its original paper. As for BFOA, we aimed to test its performance with small population size (6) and large population size (30).

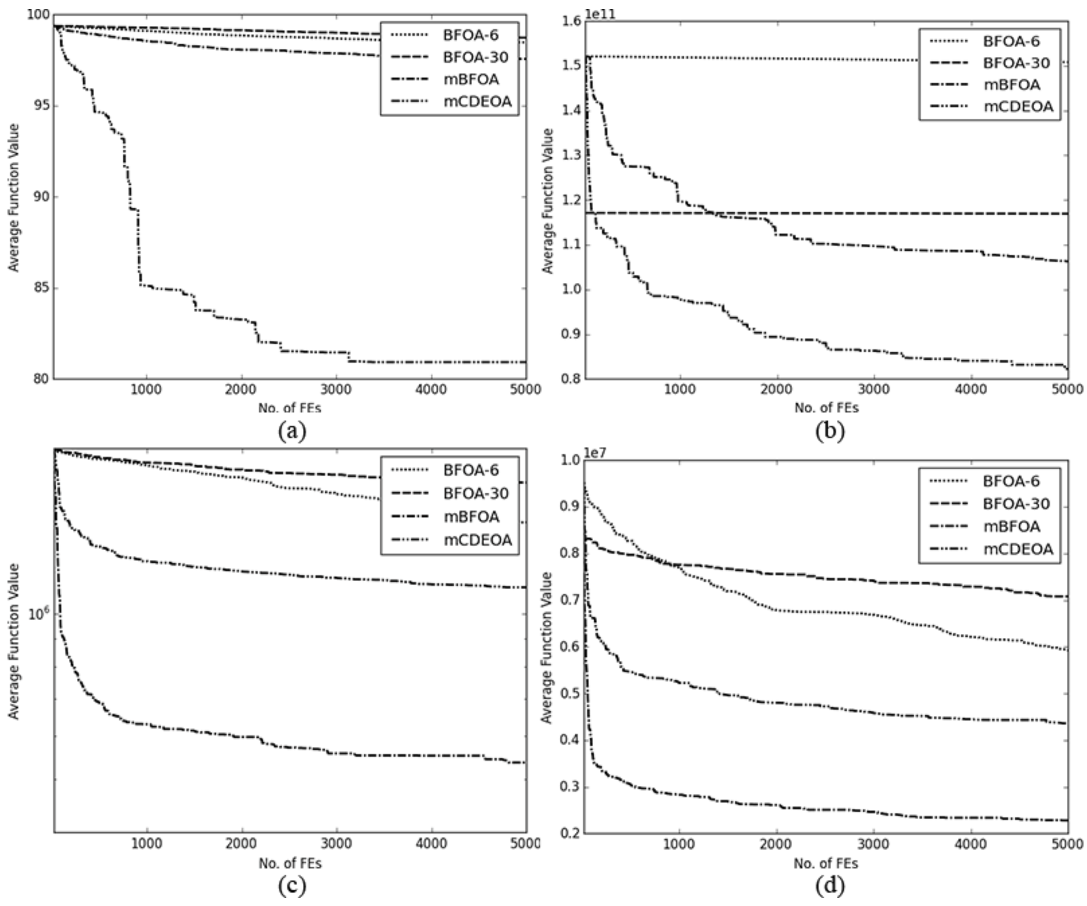


Figure 4.4: (a) F_5: Schwefel problem 2.21; (b) F_6: Shifted and rotated high conditioned elliptic; (c) F_7: Shifted Rosenbrock; (d) F_8: Shifted and rotated Rosenbrock.

Unimodal Functions F_1 - F_6

As reported in Table 4.2, in these six unimodal functions, the micro techniques exhibit their superiority to their classical counterparts in terms of quality of the final solution. µCDEOA shares the first place with µBFOA in F_1 and F_4 functions. All the algorithms show similar performances in F_4 function. We can also observe that the proposed technique performs better than that of other techniques in F_5 and F_6 functions. In contrast, µCDEOA remains behind µBFOA in F_2 and F_3 functions. The classical BFOA fails in most of the problems. We can infer that the success of a technique is problem-dependent. While BFOA-6 outperforms the

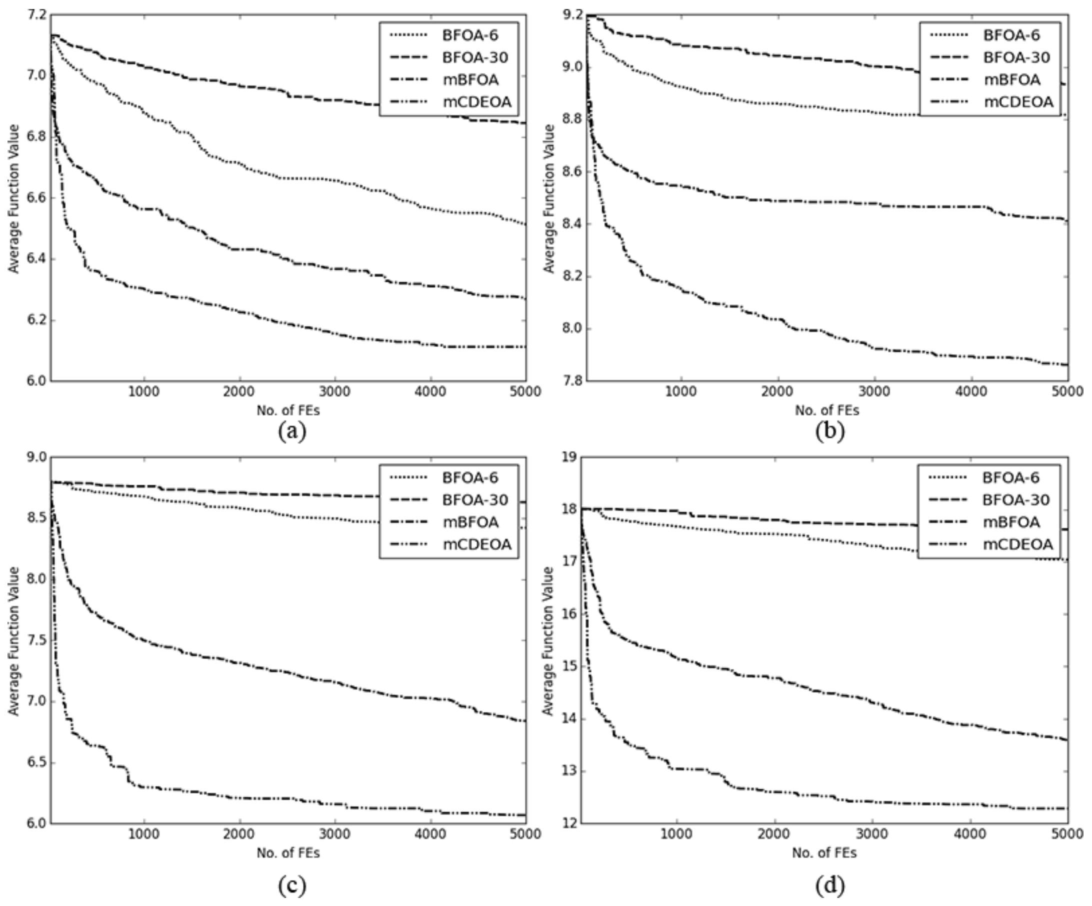
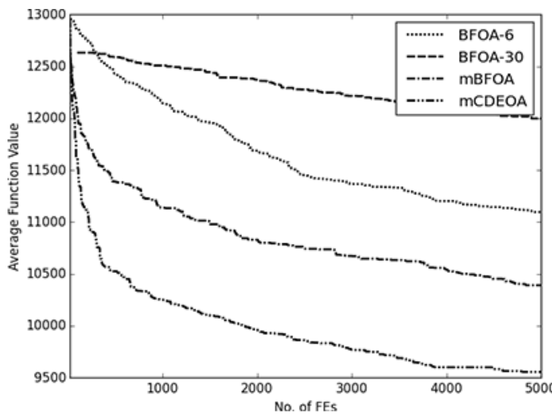


Figure 4.5: (a) F_9: Shifted Ackley. (b) F_{10}: Shifted rotated Ackley; (c) F_{11}: Shifted Griewank; (d) F_{12}: Shifted and Rotated Griewank.

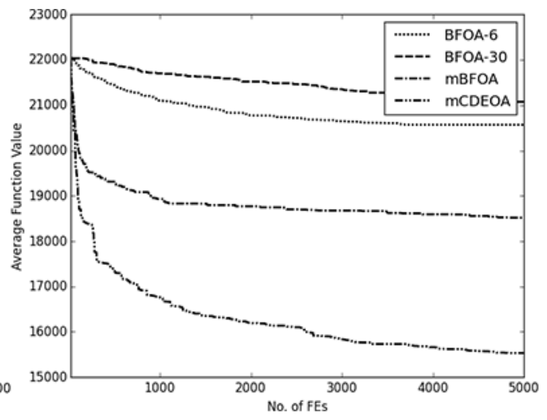
BFOA-30 in only one function F_1, BFOA-30 performs better than BFOA-6 in two functions, F_2 and F_6.

Simple Multi-modal Functions $F_7 - F_{16}$

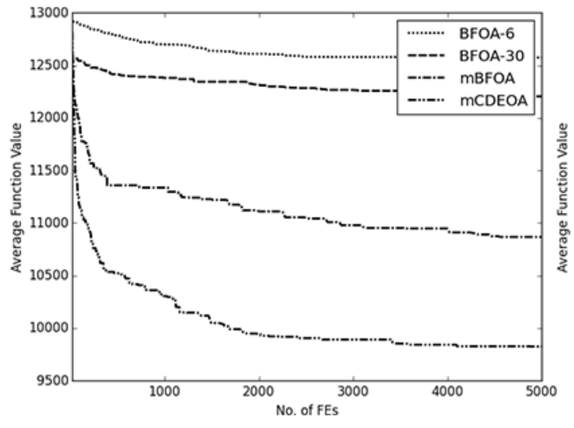
In these ten multimodal functions, overall, the proposed technique performs better than its competitors. µCDEOA exhibits better performance in 4 functions by outperforming its counterparts. On the other hand, µCDEOA exhibits similar performance with µBFOA in 4 functions while µBFOA outperforms the µCDEOA in 2 functions.



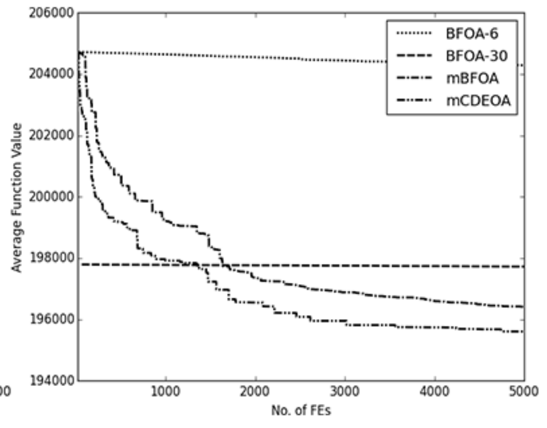
(a)



(b)



(c)



(d)

Figure 4.6: (a) F_{13}: Shifted Rastrigin; (b) F_{14}: Shifted and Rotated Rastrigin; (c) F_{15}: Shifted noncontinuous Rastrigin; (d) F_{16}: Schwefel.

The convergence map of BFOA-6, BFOA-30, µBFOA, and µCDEOA in Fig. 4.3, Fig. 4.4, Fig. 4.6, and Fig. 4.5 implies that the proposed µCDEOA technique has significantly faster and reliable convergence speed than that of its competitors.

In summary, although there are slight differences at the quality of final solution and convergence speed of the algorithms, overall, the proposed technique presents superior performance to the other techniques on unimodal and multimodal functions.

CHAPTER 5
THE EFFECTS OF DE MUTATION STRATEGIES AND ITS PARAMETERS ON CDEOA

Researchers have been investigating the performance of different Differential Evolution (DE) parameters (crossover rate and mutation factor) in solving the optimization problems. It is clear that DE parameters and mutation strategies have a huge impact in the performance of the algorithms. In this chapter, the performance of CDEOA (Chemotaxis Differential Evolution Optimization Algorithm) has been investigated and has been reported that the explorative and exploitative tendency of CDEOA on a fitness landscape depends on DE mutation strategies [68] and its parameters [70]. Bio-inspired techniques such as BFOA and DE have been employed for achieving optimal optimization performance by incorporating evolutionary operators such as mutation, crossover, selection, and reproduction. In order to increase the BFOA and DE performance, a number of approaches have been presented [5] [4] [31] [64] [27]. In our experimental study, the performance of CDEOA on different mutation and crossover rate pairs and different mutation strategies were tested using a set of 6 standard benchmark functions. Functions 1-2 (F_1: Sphere, F_2: Schwefel 2.21) are unimodal and functions 3-6 (F_3: Rosenbrock, F_4: Ackley, F_5: Rastrigin, F_6: Griewank) are multimodal functions.

5.1 The Effects of DE Mutation Strategies on CDEOA

In order to improve the performance of DE, a number of mutation strategies were performed in the literature ([60], [52]). In two studies ([20], [61]), it was reported that DE/rand/2 (Eq. 2.7 in Chapter 2) mutation strategy possesses better performance than DE/rand/1 (Eq. 2.6 in chapter 2) because it diversifies the population with more than 2 trial vectors. On the other hand, the best solution based strategies such as DE/best/1 (Eq. 2.9 in chapter 2) and DE/rand-to-best/1 (Eq. 2.11 in chapter 2) perform faster on simple unimodal optimization problems. However, it may get stuck in local minima and become unreliable in solving multimodal and high dimensional problems. In JADE ([27]), an adaptive mutation strategy was proposed with optional external archive. Although metaheuristics techniques are not problem dependent, Iorio et al. [25] proposed a new mutation strategy called rotation-invariant to solve rotated problems.

In this section, CDEOA's [73] (a hybrid approach of BFOA and DE) performance on five different DE mutation strategies (See section 2.5.1 was presented. The pseudocode of CDEOA is presented in Algorithm 1 (See section 3.1.1).

5.1.1 Experimental Study

After a series of fine tuning experiments, the control parameters F (scaling factor) and Cr (crossover rate) of DE were set 0.5 and 0.9, respectively. For the CDEOA, following parameter values of classical BFOA were chosen: N_c=100, N_s=16, N_{re}=8, $C(i)$=0.1. The study aims to test the quality of final solution and the convergence speed at the end of a fixed number of function evaluations (FEs). All of the algorithms were launched from the same initial population to make the comparison fair. All functions were tested in 2 dimensions with 2×10^4 FEs. Each method were run 30 times with a suite of functions and statistics were given in Table 5.1. "Mean error" indicates that the average of the error function values.

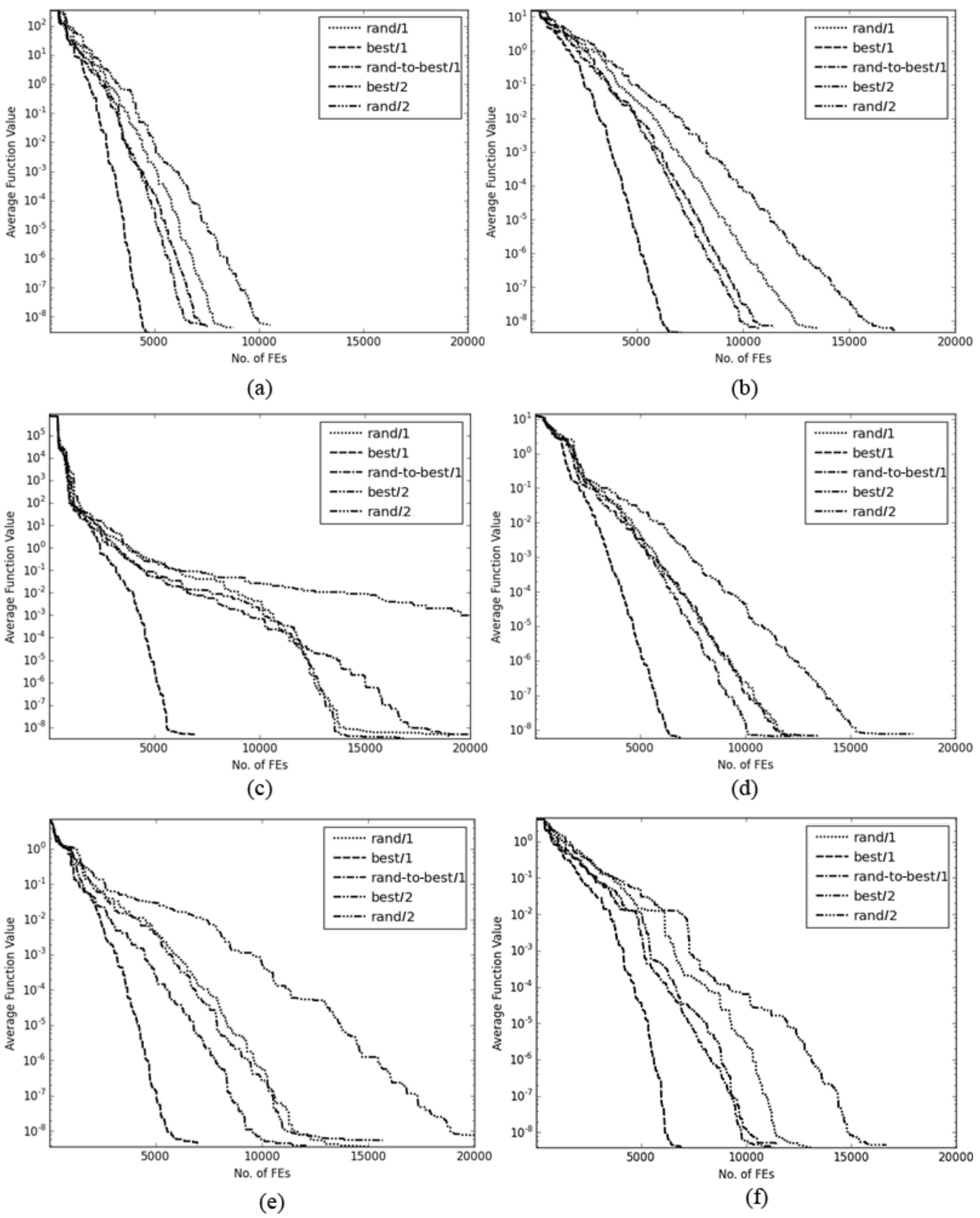


Figure 5.1: (a) F_1: Sphere; (b) F_2: Schwefel problem 2.21; (c) F_3: Rosenbrock; (d) F_4: Ackley;(e) F_5: Rastrigin; (f) F_6: Griewank.

Table 5.1: Comparison of DE mutation strategies rand/1, best/1, rand-to-best/1, best/2, and rand/2.

F	rand/1 Mean Error	best/1 Mean Error	rand-to-best Mean Error	best/2 Mean Error	rand/2 Mean Error
F_1	4.22E-09	4.02E-09	4.96E-09	5.67E-09	5.55E-09
F_2	4.78E-09	4.64E-09	1.85E-08	3.83E-09	4.85E-03
F_3	6.45E-09	6.05E-09	6.56E-09	5.95E-09	6.72E-09
F_4	4.11E-09	4.65E-09	4.10E-09	5.18E-09	1.42E-07
F_5	2.55E-05	2.55E-05	2.55E-05	2.55E-05	8.95E-05
F_6	4.24E-09	4.33E-09	5.09E-09	4.42E-09	4.78E-09

Unimodal Functions

In these two unimodal functions, we can observe the similarities at the quality of final solutions. In particular, DE/best/1 strategy outperforms the other strategies in F_1 and F_2 test functions.

Multimodal Functions

Multimodal functions are known as hard optimization problems since they tend to have many local minima. In these four multimodal functions, rand/2 strategy performs great performance on F_3 and F_5 functions at the quality of final solutions as opposed to the other strategies. We can observe the similar performances on F_4 and F_6 functions. The convergence map of CDEOA led DE mutation strategies which rely on best solution discovered (best/1 (Eq. 2.9) and best/2 Eq. (2.10) in Chapter 2) possess faster convergence speed than the other strategies. From this perspective, we can infer that the best solution strategies exhibit better performance in terms of the quality of final solution and the convergence speed on unimodal and multimodal functions.

5.2 The Effects of DE Parameters on CDEOA

Parameter adaptation of DE has been studied by several researchers to sort the real-world optimization problems out for years. It has been an important improvement in boosting the success of DE optimization techniques. In this context, a number of studies has been put forward by fine-tuning the parameters of DE ([27], [53], [75]). The fine-tuned parameters of

DE has been discussed in section 3.1.1. In this section, CDEOA's (hybrid technique of BFOA) behavior on different DE parameter pairs (mutation and crossover rate) has been reported.

5.2.1 Experimental Study

CDEOA/rand/1 implies the algorithm which employs DE/rand/1 mutation strategy (Eq. 2.6) whereas CDEOA/best/1 implies DE/best/1 (Eq. 2.9 in Chapter 2). For both CDEOA/rand/1 and CDEOA/best/1, the control parameters F (scaling factor) and CR (crossover rate) pair were set [F:0.5, CR:0.9], [F:0.5, CR:0.5], [F:0.1, CR:0.1], [F:0.1, CR:0.9], [F:0.2, CR:0.8]. In DE related studies, we see that the most effective range of F value is to be [0.4, 1.0]. Since a smaller F which is close to 0 has a tendency of helping the individuals have strong exploitative ability, we used $F = 0{,}1$ in two cases of our experiments. CR is generally to be used within the range of [0.1, 0.9] in the literature of DE. In contrast, Ronkkonen et al. [58] reported that CR should be between 0 and 0.2 for separable functions and between 0.9 and 1.0 for multimodal and non-separable functions. From this perspective, we can clearly understand that researchers agreed on F to be between [0.4, 1.0] and CR to be either close to 0 or 1.0. The algorithm-problem pair was launched from the same initial population to make the comparison fair. All functions were tested in 30 dimensions with 3×10^5 FEs. For the CDEOA, following parameter values of classical BFOA were chosen: N_c=100, N_s=16, $N_r e$=8, C(i)=0.1. Each method was run 25 times with a suite of functions and the statistics were given in Table 5.2 and Table 5.3. The convergence graph was plotted in Fig. 5.2 and Fig. 5.3. The horizontal axis of these graphs is the number of function evaluations, and the vertical axis is the mean of function values.

5.2.2 Comparison of five mutation and crossover rate paired techniques based on DE/rand/1 mutation strategy

DE/rand/1 (Eq. 2.6 in Chapter 2) is one of the most used mutation strategy that possesses slow convergence speed and exhibits much stronger exploration ability due to fact that the strategy randomly chooses three individuals which act the distinct search space information out of the current population. That being the case, the aforementioned mutation strategy empowers the exploitation ability. In contrast, it slows down the exploitation ability of an individual ([53]).

Table 5.2: Comparison of CDEOA/rand/1 mutation and crossover rate pairs over 6 benchmark functions.

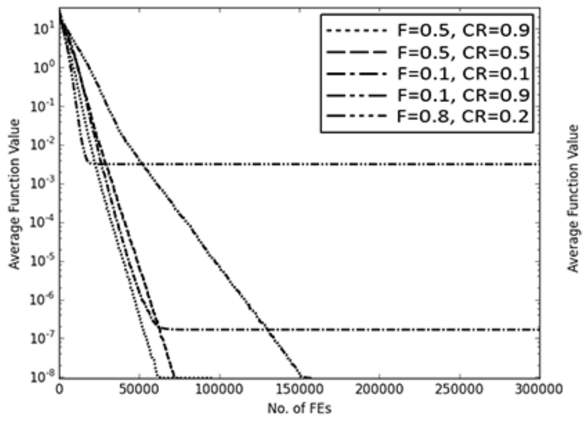
Functions	[F:0.5, CR:0.9] Mean Error	[F:0.5, CR:0.5] Mean Error	[F:0.1,CR:0.1] Mean Error	[F:0.1,CR:0.9] Mean Error	[F:0.8,CR:0.2] Mean Error
F_1	9.27E-09	9.02E-09	1.77E-06	3.31E-03	9.09E-09
F_2	1.59E-02	1.03E-02	4.29E+00	2.53E+01	9.41E+00
F_3	2.57E+01	2.37E+01	2.15E+01	3.68E+01	2.23E+01
F_4	1.66E+00	1.66E+00	1.66E+00	1.67E+00	1.66E+00
F_5	1.69E+01	3.22E+00	1.75E+00	4.00E+01	3.30E+00
F_6	4.93E-04	8.89E-09	4.19E-08	6.54E-03	8.91E-09

Unimodal Functions

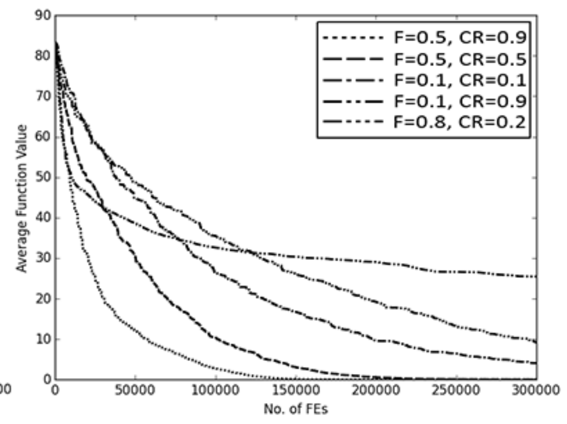
As reported in Table 5.2, in these two unimodal functions, [0.5, 0.9] and [0.5, 0.5] exhibit superior performance to the other [F,CR] pairs. Even though the CRs are different in each pair, they end up with the similar results. Although [0.8, 0.2] has a great success in F_1, it cannot maintain its performance in F_2. [0.1, 0.9] fails in two unimodal problems since F is close to 0.0.

Multimodal Functions

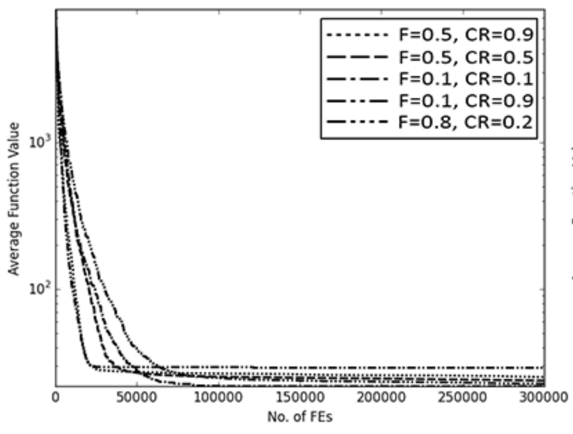
In these four multimodal functions, [0.5, 0.5] and [0.8, 0.2] pairs exhibit similar performance and outperform the others. [0.1, 0.1] is the second best pair although its F value is close to 0.0. Due to characteristics of F_3, we can also observe that [0.1, 0.1] and [0.8, 0.2] are the best in F_3 and show similar performance although they possess distinct F values.



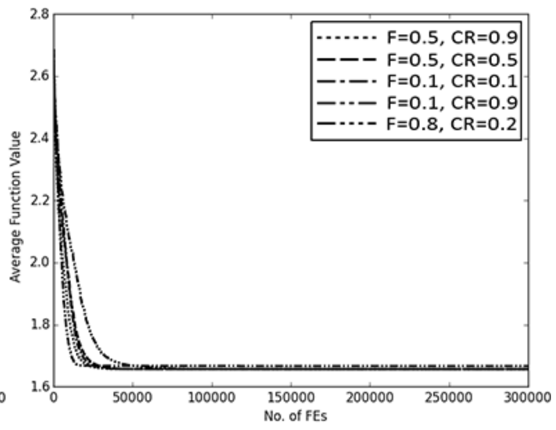
(a)



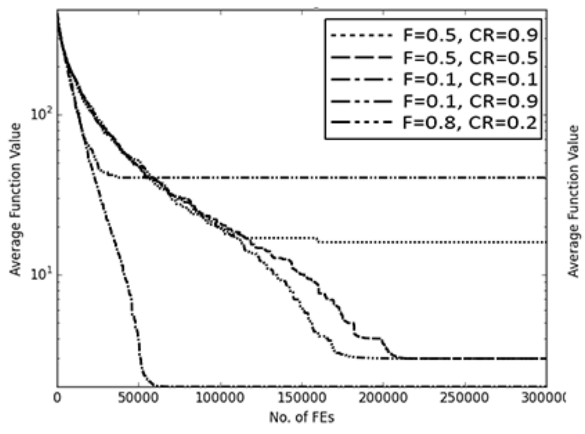
(b)



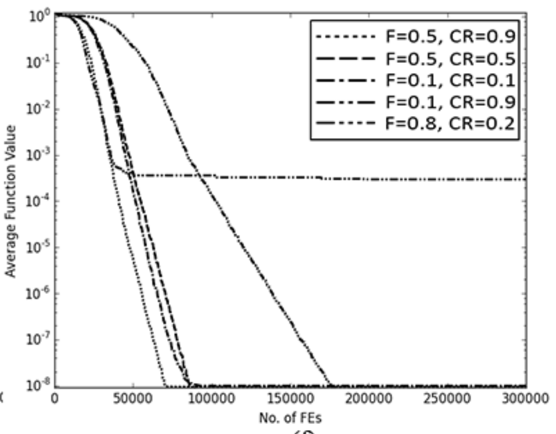
(c)



(d)



(e)



(f)

Figure 5.2: Comparison of five mutation and crossover rate paired techniques based on DE/rand/1 mutation strategy (a) F_1: Sphere; (b) F_2: Schwefel problem 2.21; (c) F_3: Rosenbrock; (d) F_4: Ackley;(e) F_5: Rastrigin; (f) F_6: Griewank.

5.2.3 Comparison of five mutation and crossover rate paired techniques based on DE/best/1 mutation strategy

Strategies based on the best solution such as "DE/best/1", "DE/best/2", and "DE/rand-to-best/1" possess the fast convergence rate and are efficient in unimodal problems. On the other hand, they can tend to get stuck at a local minima, consequently, they converge to the global optimum prematurely ([53]).

Table 5.3: Comparison of CDEOA/best/1 mutation and crossover rate pairs over 6 benchmark functions.

Functions	[F:0.5, CR:0.9]	[F:0.5, CR:0.5]	[F:0.1,CR:0.1]	[F:0.1,CR:0.9]	[F:0.8,CR:0.2]
F_1	7.87E-09	8.48E-09	2.38E-04	2.49E-03	9.31E-09
F_2	5.71E-01	7.46E-01	1.01E+01	2.24E+01	3.54E+00
F_3	7.97E-01	6.25E+00	2.77E+01	2.87E+01	2.15E+01
F_4	1.66E+00	1.66E+00	1.66E+00	1.66E+00	1.66E+00
F_5	4.14E+01	1.32E+01	1.75E+01	1.25E+02	6.17E+00
F_6	5.51E-03	4.92E-03	2.91E-05	7.50E-03	9.12E-09

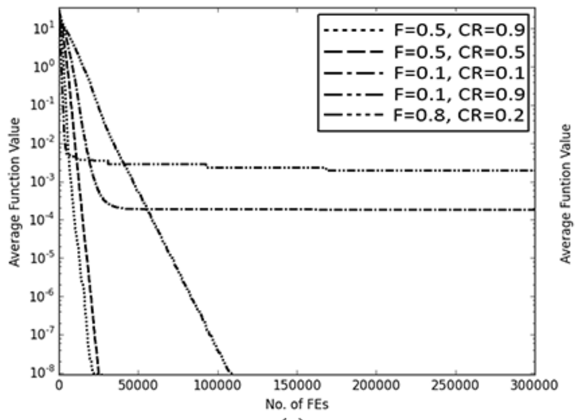
Unimodal Functions

In these two unimodal functions, we can observe similarities as opposed to the CDEOA/rand/1 simulation results in F_1 function. [0.1, 0.9] does not perform due to its small F value. On the other hand, [0.5, 0.9] performs better than that of the others since F=0.5 keeps the individual's exploration and exploitation abilities and CR=0.9 inherits most of the information from mutated vector.

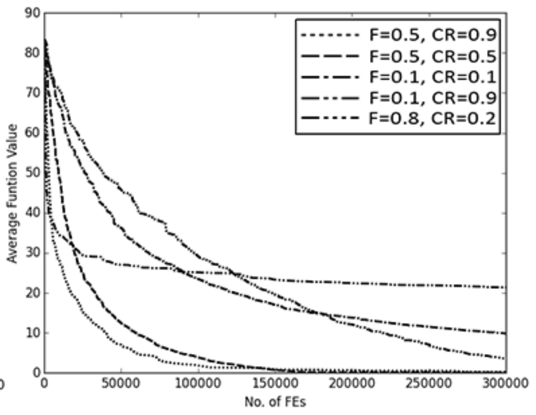
Multimodal Functions

In these four multimodal functions, [0.5, 0.9] exhibits great performance in F_3. The pairs in both CDEOA/rand/1 and CDEOA/best/1 simulation results of F_4 show identical performances. It is obvious that [0.8, 0.2] outperforms the other pairs in F_5 and F_6.

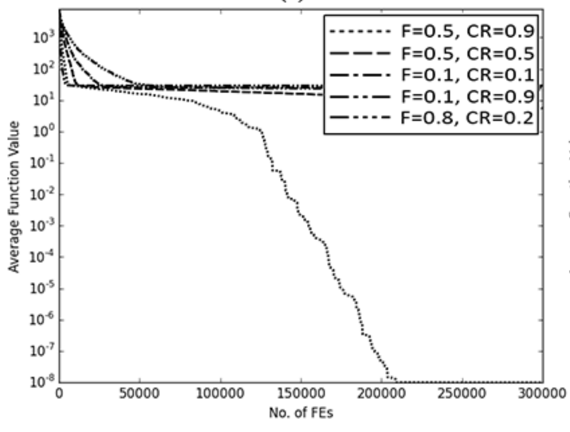
The convergence maps of [0.5, 0.9], [0.5, 0.5], [0.1, 0.1], [0.1, 0.9], and [0.8, 0.2] are reported in Fig. 5.2 and Fig. 5.3. [0.8, 0.2] converges better than the others in F_5 in both CDEOA/rand/1



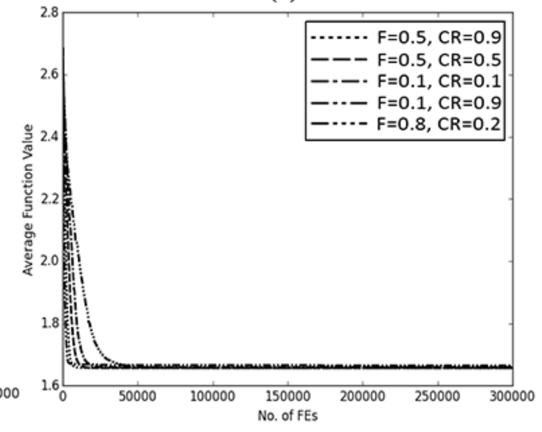
(a)



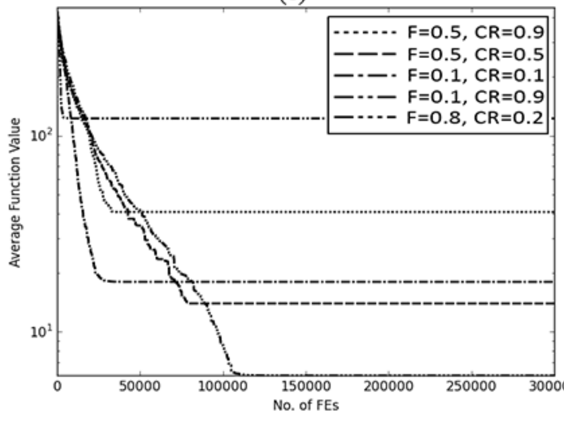
(b)



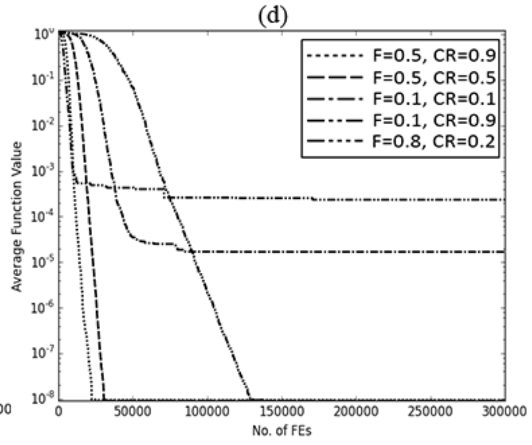
(c)



(d)



(e)



(f)

Figure 5.3: Comparison of five mutation and crossover rate paired techniques based on DE/best/1 mutation strategy (a) F_1: Sphere; (b) F_2: Schwefel problem 2.21; (c) F_3: Rosenbrock; (d) F_4: Ackley;(e) F_5: Rastrigin; (f) F_6: Griewank.

and CDEOA/best/1 simulations. It is clear that each pair in both tests converged to the global optimum in F_4 prematurely. Due to the use of the DE/best/1 (Eq. 4) strategy in CDEOA/best/1, we observe faster convergence than the others. [0.5, 0.9] possesses premature convergence in F_3 CDEOA/rand/1 simulation like the others. However, it exhibits excellent convergence performance in F_3's CDEOA/rand/1 while the rest of the pairs fail.

CHAPTER 6
CONCLUSION AND FUTURE RESEARCH

6.1 General Conclusions

A series of Bacterial Foraging Optimization Algorithm (BFOA) and Differential Evolution (DE) evolutionary based algorithms (CDEOA, iCDEOA, and µCDEOA) were presented in this thesis. The studies hybridizing BFOA with other evolutionary based approaches exhibit that it is an efficient way for improving BFOA performance by combining with different algorithms such as Genetic Algorithm (GA) , Particle Swarm Optimization (PSO), and DE . Unlike similar methods in the literature CDEOA, iCDEOA, and µCDEOA are basically based on two strategies: making weak bacterium more explorative and making strong bacterium more exploitative, where two of which can be integrated into any BFOA variant in order to improve the performance over complex fitness landscapes. The number of the failed steps of a bacterium is accumulated and if it reaches to maximum number of allowed tumble steps, the bacterium undergoes the process of making weak bacteria more explorative. The number of the lucky steps of a bacterium is accumulated and if it reaches to maximum allowed run steps, the bacterium undergoes the process of making strong bacteria more exploitative.

Improved CDEOA (iCDEOA) is to cope with the premature convergence issue of reproduction operator of BFOA. However, it is still based on exploration and exploitation strategies that CDEOA possesses . BFOA performs the reproduction operator by killing half of the population with low objective function values. The rest with high objective function values split into two. Consequently, the population tends to converge prematurely as the algorithm lacks of explorative population. iCDEOA disperses the population with poor function values whereas it sends the strong population with high function value to the vicinity of the best bacterium. Probabilistic re-positioning operator has been utilized to balance the exploration and exploitation of the search space.

Micro CDEOA (µCDEOA) was successfully used to optimize high-dimensional optimization problems. In this approach, in order to increase the convergence performance and quality of the final solution, after re-initializing the population, the bacteria are ranked according to their objective function values. The best bacterium's position is preserved in the population. The second best bacterium is reinitialized in the neighborhood of the best bacterium based on the ideas of DE technique, whereas the rest of the bacteria (four bacteria) are dispersed at random on the search space.

We have also observed the impact of DE/rand/1 and DE/best1 mutation strategies on CDEOA technique in unimodal and multimodal functions. Generally speaking, there is no common parameter settings for all the problems. Rather, there are optimum parameter values for each problem after fine-tuning experiments.

The experimental studies of CDEOA were performed on 30 single objective numerical optimization problems used in CEC2014 special session and competition. CDEOA was compared with classical BFOA, DE, two BFOA, and four DE counterparts. ICDEOA was compared with three state-of-the-art DE counterparts. µCDEOA was compared with classical BFOA with 6 and 30 population sizes, and micro BFOA (µBFOA) in a set of 16 single objective numerical optimization problems taken from IEEE CEC. Simulation results show that overall performance of CDEOA, iCDEOA, and µCDEOA was superior to, or comparable to, that of the other competitors.

6.2 Future Research

CDEOA is mostly based on BFOA which consumes quite a lot computational time. Contrary to this, DE is good at employing the local and global search operators efficiently. It is important to employ DE based BFOA in order to reduce the computational time.

ICDEOA is good at optimizing the unimodal and multimodal functions. However, it does not exhibit the same performance in some complex hybrid and composition functions. It is beneficial to employ DE based operators to increase the performance as DE coupled methods have been a powerful technique for complex hybrid and composition functions.

µCDEOA has been tested on high dimensional problems and proven to be a successful technique as a global optimizer. However, it has not been tested on large scale optimization problems. Therefore, µCDEOA needs to be improved in a way that can solve large scale optimization problems.

6.3 Source Codes

The Python source codes of the algorithms classical BFOA, classical DE, CDE, ACBSFO_DES, proposed CDEOA, iCDEOA, and µCDEOA can be downloaded from Y. Emre Yildiz's homepage (`https://sites.google.com/site/yeyildiz12/`). These codes are written to be compatible with the opn global optimization framework available in Oğuz Altun's Bitbucket repository (`https://bitbucket.org/oaltun/opn`). Algorithm 1 and Algorithm 2 in Chapter 3 does not correspond one to one to CDEOA code given, as we wanted to hide unnecessary details of the framework used. The lines 19-26 in Algorithm 1 and Algorithm 2 of Chapter 3 summarize what opn does to make the given code runnable.

Appendix A

THE NUMERICAL COMPARISON OF CDEOA AND ICDEOA WITH THEIR COUNTERPARTS

Table A.1: Comparison of BFOA, DE, CDE, ACBSFO_DES, and CDEOA over 30 test functions of 30 dimensions using 300,000 function evaluations. ”Mean Error” and ”Std Dev” indicate the average and standard deviation of the error function values obtained in 25 runs, respectively.

Functions		BFOA		DE		CDE		ACBSFO_DES		CDEOA
		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev
Unimodal Functions	F_1	1.36E+09±3.48E+08	−	4.95E+06±2.30E+06	−	2.65E+07±7.00E+06	−	6.35E+04±4.12E+04	+	1.76E+06±1.58E+06
	F_2	9.94E+10±8.78E+09	−	5.49E+01±3.46E+01	−	7.39E+06±4.37E+06	−	0.00E+00±0.00E+00	+	1.82E+00±2.64E+00
	F_3	1.40E+05±1.76E+04	−	1.13E+05±1.34E+04	−	1.60E+00±5.37E−01	−	0.00E+00±0.00E+00	≈	0.00E+00±0.00E+00
Simple Multimodal Functions	F_4	1.95E+04±4.83E+03	−	9.27E+03±1.36E+03	−	1.60E+02±1.77E+01	−	1.33E+00±6.49E+00	+	8.19E+01±2.79E+01
	F_5	2.00E+01±1.46E−03	≈	2.10E+01±4.78E−02	≈	2.00E+01±5.32E−04	≈	2.00E+01±8.32E−06	≈	2.00E+01±4.63E−05
	F_6	4.76E+01±2.08E+00	−	3.92E+01±1.10E+00	−	1.63E+01±6.23E+00	−	1.65E+01±2.89E+00	−	3.96E−01±5.90E−01
	F_7	8.86E+02±9.76E+01	−	5.33E+02±6.61E+01	−	1.06E+00±6.95E−02	−	8.86E−03±1.01E−02	−	8.87E−04±3.18E−03
	F_8	2.09E+02±3.34E+01	−	3.53E+02±1.40E+01	−	1.44E+02±1.24E+01	−	8.69E+01±2.70E+01	−	5.37E+01±7.61E+00
	F_9	3.10E+02±4.48E+01	−	4.04E+02±2.14E+01	−	1.34E+02±1.22E+01	−	9.31E+01±2.51E+01	−	6.33E+01±1.23E+01
	F_{10}	3.62E+03±5.83E+02	−	6.93E+03±2.96E+02	−	4.06E+03±2.42E+02	−	1.74E+03±4.08E+02	≈	1.77E+03±2.51E+02
	F_{11}	4.08E+03±6.49E+02	−	7.19E+03±2.77E+02	−	4.17E+03±2.54E+02	−	2.95E+03±3.86E+02	+	3.23E+03±4.88E+02
	F_{12}	2.67E+00±4.12E−01	−	2.47E+00±2.22E−01	−	1.06E+00±2.19E−01	−	2.23E−01±1.23E−01	+	4.67E−01±1.91E−01
	F_{13}	9.23E+00±9.42E−01	−	6.25E+00±3.51E−01	−	2.60E−01±3.88E−02	≈	3.19E−01±8.62E−02	−	2.91E−01±4.20E−02
	F_{14}	2.88E+02±5.43E+01	−	1.89E+02±1.80E+01	−	2.09E−01±2.84E−02	≈	2.25E−01±5.95E−02	−	2.07E−01±2.87E−02
	F_{15}	3.36E+06±1.60E+06	−	1.01E+06±4.21E+05	−	6.51E+00±1.20E+00	−	1.12E+01±3.65E+00	−	8.88E+00±1.54E+00
	F_{16}	1.29E+01±3.51E−01	−	1.31E+01±1.66E−01	−	1.16E+01±3.91E−01	≈	1.04E+01±7.11E−01	+	1.12E+01±4.67E−01
Hybrid Functions	F_{17}	3.91E+07±2.98E+07	−	1.85E+07±6.27E+06	−	2.44E+03±2.12E+02	−	1.06E+03±4.82E+02	+	1.18E+03±2.80E+02
	F_{18}	2.54E+09±1.47E+09	−	1.01E+09±2.98E+08	−	9.92E+01±8.13E+00	−	9.40E+01±2.95E+01	+	5.77E+01±7.58E+00
	F_{19}	4.40E+02±1.56E+02	−	2.46E+02±3.40E+01	−	7.62E+00±1.30E+00	−	1.22E+01±1.64E+01	−	4.59E+00±4.21E−01
	F_{20}	1.79E+05±1.17E+05	−	7.22E+04±2.89E+04	−	5.63E+01±6.70E+00	−	5.89E+01±3.20E+01	−	1.96E+01±7.21E+00
	F_{21}	9.16E+06±9.42E+06	−	5.62E+06±2.03E+06	−	1.31E+03±1.35E+02	−	4.82E+02±2.17E+02	−	3.51E+02±1.74E+02
	F_{22}	1.49E+03±6.07E+02	−	1.28E+03±2.29E+02	−	2.23E+02±7.16E+01	≈	3.08E+02±1.78E+02	−	2.46E+02±8.44E+01
Composition Functions	F_{23}	1.01E+03±1.68E+02	−	3.26E+02±5.29E+01	≈	3.16E+02±1.98E−01	≈	3.15E+02±8.75E−12	≈	3.15E+02±5.55E−07
	F_{24}	4.47E+02±2.71E+01	−	2.65E+02±1.33E+01	≈	2.31E+02±2.36E+00	≈	2.43E+02±7.32E+00	≈	2.24E+02±9.54E−01
	F_{25}	2.72E+02±1.96E+01	≈	2.31E+02±6.53E+00	≈	2.13E+02±1.40E+00	≈	2.09E+02±5.22E+00	≈	2.07E+02±7.67E−01
	F_{26}	1.43E+02±5.61E+01	≈	1.06E+02±5.50E−01	≈	1.00E+02±3.92E−02	≈	1.20E+02±3.99E+01	−	1.00E+02±5.80E−02
	F_{27}	1.60E+03±2.84E+02	−	8.30E+02±7.82E+01	−	4.26E+02±3.71E+01	−	6.35E+02±1.91E+02	−	3.63E+02±4.99E+01
	F_{28}	4.66E+03±5.28E+02	−	4.51E+03±3.33E+02	−	1.33E+03±7.36E+01	−	1.42E+03±5.46E+02	−	1.06E+03±3.07E+01
	F_{29}	3.40E+08±1.51E+08	−	1.13E+08±3.12E+07	−	8.89E+02±8.53E+01	−	7.22E+02±1.71E+02	−	4.61E+02±1.75E+02
	F_{30}	1.81E+06±9.33E+05	−	8.97E+05±2.16E+05	−	2.55E+03±4.06E+02	−	1.76E+03±7.46E+02	−	9.62E+02±1.98E+02
−		27		24		21		18		
+		0		0		0		6		
≈		3		5		9		6		

”−”, ”+”, and ”≈” denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CDEOA, respectively

Table A.2: Comparison of BFOA, DE, CDE, ACBSFO_DES, and CDEOA over 30 test functions of 30 dimensions using 300,000 function evaluations. “BFV”, “WFV”, and “MEDIAN” indicate the best final function value, the worst final function value, and the median of the final value obtained in 25 runs, respectively.

	Function	Algorithm	BFV	WFV	MEDIAN		Function	Algorithm	BFV	WFV	MEDIAN
F_1	*Rotated High Conditioned Elliptic*	BFOA	7.92E+08	2.03E+09	1.29E+09	F_9	*Shifted and Rotated Rastrigin*	BFOA	2.34E+02	4.20E+02	3.11E+02
		DE	2.17E+06	1.17E+07	4.35E+06			DE	3.57E+02	4.51E+02	4.03E+02
		CDE	1.34E+07	4.39E+07	2.52E+07			CDE	1.11E+02	1.57E+02	1.33E+02
		ACBSFO_DES	1.63E+04	1.49E+05	5.16E+04			ACBSFO_DES	5.37E+01	1.76E+02	9.15E+01
		CDEOA	5.13E+05	8.01E+06	1.05E+06			CDEOA	3.63E+01	8.08E+01	6.42E+01
F_2	*Rotated Bent Cigar*	BFOA	8.39E+10	1.15E+11	1.00E+11	F_{10}	*Shifted Schwefel*	BFOA	2.11E+03	4.74E+03	3.74E+03
		DE	1.86E+00	1.21E+02	5.29E+01			DE	6.10E+03	7.39E+03	6.99E+03
		CDE	1.29E+06	1.72E+07	6.47E+06			CDE	3.61E+03	4.53E+03	4.03E+03
		ACBSFO_DES	0.00E+00	0.00E+00	0.00E+00			ACBSFO_DES	1.07E+03	2.53E+03	1.65E+03
		CDEOA	6.82E−02	1.25E+01	1.30E+00			CDEOA	1.27E+03	2.28E+03	1.77E+03
F_3	*Rotated Discus*	BFOA	9.71E+04	1.69E+05	1.41E+05	F_{11}	*Shifted and Rotated Schwefel*	BFOA	1.94E+03	5.26E+03	4.10E+03
		DE	8.66E+04	1.36E+05	1.16E+05			DE	6.56E+03	7.57E+03	7.26E+03
		CDE	7.84E−01	2.64E+00	1.51E+00			CDE	3.52E+03	4.62E+03	4.14E+03
		ACBSFO_DES	0.00E+00	0.00E+00	0.00E+00			ACBSFO_DES	2.06E+03	3.64E+03	3.04E+03
		CDEOA	0.00E+00	1.08E−07	0.00E+00			CDEOA	2.19E+03	3.98E+03	3.37E+03
F_4	*Shifted and Rotated Rosenbrock*	BFOA	1.02E+04	3.16E+04	1.94E+04	F_{12}	*Shifted and Rotated Katsuura*	BFOA	1.69E+00	3.49E+00	2.74E+00
		DE	5.76E+03	1.23E+04	9.35E+03			DE	1.85E+00	2.87E+00	2.50E+00
		CDE	1.19E+02	1.93E+02	1.57E+02			CDE	6.00E−01	1.37E+00	1.10E+00
		ACBSFO_DES	1.00E−09	3.31E+01	1.40E−05			ACBSFO_DES	4.23E−02	4.79E−01	1.88E−01
		CDEOA	3.82E+00	1.24E+02	7.60E+01			CDEOA	1.19E−01	8.08E−01	4.83E−01
F_5	*Shifted and Rotated Ackley*	BFOA	2.00E+01	2.00E+01	2.00E+01	F_{13}	*Shifted and Rotated HappyCat*	BFOA	6.87E+00	1.07E+01	9.42E+00
		DE	2.08E+01	2.10E+01	2.10E+01			DE	5.70E+00	7.12E+00	6.22E+00
		CDE	2.00E+01	2.00E+01	2.00E+01			CDE	1.84E−01	3.20E−01	2.64E−01
		ACBSFO_DES	2.00E+01	2.00E+01	2.00E+01			ACBSFO_DES	1.79E−01	5.15E−01	3.30E−01
		CDEOA	2.00E+01	2.00E+01	2.00E+01			CDEOA	2.15E−01	3.77E−01	2.90E−01
F_6	*Shifted and Rotated Weierstrass*	BFOA	4.11E+01	5.10E+01	4.77E+01	F_{14}	*Shifted and Rotated HGBat*	BFOA	1.17E+02	3.71E+02	2.97E+02
		DE	3.60E+01	4.06E+01	3.96E+01			DE	1.43E+02	2.15E+02	1.92E+02
		CDE	5.03E+00	3.10E+01	1.59E+01			CDE	1.31E−01	2.58E−01	2.06E−01
		ACBSFO_DES	1.07E+01	2.16E+01	1.71E+01			ACBSFO_DES	1.13E−01	3.36E−01	2.36E−01
		CDEOA	7.07E−02	2.78E+00	1.70E−01			CDEOA	1.53E−01	2.66E−01	2.08E−01
F_7	*Shifted and Rotated Griewank*	BFOA	7.10E+02	1.06E+03	8.97E+02	F_{15}	*SREGR*	BFOA	8.50E+05	6.99E+06	3.43E+06
		DE	3.64E+02	6.22E+02	5.41E+02			DE	4.55E+04	1.70E+06	9.15E+05
		CDE	7.97E−01	1.18E+00	1.06E+00			CDE	4.19E+00	8.70E+00	6.32E+00
		ACBSFO_DES	0.00E+00	3.93E−02	7.40E−03			ACBSFO_DES	5.37E+00	1.84E+01	1.07E+01
		CDEOA	0.00E+00	1.48E−02	3.23E−07			CDEOA	4.65E+00	1.12E+01	8.84E+00
F_8	*Shifted Rastrigin*	BFOA	1.23E+02	2.51E+02	2.19E+02	F_{16}	*SRESF6*	BFOA	1.19E+01	1.35E+01	1.30E+01
		DE	3.26E+02	3.81E+02	3.53E+02			DE	1.28E+01	1.34E+01	1.31E+01
		CDE	1.12E+02	1.69E+02	1.49E+02			CDE	1.08E+01	1.22E+01	1.17E+01
		ACBSFO_DES	2.98E+01	1.42E+02	8.36E+01			ACBSFO_DES	8.80E+00	1.14E+01	1.05E+01
		CDEOA	3.61E+01	7.02E+01	5.46E+01			CDEOA	1.02E+01	1.20E+01	1.12E+01

SREGR: Shifted and Rotated Expanded Griewank’s plus Rosenbrock, SRESF6: Shifted and Rotated Expanded Scaffer’s F6 Function

Table A.3: Comparison of FCDE, POBL-ADE, FWA-DE, FERDE, and CDEOA over 30 test functions of 30 dimensions using 300,000 function evaluations. "Mean Error" and "Std Dev" indicate the average and standard deviation of the error function values obtained in 51 runs, respectively.

Functions		FCDE		POBL-ADE		FWA-DE		FERDE		CDEOA
		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev
Unimodal Functions	F_1	6.54E+04±4.90E+04	−	1.60E+04±1.22E+04	+	2.76E+05±1.82E+05	−	5.41E+02±5.41E+02	+	4.64E+04±4.26E+04
	F_2	0.00E+00±0.00E+00	≈	3.14E+02±7.52E+02	−	0.00E+00±0.00E+00	≈	2.39E−03±2.39E−03	−	0.00E+00±0.00E+00
	F_3	3.51E+01±1.26E+02	−	0.00E+00±0.00+00	≈	0.00E+00±0.00+00	+	1.13E−03±1.13E−03	−	1.05E−05±6.68E−05
Simple Multimodal Functions	F_4	1.62E+01±3.23E+01	−	6.34E+01±2.63E+01	−	2.04E+01±1.91E+01	−	6.25E−01±6.25E−01	+	4.46E+00±2.40E+01
	F_5	2.09E+01±7.23E−02	−	2.06E+01±5.11E−02	−	2.05E+01±5.36E−02	−	2.00E+01±2.00E+01	≈	2.00E+01±5.42E−06
	F_6	2.20E+01±3.36E+00	−	5.19E+00±1.64E+00	+	1.29E+01±8.25E+00	−	1.82E+01±1.82E+01	−	6.30E+00±2.33E+00
	F_7	2.89E−02±5.41E−02	−	2.37E−02±2.31E−02	−	8.55E−03±9.81E−03	+	0.00E+00±0.00E+00	+	5.99E−03±8.69E−03
	F_8	9.44E+01±2.46E+01	−	5.59E+01±1.10E+01	−	0.00E+00±0.00E+00	+	0.00E+00±0.00E+00	+	5.01E+01±1.32E+01
	F_9	1.33E+02±2.89E+01	−	8.46E+01±9.06E+00	−	5.66E+01±1.08E+01	+	5.50E+01±5.50E+01	+	7.92E+01±1.71E+01
	F_{10}	2.24E+03±5.02E+02	−	2.17E+03±4.92E+02	−	8.53E+00±2.42E+00	+	1.29E+00±1.29E+00	+	1.71E+03±4.13E+02
	F_{11}	3.40E+03±5.99E+02	≈	3.86E+03±3.52E+02	−	2.63E+03±2.48E+02	+	2.72E+03±2.72E+03	≈	3.09E+03±5.28E+02
	F_{12}	1.56E+00±5.83E−01	−	9.51E−01±1.35E−01	−	3.71E−01±6.66E−02	−	5.64E−01±5.64E−01	−	1.00E−01±7.04E−02
	F_{13}	5.98E−01±1.16E−01	−	2.86E−01±6.10E−02	≈	3.89E−01±5.51E−02	≈	2.84E−01±2.84E−01	≈	3.29E−01±7.19E−02
	F_{14}	4.60E−01±2.82E−01	−	2.26E−01±4.28E−02	≈	2.69E−01±7.76E−02	≈	2.14E−01±2.14E−01	≈	2.52E−01±1.02E−01
	F_{15}	1.59E+01±7.64E+00	−	7.73E+00±1.04E+00	≈	7.37E+00±8.46E−01	≈	4.14E+00±4.14E+00	+	7.77E+00±2.82E+00
	F_{16}	1.21E+01±5.74E−01	−	1.04E+01±4.58E−01	≈	1.10E+01±2.71E−01	≈	1.12E+01±1.12E+01	≈	1.07E+01±8.02E−01
Hybrid Functions	F_{17}	4.00E+03±3.17E+03	−	1.10E+03±4.14E+02	≈	6.29E+03±5.95E+03	−	1.16E+03±1.16E+03	≈	1.49E+03±1.19E+03
	F_{18}	1.21E+02±6.62E+01	−	1.10E+02±3.81E+01	−	7.67E+01±3.66E+01	−	2.24E+01±2.24E+01	≈	2.90E+01±1.28E+01
	F_{19}	1.33E+01±1.16E+01	−	8.88E+00±1.21E+01	−	9.95E+00±1.93E+00	−	7.74E+00±7.74E+00	−	6.46E+00±1.16E+01
	F_{20}	1.45E+02±9.98E+01	−	3.89E+01±2.21E+01	−	4.28E+01±2.61E+01	−	2.80E+01±2.80E+01	−	1.70E+01±7.30E+00
	F_{21}	1.88E+03±2.37E+03	−	3.86E+02±1.91E+02	≈	7.29E+02±9.49E+02	−	5.99E+02±5.99E+02	−	3.84E+02±1.92E+02
	F_{22}	5.44E+02±2.35E+02	−	2.31E+02±8.16E+01	≈	1.46E+02±8.83E+01	+	1.15E+02±1.15E+02	+	2.50E+02±1.08E+02
Composition Functions	F_{23}	3.15E+02±0.00E+00	≈	3.15E+02±1.16E−07	≈	3.14E+02±0.00E+00	≈	3.14E+02±3.14E+02	≈	3.15E+02±0.00E+00
	F_{24}	2.50E+02±6.82E+00	≈	2.22E+02±7.48E+00	≈	2.26E+02±3.59E+00	≈	2.25E+02±2.25E+02	≈	2.37E+02±8.04E+00
	F_{25}	2.05E+02±2.29E+00	≈	2.04E+02±3.22E+00	≈	2.01E+02±1.98E−01	≈	2.00E+02±2.00E+02	≈	2.04E+02±1.24E+00
	F_{26}	1.01E+02±1.10E−01	≈	1.39E+02±4.91E+01	−	1.00E+02±5.35E−02	≈	1.00E+02±1.00E+02	≈	1.00E+02±8.75E−02
	F_{27}	6.18E+02±2.32E+02	−	4.21E+02±4.64E+01	≈	4.01E+02±3.06E+01	≈	3.84E+02±3.84E+02	≈	4.26E+02±6.01E+01
	F_{28}	1.50E+03±3.75E+02	−	9.16E+02±1.63E+02	+	3.93E+02±1.46E+01	+	8.05E+02±8.05E+02	+	1.08E+03±3.38E+02
	F_{29}	1.06E+06±3.28E+06	−	3.39E+05±2.41E+06	−	2.11E+02±2.90E+00	+	1.19E+03±1.19E+03	−	7.69E+02±2.09E+02
	F_{30}	2.53E+03±9.82E+02	−	1.29E+03±5.14E+02	≈	4.51E+02±1.96E+02	+	1.07E+03±1.07E+03	≈	1.06E+03±6.48E+02
−		24		14		10		8		
+		0		5		9		9		
≈		6		11		11		13		

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of CDEOA, respectively

Table A.4: Comparison of FCDE, POBL−ADE, FWA−DE, FERDE, and CDEOA over 30 test functions of 30 dimensions using 300,000 function evaluations. “BFV”, “WFV”, and “MEDIAN” indicate the best final function value, the worst final function value, and the median of the final value obtained in 51 runs, respectively.

	Function	Algorithm	BFV	WFV	Median		Function	Algorithm	BFV	WFV	MEDIAN
F_1	*Rotated High Conditioned Elliptic*	FCDE	5.36E+03	2.65E+05	5.46E+04	F_9	*Shifted and Rotated Rastrigin*	FCDE	7.26E+01	2.22E+02	1.31E+02
		POBL−ADE	1.50E+03	6.38E+04	1.26E+04			POBL−ADE	6.35E+01	1.06E+02	8.43E+01
		FWA−DE	3.70E+04	9.94E+05	2.24E+05			FWA−DE	3.32E+01	7.82E+01	5.62E+01
		FERDE	2.82E−02	2.48E+03	2.69E+02			FERDE	3.48E+01	8.16E+01	5.57E+01
		CDEOA	8.58E+03	2.87E+05	3.65E+04			CDEOA	4.88E+01	1.19E+02	7.76E+01
F_2	*Rotated Bent Cigar*	FCDE	0.00E+00	0.00E+00	0.00E+00	F_{10}	*Shifted Schwefel*	FCDE	5.87E+02	3.51E+03	2.33E+03
		POBL−ADE	0.00E+00	3.15E+03	6.91E−01			POBL−ADE	8.94E+02	3.35E+03	2.11E+03
		FWA−DE	0.00E+00	0.00E+00	0.00E+00			FWA−DE	4.65E+00	1.54E+01	8.09E+00
		FERDE	1.09E−06	1.33E−02	1.23E−03			FERDE	8.33E−02	7.50E+00	3.33E−01
		CDEOA	0.00E+00	0.00E+00	0.00E+00			CDEOA	8.69E+02	2.58E+03	1.67E+03
F_3	*Rotated Discus*	FCDE	1.70E−05	8.56E+02	4.03E−01	F_{11}	*Shifted and Rotated Schwefel*	FCDE	2.30E+03	4.63E+03	3.33E+03
		POBL−ADE	0.00E+00	0.00E+00	0.00E+00			POBL−ADE	2.92E+03	4.56E+03	3.91E+03
		FWA−DE	0.00E+00	0.00E+00	0.00E+00			FWA−DE	2.00E+03	3.03E+03	2.65E+03
		FERDE	1.04E−05	3.43E−03	1.02E−03			FERDE	1.41E+03	3.74E+03	2.75E+03
		CDEOA	0.00E+00	4.81E−04	0.00E+00			CDEOA	1.69E+03	4.30E+03	3.15E+03
F_4	*Shifted and Rotated Rosenbrock*	FCDE	1.60E−06	1.62E+02	8.82E−01	F_{12}	*Shifted and Rotated Katsuura*	FCDE	4.36E−01	2.73E+00	1.49E+00
		POBL−ADE	3.16E−03	9.95E+01	7.08E+01			POBL−ADE	6.11E−01	1.25E+00	9.58E−01
		FWA−DE	1.22E−03	7.44E+01	1.58E+01			FWA−DE	2.08E−01	5.21E−01	3.70E−01
		FERDE	0.00E+00	3.99E+00	0.00E+00			FERDE	2.14E−01	9.38E−01	5.54E−01
		CDEOA	7.98E−07	1.62E+02	4.15E−04			CDEOA	3.05E−02	3.82E−01	8.29E−02
F_5	*Shifted and Rotated Ackley*	FCDE	2.07E+01	2.10E+01	2.09E+01	F_{13}	*Shifted and Rotated HappyCat*	FCDE	3.52E−01	9.61E−01	5.83E−01
		POBL−ADE	2.05E+01	2.07E+01	2.06E+01			POBL−ADE	1.56E−01	4.16E−01	2.79E−01
		FWA−DE	2.04E+01	2.06E+01	2.05E+01			FWA−DE	2.88E−01	4.90E−01	4.00E−01
		FERDE	2.00E+01	2.00E+01	2.00E+01			FERDE	1.91E−01	3.81E−01	2.85E−01
		CDEOA	2.00E+01	2.00E+01	2.00E+01			CDEOA	1.80E−01	5.36E−01	3.11E−01
F_6	*Shifted and Rotated Weierstrass*	FCDE	1.51E+01	2.89E+01	2.20E+01	F_{14}	*Shifted and Rotated HGBat*	FCDE	1.88E−01	1.31E+00	3.38E−01
		POBL−ADE	1.88E+00	8.50E+00	5.01E+00			POBL−ADE	1.06E−01	3.08E−01	2.28E−01
		FWA−DE	8.12E−02	2.09E+01	1.75E+01			FWA−DE	1.78E−01	7.40E−01	2.59E−01
		FERDE	1.42E+01	2.21E+01	1.82E+01			FERDE	1.07E−01	2.69E−01	2.15E−01
		CDEOA	9.89E−01	1.45E+01	6.38E+00			CDEOA	1.35E−01	7.81E−01	2.33E−01
F_7	*Shifted and Rotated Griewank*	FCDE	0.00E+00	3.65E−01	1.72E−02	F_{15}	*SREGR*	FCDE	3.69E+00	4.15E+01	1.41E+01
		POBL−ADE	0.00E+00	8.10E−02	2.21E−02			POBL−ADE	4.73E+00	9.45E+00	7.81E+00
		FWA−DE	0.00E+00	3.69E−02	7.40E−03			FWA−DE	5.64E+00	9.05E+00	7.33E+00
		FERDE	0.00E+00	0.00E+00	0.00E+00			FERDE	2.25E+00	6.07E+00	3.96E+00
		CDEOA	0.00E+00	4.42E−02	0.00E+00			CDEOA	3.47E+00	1.73E+01	7.51E+00
F_8	*Shifted Rastrigin*	FCDE	4.28E+01	1.47E+02	9.25E+01	F_{16}	*SRESF6*	FCDE	1.07E+01	1.29E+01	1.21E+01
		POBL−ADE	3.75E+01	9.30E+01	5.39E+01			POBL−ADE	9.09E+00	1.11E+01	1.06E+01
		FWA−DE	0.00E+00	0.00E+00	0.00E+00			FWA−DE	1.03E+01	1.14E+01	1.10E+01
		FERDE	0.00E+00	0.00E+00	0.00E+00			FERDE	1.00E+01	1.20E+01	1.13E+01
		CDEOA	2.69E+01	8.95E+01	4.78E+01			CDEOA	9.11E+00	1.21E+01	1.08E+01

SREGR: Shifted and Rotated Expanded Griewank’s plus Rosenbrock, SRESF6: Shifted and Rotated Expanded Scaffer’s F6 Function

Table A.5: Comparison of OptBees, FERDE, RSDE, and CDEOA over 7 test functions of 30 dimensions using 300,000 functions evaluations. "Mean Error" and "Std Dev" indicate the average and standard deviation of the error values obtained in 25 runs, respectively

Functions		OptBees		FERDE		RSDE		CDEOA		ICDEOA
		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev		Mean Error±Std Dev
Unimodal Functions	F_1	8.57E+04±6.96E+02	−	5.41E+02±6.40E+02	+	1.50E+03±1.70E+03	+	4.64E+04±4.26E+04	−	7.32E+03±2.45E+03
	F_2	0.00E+00±6.40E−02	≈	2.39E−03±3.24E−03	−	0.00E+00±0.00E+00	≈	0.00E+00±0.00E+00	≈	0.00E+00±0.00E+00
	F_3	8.41E−03±2.94E+00	−	1.13E−03±7.36E−04	−	4.74E−02±1.16E−01	−	1.05E−05±6.68E−05	−	1.20E−07±3.38E−07
Simple Multimodal Functions	F_4	1.64E+01±2.88E+00	−	6.25E−01±1.46E+00	−	3.05E+00±1.34E+01	−	4.46E+00±2.40E+01	−	1.49E−02±1.27E−02
	F_5	2.00E+01±1.29E−04	≈	2.00E+01±7.17E−05	≈	2.03E+01±9.88E−02	−	2.00E+01±5.42E−06	≈	2.00E+01±2.00E−03
	F_6	1.64E+01±1.28E+00	−	1.82E+01±1.72E+00	−	5.16E+00±2.01E+00	≈	6.30E+00±2.33E+00	−	5.36E+00±1.86E+00
	F_7	3.75E−02±1.40E−01	−	0.00E+00±0.00E+00	+	8.46E−04±1.59E−03	+	5.99E−03±8.69E−03	+	8.66E−03±1.34E−02
−		5		4		3		4		
+		0		2		2		1		
≈		2		1		2		2		

"−", "+", and "≈" denote that the performance of the corresponding algorithm is worse than, better than, and similar to that of ICDEOA, respectively

REFERENCES

[1] Global optimization test problem. retrieved september 2015, from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/hedarfiles/testgo.htm.

[2] Abraham, A., Biswas, A., Dasgupta, S., and Das, S. Analysis of reproduction operator in bacterial foraging optimization algorithm. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on* (2008), IEEE, pp. 1476–1483.

[3] Biswas, A., Das, S., Abraham, A., and Dasgupta, S. Stability analysis of the reproduction operator in bacterial foraging optimization. *Theoretical Computer Science 411*, 21 (May 2010), 2127–2139.

[4] Biswas, A., Dasgupta, S., Das, S., and Abraham, A. A synergy of differential evolution and bacterial foraging optimization for global optimization. *Neural Network World 17*, 6 (2007), 607.

[5] Biswas, A., Dasgupta, S., Das, S., and Abraham, A. Synergy of PSO and bacterial foraging optimization—a comparative study on numerical benchmarks. In *Innovations in Hybrid Intelligent Systems*. Springer, 2007, pp. 255–263.

[6] Blum, C., and Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR) 35*, 3 (2003), 268–308.

[7] Bäck, T., and Schwefel, H.-P. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation 1*, 1 (Mar. 1993), 1–23.

[8] Cabrera, J. C. F., and Coello, C. A. C. Handling constraints in particle swarm optimization using a small population size. In *MICAI 2007: Advances in Artificial Intelligence*. Springer, 2007, pp. 41–51.

[9] Caraffini, F., Neri, F., and Poikolainen, I. Micro-differential evolution with extra moves along the axes. In *Proceedings. Differential Evolution (SDE), 2013 IEEE Symposium on* (2013), IEEE, pp. 46–53.

[10] Castillo-Villar, K. K. Metaheuristic Algorithms Applied to Bioenergy Supply Chain Problems: Theory, Review, Challenges, and Future. *Energies 7*, 11 (2014), 7640–7672.

[11] Castillo-Villar, K. K., and Herbert-Acero, J. F. A Metaheuristic-Based Approach for the Capacitated Supply Chain Network Design Problem Including Imperfect Quality and Rework. *IEEE Computational Intelligence Magazine 9*, 4 (Nov. 2014), 31–45.

[12] Chu, Y., Mi, H., Liao, H., Ji, Z., and Wu, Q. A Fast Bacterial Swarming Algorithm for high-dimensional function optimization. In *IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)* (June 2008), pp. 3135–3140.

[13] Das, S., Dasgupta, S., Biswas, A., Abraham, A., and Konar, A. On Stability of the Chemotactic Dynamics in Bacterial-Foraging Optimization Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 39*, 3 (May 2009), 670–679.

[14] Dasgupta, S., Biswas, A., Das, S., Panigrahi, B. K., and Abraham, A. A micro-bacterial foraging algorithm for high-dimensional optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on* (2009), IEEE, pp. 785–792.

[15] Dasgupta, S., Das, S., Abraham, A., and Biswas, A. Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis. *IEEE Transactions on Evolutionary Computation 13*, 4 (Aug. 2009), 919–941.

[16] De Jong, K. A., and Spears, W. M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence 5*, 1 (1992), 1–26.

[17] Dorigo, M., Birattari, M., and Stützle, T. Ant colony optimization. *Computational Intelligence Magazine, IEEE 1*, 4 (2006), 28–39.

[18] Eberhart, R. C., and Kennedy, J. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (1995), vol. 1, New York, NY, pp. 39–43.

[19] Eiben, A. E., and Schippers, C. A. On evolutionary exploration and exploitation. *Fundamenta Informaticae 35*, 1 (1998), 35–50.

[20] Gamperle, R., Muller, S. D., and Koumoutsakos, P. A parameter study for differential evolution. *Advances in intelligent systems, fuzzy systems, evolutionary computation 10* (2002), 293–298.

[21] Glover, F., and Laguna, M. *Tabu Search.* Springer US, Boston, MA, 1999, pp. 2093–2229.

[22] Handfield, R. B., Nichols, E. L., et al. *Introduction to supply chain management*, vol. 1. prentice Hall Upper Saddle River, NJ, 1999.

[23] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[24] Hu, Z., Bao, Y., and Xiong, T. Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2259–2265.

[25] Iorio, A. W., and Li, X. Solving rotated multi-objective optimization problems using differential evolution. In *AI 2004: Advances in artificial intelligence*. Springer, 2005, pp. 861–872.

[26] Jarraya, Y., Bouaziz, S., Alimi, A. M., and Abraham, A. A hybrid computational chemotaxis in bacterial foraging optimization algorithm for global numerical optimization. In *Cybernetics (CYBCONF), 2013 IEEE International Conference on* (2013), IEEE, pp. 213–218.

[27] Jingqiao Zhang, and Sanderson, A. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Transactions on Evolutionary Computation 13*, 5 (Oct. 2009), 945–958.

[28] Karaboga, D., and Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization 39*, 3 (2007), 459–471.

[29] Kennedy, J. Particle Swarm Optimization. In *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer US, 2011, pp. 760–766.

[30] Kim, D. H., Abraham, A., and Cho, J. H. A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Information Sciences 177*, 18 (2007), 3918–3937.

[31] Kim, D. H., and Cho, J. H. Bacterial foraging based neural network fuzzy learning. In *Proc. in IICAI* (2005), pp. 2030–2036.

[32] Korani, W. M., Dorrah, H. T., and Emara, H. M. Bacterial foraging oriented by particle swarm optimization strategy for PID tuning. In *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on* (2009), IEEE, pp. 445–450.

[33] Krishnakumar, K. Micro-Genetic Algorithms For Stationary And Non-Stationary Function Optimization. vol. 1196, pp. 289–296.

[34] Li, Z., Shang, Z., Qu, B. Y., and Liang, J. J. Differential Evolution strategy based on the constraint of fitness values classification. In *Proceedings. Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 1454–1460.

[35] Liang, J. J., Qu, B. Y., and Suganthan, P. N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory* (2013).

[36] Liang, J. J., Suganthan, P. N., and Deb, K. Novel composition test functions for numerical global optimization. In *Proceedings. Swarm Intelligence Symposium, 2005. SIS 2005.IEEE* (2005), IEEE, pp. 68–75.

[37] Liu, Y., and Passino, K. Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors. *Journal of Optimization Theory and Applications 115*, 3 (Dec. 2002), 603–628.

[38] Luke, S. *Essentials of Metaheuristics.* Lulu, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/, accessed October 15, 2014.

[39] Maia, R. D., Castro, L. N. d., and Caminhas, W. M. OptBees-A Bee-Inspired Algorithm for Solving Continuous Optimization Problems. In *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on* (2013), IEEE, pp. 142–151.

[40] Mentzer, J. T., DeWitt, W., Keebler, J. S., Min, S., Nix, N. W., Smith, C. D., and Zacharia, Z. G. Defining supply chain management. *Journal of Business logistics 22*, 2 (2001), 1–25.

[41] Mishra, S. A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. *Evolutionary Computation, IEEE Transactions on 9*, 1 (2005), 61–73.

[42] Mishra, S., and Bhende, C. N. Bacterial foraging technique-based optimized active power filter for load compensation. *Power Delivery, IEEE Transactions on 22*, 1 (2007), 457–465.

[43] Molga, M., and Smutnicki, C. Test functions for optimization needs. *Test functions for optimization needs* (2005).

[44] Nesmachnow, S., Cancela, H., and Alba, E. A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling. *Applied Soft Computing 12*, 2 (Feb. 2012), 626–639.

[45] Olguin-Carbajal, M., Alba, E., and Arellano-Verdejo, J. Micro-differential evolution with local search for high dimensional problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (2013), IEEE, pp. 48–54.

[46] Olorunda, O., and Engelbrecht, A. P. Differential evolution in high-dimensional search spaces. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (2007), IEEE, pp. 1934–1941.

[47] Pardalos, P. *Handbook of Combinatorial Optimization.* Springer Verlag, Dordrecht, 2005.

[48] Parsopoulos, K. E. Cooperative micro-differential evolution for high-dimensional problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (2009), ACM, pp. 531–538.

[49] Parsopoulos, K. E. Parallel cooperative micro-particle swarm optimization: A master slave model. *Applied Soft Computing 12*, 11 (2012), 3552–3579.

[50] Passino, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE 22*, 3 (2002), 52–67.

[51] Poli, R., Kennedy, J., and Blackwell, T. Particle swarm optimization. *Swarm Intelligence 1*, 1 (Aug. 2007), 33–57.

[52] Price, K. *An introduction to differential evolution, New ideas in optimization.* McGraw-Hill Ltd., UK, Maidenhead, UK, 1999.

[53] Qin, A., Huang, V., and Suganthan, P. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. *IEEE Transactions on Evolutionary Computation 13*, 2 (Apr. 2009), 398–417.

[54] Qin, A. K., and Li, X. Differential evolution on the CEC-2013 single-objective continuous optimization testbed. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (2013), IEEE, pp. 1099–1106.

[55] Qu, B. Y., Liang, J. J., Xiao, J. M., and Shang, Z. G. Memetic differential evolution based on fitness Euclidean-distance ratio. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2266–2273.

[56] Rahnamayan, S., and Tizhoosh, H. Image thresholding using micro opposition-based Differential Evolution (Micro-ODE). In *IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)* (June 2008), pp. 1409–1416.

[57] Ramalhinho Dias Lourenço, H. Supply chain management: an opportunity for metaheuristics. *UPF Economics and Business Working Paper*, 538 (2001).

[58] Ronkkonen, J., Kukkonen, S., and Price, K. Real-parameter optimization with differential evolution. In *The 2005 IEEE Congress on Evolutionary Computation, 2005* (Sept. 2005), vol. 1, pp. 506–513 Vol.1.

[59] Sotelo-Figueroa, M. A., Soberanes, H. J. P., Carpio, J. M., Huacuja, H. J. F., Reyes, L. C., and Alcaraz, J. A. S. Evolving bin packing heuristic using micro-differential evolution with indirect representation. In *Recent Advances on Hybrid Intelligent Systems.* Springer, 2013, pp. 349–359.

[60] Storn, R. On the usage of differential evolution for function optimization. In *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American* (1996), IEEE, pp. 519–523.

[61] Storn, R., and Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization 11*, 4 (1997), 341–359.

[62] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL report 2005005* (2005).

[63] Tripathy, M., Mishra, S., Lai, L. L., and Zhang, Q. P. Transmission loss reduction based on FACTS and bacteria foraging algorithm. In *Parallel Problem Solving from Nature-PPSN IX.* Springer, 2006, pp. 222–231.

[64] Wang, Y., Cai, Z., and Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *Evolutionary Computation, IEEE Transactions on 15*, 1 (2011), 55–66.

[65] WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing 4*, 2 (June 1994), 65–85.

[66] XU, C., HUANG, H., AND YE, S. A Differential Evolution with Replacement Strategy for Real-Parameter Numerical Optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 1617–1624.

[67] YANG, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[68] YILDIZ, Y. E., ALTUN, O., AND TOPAL, A. O. Chemotaxis Differential Evolution Optimization Algorithm Comparative study on mutation strategies. In *Computer Science 2015* (Durres, Albania, Sept. 2015), pp. 230–235.

[69] YILDIZ, Y. E., ALTUN, O., AND TOPAL, A. O. Computational chemotaxis in micro bacterial foraging optimization for high dimensional problems: A comparative study on numerical benchmark. *International Journal of Computer Applications 124*, 4 (2015).

[70] YILDIZ, Y. E., ALTUN, O., AND TOPAL, A. O. The Effects of Crossover and Mutation Rates on Chemotaxis Differential Evolution Optimization Algorithm. *Journal of Nature and Technical Sciences XX*, 1 (Aug. 2015), 89–101.

[71] YU, C., KELLEY, L., ZHENG, S., AND TAN, Y. Fireworks algorithm with differential mutation for solving the cec 2014 competition problems. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 3238–3245.

[72] YILDIZ, Y. E., AND ALTUN, O. Chemotaxis Differential Evolution Optimization Algorithm for Minimization of Supply Chain Cost with Embedded Risk. In *International Conference on Image Processing, Production and Computer Science* (Istanbul (Turkey), June 2015), pp. 31–37.

[73] YILDIZ, Y. E., AND ALTUN, O. Hybrid achievement oriented computational chemotaxis in bacterial foraging optimization: a comparative study on numerical benchmark. *Soft Computing* (May 2015), 1–17.

[74] YILDIZ, Y. E., ALTUN, O., AND TOPAL, A. O. Improved Chemotaxis Differential Evolution Optimization Algorithm. In *International Conference on Computing Informatics 2015* (Istanbul (Turkey), Aug. 2015), pp. 312–317.

[75] ZAHARIE, D. Critical values for the control parameters of differential evolution algorithms. In *Proceedings of MENDEL* (2002), vol. 2002.

[76] ČREPINŠEK, M., LIU, S.-H., AND MERNIK, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys 45*, 3 (June 2013), 1–33.

CIRRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Yildiz, Yunus Emre

Nationality: Turkish (TC)

Date and Place of Birth: 07.05.1982, Istanbul

Marital Status: Married

Phone: 00 355 673022320

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.S.	Epoka University	2011
B.S.	Marmara University	2004
High School	Orhangazi Anatolian Technical High School	2000

PROFESSIONAL EXPERIENCE

Year	**Place**	**Enrollment**
2004 - 2009	Gulistan Education Company	IT Teacher
2009 - 2016	Turgut Ozal Education Company	IT Teacher

PUBLICATIONS

The content of this thesis has been published before in the following papers:

[1] Y. E. Yıldız and O. Altun, "Hybrid achievement oriented computational chemotaxis in bacterial foraging optimization: a comparative study on numerical benchmark," Soft Computing, SPRINGER, pp. 1–17, May 2015.

[2] Y. E. Yıldız, O. Altun, and A. O. Topal, "Computational Chemotaxis in Micro Bacterial Foraging Optimization for High Dimensional Problems: A Comparative Study on Numerical Benchmark," International Journal of Computer Applications, vol. 124, no. 4, 2015.

[3] Y. E. Yıldız, O. Altun, and A. O. Topal, "Improved Chemotaxis Differential Evolution Optimization Algorithm," presented at the International Conference on Computing and Informatics, Istanbul (Turkey), 11-13 August, pp. 321–316.

[4] Y. E. Yildiz, O. Altun, and A. O. Topal, "The effects of crossover and mutation rates on chemotaxis differential evolution optimization algorithm," JNTS, vol. XX, no. 1, pp. 89–101.

[5] Y. E. Yıldız and O. Altun, "Chemotaxis Differential Evolution Optimization Algorithm for Minimization of Supply Chain Cost with Embedded Risk," presented at the International Conference on Image Processing, Production and Computer Science, Istanbul (Turkey), June 3-5, pp. 31–37, 2015.

[6] Y. E. Yıldız, O. Altun, and A. O. Topal, "Chemotaxis Differential Evolution Optimization Algorithm Comparative study on mutation strategies," presented at the Computer Science, Durres, Albania, 08-10 September, pp. 230–235, 2015

Other papers with co-author contribution:

[7] A. O. Topal, O. Altun, and Y. E. Yıldız, "Micro Bat Algorithm for High Dimensional Optimization Problems," International Journal of Computer Applications, vol. 122, no. 12, 2015.

[8] A. O. Topal, O. Altun, and Y. E. Yıldız, "Emprical Study of Dynamic Virtual Bats Algorithm," presented at the Conference of Computer Science, Durres, Albania, 08-10

September, pp. 230–235, 2015

[9] A. O. Topal, O. Altun, and Y. E. Yıldız, “An Effective Hybrid of Bat Algorithm and Hill Climbing for Global Optimization of High- dimensional Functions,” JNTS, vol. XX, no. 2, pp. 87–100.