Hidden-Markov-Based Self-adaptive Differential Evolution

Marwa Mohamed ElSaied Hassan Keshk

B.Sc. (Information Systems), Helwan University, Egypt.

A thesis submitted in fulfilment of the requirements

for the degree of

Master of Computer Science

UNSW
AUSTRALIA

School of Engineering and Information Technology

The University of New South Wales

Australia

August 2017

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: **Mohamed ElSaied Hassan Keshk**

First name: **Marwa** Other name/s:

Abbreviation for degree as given in the University calendar:**Master**

School: **SEIT** Faculty: **UNSW Canberra**

Title**: Hidden-Markov-Based Self-adaptive Differential Evolution**

Abstract 350 words maximum: (PLEASE TYPE)

Heuristic search is an efficient way to solve complex optimization problems, and sometimes it is the only way to do so. Differential Evolution (DE) is a population-based heuristic search suitable mostly for continuous optimization problems. The efficiency of DE to optimize a problem can largely degrade if the right values for its parameters are not chosen. Finding the right values for DE's parameters is a non-trivial task. Many researchers resort to parameter tuning and self-adaptation mechanisms. Existing methods vary in their performance and design philosophies.
In this thesis, I start by introducing a semantic evolutionary visualization framework to investigate evolutionary dynamics. The different visualizations track the ongoing changes within an evolutionary run by exploring pedigree trees and the fitness landscapes. The visualization alone was not sufficient to shed light on a very high dimensional space. Consequently, I resorted to introducing a new self-adaptive algorithm using Hidden Markov Models (HMMs).
Markov models have been used extensively in the past to analyze convergence of evolutionary optimization methods. I have leveraged this opportunity to introduce a new algorithm that we call DE-HMM, where HMMs is used for real-time learning of evolutionary dynamics to allow for dynamic adjustment of the two intrinsic DE parameters: F and CR.
DE-HMM categorizes each evolutionary transition into two discrete states; low and high, representing the rate of change in a population over time. The HMM posterior and likelihood ratios are estimated to assign the values for F and CR during the evolutionary process. Two unconstrained benchmark set are used to assess DE-HMM performance, demonstrating its overall superiority in terms of solution quality and computational resources, when compared to other state-of-the art algorithms.
The self-adaptive DE-HMM is then augmented with local search to solve constrained optimization problems. A two-stage method is introduced; with the two states being either global or local based on the degree of feasibility and rate of diversity. The methodology demonstrated competitive results when tested on the constrained CEC2010 benchmark dataset.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

Copyright Statement

Authenticity Statement

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed

Data: .....

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked at UNSW or elsewhere, during my candidature, is fully acknowledged. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed

Data: .....

Acknowledgment

In the name of Allah, all the faithful gratitude, praises are due to Allah, the Almighty God, the Ubiquitous God, the most Gracious and Merciful. Thanks to ALLAH for his support and blessing to complete this work and I hope that this study would be beneficial and valuable as a scientific research in the Evolutionary Computation field.

I would like to express my genuine thanks and deepest gratitude to everyone who have supported and assisted me to successfully achieve this work. First of all, I would like to express my sincere appreciation to my supervisor, Professor Hussein Abbass, and my Co-supervisor, Dr. Hemant Singh for their outstanding supervision, continuous and precious guidance, insightful discussions and their effort for reviewing and providing me genuine feedback of my publications and thesis.

I would like to express my great thanks to Mrs. Denise Russell for her helpful suggestions by proofreading my publications and thesis. I would like to thank the University of New South Wales, Canberra at the Australian Defense Force Academy for providing me with a scholarship to study at that institution.

I thank my mother, father, brother and sister for supporting and encouraging me to complete my master degree. I wish to express my gratitude and sincere appreciation to my husband, Nour, who always helps me a lot in my work and my life. Also, I wish to thank my little daughter, Nardeen, who has always been the flower of my life.

I would like to thank everyone at the Trusted Autonomy group for the good time we spent, sharing ideas and learning from each others. Moreover, I offer my appreciation to the school administration and IT support staffs who provided me with all the necessary facilities. Finally, many thanks for my friends in canberra

and in my home country who support and assist me all the time to finish my master study.

Abstract

Heuristic search is an efficient way to solve complex optimization problems, and sometimes it is the only way to do so. Differential Evolution (DE) is a population-based heuristic search suitable mostly for continuous optimization problems. The efficiency of DE to optimize a problem can largely degrade if the right values for its parameters are not chosen. Finding the right values for DE's parameters is a non-trivial task. Many researchers resort to parameter tuning and self-adaptation mechanisms. Existing methods vary in their performance and design philosophies.

In this thesis, I start by introducing a semantic evolutionary visualization framework to investigate evolutionary dynamics. The different visualizations track the ongoing changes within an evolutionary run by exploring pedigree trees and the fitness landscapes. The visualization alone was not sufficient to shed light on a very high dimensional space. Consequently, I resorted to introducing a new self-adaptive algorithm using Hidden Markov Models (HMMs).

Markov models have been used extensively in the past to analyze convergence of evolutionary optimization methods. I have leveraged this opportunity to introduce a new algorithm that we call DE-HMM, where HMMs is used for real-time learning of evolutionary dynamics to allow for dynamic adjustment of the two intrinsic DE parameters: F and CR.

DE-HMM categorizes each evolutionary transition into two discrete states; low and high, representing the rate of change in a population over time. The HMM posterior and likelihood ratios are estimated to assign the values for F and CR during the evolutionary process. Two unconstrained benchmark set are used to assess DE-HMM performance, demonstrating its overall superiority in terms of solution quality and computational resources, when compared to other state-of-the art algorithms.

The self-adaptive DE-HMM is then augmented with local search to solve constrained optimization problems. A two-stage method is introduced; with the two states being either global or local based on the degree of feasibility and rate of diversity. The methodology demonstrated competitive results when tested on the constrained CEC2010 benchmark dataset.

Keywords

Meta-heuristic Optimization, Evolutionary Algorithm, Differential Evolution, Parameter Control, Hidden Markov Model, Multi-stages Mutation, Constrained Optimization

List of publications

Journal Articles

1. M. Keshk, H. Singh, and H. Abbass, "Automatic Estimation of Differential Evolution Parameters using Hidden Markov Models," under review. (Based on Chapter 4)

Conference Papers

1. M. Keshk, "Semantic Evolutionary Visualization," **Lecture Notes in Swarm Intelligence**, Springer, 2017. (Based on Chapter 3)

Contents

Title Page i
Copyright Statement iv
Originality statement vi
Acknowledgments vii
Abstract ix
Keywords xi
List of publications xii
Table of Contents xiii
List of Figures xvii
List of Tables xx
List of Terms xxvi
1 Introduction 1
1.1 Background 1
1.2 Problem Description 2
1.3 Gaps and Objectives of Our Research 4
1.4 Thesis Contributions 5
1.5 Thesis Structure 6
2 Background Study and Literature Review 8
2.1 Evolutionary algorithms 8
2.1.1 Differential Evolution (DE) 9
2.1.2 Significance of DE parameters 14
2.1.3 Classification of parameters control strategies 17

2.2 Recent advances of automatic approaches for DE parameter control . . . 21
2.2.1 DE adaptation methods with multiple strategies and control parameter settings . . . 22
2.2.2 Adaptation of F and CR DE control parameters . . . 27
2.2.3 DE methods with new offspring strategy . . . 31
2.2.4 DE with PS control . . . 35
2.3 Constrained Optimization . . . 38
2.3.1 Constrained Handling Techniques (CHTs) . . . 39
2.4 Recent advances of handling constraints in DE . . . 44
2.4.1 Combining CHTs into DE . . . 44
2.4.2 Repairing methods for EAs . . . 48
2.4.3 Hybrid ensemble of EAs and operators . . . 49
2.5 Hidden Markov Model (HMM) . . . 51
2.5.1 Elements of HMM . . . 52
2.6 Visualization and reasoning for EAs . . . 54
2.6.1 Basic concepts behind reasoning . . . 56
2.6.2 Existing EA visualization techniques and tools . . . 61
2.7 Chapter Summary . . . 68
3 Semantic Evolutionary Visualization framework for visualizing EA . . . 69
3.1 Introduction . . . 69
3.2 Semantic Evolutionary Visualization (SEV) . . . 71
3.2.1 Main components of SEV . . . 71
3.3 Evaluation of SEV framework . . . 78
3.3.1 Test functions and parameter setup . . . 78
3.3.2 Experiment results . . . 79

3.3.3 Comparison with existing techniques . . . 92
3.4 Chapter summary . . . 93
4 A novel self-tuning methodology for solving unconstrained optimisation problems . . . 95
4.1 Introduction . . . 95
4.2 General architecture of the proposed methodology (DE-HMM) . . . 97
4.2.1 DE combined with HMM . . . 97
4.2.2 Main components of DE-HMM . . . 102
4.3 Experimental study . . . 108
4.3.1 Benchmark problems . . . 108
4.3.2 Experiments setup . . . 108
4.3.3 Experimental results . . . 112
4.3.4 Time Complexity . . . 125
4.4 Chapter summary . . . 128
5 Two Strategies DE-HMM Combined with Local Search for Constrained Optimization Problems . . . 130
5.1 Introduction . . . 130
5.2 Algorithmic Framework of MS-DEHMM-L . . . 132
5.2.1 Constrained Handling . . . 133
5.2.2 HMM for control parameter setting (F and CR) . . . 135
5.2.3 DE combined with local search . . . 137
5.2.4 Two strategies and control parameters in MS-DEHMM-L . . . 140
5.3 Experimental Results and Analysis . . . 141
5.3.1 Benchmark problems . . . 141
5.3.2 Experiment setup . . . 142
5.3.3 Experimental Results . . . 143

5.4 Chapter summary 148

6 Conclusions and Recommendations for Future Work 150

6.1 Summary of research conducted 150

6.2 Research Findings 153

6.2.1 SEV framework 153

6.2.2 Methodology of self-adaptive DE-HMM 153

6.2.3 Multi-stage mutation MSMODE-HMM methodology 155

6.3 Recommendations for Future Work 156

References. 157

A CEC2005 Comparison of DE-HMM different variants (different parameter-choices) 179

B Analysis of using different search operators on DE-HMM 183

C CEC2005 Comparison of DE-HMM and C-DE on CEC2005 and CEC2014 datasets 185

D CEC2005 Comparison of DE-HMM and State-of-the-art algorithms . . . 190

E CEC2014 Comparison of DE-HMM and Up-to-date algorithms 204

F CEC2005 and 2014 Time Complexity 212

G Example of DE-HMM procedure 214

List of Figures

2.1 Overlaying meta-optimization conceptual model 15

2.2 Hierarchy of parameter control schema 18

2.3 Classification of parameter control methods 22

2.4 General HMM structure 52

2.5 Process of visual analytical reasoning 56

2.6 Basic reasoning process 58

2.7 Types of Reasoning 59

2.8 Visualisation techniques classification in EAs. 62

3.1 Semantic Evolutionary Framework (SEV) 72

3.2 Example of transforming DE data using HP algorithm 74

3.3 Pedigree Graph Visualization of the DeJong's two trials using DE algorithm 80

3.4 Pedigree Graph Visualization of the Rastrigin's two trials using DE algorithm 81

3.5 Frequency ofgraph nodes over generations for DeJong and Rastrigin functions, respectively. 83

3.6 Fitness graph for DeJong function (trial 1 and 2), respectively. . . . 85

3.7 Fitness graph for Rastrigin function (trial 1 and 2), respectively. . . 86

3.8 Correlation coefficient of Child & Parents for DeJong trials 1 and 2, respectively. 87

3.9 Correlation coefficient of Child & Parents for Rastrigin trials 1 and 2, respectively. 88

3.10 A Scatter diagram showing the relationship between the child and first parent fitness for DeJong and Rastrigin functions, respectively. 90

3.11 Linear regression for trial 1 and 2 of Dejong F1, respectively 91

3.12 Standard deviation of fitted values for Dejong F1 linear regression . 91

4.1 Diagrammatic example of HMM transition diagram for DE. the low and high states represent the low and high change rates of DE, respectively, and the observation set (O = {1,2,3}) reflects the three dimensions of a population 97

4.2 Encoding of control parameters with individuals 100

4.3 Complete procedure for computing DE control parameters. 103

4.4 General flowchart of the proposed DE-HMM algorithm. 106

4.5 Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and its own variants. . . 115

4.6 F and CR evolution of DE-HMM variants for C2 115

4.7 F and CR evolution of DE-HMM variants for C12 116

4.8 Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and C-DE. 118

4.9 Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and all the state-of-art variants 121

4.10 Convergence plots of DE-HMM on F1, F2, F11, F14, and F16 with 30 dimensions where y-axis in log scale. 124

4.11 Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and the up-to-date DE variants. 126

4.12 Time complexity for C3 of CEC 2005 test problems at 10, 30, and 50 dimensions. 127

4.13 Time complexity for F18 of CEC 2014 test problems at 10, 30, and 50 dimensions. 128

5.1 Flow Chart of MS-DEHMM-L methodology 139

List of Tables

3.1 Configuration of DE 79

4.1 CEC2005 Benchmark Functions. 109

4.2 CEC2014 Benchmark Functions. 110

4.3 Critical values $q_{\alpha=0.05}$ for calculating CD of Nemenyi post-hoc test at different K 112

4.4 F and CR values used for DE-HMM variants. 112

4.5 Friedman statistical analysis of different variants of DE-HMM. . . . 114

4.6 Description of the four mutation and crossover setups used in DE-HMM analysis. 118

4.7 Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2005 test functions. . . . 119

4.8 Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2014 test functions. . . . 119

4.9 Rankings obtained from Friedman's test of 10, 30, and 50 dimensional CEC2005 functions of DE-HMM and its DE variants. 121

4.10 Rankings obtained from Friedman's test of 10, 30, and 50 dimensional CEC2005 functions of DE-HMM and its non-DE variants. . . 123

4.11 Rankings obtained from Friedman's test of 10, 30 dimensional CEC2014 functions of DE-HMM and its up-to-date variants. 126

5.1 Characteristics of CEC2010's test functions where I refers to the number of inequality constraints, E is the number of equality constraints, S is seperable, Non-Sep is not seperable and feasibility pro denotes the ratio between feasible region and whole search space . . 142

5.2 Mean and std of function error values of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 10D. 144

5.3 Mean and std of function error values of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 30D. 146

5.4 Feasibility Rate (%) of MS-DEHMM-L and the state-of-art competitors on CEC2010 18-test problems at 10D and 30D 147

5.5 Comparison summary between MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 10D and 30D. 148

5.6 Rankings obtained from Friedman's test of 10, 30D of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems. . . 148

A.1 Mean function error values among DE-HMM variants at 30D of CEC2005 over 30 independent runs. (DE-HMM-minL - DE-HMM-meanE) 180

A.2 Mean function error values among DE-HMM variants at 30D of CEC2005 over 30 independent runs. (DE-HMM-maxG - DE-HMM-minG) 181

A.3 Comparison summary among DE-HMM and its variants and Randgen method on 30D CEC2005 test problems. 182

B.1 Effect of different search operators on DE-HMM at 30D CEC2005 mean function error over 30 independent runs. 183

B.2 Effect of different search operators on DE-HMM at 30D CEC2014 mean function error over 30 independent runs. 184

C.1 Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2005 25-test functions with 10, 30, and 50 dimensions (C1-C15) 186

C.2 Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2005 25-test functions with 10, 30, and 50 dimensions (C16-C25) 187

C.3 Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2014 30-test functions with 10, 30, and 50 dimensions (F1-F15) 188

C.4 Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2014 30-test functions with 10, 30, and 50 dimensions (F16-F30) 189

D.1 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 10D over 30 independent runs (C1-C15). 191

D.2 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 10D over 30 independent runs (C16-C25). 192

D.3 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 30D over 30 independent runs (C1-C15). 193

D.4 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 30D over 30 independent runs (C16-C25). 194

D.5 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 50D over 30 independent runs (C1-C15). 195

D.6 Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 50D over 30 independent runs (C16-C25). 196

D.7 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 10D CEC2005 over 30 independent runs (C1-C15). 197

D.8 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 10D CEC2005 over 30 independent runs (C16-C25). 198

D.9 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 30D CEC2005 over 30 independent runs (C1-C15). 199

D.10 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 30D CEC2005 over 30 independent runs (C16-C25). 200

D.11 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 50D CEC2005 over 30 independent runs (C1-C15). 201

D.12 Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 50D CEC2005 over 30 independent runs (C16-C25). 202

D.13 Comparison summary among DE-HMM and state-of-the-art on the 10D, 30D and 50D CEC2005 test problems. 203

E.1 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 10D over 30 independent runs(F1-F15) 205

E.2 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 10D over 30 independent runs (F16-F30) 206

E.3 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 30D over 30 independent runs(F1-F15) 207

E.4 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 30D over 30 independent runs (C16-C30) 208

E.5 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 50D over 30 independent runs(F1-F15) 209

E.6 Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 50D over 30 independent runs (F16-F30) 210

E.7 Comparison summary among DE-HMM and up-to-date variants on the 10D, 30D and 50D CEC2014 test problems. 211

F.1 Time complexity computations for CEC2005 and CEC2014 datasets. 212

F.2 Time complexity for DE variants on problem C3 of CEC2005 benchmark. 212

F.3 Time complexity for up-to-date DE variants on problem F18 of CEC2014 benchmark. 213

List of Terms

Acronyms	**Description**
EAs	Evolutionary Algorithms
DE	Differential Evolution
COPs	Constrained Optimization Problems
CHTs	Constrained Handling Techniques
HMMs	Hidden Markov Models
LS	Local Search
CPs	Control Parameters
DE-HMM	Differential Evolution based on Hidden Markov Model
MS-DE-HMM-L	Multiple strategies DE-HMM combined with local search
PBF	Population based feature
LDBF	Local Dynamics Based Features
GDBF	Global Dynamics Based Features
SEV	Semantic Evolutionary Visualization
HP	Hierarchy Profile
PCC	Pearson's Correlation Coefficient
LR	Linear regression
Cor_CP	Correlation between child and its parents (dependency ratio)
$Fitted\_GCP$	Linear regression relationship of each generation and its individuals
ES	Evolution Strategy
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
SR	Stochastic Ranking
ATM	Adaptive trade-off model
ABC	Artificial bee colony

Acronyms	**Description**
CBDEX	Center based differential exponential crossover
CI	Computational intelligence
CMA-ES	Co-variance matrix adaptation evolution strategies
CO	Conventional optimization
HS	Harmony search
SBX	Simulated binary crossover
CBDX	Center-Based Differential crossover
εDEag	Constrained differential evolution with an archive and gradient-based mutation
CO-CLPSO	Co-evolutionary comprehensive learning particle swarm optimizer
eABC	Elitist artificial bee colony
jDEsoco	An Improved Self-adaptive Differential Evolution Algorithm in Single Objective Constrained Real-Parameter Optimization
JADE	Adaptive differential evo- lution with optional external archive
SaDE	Differential evolution algorithm with strategy adaptation
jDE	Self-adapting control parameters in differential evolution
CoBiDE	Differential evolution based on co-variance matrix learning and binomial distribution
CPI-DE	Utilizing cumulative population distribution information in differential evolution
LSHADE	Improving the search performance of shade using linear population size reduction
TSDE	Differential evolution with a two-stage optimization mechanism
UMOEA	Testing united multi-operator evolutionary algorithms on the cec2014 real- parameter numerical optimization

Chapter 1

Introduction

In this introductory chapter, firstly, an overview of the background to this study is provided. Then, its problem formulation, research objectives and principal contributions are discussed and, finally, its structure presented.

1.1. Background

During the last few decades, the field of optimization has received a great deal of attention for several research application areas, such as finance, science, engineering and air-traffic control [1–3]. The need to solve optimization problems is not limited to a particular discipline and many real-world systems require optimization methods. An optimization problem can have single or multiple objective (s), which compete to obtain the best solution. Problems with a single objective, which is our focus in this thesis, can be classified based on their properties as linear or non-linear, constrained or unconstrained and with convex or non-convex functions [4].

Optimization methods are commonly divided into two main categories [5, 6]: deterministic and stochastic search methods. The former is the classical branch of optimization techniques in mathematics, which strongly depends on linear algebraic and gradient computations [6]. On the other hand, stochastic methods, such as Evolutionary Algorithms (EAs), are simple and applicable to any types of problems with various characteristics [5]. Due to their general applicability, they have become popular for solving different optimization problems. Usually, it is not necessary to make any assumption about a problem's specifications as it works as black-box optimization methods.

An EA, which mimics natural evolution, is a population-based stochastic algorithm that uses a set of genetic operators to evolve a population of candidate solutions with the aim of finding the best [7]. One of the most popular EAs is Differential Evolution (DE) [8], which has shown its superiority to other EAs with its positive features being its easy implementation, fast convergence because it does not require expensive operators and very simple data structure. It maintains three main operators, mutation, crossover and selection, which are associated with three control parameters, the scaling factor (F), crossover rate (CR) and population size (PS), for determining the optimal solution.

Although DE has a long history of success in solving optimization problems, it suffers from some drawbacks similar to any global optimization method of being incapable of delivering global optimal solutions in a reasonable time frame. [3, 8], stagnation, premature convergence and vulnerable control parameters [7]. Setting appropriate parameter values is one of the persistent challenges in the DE community for achieving better performances as no single parameter is effective for all optimization problems, that is, one value could perform well for one problem but poorly for another. Even for the same problem, there is no guarantee that a chosen parameter value is efficient throughout the evolutionary process. Therefore, it is essential to investigate and analyze the potential process of DE to provide deep insights into the effects of its control parameters on its performance, which could assist in the design of an effective parameter control methodology for solving both Constrained Optimization Problems (COPs) and unconstrained ones.

1.2. Problem Description

It is a big challenge to select the search operators (i.e., mutation and crossover schemes) and settings of the control parameter (i.e., F, CR and PS) for population-based stochastic search algorithms, particularly DE, which greatly influence their performances [7, 9]. The main purpose is to determine the optimal solution that

satisfies the objective function, variable boundaries and/or functional constraints that is affected by the choice of suitable parameter values. This can be achieved by automatically adapting these parameters and restricting them to adequate values without a user having prior knowledge of the relationship between their settings and the characteristics of the optimization problem [5, 7, 9]. Since an a-priori identification of the best parameter settings is always time-consuming and often not realistic [8, 9], one should employ an effective strategy for adjusting them according to the following three main points.

- **Parameter settings are problem dependent [10]**: it is noted that there is no 'free lunch' theory for optimization problems as individual parameter values represent different search directions. In other words, a parameter value may be optimal for one optimization problem but insufficient for another.
- **Parameter settings are evolution dependent [11]**: setting DE's control parameters involves using good parameter values through various DE stages because they generally affect the success of the evolving search process; for instance, this process may proceed quickly in its early stages but then slow down as it approaches near-optimal solutions in its later ones.
- **Interactions among the parameters can be complex [12]**: DE's parameters can affect each other and improve or deteriorate the DE's performance. These interactions are always complex due to the difficulty of selecting appropriate parameter values that reflect the best solutions.

In the literature, there are several adaptive and self-adaptive methods for controlling DE's parameters [8, 9, 12]. However, most have shortcomings as they adapt the F and CR parameters independently whereas the performance of DE depends mainly on a combination of them. Therefore, these control parameters have to be carefully adjusted to improve the quality of candidate solutions according to the ongoing evolutionary dynamics of the DE process [7, 9].

The abovementioned issues provide inspiration for developing effective methods, which investigate and provide insights into the evolutionary search process so that a better way of adjusting DE's control parameters for solving unconstrained and COPs.

1.3. Gaps and Objectives of Our Research

Setting the intrinsic parameters of DE (F and CR) through its evolutionary process can, itself, be a complex optimization problem, as discussed in the previous section. Although several methods have been introduced, none can guarantee the best choices for solving different optimization problems and different stages in a problem considering these parameters' actual dependencies.

Since it seems that these intrinsic parameters should be automatically adjusted when the DE evolutionary process is interpreted, it is necessary to develop new methods that help to explore the internal procedure of DE and effectively control these parameters.

Therefore, ***the main objective of this thesis is to develop a novel method, which aims to automatically adjust the F and CR parameters of DE by coupling the Hidden Markov Model (HMM) with the classical DE to enhance the performance of DE for solving a wide range of COPs and unconstrained ones.*** To achieve this, the following three sub-objectives are proposed.

- To develop a Semantic Evolutionary Visualization (SEV) framework for tracking and interpreting the various dynamics of the DE search process in order to conduct direct analyses of the impacts of different settings of DE's intrinsic parameters to improve its performance.

- To develop a novel self-adaptive method that automatically adapts the DE's control parameters (F and CR) by incorporating the HMM into the DE procedure,

called the DE-HMM method, for solving unconstrained optimization problems. The major motivation behind using HMM learning algorithms is that the previous studies have applied finite Markov models as a way to analyze the evolutionary methods' behavior, however, there is no research study that has focused on employing Markov Chains to enhance the performance of evolutionary computation. In contrast, differential evolution was developed for improving Markov chains [13]. It demonstrates competitive performances compared with those of its own variants and other state-of-the-art methods, as validated using two different benchmark datasets.

- To extend the self-adaptive DE-HMM method to solve COPs by combining the local search operator as an alternative way for guiding and accelerating the search toward the feasible regions. Then the DE basic procedure is applied to find the optimal solution within the feasible era. Through these two processes, two different mutation strategies and estimated F and CR are assigned separately based on the nature of each process. This proposed methodology is known as MS-DEHMM-L. It is tested and compared with previously constrained algorithms, showing promising results.

1.4. Thesis Contributions

This thesis contributes to the field of Evolutionary Computation (EC), particularly DE, by developing three methodologies that address the above three problems and improve the performance of DE for solving both unconstrained and COPs, as detailed below.

- **SEV framework:** this new visualization framework aims to provide a full understanding and analysis of all the evolutionary dynamics occurring during an EA procedure, particularly those in DE, to demonstrate the influence of different control parameter settings on the quality of its solutions.

- **Self-adaptive DE-HMM methodology**: a classical DE incorporated with the HMM as a parameter control method, which is developed to automatically adjust the F and CR parameters to efficiently solve unconstrained problems. It demonstrates outstanding performances compared with those of several DE and non-DE variants. In addition, the speed of DE is not greatly impacted by using a HMM for problems with different dimensions.
- **MS-DEHMM-L methodology**: a new DE algorithm with a HMM for controlling its F and CR parameters uses the strengths of the local search specialized operator and different mutation strategies for increasing the feasibility and optimality through the evolution until the best solution is obtained at the end of the search process. An evaluation of this new method demonstrates its superior performance compared with some state-of-the-art algorithms in the literature in terms of the quality of solutions.

1.5. Thesis Structure

This dissertation has six chapters and is systematically divided as follows.

- Chapter 1: Introduction
- Chapter 2: Background Study and Literature Review
- Chapter 3: Semantic Evolutionary Visualization Framework For Visualizing EA
- Chapter 4: A Novel Self-adaptive DE-HMM for Solving Unconstrained Optimization Problems
- Chapter 5: Two Strategies DE-HMM with Local Search For Constrained Optimization Problems
- Chapter 6: Conclusions and Future Research

Chapter 1 discusses the background to, and problem formulation, objectives and contributions of this study as well as the structure of this thesis.

Chapter 2 presents the theoretical background to this research, reviews the related literature and introduces the basics of EAs, especially DE, and visualization concepts. Then, previous work related to visualization in EAs and other research studies that adapt DE control parameters are discussed. Finally, the fundamentals of COPs and constraint-handling techniques combined with recent EA variants are illustrated.

Chapter 3 proposes a new SEV framework for visualizing DE and exploring its processes as a way of understanding the impact of its control parameters on its performance. The experimental results and analysis of this framework are reported and compared with those from previous studies.

In Chapter 4, a novel self-adaptive DE-HMM method for solving unconstrained optimization problems is proposed. It couples a HMM with the DE process to automatically adjust the DE's F and CR control parameters. The benchmark test suites used to evaluate this method are described and the DE-HMM's results and analysis provided to demonstrate its superiority to other state-of-the-art algorithms.

Chapter 5 details a new method for dealing with COPs that incorporates the best variant obtained from Chapter 4 with a local search operator. The benchmark dataset used for its assessment shows that our method performs well.

Finally, Chapter 6 presents this study's conclusions and contributions, and suggests potential research directions.

Chapter 2

Background Study and Literature Review

This chapter briefly describes the theoretical background and previous studies related to the work in this thesis. Firstly, the basic concepts of Evolutionary Algorithms (EAs), particularly Differential Evolution (DE), are discussed and then the significance of DE control parameters (CPs) and the criteria for classifying their strategies is provided, with previously proposed DE variants for automatically adapting them reviewed. Descriptions of the concepts of constrained optimization, including well-known constraint-handling techniques (CHTs), and the recent EA variants combined with them to solve constrained optimization problems (COPs) are presented. Then, the basic concepts of Hidden Markov Model are illustrated followed by the recent visualization techniques used to explore EAs' dynamics to reveal the importance of adjusting the CPs to enhance its performance. The aim of this literature review is to demonstrate the role of visualization in revealing an EA's potential processes using different parameter settings and determine the main gaps in attempts undertaken in previous studies to adjust DE's parameters to improve its performances for both unconstrained and constrained problems.

2.1. Evolutionary algorithms

An EA is a stochastic, population-based mechanism, which tries to mimic the biological concepts of the Darwinian Theory of Evolution through three basic features, namely, mutation, recombination and natural selection ('survival of the fittest'), in order to solve an optimization problem [14]. The procedures of all

current EAs are quite similar, with only slight differences in their sequences. An EA maintains an initial population consisting of a set of individuals and then uses some rules to select and recombine operators such as mutation and crossover, as described below.

1. Initialization: an initial population of individuals, which can be represented in various formats such as a real value, integer or string is randomly generated.

2. Evaluation: each individual in the population is evaluated based on a fitness function.

3. Recombination: this perturbs the individuals to generate new solutions via mutation and/or crossover operators. In a crossover operation, new offspring are created using two or more original parents while a mutation operation maintains diversity [15] by slightly altering the parent, balancing between exploitation and exploration of the population.

4. Selection: this focuses mainly on individuals with lower fitness values by exploiting their fitness information. Note that, in this thesis, the minimization problems are considered and the lower fitness value is taken to be a better solution, representing the objective value of $f(x)$

These steps are iteratively repeated until the stopping conditions are met.

2.1.1. Differential Evolution (DE)

DE was first proposed in 1995 by Kenneth Price and Rainer M. Storn [16] and, over the last few decades, has become a desirable algorithm for real and continuous optimization [7] due to its simple implementation. It belongs to population-based stochastic search methods and uses the same methodology. In more detail, a population of vectors is initialized randomly over the whole search space. Each of these

vectors (i.e., target vectors) and a new one (i.e., donor vectors) are generated using a mutation operator with a scaling factor. Then, a final trial vector (i.e., offspring) is obtained by combining the target and donor vectors using a crossover operator with a pre-determined crossover rate. Finally, a selection tournament is used to determine which of these vectors survives for the next generation. Although DE usually performs better/well for solving different complex optimization problems, it often suffers from premature convergence, stagnation and/or misadjustments of its CPs [17]. Its three main components [14] are discussed in detail in the following subsections.

A. Mutation

Several mutation schemes for exploring the entire search space with specified boundaries have been proposed. The procedure of mutation aims to produce intermediate population vectors (donor/mutant vectors) from the current ones (target vectors), including one base and some difference ones [16, 18, 19]. A mutation strategy can select a base vector, also called the target vector or parent, and a number of perturbed difference ones [14]. The basic form of mutation is DE/rand/1, where "DE" refers to a DE algorithm, "rand" is the method for selecting the parent to act as a base vector and "1" refers to a single difference vector [14]; for example, a mutant vector is produced by adding a random vector to the processed ones resulting from multiplying the difference between two random vectors by the scaling factor (F) and computed by one of the following five best-known mutation strategies [20].

1. DE/rand/1

$$v_{i,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G}) \tag{2.1}$$

2. DE/rand/2

$$v_{i,G} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G}) + F * (x_{r4,G} - x_{r5,G}) \tag{2.2}$$

3. DE/current-to-rand/1

$$v_{i,G} = x_{i,G} + F * (x_{r1,G} - x_{i,G}) + F * (x_{r2,G} - x_{r3,G}) \tag{2.3}$$

4. DE/best/1

$$v_{i,G} = x_{best,G} + F * (x_{r1,G} - x_{r2,G}) \tag{2.4}$$

5. DE/current-to-best/1

$$v_{i,G} = x_{i,G} + F * (x_{best,G} - x_{i,G}) + +F * (x_{r1,G} - x_{r2,G}) \tag{2.5}$$

6. DE/rand-to-best/1

$$v_{i,G} = x_{r1,G} + F * (x_{best,G} - x_{r1,G}) + F * (x_{r2,G} - x_{r3,G}) \tag{2.6}$$

where $r1$, $r2$, $r3$, $r4$, and $r5$ are randomly chosen integer indices within the range of $[1, PS]$, which are different from index i and x_{best} is the best fitness vector chosen at generation G. F is a positive CP specified in a range of $[0, 1]$ to scale the difference vectors, and G is the generation number.

B. Crossover

The crossover operator is applied after executing the mutation operator using the original population (i.e. parents). It combines the components of the current target and mutant vectors ($x_{i,G}$ and $v_{i,G}$), respectively, with respect to the crossover rate CR, with the aim of creating offspring that will maintain the population's diversity. In DE, there are two key types of crossover variants, namely, binomial and exponential [21], as discussed below.

- Binomial Crossover

The binomial crossover, which is similar to the uniform crossover in EAs, is the most commonly used in DE. It generates a random number from a range of $[0, 1]$ for each component as it is compared with the pre-specified probability of the CR to determine which components can be taken from the mutant and target vectors. In other words, a variable is obtained from a mutant vector with a CR probability and $1 - CR$ from the current target vector.

In order to generate a trial vector using the binomial crossover,

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & if(rand() \leq CR) \mid\mid (j = K) \\ x_{i,,jG} & otherwise \end{cases} \tag{2.7}$$

where $rand \in [0, 1]$, $k \in [1 \ldots\ldots..D]$ is an index chosen randomly to ensure that there is at least one component inherited from $v_{i,jG}$, and CR the crossover probability within the range of $[0, 1]$ for controlling the exploitation of a mutant vector towards new offspring.

- Exponential Crossover

The exponential crossover operates from starting positions randomly chosen from $1, ..., D$ and the L consecutive components are selected from the mutant vector. The crossover probability determines the number of exchanged components for obtaining the trial vector, calculated by

$$u_{i,G} = \begin{cases} v_{i.G} & \forall j = \prec l \succ_D, \prec l + 1 \succ_D, ..., \prec l + L - 1 \succ_D \\ x_{i,G} & otherwise \end{cases} \tag{2.8}$$

where l is the starting position and the brackets $\prec l \succ_D$ denotes a modulo function with modulus D.

C. Selection

In the selection stage, a greedy selection is used to determine the vector, either target or trial vector, to survive for the next generation using the fitness value (i.e., objective function value); for instance, in a minimization problem, the candidate in a population with the minimum objective function value is selected to be one of the members in the next generation as

$$x_{i,G+1} = \begin{cases} u_{i.G} & if(f(u_{i,G}) \prec f(x_{i,G})) \\ x_{i,G} & otherwise \end{cases} \tag{2.9}$$

A simple cycle of mutation, crossover and selection is repeated in subsequent generations until the termination conditions are satisfied. Algorithm 2.1 summarizes the DE steps used in this thesis to demonstrate the main procedures of DE for any optimization problem.

Algorithm 2.1 DE

1: Set initial control parameters F , CR and PS
2: Initialize randomly a population of PS individuals, called *target_vectors*
3: **while** (!*stoppingcriteria*) **do**
4: **for** ($i = 1$ to PS) **do**
5: Evaluate each individual in the population using the objective function $f(x)$
6: Apply Mutation strategy to produce *mutant_vector* using equation 2.1
7: Create a new *trial_vector* using the crossover operator using equation 2.7
8: **if** ($f(trial\_vector) < f(target\_vector)$) **then**
9: Maintain *trial_vector* in the next generation
10: **end if**
11: **end for**
12: **end while**

2.1.2. Significance of DE parameters

All the procedures in a DE algorithm have been used for different real-life problems [7]. Although the mechanism of DE appears to be simple, it has some limitations which affect its performance, such as stagnation, premature convergence and vulnerable CPs [8]. Stagnation is an undesirable situation in which a population-based algorithm does not converge to a better solution, even a sub-optimal one, while the population's diversity is still high [20]. In other words, stagnation occurs in DE when the population cannot be improved over a prolonged number of generations and is incapable of determining a new search space to find the desired candidate solution [22]. Several factors cause stagnation, including bad choices of DE's CPs and a problem's dimensions [20, 23].

Premature convergence is a situation whereby the population converges to a local optima due to a loss of diversity, that is, when no more offspring solutions better than the parents can be generated [24]. Convergence can be categorized as good, premature and slow, with good achieved when the global optimum is obtained in an acceptable amount of generations and there is a good trade-off between exploration and exploitation [25].

The sensitivity or insensitivity of its CPs is considered a significant drawback of DE because it can either improve DE's performance or decrease its efficiency, respectively [11, 26]. Several studies have investigated the relationships between the aforementioned problems to determine how to adjust DE's CPs [27]. The reasons for the success of DE are discussed in [14] in which the implicit self-adaptation within DE's algorithmic structure is highlighted. During the optimization process, for each candidate solution, the search rule states that generating new offspring solutions depends on the distribution of other solutions belonging to the current population in the search space.

In DE, as solutions in the early stage of the optimization process tend to be spread out in the search space, the mutation scaling factor seems to produce

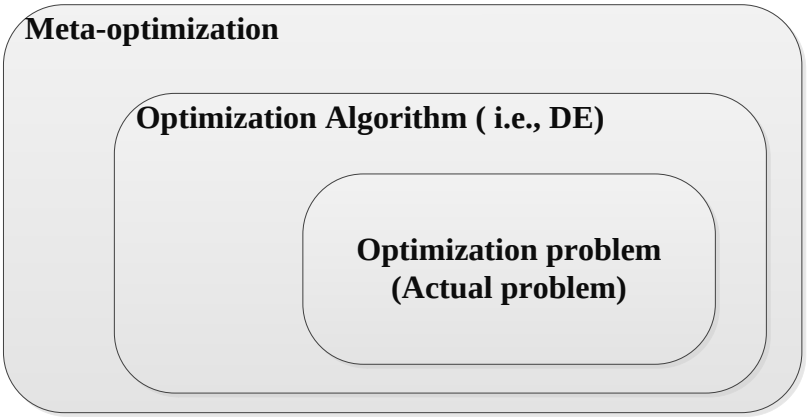


Figure 2.1: Overlaying meta-optimization conceptual model

new promising offspring ones by exploring the search space with a large step size. Then, as existing solutions of populations are concentrated in specific regions in the search space, this step size is progressively reduced. Therefore, a DE algorithmic scheme is highly explorative at the beginning of an evolutionary optimization process and successively becomes more exploitative during subsequent stages [7, 9].

The canonical DE requires very few CPs, in particular, PS, F and CR [16], each of which has a great influence on its performance in terms of effectiveness, efficiency and robustness. Determining their correct values to achieve a balance between the reliability and efficiency of running DE is a crucial task [27]. PS provides the possible moves, some of which are beneficial during the evolutionary process for reaching the optimal solution while others are not as they consume more computational resources. Therefore, suggesting a too-large PS could mislead a search in terms of convergence [23] but a too-small one could cause stagnation during the evolutionary process [28].

Several researchers [8, 17, 19] have suggested guidelines for setting the PS with different ranges of $[2D - 40D]$, where D is the number of dimensions of a particular problem. However, a high-dimensional problem cannot follow this rule as its processing time takes long. Therefore, setting the PS values depends mainly

on the problem's characteristics; for instance, separable and unimodal problems require low ones and multimodal problems high ones [9, 29].

Setting the F and CR parameters is neither a simple nor straightforward task because of its significance for guaranteeing the algorithm's effectiveness in finding the desired solution. Their optimal values depend basically on the problem's characteristics, such as linear or separable. Several studies [5, 30, 45] have reported that using $F = 1$ is not recommended as it leads to low exploration in the search space. Likewise, $CR = 1$ is not preferred because it dramatically decreases the number of offspring solutions generated [28]. Storn and Price (1997) and Liu and Lampinen (2002a) suggested settings of $F \in [0.5, 1]$ and $CR \in [0.8, 1]$ whereas, after conducting some experimental analyses, Zielinski et al. (2006) showed that $F \geq 0.6$ and $CR \geq 0.6$ provide better solutions thereby indicating that powerful parameter settings are required for different optimization problems.

The aforementioned studies reveal that the 'no free lunch' theory can be applied to different DE schemes. The F and CR parameters considerably affect the convergence speed and robustness of the search space as F impacts directly on diversity in the search space and CR provides the means for successively undertaking exploitation.

In the context of randomizing the scaling factor of DE, two new terms, jitter and dither, are defined in [14]. The former represents the procedures of individual parameters whereby different F values are generated and associated with their corresponding indices. Although jitter is not rotationally invariant, it seems to be effective for non-deceptive objective functions which possess strong global gradient information [14]. On the other hand, dither represents a situation in which a F value is created for each individual and assigned to its corresponding index, with each characteristic of the same individual evolved using the same scaling factor [8]. Although dither is rotationally invariant, when the level of variation is very small, the rotation has only a small influence. The application of these principles (i.e., dither and jitter) is used in multiple studies, for example, in [30], F is generated

for each individual in the range of $[0.4, 1]$ and CR chosen from the interval $[0.5, 0.7]$ which is fixed for each iteration. Nevertheless, setting these CPs is still a crucial problem and an active area of research.

2.1.3. Classification of parameters control strategies

Since DE's inception, it has been claimed that the task of selecting promising CPs (i.e., behavioural ones) and search operators (i.e., mutation and selection strategies) for diverse optimization problems is not simple [31]. The general idea is to have a black-box optimization method act as an overlaying meta-optimizer [26] to discover the effective parameters for an optimization technique to enhance some problems, as depicted in Figure 2.1.

Parameter tuning is divided into three layers [32], application, optimization and configuration, which are incorporated together to tackle an optimization problem. However, as shown in Figures 2.1 and 2.2, we divide the entire search scheme into two components, a main optimizer and meta-optimizer (tuner), to clarify the main steps in adjusting the DE's CPs. In Figure 2.2, the DE acts as the optimization algorithm for locating the best/ optimal solution for the problem obtained from the application layer while the other component (meta-optimizer or tuner) operates as a behavioral modifier; it acts as a modifier for the optimization algorithm configurations and parameter setting, achieving the best outcome in the optimization algorithm. On other words, the optimizer component evolves a problem in the application layer and DE in the optimizer one to determine an optimal solution. Conversely, the meta-optimizer component attempts to determine the optimal choices of the DE's parameters and configuration layer that can be used in the configuration stage.

Of the variety of factors that influence DE, setting its CP process is not a simple task. Based on the 'how' stages of parameter-setting approaches, there are two classes: parameter tuning; and parameter control [20]. The former searches for

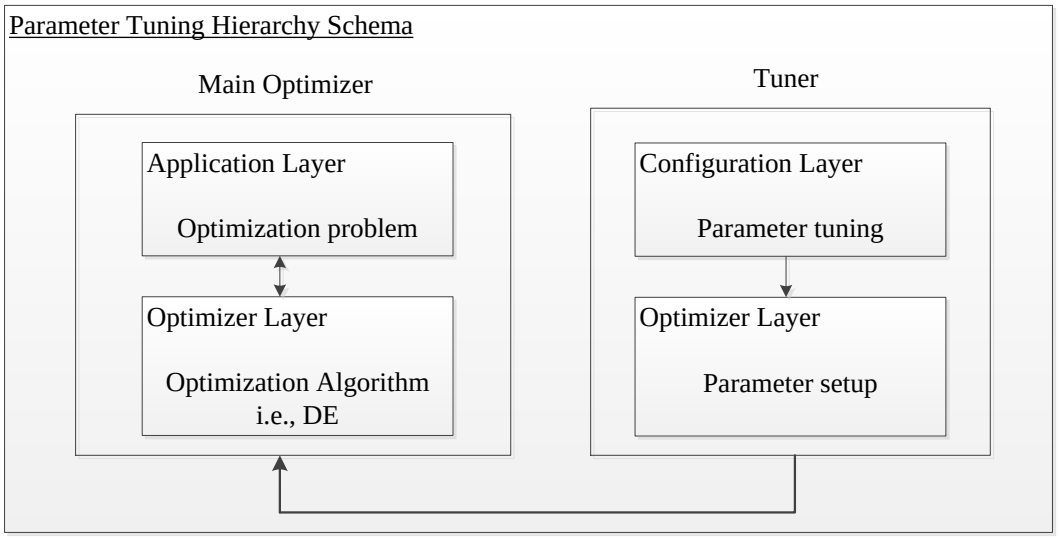


Figure 2.2: Hierarchy of parameter control schema

acceptable values before running the algorithm, with its main drawbacks related to: (a) the impossibility of trying all possible combinations; (b) its time-consuming process; (c) the parameter values selected for a given problem not necessarily being optimal even if the effort put into setting them is significant; and (d) using adaptive processes and rigid parameters is contrary to this idea because EAs are dynamic [11].

In parameter control, the CP values change during runs [33] according to some defined rules based on four aspects: which parameters change; how the control mechanisms are made; the scope of change at the population, individual and sub-individual levels; and evidence of which points change [34].

To tackle the problem of choosing the CPs, Eiben and Schut (2007) classified parameter control methods into four classes based on the 'how' criterion: (a) deterministic; (b) adaptive; (c) self-adaptive; and (d) hybrid. However, Takahama and Sakai (2012) categorized CP methods as: (a) deriving the parameter values based on observation; and (b) performing more adjustments for successful cases. A new taxonomy for organizing parameter control mechanisms [35] applied to only the F and CR parameters is given by: (a) their numbers, i.e., continuous

or discrete; (b) the number of them used per generation; and (c) the source of reflected information.

Based on the previous discussion, it is difficult to classify DE parameter control methods using a unified and well-known taxonomy. In this thesis, we focus on approaches involving 'how' criterion classes because our proposed methodologies in the next chapters follow this type of classification and categorize them as three major types: (a) **deterministic**; (b) **adaptive**; and (c) **self-adaptive**, as explained below.

A. Deterministic control methods

In deterministic control methods, the CPs are altered using pre-defined deterministic rules for assigning new parameter settings regardless of receiving any feedback from the system, such as a fixed schedule and time-dependent change in the mutation rates [14, 36]. Although replacing fixed CPs (i.e., parameter tuning) using stochastic methods could help researchers easily tune the dynamic behaviour of any optimiser, their settings should be chosen carefully at the beginning of DE processing which could require a large number of samples [25]; for example, the MDDE algorithm [37] initialises the parameters with relatively large F_0 and CR_0 values to prevent premature convergence and then monotonically decreases these values over generations (g) in a geometric sequence according to

$$F_g := F_0.exp(-a_0\frac{g}{g_{max}}) \tag{2.10}$$

$$CR_g := CR_0.exp(-a_1\frac{g}{g_{max}}) \tag{2.11}$$

: g_{max} is the maximum number of generations. Another example is L-SHADE [38] in which the PS linearly decreases during the evolution process.

B. Adaptive control methods

In adaptive control methods, the CPs are dynamically updated to incorporate some feedback from the search procedure for adjusting them and determining the directions and/or magnitudes of their changes [33]. This leads to their adaptations being enhanced during running of the optimization process to achieve a reasonable balance between exploration and exploitation in the search space [9]. Feoktistov (2006) designed two paradigms for an adaptive control method. The first refreshes a population-based approach which either changes bad individuals or pushes new ones inside the population. These procedures are repeated periodically until the diversity level reaches the stopping point to rebuild the population's diversity. It is expected that it improves the exploration of new parts in the search space and increases the convergence ratio. The second is a parameter adaptation-based approach which follows the state of the population [7].

C. Self-adaptive control methods

The parameters in self-adaptive control methods are directly encoded into the algorithm itself and follow the co-evolutionary perception for selecting the CPs of DE which is effective for decomposing the structure of a complex problem to improve its performance [14]. In other words, the algorithm reconfigures by evolving the parameters with encoded candidate solutions and combines the decision variables, with the better fitness values surviving so that the feedback from the search process is fully utilized. Although this assists users to perform any intractable task [39, 40], the convergence of such an approach is not guaranteed because the DE norm in which it operates is random-based [12].

2.2. Recent advances of automatic approaches for DE parameter control

Deterministic parameter control strategies are usually conducted through trial and error (empirical) processes which require long processing times with no guarantee of achieving the best choice for the problem at hand. Therefore, investigating adaptive and self-adaptive methods is necessary because they usually outperform classical DE algorithms (i.e., DE without parameter control), as measured in terms of their reliability and convergence rates for solving optimization problems. Technically speaking, in this thesis, we try to modify the values of the parameters of DE with respect to the actual search progress while running the algorithm. As previously discussed, there are two ways of achieving this. Firstly, some heuristic rules which obtain feedback from the current state of the search are used and then the parameter values subsequently modified (adaptive parameter control), for example, as in JADE [42]. Secondly, it is better to incorporate parameters with the candidate solutions subject to evolution (self-adaptive parameter control), as in CoDE [48].

As previously discussed, it is very difficult to classify adaptive and self-adaptive DE parameter control methods using any well-known taxonomy because some of them combine multiple mechanisms that have different procedures. Consequently, we discuss these methods according to their internal adaptation processes shown in Figure 2.3 as follows: **(a) DE adaptation methods with multiple strategies and control parameter settings, (b) DE with only control parameter (F and CR) adaptation, (c) DE that introduce new offspring strategy, (d) DE with PS control.**

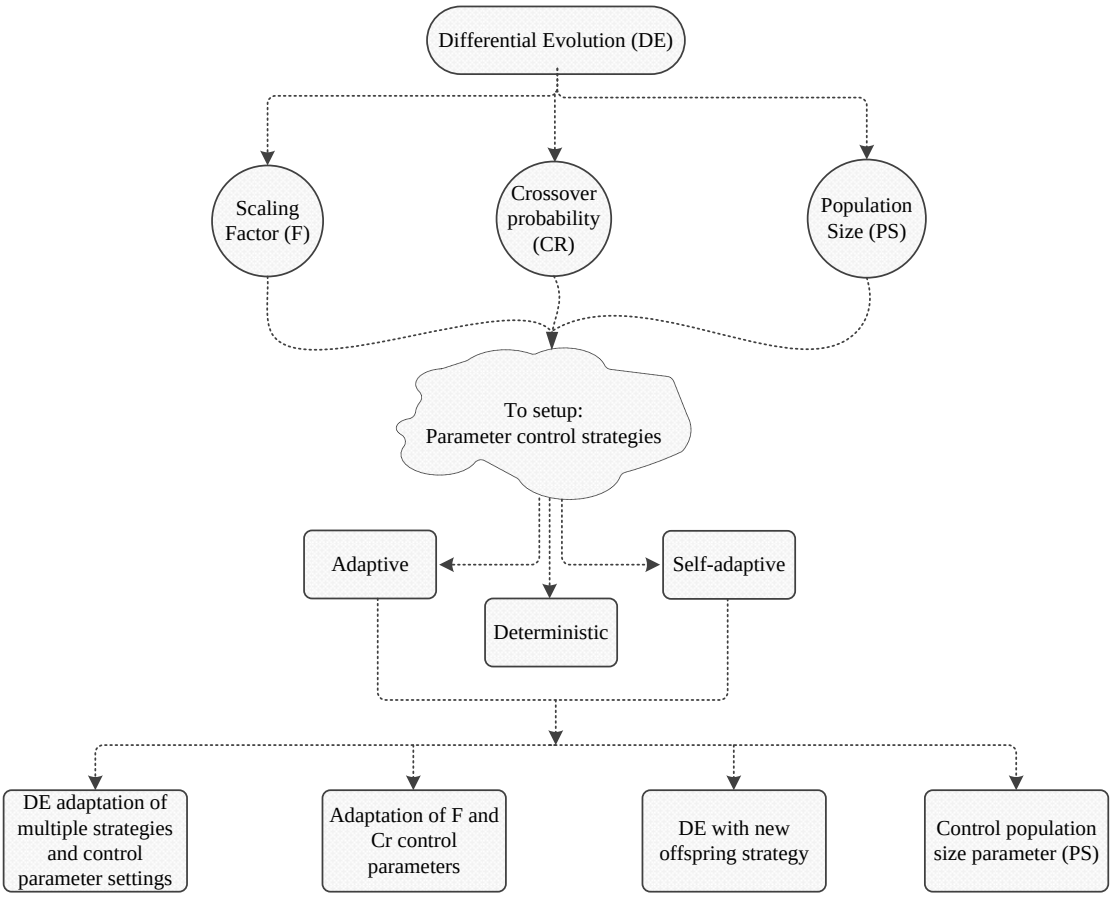


Figure 2.3: Classification of parameter control methods

2.2.1. DE adaptation methods with multiple strategies and control parameter settings

Rather than using a particular strategy during the evolutionary process, DE methods can choose one from a pool of suggested ones incorporated with some specific rules for adapting their F and CR.

To calibrate a DE algorithm's evolution using a pool of candidate strategies with different CP settings, a plethora of research studies have been undertaken over the last few decades; for example, Tang et al. [41] included an individual-dependent mechanism with a new DE, called IDE, which uses the information of fitness differences to rank the population members and balance exploration

and exploitation over runs. In this method, a collection of parameter values of the mutation and crossover operators (F and CR) and four distinct mutation strategies are used. It defines individuals based on their fitness ranks as inferior or superior and then determines the appropriate CP values and mutation strategy to be assigned for the process which is a very competitive approach. Another DE based on an ensemble of parameters and mutation strategies [29] called EPSDE defines a pool of three mutation strategies that ensure diverse characteristics (i.e., DE/best/2/bin, DE/rand/2/bin and DE/current-to-rand/1) with respect to the crossover operator (i.e., binomial or exponential). It is worth noting that the DE/current-to-rand/1 procedures use the special type of mutated recombination,

$$v_{i,G} = x_{i,G} + K * (x_{r1,G} - x_{i,G}) + F * (x_{r1,G} - x_{r2,G}) \tag{2.12}$$

where K is an additional combination coefficient. Also, the F and CR are defined within specific ranges of $[0.4, 0.9]$ and $[0.1, 0.9]$, respectively, both with step sizes of 0.1. It selects certain values from the chosen collection for each individual during the search process, particularly at the beginning (i.e., the initialization phase) when each population candidate is assigned a mutation scheme and values of F and CR for generating a trial vector, and then either the trial or target vector is chosen as the surviving one. If the trial vector is replaced by the target one, it follows the successful strategy in the following generation, otherwise the target vector can either select a new strategy or retain the old one with the same probability.

Using the diverse characteristics of mutation strategies that exhibit distinct performance characteristics during different stages in the evolution of a particular problem does not guarantee that the performance is reasonable while the evolution progresses using the combination of a mutation strategy and CPs . In EPSDE, the interactions between the intrinsic CPs are not elaborated and it is observed that its different combinations of parameter settings for each test function cannot achieve

the necessary efficiency to converge to the global optima for different test functions, especially multi-modal and scalable ones, compared with state-of-the-art methods over 10, 30 and 50 dimensions. SaDE [42] was proposed in order to incorporate two additional mutation strategies from different learning ones, that is, 'rand/2/bin' and 'current to rand/1/bin' because the first could provide more exploration and the other be effective for rotated problems, as represented in equations 2.2 and 2.3, respectively. To generate a trial vector, trial learning strategies with the two parameters (F and CR) are probabilistically assigned to each target vector in each generation using the corresponding probabilities gradually learned from previous experiences of generating promising solutions.

In SaDE, as the crossover probability is assumed to be more sensitive to a problem's characteristics [21, 42], it is accumulated from previous learning experiences over generations and randomly generated according to an independent normal distribution with a mean CR_m and standard deviation of 0.1. To properly adapt CR, the CR_m is updated based on the successful CR values recorded. Although SaDE is considered more effective and converges faster than other peer algorithms tested, it seems that it converges much more slowly for some functions (i.e., the shifted sphere, shifted Ackley, shifted rotated Ackley, shifted Griewank's, shifted Rastrigin's and other composition ones). Also, it does not successfully optimize the shifted Schwefel's function with noise, and the shifted rotated Rastrigin's and composition Ackley functions for high-dimensional problems.

Wang et al. [43] developed a new adaptive DE algorithm, called composite DE (CoDE), the main principle of which is to randomly combine three well-studied learning strategies with three parameter settings in each generation and generate a trial vector according to its function scheme. In order to generate trial vector candidates with different characteristics, the three strategies, DE/rand/1/bin, DE/rand/2/bin and DE/current-to-rand/1, and three popular parameter settings ($F = 1.0, Cr = 0.1; F = 1.0, Cr = 0.9; F = 0.8, Cr = 0.2$) chosen have distinct effects and suppose different search behaviours of the algorithm; for example,

($F = 1.0, Cr = 0.9$) provide high degrees of variation and perturbation to the donor and parent, respectively, with most components of the trial vector coming from the donor one which results in exploration of the search space while ($F = 0.8, Cr = 0.2$) do the opposite, that is, facilitate exploitation of the space around the target (parent) vector.

Wenchao et al. [44] provided a hybrid DE method (HSDE) which integrates two mutation strategies with a co-existing self-adaptive parameter control method [40]. It applies two mutation strategies (i.e., DE/rand and DE/current-to-best) to balance the exploration and exploitation strengths of DE while the F and CR are generated according to jDE [40] to effectively control them using a feedback indicator.

Ghost [45] proposed an effective adaptation strategy which aims to tune the F and CR based on their objective function values of the population members during the course of a run to run DE automatically. An executive comparison of different variants and some real-life engineering problems reveals its superiority for achieving better solutions with acceptable convergence speeds. A novel DE algorithm with an intersect mutation operator for improving the global search ability, called IMDE, is proposed in [56]. In it, the population is categorized into two groups, superior and inferior individuals, according to their fitness values, with different mutation strategies playing different roles in different groups. The experiment results showed that, empirically, this algorithm is relatively better than jDE [40] and other state-of-the-art EAs.

DMPSADE was implemented by Fan and Yan [46] to act for self-adaptive parameter control and mutation strategies. In it, each variable of an individual has its own mutation parameter and each individual is evolved using a different crossover parameter and one of five mutation strategies, i.e., DE/rand/1, DE/rand-to-best/2, DE/rand-to-best/1, DE/best/2 and DE/rand/2. The F and CR are generated by a normal distribution function and, based on their performance rankings, a successful combination for the next generation is found. In

the experiments, more than 25 test functions with different dimensions were assessed and compared with other DE variants, with the statistical results implying that DMPSADE produces an overall outstanding performance in terms of solution quality.

Elsayed et al. [47] introduced a new DE variant algorithm called SAMODE which uses different mutation strategies with distinct characteristics. It selects one during the evolutionary process from a pool of suggested mutation operators, each with an equal chance of being selected during the search stages. It divides the population into four parts, each of which executes one mutation strategy on its individuals. Recently, based on a/the multi-operator framework, Elsayed et al. [48] presented a self-adaptive parameter control technique called UMOEA. In it, the population is clustered into three sub-populations of similar sizes to each of which a multi-operator algorithm is independently applied. The success rates for some fixed numbers of generations are recorded to determine the best multi-operator for the next generations. The UMOEA demonstrates great performances for different testing functions with less time complexity than other algorithms as it uses the three operators for only half the number of maximum evaluations rather than the whole evolutionary process.

Another adaptation of a parameter candidate pool called TSED is introduced in [59]. As the evolution progresses, the process is split into two stages according to the number of fitness evaluations whereby different mutation strategies are implemented with corresponding settings to obtain balance between exploration and exploitation in different stages.

More recently, Rammohan et al. [49] proposed a multi-population DE variant with an ensemble of learning strategies called MPEDE. In it, the entire population of size NP_w is divided in each generation into four equal parts, three indicator sub-populations of the same size ($NP1$, $NP2\,and\,NP3$) and one relatively large reward one. Three distinct mutation strategies are assigned for each indicator sub-population and a constituent mutation for the reward one. Subsequently,

while the search proceeds, the best-performing mutation strategy captured will consume more computational resources in the following generations. The mutation performance can be assessed by the ratio of the fitness improvements to function evaluations consumed as

$$p_{m,i} = \frac{\Delta f_j}{\Delta FES_j} \tag{2.13}$$

2.2.2. Adaptation of F and CR DE control parameters

Many studies [19, 32, 34] have been conducted to individually adapt DE's CPs; for instance, Liu and Lampinen [50] developed a new DE called a fuzzy adaptive DE (FADE) based on the fuzzy logic technique in which a fuzzy logic controller (FLC) applies IF-THEN rules based on previous knowledge to gain broader knowledge [50]. The F and CR are adapted using FLCs, the inputs to which are provided with their function values. These parameters reflect the population's information (i.e., their parameter vectors and function values, and changes during the search process). Their results outperform those from the classical DE for higher-dimensional problems.

Some optimization algorithms rely on probabilistic models, such as covariance matrix adaptation (CMA) [51, 52], that maximize the reproduction growth in a search's path. Wang et al. [52] suggested a new approach called CoBiDE which dynamically adapts the F and CR using the covariance matrix's learning and binomial distribution parameters to achieve equilibrium between the exploration and exploitation of DE. A covariance matrix is used in CoBiDE to implement a coordinate system for the crossover operator with the binomial distribution including the two Cauchy distributions

$$F_G = \begin{cases} rand_c(0.65, 0.1) & if rand(0,1) \leq 0.5 \\ rand_c(1.0, 0.1) & otherwise \end{cases} \tag{2.14}$$

$$CR_G = \begin{cases} rand_c(1.0, 0.1) & if rand(0,1) \leq 0.5 \\ rand_c(0.95, 0.1) & otherwise \end{cases} \tag{2.15}$$

where $rand$ is a random uniform number$\in [0, 1]$. Overall, the performances of CoBiDE is effective because it can compute the central parameters precisely.

A self-adaptive DE for estimating the CPs is the jDE technique designed by Brest et al. [12] in which the F and CR values are encoded into individuals and initialised using $F = 0.5$ and $CR = 0.9$ for each individual. Then, they are reproduced using uniform distributions of $[0.1, 1]$ and $[0, 1]$, respectively, with respect to the probabilities$\tau_1 = \tau_2 =$0.1. New F and CR values generated over generations are calculated by

$$F_{G+1} = \begin{cases} F_l + rand_1.F_u & if rand_2 \leq \tau_1 \\ F_{i,G} & otherwise \end{cases} \tag{2.16}$$

$$CR_{G+1} = \begin{cases} rand_3 & if rand_4 \leq \tau_2 \\ CR_{i,G} & otherwise \end{cases} \tag{2.17}$$

where $rand1, rand2, rand3$ and $rand4$ are random numbers, which follow a uniform distribution of $[0, 1]$, whereas F_l and F_u are initialized by 0.1 and 0.9, respectively with the range of F values $[0.1, 1]$, and $\tau_1 = \tau_2$=0.1. In [53], another adaptive approach for computing the F and CR based on an exponential weighted moving average called EWMA-DECrF is proposed. The results of this study show reasonable performances compared with those of some variants. Sarker et al. [54] designed a dynamic selection method for choosing the desired combination of parameter settings, which ranks them in subsequent generations. Recently, Corriveau

et al. [55] developed a genetic adaptive method based on the Bayesian network (BN) called BNGA, where the BN represents a graphical model for optimising the relationships between the GA parameters which are updated depending on a new reward emanating from the entire DE process.

Tanabe and Fukunaga [56] presented an improved version of the JADE algorithm [42] called SHADE, both of which update the F and CR values based on the probability distribution of the successfully yielded offspring. However, SHADE samples the F and CR using the historical archives of recent generations which enhances its performances, to analyze which it was tested using the IEEE CEC2013 competition and ranked third of 21 algorithms. Tanabe and Alex [38] further extended the SHADE algorithm in a new version called L-SHADE in which the population size (PS) linearly decreases during the search progress. It can automatically adjust the F, CR values based on their success-history adaptations. The experimental results show that it outperforms SHADE and other peer variants but requires more space resources to maintain three archives for the solutions and CPs.

Elsayed et al. [57] proposed an improved algorithm called DE-APC which finds the most appropriate parameter settings for the F and CR. Two sets of the F and CR are defined, with each individual in the population generating a random combination of their values. For a specified number of iterations, the ranking of each combination is calculated to decide which will remain for the next generations. When applied to a set of problems, this algorithm demonstrated its high-quality solutions.

Yu et al. [58] presented an individual-dependent parameter control adaptation method called ADE which has two main components. The first is implemented in a new mutation strategy (DE/lbest/1) which is a variant of DE/best/. It guides the search toward multiple locally best individuals instead of only one global best in order to obtain a balance between population diversity and fast convergence during the search process. The second contains an adaptation function involving two main

steps. The first adapts the F , CR parameters at the population level according to the nature of the population (i.e., optimization states) while, in the second, an individual's F and CR values are adjusted by the population-level parameters in which the individual's fitness values and the distance of any individual from the global best are computed. In this approach, the parameters can estimate both the evolution's state (i.e., explorative or exploitative) and the individuals' characteristics, with the state of the population of a generation estimated using an indicator of the current evolution's state as

$$IOS = \sum_{i=1}^{PS} |f_i - d_i| \tag{2.18}$$

where the individuals are first sorted based on their fitness values according to their (f) fitness ranks, and then the distance of the best-fit individual with a d_i distance. An IOS refers to the exploration and exploitation states being high and low, respectively. The F, CR values at the population level are estimated using the evolution state as

$$F_p^t = \begin{cases} F_p^{t-1} + c_p \Delta F_p & if(population : explorative) \\ F_p^{t-1} - c_p \Delta F_p & if(population : exploitative) \end{cases} \tag{2.19}$$

$$CR_p^t = \begin{cases} CR_p^{t-1} + c_p \Delta CR_p & if(population : explorative) \\ CR_p^{t-1} - c_p \Delta CR_p & if(population : exploitative) \end{cases} \tag{2.20}$$

$$\Delta F_p, \Delta CR_p = \begin{cases} \frac{IOS - IOS_{min}}{IOS_{max} - IOS_{min}} & if(population : explorative) \\ \frac{IOS_{max} - IOS}{IOS_{max} - IOS_{min}} & if(population : exploitative) \end{cases} \tag{2.21}$$

where c_F=0.1 and c_{CR}=0.05 are used for estimating the F, CR. This algorithm is competitive with peer ones although it requires several computations which lead to it being complex for handling high-dimensional problems.

In a new self-adaptive DE introduced by Mallipeddi et al. [59], a Guassian Adaptation (GaA) is applied to adjust the F, CR parameters. This is a stochastic process based on maximizing the entropy (H) of a multivariate Gaussian distribution with its mean (m) and variance information (C) as

$$H = log(\sqrt{(2\pi e)^D det(C)}) \tag{2.22}$$

The GaA has a threshold for the fitness-dependent acceptancec_T, which is reduced until the convergence criteria are satisfied. This algorithm was tested on 25 problems with different dimensions and, although it showed remarkable performances in terms of solution quality, its complexity increases with increases in its algorithmic dimensions.

2.2.3. DE methods with new offspring strategy

Researchers have tried to develop new learning strategies, including different forms of decision vectors, to improve the search procedure; for instance, Zhang et al. [60] used a new mutation strategy, 'DE/current-to-pbest' with an optional archive. It is the basis of the adaptive DE algorithm for adapting the F, CR called JADE in which the strategy can interact with or without the archive whereby the mutation vectors are generated, respectively, as

$$v_{i,G} = x_{i,G} + F * (x_{pbest,G} - x_{i,G}) + +F * (x_{r1,G} - x_{r2,G}) \tag{2.23}$$

$$v_{i,G} = x_{i,G} + F * (x_{pbest,G} - x_{i,G}) + +F * (x_{r1,G} - \tilde{x}_{r2,G}) \tag{2.24}$$

where $x_{pbest,G}$ is randomly chosen from the top $100p$ of the current individuals in the population with $p \in [0, 1]$ as well as F is the scaling factor, which is updated per generation in the adaptive manner $x_{i,G}$, $x_{r1,G}$and $x_{r2,G}$ are randomly chosen

from P, but $\tilde{x}_{r2,G}$obtained from the union of the current population and the archive in equation 2.24.

This parameter adaptation is performed automatically and does not require any prior information [43]. The F, CR values are sampled using the Cauchy and normal distributions of their respective mean values which are updated by the individual ones successfully generated as better trial vectors than the target ones. JADE is considered a competitive DE variant because it incorporates the best solution information in the evolutionary process and maintains a high level of reliability. However, despite the benefits of incorporating this information, it could/-can cause problems, such as premature convergence due to the resultant reduced population diversity. Moreover, it incurs a higher computational cost than other DE variants.

Wang et al. [61] used the cumulative distribution information of a population, which implements an Eigen coordinate system to the crossover operator, called CPI-DE. This operator is executed for the original and Eigen coordinate systems, with the better offspring selected for the next generations.

Wenyin and Cai [62] developed a different strategy for the mutation operator, claiming that it is better for the difference vectors used in mutation to be selected based on their fitness rankings instead of randomly selected from the population. This proposed mutation enhances the original DE's exploitation capability with no increase in its complexity. The population is ascendingly sorted according to the individuals' fitness values and then the ranking measure for computing the decision vector (x_i) is calculated as

$$R_i = PS - i \tag{2.25}$$

In order to select the new vectors for mutation, the selection probability (p_i) is estimated as the ratio of the ranking measure to the PS as

$$P_i = \frac{R_i}{PS} \tag{2.26}$$

DE was improved by Zhou et al. [63] who modified the mutation and crossover schemes for generating new vectors. This method, IMDE, improves the search proficiency with two main processes employed for the next generation's population. In each generation, the whole population is sorted ascendingly and separated into two portions, better and worse solutions, by applying two different variants. In more detail, the better part in the first variant uses the DE/rand/1 mutation strategy with the worst part and other two difference vectors chosen from the better part. While the second variant uses a new mutation strategy (i.e., DE/ current-to-best/bin/1), its vectors are selected according to the part to which it applies. The first and last vectors are chosen from the worst part and the others from the best one. The main issue of this algorithms is that, to determine the proportions for the global search capability, the additional parameters have to be carefully chosen.

To speed up the convergence of DE and escape the local optimum, Wen et al. [64] proposed new learning strategies for mutation inspired by the 2-Opt algorithm [90] using the two mutation schemes DE/2-opt/1 and DE/2-opt/2 rather than DE/rand/1 and DE/rand/2, respectively. The original 2-Opt algorithm attempts to find routes in the travelling salesperson problem, such as, in [90], using self-crossing which offers a powerful opportunity to avoid local optima. Therefore, it replaces the DE/rand/1 and DE/rand/2 strategies by exchanging the new individuals in the population while the mutation operates. It is worth mentioning that the base vectors in these two new mutation schemes always have better objective values than the first difference vector. However, although they achieve good performances, they require long processing times to ensure the superiority of the selected vectors.

Wang et al. [65] proposed a DE framework called OXDE which uses an orthogonal crossover operation instead of a binomial or exponential one to generate enhanced solutions, with this crossover adjusted using a quantisation methodology (QOX) [91]. In it, the QXD is combined with DE in each generation to obtain the advantages of both techniques but it consumes high amounts of computational resources while running.

Several studies [19, 65, 66] have tried to enhance the performance of DE by adapting its crossover operator. The binomial crossover is widely used in different DE variants because of its capability to handle rotated and invariant function landscapes in order to provide better trial vectors. Yong et al. [52] replaced the standard co-ordinate system with an appropriate one using covariance matrix learning based on the distribution of the current population to generate a new trial vector. However, it degrades the performance of DE for some complex landscapes, such as multimodal, rotated problems, because establishing such a matrix requires using some of the best population vectors and to adjust its new CPs is an arduous task.

Although DE has a mutation operation that helps to explore new solutions in order to obtain optimal results, researchers have not focused on designing decent strategies for efficiently handling complex problems which requires careful analysis to choose the correct mutation vectors. However, these vectors are currently randomly selected without further analysis of the neighbourhood information which could properly guide the search process. Also, the best solutions in the population are exploited while the others are ignored whereas determining the latter in depth would improve the performance of DE. Cai and Wang [88] proposed two schemes for directing a search towards the desired solutions without exploring undesired areas using the neighbourhood and directional information of the population involved in the search procedure to relatively enhance DE's performance. Firstly, a neighbourhood-guided selection scheme selects the mutation vectors, such as probability selection operators calculated by the neighbourhood information (i.e.,

Euclidean distance measure), as the individual's probability is inversely proportional to its distance.

A roulette wheel method is applied to select three vectors to estimate their probabilities, with the base one the champion vector of the tournament. Secondly, a direction-induced mutation strategy uses the direction information to generate a new vector called a direction one which is accumulated with the other vectors obtained from the neighbourhood-guided selection scheme. Then, the nearest best and nearest worst neighbours are calculated by the maximum ratio of the fitness difference to Euclidean distance between the two vectors (x_i and x_j). Based on this information, three types of directional characteristics designed are Directional Attraction (DA), Directional Repulsion (DR) and Directional Convergence (DC) [67]. Finally, the two components, the neighbourhood-guided selection and direction-induced mutation, are combined to design the NDi-DE framework. Although this framework is effective in terms of results, its complexity is quite expensive because it consumes high amounts of resources to compute its components.

2.2.4. DE with PS control

As the PS is one of DE's CPs, it has some attraction for researchers and a few studies have concentrated on controlling it using a DE methodology; for example, Brest and Maucec [68] proposed a new mechanism called jDElscop for adapting the population with three different parameter-learning strategies which uses jDE as a baseline algorithm and involves three well-known variants of the F, CR (i.e., jDEbin, jDEexp and jDEbest). In each iteration, only one strategy is active and then an/the iterative DE procedure is applied, with the F, CR values updated in the iterations similarly to equations 2.16 and 2.17 but for different strategies as

$$F_{G+1,S} = \begin{cases} F_l^S + rand_1.F_u^S & if rand_2 \leq \tau_1 \\ F_{i,G}^S & otherwise \end{cases} \tag{2.27}$$

$$CR_{G+1,S} = \begin{cases} CR_l^S + rand_3.CR_u^S & if rand_2 \leq \tau_1 \\ CR_{i,G} & otherwise \end{cases} \tag{2.28}$$

where the lower and upper boundaries of F_l^S , F_u^S , CR_l^S , CR_u^S can be set for different strategies $S = 1, 2, 3$, for example, if $s = 1$, then the bounds of F and CR are $[0.1, 1]$ and $[0, 1]$, respectively. It offers a population reduction method with a pre-defined schedule for reducing the PS whereby the population is divided into some parts with some strategies applied to each one. In this way, individuals can compete with their corresponding offspring in the same strategy period which helps to avoid stagnation during the search process. The computational budget of this mechanism is expensive and increases considerably for high-dimensional problems.

Later, Brest et al. [69] improved the jDE using two mutation strategies combined with a population reduction mechanism, which are dependent on the PS that reduces with increases in the number of function evaluations. The results obtained from assessing this algorithm using a toolkit of CEC2011 proved that its performance is reasonable compared with those of two existing algorithms. In [70], a population reduction mechanism based on jDE with two learning strategies and a structured PS reduction procedure is called SPSRDEMMS.

Yang et al. [71] introduced a population adaptation technique in which the PS is regenerated based on the loss of diversity whereby, if there is poor population diversity or stagnation occurs, it is adapted using the Euclidean distance measure to increase diversity and avoid stagnation. This technique is capable of identifying the loss of diversity or stagnation time using an indicator instead of monotonically reducing it. However, computing the Euclidean distances many times is computationally expensive. Gonuguntla et al. [72] extended the technique in [59] whereby the Gaussian adaptation technique is accumulated for population adaptation. It samples a huge number of solutions and then, for each generation, a fixed number of individuals is selected from the large set either randomly or based on the

distribution of the objective values using/in the current evolution.

Zhu et al. [73] proposed an approach for dynamically adjusting the PS via the search process in a specific range. In each iteration, it monitors the status of the search space to keep track of an individual's progress to determine whether the current/concurrent PS needs to increase or decrease; for instance, if no better solution is detected in the next generations, the new population is provided with new individuals. However, if the DE finds one or two better solutions, the redundant solutions in the population should be eliminated to remove redundancy and reduce the computational load. If the population's individuals stagnate for some generations at either the lower or upper search boundary, the population is updated in accordance with this boundary. In order to decrease the PS, the iterative solutions are sorted ascendingly and their rankings calculated as

$$rank_i = quo(f_i - f_{min}, \frac{f_{max} - f_{min}}{\omega}) \tag{2.29}$$

where *quo* is calculated by considering the floor by dividing $f_i - f_{min}$, which represents the difference between the fitness of the solution i and the lower fitness boundary, and $\frac{f_{max}-f_{min}}{\omega}$, which refers to the interval length. The ranking of the ith solution should belong to $[0, \omega]$, signifying its location in the objective space, in which $\omega = \lfloor 0.8 * PS \rfloor$. For each individual, the ranking of solutions is transformed into being probability-based and calculated based on their ranks to represent their locations with $[0, 1]$ values instead of [0, ω]. Some of elite individuals are selected and new solutions are generated by perturbing them to increase the population size and avoid the stagnation as well as premature convergence. However, this approach requires other new parameters to detect the status or generate new individuals.

Teng et al. [74] proposed a self-adaptive PS methodology based on two encoding approaches, absolute and relative, namely DE-Abs and DE-Rel, respectively. The former sets the PSs of the next iterations as the average of that of the current

one while the latter uses the growth rate of the population rather than its absolute value.

A novel self-adaptive technique for dynamically controlling the PS parameter without setting it a-priori, namely DEAPS, was proposed by Leung et al. [75]. In it, the main objective is to preserve a/the proportion of the centre of the population in relation to the standard deviation of the whole population as constant. More precisely, if the proportion is greater or less than a pre-defined threshold, the PS increases or decreases, respectively.

2.3. Constrained Optimization

COPs often appear in real-world problems as either minimization or maximization problems subject to a set of constraints which have to be satisfied to obtain any feasible solution. The mathematical process of a COP is as follows.

$$\begin{aligned} Minimise \quad & f \quad (\overrightarrow{x}) \\ subject \quad & to: \\ L_d \leq x_d \leq U_d \quad & , \quad d = 1, 2, ...D \\ g_i(\overrightarrow{x}) \leq 0 \quad & , \quad i = 1, 2, ...IQ \\ h_j(\overrightarrow{x}) = 0 \quad & , \quad j = 1, 2, ...EQ \end{aligned} \tag{2.30}$$

where $f(x)$ is the objective function to be minimized and $\overrightarrow{x} = [x_0, x_1, ...x_D]$ the dimensional vector of D decision variables. Each vector should be defined within the lower and upper boundaries (L_d and U_d, respectively) with $g_i(\overrightarrow{x})$ and $h_j(\overrightarrow{x})$ the i^{th} inequality and j^{th} equality constraints, respectively. Any solution in the search space can be defined as feasible or infeasible according to its level of constraint satisfaction. If $\overrightarrow{x}$ can satisfy all the equality and inequality constraints

within the defined boundaries, it is called a feasible solution, otherwise an infeasible one. It is necessary to point out that, as EAs were originally used to solve unconstrained problems, COPs need additional techniques for handling their constraints. Therefore, different Constraint Handling Techniques (CHTs) have been combined with optimization algorithms to solve them.

A COPs' solution is considered an optimal one if two main characteristics are achieved, that is, it is feasible and has the minimum fitness value (i.e., objective function value). COPs can be categorized in different classes according to their properties, such as their functions could be unimodal or multimodal, discrete or continuous and the data types of their decision variables real or integer while the feasible area can be a small or large portion of the whole search space [4].

It is worth noting that all these characteristics influence the choice of optimization methods for solving COPs to simultaneously achieve feasibility and optimality [76]. Also, any problem without an EQ or IQ problem is considered an unconstrained optimization one that achieves optimality but not feasibility.

2.3.1. Constrained Handling Techniques (CHTs)

EAs have proven their success for solving unconstrained optimization problems, with mainly meta-heuristic algorithms used during the last few decades. However, as the other constrained functions involved in COPS (i.e., equality or inequality) lead to more difficult optimization problems, it is necessary to adopt some approaches for solving them, with CHTs having been incorporated with EAs to deal with constraints [76] (both equality and inequality) with the fitness function.

CHTs can be categorized as different basic techniques [77], with the most popular: (a) penalty functions; (b) distinction of objective functions and constraints; (c) stochastic ranking; (d) the ε-constraint method; and (c) multi-objective concepts. The first two are considered classical or early CHTs while the rest are currently used, with the most frequently used ones described below.

A. Penalty functions

This is the most common technique for handling constraints. In it, a penalty term is added to the fitness function with the aim of transforming a constrained problem into an unconstrained one based on its constraint violations. The main reason for this is to reduce the chance of obtaining infeasible solutions.

This can be accomplished by adding or subtracting the value of a current constraint violation from the objective function depending on the type of problem (i.e., minimization or maximization). Mathematically speaking, there are two kinds of penalty functions, interior and exterior. For the latter, a small penalty value is added or subtracted to the feasible points inside the feasible region while the former starts from infeasible solutions and propagates towards feasible ones [78], with the basic form of a new penalty function described as

$$newf_{penalty}(\overrightarrow{x}) = f(\overrightarrow{x}) + \left[\sum_{i=1}^{IQ} \phi_i * g_i + \sum_{j=1}^{EQ} \gamma_j * h_j\right] \tag{2.31}$$

where$\phi$and $\gamma_j$are the penalty's constant factors, $g_i$and $h_j$are the inequality and equality constraints, $f(\overrightarrow{x})$ is the original objective function. There are different approaches for using penalty functions in EAs [122, 119], as explained below.

- The **death penalty function** is the simplest and most popular method for handling COPs, where infeasible solutions are rejected from the population by assigning their fitness values to zero. However, it is only suitable and considered efficient for problems in the feasible search region which is a fairly large part of the whole search space [78] and, for a problem with high constraints, too much time is required to find a feasible solution. Moreover, exploiting the search process based on the feasible space without considering the infeasible information limits the possibility of finding better solutions [79].

- The **static penalty function** uses the same value constant as the penalty factors during the evolutionary process. Although an advantage of this approach is that its parameters do not depend on the number of generations, as it seems that it is not a good idea to generalize the penalty factors for all problems as they are problem-dependent, it is necessary to adapt a large number of parameters [79, 80].

- The **dynamic penalty function** is a type of penalty approach in which the stages in the evolutionary process that are involved as the penalty factors increase over time. In other words, for more generations, the values of these factors increase. Although many researchers have recommended using the dynamic rather than static penalty method for any arbitrary COPs [4, 81], its parameters must still be chosen carefully for good optimization.

- The **adaptive penalty function** maintains the diversity of a population. In it, the penalty factors are iteratively re-formed as the number of generations increases according to the fitness information of the best solution obtained from a set of generations. Updating these penalty parameters requires using the information of the optimal individual with the least fitness regardless of its constraints. Its main disadvantage is the way in which it chooses a threshold to guide the search towards feasibility [76, 82, 83].

- The **annealing penalty function** mimics the idea of the simulated annealing algorithm in which all the constraints involved are separated into four sets: linear equations and inequalities; and non-linear equations and inequalities. Then, it initiates a starting point for establishing an active constraint set. Its main interesting idea is its use of a cooling factor to update the penalty parameters to prevent it becoming stuck in local optima [78, 84] but it is very sensitive to the settings of these parameters' values.

- The **segregated EA** uses two penalty parameters for two different populations to balance their high and low penalties. Its main issue is that choosing these penalties takes a long processing time.

B. Distinction of objective function and constraints

To handle constraints through an evolutionary process, the search is divided into the two main steps of handling the constraints and isolating the objective function. The first aims to find feasible solutions without considering their objective values while the second attempts to obtain the best objective function value [78, 85]. Two techniques are used to handle the constraints of a COP based on the distinction between its objective function and constraints.

Firstly, the technique for determining the superiority of the feasible points [85] considers any feasible solution superior to an infeasible one and can handle them subject to the following three main rules [86].

- If there are two feasible solutions, the better objective value is chosen to survive.
- If the first solution is feasible and the second is infeasible, then the feasible is preferred.
- If the two solutions are infeasible, the solution with the least sum of constraints violations is better.

Despite the capability of this technique to direct the search towards the feasible region, it loses diversity in the population [85, 86].

Secondly, a multi-objective optimization technique based on a multi-objective optimization principle, such as Pareto dominance, population-based selection and Pareto ranking, is used [87]. In it, a constrained problem is transformed into a bi-objective one in order to handle the objective function and constraints separately, that is, each constraint is considered an additional objective function. This technique is considered simple for applying to solve a COP because there is no need for extra updating of its fitness values. However, it could require additional

parameters to handle the constraints and consumes a high amount of computational resources. It is the same idea of multi-objective optimization evolutionary algorithms, which comprise two inconsistent objectives such as the DE in feature selection (DEMOFS) [88].

C. Stochastic Ranking (SR)

SR was developed by Runarsson and Yao [89] to handle the constraints of non-linear problems. It can overcome the disadvantages of using penalty methods, particularly inappropriate selections of the penalty parameters. It uses a bubble sort based on the problem's constraint violations to rank the candidates in the population and swaps every two adjacent individuals to reduce the number of infeasible solutions reaching feasible ones until all have been swapped. In it, F_{SR} is a user-defined parameter for randomising between infeasible solutions in the selection process according to their constraint violations and objective functions [89]. Although it is simple in terms of processing, it cannot preserve the level of diversity during the evolutionary process.

D. ε-Constraint method

The ε-Constraint method method defines two levels [90]: (i) a feasibility one for the constraints of a problem called a relaxation parameter; and (ii) an ε level which defines a mechanism for lexicographic order that easily handles the equality constraints [85, 90]. A constraint violation precedes an objective value as the feasibility of any individual which is very significant as this minimises the objective function. Unlike previous approaches, the ε-method can retain the diversity of the population but its extra parameters are an issue.

2.4. Recent advances of handling constraints in DE

Because of the significance of solving COPs in the real world, CHTs have become essential supplements for improving the performances of EAs, particularly DE. Several researchers [91] have undertaken to improve DE methods to address constraint problems using different approaches: (i) **directly combining CHTs with DE**; (ii) **incorporating repaired methods in DE**; and (iii) **hybridising an ensemble of CHTs or any other EA operators and methods.**

2.4.1. Combining CHTs into DE

Over the last few decades, many studies [82, 92] have suggested solving COPs by incorporating CHTs with EAs, in particular DE, through either proposing a new CHT, new DE operator schemes or multi-objective optimisation techniques, as discussed below.

A bi-objective optimisation problem degrades the constraint violations for infeasible individuals and enhances the objective functions for feasible solutions [76]. In [84], the authors formulated a co-evolutionary dual-population DE (DPDE) algorithm in which the initial population is divided into two sub-populations, one containing infeasible solutions and the other feasible ones, with each handled separately by randomly selecting the mutation difference vectors as follows. Firstly, the base vector and terminal point of the difference vector are chosen from the target vector's sub-population, secondly, the third vector is arbitrarily selected from the entire population and, finally, information is shared amongst the difference vectors. This algorithm was evaluated on 35 test functions (CEC2006 and CEC2010 datasets) and shown to be significant compared with other algorithms. However, in some cases, this strategy could drive some feasible solutions towards infeasible regions.

A new CHT called MRS (multi-objective optimisation-based reverse strategy) [82] uses two main steps. The first, which is based on transforming a C-constraints problem into a C-objective one and then selecting the best feasible individual from the objective value of the current generation, is denoted as gbest [82]. In MRS, the objective function is a constraint and its value has to be less than or equal to gbest, with the Pareto dominance used as the selection operator in DE to determine the individuals to survive in the next generations. The experiments showed that it performs better than other algorithms except that, for equality constraints, it considers finding feasible solutions a difficult optimisation task.

In 2012, Haibo and Rangaiah[92] introduced a new CHT for handling equality and inequality constraints in an integrated DE (IDE). It uses an adaptive constraint relaxation function incorporated with a selection feasibility approach, with the constraint relaxation rate dependent on the proportion of feasibility (i.e., number of feasible individuals). Although it performs well for solving test benchmark functions and some application problems, solving the latter requires expensive computations to obtain promising results. Later, Guohua et al. [93] proposed a new strategy for reducing both the equality constraints and variables involved in COPs (ECVRS). It exploits the relationships between them using a local search technique with equality constraints and demonstrates significant results.

In addition to the new CHTs, a new DE algorithm for determining the most suitable environmental economic dispatch (EED) strategy called DEFS was developed by Pandit et al. [94]. It employs a fuzzy selection mechanism in which the fuzzy logic used in the selection phase ranks each individual solution in the population with respect to multiple objectives. It also maintains diversity while satisfying the objective constraints and demonstrates effective results for an EED problem.

Zhang and Duan [95] introduced an improved selection DE operator called mDELC. This approach is based on involving an ε-CHT with the selection operator in DE so that the global search performance of DE can be enhanced. The ε value

indicates the priority level (i.e., scaler parameter) for comparing two solutions and its value is dynamically updated when dealing with a small feasible region, where the information of the early infeasible solutions obtained is used. This proposed algorithm was compared with the original DELC and other state-of-the-art algorithms using a route-planning problem, which resulted in it being capable of determining an optimal feasible route. However, although this approach can be appropriate for solving some constraint problems, it is not suitable for all.

As the mutation operation is one of the core procedures in DE, many researchers have focused on improving its search capability; for instance, an adaptive-ranking mutation operator (ARMOR) was employed in DE [96] in 2015. In it, the solutions are adaptively ranked according to their status in the current generation and classified as feasible, infeasible and semi-feasible. Feasible solutions are treated as unconstrained problems and ranked based on their objective values, infeasible ones sorted according to their constraint violations and semi-feasible ones ranked based on transforming the fitness model. The selection probability for each solution in the current population is calculated according to its classification for generating a trial vector. The base vector and terminal point of the difference vector in the mutation strategy are randomly chosen based on the selection probability and the other vectors.

Recently, Wei et al. [97] proposed a novel constrained DE framework called MS-CDE in which the current population is sorted using the fitness and diversity information. It uses non-dominated sorting of previous information to rank the parents in the same front as that chosen for the mutation operator. Although the authors consider that its generality can be applied to other CDE variants, despite this and its simplicity, it incurs additional computational costs for its diversity and sorting calculations.

A new mutation scheme called COMDE for obtaining fitter solutions in DE was proposed [99]. In it, the difference vectors are chosen based on the global

best and worst vectors in the population with a 0.5 probability, instead of being randomly selected, as

$$
\begin{aligned}
if\,(rand[0,1] \quad &\leq \quad 0.5) \\
then \quad v_{i,G+1} \quad &= x_{i,G} + F_l * (x_{best,G} - x_{worst,G}) \\
Else \quad v_{i,G+1} \quad &= x_{r1,G} + F_g * (x_{r2,G} - x_{r3,G})
\end{aligned}
\tag{2.32}
$$

Also, the selection procedure is used to choose the trial vector in subsequent generations according to the following three criteria for handling a problem's constraints.

- If both vectors are feasible, the one with the lowest fitness value has to survive.
- If both vectors are infeasible, the one with the fewest constraint violations is preferred.
- If the mutant vector is feasible and the target one infeasible, the mutant is better.

Constrained problems are considered complex because they not only have to optimise the objective function but also determine the imposed constraints. Therefore, the best way to handle one is to quantify its constraint violations and its objective function so that each solution can be properly optimised [76].

Wang and Cai [96] integrated a (μ+λ)-DE with an improved ATM, named (IATM), which first normalises each constraint violation ((μ+λ)-CDE) in order to proficiently handle COPs. Each solution vector uses three mutation strategies (rand/1, current-to-best/1 and rand/2) and then the selection process-based ES is applied to determine which of a parent and new offspring can be allocated to

the next generation. The main disadvantage of this technique is that the tolerance parameter for the equality constraints is set as a constant value. Later, Jia et al. [97] proposed a new variant of (μ+λ)-DE called ICDE that dynamically changes the tolerance, with an archive-based ATM (ArATM) using infeasible individuals to maintain the population's diversity.

2.4.2. Repairing methods for EAs

Since constraint gradient information can be used to steer an infeasible solution towards a feasible one, some researchers have focused on using it to modify a/the constrained DE to enhance CHTs; for instance, Tetsuyuki and Sakai [90] introduced a ε-constraint method with an archive gradient-based mutation incorporated in DE called εDEag. It can maintain diversity by storing solutions in an archive with the ε-level parameter adopted to solve some well-defined constrained problems. The infeasible solutions obtained in each generation are repaired to feasible ones using the gradient matrix. The authors claim that the results of this technique are competitive with those of peer techniques but its processing is expensive.

More recently, Luo et al. [98] suggested an approach for selecting the infeasible solutions to be repaired rather than choosing them randomly as in [90]. It combines SRS with the εDEag algorithm and is called SRS-εDEag. In it, the whole population is classified into species using a clustering method [99] and then a number of the fittest infeasible solutions in each, as determined by the ratio of them to the feasible solutions in the same species, are repaired. The experimental results demonstrated that this algorithm achieved better results for most of the tested problems than the original εDEag.

Alducin et al. [100] proposed a coupling of two DE variants for solving dynamic COPs called DDECV+. In it, the two variants, DE/ rand/1 and DE/best/1, are combined with a repair method for re-sampling based on a mutation scheme, with the repair method applied only if the mutant vector is infeasible. From extensive

comparisons of its results, DDECV+ shows faster recovery than other algorithms but the main concern is the time it takes to repair all the generated infeasible solutions.

A gradient-based repair method was introduced into a DE/rand/1/bin scheme with a new constrained optimiser [81]. In it, if the solutions generated in each generation are infeasible, the repair method converts them to feasible ones. Overall, the proportion of the feasible to whole search space increases which leads to both the infeasible solutions and algorithm's performance improving.

2.4.3. Hybrid ensemble of EAs and operators

Motivated by the fact that no single CHT is preferable for all problems, recent research studies have introduced hybrid ensembles of different CHTs and/or different EA operators, as briefly described below.

Mallipeddi and Suganthan [87] proposed a new DE with an ensemble of CHTs (ECHT) in which a population is clustered into parts with each sub-population having its own CHT. The CHTs are chosen for different search phases according to some characteristics of the problem, such as the fraction of the feasible search space in the whole one, the problem's modality and the current stage in the search process (i.e., global exploration, local exploitation). The key issue in the ECHT is how to use each function call by each population associated with each CHT and adapt their evolutionary requirements.

In the multi-operator DE (SAMODE) method for solving COPs introduced in [101], the population is split into four sub-populations to which different DE operators are applied to generate new trial vectors. While the search proceeds, emphasis is placed on selecting the best-performing one for producing feasible solutions. It was observed that this technique performs better than other state-of-the-art algorithms.

Another DE-based algorithm with multiple search operators and two CHTs (i.e., the feasibility rule and epsilon constraint method) adopted by Elsayed et al. [80] is called ISDE-L. In it, each individual is assigned to a random combination of operators and then an improvement measure calculated for each combination with the best one selected to be re-used in the next set of generations. A local search method is applied to speed up its convergence while solving COPs. The results showed that this proposed algorithm is superior to state-of-the-art ones.

To accomplish a better CDE, a surrogate model integrated with DE, called ESMDE, is applied to generate competitive offspring in [102]. Specifically, different candidate solutions are generated using various combinations of mutation strategies and parameter values, and then the surrogate model helps to select the solutions for use in the next generation. ESMDE's performance was shown to be superior to those of self-adaptive DEs.

Based on hybridising different EAs, Zhou et al. [103] applied a hybrid of DE and PSO called HPSODE to solve constrained numerical problems. It aims to solve COPs in two phases using: (i) an information-sharing mechanism to prevent premature convergence; and (ii) a feasibility rule to rapidly find feasible solutions. The simulation results showed that it can effectively solve constrained engineering problems.

A fuzzy control scheme for quantifying the feasibility of a solution is proposed in [104]. In it, each candidate solution is given a contentment level based on its constraint penalty using a basic DE framework (rand/1/bin) and modified selection procedure. It compares an offspring with its parent using the α-comparison strategy, with the value of α iteration-dependent, while the number of feasible solutions is increased in later sets of generations.

2.5. Hidden Markov Model (HMM)

A HMM is a powerful and popular stochastic tool for modelling and analyzing a large volume of sequential data characterized as a sequence of observations with probabilistic measures [105]. It has been successfully applied to a wide range of applications, such as speech and image recognition [106, 107], weather forecasting [108] and financial stock market predictions [109]. Generally, it is used to represent a sequence of observations using probability distributions and modelling stochastic processes in which two types of states, observed and hidden, are involved [110]. Baum and Petrie [111] introduced the basic theory of HMM in the 1960s based on Bayes' theorem which is given as

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}, \tag{2.33}$$

where A and B are events and $P(A)$ and $P(B)$ are the prior probability and marginal probability used for normalization targets (i.e., towards obtaining evidence), respectively. $P(B|A)$is the conditional probability of B assuming that A is true and $P(A|B)$ denotes to the posterior of A.

Rabiner et al. [112] stated that the HMM is determined by a double stochastic procedure in which one process is a discrete-time finite-state Markov chain with unobserved (hidden) states while the other, which is observable, forms a sequence of observations for identifying its corresponding states. As a further explanation of how the HMM works, we use a simple example in which the weather is considered to have two states in one day, rainy or sunny, depending on observations. As the probabilities of a rainy day being followed by another rainy day and a sunny day by another sunny day are 0.6 and 0.8, respectively, we can express these transition probabilities by the state-transition matrix

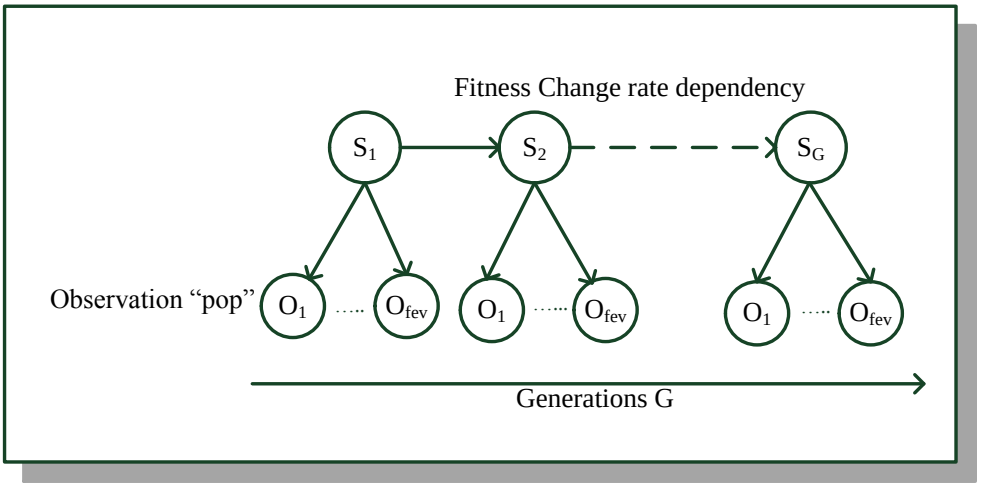


Figure 2.4: General HMM structure

$$\begin{array}{ccc} & R & S \\ R & 0.6 & 0.4 \\ S & 0.2 & 0.8 \end{array} \tag{2.34}$$

To predict the probable daily weather for the next five days, we need to define an observation sequence that reflects the previous states which is called a HMM.

2.5.1. Elements of HMM

The HMM is formulated by the following set of elements characterized by the Rabiner and Juang notations

- N: the number of hidden states,
- M: the number of observation symbols for each state in the N states;
- T: the state transition probability matrix (i.e., hidden states);
- E: the observed probability matrix (Emission matrix); and

- π: the initial state probabilities (at t=1) with some states with $\pi = 0$ unable to be initial states.http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf

Its general structure is depicted in Figure 2.4 in which any hidden state is determined by the current state and previous observation sequence.

Using the Rabiner notation, the HMM is expressed as follows. The set of hidden states is denoted by $S = (s_1, s_2, ..., s_N)$ with each state producing one observation set $V = (v_1, v_2, ..., v_M)$. For any system that can change one state to another, a set of states $Q = (q_1, q_2, ..., q_T)$ and their associated sequences of observations $O = (O_1, O_2, ..., O_T)$, such that $O_t \in V$, are attained. The transition probabilities from states i to j , following a Markovian property representing a $N \times N$ transition probability matrix, namely (T_{ij}), should fulfills:

$$T_{ij} = P(q_t = s_j \mid q_{t-1} = s_i) \tag{2.35}$$

Using the sets of states and observations, an emission matrix E can be expressed as $N \times M$ observation with the probability of observation (V) being created through state i as

$$E_i(v_m) = P(q_t = v_m \mid q_t = s_i) \tag{2.36}$$

A complete HMM includes three core parameter set is symbolized as $M = (T, E, \pi)$ whereas it should satisfy the following conditions:

- $\sum_{j=1}^{N} T_{ij} = 1$ where $1 \leq i \leq N$
- $\sum_{j=1}^{M} E_{ij} = 1$ where $1 \leq i \leq N$
- $\sum_{i=1}^{N} \pi_i = 1$ where $\pi \geq 0$

2.6. Visualization and reasoning for EAs

Visualization is defined as an interactive interface for analyzing and providing powerful depictions of large solution spheres [113]. It is an appropriate way for a family of population-based search algorithms to gain understanding of the evolutionary process. While using a visualization technique, users need a great deal of flexibility to anatomise various slices of the data [114]. Interactive interfaces and tools can help non-expert users accomplish the processes of data manipulation and analysis in terms of the search algorithm used [115]; for example, evaluating the huge amount of data produced by population-based algorithms, such as DE, can be significant using an efficient visualization tool to explore and analyze the potential procedure of DE, investigating different parameter setting's influence on its performance.

This comprises transmuting many of the data's values to a mental model of the underlying procedure which created them. Visualization is a method for analyzing information using different approaches, such as statistical measures regarding clusters, correlations and data distributions [113, 116]. Visualization techniques can be integrated with reasoning theory facilitated by interactive visualization so that visualizations can provide different ways of revealing correlated and complex information to users. They can simplify the means of obtaining the information, analyses and knowledge required to be manipulated to gain insights into advanced reasoning [115].

Visualization is described in [116] as the process for establishing a mental understanding and new insights using interactive abstract visual representations of the internal procedure of EA (i.e., DE). Interactive investigations of search data can produce many findings in terms of relations, patterns, outliers, etc. Due to the difficulty of tracking all the discovered patterns, synthesizing several discoveries and their relationships increases the cognitive overload [117] and, thus, controls the reasoning process, as discussed in the following section. Visual analytics is the

discipline of analytical reasoning expedited by interactive visual interfaces [115], which syndicates approaches from information visualization and computational data analysis to boost the analytical reasoning process [115, 117, 118].

Analytical reasoning is generally not a systematic process for choosing a suitable visual representation that simplifies understanding and explores iterations of searching or hypothesis testing [118]. However, in visualization, information seeking can be represented as steering in the ground pool with various interactions which support an analyst sighting the data in some way through the search process and interpreting the potential data generated. Users can extract and explore what has occurred during an evolution to direct them towards an effective approach for improving search and achieving better performance.

Based on the above discussion of visualization and analytical reasoning, the former is an essential part of a user interface rather than a final outcome of an analytical process [113, 118]. Moreover, representing the reasoning process using different visual methods could aid in determining and enhancing the next step in reasoning towards better solutions, as in DE. In DE, revealing different stages with different parameter settings through its process can provide a deeper understanding of the significance of parameter adaptation. This type of visualization is a significant part of an EA's cycle for obtaining the desired results. The entire visual analytical reasoning process is depicted in Figure 2.5.

Visualization techniques act as interactive visual paradigms to analyze and provide users with insights into the significance of enormous solution domains. Users with limited awareness of the underlying search process need an overview of the data, the effects of different parameter settings and resilience for analyzing specific parts of the data that can be manipulated proficiently; for instance, using an interactive visualization methodology can be beneficial for evaluating the huge amounts of data emanating from population-based algorithms, particularly DE which is considered in this thesis [16]. In general, EA visualization techniques have been developed for both on-line and off-line systems. The former provide

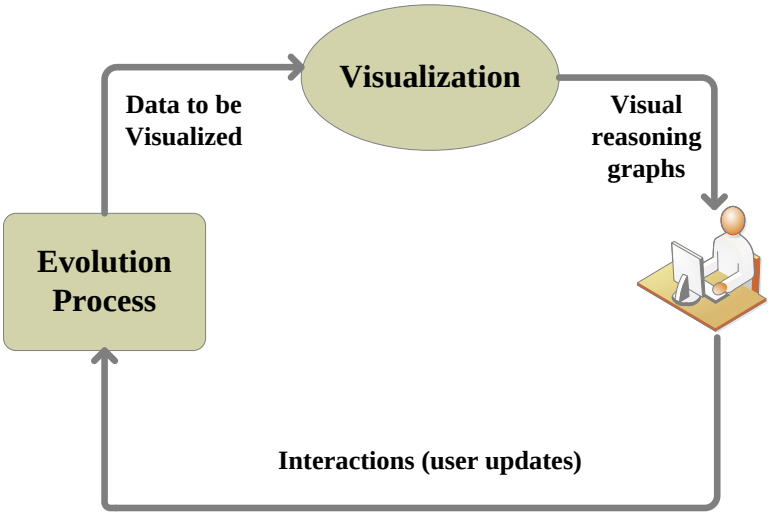


Figure 2.5: Process of visual analytical reasoning

users with the flexibility to interactively guide the progress of an EA while the latter allow users to evaluate the EA process only after their runs are completed [119]. Indeed, without using an on-line visualization of an EA, a user can only retrospectively detect its impact based on the yield data.

2.6.1. Basic concepts behind reasoning

Reasoning is defined as how to draw some inferences (i.e., outcomes) from pre-defined information (i.e., propositions) which requires going beyond the given propositions to obtain the correct inferences [115]. Basically, an inference process is categorized as deductive, inductive or abductive based on the level of validity of the inference [120]. On the one hand, if the given propositions or information can guarantee a true inference, this is called deductive reasoning. On the other, if the initial information can provide some conclusion with no assertion regarding its truth, this is called inductive or abductive reasoning. Researchers interested in developing a task have to more clearly distinguish between these two types of logical reasoning than those examining the performance of a task [121].

Generally, reasoning is the attempt to infer some rules, patterns or conclusions to describe the common relationships among all the elements involved and deduce the consequences from a certain premise that is assumed to be true by the community; for example, suppose that we have the following information about natural phenomena.

- If we observe that the sky is dark, we can infer that it may rain. Therefore, the information about the contours of the dark sky can be considered premises, which can help to correctly infer the high likelihood of rain.
- In particular, the process of inferring a conclusion from a set of premises usually assists people to scrutinise the internal steps taken towards it, as shown in Figure 2.6. For further explanation, the ground information and yield inferences can be expressed as statements containing either words or numbers that simplify the reasoning process, such as the following syllogism of the nature of an animal, which consists of two given premises from which to draw a conclusion.

We can explicitly define reasoning as the process of drawing some conclusions in different forms given a set of propositions containing assumed information. People use all the available connections between their experiences and knowledge in reasoning to see, think and then arrive at a pattern [115, 120, 122]. Mathematically speaking, reasoning can be used to validate and provide some proof for different arguments with the aim of verifying and explaining why the mathematical conclusions arrived at are correct [121, 123]. Different forms of reasoning are used in different contexts and it is desirable to model it using a variety of visual graphs through a sequentially directed process. In an analytical reasoning process, visualization of the information describes the search space based on different interactions, which enable a user to explore and analyze the findings obtained and infer the importance of applying an automatic parameter tuning method in DE [113].

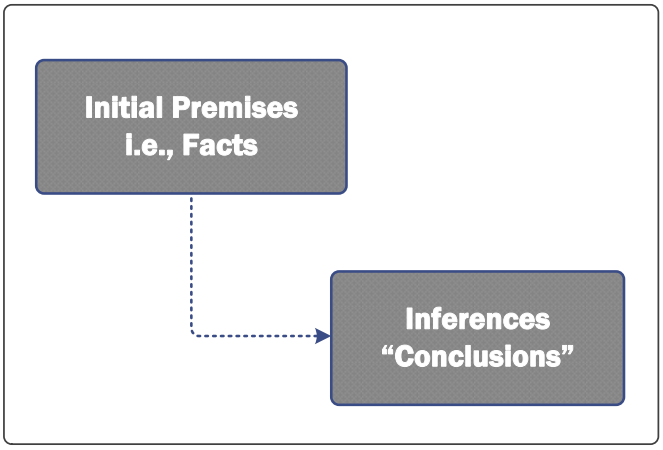


Figure 2.6: Basic reasoning process

Reasoning can be applied to an EA to analyze its vast amount of information in depth and explore that to obtain a full insight into its evolutionary state and dynamics like in DE process. For any iterative search process, its explanations of the actual/current states of the individuals involved should provide some clues about the evolutionary process and determine the proper configuration and setup of an EA for achieving outstanding outcomes. Data analysis and visualization are considered the fundamentals of visual analytics for EA data because of the usefulness and importance of this data, especially that of DE, for guiding its search algorithm. Through the data analysis phase, an EA user looks for some indications from the data to validate and emphasize pre-defined problem hypotheses.

Reasoning in an EA is best defined in relation to providing: (1) a suitable methodology for a DE algorithm which looks for solutions that interpret its information; (2) some justification for the suggested settings of the DE control parameters; and (3) critical thinking that involves the evaluation and meta-cognition of the merits of the DE methodology.

A. Types of reasoning

There are three major types of reasoning, as shown in Figure 2.7, logical, case-based and memory-based, the natures, functionalities and epistemologies of which

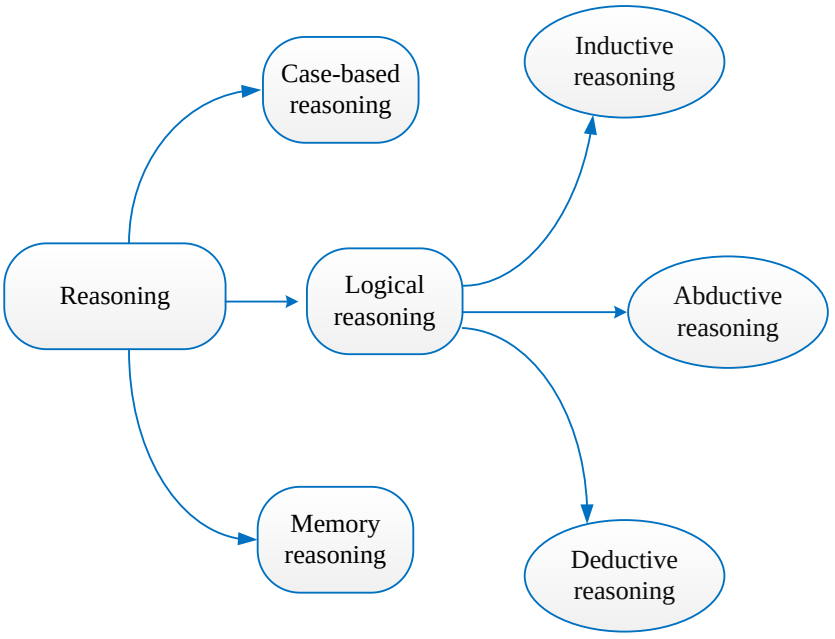


Figure 2.7: Types of Reasoning

are discussed below.

- **Logical reasoning**

Logical reasoning is a basic kind of reasoning which depends on arguments in different forms. It is rule-based, whereby its most common logic is the syllogism, which states that "an argument can be both a major or minor proposition and a conclusion" [124]. The validity of a conclusion relies on the forms of the argument and premises, with a true premise leading to a true conclusion. This is an umbrella definition that encompasses the sub-types of deductive, inductive and abductive reasoning, as shown in Figure 2.7. Logic can provide distinctions between its sub-types of reasoning which are forms of logic that arrive at a conclusion or inference based on assuming that the given hypothesis is true [125]. They require conceptual knowledge with respect to their roles during justification of the truth. More specifically, deductive reasoning can achieve a specific conclusion from generalized premises while inductive and abductive reasoning move from some events to achieve generalisations, as described below.

1. *Deductive Reasoning*- derives specific conclusions from a given set of general facts. It is a form of logical reasoning comprising a chain of basic statements (i.e., premises) that is followed by some complex truths assembled from ground statements [121], that is, when studying some mathematical theories, such as those of geometry, we can begin with some basic assumptions, inherit a proposition proven by these assumptions and so on for increasingly complex propositions.

2. *Inductive* Reasoning- attempts to derive general principles ('propositions') from several particular occurrences ('conclusions'). It incorporates all relevant experiences and then builds specific principles which are very beneficial for explaining another procedure [125], for example, the internal dynamics of DE. Inductive reasoning cannot provide actual proof of a theory with a 100% certainty but progressively finds some patterns and achieves generality by inferring from some specific situations or knowledge. It includes statistical and demographic reasoning, predictions, analogies and explanations based on a large number of observations with the possibility that a conclusion can easily be incorrect [126].

3. *Abduction Reasoning*- is the third form of logical reasoning, is quite similar to inductive reasoning. It was first introduced by 'guessing' [126], which suggested that conclusions inferred based on probabilities are used as premises for further predictions. It has gained a great deal of attention from philosophers of science as a means of scientific innovation It is the process of deducing or inferring the best rules and guesses that reveal conclusions and discover some new phenomena or information. It is thought of as a theory-forming or explanatory corollary, which shares inductive generalizations that are implicative inferences, that is, at the end of an abductive process; there is a putatively superlative explanation [124].

- **Case based reasoning**

CBR is a procedure for solving new issues based on the solutions to similar previous ones. It involves reasoning by remembering which includes analysing theoretical basics, system improvements and real applications using experience-based problem solving.

- **Memory based reasoning**

MBR is a way of categorizing cases with the same characteristics based on experience and stratifying the information from them to a problem. It discovers neighbours similar to a new feature and customizes them for prediction and classification purposes. The hidden Markov model (HMM), which is a technique that depends on the MBR, involves two processes, combination and distance functions. The former associates the consequences from the neighbours to obtain an answer while the latter computes the distance between any two records. Therefore, we use it as a basic component for adjusting DE control parameters, as will discuss in the next chapter.

In this thesis, we use the benefits of reasoning to visualize the internal processes of EA techniques, especially DE presented in Chapter 3, which helps to understand DE's random normality and the difficulty of finding its optimal parameters (F, CR). In Chapter 4 and 5, the HMM is used to adapt these parameters due to its advantages for handling random variables and time series data.

2.6.2. Existing EA visualization techniques and tools

The visualization of EAs can be helpful for analyzing a search space in depth. It can explore the convergence behaviours, fitness landscapes and dynamic parameter rates (i.e., CR and mutation) of evolutionary mechanisms. Visualization techniques can be executed to study EAs both on-line and off-line, and can be categorized as: **(1) Population Based Feature (PBF); (2) Local Dynamics Based Features (LDBF); and (3) Global Dynamics Based Features**

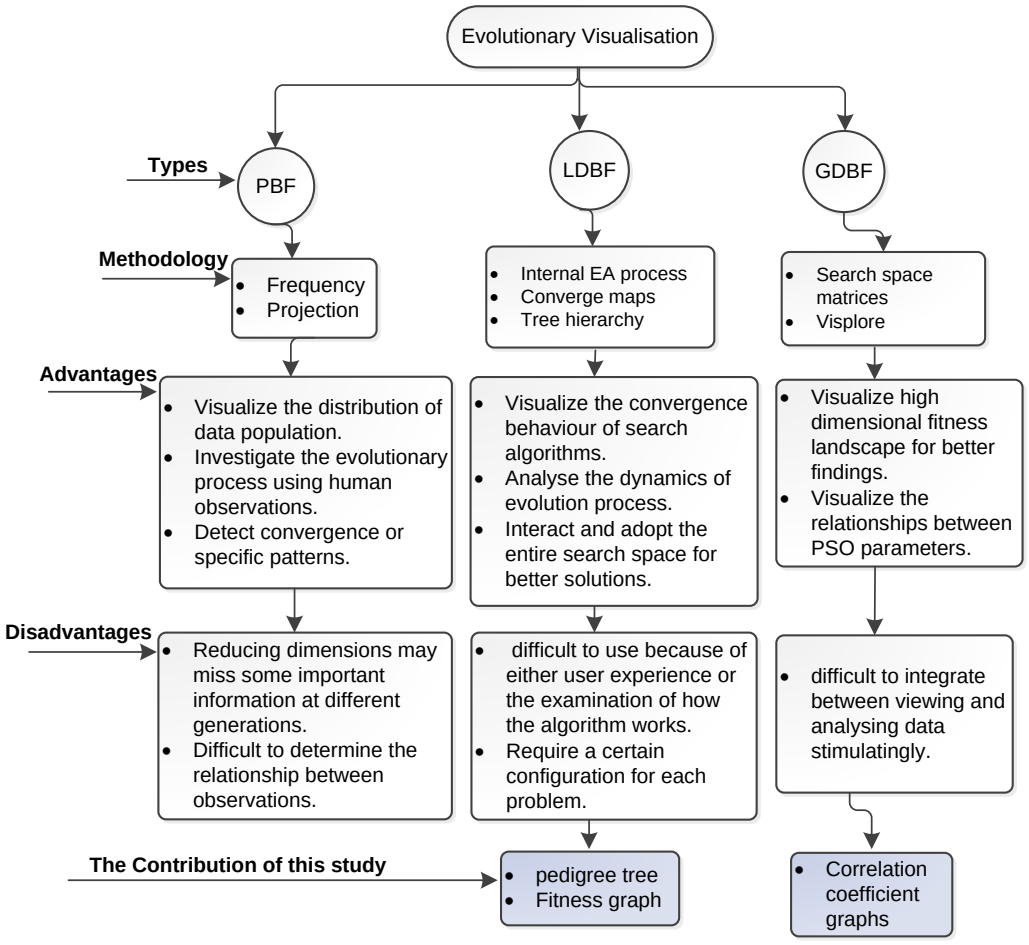


Figure 2.8: Visualisation techniques classification in EAs.

(GDBF). Figure 2.8 elaborates on them with respect to their types, methodologies used, advantages and drawbacks to demonstrate their relationships to our work in this chapter.

A. Population Based Feature (PBF) technique

Some forms of data visualization have been developed using population data matrices, which represent data sets; for instance, an imaging matrix shows the frequency of individual values (i.e., chromosome values) at their positions (i.e., chromosome positions) rather than illustrating broad population data sets [127]. Because of the massive amount of information in a data matrix, it is difficult to determine

the relationship between observations just by looking at the matrix. Therefore, multi-dimensional scaling techniques were developed to reduce high-dimensional data to two or three dimensions [116, 128], which led to the evolutionary process being efficiently determined by human observations. However, these techniques [116], which include glyphs, projections, parallel coordinates and Principal Component Analysis (PCA), fail to handle non-linear structures and biplots, and present graphs without meaningful information in different generations. Although self-organizing maps (SOMs)[18] have been used as visualization techniques to reduce high-dimensional data, they face some problems of arbitrarily high dimensional counts. ViSNE [129] is a mapping tool for transposing high-dimensional cytometry data into two dimensions. It can plot individual cells in a visual graph similar to a scatter plot but, as it uses a pairwise distance to locate each cell point, it does not ensure an efficient outcome.

B. Local Dynamics Based Features (LDBF) technique

Visualizing the search space has been applied in classical techniques, such as PCA [130] and glyphs [131] , to represent one generation. The Sammon Mapping [132] technique analyses the parental relationship between generations while the GEATbx toolbox [133] presents the off-line global optimisation capabilities over Genetic Algorithms (GAs) and EAs. It provides some graphs that visualize the convergence behaviours of search algorithms and supports multi-objective optimisation as well as multi-population and constraint-handling evolutions. Although it has large built-in plotting tools and runs on different operating systems (e.g., Windows and Linux), its environment requires mathematical and programming skills to manipulate the data and generate different graphs.

Another off-line tool is the VIS [134]which aims to visualize the information of an individual and describes parental information to reflect the effect of genetic operators when creating or removing an individual. Hart [135] proposed the GAVEL

system which is very similar to the VIS tool but extends it by including its capabilities for viewing individual parental information and showing the ancestry of single genes. The GAVEL tool assists a GA user to analyze and determine the dynamics of an evolutionary process. The GEVOL graph-based system [136] can help to understand changes in the evolution of software by generating a series of graphs which characterize its evolving states at different times using different colours. It also uses advanced force-directed layout algorithms to preserve a viewer's mental map obtained while observing different graphs. However, it cannot be a stand-alone system and, indeed, is intended to be integrated with other tools to provide a programmer with the capabilities to examine all aspects of a system.

The GONZO tool, which was developed by [119] for both on-line and off-line simulations, enables a system user to interact and adapt its control parameters. It is very helpful for obtaining good results from algorithms which is considered its key advantage. Also, it displays the decision space using metrics and provides summary graphs of a population's members and information of their ancestry. However, it does not offer a way of improving our understanding of how an algorithm works. Its on-line interactive tuning is criticized because the adoption of an algorithm is problem-dependent [119]. Rysselberghe and Demeyer [137] proposed a technique called Tomcat for visualising the histories of different states of software applications. It enables unstable components, coherent entities, and design and architecture evolutions to be detected but does not use a colouring approach.

Some other off-line methods for manipulating and visualizing the historical data of evolutionary studies have been proposed. The work in [138] focuses on two ideas: mapping a massive amount of historical data into a network graph to represent nodes and edges through an evolutionary process; and determining how to use spectrographs rather than different aggregation techniques. The data studied is elaborated at a relatively high level which leads to the discovery of more interesting patterns; for example, spectrographs are useful for highlighting trends and anomalies in the entities of data metrics.

Shine and Eick [139] described another methodology for visualizing the search space which generates coverage maps that include some means of estimating the solutions explored with respect to the whole search space. In particular, a map works as if there is an individual acting in more than one generation, always depicted in an identical locus, but the referred maps cannot provide any genetic information about the distance between individuals in the population. A tree-based methodology proposed in [140] visualizes the relationships among the objectives in multiple-objective optimization problems. It uses an iterative reduction approach and non-parametric metrics to gain an insight into the relationships among the objectives and conflict areas in these problems to help the decision-maker group the more effective objectives.

EvoLens [141] is a developmental approach for navigating and visualizing the software development process off-line which was specifically designed for object-oriented systems. It is based on a temporal lens view and its visual representation incorporates enhanced zooming using software hierarchy navigation. An illustration of visualization is presented in [142] in which a comprehensive set of graphs is reported to be capable of detecting the behaviours of evolutionary techniques and the type of information required to understand them mentioned. Some computer technologies for cool visualization have been introduced (e.g., Java/Java3D, VRML, MATLAB and NetMap) and the main criteria for choosing them indicated. However, they do not adequately cover evolutionary visualization. EAVis is a visualization tool that can assist the understanding of how EAs act by using several graphs [143]. It is considered to be an on-line rather than off-line visualization system, such as the VIS one [143], and enables an EA user to supervise the generation process on-line. Pohlheim [131] proposed a set of 'standard' visualization techniques for providing information on the progress of the evolutionary process which, combined with other advanced techniques, helps an EA user. However, using multi-dimensional scaling to visualize an explored search space is still an open research field.

SwarmViz [144] is an open-source visualization tool for Particle Swarm Optimization (PSO), which aims to monitor the progress of a specific optimization problem and adjust the relevant PSO parameters through a dedicated Graphical User Interface (GUI) to provide a visual insight into PSO for students studying optimization techniques. Although it enables a user to set the corresponding parameters and follow the algorithm's progress step by step, it cannot directly adapt the parameters of the evolutionary process. Also, extending it to the visualization of high dimensions incurs other problems.

Paterson [145] produced the VIPER tool which explores large animal pedigrees off-line in two stages of evolution represented in interactive visualization graphs. The first stage ensures that VIPER can handle and exhibit the types of data required in the next stage which leads to identification of the critical data features and documentation of errors. The second stage reveals the consequence of injecting errors into real and artificial pedigree data sets and, moreover, explores the possibility of visualizing them in a recognizable form for a domain expert.

C. Global Dynamics Based Features (GDBF) technique

The task of visualizing the whole search space of a problem has been addressed in the following different ways.

The Search Space Matrices (SSM) approach introduced in [130] visualizes low-dimensional graphs based on high-dimensional evolutionary data. It is based on a configuration space analysis which uses a single mapping for the entire search space and then generates comparisons of parental relationships between individuals over a set of iterations. This visualization can provide a user with the opportunity to observe and analyze a search's evolutionary path and be informed of the quality of the findings.

Khemka and Jacob [114] suggested an off-line toolkit called VISPLORE, the main aim of which is to run and analyze experiments using the PSO algorithm

which shares some similarities with evolutionary computational techniques such as GAs and DE. It is much easier to visualize the individuals, population or set of populations in an experiment during all its independent runs using this tool rather than other methods. Also, it is capable of visualizing the relationships among a PSO's parameters, clusters and other shapes in an analysis. Its visualization is very interactive and allows users to customize different graphs to explore the solutions generated by the algorithm on different levels. It begins with individual solutions, passes to the population that includes a set of individuals, and then a single experiment is conducted using a set of populations over generations followed by a pool of experiments. Although it is implemented as a mathematical palette, it provides a user with valuable insights into how to improve the PSO algorithm and, consequently, gain a better understanding of the high-dimensional fitness landscape.

The various visualization techniques discussed above facilitate the study of the convergence attitude of an EA's fitness landscape to provide a better understanding of the dynamics in the evolutionary process. However, they cannot provide a methodology for tracing the progress of evolutionary visualization to help a user follow the changes throughout this process, with respect to different DE configurations in order to improve performance due to:

- the high dimensionality of data which is very complicated to display;
- the inability to integrate existing tools to view and analyze the data; and
- existing tools not directly reflecting the adaptive parameters of an EA to achieve the best solution as they still work on a trial and error principle.

In summary, as attempts to use visualization techniques to analyze the potential processes of DE are limited, we propose a visualization framework as a possible alternative for analyzing a typical DE search to investigate its procedures as well as providing a deeper insight of the influence of DE's control parameters for enhancing its performance.

2.7. Chapter Summary

Since visualisation has a great effect on representing and investigating the potential procedures of EAs, particularly DE, with different parameter settings, in this chapter, a brief review of the visualisation theory and some recent studies in this area followed by an overview of EAs, including DE, are provided. Then, the CPs of DE, their significance for improving its performance and how they can be adjusted are discussed. Although several studies have attempted to solve the problem of controlling these parameters, most have adapted them independently.

Based on the literature illustrated in this chapter for both visualisation and parameter control, although we determine that DE's performance is highly dependent on controlling the combination of the F and CR parameters, their actual interdependencies have not been effectively investigated. Therefore, in Chapter 4, we aim to develop a novel algorithm by incorporating the Hidden Markov Model (HMM) in DE as a self-adaptive method for dynamically adapting these parameters through its procedure and improving its performance in terms of reliability and efficiency.

This chapter also presents a brief introduction to constrained optimisation with a review of different CHTs incorporated with EAs (i.e., DE) for COPs. The literature claims that a CHT is important for facilitating the rapid achievement of feasibility while solving evolutionary constrained optimisation. This is the motivation for extending the adaptation of DE's parameters to solve COPs by combining a specialised local search operator with HMM, as discussed in Chapter 5.

Chapter 3

Semantic Evolutionary Visualization framework for visualizing EA

This chapter discusses a proposed Semantic Evolutionary Visualization (SEV) framework [1] for demonstrating the depth relationship between the individuals of an Evolutionary Algorithm (EA), particularly Differential Evolution (DE), and retrieving information that can help EA users control its evolutionary parameters to considerably improve the performance of DE. Then, brief descriptions of the concepts of reasoning and evolutionary visualization are provided to identify the relationships among EA information. Following that, a review of recent techniques for visualizing EAs is presented to demonstrate the importance of the proposed framework. Finally, the benchmark dataset and experimental configurations used to visualize DE information and the results obtained are illustrated.

3.1. Introduction

Visualization has a great impact on different applications as it graphically represents the potential information of any task. The conceptual images provided by

[1]

- The work presented in this chapter has been published in:
  M. Keshk."Semantic Evolutionary Visualization", In International Conference in Swarm Intelligence (pp. 624-635). Springer, Cham. (July 2017)

various visual methods assist a user to conduct an in-depth analysis of any problem/procedure [115]. Generally, visualization is considered a form of the theoretical and analytical analyses [113] used in the EA literature as one of the primary means of gaining deeper insights into the dynamics of an EA during a search process. Thomas and Cook (2006) defined visual analytics as "the science of analytical reasoning facilitated by visual interactive interfaces". In analytical reasoning, the visual representation of EA information provides users with different graphs for exploring and analyzing the internal processes in a semantic visualization [113].

Visualization techniques can be applied to EAs to interpret the arbitrary information that changes over time. In an evolutionary process, the individuals in a population undergo cycles of selection, crossover, mutation and other parameters aimed at obtaining the optimal solution [117]. These operators enrich evolutionary dynamics where the population data is complicated to analyze. Therefore, EA researchers use visualizations to understand and test the search dynamics of a changing population. Although there is a long history of off-line EA visualizations reported in the literature, interpretations of their characteristics and changes during a search are often inadequate [32, 131, 143].

DE [16], which is a simple, nature-inspired meta-heuristic method, is one of the most popular EAs. Like any EA, it is a population-based algorithm that evolves a set of individuals which represent solutions to different types of optimization problems. Because of its various advantages, it has become a powerful alternative to other EAs, with its main difference of its differential mutation operator [16, 117]. DE has three main control parameters, namely, a scaling factor (F), crossover rate (CR) and population size (PS), which can considerably affect its overall performance because of its natural random process. While these intrinsic parameters should be automatically adjusted, the randomness norm of DE requires an accurate analysis using visualization techniques to display its information.

In this chapter, a visualization framework called SEV, which depends mainly

on the theory of reasoning defined as the process for drawing conclusions from predefined propositions [2] to infer that they are true, is designed. Using reasoning to describe the internal procedure of an EA, particularly DE, has become an active research area. Its methodology provides some ways of determining the main problems in the DE algorithm, in particular, proper settings of its intrinsic parameters. As dynamic changes in the process of DE affect its overall performance over time, it is difficult to achieve optimal solutions. This proposed SEV framework is specially designed to track and reflect the various dynamics of DE, and conduct direct analyses of the influence of different parameter settings on its effectiveness. To demonstrate its concepts, the DE algorithm uses two test functions with different parameter settings and characteristics.

The rest of this chapter is organized as follows. Section 2 introduces the proposed framework, explaining its major components. The experimental setup and results in section 3. Finally, a summary of this chapter is presented in section 4.

3.2. Semantic Evolutionary Visualization (SEV)

3.2.1. Main components of SEV

In this section, the SEV framework designed to integrate a visualization of potential evolutionary functions, especially parental information, with an analysis of the parameters (the CR and F) using different kinds of graphs, for instance, fitness of generations, correlation and linear regression (LR) ones, is discussed. It semantically analyses the possible relationships among individuals over generations of evolution to adapt the parameters and evaluate the performances of EAs.

As the main motivation is to offer visualization tools which can help a user with no experience to reason, we choose the word 'semantic' to reflect the evolutionary dynamics at the right level of resolution which can be identified by direct and

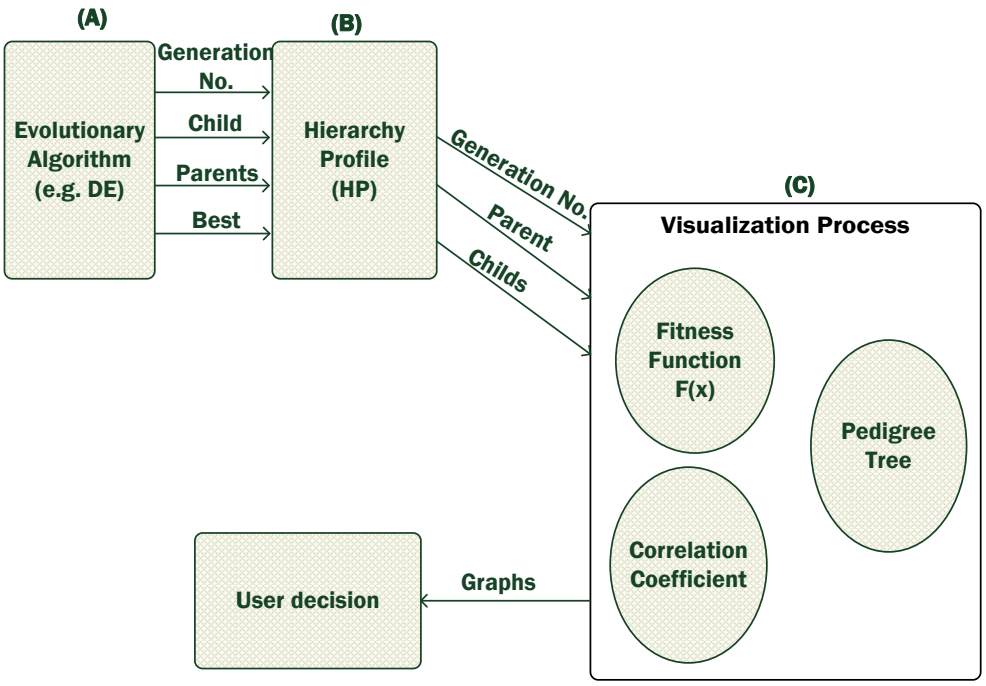


Figure 3.1: Semantic Evolutionary Framework (SEV)

readily available information that helps a user to steer the evolutionary process. This framework contains two new proposed visualization tools as well as a conventional one. Managing the three together triangulates three different pieces of complementary information which aids a user to understand the internal changes in evolution over time.

The DE algorithm, which is employed as an evolutionary heuristic optimization, is known as one of the most popular and fastest EA. However, any evolutionary optimization method can fit the framework without any modifications. Figure 3.1presents the three major components of SEV and their relationships: the evolutionary optimization algorithm, DE in this thesis; hierarchy profiling which works as a pre-processing stage for formulating the generated data for visualization; and the visualization procedure which involves three different types of plots, as detailed below.

A. Hierarchy Profile algorithm (HP)

Because of the vast amount of data generated from DE, the HP algorithm described in Algorithm 3.2 is developed. It acts as an intermediate stage in the corresponding visualization process, repeatedly pre-processing the data produced by the evolutionary process and fixing it in a proper format for the visualization stage.

The HP algorithm recognizes the best individual (solution) found and then uses it recursively to create an ancestry table from it up to the first generation via evolution. In particular, its inputs are the outputs generated from the DE algorithm (i.e., $IDdata$) and the best individual found (i.e., $best\_fitness\_individual$). The $IDdata$ of the child and parents are stored in $NodeParts$ to trace the hierarchy of each parent while $Vertices$ logs the child attribute to institute the relationship between the parents and their children to avoid replication during the representation of a pedigree tree. DE algorithm starts its procedure seeking to find the optimal solution ($best\_fitness\_individual$). We employ the HP algorithm as a pre-processing stage to re-format the targeted data (data needed for analysis and visualization) into such format that can simplify the visualization stage. DE outputs a set of elements $IDdata$ (Gen, parent, child1, child2, child3) whereas Gen refers to the generation-number and (parent, child1, child2, child3) represent their numeric identifiers (IDs). HP uses the $best\_fitness\_individual$ for starting the recursive process for creating ancestry table in which it includes the IDs for each child and its corresponding parents that generate it. After finalizing the process of HP algorithm, the hierarchy of each parent and the child attribute are stored in $NodeParts$ and $Vertices$, respectively for use by next visualization component.

This format facilitates the representation and analysis of the evolutionary hierarchy process since it declares the direction of the best solution towards the first generation. Figure 3.2 shows the parameters extracted from the results of

Algorithm 3.2 Hierarchy Profile algorithm (HP)

Input: IDdata , best_fitness_individual
Output: $Nodes$, $Vertices$

Set d=1
for (r =1 to *length* ($IDdata$)) **do**
  if ($IDdata[r].child$ =best_fitness_individual) **then**
    $Node1 \leftarrow IDdata(child, parents)$
  end if
end for
while ($d < length(Node1)$) **do**
  for (k =1 to *length*($IDdata$)) **do**
    for (u =1 to *length*($Node1$)) **do**
      if ($Nodes[u, d] = IDdata[k].child$) **then**
        $NodeParts \leftarrow$ *IDdata(child,parents)*
        $d = d + 1$
      end if
    end for
  end for
end while
$Nodes \leftarrow Node1$ *and* $NodeParts$
$Vertices \leftarrow IDdata.child$
return ($Nodes$, $Vertices$)

the DE algorithm using the HP algorithm which pass to the visualization process (component C).

Gen	Parent	Child1	Child2	Child3
1	21	1	13	4
⋮				
8	84	57	61	79
9	88	74	84	48
10	100	60	96	87
10	101	71	88	48

Child	Parent1	Parent2	Parent3
101	71	88	48
88	74	84	48
⋮			
23	6	17	9
22	3	2	12
21	1	13	4

Figure 3.2: Example of transforming DE data using HP algorithm

B. Visualization process

The visualization component in the SEV framework describes the techniques used to build different visualizations. This process consists of three visualizations, one classical and the others newly proposed which, for an EA, can be categorized as off-line and on-line and are capable of working in both modes. The classical one is the fitness graph which aims to determine the changes undergone during the evolutionary process and can be drawn as a line graph showing the best, average and worst fitness values in each generation. Although it is considered classical, it is important to provide an analyst with the critical information of the evolutionary dynamics (i.e., convergence). The other two are the pedigree tree and correlation graph, which are proposed for the first time. The former tracks and explores the ancestry information of the best solution found while the latter is a measure of the inheritance between the children and their parents. We illustrate each type of visualization in the following subsections.

- **Pedigree Graphs**

The main aim of a pedigree graph is to show the parental hierarchy from the best solution and its corresponding parents back to the first generation. It is a multipartite graph in which there are multiple connections among the nodes. We can see that all the parents in the graph's leaves represent solutions from the first generation. This graph is proposed mainly to visualize the evolutionary trajectories of the best solutions as well as those of their ancestors. Consequently, it can be drawn concurrently while the evolutionary process progresses and dynamically recalculated for each better solution found, with no need to wait for the completion of the evolution. There are two profiles with different settings for plotting the pedigree graph with different labels, one of which has a unique identifier id to show the ancestry order while the other is its fitness value which shows the relationship between each child and its parents from a fitness perspective.

To further explore the dynamics analysis, we consider the frequencies of different solutions for each generation. The frequencies of different solutions for each generation means that the number of nodes (pedigree tree) generated over generations is calculated and visualized to reflect the influence of changing CR values and the fitness function norm on the generated nodes. This can be helpful for measuring the performance of DE with a certain CR (i.e., convergence) as well as the nature of the fitness function.

- **Fitness Graphs**

These graphs are common visualizations used mainly in the community of EA computations. They usually describe different statistics of the fitness values in each generation over time, with their plots showing the best, average and worst ones based on the minimum, average and maximum objective values.

They can also comprise other statistical measures as well as plus the mode, median and exact percentile values. They help users to monitor the changes occurring in different setups, achieve the optimal solutions and determine how fast a population could lose diversity. Therefore, a EA user can directly control and adapt the parameters to enhance an EA's performance.

- **Correlation Graphs**

We use the correlation coefficient between two different solutions in a continuous space which acts as a measurement of inheritance equivalent to the schema length in a binary representation, that is, we estimate the correlations of the fitness values between children and their parents. In order to measure a correlation score between parents and children, the scores are ranked ascendingly for estimating to what extent the parents are close to their children. This type of correlation can be used as an indicator of the fitness landscape; for instance, if it is high, it may imply a non-rugged fitness landscape while, if it is low, we can conclude that, as

the local fitness values of individuals vary significantly, the fitness landscape is very rugged. We use the Pearson's Correlation Coefficient (PCC) [146] to measure the strength of the relationship between the fitness values of parents and children as

$$PCC\left(Children, Parents\right) \leftarrow \frac{cov\left(Children, Parents\right)}{\delta\left(Children\right).\delta\left(Parents\right)} \tag{3.1}$$

This is the classic correlation coefficient that measures the rate of variations in the fitness values among children and their parents in each generation which demonstrates the dependency ratio between the generated child and its parents as

$$Cor\_CP \leftarrow Child \sim Parents \tag{3.2}$$

This equation reflects the general relationship among a child and its parents which indicates their proximity, that is, it calculates the standard deviation (*std*) of the correlation. As it is observed that, if the CR increases, the *std* increases, this helps to adapt the CR to achieve the best solution; in other words, if the std decreases, it is necessary to change the crossover as

$$\delta(fitted\_value\_generation)\alpha\, Crossover \tag{3.3}$$

Also, we use a LR analysis as an extension of the correlation one of DE to determine the dependency relationship among the children and their corresponding parents. This is a well-established technique for analyzing and predicting the behaviours of data [147]. Equation 3.4 represents the potential process of executing the LR method where α is the intercept and β is the slope. In this chapter, a linear regression model formulated from the trend of fitness values over generations is computed against their child and parents to determine the relationship between a

dependent variable (i.e., generation) and one or more independent ones (i.e., child and each of its parents) as follows

$$f(x) = \alpha + \beta x \tag{3.4}$$

$$Fitted\_GCP \leftarrow [Generation \sim (Child + Parents)] \tag{3.5}$$

In this equation, LR fits the correlation between each generation and its individuals (children and parents) whereas a correlation analysis focuses on the relationship between children and their parents, that is, this correlation indicates the dependency ratio between a generation and its individuals while LR (Equation (34)) reflects the general relationship between a child and its parents which displays their proximity.

3.3. Evaluation of SEV framework

In this section, we discuss the experimental results obtained from our SEV framework based on its components.

3.3.1. Test functions and parameter setup

In this section, we present an example that illustrates the SEV's performance and the potential information that can be gained from these graphs. Table 3.1 presents the configuration of DE for implementing two trials with two different fitness functions, a simple unimodal one (DeJong F1) and a multimodal one (Rastrigin F2) in 2 dimensions (d), respectively defined as

$$f_1(x) = \sum_{i=1}^{d} x_i^2, i = 1.....d, -5.12 < x_i < 5.12 \tag{3.6}$$

Table 3.1: Configuration of DE

Trials	Trial 1	Trial 2
Strategy	DE/rand/1/bin	DE/rand/1/bin
population size	20	20
No. of generation	10-dimensional	10-dimensional
Mutation rate	0.5	0.5
Crossover rate	**0.1**	**0.5**

$$f_2(x) = 10.d + \sum_{i=1}^{d} x_i^2 - 10.cos(2.\pi.x_i), i = 1...d, -5 < x_i < 5 \tag{3.7}$$

We use DE with the basic strategy of DE/rand/1/bin, a mutation rate of 0.5 and PS of 20 for 10 generations, with the difference between the two trials that the CR is 0.1 for trial 1 and 0.5 for trial 2. Then, the outputs from both trials are used in the hierarchy-profiling algorithm to prepare the data for the pedigree graph visualization. The purpose of this configuration is to evaluate the performances of DE and how visualization can reflect changes in the parameters, specifically the CR. The findings from these trials are used to build the HP profile illustrated in Figure 3.2 which reconfigures the DE results to easily visualize a pedigree graph.

3.3.2. Experiment results

In this section, we discuss how visualization can help to understand the potential processes of DE and how, by interpreting the graphs, the parameters, such as CR in this study, can be adapted to enhance the performance of DE. Visualizations of the two test functions for the two trials using the DE algorithm and the Pajek [122] tool that designs the two outlines for exploring the best individuals in each trial, are described as follows.

- **Pedigree tree analysis**

Firstly, we use the HP profile to easily produce a pedigree graph. As previously mentioned, the Pajek tool [122] helps to generate two plots with different labels for

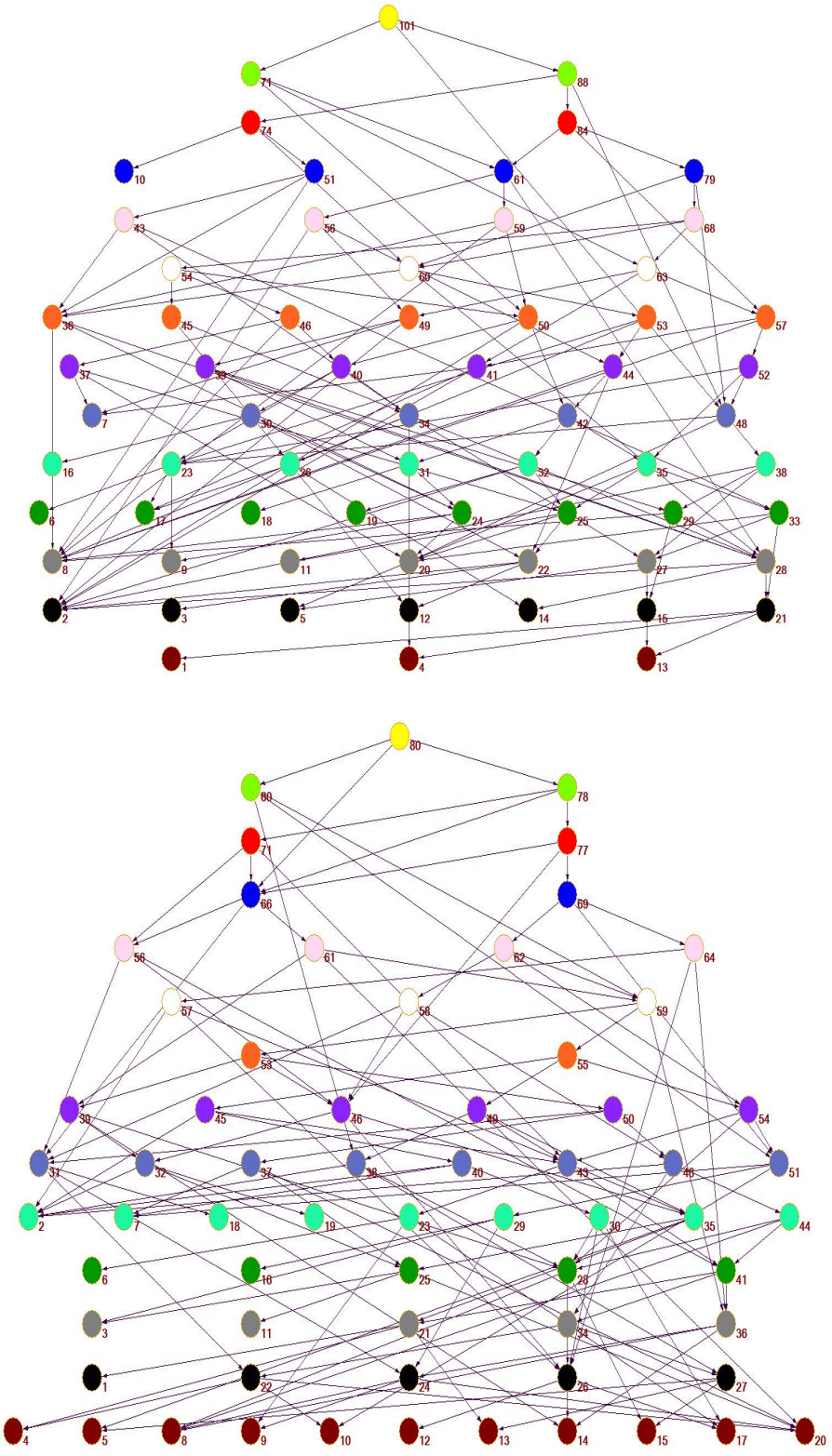


Figure 3.3: Pedigree Graph Visualization of the DeJong's two trials using DE algorithm

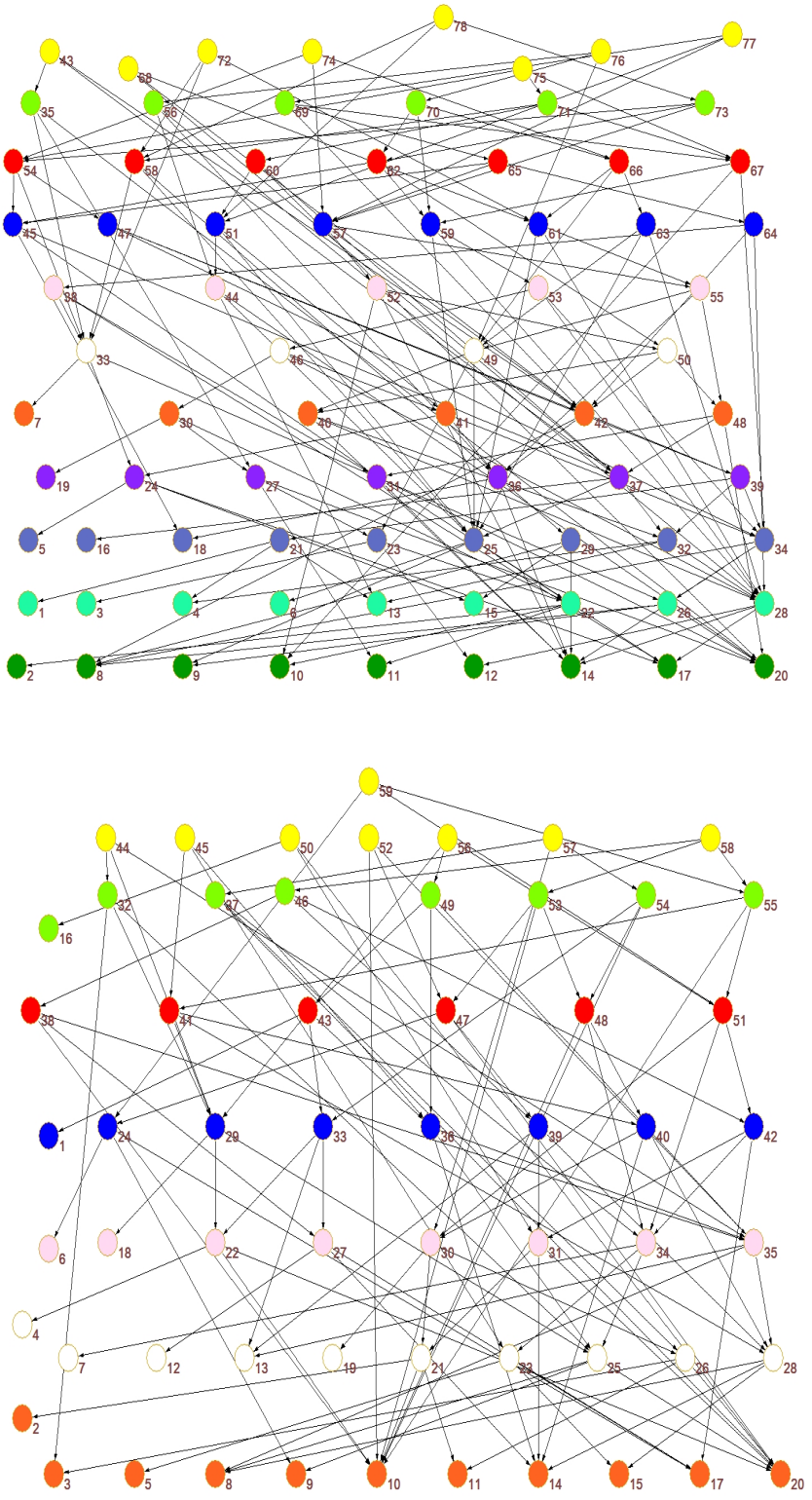


Figure 3.4: Pedigree Graph Visualization of the Rastrigin's two trials using DE algorithm

each test trial for each function. The main aim of this graph is to explore the best individuals and their ancestral hierarchy back to their first-generation parents. Figures 3.3 and 3.4 show the tree representations of the best solutions obtained from the Dejong F1 and Rasrigin F2 trials, respectively, as described in Table 3.1. These figures indicate the backward recursion flow of the best solution obtained from the last iteration until it reaches its parents in the first generation. It is worth providing the hierarchies of the best solutions labelled with their identifiers to enable the orders of generations to be easily understood. The pedigree tree using the HP profile can also facilitate observations of the potential operations of DE in a semantic way by representing the useful individuals that influence each generation moves towards the best solution.

We can see in Figure 3.3 that the identifier of the unimodal F1 trial's best solution is 101 and that of the second 80. We can see the effect of changing the CR value over the two trials as its higher value converges faster than its lower one. For the multimodal F2, Figure 3.4 displays the identifiers of the best solutions as 78 and 59 for the first and second trials, respectively. Both these figures provide us with a verification network for query purposes rather than explicit information on a screen.

Like any verification graph, although it can be more complicated, we simplify it by using the graph rewrite rule to skip the transitional parents generated between the DE's recombination stages; for example, if parent1 acts as the parent of parent2 and parent2 the parent of parent3, we can leverage from the transitive nature of this clerestory relationship and rewrite it as parent1 is the parent of parent3. At this time, parent2 is dropped and replaced by its father to be maintained as the grandfather of a child. In this way, we can skip and replace as many as possible of the intermediate parents to simplify the graph and it can be observed that all the solutions generated in the first generation are parents.

Using the fitness function as the other pedigree profile will help to examine this graph in order to understand evolutionary dynamics; for example, a user can

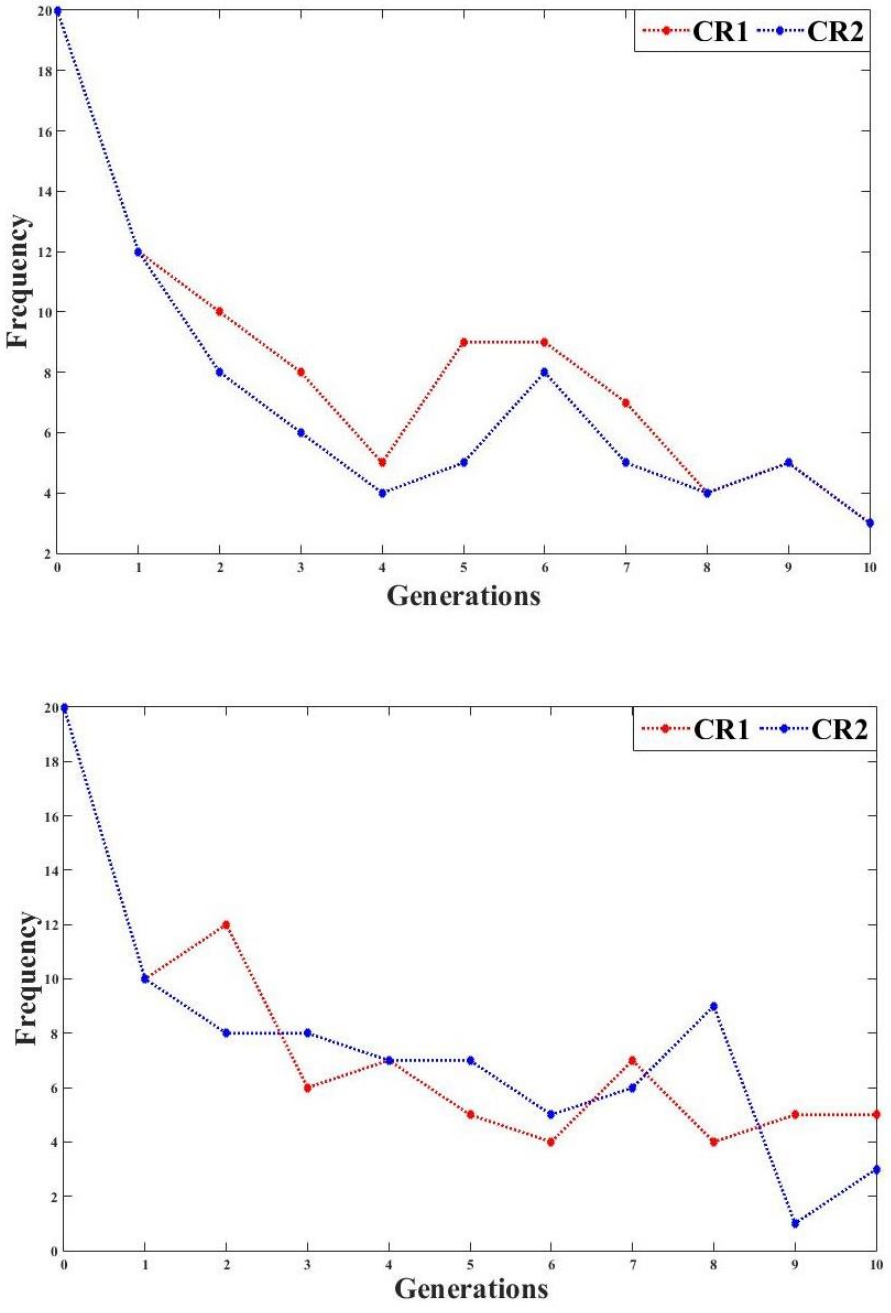


Figure 3.5: Frequency ofgraph nodes over generations for DeJong and Rastrigin functions, respectively.

estimate the many candidate solutions in this graph from some queries. Similarly, we can analyse the ruggedness of the fitness landscape by applying a social network analysis, such as visualizing the energy of the network's evolution. After the evolution suffers from stagnation, there are no more changes to the best solution from one generation to the next as no new node representing a new better solution is considered and the energy equals zero. However, if new nodes are generated with better solutions over generations as the evolution continues to move from one good area to another, the energy will be high.

Secondly, it is interesting to show the number of nodes generated over generations as this helps to perceive the effects of changing the CR and frequency of new nodes in the pedigree graph over generations. Figure 3.5 displays the frequencies of new nodes generated in the tree plots for both the test functions and number of nodes generated over time. The first plot in Figure 3.5 shows the DeJong unimodal function, using which the lower CR in trial 1 leads to more solution nodes than those generated by the high CR in trial 2. The second plot in Figure 3.5 presents the nodes generated by the Rastrigin multimodal function with multiple local minima which highlights that the numbers of solution nodes obtained from both trials fluctuate over time but are more consistent in trial 2 than 1.

- **Fitness graph explanation**

A fitness graph is a direct illustration of the fitness values of each generation, with the y-axes in Figures 3.6 and 3.7 expressing their objective values. Its plots are beneficial for revealing the influence of the recombination strategy (i.e., crossover) on an/the algorithm's performance. Figures 3.6 and 3.7 depict the two trials of both the Dejong and Rastrigin functions, in each of which the three lines that represent the best, average and worst fitness values over generations lose diversity over time.

For the Dejong function in Figure 3.6, the best and average solutions are extremely close and lose diversity linearly; in contrast, the diversity of the worst

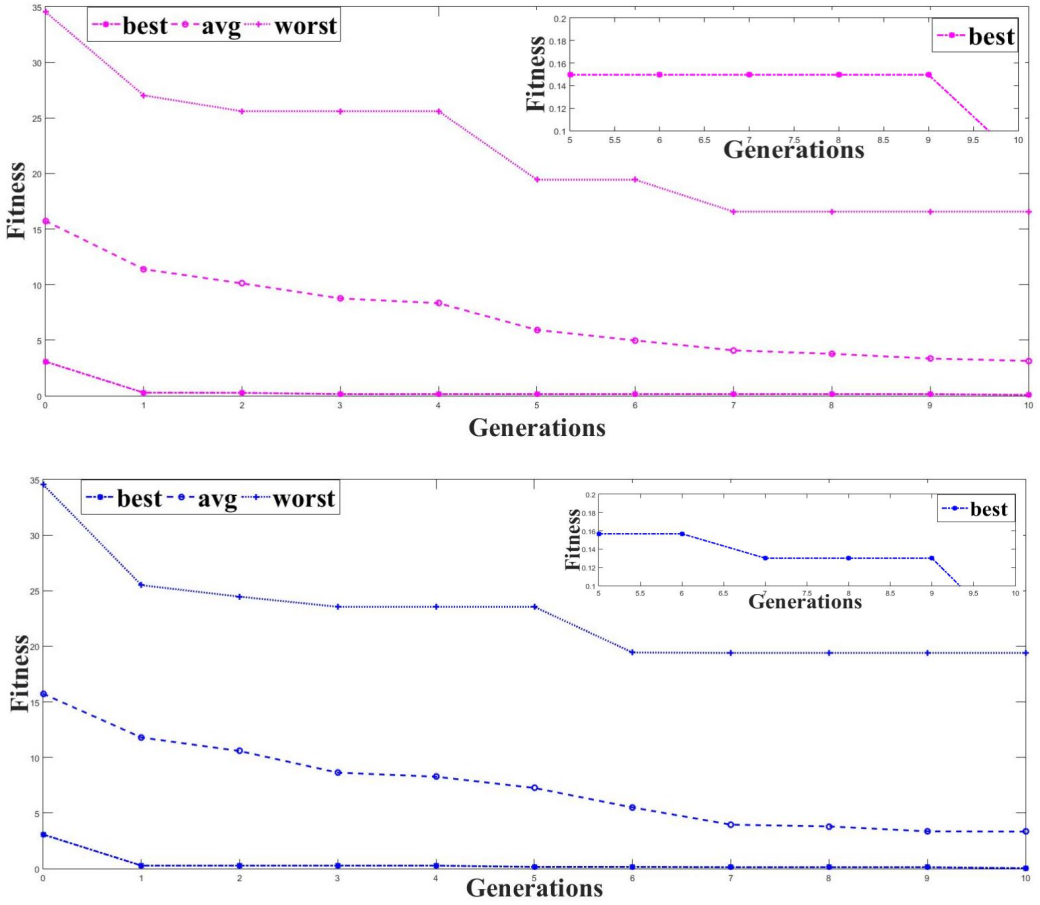


Figure 3.6: Fitness graph for DeJong function (trial 1 and 2), respectively.

one shifts gradually over generations. In the first trial, the best individual moves during the first 5 generations to find a stable position. Conversely, in the second, it doesn't settle in a stable position until generation 7. It is interesting to use the best solution as the best indicator of convergence because that in the first trial converges before that in the second as it reaches a stable position first and, therefore, the lower CR in trial 1 reflects lower diversity. Similarly, the Rastrigin function in Figure 3.7 describes the potential evolutions of solutions over generations in the two trials. We can see that a smooth convergence is not easy because of the multimodal nature of the function. However, the second trial is more efficient in finding the best solution with the lowest optima (close to the global ones) than the first, which may become trapped in local optima during convergence. As this graph is generally considered a very popular visualization, it is discussed further

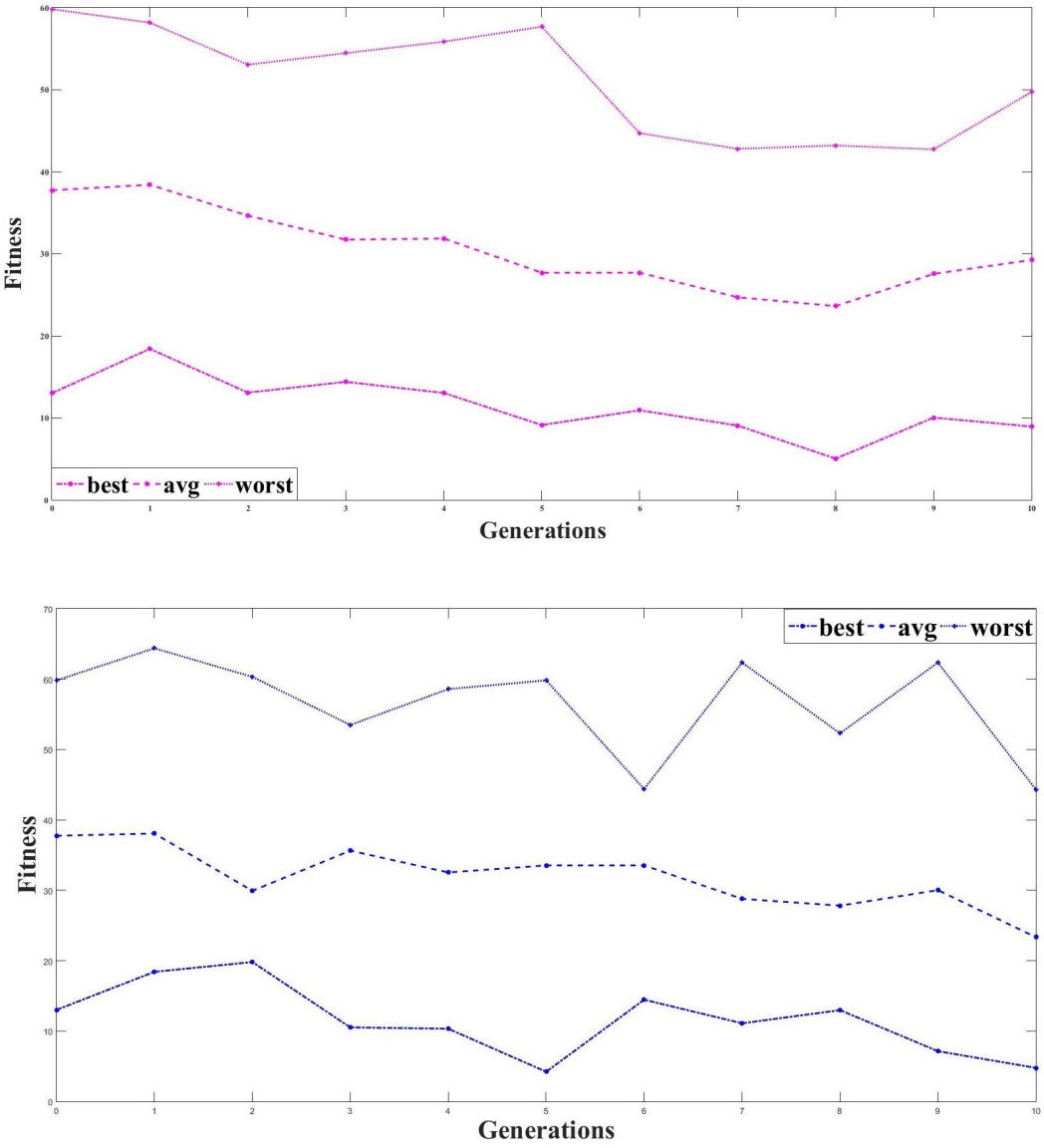


Figure 3.7: Fitness graph for Rastrigin function (trial 1 and 2), respectively.

with the corresponding visualizations for the two evolutionary test functions in both test trials.

- **Correlation analysis interpretations**

To demonstrate the strength of the relationship between a child and its parents over generations, Figures 3.8 and 3.9 show the dependency correlations between

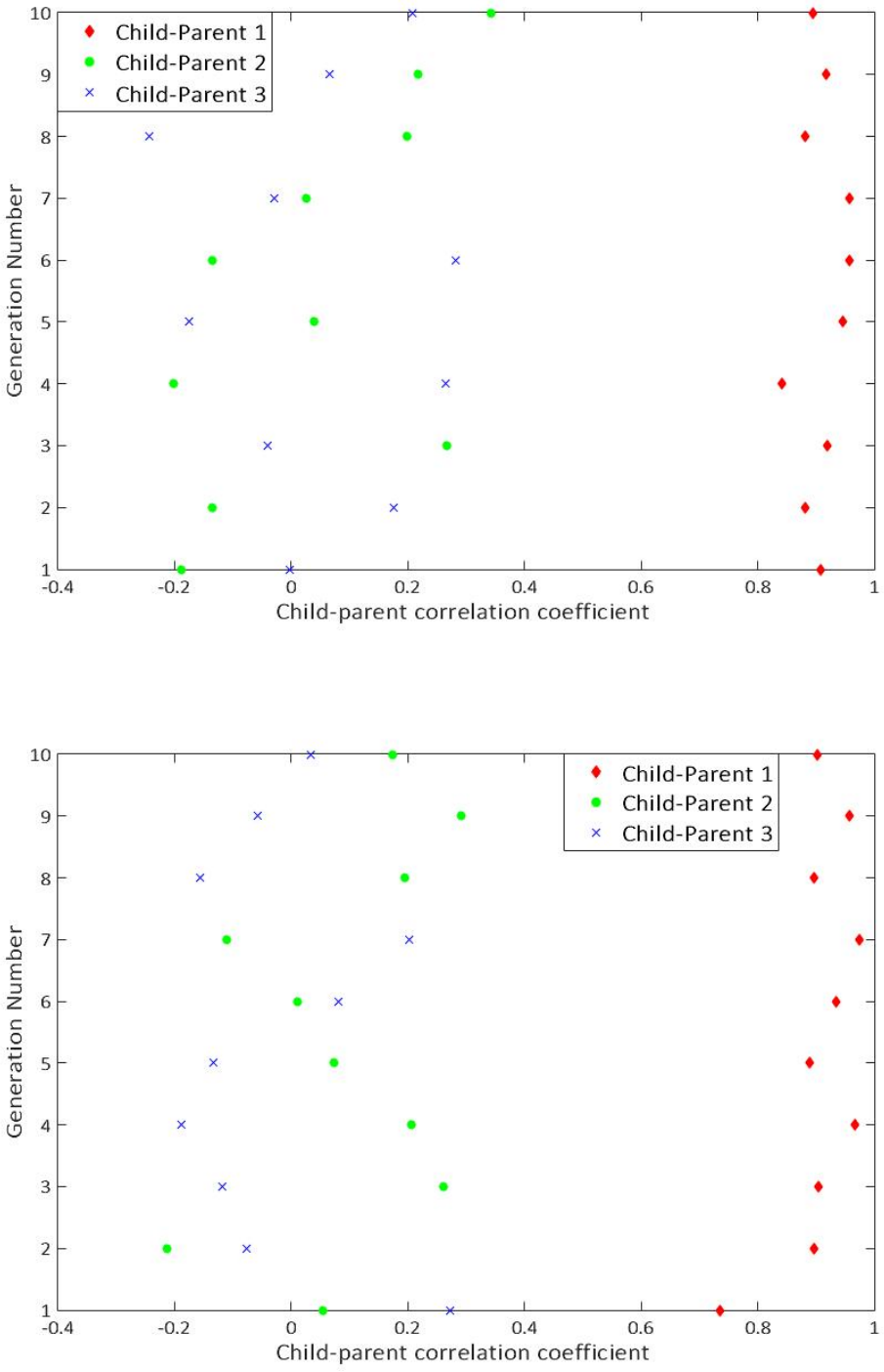


Figure 3.8: Correlation coefficient of Child & Parents for DeJong trials 1 and 2, respectively.

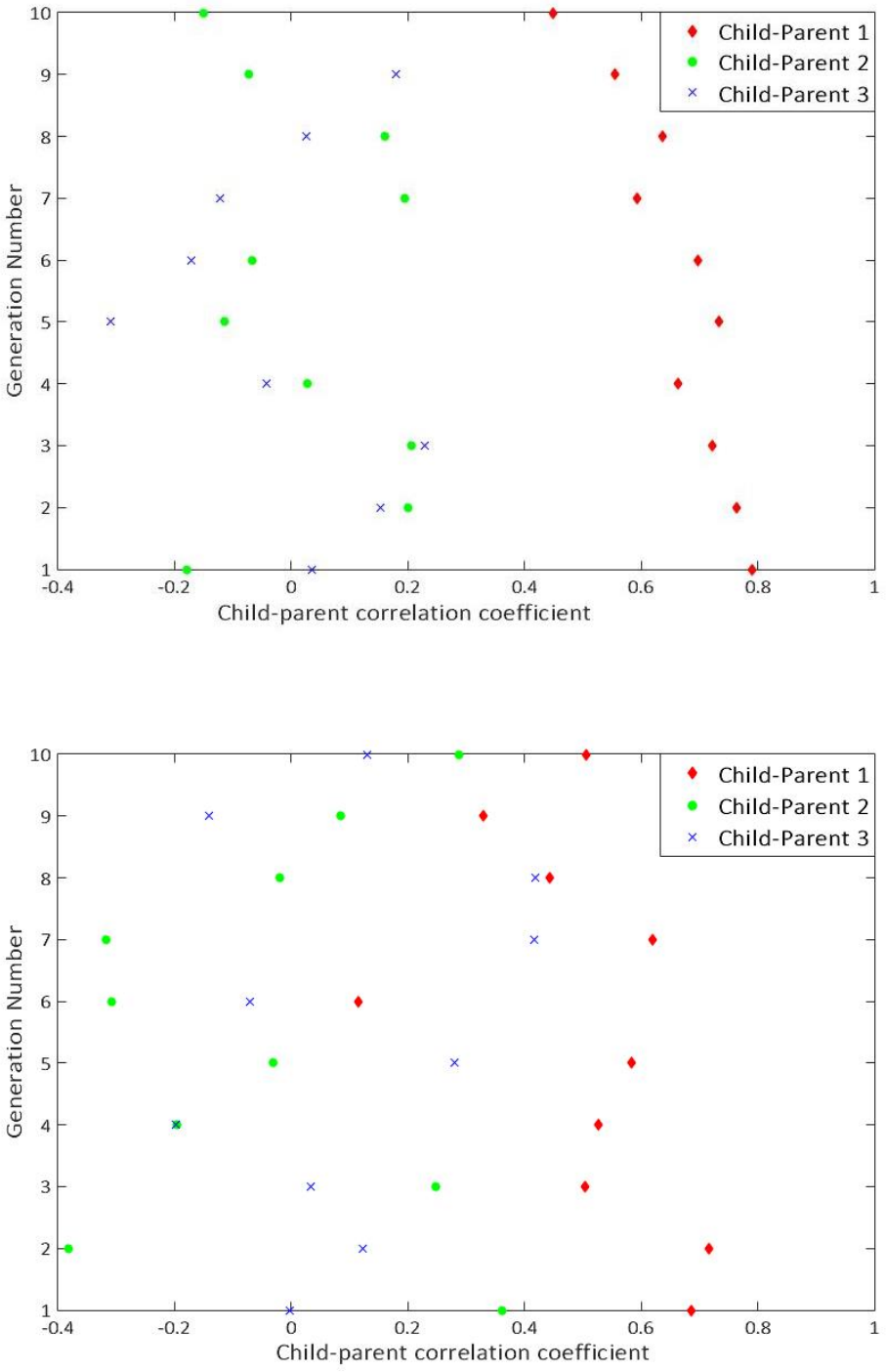


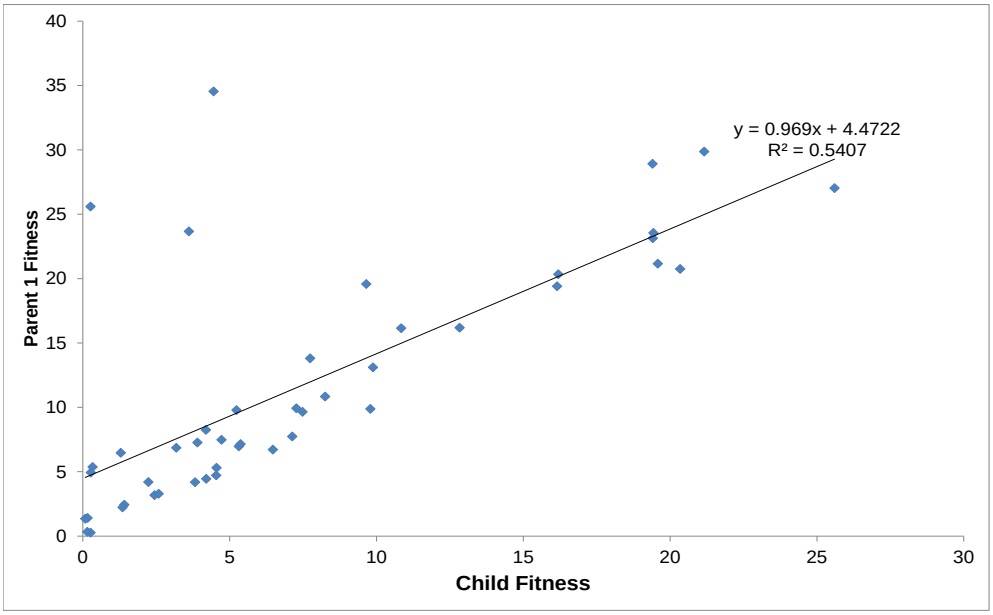
Figure 3.9: Correlation coefficient of Child & Parents for Rastrigin trials 1 and 2, respectively.

their movements over generations in the trials of test functions F1 and F2, respectively. In more detail, in the F1 correlation relationship depicted in Figure 3.8, there is a high correlation between the child and parent1 which is normal because of the nature of the F1 recombination process that is consistently high because of its fixed step and length and non-rugged nature of the fitness landscape. We can use this correlation in rugged landscapes and for self-adaptive algorithms, as in Figure 3.9 in which the correlation between the child and its first parent is reduced compared with that for F1. It also indicates the equilibrium between exploration and exploitation as it is expected that the correlation between the fitness values (objective function values) of the parents and child will be low during the former and, assuming a less rugged fitness landscape, high during the latter.

Visualising a scatter diagram of the fitness values of the child and first parent provides an indication of the strong correlation between the two for the DeJong function as in the first plot of Figure 3.10, a significant sign that the fitness landscape is smooth and consistent, as is the case for our unimodal function. Conversely, the correlation in the Rastrigin function in the second plot of Figure 3.10 is dropped because the nature of its landscape is relatively rugged and difficult to explore, as is the norm for the multimodal function.

Finally, the dependency correlation of each child and its parents over time is interpreted using a LR analysis, with LR applied on Dejong F1 to explain the concept.

Figures 3.11 show the dependency correlations between the movements of the child and its parent over time. The graphs on the left-hand side represent the correlations between each generation and its individuals (i.e., the child and parents), with the fitted line showing that the individuals start and end with almost no correlations over generations. This indicates the diversity rates obtained by computing the cumulative standard deviation for the correlated generations with individuals shown in detail in Figure 3.11. On the other hand, the graphs on the right-hand side represent the correlations between a child and each of its parents.



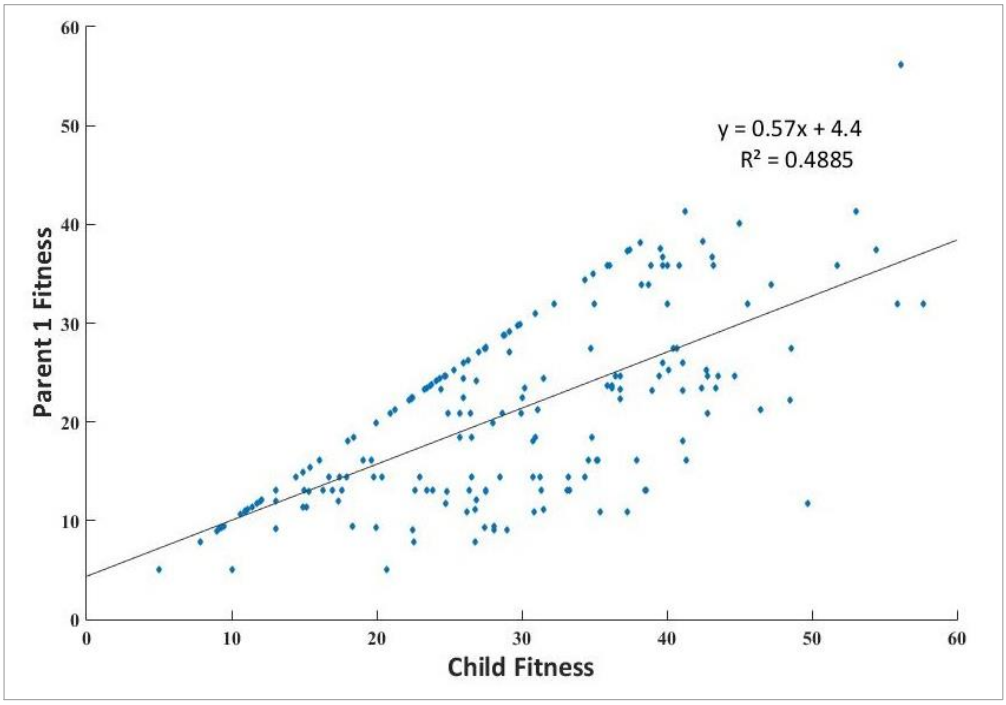


Figure 3.10: A Scatter diagram showing the relationship between the child and first parent fitness for DeJong and Rastrigin functions, respectively.

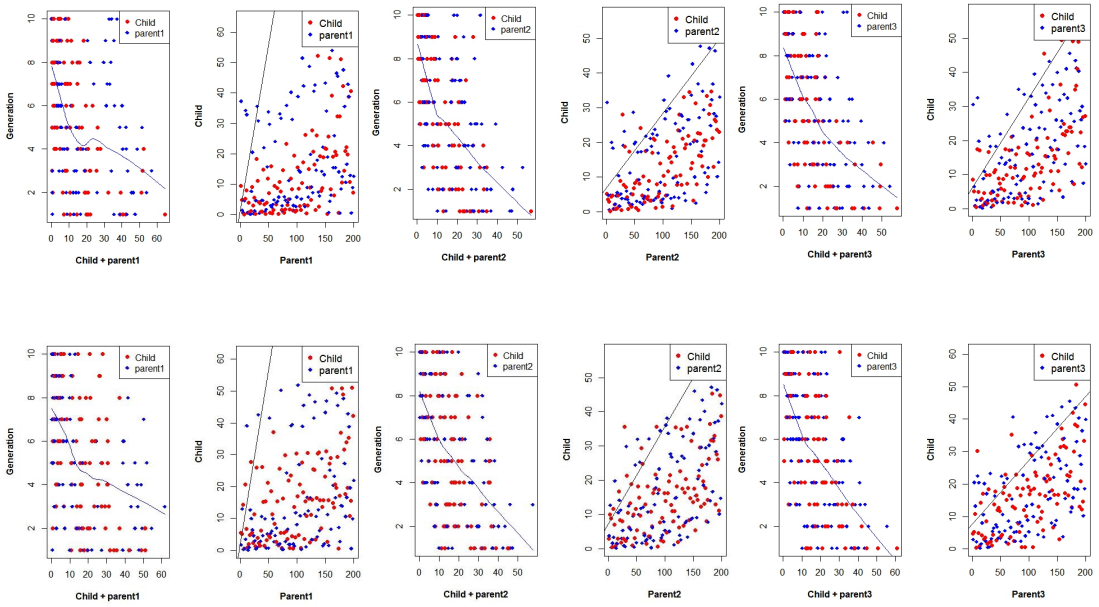


Figure 3.11: Linear regression for trial 1 and 2 of Dejong F1, respectively

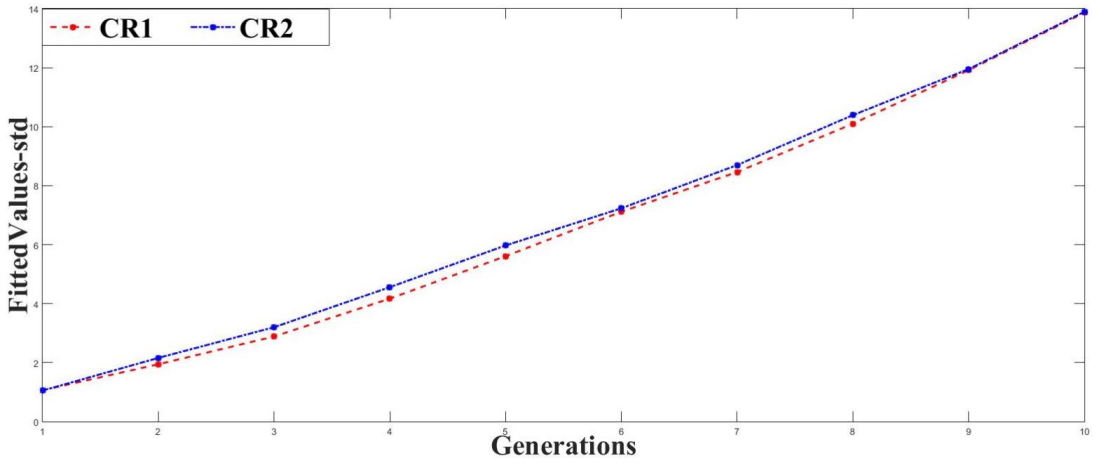


Figure 3.12: Standard deviation of fitted values for Dejong F1 linear regression

As comparing these two figures highlights a slight difference between the two CRs over generations, it is necessary to adapt these values to improve the population's dynamics over generations.

Based on Equations 3.3 and 3.5, for each trial, the cumulative standard deviation is computed for the fitted values of LR to measure the variations between individuals and their relationships with the CR, with Figure 3.12 demonstrating

the influence of the CR on the fitted correlation values for Dejong F1. The standard deviation in the first trial is less than that in the second which means that the higher CR leads to a higher standard deviation. Consequently, the diversity will increase and the correlation between individuals decrease. Ultimately, this relationship shows that, if the standard deviation of the fitted values decreases, the CR should be adjusted to preserve diversity and enhance the performance of DE.

The visualizations proposed in this chapter can help in determining and analyzing the potential evolutionary process of the optimization mechanisms, however, with the increasing of population size and problem dimensionality, the visualization needs to be refined to match an increase in problem dimensionality.

3.3.3. Comparison with existing techniques

Our proposed SEV framework is compared with visualization techniques to determine its influence. Firstly, the techniques in the PBF category [127–129]usually denote the occurrence of a population of individuals but do not reveal the core interactions between these individuals. In this chapter, we customize the frequency of graph nodes (candidate solutions) in Figure 3.5 and the scatter diagram in Figure 3.10 to determine the connections among each child and its parents, which can refer to the paths produced through evolution in the fitness landscape.

Secondly, the techniques in the LDBF category [140, 144, 145] express the evolutionary changes whereby pedigree tree and fitness demonstrations are developed to understand the internal evolutionary process, with these visualization methods complementary in nature. However, some LDBF methods suppose that users have sufficient experience to examine the dynamics during an evolution whereas pedigree trees are simple heredity concepts which can be equally understood by expert users and those with less or no experience.

Finally, the techniques in the GDBF category [114, 130] demonstrate the search domain features of data matrices while using second-order statistics, especially the correlation coefficient and regression analysis, as shown in Figure 3.8, 3.9 and 3.11, which demonstrate the dependency between individuals as they help to recognize actual deviations of the vectors.

3.4. Chapter summary

Visualization is one of the best ways of understanding, and gaining an instant impression and representation, of an evolutionary progression. Many studies have visualized the internal procedure of an EA, with this thesis focusing on DE. Extracting and using relevant information to determine the progress and concurrent states of the evolutionary process assists inexpert users to understand complex evolutionary dynamics in order to work towards achieving better performances.

The amount of data produced by DE to enable analytical reasoning to describe and explore the findings from the search space present challenges in terms of obtaining an ultimate visual representation of DE's useful information to facilitate the filtering and recognition of some specific inferences. Therefore, in this chapter, we present a visualization framework called the SEV for achieving these goals, which proposes visualizing specific features and offering an explicit reflection of the evolutionary dynamics.

The features of this framework include two new visualizations and a classical one which provide comprehensive information that can assist users to semantically reason about the internal process/dynamics of evolution. It presents a novel hierarchy-profiling algorithm for arranging the data generated from DE to fit the visualization process which encompasses three types of graphs, pedigree, correlation with LR analysis and fitness, the first two of which are newly proposed while the third is a conventional one.

Initially, as an example of EAs, DE is tested on two different functions with distinct characteristics and parameter settings for two trials. The resultant visualizations show the relationships between children and their parents as well as the internal evolutionary dynamics evidenced by correlation and pedigree graphs, respectively. Overall, this framework can be applied without modification to any optimization algorithm. The plots track the ongoing changes within an evolutionary run by exploring pedigree trees and the fitness landscapes produced and will help to investigate the influence of different parameter settings so that a new self-adaptive DE using HMM is introduced for solving unconstrained and constrained problems in Chapters 4 and 5, respectively.

Chapter 4

A novel self-tuning methodology for solving unconstrained optimisation problems

In this chapter, a novel self-adaptive technique for solving unconstrained optimization problems, which couples the Hidden Markov Model (HMM) with the classical Differential Evolution algorithm (DE) to automatically adjust the DE control parameters over its search, is introduced [1]. It is called the DE-HMM, with an overview of the HMM explaining the main elements needed for building it and a description of our technique presented to indicate the role of HMM in adapting the DE's parameters. Details of the benchmark test suites used to evaluate the performance of the DE-HMM are provided. Finally, the experimental results obtained and an analysis of its components for solving these problems are discussed, and a comparative study of it and some existing techniques conducted to compare their performances.

4.1. Introduction

DE is one of the most popular and effective evolutionary algorithms (EAs) [34] and, similar to some of the others, is a population-based search method [29] in

[1]

- The study introduced in this chapter has been submitted in:
  M. Keshk, H. Singh, and H. Abbass, "Automatic Estimation of Differential Evolution Parameters using Hidden Markov Models" (Under review).

which three major operators (i.e., mutation, crossover and selection) with three control parameters (i.e., a scaling factor (F), crossover probability/rate (CR) and population size (PS)) are employed during its search process to guide the progression of its evolution towards the global optimal solution [35]. Although it is easy to use as it has a simple structure and is faster than other EAs, its performance is highly influenced by the settings of its control parameters [34].

The most proper configuration of DE's control parameters can differ from both one problem to another [148] and in the same problem during the evolutionary process with respect to their interactions [34, 148]. Therefore, determining their settings for various optimization problems is a complex issue [48]. Although several studies have introduced adaptive/self-adaptive methods for parameter control that tune DE's intrinsic parameters, as reviewed in Chapter 2, the interactions among these parameters have rarely been investigated.

The core of this chapter is the proposal of a novel self-adaptive methodology for automatically adjusting the DE's intrinsic parameters (the F and CR) in order to address the parameter control challenges by incorporating the HMM into its procedure to enhance its performance. The resultant algorithm is denoted as DE-HMM, which two parameters are self-adaptively chosen in each generation.

The DE-HMM methodology is tested using two benchmark data sets [19, 20] and its performance extensively compared with those of the classical DE (i.e., fixed values of the F and CR) and twelve comparable algorithms, eight of which are competitive DE variants and four non-DE EAs, with respect to effectiveness and efficiency.

This chapter is structured as follows: section 1 discusses the general architecture of the proposed DE-HMM methodology ; in section 2, the experimental study, including details of the benchmark data sets, configuration used and results obtained are provided ; and, finally, this chapter is concluded in section 3.

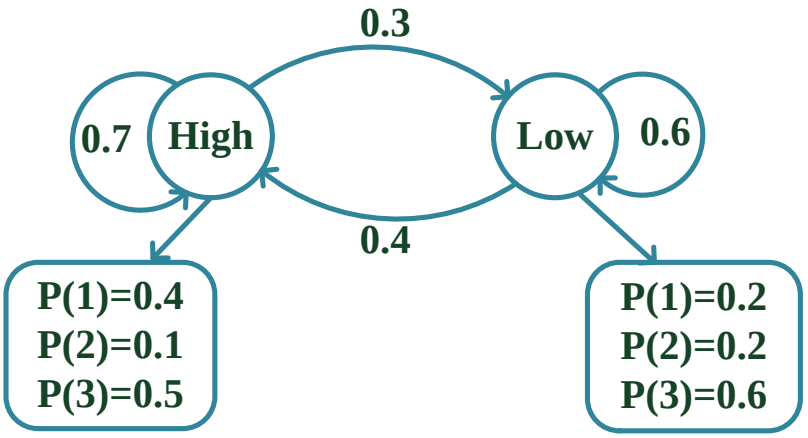


Figure 4.1: Diagrammatic example of HMM transition diagram for DE. the low and high states represent the low and high change rates of DE, respectively, and the observation set (O = {1,2,3}) reflects the three dimensions of a population

4.2. General architecture of the proposed methodology (DE-HMM)

4.2.1. DE combined with HMM

As previously mentioned in chapter 2, the basic control parameters of DE (i.e., the F and CR) need to be carefully set for both different problems and throughout the same problem. Several studies in the literature present various ways of adjusting them with most adapting them independently; for instance, the JADE algorithm [60] separately adapts the F and CR using Cauchy and Gaussian distributions, respectively. In [29], Mallipeddi et al. claimed that, although combining the F and CR greatly influences the performance of DE, rather than setting their values separately, their actual dependencies have not been successfully investigated.

Therefore, in this study, a new self-adaptive algorithm is proposed for automatically adjusting the F and CR using the HMM, namely DE-HMM. It incorporates the HMM in the DE procedure for addressing the parameter setting

challenges (i.e., problem-dependent, evolution-dependent, and the complex interaction among them) where the DE parameters are adapted per generation. The first challenge can be overcome via employing HMM for dynamically setting these parameters. The second challenge is overcome by coupling each individual vector with its own parameter set (F and CR) and the third one is achieved by estimating F as a function of CR.

The speed of DE does not significantly get affected because all the required statistics for the HMM are calculated based on first-order mathematical measures, such as the mean and standard deviation at a linear computational cost.

Using such HMMs learning algorithms is motivated from two perspectives. Firstly, several studies have employed finite Markov models to analyze the behavior of evolutionary computation methods. At the beginning of 1980s, Goldberg and Segrest [149] used finite Markov chains to anatomize the behavior of simple genetic algorithms (GAs). Mahfoud [150] continues based on that work and used finite Markov models to forecast the expected drift time for a Boltzmann tournament selection strategy. Davis and Principe [151] claimed that there is a unique stationary distribution for simple GA while mutation is evolved for parameter control. In essence, the mutation operator represents a source of disorders to the population, whereas selection influences the transitions from one population to another. Markov chains have been used widely to examine and demonstrate the behavior of evolutionary computation in recent years. An old, but very useful survey is presented in [152], and more recent research studies on the matter including [153, 154].

Secondly, and Although a great volume of work that focuses on using Markov Chains for analyzing evolutionary computation techniques, no study has actually applied Markov Chains to help evolutionary computation, improving its performance. In contrast, differential evolution was customized to enhance Markov chains by building a population of chains [13].

HMM are known to be more precise and generates smaller models than simple Markov models. The only research study that we are aware of using HMM for evolutionary computation is Rees and Koehler study [155]. Their work showed that HMM can accurately estimate the evolutionary parameters. In their work, many evolutionary computation algorithms with fixed parameters were ran and used HMM to estimate these parameters in an offline mode, which conclude that it was able to precisely estimate the parameters using HMMs. This work basically motivates this study by enquiring of whether HMM can in fact be used to estimate the parameters in real-time and use these estimates to adjust the DE's parameters to improve the performance of the optimization process?

To model the HMM, it is necessary to define its states (T) and observations (E) based on actual data. In multivariate time-series models (MTSM) [156], $x_0, x_1.....; x_t$, where t is the time consumed for function evaluations, as in DE and the HMM, with these variables consisting of some values observed in the DE population. However, as there is no information of the observed sequences of these values during processing (i.e., the hidden states), the HMM is used to approximate them, as schematically depicted in Figure 4.1. The types of these states, either discrete or continuous, depend on the actual data norm [157]. To identify the internal variations in a population's individuals over time, as a form of MTSM, two discrete states are defined in advance in the HMM to measure the changes in the DE's fitness probabilities.

The DE-HMM adapts the F and CR by calculating the posterior and likelihood ratios of the HMM, respectively. Due to the similar functioning of the HMM and DE, as the basic idea of DE is to change its population over time, which is the same as the core process of the HMM that reflects those of time-series models [156], we use the former to adjust the latter's control parameters. To identify population changes, we assume two states, 'low' and 'high' which indicate negative and positive fitness function behaviours, where the fitness function does not decrease and increases over time, respectively. These states can be evaluated

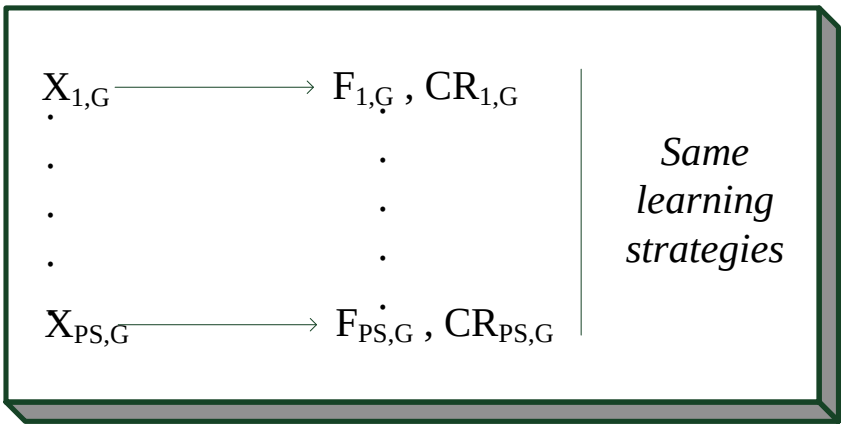


Figure 4.2: Encoding of control parameters with individuals

using the HMM procedure to calculate their posterior probabilities, with values from 0 to 0.5 considered low and from 0.5 to 1 as high representing the HMM transition matrix. The advantage for using categories is to avoid over-fitting an HMM to a certain population as our purpose is not reverse engineer the parameters as Rees and Koehler [155] did, but to approximate the parameter values that have to be adopted. The HMM emission matrix is computed by the probability that a population is generated through the DE stochastic process. A complete example demonstrating the working of DE-HMM is presented in Appendix G.

To represent the HMM, a weighted graph is used, which is represented as a transition diagram in Figure 4.1. The nodes denote the states and the edges the potential transitions among these states weighted by their transition probabilities, with the summation of them equalling one for each node. Each state has its own observations represented by an emission probability matrix as HMM deals with both observed and hidden events.

The DE-HMM estimates the intrinsic F and CR parameters for successful individual ($x_{i,G}$) in the population to generate new offspring (i.e., a trial vector), as shown in Figure 4.2. It self-adapts them by concentrating on their inter-correlations whereas the preferred parameter values have to produce the most likely individuals to survive in the next generation. This inter-correlation is modelled by

first estimating CR then estimate F using the estimated CR in conjunction with the minimum posterior probability for the given population observation emission matrix (EM_{seq}).

Figure 4.3 presents the complete procedure for computing the F and CR of DE. The relationship between DE and HMM is shown, the F and CR calculations based on various HMM probabilities and the posterior and Likelihood ratio for F and CR, respectively. The DE population is converted into probabilities to fit the HMM requirement where the initial population is taken as prior probability; the parent vectors, referring to the population observations are taken as likelihood probability; and the mutant vectors, denoting the change from the parent vectors are taken as the posterior probability. HMM uses these probabilities to compute CR and F.

Algorithm 4.1 Transformation of population values to probabilities.

Input $\leftarrow$ Population pop_m as a matrix of size $R \times C$

Create a new matrix pop_r of size $R \times C$;
Create two vectors μ and σ of length C;
for each column j in pop_m **do**
  $\mu(j) \leftarrow mean(pop_m)$ over all rows for column j;
  $\sigma(j) \leftarrow stdev(pop_m)$ over all rows for column j;
end for
for each row i in pop_m **do**
  for each column j in pop_m **do**
    $pop_r(i,j) \leftarrow prob(pop_m(i,j) \leq \zeta),\ \zeta \in N(\mu(j), \sigma(j))$;
  end for
end for

Output$\rightarrow pop_r$

The HMM requires three elements to be computed for its procedure, that is, $M = (T, E, \pi)$. We use the DE population values as input for calculating these parameters, with the HMM built through the running of DE. Firstly, the state-transition matrix of the HMM (T) reflects the changes in these values during the evolutionary process, either low or high, measured internally based on observations of the DE population [158, 159]. Secondly, the emission matrix (E) is calculated from those observations, which reflect the interdependencies of the evolutionary

process. As a population sample has to be transformed into probabilities to achieve the requirements of the HMM, the parent and mutant vectors of DE are converted into probabilities using Algorithm 4.1, where it loads a population of size R rows and C columns and determines the probability distribution for individuals in the population, pop_m. The initial states (π) are set at $0.5, 0.5$ for $low, high$ to start building the model with equal probabilities.

To compute the F and CR, we use the likelihood ratio between the actual sequence of states ST_{seq} and the most likely state sequence $best_{seq}$ estimated using the Viterbi algorithm [160, 161] (discussed later) to calculate the CR, where the emission matrix (E) denotes the likelihood. Similarly, an EM_{seq} observation sequence is derived from the Viterbi algorithm to compute the posteriors, with the minimum ones averaged with the estimated CR to reveal the F values. The steps in the procedure for estimating the F and CR parameters are presented in Algorithm 4.2, and details of the main components of the DE-HMM for adjusting the F and CR in the following section.

4.2.2. Main components of DE-HMM

A. Computing crossover rate (CR)

The CR usually needs to be carefully set according to the problems' characteristics, such as linearity and modality (i.e., unimodal or multimodal) [34]. It is the main component for determining the potential probability of mixing both the current target vector (parent) and donor vector (mutant) produced from the mutation operator to generate a new trial vector (offspring), which is necessary for improving a population's diversity as the evolution progresses. In this thesis, we use the simple binomial crossover scheme that customs the CR to construct the offspring, as discussed in Chapter 2.

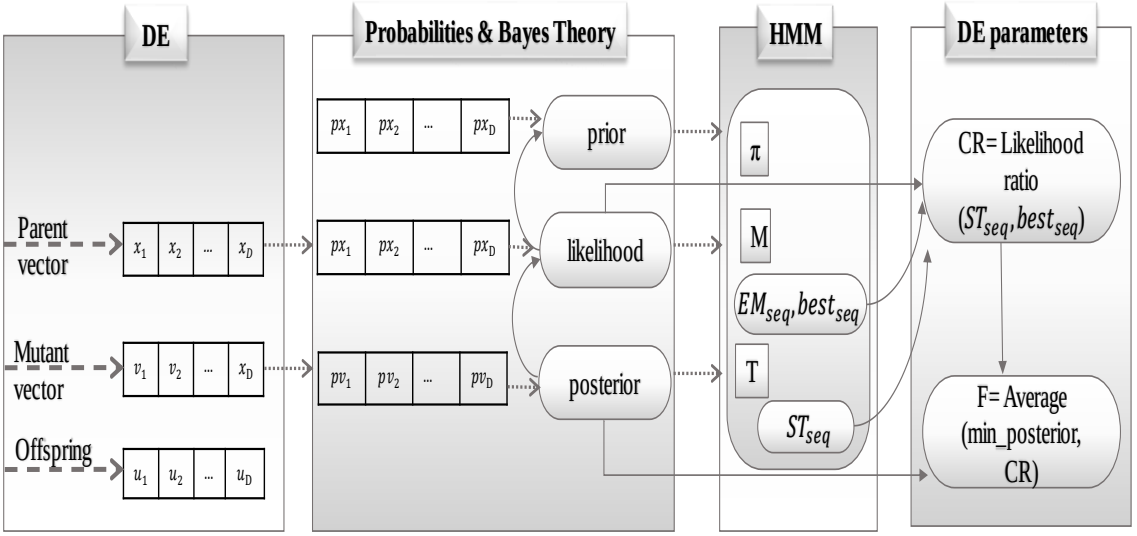


Figure 4.3: Complete procedure for computing DE control parameters.

The Viterbi algorithm [160] is applied through the process of adapting the CR, which is the best way of choosing the best state sequence that maximizes its probability given the observation sequence. This is a type of dynamic programming that acts as a decoder given the model's parameters (T, E) and observation sequences $O = (O_1, O_2, ..., O_T)$ derived from the DE population. It can optimize the CR by finding the most likely state sequence considering the maximum of all previous ones $max_{q_0,q_1....q_{t-1}}$. In this way, the Viterbi probability $v_t(j)$ for a current state (q_j) at time (t), is estimated as

$$v_t(j) = \max_{t=1}^{N} v_{t-1}(i) * T_{ij} * E_j(O_t) \tag{4.1}$$

where, $v_{t-1}(i)$ is the prior probability of the Viterbi path, T_{ij} denotes the probability of transition from states q_i to q_j, and $E_j(O_t)$ the emitted observation probability with the current state j.

The proposed DE-HMM algorithm estimates the CR value by computing the likelihood ratio of the actual state sequence (ST_{seq}) produced, being better than the best state sequence $(best_{seq})$ produced by the Viterbi algorithm, given as:

$$CR = Likelihoodratio\ (ST_{seq} \geq best_{seq}) \tag{4.2}$$

where we use $\geq$ to indicate that the sequence ST_{seq} is better than the sequence $best_{seq}$. The main reason for calculating the CR using the likelihood ratio is that, using the ratio of a target vector to its mutant vector, constructing new offspring can be considered the likelihood of the actual state sequence (ST_{seq}) being better than the best state sequence ($best_{seq}$) (i.e., being in a state $ST_{seq} \geq best_{seq}$), as depicted in Figure 4.3. Therefore, the changes between (ST_{seq}) and ($best_{seq}$) produce the CR values, and identify the improvement of the new vector being through its ratio in each sequence. In other words, comparing the values of the state sequences (ST_{seq}) and ($best_{seq}$) and calculating the ratio of obtaining a number of states with greater values in the actual state sequence ST_{seq} than the best state sequence $best_{seq}$ would indicates the improvement ratio of the target and mutant vectors for better offspring.

Algorithm 4.2 Steps for building the HMM from DE information

Input $\leftarrow$ Population pop_m as a matrix of size $R \times C$

1: Convert pop_m to probability emission matrix E using Algorithm 4.1
2: Initialize T with low, high states
3: Sample random sequence of states ST_{seq} and emission symbols EM_{seq}
4: Evaluate the max likelihood of ST_{seq} and EM_{seq}
5: Calculate the best state sequence $best_{seq}$ that maximizes the likelihood of EM_{seq} using Viterbi algorithm [160, 161]
6: CR is computed as in equation 4.2
7: F is calculated as in equation 4.4

Output$\rightarrow$ F, CR

B. Computing mutation factor (F)

The F is an important factor in the core mutation operation of DE. It is essential for balancing exploration and exploitation and perturbing parents towards a new offspring in the preferred area of the search space [162] and we use the HMM posterior to automatically adapt it. .A dynamic F is measured by the minimum

HMM posterior of a particular state to designate a change in that state (low or high) in the population sequence with respect to the emission sequence (observation). The posterior distribution is used to make a Bayesian inference for exploring the subsequent search region, which is computed as the conditional probability of terminating the sequence at this particular state given the observed population sequence, as shown in Figure 4.3 . In more detail, the minimum posterior is calculated in the estimated state-posterior sequence (i.e., the minimum step size) and averaged with the pre-calculated CR values. The motivation for adapting the F, which is the ratio of the jump from a parent to its vectors generated from a mutation operator, is that the posterior in Bayes's theory is the probability of a random variable given its prior and likelihood as

$$posterior \approx likelihood \times prior \tag{4.3}$$

The CR is estimated by computing the likelihood ratios over the changes between ST_{seq} and $best_{seq}$ as prior information for employing the mutation operator. In DE, a new mutant vector is produced to explore various candidate solutions. The probability of movement between the two vectors (parent and mutant) can be assessed as the lower probability that occurs in the mutant vector for exploring more possible sub-regions based on all the preceding information in the original parents with the emission observations (EM_{seq}) and changes in their CR values as

$$F = Average\ (min_{posterior}|EM_{seq}, CR) \tag{4.4}$$

Because of the large number of vectors in the population, their average value is considered to obtain the expected movement rate for any one.

Algorithm 4.3 An iteration of the DE-HMM algorithm.

Define fitness function f; population size PS; population dimensions D; the maximum number of function evaluation Max_{Eval};
Set $G = 0$; $Feval = PS$; $F = CR = 0.5$;
$pop_0 \leftarrow rand()$
Evaluate pop_0 with f;
while $Feval \leq Max_{Eval}$ **do**
  for each $x_{i,G} \in pop$ **do**
    $v_{i,G} \leftarrow$ Mutation (Eq 2.1);
    $u_{i,G} \leftarrow$ Crossover (Eq 2.7);
    Evaluate $u_{i,G}$;
    Select($u_{i,G}$,$x_{i,G}$) (Eq 2.9);
  end for
  Sort(pop_G,ascending on fitness);
  Update F,CR for the sorted, successful vectors using HMM in Algorithm 4.2;
  $Feval$=$Feval$+PS;
end while

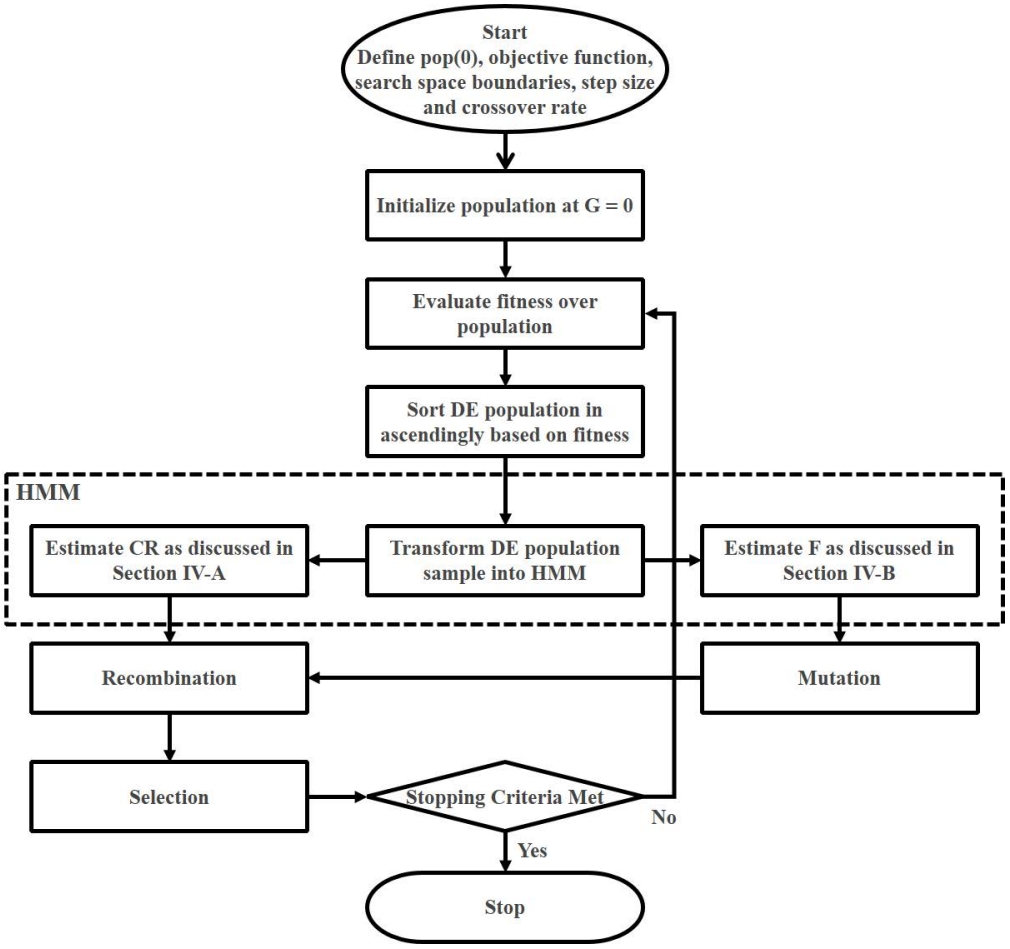


Figure 4.4: General flowchart of the proposed DE-HMM algorithm.

C. The complete architectural of DE-HMM

The DE-HMM architecture is presented in Algorithm 4.3, which describes the way of implementing the HMM's processes in the DE evolution procedure for controlling the two intrinsic parameters (F, CR), as previously discussed. The core difference between the classical DE and DE-HMM is the stage that uses the DE's sorted population to build the HMM, shown in Figure 4.4.

Firstly, a population of PS individuals is uniformly initialized in random way, with initially uses $F = 0.5$ and $CR = 0.5$. Subsequently, all the individuals are evaluated according to $f(x)$ producing the objective value, here represented by fitness value. Secondly, the mutation operation is employed at each generation via Equation 2.1 to generate the mutant individuals, followed by the crossover operation as we apply the basic binomial crossover scheme in Equation 2.7 to produce the trial individuals (offspring).

The procedure for updating the values of F and CR depends on how the DE population changes over time. In DE-HMM, the DE population is sorted in an ascending order and gets transformed into probabilities as shown in Algorithm 4.1 in order to fit the HMM parameters. Then random sequences of ST_{seq} states and EM_{seq} observations are sampled from the HMM.

The CR value is estimated by calculating the likelihood ratio between the actual state sequence ST_{seq} that is better than the best state sequence $best_{seq}$, generated using the Viterbi algorithm, as in Equation 4.1. This is derived from DE where CR in crossover operation determines the improvement ratio of the new offspring from the original and mutant vectors. Also, the F value is estimated based on averaging the minimum posterior (i.e., reflects the change of DE state) and the estimated CR to recognize the dependency between F and CR, as in Equation 4.4. Finally, each original individual is compared with its corresponding offspring and the individual with less fitness value survives to the next generation.

4.3. Experimental study

4.3.1. Benchmark problems

We use two benchmark test sets from the IEEE CEC2005 [163] and IEEE CEC2014 [164] special sessions on real-parameter optimization, which are mainly employed to test unconstrained optimisation algorithms, to evaluate the performance of our proposed algorithm. There is a total of 55 test functions with 10, 30 and 50 dimensions (D), as listed in Tables 4.1 and 4.2. They have different characteristics, such as uni- and multimodal test functions, which are labelled C1 ~ C25 for the 25 test functions from CEC2005 and F1 ~ F30 for the 30 from CEC2014. For further details of these benchmarks, refer to [163] and [164].

To demonstrate its effectiveness, the proposed DE-HMM algorithm is compared with the canonical DE that uses pre-defined F and CR values, other peer algorithms previously introduced to control the parameters and:

- Eight DE variants, four of which are commonly used in the literature and known as competitive algorithms and the other four the most recent variants that reflect the latest progress of DE; and
- Four non-DE variants frequently chosen to provide more comprehensive comparisons.

4.3.2. Experiments setup

DE-HMM was are coded in Matlab R2015a, and run on a PC with a 3.40 GHz TM-Core i7 processor, 16 GB RAM and Windows 7 with 64 bits. The PS value of DE-HMM to 60. The total number of function evaluations (FEs) is set to $10000D$, where D indicates a problem's dimensionality and each problem has 30

Table 4.1: CEC2005 Benchmark Functions.

	No.	Functions	Bounds	Fi*=Fi(x*)
"Unimodal Functions"	1	Shifted Sphere Function	[-100 , 100]	f_bias= -450
	2	Shifted Schwefel's Problem	[-100 , 100]	f_bias= -450
	3	Shifted Rotated High Conditioned Elliptic Function	[-100 , 100]	f_bias= -450
	4	Shifted Schwefel's Problem with Noise in Fitness	[-100 , 100]	f_bias= -450
	5	Schwefel's Problem with Global Optimum on Bounds	[-100 , 100]	f_bias= -310
"Basic Multimodal Functions"	6	Shifted Rosenbrock's Function	[-100 , 100]	f_bias= 390
	7	Shifted Rotated Griewank's Function without Bounds	Initialization range[0, 600]	f_bias= -180
	8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	[-32 , 32]	f_bias= -140
	9	Shifted Rastrigin's Function	[-5, 5]	f_bias= -330
	10	Shifted Rotated Rastrigin's Function	[-5, 5]	f_bias= -330
	11	Shifted Rotated Weierstrass Function	[-0.5, 0.5]	f_bias=90
	12	Schwefel's Problem	[-100 , 100]	f_bias= -460
"Expanded Multimodal Functions"	13	Shifted Expanded Griewank's plus Rosenbrock's Function	[-3 , 1]	f_bias= -130
	14	Shifted Rotated Expanded Scaffer's	[-100 , 100]	f_bias= -300
"Hybrid Composition Functions"	15	Hybrid Composition Function 1	[-5, 5]	f_bias= 120
	16	Rotated Hybrid Composition Function 1	[-5, 5]	f_bias= 120
	17	Rotated Hybrid Composition Function 2 with Noise in Fitness	[-5, 5]	f_bias= 120
	18	Rotated Hybrid Composition Function 2	[-5, 5]	f_bias= 10
	19	Rotated Hybrid Composition Function 2 with a Narrow Basin for the Global Optimum	[-5, 5]	f_bias= 10
	20	Rotated Hybrid Composition Function 2 with the Global Optimum on the Bounds	[-5, 5]	f_bias= 10
	21	Rotated Hybrid Composition Function 3	[-5, 5]	f_bias= 360
	22	Rotated Hybrid Composition Function 3 with High Condition Number Matrix	[-5, 5]	f_bias= 360
	23	Non-Continuous Rotated Hybrid Composition Function 3	[-5, 5]	f_bias= 360
	24	Rotated Hybrid Composition Function 4	[-5, 5]	f_bias= -473
	25	Rotated Hybrid Composition Function 4 without Bounds	Initialization range[-2, 5]	f_bias= -474

Table 4.2: CEC2014 Benchmark Functions.

	No.	Functions	Fi*=Fi(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 plus Rosenbrock's Function	1500
	16	Shifted and Rotated Expanded Scaffer's F6 Function	1600
"Hybrid Functions"	17	Hybrid Function 1 (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=$)	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

independent runs. We use the mean and standard deviation ($mean(std)$) statistical measures of the fitness error values ($f(x)$–$f(x*)$) to gauge deviations among the calculated and optimal results, with values less than 1.0E-08 treated as zero. Because of the random natures of stochastic algorithms (i.e., DE), we calculate the mean results over 30 runs to reflect an algorithm's performance instead of accepting a good solution in only one run. The detailed results obtained from all the algorithms are provided in the online appendices, with the best ones presented in boldface type and the number of best results ($No - Best$) for each corresponding algorithm for all problems are summarized at the end of the comparison tables.

Two non-parametric statistical hypothesis tests, the Friedman and Wilcoxon ones [165], are used to evaluate and assess the output from the DE-HMM algorithm which resembles those from the augmented algorithms using the well-known SPSS statistical tool [166] to execute them. The Friedman test ranks the algorithms' means while the Wilcoxon one evaluates the significance of performances of DE-HMM and other compared algorithms. In particular, the null hypothesis of these tests supposed that there is no significant difference between the mean error values of two samples while the alternative one attempts to determine if there is a significant difference using a 5% significance level. From the Friedman and Wilcoxon tests, three marks " + " or " − "or " = " are considered to reflect the significant difference in the average fitness values between DE-HMM and the peer algorithms, where " + " or " − " mean that DE-HMM is significantly better or worse than the competing algorithm, respectively while " = " means that no significant difference between the two algorithms. The critical difference (CD) of the Nemenyi post-hoc test [165] is required to detect the significant differences between the estimated ranks whereas the rejection of the null hypothesis is followed by a post-hoc analysis (one common test is Nemenyi) for assessing the significance between these algorithms. CD is calculated using equation 4.5 as discussed in [165], whereas q_α is the critical value at α significance level, listed in [165], which get estimated based on studenized range statistic divided by $\sqrt{2}$ and K and N refer to the number of compared algorithms and the number of tested problems,

Table 4.3: Critical values $q_{\alpha=0.05}$ for calculating CD of Nemenyi post-hoc test at different K .

$K =$	10	6	5	4
$q_{\alpha=0.05} =$	3.164	2.850	2.728	2.569

Table 4.4: F and CR values used for DE-HMM variants.

Variant	F posterior	CR likelihood ratio
DE-HMM-minG	min	current > best
DE-HMM-minL	min	current < best
DE-HMM-minE	min	current = best
DE-HMM-meanG	mean	current > best
DE-HMM-meanL	mean	current < best
DE-HMM-meanE	mean	current = best
DE-HMM-maxG	max	current > best
DE-HMM-maxL	max	current < best
DE-HMM-maxE	max	current = best

respectively. Since the significance level is 5%, the q_α for each group of comparison is shown in Table 4.3. The smaller CD means more significance differences can be detected than larger CD.

$$CD = q_\alpha \sqrt{k(k+1)/6N} \tag{4.5}$$

4.3.3. Experimental results

A. Analysis of performances of DE-HMM and its own variants

When DE is used to solve an optimization problem, a combination of different parameter settings with different learning strategies significantly influences its performance. We use the HMM as an adaptation method for automatically adjusting the two intrinsic parameters (F and CR) which are estimated by applying the two main components using the posterior of the HMM and likelihood ratio, respectively.

In this experiment, the different roles of the two components previously described are used to analyze their behaviours, with the best chosen as the better one for adaptation. We employ a case study using the 30D CEC2005 benchmark test problems to compare eight different selected parameters. The best choice of DE-HMM is denoted in this section by DE-HMM-minG to differentiate it from other variants and to indicate it uses minimum posterior and maximum fitness improvements. It is compared with a DE which has its parameters set randomly for each generation, namely Rand-gen. The comparison involves all variants and Rand-gen to allow for an appropriate test of significance to be used [165].

The eight DE-HMM variants are displayed in Table 4.4 in which the second column denotes the posterior movement of the next area, that is, the minimum, maximum or mean step size, and the third the number of candidate individuals that can be improved ($>$, $<$ or no improvement), also, compare DE-HMM against a DE with random control parameters that we call (Rand-gen). The complete results, including the mean function error values, presented in Tables A.1 and A.2 in Appendix A. As previously discussed, we use the Wilcoxon's and Friedman tests to summarize the performances of all the DE-HMM variants and Rand-gen for the 30D CEC2005 test problems, shown in Figure 4.5 and Table 4.5, respectively. In Figure 4.5, it is clear that our proposed DE-HMM algorithm with the minimum posterior and greater suggested improvement values achieves better average fitness values (i.e., fewer errors) for more problems than the other DE-HMM variants where "Better" means that DE-HMM outperforms each of the compared variants while "Worse" indicates that the competitor variant has better results relative to DE-HMM. Moreover, the significance difference is shown in the last column in Table A.3 in Appendix A based on the p-values as it shows that DE-HMM-minG is significantly better than all the DE-HMM variants. Friedman statistical analysis is reported in table 4.5, which demonstrates that DE-HMM attains the highest ranking. Since the p-value computed by the Friedman test is smaller than the significance level (0.05), the hypothesis that there is no significance difference

Table 4.5: Friedman statistical analysis of different variants of DE-HMM.

Method	30D
	Ranks
DE-HMM-minG "DE-HMM"	**2.42**
Rand-gen	3.2
DE-HMM-minL	3.62
DE-HMM-minE	4.18
DE-HMM-meanG	4.52
DE-HMM-meanL	5.16
DE-HMM-maxG	7.4
DE-HMM-meanE	7.94
DE-HMM-maxL	8.28
DE-HMM-maxE	8.28
"p-value"	**3.23E-26**
"Significance"	+
"CD"	2.709

in that 10 competing algorithms is rejected. The smaller CD value detects the significance difference in that test.

For further analysis, we select two test problems to act as typical ones with different characteristics (i.e., C2 and C12) to validate the potential variations in the ranges of their F and CR values through the evolutionary process.

They evidence the consequences of our choices for assessing the DE parameters. It can be seen that the DE-HMM-minG performs significantly better than the other eight, that is, it achieves better optimal solutions for the first two unimodal functions while DE-HMM-minL and DE-HMM-meanL obtain better ones for both C3 and C4, and C5, respectively. Furthermore, none of the eight variants can provide better results better than our DE-HMM for any other test functions. The main reason for this is that the proposed DE-HMM improves solutions by seeking a higher ratio while estimating the CR (i.e., CR $[0.7, 1]$) and the minimum posterior acts as the minimum step size when averaged with the configured CR (i.e., $F[0.3, 0.5]$), as preferred in the literature discussed in Chapter 2.

Regarding the ranges of the F and CR, two main observations are shown in Figures 4.6 and 4.7. The first is that the proper settings of the DE-HMM variants

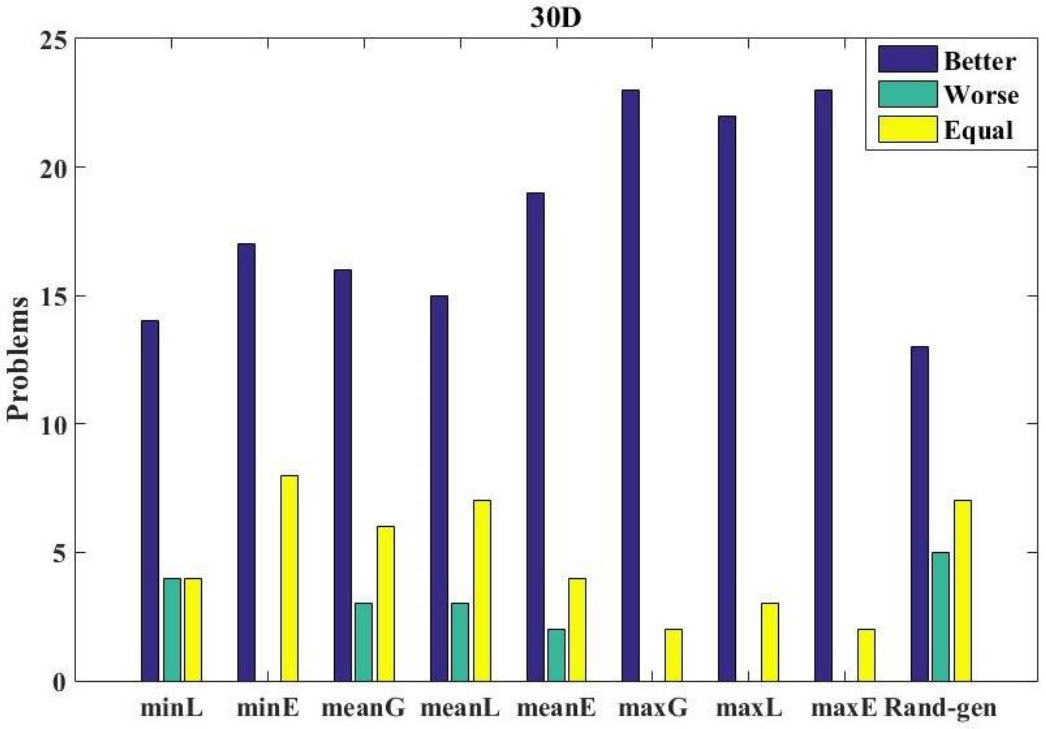


Figure 4.5: Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and its own variants.

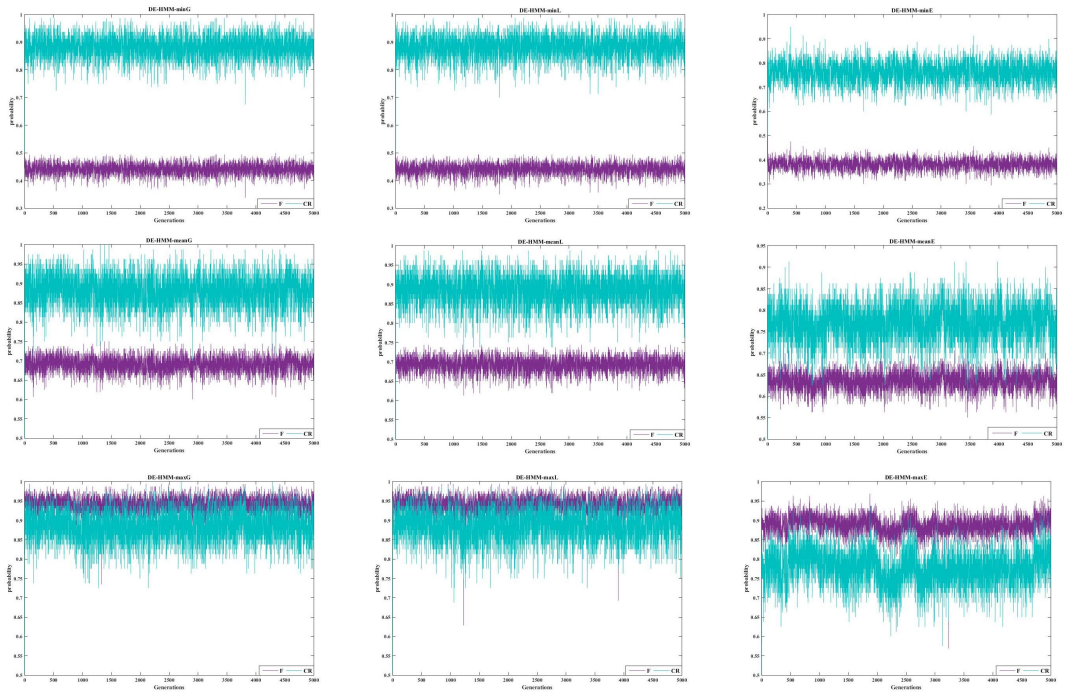


Figure 4.6: F and CR evolution of DE-HMM variants for C2

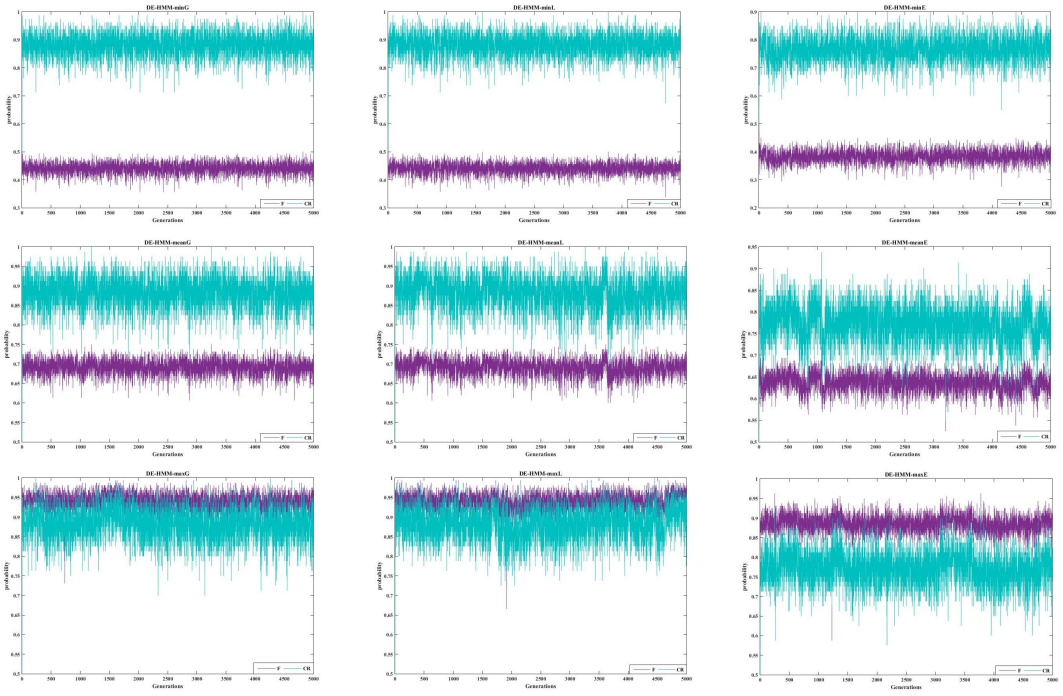


Figure 4.7: F and CR evolution of DE-HMM variants for C12

are changed for solving one optimisation problem; for instance, for C2 in Figure 4.6, the F value with the minimum posterior is in the range of $[0.3, 0.5]$ and the mean and maximum posterior within $[0.6, 0.75]$ and $[0.85, 1]$, respectively. Similarly, the CR for greater proportions of improvement has values within $[0.7, 1]$ while those for lower ones or no further improvement are within $[0.6, 0.9]$ or $[0.5, 1]$. Secondly, the DE-HMM exhibits the same behaviour when dynamically estimating the F and CR for solving different optimization problems (C2 and C12).

In summary, it is proven that our DE-HMM is superior to Rand-gen in terms of robustness and effectiveness because the chance of producing a good solution by choosing the F and CR in a random way is likely to be much lower. Conversely, choosing the appropriate F and CR probabilities through the evolutionary procedure using the proposed approach exhibits significantly improved results.

B. Performance analysis of DE-HMM

Based on the aforementioned analysis of the DE-HMM and all its variants, we determine that the best variant is DE-HMM-minG (i.e., that with the minimum posterior and greatest improvement). Therefore, it is compared with other state-of-the-art algorithms in the next sections to demonstrate its performance in terms of the quality of the solutions obtained.

- **Comparisons of DE-HMM and classical DEs performances**

Firstly, to validate its performance, our DE-HMM is compared with the canonical DE DE/rand, namely C-DE, which has its control parameters fixed as $PS = 100$, $F = 0.5$ and $CR = 0.9$, the detailed results for which are presented in Tables C.1, C.2, C.3 and C.4 in Appendix C. A summary of comparisons of these algorithms with respect to the results obtained from the Wilcoxon statistical test is depicted in Figure 4.8 in which it can be seen that our algorithm achieves superior performances for most of the problems used in the experiment and similar improvements for problems with higher dimensions using different datasets.

We extend our comparison to differ the crossover operator and mutation strategy in DE-HMM. The four DE-HMM distinct setups, shown in Table 4.6 where the first column shows the best strategy against binomial crossover shown in Equations 4.6 and 4.7, respectively. In Equation 4.6, $x_{best,G}$ is the best fitness vector chosen at generation G and F is the scaling factor, generated by DE-HMM method. l and$\prec l \succ_D$, in Equation 4.7 are the starting position, and the modulo function with modulus D, respectively. The second column shows the mutation strategies 'rand' versus 'best'. The third column refers to the crossover scheme binomial and exponential, used for generating offspring solutions.

$$v_{i,G} = x_{best,G} + F * (x_{r1,G} - x_{r1,G}) \tag{4.6}$$

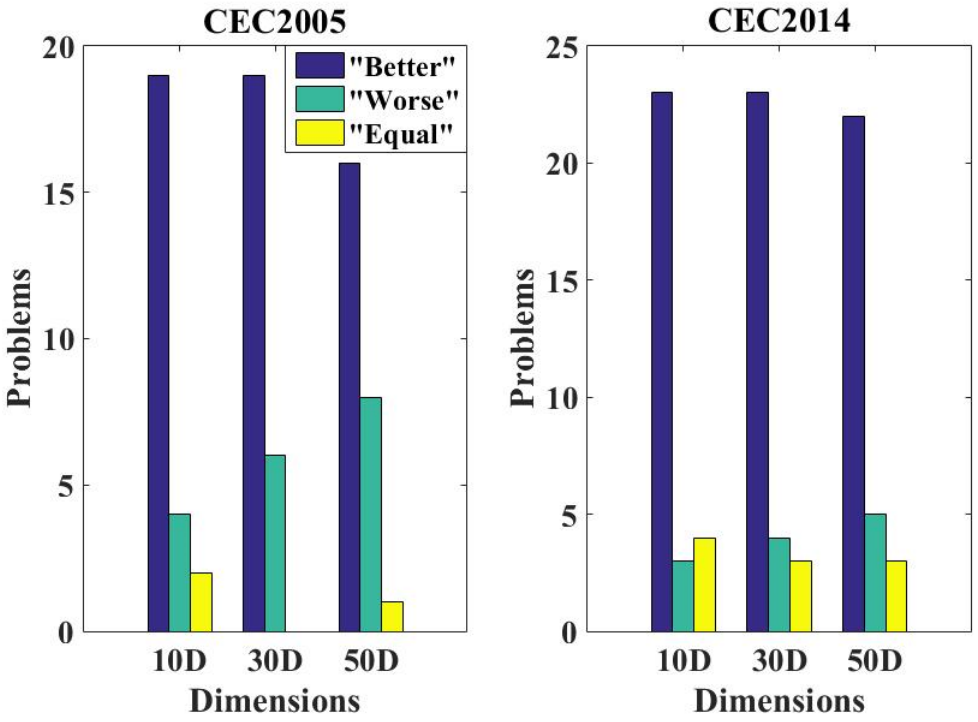


Figure 4.8: Comparison of numbers of problems with better or worse or equivalent performances achieved by DE-HMM and C-DE.

Table 4.6: Description of the four mutation and crossover setups used in DE-HMM analysis.

	Mutation strategy	Crossover scheme
DE-HMM "rand/bin"	rand	binomial
DE-HMM “rand/exp”	rand	exponential
DE-HMM “best/bin”	best	binomial
DE-HMM “best/exp”	best	exponential

$$u_{i,G} = \begin{cases} v_{i.G} & \forall j = \prec l \succ_D, \prec l+1 \succ_D, ..., \prec l+L-1 \succ_D \\ x_{i,G} & otherwise \end{cases} \tag{4.7}$$

We use the CEC2005 and CEC2014 benchmark functions at 30 dimensions to assess their performances in terms of the obtained solution quality, where the detailed results are reported in Tables B.1 and B.2 in Appendix B, respectively. The Wilcoxon statistical results for the four setup are shown in Tables 4.7 and

Table 4.7: Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2005 test functions.

DE-HMM "rand/bin" vs.	"Better"	"Worse"	"Equal"	"p-value"	"Significance"
DE-HMM/rand/exp	18	2	5	**0.005**	+
DE-HMM/best/bin	12	3	10	0.11	=
DE-HMM/best/exp	14	3	8	**0.0125**	+

Table 4.8: Comparison summary between DE-HMM with rand and binomial operators versus other setups at 30D CEC2014 test functions.

DE-HMM "rand/bin" vs.	"Better"	"Worse"	"Equal"	"p-value"	"Significance"
DE-HMM/rand/exp	21	5	4	**0.001**	+
DE-HMM/best/bin	15	9	6	0.155	=
DE-HMM/best/exp	16	10	4	0.146	=

4.8 for both CEC2005 and CEC2014 test set, respectively. In Table 4.7, DE-HMM employing best strategy with binomial crossover achieves better results in three unimodal and one multimodal functions while using best and exponential operators get optimal results of two unimodal functions. Nevertheless, DE-HMM using rand strategy and binomial crossover is better for the majority of time. Moreover, for CEC2014 results in Table 4.8, DE-HMM with rand and binomial crossover operators is superior to the other variants.

- **Comparisons of DE-HMM and the state-of-the-art variants**

Veček et.al [167] stated that comparisons in the literature should be made either against parameter settings that has been tuned using a parameter tuning algorithm, or against algorithms with abilities for parameter control. Given that our proposed method falls in the second category, we have compared against similar algorithms.

The DE-HMM is compared with some state-of-the-art control algorithms containing four DE variants (i.e., SaDE [42], JADE [60], jDE [12] and CoBiDE [52]) and four non-DE EAs (i.e. BNGA [55], CLPSO [168], CMA-ES [169] and IPOP-CMA-ES [170]), procedures which are regularly used for comparison in the literature. IPOP-CMA-ES was the champion of the CEC2005 competition. Note that

the results of BNGA are presented for 10D only because the original paper only reported for 10D.

It is clear that the DE-HMM, JADE and CoBiDE produce significant results better than SaDE and jDE and, particularly, DE-HMM provides more substantial improvements for many test problems. It also produces promising results for various problem sizes, as shown in Table 4.9, which presents the algorithms' ranks determined by the Friedman test, reporting the p-values at 10, 30, 50 dimensions and the CD value for this comparison results. From that table, it is observed that there are significance differences for all assessed dimensions based on the p-values as well as the calculated CD value.

For the 10D problems, similar to JADE, DE-HMM obtains the optimal solutions for all the unimodal functions except C4 and can proficiently solve most of the multimodal and hybrid test problems. The DE-HMM also shows superior outcomes for most functions, particularly those with multimodalities and complex hybrid compositions for the 30D problems. Nevertheless, JADE is the best algorithm for solving the unimodal functions and produces optimal results for C9, as do jDE and CoBiDE. For the 50D problems, the DE-HMM is still the best algorithm, followed by CoBiDE. Overall, the performances of DE-HMM are better than or competitive with the other algorithms as its way of estimating the F and CR parameters basically analyses the potential dynamics in the DE evolutionary process. The computational results obtained from these experiments are presented in Appendix D.

The DE-HMM is also compared with the non-DE variants, as detailed in Appendix D, and succeeds in obtaining better results for the majority of the functions tested. DE-HMM obtains the best results for most of the unimodal functions with 10D and 30D similar to CMA-ES while CMA-ES can achieve better results at

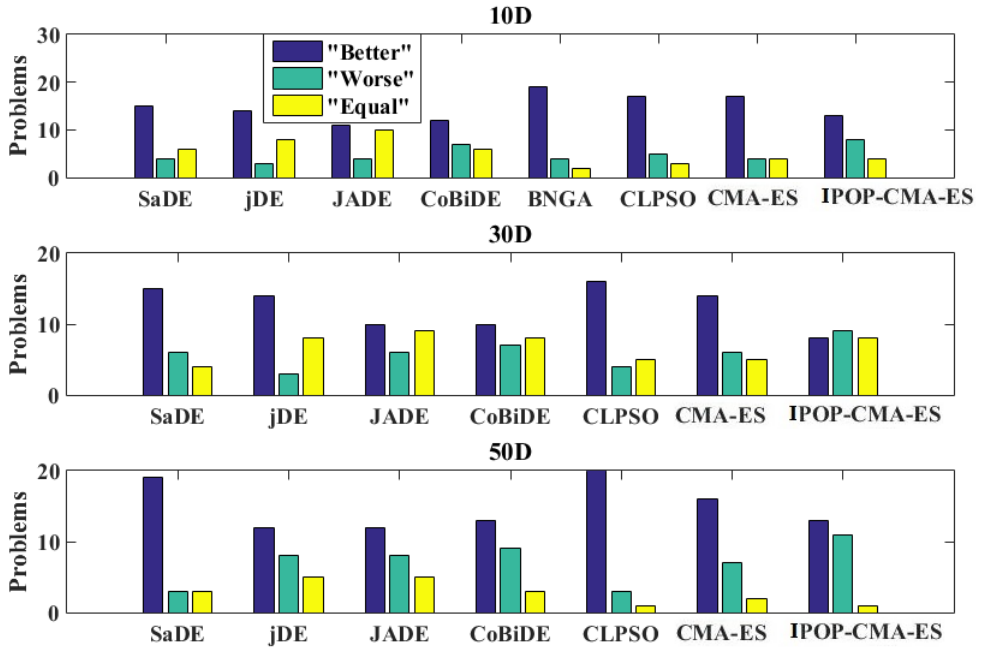


Figure 4.9: Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and all the state-of-art variants

Table 4.9: Rankings obtained from Friedman's test of 10, 30, and 50 dimensional CEC2005 functions of DE-HMM and its DE variants.

Method	Ranks		
	10D	30D	50D
DE-HMM	**2.56**	**2.72**	**2.80**
CoBiDE	2.60	2.74	2.90
JADE	3.32	3.12	3.16
jDE	3.52	4.04	3.52
SaDE	4.10	3.88	4.56
C-DE	4.90	4.50	4.06
"p-value"	**7.75e-07**	**3.12e-04**	**0.002**
"Significance"	+	+	+
"CD"	1.508		

50D. Considering the general trends of these algorithms for the multimodal problems, DE-HMM and IPOP-CMA-ES achieve good results for the 10D, 30D and 50D ones. For the group of functions with complex compositions, the DE- HMM clearly demonstrates better performances than the other algorithms.

The statistical analysis by Friedman test confirms the superiority of the DE-HMM over the non-DE algorithms for the 10D, 30D and 50D test functions, as shown in Table 4.10, which includes the average ranking for each competing algorithm, p-values at each dimensions and the CD value, all show that DE-HMM has the highest ranking and the small p-values and CD reject the hypothesis that there is no significance difference between the algorithms. Moreover, the Wilcoxon statistical test provides results that demonstrate the significant performances of the DE-HMM as it obtains greater number of better results than the other algorithms, as shown in Figure 4.9. More specifically, Table D.13 in Appendix D shows test of significance results and demonstrate that DE-HMM is significantly better than all competing algorithms (except JADE and CoBiDE), based on the average results obtained for all test problems at 10D and 30D problems. For JADE and CoBiDE, DE-HMM offers significantly better solutions for 10D test problems and equivalent performance for 30D and 50D. DE-HMM in Table D.13 achieves significance results better than the augmented non-DE variants at 10D problems whilst there is no significance difference for the other problems' dimensionality. Table 4.10 shows that DE-HMM still has the best average ranking compared with non-DE variants, whereas the CD is calculated at 10D and (30 and 50D) for 5 and 4 competing algorithms, respectively.

To further illustrate the DE-HMM's performance compared with those of the state-of-the-art-techniques, convergence graphs of the different algorithms for 6 test problems (F1, F2, F4, F11, F14 and F16) are presented in Figure 4.10 in which the x-axis refers to evolutions of generations (within the same function evaluations) and the y-axis the median function error values over 30 independent runs. It is clear that our DE-HMM algorithm is able to converge to better solutions

Table 4.10: Rankings obtained from Friedman's test of 10, 30, and 50 dimensional CEC2005 functions of DE-HMM and its non-DE variants.

Method	Ranks		
	10D	30D	50D
DE-HMM	**2.10**	**2.12**	**1.92**
IPOP-CMA-ES	2.42	2.16	2.40
CLPSO	2.92	2.84	2.56
BNGA	3.74	-	-
CMAES	3.82	2.90	3.12
"p-value"	**3.39E-05**	**0.016**	**0.008**
"Significance"	+	+	+
"CD"	1.21	0.938	

for most test problems. From the graphs, one can see that HMM works well on F11, F14, and F16 and not on F1, F2, and F4, this could be because The greedy mutation strategy, i.e., current-to-pbest/1in JADE algorithm results in fast convergence speed and high convergence precision for unimodal functions. For example, all algorithms almost perform similar performance in F1, the only difference in the generation period to obtain these results. JADE is the best in F2 as mentioned because of its greedy strategy while the difference between DE-HMM and CoBiDE is slight. The same behaviour for F4 as DE-HMM is the second best after JADE algorithm. For multi-modal function such as F11, F14, and F16, the best performing algorithm is DE- HMM as it can explore better regions in such complex fitness landscapes.

- **Performance evaluations of DE-HMM versus recent DE variants**

In this subsection, we employ the four DE variants most recently developed to improve the performance of DE (i.e., CPI-DE [61], TSDE [148], LSHADE [38] and UMOEA [48]) for comparison with our proposed DE-HMM. The reason for choosing these algorithms is that they all have either been published in the proceedings of CEC2014 and/or recently proposed to imitate the current progress of DE. This

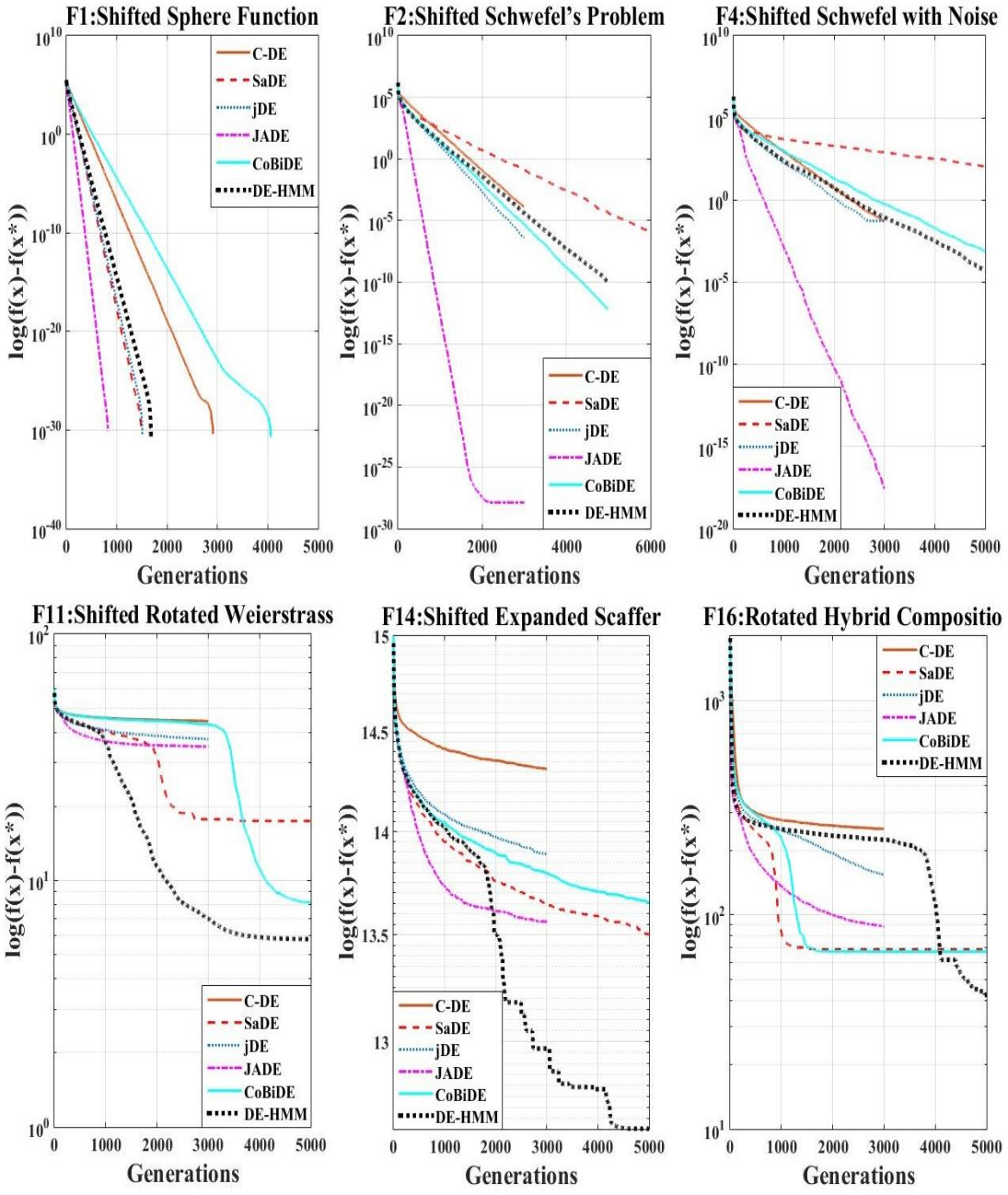


Figure 4.10: Convergence plots of DE-HMM on F1, F2, F11, F14, and F16 with 30 dimensions where y-axis in log scale.

experiment is conducted for 30 functions with 10, 30 and 50 decision variables over 30 independent runs, with their computational outcomes reported in Appendix E.

It is obvious in Table E.1 in Appendix E that both the DE-HMM and other compared variants perform well for the 10D problems, especially the unimodal and multimodal functions and most hybrid and composition instances. It reaches optimal solutions for two multimodal functions (F6 and F8) and is considered the best of the peer algorithms for 6 functions while the LSHADE algorithm achieves the best results for 4 and UMOEA outperforms the others for 3. Table E.3 in Appendix E recaps the experimental results for the 30D problems for which UMOEA, LSHADE and TSDE obtain optimal solutions for 3, 2 and 1 unimodal functions, respectively, while DE-HMM's are close to the optimal solution for F3. The DE-HMM is able to find the optimal solution for the multimodal F7 and better results for 3 functions as well as competitive results for the other hybrid and composition ones. The results for the 50D problems show that DE-HMM exhibits superior performances for the majority of multimodal, hybrid and composition functions.

Finally, as a further comparison, the Wilcoxon and Friedman tests are conducted, to detect the significance difference of these algorithms, and their results shown in Figure 4.11 and Table 4.11, respectively, for all the dimensions studied in our experiment. In conclusion, the proposed DE-HMM yields significantly better results than the four CPI-DE for 10D and 50D, while it is significantly better than TSDE, LSHADE, and UMOEA for 30D, 50D, and 10D, respectively, as illustrated in Table E.7 in Appendix E.

4.3.4. Time Complexity

This section describes the time complexity for comparing DE-HMM against other algorithms (either compared based on CEC2005 or CEC2014 benchmark datasets). The computational time is measured as defined in [163, 164] for CEC2005 and

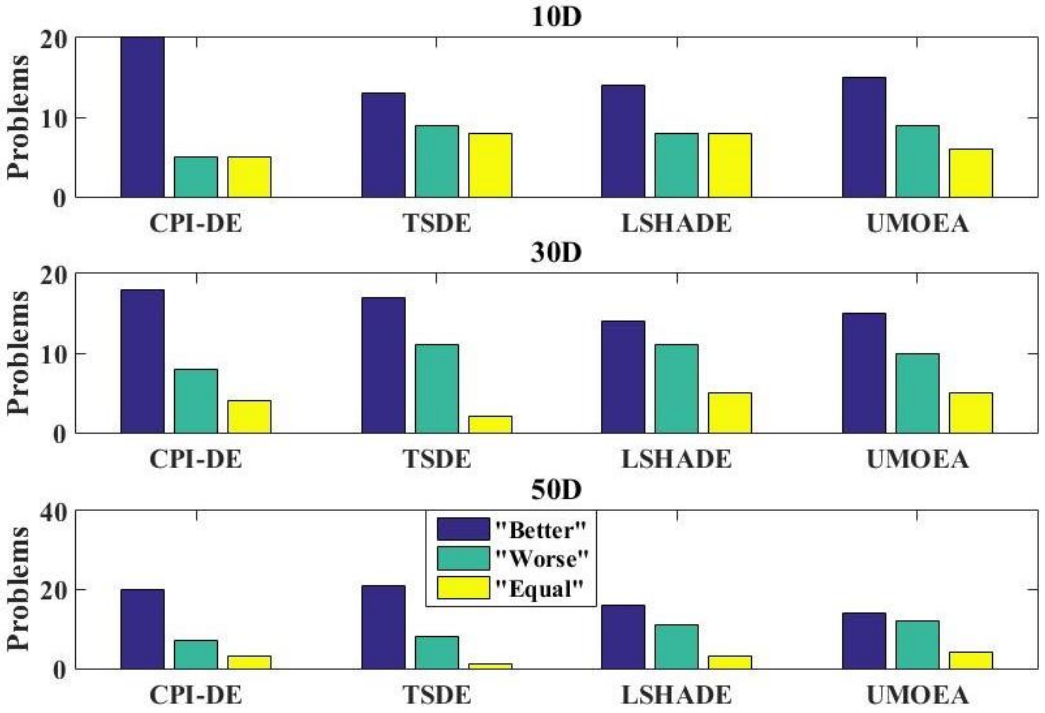


Figure 4.11: Comparison of numbers of problems with better or worse or equivalent performance achieved by DE-HMM and the up-to-date DE variants.

Table 4.11: Rankings obtained from Friedman's test of 10, 30 dimensional CEC2014 functions of DE-HMM and its up-to-date variants.

Method	Ranks		
	10D	30D	50D
DE-HMM	**2.67**	**2.78**	**2.67**
LSHADE	2.82	2.90	2.72
TSDE	2.83	3.68	4.22
UMOEA	3.43	3.08	2.78
C-DE	4.55	4.10	4.38
CPI-DE	4.70	4.45	4.23
"p-value"	**2.64e-08**	**4.50e-04**	**5.57e-05**
"Significance"	+	+	+
"CD"	1.376		

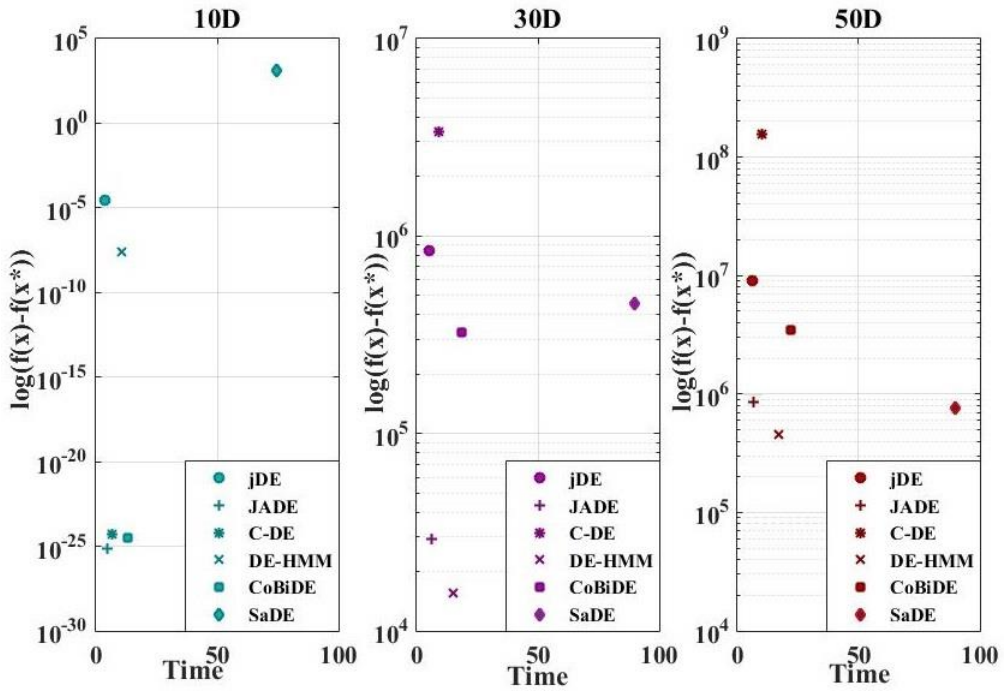


Figure 4.12: Time complexity for C3 of CEC 2005 test problems at 10, 30, and 50 dimensions.

CEC2014, respectively. The comparisons are reported in Tables F.2 and F.3 for CEC2005 and CEC2014 benchmark datasets, accordingly. We employ the two standard functions used in the literature for this comparison.

Two plots are drawn to depict the performance of each algorithm in terms of the quality of the obtained solution and the time consumed for achieving it. Firstly, Figure 4.13 shows the trade off between the effectiveness and efficiency of DE-HMM and C-DE and DE variants, use the CEC2005 test set in the previous comparison. Function 3 is employed as defined in [163] at different dimensions (10, 30, and 50 dimensions). From this figure, it is clear that DE-HMM could consume more time than C-DE, JADE, and jDE but it achieves less error (better result) than them at 30D and 50D. In regard to the 10D results, although C-DE, jDE, and JADE are better in terms of time and quality for 10D, the difference when compared to the performance of DE-HMM is negligible.

Secondly, we test the time consumed for F18 in CEC2014 dataset as defined

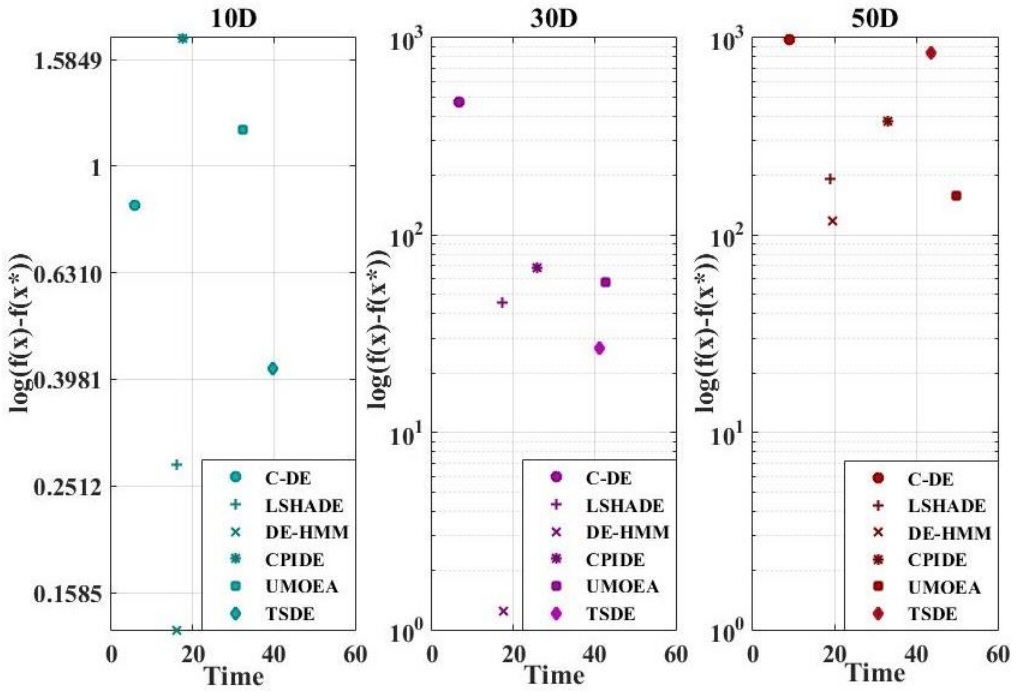


Figure 4.13: Time complexity for F18 of CEC 2014 test problems at 10, 30, and 50 dimensions.

in [164] to evaluate the performance of the proposed DE-HMM algorithm and up-to-date DE variants. The comparisons are described in Figure 4.13, where DE-HMM performs better in terms of quality of solutions obtained, albeit with a slight increase in computational cost when compared to classical DE and LSHADE algorithms.

4.4. Chapter summary

Setting the intrinsic parameters of a DE stochastic algorithm has a great effect on the efficiency and effectiveness of the obtained solutions, with less processing time required to adjust them through the evolutionary process than tune them manually using a trial and error approach.

In this chapter, a combination of the HMM procedure and classical DE process, the DE-HMM self-adaptive parameter control method for effectively adapting the F and CR control parameters considering the three main issues mentioned in Chapter 2 (i.e., problem and evolution dependencies, and the complex interactions between the F and CR) is discussed. It improves DE's performance as better solutions are obtained in less computational time than required by other adaptive/self-adaptive algorithms.

In DE-HMM method, to assign the F and CR values during the evolutionary process, the HMM posterior and likelihood ratios are calculated, as inspired by the usefulness of the HMM and its similar way of operating to DE as both change naturally over time. Experiments were conducted using two groups of benchmark problems comprising 55 test functions (i.e., IEEE CEC2005 and CEC2014) with different dimensions (i.e., 10, 30 and 50) to demonstrate the DE-HMM algorithm's performance in terms of the quality of solutions obtained. Also, the computational times for different dimensions were calculated.

Subsequently, DE-HMM was compared with different algorithms, the classical DE and other competitive ones with four DE and four non-DE variants, and also other recent state-of-the-art ones using two non-parametric statistical tests to analyze and demonstrate its overall outstanding performance.

Based on the results, DE-HMM can obtain better solutions than the other algorithms for different problems even with increased number of dimensions.

Chapter 5

Two Strategies DE-HMM Combined with Local Search for Constrained Optimization Problems

In dealing with constrained problems (COPs), this chapter introduces a DE framework that extends the DE-HMM self- adaptation method discussed in Chapter 4. This framework incorporates DE-HMM with a local search method to solve various types of COPs. The benchmark dataset, experimental settings and computational results and analysis are provided. The results are compared with other state-of-art algorithms, demonstrating the effectiveness of the proposed method in solving COPs.

5.1. Introduction

Most real world optimization problems in various fields like engineering and computer science frequently involve additional requirements on the variables and/or objective functions, which comprise " constraints". These problems are termed as constrained optimization problems (COPs) [19]. In COPs, the objective function is minimized or maximized subject to certain constraints (such as cost, geometry, sequence, etc.) [19]-[84]. Evolutionary algorithms (EAs) have proved their potential success in solving COPs over the last few decades; including Differential Evolution (DE), which is one of the highly competitive EA [91]. Handling any

COP requires two main elements; the search mechanism and constraint handling techniques (CHTs) [78]. The search performance primarily depends on these elements. The main purpose of DE here is to adjust the exploration and exploitation of the evolving population, achieving promising performance while CHTs are introduced to find the best way of handling the constraints during the evolution process.

As previously discussed in Chapter 2, DE, as a simple and highly competitive EAs, has the characteristics of global direct search, including three main operators (i.e., mutation, crossover and selection) and their associated control parameters (i.e., scaling factor F, crossover rate CR and population size PS) [91] for finding the optimal solution. However, DE still needs specialized mechanisms (CHTs) to discover both feasible and optimal solutions. The main operator in DE is mutation, where the original parents are selected randomly to generate new solutions, improving the exploration of the search space. In the literature, choosing the search operators and control parameters have proven to be a complex task due to the variability and complexity of COPs characteristics [16]. In other words, no specialized processes, operators or parameters are suitable for all problems, or even for a problem at hand.

I propose the idea of augment DE with different operators and/or different processes. This idea is inspired from memetic algorithms (MAs) [171, 172] research. Qin et al. [171] state that the MAs are often more competitive than the global search methods for solving complicated optimization problems. Also, their performance is sensitive to the frequency and the strength of the local search, how it operates and the interaction between the global and local search operator.

In this study, a local search is combined with the basic DE search, as a specialized search operator, representing two stages, each with their own mutation strategy and its F and CR based on the ongoing dynamics in the search process. One strategy is to enhance the exploration abilities and the other is to accelerate

the convergence towards the optimal solution. In addition, the HMM model is employed for automatically adjusting F and CR parameters, as discussed in Chapter 4, to tackle the parameter control issue.

This research focuses on a general framework that capitalizes on the power of local search to guide and accelerate the search convergence for the feasible regions as well as the strength of mutation characteristics that fit each stage. Furthermore, the application of HMM for adjusting F and CR parameters is extended to COPs. The resultant framework is referred to MS-DEHMM-L.

The proposed framework is evaluated using a well studied constrained benchmark suite, which was introduced in the CEC2010 IEEE competition [173]. The results obtained confirm the effectiveness of the MS-DEHMM-L framework in solving COPs compared to some of the state-of-art constrained algorithms.

This chapter is structured as follows: section 2 discusses the algorithmic framework components; the experimental study, including details of the benchmark data set, configuration used and results obtained are provided in section 3; and, finally, this chapter is concluded in section 4.

5.2. Algorithmic Framework of MS-DEHMM-L

There is no guarantee that a single search procedure or operator would be suitable for all problem types, as it could perform best for a class of problems but its behaviour is not satisfactory for another class of problems, especially for COPs. This is formally recognized in the literature as the No Free Lunch theorem [10]. Therefore, the focus in this chapter is to determine the appropriate search method, either global or local search, through the evolution procedure based on the performance of such population's individuals regards to the feasibility and diversity ratios. For each generation, the feasibility and diversity ratios are calculated of the entire population for deciding which search procedure should be employed? Then,

different mutation strategies and control parameters are assigned to each process according to their nature. The constraints are managed using the multi-objective optimization concepts, detailed below.

Three major components are incorporated in the proposed framework to solve COPs. Firstly, the local search method, which aims to explore and guide the search towards the feasible regions instead of focusing on guiding the whole population to optimal solution using the global search only. Therefore, the proper balance between finding the best solution and maintaining population diversity (i.e., exploitation and exploration) can be achieved by integrating the local search within the main DE procedure as two search stages; local for exploration and global for exploitation. The second component is to set the F and CR parameters dynamically during the evolutionary process, which is accomplished by using HMM as an extension of the work in Chapter 4. The last one is to assign two different strategies and estimated F and CR values for the local and global stages' processes.

5.2.1. Constrained Handling

As reviewed in Chapter 2, there are two types of constraints, equality and inequality in which the equality constraints are transformed into inequalities using a smaller tolerance δ. In this study, an approach based on multi-objective optimization concepts [78] is implemented as CHT in which the constraint violations $G(x)$ are treated as an additional objective besides the objective function (i.e., fitness function $f(x)$), calculated as equation 5.1

$$G(\overrightarrow{x}) = \begin{cases} max\{0, g_i(\overrightarrow{x})\} & i = 1, 2, ...IQ \\ max\{0, |h_j(\overrightarrow{x})| - \delta\} & j = 1, 2, ...EQ \end{cases} \tag{5.1}$$

$$G(\overrightarrow{x}) = \sum_{i=1}^{IQ} G_i(\overrightarrow{x}) + \sum_{j=1}^{EQ} G_j(\overrightarrow{x}) \tag{5.2}$$

where $\overrightarrow{x} = [x_0, x_1, ...x_D]$, D is the number of dimensions, each vector has $g_i(\overrightarrow{x})$ and $h_j(\overrightarrow{x})$ inequality and equality constraints, thus $G(\overrightarrow{x})$ is considered as the summation of constraint violations for the inequality and equality constraints as in equation 5.2.

Two objectives are considered to select better individuals to survive in the next generation. The Pareto-dominance ($\prec$) is usually used in this CH category (i.e., multi-objective optimization) for comparing and selecting the desired coming solutions [19, 78] whereas the objective function $f(x)$ and the constraint violations $G(x)$ are considered as two conflicting objectives to be achieved. To explain the Pareto-dominance concept, assume that there is a population *pop* with solutions $x1$ and $x2$. x1 is considered to dominate x2 if two conditions are satisfied, shown in equation 5.3

c1) if the values of $f(x1)$ and $G(x1)$ are less than or equal to the values of $f(x2)$ and $G(x2)$, respectively ($x1$ is not worse than $x2$).

c2) if the value of $f(x1)$ is less than to the value of $f(x2)$ or the value of $G(x1)$ is less than to the value of $G(x2)$ ($x1$ is better than $x2$ for one objective at least).

$$x1 \prec x2 \longleftarrow if \begin{cases} c1: & f(x1) \leq f(x2) \,\&\, G(x1) \leq G(x2) \\ c2: & f(x1) < f(x2) \,||\, G(x1) < G(x2) \end{cases} \tag{5.3}$$

In particular, $x1$ is chosen for the next generation if $x1$ Pareto dominates $x2$ (achieves c1 and c2). However, if $x1$ and $x2$ cannot hold the Pareto-dominance between them, then they are called non-dominated with each other. The same for $x2$ with respect to $x1$, the former can be survived to the next generation if it Pareto dominates to $x1$.

5.2.2. HMM for control parameter setting (F and CR)

In this section, the DE parameter control self-adaptive methodology proposed in Chapter 4 is extended. As mentioned in the previous chapter, HMM has three inputs to be defined from the DE internal information: the state transition matrix (T), the emission matrix (E) and the initial state probabilities (π). HMM defines two states to reflect the status of DE individuals as feasible or infeasible states, with initial values of 0.1 and 0.9, respectively because the infeasible solutions have are likely to be higher in proportion in the start of the search. The emission matrix including the population observations is transformed into probabilities as in Algorithm 4.1 to reveal the interdependencies in the evolutionary process, representing the HMM likelihood. According to the ongoing process, either global or local process, the population individuals are sorted regarding its target, either obtaining the best solution or achieving the feasible region, respectively. For example, the population is sorted ascendingly according to the constraint violations in the local search process as it focuses on reaching the feasible region while the global search focuses on achieving the optimal solution so that its individuals are sorted in ascending order according to the fitness function value.

CR is calculated by computing the likelihood proportion between the current state sequence and the best state sequence by the Viterbi algorithm [161] and F is calculated from the posterior probability of a certain state with respect to the observation sequence (population). Two different values for F and CR are computed for the two processes' stages based on their procedures. Since the local search guides the population to the feasible region from different directions, F and CR should be relatively large to preserve the population diversity while exploring more promising eras. However, they should be slightly smaller for the basic DE search as the main target in that case is to locate the optimal solution.

The updated steps of estimating both F, CR for the two processes are described in Algorithm 5.1 and illustrated in detail in the previous chapter.

Algorithm 5.1 Updated steps of building the HMM from constrained DE problems' information

Input ← Population pop_m as a matrix of size $R \times C$

1: Transform pop_m to probability emission matrix B using Algorithm 4.1
2: Initialize A with feas, infeas states
3: Sample random sequence of states ST_{seq} and emission symbols EM_{seq}
4: Evaluate the max likelihood of ST_{seq} and EM_{seq}
5: Calculate the best state sequence $best_{seq}$ that maximizes the likelihood of EM_{seq} using Viterbi algorithm [15, 16]
6: $CR_L = Likelihoodratio\ (ST_{seq} \geq best_{seq})$
7: $CR_G = Likelihoodratio\ (ST_{seq} \leq best_{seq})$
8: $F_L = Average\ (max_{posterior}|EM_{seq}, CR_L)$
9: $F_G = Average\ (min_{posterior}|EM_{seq}, CR_G)$

Output→ F_L, CR_L,F_G, CR_G

The MS-DEHMM-L mainly estimates the F and CR control parameters based on the evolution changes for the two stages' processes. The core difference between the DE-HMM and MS-DEHMM-L regading computing the F and CR is that the latter computes two values for each of F and CR according to the entire process, described in Algorithm 5.1. Initially, the population observation is transformed into probabilities using Algorithm 4.1 to achieve the HMM requirements, then two sequences are randomly sampled ST_{seq} and EM_{seq}, representing the sequence of states and observations, respectively from HMM.

CR value is calculated by estimating the likelihood ratio between the actual state sequence ST_{seq} that is better or worse than the best state sequence $best_{seq}$ (produced from the Viterbi algorithm) for local and global search stages, respectively. Since CR in the crossover operator identifies the improvement in the new offspring from the parent and mutant vectors, the local search needs high change in the offspring (inherit little information from the parent one) but the global search here can accelerate the convergence by getting more information from the original vectors.

F value is calculated using two values; the posterior probability and the estimated CR, therefore, two values are estimated for the local and global search

stages. A higher probability of change/step is needed in the local search procedure for distributing the mutant vector widely while the jumping step of F in the global search should be smaller as this search process focuses on finding the optimal solution so exploring the most possible individuals is essential. Therefore, F is estimated by averaging the posterior probability (that reflects evolutionary changes) and the pre-estimated CR to realize the F and CR dependencies, as discussed before in Chapter 4.

5.2.3. DE combined with local search

In this section, the optimization process is classified into two different stages; global and local DE processes. The main search process focuses on exploring the whole search space and obtaining the best individual and this could lead to continuing the evolution until finding an individual with better performance. It could be time-consuming either by exploring more and more regions when the best solution is located in small region or in case of no new solutions having better performance than their parents for large number of generations.

To tackle these issues, the evolutionary process is divided into two processes based on two factors: the feasibility ratio FP [78] and the diversity ratio φ [174, 175] of the population, computed as follows.

$$FP = \frac{No\_Feas\_Sols}{PS} \tag{5.4}$$

$$\varphi = 1 - \left| \frac{s_{avg} - s_{best}}{s_{worst} - s_{best}} \right| \tag{5.5}$$

$$L\_firing = \begin{cases} 1 & if\ rand(0,1) < 1 - FP \vee rand(0,1) < \varphi \\ 0 & otherwise \end{cases} \tag{5.6}$$

where No_Feas_Sols is the total number of feasible solutions per generation and PS is the population size. In equation 5.5, s_{avg} , s_{best} and s_{worst} are the average, best and worst fitness values of the population solutions.

At the beginning of the evolutionary process, the majority of the population are infeasible solutions and the population diversity still high. Therefore, the offspring could be infeasible or nondominated with its parent individual, which could affect on the population so as to stagnate in the infeasible region and cannot progress in the search process. To address and tackle these problems, the local search is used for locally exploring a solution's neighbourhood, while approaching the feasible region. During the local search procedure (steps 5-10 in algorithm 5.2), the population *pop* is divided into a number of subpopulations with equal sizes (assume that subpopulation with slight large size is better). Using large size for the subpopulation is preferred as it would contain both feasible and infeasible individuals with higher probability than those with small subpopulation, which increases the chance of finding the desired optimal solution in the feasible region. The subpopulations are evaluated by the fitness and constraint functions and then mutation and crossover are invoked for generating new offspring. After obtaining the new offspring populations, the selection comparison using the Pareto-dominance criteria is applied to replace the non-dominated individuals by dominated ones so that each subpopulation can motivate the search towards the feasible region. Finally, all subpopulations are combined to gather the whole population. The steps from 5 to 10 keep executing along the evolution until the population converge to the feasible region and there are enough feasible solutions with low diversity (L_firing is not satisfied). In this stage of the search, the probability of applying the global process becomes higher than local one for improving the solution quality until finding the optimal solution.

In other words, The local searcher operates when most of the population individuals are infeasible and the diversity still high, therefore, it is essential to use the local search method for locally exploring the neighbourhood approaching

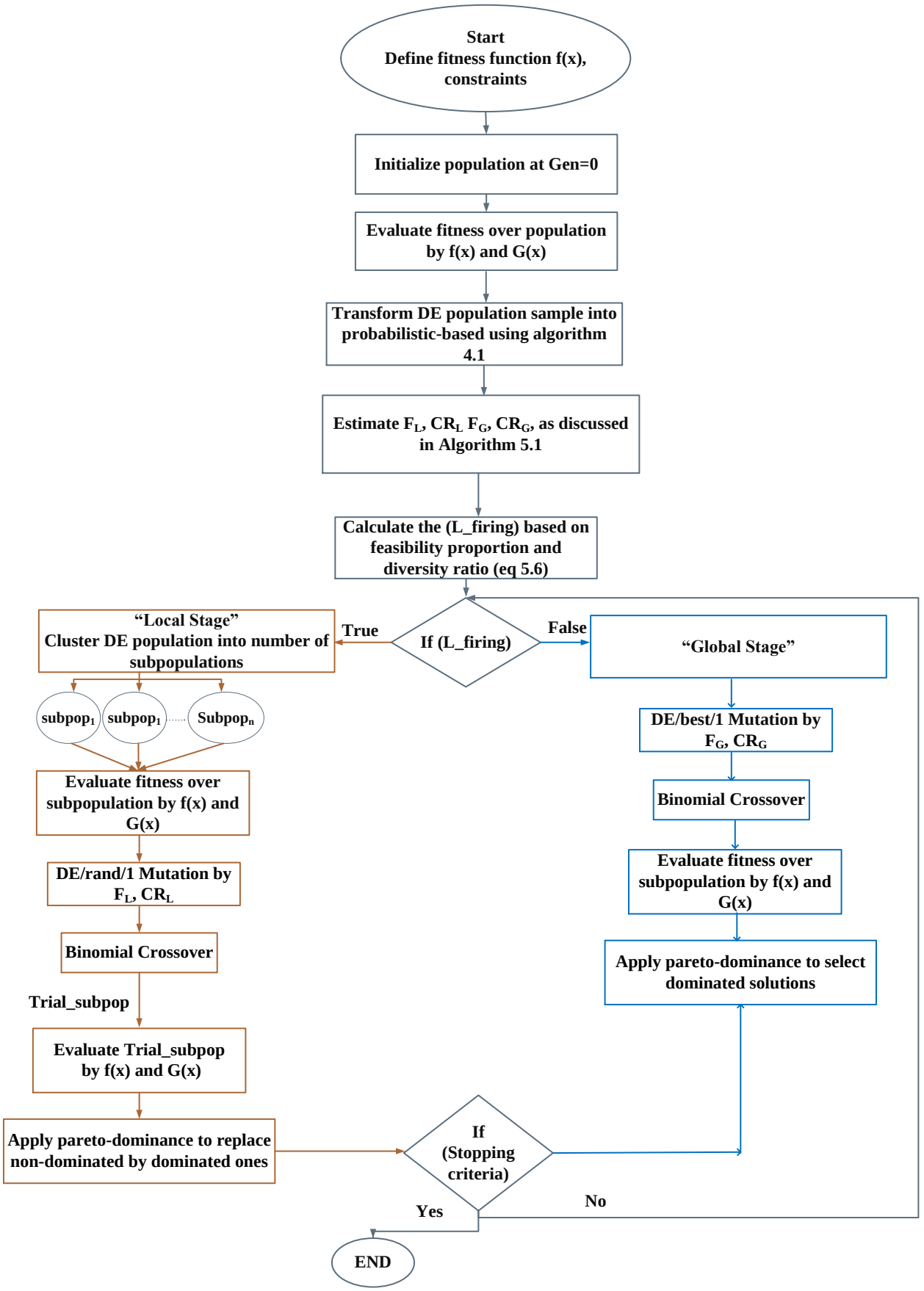


Figure 5.1: Flow Chart of MS-DEHMM-L methodology

the feasible region instead of applying global one. In global search, the search may stagnate in infeasible region because the newly generated offspring could be infeasible or non-dominated with its parents.

5.2.4. Two strategies and control parameters in MS-DEHMM-L

The variability of COPs' characteristics leads to the difficulty of using single strategy or control parameter while the evolutionary progresses. Therefore, there is a requirement to use multiple strategies and parameter values in order to consistently solve a wide range of COPs effectively. The proposed MS-DEHMM-L based on two mutation strategies, shown in Figure ,should utilize the strengths of their various characteristics for various search stages. Two distinct mutation strategies and two parameter sets are implemented according to the nature of each search stage dynamics and behaviour. In particular, DE/rand/1 and DE/best/1 is chosen to be applied in the local and global DE search processes, respectively. The main reasons for choosing those strategies are the powerful exploration capability of DE/rand/1 that can improve the search ability while DE/best/1 has greedy exploitation capability, which it comprises the best individual information to direct the search evolution with no bias.

In the early stages of the evolution process, the number of feasible solutions is usually small or there is no feasible solutions and the population diversity is high so that employing local search with the DE/rand mutation and high F and CR values is recommended as a better way for approaching the feasible region. However, the global search is encouraged when the population achieves the feasible region, then applying the greedy DE/best strategy with slightly smaller values, reaching the best solution. In general, the intent of applying both local and global search as two stages, associated with different strategies and parameter values is to balance between the feasibility and optimality. In other words, the way of maintaining

the balance between approaching the feasible region and converging to the best solution.

The basic steps of MS-DEHMM-L procedure are illustrated below.

Algorithm 5.2 Steps of MS-DEHMM-L

Input $\leftarrow f(x)$, $g(x)$, $h(x)$, PS, D and $Max_{eval} = 20000 * D$

1: Initialize *pop* of $PS * D$ randomly under uniform distribution.
2: $f(x), G(x) \leftarrow$ Evaluate *pop* by $f(x)$, $g(x)$ and $h(x)$
3: Apply algorithm 5.1 for F_L, CR_L,F_G, CR_G
4: **if** (L_firing) **then**
5: Cluster *pop* into *sub_pops* with equal sizes
6: Evaluate *sub_pops*
7: *trial_sub_pops* $\leftarrow$Evolve mutation and crossover using equations 2.1, 2.7, with F_L, CR_L
8: Evaluate *trial_sub_pops*
9: Using Pareto-dominance, replace the non-dominated by another dominated solution in *sub_pops*
10: Combine whole *sub_pops* to form *pop*
11: **end if**
12: Evolve mutation and crossover as in equations 2.4, 2.7, with F_G, CR_G
13: Evaluate *pop*
14: Using Pareto-dominance, select the dominated solutions to $G + 1$
15: **if** (Stopping condition is met) **then**
16: Stop
17: **else**
18: Update $eval = eval + PS$ and $G = G + 1$
19: Go Step 4
20: **end if**

Output$\rightarrow$ *best_solution*

5.3. Experimental Results and Analysis

5.3.1. Benchmark problems

18 benchmark test problems are collected from the CEC2010 special competition on constrained optimization problems [173], as one of the most widely used test suite for evaluating the evolutionary constrained optimization algorithms. We

Table 5.1: Characteristics of CEC2010's test functions where I refers to the number of inequality constraints, E is the number of equality constraints, S is seperable, Non-Sep is not seperable and feasibility pro denotes the ratio between feasible region and whole search space

prob	Range	Objective	No of Constraints		Feas-ratio	
			E	I	10D	30D
H1	[0,10]	Non-S	0	2 Non-S	0.99768	1
H2	[5.12,5.12]	S	1 S	2 S	0	0
H3	[1000,1000]	Non-S	1 S	0	0	0
H4	[50,50]	S	2 Non-S , 2 S	0	0	0
H5	[600,600]	S	2 S	0	0	0
H6	[600,600]	S	2 Rotated	0	0	0
H7	[140,140]	Non-S	0	1 S	0.50512	0.50372
H8	[140,140]	Non-S	0	1 Rotated	0.37951	0.37527
H9	[500,500]	Non-S	1 S	0	0	0
H10	[500,500]	Non-S	1 Rotated	0	0	0
H11	[100,100]	Rotated	1 Non-S	0	0	0
H12	[1000,1000]	S	1 Non-S	1 S	0	0
H13	[500,500]	S	0	2 S, 1 Non-S	0	0
H14	[1000,1000]	Non-S	0	3 S	0.00311	0.00612
H15	[1000,1000]	Non-S	0	3 Rotated	0.00321	0.00602
H16	[10,10]	Non-S	2 S	1 S, 1 Non-S	0	0
H17	[10,10]	Non-S	1 S	2 Non-S	0	0
H18	[50,50]	Non-S	1 S	1 S	0.00001	0

use these functions, known as CEC2010 benchmark suite to evaluate/ judge the efficiency of our MS-DEHMM-L algorithm with 10 and 30 dimensions (D), as provided in Table5.1. In this table, different problems, labelled as H1 ~ H18, with different characteristics, such as linearity or non-linearity or equality or inequality for constrained functions are evolved either objective or constrained functions.

5.3.2. Experiment setup

The proposed algorithm was implemented in Matlab R2015a, and run on a PC with a 3.40 GHz TM-Core i7 processor, 16 GB RAM and Windows 7 with 64 bits. The PS value of MS-DE-HMM-L is set to 100. The total number of function evaluations ($eval$) is set to $20000*D$, where D is the number of dimensions for each problem, 25 independent runs are performed as suggested in [173]. For describing

the best results, we highlight them in boldface type and the number of best results ($No-Best$) for each competing algorithm are reported at the end of results' tables.

In this chapter, the Wilcoxon and Friedman statistical tests [176] are applied to show the significant difference between the proposed MS-DEHMM-L and the comparable algorithms. SPSS statistical analysis software [166] is used. These tests assume in the null hypothesis that there is no significant difference between the mean values of two samples whilst the alternative hypothesis tries to find the significant difference by conducting a 5% significance level. Based on the Wilcoxon and Friedman tests, three different symbols " + " or " − " or " = " (shown in the significance columns) are considered to reflect the significant difference in the average fitness values between MS-DE-HMM-L and the peer algorithms, where " + " or " − " mean that DE-HMM is significantly better or worse than the peer algorithm, respectively, while " = " means that there is no significant difference between the two algorithms. The critical difference (CD) of the Nemenyi post-hoc test [165] that is used to identify the significant differences between the computed ranks when the null hypothesis is rejected, calculated as discussed in Chapter 4 (section 4.4).

5.3.3. Experimental Results

Four well-known state-of-the-art algorithms representing different EA are chosen as competitors for demonstrating the performance of MS-DEHMM-L algorithm against them. All of them were published in the competition of CEC that was held in 2010 of the evolutionary constrained optimization competition. Note that all the results of the competing algorithms are taken from their original papers.

1. Constrained DE with an archive and gradient-based mutation (εDEag) [90] that won the CEC2010 constrained optimization competition;

2. Co-evolutionary comprehensive learning particle swarm optimizer (Co-CLPSO) [177];

Table 5.2: Mean and std of function error values of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 10D.

10D					
	εDEag	CO-CLPSO	eABC	jDEsoco	MS-DEHMM-L
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
H1	-7.470402E-01 (1.323339E-03)	-7.335800E-01 (1.784800E-02)	-7.162570E-01 (2.689780E-02)	-7.383600E-01 (1.600600E-02)	**-7.473104E-01** **(1.350625E-03)**
H2	-2.258870E+00 (2.389779e)	-2.266600E+00 (1.461600E-02)	-1.248950E-01 (1.583590E+00)	5.635900E-01 (1.104400E+00)	**-2.277710E+00** **(1.447037E-02)**
H3	**0.000000E+00** **(0.000000E+00)**	3.550200E-01 (1.775100E+00)	2.445070E+12 (1.009670E+01)	7.810500E+00 (2.943700E+00)	9.262631E-21 (1.678290E-07)
H4	-9.918452E-06 (1.546730E-07)	-9.338500E-06 (1.074800E-06)	8.563060E-01 (3.006280E+00)	**-1.00000E-05** **(9.483100E-16)**	**-1.000000E-05** **(4.887269E-07)**
H5	**-4.836106E+02** **(3.890350E-13)**	-4.836000E+02 (1.957700E-02)	3.652470E+02 (1.172050E+02)	-3.021700E+02 (3.025600E+02)	-4.835068E+02 (1.426070E-01)
H6	-5.786528E+02 (3.627169E-03)	-5.786600E+02 (5.728900E-04)	4.381680E+02 (8.595360E+01)	-5.740800E+02 (1.646100E+01)	**-5.796248E+02** **(1.581617E+01)**
H7	**0.000000E+00** **(0.000000E+00)**	7.973200E-01 (1.627500E+00)	7.160940E+01 (5.191130E+01)	6.419200E-27 (1.351500E-26)	3.231059E-05 (7.913326E-01)
H8	6.727528E+00 (5.560648E+00)	6.087600E-01 (1.425500E+00)	4.107890E+02 (9.356030E+02)	3.742100E+00 (1.033000E+01)	**2.125510E-20** **(4.466395E-02)**
H9	**0.000000E+00** **(0.000000E+00)**	1.993800E+10 (9.968800E+10)	2.019340E+12 (1.810830E+12)	**5.289800E-01** **(1.462000E+00)**	2.136711E+01 (2.292318E+00)
H10	**0.000000E+00** **(0.000000E+00)**	1.395100E+09 (2.487100E+11)	1.746200E+12 (2.583000E+12)	3.171200E+01 (1.818800E+01)	6.322286E-03 (4.706813E+01)
H11	-1.522713E-03 (6.341035E-11)	-1.612500E-01 (6.602500E-01)	**-1.230170E+00** **(3.035560E+00)**	-8.255500E-03 (2.380700E-02)	-3.342758E-03 (1.668680E-10)
H12	-3.367349E+02 (1.782166E+02)	-2.336900E+00 (2.432900E+01)	-1.800890E+02 (2.757580E+02)	-2.236500E+01 (1.108300E+02)	**-8.870370E+02** **(3.131391E+02)**
H13	-6.842936E+01 (1.025960E-06)	-6.527800E+01 (2.576300E+00)	-6.568060E+01 (2.502700E+00)	-6.831500E+01 (5.701800E-01)	**-6.842937E+01** **(2.271798E-05)**
H14	**0.000000E+00** **(0.000000E+00)**	3.189300E-01 (1.103800E+00)	8.004100E+10 (2.366080E+11)	9.122100E-01 (2.453800E+00)	3.262775E-01 (7.633829E-01)
H15	**1.798978E-01** **(8.813156E-01)**	2.988500E+00 (3.314700E+00)	2.565760E+13 (2.862760E+13)	1.245200E+09 (3.812700E+09)	2.170927E+08 (3.326661E+12)
H16	3.702054E-01 (3.710479E-01)	5.986100E-03 (1.331500E-02)	8.347410E-02 (9.106390E-02)	4.110200E-01 (3.835900E-01)	**0.000000E+00** **(0.000000E+00)**
H17	1.249561E-01 (1.937197E-01)	3.798600E-01 (4.528400E-01)	2.680100E+01 (6.831360E+00)	8.895800E+01 (9.913100E+01)	**2.957758E-02** **(8.591504E+00)**
H18	**9.678765E-19** **(1.811234E-18)**	2.319200E-01 (9.955900E-01)	3.470230E+02 (3.710760E+02)	4.050000E+02 (7.376200E+02)	1.350433E+00 (2.564924E+02)
No-Best	8	0	1	2	9

3. Elitist artificial bee colony (eABC) [178][13]; and

4. An Improved Self-adaptive Differential Evolution Algorithm in Single Objective Constrained Real-Parameter Optimization (jDEsoco) [179].

The detailed results at 10 and 30 dimensions are shown in Table 5.2 and 5.3, respectively. Some promising observations are derived from these tables.

- ***For 10D test functions***, MS-DEHMM-L and εDEag can produce better results than Co-CLPSO, eABC and jDEsoco on the majority of the test functions. Co-CLPSO achieves promising findings in some cases such as H1, H2, H4 and H6 while jDEsoco can reveal acceptable findings in three functions (H1, H4 and H6). MS-DEHMM-L is able to achieve the 100% feasibility rate for all test functions except H10 and H11 as they record 80% and 40%, respectively as listed in Table 5.4. In addition, it can be seen from this table that the feasibility ratio of εDEag is 100% for all problems. Co-CLPSO reported 100% proportion for all problems but it cannot produce any feasible solutions for H11 while jDEsoco could not achieve 100% feasibility ratio for seven functions.

- ***On 30D test functions***, although MS-DEHMM-L is still the superior method to solve most instance against the compared competitors with respect to the quality of solution; its feasibility ratio is not as good as those in 10D. In particular, the proposed methodology is unable to achieve feasible outcomes for functions H3, H9 and H15 but it is still 100% for the remaining functions. As shown in Table 5.4, εDEag is able to achieve 100% for all problems except H12 (12% only) while Co-CLPSO and jDEsoco have 12 functions out of 16 achieving 100% feasibility proportion.

- ***Regarding the Wilcoxon test results***, from Table 5.5 , it is clear that MS-DEHMM-L shows significantly better results than all the compared algorithms for 10D except εDEag. On the other hand, MS-DEHMM-L is only

Table 5.3: Mean and std of function error values of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 30D.

30D					
	εDEag	CO-CLPSO	eABC	jDEsoco	MS-DEHMM-L
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
H1	-8.208687E-01 (7.103893E-04)	-7.159800E-01 (5.025200E-02)	-7.305540E-01 (4.875890E-02)	-8.123800E-01 (1.318700E-02)	**-8.218841E-01** **(1.760511E-03)**
H2	-2.151424E+00 (1.197582E-02)	-2.202900E+00 (1.926700E-01)	2.556470E+00 (9.429490E-01)	1.560300E+00 (8.470500E-01)	**-2.271995E+00** **(2.246615E-02)**
H3	**2.883785E+01** **(8.047159E-01)**	3.510600E+01 (3.310100E+01)	1.070870E+13 (2.208900E+12)	6.144700E+01 (5.757700E+01)	1.615494E+10 (7.253518E+11)
H4	8.162973E-03 (3.067785E-03)	1.126900E-01 (5.633500E-01)	2.145720E+01 (6.219130E+00)	3.518700E-04 (2.394800E-04)	**2.198817E-06** **(3.004453E-08)**
H5	**-4.495460E+02** **(2.899105E+00)**	-3.124900E+02 (8.833200E+01)	3.718920E+02 (7.888540E+01)	1.082200E+02 (1.520300E+02)	3.409147E+02 (6.397773E+01)
H6	-5.279068E+02 (4.748378E-01)	-2.447000E+02 (3.948100E+01)	4.738410E+02 (6.302590E+01)	-4.728400E+02 (1.274300E+02)	**-5.306080E+02** **(3.349227E-02)**
H7	2.603632E-15 (1.233430E-15)	1.116300E+00 (1.826900E+00)	1.332900E+02 (2.058280E+02)	8.739600E-02 (3.252900E-23)	**3.029077E-18** **(5.720746E-14)**
H8	7.831464E-14 (4.855177E-14)	4.751700E+01 (1.125900E+02)	1.501960E+02 (7.148770E+01)	8.258500E+01 (2.439500E+02)	**3.886901E-21** **(4.264525E-17)**
H9	1.072140E+01 (2.821923E+01)	1.482200E+08 (2.450900E+08)	1.607560E+13 (9.287410E+12)	**2.474300E+00** **(8.778200E+00)**	2.966268E+06 (1.522197E+05)
H10	3.326175E+01 (4.545577E-01)	1.395100E+09 (5.843800E+09)	1.498640E+13 (9.773360E+12)	2.938600E+01 (7.178600E+00)	**2.835986E+00** **(1.533118E-01)**
H11	-2.863882E-04 (2.707605E-05)	2.818600E-02 (3.212400E-02)	-5.885290E-01 (6.486250E-01)	1.166700E-03 (5.269000E-03)	**-1.250024E+00** **(1.688888E+00)**
H12	3.562330E+02 (2.889253E+02)	-1.991100E-01 (1.184000E-04)	5.071100E+01 (3.704680E+02)	-1.992500E-01 (2.345300E-05)	**-3.910930E+02** **(4.312008E+00)**
H13	-6.535310E+01 (5.733005E-01)	-6.077400E+01 (1.117600E+00)	-6.485590E+01 (1.382130E+00)	**-6.753700E+01** **(5.055300E-01)**	-6.741601E+01 (1.480811E+00)
H14	**3.089407E-13** **(5.608409E-13)**	1.275700E+00 (1.898000E+00)	9.947190E+03 (1.916060E+04)	1.594600E-01 (7.973200E-01)	2.881985E-04 (2.312628E+02)
H15	**2.160376E+01** **(1.104834E-04)**	5.105900E+01 (9.175900E+01)	3.785130E+13 (3.440730E+13)	1.535700E+09 (1.604500E+09)	9.639835E+12 (5.962591E+13)
H16	**2.168404E-21** **(1.062297E-20)**	5.240300E-16 (4.672200E-16)	8.207220E-01 (2.569880E-01)	7.320600E-01 (2.994300E-01)	7.838309E-16 (1.390063E-11)
H17	6.326487E+00 (4.986691E+00)	1.391900E+00 (4.262100E+00)	2.680100E+01 (1.634550E+01)	5.039800E+02 (4.483200E+02)	**1.229338E+00** **(6.437713E+02)**
H18	8.754569E+01 (1.664753E+02)	**1.087700E+01** **(3.716100E+01)**	2.933360E+02 (3.528430E+02)	3.084900E+02 (3.053800E+02)	2.056003E+02 (1.167261E+04)
No-Best	5	1	0	2	10

Table 5.4: Feasibility Rate (%) of MS-DEHMM-L and the state-of-art competitors on CEC2010 18-test problems at 10D and 30D

prob	D	εDEag	CO-CLPSO	eABC	jDEsoco	MS-DEHMM-L
H1	10	100	100	Not-reported	100	100
	30	100	100		100	100
H2	10	100	100		88	100
	30	100	100		100	100
H3	10	100	100		100	100
	30	100	0		100	0
H4	10	100	100		100	100
	30	100	80		100	100
H5	10	100	100		88	100
	30	100	100		88	100
H6	10	100	100		100	100
	30	100	100		88	100
H7	10	100	100		100	100
	30	100	100		100	100
H8	10	100	100		100	100
	30	100	100		100	100
H9	10	100	100		100	100
	30	100	100		100	92
H10	10	100	100		100	80
	30	100	100		100	100
H11	10	100	0		92	40
	30	100	0		88	100
H12	10	100	100		96	100
	30	12	92		100	100
H13	10	100	100		100	100
	30	100	100		100	100
H14	10	100	100		100	100
	30	100	100		100	100
H15	10	100	100		100	100
	30	100	100		100	65
H16	10	100	100		100	100
	30	100	100		100	100
H17	10	100	100		100	100
	30	100	100		92	100
H18	10	100	100		100	100
	30	100	100		100	100

Table 5.5: Comparison summary between MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems at 10D and 30D.

MS-DE-HMM-L		"Better"	"Worse"	"Equal"	"p-value"	"Significance"
εDEag	10D	**8**	4	6	0.4225	=
	30D	**11**	5	2	0.474	=
CO-CLPSO	10D	**14**	4	0	**0.039**	+
	30D	**13**	5	0	0.143	=
eABC	10D	**17**	1	0	**0.0195**	+
	30D	**18**	0	0	**0.02**	+
jDEsoco	10D	**14**	3	1	**0.003**	+
	30D	**12**	4	2	0.223	=

Table 5.6: Rankings obtained from Friedman's test of 10, 30D of MS-DEHMM-L and the state-of-art algorithms on CEC2010 18-test problems.

Method	Ranks	
	10D	30D
MS-DEHMM-L	**1.97**	**1.94**
εDEag	2.11	2.22
CO-CLPSO	3.00	3.22
eABC	4.33	4.56
jDEsoco	3.59	3.06
"p-value"	**9.32E-06**	**4.50E-06**
"Significance"	+	+
"CD"	1.44	

significantly better than eABC algorithm for 30D problems. For a further step of comparison, the Friedman statistical test is employed and reported in Table 5.6 for both 10 and 30D. Based on these results, it is observed that our proposed MS-DEHMM-L occupies the best ranking followed by εDEag.

5.4. Chapter summary

Building upon the encouraging results achieved from DE-HMM proposed in Chapter 4 for solving the unconstrained problems, in this chapter, a special local search operator is incorporated with the DE-HMM as well as employing two different mutation strategies and estimated parameter values for both the global and local search stages, separately. The new framework is named MS-DEHMM-L.

In this framework, the local search is fired at the beginning of the search procedure, as the majority of population's individuals are infeasible and the diversity still high, for exploring the search space to reach the feasible area. In this stage, the global search starts to exploit the feasible region, reaching the optimal solution at the end of the search.

MS-DEHMM-L is tested on the CEC2010 constrained problems and compared with four different constrained EA, with the results showing that our proposed algorithm has the first ranking between them and outperforms them in most cases.

Chapter 6

Conclusions and Recommendations for Future Work

This chapter summarizes the research work undertaken for this thesis, the conclusions drawn from it and potential research directions

6.1. Summary of research conducted

In this study, the effects of setting the control parameters of Evolutionary Algorithms (EAs), especially Differential Evolution (DE), for obtaining successful solutions to various unconstrained or constrained optimization problems are investigated. Chapters 1 and 2 discuss the importance of selecting the search operators and control parameters of DE that can improve its performance. Setting the control parameters is a common challenge in the DE community because: no single value is successful for all problems; for the same problem, each stage in the evolutionary process requires different parameter values; and obtaining all the interactions between these parameters is complex. Although, in the literature, several studies introduce different ways of adapting the parameter values, most do not consider their actual inter-dependencies. Therefore, three methods for exploring and interpreting the potential process of DE which can help to effectively control its parameters (F and CR) for solving a wide range of unconstrained and constrained optimization problems are developed in this study.

The main objective of this research is to dynamically control the parameters of DE which could remove a burden from users by reducing the time-consuming effort required to manually determine their most appropriate values and improve the quality of solutions. To achieve this, the following three methods are implemented.

Firstly, in Chapter 3, a Semantic Evolutionary Visualization (SEV) framework that uses the theory of reasoning to visualize the evolutionary dynamics of DE's internal procedure helps an analysis of the effects of different parameter settings on DE's performance. It contains three primary components: an EA (i.e., DE); a hierarchical profiling algorithm, namely HP; and a visualization process. Firstly, although the DE algorithm is an evolutionary optimization heuristic that is fast and popular, SEV can be applied for any other EA. The DE algorithm has two main parameters (F and CR) which need to be set up and slightly modified to fit the requirements of the SEV framework. Secondly, HP is a pre-processing step that recursively transforms the great amount of DE data into a suitable format for the visualization stage. Finally, this framework focuses on three types of visualizations, two new and one classical that is, pedigree, correlations associated with a linear regression analysis and fitness graphs, respectively, where the DE ongoing evolutionary procedure can be semantically reasoned under different setups.

In Chapter 4, a novel methodology for dynamically adjusting the control parameters of DE (i.e., F and CR) is developed to potentially improve the performance of the conventional DE for solving unconstrained problems in terms of solution quality and computational resources. In it, the classical DE is coupled with the procedure for a Hidden Markov Model (HMM), resulting in the DE-HMM self-adaptive algorithm for tackling the challenges of setting the parameters (i.e., the dependencies of the problem and its evolution, and the complex interactions among the F and CR parameters) with no need for any external information.

The DE-HMM starts by initializing the whole population according to predefined boundaries and then evaluates each individual in it by the fitness function to be optimized (minimization in this study). New offspring are generated with

regard to the main operators of DE, mutation, crossover and selection, with the first two requiring the parameters F and CR to be set up, respectively. These parameters are automatically adapted for each individual during the evolutionary process by estimating the posterior and likelihood proportions of the HMM, respectively, while the approach for estimating them is inspired by the nature of the HMM's operation. HMM functions similarly to DE whereby, like DE's population changes over time, its core procedure reflects time-series models. The speed of DE is not greatly impacted because the statistics required for HMM calculations are based mainly on first-order mathematical computations which consume linear costs.

Furthermore, the self-adaptive DE-HMM is extended by classifying DE procedure into two main stages' processes; global and local for solving constrained problems and achieving both feasibility and optimality, called the MS-DE-HMM-L framework. In it, the local search is integrated within the DE process based on the feasibility status and diversity ratio and the strengths of two different mutation strategies provided in the literature is utilized to employed through the two search processes. In it, the population is initialized at the start of an evolution and candidate solutions evaluated by the fitness and constraint functions. The HMM estimates the required F and CR for each process, then check the local search firing condition to start working. The COP is treated as a multi-objectives problem for handling the evolved constraints.

The three methodologies introduced in this thesis demonstrate outstanding performance of our method compared with those of both the conventional DE or other state-of-art algorithms as enhanced solutions are obtained without no much computational resources.

6.2. Research Findings

Each of the proposed algorithms implemented in this thesis is evaluated using different benchmark datasets containing unconstrained and constrained optimization problems. Their results are summarized and compared with those from different popular and recently published algorithms in the literature. After conducting a comprehensive and statistical analysis, the following conclusions are drawn.

6.2.1. SEV framework

In Chapter 3, DE is used as a testbed algorithm for evaluating and demonstrating the effectiveness of the proposed SEV framework. Two test functions with different characteristics (i.e., unimodal and multimodal functions) are developed with different settings of DE's control parameters, the influence of which on DE's performance is determined. The graphs obtained provide insights into all the relationships among the children solutions and their parental candidates evidenced by pedigree visualisation. Although the fitness graphs are quite traditional, they assist tracking of the ongoing dynamics during evolutionary convergence. The correlation graphs supported by a linear regression analysis are significant measurements of the inheritance of the fitness landscape that indicate the features of the search domain as either smooth or rugged. As these plots can demonstrate the actual progression of the evolutionary procedure with different parameter settings, this simplifies the design of a self-adaptive method.

6.2.2. Methodology of self-adaptive DE-HMM

In Chapter 4, two benchmark problem sets with 55 test functions (i.e., IEEE CEC2005 and CEC2014) and 10, 30 and 50 dimensions are used to conduct comprehensive experiments to analyse the effect of the HMM on the performance of DE. Different combinations of learning strategies and crossover operators for the

30D CEC2005 and CEC2014, and different variants of the DE-HMM for the 30D CEC2005 test problems, indicate the roles of different components and/or different operators in the DE-HMM. The findings from this experiment demonstrate the superiority of the DE-HMM (minimum posterior for F and greater improvement for CR) over the other DE-HMM variants, including random generations of the parameters. Comparing the DE-HMM using a rand mutation strategy and binomial crossover, which are the most common and basic operators of DE, with other variants, including the best mutation strategy and exponential crossover, shows that the former achieve better results for most of the instances/problems in the CEC2005 and CEC2014 test sets.

Furthermore, the DE-HMM is compared with twelve DE- and non-DE-based state-of-the-art methods comprising the most common, competitive and recently published. Firstly, the results obtained for the two test sets are analyzed using two non-parametric statistical tests, the Wilcoxon signed-rank and Friedman. It is clear that the DE-HMM outperforms both the classical DE and other algorithms regarding solution quality. Although some algorithms produce better results for some problems, as the differences between them and those from the DE-HMM are small, the majority of test functions can be effectively solved by the DE-HMM. Secondly, while the computational times taken are measured based on the pre-defined benchmark, those consumed to achieve the best solutions for some functions with 10D, 30D and 50D are computed. The performances of the DE-HMM and each competing algorithm are described as trade-offs between the quality of solutions obtained and times consumed to achieve them. Although the DE-HMM consumes more time than the classical DE and the other two peer algorithms to solve the CEC2005 test functions at different dimensions, it can obtain better solutions.

In conclusion, combining a HMM with a DE algorithm improves the performance of DE in terms of both the solution quality and computational cost. As it deals with the evolutionary process as a black box, there is no need for any external information, with the DE population used to build the HMM model and

the control parameters F and CR automatically computed regardless of any user reward.

6.2.3. Multi-stage mutation MSMODE-HMM methodology

In Chapter 5, MS-DEHMM-L algorithm is compared with other four constrained methods developed and published in CEC2010 competition to solve COPs. It is evaluated using the CEC2010 benchmark dataset, which was designed specifically for constrained optimization.

Based on the results, MS-DEHMM-L outperforms the other competing state-of-art-algorithms as it can obtain better feasible solution on most of the test function, followed by ε DEag. The feasibility ratio for our proposed algorithm to reach these results are reported as it achieves 100% feasibility rate for 16 out of 18 test problems at both 10 and 30 dimensions. Two non-parametric statistical tests, namely Wilcoxon and Friedman tests are applied for assessing the significant difference and ranking of our MS-DEHMM-L against other compared constrained methods. MS-DEHMM-L performs significantly better than all peer algorithms except ε DEag at 10D while it is significantly better than eABC method for 30D. Finally, our proposed method achieves the best ranking. Therefore, the idea of using HMM to estimate the F and CR parameters as well as employing different mutation strategies through the two stages' processes facilitates the movement of search from infeasible solutions to explore feasible regions towards optimal solutions.

Overall, the three proposed framework in this thesis study are beneficial to explore and analyze the search space efficiently and automatically adjusting F and CR intrinsic parameters of DE, improving the capability of solving unconstrained and constrained optimization problems.

6.3. Recommendations for Future Work

The current methodologies developed in this thesis could be extended in the various following ways.

1. Applying the HMM to adapt the parameters of other EAs.
2. Developing new search operators that can facilitate the search procedure and save computational resources.
3. Applying other learning strategies through DE's evolutionary stages to investigate their influence on enhancing its capabilities to obtain better solutions.
4. Solving a number of real-world problems using either the proposed unconstrained or constrained approaches.
5. Testing the proposed algorithms in this thesis on a variety of difficult benchmark suites, including dynamic, niching and large-scale problems.
6. Improving the visualization framework by considering other landscape matrices.
7. Incorporating other constraint-handling mechanisms in the proposed methods to provide alternatives for obtaining optimal feasible solutions rather than inferior infeasible ones.
8. Evaluating the capabilities of the proposed algorithms to solve multi-objective optimization problems.
9. Employing the concept of local search techniques with different settings to accelerate the convergence of search.

References

[1] Lean Yu, Lunchao Hu, and Ling Tang. Stock selection with a novel sigmoid-based mixed discrete-continuous differential evolution algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1891–1904, 03 2016.

[2] Juliana Almansa Malagoli, Jose Roberto Camacho, Mauricio Valencia Ferreira da Luz, Jacson Hudson Inacio Ferreira, and Adelicio Maximiano Sobrinho. Design of three-phase induction machine using differential evolution algorithm. *IEEE Latin America Transactions*, 13(7):2202–2208, 2015.

[3] Dong Han, Xuejun Zhang, and Xueyuan Li. Aircrafts conflict resolution using differential evolution. In *Intelligent Control and Automation (WCICA), 2016 12th World Congress on*, pages 9–12. IEEE, 2016.

[4] Abu SSM Barkat Ullah, Ruhul Sarker, and David Cornforth. Search space reduction technique for constrained optimization with tiny feasible space. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 881–888. ACM, 2008.

[5] Erik Cuevas, Alonso Echavarría, and Marte A Ramírez-Ortegón. An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. *Applied intelligence*, 40(2):256–272, 2014.

[6] Xin-She Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

[7] Elena-Niculina Dragoi and Vlad Dafinescu. Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review*, 45(4):447–470, 2016.

[8] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.

[9] Magnus Erik Hvass Pedersen and Andrew John Chipperfield. Parameter tuning versus adaptation: proof of principle study on differential evolution. *HL0803. Hvass Laboratories*, 2008.

[10] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[11] Ágoston E Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 3(2):124–141, 1999.

[12] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[13] Cajo JF Ter Braak. A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249, 2006.

[14] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[15] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: a survey. *ACM Computing Surveys (CSUR)*, 45(3):35, 2013.

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

[17] Daniela Zaharie. Critical values for the control parameters of differential evolution algorithms. In *Proceedings of MENDEL*, volume 2, page 6267, 2002.

[18] Parc Scientifique Georges Besse. Overview of differential evolution. *Conference Francophone de MOdÂŕelisation et SIMulation*, 2004.

[19] Swagatam Das, Sankha Subhra Mullick, and Ponnuthurai N Suganthan. Recent advances in differential evolution–an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.

[20] Ferrante Neri and Ville Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2):61–106, 2010.

[21] Daniela Zaharie. A comparative analysis of crossover variants in differential evolution. *Proceedings of IMCSIT*, 2007:171–181, 2007.

[22] Donald Davendra and Godfrey Onwubolu. Forward backward transformation. In *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*, pages 35–80. Springer, 2009.

[23] Ayed Salman, Andries P Engelbrecht, and Mahamed GH Omran. Empirical analysis of self-adaptive differential evolution. *European Journal of operational research*, 183(2):785–804, 2007.

[24] Elena Simona Nicoară. Mechanisms to avoid the premature convergence of genetic algorithms. *Petroleum–Gas University of Ploieşti Bulletin, Math.–Info.–Phys. Series*, 61:87–96, 2009.

[25] Daniela Zaharie. Parameter adaption in differential evolution by controlling the population diversity. In *4th Int. Workshop Symbolic Numeric Algorithms Scientific Computing, Timi soara, Romania, October*, pages 9–12, 2002.

[26] Magnus Erik Hvass Pedersen. *Tuning & simplifying heuristical optimization*. PhD thesis, University of Southampton, 2010.

[27] Janez Brest. *Constrained real-parameter optimization with ε-self-adaptive differential evolution*. Springer, 2009.

[28] Junhong Liu and Jouni Lampinen. On setting the control parameter of the differential evolution method. In *Proc. 8th Int. Conf. Soft Computing MENDEL 2002*, pages 11–18, 2002.

[29] Rammohan Mallipeddi, Ponnuthurai N Suganthan, Quan-Ke Pan, and Mehmet Fatih Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696, 2011.

[30] P Kaelo and MM Ali. Differential evolution algorithms using hybrid mutation. *Computational Optimization and Applications*, 37(2):231–246, 2007.

[31] Ali Wagdy Mohamed, Hegazy Zaher Sabry, and Tareq Abd-Elaziz. Real parameter optimization by an effective differential evolution algorithm. *Egyptian Informatics Journal*, 14(1):37–53, 2013.

[32] Selmar Kagiso Smit. *Parameter tuning and scientific testing in evolutionary algorithms*. Vrije Universiteit, 2012.

[33] Janez Brest, Borko Bošković, Sašo Greiner, Viljem Žumer, and Mirjam Sepesy Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing*, 11(7):617–629, 2007.

[34] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015.

[35] Tsung-Che Chiang, Cheng-Nan Chen, and Yu-Chieh Lin. Parameter control mechanisms in differential evolution: a tutorial review and taxonomy. In *Differential Evolution (SDE), 2013 IEEE Symposium on*, pages 1–8. IEEE, 2013.

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

[37] Mingming Zhang, Shuguang Zhao, and Xu Wang. Multi-objective evolutionary algorithm based on adaptive discrete differential evolution. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 614–621. IEEE, 2009.

[38] Ryoji Tanabe and Alex S Fukunaga. Improving the search performance of shade using linear population size reduction. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1658–1665. IEEE, 2014.

[39] Radha Thangaraj, Millie Pant, and Ajith Abraham. A simple adaptive differential evolution algorithm. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 457–462. IEEE, 2009.

[40] Hu Chunping and Yan Xuefeng. An immune self-adaptive differential evolution algorithm with application to estimate kinetic parameters for homogeneous mercury oxidation. *Chinese Journal of Chemical Engineering*, 17(2):232–240, 2009.

[41] Lixin Tang, Yun Dong, and Jiyin Liu. Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, 19(4):560–574, 2015.

[42] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[43] Yong Wang, Zixing Cai, and Qingfu Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66, 2011.

[44] Wenchao Yi, Liang Gao, Xinyu Li, and Yinzhi Zhou. A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. *Applied Intelligence*, 42(4):642–660, 2015.

[45] Arnob Ghosh, Swagatam Das, Aritra Chowdhury, and Ritwik Giri. An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Information Sciences*, 181(18):3749–3765, 2011.

[46] Qinqin Fan and Xuefeng Yan. Self-adaptive differential evolution algorithm with discrete mutation control parameters. *Expert Systems with Applications*, 42(3):1551–1572, 2015.

[47] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1041–1048. IEEE, 2011.

[48] Saber M Elsayed, Ruhul A Sarker, Daryl L Essam, and Noha M Hamza. Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1650–1657. IEEE, 2014.

[49] Guohua Wu, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, Rui Wang, and Huangke Chen. Differential evolution with multi-population based ensemble of mutation strategies. *Information Sciences*, 329:329–345, 2016.

[50] Junhong Liu and Jouni Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6):448–462, 2005.

[51] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[52] Yong Wang, Han-Xiong Li, Tingwen Huang, and Long Li. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Applied Soft Computing*, 18:232–247, 2014.

[53] Johanna Aalto and Jouni Lampinen. A mutation and crossover adaptation mechanism for differential evolution algorithm. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 451–458. IEEE, 2014.

[54] Ruhul A Sarker, Saber M Elsayed, and Tapabrata Ray. Differential evolution with dynamic parameters selection for optimization problems. *IEEE Trans. Evolutionary Computation*, 18(5):689–707, 2014.

[55] Guillaume Corriveau, Raynald Guilbault, Antoine Tahan, and Robert Sabourin. Bayesian network as an adaptive parameter setting approach for genetic algorithms. *Complex & Intelligent Systems*, 2(1):1–22, 2016.

[56] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 71–78. IEEE, 2013.

[57] Saber M Elsayed, Ruhul A Sarker, and Tapabrata Ray. Differential evolution with automatic parameter configuration for solving the cec2013 competition on real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1932–1937. IEEE, 2013.

[58] Wei-Jie Yu, Meie Shen, Wei-Neng Chen, Zhi-Hui Zhan, Yue-Jiao Gong, Ying Lin, Ou Liu, and Jun Zhang. Differential evolution with two-level parameter adaptation. *IEEE Trans. Cybernetics*, 44(7):1080–1099, 2014.

[59] Rammohan Mallipeddi, Guohua Wu, Minho Lee, and Ponnuthurai Nagaratnam Suganthan. Gaussian adaptation based parameter adaptation for differential evolution. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1760–1767. IEEE, 2014.

[60] Jingqiao Zhang and Arthur C Sanderson. Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation*, 13(5):945–958, 2009.

[61] Yong Wang, Zhi-Zhong Liu, Jianbin Li, Han-Xiong Li, and Gary G Yen. Utilizing cumulative population distribution information in differential evolution. *Applied Soft Computing*, 48:329–346, 2016.

[62] Wenyin Gong and Zhihua Cai. Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics*, 43(6):2066–2081, 2013.

[63] Yinzhi Zhou, Xinyu Li, and Liang Gao. A differential evolution algorithm with intersect mutation operator. *Applied Soft Computing*, 13(1):390–401, 2013.

[64] Cheng-Wen Chiang, Wei-Ping Lee, and Jia-Sheng Heh. A 2-opt based differential evolution for global optimization. *Applied Soft Computing*, 10(4):1200–1207, 2010.

[65] Yong Wang, Zixing Cai, and Qingfu Zhang. Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, 185(1):153–177, 2012.

[66] Shu-Mei Guo and Chin-Chang Yang. Enhancing differential evolution utilizing eigenvector-based crossover operator. *IEEE Transactions on Evolutionary Computation*, 19(1):31–49, 2015.

[67] Yiqiao Cai and Jiahai Wang. Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Transactions on Cybernetics*, 43(6):2202–2215, 2013.

[68] Janez Brest and Mirjam Sepesy Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, 15(11):2157–2174, 2011.

[69] Aleš Zamuda and Janez Brest. Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In *Swarm and evolutionary computation*, pages 154–161. Springer, 2012.

[70] Ales Zamuda, Janez Brest, and Efrén Mezura-Montes. Structured population size reduction differential evolution with multiple mutation strategies on cec 2013 real parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1925–1931. IEEE, 2013.

[71] Ming Yang, Zhihua Cai, Changhe Li, and Jing Guan. An improved adaptive differential evolution algorithm with population adaptation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 145–152. ACM, 2013.

[72] Venkateswarlu Gonuguntla, Rammohan Mallipeddi, and Kalyana C Veluvolu. Differential evolution with population and strategy parameter adaptation. *Mathematical Problems in Engineering*, 2015, 2015.

[73] Wu Zhu, Yang Tang, Jian-An Fang, and Wenbing Zhang. Adaptive population tuning scheme for differential evolution. *Information Sciences*, 223:164–191, 2013.

[74] Nga Sing Teng, Jason Teo, and Mohd Hanafi A Hijazi. Self-adaptive population sizing for a tune-free differential evolution. *Soft Computing*, 13(7):709–724, 2009.

[75] Edwin C Shi, Frank HF Leung, and Bonnie NF Law. Differential evolution with adaptive population size. In *Digital Signal Processing (DSP), 2014 19th International Conference on*, pages 876–881. IEEE, 2014.

[76] Yong Wang and Zixing Cai. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 16(1):117–134, 2012.

[77] Rammohan Mallipeddi, Swagatam Das, and Ponnuthurai Nagaratnam Suganthan. Ensemble of constraint handling techniques for single objective constrained optimization. In *Evolutionary Constrained Optimization*, pages 231–248. Springer, 2015.

[78] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002.

[79] Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997.

[80] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Integrated strategies differential evolution algorithm with a local search for constrained optimization. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2618–2625. IEEE, 2011.

[81] Soumen Sardar, Sayan Maity, Swagatam Das, and Ponnuthurai Nagaratnam Suganthan. Constrained real parameter optimization with a gradient repair based differential evolution algorithm. In *Differential Evolution (SDE), 2011 IEEE Symposium on*, pages 1–8. IEEE, 2011.

[82] Liang Gao, Yinzhi Zhou, Xinyu Li, Quanke Pan, and Wenchao Yi. Multi-objective optimization based reverse strategy with differential evolution algorithm for constrained optimization problems. *Expert Systems with Applications*, 42(14):5976–5987, 2015.

[83] Wenhong Wei, Jiahai Wang, and Ming Tao. Constrained differential evolution with multiobjective sorting mutation operators for constrained optimization. *Applied Soft Computing*, 33:207–222, 2015.

[84] Wei-Feng Gao, Gary G Yen, and San-Yang Liu. A dual-population differential evolution with coevolution for constrained optimization. *IEEE transactions on cybernetics*, 45(5):1108–1121, 2015.

[85] Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[86] Ali Wagdy Mohamed and Hegazy Zaher Sabry. Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, 194:171–208, 2012.

[87] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Differential evolution with ensemble of constraint handling techniques for solving cec 2010 benchmark problems. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[88] Bing Xue, Wenlong Fu, and Mengjie Zhang. Differential evolution (de) for multi-objective feature selection in classification. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 83–84. ACM, 2014.

[89] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on evolutionary computation*, 4(3):284–294, 2000.

[90] Tetsuyuki Takahama and Setsuko Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9. IEEE, 2010.

[91] Wenyin Gong, Zhihua Cai, and Dingwen Liang. Adaptive ranking mutation operator based differential evolution for constrained optimization. *IEEE transactions on cybernetics*, 45(4):716–727, 2015.

[92] Haibo Zhang and GP Rangaiah. An efficient constraint handling method with integrated differential evolution for numerical and engineering optimization. *Computers & Chemical Engineering*, 37:74–88, 2012.

[93] Guohua Wu, Witold Pedrycz, Ponnuthurai N Suganthan, and Rammohan Mallipeddi. A variable reduction strategy for evolutionary algorithms handling equality constraints. *Applied Soft Computing*, 37:774–786, 2015.

[94] Manjaree Pandit, Laxmi Srivastava, and Manisha Sharma. Environmental economic dispatch in multi-area power system employing improved differential evolution with fuzzy selection. *Applied Soft Computing*, 28:498–510, 2015.

[95] Xiangyin Zhang and Haibin Duan. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Applied Soft Computing*, 26:270–284, 2015.

[96] Yong Wang and Zixing Cai. Constrained evolutionary optimization by means of ($\mu+ \lambda$)-differential evolution and improved adaptive trade-off model. *Evolutionary Computation*, 19(2):249–285, 2011.

[97] Guanbo Jia, Yong Wang, Zixing Cai, and Yaochu Jin. An improved ($\mu+ \lambda$)-constrained differential evolution for constrained optimization. *Information Sciences*, 222:302–322, 2013.

[98] Chenyang Bu, Wenjian Luo, and Tao Zhu. Differential evolution with a species-based repair strategy for constrained optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 967–974. IEEE, 2014.

[99] Xiaodong Li. Efficient differential evolution using speciation for multimodal function optimization. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 873–880. ACM, 2005.

[100] María-Yaneli Ameca-Alducin, Efrén Mezura-Montes, and Nicandro Cruz-Ramírez. Differential evolution with a repair method to solve dynamic constrained optimization problems. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1169–1172. ACM, 2015.

[101] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Computers & operations research*, 38(12):1877–1896, 2011.

[102] Rammohan Mallipeddi and Minho Lee. An evolving surrogate model-based differential evolution algorithm. *Applied Soft Computing*, 34:770–787, 2015.

[103] Junli Zhang, Yongquan Zhou, and Hui Deng. Hybridizing particle swarm optimization with differential evolution based on feasibility rules. In *2012 International Conference on Graphic and Image Processing*, pages 876807–876807. International Society for Optics and Photonics, 2013.

[104] Ling Wang and Ling-Po Li. Fixed-structure h controller synthesis based on differential evolution with level comparison. *IEEE Transactions on Evolutionary Computation*, 15(1):120–129, 2011.

[105] Magdi A Mohamed and Paul Gader. Generalized hidden markov models. i. theoretical frameworks. *IEEE Transactions on fuzzy systems*, 8(1):67–81, 2000.

[106] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[107] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

[108] Mars Xu and Yue-Xia Wang. Quantifying pm 2.5 concentrations from multi-weather sensors using hidden markov models. *IEEE Sensors Journal*, 16(1):22–23, 2016.

[109] Maryam Farshchian and Majid Vafaei Jahan. Stock market prediction with hidden markov model. In *Technology, Communication and Knowledge (ICTCK), 2015 International Congress on*, pages 473–477. IEEE, 2015.

[110] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International journal of pattern recognition and artificial intelligence*, 15(01):9–42, 2001.

[111] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[112] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[113] Alex Endert, Chris North, Remco Chang, and Michelle Zhou. Toward usable interactive analytics: Coupling cognition and computation. In *Proceedings of the 2014 Workshop on Interactive Data Exploration and Analytics at KDD (IDEA)*, 2014.

[114] Namrata Khemka and Christian Jacob. Visplore: a toolkit to explore particle swarms by visual inspection. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 41–48. ACM, 2009.

[115] Joerg Meyer, Jim Thomas, Stephan Diehl, Brian Fisher, and Daniel A Keim. From visualization to visually enabled reasoning. In *Dagstuhl Follow-Ups*, volume 1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[116] William M Spears. An overview of multidimensional visualization techniques. In *Evolutionary Computation Visualization Workshop*, 1999.

[117] Teemu Peltonen. Comparative study of population-based metaheuristic methods in global optimization. 2015.

[118] Torre Zuk and Sheelagh Carpendale. Visualization of uncertainty and reasoning. In *Smart graphics*, pages 164–177. Springer, 2007.

[119] Trevor D Collins. Understanding evolutionary computing: A hands on approach. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 564–569. IEEE, 1998.

[120] Willis F Overton. *Reasoning, necessity, and logic: Developmental perspectives*. Psychology Press, 2013.

[121] Oliver Wilhelm. Measuring reasoning ability. *Handbook of understanding and measuring intelligence*, pages 373–392, 2005.

[122] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek*, volume 27. Cambridge University Press, 2011.

[123] Aiso Heinze and Kristina Reiss. Reasoning and proof in the mathematics classroom. *Analysis*, 27(2-3):333–357, 2007.

[124] David Moshman and Pina Tarricone. Logical and causal reasoning. *Handbook of epistemic cognition*, pages 54–67, 2016.

[125] Wim Staat. On abduction, deduction, induction and the categories. *Transactions of the Charles S. Peirce Society*, 29(2):225–237, 1993.

[126] Andrés Rivadulla. Abductive reasoning, theoretical preduction, and the physical way of dealing fallibly with nature. *Abduction and the process of scientific discovery*, pages 199–210, 2007.

[127] Trevor D Collins. Visualizing evolutionary computation. In *Advances in evolutionary computing*, pages 95–116. Springer, 2003.

[128] Gustavo Romero, JJ Merelo, PA Castillo, JG Castellano, and Maribel García Arenas. Genetic algorithm visualization using self-organizing maps. In *International Conference on Parallel Problem Solving from Nature*, pages 442–451. Springer, 2002.

[129] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013.

[130] Trevor D Collins. Using software visualisation technology to help evolutionary algorithm users validate their solutions. In *ICGA*, pages 307–314, 1997.

[131] Hartmut Pohlheim. Visualization of evolutionary algorithms-set of standard techniques and multidimensional visualization. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 533–540. Morgan Kaufmann Publishers Inc., 1999.

[132] Richard Dybowski, Trevor D Collins, and PR Weller. Visualization of binary string convergence by sammon mapping. In *Evolutionary Programming*, pages 377–383. Citeseer, 1996.

[133] Andrew Chipperfield, Peter Fleming, Hartmut Pohlheim, and Carlos Fonseca. Genetic algorithm toolbox for use with matlab. 1994.

[134] Annie S Wu, Kenneth A De Jong, Donald S Burke, John J Grefenstette, and C Loggia Ramsey. Visual analysis of evolutionary algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 1419–1425. IEEE, 1999.

[135] Emma Hart and Peter Ross. Gavel-a new tool for genetic algorithm visualization. *IEEE Transactions on Evolutionary Computation*, 5(4):335–348, 2001.

[136] Christian Collberg, Stephen Kobourov, Jasvir Nagra, Jacob Pitts, and Kevin Wampler. A system for graph-based visualization of the evolution of software. In *Proceedings of the 2003 ACM symposium on Software visualization*, pages 77–ff. ACM, 2003.

[137] Filip Van Rysselberghe and Serge Demeyer. Studying software evolution information by visualizing the change history. In *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, pages 328–337. IEEE, 2004.

[138] Ahmed E Hassan, Jingwei Wu, and Richard C Holt. Visualizing historical data using spectrographs. In *Software Metrics, 2005. 11th IEEE International Symposium*, pages 10–pp. IEEE, 2005.

[139] William B Shine and CF Eick. Visualizing the evolution of genetic algorithm search processes. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 367–372. IEEE, 1997.

[140] Alan RR de Freitas, Peter J Fleming, and Frederico G Guimarães. Aggregation trees for visualization and dimension reduction in many-objective optimization. *Information Sciences*, 298:288–314, 2015.

[141] Jacek Ratzinger, Michael Fischer, and Harald Gall. Evolens: Lens-view visualizations of evolution data. In *Principles of Software Evolution, Eighth International Workshop on*, pages 103–112. IEEE, 2005.

[142] Michael Barlow, John Galloway, and Hussein A Abbass. Mining evolution through visualization. In *ALife VIII workshops*, pages 103–112. Citeseer, 2002.

[143] Andreas Kerren and Thomas Egger. Eavis: A visualization tool for evolutionary algorithms. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 299–301. IEEE, 2005.

[144] Guillaume Jornod, Ezequiel Di Mario, Inaki Navarro, and Alcherio Martinoli. Swarmviz: An open-source visualization tool for particle swarm optimization. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 179–186. IEEE, 2015.

[145] Trevor Paterson, Martin Graham, Jessie Kennedy, and Andy Law. Evaluating the viper pedigree visualisation: Detecting inheritance inconsistencies in genotyped pedigrees. In *Biological Data Visualization (BioVis), 2011 IEEE Symposium on*, pages 119–126. IEEE, 2011.

[146] J Martin Bland and Douglas G Altman. Calculating correlation coefficients with repeated observations: Part 2âĂŤcorrelation between subjects. *Bmj*, 310(6980):633, 1995.

[147] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013.

[148] Zhi-Zhong Liu, Yong Wang, Shengxiang Yang, and Zixing Cai. Differential evolution with a two-stage optimization mechanism for numerical optimization. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 3170–3177. IEEE, 2016.

[149] David E Goldberg and Philip Segrest. Finite markov chain analysis of genetic algorithms. In *Proceedings of the second international conference on genetic algorithms*, volume 1, page 1, 1987.

[150] Samir W Mahfoud. Finite markov chain models of an alternative selection strategy for the genetic algorithm. *Complex systems*, 7(2):155, 1993.

[151] Thomas E Davis and Jose C Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary computation*, 1(3):269–288, 1993.

[152] Günter Rudolph. Finite markov chain results in evolutionary computation: a tour d'horizon. *Fundamenta informaticae*, 35(1-4):67–89, 1998.

[153] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[154] Jun He and Xin Yao. Average drift analysis and population scalability. *IEEE Transactions on Evolutionary Computation*, 21(3):426–439, 2017.

[155] Jackie Rees and Gary J Koehler. Learning genetic algorithm parameters using hidden markov models. *European Journal of Operational Research*, 175(2):806–820, 2006.

[156] Flore Harlé, Florent Chatelain, Cédric Gouy-Pailler, and Sophie Achard. Bayesian model for multiple change-points detection in multivariate time series. *IEEE Trans. Signal Processing*, 64(16):4351–4362, 2016.

[157] Meng-Lin Ku, Yan Chen, and KJ Ray Liu. Data-driven stochastic models and policies for energy harvesting sensor communications. *IEEE Journal on Selected Areas in Communications*, 33(8):1505–1520, 2015.

[158] Hiroshi Morimoto. Hidden markov models and self-organizing maps applied to stroke incidence. *Open Journal of Applied Sciences*, 6(03):158, 2016.

[159] Yi Cao, Yuhua Li, Sonya Coleman, Ammar Belatreche, and Thomas Martin McGinnity. Adaptive hidden markov model with anomaly states for price

manipulation detection. *IEEE transactions on neural networks and learning systems*, 26(2):318–330, 2015.

[160] Klas EG Magnusson, Joakim Jaldén, Penney M Gilbert, and Helen M Blau. Global linking of cell tracks using the viterbi algorithm. *IEEE transactions on medical imaging*, 34(4):911–929, 2015.

[161] H-L Lou. Implementing the viterbi algorithm. *IEEE Signal processing magazine*, 12(5):42–52, 1995.

[162] Markus Rickert, Oliver Brock, and Alois Knoll. Balancing exploration and exploitation in motion planning. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2812–2817. IEEE, 2008.

[163] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL report*, 2005005:2005, 2005.

[164] JJ Liang, BY Qu, PN Suganthan, and Q Chen. Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2014.

[165] Niki Veček, Matej Črepinšek, and Marjan Mernik. On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms. *Applied Soft Computing*, 54:23–45, 2017.

[166] The spss statistical tool. November 2016.

[167] Niki Veček, Marjan Mernik, and Matej Črepinšek. A chess rating system for evolutionary algorithms: a new method for the comparison and ranking of evolutionary algorithms. *Information Sciences*, 277:656–679, 2014.

[168] Jing J Liang, A Kai Qin, Ponnuthurai N Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3):281–295, 2006.

[169] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[170] Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776. IEEE, 2005.

[171] A Kai Qin, Ke Tang, Hong Pan, and Siyu Xia. Self-adaptive differential evolution with local search chains for real-parameter single-objective optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 467–474. IEEE, 2014.

[172] Ferrante Neri and Ville Tirronen. Scale factor local search in differential evolution. *Memetic Computing*, 1(2):153–171, 2009.

[173] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 24, 2010.

[174] Saúl Domınguez-Isidro, Efrén Mezura-Montes, and Guillermo Leguizamón. Performance comparison of local search operators in differential evolution for constrained numerical optimization problems. In *Differential Evolution (SDE), 2014 IEEE Symposium on*, pages 1–8. IEEE, 2014.

[175] Ferrante Neri, Jari Toivanen, Giuseppe Leonardo Cascella, and Yew-Soon Ong. An adaptive multimeme algorithm for designing hiv multidrug therapies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2), 2007.

[176] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithmsâĂŹ behaviour: a case study on the cecâĂŹ2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009.

[177] Jing J Liang, Shang Zhigang, and Li Zhihui. Coevolutionary comprehensive learning particle swarm optimizer. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[178] Efrén Mezura-Montes and Ramiro Ernesto Velez-Koeppel. Elitist artificial bee colony for constrained real-parameter optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[179] Janez Brest, Borko Bošković, and Viljem Žumer. An improved self-adaptive differential evolution algorithm in single objective constrained real-parameter optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

Appendix A

CEC2005 Comparison of DE-HMM different variants (different parameter-choices)

Table A.1: Mean function error values among DE-HMM variants at 30D of CEC2005 over 30 independent runs. (DE-HMM-minL - DE-HMM-meanE)

	DE-HMM-minL	DE-HMM-minE	DE-HMM-meanG	DE-HMM-meanL	DE-HMM-meanE
	mean	mean	mean	mean	mean
C1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
C2	**0.00E+00**	6.35E-05	5.84E+00	6.37E+00	6.35E+02
C3	**2.94E+05**	2.72E+06	2.79E+06	3.02E+06	3.89E+07
C4	**3.99E-04**	6.04E-02	7.77E+01	6.88E+01	1.88E+03
C5	3.65E+02	6.56E+02	6.34E+01	5.09E+01	5.12E+01
C6	7.11E+01	4.85E+04	1.48E+01	1.58E+01	**7.46E+00**
C7	**4.70E+03**	**4.70E+03**	**4.70E+03**	**4.70E+03**	**4.70E+03**
C8	**2.09E+01**	**2.09E+01**	2.10E+01	**2.09E+01**	2.10E+01
C9	**1.14E+01**	3.76E+01	7.84E+01	7.50E+01	1.50E+02
C10	9.18E+01	1.68E+02	2.02E+02	2.07E+02	1.95E+02
C11	2.13E+01	3.93E+01	3.92E+01	3.97E+01	3.94E+01
C12	4.01E+03	5.28E+03	**2.04E+03**	2.28E+03	5.18E+04
C13	5.82E+00	1.28E+01	1.52E+01	1.55E+01	1.67E+01
C14	1.31E+01	1.32E+01	1.33E+01	1.34E+01	1.33E+01
C15	3.42E+02	3.70E+02	3.77E+02	3.97E+02	3.43E+02
C16	1.06E+02	1.92E+02	2.22E+02	2.33E+02	2.26E+02
C17	2.20E+02	2.23E+02	2.48E+02	2.57E+02	2.56E+02
C18	**9.05E+02**	**9.05E+02**	**9.05E+02**	**9.05E+02**	9.06E+02
C19	**9.05E+02**	**9.05E+02**	**9.05E+02**	**9.05E+02**	9.06E+02
C20	9.05E+02	9.05E+02	9.05E+02	9.05E+02	9.06E+02
C21	5.10E+02	**5.00E+02**	**5.00E+02**	**5.00E+02**	**5.00E+02**
C22	8.83E+02	8.83E+02	8.83E+02	8.85E+02	8.85E+02
C23	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**	**5.34E+02**
C24	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**
C25	**1.63E+03**	**1.63E+03**	1.64E+03	1.64E+03	1.64E+03
No-Best	12	9	8	8	6

Table A.2: Mean function error values among DE-HMM variants at 30D of CEC2005 over 30 independent runs. (DE-HMM-maxG - DE-HMM-minG)

	DE-HMM-maxG	DE-HMM-maxL	DE-HMM-maxE	Rand-gen	DE-HMM-minG
	mean	mean	mean	mean	mean
C1	1.16E+01	1.16E+01	2.07E+01	**0.00E+00**	**0.00E+00**
C2	6.27E+03	7.10E+03	1.49E+04	1.13E-03	**0.00E+00**
C3	4.67E+07	3.17E+07	1.12E+08	9.56E+06	3.56E+05
C4	1.65E+04	1.41E+04	1.98E+04	5.39E-01	1.49E-03
C5	3.94E+03	3.32E+03	5.61E+03	**1.09E+02**	1.83E+02
C6	1.43E+04	1.26E+04	1.66E+05	3.31E+01	2.19E+02
C7	**4.70E+03**	**4.70E+03**	**4.70E+03**	**4.70E+03**	**4.70E+03**
C8	2.10E+01	**2.09E+01**	2.09E+01	2.10E+01	**2.09E+01**
C9	7.70E+01	7.61E+01	1.65E+02	1.94E+01	1.21E+01
C10	2.54E+02	2.48E+02	2.33E+02	1.73E+02	**2.44E+01**
C11	3.91E+01	3.97E+01	3.92E+01	3.89E+01	**7.53E+00**
C12	8.08E+04	7.59E+04	4.59E+05	2.53E+03	3.11E+03
C13	1.75E+01	1.65E+01	2.30E+01	1.36E+01	**2.68E+00**
C14	1.35E+01	1.35E+01	1.35E+01	1.32E+01	**1.29E+01**
C15	4.21E+02	4.12E+02	4.20E+02	3.57E+02	**3.37E+02**
C16	2.69E+02	2.76E+02	2.55E+02	1.91E+02	**4.43E+01**
C17	3.05E+02	3.02E+02	2.93E+02	2.28E+02	**2.18E+02**
C18	9.08E+02	9.08E+02	9.07E+02	9.04E+02	**9.05E+02**
C19	9.08E+02	9.08E+02	9.07E+02	9.04E+02	**9.05E+02**
C20	9.08E+02	9.08E+02	9.07E+02	**9.04E+02**	**9.04E+02**
C21	5.02E+02	5.01E+02	5.04E+02	**5.00E+02**	**5.00E+02**
C22	9.33E+02	9.31E+02	9.39E+02	8.83E+02	**8.76E+02**
C23	**5.34E+02**	**5.34E+02**	5.43E+02	**5.34E+02**	**5.34E+02**
C24	2.04E+02	2.03E+02	2.08E+02	**2.00E+02**	**2.00E+02**
C25	1.66E+03	1.66E+03	1.65E+03	**1.63E+03**	**1.63E+03**
No-Best	2	3	1	9	20

Table A.3: Comparison summary among DE-HMM and its variants and Rand-gen method on 30D CEC2005 test problems.

30D					
DE-HMM "DE-HMM-minG"	"Better"	"Worse"	"Equal"	"p-value"	"Significance"
DE-HMM-minL	**14**	4	7	**0.0265**	+
DE-HMM-minE	**17**	0	8	**0.0018**	+
DE-HMM-meanG	**16**	3	6	**0.032**	+
DE-HMM-meanL	**15**	3	7	**0.0425**	+
DE-HMM-meanE	**19**	2	4	**0.0013**	+
DE-HMM-maxG	**23**	0	2	**0.0015**	+
DE-HMM-maxL	**22**	0	3	**0.002**	+
DE-HMM-maxE	**23**	0	2	**0.003**	+
Rand-gen	**13**	5	7	0.107	=

Appendix B

Analysis of using different search operators on DE-HMM

Table B.1: Effect of different search operators on DE-HMM at 30D CEC2005 mean function error over 30 independent runs.

function	30D			
	/rand/binomial	/rand/exponential	/best/binomial	/best/exponential
	Mean	Mean	Mean	Mean
C1	**0.00E+00**	2.68E+00	**0.00E+00**	**0.00E+00**
C2	**0.00E+00**	2.75E-03	**0.00E+00**	**0.00E+00**
C3	**3.56E+05**	5.75E+05	5.96E+05	1.15E+06
C4	**1.49E-03**	2.41E+00	1.02E-02	1.85E+00
C5	1.83E+02	1.81E+03	**1.01E+02**	1.39E+03
C6	2.19E+02	4.73E+06	**1.50E+00**	7.86E+00
C7	**4.70E+03**	**4.70E+03**	**4.70E+03**	**4.70E+03**
C8	**2.09E+01**	**2.09E+01**	2.10E+01	**2.09E+01**
C9	**1.21E+01**	2.48E+01	1.57E+01	2.21E+01
C10	**2.44E+01**	4.52E+01	1.20E+02	3.66E+01
C11	**7.53E+00**	1.96E+01	2.91E+01	2.20E+01
C12	3.11E+03	1.22E+04	**2.84E+03**	3.58E+03
C13	**2.68E+00**	3.03E+00	8.38E+00	2.72E+00
C14	1.29E+01	**1.25E+01**	1.31E+01	1.27E+01
C15	**3.37E+02**	3.97E+02	3.67E+02	3.88E+02
C16	**4.43E+01**	1.12E+02	1.25E+02	6.81E+01
C17	2.18E+02	**6.96E+01**	2.18E+02	8.35E+01
C18	**9.05E+02**	**9.05E+02**	**9.05E+02**	**9.05E+02**
C19	**9.05E+02**	9.09E+02	**9.05E+02**	**9.05E+02**
C20	**9.04E+02**	9.09E+02	**9.04E+02**	9.06E+02
C21	**5.00E+02**	5.47E+02	**5.00E+02**	5.40E+02
C22	**8.76E+02**	8.91E+02	8.79E+02	8.87E+02
C23	**5.34E+02**	6.24E+02	5.61E+02	5.50E+02
C24	**2.00E+02**	**2.00E+02**	**2.00E+02**	**2.00E+02**
C25	**1.63E+03**	**1.63E+03**	**1.63E+03**	**1.63E+03**
No-Best	20	7	12	8

Table B.2: Effect of different search operators on DE-HMM at 30D CEC2014 mean function error over 30 independent runs.

function	30D			
	/rand/binomial	/rand/exponential	/best/binomial	/best/exponential
	Mean	Mean	Mean	Mean
F1	**9.58E+04**	2.74E+05	1.19E+05	3.41E+05
F2	1.30E+03	4.41E+06	**0.00E+00**	**0.00E+00**
F3	2.35E-01	2.38E+02	**0.00E+00**	**0.00E+00**
F4	3.25E+01	7.70E+01	**2.84E-03**	2.63E+00
F5	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
F6	1.46E+00	4.04E+00	**9.56E-01**	3.99E+00
F7	**0.00E+00**	3.71E-02	2.47E-04	5.83E-03
F8	**1.10E+01**	2.36E+01	1.44E+01	1.96E+01
F9	**1.77E+01**	3.64E+01	4.52E+01	3.05E+01
F10	**2.13E+01**	5.42E+02	1.11E+03	4.59E+02
F11	6.19E+03	**2.73E+03**	6.49E+03	**2.73E+03**
F12	2.43E+00	**1.46E+00**	2.51E+00	2.16E+00
F13	**2.59E-01**	**2.59E-01**	**2.59E-01**	**2.59E-01**
F14	2.82E-01	2.64E-01	2.77E-01	**2.54E-01**
F15	1.31E+01	3.83E+00	1.33E+01	**3.29E+00**
F16	1.20E+01	**1.08E+01**	1.18E+01	1.11E+01
F17	**1.13E+03**	7.76E+03	1.52E+03	2.00E+03
F18	**1.33E+01**	7.26E+01	2.50E+01	5.52E+01
F19	**3.50E+00**	5.39E+00	4.76E+00	4.13E+00
F20	**9.07E+00**	2.84E+01	1.10E+01	1.93E+01
F21	2.71E+02	2.95E+02	**2.24E+02**	2.95E+02
F22	**6.87E+01**	1.46E+02	7.10E+01	6.93E+01
F23	**3.15E+02**	**3.15E+02**	**3.15E+02**	**3.15E+02**
F24	**2.25E+02**	2.33E+02	2.26E+02	2.28E+02
F25	**2.03E+02**	2.04E+02	**2.03E+02**	2.04E+02
F26	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**
F27	**3.59E+02**	4.17E+02	**3.59E+02**	3.95E+02
F28	**8.27E+02**	8.96E+02	8.45E+02	8.72E+02
F29	9.75E+02	6.83E+05	**8.05E+02**	8.62E+02
F30	1.33E+03	1.70E+03	**8.81E+02**	1.17E+03
No-Best	18	7	13	9

Appendix C

CEC2005 Comparison of DE-HMM and C-DE on CEC2005 and CEC2014 datasets

Table C.1: Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2005 25-test functions with 10, 30, and 50 dimensions (C1-C15)

	10D		30D		50D	
	C-DE	DE-HMM	C-DE	DE-HMM	C-DE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.15E-04 (9.65E-05)	**0.00E+00 (0.00E+00)**	7.47E+00 (4.56E+00)	**1.05E-04 (1.00E-04)**
C3	**0.00E+00 (0.00E+00)**	1.73E-01 (5.20E-02)	1.37E+06 (9.68E+05)	**3.56E+05 (1.67E+04)**	1.57E+07 (6.03E+06)	**4.51E+05 (2.09E+04)**
C4	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	3.86E-02 (3.58E-02)	**1.49E-03 (1.05E-03)**	5.26E+02 (2.69E+02)	**1.27E+02 (1.50E+02)**
C5	8.57E-05 (3.61E-05)	**0.00E+00 (0.00E+00)**	2.31E+02 (1.63E+02)	**1.83E+02 (1.55E+02)**	4.58E+03 (5.01E+02)	**2.98E+03 (4.87E+02)**
C6	**8.67E-05 (4.66E-04)**	4.99E+00 (1.61E+00)	**2.84E+00 (2.22E+00)**	2.19E+02 (5.76E+02)	**2.81E+01 (1.46E+01)**	3.05E+05 (9.48E+03)
C7	**2.93E-01 (1.41E-01)**	1.27E+03 (1.69E-13)	**3.29E-04 (1.80E-03)**	4.70E+03 (2.82E-12)	**1.23E-03 (3.28E-03)**	6.20E+03 (3.44E-12)
C8	**2.03E+01 (8.86E-02)**	**2.03E+01 (9.46E-02)**	2.10E+01 (6.33E-02)	**2.09E+01 (4.74E-02)**	**2.11E+01 (3.65E-02)**	**2.11E+01 (3.97E-02)**
C9	1.87E+01 (2.70E+00)	**0.00E+00 (0.00E+00)**	1.34E+02 (2.23E+01)	**1.21E+01 (3.28E+00)**	2.02E+02 (5.31E+01)	**3.40E+01 (7.46E+00)**
C10	2.65E+01 (3.01E+00)	**5.17E+00 (1.95E+00)**	1.81E+02 (6.89E+00)	**2.44E+01 (1.03E+01)**	3.55E+02 (1.29E+01)	**9.15E+01 (9.39E+00)**
C11	8.58E+00 (5.60E-01)	**1.61E-01 (3.37E-03)**	3.93E+01 (9.21E-01)	**7.53E+00 (2.02E+00)**	7.28E+01 (1.70E+00)	**4.59E+01 (1.99E+01)**
C12	1.04E+04 (5.83E+03)	**6.65E-06 (3.64E-05)**	7.88E+05 (1.70E+05)	**3.11E+03 (3.32E+03)**	2.96E+06 (5.57E+05)	**1.82E+04 (1.07E+04)**
C13	2.01E+00 (3.64E-01)	**1.65E+00 (4.56E-01)**	1.51E+01 (9.96E-01)	**2.68E+00 (8.11E-01)**	3.05E+01 (8.24E-01)	**4.77E+00 (7.74E-01)**
C14	3.65E+00 (2.16E-01)	**2.98E+00 (2.10E-01)**	1.38E+01 (2.16E-01)	**1.29E+01 (2.08E-01)**	2.37E+01 (1.90E-01)	**2.28E+01 (2.10E-01)**
C15	3.49E+02 (1.07E+02)	**2.65E+02 (1.52E+02)**	7.83E+02 (5.60E+01)	**3.37E+02 (1.07E+01)**	8.28E+02 (4.90E+01)	**3.43E+02 (2.02E+01)**

Table C.2: Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2005 25-test functions with 10, 30, and 50 dimensions (C16-C25)

	10D		30D		50D	
	C-DE	DE-HMM	C-DE	DE-HMM	C-DE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C16	1.49E+02 (1.00E+01)	**9.41E+01 (1.88E+00)**	2.29E+02 (4.85E+01)	**4.43E+01 (8.77E+00)**	2.68E+02 (2.52E+01)	**8.97E+01 (3.34E+01)**
C17	1.66E+02 (1.00E+01)	**1.17E+02 (3.36E+01)**	2.52E+02 (4.01E+01)	**2.18E+02 (5.66E+01)**	**2.81E+02 (2.61E+01)**	2.83E+02 (1.17E+00)
C18	8.25E+02 (2.13E+01)	**3.00E+02 (0.00E+00)**	**8.17E+02 (4.04E-01)**	9.05E+02 (2.07E+00)	**8.37E+02 (1.97E-01)**	9.23E+02 (4.15E+00)
C19	8.21E+02 (4.90E-02)	**4.33E+02 (2.25E+01)**	**8.17E+02 (4.60E-01)**	9.05E+02 (1.75E+00)	**8.36E+02 (1.49E-01)**	9.20E+02 (1.79E-02)
C20	8.33E+02 (3.56E+01)	**4.18E+02 (2.14E+02)**	**8.17E+02 (5.16E-01)**	9.04E+02 (1.41E+00)	**8.37E+02 (1.58E-01)**	9.21E+02 (1.08E-01)
C21	1.02E+03 (1.42E+02)	**4.67E+02 (7.58E-01)**	8.57E+02 (3.29E-01)	**5.00E+02 (1.95E-13)**	7.18E+02 (2.00E+00)	**6.69E+02 (2.44E+02)**
C22	8.02E+02 (1.27E+02)	**7.65E+02 (4.08E+00)**	9.00E+02 (1.85E-01)	**8.76E+02 (1.43E+01)**	**5.00E+02 (7.65E-02)**	9.19E+02 (1.97E+01)
C23	1.07E+03 (6.75E+01)	**5.65E+02 (2.95E+01)**	8.65E+02 (5.25E-01)	**5.34E+02 (3.12E-04)**	7.23E+02 (1.17E+00)	**6.87E+02 (2.21E+02)**
C24	3.75E+02 (2.22E+00)	**2.00E+02 (0.00E+00)**	2.09E+02 (3.18E-01)	**2.00E+02 (2.89E-14)**	2.14E+02 (7.90E-01)	**2.00E+02 (1.66E-12)**
C25	**3.74E+02 (2.90E+00)**	1.73E+03 (4.23E+00)	**2.09E+02 (3.82E-01)**	1.63E+03 (4.90E-01)	**2.14E+02 (5.27E-01)**	1.66E+03 (5.69E+00)
No-Best	8	21	7	19	10	17

Table C.3: Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2014 30-test functions with 10, 30, and 50 dimensions (F1-F15)

	10D		30D		50D	
	C-DE	DE-HMM	C-DE	DE-HMM	C-DE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**8.85E+04 (1.31E+05)**	9.58E+04 (8.97E+03)	**5.58E+05 (1.89E+05)**	**5.58E+05 (1.67E+04)**
F2	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	5.58E+03 (2.62E+03)	**1.30E+03 (2.35E-03)**	4.72E+03 (5.41E+03)	**1.81E+03 (2.95E+01)**
F3	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.16E+01 (2.01E+00)	**2.35E-01 (2.25E-01)**	**2.42E+02 (2.30E+02)**	3.80E+03 (2.93E-02)
F4	1.48E+01 (1.67E+01)	**7.25E+00 (1.27E-01)**	**7.44E+00 (2.02E+01)**	3.25E+01 (3.64E+00)	7.86E+01 (2.70E+01)	**6.28E+01 (3.21E+01)**
F5	2.00E+01 (5.21E+00)	**1.63E+01 (8.28E+00)**	2.10E+01 (5.21E-02)	**2.09E+01 (3.36E-02)**	2.11E+01 (3.10E-02)	**2.11E+01 (3.76E-02)**
F6	1.49E-01 (3.39E-01)	**0.00E+00 (0.00E+00)**	1.91E+00 (9.76E-01)	**1.46E+00 (1.25E+00)**	7.23E+00 (1.68E+00)	**4.46E+00 (1.74E+00)**
F7	3.26E-02 (2.53E-02)	**6.24E-03 (1.28E-03)**	6.57E-04 (2.50E-03)	**0.00E+00 (0.00E+00)**	4.27E-03 (6.23E-03)	**0.00E+00 (0.00E+00)**
F8	2.96E+00 (4.82E+00)	**0.00E+00 (0.00E+00)**	1.42E+01 (4.93E+00)	**1.10E+01 (2.75E+00)**	5.97E+01 (7.18E+00)	**3.39E+01 (8.89E+00)**
F9	7.98E+00 (5.21E+00)	**2.71E+00 (1.19E+00)**	9.06E+01 (6.49E+01)	**1.77E+01 (4.03E+00)**	1.60E+02 (1.40E+02)	**4.85E+01 (1.19E+01)**
F10	1.32E+02 (2.07E+02)	**8.92E+01 (2.07E+00)**	3.20E+02 (4.33E+02)	**2.13E+01 (1.19E-01)**	8.24E+02 (3.26E+02)	**7.49E+02 (3.66E+02)**
F11	7.01E+02 (3.58E+02)	**3.27E+02 (3.50E-02)**	6.59E+03 (3.64E+02)	**6.19E+03 (1.21E-07)**	1.30E+04 (3.15E+02)	**1.20E+04 (2.30E-13)**
F12	**5.65E-01 (2.64E-01)**	1.05E+00 (2.65E-01)	2.67E+00 (5.64E-01)	**2.43E+00 (4.59E-01)**	3.64E+00 (3.05E-01)	**3.39E+00 (2.67E-01)**
F13	1.23E-01 (3.61E-02)	**8.49E-02 (1.27E-02)**	2.90E-01 (5.01E-02)	**2.59E-01 (3.44E-02)**	4.13E-01 (4.56E-02)	**3.68E-01 (5.26E-02)**
F14	1.56E-01 (4.33E-02)	**1.47E-01 (3.94E-02)**	2.84E-01 (3.74E-02)	**2.82E-01 (2.81E-02)**	3.50E-01 (1.47E-01)	**3.17E-01 (2.89E-02)**
F15	1.64E+00 (4.45E-01)	**1.57E+00 (4.42E-01)**	1.46E+01 (1.08E+00)	**1.31E+01 (2.71E+00)**	2.96E+01 (1.58E+00)	**5.85E+00 (1.05E+00)**

Table C.4: Mean and standard deviation (Mean and std) function error values of DE-HMM and C-DE over 30 independant runs on CEC2014 30-test functions with 10, 30, and 50 dimensions (F16-F30)

	10D		30D		50D	
	C-DE	DE-HMM	C-DE	DE-HMM	C-DE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F16	2.14E+00 (6.86E-01)	**1.97E+00 (3.17E-01)**	1.21E+01 (3.87E-01)	**1.20E+01 (3.33E-01)**	2.20E+01 (2.68E-01)	**2.15E+01 (4.36E-01)**
F17	1.15E+01 (3.16E+01)	**4.56E-01 (4.64E-01)**	1.87E+03 (1.76E+03)	**1.13E+03 (8.16E+02)**	3.66E+04 (1.87E+04)	**3.33E+04 (2.17E+04)**
F18	4.71E-01 (5.61E-01)	**1.81E-01 (2.24E-01)**	1.55E+01 (1.07E+01)	**1.33E+01 (9.31E+00)**	6.29E+02 (9.00E+02)	**1.55E+02 (8.05E+01)**
F19	1.97E-01 (1.31E-01)	**1.30E-01 (1.07E-01)**	3.66E+00 (1.21E+00)	**3.50E+00 (8.90E-01)**	3.19E+01 (1.26E+01)	**1.28E+01 (4.13E+00)**
F20	1.74E-01 (1.89E-01)	**5.67E-02 (9.39E-02)**	1.02E+01 (3.92E+00)	**9.07E+00 (3.91E+00)**	3.58E+02 (4.27E+02)	8.04E+02 (1.81E+01)
F21	1.09E+00 (3.07E+00)	**3.10E-01 (2.82E-01)**	3.18E+02 (1.17E+02)	**2.71E+02 (2.02E+02)**	5.14E+04 (2.23E+04)	**4.00E+04 (3.87E+04)**
F22	2.91E-01 (2.41E-01)	**7.88E-02 (1.62E-01)**	6.96E+01 (9.21E+01)	**6.87E+01 (7.15E+01)**	7.04E+02 (3.50E+02)	**2.35E+02 (1.47E+02)**
F23	**3.29E+02 (2.89E-13)**	**3.29E+02 (2.89E-13)**	3.15E+02 (1.45E-13)	**3.15E+02 (3.72E-02)**	3.44E+02 (2.89E-13)	**3.44E+02 (2.66E-03)**
F24	1.11E+02 (4.63E+00)	**1.09E+02 (1.95E+00)**	2.26E+02 (4.32E+00)	**2.25E+02 (1.94E+00)**	2.73E+02 (2.67E+00)	2.76E+02 (2.53E+00)
F25	1.69E+02 (4.10E+01)	**1.59E+02 (4.44E+01)**	**2.03E+02 (5.41E-01)**	**2.03E+02 (4.03E-01)**	2.09E+02 (7.52E-01)	**2.08E+02 (1.53E+00)**
F26	**1.00E+02 (4.12E-02)**	**1.00E+02 (2.45E-02)**	**1.00E+02 (3.77E-02)**	**1.00E+02 (3.25E-12)**	1.04E+02 (1.82E+01)	**1.00E+02 (4.62E-02)**
F27	**9.47E+01 (1.71E+02)**	1.41E+02 (1.64E+02)	3.68E+02 (4.70E+01)	**3.59E+02 (4.70E+01)**	5.57E+02 (5.65E+01)	**5.00E+02 (5.15E+00)**
F28	3.88E+02 (4.29E+01)	**3.64E+02 (5.59E+00)**	8.40E+02 (4.90E+01)	**8.27E+02 (4.82E+01)**	1.19E+03 (5.34E+01)	**1.17E+03 (6.91E+01)**
F29	2.19E+02 (1.54E+01)	**2.17E+02 (1.73E+01)**	**8.07E+02 (1.37E+02)**	9.75E+02 (2.20E+02)	1.35E+03 (2.30E+02)	1.20E+06 (6.58E+06)
F30	**4.68E+02 (2.03E+01)**	4.70E+02 (1.81E+01)	**1.02E+03 (4.98E+02)**	1.33E+03 (5.87E+01)	9.02E+03 (5.75E+02)	9.57E+03 (6.52E+02)
No-Best	8	27	6	26	2	25

Appendix D

CEC2005 Comparison of DE-HMM and State-of-the-art algorithms

Table D.1: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 10D over 30 independent runs (C1-C15).

10D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C3	3.30E+03 (4.95E+03)	2.13E-06 (7.71E-06)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.73E-01 (5.20E-02)
C4	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C5	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C6	5.49E+00 (1.44E+00)	3.05E-02 (5.22E-02)	5.52E+00 (2.23E+00)	**1.15E-07 (5.48E-07)**	4.99E+00 (1.61E+00)
C7	1.27E+03 (9.44E-14)	1.27E+03 (5.97E-14)	1.27E+03 (5.97E-14)	**3.45E-02 (1.82E-02)**	1.27E+03 (1.69E-13)
C8	2.04E+01 (7.72E-02)	**2.03E+01 (8.51E-02)**	**2.03E+01 (8.87E-02)**	**2.03E+01 (1.19E-01)**	**2.03E+01 (9.46E-02)**
C9	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C10	6.17E+00 (2.86E+00)	9.75E+00 (1.94E+00)	5.85E+00 (8.82E-01)	6.14E+00 (2.79E+00)	**5.17E+00 (1.95E+00)**
C11	9.39E-01 (9.28E-01)	6.10E+00 (5.82E-01)	4.73E+00 (5.52E-01)	2.44E-01 (3.10E-01)	**1.61E-01 (3.37E-03)**
C12	4.62E+00 (6.07E+00)	4.54E+00 (8.85E+00)	5.89E+01 (2.44E+02)	1.85E+00 (6.84E+00)	**6.65E-06 (3.64E-05)**
C13	5.65E-01 (7.83E-02)	4.24E-01 (7.21E-02)	3.40E-01 (4.33E-02)	**5.58E-01 (1.30E-01)**	1.65E+00 (4.56E-01)
C14	2.96E+00 (2.96E-01)	3.32E+00 (1.80E-01)	**2.64E+00 (2.45E-01)**	2.78E+00 (5.22E-01)	2.98E+00 (2.10E-01)
C15	2.66E+01 (7.64E+01)	**2.18E+01 (4.27E+01)**	6.39E+01 (1.36E+02)	2.31E+01 (7.34E+01)	2.65E+02 (1.52E+02)

Table D.2: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 10D over 30 independent runs (C16-C25).

10D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C16	9.93E+01 (5.51E+00)	1.11E+02 (7.04E+00)	9.54E+01 (6.59E+00)	1.00E+02 (7.08E+00)	**9.41E+01 (1.88E+00)**
C17	**9.94E+01 (7.92E+00)**	1.30E+02 (9.22E+00)	1.29E+02 (3.32E+00)	1.18E+02 (1.46E+01)	1.17E+02 (3.36E+01)
C18	7.30E+02 (1.99E+02)	5.33E+02 (2.54E+02)	7.27E+02 (1.96E+02)	5.17E+02 (2.52E+02)	**3.00E+02 (0.00E+00)**
C19	7.46E+02 (1.82E+02)	5.33E+02 (2.54E+02)	7.82E+02 (1.35E+02)	5.17E+02 (2.52E+02)	**4.33E+02 (2.25E+01)**
C20	7.33E+02 (1.73E+02)	4.67E+02 (2.40E+02)	7.22E+02 (2.17E+02)	5.17E+02 (2.52E+02)	**4.18E+02 (2.14E+02)**
C21	5.67E+02 (2.23E+02)	5.20E+02 (7.61E+01)	4.67E+02 (1.62E+02)	4.90E+02 (1.06E+02)	**4.67E+02 (7.58E-01)**
C22	7.85E+02 (1.76E+02)	7.67E+02 (3.31E+00)	7.68E+02 (4.47E+00)	7.66E+02 (1.79E+01)	**7.65E+02 (4.08E+00)**
C23	7.75E+02 (1.80E+02)	6.20E+02 (1.47E+02)	7.34E+02 (1.61E+02)	5.89E+02 (8.49E+01)	**5.65E+02 (2.95E+01)**
C24	2.10E+02 (5.48E+01)	**2.00E+02 (0.00E+00)**	**2.00E+02 (0.00E+00)**	**2.00E+02 (0.00E+00)**	**2.00E+02 (0.00E+00)**
C25	1.75E+03 (4.44E+00)	1.74E+03 (4.87E+00)	1.73E+03 (5.56E+00)	**3.75E+02 (3.36E+00)**	1.73E+03 (4.23E+00)
No-Best	6	8	9	12	17

Table D.3: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 30D over 30 independent runs (C1-C15).

30D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	6.71E-06 (1.99E-05)	1.32E-06 (3.72E-06)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C3	4.16E+05 (1.60E+05)	1.41E+05 (8.29E+04)	**6.92E+03 (5.66E+03)**	7.21E+04 (4.75E+04)	3.56E+05 (1.67E+04)
C4	1.00E+02 (1.32E+02)	6.27E-02 (1.63E-01)	**0.00E+00 (0.00E+00)**	2.05E-03 (5.19E-03)	1.49E-03 (1.05E-03)
C5	3.30E+03 (6.95E+02)	5.12E+02 (3.97E+02)	**1.01E-07 (2.93E-07)**	6.76E+01 (9.32E+01)	1.83E+02 (1.55E+02)
C6	4.54E+01 (3.13E+01)	2.68E+01 (2.91E+01)	8.49E+00 (2.66E+01)	**2.40E-02 (2.52E-02)**	2.19E+02 (5.76E+02)
C7	4.70E+03 (3.71E-12)	4.70E+03 (2.17E-12)	4.70E+03 (1.86E-12)	**1.89E-03 (3.97E-03)**	4.70E+03 (2.82E-12)
C8	2.10E+01 (4.82E-02)	2.10E+01 (4.53E-02)	2.09E+01 (1.80E-01)	**2.09E+01 (2.31E-01)**	**2.09E+01 (4.74E-02)**
C9	1.99E-01 (4.05E-01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.21E+01 (3.28E+00)
C10	5.04E+01 (1.21E+01)	5.84E+01 (8.73E+00)	2.60E+01 (6.38E+00)	4.19E+01 (1.42E+01)	**2.44E+01 (1.03E+01)**
C11	1.64E+01 (3.25E+00)	2.82E+01 (1.81E+00)	2.52E+01 (1.76E+00)	8.10E+00 (2.65E+00)	**7.53E+00 (2.02E+00)**
C12	4.29E+03 (3.35E+03)	9.25E+03 (6.64E+03)	6.83E+03 (4.68E+03)	3.39E+03 (4.01E+03)	**3.11E+03 (3.32E+03)**
C13	3.84E+00 (4.48E-01)	4.70E+00 (1.57E-01)	3.47E+00 (9.83E-02)	2.88E+00 (1.06E+00)	**2.68E+00 (8.11E-01)**
C14	1.30E+01 (2.18E-01)	1.31E+01 (2.11E-01)	1.29E+01 (3.25E-01)	1.31E+01 (4.73E-01)	**1.29E+01 (2.08E-01)**
C15	3.90E+02 (6.11E+01)	3.94E+02 (5.62E+01)	3.57E+02 (1.21E+02)	4.13E+02 (5.71E+01)	**3.37E+02 (1.07E+01)**

Table D.4: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 30D over 30 independent runs (C16-C25).

30D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C16	9.61E+01 (8.60E+01)	7.61E+01 (1.73E+01)	9.74E+01 (1.29E+02)	7.21E+01 (2.49E+01)	**4.43E+01 (8.77E+00)**
C17	**6.50E+01 (1.27E+01)**	1.44E+02 (5.94E+01)	1.40E+02 (1.42E+02)	8.02E+01 (3.22E+01)	2.18E+02 (5.66E+01)
C18	**8.73E+02 (6.05E+01)**	9.05E+02 (1.08E+00)	9.05E+02 (7.52E-01)	9.05E+02 (8.60E-01)	9.05E+02 (2.07E+00)
C19	**8.76E+02 (6.33E+01)**	9.05E+02 (9.57E-01)	9.06E+02 (1.91E+01)	9.05E+02 (7.34E-01)	9.05E+02 (1.75E+00)
C20	**8.73E+02 (6.05E+01)**	9.05E+02 (8.38E-01)	9.05E+02 (8.83E-01)	9.05E+02 (5.91E-01)	9.04E+02 (1.41E+00)
C21	**5.00E+02 (3.06E-13)**	**5.00E+02 (1.95E-13)**	**5.00E+02 (1.80E-13)**	**5.00E+02 (1.99E-13)**	**5.00E+02 (1.95E-13)**
C22	9.30E+02 (1.42E+01)	8.84E+02 (2.03E+01)	8.88E+02 (2.23E+01)	8.85E+02 (3.42E+01)	**8.76E+02 (1.43E+01)**
C23	5.55E+02 (1.15E+02)	5.34E+02 (2.35E-04)	5.48E+02 (7.34E+01)	5.34E+02 (3.22E-07)	**5.34E+02 (3.12E-04)**
C24	**2.00E+02 (5.68E-13)**	**2.00E+02 (2.89E-14)**	**2.00E+02 (2.89E-14)**	**2.00E+02 (2.89E-14)**	**2.00E+02 (2.89E-14)**
C25	1.64E+03 (5.62E+00)	1.63E+03 (2.89E+00)	1.63E+03 (4.42E+00)	**2.10E+02 (6.31E-01)**	1.63E+03 (4.90E-01)
No-Best	7	4	8	9	14

Table D.5: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 50D over 30 independent runs (C1-C15).

50D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	7.86E-02 (1.25E-01)	1.05E-02 (7.41E-03)	**0.00E+00 (0.00E+00)**	1.22E-06 (1.01E-06)	1.05E-04 (1.00E-04)
C3	9.89E+05 (2.90E+05)	5.03E+05 (2.24E+05)	**1.62E+04 (8.11E+03)**	5.50E+05 (9.41E+04)	4.51E+05 (2.09E+04)
C4	6.43E+03 (3.18E+03)	3.48E+02 (2.27E+02)	**8.86E-01 (2.96E+00)**	1.56E+02 (1.17E+02)	1.27E+02 (1.50E+02)
C5	8.20E+03 (1.14E+03)	3.32E+03 (6.41E+02)	3.51E+03 (5.24E+02)	3.16E+03 (5.91E+02)	**2.98E+03 (4.87E+02)**
C6	9.47E+01 (3.63E+01)	4.60E+01 (2.98E+01)	**9.30E-01 (1.71E+00)**	2.82E+01 (2.29E+01)	3.05E+05 (9.48E+03)
C7	6.20E+03 (3.10E-12)	6.20E+03 (3.20E-12)	6.20E+03 (3.27E-12)	3.69E-03 (7.15E-03)	6.20E+03 (3.44E-12)
C8	**2.11E+01 (3.65E-02)**	**2.11E+01 (2.38E-02)**	**2.11E+01 (1.86E-01)**	2.12E+01 (5.00E-01)	**2.11E+01 (3.97E-02)**
C9	1.82E+00 (1.50E+00)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (1.88E-11)**	3.40E+01 (7.46E+00)
C10	1.27E+02 (1.78E+01)	1.01E+02 (1.48E+01)	9.40E+01 (6.67E+00)	9.97E+01 (1.77E+01)	**9.15E+01 (9.39E+00)**
C11	4.64E+01 (3.15E+00)	5.40E+01 (2.30E+00)	5.24E+01 (2.24E+00)	5.10E+01 (5.07E+00)	**4.59E+01 (1.99E+01)**
C12	4.66E+04 (1.19E+04)	2.61E+04 (1.45E+04)	5.55E+04 (1.81E+04)	2.40E+04 (1.31E+04)	**1.82E+04 (1.07E+04)**
C13	8.67E+00 (1.16E+00)	2.95E+00 (1.97E-01)	**2.79E+00 (1.66E-01)**	5.15E+00 (9.90E-01)	4.77E+00 (7.74E-01)
C14	2.29E+01 (2.75E-01)	2.30E+01 (2.93E-01)	2.29E+01 (4.71E-01)	2.29E+01 (4.99E-01)	**2.28E+01 (2.10E-01)**
C15	3.77E+02 (6.25E+01)	**3.03E+02 (9.99E+01)**	3.45E+02 (9.10E+01)	3.87E+02 (5.07E+01)	3.43E+02 (2.02E+01)

Table D.6: Mean and standard deviation (Mean and std) function error values among DE-HMM and DE variants on CEC2005 25-test functions with 50D over 30 independent runs (C16-C25).

50D					
	SaDE	jDE	JADE	CoBiDE	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C16	9.42E+01 (3.76E+01)	8.99E+01 (9.96E+00)	9.48E+01 (1.09E+02)	9.89E+01 (2.65E+01)	**8.97E+01 (3.34E+01)**
C17	9.85E+01 (6.47E+01)	1.74E+02 (1.91E+01)	1.12E+02 (6.42E+01)	**7.82E+01 (2.96E+01)**	2.83E+02 (1.17E+00)
C18	9.86E+02 (1.04E+01)	9.25E+02 (3.02E+00)	9.25E+07 (4.82E+00)	9.23E+02 (2.93E+00)	9.23E+02 (4.15E+00)
C19	9.87E+02 (1.72E+01)	9.19E+02 (2.93E+00)	9.21E+02 (4.88E+00)	9.07E+02 (3.62E+01)	9.20E+02 (1.79E-02)
C20	9.89E+02 (1.39E+01)	9.20E+02 (3.50E+00)	9.22E+02 (5.38E+00)	9.14E+02 (2.18E+01)	9.21E+02 (1.08E-01)
C21	7.54E+02 (3.40E+02)	7.85E+02 (2.54E+02)	8.17E+02 (9.29E+01)	6.84E+02 (1.29E+02)	**6.69E+02 (2.44E+02)**
C22	9.86E+02 (9.45E+00)	9.02E+02 (1.32E+01)	8.96E+02 (2.40E+01)	**8.82E+02 (2.01E+01)**	9.19E+02 (1.97E+01)
C23	7.83E+02 (1.66E+02)	8.24E+02 (2.34E+02)	6.90E+02 (5.10E+00)	7.12E+02 (2.31E+02)	**6.87E+02 (2.21E+02)**
C24	4.93E+02 (4.55E+02)	**2.00E+02 (1.65E-12)**	**2.00E+02 (1.62E-12)**	**2.00E+02 (1.33E-12)**	**2.00E+02 (1.66E-12)**
C25	1.69E+03 (6.32E+00)	1.66E+03 (3.60E+00)	1.66E+03 (5.82E+00)	**2.16E+02 (9.95E-01)**	1.66E+03 (5.69E+00)
No-Best	2	5	9	6	11

Table D.7: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 10D CEC2005 over 30 independent runs (C1-C15).

(a) 10D					
	BNGA	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	1.05E-08 (1.13E-08)	5.52E-03 (3.69E-03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C3	1.43E+05 (9.73E+04)	2.61E+05 (1.46E+05)	**0.00E+00 (0.00E+00)**	5.68E-07 (2.82E-08)	1.73E-01 (5.20E-02)
C4	**0.00E+00 (0.00E+00)**	2.12E-01 (1.48E-01)	6.07E+04 (1.14E+05)	6.13E-08 (1.92E-09)	**0.00E+00 (0.00E+00)**
C5	3.33E-07 (1.47E-06)	7.65E-01 (9.08E-01)	**0.00E+00 (0.00E+00)**	6.58E-08 (1.65E-08)	**0.00E+00 (0.00E+00)**
C6	2.22E+01 (7.40E+01)	6.67E+00 (2.71E+00)	9.30E-01 (1.71E+00)	**0.00E+00 (0.00E+00)**	4.99E+00 (1.61E+00)
C7	4.82E-01 (3.45E-01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	3.43E-06 (5.33E-07)	1.27E+03 (1.69E-13)
C8	2.04E+01 (7.39E-02)	**2.03E+01 (9.65E-02)**	**2.03E+01 (1.20E-01)**	**2.03E+01 (3.88E-03)**	**2.03E+01 (9.46E-02)**
C9	7.24E+00 (4.08E+00)	2.19E-04 (6.40E-04)	1.08E+02 (5.67E+01)	2.39E-01 (4.48E-01)	**0.00E+00 (0.00E+00)**
C10	1.93E+01 (7.38E+00)	1.14E+01 (2.92E+00)	5.70E+01 (9.11E+01)	**7.56E-02 (3.77E-01)**	5.17E+00 (1.95E+00)
C11	4.03E+00 (1.22E+00)	5.05E+00 (5.37E-01)	3.06E+00 (1.62E+00)	9.34E-01 (9.00E-01)	**1.61E-01 (3.37E-03)**
C12	9.21E+02 (2.31E+03)	1.98E+02 (1.01E+02)	6.88E+03 (8.06E+03)	2.93E+01 (1.42E+01)	**6.65E-06 (3.64E-05)**
C13	1.31E+00 (6.43E-01)	**9.97E-01 (1.97E-01)**	1.19E+00 (4.26E-01)	8.66E+00 (1.65E-01)	1.65E+00 (4.56E-01)
C14	3.03E+00 (3.50E-01)	3.26E+00 (1.82E-01)	4.85E+00 (1.47E-01)	4.21E+00 (2.54E-01)	**2.98E+00 (2.10E-01)**
C15	1.94E+02 (1.10E+02)	**5.78E+01 (3.33E+01)**	5.96E+02 (3.28E+02)	2.88E+02 (6.50E+01)	2.65E+02 (1.52E+02)

Table D.8: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 10D CEC2005 over 30 independent runs (C16-C25).

(a) 10D					
	BNGA	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
C16	1.52E+02 (3.47E+01)	1.22E+02 (8.97E+00)	3.01E+02 (3.11E+02)	9.53E+01 (2.21E+00)	**9.41E+01 (1.88E+00)**
C17	1.42E+02 (2.18E+01)	1.37E+02 (1.26E+01)	5.29E+02 (5.02E+02)	1.23E+02 (1.29E+02)	**1.17E+02 (3.36E+01)**
C18	9.20E+02 (1.23E+02)	6.16E+02 (1.65E+02)	7.05E+02 (2.56E+02)	3.32E+02 (1.54E+02)	**3.00E+02 (0.00E+00)**
C19	8.19E+02 (2.19E+02)	5.78E+02 (1.72E+02)	7.85E+02 (2.07E+02)	6.26E+02 (8.94E+02)	**4.33E+02 (2.25E+01)**
C20	7.92E+02 (2.17E+02)	5.85E+02 (1.54E+02)	8.34E+02 (3.33E+02)	**3.00E+02 (0.00+00)**	4.18E+02 (2.14E+02)
C21	7.49E+02 (3.13E+02)	6.35E+02 (6.94E+01)	8.45E+02 (2.89E+02)	5.00E+02 (7.33E-12)	**4.67E+02 (7.58E-01)**
C22	7.91E+02 (3.39E+01)	**7.10E+02 (1.52E+02)**	7.66E+02 (2.93E+01)	7.29E+02 (1.02E+01)	7.65E+02 (4.08E+00)
C23	8.55E+02 (2.83E+02)	**5.51E+02 (3.45E+01)**	1.12E+03 (1.36E+02)	5.59E+02 (6.21E-10)	5.65E+02 (2.95E+01)
C24	2.55E+02 (1.91E+02)	**2.00E+02 (2.86E-04)**	2.77E+02 (1.30E+02)	**2.00E+02 (2.14E-06)**	**2.00E+02 (0.00E+00)**
C25	**2.28E+02 (1.40E+02)**	1.75E+03 (5.82E+00)	1.77E+03 (1.08E+01)	3.74E+02 (3.43E+00)	1.73E+03 (4.23E+00)
No-Best	3	8	6	7	15

Table D.9: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 30D CEC2005 over 30 independent runs (C1-C15).

(b)30D				
	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	9.16E+02 (2.26E+02)	**0.00E+00 (0.00E+00)**	6.22E-08 (8.95E−10)	**0.00E+00 (0.00E+00)**
C3	1.62E+07 (3.74E+06)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	3.56E+05 (1.67E+04)
C4	7.08E+03 (1.65E+03)	4.21E+05 (1.07E+06)	1.11E+04 3.02E+04	**1.49E-03 (1.05E-03)**
C5	3.87E+03 (4.55E+02)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.83E+02 (1.55E+02)
C6	5.98E+00 (5.58E+00)	3.99E-01 (1.22E+00)	**0.00E+00 (0.00E+00)**	2.19E+02 (5.76E+02)
C7	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	4.70E+03 (2.82E-12)
C8	**2.09E+01 (4.94E-02)**	**2.09E+01 (4.72E-01)**	**2.09E+01 (2.36E-01)**	**2.09E+01 (4.74E-02)**
C9	**0.00E+00 (0.00E+00)**	3.96E+02 (1.63E+02)	9.38E-01 (1.18E+00)	1.21E+01 (3.28E+00)
C10	1.02E+02 (1.67E+01)	4.70E+01 (1.14E+01)	**1.65E+00 (1.27E+00)**	2.44E+01 (1.03E+01)
C11	2.56E+01 (2.13E+00)	7.87E+00 (2.38E+00)	8.34E+00 (3.11E+00)	**7.53E+00 (2.02E+00)**
C12	1.62E+04 (4.99E+03)	1.53E+04 (1.70E+04)	4.43E+04 (2.19E+05)	**3.11E+03 (3.32E+03)**
C13	2.80E+00 (2.67E-01)	3.62E+00 (1.05E+00)	3.49E+00 (5.13E-01)	**2.68E+00 (8.11E-01)**
C14	1.30E+01 (2.34E-01)	1.47E+01 (2.79E-01)	**1.29E+01 (4.53E-01)**	**1.29E+01 (2.08E-01)**
C15	**6.59E+01 (2.65E+01)**	3.82E+02 (2.43E+02)	4.08E+02 (2.22E+01)	3.37E+02 (1.07E+01)

Table D.10: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 30D CEC2005 over 30 independent runs (C16-C25).

(b)30D				
	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)
C16	1.73E+02 (3.29E+01)	3.52E+02 (3.11E+02)	5.50E+01 (1.26E+01)	**4.43E+01 (8.77E+00)**
C17	2.37E+02 (4.40E+01)	5.58E+02 (3.24E+02)	2.91E+02 (3.21E+02)	**2.18E+02 (5.66E+01)**
C18	9.07E+02 (2.39E+01)	9.06E+02 (2.59E-01)	**9.05E+02 (2.87E-01)**	**9.05E+02 (2.07E+00)**
C19	9.06E+02 (2.63E+01)	9.06E+02 (3.11E-01)	**9.05E+02 (2.71E-01)**	**9.05E+02 (1.75E+00)**
C20	9.11E+02 (1.89E+01)	9.33E+02 (1.19E+02)	**9.04E+02 (2.44E-01)**	**9.04E+02 (1.41E+00)**
C21	**5.00E+02 (7.11E-13)**	**5.00E+02 (2.31E-12)**	**5.00E+02 (1.31E-13)**	**5.00E+02 (1.95E-13)**
C22	9.67E+02 (1.17E+01)	8.27E+02 (1.76E+01)	**8.03E+02 (1.68E+01)**	8.76E+02 (1.43E+01)
C23	**5.34E+02 (1.15E-04)**	5.37E+02 (3.99E+00)	**5.34E+02 (2.22E-04)**	**5.34E+02 (3.12E-04)**
C24	**2.00E+02 (1.19E-12)**	**2.00E+02 (7.86E-14)**	9.10E+02 (1.46E+02)	**2.00E+02 (2.89E-14)**
C25	1.65E+03 (4.32E+00)	1.69E+03 (6.20E+01)	**2.11E+02 8.21E-01)**	1.63E+03 (4.90E-01)
No-Best	8	8	15	16

Table D.11: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 50D CEC2005 over 30 independent runs (C1-C15).

(b)50D				
	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)
C1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
C2	1.59E+04 (1.88E+03)	5.29E-03 (7.64E−05)	**0.00E+00 (0.00E+00)**	1.05E-04 (1.00E-04)
C3	6.53E+07 (1.30E+07)	**0.00E+00 (0.00E+00)**	2.63E-08 (3.55E−12)	4.51E+05 (2.09E+04)
C4	4.30E+04 (6.97E+03)	8.81E+05 (4.38E+06)	4.68E+05 (3.11E+05)	**1.27E+02 (1.50E+02)**
C5	1.10E+04 (8.50E+02)	**1.32E-02 (7.26E-02)**	2.85E+00 (4.32E+00)	2.98E+03 (4.87E+02)
C6	2.84E+01 (1.82E+01)	6.64E-01 (1.51E+00)	**0.00E+00 (0.00E+00)**	3.05E+05 (9.48E+03)
C7	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	6.20E+03 (3.44E-12)
C8	4.18E+01 (4.49E-02)	8.26E+01 (7.39E-01)	5.41E+02 (2.15E+00)	**2.11E+01 (3.97E-02)**
C9	1.48E+02 (1.69E+00)	7.06E+02 (2.10E+02)	5.79E+01 (1.64E+00)	**3.40E+01 (7.46E+00)**
C10	3.12E+02 (4.10E+01)	9.77E+01 (1.86E+01)	3.52E+02 (4.22E+00)	**9.15E+01 (9.39E+00)**
C11	5.29E+01 (2.42E+00)	1.21E+02 (3.91E+00)	6.36E+01 (1.21E+01)	**4.59E+01 (1.99E+01)**
C12	1.02E+05 (2.28E+04)	2.61E+04 (2.83E+04)	2.27E+05 (1.11E+06)	**1.82E+04 (1.07E+04)**
C13	5.07E+00 (4.92E-01)	6.57E+01 (1.04E+00)	4.59E+00 (5.15E-01)	**4.77E+00 (7.74E-01)**
C14	2.26E+01 (2.30E-01)	2.45E+01 (2.53E-01)	2.29E+01 (5.79E-01)	**2.28E+01 (2.10E-01)**
C15	6.02E+02 (5.06E+01)	4.49E+02 (2.04E+02)	4.74+02 (1.66E+01)	**3.43E+02 (2.02E+01)**

Table D.12: Mean and standard deviation (Mean and std) function error values among DE-HMM and non-DE variants at 50D CEC2005 over 30 independent runs (C16-C25).

(b)50D				
	CLPSO	CMA-ES	IPOP-CMA-ES	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)
C16	2.67E+02 (4.59E+01)	2.67E+02 (2.05E+02)	**3.09E+01 (2.53E+01)**	8.97E+01 (3.34E+01)
C17	3.39E+02 (3.04E+01)	3.74E+02 (3.05E+02)	2.91E+02 (1.44E+02)	**2.83E+02 (1.17E+00)**
C18	9.51E+02 (5.70E+00)	**9.12E+02 (4.63E-01)**	9.13E+02 (8.42E-01)	9.23E+02 (4.15E+00)
C19	9.55E+02 (6.38E+00)	9.22E+02 (4.34E-01)	9.32E+02 (1.23E-01)	**9.20E+02 (1.79E-02)**
C20	9.46E+02 (2.65E+01)	**9.12E+02 (4.79E-01)**	**9.12E+02 (5.05E-01)**	9.21E+02 (1.08E-01)
C21	5.00E+03 (6.65E-11)	6.98E+02 (2.56E+02)	1.00E+03 (8.19E-01)	**6.69E+02 (2.44E+02)**
C22	1.01E+03 (8.32E+00)	8.55E+02 (9.20E+00)	**8.05E+02 (8.86E+00)**	9.19E+02 (1.97E+01)
C23	8.39E+02 (1.63E-04)	7.78E+02 (1.37E+02)	1.01E+03 (1.86E+00)	**6.87E+02 (2.21E+02)**
C24	8.58E+02 (3.62E+02)	**2.00E+02 (1.62E-12)**	9.55E+02 (1.58E+02)	**2.00E+02 (1.66E-12)**
C25	1.71E+03 (7.62E+00)	1.84E+03 (3.03E+01)	**2.15E+02 (9.07E-01)**	1.66E+03 (5.69E+00)
No-Best	2	7	8	15

Table D.13: Comparison summary among DE-HMM and state-of-the-art on the 10D, 30D and 50D CEC2005 test problems.

DE-HMM		"Better"	"Worse"	"Equal"	"p-value"	"Significance"
C-DE	10D	**17**	4	4	**0.008**	+
	30D	**1817**	6	1	**0.0415**	+
	50D	**15**	8	2	0.173	=
SaDE	10D	**15**	4	6	**0.0055**	+
	30D	**15**	6	4	**0.0395**	+
	50D	**19**	3	3	**0.0035**	+
jDE	10D	**14**	3	8	**0.0085**	+
	30D	**14**	3	8	**0.0245**	+
	50D	**12**	8	5	0.185	=
JADE	10D	**11**	4	10	**0.0135**	+
	30D	**10**	6	9	0.398	=
	50D	**12**	8	5	0.3405	=
CoBiDE	10D	**12**	7	6	**0.0260**	+
	30D	**10**	7	8	0.246	=
	50D	**13**	9	3	0.4165	=
BNGA	10D	19	4	2	**0.0105**	+
	30D	-				
	50D	**-**				
CLPSO	10D	17	5	3	**0.031**	+
	30D	16	4	5	**0.020**	+
	50D	21	3	1	**0.001**	+
CMAES	10D	17	4	4	**0.001**	+
	30D	14	6	5	0.085	=
	50D	16	7	2	0.130	=
IPOP-CMA-ES	10D	13	8	4	**0.0472**	+
	30D	8	9	8	0.301	**-**
	50D	13	11	1	0.352	=

Appendix E

CEC2014 Comparison of DE-HMM and Up-to-date algorithms

Table E.1: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 10D over 30 independent runs(F1-F15)

10D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F1	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
F2	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
F3	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
F4	1.01E+01 (1.52E+01)	1.16E+01 (1.55E+01)	3.13E+01 (1.06E+01)	2.13E+01 (1.68E+01)	**7.25E+00 (1.27E-01)**
F5	2.01E+01 (5.53E-01)	1.84E+01 (6.10E+00)	1.82E+01 (8.09E+00)	1.91E+01 (3.72E+00)	**1.63E+01 (8.28E+00)**
F6	1.90E-08 (9.32E-08)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	9.48E-02 (2.90E-01)	**0.00E+00 (0.00E+00)**
F7	3.65E-01 (7.41E-02)	4.06E-02 (1.82E-02)	6.57E-03 (2.50E-03)	**1.45E-04 (0.00E+00)**	6.24E-03 (1.28E-03)
F8	3.32E-02 (1.82E-01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	3.32E-02 (1.82E-01)	**0.00E+00 (0.00E+00)**
F9	1.96E+01 (3.38E+00)	4.41E+00 (2.00E+00)	2.86E+00 (8.86E-01)	3.18E+00 (1.39E+00)	**2.71E+00 (1.19E+00)**
F10	2.01E+01 (1.46E+01)	5.20E-02 (6.37E-02)	**8.33E-03 (2.16E-02)**	8.45E-01 (1.64E+00)	8.92E+01 (2.07E+00)
F11	9.30E+02 (1.39E+02)	9.58E+01 (9.46E+01)	**3.39E+01 (4.62E+01)**	1.35E+02 (1.46E+02)	3.27E+02 (3.50E-02)
F12	7.24E-01 (1.27E-01)	2.95E-02 (4.44E-02)	6.32E-02 (1.91E-02)	**8.24E-03 (2.07E-02)**	1.05E+00 (2.65E-01)
F13	2.21E-01 (3.26E-02)	8.98E-02 (2.71E-02)	8.51E-02 (1.75E-02)	**1.75E-02 (8.98E-03)**	8.49E-02 (1.27E-02)
F14	1.45E-01 (3.13E-02)	1.11E-01 (4.15E-02)	**8.27E-02 (3.20E-02)**	1.14E-01 (4.37E-02)	1.47E-01 (3.94E-02)
F15	2.12E+00 (3.76E-01)	5.76E-01 (1.37E-01)	**3.85E-01 (6.11E-02)**	7.26E-01 (2.14E-01)	1.57E+00 (4.42E-01)

Table E.2: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 10D over 30 independent runs (F16-F30)

10D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F16	2.81E+00 (2.86E-01)	2.11E+00 (4.62E-01)	2.23E+00 (2.79E-01)	2.05E+00 (4.83E-01)	**1.97E+00 (3.17E-01)**
F17	1.74E+01 (8.79E+00)	1.76E+00 (3.40E+00)	7.99E-01 (7.17E-01)	1.08E+01 (1.27E+01)	**4.56E-01 (4.64E-01)**
F18	1.36E+00 (5.83E-01)	3.08E-01 (5.22E-01)	1.98E-01 (1.72E-01)	1.09E+00 (6.05E-01)	**1.81E-01 (2.24E-01)**
F19	6.05E-01 (2.17E-01)	1.35E-01 (9.34E-02)	2.11E-01 (3.30E-02)	2.05E-01 (3.19E-01)	**1.30E-01 (1.07E-01)**
F20	3.11E-01 (1.56E-01)	5.85E-02 (8.29E-02)	1.30E-01 (1.25E-01)	3.88E-01 (3.97E-01)	**5.67E-02 (9.39E-02)**
F21	8.01E-01 (5.39E-01)	5.93E-01 (2.03E-01)	3.92E-01 (2.78E-01)	1.11E+00 (2.97E+00)	**3.10E-01 (2.82E-01)**
F22	8.34E-01 (1.20E+00)	**6.82E-02 (9.69E-02)**	1.16E-01 (9.31E-02)	2.46E-01 (1.78E-01)	7.88E-02 (1.62E-01)
F23	**3.29E+02 (2.89E-13)**	**3.29E+02 (2.89E-13)**	**3.29E+02 (2.89E-13)**	**3.29E+02 (2.89E-13)**	**3.29E+02 (2.89E-13)**
F24	1.29E+02 (3.79E+00)	1.12E+02 (3.04E+00)	1.09E+02 (1.93E+00)	**1.09E+02 (2.00E+00)**	**1.09E+02 (1.95E+00)**
F25	1.64E+02 (3.51E+01)	**1.26E+02 (2.57E+01)**	1.32E+02 (3.93E+01)	1.41E+02 (3.14E+01)	1.59E+02 (4.44E+01)
F26	**1.00E+02 (3.43E-02)**	**1.00E+02 (5.64E-02)**	**1.00E+02 (1.40E-02)**	**1.00E+02 (1.82E-02)**	**1.00E+02 (2.45E-02)**
F27	6.55E+01 (1.29E+02)	7.52E+01 (1.50E+02)	3.79E+01 (1.13E+02)	**3.52E+01 (1.02E+02)**	1.41E+02 (1.64E+02)
F28	3.72E+02 (3.40E+01)	**3.64E+02 (3.76E+00)**	3.83E+02 (3.45E+01)	3.66E+02 (8.40E+01)	**3.64E+02 (5.59E+00)**
F29	2.22E+02 (6.39E-01)	2.19E+02 (1.64E+01)	2.22E+02 (5.91E-01)	2.19E+02 (1.41E+01)	**2.17E+02 (1.73E+01)**
F30	**4.64E+02 (4.80E+00)**	4.66E+02 (1.04E+01)	4.64E+02 (7.29E+00)	4.87E+02 (3.04E+01)	4.70E+02 (1.81E+01)
No-Best	6	10	11	10	19

Table E.3: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 30D over 30 independent runs(F1-F15)

30D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F1	2.50E-05 (2.35E-05)	4.34E+04 (3.26E+04)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	9.58E+04 (8.97E+03)
F2	5.58E-06 (2.95E-06)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.30E+03 (2.35E-03)
F3	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	2.35E-01 (2.25E-01)
F4	2.11E+00 (1.16E+01)	6.25E+01 (7.26E-01)	**0.00E+00 (0.00E+00)**	3.51E+02 (1.16E+01)	3.25E+01 (3.64E+00)
F5	2.09E+01 (5.08E-02)	**2.01E+01 (7.65E-02)**	2.10E+01 (2.17E-02)	2.02E+01 (1.57E-01)	2.09E+01 (3.36E-02)
F6	2.32E+01 (8.96E+00)	2.01E+00 (1.99E+00)	**8.86E-06 (5.36E-04)**	1.80E+00 (1.45E+00)	1.46E+00 (1.25E+00)
F7	2.87E-08 (6.98E-09)	2.47E-04 (1.35E-03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
F8	9.94E+01 (8.26E+00)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.96E+00 (1.70E+00)	1.10E+01 (2.75E+00)
F9	1.94E+02 (8.66E+00)	3.59E+01 (1.13E+01)	1.89E+01 (1.49E+00)	1.98E+01 (3.22E+00)	**1.77E+01 (4.03E+00)**
F10	3.77E+03 (3.27E+02)	1.43E+02 (1.34E+00)	**4.86E-03 (1.05E-02)**	3.53E+02 (3.04E+01)	2.13E+01 (1.19E-01)
F11	6.76E+03 (2.04E+02)	8.92E+03 (6.20E+02)	6.24E+03 (1.69E+02)	7.94E+03 (7.97E+02)	**6.19E+03 (1.21E-07)**
F12	2.57E+00 (3.26E-01)	8.77E-02 (4.62E-02)	1.60E-01 (3.22E-02)	**1.95E-02 (9.35E-02)**	2.43E+00 (4.59E-01)
F13	4.44E-01 (3.93E-02)	3.49E-01 (5.07E-02)	3.15E-01 (1.56E-02)	**6.36E-02 (2.18E-02)**	2.59E-01 (3.44E-02)
F14	3.11E-01 (2.44E-02)	3.04E-01 (3.90E-02)	2.92E-01 (2.13E-02)	2.96E-01 (4.22E-02)	**2.82E-01 (2.81E-02)**
F15	1.64E+01 (1.25E+00)	3.18E+00 (1.04E+00)	2.14E+01 (1.90E-01)	**3.16E+00 (7.35E-01)**	1.31E+01 (2.71E+00)

Table E.4: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 30D over 30 independent runs (C16-C30)

30D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F16	1.25E+01 (2.89E-01)	9.38E+00 (8.16E-01)	**8.56E+00 (3.06E-01)**	1.08E+01 (4.63E-01)	1.20E+01 (3.33E-01)
F17	1.34E+03 (1.65E+02)	2.32E+03 (2.23E+03)	1.30E+03 (1.13E+02)	1.19E+03 (4.01E+02)	**1.13E+03 (8.16E+02)**
F18	4.65E+01 (7.11E+00)	1.54E+01 (5.12E+00)	6.16E+01 (3.32E+00)	2.88E+01 (1.85E+01)	**1.33E+01 (9.31E+00)**
F19	5.76E+00 (1.44E+00)	3.88E+00 (6.09E-01)	4.46E+00 (5.79E-01)	4.50E+00 (9.14E-01)	**3.50E+00 (8.90E-01)**
F20	3.51E+01 (3.37E+00)	1.41E+01 (8.38E+00)	1.43E+01 (1.04E+00)	1.22E+01 (7.11E+00)	**9.07E+00 (3.91E+00)**
F21	7.82E+02 (1.38E+02)	2.90E+02 (1.67E+02)	2.92E+02 (7.29E+01)	3.39E+02 (2.51E+02)	**2.71E+02 (2.02E+02)**
F22	1.82E+02 (7.89E+01)	1.54E+02 (8.02E+01)	7.51E+01 (3.23E+00)	9.59E+01 (7.06E+01)	**6.87E+01 (7.15E+01)**
F23	**3.15E+02 (1.93E-13)**	**3.15E+02 (1.45E-13)**	**3.15E+02 (5.78E-14)**	**3.15E+02 (2.31E-13)**	**3.15E+02 (3.72E-02)**
F24	**2.00E+02 (2.84E-02)**	2.26E+02 (4.01E+00)	2.25E+02 (2.38E+00)	2.25E+02 (1.39E+00)	2.25E+02 (1.94E+00)
F25	**2.03E+02 (2.83E-02)**	2.04E+02 (1.70E+00)	**2.03E+02 (4.99E-02)**	**2.03E+02 (1.12E+00)**	**2.03E+02 (4.03E-01)**
F26	**1.00E+02 (4.72E-02)**	**1.00E+02 (5.04E-02)**	**1.00E+02 (1.44E-02)**	**1.00E+02 (5.88E-02)**	**1.00E+02 (3.25E-12)**
F27	3.32E+02 (4.53E+01)	3.71E+02 (4.48E+01)	**3.00E+02 (0.00E+00)**	3.60E+02 (3.49E+01)	3.59E+02 **(4.70E+01)**
F28	9.98E+02 (4.80E+01)	8.39E+02 (3.19E+01)	8.33E+02 (2.06E+01)	8.63E+02 (4.65E+01)	**8.27E+02 (4.82E+01)**
F29	7.56E+02 (1.01E+02)	8.30E+02 (1.17E+02)	7.16E+02 (2.27E+00)	**6.94E+02 (1.56E+02)**	9.75E+02 (2.20E+02)
F30	9.78E+02 (1.91E+02)	**9.71E+02 (4.32E+02)**	1.89E+03 (4.89E+02)	1.53E+03 (4.02E+02)	1.33E+03 (5.87E+01)
No-Best	7	6	13	11	14

Table E.5: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 50D over 30 independent runs(F1-F15)

(c) 50D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F1	5.77E+05 (6.44E+05)	6.47E+05 (3.25E+05)	8.91E+03 (1.67E+03)	**0.00E+00 (0.00E+00)**	5.58E+05 (1.67E+04)
F2	7.69E+04 (2.70E+04)	2.00E+03 (1.32E+03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.81E+03 (2.95E+01)
F3	3.60E-03 (1.09E-02)	5.42E+02 (6.01E+02)	**0.00E+00 (0.00E+00)**	5.79E-02 (2.74E-01)	3.80E+03 (2.93E-02)
F4	9.21E+01 (3.44E+00)	7.08E+01 (2.74E+01)	9.82E+01 (4.65E+01)	8.25E+02 (3.57E+02)	**6.28E+01 (3.21E+01)**
F5	**2.11E+01 (3.83E-02)**	2.17E+01 (2.25E-01)	2.13E+01 (2.94E-02)	2.14E+01 (1.83E-01)	**2.11E+01 (3.76E-02)**
F6	3.62E+01 (1.55E+01)	8.17E+02 (4.10E+00)	4.74E+01 (5.34E-01)	6.00E+00 (2.81E+00)	**4.46E+00 (1.74E+00)**
F7	6.64E-02 (1.93E-04)	1.89E-03 (4.37E-03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
F8	2.88E+02 (1.18E+01)	9.12E+00 (5.45E+00)	**0.00E+00 (0.00E+00)**	5.84E+00 (4.04E+00)	3.39E+01 (8.89E+00)
F9	3.77E+02 (1.33E+01)	7.77E+01 (1.64E+01)	6.12E+01 (2.29E+00)	5.24E+01 (6.89E+00)	**4.85E+01 (1.19E+01)**
F10	9.36E+03 (3.50E+02)	4.35E+03 (1.12E+02)	**3.80E-02 (2.23E-02)**	8.52E+02 (1.79E+02)	7.49E+02 (3.66E+02)
F11	1.31E+04 (3.01E+02)	4.49E+03 (7.98E+02)	**3.31E+03 (2.29E+02)**	4.00E+03 (9.86E+02)	1.20E+04 (2.30E-13)
F12	3.52E+00 (3.30E-01)	7.27E-01 (4.55E-01)	2.16E-01 (3.34E-02)	**3.57E-03 (1.09E-02)**	3.39E+00 (2.67E-01)
F13	5.99E-01 (5.18E-02)	3.78E-01 (6.05E-02)	3.73E-01 (2.00E-02)	5.11E-01 (2.87E-02)	**3.68E-01 (5.26E-02)**
F14	3.31E-01 (3.47E-02)	3.43E-01 (4.06E-02)	3.31E-01 (2.14E-02)	3.88E-01 (4.95E-02)	**3.17E-01 (2.89E-02)**
F15	3.24E+01 (1.53E+00)	6.90E+00 (1.35E+00)	5.98E+00 (3.94E-01)	5.89E+00 (7.08E-01)	**5.85E+00 (1.05E+00)**

Table E.6: Mean and standard deviation function error values among DE-HMM and up-to-date DE variants on CEC2014 with 50D over 30 independent runs (F16-F30)

(c) 50D					
	CPI-DE	TSDE	LSHADE	UMOEA	DE-HMM
	mean (std)	mean (std)	mean (std)	mean (std)	mean (std)
F16	2.24E+01 (1.69E-01)	3.25E+01 (7.43E-01)	2.70E+01 (3.97E-01)	2.63E+01 (7.36E-01)	**2.15E+01 (4.36E-01)**
F17	3.21E+03 (3.60E+02)	4.38E+04 (2.90E+04)	3.51E+04 (3.68E+02)	2.52E+03 (6.68E+02)	3.33E+04 (2.17E+04)
F18	1.66E+02 (1.16E+01)	5.95E+02 (6.45E+02)	1.81E+02 (1.84E+01)	1.66E+02 (6.01E+01)	**1.55E+02 (8.05E+01)**
F19	1.29E+01 (2.27E+00)	1.29E+01 (1.77E+00)	1.29E+01 (1.92E+00)	1.36E+01 (2.09E+00)	**1.28E+01 (4.13E+00)**
F20	1.01E+02 (9.68E+00)	5.36E+02 (3.68E+02)	8.95E+02 (5.28E+00)	**8.50E+01 (5.32E+01)**	8.04E+02 (1.81E+01)
F21	2.07E+03 (2.10E+02)	4.29E+04 (4.24E+04)	**4.98E+02 (1.20E+02)**	1.63E+03 (4.71E+02)	4.00E+04 (3.87E+04)
F22	4.21E+03 (1.44E+02)	6.11E+02 (1.94E+02)	3.07E+02 (6.47E+01)	2.90E+02 (1.52E+02)	**2.35E+02 (1.47E+00)**
F23	**3.44E+02 (3.29E-07)**	**3.44E+02 (2.89E-13)**	**3.44E+02 (2.59E-13)**	**3.44E+02 (2.60E-13)**	**3.44E+02 (2.66E-03)**
F24	**2.60E+02 (3.91E+00)**	2.72E+02 (2.37E+00)	2.75E+02 (4.37E-01)	2.73E+02 (2.16E+00)	2.76E+02 (2.53E+00)
F25	**2.08E+02 (1.55E-01)**	2.09E+02 (4.26E+00)	**2.08E+02 (3.76E-01)**	**2.08E+02 (3.40E+00)**	**2.08E+02 (1.53E+00)**
F26	1.01E+02 (7.18E-02)	1.07E+02 (2.53E+01)	**1.00E+02 (1.69E-02)**	**1.00E+02 (8.96E-02)**	**1.00E+02 (4.62E-02)**
F27	5.49E+03 (4.02E+02)	5.34E+02 (7.38E+01)	5.30E+02 (3.30E+01)	5.25E+02 (3.55E+01)	**5.00E+02 (5.15E+00)**
F28	1.45E+03 (1.47E+01)	1.19E+03 (4.43E+01)	1.18E+03 (2.66E+01)	1.23E+03 (8.34E+01)	**1.17E+03 (6.91E+01)**
F29	8.79E+03 (6.60E+01)	1.23E+03 (2.11E+02)	**8.20E+02 (4.78E+01)**	1.11E+03 (2.77E+02)	1.20E+06 (6.58E+01)
F30	**8.14E+03 (2.70E+02)**	9.12E+03 (5.13E+02)	8.75E+03 (5.69E+02)	8.76E+03 (5.28E+02)	9.57E+03 (6.52E+02)
No-Best	5	1	11	8	17

Table E.7: Comparison summary among DE-HMM and up-to-date variants on the 10D, 30D and 50D CEC2014 test problems.

DE-HMM		"Better"	"Worse"	"Equal"	"p-value"	"Significance"
C-DE	10D	**22**	3	5	**0.001**	+
	30D	**23**	4	3	**0.008**	+
	50D	**22**	5	3	**0.0155**	+
CPI-DE	10D	**20**	5	5	**0.008**	+
	30D	**18**	8	4	0.129	=
	50D	**20**	7	3	**0.012**	+
TSDE	10D	**13**	9	8	0.455	=
	30D	**17**	11	2	**0.0325**	+
	50D	**21**	8	1	0.0815	=
LSHADE	10D	**14**	8	8	0.480	=
	30D	**14**	11	5	0.4625	=
	50D	**16**	11	3	**0.0355**	+
UMOEA	10D	**15**	9	6	**0.022**	+
	30D	**15**	10	5	0.0565	=
	50D	**14**	12	4	0.0955	=

Appendix F

CEC2005 and 2014 Time Complexity

Table F.1: Time complexity computations for CEC2005 and CEC2014 datasets.

	CEC2005 dataset	CEC2014 dataset

$T0$	$T0$ is the time computed by running the following test program:
	for i=1:1000000
	x= (double) 5.55;
	x=x + x; x=x./2; x=x*x; x=sqrt(x); x=ln(x); x=exp(x); y=x/x;
	end
	Same for 10, 30 and 50D
		$T0$ is calculated by running the following test program:
		for i=1:1000000
		x=0.55+(double) i; x=x+x; x=x/2; x=x * x;
		x=sqrt(x); x=log(x); x=exp(x); x=x/(x+2);
		end
		Same for 10, 30 and 50D

$T1$	$T1$is the time to execute 200,000 evaluations of benchmark function C3 by itself with D dimensions,	$T1$is the time to execute 200,000 evaluations of benchmark function F18 by itself with D dimensions,
$T2$	$T2$ is the time to execute an algorithm with 200,000 evaluations of C3 in D dimensions. $\hat{T}2$ is the mean T2 values of 5 runs.	$T2$ is the time to execute an algorithm with 200,000 evaluations of F18 in D dimensions. $\hat{T}2$ is the mean T2 values of 5 runs.

Table F.2: Time complexity for DE variants on problem C3 of CEC2005 benchmark.

2005										
Algorithm	10D				30D			50D		
	T0	T1	avg-T2	comp-T	T1	avg-T2	comp-T	T1	avg-T2	comp-T
jDE	0.16	0.172	0.8043	3.94E+00	0.328	1.1345	5.02E+00	0.3779	1.3504	6.06E+00
JADE	0.16	0.1446	0.9187	4.82E+00	0.3259	1.3384	6.31E+00	0.4009	1.5524	7.17E+00
C-DE	0.16	0.2098	1.2843	6.69E+00	0.3341	1.7879	9.05E+00	0.4632	2.0959	1.02E+01
DE-HMM	0.16	0.1685	1.8554	1.05E+01	0.2770	2.6917	1.50E+01	0.4356	3.1988	1.72E+01
CoBiDE	0.16	0.2193	2.2948	1.29E+01	0.3211	3.2968	1.85E+01	0.4735	4.0006	2.20E+01
SaDE	0.16	0.2027	12.1465	7.44E+01	0.3293	14.7644	8.99E+01	0.4430	14.8942	9.00E+01

Table F.3: Time complexity for up-to-date DE variants on problem F18 of CEC2014 benchmark.

2014										
Algorithm	10D				30D			50D		
	T0	T1	avg-T2	comp-T	T1	avg-T2	comp-T	T1	avg-T2	comp-T
C-DE	0.10	0.1458	0.7704	5.73E+00	0.3815	1.0978	6.57E+00	0.7140	1.7073	9.11E+00
LSHADE	0.10	0.1211	1.8671	1.60E+01	0.2897	2.1664	1.72E+01	0.5768	2.6573	1.91E+01
DE-HMM	0.10	0.1061	1.8579	1.61E+01	0.2730	2.1754	1.75E+01	0.5499	2.6928	1.97E+01
CPIDE	0.10	0.2069	2.1326	1.77E+01	0.3988	3.2362	2.60E+01	0.6679	4.2898	3.32E+01
UMOEA	0.10	0.9357	4.4822	3.25E+01	1.2194	5.8698	4.27E+01	1.4831	6.9143	4.98E+01
TSDE	0.10	0.2094	4.5450	3.98E+01	0.4278	4.9192	4.12E+01	0.7177	5.4962	4.38E+01

Appendix G

Example of DE-HMM procedure

I have started the execution using the steps of Algorithm 3, and in Step 11, Algorithm 2 will be implemented with the given population pop-m= [3 4 ; 2 1; 4 5]. While Algorithm 2 is running, the first step will be the start of Algorithm 1.

Algorithm3: DE-HMM

1. Initialise pop G=0 uniformly
2. Evaluate pop with fitness function
3. While (Feval < Max-Eval)
4. For each target-vector in pop
5. Mutant-vectors← Mutation
6. Trial-vectors←Crossover
7. Evaluate Trial-vectors , EndFor
8. Pop G=1←Selection between target-vectors and trial-vectors
9. Sort Pop G=1 according to fitness value ascendingly
10. Input Pop G=1 to **Algorithm2** to generate newF and newCR
11. EndWhile

Agorithm2: Apply hmm on the population of DE

Inputs: transition matrix A=[0.5 0.5;0.5 0.5]; Population matrix B as emission matrix // Pop G=1 from Algorithm3 —- Steps:

1. Apply **Algorithm1** to transform population sample into probability as following : // pop-m ←Pop G=1 , Assume that the population input at G=1 is the given matrix pop-m

   (a) Pop-m= $\begin{bmatrix} 3 & 4 \\ 2 & 1 \\ 4 & 5 \end{bmatrix}$

   (b) create matrix pop-r of the size of pop-m (3,2).

   (c) calculate for each variable the mean $\mu(j) \leftarrow mean(pop_m)$of pop-m and standard deviation $\sigma(j) \leftarrow stdev(pop_m)$

   $\mu(j) = \begin{bmatrix} 3 & 3.33333333 \end{bmatrix}$

   $\sigma(j) = \begin{bmatrix} 1 & 2.08166599 \end{bmatrix}$

   (d) For each row in pop-m, calculate the normal distribution under$\mu$and σ

   (e) return pop-r= $\begin{bmatrix} 0.5 & 0.6256 \\ 0.158 & 0.1312 \\ 0.8413 & 0.7883 \end{bmatrix}$

2. Sample a random sequence of states ST_{seq} and emission symbols EM_{seq} (Note 1 refers to state1 and 2 refers to state2), for example 80 sequence.

   ST_{seq} =2 1 2 1 2 1 1 2 2 2 2 2 1 1 2 1 2 1 1 1 1 1 2 2 2 1 2 2 2 1 2 2 2 2 1 1 2 1 2 2 2 2 2 1 2 2 2 1 1 2 1 2 2 1 1 2 2 2 1 1 1 2 2 2 2 2 1 2 2 1 1 1 1 1 2 2 1 2 1 1

   EM_{seq}=1 2 2 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 2 2 2 1 2 1 1 1 2 1 1 2 1 1 2 2 1 2 1 2 1 1 1 1 2 1 1 1 1 2 2 2 2 2 2 1 1 2 2 2 2 1 2 2 2 1 2 2 1 1 2 2 1 1 1 2 2 2 1 2

3. Apply Viterbi algorithm to find out the best state sequence that can maximise the likelihood of the estimated state sequence for given EM_{seq}

   $best_{seq}$=2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

4. Calculate CR by the probability ratio of the actual sequence state that overhead the best state sequence found

   $CR = ratio(ST_{seq} >= best_{seq})$

   $CR = 0.9625$

5. Calculate minimum posterior given EM_{seq}

   $min_{posterior}$=

   0.409877979685073 0.492632938889084 0.493981187346759 0.380892740874182
   0.375386516494484 0.380903354898191 0.493750623681496 0.487920153121753
   0.493750622570172 0.380903377678586 0.375387007339577 0.380903377608924
   0.493750624083418 0.487920184052916 0.493751256190501 0.380890420399688
   0.375107820672939 0.374853333463722 0.375107822111223 0.380890451642884
   0.493750578946908 0.487906341071534 0.493467550013981 0.386692103292816
   0.499885028923972 0.380619742189386 0.375360281351111 0.380606610183262
   0.499829710418849 0.380861295721322 0.380861873528882 0.499842839679907
   0.380875002857084 0.381144909034776 0.494024203752374 0.493766210692649
   0.386691357395305 0.499602558354115 0.386407427225138 0.499899531081088
   0.380607322195910 0.375106415299573 0.375105837944435 0.380594203215016
   0.499816068558082 0.380594231238201 0.375106443287529 0.375119533479994
   0.380891673788547 0.493735811017673 0.487605022467576 0.487305935330151
   0.487305904086064 0.487604350768443 0.493722073477269 0.381172600239570
   0.381172568668759 0.493722790846604 0.487619015468394 0.487605781509730
   0.493438281377968 0.386991335032673 0.493452206871298 0.487891070978735
   0.493452917665603 0.386976764858137 0.493752222319010 0.494022880001193
   0.381158046869324 0.381157406015310 0.494037441838434 0.494049900186457
   0.380889570965639 0.375386371691094 0.380903380161023 0.493749929251851
   0.487905952084370 0.493460226308872 0.386841838606740 0.496658727818858

6. $F = average(min_{posterior}, CR)$=0.668676666731861