

**ADAPTIVE DIFFERENTIAL EVOLUTION ALGORITHM WITH
FITNESS BASED SELECTION OF PARAMETERS AND
MUTATION STRATEGIES**

RAWAA DAWOUD HASSAN AL-DABBAGH

**THESIS SUBMITTED IN FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF DOCTOR
OF PHILOSOPHY**

**FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2015

ABSTRACT

Differential evolution (DE) is a simple yet powerful evolutionary algorithm (EA). It has demonstrated good convergence, and its principles are easy to understand. DE has effectively solved various global optimization problems, including benchmark functions. These problems have shown different challenging characteristics such as non-convexity, non-linearity, and/or multi-modality which became difficult for traditional non-linear programming to deal with.

The performance of DE algorithm depends heavily on the selected mutation strategy and its associated control parameters. The sensitivity of the DE algorithm to its mutation strategy and to the corresponding control parameters can significantly deteriorate its performance if the strategy is improperly selected. Hence, the process of choosing a suitable DE strategy and setting its control parameters is difficult and requires much user experience. In this thesis, the fundamental contributions include the analysis, design, and evaluation of the adaptive DE algorithms.

Firstly, a comprehensive procedural analysis is conducted to investigate the various adaptive schemes that have been utilized to automatically control the DE parameters and/or its mutation strategies. In the same analysis, two taxonomies are proposed for the purpose. The first one is proposed to eliminate any ambiguity related to classify any adaptive EA. The new classification comprises three levels of categories instead of two regarding the parameter control type (deterministic, adaptive, self-adaptive) and the evidence (absolute, relative) used for determining the change of the parameter. The second taxonomy is a new taxonomy proposed to classify the adaptive DE algorithms in particular into two categories (DE with adaptive parameters and DE with adaptive parameters and strategies) considering the adaptive components used in this algorithm.

Secondly, a new DE algorithm (ARDE-SPX) is introduced that automatically adapts a repository of DE strategies and parameters control schemes to avoid the problems of stagnation and make DE respond to a wide range of function characteristics at different stages of evolutionary search. ARDE algorithm makes use of JADE strategy and the MDE_pBX parameters adaptive schemes as frameworks. Then a new adaptive procedure called adaptive repository (*AR*) is developed to select the appropriate combinations of the JADE strategies and the parameter control schemes of the MDE_pBX to generate the next population based on their fitness values. The adaptive repository mechanism is a general scheme and can be embedded with high flexibility into any population-based evolutionary algorithm. Moreover, this work is extended to integrate the SPX crossover operator with the adaptive ARDE algorithm in a new way of implementation in order to make the adaptive ARDE algorithm satisfy both the global and local search requirements.

Thirdly, experimental results are presented to confirm the reliability of the proposed ARDE-SPX over several existing adaptive DE variants. These comparisons are conducted in terms of the solution precision, successful rate and robustness over thirty-three standard and transformed benchmark functions. ARDE is also used to develop a new dynamic parameter identification framework to estimate the barycentric parameters of the CRS A456 robot manipulator. The simulation results show the effectiveness of the ARDE method over other conventional techniques, transcending the limits of the existing state-of-the-art algorithms in estimating the parameters of robot.

ABSTRAK

Evolusi Pengkamiran (DE) adalah satu algoritma evolusi (EA) mudah lagi berkuasa. Ia berkesan menghasilkan keputusan yang hebat dalam menyelesaikan pelbagai masalah pengoptimuman global dari pelbagai disiplin seperti kejuruteraan dan sains. Masalah-masalah ini telah menunjukkan ciri-ciri cabaran yang berbeza seperti bukan kecembungan, bukan-kelinearan, dan / atau multi-modaliti yang menjadi sukar bagi pengaturcaraan tidak linear tradisional untuk ditangani. DE telah menunjukkan penumpuan yang baik, dan prinsipnya mudah difahami. Oleh itu, popularitinya telah beransur-ansur meningkat dan ia telah digunakan dalam banyak aplikasi dunia sebenar.

Walaubagaimanapun, prestasi algoritma DE adalah peka kepada jenis strategi yang dipilih dan kawalan parameter yang berkaitan kerana kelakuan yang berbeza bagi pelbagai masalah di pelbagai peringkat proses evolusi. Kepekaan algoritma DE terhadap strateginya dan kawalan parameter boleh membawa kepada kemerosotan prestasi yang ketara jika strategi tidak dipilih secara wajar. Oleh itu, proses pemilihan strategi DE yang sesuai dan menetapkan kawalan parameter adalah sukar dan memerlukan pengalaman pengguna yang lebih. Dalam tesis ini, sumbangan asas termasuk analisis, reka bentuk, dan penilaian penyesuaian DE algoritma seperti berikut,

Pertama, analisis prosedur yang menyeluruh telah dijalankan untuk menyiasat pelbagai skim yang sesuai yang telah digunakan untuk mengawal nilai parameter DE dan / atau strategi mutasinya secara automatik. Dalam analisis yang sama, dua taksonomi telah dicadangkan untuk tujuan itu. Yang pertama adalah taksonomi lanjutan kepada klasifikasi penetapan parameter EA yang umum. Adalah dicadangkan untuk menghapuskan apa-apa kekaburan berkaitan dengan mengelaskan mana-mana EA penyesuaian. Dengan itu, klasifikasi baru adalah tiga tahap kategori dan bukannya dua dengan mengambilkira jenis kawalan parameter (berketentuan, penyesuaian,

penyesuaian diri) dan bukti (mutlak, relatif) yang digunakan untuk menentukan perubahan parameter. Taksonomi kedua adalah satu taksonomi baru yang dicadangkan untuk mengklasifikasikan algoritma DE penyesuaian khususnya kepada dua kategori (DE dengan parameter penyesuaian; DE dengan parameter penyesuaian dan strategi) dengan mengambil kira komponen penyesuaian yang digunakan dalam algoritma ini.

Kedua, algoritma DE baru (ARDE-SPX) diperkenalkan yang menyesuaikan diri secara automatik repositori strategi DE dan skim kawalan parameter untuk mengelakkan masalah genangan dan membuat DE respons kepada pelbagai ciri-ciri fungsi di pelbagai peringkat carian evolusi. Algoritma ARDE menggunakan strategi JADE dan skim penyesuaian parameter MDE_pBX sebagai rangka kerja. Kemudian, prosedur penyesuaian baru yang dikenali sebagai repositori penyesuaian (AR) dibangunkan untuk memilih kombinasi yang sesuai bagi strategi JADE dan skim kawalan parameter MDE_pBX bagi menjana penduduk akan datang berdasarkan kepada nilai-nilai kecergasan mereka. Mekanisme repositori penyesuaian adalah skim umum dan boleh digunakan dengan fleksibiliti yang tinggi di dalam mana-mana algoritma evolusi berasaskan populasi. Selain itu, kerja ini telah dilanjutkan untuk mengintegrasikan pengendali *crossover* SPX dengan algoritma ARDE penyesuaian dengan cara pelaksanaan yang baru untuk membuat algoritma ARDE penyesuaian memuaskan kedua-dua keperluan carian global dan tempatan.

Ketiga, keputusan eksperimen dibentangkan untuk mengesahkan kebolehpercayaan ARDE-SPX yang dicadangkan terhadap beberapa varian DE penyesuaian yang sedia. Perbandingan ini dijalankan dari segi ketepatan penyelesaian, kadar kejayaan dan kemantapan terhadap lebih 33 piawaian dan fungsi penanda aras berubah. ARDE juga telah digunakan untuk membangunkan satu rangka kerja pengenalan parameter dinamik yang baru untuk menganggarkan parameter barycentric daripada pengoperasi robot CRS A456. Keputusan simulasi menunjukkan kaedah ARDE adalah lebih berkesan

berbanding teknik konvensional yang lain, melampaui batas algoritma canggih sedia ada dalam menyelesaikan masalah robot.

DEDICATION

I dedicate this thesis to my parents and my brothers for their endless love, support and encouragement throughout my life ...

ACKNOWLEDGMENTS

To my Lord Allah Almighty, I am thankful for the blessings and virtues, and for reconcile, strength, patience, courage, and determination he gave me to complete this work, Alhamdulillah. This work would not be accomplished without the help of so many people. In the following lines is a brief account of some but not all who deserve my thanks.

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Mohd Sapiyan Baba for the continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my supervisor, I would like to thank the rest of my thesis supervisory committee: Dr. Norisma Idris and Prof. Dr. Saad Mekhilef for their support, encouragement and insightful comments.

My sincere thanks also goes to Dr. Azeddien Kinsheel from University of Tripoli, Prof. Dr. Bara'a A. Attea from University of Baghdad and Dr. János Botzheim from Tokyo Metropolitan University for offering me valuable information which helped me to understand many things related to my work and leading me working on diverse exciting projects.

My warmest gratitude goes to all my family members, especially my mother and my father who always believed in me, gave me all the possible support, and being patient

with me for years. I would like to thank my brother Mohanad for his endless support in so many aspects, by giving me help throughout my research, and, of course, sharing my happiness and sorrow by being my wonderful home mate for years. I am also thankful for my second brother Zaid for his support and concern about my study.

I must extend my sincere thanks to the University of Baghdad and the Ministry of Higher Education in Iraq for their support by sponsoring me for the three years and half of my study. None the less, my gratitude to the Malaysian people in general for their perfect hospitality in their green land during my study.

I am deeply thankful for every wonderful friend I have in my country for their endless friendship and support ...

TABLE OF CONTENTS

ORIGINAL LITERARY WORK DECLARATION	ii
ABSTRACT	iii
ABSTRAK	v
DEDICATION	viii
ACKNOWLEDGEMENTS	ix
TABLE OF CONTENTS	xi
LIST OF TABLES	xvi
LIST OF FIGURES	xviii
LIST OF ALGORITHMS	xx
LIST OF SYMBOLS AND ABBREVIATIONS	xxi
LIST OF APPENDICES	xxiv

CHAPTER 1: INTRODUCTION

1.1 Research Background	1
1.2 Research Motivation	3
1.3 Problem Statement	4
1.4 Research Objectives	5
1.5 Research Questions	6
1.6 Scope of Research	7
1.7 Research Significance	8
1.8 Research Processes	9
1.9 Outline of the Thesis	12

CHAPTER 2: DIFFERENTIAL EVOLUTION: A REVIEW

2.1 Introduction	14
2.1.1 Differential Evolution: Definition.....	16
2.2 Why Differential Evolution?	18
2.3 Differential Evolution: Basic Concept and Variants	21
2.3.1 Mutation Operator	24
2.3.2 Crossover Operator	27
2.3.3 Selection Operator	29
2.4 Parameter Settings of Differential Evolution	31
2.5 Unconstraint Optimization Problems	32
2.6 No-Free Lunch Theorem and Domain Knowledge Utilization	35
2.7 Summary	38

CHAPTER 3: ADAPTIVE DIFFERENTIAL EVOLUTION: TAXONOMY AND ANALYSIS

3.1 Introduction	39
3.2 Evolutionary Algorithms Parameter Settings: Extended Taxonomy	41
3.3 Differential Evolution Parameters Tuning	45
3.4 Adaptive Differential Evolution: Procedural Analysis and Comparison	49
3.4.1 DE with Adaptive Parameters and Single Strategy	53
3.4.1.1 Adaptive DE with Single Standard Strategy	53
3.4.1.2 Adaptive DE with Single Advanced Strategy	54
3.4.2 DE with Adaptive Parameters and Multiple Strategies	65

3.4.2.1 Adaptive DE with Multiple Standard Strategies	66
3.4.2.2 Adaptive DE with Multiple Advanced Strategies	69
3.4.3 Adaptive DE Comparisons	71
3.4.3.1 Adaptive DE Conceptual Similarities and Differences	72
3.4.3.2 Adaptive DE Strengthens and Drawbacks	73
3.4.3.3 Discussion and Conclusion	80
3.5 Summary	84

CHAPTER 4: DIFFERENTIAL EVOLUTION WITH ADAPTIVE REPOSITORY OF STRATEGIES AND PARAMETER CONTROL SCHEMES INTEGRATED WITH LOCAL SEARCH METHOD

4.1 Introduction	85
4.2 General Steps to an Adaptive EA	85
4.3 Adaptive Repository of DE Strategies and Parameters Control Schemes Integrated with SPX-Crossover (ARDE-SPX)	88
4.3.1 ARDE-SPX: The Repository of DE Strategies	89
4.3.2 ARDE-SPX: The Repository of Parameters Control Schemes	91
4.3.3 ARDE-SPX: Adaptive Repository with Fitness Based Selection	95
4.3.4 ARDE-SPX: The Local Search of Hill-Climbing Crossover (SPX)	102
4.3.4.1 SPX: Hill-Climbing Simplex Crossover	103
4.3.4.2 SPX Crossover with Group Based Replacement	105
4.3.5 ARDE-SPX: Overall Algorithm Implementation	106
4.3.6 ARDE-SPX: Algorithm Complexity Analysis	107

4.3.7 ARDE-SPX: Comparison with Other Adaptive DE Variants	109
4.4 Summary	111

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Introduction	112
5.2 Experimental Setup	112
5.2.1 Unconstrained Benchmark Functions	112
5.2.2 Algorithms for Comparison	118
5.2.3 Comparison Strategies and Metrics	120
5.3 Experimental Results and Discussions	123
5.3.1 Comparison of multiple DE variants based parameter tuning	123
5.3.2 Comparison of multiple adaptive DE variants.....	129
5.3.2.1 Final solution precision (Mean \pm Std)	129
5.3.2.2 Convergence speed and robustness (FESS, S_r)	138
5.3.2.3 Convergence plot	141
5.4 Summary	154

CHAPTER 6: SYSTEM IDENTIFICATION AND CONTROL OF ROBOT MANIPULATOR BASED ON ARDE ALGORITHM

6.1 Introduction	155
6.2 Research Background	155
6.3 Dynamic Model of the CRS A456 Robot Manipulator	158
6.4 System Implementation	161
6.5 Summary	166

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Introduction167

7.2 Research Conclusions167

7.3 Research Future Work170

REFERENCES173

APPENDIX A186

APPENDIX B192

APPENDIX C194

LIST OF PUBLICATIONS AND PAPERS PRESENTED198

LIST OF TABLES

Table 2.1	Historical elucidation of the Differential Evolution algorithm invention and development	22
Table 2.2	Problem types	32
Table 2.3	Definitions and mathematical formulations of different terminologies related to optimization	34
Table 3.1	Table of DE algorithm control parameters with their estimated corresponding setting guidelines	50
Table 3.2	A summary of the type of mutation strategies used in the ten adaptive DE algorithms	74
Table 3.3	This table encompasses taxonomy of the adaptation scheme used to update the main control parameters in the ten adaptive DE algorithms	75
Table 3.4	DE algorithms points of strengthens based on mutation strategy	76
Table 3.5	DE algorithms drawbacks based on mutation strategy	77
Table 3.6	DE algorithms points of strengthens based on parameter control schemes	78
Table 3.7	DE algorithms drawbacks based on parameter control schemes	79
Table 4.1	The x^2 example of fitness tournament selection detailed steps	97
Table 4.2	The FTS method used to assign the DE strategies and parameter control schemes to the successful individuals	99
Table 5.1	Problem dimension, global optimum parameters set, global optimum value, search range, and initialization range of thirty-three benchmark functions	116
Table 5.2	The F and CR values tuned for each pair of DE mutation variant-benchmark functions	126

Table 5.3	Mean and standard deviation of 30-dimensional and low dimensional problems achieved for multiple DE mutation strategies averaged over 30-independent runs	127
Table 5.4	Mean and standard deviation of 30-dimensional problems, averaged over 50-independent runs for the high dimensional test problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$	132
Table 5.5	Mean and standard deviation of 100-dimensional problems, averaged over 50 independent runs for the high dimensional test problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$	135
Table 5.6	Mean and standard deviation of the low dimensional problems f_8 and $f_{19} - f_{28}$, averaged over 50 independent runs	137
Table 5.7	Mean of the NFEs required to obtain the accuracy level Ter_Err and success rate S_r for 50-independent runs of the 30-dimensional problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$	139
Table 5.8	Mean of the NFEs required to obtain the accuracy level Ter_Err and success rate S_r for 50-independent runs of the 100-dimensional problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$	140
Table 6.1	Barycentric parameters estimation of the single joint CRS A465 robot arm	164
Table 6.2	Mean square error and standard deviation of the estimation methods for the estimated model averaged over 30-independent runs	164

LIST OF FIGURES

Figure 1.1	Combination of Differential Evolution state-of-the-art work	8
Figure 1.2	The most significant applications of Differential Evolution algorithm	9
Figure 1.3	Research processes of developing and evaluating a new adaptive DE algorithm. The parts of contributions have been highlighted	11
Figure 2.1	The typical EA cycle	15
Figure 2.2	Differential Evolution research trend (based on information extracted from Web of Science database site)	20
Figure 2.3	A generate-and-test DE flowchart loop	30
Figure 2.4	Function with multi global and local maximum and minimum points	35
Figure 2.5	General framework of a problem solver	37
Figure 2.6	General classification of domain knowledge methods	37
Figure 3.1	Extended taxonomy of parameters settings in EAs	43
Figure 3.2	The evolution trend of the population variance of a single control parameter F for different values	47
Figure 3.3	Correspondence's tendencies between the mutation probability, P_m and the crossover probability, CR for binomial and exponential crossover. (a) For 30 dimensions problems. (b) For 100 dimensions problems	48
Figure 3.4	Simple classification illustrates the position of each adaptive DE variant with respect to the type of adaptive procedure it applies	51
Figure 3.5	An estimated rank of the adaptive DE algorithms based on their recorded experimental results	83
Figure 4.1	Comparison between Cauchy and Gaussian probability density functions	93

Figure 4.2	The complete structure of the adaptive repository, AR in the ARDE algorithm	101
Figure 4.3	$SPX-2-3-\varepsilon$	104
Figure 5.1	General classification of twenty eight standard benchmark functions	113
Figure 5.2	A snapshot of the Microsoft Office Excel package of t -test	122
Figure 5.3	Convergence performance of the algorithms for nine 30-dimensional functions. (a) f_1 . (b) f_2 . (c) f_6 . (d) f_9 . (e) f_{12} . (f) f_{15} . (g) f_{18} . (h) F_2 . (i) F_9 .	146
Figure 5.4	Convergence performance of the algorithms for nine 100-dimensional functions. (a) f_1 . (b) f_2 . (c) f_6 . (d) f_9 . (e) f_{12} . (f) f_{15} . (g) f_{18} . (h) F_2 . (i) F_9 .	151
Figure 5.5	Adaptation characteristics of F_m and CR_m on the selected functions. (a) $f_1(D = 30)$. (b) $f_1(D = 100)$. (c) $f_9(D = 30)$. (d) $f_9(D = 100)$	153
Figure 6.1	Structure of a single robotic cell for robot assisted orthopaedic surgery	160
Figure 6.2	Coordinate frame assignment of single joint CRS A465	161
Figure 6.3	The behavior of the F and CR values in ARDE algorithm during 200 generations	165
Figure 6.4	Measured torque compared with estimated torque using different methods	166

LIST OF ALGORITHMS

Algorithm 2.1	General scheme of an Evolutionary Algorithm pseudo-code	17
Algorithm 2.2	General pseudo-code fashion of DE algorithm	29
Algorithm 4.1	DE/current-to- p Best/1 with archive strategy	90
Algorithm 4.2	DE/rand-to- p Best/1 with archive strategy	90
Algorithm 4.3	Update the value of F_m scheme	94
Algorithm 4.4	Update the value of CR_m scheme	95
Algorithm 4.5	Generate F value scheme	95
Algorithm 4.6	Generate CR value scheme	95
Algorithm 4.7	The standard FTS algorithm	97
Algorithm 4.8	The standard SPX crossover	105
Algorithm 4.9	The ARDE-SPX algorithm	106

LIST OF SYMBOLS AND ABBREVIATIONS

CI	Computational Intelligence
EA	Evolutionary Algorithm
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
ES	Evolution Strategy
EP	Evolutionary Programming
DE	Differential Evolution
ARDE-SPX	Adaptive Repository Differential Evolution Integrated with SPX
	Crossover
P	Population
t	Index of the current iteration (generation)
T	Total number of iterations (generations)
X	Individual (Vector, Chromosome)
x_i	The individual's variables in global optimization problem, $i = 1, 2, \dots, Np$
Np	Population size
X_{min}	Lower bound of the variable's domain
X_{max}	Upper bound of the variable's domain
f	Fitness function
r	Crossover operator
p_c	Recombination probability
m	Mutation operator
p_m	Mutation probability
X^*	Best solution that has the optimal fitness value
\mathbb{X}	Free search space
\mathbb{S}	Solution search space
ϵ	Precision factor
τ	Stopping condition criteria
F	DE mutation scaling factor
CR	DE crossover rate
D	The individual's dimension (Problem dimension)
$r_1, r_2, \text{ and } r_3$	Mutually different individuals
$\alpha_{i,j}$	Uniform random number generator within the range [0,1)
V_i	Donor vector
$DE/x/y/z$	DE nomenclature scheme reference
bin	Binomial crossover in DE
exp	Exponential crossover in DE
U_i	Trial vector
$\beta_{i,j}$	Uniform random number within the range [0,1]
$rand(0,1)$	Uniform random number generator within the range [0,1)
$randn(m, d)$	Normal random number distribution generator with mean m and standard deviation d .

$randc(m, d)$	Cauchy random number distribution generator with mean m and standard deviation d .
η	Mutation scale factor in DESAP
δ	Crossover rate in DESAP
π	Population size in DESAP
A	The archive in the JADE
x_{best}^p	Random individual from the $p\%$ good solutions in the current population in JADE, and $p \in (0, 1]$
\tilde{x}_{r2}	Random individual selected from $A \cup P$
μ_F	Mean parameter of mutation probability distribution in JADE
μ_{CR}	Mean parameter of crossover probability distribution in JADE
S_F	Set of all the successful mutation probabilities F_i in generation t in JADE
S_{CR}	Set of all the successful crossover probabilities CR_i in generation t in JADE
$mean_L(\cdot)$	The Lehmer mean
$mean_A(\cdot)$	the usual arithmetic mean
LR, c	Learning rate
$x_{gr_{best}}$	The best individual from the $q\%$ group of individuals in MDE_pBX
F_m	Mean parameter of mutation probability distribution in MDE_pBX
CR_m	Mean parameter of crossover probability distribution in MDE_pBX
$mean_{pow}(\cdot)$	Mean power
$F_{success}$	Set of all the successful mutation probabilities F_i at generation t in MDE_pBX
$CR_{success}$	Set of all the successful crossover probabilities CR_i at generation t in MDE_pBX
w_F	Weight factor of F_m
w_{CR}	Weight factor of CR_m
$W, K, \text{ and } F$	Mutation scalar factors in p -ADE
x_{best}	The best individual in the current generation t in p -ADE
x_{pbest}	The best previous individual picked up from the previous generation in p -ADE
$E(\cdot)$	Second moment value
p_k	The probability of applying each DE strategy in SaDE
ns_k	The successful memory in SaDE
nf_k	The failure memory in SaDE
CR_{suc}	Set of all the successful crossover probabilities CR_i in SaDE
η_i	Strategy parameter control variable in SaM within the range $[0,1)$
S_i	The selected DE strategy in SaM
μ_s	Mean parameter of strategy probability distribution in SaM
H_s	Set of all successful DE strategy parameters at generation t in SaM

FTS	Fitness Tournament Selection
S_p	Selection probability in FTS
T_s	Tournament size in FTS
AR	Adaptive Repository in ARDE
$cell_i$	The cell index in AR
nfc_i	Total number of fitness values in cell i
SPX- $n-m-\varepsilon$	The Simplex Crossover
n	The dimension of the search space in SPX
m	The number of parents in SPX
ε	The expanding rate in SPX
O	The centroid of the m parents in SPX
C	The descent solution in SPX
Ter_Err	Termination Error
MAX_FEs	Maximum Number of Fitness Evaluations
std	Standard Deviation
S_r	Success rate
FESS	Average number of function evaluations over successful runs
τ_i	The torque acting on joint i
$\hat{\tau}_i$	The estimated torque acting on joint i
q, \dot{q}, \ddot{q}	The position, velocity and acceleration of robot joints
χ	The robot model parameters
MSE	The mean square error
OLS	Ordinary Least Square method

LIST OF APPENDICES

APPENDIX A: STANDARD BENCHMARK FUNCTIONS	186
APPENDIX B: TRANSFORMED BENCHMARK FUNCTIONS	192
APPENDIX C: STANDARD DE/RAND/1/BIN DELPHI 7 SOURCE CODE	194

CHAPTER 1

INTRODUCTION

1.1 Research Background

Many applications (or global optimization problems), including benchmark problems, have been proliferated in diverse disciplines, such as engineering, science, and medicine. The process of solving such problems has yielded new practical solvers, known as computational intelligence (CI), which are mainly inspired by biological processes such as artificial neural networks and evolutionary computations. These solvers determine the most suitable solution for a certain feasible region when they are applied to different problems. A prominent example of a CI method is the evolutionary algorithm (EA) (Bongard, 2009; Brownlee, 2011; Kephart, 2011), which is a population-based optimizer whose mechanisms are inspired by biological evolutionary processes such as mutation, crossover, and survival selection. EAs have many dialects, including genetic algorithm (GA) (Holland, 1992), particle swarm optimization (PSO) (Kennedy & Eberhart, 1995), and differential evolution (DE) (Storn & Price, 1997). They are all derivative-free methods and require only information regarding the objective function itself, without auxiliary properties (Eiben & Smith, 2003). EAs have successfully solved many numerical and combinatorial optimization problems (Blum & Roli, 2003; Niu & Xu, 2014).

DE is a simple yet powerful evolutionary algorithm (Storn & Price, 1997). It has effectively solved various global optimization problems, including benchmark functions. Moreover, DE has demonstrated good convergence, and its principles are easy to understand. Hence, its popularity has gradually increased and it has been used in many real-world applications (Chakraborty, Abbott, & Das, 2012; Dragoi, Curteanu, Galaction, & Cascaval, 2013).

Current studies on EA and its dialects mainly concentrates on three aspects (Wang, 2011):

- *EAs design*: it refers to the process of balancing the exploration and exploitation capabilities of the algorithm. This process is deemed to be a key factor to the performance of the algorithm. Exploration indicates that an algorithm should be capable of probing extensively into search regions. This capability is closely related to the robustness of the algorithm. By contrast, exploitation focuses the search on the neighborhoods of the current solutions. It directly affects the convergence speed.
- *EAs analysis*: The global convergence properties and the time complexity of EAs have been actively researched through theoretical analysis (He & Yao, 2002).
- *EAs application*: EAs have been widely applied to various fields, such as project scheduling, control system design, task assignment, antenna array optimization, and power system optimization.

The current study considers EAs design. In particular, it examines the implementation of adaptive EAs. This type of EAs, if well designed, can enhance the robustness and convergence performance of the algorithm by dynamically updating the EA parameters for different objective function landscapes during evolution. DE is a representative EA and is very sensitive to its parameter settings; thus, this study aims to investigate these settings with the diverse versions of adaptive DE algorithms. Then, a new adaptive DE algorithm will be designed. EAs application is also considered. Specifically, the new adaptive DE is applied to estimate the parameters of a robot manipulator.

1.2 Research Motivation

The use of hand-tuning is more difficult than expected, as is the preliminary testing of the parameters of any EA, including DE. Given a specific task, one may have to spend much time attempting to fine-tune corresponding parameters. In addition, some objective functions are highly sensitive to the settings algorithmic parameters. This dilemma motivates many researchers to either limit these parameters or to develop a new algorithm. In the new algorithm, control parameters are adapted by employing an adaptive/self-adaptive procedure. These automatic and dynamic adaptive mechanisms address the problems stemming from inappropriate parameter setting and may facilitate the desired increase in convergence rate.

Efficient recently developed adaptive DE algorithms have efficiently addressed various benchmark problems with different characteristics and real-world applications. However, other major issues in adaptive DE must be addressed to test the algorithm comprehensively for users in view of future problem optimization. One of these issues involves the adaptation of various DE mutation strategies through evolution in addition to its control parameters. This trend in the theoretical insight into adaptive DE is highly favorable and remains a progressive research area, thus motivating us to contribute a new adaptive DE algorithm whose performance is competitive with that of other state-of-the-art adaptive DE algorithms, as discussed in Chapter 4.

Moreover, few review studies have captured the overall performance of adaptive DE thus far. Das and Suganthan (2011) conducted a comprehensive survey that addresses almost all of the issues in current research on DE. However, DE parameters control is highlighted only in a short section. This research follows the review conducted by Neri and Tirronen (2010) which presents DE and its most recent advances in a classification format. Detailed experiments have been conducted according to a large set of various benchmark problems to test the overall performance of these algorithmic classes.

Selected adaptive DE variants were included and then discussed under this classification. Recently, Chiang, Chen, and Lin (2013) published a new taxonomy on DE parameters control mechanisms based on the type of parameter values, the number of parameter values, and the information used to adjust the parameter values. However, the review articles that address DE parameters control are rare and out-of-date. This drawback has motivated us to contribute the comprehensive review presented in Chapter 3 to examine some of the major aspects related to DE parameter setting.

1.3 Problem Statement

Unlike other EA dialects, the DE algorithm tends to suffer from stagnation rather than premature convergence (Lampinen & Zelinka, 2000). In DE stagnation, the algorithm may occasionally stop approaching the global minimum even if the population has not converged to local minimum or any other point. The population remains diverse, and new individuals may still enter the population. Nonetheless, the algorithm does not search for better solutions. It may converge but it is unlikely to do so.

Recent studies such as (Mallipeddi, Suganthan, Pan, & Tasgetiren, 2011; Qin, Huang, & Suganthan, 2009) have shown that the effectiveness, efficiency, and robustness of the DE algorithm depends heavily on the selected mutation strategy and its associated control parameters. The sensitivity of the DE algorithm to its mutation strategy and to the corresponding control parameters can significantly deteriorate its performance if the strategy is improperly selected. Hence, the process of choosing a suitable DE strategy and setting its control parameters is difficult and requires much user experience. These studies have also indicated that the use of different DE mutation strategies with various control parameter settings can be appropriate during the evolution and can alleviate the decline in DE performance. However, these adaptive DE algorithms often lose their

efficiency when they are applied to solve complex problems with high dimensions given the distinct characteristics of such algorithms.

Additionally, some problems require different parameter settings at different stages of the evolution. These optimization problems vary in terms of complexity, as follows:

- Some problems require an algorithm with high exploration capability when the solution space increases exponentially with the problem dimension.
- Problem characteristics may change with the increase in the problem dimension; for instance, unimodal problems may become multimodal ones with high dimension.

Therefore, in order to address the aforementioned complex problem instances, in this study an improved DE algorithm that attempts to adaptively choose the suitable DE strategies and parameter control schemes during the different evolution stages is investigated. Moreover, this algorithm has been integrated with a local search method to further improve its performance.

1.4 Research Objectives

The primary objective of this study is to generate an overview of EAs parameter settings. Hence, it has been designed from the ground up to support the control of the various parameters of the EAs in general and of the DE algorithm in particular. The present study has three major objectives:

- 1.** To investigate the adaptation properties of different adaptive DE variants by conducting a structural analysis of each variant. To achieve this objective, the following two sub-objectives are to be accomplished:
 - To present the rudiments of EAs parameter control settings using an extended taxonomy of different approaches used to set these parameters on-the-fly while solving the problem.

- To analyze and describe in depth the working principles, structural modifications, and similarities and differences of certain selected state-of-the-art adaptive DE variants based on the extended taxonomy.
2. To develop an improved DE algorithm (ARDE-SPX) that automatically adapt a repository of advanced DE strategies and parameters control schemes to avoid the problem of stagnation and make DE respond to a wide range of function characteristics at different stages of evolutionary search. Then, to integrate the new algorithm with a local search (LS) method to further improve its performance.
 3. To compare the performance of the proposed ARDE-SPX algorithm with the standard DE and several state-of-the-art adaptive DE versions as follows,
 - To implement the ARDE-SPX on a set of benchmark functions of different characteristics such as, convexity, non-convexity, multimodality, and non-linearity.
 - To develop a new dynamic parameter identification framework to estimate the barycentric parameters of the CRS A456 robot manipulator based on the new ARDE algorithm.

1.5 Research Questions

In this study, the main question is:

- Can the DE algorithm be implemented such that it adapts the mutation strategies and their associated parameter control schemes without explicit tuning?

Four additional questions that can be derived from the main one:

Question 1) How are changes made to the EAs such that they can be considered as self-adapted algorithms?

Question 2) What is the evidence of the change in self-adapted EAs?

Question 3) What main components of the DE algorithm affect its overall performance?

Question 4) Can the integration of an EA with a heuristic method perform better than either of its parent algorithms?

Therefore, our hypothesis is that a DE algorithm with adaptive mutation strategies and parameter control schemes can be designed and implemented for efficient and effective function optimizations.

1.6 Scope of Research

In consideration of the strong and multifaceted contribution trends of the DE algorithm and our tendency toward simplicity, we design a state-of-the-art schematic flow diagram of DE by customizing a distinct alphabetic index for each contribution aspect, as depicted in Figure 1.1. As the figure shows, some or even all areas of DE algorithm can overlap in a common work. The scope of this research has been highlighted in “pink”, and the alphabetic index is labeled as (a.1), (b.2), and (b.3).

This study emphasizes DE parameter settings and how the algorithm performance can be significantly improved by integrating parameter-setting schemes during evolution. This process relieves users of the task of performing difficult and time-consuming manual settings. This area of research related to adaptive DE has been extended further to adapt different DE mutation strategies and to control their parameters. This adaptive DE trend has generated promising solutions given that some DE strategies may effectively solve certain problems but are ineffective with other problems. The scope of this study also covers the integration of the new adaptive DE with a local search (LS) technique. This integration always outperforms its predecessors in benchmark problems or specific applications.

This research also applies the adaptive DE to real-world problems by estimating the parameters of a robot manipulator system.

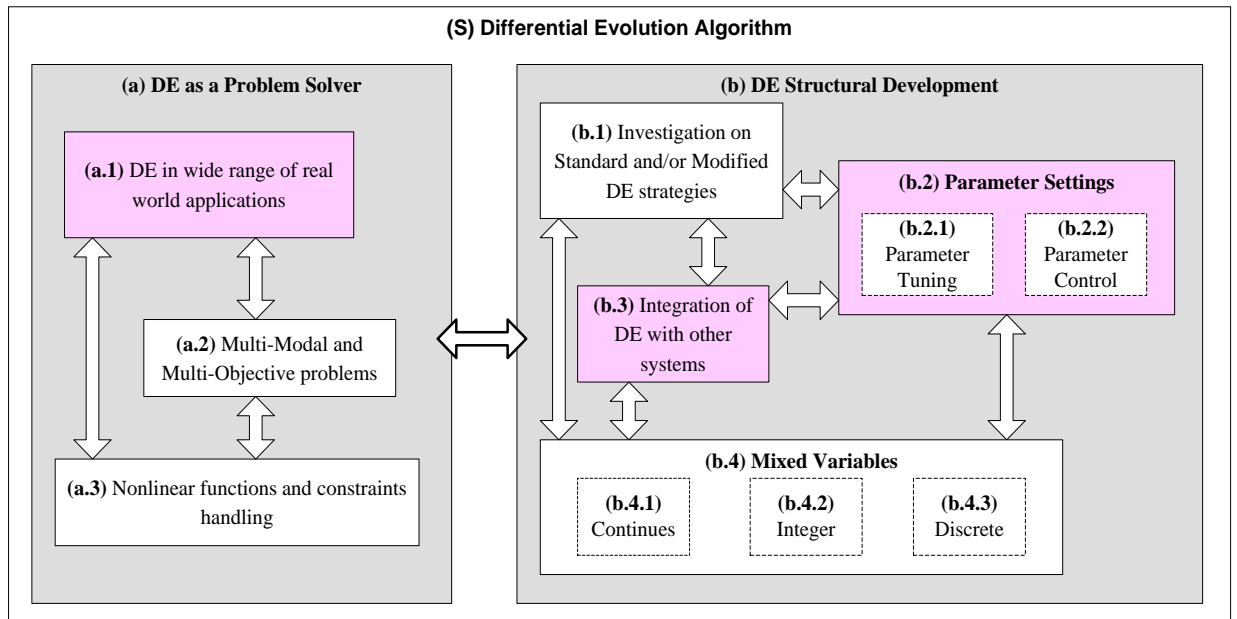


Figure 1.1: Combination of Differential Evolution state-of-the-art work

1.7 Research Significance

DE and its numerous variants have developed rapidly as simple and robust algorithms. Practitioners from different disciplines of science and engineering have applied DE algorithms to address various optimization problems in their own fields. Thus, it can be applied to almost any optimization problem, regardless of whether it is continuous, combinatorial, or mixed-variable. Claims and counterclaims have recently been proposed, especially by engineers, regarding the rules to be followed in choosing the appropriate control values of standard DE parameters with which to solve practical problems. Adaptive DE variants can automatically determine suitable parameter settings; thus, the use of an adaptive DE variant in the present study can significantly benefit many applications. To highlight the significance of the DE method and its variants, Figure 1.2 outlines the applications in which the DE algorithm can be

successfully implemented.

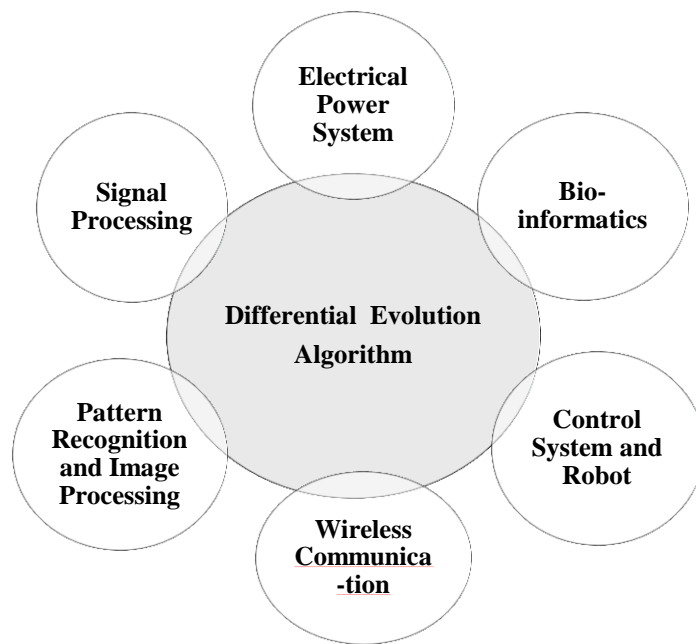


Figure 1.2: The most significant applications of Differential Evolution algorithm (Das & Suganthan, 2011)

1.8 Research Processes

The research process in this study is consists of certain phases and steps, as summarized in Figure 1.3. The main steps to completing the research are outlined in an ordered format. This order emphasizes the importance of achieving one step before transitioning to the next. The steps that detail the contributions of this study are also highlighted in the figure. The processes are briefly described as follows:

Process 1 (EA Definition and Parameters Setting): The general structure, steps and standard classification of the parameter settings of the EAs are investigated.

Process 2 (DE Standard Structure and Variants): The DE algorithm is examined as a prominent example of EA. The process analyzes the overall structure, mechanism, and variants of the DE algorithm.

Process 3 (DE Parameter Setting): The study of the parameter settings of the DE algorithm is divided into two processes:

Process 3.1 (DE Parameter Tuning): The theory regarding the manual tuning of DE parameters control is reviewed. Then setting values for the parameter control of DE are suggested for rapid and good performance.

Process 3.2 (Adaptive DE algorithms): The DE algorithms with adaptive parameters and/or mutation strategies are investigated.

Process 3.2.1 (Analysis of Adaptive DE Algorithms): Different adaptive DE variants are subject to a comprehensive procedural analysis and classified.

Process 3.2.2 (Development of a New Adaptive DE algorithm): The standard DE algorithm is improved by adding an adaptive strategy.

Process 3.2.3 (DE and LS Method): The new adaptive DE algorithm is integrated with a LS technique, and its performance is investigated.

Process 3.2.4 (Evaluation): The developed DE algorithm is evaluated according to a set of benchmark functions with different characteristics. The results of the evaluation are then validated.

Process 3.2.5 (Real-World Application): The new adaptive DE algorithm is applied to a real-world problem by estimating the parameters of a robot manipulator system.

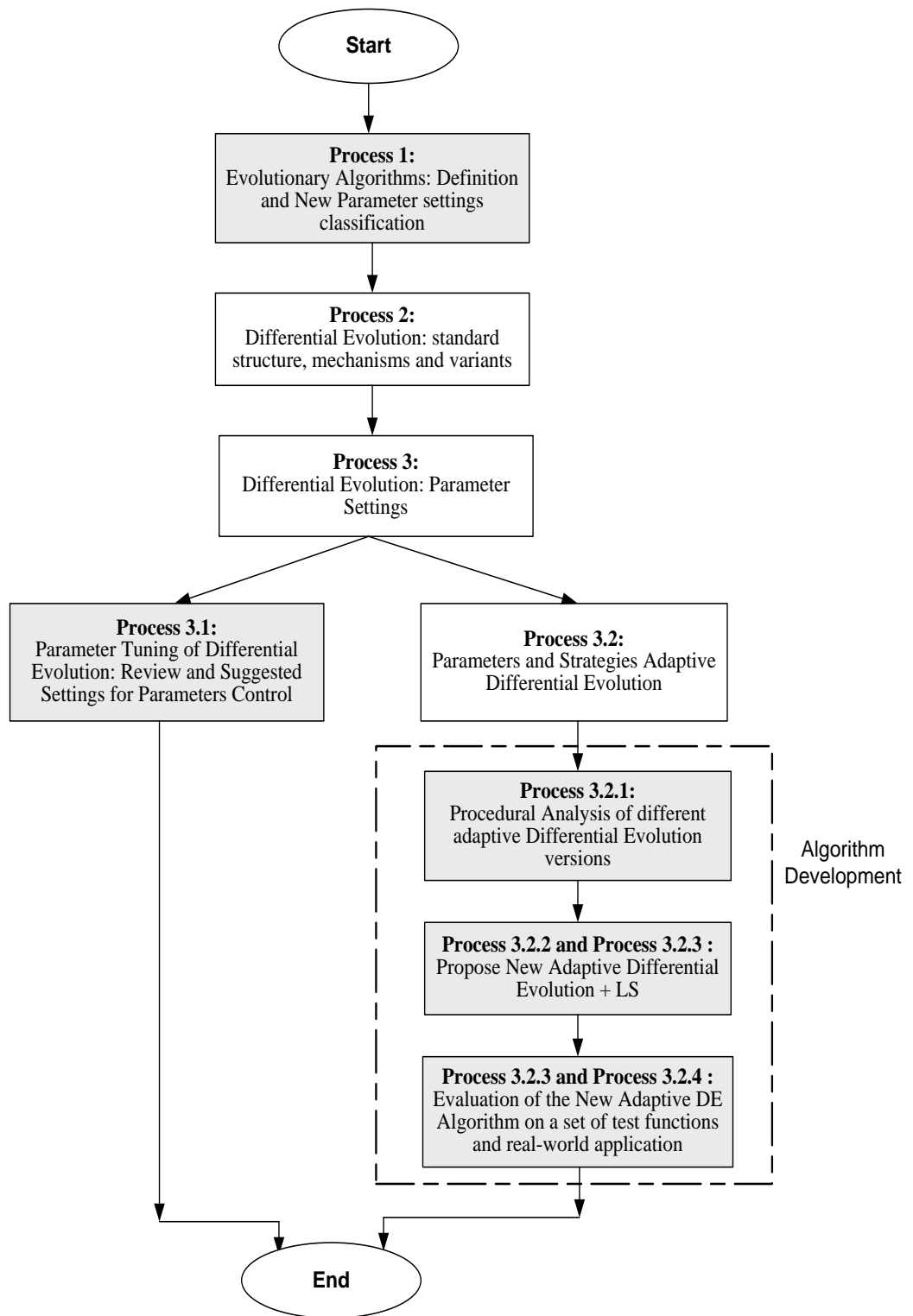


Figure 1.3: Research processes of developing and evaluating a new adaptive DE algorithm. The parts of contributions have been highlighted

1.9 Outline of the Thesis

The remainder of this thesis is organized as follows:

CHAPTER 2 discusses various topics to provide sufficient background on diverse issues that concern the general definition of EAs and their importance as a problem solver in continuous and non-continuous optimization problems, the classical DE algorithm, DE literature, and DE control parameters. This chapter also describes the single-objective optimization function and its associated terminologies, and the general concept of the No-Free-Lunch Theorem.

CHAPTER 3 presents a comprehensive description of EAs parameter settings then provides an extended EA parameter settings' taxonomy. The new extended EA parameter setting taxonomy is applied to multiple adaptive DE algorithms in specific, as an example to convey the main purpose of this taxonomy. Then, a procedural analysis study is established on these algorithms to elucidate the conceptual similarities and differences among them, the pros and cons of the adaptive schemes.

CHAPTER 4 presents the general steps that should be considered to create an EA with parameter control. Then the mechanism of the developed adaptive DE algorithm (ARDE-SPX) is described in details. The description encompasses the mechanism adopted to create the repository of the DE strategies and the parameters control schemes. The description also includes the local search method (SPX) that has been used to improve the performance of the adaptive ARDE algorithm.

CHAPTER 5 provides an experimental study to identify the competitive nature of six DE variants in solving different optimization problems and compare their results. In addition, this chapter presents the results of evaluating the developed adaptive DE algorithm (ARDE-SPX) on a set of benchmark functions with different characteristics in terms of the solution accuracy, convergence speed, and robustness. ARDE-SPX are also compared with several state-of-the-art adaptive DE variants.

CHAPTER 6 presents the application of the new adaptive DE algorithm on the parameter estimation of the CRS A456 robot manipulator system. Simulation results are presented to show the effectiveness of the ARDE method over other conventional techniques in solving the problem of the robot.

CHAPTER 7 concludes the thesis and summarizes the objectives addressed in it. Suggestions and future work development are also offered in this final chapter.

CHAPTER 2

DIFFERENTIAL EVOLUTION: A REVIEW

2.1 Introduction

Real-world optimization problems are encountered in various important applications such as machine learning and system design, and in science and engineering disciplines. Continuous and extensive research efforts have been made to find the best solutions for these problems. The main challenge in finding such solutions is that they often involve uncertainties and/or noise that lead to theoretical optima that are not optimal or practical in real life. Optimization algorithms have been widely employed to address these challenges, and they are used to find parameters (or structures) that maximize or minimize user-defined objective functions. For some typical optimization problems, efficient algorithms are used, whereas for many continuous or non-linear problems heuristics are used to solve them. Moreover, as problem complexity increases in conjunction with the need to reduce the time available for thorough problem analysis and tailored algorithm design, robust algorithms with satisfactory performance become urgently needed. These algorithms should not only be applicable to specific problem, but should also be applicable to a wide range of problems, and yield good (not necessarily optimal) solutions within an acceptable time (Eiben & Smith, 2003). Robust optimization aims to find solutions that not only perform optimally in the theoretical optimization model, but also remain stable despite variations caused by uncertainties and/or noise. Developing such heuristics remains a hot research topic (Weise, 2009).

Natural computation is an emerging interdisciplinary field of computational systems. It utilizes concepts and ideas, and gains insights from natural systems, including ecological, biological, and physical, in which a range of methodologies and approaches

are studied to address large, complex, and dynamic problems. Evolutionary algorithms (EAs) are population-based stochastic search methods that include genetic algorithms (GA), evolution strategy (ES), evolutionary programming (EP) and differential evolution (DE). These algorithms form a rich class of meta-heuristic algorithms and computational intelligence techniques that derive from biological notions such as variation operations (crossover, mutation) and survival of the fittest (selection), for incrementally directing the search course (population) toward a prospective set of better candidate solutions (individuals or chromosomes) (see Figure 2.1). The cost function (evaluation) determines which of the solution “lives” with or without considering the level of constraint handling. These operations compose a loop (generation), and EAs usually execute a number of generations until the obtained best-so-far solution is satisfactory or other termination criterion is fulfilled.

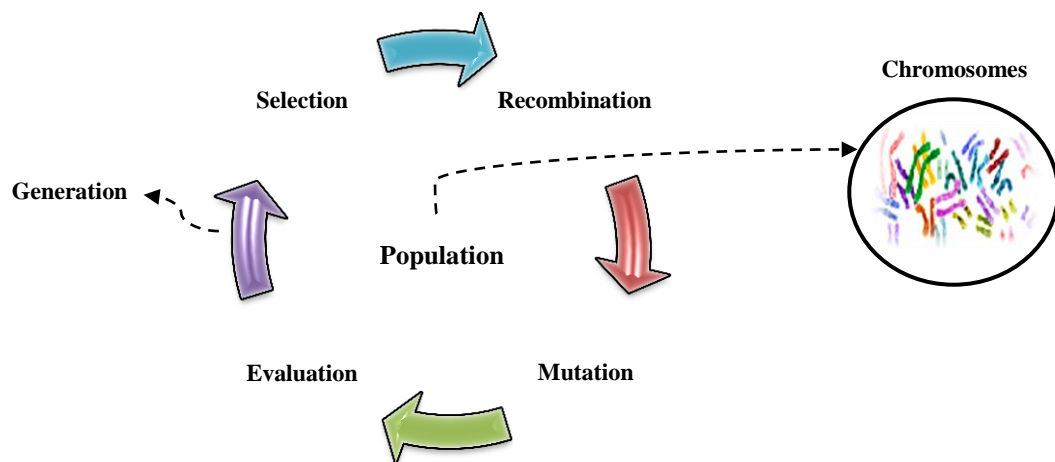


Figure 2.1: The typical EA cycle (Eiben & Smith, 2003)

Although all EAs rely on the same concept of this basic procedure and its probabilistic operators, differences still exist among these algorithms. For example, GA strongly stresses on crossover operator as the main operator to create variations, and decision variables are originally encoded as a bit-string and are manipulated by logical operators, thereby making GA suitable for discrete optimization. As opposed to GA, ES

is continuous optimizer because it encodes variables as floating-point numbers and modifies them by using arithmetic operators, which makes it suitable for continuous optimization. For the main variation operator, ES concentrates more on the mutation operation, although it may also integrate the crossover as an additional operator (Brownlee, 2011; Eiben & Smith, 2003; Goldberg, 1989; Mitchell, 1998).

Globally, EAs provide effective and reliable answers to the challenges of deploying automated solution methods for solving various optimization problems within a minimal amount of time. EAs' features, such as population of solution and variation operators, provide the ability to create a proper balance between exploitation and exploration of the search process when solving problems with complex characteristics (noise, chaos, discontinuous and nonlinearity); by providing a means of escaping from local optima and maintaining solutions diversity. These features distinguish EAs from other traditional optimization methods, such as local search algorithms, and other stochastic algorithms, such as simulated annealing (SA) and various hill-climbing algorithms (HC) (Eiben & Smith, 2003; Fogel, 1994). The general pseudo-code fashion of EAs is illustrated in Algorithm 2.1, which shows the usual approach that reflects the evolution from biology to computer science.

2.1.1 Differential Evolution: Definition

More than a decade ago, DE emerged as a very competitive form of EAs. DE is a population-based optimizer that is effective on a large range of classical optimization problems, primarily to continuous search spaces (Ahandani, Shirjoposh, & Banimahd, 2011; Feoktistov, 2006; Price, Storn, & Lampinen, 2005; Storn & Price, 1997; Storn & Price, 1995; Zou, Liu, Gao, & Li, 2011) like ES. Its effectiveness is due to its properties, such as simplicity, efficiency, and ease of implementation/coding with very few control parameters. Recent studies found that DE produces outstanding results in widely used

benchmark functions (Ahandani, Shirjoposh, & Banimahd, 2011; Piotrowski, Napiorkowski, & Kiczko, 2012; Spadoni & Stefanini, 2012; Storn & Price, 1996; Wang, Cai, & Zhang, 2012) and real-world applications (Goudos, Siakavara, Samaras, Vafiadis, & Sahalos, 2011; Peng, Dai, Wang, Hu, Chang, & Chen, 2011; Zhang, Chen, Dai, & Cai, 2010).

Algorithm 2.1: General scheme of an Evolutionary Algorithm pseudo-code

01: BEGIN

0:2 Step 1 (INITIALIZATION) generate an initial population $P(t = 0)$ with random candidate solutions,
 $\langle x_1^t, x_2^t, \dots, x_{N_p}^t \rangle \in [X_{min}, X_{max}]$;

03: Step 2 (EVALUATION) evaluate each candidate solution $P(t = 0) = \{f(x_1^t), f(x_2^t), \dots, f(x_{N_p}^t)\}$;

04: Step 3 (WHILE STOPPING CRITERION(S) IS NOT FULFILLED) $\tau(P(t)) \neq true$

05: DO

06: Step 3.1 (SELECTION) select the best parents by generating an intermediate population $P(\hat{t})$ from the current population $P(t)$;

07: Step 3.2 (RECOMBINATION) set the parents' pool $SP(\hat{t})$ from interchanging information and genes among two or more parents, randomly and repeatedly, selected from $P(\hat{t})$ as follows,
 $SP(\hat{t}) = r\{p_c\}(P(\hat{t}))$;

08: Step 3.3 (MUTATION) set the mutated pool $C\hat{P}(t)$ by applying a stochastic variability on $SP(\hat{t})$ as follows, $C\hat{P}(t) = m\{p_m\}(SP(\hat{t}))$;

09: Step 3.4 (EVALUATION) evaluate the fitness function for each chromosome $\hat{P}(t) = f(C\hat{P}(t))$;

10: Step 3.5 (SELECTION) select the individuals for the next generation $P(t + 1) = S(\hat{P}(t))$;

11: $t = t + 1$;

12: OD

13: END

The first published article on DE is a technical report entitled “Differential evolution- A simple and efficient adaptive scheme for global optimization over continuous spaces” (Storn & Price, 1995) of the ICSI. Originally, the concept was based on the population-based genetic annealing algorithm that was developed by Price in 1994. After its development, Price modified the annealing algorithm to use arithmetic floating-point vector operations instead of logical ones. Therefore, the annealing mechanism was removed and replaced with differential mutation combined with discrete recombination and pair-wise selection. The recasts changed genetic annealing from a bit-string into a *continuous optimizer* and thus, the obtained algorithm gave rise to DE.

A recent suggestion to create a website that provides useful information about DE was met with interest. Accordingly, the two forefathers of the field, Storn and Lampinen, published their own official bibliography Web sites that supplied all DE materials, such as source codes and some useful links dated from 1995 up to 2002. These sites can be accessed at <http://www.icsi.berkeley.edu/~storn/code/> and <http://www.lut.fi/~jlapine/debiblio.html>.

2.2 Why Differential Evolution?

DE was selected as a parent algorithm in this thesis because the literature indicates that DE is superior to its more traditional EAs cousins, such as GA and ES in fundamental and explicit ways (Das & Suganthan, 2011; Price, Storn, & Lampinen, 2005) as follows:

- *Implementation/coding are easy, flexible, and straightforward.* The main body of the algorithm may require only few lines to code in any programming language. In addition, many references (Feoktistov, 2006; Lin, Qing, & Feng, 2011; Price, Storn, & Lampinen, 2005; Storn & Price, 1997) and Web sites (Beuhren, 22 Sep 2011; Storn, 2000) provide DE open source code written in different programming languages, such as C, MATLAB, Fortran, and Java, which is particularly beneficial for those who unfamiliar with programming as well as practitioners from other fields. The simplicity of DE makes these codes easy to analyze and then modify/amend according to users domain-specific problems. Moreover, although particle swarm optimization (PSO) is also very simple to code, the performance of DE and its variants is largely better than the PSO variants over a wide variety of problems, as indicated by studies like (Das, Abraham, Chakraborty, & Konar, 2009; Rahnamayan, Tizhoosh, & Salama, 2008; Vesterstrom & Thomsen, 2004).

- *Limited number of parameters* to be adjusted (F , CR , and Np in classical DE). The effect of these parameters on the performance of DE and the different ways they have been tuned is discussed in a later chapter.
- *Low and efficient memory consumption* compared with other well-known and competitive real/deterministic parameter optimizers, such as CMA-ES (Hansen & Ostermeier, 2001; Rahnamayan & Dieras, 2008). This advantage, in addition to *lower computational complexity*, increased the demand to utilize DE on a large scale and in expensive optimization problems that have dimensions that may exceed 100 variables. Based on the distinctive characteristics of both algorithms, researchers have recently been encouraged to launch new forms of algorithms that combine DE and CMA-ES into one hybrid system and demonstrated their capability in a number of analytical and empirical studies (Ghosh, Das, Roy, Islam, & Suganthan, 2012; Kaempf & Robinson, 2009).
- It is a significantly *faster optimization algorithm with a high convergence rate* in finding optimal solutions because of two main features. First, when working directly with continuous variables, the use of arithmetic operators instead of logical operators saves more time and removes inaccuracy more efficiently compared with traditional GA. Second, the faster performance of the algorithm is due to the dexterity of the different variants of DE mutation strategies and the greatest freedom in terms of constructing different variations in mutation distributions. Some of these strategies are already based on constructing candidate solutions from the current and superior solution, thereby accelerating the convergence rate (Jeyakumar & Shanmugavelayutham, 2009, 2011; Jeyakumar & Velayutham, 2010; Mezura-Montes, Edith Miranda-Varela, & del Carmen Gomez-Ramon, 2010; Mezura-Montes, Velazquez-Reyes, & Coello, 2006).

All aforementioned advantages may encourage any researcher or practitioner to use DE. DE is still in its infancy (approximately 14 years old) and is being improved gradually, but it has already been established as a universal optimization tool (Das & Suganthan, 2011). An Internet search through Web of Science by using the keyword “*Differential Evolution Algorithm*” revealed about 6,217 relevant articles published between 1996 and 2014. Several thousand application articles in diverse areas were found. The robustness and versatility of DE have encouraged researchers and practitioners from several domains of science and engineering to use DE in solving optimization problems that arise in their own fields. Figure 2.2 shows an abrupt growth in the number of publications and citations related to DE because of its growing popularity as a simple and robust optimizer.

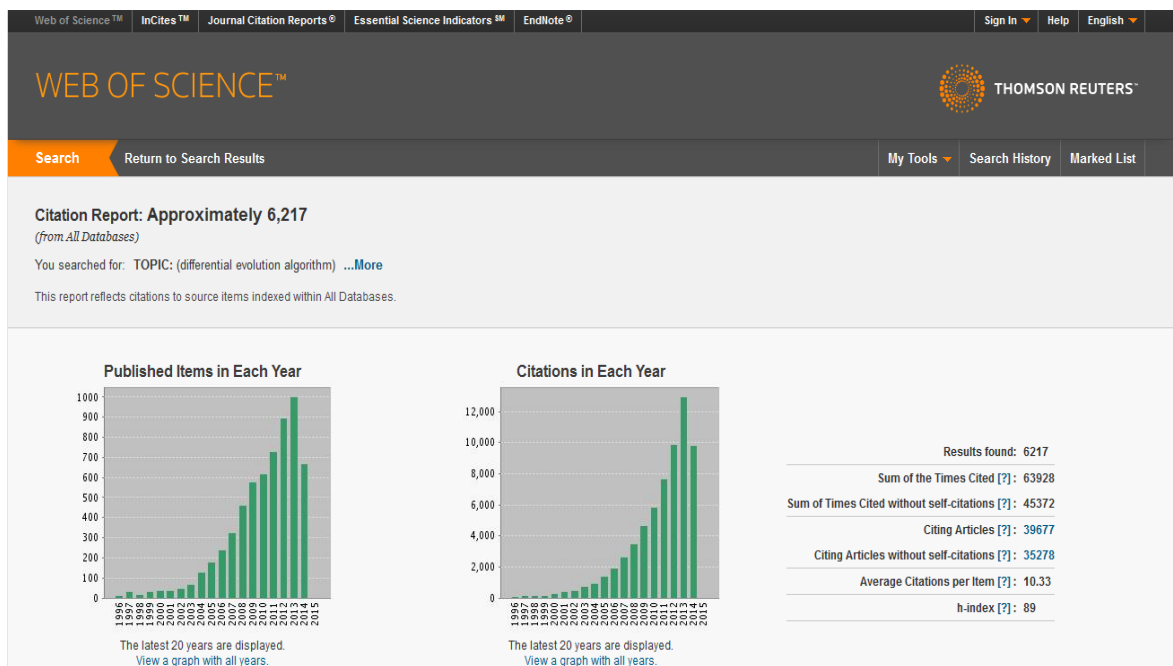


Figure 2.2: Differential Evolution research trend (based on information extracted from Web of Science database site)

The same figure indicates that research on DE from 2008 and 2012 increased and became multifaceted all around the globe within a short time.

Despite the popularity of DE, however, DE performed poorly in many cases. Over the past few years, many researchers have contributed to making DE a general and fast optimization method for any kind of optimization problem by adjusting various constitutes (aspects) of DE (e.g., initialization, mutation, diversity enhancement, and selection), and conducting multiple attempts to automatically adjust the algorithm's parameters for single or multiple problems. Table 2.1 covers excerpts of some of the most prominent milestones and epochs in the DE full story since its discovery in 1996 until current time.

2.3 Differential Evolution: Basic Concept and Variants

Like nearly all EAs, DE is a group-based optimizer that tackles a starting point problem by sampling the objective function at multiple, randomly chosen initial points instead of conducting a point-to-point search. The group is called as *population*, and denoted by $P(t = 0) = \{X_1^t, X_2^t, \dots, X_{Np}^t\}$.

In its basic concept, DE searches for a global optimum point in a D -dimensional real parameter space \mathbb{R}^D . Before a population can be initialized, the upper $X_{max} = \{x_{max,1}, x_{max,2}, \dots, x_{max,D}\}$ and lower $X_{min} = \{x_{min,1}, x_{min,2}, \dots, x_{min,D}\}$ bounds of the preset parameters are specified to define the domain from which the values $x_{i,j}^t; j \in \{1, \dots, D\}$ of each Np real-valued parameter vectors in the initial population are chosen. Hence, the initial population is nourished by candidate solutions using the standard equation:

$$x_{i,j}^{t=0} = X_{min,j} + \alpha_{i,j} \cdot (X_{max,j} - X_{min,j}) \quad (2.1)$$

Table 2.1: Historical elucidation of the Differential Evolution algorithm invention and development







Details	Citation (papers)	Juncture																	
		1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
The first technical report was written on DE.	(Storn & Price, 1995)	●																	
The successful of DE was demonstrated at the First International Contest on Evolutionary Optimization (1 st ICEO).	(Storn & Price, 1996)		●																
Two journal articles published describing DE in details and its very good results	(Price & Storn, 1997; Storn & Price, 1997)			●															
DE was presented at the Second International Contest on Evolutionary Optimization (2 nd ICEO)	(Price, 1997)			●															
A compendium on DE ‘New Ideas on Optimization’ has been summarized by Price.	(Price, 1999)					●													
New mutation schemes were presented and some other developed strategies.	(Fan & Lampinen, 2002; Fan & Lampinen, 2003; Feoktistov & Janaqi, 2004a, 2004b; Feoktistov & Janaqi, 2004d)								●										
In 2007, two types of crossover operations were considered: binomial and exponential schemes. In 2011, two new crossover schemes were designed: consecutive binomial crossover and non-consecutive exponential crossover.	(Lin, Qing, & Feng, 2011; Price, 1999; Zaharie, 2007)					●							●					●	
The first modifications of DE selection rule for constraints handling, were presented.	(Lampinen, 2001, 2002; Montes, Coello Coello, & Tun-Morales, 2004)							●	●		●								
Preliminary recommendations on how to choose appropriate parameter settings of DE	(Lampinen & Zelinka, 2000; Price & Storn, 1997; Price, 1997, 1999; Storn, 1996; Storn & Price, 1996; Storn & Price, 1997; Storn & Price, 1995)	●																	
 Specified number of DE publications in general fields  Unspecified number of DE publications in general fields																			
 Unspecified number of DE publications in the field of parameter settings																			

Table 2.1- Continued

Details	Citation (papers)	Juncture																		
		1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	
The era of developing adaptive DE parameter control methods started in 2002 and continues to be a very hectic trend until current time.	(Abbass, 2002; Brest & Maucec, 2011; Das, Mandal, & Mukherjee, 2014; Liu & Lampinen, 2002a; Liu & Lampinen, 2005; Teo, 2006; Tvrdik, 2009; Urfalioglu & Arikan, 2011; Wang, Rahnamayan, & Wu, 2013; Yin, Wang, & Hu, 2012; Zaharie, 2002b; Zhao, Wang, Chen, & Zhu, 2014; Zhu, Tang, Fang, & Zhang, 2013)																			
DE in a wide range of real-world applications with good results.	(Chandra & Chattopadhyay, 2014; Develi & Yazlik, 2012; Liu, Ni, Liu, & Xu, 2012; Oh, Kim, & Pedrycz, 2012; Ponsich & Coello Coello, 2011; Storn, 1996; Subudhi & Jena, 2011; Titare, Singh, Arya, & Choube, 2014; Zhang, Chen, Dai, & Cai, 2010)																			
DE framework significant improvements have been presented in many aspects: - Structure based randomization of individuals: Compact DE. - Integration based randomization of individuals with explicit exploitative component: Memetic Differential Evolution and its variants. - Increase the search moves of DE with structured population: Parallel DE; Distributed DE; Micro-DE - Other hybridization and improvements in the overall structure of DE which is still an ongoing research field.	(Caraffini, Neri, & Poikolainen, 2013; Feoktistov & Janaqi, 2004c; Kukkonen & Lampinen, 2005; Mininno, Neri, Cupertino, & Naso, 2011; Neri & Mininno, 2010; Neri & Tirronen, 2008; Qasem & Shamsuddin, 2011; Sayah & Hamouda, 2013; Sindhya, Ruuska, Haanpaa, & Miettinen, 2011; Weber, Neri, & Tirronen, 2011; Weber, Tirronen, & Neri, 2010; Zaharie, 2004)																			
 Specified number of DE publications in general fields  Unspecified number of DE publications in general fields																				
 Unspecified number of DE publications in the field of parameter settings																				

where $\alpha_{i,j}$ is a random number generator that returns a uniformly distributed random number within the range $[0,1)$, that is, $0 \leq \alpha_{i,j} < 1$, and is instantiated independently for each component of the i^{th} vector. The classical DE pseudo-code is depicted in Algorithm 2.2. The algorithm scheme shows that after the initialization step (lines 1-10), DE circulates in a loop of evolutionary operations, that is, mutation (lines 12-17), crossover (lines 24-28), and evaluation and selection (lines 29-32). *selectfloat* denotes that the user chooses a floating-point number within a specified range, whereas *selectnum* denotes that the user chooses an integer number within a specified range. For clarification, Figure 2.3 shows a flowchart of the steps of DE.

2.3.1 Mutation Operator

Almost all EAs methods subscribe to an important scheme that is responsible for producing the noisy vectors, preventing the risk of stagnation, responsible for the convergence rate of the algorithm, and diversity of population, throughout the evolution process; this strategy is referred to as *mutation operation*. Specifically, in DE algorithm mutation operation is considered as the main operation, it plays a key role in the performance of the algorithm. Moreover, the name of DE algorithm is itself drawn from the mechanism of this operation since the mutation amount which is called *mutant vector* $V_i(t) = \{v_{i,1}^t, v_{i,2}^t, \dots, v_{i,D}^t\}$ is derived from differentiate multiple randomly selected members, r_1 and $r_2 \in \{1, 2, \dots, Np\}$, of the current population to produce the mutant vector. These indices should be mutually different and also different from the current index i . In DE literature, these parent vectors are called *target vectors* X_i^t . A parameter called *scaling factor*, $F_y \in [0, 2]$, is then multiplied by the y^{th} difference value $(x_{r_1,y}^t - x_{r_2,y}^t)$ to control the amplification of the differential variation. In many recent DE algorithms, each pair of difference vectors might be associated with its own F_y , this

has been introduced to alleviate the greediness tendency of the previous schemes by providing an appropriate means when incorporating the best individual value, X_{best} , from the current population in the difference scheme. According to DE logic, a provisional (interim) offspring so-called *donor vector* $V_i(t)$ is then generated using the standard mutation equation,

$$v_i^t = b_i^t + \sum_{y \geq 1} F_y (x_{r_{1y}}^t - x_{r_{2y}}^t), \quad 1 \leq i \neq r_{1y} \neq r_{2y} \leq Np \quad (2.2)$$

In several well-known DE mutation strategies the base vector, b_i^t , is the best member of the running population, for the intension that the information of the superior individual could be propagated among the population. Diverse mutation schemes have been extracted from the above equation and have been subsequently proposed in literature (Feoktistov, 2006; Liu & Lampinen, 2005; Price & Storn, 1997; Price, Storn, & Lampinen, 2005). Equations (2.3-2.10) list the eight most frequently used mutation strategies; they are as follows connected with the name of each corresponding strategy,

$$DE/rand/1 \quad v_{i,j}^t = x_{r1,j}^t + F_1 \cdot (x_{r2,j}^t - x_{r3,j}^t) \quad (2.3)$$

$$DE/best/1 \quad v_{i,j}^t = x_{best,j}^t + F_1 \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.4)$$

$$DE/rand/2 \quad v_{i,j}^t = x_{r1,j}^t + F_1 \cdot (x_{r2,j}^t - x_{r3,j}^t + x_{r4,j}^t - x_{r5,j}^t) \quad (2.5)$$

$$DE/best/2 \quad v_{i,j}^t = x_{best,j}^t + F_1 \cdot (x_{r1,j}^t - x_{r2,j}^t + x_{r3,j}^t - x_{r4,j}^t) \quad (2.6)$$

$$DE/current - to - rand/1 \quad v_{i,j}^t = x_{i,j}^t + F_1 \cdot (x_{r3,j}^t - x_{i,j}^t) + F_2 \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.7)$$

$$DE/current - to - best/1 \quad v_{i,j}^t = x_{i,j}^t + F_1 \cdot (x_{best,j}^t - x_{i,j}^t) + F_2 \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.8)$$

$$DE/rand - to - best/1 \quad v_{i,j}^t = x_{r3,j}^t + F_1 \cdot (x_{best,j}^t - x_{r3,j}^t) + F_2 \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.9)$$

$$DE/current - to - rand/2 \quad v_{i,j}^t = x_{i,j}^t + F_1 \cdot (x_{r3,j}^t - x_{i,j}^t) + F_2 \cdot (x_{r1,j}^t - x_{r2,j}^t) + F_3 \cdot (x_{r4,j}^t - x_{r5,j}^t) \quad (2.10)$$

The eight above strategies can be simplified to the standard convention $DE/x/y/z$ where x is the manner in which the individual to be perturbed, y is the number of difference vector pairs involved in the construction of the mutant vector, where a difference vectors means the difference between two randomly selected members from the current population excluding the current and the best vectors from them. More often, y is only associated with 1 or 2. z is the type of crossover used [which we will be taking a closer look at in the next definition]. It is worth noting that some of these DE strategies have been misnamed in many publications, which has led to a misconception to the mechanism of some of the new proposed DE methods, as in (Qin & Suganthan, 2005; Zhang & Sanderson, 2009a). Thus, in this study we preferred to recall all the well-known DE strategies with their corresponding names in order to eliminate any confusions related to them in the future.

Recently, $DE/current-to-rand/1$ without crossover has been proposed in (Iorio & Li, 2004) as a rotation-invariant mutation strategy and has proved to provide good results for multi-objective optimization problems and rotational problems.

It is also worth remembering that after applying mutation operation to the individual values, we may run the risk of exceeding the boundaries limitations of all/some of the donor vector values; therefore (Algorithm 2.2, lines 18-23) are used to tackle this problem and to guarantee that the vector components are always inside the

valid boundary values. In literature, there are some other proposed solutions (Price, 1999; Price, Storn, & Lampinen, 2005; Rönkkönen, Kukkonen, & Price, 2005; Storn & Price, 1997; Storn & Price, 1995) for the ‘out of bounds’ problem, they are either to set the values on bound (reflected from the specified boundaries), or use them as they are (out of bound) and let the algorithm itself adjusts them automatically through the evolution process.

2.3.2 Crossover Operator

After the mutation phase, another substantial perturbation operation called *crossover* or *recombination* process comes into play, which is subsequently applied to further raise the potential of population’s diversity. Two standard crossover interpretations are considered: *binomial crossover (bin)* and *exponential crossover (exp)*, as illustrated in Equation 2.11 and 2.12 respectively, to deliver the so-called *trial vector* $U_i(t) = \{u_{i,1}^t, u_{i,2}^t, \dots, u_{i,D}^t\}$. For both interpretations a user-specified real parameter value called *crossover probability* or *crossover rate*, $CR \in [0,1]$, is used to control the mixing process. Considering once again the aforementioned convention $DE/x/y/z$, the letter z is substituted to *bin* or *exp* regarding to which crossover strategy we are referring to. Accordingly, the eight DE variants are then extended to be a total of $8 \times 2 = 16$ variants after combining one type of mutation scheme with either “binomial” or “exponential” crossover scheme (Price, Storn, & Lampinen, 2005; Storn & Price, 1995).

In binomial genewise crossover, a component of the donor vector $v_{i,j}^t$ is inherited with probability CR for the offspring $u_{i,j}^t$, and with probability $1 - CR$ from the target vector $x_{i,j}^t$.

$$DE/x/y/bin \quad u_{i,j}^t = \begin{cases} v_{i,j}^t & (\beta_{i,j} \leq CR) \text{ or } (j = j_{rand}) \\ x_{i,j}^t & \text{otherwise} \end{cases} \quad (2.11)$$

where $\beta_{i,j}$ is a real number, uniformly generated in the range $[0,1]$. j_{rand} is a randomly generated integer in the range $[1, D]$ to ensure that the trial vector $U_i(t)$ will differ from its corresponding target vector $X_i(t)$ by at least one component. This crossover convention is analogue with the so-called uniform crossover which is often used in EAs. On the other hand, in the case of exponential crossover a consecutive component(s) from the donor vector are truncated and then donated to the trial vector after the latter has already inherited all its parameters from the target vector. Initially, two cut points are randomly chosen and then applied to the donor vector; They are respectively: the first cut point, $n \in \{1, \dots, D - 1\}$ and the second cut point, $L \in \{1, \dots, D - 1\}$ with probability $Prob(L = h) = CR^h$; where h is the number of the mutated components. The number of components between the points n and L are counted in a circular manner depending on either a series of Bernoulli experiments of probability CR or the crossover length has already achieved the $D - 1$. New random decisions are made for both n and L for each trial vector $U_i(t)$.

$$DE/x/y/exp \quad u_{i,j}^t = \begin{cases} v_{i,j}^t & \text{for } j = \langle n \rangle_D, \langle n + 1 \rangle_D, \dots, \langle n + L - 1 \rangle_D \\ x_{i,j}^t & \text{for all other } j \in [1, D] \end{cases} \quad (2.12)$$

where the acute brackets $\langle \ \rangle_D$ denote a modulo function with modulus D . This type of crossover is similar to the so-called two-point crossover used in EAs. Finally, the pseudo-code of the *bin* crossover strategy is given in (Algorithm 2.2, line 24-28). In this strategy, *irand* is a function used to generate a random integer number within the interval $\{1, \dots, D\}$.

2.3.3 Selection Operator

Then one-to-one greedy selection operation rises to decide whether the trial vector U_i^t would win the competition over its corresponding target vector X_i^t to be a member X_i^{t+1} in the population of the next generation. This competition is normally based on the evaluation of both individuals' fitness function, and since DE has a minimization propensity, the comparison will end up with the solution that has less or equal fitness value, as illustrated in Equation 2.13. This selection strategy has one advantage over many other selection strategies such as tournament selection, rank based selection, and fitness proportional selection, this is so because its unique merit of reserving the old (i.e. target vectors) and the new (i.e. trial vectors) candidate solutions, then to set off the comparison process on these individuals alike.

$$X_i^{t+1} = \begin{cases} U_i^t & \text{if } f(U_i^t) \leq f(X_i^t) \\ X_i^t & \text{otherwise} \end{cases} \quad (2.13)$$

Algorithm 2.2: General pseudo-code fashion of DE algorithm

0: BEGIN

1: Step1 (INITIALIZATION) Initialize the generation counter $t = 0$. Set the mean value of CR and the value of F . Generates an initial population $P(t = 0)$ with random candidate solutions *target vectors*; $\langle X_1^t, X_2^t, \dots, X_{Np}^t \rangle \in [X_{min}, X_{max}]$

2: FOR $i = 1$ to Np do

3: FOR $j = 1$ to D do

4: $x_{i,j}^t = x_{j,min} + \alpha_{i,j}(x_{j,max} - x_{j,min})$

5: END FOR

6: END FOR

7: FOR $i = 1$ to y do

8: $F = \text{selectfloat}[0, 2]$;

9: ENDFOR

10: $CR = \text{selectfloat}[0, 1]$;

11: Step2 (EVOLUTION) DO WHILE budget condition

12: Step 2.2 (*DE/current - to - rand/1* MUTATION) is applied to obtain the *donor vector* V_i^t . Generate three mutually different vectors $r1$, $r2$ and $r3$, and are different from i .

13: FOR $i = 1$ to Np do

14: FOR $j = 1$ to D do

15: $v_{i,j}^t = x_{i,j}^t + F \cdot (x_{r3,j}^t - x_{i,j}^t) + F \cdot (x_{r1,j}^t - x_{r2,j}^t)$

16: END FOR

17: END FOR

18: Step 2.3 (BOUNDARIES CONSTRAINTS) Regularize infeasible mutant vector V_i^t

```

19: FOR  $i = 1$  to  $Np$  do
20:   FOR  $j = 1$  to  $D$  do
21:     IF  $(v_{i,j}^t < x_{j,min})$  or  $(v_{i,j}^t > x_{j,max})$  THEN  $v_{i,j}^t = x_{j,min} + (x_{j,max} - x_{j,min}) \times rand(0,1)$ 
22:   END FOR
23: END FOR
24: Step 2.4 (BINOMIAL CROSSOVER) applies one of the two crossover schemes to obtain the trial
vector  $U_i^t$ 
25:    $k = irand(\{1, \dots, D\})$ 
26:   FOR  $j = 1$  to  $D$  do
27:     IF  $rand(0,1) < CR$  or  $j = k$  THEN  $u_{i,j}^t = v_{i,j}^t$  ELSE  $u_{i,j}^t = x_{i,j}^t$ 
28:   END FOR
29: Step 2.6: (SELECTION) select the individuals with the minimum fitness value for the next
generation
30:   FOR  $i = 1$  to  $Np$  do
31:     IF  $f(U_i^t) \leq f(X_i^t)$  THEN  $X_i^{t+1} = U_i^t$  ELSE  $X_i^{t+1} = X_i^t$ 
32:   END FOR
33: Step 2.7 increments the generation count  $t = t + 1$ 
34: END WHILE
35: END

```

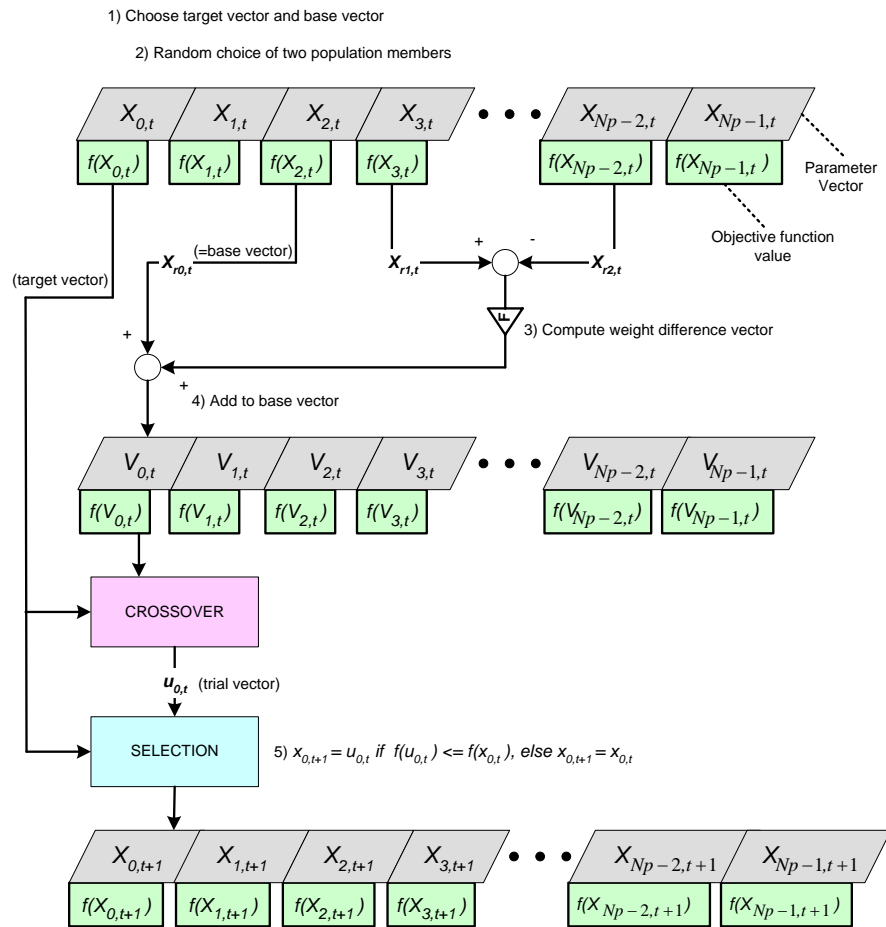


Figure 2.3: A generate-and-test DE flowchart loop (Price, Storn, & Lampinen, 2005)

2.4 Parameter Settings of Differential Evolution

All dialects of EAs are based on the same generic framework whose details need to be specified to obtain a particular EA. These details are commonly known as EA parameters. They include probability of mutation, tournament size of selection, or population size. Designing an EA for a given application requires selecting appropriate values for these parameters. The values of these parameters (which are also called algorithm parameters or strategy parameters) significantly determine whether the algorithm will find an optimal or near-optimal solution or will find such a solution efficiently (Cotta, Sevaux, & Sörensen, 2008; Lobo, Lima, & Michalewicz, 2007).

Fine-tuning is a complicated task, and any attempt to choose the right parameter values is lost a priori. As a result of this drawback, optimal convergence can be achieved and the user can be liberated from tedious parameter tuning trials by altering these parameters on-the-fly during the evolutionary process by taking the actual search progress into account. The main idea is not to choose the parameters semi-arbitrarily¹ but to allow the parameters to adapt to the problem (Eiben, Hinterding, & Michalewicz, 1999; Eiben & Smith, 2003); this type of setting is called *parameter control*. EA literature presents three types of parameter control, namely, *deterministic*, *adaptive*, and *self-adaptive*. DE is a particular instance of EA. Therefore, the issue of parameter control has been investigated in the DE literature.

Recently, the development of adaptive DE algorithms shows faster and more reliable convergence performance than the classical DE algorithms with manual parameter settings over many problems. There are some adaptive DE variants in literature that piqued our interest with their superior performance such as DESAP with self-adapting populations (Teo, 2006); FADE, which is a fuzzy-based DE algorithm (Liu & Lampinen, 2005); improved SaDE (Qin & Suganthan, 2005) for parameters and

¹ The choices were often made from experience

strategies adaptive of DE; jDE, which is a self-adaptive DE algorithm, and its improved version jDE-2, see (Brest, Boskovic, Greiner, Zumer, & Maucec, 2007; Brest, Greiner, Boskovic, Mernik, & Zumer, 2006); and JADE with and without archive (Zhang & Sanderson, 2009b), and so on. The topic of EA parameter settings and adaptive DE algorithms will be discussed in a later chapter.

2.5 Unconstraint Optimization Problems

When optimizing a function or a process, we must first specify our objectives. Any optimization problem can be distinguished according to the presence or absence of (1) an objective function and (2) Constraints. The resulting four categories are shown in Table 2.2 (Eiben & Smith, 2003). When the main task in optimization (minimization or maximization) is to provide the value of a predefined cost or objective function by determining a set of model parameters or state variables, in which the absence of constraints make it less challenging, the problem is called a *free optimization problem* (*unconstraint optimization problem*).

Table 2.2: Problem types (Eiben & Smith, 2003)

Constraints	Objective Function	
	Yes	No
Yes	<i>Constrained optimization problem</i>	<i>Constraint satisfaction problem</i>
No	<i>Free optimization problem</i>	<i>No problem</i>

In free optimization problem, a common (standard) model form $\langle \mathbb{X}, f \rangle$ is frequently defined as the minimization of an objective function (or fitness function) f on \mathbb{X} . If a criterion f is subjected to maximization, then it is of equivalent to minimize its negation ($-f$). A mathematical formulation of the objective function is $f: \mathbb{X} \rightarrow \mathbb{Y}$ with $\mathbb{Y} \subseteq \mathbb{R}$, where \mathbb{Y} is the co-domain of f , and its range should be a subset of real numbers \mathbb{R} . \mathbb{X} is called a *free search space*, which is the set of all elements \mathfrak{x} that can be processed by

search operations. Hence, the optimization target is to find the best element $X^* \in \mathbb{X}$ with respect to such criteria $f \in \mathbb{F}$, that governs the overall performance of the system to be the best under certain conditions (Eiben & Smith, 2003; Weise, 2009). For real parameter optimization, each value of X is a real number. Without losing generality, we will consider only the type of *free optimization problem* with minimization propensity to a *single-objective function*. This type of optimization problems is commonly used to evaluate the performance of any algorithm, for example, DE, GA and PSO, or to verify a rigorous and fair performance comparison of various optimization methods, for example DE and its variants. It also exists in real-world applications such as engineering and scientific applications. De Jong first presented numerous unconstrained test problems, which are then recommended for the purpose (De Jong, 1975; Storn & Price, 1995). These functions are commonly called benchmark test functions. Thirty-three benchmark functions are considered in this study, as given in Appendix A and B. The behavior (complexity) of these problems varies to cover most difficulties faced in the area of continuous global optimization and real-world applications. Most of these functions are considered as difficult to optimize.

The definitions and mathematical formulation of some terminologies, difficulties and challenges related to such functions, are illustrated in Table 2.3 (Weise, 2009).

Figure 2.4 illustrates such a function f defined over a three-dimensional space $\mathbb{F} = (\mathbb{X}, \mathbb{Y}, \mathbb{Z})$. As outlined in the graph, we distinguish between local and global optima. A global optimum is an optimum of the whole domain \mathbb{F} while a local optimum is an optimum of only a subset of \mathbb{F} .

Table 2.3: Definitions and mathematical formulations of different terminologies related to optimization

Terminology	General Definitions	Sub-Definitions	Mathematical formulation
Local Optimum	A local optimum $X^* \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \rightarrow \mathbb{R}$ is either a local maximum or a local minimum	Local Maximum	If $\mathbb{X} \in \mathbb{R}^n$ we can write, $\forall \hat{X} \exists \epsilon > 0: f(\hat{X}) \geq f(X) \forall X \in \mathbb{X}, X - \hat{X} < \epsilon$
		Local Minimum	If $\mathbb{X} \in \mathbb{R}^n$ we can write, $\forall \hat{X} \exists \epsilon > 0: f(\hat{X}) \leq f(X) \forall X \in \mathbb{X}, X - \hat{X} < \epsilon$
Global Optimum	A global optimum $X^* \in \mathbb{X}$ of one (objective) function $f: \mathbb{X} \rightarrow \mathbb{R}$ is either a global maximum or a global minimum. Even a 1-D function $f: \mathbb{X} = \mathbb{R} \rightarrow \mathbb{R}$ may have more than one global maximum, multiple global minima, or even both in its domain \mathbb{X}	Global Maximum	-
		Global Minimum	-
Solution Space	The union of all solutions of an optimization problem is called its solution space \mathbb{S} , where $\mathcal{X}^* \subseteq \mathbb{S} \subseteq \mathbb{X}$. A solution space contains (and can be equal to) the global optimal set \mathcal{X}^* . A valid solution $X \in \mathbb{S}$, which is not an element of \mathcal{X}^* , may exist, particularly within the context of constraint optimization		
Candidate Solution	A candidate solution X is an element of the problem space \mathbb{X}		

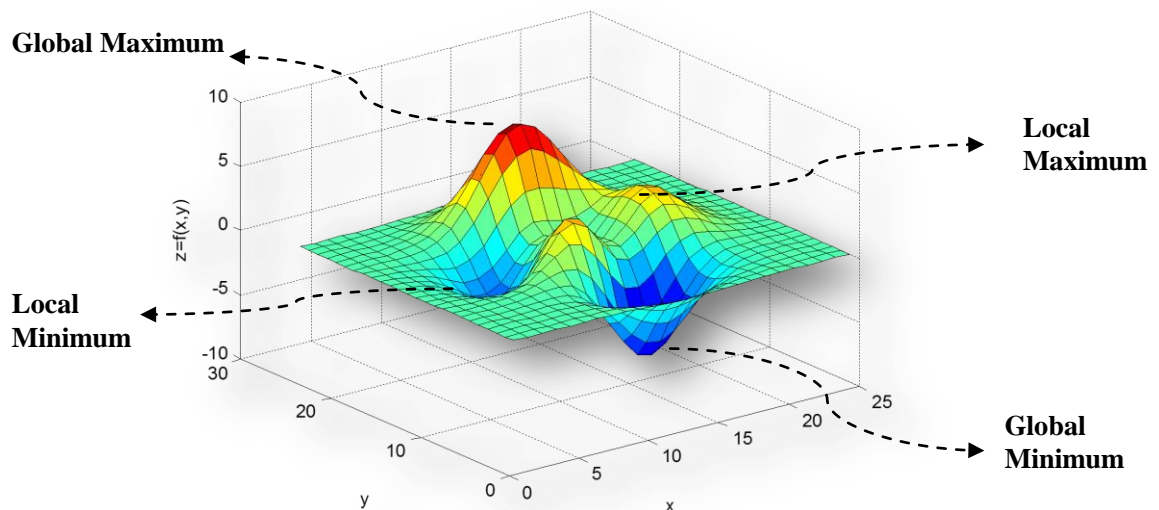


Figure 2.4: Function with multi global and local maximum and minimum points

2.6 No-Free Lunch Theorem and Domain Knowledge Utilization

Most search heuristic algorithms have little or no problem-specific knowledge utilization. For this reason, they are called “black-box” optimization algorithms. Primary examples of the black-box approaches are EAs, simulated annealing (SA), hill climbing (HC), and hill descending (HD). Brute-force approaches or random search approaches are also black box optimization algorithms. Thus, they represent an important benchmark against which the performance of other algorithms may be measured. The general search process idea of these algorithms is to describe how to search the solution space without being restricted to the type of problem. Some of these algorithms may be tailored (i.e. customized) to make them suitable for a particular problem.

Recently, no free lunch (NFL) theorem for search and optimization (Wolpert & MacReady, 1997) has sparked intense debate in the computational intelligence community. NFL theorem has had a considerable effect on the field of optimization research. This theorem states that any optimization algorithm a_i that searches for an

extremum of a cost function performs exactly the same as all other optimization algorithms a_{i-1} s, such as a random search algorithm, when averaged over all possible cost functions f if and only if f is closed under permutation (c.u.p) and each target function in f is equally likely (a uniform probability distribution over fitness functions). NFL can be defined by using Equation 2.14 as follows: For any two black box optimization algorithms a_1 and a_2 , the performance P , averaged over all combinatorial optimization problems f , is constant for any pair of algorithms,

$$\sum_f P(d_m|f, m, a_1) = \sum_f P(d_m|f, m, a_2) \quad (2.14)$$

where m is the number of algorithm iterations, and d_m is the time-ordered set of m distinct points visited. In (Igel & Toussaint, 2004), it is proved that as the cardinality of the search space increases, the fraction of nonempty subsets that are closed under permutation rapidly approaches zero, and the number of local minima and constraints on steepness lead to subsets that are not closed under permutation. Thus, the result is consistent with intuition that on average, there are some algorithms that will perform better than others. However, NFL theorem suggests that this assertion is not precise, and finding an optimal algorithm with exceptional performance is applicable only for limited problems (particularly those of similar characteristics). This idea means that if an algorithm performs well on a set of problems, then it will perform poorly on all others, and any algorithm can outperform another algorithm if it is specialized to the structure of a problem under consideration. Based on the NFL theorem, super-algorithms are non-existent.

As emphasized above, NFL theorem implies that across all optimization problems, the average performance of all algorithms is the same unless it incorporates prior problem domain knowledge into the behavior of the algorithms (such as EAs and its

variant DE as they are not problem-specific in nature) for performance improvement and to match algorithms to problems (Ho & Pepyne, 2002). Therefore, the general framework of a problem solver will be as depicted in Figure 2.5.

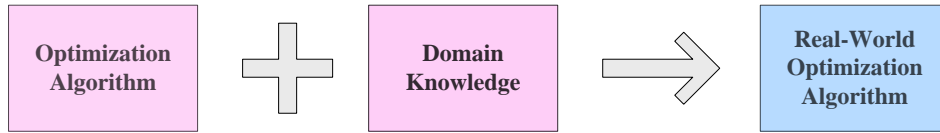


Figure 2.5: General framework of a problem solver

As discussed in (Bonissone, Subbu, Eklund, & Kiehl, 2006), such knowledge can be embedded by using two methods, which are classified in Figure 2.6. It indicates that the performance of any EA can be remarkably improved if integrated with additional domain knowledge approaches.

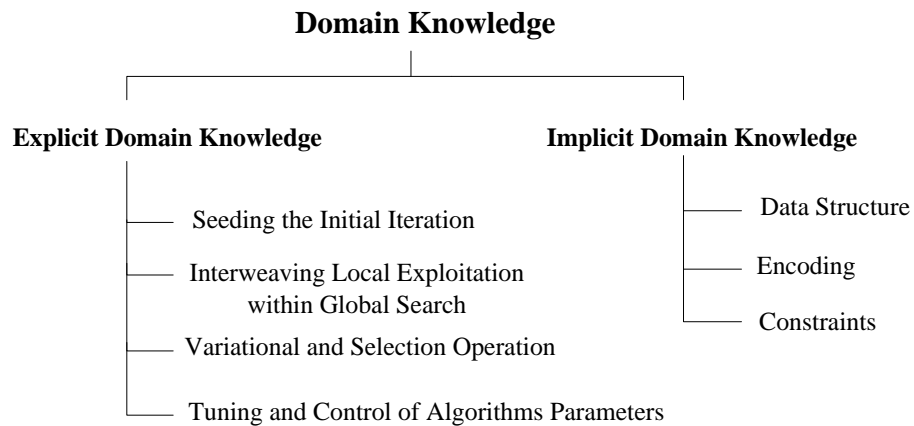


Figure 2.6: General classification of domain knowledge methods

In this thesis, an adaptive DE algorithm that is not problem specific in nature was proposed. This algorithm was compared with other state-of-the-art algorithms on a variety of benchmark functions with different characteristics. The utilization of the

domain knowledge was considered in the application of the proposed algorithm for real-world robot manipulator application.

2.7 Summary

This chapter discusses various topics to provide sufficient background on diverse issues that concern DE literature.

An overview of the general concept of EAs and their importance as a problem solver in continuous and non-continuous optimization problems is initially presented.

The main reasons for selecting DE as a parent algorithm are briefly explained. The classical version that encompasses DE basic operators and variants, are then discussed. The presented DE algorithm in this chapter (Algorithm 2.2) is utilized as a parent algorithm for introducing any other enhanced version. Adaptive DE exhibits superior performance and advancement over classical DE with manual settings, according to the reported results of numerous comparative studies of DE parameter settings. Studies on adaptive DE have asserted the efficiency of this algorithm.

The obtained details of classical DE and adaptive DE will serve as a basis for the procedural analysis and algorithm development in later chapters.

An unconstraint single-objective optimization problem and its classes are illustrated to function as an evaluation and application domain of adaptive DE algorithms. Finally, a brief discussion on the no-free lunch theorem and domain knowledge utilization is presented.

Adaptive DE algorithms have shown efficiency over the standard DE algorithms with fixed parameters over many optimization problems. Therefore, Chapter 3 provides a comprehensive review and analysis on this type of algorithms and their adaptive mechanisms.

CHAPTER 3

ADAPTIVE DIFFERENTIAL EVOLUTION: TAXONOMY AND ANALYSIS

3.1 Introduction

Globally, the goal of parameter settings is to keep up the optimal exploration/exploitation balance so that the algorithm is able to find a global optimum in a minimum amount of time. A well-chosen set of these parameters brings about the convergence performance and robustness of an algorithm with various degrees. The setting of these parameters is problem-dependent, neither an intuitive nor a straightforward task, and requires previous experience of the user. The standard procedure of DE generally disposes the following four main control parameters:

- Population size (Np), which determines the total number of potential solutions in the same generation.
- Mutation scaling factor (F), which determines the amount of differentiation ratios that the perturbed solution could acquire.
- Crossover rate (CR), which determines the probability that the yielded offspring inherits the actual genes of an individual.
- Number of generations (t), which determines the period needed to determine when the DE run would be terminated.

Tremendous research efforts have been focused on finding reasonably good settings for the control parameters of DE, either by manual tuning or alternative adaptive setting techniques. Adaptive DE algorithms exhibit faster and more reliable convergence performance than classical DE with manual parameter settings.

To the best of our knowledge, only a few significant review studies on DE have been reported: Das and Suganthan (2011) published a comprehensive survey article that,

almost, addresses all the issues concerning current DE-research, such as DE and constrained optimization functions, important schemes of DE for single-objective functions, DE in complex environments, theoretical analysis and development of DE, most contemporary engineering applications of DE, and so on. However, the share of DE parameters control topic in this survey is only a terse section that discusses some of the most prominent and recent DE variants in the field. This article followed Neri and Tirronen (2010) survey paper that presents DE and its most recent advances in a classification format, whereby, these reviewed methods were categorized into two main classes: the first class is based on integrating DE with an extra component like local search methods. The second class is based on modifying DE structure. Detailed experiments have been conducted based on a broad set of various benchmark problems to test the overall performance of these algorithmic classes. Recently, Chiang, Chen, and Lin (2013) published a new taxonomy on DE parameters control mechanisms based on the type of parameter values (discrete, continuous), number of parameter values (multiple, individual, variable), and the information used to adjust the parameter values (random, population, parent, individual).

Accordingly, the present study in this chapter is devoted to provide two types of review and analysis on DE parameter settings. First, in order not to overlook the importance of parameter tuning, this chapter presents a short review on DE parameters tuning with a table composed of some estimated recommended guidelines gleaned from literature for setting these parameters. These settings are chosen to almost fit different optimization problems. Second, an overall review and analysis are presented of the state-of-the-art research on certain selected adaptive DE versions, which, according to our judgment, are the most promising and successful parameter control solutions that have been published on relevant forums. This analysis is addressed using a classification of adaptive DE provided in the same section and using the new classification of the

extended parameter control taxonomy of EAs.

3.2 Evolutionary Algorithms Parameter Settings: Extended Taxonomy

The critical decision in implementing any EA is on how to set the values for various parameters of that algorithm. These values greatly affect the performance of the evolution. Parameter settings are commonly composed of crossover rate, mutation step, population size, selection pressure, and penalty coefficient. It is an important and promising aspect of evolutionary computation. The efficiency of any EA greatly depends on the setting of these parameters, that is, by *parameter tuning* or *parameter control*. *Parameter tuning* is also called *off-line setting* and involves using several “standard” parameter setting values in advance and keeping these settings *fixed* during the run, whereas *parameter control* is also called *on-line setting* and involves using another class of approaches where parameters are subject to *change* or *evolve* as problem parameters are optimized. Scientists and practitioners typically tune EA parameters manually and are guided only by their experience and some rules of thumb. Parameter tuning often requires tedious and time-consuming human involvement. Moreover, the process of any EA, and not necessarily DE, is essentially adaptive and dynamic process; thus, using fixed parameters with constant values opposes this essence. Intuitively, the values of these parameters might be optimal at different stages of the evolution; any effort spent toward this direction is indeed lost a priori (Angeline, 1995; Brest, Boskovic, Greiner, Zumer, & Maucec, 2007; Cotta, Sevaux, & Sörensen, 2008; Eiben, Hinterding, & Michalewicz, 1999; Eiben & Smith, 2003). The downsides or limitations of parameter tuning are as follows (Lobo, Lima, & Michalewicz, 2007):

- Parameter values tuned for a single problem may lead to a large difference in performance if these parameters were set to different values.

- A parameter tuned for one test problem and produced superior results may not be as effective in other problems.
- EA parameters are intrinsically dependent; thus, tuning them independently is inconvenient.

An alternative form is parameter control, which refers to when an *automated* setting is applied on EA parameter values. Globally, the automation of parameter settings encompasses three main categories (Cotta, Sevaux, & Sörensen, 2008; Eiben & Smith, 2003; Lobo, Lima, & Michalewicz, 2007):

- ***Deterministic parameter control*** – automation occurs when a deterministic rule is triggered to modify the value of a strategy parameter in a fixed, predetermined manner without using any feedback from the search.
- ***Adaptive parameter control*** – automation occurs during the evolution when the strategy parameter direction and/or magnitude are adjusted according to a pre-designed rule. Basically, automation incorporates information gleaned from the feedback based on algorithm performance, such as the quality of the individual fitness value, without being part of the evolution, where the new control parameter value may or may not persist or propagate throughout the next iterations.
- ***Self-adaptive parameter control*** – automation occurs when the strategy parameters undergo genetic encoding and when the alteration is subject to evolution and pressure (i.e., mutation and crossover); better parameter values tend to produce better individuals (i.e., solutions) with the highest chance to survive and propagate for more off-springs.

Another important criterion that should be considered when discussing parameter control techniques is the evidence of change in parameter value, which can be observed from the performance of operators, the diversity of the population, and fitness values. Evidence can be *absolute* or *relative*. *Absolute evidence* is when a rule is applied to alter

a strategy parameter value on the basis of a predefined event feedback, such as updating the probability of mutation rate in accordance with a fuzzy rule set, population diversity drops at some given value, and even time elapses, rather than being relative to the performance of other values. By contrast, *relative evidence* is when the strategy parameter value is altered according to the fitness of the offspring produced and the better is rewarded; this change is specified relative, not deterministically, to one value present at any time. Therefore, deterministic parameter control is impossible with relative evidence and thus for self-adaptive parameter control with absolute evidence (Angeline, 1995; Cotta, Sevaux, & Sörensen, 2008; Eiben, Hinterding, & Michalewicz, 1999; Eiben & Smith, 2003).

The aforementioned terminologies of the parameter setting of EAs have led to the taxonomy illustrated in Figure 3.1. The new taxonomy is an extension of a former one suggested in (Eiben & Smith, 2003) which caused some confusion among a number of researchers working in this field, particularly in distinguishing deterministic and absolute adaptive rule, as well as relative adaptive rule and self-adaptive rule.

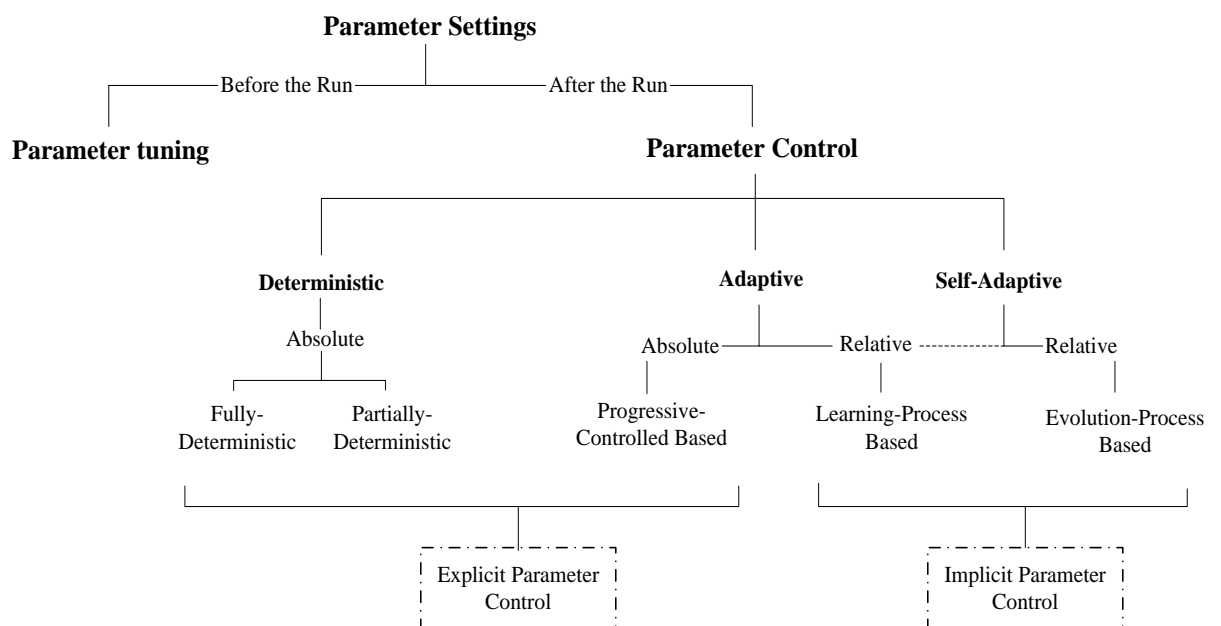


Figure 3.1: Extended taxonomy of parameters settings in EAs

Accordingly, we investigated the subject of parameter control and found new subcategories that can be added to the main one to address the ambiguity in classification. The definitions of these subcategories are as follows:

- ***Fully-Deterministic Scheme and Partially-Deterministic Scheme***, these two subcategories fall under *Deterministic Parameter Control* category. Their main feature is not receiving any feedback from the search during the evolution. Technically, a *fully-predetermined* rule is when a user makes a complete counter-intuition on how to steer the control parameter to the desired direction and/or magnitude; for instance, a rule is triggered on the basis of a certain number of generations that elapsed. By contrast, a *partially-predetermined* rule is when one uses random based scheme to alter, for example, mutation probability after every 100 generations (Fogel, Fogel, & Atmar, 1991).
- ***Progressive-Controlled Based***, this subcategory is applied when some feedback from the search is discriminated as measurements on the basis of user pre-determined rules. When these measurements achieve a threshold, a corresponding adaptive rule is applied to update its relative parameter control. Thus, the progress of updating parameter values, as well as the search, is controlled under inconsiderable intuition. For instance, these measurements may be based on gathering information from previous runs through data mining-based fuzzy-knowledge control (Liu & Lampinen, 2005), theoretical considerations (Smith & Smuda, 1995), or practical experience encapsulation (Lis, 1996). A prominent example of this type of parameter control is the 1/5 success rule of Rechenberg (Rechenberg, 1973; Schwefel, 1977); this rule is applied at certain periodic intervals deterministically.
- ***Progressive-Uncontrolled Based (Learning Process-Based)***, this subcategory is the most prominent in literature. This sub-classification falls between adaptive rule

with relative evidence and self-adaptive with evolution-process rule, because both rules are intersected on the basis of updating the control parameter values associated with each individual, at each generation based on their corresponding fitness value; better features of individuals will be propagated to the next populations over time. The only difference is that in *progressive-uncontrolled rule* feedback is gained from the search to allow the parameter control to gradually adapt by applying a pre-specified strategy, which is most likely analogue to that of crossover and mutation strategies (Hansen & Ostermeier, 1996; Qin & Suganthan, 2005; Zhang & Sanderson, 2009b); most of these strategies are learning schemes that collect experience from previous search. In such methods, the changes performed on the parameter values are fully uncontrolled, because the adaptive rule is associated only with the “fittest” solutions and its corresponding control parameters. This subcategory causes much confusion for some researchers working in this field (Zhang & Sanderson, 2009b). For convenience, we use dashed-line connector to make it optional for researchers who desire to stick with the former taxonomy, not to use the latter one, and include the learning style under self-adaptive category, as shown in Figure 3.1.

Ultimately, categories that are involved in the search and gradually evolved by either learning or evolution strategy as long as the search is not yet terminated, are considered as *implicit parameter control*, otherwise, are *explicit parameter control*.

3.3 Differential Evolution Parameters Tuning

As a starting point, (Storn & Price, 1997; Storn & Price, 1995), in his early studies, examined the DE performance towards different F and CR settings. He concluded that the choice of F has a higher priority impact on the performance of DE than that of CR . This is because of its ability to change the characteristics of the search. Later, and based

on the basis of discussion reported in (Price & Storn, 1997), the study (Liu & Lampinen, 2005) recommends the use of 0.9 as a control parameters setting for both F and CR , whereas the empirical analysis reported in (Zielinski, Weitkemper, Laur, & Kammeyer, 2006) demonstrated that a setting of $F \geq 0.6$ and $CR \geq 0.6$ leads to DE having better performance. Zaharie draws our attention to her two distinctive seminal studies.

The first one (Zaharie, 2002a) was a complete theoretical and empirical analysis of DE control parameter settings and their effects on the population diversity; then she pointed out to a simple relation (see Equation 3.1) that can frequently be used to find appropriate values for the control parameters.

$$\left(2F^2CR - \frac{2CR}{Np} + \frac{CR^2}{Np} + 1 \right) \cdot var(P_x^t) \quad (3.1)$$

where $var(P_x^t)$ indicates the population variance as in Equation 3.2,

$$Var(P_x^t) = \frac{1}{Np} \sum_{i=0}^{Np-1} (x_i^t - \langle x \rangle^t)^2; \quad \langle x \rangle^t = \frac{1}{Np} \sum_{i=0}^{Np-1} x_i^t \quad (3.2)$$

From Equation 3.1 we can see that if the factor $2F^2CR - \frac{2CR}{Np} + \frac{CR^2}{Np} + 1 > 1$ then the variation operators induce an increase in the population diversity, while if $2F^2CR - \frac{2CR}{Np} + \frac{CR^2}{Np} + 1 < 1$ then the population variance decreases. Thus, accordingly, and base on what has already been proved experimentally that the selection operator usually reduces the population diversity, the values of the control parameters which satisfy Equation 3.3 can be considered to be critical; this is only to assure that the factor $2F^2 - \frac{2}{Np} + \frac{CR}{Np}$ is always greater than one.

$$2F^2 - \frac{2}{Np} + \frac{CR}{Np} = 0 \quad (3.3)$$

As such, Figure 3.2 confirms what has been predicted in (Zaharie, 2002a) that $F = 0.1341$ is a critical value when $Np = 50$ and $CR = 0.2$, in the sense that the population variance is close in proximity to the constant. These results were generated using Zaharie's modified version of DE over 100 run averages.

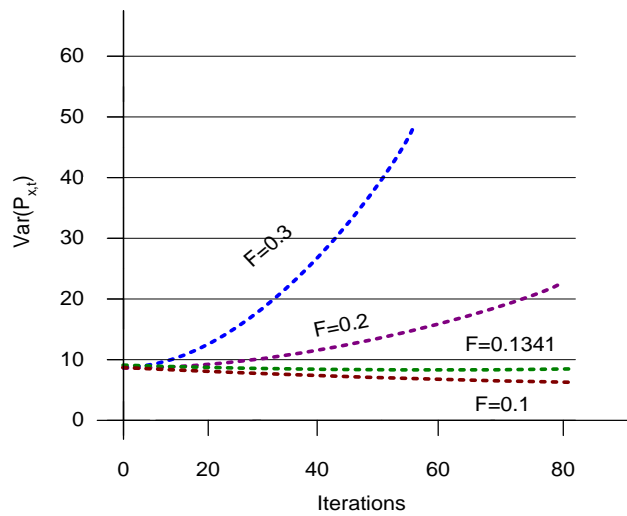


Figure 3.2: The evolution trend of the population variance of a single control parameter F for different values (Price, Storn, & Lampinen, 2005)

Thereafter, she presented her second work in (Zaharie, 2007) as a way of letting the practitioners have sufficient analysis about the influence of the crossover probability CR on P_m for both binomial and exponential crossover variants. Where P_m denotes the probability that a component of an individual is mutated, and can be measured by first identifying the type of the crossover being used, then applying one of the following probability equations:

Binomial Crossover Probability $P_m = CR \left(1 - \frac{1}{D}\right) + 1/D$ (3.4)

Exponential Crossover Probability $P_m = 1 - CR^D/D(1 - CR)$ (3.5)

Figure 3.3 depicts the correspondence tendency between CR value and P_m value for binomial and exponential crossover for two dimensions of the problem ($D = 30, D = 100$).

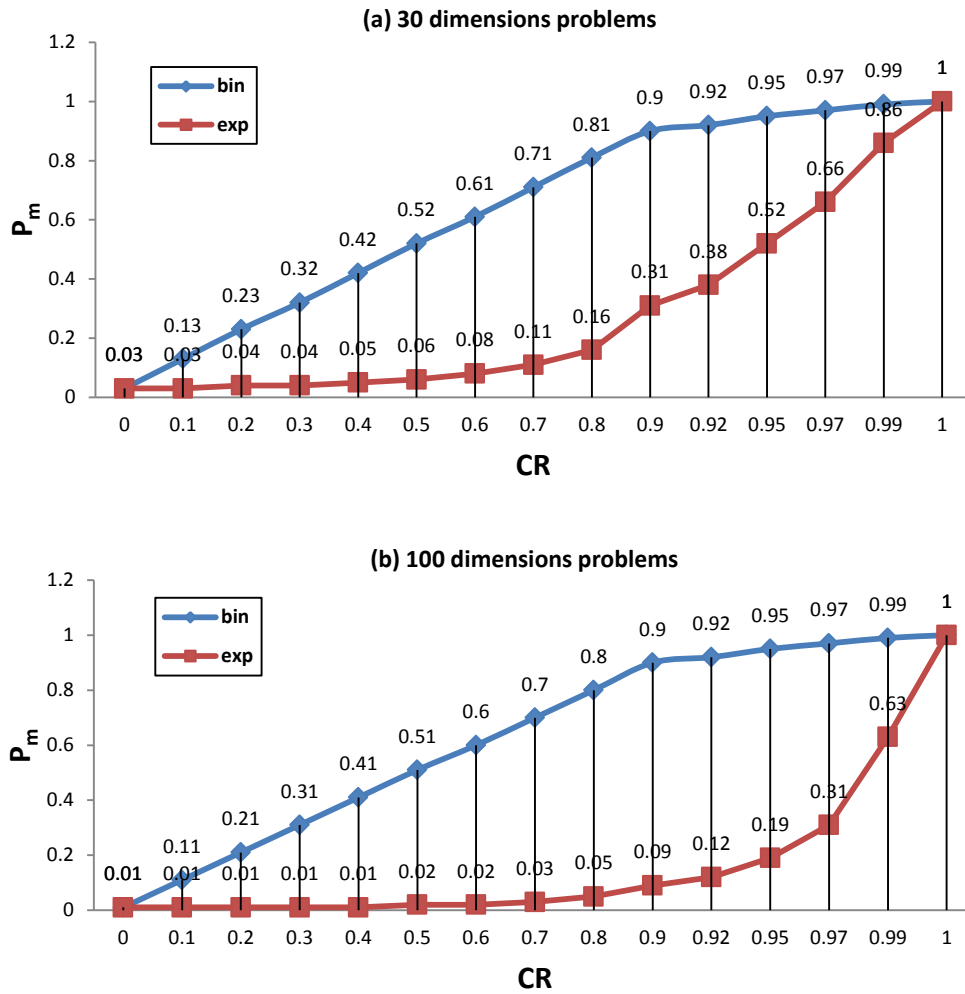


Figure 3.3: Correspondence's tendencies between the mutation probability, P_m and the crossover probability, CR for binomial and exponential crossover. (a) For 30 dimensions problems. (b) For 100 dimensions problems

The same figure shows the linear dependence between P_m and CR in the case of binomial crossover and nonlinear dependence in the case of exponential crossover

combined with and the fact that the difference between both variants is more significant when D is larger. In addition to the previous analysis, she presented in the same article a symmetrical comparison on the performance of the two crossover variants.

Throughout this literature, it has been noted that there is no exact conclusive result on what is preferable for DE parameter tuning. For example, Pedersen in (Pedersen, 2010) has pointed out in his technical report that Rastrigin function (a multi model separable function) (Liu & Lampinen, 2005) has not been solved by DE with ($D = 30, Np = 75, CR = 0.8803$, and $F = 0.4717$) parameter settings and suggested the practitioners try other parameter settings, or to choose another optimization method such as PSO. While Zaharie in (Zaharie, 2007) offered a rigorous analysis for this function in particular and argued that this function can be solved by DE with parameters tuned to ($D = 30, Np = 50, F = 0.5$, $CR < 0.3$ for binomial crossover and $CR \leq 0.97$ for exponential crossover), and has reached the desired optimum with the accuracy $\epsilon = 10^{-6}$.

More often than not, people use what has already worked well in previously reported cases. Hence, we enlisted DE parameter control tuning in Table 3.1. This table includes the corresponding guideline for each parameter based on the encapsulation of practical experience which are the most promising and successful parameter control settings and have already been tested and showed to be effective on relevant forums.

3.4 Adaptive Differential Evolution: Procedural Analysis and Comparison

In literature, there are many recent and prominent adaptive DE variants that show efficiency and reliability in their performance. In this study, most of these algorithms have been reviewed and analyzed on case-by-case bases each according to the facts of its particular situation (parameter control scheme and/or adaptive DE mutation strategy) that has been implemented (see Figure 3.4).

Table 3.1: Table of DE algorithm control parameters with their estimated corresponding setting guidelines

Parameter	Guideline
Np	Generally, the large Np the more robust will be the search, although, sometimes due to complex objective function, the generation will be cumbersome by a considerable amount of computational time, as well as the possibility of increasing in the set of potential ineffective moves. On the other hand, it should not be small in order to avoid stagnation and have adequate population diversity (Liu & Lampinen, 2005; Storn & Price, 1997). Arguably, a plausible choice of Np is recommended in the range $2D \leq Np \leq 20D$.
F	Many studies have been reported, for example (Lampinen & Zelinka, 2000), and arrived at similar conclusions: a small F increases in the probability of DE gets trapped in a local minimum and leads to a premature convergence due to the extensive exploitation caused, whilst a large F makes DE over explorative and significantly slows down the convergence speed. The empirical analyses also suggest, that the use of ($F = 1$) is not recommended since it leads to a significant decrease in the exploration power (Liu & Lampinen, 2005; Storn & Price, 1997). Literature recommends $F \in \left[\frac{2}{Np}, 1 \right]$. Eventually and based on what Zaharie has concluded in (Zaharie, 2002a), the setting ($F \in \left[\frac{2}{Np}, \frac{D}{Np} \right]$), where $D \leq Np$) is highly recommended if wanting to narrow down the range of F ; otherwise a gradual increase of the value within the range ($\frac{D}{Np} < F \leq 1$) is recommended until satisfied.
CR	It is arguable that the small value of CR will lead to a small probability change in the number of parameters being swapped between the two engaged individuals. Otherwise, a high value of CR ($CR = 1$) will dramatically reduce the diversity amount of the offspring solution to be only inherited from the interim individual, which makes the crossover operation useless (Lampinen & Zelinka, 2000). In many studies (Liu & Lampinen, 2002b; Storn & Price, 1997), the setting $CR \in [0.7, 0.9]$ is recommended.
t	Usually determined by assigning a user-defined parameter for an acceptable stopping or convergence criteria (CPU time, total number of fitness evaluations, and so on) then connecting the metric t directly to the evolutionary process (i.e. the aforementioned criteria) (Eiben & Smith, 2003). The range $300 \leq t \leq 5000$ is recommended for high dimensional and noisy problems.

Then, an extensive comparison among these methods has been conducted based on their conceptual similarities and differences, and the pros and cons of each algorithm. These algorithms are as follows:

- **FADE:** is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using fuzzy logic (Liu & Lampinen, 2005).
- **jDE** is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive scheme (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006).

- **DESAP** is a parameter adaptive DE in which the control parameters (F , CR and Np) of DE are all adjusted through evolution (Teo, 2006).
- **JADE** is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive learning scheme (Zhang & Sanderson, 2009b).

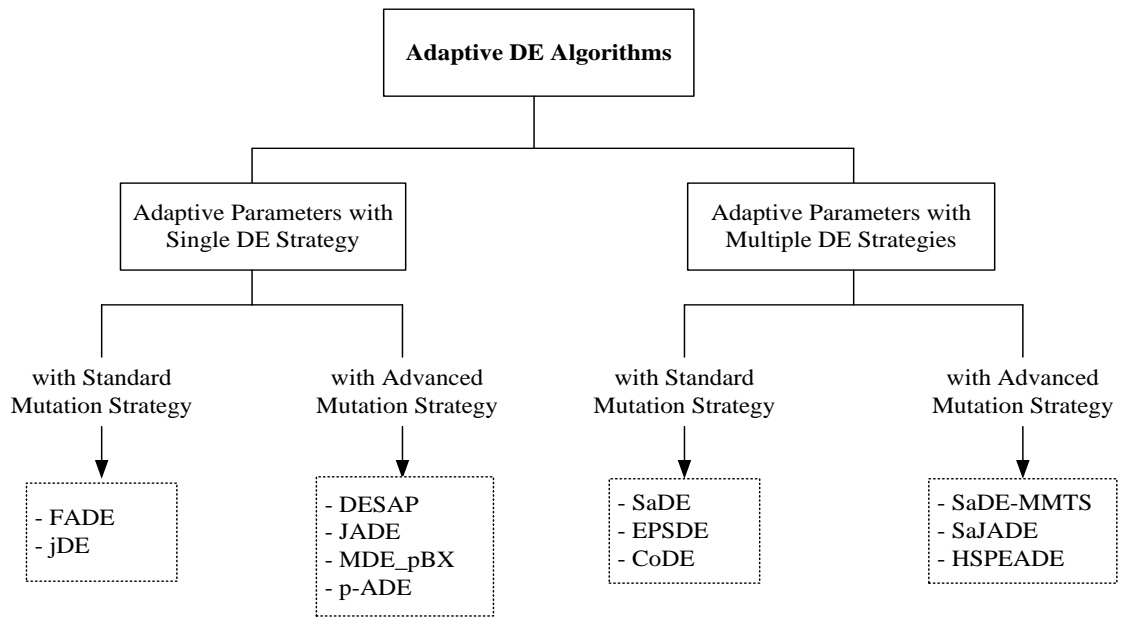


Figure 3.4: Simple classification illustrates the position of each adaptive DE variant with respect to the type of adaptive procedure it applies

- **MDE_pBX** is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive learning scheme (Islam, Das, Ghosh, Roy, & Suganthan, 2012).
- **p-ADE** is a parameter adaptive DE that adjusts the parameters of F and CR and other control parameters related to its mutation scheme in an adaptive manner (Bi & Xiao, 2011).

- **SaDE** is a parameter and strategy adaptive DE in which the control parameters (F and CR) of DE as well as the DE strategies are adjusted using adaptive techniques (Qin, Huang, & Suganthan, 2009).
- **EPSDE** is a new version of adaptive DE in which an ensemble of control parameters and strategies are created then selected randomly for each individual. (Mallipeddi, Suganthan, Pan, & Tasgetiren, 2011);
- **CoDE** is a composition based DE in which three DE strategies and three control parameters values of F and CR are combined randomly to generate the trial vectors (Wang, Cai, & Zhang, 2011).
- **SaDE-MMTS** is a parameter and strategy adaptive DE in which the control parameters (F and CR) of DE as well as the DE strategies are adjusted using adaptive techniques. This algorithm is an integration of SaDE, JADE and local search algorithms (Zhao, Suganthan, & Das, 2011).
- **SaM (SaJADE)** is a parameter and strategy adaptive DE in which the control parameters (F and CR) of DE as well as the JADE strategies are adjusted using adaptive techniques. This algorithm is an improvement to the JADE (Gong, Cai, Ling, & Li, 2011).
- **HSPEADE** is a parameter and strategies adaptive DE. This algorithm is an improvement for the EPSDE algorithm. In this algorithm the Harmony Search algorithm (HS) is used to select the control parameter values of F and CR as well as the DE strategies from the ensemble instead of the random manner (Mallipeddi, 2013).

To convey the aforementioned information, this section is subdivided into three major subsections.

3.4.1 DE with Adaptive Parameters and Single Strategy

In this subsection the main characteristics and mechanisms of six remarkable adaptive DE versions are stated in details, on the basis of parameter adaptive schemes and DE strategies.

3.4.1.1 Adaptive DE with Single Standard Strategy

- ***FADE Algorithm***

FADE uses the standard DE scheme DE/rand/1/bin. It updates the values of F and CR at each generation using a mechanism, which is based on the fuzzy logic controller (FLC); whereby a fuzzy knowledge-based system is used to update the control parameters on-line, in a dynamic adaptive manner to the inconsistent situation.

- The values of function values, population diversity (FC) and parameter vectors (PC), and their updates after n^{th} generations are calculated and then used as input to the FLCs, and the values of the control parameters (i.e. F and CR) are the outputs.
- The values of F and CR are then assigned to the fuzzy sets membership functions.
- “ 9×2 ” IF-THEN fuzzy rules statements are used to formulate the conditional statements that comprise fuzzy logic.
- Mamdani fuzzy inference method is used as the fuzzy control strategy to map from the given inputs to an output.
- The defuzzification process is held to map from a space of fuzzy output into a space of real output.

- ***jDE Algorithm***

jDE uses the standard DE scheme DE/rand/1/bin. It updates the values of F and CR in a self-adaptive manner based on adjusting the control parameters F and CR by means of

evolution and applied at the individual level. First, each individual x_i^t , $i = 1, 2, \dots, Np$ is associated with its corresponding control parameters F and CR . These parameters are then initialized to $F_i^t = 0.5$ and $CR_i^t = 0.9$. The new control parameters of F_i^{t+1} and CR_i^{t+1} are then assigned to random values according to uniform distributions on $[0.1, 1]$ and $[0, 1]$ respectively as follows,

$$F_i^{t+1} = \begin{cases} 0.1 + rand_1 \times 0.9, & \text{if } rand_2 < \tau_1 \\ F_i^t, & \text{otherwise} \end{cases} \quad (3.6)$$

$$CR_i^{t+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_i^t, & \text{otherwise} \end{cases} \quad (3.7)$$

where $rand_j$; $j \in \{1, 2, 3, 4\}$ are uniform random values $\in [0, 1]$. In this algorithm, τ_1 and τ_2 represent the probabilities limits that permit the adjustment of F and CR values; they are both assigned to the same value 0.1.

3.4.1.2 Adaptive DE with Single Advanced Strategy

- **DESAP Algorithm**

- *Advanced DESAP Mutation and Crossover Schemes*

In DESAP the base strategy used is a bit different from the standard DE/rand/1/bin and of some sort similar to the strategy introduced in (Abbass, 2002).

Crossover Scheme: The crossover operator is performed first with some probability, $rand(0,1) < \delta^{r1}$ or $i = j$, where j is a randomly selected variable within individual i . The updating strategy is as follows,

$$X^{child} = X^{r1} + F \cdot (X^{r2} - X^{r3}) \quad (3.8)$$

The ordinary amplification factor F is set to 1, thereby at least one variable in X must be changed. Otherwise the value of X^{child} and its control parameters will be set to the same values associated with X^{r1} .

Mutation Scheme: The mutation stage is implemented with some mutation probability, $rand(0,1) < \eta^{r1}$, otherwise all the values will remain fixed.

$$X^{child} = X^{child} + randn(0, \eta^{r1}) \quad (3.9)$$

As can be seen from the equation above, that DESAP mutation is not derived from one of the DE standard mutation schemes.

○ *DESAP Parameter Control Schemes*

DESAP is proposed not only to update the values of the mutation and crossover control parameters, η and δ , but, rather, it adjusts the population size parameter, π as well in a self-adaptive manner. All parameters undergo the evolution and pressure (i.e. crossover and mutation) in a way analogue to their corresponding individuals. The terms δ and π have the same meaning as CR and Np , respectively, η refers to the probability of applying the mutation scheme whereas the ordinary F is kept fixed during the evolution process. Mainly, two versions of DESAP have been applied. The population size of both DESAP versions (Rel and Abs) are initialized by generating, randomly, a population of $(10 \times n)$ initial vectors X , where n denotes the number of design variables which are already recommended by the authors of the original DE method (Storn & Price, 1995). The mutation probability η_i and crossover rate δ_i are both initialized to random values generated uniformly between $[0,1]$. The population size parameter π_i is initialized in DESAP-Abs version to,

$$\pi_i = \text{round}(\text{initial population size} + \text{randn}(0,1)) \quad (3.10)$$

whereas in DESAP-Rel to,

$$\pi_i = \text{rand}(-0.5,0.5) \quad (3.11)$$

the updating process is then applied on the parameters δ , η and π , at the same level with their corresponding individuals using the same crossover and mutation schemes (see Equation 3.8-3.9).

Updating the crossover rate δ

$$\delta^{child} = \delta^{r1} + F \cdot (\delta^{r2} - \delta^{r3}) \quad (3.12)$$

$$\delta^{child} = \text{randn}(0,1) \quad (3.13)$$

Updating the mutation probability η

$$\eta^{child} = \eta^{r1} + F \cdot (\eta^{r2} - \eta^{r3}) \quad (3.14)$$

$$\eta^{child} = \text{randn}(0,1) \quad (3.15)$$

Updating the population size π

$$\text{DESAP-Abs: } \pi^{child} = \pi^{r1} + \text{int}(F \cdot (\pi^{r2} - \pi^{r3})) \quad (3.16)$$

$$\text{DESAP-Rel: } \pi^{child} = \pi^{r1} + \text{int}(F \cdot (\pi^{r2} - \pi^{r3})) \quad (3.17)$$

$$\text{DESAP-Abs: } \pi^{child} = \pi^{child} + \text{int}(\text{randn}(0.5,1)) \quad (3.18)$$

$$\text{DESAP-Rel: } \pi^{child} = \pi^{child} + \text{randn}(0, \eta^{r1}) \quad (3.19)$$

The ordinary amplification factor F is set to 1. The evolution process of DESAP continues until it achieves a pre-specified population size M , then the new population size is calculated for the next generation as,

$$\text{DESAP-Abs: } M_{new} = \text{round}(\sum_1^M \pi/M) \quad (3.20)$$

$$\text{DESAP-Rel: } M_{new} = \text{round}(M + (\pi \times M)) \quad (3.21)$$

For the next generation and in an attempt to carry forward all the individuals with the remaining $(M_{new} - M)$ individuals, the condition $(M_{new} > M)$ should be satisfied; otherwise, carry forward only the first M_{new} individuals of the current generation.

- **JADE Algorithm**

- *Advanced JADE Mutation Schemes*

There are different mutation versions of JADE have been proposed in (Zhang & Sanderson, 2009a) and (Zhang & Sanderson, 2009b), which we refer to in our study. The first new mutation scheme is called DE/current-to-pbest/1/bin (see Equation 3.22), which it has less greedy property than its previous specification scheme, DE/current-to-best/1/bin, since it utilizes not only the information of the best individual, but the information of the $p\%$ good solutions in the current population indeed.

$$v_{i,j}^t = x_{i,j}^t + F_i \cdot (x_{best,j}^{p,t} - x_{i,j}^t) + F_i \cdot (x_{r1,j}^t - x_{r2,j}^t), \quad (3.22)$$

where $p \in (0, 1]$ and $x_{best,j}^{p,t}$ is a random uniform chosen vector as one of the superior $100p\%$ vectors in the current population. The second mutation scheme with an *external archive*, denoted as A , that has been introduced to store the recent explored *inferior individuals* that have been excluded from the search process and their differences from the individuals in the running population, P . The archive vector A is first initialized to be empty. Thereafter, solutions that are failed in the selection operation of each generation are added to this archive. The new mutation operation is then reformulated as follows,

$$v_i^t = x_i^t + F_i \cdot (x_{best}^{p,t} - x_i^t) + F_i \cdot (x_{r1}^t - \tilde{x}_{r2}^t), \quad (3.23)$$

where x_i^t and x_{r1}^t are generated from P in the same way as in the original JADE, whereas \tilde{x}_{r2}^t is randomly generated from the union, $A \cup P$. Eventually, randomly selected solutions are going to be removed from the archive if its size exceeds a certain threshold, say population size Np , just to keep the archive within a specified dimension. It is clear that if the archive size has been set to be zero then Equation 3.22 is a special case of Equation 3.23.

Another variant has been proposed to further increase the population diversity, named archive-assisted DE/rand-to-pbest/1 as follows,

$$v_i^t = x_{r1}^t + F_i \cdot (x_{best}^{p,t} - x_{r1}^t) + F_i \cdot (x_{r2}^t - \tilde{x}_{r3}^t) \quad (3.24)$$

○ *JADE Parameter Control Schemes*

JADE updates four control parameters (F , CR , μ_F and μ_{CR}) during the evolution process.

Mutation factor (F) and location parameter of mutation probability distribution (μ_F):

The mutation probability F_i is independently generated at each generation for each individual i according to the following formula,

$$F_i = randc_i(\mu_F, 0.1) \quad (3.25)$$

where $randc_i$ is a Cauchy distribution with location parameter μ_F and scale parameter 0.1. If $F_i \geq 1$ then the value is truncated to be 1 or regenerated if $F_i \leq 0$. The location parameter μ_F is first initiated to be 0.5. In this step, JADE shows some similarity in updating the mean of the distribution, μ_{CR} , to the learning style used in Population

Based Incremental Learning (PBIL) algorithm (Baluja, 1994; Baluja & Caruana, 1995). The standard version of the PBIL uses learning rate $LR \in (0,1]$ that must be fixed a priori. Then, by utilizing Hebbian-inspired rule the difference rate $(1 - LR)$ is multiplied by the probability vector (PV) that represents the combined experience of the PBIL throughout the evolution process, whereas LR is multiplied by each bit (i.e. gene's value) of the current individual(s) used in the updating process. Likewise, JADE updates the mutation distribution mean location, μ_F is updated at the end of each generation after accumulating the set of all the successful mutation probabilities F_i 's at generation t , denoted by S_F . The new μ_{CR} is updated as,

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F), \quad (3.26)$$

where $mean_L(.)$ is Lehmer mean,

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (3.27)$$

Crossover probability (CR) and mean of crossover probability distribution (μ_{CR}): The crossover probability CR_i is updated, independently, for each individual according to a normal distribution,

$$CR_i = randn_i(\mu_{CR}, 0.1), \quad (3.28)$$

with mean μ_{CR} and standard deviation 0.1 and truncated to the interval $(0, 1]$. The mean μ_{CR} is first initiated to be 0.5. Then, similar to the updating scheme of the mutation probability mean, the distribution of the crossover mean, μ_{CR} , is updated at each generation after accumulating the set of all the successful crossover probabilities CR_i 's at generation t , denoted by S_{CR} , hence calculate its $mean_A(S_{CR})$. The new μ_{CR} is updated by the equation,

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}), \quad (3.29)$$

where c is a positive constant $\in (0,1]$ and $\text{mean}_A(\cdot)$ is the usual arithmetic mean.

- **MDE_pBX Algorithm**

- *Advanced MDE_pBX Mutation and Crossover Schemes*

Mutation Scheme: The new proposed mutation scheme DE/current-to-gr_{best}/1/bin, utilizes the best individual $x_{gr_{best}}^t$ chosen from the $q\%$ group of individuals *randomly* selected from the current population for each target vector. The group size q of the MDE_pBX is varying from 5% to 65% of the Np . The new scheme can be described as,

$$v_i^t = x_i^t + F_y \cdot (x_{gr_{best}}^t - x_i^t + x_{r1}^t - x_{r2}^t), \quad (3.30)$$

where x_{r1}^t and x_{r2}^t are two different individuals randomly selected from the current population and they are also mutually different from the running individual x_i^t and $x_{gr_{best}}^t$.

Crossover Scheme: The new proposed recombination scheme *p-Best*, has been defined as a greedy strategy; it is based on the incorporation between a randomly selected mutant vector perturbed by one of the p top-ranked individual selected from the current population to yield the trial vector at the same index. Throughout evolution the value of parameter p is reduced linearly in an adaptive manner (see Equation 3.37).

- *MDE_pBX Parameters Control Schemes*

Modifications applied to the adaptive schemes in MDE_pBX: The scalar factor F_i and the crossover rate CR_i of each individual are both altered independently at each generation using JADE schemes (see Equation 3.25 and Equation 3.28). The new

modifications have been applied only to F_m and CR_m adapting schemes. In MDE_pBX, both F_m and CR_m are subscribed to the same rule of adjusting. Firstly, the values of F_m and CR_m are initialized to 0.5 and 0.6 respectively, then are updated at each generation in the following way,

$$F_m = w_F \cdot F_m + (1 - w_F) \cdot \text{mean}_{pow}(F_{success}) \quad (3.31)$$

$$CR_m = w_{CR} \cdot CR_m + (1 - w_{CR}) \cdot \text{mean}_{pow}(CR_{success}) \quad (3.32)$$

where a set of successful scale factors $F_{success}$ and a set of successful crossover probability $CR_{success}$ are generated from the current population. And $| \cdot |$ stands for the cardinality of each successful set. The variable n is set to 1.5 as it proves to give better results on a wide range of test problems. Then the mean power mean_{pow} of each set is calculated as follows,

$$\text{mean}_{pow}(F_{success}) = \sum_{x \in F_{success}} (x^n / |F_{success}|)^{\frac{1}{n}} \quad (3.33)$$

$$\text{mean}_{pow}(CR_{success}) = \sum_{x \in CR_{success}} (x^n / |CR_{success}|)^{\frac{1}{n}} \quad (3.34)$$

Together with calculating the weight factors w_F and w_{CR} as,

$$w_F = 0.8 + 0.2 \times \text{rand}(0, 1) \quad (3.35)$$

$$w_{CR} = 0.9 + 0.1 \times \text{rand}(0, 1) \quad (3.36)$$

the F_m and CR_m are formulized. As can be seen from Equations 3.35-3.36, the value of w_F uniformly randomly varies within the range $[0.8, 1]$, while the value of w_{CR}

uniformly randomly varies within the range[0.9, 1]. The small random values used to perturb the parameters F_m and $mean_{pow}$ will reveal an improvement in the performance of MDE_pBX as it emphasizes slight varies on these two parameters each time F is generated.

Crossover amplification factor (p): Throughout evolution the value of parameter p is reduced linearly in the following manner,

$$p = \text{ceil} \left[\frac{Np}{2} \cdot \left(1 - \frac{G - 1}{G_{max}} \right) \right] \quad (3.37)$$

where $\text{ceil}(y)$ is the “ceiling” function that outputs the smallest integer $\geq y$. $G = [1,2,3, \dots G_{max}]$ is the running generation index, G_{max} is the maximum number of generations, and Np is the population size. The reduction monotony of the parameter p creates the required balance between exploration and exploitation.

- ***p-ADE Algorithm***

- *Advanced p-ADE Mutation scheme*

A new mutation strategy called DE/rand-to-best/pbest/bin is used; which is, essentially, based on utilizing the best global solution and the best previous solution of each individual that are involved in the differential process, thus bringing in more effective guidance information to generate new individuals for the next generation.

The detailed operation is as follows,

$$v_i^t = W_i^t \cdot x_{r1}^t + K_i^t \cdot (x_{best}^t - x_i^t) + F_i^t \cdot (x_{pbesti}^t - x_i^t) \quad (3.38)$$

where x_{best}^t denotes the best individual in the current generation t . x_{r1}^t is a random

generated individual where $r1 \in [1, Np]$ and $r1 \neq i$. x_{pbesti}^t denotes the best i^{th} 's previous individual picked up from the previous generation. The mutation's control parameters W_i^t, K_i^t , and F_i^t of the i^{th} individual are updated using a dynamic adaptive manner. The most remarkable merit of this mutation technique is the inclusion of three different working parts at the same time:

Inertial Part (Inheriting part) represented by $W_i^t \cdot x_{r1}^t$ where the current individual, v_i^t , inherits traits from another individual at generation t .

Social Part (Learning Part) represented by $K_i^t \cdot (x_{best}^t - x_i^t)$ where the current individual, v_i^t , gains information from the superior individual in the current generation t .

Cognitive Part (Private Thinking) represented by $F_i^t \cdot (x_{pbesti}^t - x_i^t)$ where the current individual, v_i^t , reinforces its own perception through the evolution process.

The high values of both the inertial and the cognitive part play a key role in intensifying the exploration searching space, thus improving its ability for finding the global solution. While the large values of the social part promotes connections among individuals, thus resulting to speed up the convergence rate. From the previous description of the main mechanism of p -ADE mutation scheme and the PSO standard perturbation scheme (Kennedy & Eberhart, 1995; Xin, Chen, Zhang, Fang, & Peng, 2012), we can observe that they are closely related in origin, in particular, for the case where the mutation (see Equation 3.38) is divided into three learning parts in the same manner applied by PSO algorithm. In p -ADE there is an additional mechanism which is called *classification mechanism*. This *classification mechanism* is coupled with the mutation scheme to be implemented on the whole population at each generation. Accordingly, the new mechanism divides the population's individuals into three classes:

Superior individuals: The first individuals' category where the fitness values of these individuals fall in the range $f_i - f_{mean} < -E(f^2)$, where f_{mean} is the mean fitness values and $E(f^2)$ is the second moment of the fitness values of all individuals in the current generation. In this case, the exploration ability of the search process is enhanced by further intensifying the inertial and cognitive parts in order to increase the likelihood of the excellent individual to find the global solution in its neighborhood area. So, the corresponding individual is generated as follows,

$$v_i^t = W_i^t \cdot x_{r1}^t + F_i^t \cdot (x_{pbesti}^t - x_i^t) \quad (3.39)$$

Inferior individuals: The second individuals' category where the fitness values of these individuals fall in the range $f_i - f_{mean} > E(f^2)$. The individual in this case has poor traits since its place in the search space is far away from the global optimum. Therefore, the exploration search ability is also intensified for rapid convergence rate. So, the corresponding individual is generated as follows,

$$v_i^t = W_i^t \cdot x_{r1}^t + K_i^t \cdot (x_{best}^t - x_i^t) \quad (3.40)$$

Medium Individuals: The third individuals' category where the fitness values of these individuals fall in the range $-E(f^2) < f_i - f_{mean} < E(f^2)$. The individuals in this category are not superior nor are they inferior; therefore, the complete perturbation scheme (see Equation 3.38) should be implemented entirely for further enhancing both the exploitation and exploration abilities.

- *p-ADE Parameter Control Schemes*

p-ADE comprises four control parameters involved in the search process, including three mutation scheme parameters (W , F and K) and crossover rate CR . A dynamic

adaptive scheme has been proposed to commonly update the four parameters through the run as follows,

$$W_i^t = W_{min} + (W_{max} - W_{min}) \times \left((2 - \exp\left(\frac{t}{Gen} \times \ln(2)\right)) \times \frac{1}{2} + \frac{f_i^t - f_{min}^t}{f_{max}^t - f_{min}^t} \times \frac{1}{2} \right) \quad (3.41)$$

$$K_i^t = K_{min} + (K_{max} - K_{min}) \times \left((\exp\left(\frac{t}{Gen} \times \ln(2)\right) - 1) \times \frac{1}{2} + \frac{f_i^t - f_{min}^t}{f_{max}^t - f_{min}^t} \times \frac{1}{2} \right) \quad (3.42)$$

$$F_i^t = F_{min} + (F_{max} - F_{min}) \times \left((2 - \exp\left(\frac{t}{Gen} \times \ln(2)\right)) \times \frac{1}{2} + \frac{f_{max}^t - f_i^t}{f_{max}^t - f_{min}^t} \times \frac{1}{2} \right) \quad (3.43)$$

$$CR_i^t = CR_{min} + (CR_{max} - CR_{min}) \times \left((2 - \exp\left(\frac{t}{Gen} \times \ln(2)\right)) \times \frac{1}{2} + \frac{f_i^t - f_{min}^t}{f_{max}^t - f_{min}^t} \times \frac{1}{2} \right) \quad (3.44)$$

As can be seen from the above equations, the main adaptive scheme is equally captive to the influence of the number of generations achieved, as well as the fitness values. Technically, the value of each control parameter varies within its specified range as, $W \in [0.1, 0.9]$, $K \in [0.3, 0.9]$, $F \in [0.3, 0.9]$ and $CR \in [0.1, 0.9]$ during the run of the algorithm. Throughout the evolution process, the values of these parameters will gradually decrease; thereby transiting the search from exploration to exploitation.

3.4.2 DE with Adaptive Parameters and Multiple Strategies

In this subsection the main characteristics and mechanisms of four remarkable adaptive DE versions are stated in details, on the basis of parameter adaptive schemes

and multiple adaptive DE strategies.

3.4.2.1 Adaptive DE with Multiple Standard Strategies

- **SaDE Algorithm**

- *SaDE Strategies Adaptive*

The main feature of SaDE is to automatically adapt multiple standard DE mutation strategies (DE/rand/1/bin; DE/rand-to-best/2/bin; DE/rand/2/bin; and DE/current-to-rand/1 with no crossover) and update the corresponding control parameters during the evolution process using parameter, $p_k(k = 1,2,3,4)$.

Determine the probability of applying each candidate strategy to the current population (p_k): Initially, the probabilities of applying each scheme, p_k^t , is set to $1/K$ so as to assign an equally likely probability for all strategies. Then, the probability of applying each strategy is then updated every 50 generations in the following manner,

$$p_k^t = \frac{S_k^T}{\sum_{k=1}^K S_k^T} \quad (3.45)$$

where

$$S_k^t = \frac{\sum_{t=T-LP}^{T-1} ns_k^t}{\sum_{t=T-LP}^{T-1} ns_k^t + \sum_{t=T-LP}^{T-1} nf_k^t} + \varepsilon, \quad \text{for } k = 1,2, \dots, K; T > LP$$

where K is the number of strategies available for perturbation. LP is the period assigned for learning in which the learning process is activated only when $T > LP$; in the current study it has been set to 50 generations. ns_k^t (*Success Memory*) and nf_k^t (*Failure Memory*) are both memories generated by the k^{th} strategy and used to record the number of trial vectors that have been succeeded or failed to enter the search process respectively. $\varepsilon = 0.001$ this small value is added to avoid the possibility of the null rate of success. Once these memories' sizes reach to a certain threshold, i.e. after LP

iterations, all previous records will be eliminated from these memories, i.e. ns^{T-LP} and nf^{T-LP} in order to allow those vectors that are generated in the current iteration to be stored. Finally, S_k^T is divided by $\sum_{k=1}^K S_k^T$ to guarantee that the resultant p_k^t is always summed to 1.

○ *SaDE Parameters Control Schemes*

Set the mutation factor F_i values to be independently generated at each generation according to Gaussian distribution with mean 0.5 and standard deviation 0.3 as follows,

$$F_i = randn(0.5,0.3) \quad (3.46)$$

Accordingly, both the local (with small F_i values) and global (with large F_i values) search ability will be kept throughout the evolutionary process, hence to generate, good mutant vectors.

Crossover Probability (CR_i) and the Mean Crossover Probability Distribution (CR_m):

The strategy of controlling the crossover probability CR is an adaptive learning based. It starts with independently generating crossover probabilities CR_i under Gaussian distribution with mean CR_m and standard deviation 0.1 as follows,

$$CR_i = randn(CR_m, 0.1) \quad (3.47)$$

The CR_i values will remain fixed for 5 generations before the next generation has launched. Throughout these generations CR_i values that are associated with successful trial vectors are recorded. While the value of the median, CR_m , is first initialized to 0.5, then updated every 25 generations based on the successful CR_i values,

$$CR_m = \frac{1}{K} \sum_{k=1}^K CR_{suc}(k), \quad (3.48)$$

where K denotes the number of successful CR_i values accumulated over 25 generations and CR_{suc} is the k^{th} CR successful value.

- ***EPSDE Algorithm***

- *EPSDE Parameters Control Schemes and Strategies*

EPSDE is unlike other adaptive DE variants, it is an ensemble of mutation strategies and parameter values of DE. EPSDE does not involve certain equation to modify the values of the control parameters, but rather it assigns for each member of the initial population a mutation strategy randomly selected from a pool of mutation strategies with diverse characteristics, and randomly takes values for the associated parameter from a pool of values. Throughout evolution, the population members that produce individuals better than the target vectors, their mutation strategies and associated parameter values retained for the next generation, while those fail to produce better individuals are reinitialized with a new mutation strategy and associated parameter values from the respective pools or from the successful combinations stored with equal probability. In EPSDE, there are two pools:

Pool of mutation strategies: this pool includes the DE strategies that are involved in the evolution. These strategies have been selected with diverse characteristics:

1. Strategies rely on the best individual in the current population, DE/best/2/bin.
2. Strategies which bear stronger exploration capabilities, DE/rand/1/bin.
3. Strategies being rotational invariant without crossover DE/current-to-rand/1.

Pool of parameter control values: in this pool the three crucial parameter values (F and CR) are set to different ranges. The pool of CR values is taken in the range 0.1 – 0.9 in steps of 0.1. The pool of F values is taken in the range 0.4 – 0.9 in steps of 0.1.

3.4.2.2 Adaptive DE with Multiple Advanced Strategies

- ***SaDE-MMTS Algorithm***

SaDE-MMTS has been proposed to enhance the performance of the standard SaDE algorithm; by *incorporating* SaDE with the JADE mutation strategy (JADEw) and *integrating* it with the modified multi-trajectory algorithm (MMTS), in order to solve problems with complex characteristics and high dimensionality. This integration can be encapsulated into three main parts: *SaDE-MMTS = JADE mutation scheme + SaDE algorithm + MMTS method* (Local Search), as follows:

- *SaDE-MMTS Advanced Adaptive DE Strategies*

JADE mutation strategy with external archive (JADEw) as in Equation 3.23 is adopted and engaged with three crossover operators (binomial and exponential), and no crossover option as well, to generate the trail vectors for the new population. Hence, the expected number of perturbation strategies is three and they are applied according to the strategy probability, as in the SaDE algorithm. The selection of the mutation strategy is according to the probability, p_k^t , of applying each JADE with archive strategy in the current population (see Equation 3.45).

- *SaDE Parameters Control Schemes*

The control parameters F and CR are updated through the evolution process in the same manner used in SaDE (see Equations 3.46-3.47).

- *MMTS method*

The original MTS (Tseng & Chen, 2007, 2008) algorithm is first proposed to solve large scale global optimization problems. The main underlying idea of this algorithm is

the employment of randomly selected search combinations (i.e. agents) uniformly distributed over the whole search space to seek out for better solutions. Each combination applies one of three candidate local search methods that is better fit the search space characteristics of a solution's neighborhoods; these combinations are generated using the basic orthogonal array $OA_{n \times k}$ where n is the number of testing experiments and k is the number of factors in each experiment.

- **SaM**

SaM is a strategy adaptation mechanism that can be integrated with any DE variant to make it strategy adaptive. SaM creates a pool of strategies and selects the candidate strategy to be applied on the running individual X_i from this pool according to Equation 3.49,

$$S_i = \lfloor \eta_i \times K \rfloor + 1 \quad (3.49)$$

where $\eta_i \in [0,1)$ is a strategy parameter control variable. K is the total number of strategies in the pool and $S_i = 1, 2, \dots, K$ the selected DE strategy. For example, suppose $K = 4$ and at a certain generation $\eta_i \in [0,0.25)$, then based on the calculation of Equation 3.49 the value of S_i is 1.

SaM has suggested three approaches to update the value of η_i during evolution. In this study, the first approach has been considered which is inspired by the parameter adaptation equation of JADE. For each individual X_i at generation t , a new value for η_i is generated as,

$$\eta_i = \begin{cases} randn_i(\mu_s, \frac{1}{6}), & t = 1 \\ randn_i(\mu_s, 0.1), & otherwise \end{cases} \quad (3.50)$$

where $randn_i(\mu_s, 0.1)$ indicates a normal random distribution of mean μ_s and standard

deviation 0.1. The mean μ_s is initialized to be 0.5 and then updated at the end of each generation in the same adaptation equation used in JADE (see Equation 3.29) as follows:

$$\mu_s = (1 - c) \times \mu_s + c \times \text{mean}_A(H_s) \quad (3.51)$$

where H_s denotes as the set of all successful DE strategy parameters η_i 's at generation t . SaM mechanism has been applied on JADE strategies to create a new approach called SaJADE. SaJADE employed a pool of different JADE strategies: (1) DE/current-to- p Best/1/bin with no archive; (2) DE/current-to- p Best/1/bin with archive; (3) DE/rand-to- p Best/1/bin with no archive; (4) DE/rand-to- p Best/1/bin with archive. The parameter adaptive equations used to update the values of F_i and CR_i are also JADE schemes as in Equations 3.25-3.29. In SaJADE, assigning the parameter adaptive scheme to update F_i and CR_i is fixed and determined for each DE strategy. For updating the value of CR_i , all the strategies use Equation 3.28. The difference is in updating the value of F_i in which the strategies 1 and 3 use Equation 3.25; whereas strategies 2 and 4 use a modified version of Equation 3.25 which is normal distribution as follows,

$$F_i = \text{randn}_i(\mu_F, 0.1) \quad (3.52)$$

3.4.3 Adaptive DE Comparisons

Ten adaptive DE algorithms have been presented in the previous subsections. These algorithms clearly exhibit diversity in terms of characteristic, structure, complexity, and algorithmic logic. Beside their advantages, these algorithms also show some shortcomings in particular cases. In this subsection, comparisons have been established among these methods based on their conceptual similarities and differences, and pros

and cons of each algorithm.

3.4.3.1 Adaptive DE Conceptual Similarities and Differences

In this subsection, we discuss how the aforementioned methods relate and differ from each other and from the standard DE algorithm based on mutation strategy and parameter control schemes used in each algorithm.

- ***Comparison Based DE Mutation Scheme***

The ten algorithms presented under this comparison differ in their mutation strategies, as can be seen in Table 3.2, which gives a summary of the mutation schemes employed in each algorithm. All the algorithms adopt the classical crossover (*bin*) and selection operations in their main work, except DESAP which uses a modified crossover scheme similar to that of DE/rand/1 mutation scheme and MDE_ *p* BX which uses a new modified crossover scheme called *p*-Best; while both SaDE, EPSDE and SaJADE invent new adaptive schemes that can make a selection among a pool of candidate mutation schemes. SaDE-MMTS uses three crossover aspects (binomial, exponential, and no crossover). SaDE-MMTS creates a pool of DE strategies to automatically adapt one of them.

- ***Comparison Based DE Parameter Control Schemes***

The comparison in this subsection is based on the parameter setting taxonomy explained in Section 3.2. Table 3.3, elucidates the main features of the adaptation scheme being used to control the main four parameters (F , CR , Np and t) in the eleven adaptive DE variants. This table shows how the concept of adaptive has revealed by applying an adaptive rule to at least one of the algorithm component. From the same

table we can see that none of the algorithms has adapted the number of generations' parameter, t and none of them has also considered adapting the population size parameter, Np except DESAP algorithm. The main focus was only on F and CR .

3.4.3.2 Adaptive DE Strengthens and Drawbacks

In this subsection, conceptual strengthens and drawbacks of each of the ten algorithms have been discussed in terms of modifying the main DE strategies, and the additional components that have been added to the original DE algorithm and function as adaptive features. It is clear that all the modifications and integrations proposed tend to include extra moves to the original DE, as well as create the proper balancing between the exploitation and exploration characteristics. However, there are also some drawbacks need to be considered.

- ***Comparison Based DE Mutation Strategy***

In this subsection the ten DE algorithms have been compared based on the mutation scheme used in each algorithm. Table 3.4 and Table 3.5 illustrate the algorithms comparison.

- ***Comparison Based DE Parameter Control Schemes***

Table 3.6 and Table 3.7, elucidate the points of strengthens and drawbacks of the adaptation scheme being used to control the main four parameters (F , CR , Np and t) in the ten DE variants. From the two tables we can see that there are some common pros and cons among certain algorithms. This is due to the fact that they use the same adaptation scheme.

Table 3.2: A summary of the type of mutation strategies used in the ten adaptive DE algorithms

Algorithm	Mutation Scheme	Main Feature of the Scheme	Base Scheme
FADE	DE/rand/1	Fixed	DE/rand/1
JADE wo	DE/current-to- p best/1	Utilization of the $100p\%$ best solutions	DE/current-to-best/1
JADE w archive	DE/current-to- p best/1 w archive	Utilization of the $100p\%$ best solutions and the A inferior solutions	DE/current-to- p best/1
MDE- p BX	DE/current-to- gr_{best} /1	Utilization of the best individual selected from $q\%$ group of individuals	DE/current-to- p best/1
SaDE	Pool of standard DE strategies	Adaptive PCB among the schemes	DE/rand/1; DE/rand/2; DE/current-to-rand/1; DE/rand-to-best/1
jDE	DE/rand/1	Fixed	DE/rand/1
DESAP	(non-DE standard scheme)	With deterministic mutation probability updates	Simple Perturbation Scheme
p -ADE	DE/rand-to-best/ p best	Adaptive Dynamic Structure	PSO Standard Scheme
SaJADE	Pool of advanced DE strategies	Adaptive LPB among the schemes	DE/current-to- p best/1 wo; DE/rand-to- p best/1 wo; DE/current-to- p best/1 w; DE/rand-to- p best/1 w;
SaDE-MMTS	Pool of DE/current-to- p best/1 w merged with two crossover schemes and no crossover	Adaptive PCB among the schemes	DE/current-to- p best/1 w
EPSDE	Pool of standard DE strategies	Deterministic/ Partial-predetermined	DE/best/1; DE/best/2; DE/rand-to-best/1; DE/rand-to-best/2; DE/rand/1; DE/rand/2; DE/current-to-rand/1

Note: **LPB:** Learning Process Based **PCB:** Progressive-Controlled Based **EPB:** Evolution Process Based

Table 3.3: This table encompasses taxonomy of the adaptation scheme used to update the main control parameters in the ten adaptive DE algorithms

Algorithm Name	Parameter control strategies based taxonomy							
	<i>Np</i>		<i>F</i>		<i>CR</i>		<i>t</i>	
	Type of Setting	Type of Evidence	Type of Setting	Type of Evidence	Type of Setting	Type of Evidence	Type of Setting	Type of Evidence
FADE	Tuned	×	Adaptive/PCB	Absolute	Adaptive /PCB	Absolute	Tuned	×
JADE wo	Tuned	×	Adaptive/ LPB	Relative	Adaptive/ LPB	Relative	Tuned	×
JADE w	Tuned	×	Adaptive/ LPB	Relative	Adaptive/ LPB	Relative	Tuned	×
MDE_ <i>p</i> BX	Tuned	×	Adaptive/ LPB	Relative	Adaptive/ LPB	Relative	Tuned	×
SaDE	Tuned	×	Deterministic/ Partial-predetermined	Absolute	Adaptive/ LPB	Relative	Tuned	×
jDE	Tuned	×	Self-adaptive/EPB	Relative	Self-adaptive/ EPB	Relative	Tuned	×
DESAP	Self-adaptive/ EPB	Relative	Self-adaptive/ EPB	Relative	Self-adaptive/ EPB	Relative	Tuned	×
<i>p</i> -ADE	Tuned	×	Adaptive/ LPB	Relative	Adaptive/ LPB	Relative	Tuned	×
SaDE-MMTS	Tuned	×	Deterministic/ Partial-predetermined	Absolute	Adaptive/ LPB	Relative	Tuned	×
EPSDE	Tuned	×	Deterministic/ Partial-predetermined	Absolute	Deterministic/ Partial- predetermined	Absolute	Tuned	×
SaJADE	Tuned	×	Adaptive/ LPB	Relative	Adaptive/ LPB	Relative	Tuned	×

Note: **LPB:** Learning Process Based **PCB:** Progressive-Controlled Based **EPB:** Evolution Process Based

Table 3.4: DE algorithms points of strengthens based on mutation strategy

Algorithm	Strengthens of the DE Mutation
JADE	Alleviate the problem of premature convergence because of its ability to intensify the population diversity.
JADE with archive	<ul style="list-style-type: none"> • Increase the population diversity as such the problem stem from the convergence rate was further reduced. This is so because, the superior and inferior solutions are both incorporated into the mutation strategy. • No significant computational overhead as the archive operation has been made very simple.
<i>p</i> -ADE	The ability to be dynamically changed to three different schemes according to the classification conditions upon the individuals' quality that are located in the same population.
MDE_pBX	It weakens the tendency of premature convergence and alleviates the attraction of any fixed point in the fitness landscape. This small modification has led to reduce the greediness feature of the DE mutation scheme towards choosing the superior solutions for perturbation and making it converges fast to a local point.
DESAP	No significant improvements over the standard DE.
SaDE; SaDE-MMTS; SaJADE	A learning strategy has been applied to gradually evolve the selection of one mutation scheme from a pool of mutation schemes in hand throughout generations. This characteristic allows further improvements in the DE moves, thus increasing the exploitation features of the algorithm. This learning strategy affords flexibility to be extended to include more candidate mutation schemes in an attempt to solve complex optimization problems.
EPSDE	The selection of the mutation strategies is made randomly from a pool of mutations with different characteristics to avoid the influence of less effective mutation strategies.
FADE; jDE	The standard DE/rand/1 scheme is always considered as the fastest non greedy scheme with good convergence performance.

Table 3.5: DE algorithms drawbacks based on mutation strategy

Algorithm	Drawbacks of the DE mutation
JADE; JADE with archive; SaDE-MMTS; SaJADE	<p style="text-align: center;">Stagnation</p> <p>The population selective factor, p, is tuned before the run and kept fixed during the evolution process. This strategy might lead to stagnation problem especially after several epochs of evolution when the population diversity rate is very low and the superior solutions in the $p\%$ of the current population start to be close in values if not exactly same.</p>
p -ADE	<p style="text-align: center;">Local optimum caused by greedy mutation</p> <p>Selecting the best solutions from the previous and current population may lead the new strategy become greedier toward good solutions, thus running the risk of falling into local optimum.</p>
MDE_ p BX	<p style="text-align: center;">Lack of strategy and parameter analysis & Crossover greediness tendency</p> <ul style="list-style-type: none"> • The influence of the new mutation scheme on the diversity of population and convergence rate is not investigated. • The greediness of the new crossover scheme towards superior solutions and the associated parameter, p of the top-ranking vectors, is fixed during the run. This may lead to premature convergence problem.
DESAP	<p style="text-align: center;">Lack of Exploitation and Exploration Ability</p> <p>The new crossover and mutation operations are simple and straightforward schemes, they did not bring about the desired performance and DESAP outperform the standard DE in only one out of five test problems.</p>
FADE; jDE;	<p style="text-align: center;">Lack of Population diversity</p> <p>This problem will arise in optimizing high dimensional functions and also when the characteristic of the test problem is challenging.</p>

Table 3.6: DE algorithms points of strengthens based on parameter control schemes

Algorithm	Strengthens of Parameter Control Schemes
JADE; JADE with archive; SaJADE	<ul style="list-style-type: none"> • Adjusting the values of F and CR in an adaptive characteristic based on a learning strategy that gains knowledge from previous iterations and cumulates it into two learning parameters, μ_F and μ_{CR} to be retained and used in the current population. • Create the proper balance in maintaining the pressure on the population to move towards exploring more optimal solutions, as well as not to lose the exploitation features.
p -ADE	<ul style="list-style-type: none"> • Possess a unique merit over other adaptive DE algorithms by involving the number of iterations passed over and the fitness value in the updating process; for the sake of accelerating the convergence rate, • Creating the required balance between the exploitation and exploration features. This has been achieved through selecting the best values for the control parameters W, K, F, and CR.
MDE_ p BX	<p>Modifies the original JADE scheme of adapting the values of F and CR. The new modifications to the control parameters schemes with the combination of the new mutation and crossover schemes in MDE_pBX have greatly increased the ability of exploitation and exploration, and the search process is directed to explore better search space regions, hence, escaping from the possibility of getting trapped in suboptimal solutions.</p>
DESAP	<ul style="list-style-type: none"> • The first attempt to demonstrate the possibility to produce an adaptive algorithm that not only updates the crossover and mutation rates but also the population size parameter as well. • Downsize DE parameters' setting by updating the population size and other control parameters in an adaptive manner.
FADE	<ul style="list-style-type: none"> • The first attempt in using Fuzzy Control in DE parameter settings for the sake of reducing the user load from parameter tuning. • Possess robustness and fuzziness with problems in an imprecise environment
SaDE ; SaDE-MMTS	<p>Utilizes an adaptive learning strategy to adjust the crossover rate and mutation rate during the search process.</p>
jDE	<p>Adjust both F and CR in a self-adaptive manner with few additional parameters.</p>
EPSDE	<p>The selection of the mutation and crossover parameter values is made randomly from a pool of values to avoid the influence of less effective parameter settings.</p>

Table 3.7: DE algorithms drawbacks based on parameter control schemes

Algorithm	Drawbacks of Parameter Control Schemes
JADE; JADE with archive; SaJADE	<p style="text-align: center;">Problem-dependent parameter selection</p> <p>The parameters c and p determine the adaptation rate of μ_{CR} and μ_F and the greediness of the mutation strategy are kept fixed during the run. These two parameters have the ability to affect the overall performance of JADE mutation.</p>
p -ADE	<p style="text-align: center;">Time Consumption</p> <p>All of the four parameters (W, K, F, and CR) should be adjusted through the run, simultaneously with adjusting the mutation scheme. This may require increasing the number of iterations needed to achieve the optimal solution.</p>
MDE_pBX	<p style="text-align: center;">Lack of theoretical guidelines</p> <p>There are two additional control parameters q (the group size in the mutation operation) and p (the number of the top-ranking vectors in the crossover operation). These parameters bring about the effect to the performance of the mutation and crossover, since both parameters were set to fixed values.</p>
DESAP	<p style="text-align: center;">Lack of Population diversity</p> <ul style="list-style-type: none"> • It outperforms the standard DE over five test functions only. • It was found that both DESAP's versions yielded highly similar results in terms of the best solution obtained; although, in providing more stability DESAP with absolute encoding was more favorable than DESAP with relative encoding.
FADE	<p style="text-align: center;">Lack of Algorithm Performance Analysis</p> <p>It does not involve any relative consideration for the individual fitness value only absolute based on the knowledge gained from the fuzzy controller.</p>
SaDE; SaDE-MMTS	<p style="text-align: center;">Deterministic evidence rule for updating the mutation factor</p> <ul style="list-style-type: none"> • The mutation factor, F, is updated through a deterministic rule based, though, it has been set to different random values through the evolution process. • It introduces additional learning parameters such as ns and nf to steer both learning strategies, thus making the algorithm cumbersome with too many parameters.
jDE	<p style="text-align: center;">Lack of balance between the exploitation and exploration</p> <p>There is a weakness in the relative consideration of the individual fitness value since the values of F and CR are selected according to the best fitness picked up from the current generation only, then updated according to a uniform distribution. This may lead the jDE's moves to be biased towards exploration.</p>

Table 3.7- Continued

Algorithm	Drawbacks of Parameter Control Schemes
EPSDE	<p style="text-align: center;">Local Optima</p> <p>The procedure of adjusting the control parameter values is mostly implemented in a random way. There is no accumulating knowledge through the evolution and the parameter value that produce inferior solution will be reinitialized at the same generation. This procedure in conjunction with the random selection of the mutation scheme will in some cases divert the population to a wrong direction of the search space and fall in local optima if the algorithm fails to select the appropriate parameter values and scheme.</p>

3.4.3.3 Discussion and Conclusion

From this extensive review, we found two main points which are:

1) Review and analysis study on adaptive DE

Although few significant review studies on DE have been written, no extensive review or analysis study on adaptive DE has been published to date only the study of Chiang, Chen, and Lin (2013). Das and Suganthan (2011) published a comprehensive survey article that addresses general issues concerning DE research and the survey of Neri and Tirronen (2010) who presented a number of well-known DE versions of different characteristics in a classification format.

For this reason, this chapter (Chapter 3) has been devoted to present a thorough review and analysis concerning the parameter settings of EAs and provide a new parameter settings' classification. In this review, DE has been selected as an EA example to apply this classification on its adaptive versions. Then a new taxonomy on adaptive DE algorithms is also presented.

2) No DE algorithm applies the selection of different parameters adaptive schemes

Although many DEs with different adaptive characteristics have been discovered

through this review, no algorithm has yet been discovered that applies the selection of different parameter adaptive schemes through optimization besides adapting the DE strategies. For instance, the adaptive DE algorithms (jDE; SaDE; DESAP; JADE; SaDE-MMTS; MDE_pBX) have applied two parameter control strategies; one to update F value and another strategy to update CR value, except FADE, p -ADE which uses the same strategy to update F and CR . The adaptive DE algorithms (EPSDE; CoDE; HSPEADE) apply no adaptive schemes but rather the control parameters as well as the DE strategies have been selected as combinations from either the pool of successful combinations or from the initial pool based on the situation. The only adaptive DE algorithm that suggests using different parameter adaptive schemes is SaM (SaJADE). In SaJADE, each DE strategy has been assigned with certain selected parameter adaptive schemes as it may perform the best with this strategy, however, this selection has been predetermined before the run. Moreover, the integration with local search methods is limited to SaDE-MMTS as it can be observed from the review.

Figure 3.5 depicts an estimated rank of the aforementioned algorithms based on the experimental results (Mean \pm STD) presented in their corresponding articles. The distinct layers in the figure refers that the DE algorithms in the same layer have almost equal performance. The sold arrow indicates that the algorithm in the source node has inherited the mutation strategy from the destination node. The intermittent line refers to the DE algorithms with less performance than their corresponding DE algorithm in the source node. The figure shows that, generally, the adaptive DE algorithms with multiple DE strategies such as ESPEDE and with single advanced DE strategy such as MDE_pBX always have the best performance over other algorithms. Then the superior performance is for the adaptive DE algorithms that employ both multiple and advanced DE strategies, such as HSPEADE and SaJADE.

Moreover, it has also been found that the use of different parameter adaptive schemes such as in SaJADE has no less effect as applying multiple DE strategies to improve the search process of DE. In the case of multimodal problems with high dimensionality, the algorithm may require different steps size during the search process to escape from the local optima and speed up the convergence rate. It has also been found that applying different parameter adaptive schemes induces the population diversity.

In addition, it has already been proved by the No Free Launch Theorem that the quest for good EA is lost a priori and the integration of the different characteristics of two or more algorithms will serve the search process.

For this reason, a new version of adaptive DE called ARDE-SPX is proposed in (Chapter 4). In this algorithm, a repository of different advanced DE strategies and parameter adaptive schemes for F and CR has been created. The selection among the combinations of the DE strategies and the adaptive parameters schemes from the repository is implemented based on the fitness values of their corresponding trial vectors. Moreover, the ARDE has been integrated with the local search method (SPX crossover) to further improve its performance.

3.5 Summary

This chapter is designed from the ground up to support the issue of EAs parameter control values represented by,

- 1) New extended EA parameter setting taxonomy has been proposed to eliminate any confusion related to identify the type of the scheme used to control the algorithm's parameters. This problem has been overlooked by many previous related studies. For example, in literature many algorithms have been defined as "self-adaptive" algorithms, although their own parameters do not undergo mutation and crossover during the evolution process, thus their schemes should be considered under the definition of "adaptive" algorithms. The new extended EA parameter setting taxonomy has been applied to multiple adaptive DE algorithms in specific, as an example to convey the main purpose of this taxonomy.
- 2) A comprehensive procedural analysis study has been established on these algorithms to elucidate the conceptual similarities and differences among them, the pros and cons of the adaptive schemes, as well as proposed solutions. In addition, the study has been extended to involve the DE mutation schemes employed by each method.

Finally, based on the review study presented in this chapter and to tackle some of the limitations in other adaptive DE algorithms, Chapter 4 presents a new adaptive DE algorithm that automatically tunes the mutation and crossover strategies, and the parameters control schemes of F and CR using a simple and efficient mechanism.

CHAPTER 4

DIFFERENTIAL EVOLUTION WITH ADAPTIVE REPOSITORY OF STRATEGIES AND PARAMETER CONTROL SCHEMES INTEGRATED WITH LOCAL SEARCH METHOD

4.1 Introduction

Throughout literature, the adaptive (or self-adaptive) parameters of DEs have shown tremendous successful performance in solving different types of optimization problems and overcome the problem of the tedious and time-consuming manual (or tuning) settings. Moreover, it has also been proved that the use of multiple DE strategies besides the adaptive parameters of the algorithm will make DE algorithm more efficient.

In this chapter, a new adaptive DE algorithm named ARDE-SPX is presented. In ARDE, the DE strategies as well as the parameters adaptation schemes of the mutation factor F and crossover rate CR are evolved through the run using a new adaptive mechanism. The ARDE has also been integrated with a local search technique called SPX-crossover to further increase its performance. Additionally, and in order not to overlook the general rudiments of how to build an adaptive EA, the first section of this chapter has been devoted for this purpose.

4.2 General Steps to an Adaptive EA

Attention must be paid to the general steps that are required to integrate any proposed EA into the adaptation of parameter control concept; these steps are listed below. However, we should consider the different aspects of EAs in both informal and formal languages before going through these steps. This constitutes the various components of an EA, including selection, recombination, mutation, and survival

operators. The steps are as follows:

Step 1 (Individual Encoding and Population Representation): A population is a set or array of individuals or chromosomes, where pop_{size} refers to the number of individuals in each generation. We have to encode the information required for the problem analysis in the chromosome structure. Each chromosome should represent a complete solution to the problem at hand. Altogether, these parameters are called solution parameters. Additionally, if we intend to work on self-adaptive algorithm, the chromosome should implicitly include the encoding of the control parameter(s) (i.e., strategy parameter) to undergo evolution via recombination and mutation. Including this parameter requires considering an intelligent decision in such a way that better parameter values tend to produce better individuals (i.e., solutions) with the highest chance to survive and propagate for more off-springs.

Step 2 (Chromosome Evaluation): The definition of the fitness function is crucially important for a successful application. The fitness of each chromosome must be evaluated in any evolutionary algorithm. Many standard fitness functions can be used to test any proposed EA method. Stating the fitness function as minimization with or without constraints is more natural than that as maximization of some utility objectives throughout many problems; however, the result depends on the type of problem or application at hand.

Step 3 (Chromosome Operations): An important step in using EA is changing the candidate solutions to diversify the population with new solutions. Exploration and exploitation are the two cornerstones of problem solving by search. Proper balance must be achieved between exploration (i.e., to cover sufficiently the solution space seeking out for good solutions) and exploitation (i.e., refining the solutions by combining information gathered from good ones during the exploration phase) to achieve a successful evolution. Moreover, diversity maintenance is important to prevent

premature convergence and serves as a motivation to study and provide sufficient exploration and exploitation techniques to be incorporated into the main paradigm of EA. The main evolution operators that should be understood thoroughly and applied well are the following: selection, recombination, mutation, and survival operators.

Step 4 (Stopping Criteria): The most common stopping condition used in the literature is allowing the algorithm to run to a maximum number of iterations. A small number of iterations may not offer enough time for the algorithm to attain an optimum, especially when the search space is large. By contrast, nothing is gained from a very large number of iterations once the optimum solution is reached. In general, the product of the maximum number of iterations allowed and swarm size indicates the number of particles required to be evaluated by the EA algorithm.

Step 5 (Investigate the Influence of Different Parameters of EA): This step includes the following:

- Population size (pop_{size}). Draw the effect of pop_{size} on the population diversity and performance of the proposed EA, and determine whether increasing the size of the population may prevent or, at least, reduce the chance of the algorithm to be trapped in a local minimum.
- Selection and perturbation operators (recombination and mutation) with various qualitative and quantitative parameter settings. The type of parameter control settings to be adopted (i.e., deterministic, adaptive, and/or self-adaptive), change evidence, parameter(s), and method or approach to be applied should be determined in advance.
- Scope of change. The scope of change can be at the gene, individual, or whole population level and it must be determined.
- Termination criteria and number of EA generations

Step 6 (Experimental Results): This step is performed to reach twofold goals. First, experimental results should be evaluated with various test problems to show the accuracy and efficiency of the proposed methods. Second, the relative performance of the proposed methods should be evaluated both in qualitative and/or quantitative terms when compared with other well-known state-of-the-art algorithms and reported when applied to the same test problem or application. The influences of different versions, components, and parameters of EA on performance in all these experiments must be investigated, analyzed, and reported.

4.3 Adaptive Repository of DE Strategies and Parameters Control Schemes Integrated with SPX-Crossover (ARDE-SPX)

The idea of ARDE-SPX has been motivated by the prolonged experimental observations that different optimization problems require different mutation strategies with different parameter values depending on the problem characteristic (uni-model and multi-model). Basically, the proposed ARDE algorithm makes use of the JADE mutation strategies and the MDE_pBX parameters adaptive schemes of F and CR as frameworks. Then a new adaptive procedure has been developed to select the appropriate combinations of the JADE strategies and the parameter control schemes to generate the next generation. Moreover, the hill climbing simplex crossover (SPX) has been adopted as a local search (LS) engine. SPX is an interesting technique and according to the recorded results in (Noman & Iba, 2008) it shows a promising ability to improve the performance of the DE algorithm. Based on this, this work has been extended to integrate the SPX crossover operator with the adaptive ARDE algorithm in a new way of implementation in order to make the adaptive ARDE algorithm satisfying both the global and local search requirements. The detailed characteristics and steps of the ARDE-SPX are provided in this section.

4.3.1 ARDE-SPX: The Repository of DE Strategies

In DE literature there are many DE strategies with diverse characteristics in order to create the strategy repository. The candidate strategies should be restrictive and selected neatly in order to avoid the undesirable influence of less effective strategies. In this study, six DE strategies have been selected to be included in the strategy repository. These six DE strategies are gleaned from the JADE mutation variants (DE/current-to- p best/1 with archive and DE/rand-to- p best/1 with archive). These new strategies have been inspired by the standard DE mutation strategies that rely on the best solution found so far, such as DE/best/1. They have proved to be very powerful strategies and can significantly increase the convergence rate. That is so because the JADE strategies could solve the high greediness tendency of the DE/best/1 and provide the proper balance between exploitation and exploration capabilities. The candidate DE strategies that have been chosen to be included in the repository of DE strategies are described as follows:

- DE/current-to- p best/1 with archive, as described in Equation 3.23, and coupled with two basic crossover strategies (binomial and exponential) to form two strategies, i.e., DE/current-to- p best/1/bin with archive and DE/current-to- p best/1/exp with archive.
- DE/rand-to- p best/1 with archive, as described in Equation 3.24, and coupled with two basic crossover strategies (binomial and exponential) to form two strategies, i.e., DE/rand-to- p best/1/bin with archive and DE/rand-to- p best/1/exp with archive. These strategies have been proposed to solve large-scale problems and further increase the population diversity. That is so because they imitate the standard DE/rand/1 strategy which is already fast and robust, and it also bears strong exploration ability.

- DE/current-to-*p*best/1 with archive and DE/rand-to-*p*best/1 with archive, and no crossover are very effective strategies and they bear strong exploitation capabilities. These two strategies have been inspired by the DE/current-to-rand/1 with no crossover strategy which has been widely used in the literature of multiple DE strategies algorithms. This type of strategy is a rotational-invariant strategy. It has been proposed to solve multi-objective optimization problems and it has also proved effectiveness in solving rotation problems better than other strategies.

As can be noted above, the number of DE mutation strategies is only two (DE/current-to-*p*best/1 with archive and DE/rand-to-*p*best/1 with archive). Then this number has increased to six DE strategies after incorporating the three DE crossover strategies (bin, exp, and no crossover) with them. The pseudo-code of the two mutation strategies (DE/current-to-*p*best/1 with archive and DE/rand-to-*p*best/1 with archive) are provided in Algorithm 4.1 and Algorithm 4.2, respectively.

Algorithm 4.1: DE/current-to-*p*Best/1 with archive strategy

01: Input: $P(X)$ (Target Population), Np (Population size), A (Archive), D (individual dimension), F (Set of mutation factors)
 02: Output: $P(V)$ (Donor Population)
 03: Begin
04: Step 1: Select two mutually randomly different vectors $r1$ and i from $P(X)$. Select random vector $r2$ from $P(X) \cup A$. Select random vector $x_{best}^{p,t}$ as one of the 100*p*% best solutions in $P(X)$.
05: Step 2: $v_{i,j}^{t+1} = x_{i,j}^t + F_i \cdot (x_{best,j}^{p,t} - x_{i,j}^t) + F_i \cdot (x_{r1,j}^t - \tilde{x}_{r2,j}^t)$ ($i = 1, 2, \dots, Np$) ($j = 1, 2, \dots, D$)
 06: END

Algorithm 4.2: DE/rand-to-*p*Best/1 with archive strategy

01: Input: $P(X)$ (Target Population), Np (Population size), A (Archive), D (individual dimension), F (Set of mutation factors)
 02: Output: $P(V)$ (Donor Population)
 03: Begin
04: Step 1: Select three mutually randomly different vectors $r1$, $r2$ and i from $P(X)$. Select random vector $r3$ from $P(X) \cup A$. Select random vector $x_{best}^{p,t}$ as one of the 100*p*% best solution in $P(X)$.
05: Step 2: $v_{i,j}^{t+1} = x_{r1,j}^t + F_i \cdot (x_{best,j}^{p,t} - x_{r1,j}^t) + F_i \cdot (x_{r2,j}^t - \tilde{x}_{r3,j}^t)$ ($i = 1, 2, \dots, Np$) ($j = 1, 2, \dots, D$)
 06: END

The reason of using the two crossover operators (binomial and exponential) is so because the binomial crossover is efficient to solve separable problems when CR is low while it is also efficient to solve non-separable problems when CR is high; whereas, the exponential crossover has been selected because it is appropriate for solving linked problems.

4.3.2 ARDE-SPX: The Repository of Parameters Control Schemes

In conjunction with choosing the suitable DE strategy to be adopted, the proper selection of the schemes that update the values of the parameters F and CR is also important. The adaptation of F and CR values are based on the rule that better control parameter values tend to generate better individuals thus these control parameter values should be propagated to the next generations.

Basically, the adaptation of the control parameter values depends on two major issues (1) the way to accumulate the previous experience of the successful control parameter values throughout generations (2) the scheme of applying the adaptation steps to the control parameter values.

In this study, in order to measure the first point, the accumulation process has been inspired by the adaptation scheme of MDE- p BX which is also inspired by the adaptation scheme of JADE. In MDE- p BX the experience accumulation of the successful individuals' parameters F_i and CR_i are stored in the variables F_m and CR_m respectively, as described in details in Chapter 3 using Equations 3.31-3.36.

For the second point of how to update the values of F_i and CR_i is also inspired by JADE, where the F_m and CR_m are used to be the mean for the Cauchy and the Gaussian distribution respectively, to generate new values for the F_i and CR_i for each individual i at every generation. This process has also been described in details in Chapter 3 using Equations 3.25 and 3.28. In this work, the JADE adaptation schemes for F_i and CR_i

have been further extended to be generated using Gaussian and Cauchy for both F_i and CR_i , as follows,

$$F_i = randn(F_m, 0.1) \quad (4.1)$$

$$CR_i = randc(CR_m, 0.1) \quad (4.2)$$

So, the result is four adaptation schemes including the previous two schemes which are contained in the repository of the parameters control schemes.

The reason of using two distributions (Gaussian and Cauchy) is so because each distribution provides different adaptation steps based on its characteristics that can fit different stages of evolution for different problems scenarios. The Gaussian distribution (normal distribution) provides relatively small step size from the mean in comparison with the Cauchy distribution which has a far wider tail. This merit in Cauchy distribution makes it more likely to generate offspring further away from its parents and avoid the local optimum or escaping from a plateau when the population starts to be a basin of attraction of local optimum or plateau. On the other hand, the smaller hill around the mean in the Cauchy distribution makes it less effective in the exploitation of the local neighborhood area and thus has powerless ability in the small to mid-range search area. In Figure 4.1 the comparison between the Gaussian and the Cauchy distribution is depicted. From the same figure it can be observed the difference between the two distributions with respect to the distribution tail length and the hill height. This figure has been plotted using the probability density functions (PDF) of the Gaussian and Cauchy distribution respectively as follows,

$$\text{Gaussian distribution PDF} \quad Fn(x; \mu; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.3)$$

where μ denotes the mean (i.e. distribution location) which has been set to be 0, and σ is the standard deviation which has been set to be 1.

Cauchy distribution PDF
$$Fc(x; l; s) = \frac{1}{\pi} \arctan\left(\frac{x - l}{s}\right) + \frac{1}{2} \quad (4.4)$$

where l denotes the distribution location which has been set to be 0, and s is the scaling factor which has been set to be 1. The values of x have been uniformly randomly generated within the range $[-5, 5]$.

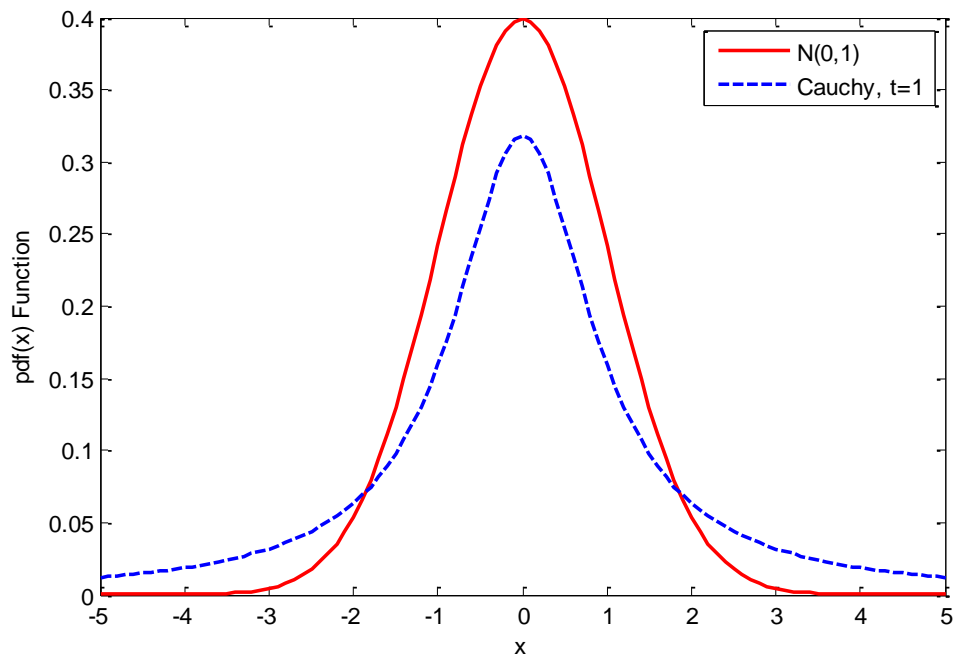


Figure 4.1: Comparison between Cauchy and Gaussian probability density functions

Accordingly, for F_i control parameter, the usage of the two distributions is when the mutation strategy consternates highly around a certain value (i.e. best individual) especially in DE/current-to- p best/1/bin with archive, DE/current-to- p best/1/exp with archive, and DE/current-to- p best/1 with archive then the Cauchy distribution is needed to diversify the mutation and prevent the premature convergence. Whereas when the

mutation strategies are already can provide mutation diversity such as DE/rand-to- p best/1/bin with archive, DE/rand-to- p best/1/exp with archive and DE/rand-to- p best/1 with archive then the use of the Gaussian distribution can be adopted to balance the high population diversity produced by the mutation strategy and the small value of F_i produced by the Gaussian equation. However, the use of Cauchy distribution at the early stage of all DE variants is very useful, as the higher values of F_i encourages the exploration ability of the population. On the other hand, at the late stage of the DE variants, the use of Gaussian distribution is useful, as it encourages the exploitation ability and stability of the population.

For CR_i control parameter, it is related to the diversity of population. The high values of CR_i generated by the Cauchy distribution can increase the population diversity especially at the early stage of DE. However, when applying Cauchy distribution using Equation 4.2, the long tail of Cauchy would generate higher values of CR_i , which these values would have to be truncated to 0 if the CR_i is less than 0, and truncated to 1 if the CR_i is greater than 1. On the other hand, the usage of the Gaussian distribution is better useful at some search stages, as it provides CR_i values within the range because of its short tail, moreover, the values of CR_i may become independent of the Cauchy distribution.

The pseudo-code of the parameter adaptation schemes for F and CR values are provided in Algorithms 4.3-4.6.

Algorithm 4.3: Update the value of F_m scheme

01: Input: F_m (mean value of previous successful F), S_F (set of successful mutation scalars), $n = 1.5$

02: Output: F'_m (updated mean value of F)

03: BEGIN

04: Step 1: $mean_{pow} = \sum_{i=1}^{|S_F|} power(\frac{power(F_i, n)}{|S_F|}, 1/n)$ where $F_i \in S_F$

05: Step 2: $w_F = 0.8 + 0.2 \times rand(0,1)$

06: Step 3: $F'_m = w_F \times F_m + (1 - w_F) \times mean_{pow}$

07: END

Algorithm 4.4: Update the value of CR_m scheme

01: Input: CR_m (mean value of previous successful CR), S_{CR} (set of successful crossover rates), $n = 1.5$
02: Output: \hat{CR}_m (updated mean value of CR)
03: BEGIN
04: **Step 1:** $mean_{pow} = \sum_{i=1}^{|S_{CR}|} power(\frac{power(CR_i n)}{|S_{CR}|}, 1/n)$ where $CR_i \in S_{CR}$
05: **Step 2:** $w_{CR} = 0.9 + 0.1 \times rand(0,1)$
06: **Step 3:** $\hat{CR}_m = w_{CR} \times CR_m + (1 - w_{CR}) \times mean_{pow}$
07: END

Algorithm 4.5: Generate F value scheme

01: Input: s (the cell's index in the repository), F_m (mean value of F)
02: Output: F (new F value)
03: BEGIN
04: IF (' F_n ' in $Cell_s$) THEN $F = randn(F_m, 0.1)$
 ELSE $F = randc(F_m, 0.1)$
05: END

Algorithm 4.6: Generate CR value scheme

01: Input: s (the cell's index in the repository), CR_m (mean value of CR)
02: Output: CR (new CR value)
03: BEGIN
04: IF (' CR_n ' in $Cell_s$) THEN $CR = randn(CR_m, 0.1)$
 ELSE $CR = randc(CR_m, 0.1)$
05: END

4.3.3 ARDE-SPX: Adaptive Repository with Fitness Based Selection

After constructing the two repositories of the DE strategies and the parameter control schemes, the next step is to make these repositories adaptive through generations. Accordingly, a new mechanism has been used to implement the adaptive process based on the fitness value of each strategy. This mechanism is inspired by the *fitness tournament selection* (FTS) procedure used in GA.

The Fitness Tournament Selection (FTS) is one of the most commonly used selection method in GA to choose the parent individuals for crossover process. FTS properties were characterized in (Blickle & Thiele, 1997). In this study, FTS has been chosen in preference to other selection methods like (roulette wheel selection and rank selection) because it has a very simple and easy concept to be implemented, and it requires no knowledge about the population. It can handle both minimization and maximization problems without any structural change in the fitness function. Moreover, it has no

restriction on negative fitness function. Finally, it is a fast technique because it does not require any further calculations for the average of fitness or any other population statistics, all the solutions are sent to the same processor using only their relative fitness values (Elsayed, Sarker, & Essam, 2014).

The basic idea of FTS is to stochastically select individuals from the current generation to create the basis of the next generation. The selection process has replicated nature in that the individuals with better fitness values will tend to have better chance to survive to the next generation; whereas the individuals with weaker fitness values will have less probability to survive. The general idea of the FTS can be encapsulated as follows:

- Evaluate each individual using the fitness function to get the fitness values f_i .
- Set a value to the selection probability $S_p \in [0,1]$. In practice, it has been found that it is always better to set the value of S_p to be greater than 0.5 in order to favor the fittest individuals and increase the selection pressure. The tournament selection with $S_p = 1.0$ is called *deterministic tournament* and the tournament selection with $S_p < 1.0$ is called *stochastic tournament*.
- Determine the tournament size $T_s \in [2, Np]$, because at least two individuals should be involved in the competition. In practice, the larger the tournament size, the higher probability that the new population will contain individuals with fitness values above average. The lower tournament size will produce population consists of individuals with low fitness values.
- Generate randomly uniformly distributed number $r \in [0,1]$. Then if $r \leq S_p$, the fitter candidate individual among the T_s individuals is selected; otherwise the weaker individual is chosen.

The last two steps are repeated until the desired number of survival individuals is obtained. Table 4.1 illustrates step-by-step on how to get the number of copies for 5

the selection step of DE, as follows:

Step 1 (Repository Construction): In the same selection step, a *repository* of empty *cells* is created. This repository is a result of uniting the two initial repositories of the DE strategies and the parameter control schemes into one repository called *adaptive repository (AR)*. The address of each cell in this repository is one combination of the DE strategies and the parameter control schemes (DE strategy, *F* adaptation scheme and/or *CR* adaptation scheme) that are involved in the ARDE algorithm. So, the total number of the cells is 20 because there are six DE strategies, two *F* adaptation schemes, and two *CR* adaptation schemes to be involved in constructing the *AR*; so the total number is $(6 \times 2 \times 2) - 4 = 20$ cells. This number is subtracted by 4 because the DE/current-to-*p*best/1 with archive and DE/rand-to-*p*best/1 with archive strategies do not require crossover strategy. The content of the cell is the fitness value(s) that has been obtained from applying its relative combination to the corresponding individual(s); otherwise, it is an empty cell if its relative combination has not been attempted in the search process. Figure 4.2 depicts the strategies and schemes combinations repository in the ARDE.

Step 2 (Individuals Classification): In the selection step of ARDE, the target vectors $X(t)$'s and the trial vectors $U(t)$'s are first evaluated. Then a one-by-one competition is started to determine the vectors of the next generation $X(t + 1)$'s. These new vectors are either *successful individuals* (i.e. the trial vectors that their fitness values are better than their corresponding target vectors after applying the DE strategy and parameter control schemes) or *failure individuals* (i.e. the target vectors that their fitness values are still better than their corresponding trial vectors even after applying the DE strategy and the parameter control schemes).

Step 3 (Averaging Cells): In this step, the cell that contains more than one fitness value is averaged using the equation,

$$cell_i = \frac{\sum_{j=1}^{nfc_i} f_{i,j}}{nfc_i} \quad (4.5)$$

where $f_{i,j}$ is the fitness values and nfc_i is the total number of fitness values in cell i . So, the result is a repository with cells that are either empty from a value (non-attempted in the search process) or occupied with a value (attempted in the search process).

Step 4 (Cells Selection): In this step of ARDE, the $U(t + 1)$ individuals are generated by assigning DE strategies and parameter control schemes to the $X(t + 1)$ s' successful and failure individuals using an adaptive technique as,

Step 4.1: For the successful individuals, the FTS mechanism is used to assign the DE strategies and parameter control schemes to them from the AR created in Step 1. In this adaptive method, the same table of FTS is created with only one difference is that the column of individuals is replaced with a column that includes the attempted cells of the repository with their average fitness values, as illustrated in Table 4.2. Then, for each successful individual, the FTS procedure determines which combination is to be assigned to it.

Table 4.2: The FTS method used to assign the DE strategies and parameter control schemes to the successful individuals

Successful Individuals	Repository of Strategies (AR)					Compete Cells	r	Winner Cell	
	Cells				Cells' Avg. Fitness				
	Cell No.	DE Strategies	F Scheme	CR Scheme					
Individual 11	1	DE/current-to-pbest/1/bin	F_c	CR_c	9.703E+04	2	1	0.46	1
Individual 23	2	DE/current-to-pbest/1	F_c	-	1.090E+05	4	2	0.39	4
Individual 15	3	DE/rand-to-pbest/1/bin	F_n	CR_n	8.386E+04	1	3	0.78	1
Individual 30	4	DE/current-to-pbest/1/exp	F_n	CR_c	6.236E+04	3	5	0.50	3
Individual 45	5	DE/rand-to-pbest/1/exp	F_c	CR_n	1.129E+05	5	4	0.14	4

In the same table, the problem is minimization and the values of S_p and T_s are set to 0.7 and 2, respectively.

In the same step, two counters are created. The first counter is called *Counts* which records the number of times its corresponding cell has been selected during one generation. The second counter is called *Countw* which records the number of times in which this cell has won the competition in FTS.

Step 4.2: For the failure individuals, a re-initialization mechanism is used to assign a combination of the strategies and schemes from the attempted and non-attempted cells of the *AR* with equal probability. In case there is not non-attempted cell, then the selection of combinations from the *AR* is applied among the attempted cells with equal probability.

Step 5 (Repository Cleaning, Rewarding, and Penalizing Processes):

Step 5.1 (Cleaning Process): In this step a cleaning mechanism is applied to discriminate the individuals on the basis of the age. In this mechanism the concept of "history" or "first-in-first-out (FIFO)" technique is used to eliminate the oldest fitness values in the cells. For each fitness value stored in a cell, its generation number (i.e., its history of birth) is stored too. Then cleaning is applied to those old fitness values existing in the *AR*'s cells. A fitness value in a cell of the *AR* can be tagged as "old" if its history is earlier than the current generation number by 10 generations. For example, if the current number of generations is 11, then the clean operation is applied to all those fitness values with history equals to 1. And at generation 12, the clean operation removes all fitness values with history 2, and so on. Accordingly, the cell that contains one fitness value with "old" tag is turned to be empty cell (non-attempted).

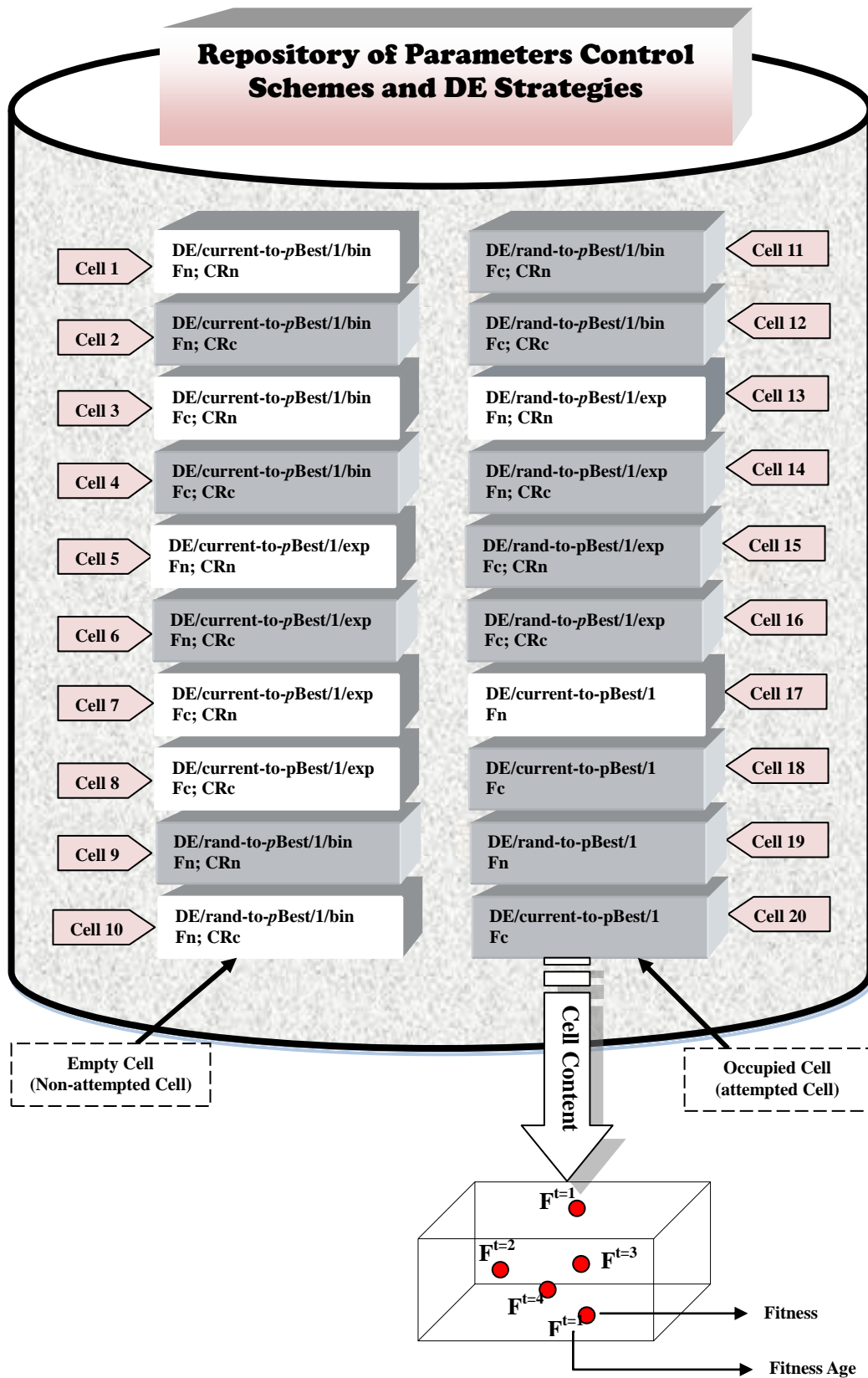


Figure 4.2: The complete structure of the adaptive repository, *AR* in the ARDE algorithm

Step 5.2 (Rewarding and Penalizing Processes): At each generation, the superior cell (cell with the highest p) is rewarded by eliminating its worst fitness value. The inferior cell (cell with the lowest p) is penalized by eliminating its best fitness value; where p is the cell's winning probability for each attempted-cell selected in the FTS mechanism and calculated as,

$$Cell_s.p = \frac{Cell_s.Countw}{Cell_s.Counts} \quad (4.6)$$

where s is the index of the attempted-cell that participated in the FTS mechanism during the current generation.

These steps of ARDE evolution are repeated till the stopping criteria are met. In Figure 4.2 and Table 4.2, the letters n and c attached with F and CR refer to the parameters adaptation schemes of Normal distribution and Cauchy distribution, respectively.

4.3.4 ARDE-SPX: The Local Search of Hill-Climbing Crossover (SPX)

As we discussed in Chapter 2, that general problem solvers with an overall successful and efficient performance do not exist. The No Free Lunch theorem (NFL) and other empirical evidences have strongly support this view. In addition, (Torn & Zilinskas, 1989) in the section entitled: *Global Search Method: Exploration and Exploitation*, have provided that there are two important aspects in the design of a reliable global search should be considered: exploration, which means that every part of the search space is search enough to ensure global reliability; exploitation, which means that search efforts are to be concentrated around the best solutions found so far through searching their neighborhoods to generate better solutions. These two goals have been achieved by many search algorithms using a combination of dictated global and local search techniques.

In many studies, local search has been used to enhance the overall performance of the DE algorithm. For example, in (Noman & Iba, 2008) a local search approach, called Differential Evolution with Adaptive Hill Climbing Simplex Crossover (DEahcSPX), has been integrated with the standard DE/rand/1/bin. The new algorithm has shown better or at least comparably performance to the standard DE on a wide range of benchmark functions. Another study in (Qin & Suganthan, 2005), Quasi-Newton method has been used as a local search method to speed up the convergence performance of the self-adaptive algorithm (SaDE) at different stages of the evolution. For the same algorithm, (Zhao, Suganthan, & Das, 2011) has integrated the local search method, called modified multi-trajectory search (MMTS) with SaDE, at different phases of the search process to increase the population diversity and provide approximate direction of evolution. Finally, in (Dong & Wang, 2014), the simplex crossover (SPX) has also been used to improve the DE performance in constraint optimization. It has been employed as a search engine in the neighborhood of the best feasible and infeasible solutions to guide the search to the optimal solution.

4.3.4.1 SPX: Hill-Climbing Simplex Crossover

Simplex crossover (SPX) (Tsutsui, Yamamura, & Higuchi, 1999) has first proposed for real-coded GAs and it is now considered as one of the most common used LS techniques in EAs. It has a very simple concept and it is easy to realize. SPX ensures a good ability of exploration at the early stages of evolution and good exploitation ability at the late stages of evolution. This so because SPX generates offspring based on uniform distribution and its search region is adaptively adjusted during evolution. The SPX variant that has been used in this study is called $SPX-n-m-\varepsilon$, where n is the dimension of the search space, m is the number of parents selected from the parental pool (population) for recombination in the range $[2, n + 1]$, and ε is the SPX control

parameter that determines the expanding rate.

The main idea of SPX is to start with selecting mutually independent parents, $X = \{x_1, x_2, \dots, x_m\}$ from the population; then it generates y_m offspring uniformly distributed around the center mass of their parents to form an area defined by m -simplex. This simplex is expanded in each direction of the search by $(x_i - O)$ with $(\varepsilon > 0)$, where O is the centroid of the m parents, calculated as:

$$O = \frac{1}{m} \sum_{i=1}^m x_i \quad (4.7)$$

and

$$y_i = O + \varepsilon(x_i - O), \quad i = 1, 2, \dots, m \quad (4.8)$$

The new solutions are then generated using the hill-climbing crossover operation. The SPX crossover depicted in Figure 4.3 shows a three parental SPX in a two-dimensional problem which can be denoted as $SPX-2-3-\varepsilon$. In this figure d refers to the difference value of $(x_i - O)$. Finally, the standard procedure of the SPX crossover is shown in Algorithm 4.8.

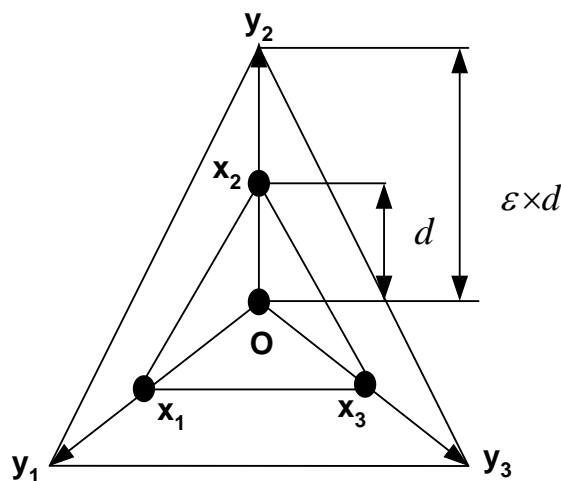


Figure 4.3: $SPX-2-3-\varepsilon$

Algorithm 4.8: The standard SPX crossover

01: Input: m (number of the parents), X (solution vectors), and ε (expansion rate of SPX)

02: Output: C (solution vector)

03: Begin

04: **Step 1:** Calculate the central mass of the parents,

$$O = \frac{1}{m} \sum_{i=1}^m x_i$$

05: **Step 2:** Generate the new offspring

06: **Step 2.1:** Generate uniformly random numbers,

$$r_i = u^{\frac{1}{i+1}}, \quad (i = 1, 2, \dots, m-1)$$

where u is a uniform random number $\in [0,1]$

07: **Step 2.2:** Calculate the m -simplex area, $y_i = O + \varepsilon(x_i - O)$, ($i = 1, 2, \dots, m$)

08: **Step 2.3:** Compute the new solutions by applying the hill-climbing SPX equation,

$$C_i = \begin{cases} 0, & (i = 1) \\ r_{i-1}(y_{i-1} - y_i + C_{i-1}), & (i = 2, \dots, m) \end{cases}$$

09: **Step 2.4:** Calculate the last descendent solution as, $C = C_m + y_m$

End

4.3.4.2 SPX Crossover with Group Based Replacement

In this study, a new way in employing the SPX crossover local search has been proposed and integrated with the adaptive ARDE algorithm.

The new implementation adopts the SPX mechanism previously explained with few additional modifications. First, in order to form the m -simplex, a group of m randomly individuals are selected from the current population. After that, the SPX crossover is applied to this group of individuals to generate new solution as in Algorithm 4.8. The new solution is then replaced with the worst individual in the group of parents and then in the population. This replacement is implemented without the need to evaluate the new solution and increase the number of fitness evaluations. This is so because, through experiment it has been found that the new solution generated from SPX crossover is mostly better than at least one individual in the group of the simplex with a very small probability to be worst. Moreover, it has also been found that even if the new individual is worst in its fitness value, the difference between its fitness and the fitness of the worse individual in the group is very small. In this way the new replacement will always have better effect on the population or at the very least no effect. This SPX process is applied once to the population individuals at every generation.

4.3.5 ARDE-SPX: Overall Algorithm Implementation

The first step in ARDE-SPX is the same step in all EAs dialects which is the population initialization, $P(t = 0) = \{X_1^0, X_2^0, \dots, X_{Np}^0\}$ with randomly generated candidate individuals using Equation 2.1. After the population initialization step, each individual is assigned with randomly selected combination of (DE strategy, F adaptation scheme and/or CR adaptation scheme) from the AR to produce the population of trial vectors $U(t) = \{U_1^t, U_2^t, \dots, U_{Np}^t\}$. In Step 4, the selection step of ARDE, the standard individual selection strategy of DE is implemented to obtain the population of $X(t + 1) = \{X_1^{t+1}, X_2^{t+1}, \dots, X_{Np}^{t+1}\}$; in addition, the repository AR , the set of the successful values of F and CR , and the archive of the inferior solution are all updated based on the fitness values of the yielded individuals. To improve the quality of the individuals in the $X(t + 1)$ population, the SPX-crossover is applied on randomly selected individuals from the current population in Step 5. In Step 6, and before the selection of the combinations, a cleaning scheme is applied to remove all the oldest fitness values from the cells according to a predefined epoch which is in this work set to be 10 generations. After this step, the selection of the strategies and schemes combinations can be implemented with regard to the average cells values of the AR in order to produce the new population $U(t + 1)$.

Algorithm 4.9: The ARDE-SPX algorithm

BEGIN

01: Step 1: (INITIALIZATION) Initialize the generation counter $t = 0$. Generates an initial population $P(t = 0)$ with random candidate solutions *target vectors*; $\langle X_1^t, X_2^t, \dots, X_{Np}^t \rangle \in [X_{min}, X_{max}]$

$$x_{i,j}^t = x_{j,min} + rand(0,1)_{i,j} \times (x_{j,max} - x_{j,min}) \quad (i = 1, 2, \dots, Np) \quad (j = 1, 2, \dots, D)$$

Set the mean values of $F_m = 0.5$; the mean value of $CR_m = 0.5$; the tournament size $T_s = 2$; the tournament probability $S_p = 1.0$; the archive $A = \emptyset$; the repository cells $AR = \emptyset$; the SPX crossover simplex size $m = 4$;

02: Step 2: (ASSIGN RANDOM STRATEGIES COMBINATIONS)

(* The strategies pseudo-codes associated for each cell in AR are already stated in Algorithm 4.1 and 4.2 for the mutation strategies (DE/current-to- p Best/1 and DE/rand-to- p Best/1), Algorithm 2.2 for the crossover strategies (bin, exp) and Algorithms 4.5-4.6 for the schemes of generating the values of F and CR *)

$$i = 1$$

WHILE ($i \leq Np$) DO

total ARDE-SPX complexity is calculated as follows,

(1) The complexity of both the mutation and crossover operations in the standard DE and ARDE-SPX is $O(Np \cdot D)$. In addition, the complexity of finding the p Best solution takes $O(Np \cdot \log Np)$; so, the overall complexity of the mutation and the crossover operations in ARDE-SPX is $O(Np \cdot (D + \log Np))$. (2) The selection operation in the standard DE and ARDE-SPX takes $O(Np)$ arithmetic operations. Since the parameters adaptation is already embedded in the selection operation, thus the overall complexity of the selection and parameters adaptation in ARDE-SPX is the same, $O(Np)$. (3) The complexity of the FTS selection scheme of the adaptive repository and its cleaning, rewarding and penalizing processes in its worst case is $O(Np)$. (4) In ARDE, the complexity of the group base selection of SPX is $O(m \cdot D)$ and since m can be set to D as the maximum simplex size, thus the complexity of SPX is $O(D^2)$. Hence, the total complexity of the ARDE-SPX can be calculated as,

$$\rightarrow O(Np \cdot (D + \log Np)) + O(Np) + O(Np) + O(D^2)$$

In literature, the population size Np is set to be proportional to the problem dimension D ; so, Np has been substituted by D in the above formula as,

$$\rightarrow O(D \cdot (D + \log D)) + O(D) + O(D) + O(D^2)$$

By eliminating the terms with less than D^2 , we have

$$\rightarrow O(D^2) + O(D \log D) + O(D) + O(D) + O(D^2)$$

$$\rightarrow O(2D^2)$$

when the constant 2 is dropped, the final complexity becomes

$$\rightarrow O(D^2) \xrightarrow{\text{total complexity}} O(T \cdot D^2)$$

where T is the total number of generations. The total complexity of ARDE-SPX is the same as in many other adaptive DE variants like JADE and SaJADE (Gong, Cai, Ling, & Li, 2011).

4.3.7 ARDE-SPX: Comparison with Other Adaptive DE Variants

ARDE-SPX is an adaptive algorithm that updates the values of F and CR in a learning controlled based and updates the selection of the DE strategies in a progressive controlled based. Compared with the variants of DE where the parameters and multi-strategies of DE are adaptive (SaDE, SaJADE, EPSDE, CoDE, SaDE-MMTS and HSPEADE), the main differences between ARDE-SPX and these algorithms are as follows:

- **The repository of DE strategies and parameter adaptive schemes:** ARDE-SPX is different from all the aforementioned algorithms in one major point which is the adaptation of the combinations of the parameters control schemes in addition to the adaptation of the DE strategies. This mechanism is called *adaptive repository, AR*. In this mechanism the algorithm decides which better combination to select through the adaptation process. The adaptive repository is a new attempt in the field of adaptive DE algorithms.
- **Ease of Implementation:** ARDE-SPX uses a very simple and straightforward mechanism to implement the adaption process of *AR* which is based on the fitness tournament scheme of selection (FTS) method used in GA. This mechanism decides which combinations of DE strategy and parameters control scheme are assigned for each individual based on the fitness function values accumulated every 10 generations. ARDE considers the successful and failure combinations at the same time, then the selection will be implemented based on the average of their fitness values; so, all the combinations attempted through every 10 generations have equal chances to survive. In addition, in ARDE the successful individual is more likely not to be assigned to the same combination that generates it. FTS establishes a competition between two selected combinations from the repository and selects the fitter one based on its fitness

value. One advantage of such a mechanism is that it guarantees that all the attempted combinations will participate in generating the next population and FTS decides which combination to choose with respect to its average fitness value. In this way, it increases the population diversity and increases the chance that the failure combination may become successful combination if it has been assigned to another individual.

- **Local Search Integration:** few adaptive DE algorithms have been yet integrated with local search technique except for SaDE-MMTS. In ARDE, the SPX crossover has been integrated to improve the quality of the solutions at each generation.
- **No Extra Parameters Control:** in ARDE-SPX, there are any extra control parameters need to be adapted during the run like in SaDE, SaJADE, SaDE-MMTS and HSPEADE. There are only few setting parameters (tournament size, T_s , tournament probability, S_p and the simplex size m) which can easily be tuned before the run according to predefined values.
- In general, using this simple mechanism of *AR* makes ARDE easy to implement more than SaDE, SaDE-MMTS and HSPEADE that are complex to implement; and more efficient than EPSDE, CoDE and SaJADE in the adaptation process. In addition, the ARDE-SPX algorithm updates the control parameters F and CR except for the population size Np . Because through experiments it has been found that the NP does not have significant effects on the performance of DE as the other control parameters, F and CR .

4.4 Summary

In this chapter, two major points have been addressed:

- 1) The general steps of how to develop an adaptive EA have been provided in details. These steps represent the general methodology to modify any standard EA and make it an adaptive one. In this study the Differential Evolution has been considered as the case study to apply these steps.
- 2) A new mechanism in adapting the parameters control schemes and the DE strategies have been proposed in this chapter. This proposed mechanism is based on creating a repository that involve all the possible combinations of these strategies and schemes and embedded them inside the standard DE to introduce the adaptive repository, ARDE. The evolutionary part of the DE is kept untouched except the selection step where the repository is updated. Then an additional step has been added to assign a combination for each individual in the population. The use of different schemes for the parameters adaptation and the mutation strategies makes DE more general algorithm that can solve different optimization problems. It also increases the population diversity and at the same time decreases the risk to fall into local optima. Finally, the ARDE has been combined with a local search method, SPX-crossover to make the algorithm more reliable and robust. In the following chapter, the computer experimental results will support this expectation.

Finally, in order to validate the new proposed adaptive DE algorithm (ARDE-SPX), several benchmark problems are used for the purpose in Chapter 5. Then, the performance of ARDE-SPX is compared with several adaptive DE algorithms. The ARDE-SPX is also validated by implementing it to estimate the parameters of a robot manipulator in Chapter 6.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, two types of experiments have been conducted. In the first experiment, six standard DE variants with different mutation schemes have been compared over 28 standard benchmark (unimodal and multimodal) functions. The objective of this experiment is to provide a table of F and CR settings associated with their corresponding DE variants. This table provides the best settings of each DE variant in which it could perform the best over each test problem. It is aimed at practitioners from different disciplines to help them achieve better results when adopting DE as an optimizer with less time and effort, as well as reducing the computational complexity when solving some applications that require the algorithm simplicity to ensure the implementation speed. In the second experiment, extensive experimental results and discussions have been presented to confirm the reliability of the proposed ARDE-SPX over several existing adaptive DE variants. These comparisons have been conducted in terms of the solution precision, successful rate and robustness over 33 standard and transformed benchmark (unimodal and multimodal) functions.

5.2 Experimental Setup

5.2.1 Unconstrained Benchmark Functions

Twenty-eight standard benchmark functions (i.e. functions with no modifications on their main characteristics, such as shifting, rotating or hybrid with other functions) are first considered in this study. These functions have been gleaned from different sources, for instance, the functions $(f_1, f_5 - f_6, f_8, f_{23})$ are called De Jong's functions

test suite (De Jong, 1975), functions ($f_{20} - f_{21}, f_{24} - f_{28}$) are called Dixon-Szegö functions (Dixon & Szego, 1978), and function (f_{13}) is called Griewank function (Griewank, 1981). They are either called *Convex* or *Non-Convex* depending on the characteristics of the function's landscape. More details about these functions can be found in (Lee & Yao, 2004; Torn & Zilinskas, 1989; Yao, Liu, & Lin, 1999). These functions have been classified in terms of their modality feature into two main classes (unimodal and multimodal) functions (see Figure 5.1). The word modality refers to the number of peaks in the fitness landscape.

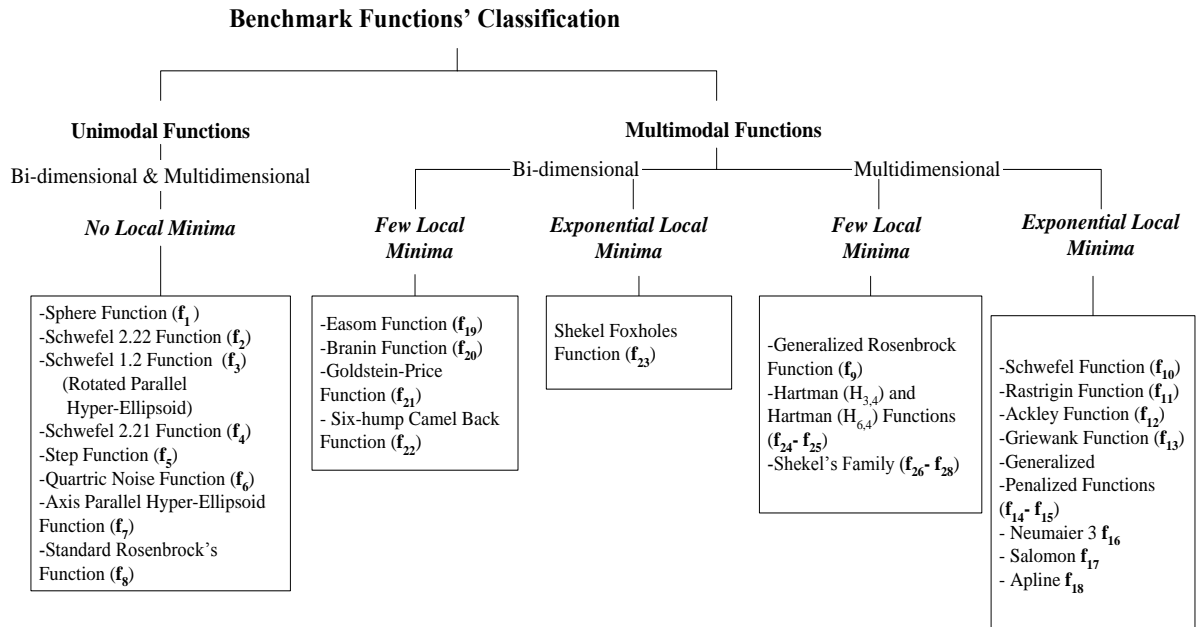


Figure 5.1: General classification of twenty eight standard benchmark functions

Class 1 (Unimodal Functions), these functions have only one global minimum solution in the feasible region of the search, they can be continuous ($f_1 - f_4, f_6 - f_8$) or discontinuous function (f_5) with no or single local minimum. In the presence of a local minimum, the task of optimization becomes a cumbersome task; as such, an improperly designed algorithm can get stuck in the local minima. An additional modulation can

also be padded to a function formulation in order to produce frequent local minima, as in problem (f_6) which is called a noisy quadratic function, without affecting the overall general characteristic of this function as being unimodal; this function is used to assess the performance of an algorithm on noisy data.

Some of the unimodal functions inherent a certain kind of deceptiveness and they are hard to optimize. The deceptiveness could be that the global minimum is either located on a flat surface (f_5) or located very near to the local minimum. Sometimes, deceptiveness is due to the fact that the global minimum lies in a narrow curving valley (f_8) or it shows fractal properties around the global minimum (f_6). These functions are considered as high-dimensional functions except the Rosenbrock function (f_8).

Class2 (Multimodal Functions), these functions ($f_9 - f_{28}$) have more than one (few or exponential) local minima with one or multiple global minima depending on the function characteristics. In most cases of multimodal functions, the number of local minima will dramatically increases with increasing the problem dimension. Functions ($f_{10} - f_{18}$) are high-dimensional multimodal functions where the number of local minima increases exponentially with the problem dimension. These functions are the most difficult class of problems for many optimization algorithms. Functions ($f_{19} - f_{22}, f_{24} - f_{25}$) are low-dimensional multimodal functions which have only few local minima. Function ($f_{23}, f_{26} - f_{28}$) are low-dimensional multimodal functions where the number of local minima increases exponentially and dependently on the problem characteristics. The number of local minima in these functions is determined by the problem characteristic. Generalized Rosenbrock Function (f_9) is a high-dimension multimodal function with few local minima. Over the last decade this function has been considered as a unimodal function with no local minimum to test the performance of many algorithms. Recent investigation (Shang & Qiu, 2006) has

proved that for $4 \leq D$, the Rosenbrock function has two minima; as such it has been considered as a multimodal function in this study. Easom function (f_{19}) is used to be considered as a unimodal function. However, this function has one global minimum located in a very small area relative to the search space. This global minimum is surrounded by a small number of local minima. Thus, it has been considered as a multimodal function in this study.

Then, five transformed (shifted and/or rotated) test functions (F_2, F_6, F_8, F_9 and F_{10}) are also considered in this study. The detailed description of these functions can be found in (Liang, Suganthan, & Deb, 2005; Qin, Huang, & Suganthan, 2009). They are:

- F_2 : Shifted Schwefel Function 1.2 (Unimodal)
- F_6 : Shifted Rotated Ackley Function (Multimodal)
- F_8 : Shifted Rotated Griewank Function (Multimodal)
- F_9 : Shifted Rastrigin Function (Multimodal)
- F_{10} : Shifted Rotated Rastrigin Function (Multimodal)

In Table 5.1, the problem dimension is denoted as D . the initial range of the variable is denoted as \mathbb{S} . The global minima values of the given functions are denoted as f_{min} . The complete functions formulations are provided in Appendices A and B.

Table 5.1: Problem dimension, global optimum parameters set, global optimum value, search range, and initialization range of thirty-three benchmark functions

f	D	Global Optimum x^*	Local Minima	$F(x^*)$	Search Range (S)	Initialization Range
Unimodal Standard Functions (high-dimensional Functions)						
f_1	30, 100	(0,0,...,0)	No	0	[-100,100]	[-100,100]
f_2	30, 100	(0,0,...,0)	No	0	[-10,10]	[-10,10]
f_3	30, 100	(0,0,...,0)	No	0	[-100,100]	[-100,100]
f_4	30, 100	(0,0,...,0)	No	0	[-100,100]	[-100,100]
f_5	30, 100	(0,0,...,0)	No	0	[-100,100]	[-100,100]
f_6	30, 100	(0,0,...,0)	No	0+noise	[-1.28,1.28]	[-1.28,1.28]
f_7	30, 100	(0,0,...,0)	No	0	[-100,100]	[-100,100]
Unimodal Standard Functions (low-dimensional Functions)						
f_8	2	(1,1)	No	0	[-30,30]	[-30,30]
Multimodal Standard Functions (high-dimensional Functions)						
f_9	30, 100	(1,1,...,1)	2	0	[-30,30]	[-30,30]
f_{10}	30, 100	(420.9687,..., 420.9687)	exponential	-418.9829·D	[-500, 500]	[-500, 500]
f_{11}	30, 100	(0,0,...,0)	exponential	0	[-5.12, 5.12]	[-5.12, 5.12]
f_{12}	30, 100	(0,0,...,0)	exponential	0	[-32,32]	[-32, 32]
f_{13}	30, 100	(0,0,...,0)	exponential	0	[-600, 600]	[-600, 600]
f_{14}	30, 100	(-1,-1,...,-1)	exponential	0	[-50, 50]	[-50, 50]
f_{15}	30, 100	(1,1,...,1)	exponential	0	[-50, 50]	[-50, 50]
f_{16}	30, 100	((D+1-1), 2(D+1-2), ..., D(D+1-D))	exponential	-D(D+4)(D-1)/6	[-D ² , D ²]	[-D ² , D ²]
f_{17}	30, 100	(0,0,...,0)	exponential	0	[-100,100]	[-100,100]
f_{18}	30, 100	(0,0,...,0)	exponential	0	[-10,10]	[-10,10]

Table 5.1- Continued

f	D	Global Optimum x^*	Local Minima	$F(x^*)$	Search Range (S)	Initialization Range
Multimodal Standard Functions (low-dimensional Functions)						
f_{19}	2	(π, π)	few	-1	[-100,100]	[-100,100]
f_{20}	2	$(-\pi, 12.275); (\pi, 2.275); (9.42478, 2.475)$	3	0.397887	[-5,10]x[0,15]	[-5,10]x[0,15]
f_{21}	2	(0, -1)	4	3.0000	[-2, 2]	[-2, 2]
f_{22}	2	$(-0.0898, 0.7126); (0.0898, -0.7126)$	4	-1.0316	[-3,3]x[-2,2]	[-3,3]x[-2,2]
f_{23}	2	$(-32, -32)$	= m	≈ 1	[-65.536, 65.536]	[-65.536, 65.536]
f_{24}	3	(0.114, 0.556, 0.852)	4	-3.86278	[0, 1]	[0, 1]
f_{25}	6	$(0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$	6	-3.3237	[0, 1]	[0, 1]
f_{26}	4	(4, 4, 4, 4)	5	-10.1532	[0, 10]	[0,10]
f_{27}	4	(4, 4, 4, 4)	7	-10.4029	[0, 10]	[0, 10]
f_{28}	4	(4, 4, 4, 4)	10	-10.5364	[0, 10]	[0, 10]
Unimodal Transformed Functions (high-dimensional Functions)						
F_2	30, 100	$(o, o, o \dots o)$	No	0	[-100,100]	[-100,100]
Multimodal Transformed Functions (high-dimensional Functions)						
F_6	30, 100	$(o, o, o \dots o)$	exponential	0	[-32,32]	[-32,32]
F_8	30, 100	$(o, o, o \dots o)$	exponential	0	No bounds	[0,600]
F_9	30, 100	$(o, o, o \dots o)$	exponential	0	[-5,5]	[-5,5]
F_{10}	30, 100	$(o, o, o \dots o)$	exponential	0	[-5,5]	[-5,5]

o is the shifted vector

5.2.2 Algorithms for Comparison

In this study two main experiments are conducted:

- *Experiment 1*, an empirical study on the competitive convergence nature of six DE mutation variants (with different trial vector generation strategies) are compared to solve twenty-eight unconstrained optimization problems. The general purposes of this comparison are: (1) to present a list of good choices of DE parameters for various optimization problems which would help the practitioners from different field achieve better solutions with little efforts to solve their optimization problems (2) to identify the best DE mutation variants to constitute the adaptive pool of strategies in the ARDE-SPX algorithm. For fair comparison, the crossover type that has been adopted in all strategies is the binomial crossover. The DE mutation strategies are:

- Four standard DE strategies have been selected for the comparison (DE/rand/1, DE/current-to-rand/1, DE/best/1, and DE/current-to-best/1), as in Equations (2.3-2.4, and 2.7-2.8) respectively. Here, the DE variants in Equations 2.3 and 2.7 are chosen because they are fast, robust, and they bear very strong exploration ability. Moreover, they employ the most commonly used trial vector generation strategy. While the DE variants in Equations 2.4 and 2.8 are chosen because they rely on the best solution found so far and this increases the reliability as well as the convergence rate.
- Two advanced DE strategies have been selected for the comparison (DE/current-to- p Best/1/bin without archive and DE/current-to- p Best/1/bin with archive), as in Equations 3.25 and 3.26 respectively. These strategies have shown powerful performance and improve the population diversity especially for high-dimension problems.

In this experiment, the comparison is conducted on 28 benchmark functions in terms of the solution precision (Mean \pm Std).

- *Experiment 2*, simulations are carried out to obtain a comparative performance analysis of the ARDE-SPX with respect to five recent and well known state-of-the-art adaptive DE variants such as jDE, SaDE, JADE without archive, JADE with archive and SaJADE. These algorithms have different adaptive characteristics, as explained in Chapter 4. The parameters of these algorithms have been set based on what have already been implemented in their corresponding references unless different settings used will be referred to it in the corresponding subsection. In this experiment, the parameter settings are as follow unless a change is mentioned:

- ARDE-SPX has been initialized with $CR_m = 0.5$, $F_m = 0.5$, $p = 0.05$, and *archive size* $= Np$. For the tournament selection $T_s = 2$ and $S_p = 1.0$. For SPX crossover $\varepsilon = 1.0$, and $m = 3$ for $D \leq 30$ and $m = 4$ for $D = 100$.
- jDE has been initialized with $\tau_1 = \tau_2 = 0.1$, as suggested in (Brest, Boskovic, Greiner, Zumer, & Maucec, 2007).
- SaDE has been initialized with $p_k = 1/4$ where $(k = 1, 2, 3, 4)$ and $CR_m = 0.5$, as suggested in (Qin, Huang, & Suganthan, 2009).
- JADE with and without archive have been initialized with $\mu_{CR} = 0.5$, $\mu_F = 0.5$, $p = 0.05$ and $c = 0.1$, and *archive size* $= Np$, as suggested in (Zhang & Sanderson, 2009b).
- SaJADE with $\mu_{CR} = 0.5$, $\mu_F = 0.5$, $\mu_S = 0.5$, $p = 0.05$ and $c = 0.1$, and *archive size* $= Np$, as suggested in (Gong, Cai, Ling, & Li, 2011).

In this experiment, all the comparisons have been conducted on 33 benchmark functions in terms of the convergence speed, solution precision and robustness. Finally, for all experiments, the following settings have been used unless a change is mentioned:

- The function dimension: $D = 30$ and $D = 100$.
- The population size: $Np = 30$ if $D \leq 10$, $Np = 100$ if $D = 30$, and $Np = 400$ if $D = 100$.

- Termination Error (Ter_Err): For functions $f_1 - f_5$, f_7 , and $f_9 - f_{18}$, Ter_Err= 10^{-8} ; for function F_2 , Ter_Err= 10^{-6} ; for functions f_6, F_6, F_8, F_9 and F_{10} , Ter_Err= 10^{-2} .
- Maximum Number of Fitness Evaluations (MAX_FEs): If $D = 30$ –for f_1, f_5, f_7, f_{14} and f_{15} , MAX_FEs= 150 000; for f_2 and f_{12} , MAX_FEs= 200 000; for $f_6, f_{13}, f_{16} - f_{18}, F_2, F_6, F_8, F_9$ and F_{10} , MAX_FEs= 300 000; for f_3, f_4 , and $f_9 - f_{11}$, MAX_FEs= 500 000. If $D = 100$ – for f_1 and f_7 , MAX_FEs= 800 000; for $f_5, f_6, f_{10}, f_{16} - f_{18}, F_2, F_6, F_8, F_9$ and F_{10} , MAX_FEs= 1 000 000; for f_2 and $f_{11} - f_{15}$, MAX_FEs= 1 200 000; for f_3, f_4 , and f_9 , MAX_FEs= 2 000 000.

5.2.3 Comparison Strategies and Metrics

In general, for fair comparison, the population size for all the DE algorithms over each problem have been initialized using the same population size, the same initial population values, and the same termination condition in each run; so, any difference in their performance is mainly belong to the algorithms' internal search operators. The results of all the experiments in this study have been validated on the basis of the paired two-sample t -test. Two important aspects have been considered in conducting the comparisons:

Statistical Testing: Since all the algorithms start with the same initial population over each problem instance, the paired t -test is used to compare their final accuracies which are assumed to be normally distributed (Bi & Xiao, 2011; Zhu, Tang, Fang, & Zhang, 2013). T-test is a popular tool in evolutionary computing used to measure the significant statistical difference between two sets (algorithms) of results. It is calculated using the direct difference method (Runyon, Haber, Pittenger, & Coleman, 1996) which is given by the following equation:

$$t = \frac{\bar{D}}{\sqrt{\frac{\sum D^2 - \frac{(\sum D)^2}{N}}{N(N-1)}}} \quad (5.1)$$

where $\bar{D} = \bar{X}_1 - \bar{X}_2$ and \bar{X}_1, \bar{X}_2 are the sample average of the two groups. D is the difference between the two pairs of random groups. N is the sample size of a single group, then the degree of freedom is calculated as $N - 1$; so, if the experiments have been implemented over 50 independent-runs, then the degree of freedom is approximately equal to 49. A two-tailed test is conducted at significance levels of $\alpha = 0.05$ (95% confidence interval, t -value= 2.0100) of all the statistical testing of results. The t -test is easy to be implemented since it is now included in most of the well-known software packages (e.g. Microsoft Office Excel, SPSS, etc.). In this study, the Microsoft Office Excel package of t -test has been used as shown in Figure 5.2. From the same figure it can be noted that the “ t -test: Pairs two sample for means” has been selected, because this test is suitable in case when each pair of two algorithms has the same initial condition. Moreover, the reason of choosing the Microsoft Office Excel package to do the t -test is because in this study Delphi 7 has been used to implement the algorithms and this software has a big number library that gives results in floating point number with high accuracy (e.g. 1.432E-90) which makes Excel better suited software for this type of data.

The Criteria of Comparisons: The comparative study focuses on four important criteria:

- (1) *The solution precision*, which is measured by the objective function value achieved till the maximum number of function evaluations (MAX_FEs). The mean and standard deviation of the optimized fitness values are calculated.
- (2) *The number of function evaluations (FESS)*, which is defined as the number of function evaluations required by an algorithm to reach the predefined

termination error value (Ter_Err) before reaching the MAX_FEs. FESS is used to measure the convergence speed and reliability of an algorithm.

(3) *The success rate (S_r)*, which is calculated as the number of successful runs divided by the total number of runs. Successful run is defined as the run where the accuracy of the final function value can reach the Ter_Err before reaching the MAX_FEs. S_r is used to measure the robustness of an algorithm.

(4) *The convergence graph*, which shows the convergence performance of an algorithm by plotting the average of the best fitness values achieved through the run.

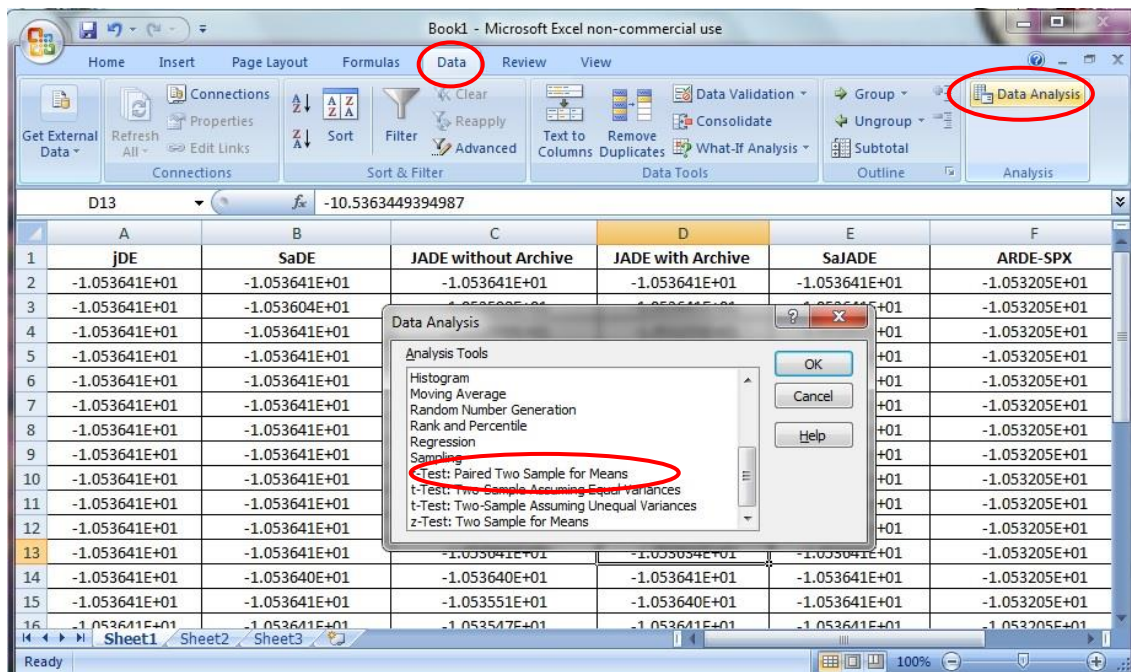


Figure 5.2: A snapshot of the Microsoft Office Excel package of t -test

System Configuration: Simulations were implemented on an Intel® Core™ Duo CPU T6670 @ 2.20GHz 2.20, GHz 4.00 GB of Memory, and Windows 7 Professional Version 2009 using Borland Delphi 7.

5.3 Experimental Results and Discussions

5.3.1 Comparison of Multiple DE Variants Based Parameter Tuning

In this comparison the problem dimension is set to ($D = 30$) for the high-dimensional problems only. The population size is set to ($Np = 100$) if ($D = 30$) and is set to ($Np = 30$) if ($D < 30$) for all test problems, except for the standard Rosenbrock function, f_8 . The population size of this function has been set to 100 because this function needs high solution diversity; otherwise it always stuck in the flat area of the solutions nearby the optimum. In this experiment, all of the six DE variants have been taken with no adaptive operation to any of their control parameters, and F and CR are set manually to values within the range $[0.1, 0.95]$ and 0.05 steps that give, approximately, good convergence performance on average. The p value in both DE/current-to- p Best/1/bin with no archive and DE/current-to- p Best/1/bin with archive is set to be 0.05. The solution precision has been used to detect which variant is more competitive. To measure the solution precision, the mean and standard deviation of the best fitness values achieved over 30 runs are calculated for each DE mutation variant. In this experiment, the preference of the best variant has been determined manually based on the best mean and standard deviation achieved without using t -test because the comparison here is not crucial.

Table 5.2 displays the F and CR values tuned by trial-and-error for the various pairs DE variant-test function that provide the best performance. The mean and standard deviation of the objective function values obtained for the unimodal functions: $f_1 - f_8$ and the multimodal functions: $f_9 - f_{28}$ are presented in Table 5.3. For space purposes, in these two tables the name of the DE variant has been substituted with the word “scheme” and a number that indicates the index of the variant, i.e, DE/rand/1/bin is Scheme1, DE/current-to-rand/1/bin is Scheme 2, DE/best/1/bin is Scheme 3,

DE/current-to-best/bin is Scheme 4, DE/current-to- p Best/1/bin without archive is Scheme 5 and DE/current-to- p Best/1/bin with archive is Scheme 6.

In the case of the unimodal functions $f_1 - f_5$, the results in Table 5.3 show that the best performance is provided by DE/current-to- p Best/1/bin with archive. For the noise function f_6 , the DE/current-to-best/1/bin could outperform other variants and the DE/current-to- p Best/1/bin with archive comes in the second place; whereas for the standard Rosenbrock function f_8 , the variant DE/best/1/bin could significantly outperform other variants. In general, it is clear that in the case of all unimodal functions the DE variants that involve the best individual found so far in their difference pairs can display better performance than other variants with random individuals only. This is mainly due to the characteristics of the unimodal functions that do not possess any local minimum that cause these types of greedy variants to fall in. Moreover, the greedy variants of DE can lead to very fast convergence in the unimodal functions.

In the case of multimodal functions with high-dimensions $f_9 - f_{18}$, the results in Table 5.3 show that the performance of the DE variants varies depending on the problem complexity. For f_9 the greedy DE variants present better results than the other variants and DE/current-to- p Best/bin with no archive performs the best except the DE/best/1/bin which always falls into the local minimum during the experiment because of its high greedy tendency. For f_{10} the entire DE variants could provide the same performance and almost with the same F and CR settings. Whereas for $f_{11} - f_{13}$ the DE variants with only random individuals (DE/rand/1/bin and DE/current-to-rand/1/bin) could significantly outperform other variants, as the characteristics of these functions demand an algorithm with high exploration and exploitation capability to escape from the local minima. However, in the case of f_{13} the DE variants DE/current-to- p Best with and without archive could also perform well. This proves the reliability of these two variants to create the proper balance between the exploitation and exploration, as well as the fast

convergence. This also has made these two variants perform the best on $f_{14} - f_{18}$.

In the case of the multimodal low dimensional problems $f_{19} - f_{28}$, the settings of F and CR could bring about the desired performance for all the variants in $f_{19} - f_{24}$. The difference appeared in $f_{25} - f_{28}$ as these functions have the local minimum very near to the global minimum which makes these functions demand algorithms with different characteristics in order to escape from the local entrapment. For example, in f_{25} the greedy variants DE/current-to-best/1/bin, and DE/current-to-pBest/1/bin with and without archive could outperform other variants, whereas in $f_{26} - f_{27}$ the variant DE/rand/1/bin outperform all other variants because of its high exploration capability.

Finally, the results displayed in Table 5.3 are based on the settings provided in Table 5.2 which are obtained by trial-and-error experiments. This implies that these values can be changed if future experiments provide better results. However, based on the current experimental settings it can be noted that the selection of the DE variants and the associated parameter settings are problem dependent. There are some problems demand greedy variants whereas there are some other problems demand high randomness in the algorithm. At the same time, there are some problems require different settings for F and CR . For the powerful DE variants, it can be noted that the DE/current-to-pBest/1/bin with and without archive are very powerful variants because of their characteristics to involve the greediness tendency as well as the proper randomness in the same time.

Table 5.2: The F and CR values tuned for each pair of DE mutation variant-benchmark functions

Fun.	Scheme 1		Scheme 2		Scheme 3		Scheme 4		Scheme 5		Scheme 6	
	F	CR	F	CR	F	CR	F	CR	F	CR	F	CR
f_1	0.4	0.9	0.5	0.5	0.6	0.8	0.6	0.8	0.5	0.6	0.5	0.9
f_2	0.4	0.9	0.5	0.7	0.6	0.8	0.6	0.8	0.5	0.7	0.5	0.9
f_3	0.4	0.9	0.6	0.9	0.6	0.9	0.6	0.8	0.6	0.9	0.5	0.9
f_4	0.4	0.5	0.6	0.4	0.6	0.3	0.7	0.8	0.6	0.6	0.5	0.9
f_5	0.4	0.9	0.4	0.4	0.5	0.5	0.6	0.6	0.5	0.5	0.5	0.9
f_6	0.4	0.9	0.4	0.1	0.4	0.2	0.5	0.1	0.5	0.1	0.5	0.9
f_7	0.4	0.9	0.5	0.5	0.6	0.8	0.6	0.8	0.5	0.6	0.5	0.9
f_8	0.2	0.95	0.3	0.95	0.6	0.95	0.6	0.95	0.75	0.95	0.95	0.95
f_9	0.5	0.9	0.7	0.9	0.7	0.8	0.7	0.8	0.7	0.8	0.6	0.9
f_{10}	0.95	0.1	0.95	0.1	0.95	0.1	0.95	0.1	0.95	0.1	0.95	0.1
f_{11}	0.2	0.1	0.5	0.1	0.5	0.1	0.5	0.1	0.5	0.1	0.95	0.1
f_{12}	0.4	0.9	0.6	0.8	0.5	0.1	0.7	0.4	0.5	0.1	0.7	0.9
f_{13}	0.4	0.9	0.6	0.8	0.7	0.2	0.7	0.2	0.7	0.7	0.5	0.9
f_{14}	0.2	0.1	0.6	0.8	0.5	0.2	0.5	0.2	0.5	0.2	0.6	0.9
f_{15}	0.4	0.9	0.5	0.3	0.5	0.3	0.5	0.3	0.5	0.3	0.6	0.9
f_{16}	0.5	0.95	0.6	0.9	0.95	0.9	0.95	0.95	0.7	0.95	0.7	0.95
f_{17}	0.4	0.9	0.6	0.9	0.7	0.9	0.8	0.9	0.7	0.9	0.7	0.95
f_{18}	0.3	0.9	0.5	0.1	0.5	0.1	0.5	0.1	0.6	0.9	0.6	0.9
f_{19}	0.4	0.9	0.6	0.9	0.6	0.9	0.4	0.9	0.4	0.9	0.6	0.9
f_{20}	0.4	0.9	0.4	0.9	0.6	0.8	0.6	0.8	0.6	0.8	0.5	0.9
f_{21}	0.4	0.9	0.4	0.9	0.6	0.7	0.4	0.9	0.4	0.9	0.5	0.9
f_{22}	0.4	0.9	0.6	0.8	0.6	0.8	0.6	0.8	0.6	0.8	0.5	0.9
f_{23}	0.5	0.9	0.95	0.8	0.95	0.8	0.95	0.8	0.95	0.8	0.95	0.8
f_{24}	0.5	0.9	0.6	0.8	0.7	0.8	0.7	0.8	0.7	0.8	0.7	0.8
f_{25}	0.5	0.95	0.7	0.4	0.95	0.1	0.7	0.2	0.7	0.2	0.7	0.2
f_{26}	0.6	0.9	0.65	0.9	0.95	0.2	0.2	0.1	0.2	0.1	0.95	0.2
f_{27}	0.6	0.9	0.65	0.9	0.95	0.2	0.2	0.1	0.2	0.1	0.65	0.95
f_{28}	0.5	0.9	0.6	0.8	0.95	0.2	0.2	0.1	0.95	0.95	0.95	0.95

Table 5.3: Mean and standard deviation of 30-dimensional and low dimensional problems achieved for multiple DE mutation strategies averaged over 30-independent runs

Fun.	MAX-NFEs	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	Scheme 6
		Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std
f_1	150,000	5.724E-28 \pm 5.924E-28	5.415E-52 \pm 1.798E-51	1.111E-73 \pm 2.140E-73	7.570E-77 \pm 8.227E-77	8.406E-86 \pm 4.660E-86	3.357E-112\pm 1.117E-111
f_2	200,000	3.214E-20 \pm 1.919E-20	1.252E-39 \pm 1.082E-39	1.331E-51 \pm 1.862E-51	9.462E-51 \pm 4.366E-51	2.878E-63 \pm 1.844E-63	2.620E-74\pm 3.186E-74
f_3	500,000	1.455E-13 \pm 2.650E-13	2.931E-23 \pm 5.562E-23	1.911E-68 \pm 6.323E-68	8.280E-48 \pm 1.352E-47	5.536E-69 \pm 1.953E-68	8.549E-206\pm 0.000E+00
f_4	500,000	7.941E-16 \pm 2.922E-16	9.708E-24 \pm 4.153E-24	4.293E-22 \pm 8.694E-22	3.399E-26 \pm 6.883E-26	2.554E-44 \pm 1.330E-43	1.875E-140\pm 8.4231E-140
f_5	10,000	2.000E+02 2.632E+02	1.267E+00 \pm 1.143E+00	2.000E-01 \pm 4.068E-01	0.000E+00\pm 0.000E+00	0.000E+00\pm 0.000E+00	0.000E+00\pm 0.000E+00
f_6	300,000	5.541E-02 \pm 1.692E-02	3.486E-03 \pm 1.823E-03	9.941E-03 \pm 2.986E-03	9.648E-04\pm 3.828E-04	1.305E-03 \pm 5.360E-04	2.867E-03 \pm 2.194E-03
f_7	150,000	4.764E-27 \pm 4.682E-27	1.953E-50 \pm 5.365E-50	6.153E-72 \pm 1.466E-71	1.139E-75 \pm 1.277E-75	1.749E-84 \pm 2.452E-84	1.204E-111\pm 3.597E-111
f_8	10,000	6.666E-23 \pm 2.264E-22	6.935E-04 \pm 9.535E-04	0.000E+00\pm 0.000E+00	1.531E-14 \pm 7.916E-14	9.379E-15 \pm 2.365E-14	9.624E-10 \pm 1.918E-09
f_9	500,000	6.269E-12 \pm 2.165E-11	2.358E-10 \pm 3.725E-10	3.998E-01 \pm 1.220E+00	2.909E-30 \pm 1.459E-29	2.058E-31\pm 1.127E-30	1.923E-30 \pm 5.867E-30
f_{10}	150,000	-1.257E+04 \pm 2.013E-05	-1.257E+04 \pm 1.056E-04	-1.257E+04\pm 8.946E-09	-1.257E+04 \pm 1.027E-07	-1.257E+04 \pm 2.676E-07	-1.257E+04 \pm 3.688E-02
f_{11}	200,000	0.000E+00\pm 0.000E+00	6.239E-12 \pm 5.289E-12	6.633E-02 \pm 2.524E-01	3.987E-16 \pm 6.214E-16	3.268E-16 \pm 3.018E-16	4.071E-17 \pm 2.476E-17
f_{12}	200,000	2.385E-18\pm 0.000E+00	2.385E-18\pm 0.000E+00	4.120E-18 \pm 2.351E-33	4.120E-18 \pm 2.351E-33	2.790E-18 \pm 7.462E-19	2.848E-18 \pm 7.802E-19
f_{13}	300,000	0.000E+00\pm 0.000E+00	0.000E+00\pm 0.000E+00	4.156E-20 \pm 2.332E-20	2.349E-20 \pm 2.732E-20	0.000E+00\pm 0.000E+00	0.000E+00\pm 0.000E+00
f_{14}	150,000	1.032E-31 \pm 6.372E-32	4.001E-33 \pm 1.859E-32	3.077E-39\pm 6.637E-55	3.077E-39\pm 6.637E-55	3.077E-39\pm 6.637E-55	1.721E-34 \pm 9.426E-34

Table 5.3- Continued

Fun.	MAX-NFEs	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	Scheme 6
		Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
f_{15}	150,000	8.638E-28± 1.110E-27	1.315E-33± 7.201E-33	1.440E-38± 2.655E-54	1.440E-38± 2.655E-54	1.440E-38± 2.655E-54	4.800E-33± 2.629E-32
f_{16}	150,000	-4.903E+03± 3.261E+01	-4.891E+03± 7.604E+01	-4.280E+03± 4.799E+02	-4.586E+03± 3.519E+02	-4.930E+03± 2.457E-10	<i>-4.930E+03±</i> <i>7.425E-04</i>
f_{17}	150,000	1.965E-01± 1.824E-02	<i>1.472E-01±</i> <i>4.124E-02</i>	2.898E-01± 5.477E-02	2.303E-01± 4.634E-02	1.913E-01± 2.680E-02	1.098E-01± 3.051E-02
f_{18}	150,000	3.394E-13± 1.819E-12	6.445E-03± 1.047E-03	1.839E-16± 2.268E-16	1.335E-04± 1.032E-04	<i>9.754E-17±</i> <i>1.999E-16</i>	1.112E-17± 3.880E-17
f_{19}	6,000	-1.000E+00± 0.000E+00	-1.000E+00± 0.000E+00	-1.000E+00± 0.000E+00	-1.000E+00± 0.000E+00	-1.000E+00± 0.000E+00	-1.000E+00± 2.132E-13
f_{20}	6,000	3.979E-01± 1.129E-16	3.979E-01± 1.129E-16	3.979E-01± 1.129E-16	3.979E-01± 1.129E-16	3.979E-01± 1.129E-16	3.979E-01± 1.129E-16
f_{21}	6,000	3.000E+00± 0.000E+00	3.000E+00± 0.000E+00	3.000E+00± 0.000E+00	3.000E+00± 0.000E+00	3.000E+00± 0.000E+00	3.000E+00± 0.000E+00
f_{22}	6,000	-1.032E+00± 0.000E+00	-1.032E+00± 0.000E+00	-1.032E+00± 0.000E+00	-1.032E+00± 0.000E+00	-1.032E+00± 0.000E+00	-1.032E+00± 0.000E+00
f_{23}	6,000	9.980E-01± 3.388E-16	9.980E-01± 3.388E-16	9.980E-01± 3.388E-16	9.980E-01± 3.388E-16	9.980E-01± 3.388E-16	9.980E-01± 3.388E-16
f_{24}	6,000	-3.863E+00± 2.710E-15	-3.863E+00± 2.710E-15	-3.863E+00± 2.710E-15	-3.863E+00± 2.710E-15	-3.863E+00± 2.710E-15	-3.863E+00± 2.710E-15
f_{25}	6,000	-3.290E+00± 5.348E-02	-3.322E+00± 4.136E-04	-3.322E+00± 4.223E-05	-3.322E+00± 2.042E-06	-3.322E+00± 1.485E-06	-3.322E+00± 3.072E-04
f_{26}	6,000	-1.0153E+01± 1.807E-15	<i>-1.01525E+01±</i> <i>3.237E-03</i>	-9.462E+00± 1.742E+00	-9.635E+00± 1.538E+00	-1.013E+01± 4.828E-02	-9.881E+00± 3.339E-01
f_{27}	6,000	-1.040E+01± 2.471E-10	-1.0402E+01± 1.372E-05	-1.037E+01± 1.222E-01	-1.040E+01± 1.191E-02	-1.040E+01± 4.191E-04	<i>-1.040E+01±</i> <i>3.076E-06</i>
f_{28}	6,000	-1.053E+01± 9.034E-15	-1.053E+01± 6.648E-05	-1.053E+01± 8.776E-03	-1.053E+01± 6.320E-04	-1.053E+01± 9.034E-15	-1.053E+01± 3.039E-08

5.3.2 Comparison of Multiple Adaptive DE Variants

In this subsection, a performance comparison of ARDE-SPX with that of jDE, SaDE, JADE without archive (JADEwo), JADE with archive (JADEw), and SaJADE is presented in terms of three aspects: 1) Final solution precision; 2) the success rate; 3) Convergence speed. The parameters settings of all algorithms are used as mentioned in Section 5.2.2-*Experiment 2*. For all the test problems the dimensions are provided in Table 5.1.

5.3.2.1 Final Solution Precision (Mean \pm Std)

In this subsection, the performance comparison between ARDE-SPX with the other adaptive DE variants is conducted based on the quality of the final solution achieved after the optimization process has terminated. This is measured by averaging the final solution and the standard deviation over 50-independent runs using the pre-specified MAX_NFEs as mentioned in Section 5.2.2.

Tables 5.4-5.6 show the mean and standard deviation of the final solutions for 50-independent runs of each of the six algorithms on 33 benchmark functions for the high ($D = 30$, $D = 100$) and low dimensional problems. The statistics of this comparison is calculated at the end of the optimization. Since all the algorithms start with the same initial population, the paired t -test is used for these statistics to compare the means of the best and second best algorithms. In the last row of each table the statistical symbols ‘†’ and ‘‡’ are defined. ‘†’ indicates that the ARDE-SPX algorithms performs significantly better than the other algorithms at a 0.05 level of significance of 49 degrees of freedom by the paired two tailed t -test; whereas the symbol ‘‡’ indicates that the corresponding algorithm is better than the ARDE-SPX algorithm. The best results are typed in bold and the second best in italic.

The results in these tables suggest that the ARDE-SPX performs the best and second best for most of the test problems.

From Table 5.4, which shows the results of $D = 30$ problems, it can be noted that the ARDE-SPX has significantly best performance for the test problems $f_1, f_3 - f_4, f_7, f_9, f_{12}, f_{14} - f_{15}, f_{17}, F_2, F_6,$ and F_{10} , and has second best performance for the test problems $f_2, f_6, f_{16}, f_{18},$ and F_8 over all the adaptive DE methods. In the case of f_{10} , ARDE-SPX performs worse than jDE and SaDE because this function requires a method with high randomness ability to overcome the difficulties of exploring its landscape. In the same way, jDE outperforms ARDE-SPX over the test function f_{18} . In the case of f_{11} , it can be noted that the methods with high greediness tendency such as JADEwo and SaJADE outperform other methods with less greediness such as ARDE-SPX and jDE. For the same reason, SaJADE outperforms ARDE-SPX in the convergence speed for the test functions f_2 and f_6 , because it involves the JADE mutation with no archive in addition to the mutation with archive. Accordingly, the JADEwo focuses on the best fitness area and this increases the convergence speed especially in somehow low dimensional problems (i.e. $D = 30$).

From Table 5.5, which shows the results of the high dimension $D = 100$ problems, it can be noted that on the majority of the test problems, ARDE-SPX performs significantly better than other DE variants. This is because ARDE-SPX could diversify the yielded offspring using its multiple mutation and crossover schemes even with the small population size of 400 that is probably not sufficient for most of the test cases. The only competitive method in this comparison is jDE which shows significant better performance than ARDE-SPX on the test functions $f_{10}, f_{11},$ and F_9 . This is because the mutation DE/rand/1 in jDE is more robust than the greedy mutations of JADE and this may improve the algorithm performance on some of the test problems. For the functions f_5 and f_{16} , the algorithms SaJADE and JADEwo could outperform ARDE-SPX in the

convergence speed only but not on the quality of the final solution achieved.

From Table 5.6, which shows the simulation results of the low dimensional problems $f_8, f_{19} - f_{28}$, it can be noted that there is no superior adaptive DE algorithm to solve these set of functions. The simulation results also indicate that DE with adaptive parameter control does not work efficiently within the small number of fitness evaluations required to optimize these problems. In these low dimensional problems, the adaptive DE algorithm fluctuates around the optimal fitness area and more often falls into local optima because of the limited number of generations, as well as the small population size; so, it is recommended to use the standard DE with its multiple mutation variants to solve these low dimensional problems as suggested in Table 5.2.

In general, the ARDE-SPX approach is the first in terms of the quality of the final solution followed by jDE, SaJADE, JADEw, JADEwo and then SaDE. This high reliable performance of ARDE-SPX stems from both the diversity provided by the multiple mutation schemes and crossover schemes incorporated with the DE/current-to- p Best/1 and the multiple adaptive parameter control schemes. In addition, an important observation is in the case of the difficult problems (transformed functions) the ARDE-SPX shows significant performance better than the other methods especially in the case of rotation problems. This is due to incorporating the DE/current-to- p Best/1 with no crossover that shows reliability in solving these types of problems.

Table 5.4: Mean and standard deviation of 30-dimensional problems averaged over 50-independent runs for the high dimensional test problems $f_1 - f_7$; $f_9 - f_{18}$; $F_2, F_6, F_8 - F_{10}$

Fun.	MAX-NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std
f_1	150,000	1.461E-28 \pm 2.511E-28 [†]	2.523E-35 \pm 2.524E-36 [†]	2.334E-61 \pm 4.244E-61 [†]	3.245E-56 \pm 4.332E-55 [†]	2.556E-76 \pm 3.432E-76 [†]	3.856E-80\pm 2.173E-80
f_2	200,000	1.472E-23 \pm 1.121E-23 [†]	5.615E-25 \pm 5.122E-25 [†]	1.796E-26 \pm 2.334E-26 [†]	1.025E-24 \pm 1.761E-23 [†]	5.820E-45\pm 2.001E-45[‡]	6.544E-43 \pm 7.376E-43
f_3	500,000	4.233E-14 \pm 2.162E-13 [†]	1.267E-37 \pm 2.372E-37 [†]	3.874E-61 \pm 4.556E-60 [†]	4.231E-79 \pm 4.002E-80 [†]	1.112E-77 \pm 4.121E-77 [†]	7.099E-85\pm 7.603E-85
f_4	500,000	2.217E-15 \pm 2.000E-15 [†]	5.313E-27 \pm 4.581E-26 [†]	6.884E-23 \pm 7.332E-23 [†]	5.431E-54 \pm 5.899E-55 [†]	2.731E-20 \pm 2.877E-20 [†]	1.156E-59 1.390E-60
f_5	10,000	2.978E+02 \pm 1.885E+02	5.061E+01 \pm 6.231E+01	2.878E+00 \pm 1.322E+00	4.556E+00 \pm 1.778E+00	0.00E+00 \pm 0.00E+00	0.00E+00 \pm 0.00E+00
	150,000	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.00E+00 0.00E+00	0.000E+00 \pm 0.000E+00
f_6	300,000	2.654E-03 \pm 7.751E-04 [†]	1.962E-03 \pm 2.331E-04 [†]	5.891E-04 \pm 1.885E-04 [†]	5.988E-04 \pm 2.455E-04 [†]	4.221E-04\pm 2.121E-04[‡]	5.132E-04 \pm 1.561E-04
f_7	150,000	2.521E-27 \pm 2.361E-27 [†]	3.415E-30 \pm 2.433E-30 [†]	1.234E-58 \pm 3.445E-58 [†]	2.135E-55 \pm 2.422E-54 [†]	5.011E-70 \pm 4.212E-70 [†]	7.900E-75\pm 6.438E-75
f_9	500,000	1.480E-03 \pm 2.137E-03 [†]	6.785E-02 \pm 4.862E-01 [†]	2.223E-30 \pm 2.334E-30 [†]	1.923E-29 \pm 1.923E-29 [†]	2.134E-30 \pm 5.231E-30 [†]	1.343E-30\pm 1.345E-30
f_{10}	100,000	-1.257E+04\pm 6.324E-11[‡]	-1.257E+04 \pm 5.223E-10 [‡]	-1.257E+04 \pm 3.568E-05 [†]	-1.257E+04 \pm 3.887E-04 [†]	-1.257E+04 \pm 6.877E-07 [†]	-1.257E+04 \pm 5.572E-09
	500,000	-1.257E+04 \pm 0.000E+00	-1.257E+04 \pm 0.000E+00	-1.257E+04 \pm 0.000E+00	-1.257E+04 \pm 0.000E+00	-1.257E+04 \pm 0.000E+00	-1.257E+04 \pm 0.000E+00

Table 5.4- Continued

Fun.	MAX_NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
f_{11}	100,000	4.799E-04± 1.006E-04 [†]	2.623E-03± 7.966E-04 [†]	2.111E-04± 2.311E-04[‡]	3.024E-04± 3.459E-04 [‡]	2.654E-04± 2.332E-04 [‡]	3.568E-04± 6.775E-05
	500,000	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00
f_{12}	50,000	2.465E-04± 5.211E-05 [†]	4.765E-06± 7.980E-07 [†]	1.098E-09± 6.532E-10 [†]	4.102E-09± 1.877E-09 [†]	1.145E-12± 2.000E-12 [†]	5.234E-15± 1.312E-15
	200,000	4.781E-15± 8.776E-15	4.144E-15± 0.000E+00	4.144E-15± 0.000E+00	4.144E-15± 0.000E+00	4.144E-15± 0.000E+00	4.144E-15± 0.000E+00
f_{13}	50,000	2.002E-05 4.781E-05	3.351E-09± 3.592E-08	6.771E-10± 5.664E-11	9.322E-07± 8.558E-07	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00
	300,000	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00
f_{14}	50,000	4.445E-07 3.564E-07 [†]	7.146E-11± 6.211E-12 [†]	1.587E-17± 2.344E-16 [†]	3.458E-16± 1.763E-15 [†]	1.354E-19± 2.179E-19 [†]	4.811E-21± 5.433E-21
	150,000	2.595E-29± 6.734E-29	1.601E-32± 0.000E+00	1.601E-32± 0.000E+00	1.601E-32± 0.000E+00	1.601E-32± 0.000E+00	1.601E-32± 0.000E+00
f_{15}	50,000	1.843E-06± 2.844E-07 [†]	1.223E-09± 2.367E-09 [†]	1.986E-15± 5.432E-15 [†]	3.671E-13± 4.112E-13 [†]	8.416E-17± 7.328E-17 [†]	2.455E-19± 1.223E-19
	150,000	1.802E-28± 2.023E-28	1.401E-32± 0.000E+00	1.401E-32± 0.000E+00	1.401E-32± 0.000E+00	1.401E-32± 0.000E+00	1.401E-32± 0.000E+00
f_{16}	300,000	-1.003E+03± 1.211E+00 [†]	-2.891E+02± 1.455E+02 [†]	-3.390E+03± 4.005E-04 [†]	-4.930E+03± 3.645E-10[‡]	-4.925E+03± 2.433E-07 [†]	-4.930E+03± 1.889E-09
f_{17}	300,000	1.962E-01± 1.372E-02 [†]	1.487E-01± 4.963E-02 [†]	2.001E-01± 2.132E-02 [†]	1.982E-01± 2.680E-02 [†]	1.821E-01± 3.900E-02 [†]	1.128E-01± 2.141E-02
f_{18}	300,000	5.864E-10± 7.521E-10[‡]	2.875E-06± 3.363E-06 [†]	3.667E-06± 3.001E-06 [†]	2.766E-05± 3.112E-06 [†]	3.120E-07± 5.131E-07 [†]	4.242E-08± 5.643E-09

Table 5.4- Continued

Fun.	MAX_NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std	Mean ± Std
F_2	300,000	5.043E-07± 4.790E-06 [†]	1.331E-08± 1.746E-09 [†]	5.823E-23± 4.874E-23 [†]	3.473E-26± 3.216E-26 [†]	1.782E-25± 2.625E-25 [†]	8.245E-28± 7.616E-29
F_6	300,000	2.033E+01± 1.451E-02 [†]	2.048E+01± 2.165E-02 [†]	2.031E+01± 5.098E-01 [†]	2.033E+01± 2.761E-01 [†]	2.024E+01± 6.801E-02 [†]	2.020E+01± 3.648E-02
F_8	300,000	2.905E-02± 6.122E-03 [†]	2.431E-03± 2.614E-02 [†]	4.921E-02± 3.271E-02 [†]	2.562E-03± 4.627E-02[‡]	1.453E-02± 2.672E-02 [†]	5.542E-03± 2.113E-03
F_9	300,000	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00	0.000E+00± 0.000E+00
F_{10}	300,000	4.651E+01± 3.628E+00 [†]	2.980E+01± 2.518E+00 [†]	3.241E+01± 5.017E+00 [†]	3.102E+01± 2.451E+00 [†]	2.931E+01± 3.267E+00 [†]	1.029E+01± 2.813E+00

[†] indicates that ARDE-SPX performs better than other algorithms with 95% confidence level by *t*-test.
[‡] indicates that the corresponding algorithm is better than ARDE-SPX.

Table 5.5: Mean and standard deviation of 100-dimensional problems averaged over 50 independent runs for the high dimensional test problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$

Fun.	MAX_NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std
f_1	800,000	4.825E-17 \pm 2.113E-17 [†]	3.111E-20 \pm 3.045E-20 [†]	1.566E-50 \pm 2.332E-50 [†]	8.511E-69 \pm 8.677E-70 [†]	2.953E-81 \pm 1.811E-81 [†]	3.237E-88 \pm 3.433E-88
f_2	1,200,000	3.445E-14 \pm 3.233E-13 [†]	2.112E-16 \pm 1.113E-15 [†]	2.662E-39 \pm 2.222E-39 [†]	6.755E-46 \pm 5.882E-47 [†]	1.394E-50 \pm 2.171E-49 [†]	1.603E-52 \pm 2.211E-51
f_3	2,000,000	5.466E+00 \pm 5.332E+00 [†]	2.889E-02 \pm 1.674E-02 [†]	4.534E-06 \pm 4.221E-06 [†]	3.235E-09 \pm 3.792E-09 [†]	5.001E-09 \pm 4.551E-08 [†]	8.201E-10 \pm 7.941E-11
f_4	2,000,000	4.121E-01 \pm 4.521E+00 [†]	3.711E-05 \pm 2.811E-04 [†]	1.005E-02 \pm 1.112E-03 [†]	7.223E-08 \pm 6.611E-07 [†]	1.020E-05 \pm 2.609E-04 [†]	6.351E-09 \pm 5.419E-08
f_5	40,000	2.122E+04 \pm 7.172E+03 [†]	2.045E+03 \pm 1.971E+03 [†]	2.331E+02 \pm 3.001E+01 [†]	2.822E+02 \pm 2.983E+01 [†]	6.768E+01 \pm 4.275E+01 [‡]	8.104E+01 \pm 7.631E+01
	1,000,000	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00
f_6	1,000,000	1.788E-02 \pm 2.776E-03 [†]	5.489E-03 \pm 4.322E-03 [†]	2.854E-03 \pm 3.211E-04 [†]	1.993E-03 \pm 2.533E-04 [†]	7.344E-04 \pm 3.109E-03 [†]	3.864E-04 \pm 2.283E-04
f_7	800,000	3.624E-16 \pm 3.233E-16 [†]	2.854E-18 \pm 2.110E-19 [†]	4.377E-48 \pm 4.566E-49 [†]	6.231E-68 \pm 6.129E-68 [†]	1.248E-78 \pm 1.177E-78 [†]	7.206E-83 \pm 6.350E-84
f_9	2,000,000	3.121E+00 \pm 2.111E+00 [†]	2.551E+00 \pm 3.056E-01 [†]	9.376E-01 \pm 2.002E+00 [†]	8.775E-01 \pm 1.155E+00 [†]	8.675E-01 \pm 1.578E-01 [†]	5.208E-01 \pm 4.462E-01
f_{10}	1,000,000	-1.257E+04 \pm 6.324E-5 [‡]	-1.073E+04 \pm 7.985E+01 [‡]	-9.981E+04 \pm 5.576E+02 [†]	-9.176E+04 \pm 4.122E+02 [†]	-9.345E+04 \pm 3.642E+02 [†]	-1.001E+04 \pm 8.521E+01
f_{11}	1,200,000	3.122E-04 \pm 3.212E-03 [‡]	7.435E-03 \pm 2.887E-03 [‡]	2.024E-01 \pm 2.112E-02 [†]	3.514E-01 \pm 3.332E-02 [†]	2.140E-01 \pm 1.579E-02 [†]	8.729E-02 \pm 7.435E-02
f_{12}	200,000	7.645E-01 \pm 3.729E-01 [†]	6.013E-03 \pm 5.434E-04 [†]	8.332E-06 \pm 1.266E-06 [†]	3.133E-07 \pm 1.572E-07 [†]	3.411E-08 \pm 1.662E-09 [†]	6.811E-11 \pm 5.309E-10
	1,200,000	9.677E-14 \pm 1.032E-14 [†]	1.711E-14 \pm 8.323E-15 [†]	7.827E-15 \pm 5.011E-16 [†]	7.291E-15 \pm 0.000E+00 [†]	7.688E-15 \pm 0.000E+00 [†]	6.772E-15 \pm 0.000E+00

Table 5.5- Continued

Fun.	MAX-NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std
f_{13}	200,000	8.455E-01 \pm 4.766E-02 [†]	2.545E-03 \pm 7.612E-03 [†]	5.111E-04 \pm 1.456E-03 [†]	6.557E-09 \pm 7.122E-09 [†]	2.579E-12 \pm 5.051E-13 [†]	2.250E-16 \pm 1.354E-15
	1,200,000	0.000E+00 \pm 0.000E+00	7.112E-12 \pm 5.223E-13	5.125E-05 \pm 4.668E-04	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00	0.000E+00 \pm 0.000E+00
f_{14}	200,000	2.445E+00 \pm 2.311E-01 [†]	7.877E-06 \pm 3.221E-06 [†]	1.483E-11 \pm 2.334E-11 [†]	3.412E-14 \pm 3.566E-13 [†]	4.425E-16 \pm 4.032E-17 [†]	8.001E-19 \pm 8.221E-20
	1,200,000	2.980E-23 \pm 2.676E-23	5.455E-30 \pm 4.326E-30	4.706E-33 \pm 0.000E+00	4.706E-33 \pm 0.000E+00	4.706E-33 \pm 0.000E+00	4.706E-33 \pm 0.000E+00
f_{15}	200,000	2.112E+01 \pm 6.875E+00 [†]	5.213E-03 \pm 4.723E-03 [†]	2.603E-08 \pm 1.455E-09 [†]	1.304E-11 \pm 2.511E-12 [†]	8.592E-14 \pm 5.382E-14 [†]	3.471E-16 \pm 3.005E-16
	1,200,000	2.327E-22 \pm 1.456E-22	2.891E-27 \pm 2.356E-27	1.355E-32 \pm 0.000E+00	1.355E-32 \pm 0.000E+00	1.355E-32 \pm 0.000E+00	1.355E-32 \pm 0.000E+00
f_{16}	1,000,000	-3.629E+00 \pm 1.265E+04 [†]	-5.811E+00 \pm 3.453E+02 [†]	-5.520E+00 \pm 2.633E+02 [†]	-7.147E+01\pm 6.112E+03\pm	-1.629E+01 \pm 1.265E+04 [†]	-9.130E+01 \pm 8.702E+03
f_{17}	1,000,000	3.902E-01 \pm 3.831E-02 [†]	3.554E-01 \pm 5.044E-02 [†]	3.322E-01 \pm 4.245E-02 [†]	2.657E-01 \pm 2.835E-02 [†]	2.785E-01 \pm 2.131E-02 [†]	1.937E-01 \pm 1.119E-02
f_{18}	1,000,000	5.432E-03 \pm 5.433E-04 [†]	6.011E-03 \pm 2.776E-03 [†]	3.887E-11 \pm 4.108E-11 [†]	7.623E-06 \pm 4.002E-06 [†]	1.663E-19 \pm 1.645E-20 [†]	6.231E-24 \pm 5.292E-23
F_2	1,000,000	6.102E+01 \pm 7.932E+01 [†]	4.234E+02 \pm 8.190E+01 [†]	6.467E-12 \pm 5.524E-13 [†]	2.951E-13 \pm 2.721E-13 [†]	1.946E-14 \pm 2.478E-13 [†]	8.798E-15 \pm 8.532E-15
F_6	1,000,000	2.136E+01 \pm 4.102E-02 [†]	2.134E+01 \pm 5.332E-02 [†]	2.321E+01 \pm 4.433E-01 [†]	2.130E+01 \pm 1.761E-01 [†]	2.282E+01 \pm 3.965E+00 [†]	2.052E+01 \pm 1.256E-02
F_8	1,000,000	4.941E-03 \pm 2.312E-03 [†]	2.012E-02 \pm 2.002E-02 [†]	6.288E-03 \pm 5.211E-02 [†]	7.412E-03 \pm 7.541E-03 [†]	4.134E-03 \pm 5.121E-03 [†]	3.781E-03 \pm 7.811E-03
F_9	1,000,000	0.000E+00\pm 0.000E+00\pm	2.531E+01 7.313E+00 [†]	3.468E+00 2.329E+00 [†]	2.011E+00 \pm 2.857E+00	1.654E+00 2.020E+00 [†]	1.323E-02 2.000E+00
F_{10}	1,000,000	2.103E+02 \pm 3.345E+01 [†]	4.212E+02 \pm 5.921E+01 [†]	2.200E+02 \pm 1.548E+02 [†]	2.082E+02 \pm 3.437E+02 [†]	1.833E+02 \pm 1.243E+01 [†]	1.322E+02 \pm 1.254E+01

† indicates that ARDE-SPX performs better than other algorithms with 95% confidence level by *t*-test.
‡ indicates that the corresponding algorithm is better than ARDE-SPX.

Table 5.6: Mean and standard deviation of the low dimensional problems f_8 and $f_{19} - f_{28}$, averaged over 50 independent runs

Fun.	MAX_NFEs	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
		Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std	Mean \pm Std
f_8	10,000	5.561E-27 \pm 1.175E-28 [‡]	5.814E-05 \pm 8.632E-04 [‡]	1.322E-09 \pm 2.332E-10 [‡]	8.655E-10 \pm 2.877E-10 [‡]	2.805E-10 \pm 7.869E-10 [‡]	3.365E-12 2.481E-12
f_{19}	6,000	-1.000E+00 \pm 0.000E+00	-1.000E+00 \pm 0.000E+00	-1.000E+00 \pm 0.000E+00	-1.000E+00 \pm 0.000E+00	-1.000E+00 \pm 0.000E+00	-1.000E+00 \pm 0.000E+00
f_{20}	6,000	3.979E-01 \pm 3.324E-16	3.979E-01 \pm 3.324E-16	3.979E-01 \pm 3.324E-16	3.979E-01 \pm 3.324E-16	3.979E-01 \pm 1.129E-16	3.979E-01 \pm 3.324E-16
f_{21}	6,000	3.000E+00 \pm 0.000E+00	3.000E+00 \pm 0.000E+00	3.000E+00 \pm 0.000E+00	3.000E+00 \pm 0.000E+00	3.000E+00 \pm 0.000E+00	3.000E+00 \pm 0.000E+00
f_{22}	6,000	-1.032E+00 \pm 0.000E+00	-1.032E+00 \pm 0.000E+00	-1.032E+00 \pm 0.000E+00	-1.032E+00 \pm 0.000E+00	-1.032E+00 \pm 0.000E+00	-1.032E+00 \pm 0.000E+00
f_{23}	6,000	9.980E-01 \pm 3.388E-16	9.980E-01 \pm 3.388E-16	9.980E-01 \pm 3.388E-16	9.980E-01 \pm 3.388E-16	9.980E-01 \pm 3.388E-16	9.980E-01 \pm 3.388E-16
f_{24}	6,000	-3.863E+00 \pm 0.000E+00	-3.863E+00 \pm 2.788E-15	-3.863E+00 \pm 0.000E+00	-3.863E+00 \pm 1.277E-16	-3.863E+00 \pm 2.710E-15	-3.863E+00 \pm 0.000E+00
f_{25}	6,000	-3.268E+00 \pm 4.842E-02	-3.322E+00 \pm 2.771E-02	-3.310E+00 \pm 3.557E-02	-3.311E+00 \pm 4.001E-02	-3.282E+00 \pm 5.607E-02	-3.322E+00 \pm 1.223E-02
f_{26}	6,000	-1.0153E+01 \pm 1.983E-12 [‡]	-1.0052E+01 \pm 6.886E-02 [‡]	-1.0153E+01 \pm 3.988E-14 [‡]	-9.788E+00 \pm 1.556E+00 [‡]	-1.012E+00 \pm 2.691E+00 [‡]	-1.013E+00 \pm 3.542E+00
f_{27}	6,000	-1.0402E+01 \pm 2.654E-15	-1.0402E+01 \pm 6.554E-12	-1.0402E+01 \pm 8.955E-16	-1.0402E+01 \pm 1.998E-12	-1.0402E+01 \pm 1.369E-15	-1.0402E+01 \pm 8.955E-13
f_{28}	6,000	-1.0536E+01 \pm 7.133E-16	-1.0536E+01 \pm 7.982E-05	-1.0536E+01 \pm 7.833E-12	-1.0536E+01 \pm 5.664E-14	-1.0536E+01 \pm 2.205E-15	-1.0536E+01 \pm 5.664E-17

Moreover, the local search represented by the SPX crossover plays an important role in improving the quality of the solutions in the population by replacing the worst solution found so far in the current population with the one generated by the SPX crossover which is most likely to be better than its previous one.

5.3.2.2 Convergence Speed and Robustness (FESS, S_r)

In Tables 5.7 and 5.8, the success rate (S_r) and the average number of function evaluations over successful runs (FESS) of each adaptive DE algorithm have been summarized at $D = 30$ and $D = 100$. S_r and FESS are useful to compare the reliability and the convergence velocity of any proposed algorithm, respectively.

From these two tables, it can be seen that ARDE-SPX requires the minimum FESS to reach the Ter_Err on the majority of the test functions compared with the other adaptive DE variants. ARDE-SPX also obtains the greatest overall success rate measured by $\sum S_r = \frac{\sum_{i=1}^{nf} (S_r \text{ in } \%)}{100}$ where nf is the total number of the test functions which is in our experiment = 22. At the problem dimension $D = 30$, the ARDE-SPX has achieved $\sum S_r = 18.70$ and $\sum S_r = 14.85$ at the problem dimension $D = 100$.

The difference between ARDE-SPX and the other adaptive DE variants in terms of the FESS and the convergence rate S_r is significant and it shows the reliability and robustness of the proposed ARDE-SPX algorithm. This is due to the different mutation and crossover strategies of DE as well as the diversity in employing different distributions in the parameter control schemes represented by the Normal and Cauchy distributions that result to guide the search to better directions within minimum cost and time.

Table 5.7: Mean of the NFEs required to obtain the accuracy level Ter_Err and success rate S_r for 50-independent runs of the 30-dimensional problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$

Fun.	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)
f_1	5.881E+04 (100)	4.223E+04 (100)	2.911E+04 (100)	3.113E+04 (100)	2.381E+04 (100)	2.032E+04 (100)
f_2	7.983E+04 (100)	7.192E+04 (100)	4.778E+04 (100)	5.563E+04 (100)	4.012E+04 (100)	3.892E+04 (100)
f_3	3.211E+05 (100)	3.087E+05 (100)	8.778E+04 (100)	7.334E+04 (100)	8.532E+04 (100)	5.954E+04 (100)
f_4	4.001E+05 (100)	1.577E+05 (100)	2.032E+05 (100)	2.311E+04 (100)	3.243E+05 (100)	2.122E+04 (100)
f_5	2.213E+04 (100)	1.643E+04 (100)	1.102E+04 (100)	1.326E+04 (100)	8.764E+03 (100)	8.371E+03 (100)
f_6	1.100E+05 (100)	5.340E+04 (100)	2.866E+04 (100)	3.011E+04 (100)	2.258E+04 (100)	2.558E+04 (100)
f_7	6.233E+04 (100)	3.667E+04 (100)	3.021E+04 (100)	3.412E+04 (100)	2.611E+04 (100)	2.300E+04 (100)
f_9	4.021E+05 (25)	2.722E+05 (70)	1.533E+05 (100)	1.232E+05 (100)	1.201E+05 (100)	1.0233E+05 (100)
f_{10}	9.071E+04 (100)	9.342E+04 (100)	1.277E+05 (100)	1.131E+05 (95)	1.128E+05 (100)	1.059E+05 (100)
f_{11}	1.187E+05 (100)	1.421E+05 (100)	1.316E+05 (100)	1.342E+05 (100)	1.304E+05 (100)	1.335E+05 (100)
f_{12}	9.110E+04 (100)	5.765E+04 (100)	4.544E+04 (100)	7.680E+04 (100)	3.476E+04 (100)	2.871E+04 (100)
f_{13}	6.432E+04 (100)	5.102E+04 (100)	3.223E+04 (100)	3.655E+04 (100)	2.652E+04 (100)	2.492E+04 (100)
f_{14}	5.437E+04 (100)	4.432E+04 (100)	2.698E+04 (100)	2.920E+04 (100)	2.541E+04 (100)	2.531E+04 (100)
f_{15}	6.230E+04 (100)	4.549E+04 (100)	3.000E+04 (100)	3.207E+04 (100)	2.833E+04 (100)	2.581E+04 (100)
f_{16}	NA (0)	NA (0)	NA (0)	2.210E+05 (100)	2.339E+05 (100)	2.250E+05 (100)
f_{17}	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
f_{18}	2.178E+05 (100)	2.544E+05 (15)	2.9211E+05 (20)	2.872E+05 (20)	1.612E+05 (60)	1.483E+05 (85)
F_2	2.022E+05 (90)	2.097E+05 (100)	1.232E+5 (100)	1.038E+5 (100)	9.998E+04 (100)	8.781E+04 (100)
F_6	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
F_8	1.211E+05 (55)	1.235E+05 (65)	4.426E+04 (70)	3.501E+04 (78)	3.433E+04 (70)	2.850E+04 (85)
F_9	8.421E+04 (100)	2.021E+05 (100)	9.549E+04 (100)	1.143E+05 (100)	1.125E+05 (100)	1.110E+05 (100)
F_{10}	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
$\sum S_r$	16.70	16.50	16.90	17.93	18.30	18.70

Table 5.8: Mean of the NFEs required to obtain the accuracy level Ter_Err and success rate S_r for 50-independent runs of the 100-dimensional problems $f_1 - f_7 ; f_9 - f_{18}; F_2, F_6, F_8 - F_{10}$

Fun.	jDE	SaDE	JADE wo	JADE w	SaJADE	ARDE-SPX
	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)	FESS (Sr%)
f_1	5.232E+05 (100)	4.122E+05 (100)	2.102E+05 (100)	1.677E+05 (100)	1.412E+05 (100)	1.282E+05 (100)
f_2	8.100E+05 (100)	6.032E+05 (100)	2.988E+05 (100)	2.564E+05 (100)	2.304E+05 (100)	2.209E+05 (100)
f_3	NA (0)	1.712E+06 (95)	1.431E+06 (100)	9.458E+05 (100)	7.112E+05 (100)	6.521E+05 (100)
f_4	1.211E+06 (2)	5.766E+05 (80)	1.122E+06 (20)	4.322E+05 (91)	7.533E+05 (85)	3.919E+05 (95)
f_5	2.232E+05 (100)	1.432E+05 (100)	1.097E+05 (100)	3.125E+04 (100)	3.243E+04 (100)	3.311E+04 (100)
f_6	1.100E+05 (100)	1.568E+05 (100)	3.021E+04 (100)	3.620E+04 (100)	2.241E+04 (100)	2.055E+04 (100)
f_7	5.587E+05 (100)	5.022E+05 (100)	2.780E+05 (100)	2.512E+05 (100)	1.713E+05 (100)	1.540E+05 (100)
f_9	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
f_{10}	5.342E+05 (100)	8.501E+05 (100)	8.340E+05 (100)	8.546E+05 (100)	8.012E+05 (100)	6.821E+05 (95)
f_{11}	1.159E+06 (2)	NA (0)	NA (0)	NA (0)	NA (0)	1.192E+06 (10)
f_{12}	8.021E+05 (100)	6.122E+05 (100)	2.780E+05 (100)	2.551E+05 (100)	2.190E+05 (100)	1.861E+05 (100)
f_{13}	5.311E+05 (100)	4.021E+05 (98)	1.850E+05 (95)	1.743E+05 (100)	1.545E+05 (100)	1.322E+05 (100)
f_{14}	5.329E+04 (100)	3.863E+04 (100)	1.556E+04 (100)	1.374E+04 (100)	1.413E+04 (100)	1.301E+04 (100)
f_{15}	6.129E+04 (100)	4.095E+04 (100)	1.908E+04 (100)	1.655E+04 (100)	1.561E+04 (100)	1.221E+04 (100)
f_{16}	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
f_{17}	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
f_{18}	7.541E+05 (75)	8.145E+05 (70)	4.343E+05 (100)	3.632E+05 (90)	3.472E+05 (100)	2.810E+05 (100)
F_2	NA (0)	NA (0)	9.475E+05 (87)	8.895E+05 (88)	7.822E+05 (90)	7.120E+05 (92)
F_6	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
F_8	4.341E+05 (77)	5.632E+05 (63)	1.670E+05 (88)	1.425E+05 (92)	1.791E+05 (88)	1.353E+05 (93)
F_9	4.971E+05 (97)	NA (0)	NA (0)	NA (0)	NA (0)	5.632E+05 (35)
F_{10}	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)	NA (0)
ΣS_r	12.53	13.06	13.90	14.61	14.63	14.85

5.3.2.3 Convergence Plot

In this subsection, a comparison on the convergence performance of the six algorithms (jDE, SaDE, JADEwo, JADEw, SaJADE, and ARDE-SPX) is conducted using the convergence graph for nine benchmark functions ($f_1, f_2, f_6, f_9, f_{12}, f_{15}, f_{18}, F_2, F_9$) with 30-dimension and 100-dimension problems. Because the convergence graphs of most of the test functions are similar in their characteristics, these nine functions have been selected as representative instances. These graphs illustrate the convergence characteristics in terms of the best fitness value of the run of each algorithm. In addition, the evolution trend of the μ_F and μ_{CR} is also illustrated.

Figure 5.3 illustrates the performance of the six algorithms for nine 30-dimensional benchmark functions. From this figure, it can be noted that ARDE-SPX has the best performance in the convergence speed for the functions ($f_1, f_6, f_9, f_{12}, f_{15}, F_2, F_9$), and the second best performance after SaJADE and jDE for the functions f_2 and f_{18} , respectively.

Figure 5.4 illustrates the performance of the six algorithms for nine 100-dimensional benchmark functions. From this figure, it can be noted that ARDE-SPX has the best performance in the convergence speed for the functions ($f_1, f_2, f_6, f_9, f_{12}, f_{15}, f_{18}, F_2$), and the second best performance after jDE for the function F_9 .

As can be seen from these figures, the importance of ARDE-SPX appears significantly in the high dimension problems. The convergence rate of ARDE-SPX in 100-dimensional problems is relatively high compared with the other adaptive DE; except in the case of F_9 where jDE performs the best. This is due to the high randomness possessed by jDE which can make this algorithm escape from the local entrapment.

Finally, Figure 5.5 depicts the evolution trend of the parameters F_m and CR_m on some selected functions with the mean curve. It clearly illustrates the adaptation characteristics of ARDE-SPX.

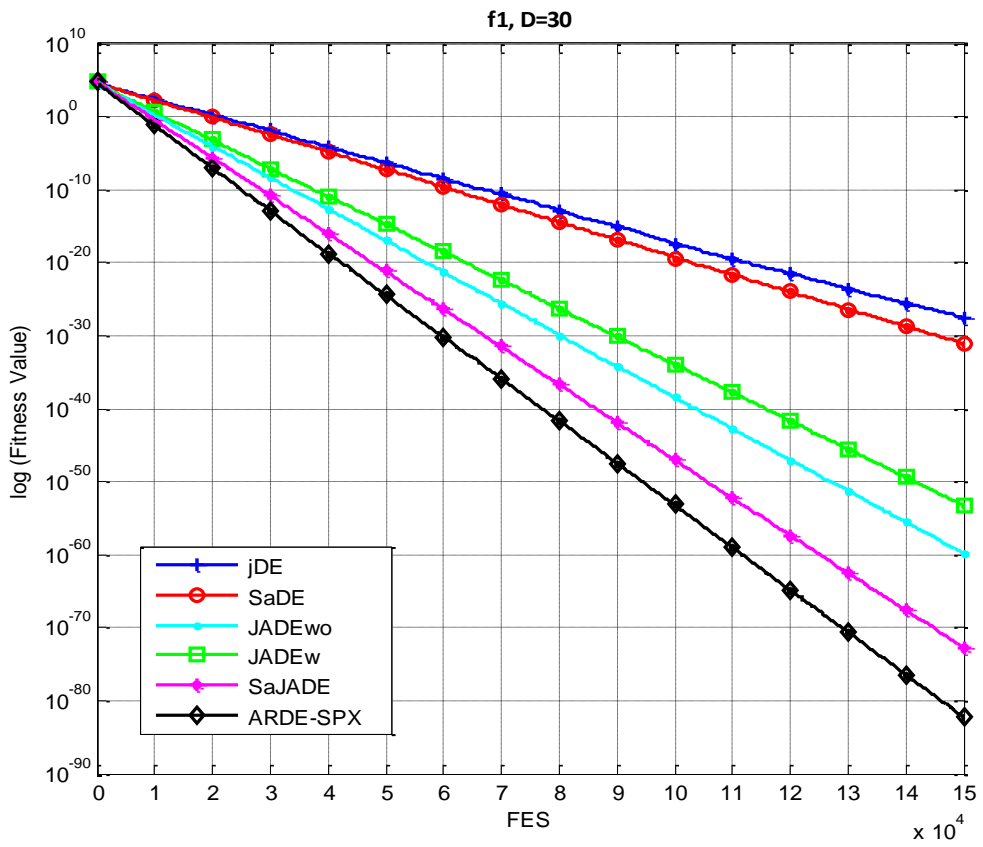


Figure 5.3-(a)

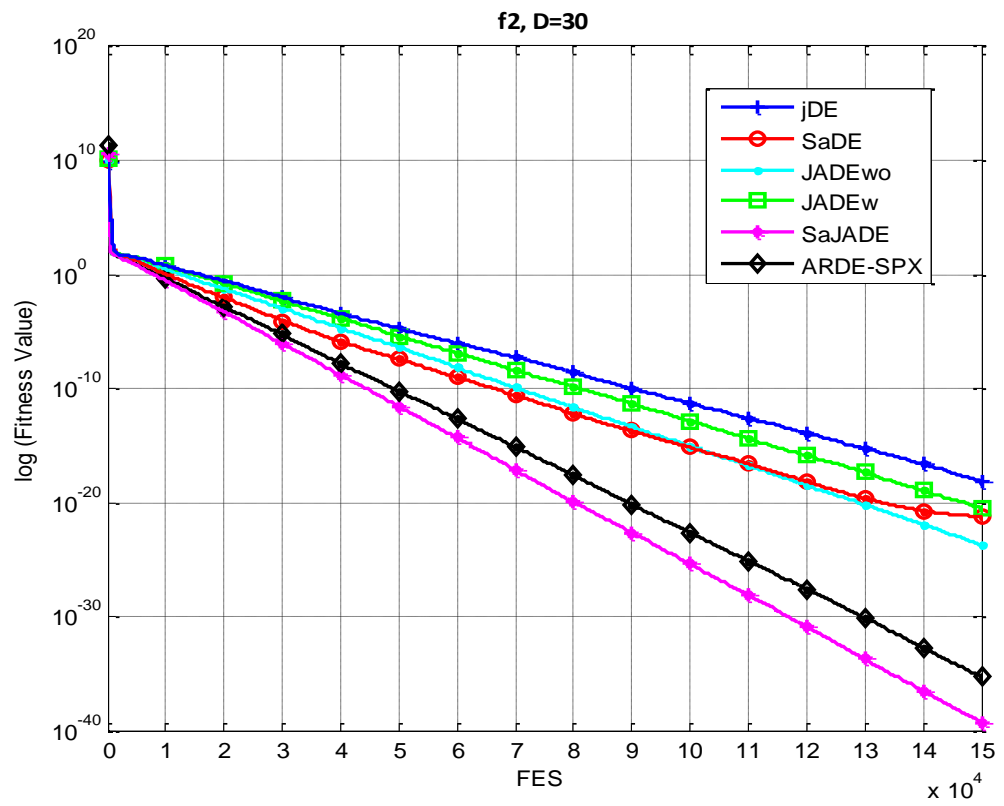


Figure 5.3-(b)

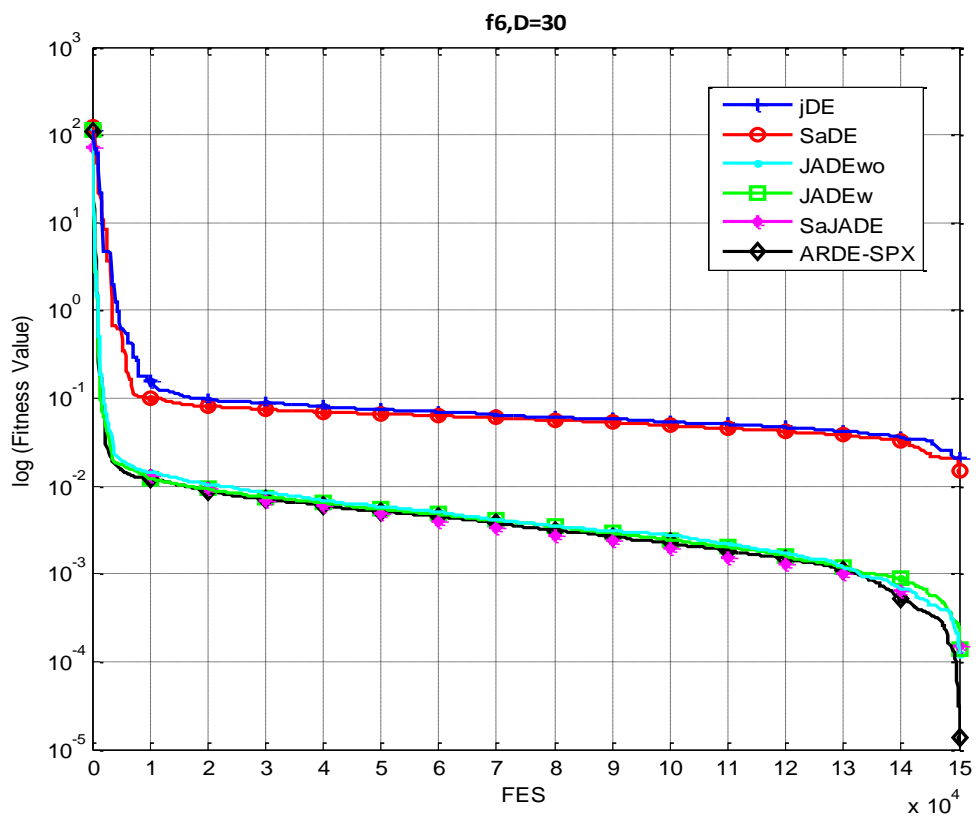


Figure 5.3-(c)

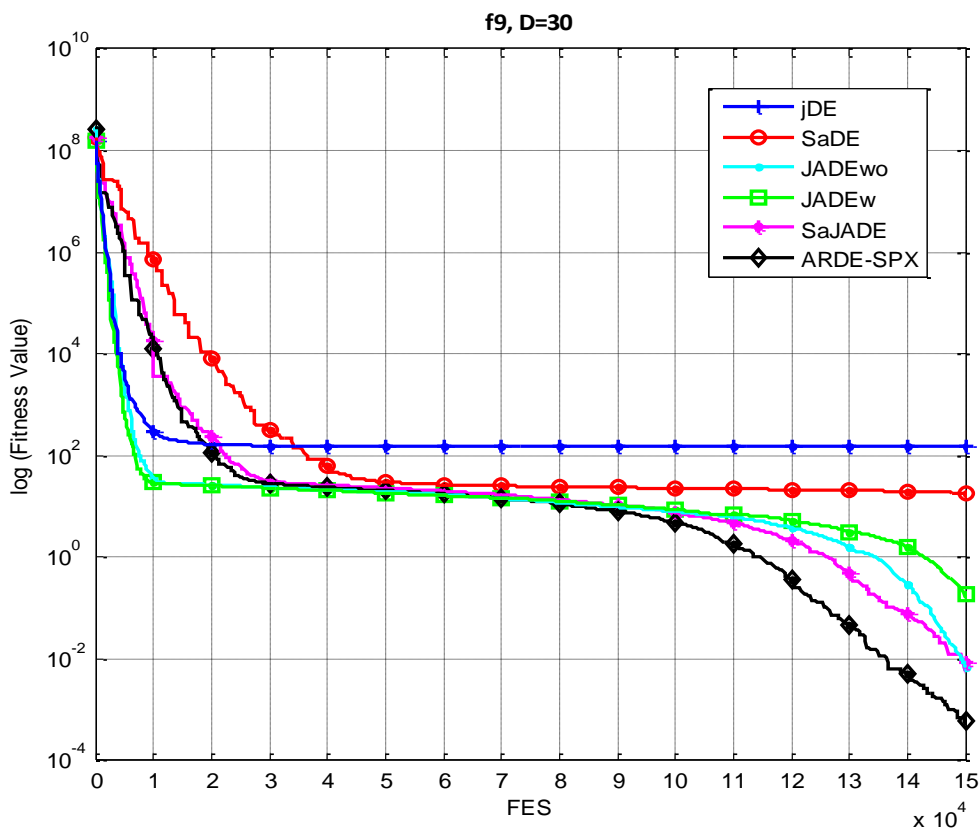


Figure 5.3-(d)

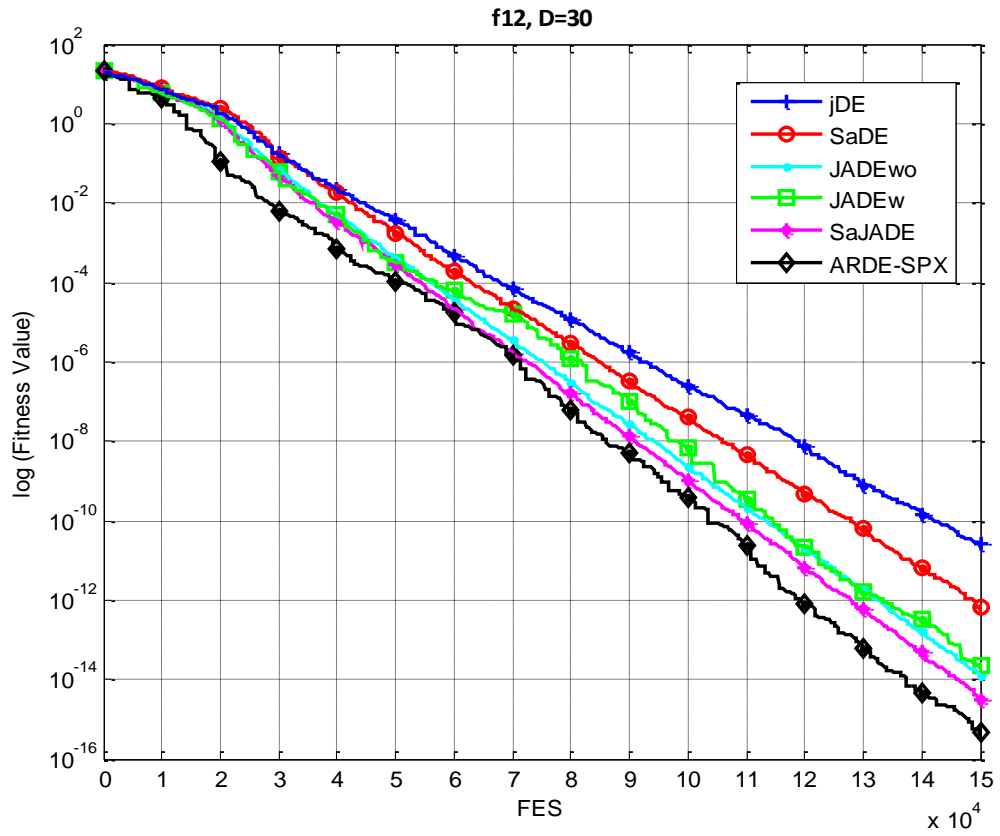


Figure 5.3- (e)

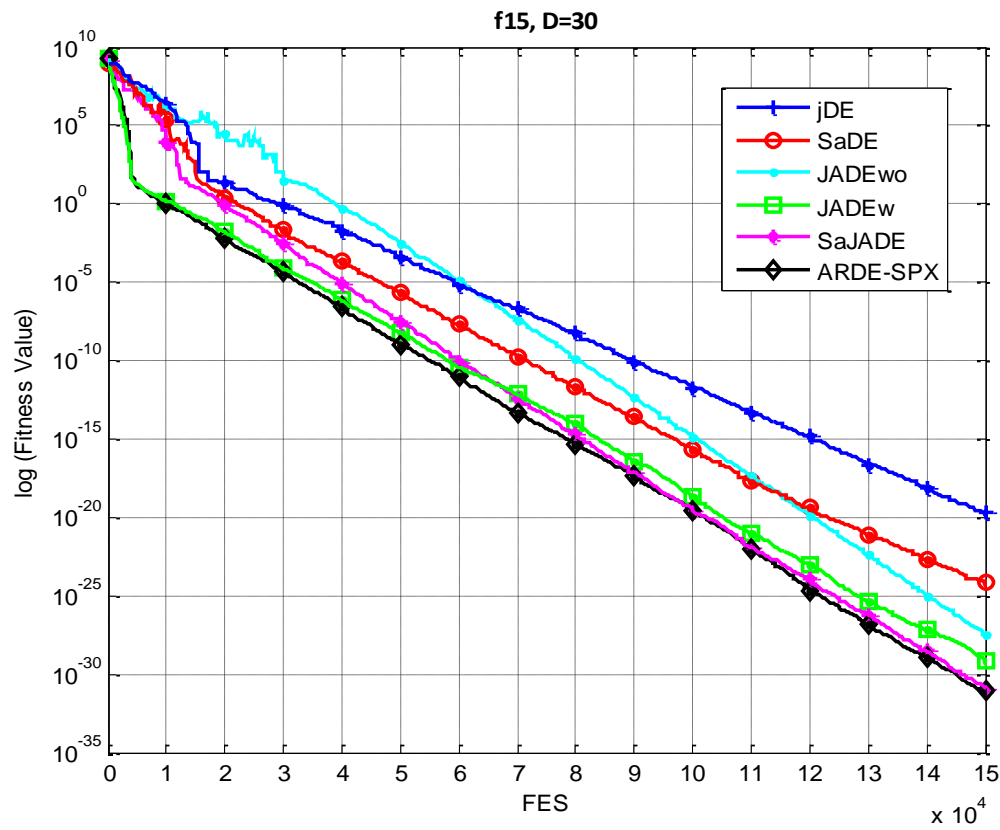


Figure 5.3- (f)

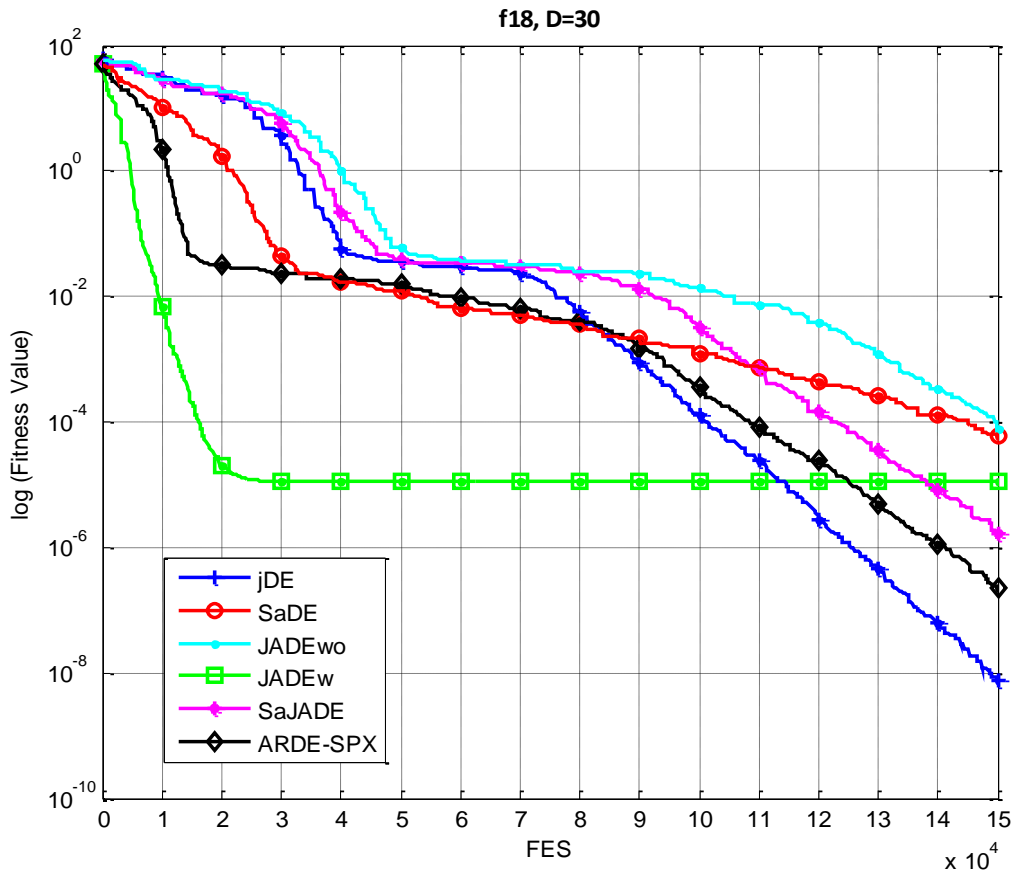


Figure 5.3- (g)

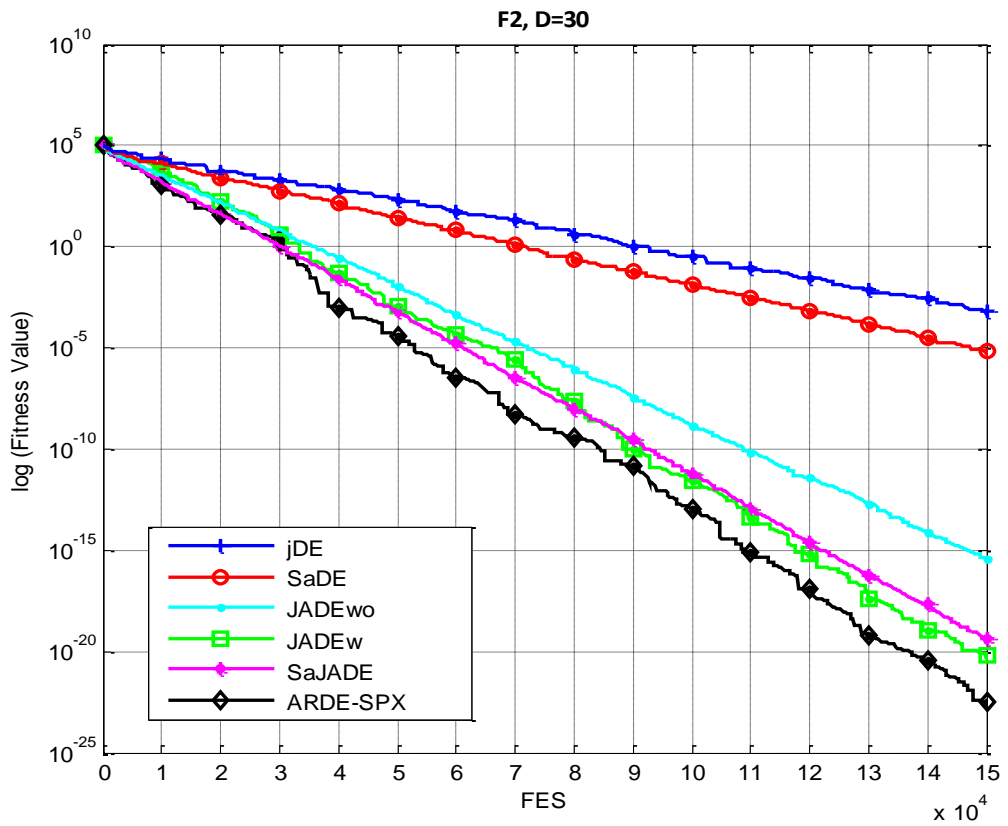


Figure 5.3- (h)

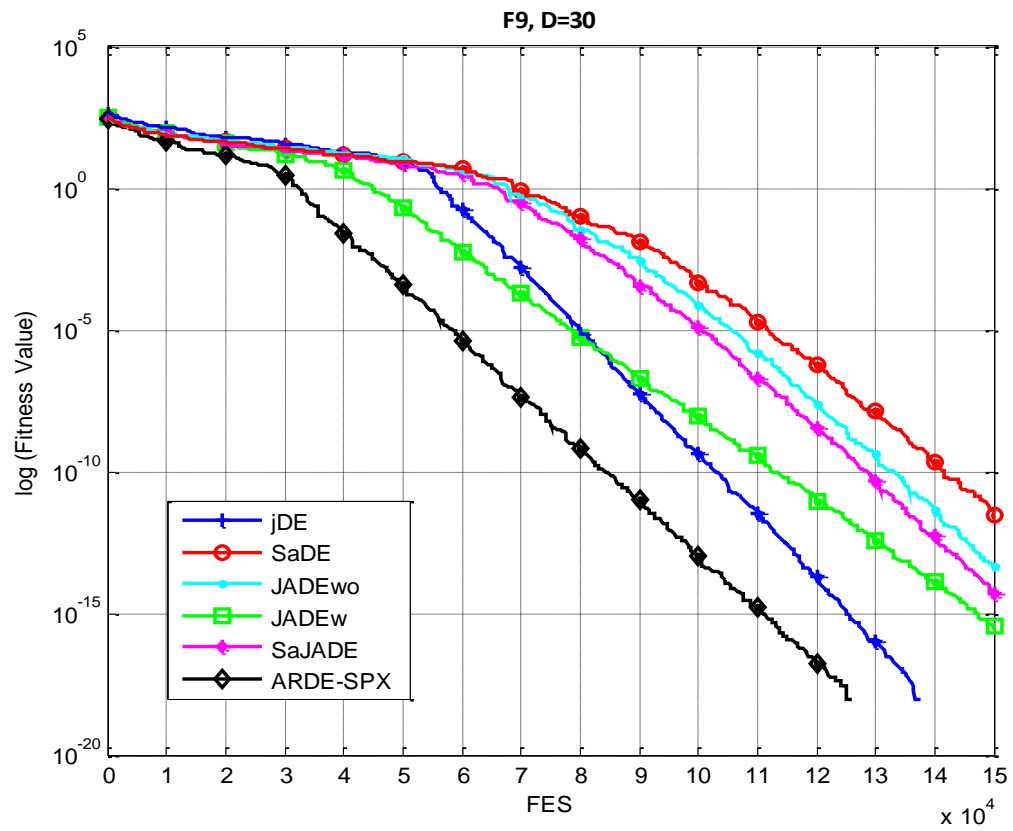


Figure 5.3- (i)

Figure 5.3: Convergence performance of the algorithms for nine 30-dimensional functions. (a) f_1 . (b) f_2 . (c) f_6 . (d) f_9 . (e) f_{12} . (f) f_{15} . (g) f_{18} . (h) F_2 . (i) F_9 .

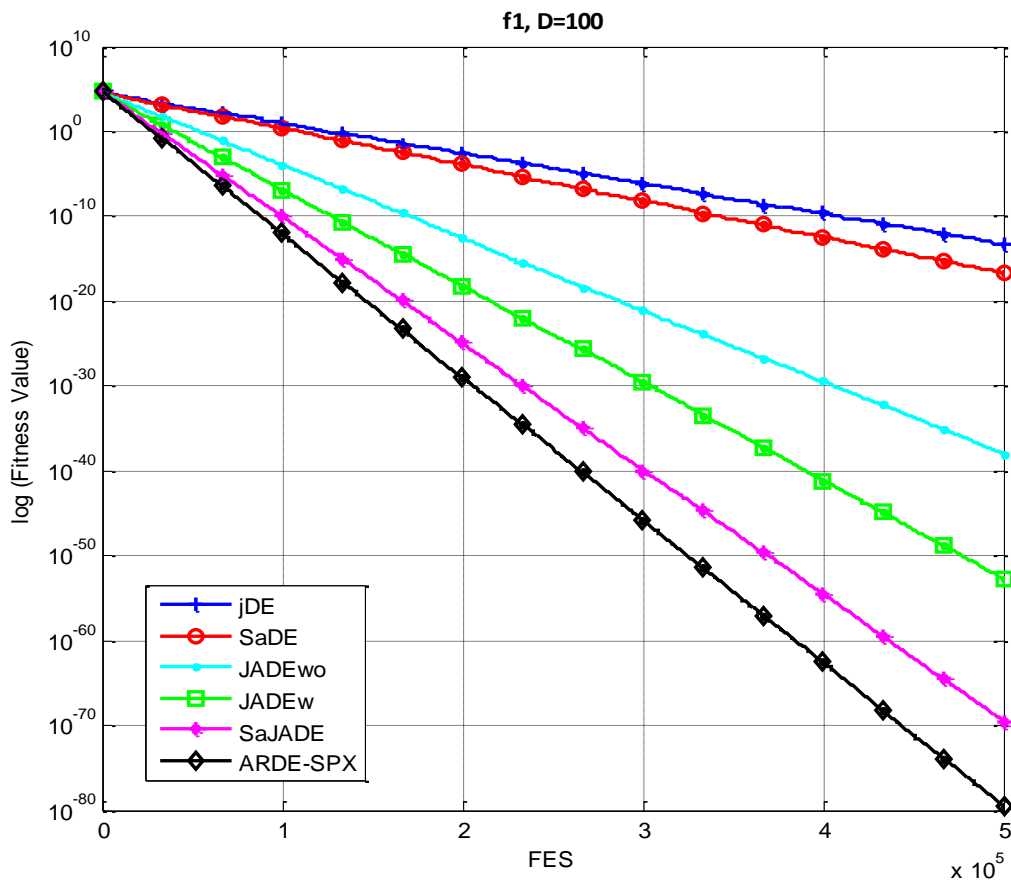


Figure 5.4-(a)

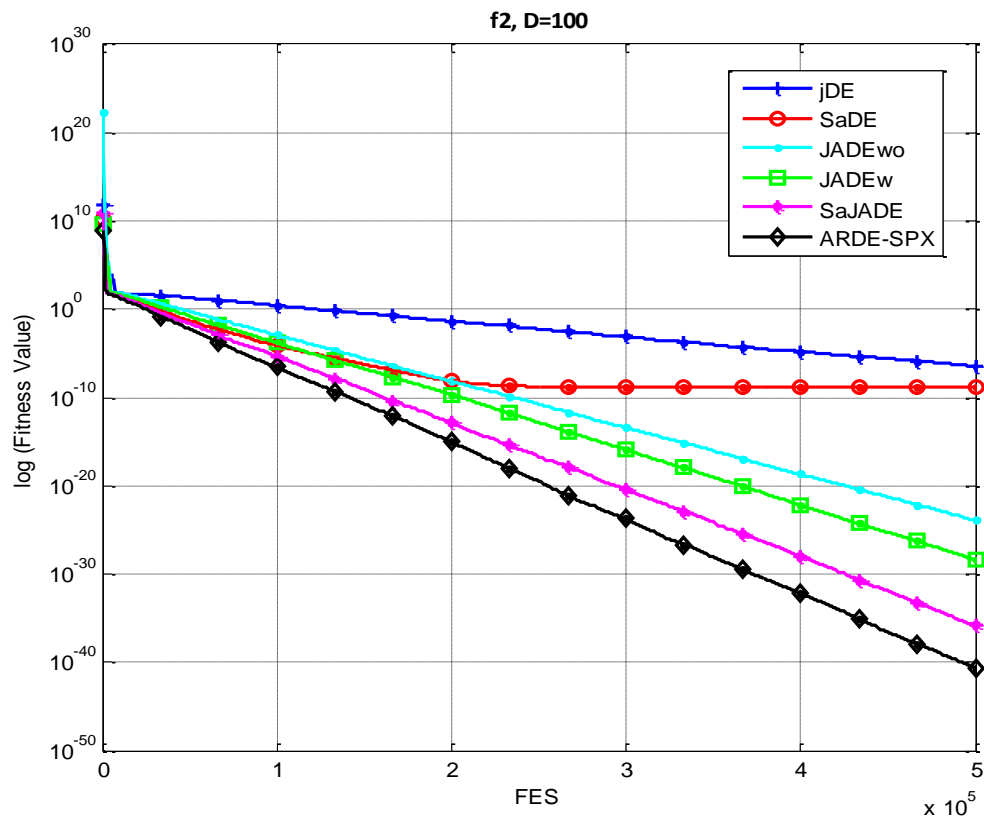


Figure 5.4-(b)

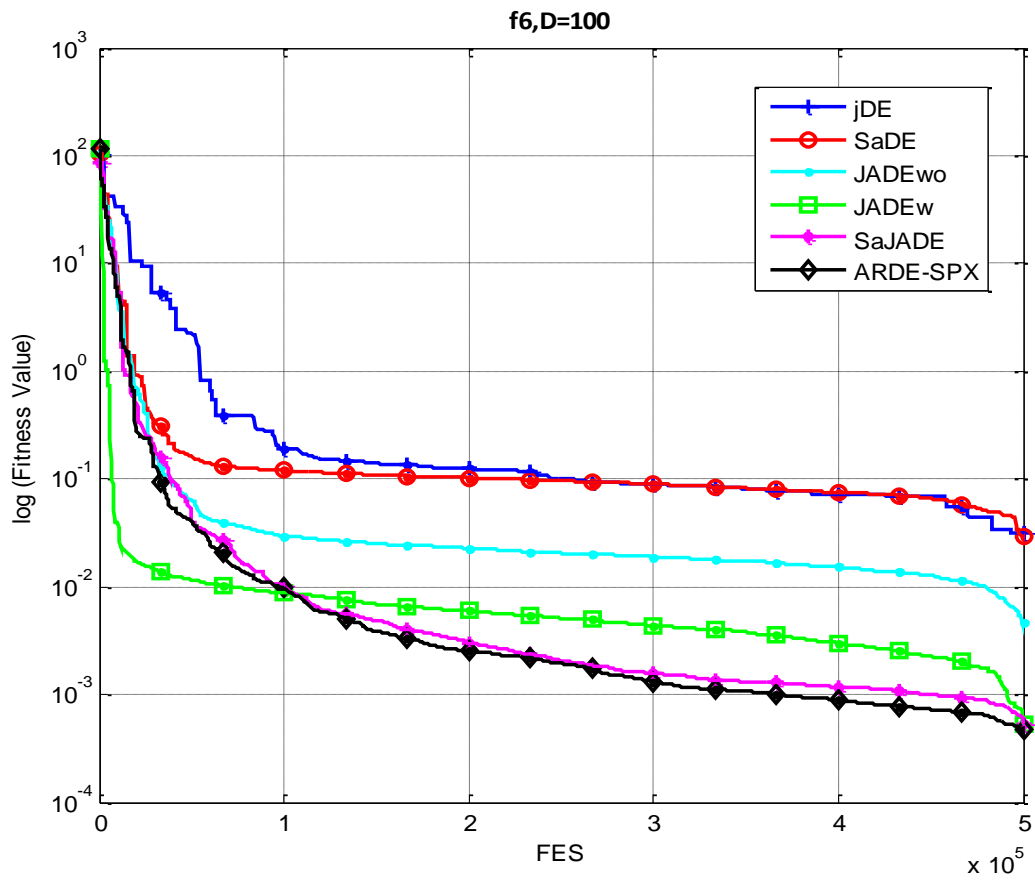


Figure 5.4-(c)

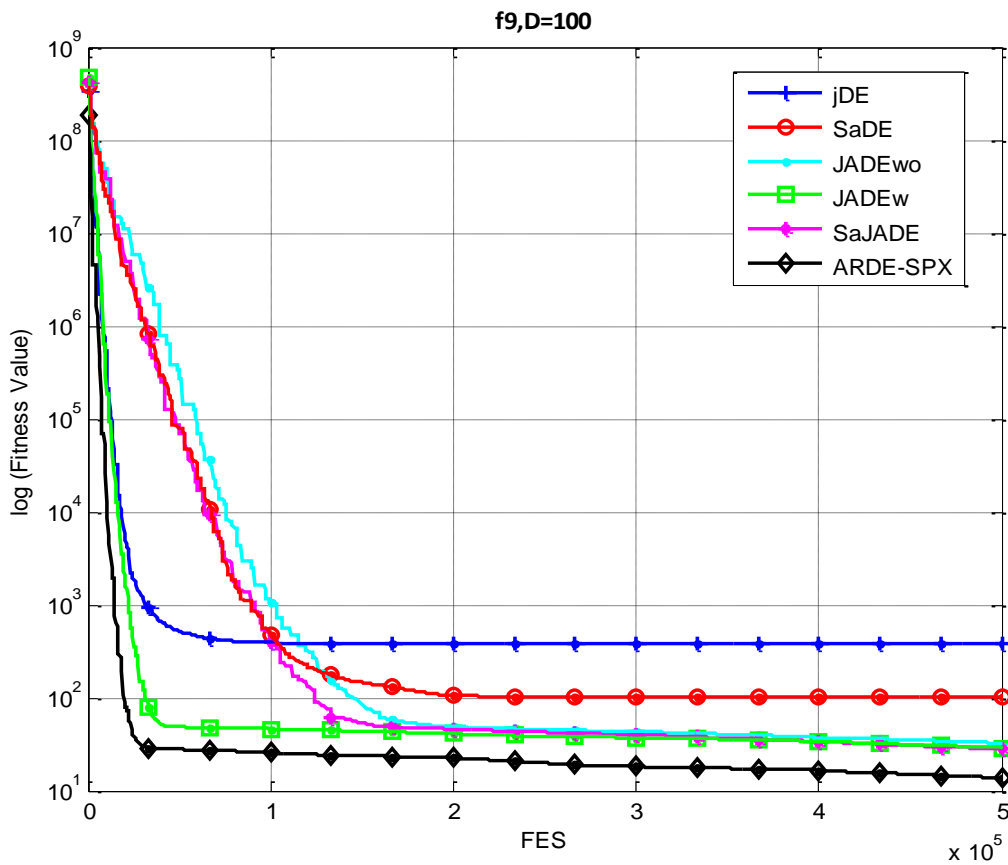


Figure 5.4-(d)

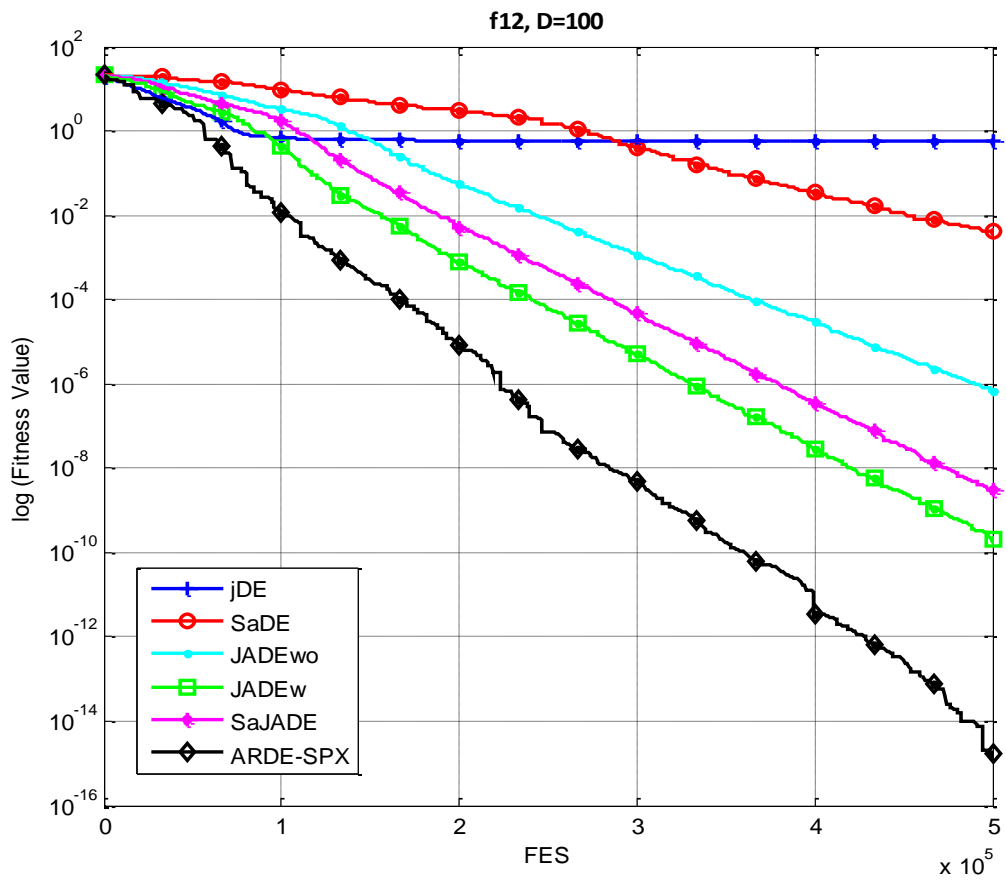


Figure 5.4-(e)

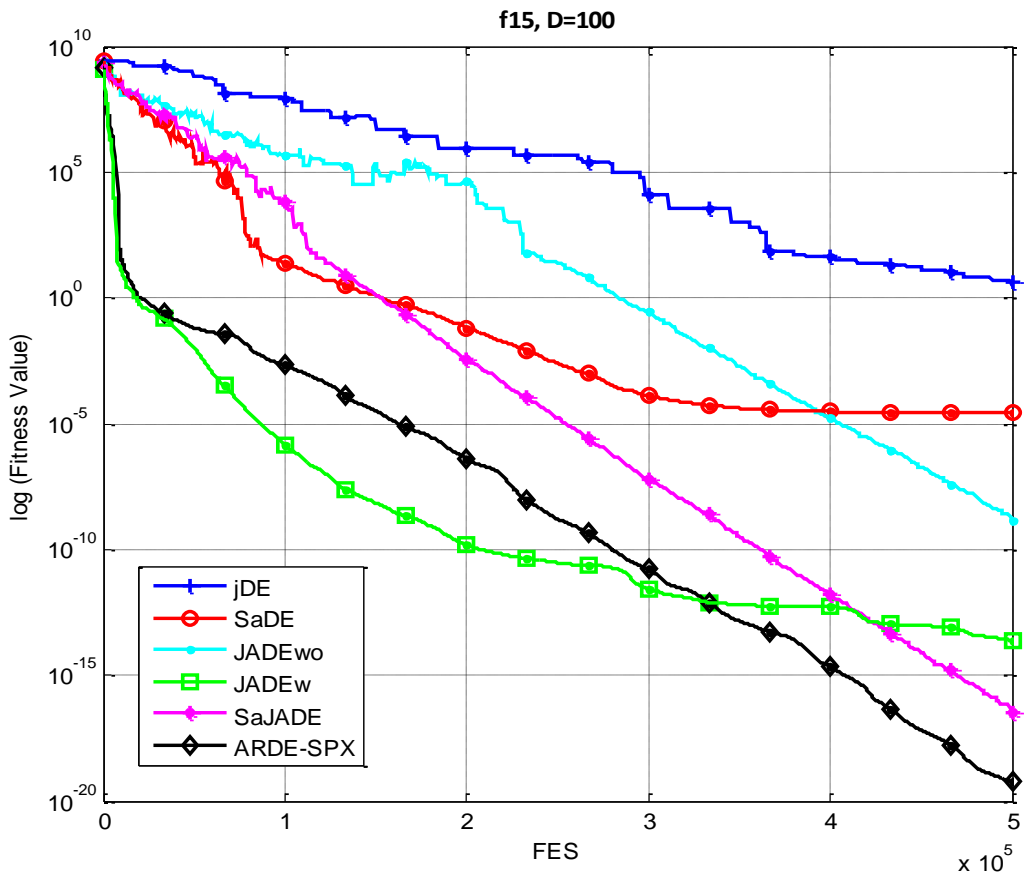


Figure 5.4-(f)

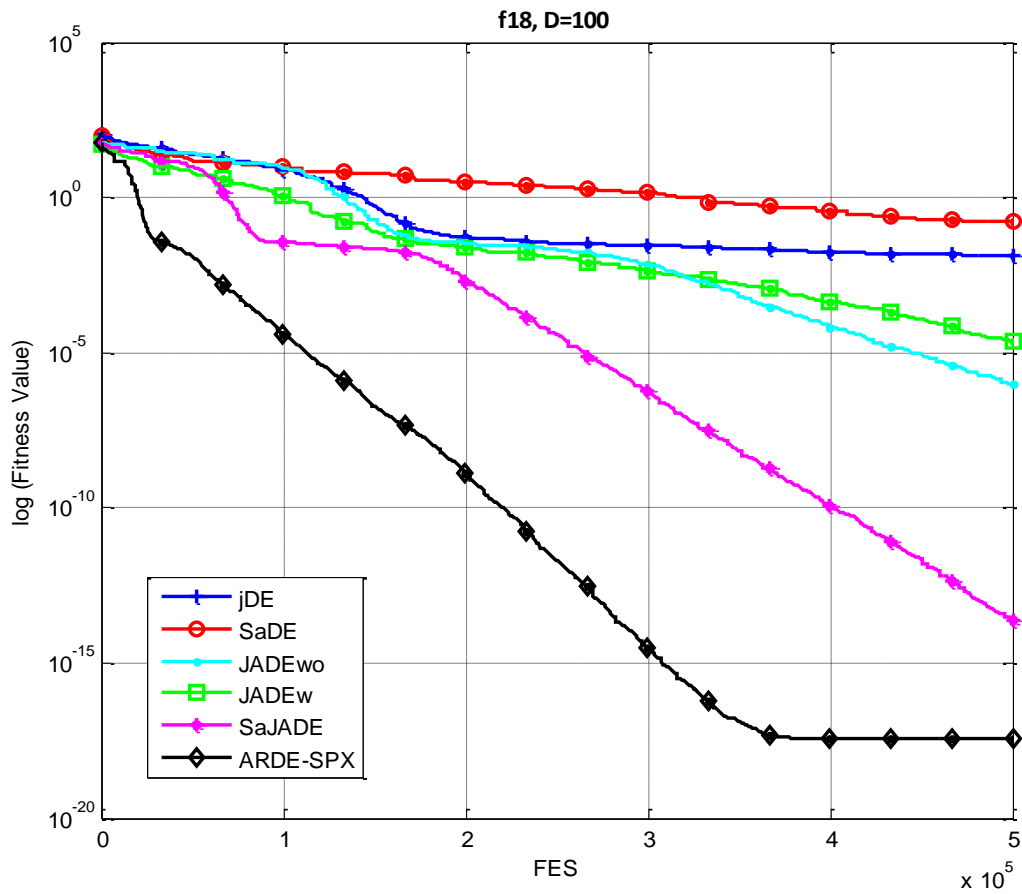


Figure 5.4-(g)

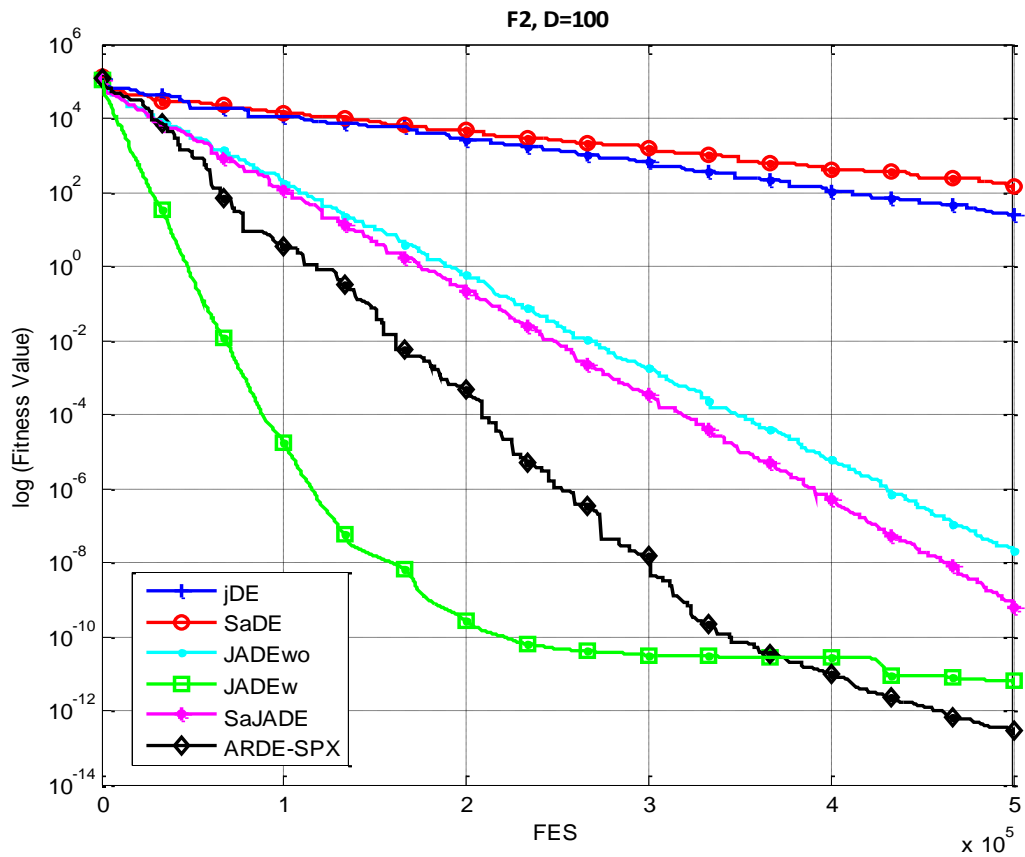


Figure 5.4-(h)

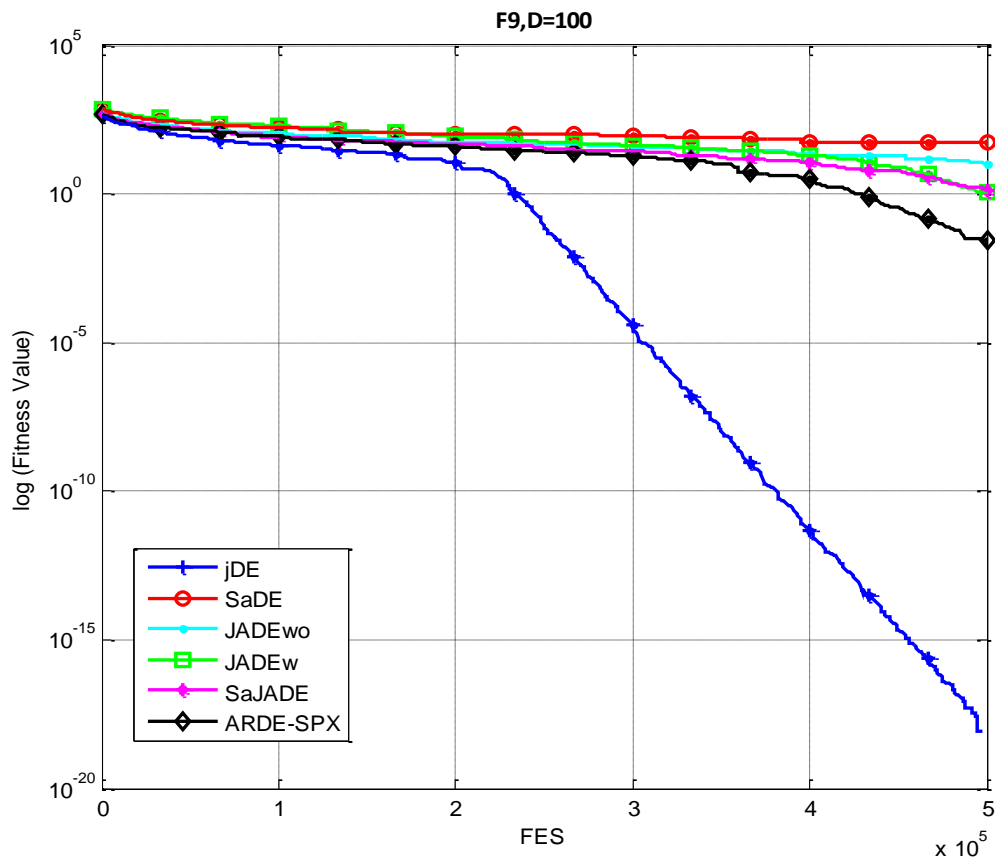


Figure 5.4-(i)

Figure 5.4: Convergence performance of the algorithms for nine 100-dimensional functions. (a) f_1 . (b) f_2 . (c) f_6 . (d) f_9 . (e) f_{12} . (f) f_{15} . (g) f_{18} . (h) F_2 . (i) F_9 .

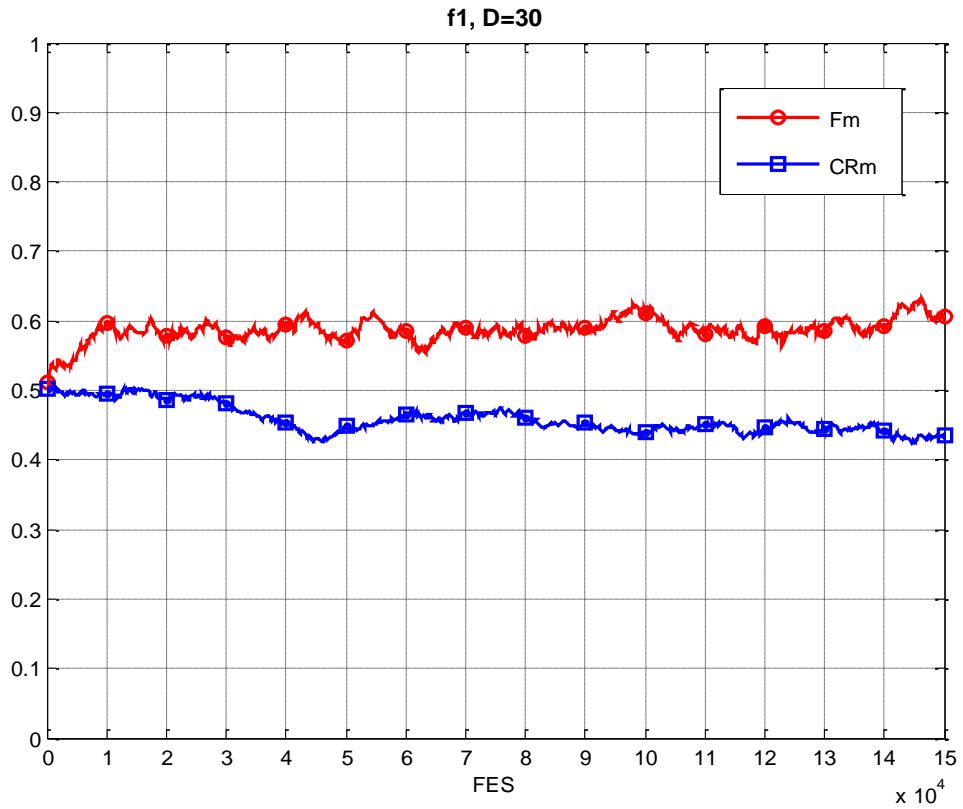


Figure 5.5-(a)

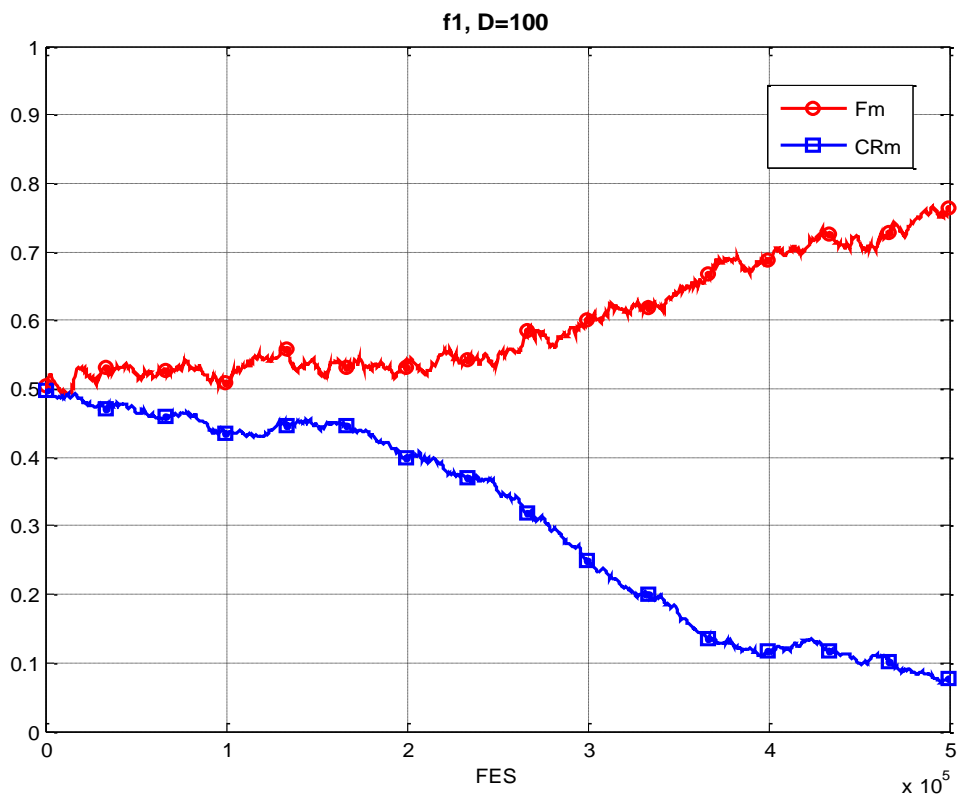


Figure 5.5-(b)

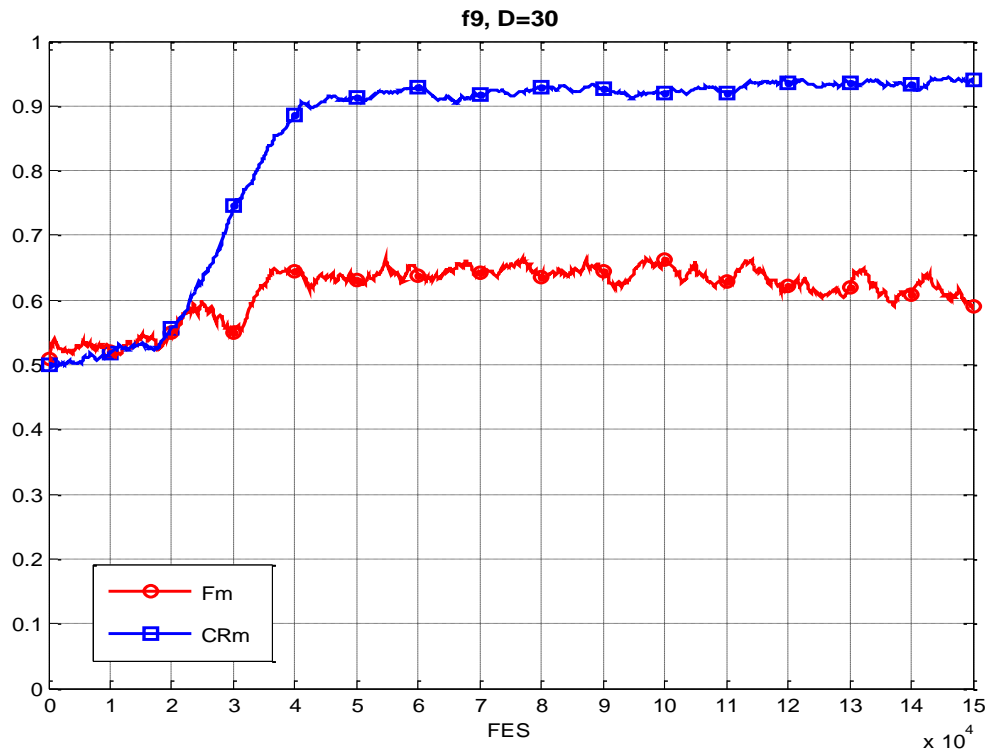


Figure 5.5-(c)

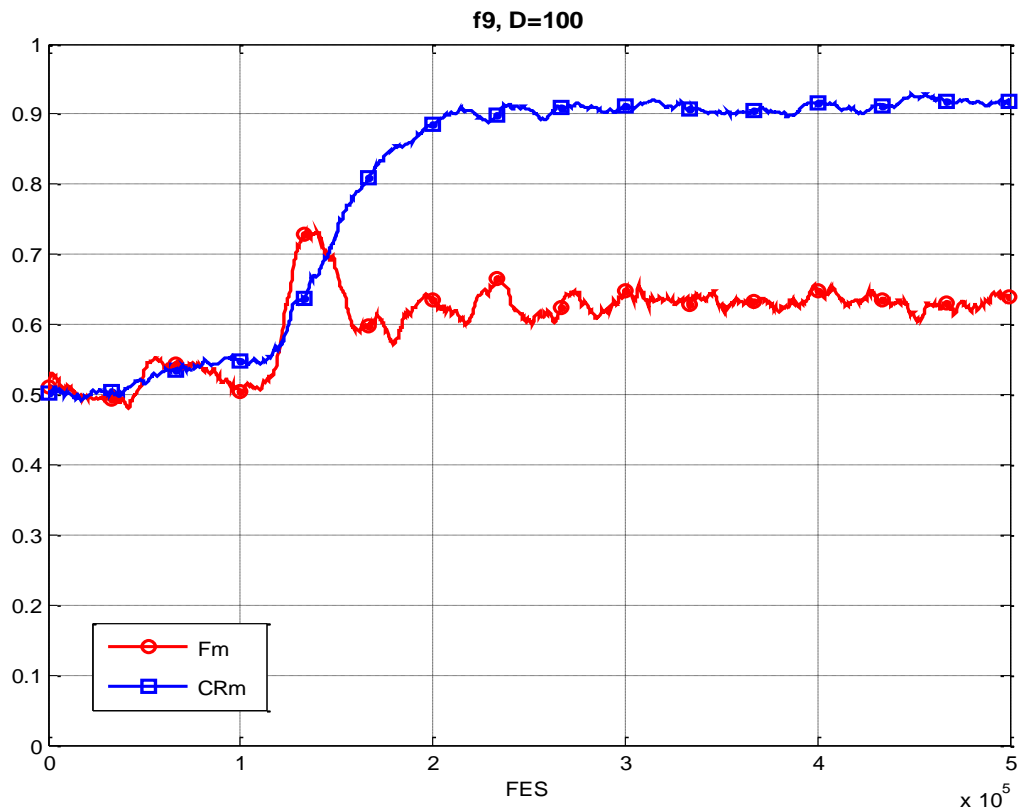


Figure 5.5-(d)

Figure 5.5: Adaptation characteristics of F_m and CR_m on the selected functions. (a) $f_1(D = 30)$. (b) $f_1(D = 100)$. (c) $f_9(D = 30)$. (d) $f_9(D = 100)$

5.4 Summary

In this chapter, two performance comparisons have been presented.

- 1) The first comparison is established among different DE mutation schemes (DE/rand/1, DE/current-to-rand/1, DE/best/1, DE/current-to-best/1, DE/current-to- p Best/1 without archive and DE/current-to- p Best/1 with archive) in terms of the solution precision over 28 benchmark functions. In this comparison, a table of F and CR parameter settings is provided where each DE scheme could perform the best over the different test problems. This table can be used by the practitioners when they want to tune the parameters of DE to perform well on optimization problems with different characteristics and dimensionality with less time and effort.
- 2) The second comparison is established among ARDE-SPX with five significant state-of-the-art adaptive DE variants (jDE, SaDE, JADE with no archive, JADE with archive, and SaJADE) over 33 test problems with different characteristics (unimodal, multimodal, shifted, rotated, etc.) and different dimensionality (6-D, 30-D, and 100-D). This comparison has been implemented in terms of the solution precision, robustness, and convergence speed. All the tests conducted could lead to one conclusion that ARDE with its incorporated local search SPX crossover can outperform the other adaptive DE variants on a wide variety of objective functions; while at the same time it enhances the ability of the standard DE to accurately find good solutions in the search space during optimization.

CHAPTER 6

SYSTEM IDENTIFICATION AND CONTROL OF ROBOT MANIPULATOR BASED ON ARDE ALGORITHM

6.1 Introduction

A requirement for new robotic manipulators is the ability to detect and manipulate objects in their environments. Robotic manipulators are highly nonlinear systems, and an accurate mathematical model is difficult to obtain using conventional techniques. Therefore, an efficient technique is required to deal with these types of complex and dynamic systems. The objective of this chapter is to develop a new dynamic parameter identification framework to estimate the barycentric parameters of the CRS A456 robot manipulator based on ARDE algorithm. The simulation results presented in this chapter show the effectiveness of the ARDE method over other conventional techniques, transcending the limits of the existing state-of-the-art algorithms in solving the problem of robot.

6.2 Research Background

There are many industrial applications where the robot manipulator is required to carry out precise task with high accuracy and repeatability. Recently, the application of robotic technology in clinical medicine has been a very active research area. For instance, in surgical operations the robot manipulator serves as an assistant to the doctor or as an extension of the doctor capabilities (Gomes, 2011; Pislă, Gherman, Vaida, Suciū, & Plitea, 2013). These kinds of advanced robot applications require an accurate model of the robotic system, which in turn, requires sufficiently accurate knowledge of the parameters of robot dynamics to be applied in advanced control system design,

preoperative planning, process supervision, and simulation and training.

Dynamic models of robot arms used in model-based control schemes are designed in terms of various inertial and friction parameters that must be either measured directly or determined experimentally. However, direct measurements of such characteristics are rather impractical or even impossible in many cases. Inertial parameters of robot links cannot be measured without dismantling the robot arm, while highly nonlinear inherent phenomena at robot joints cannot be directly quantified. Therefore, models describing nonlinear effects such as friction should be addressed in conjunction with methods of determining parameters of the dynamic model of the arm based on experiments, in order to fully identify the dynamic model of the robot arm (Mavroidis, Flanz, Dubowsky, Drouet, & Goitein, 1998).

There are many traditional methods that have been used for dealing with dynamic robot parameter identification including Kalman Filter (Gautier & Poignet, 2001) and least square method (Karahan & Binguel, 2008; Khalil, Gautier, & Lemoine, 2007), etc. However, some model parameters such as link mass and link lengths cannot be easily measured using these methods especially with the effect of noise factor, or in other words their measurements relatively difficult (Fleming & Purshouse, 2002). Moreover, these traditional techniques are relatively effective for a class of specific issues. For example, the structural model is reliable but the data has limited accuracy. Furthermore, they depend on unrealistic assumptions that models must be unimodal, continuous and derivable. These methods sometimes converge slowly, and sometimes at local optimum, or even not at all.

Recently, there have been intelligent proposed methods for estimation based on the use of universal approximations such as fuzzy logic and neural network methodologies. These methods seem to be very attractive because in the ideal case they allow the modeling of the dynamic effects even 'bad'-modeled, for example, friction. In recent

years, Evolutionary algorithms such as Genetic Algorithm (GA), Differential Evolution (DE) and Particle Swarm Optimization (PSO) have been studied extensively. They have been used to improve the dexterity of robot manipulators in many fields such as control, parameter identification, robot design and planning (Bingul & Karahan, 2011; Vuong & Ang, 2009; Zakharov & Halasz, 2001). They have been known to be better suited for noisy, discontinuous functions because they require no knowledge or gradient information about the response surface. This ability of Evolutionary algorithms has encouraged researchers to use these methods in order to moderate the difficulties of noise and nonlinearity that often arise in dynamic models. GAs is better suited for noisy, discontinuous functions because there is no requirement for a derivative in the fitness function. Moreover, GAs accumulate information about the system during the search process, which makes them more desirable than the traditional numerical methods (Adamson & Liu, 2006) through the use of real-coded GA to estimate friction and torque sensor model parameters. The simulation approach demonstrates the effectiveness of the GA. By identifying the parameters, the position tracking error and the velocity tracking of the joint is enhanced. The performance of GA has been also analyzed and evaluated in optimizing the precision of kinematic parameters of the robot manipulator by developing a forward calibration algorithm which is based on GAs. The main problem for this approach is to find a good mathematical correction function and in (Wang, 2009) a suggestion has been made to enhance the accuracy of the robot manipulator by using some new techniques such as ANN and Fuzzy Logic technique.

Differential Evolution (DE) Algorithm has extensively been used to minimize nonlinear and non-differentiable continuous space functions. So far, there has been no attempt to optimize the design parameters of manipulator by which performance variations will be minimal. In (Rout & Mittal, 2010) a modification in differential evolution is proposed to incorporate the effect of noise in the optimization process and

obtain the optimal design of manipulator, which is insensitive to noise. In this optimization process, the kinematic and dynamic models of the manipulators are used. The results indicate that the DE converges quickly with fewer generations and function evaluations than GA. Hence, fast performance of DE indicates that this approach can be a viable optimization technique.

However, the performance of DE is still sensitive to its control parameters such as mutation factor (F) and crossover rate (CR). Recently, the development of adaptive DE has shown more reliable performance than DE with manual settings (Liu & Lampinen, 2005; Tvrdik, 2009; Zhu, Tang, Fang, & Zhang, 2013).

In this chapter, the application of ARDE algorithm is used to estimate the barycentric parameters of the CRS A456 robot manipulator. This algorithm is used to off-line estimate the optimal parameters of the inverse dynamic model of the CRS A465 robot arm, which is expected to be insensitive to noise.

6.3 Dynamic Model of the CRS A456 Robot Manipulator

The CRS A465 arm considered in this work is used as a slave robot in a research cell for orthopedic robot-assisted surgery (see Figure 6.1). In this application, the end effector of the arm carries the surgical tool - the “drilling/machining tool”. Due to the symmetry of the drilling tool, only five degrees of freedom is required. Therefore, only the first five joints of the arm are considered to be the subject for the modeling task in this work.

The equation of motion for the robot is developed using the L-E formulation. The L-E is non-recursive method that allows the development of the robot model using a set of equations derived from the energy model (Mittal & Nagrath, 2003). Based on this formulation the torque acting on any joint axis is:

$$\tau_i = \sum_{j=1}^N D_{ij}(q, \chi) \ddot{q}_j + \sum_{j=1}^N \sum_{k=1}^N H_{ijk}(q, \chi) \dot{q}_j \dot{q}_k + G_i(q, \chi) + \tau_{fi} \quad (6.1)$$

where

τ_i is the torque acting on joint i , $i = 1, 2, \dots, N$, N is the number of degrees of freedom,

q, \dot{q}, \ddot{q} are the position, velocity and acceleration of robot joints, respectively,

χ is the model parameters,

D_{ij} is the effective and coupling inertia,

H_{ijk} is the centripetal and Coriolis effect,

G_i is the Gravity loading, and

τ_{fi} is the joint friction.

The details of the coefficients D_{ij} and H_{ijk} is given in (Mittal & Nagrath, 2003) through examination of Equation 6.1 shows that the equation of motion is linear in the robot physical parameters, χ , that is the mass, center of gravity locations moments and products of inertia of each link (see Figure 6.2). Therefore Equation 6.1 can be written as,

$$\tau = \phi(q, \dot{q}, \ddot{q})\chi \quad (6.2)$$

where τ is the torque vector, $\phi(q, \dot{q}, \ddot{q})$ represents an $(N \times R)$ observation matrix, and the R -length vector χ , contains the effective inertial parameters of the manipulator grouped in the barycentric or base parameters.

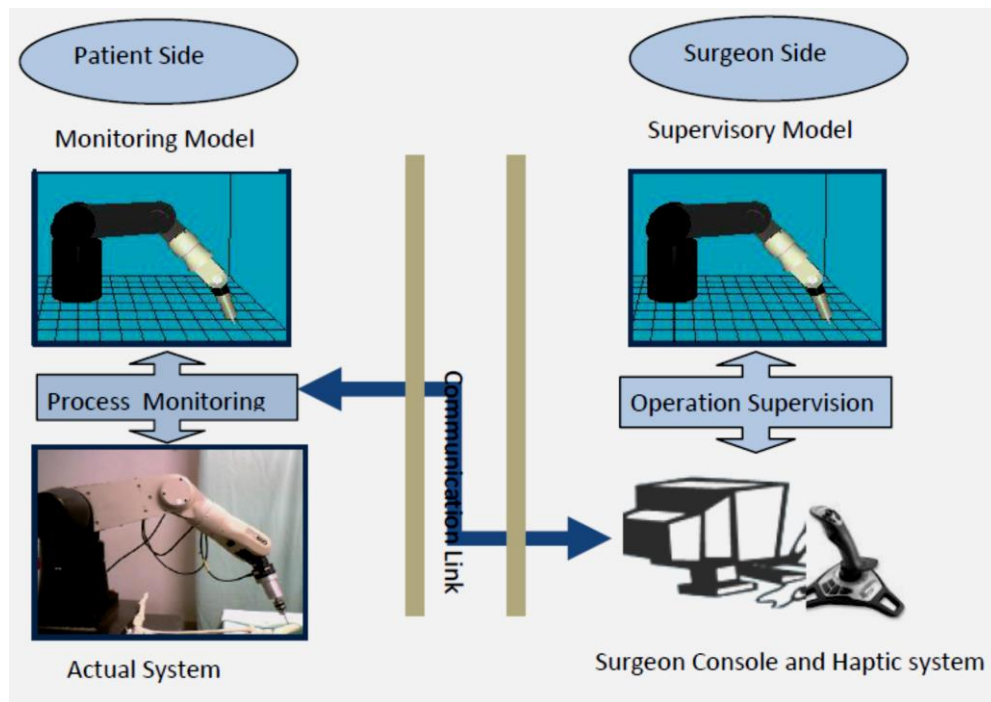


Figure 6.1: Structure of a single robotic cell for robot assisted orthopaedic surgery (Kinsheel, Taha, Deboucha, & Ya, 2012)

The identification “observation” matrix $\phi(q, \dot{q}, \ddot{q})$ depends on the joint angles, velocities, and accelerations. The barycentric parameters of a link are combinations of its inertial parameters and its descendants in the kinematic chain (Renaud, 1987). The categorization and grouping of the barycentric parameters is done symbolically or by applying a set of rules. Normally, special computer programs are developed for automatic generation of the symbolic model and the associated barycentric parameters. For the CRS A465 the set of the barycentric parameters, χ are given in (Kinsheel, Taha, Deboucha, & Ya, 2012).

In this study, in order to make a clear comparison among the estimation methods, the problem has been simplified to consider only a single joint arm of the CRS A465 to estimate its parameters. The CRS 465 single joint arm has four parameters a_i , $i = 1, \dots, 4$ to be identified; they are the inertia, the viscous friction coefficient, the positive side Coulomb friction, and the negative side Coulomb friction, respectively. The system

equation becomes:

$$\tau = a\chi \quad (6.3)$$

where τ is the torque, and χ is the barycentric parameters that have been reduced to four parameters, they are the angular acceleration x_1 , the angular velocity x_2 , the positive sign of the velocity x_3 ($=1$ if x_2 is positive, 0 otherwise) and the negative sign of the velocity x_4 ($=1$ if x_2 is negative, 0 otherwise).

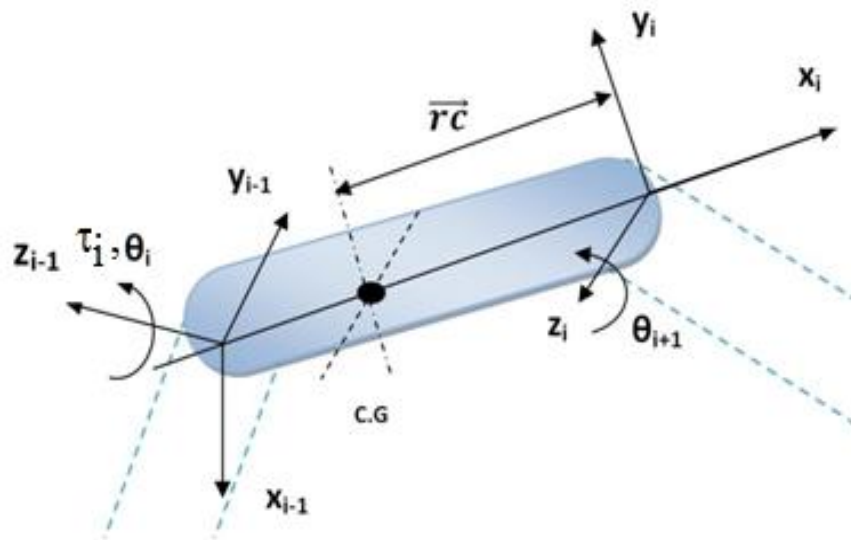


Figure 6.2: Coordinate frame assignment of single joint CRS A465

6.4 System Implementation

The kinematic and dynamic models of manipulators are nonlinear and coupled. Thus, explicit modeling of noises will make dynamic modal complex. To overcome this problem the ARDE algorithm has been utilized for improving the parameter estimation of the robot manipulator and to deliver minimum performance variation. As a case study, the single joint arm model of the CRS A465 is considered. The CRS 465 single joint arm has $A = 4$ parameters to be estimated, as discussed in Section 6.3. In the simulation, a_1 is the inertia, a_2 is the viscous friction coefficient, a_3 is the positive side

Coulomb friction, and a_4 is the negative side Coulomb friction.

To develop the new dynamic parameter identification framework based on ARDE, attention has to be paid to the following setting points that characterize the ARDE algorithm as well as the standard DE algorithm to the robot application:

- **Individual (solution encoding) and Population representation:** A population with pop_{size} ($NP = 30$) of individuals, refers to the number of individuals at each generation. First, we have to encode the necessary information required for the parameter estimation in the individual structure. Each individual should represent a complete solution to the problem at hand. In our application the individual is a vector of 4 real-coded parameters known as *solution parameters*.
- **Parameter control:** The control parameters that are going to be considered are the mutation factor, F and the crossover rate, CR . In the standard DE/rand/1/bin, these parameters have been set to 0.5 and 0.9, respectively. In ARDE, F and CR values are undergo the evolution via the adaptive system; in such a way that better values of these parameters would lead to better individuals which in turn are more likely to survive and produce offspring and hence propagate these better parameter values to the next generation. The other parameters in ARDE have been initialized with $CR_m = 0.5$, $F_m = 0.5$, $p = 0.05$, and *archive size* $= NP$. The local search part of the SPX has been switched off.
- **Individual evaluation (solution validation):** The definition of the fitness function is crucially important for a successful application. In this work, we have to evaluate the fitness of each individual based on the mean square error (MSE) of the estimated model, as in Equation 6.4.

$$MSE = \sum_{i=1}^n (\tau_i - \hat{\tau}_i)^2 \quad (6.4)$$

where τ represents the measured torque in our application and $\hat{\tau}$ is the estimated torque using the DE and ARDE algorithms. n is the dimension of τ vector.

- **Stopping criteria:** The most common stopping condition used in literature is to allow the algorithm to run to a maximum number of iterations. A small number of iterations may not give the algorithm enough time to reach an optimum especially when the size of the search space is large. On the other hand, a very large number of iterations may be unnecessary because there can no further gain once the optimum solution is reached; so, the number of iterations for the standard DE and ARDE is set at 200.

In this comparison the ordinary least square (OLS) identification method (see Equation 6.5) has also been included to estimate the unknown parameters X_{OLS} by minimizing the sum of the squared error between the actual torque τ and the predicted torque $\phi_{X_{OLS}}$, as follows:

$$X_{OLS} = (\phi^T \phi)^{-1} \phi^T \tau \quad (6.5)$$

Using the aforementioned methods and their corresponding settings the barycentric parameters of the CRS 465 single joint arm are estimated. The results of the three estimation techniques are presented in Table 6.1. These results have been averaged over 30-independent runs.

A clear comparison among these methods is presented in Table 6.2 which illustrates the mean square error and the standard deviation of the prediction error. From the same table it can be observed that the ARDE could outperform the OLS and the standard DE for both values. This is so because ARDE updates the values of the control parameters each generation and this change can deal with difficult problems such as noise.

Table 6.1: Barycentric parameters estimation of the single joint CRS A465 robot arm

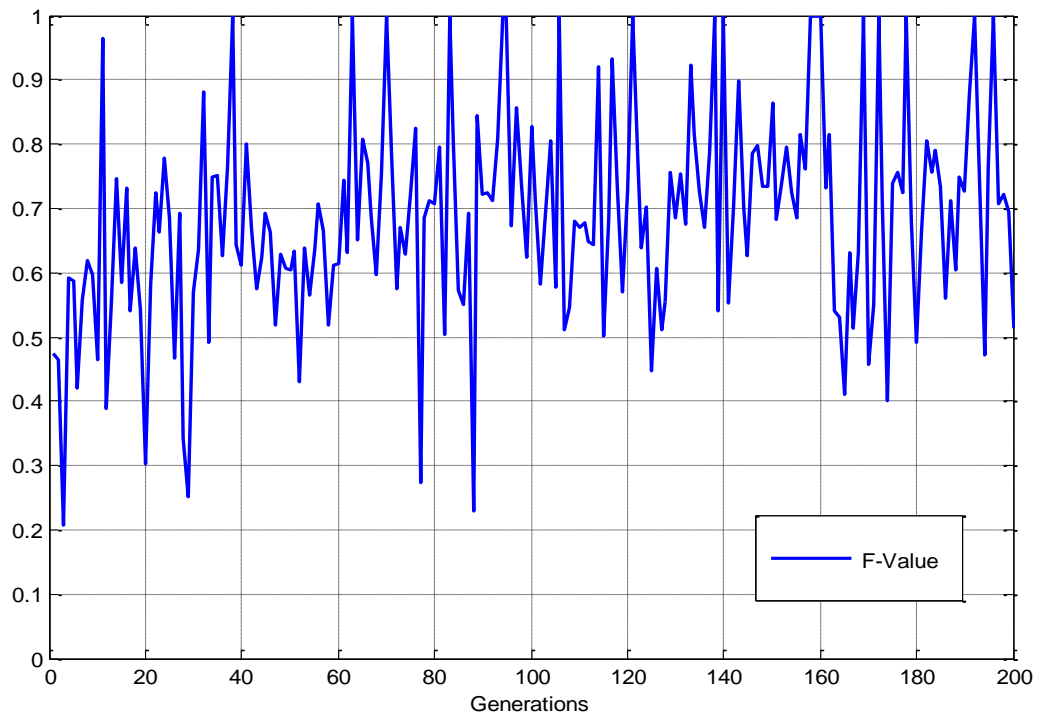
Single Joint Parameters	OLS	Standard DE	ARDE
a_1	0.0036	0.0037	0.0038
a_2	0.0164	0.0143	0.0169
a_3	0.0089	0.0594	0.0112
a_4	-0.2582	-0.3060	-0.2261

Table 6.2: Mean square error and standard deviation of the estimation methods for the estimated model averaged over 30-independent runs

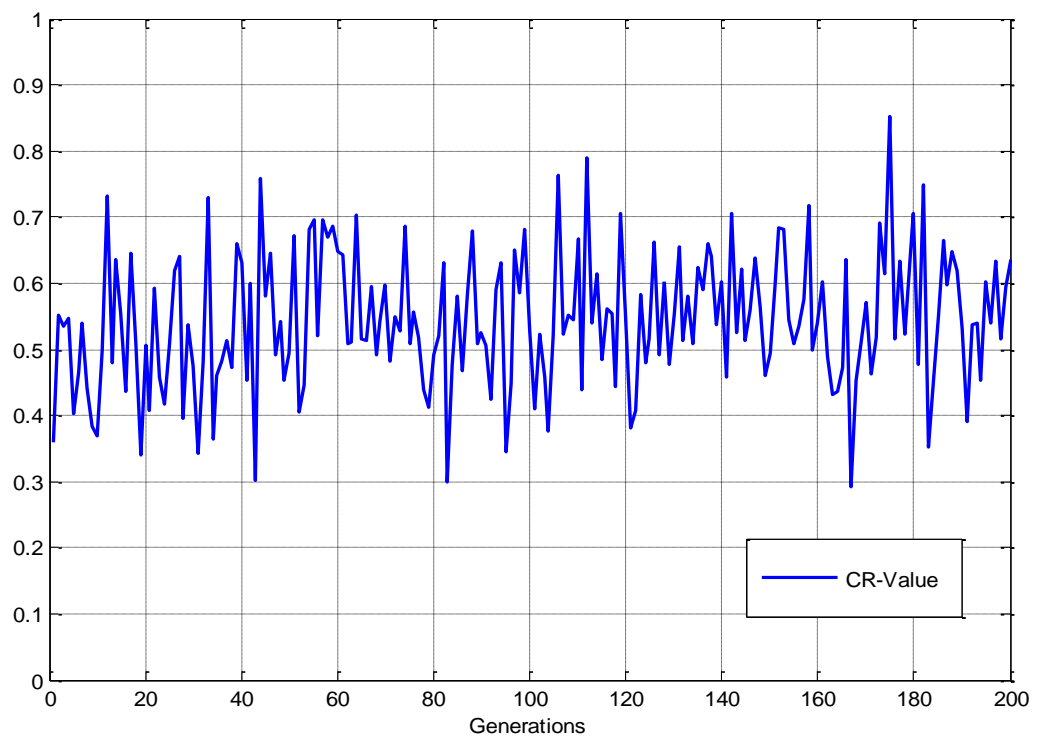
OLS	Standard DE	ARDE
MSE (Std Dev)	MSE (Std Dev)	MSE (Std Dev)
9.259E-02 (2.099E+00)	8.872E-02 (1.415E+00)	5.143E-02 (1.390E+00)

Figure 6.3 depicts the different behavior of F and CR values during the 200 generations due to the population information. The plot of the figure shows a significant high fluctuation at the early stages of the run then begins to stabilize due to stability in the population. This change in F and CR values helps ARDE to escape from the local optimums generated by the noisy components in the dataset.

The distinct performance of the ARDE in comparison with the standard DE and the OLS is further proved in the torque prediction, as depicted in Figure 6.4. From this figure, and based on the barycentric parameters, it is worth noting that the standard DE and ARDE are both nearer to the measured torque than the OLS. The difference is clearer in the accuracy of the model as already presented in Table 4. However, the difference between the standard DE and ARDE performance plot will be more significant as the number of the estimated parameters is increased.



(a) *F*-Value



(b) *CR*-Value

Figure 6.3: The behavior of the *F* and *CR* values in ARDE algorithm during 200 generations

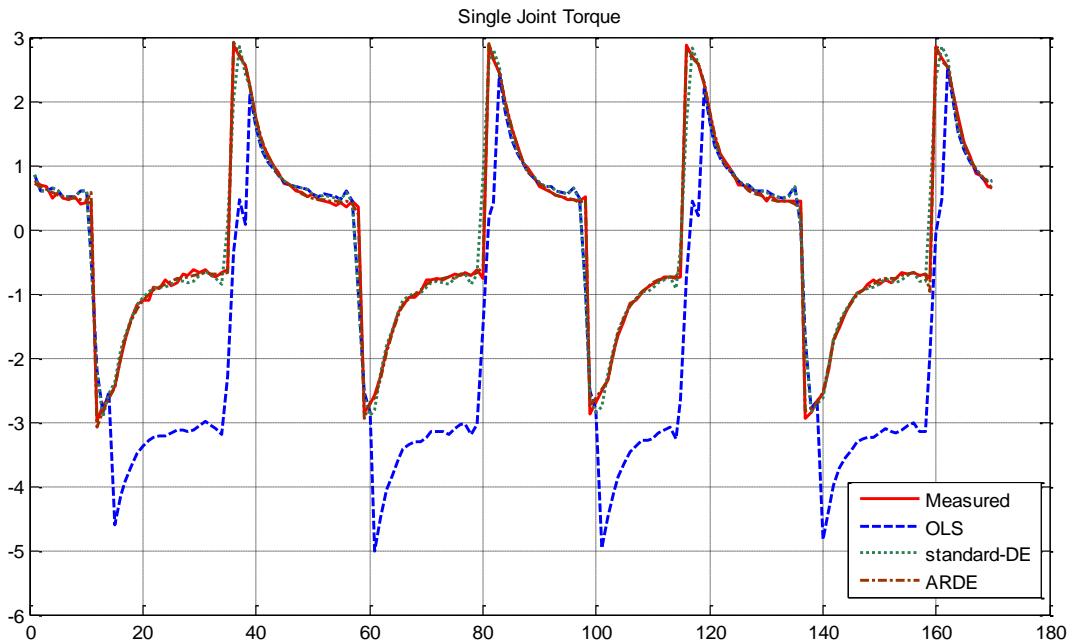


Figure 6.4: Measured torque compared with estimated torque using different methods

6.5 Summary

In this chapter, the ARDE is utilized to estimate the barycentric parameters of single joint CRS A465 robot arm dynamics. In this method the values of the control parameters F and CR are adapted using the adaptive repository mechanism of parameters and mutation strategies. The main advantages of this approach are: computationally efficient, and well-adaptable with optimization techniques. ARDE is not only a simple approach in comparison with other adaptive DE variants, but is also reliable and easy to be implemented in real time applications such as robot system identification. The barycentric parameters of a single joint CRS A465 robot are also estimated using OLS and the standard DE, and the experimental results suggest that ARDE provides better overall performance than the ordinary least square method and the standard DE with fixed parameters.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Introduction

This thesis contains an exploration of adaptive EA strategies for global optimization. The work was not driven by an existing algorithm nor by a single application. Rather, this work was motivated by a desire to cover and understand the subject of adaptive EA algorithms in general from the most basic level. In this study, adaptive DE has been treated as a case study of this type of algorithms then a new adaptive DE algorithm called ARDE-SPX has been proposed that could overcome the limitations over other state-of-the-art adaptive DE variants existing in literature. While a number of benchmark functions were explored to examine in-depth the performance of the ARDE-SPX algorithm. Robot manipulator system identification has also been considered as a real-world application to study the performance of the ARDE-SPX in comparison with standard methods.

In this final chapter, a summary of conclusions, contributions and future work directions on developments and improvements of adaptive EA algorithms are presented.

7.2 Research Conclusions

The fundamental conclusions of this research work include the analysis and development of adaptive DE algorithm and its application to a real-world application are summarized below.

- *The classification of adaptive Evolutionary Algorithms and adaptive Differential Evolution algorithms is always an on-going research area. The research on*

developing adaptive EA algorithms has been such a hot topic. As more and more new adaptive algorithms are proposed with new characteristics, the need for a general classification that can cover all these types of algorithms becomes a large demand. These classifications provide knowledge to those researchers who are interested in this type of algorithms on what have been implemented and what improvements or developments can be added in this area as future work. In this thesis, two classifications have been proposed for the purpose: First, extension taxonomy to the EA parameter settings that covers in general the type of parameters settings in evolutionary computations. Second, general classification to the adaptive DE algorithms that classifies these algorithms based on the parameters control of the algorithm as well as the number of DE strategies employed in the implementation.

- *Differential Evolution parameters tuning is no less important than the adaptive Differential Evolution.* Many researchers from different disciplines like engineering seek out the simplicity for their applications. Some of the real-world applications such on-line systems demand fast algorithms with less number of parameters to be used such as the standard DE, and this requires a prior knowledge on which parameter settings can give the best performance of the algorithm. In general, parameter setting is a problem dependent; there is no possibility that any algorithm can be tuned once to optimize all types of problems. A table that composed of the DE parameters tuning for different problems has been granted for those practitioners who are interested to use the standard DE.
- *Results of the Differential Evolution with adaptive repository and local search (ARDE-SPX) are promising; however, the ARDE-SPX is still in its infancy.* The results of the final solution precision based on the mean and standard deviation,

as well as the robustness of the ARDE-SPX represented by the mean of the successful rate have confirmed that ARDE-SPX has the advantage over other adaptive DE variants presented in the comparison. The comparison between ARDE-SPX and other adaptive DE methods based on the number of functions that each algorithm has achieved the best results over the other algorithms. ARDE-SPX has the superior performance in both 30-dimension and 100-dimension problems. But, it is also important to mention that this current study, like many other newly proposed algorithms, needs further study to shed the lights more on its benefits, weaknesses and limitations.

- *The advantages of the ARDE-SPX are not the same for different problems.* Similar to all optimization algorithms, ARDE-SPX does not present consistent behavior over different problems. However, the overall performance of ARDE-SPX has shown to be better than the other five adaptive DE algorithms over the selected benchmark test suite.
- *The proposed adaptive repository is general enough to be applied on different algorithms.* The adaptive repository mechanism of strategies and parameters adaptation schemes is a general mechanism and can be embedded with high flexibility inside any population-based evolutionary algorithm for further investigation.
- *There is fewer control parameters in ARDE-SPX than most of the adaptive DE algorithms.* ARDE-SPX has no extra control parameters added to its main procedure. The only way that may increase the ARDE-SPX's control parameters depends on the DE strategies and parameters control schemes involved in the repository.
- *ARDE has shown good results in real-world application.* In system identification and control of robot manipulator, ARDE has shown better

performance than the standard DE and the ordinary least square method OLS. Because of the high randomness of ARDE in terms of its adaptive manner, it could overcome the problem of the robot noisy data.

7.3 Research Future Work

Many studies can be conducted to extend or enhance the adaptive DE algorithms based on the analysis of these methods or from the new proposed ARDE-SPX algorithm. Some of these directions can be stated as follows:

- *The use of alternative DE strategies in ARDE-SPX.* Investigate the use of the adaptive repository mechanism of ARDE on other DE strategies rather than JADE with archive. So far, there are many DE strategies and can be either replaced or integrated with the existing JADEw strategies.
- *The use of alternative parameter control schemes in ARDE-SPX.* Investigate the use of the adaptive repository mechanism of ARDE-SPX on other parameters adaptive schemes of F and CR rather than the adaptation scheme of MDE_pBX. There are many parameter adaptive/self-adaptive schemes that can be integrated with the ARDE-SPX.
- *Generalize ARDE-SPX to handle constraint and multi-objective optimization.* In most of the practical applications there are the problems of constraint handling and multi-objective. There are many DE approaches for handling these kinds of problems; they can be integrated with ARDE-SPX to solve multi-objective constrained problems.
- *Extend the adaptive repository of DE (ARDE-SPX) to non-continuous optimization (discrete/integer).* In this study, the continuous optimization of ARDE-SPX is considered. There are many optimization problems that require individuals with discrete values, these types of problems are called

combinatorial problems. There some previous work has been done on modifying DE to deal with discrete variables; these components can be added to ARDE-SPX to solve these problems.

- *Investigate the use of different local search algorithms.* There many other local search algorithms than SPX, such as Hill-Climbing and Tabu search methods. These algorithms can be added to the ARDE algorithm, then a comparison analysis can be conducted to investigate the effect of each of these algorithms on the performance of ARDE.
- *Multi-comparison statistical test.* It would be interesting to use some multi-comparison statistical test such as Friedman test, ANOVA and Wilcoxon Rank to analyze the differences among the state-of-the-art adaptive DE variants and ARDE-SPX algorithm.
- *Increase the number of joints in the robot part.* In order to further investigate the performance of the ARDE as an estimator technique and any possible shortcomings, further work is considered to increase the number of joints of the robot arm which in turn will increase the number of parameters of the predicted model.
- *Improve the performance of the JADE mutation strategy and its variants (JADE with archive, SaDE-MMTS, and SaJADE).* The selection of the best individuals, $p\%$ of the population size in the mutation strategy can be implemented in an adaptive manner based on the population diversity.
- *Improve the performance of the MDE_pBX algorithm in different directions.* The MDE_pBX algorithm is a platform for many modifications. 1) An analytical investigation on the effects of the two new strategies (mutation and crossover) on the population diversity and convergence rate. 2) The connotation of a dynamic grouping can be a future MDE_pBX development to include new

operators such as $DE/gr_best/1$, $DE/gr_best/2$, etc., then their effectiveness could be measured on different types of test functions. 3) The parameter, p , may also be modified to be adaptive or at the very least dynamic during the evolution process, hence its performance effectiveness can further be investigated. 4) There are two additional control parameters q (the group size in the mutation operation) and p (the number of the top-ranking vectors in the crossover operation), a theoretical guidelines of how to select the values of p and q can be investigated.

- *Enhance the adaptive scheme of the parameters control in the SaDE and its variant SaDE-MMTS.* In these two algorithms, the parameter F can be set to an adaptive rule that accumulate knowledge from the previous generations.
- *Improve the adaptive ensemble of EPSDE.* The random strategy of the EPSDE in selection the parameters control and DE strategies can be improved by accumulating knowledge regarding the performance of the control parameter values through certain number of generations.

REFERENCES

- Abbass, H. A. (2002). *The self-adaptive Pareto differential evolution algorithm*. Paper presented at the Proceedings of the IEEE Congress on Evolutionary Computation (CEC2002), Honolulu, HI.
- Adamson, M., & Liu, G. (2006). *A genetic algorithms approach to model parameter estimation of a robot joint with torque sensing*.
- Ahandani, M. A., Shirjoposh, N. P., & Banimahd, R. (2011). Three modified versions of differential evolution algorithm for continuous optimization. *Soft Computing*, 15(4), 803-830. doi: 10.1007/s00500-010-0636-5
- Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. *Computational Intelligence: A Dynamic Systems Perspective*, 152-163.
- Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Pittsburgh, PA: Carnegie Mellon University.
- Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm Pittsburgh, PA: Carnegie Mellon University.
- Beuhren, M. (2011). Differential Evolution, from <http://www.mathworks.com/matlabcentral/fileexchange/18593-differential-evolution%20>
- Bi, X. J., & Xiao, J. (2011). Classification-based self-adaptive differential evolution with fast and reliable convergence performance. *Soft Computing*, 15(8), 1581-1599. doi: 10.1007/s00500-010-0689-5
- Bingul, Z., & Karahan, O. (2011). Dynamic identification of Staubli RX-60 robot using PSO and LS methods. *Expert Systems with Applications*, 38(4), 4136-4149. doi: 10.1016/j.eswa.2010.09.076
- Blickle, T., & Thiele, L. (1997). A comparison of selection schemes used in genetic algorithms *Evolutionary Computation*, 4(4), 261-294.
- Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Acm Computing Surveys*, 35(3). doi: 10.1145/937503.937505
- Bongard, J. (2009). Biologically Inspired Computing. *Computer*, 42(4), 95-98.
- Bonissone, P. P., Subbu, R., Eklund, N., & Kiehl, T. R. (2006). Evolutionary algorithms plus domain knowledge equals Real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3), 256-280. doi: 10.1109/tevc.2005.857695

- Brest, J., Boskovic, B., Greiner, S., Zumer, V., & Maucec, M. S. (2007). Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing*, *11*(7), 617-629. doi: 10.1007/s00500-006-0124-0
- Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, *10*(6), 646-657. doi: 10.1109/tevc.2006.872133
- Brest, J., & Maucec, M. S. (2011). Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, *15*(11), 2157-2174. doi: 10.1007/s00500-010-0644-5
- Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes* (First ed.). Australia: LuLu Enterprises.
- Caraffini, F., Neri, F., & Poikolainen, I. (2013). Micro-differential evolution with extra moves along the axes. *Proceedings of the 2013 IEEE Symposium on Differential Evolution (SDE)*.
- Chakraborty, U. K., Abbott, T. E., & Das, S. K. (2012). PEM fuel cell modeling using differential evolution. *Energy*, *40*(1), 387-399. doi: 10.1016/j.energy.2012.01.039
- Chandra, A., & Chattopadhyay, S. (2014). A novel approach for coefficient quantization of low-pass finite impulse response filter using differential evolution algorithm. *Signal Image and Video Processing*, *8*(7), 1307-1321. doi: 10.1007/s11760-012-0359-4
- Chiang, T. C., Chen, C. N., & Lin, Y. C. (2013). Parameter control mechanisms in differential evolution: A tutorial review and taxonomy. *Proceedings of the 2013 IEEE Symposium on Differential Evolution (SDE)*.
- Cotta, C., Sevaux, M., & Sörensen, K. E. (2008). *Adaptive and multilevel metaheuristics* (Vol. 136). Berlin, Germany: Springer-Verlag.
- Das, S., Abraham, A., Chakraborty, U. K., & Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, *13*(3), 526-553. doi: 10.1109/tevc.2008.2009457
- Das, S., Mandal, A., & Mukherjee, R. (2014). An adaptive differential evolution algorithm for global optimization in dynamic environments. *IEEE Transactions on Cybernetics*, *44*(6), 966-978. doi: 10.1109/tevc.2013.2278188
- Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 27-54. doi: 10.1109/tevc.2010.2059031
- De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD Thesis, University of Michigan, Dissertation Abstracts International 36(10), 5140B. (University Microfilms No. 76-9381)

- Develi, I., & Yazlik, E. N. (2012). Optimum antenna configuration in MIMO systems: a differential evolution based approach. *Wireless Communications & Mobile Computing*, 12(6), 473-480. doi: 10.1002/wcm.974
- Dixon, L. C. W., & Szego, G. (1978). *The global optimization problem: An introduction*. Paper presented at the Proceeding Toward Global Optimization 2, Amsterdam, Netherlands: North-Holland.
- Dong, N., & Wang, Y. (2014). A memetic differential evolution algorithm based on dynamic preference for constrained optimization problems. *Journal of Applied Mathematics 2014*, 1-15.
- Dragoi, E. N., Curteanu, S., Galaction, A. I., & Cascaval, D. (2013). Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process. *Applied Soft Computing*, 13(1), 222-238. doi: 10.1016/j.asoc.2012.08.004
- Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2), 124-141. doi: 10.1109/4235.771166
- Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing* (Second ed.). Berlin, Germany: Springer-Verlag.
- Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2014). A self-adaptive combined strategies algorithm for constrained optimization using differential evolution. *Applied Mathematics and Computation*, 241, 267-282. doi: 10.1016/j.amc.2014.05.018
- Fan, H.-Y., & Lampinen, J. (2002). *A trigonometric mutation approach to differential evolution*. Paper presented at the Giannakoglou KC, Tsahalis DT, Papailiou JPKD, Fogarty T (eds) Evolutionary methods for design, optimization and control with Applications to Industrial Problems, CIMNE, Barcelona.
- Fan, H. Y., & Lampinen, J. (2003). A trigonometric mutation operation to differential evolution. *Journal of Global Optimization*, 27(1), 105-129. doi: 10.1023/a:1024653025686
- Feoktistov, V. (2006). *Differential evolution: In search of solutions* (Vol. 5). New York, United State: Springer-Verlag.
- Feoktistov, V., & Janaqi, S. (2004a). Differential evolution. France: LGI2P-l'Ecole des Mines d'Ales, Parc Scientifique G. Besse, 30035 Nîmes.
- Feoktistov, V., & Janaqi, S. (2004b). *Generalization of the strategies in differential evolution*. Paper presented at the 18-th Annual IEEE International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, USA.
- Feoktistov, V., & Janaqi, S. (2004c). *New energetic selection principle in differential evolution*. Paper presented at the 6th International Conference on Enterprise Information Systems - ICEIS 2004, Porto-Portugal.

- Feoktistov, V., & Janaqi, S. (2004d). New strategies in differential evolution - Design principle *I.C. Parmee editor, Adaptive computing in design and manufacture VI* (pp. 335-346). UK, London: Springer - Verlag London Limited.
- Fleming, P. J., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10(11), 1223-1241. doi: 10.1016/s0967-0661(02)00081-3
- Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1), 3-14. doi: 10.1109/72.265956
- Fogel, D. B., Fogel, L. J., & Atmar, J. W. (1991, Nov 4-6). *Meta-evolutionary programming*. Paper presented at the Conference on Signals, Systems and Computers. 1991 Conference Record of the Twenty-Fifth Asilomar, San Diego, CA, USA.
- Gautier, M., & Poignet, P. (2001). Extended Kalman filtering and weighted least squares dynamic identification of robot. *Control Engineering Practice*, 9(12), 1361-1372. doi: 10.1016/s0967-0661(01)00105-8
- Ghosh, S., Das, S., Roy, S., Islam, S. K. M., & Suganthan, P. N. (2012). A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization. *Information Sciences*, 182(1), 199-219. doi: 10.1016/j.ins.2011.08.014
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. New Jersey, United States: Pearson Education (US).
- Gomes, P. (2011). Surgical robotics: Reviewing the past, analysing the present, imagining the future. *Robotics and Computer-Integrated Manufacturing*, 27(2), 261-266. doi: 10.1016/j.rcim.2010.06.009
- Gong, W., Cai, Z., Ling, C. X., & Li, H. (2011). Enhanced differential evolution with adaptive strategies for numerical optimization. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 41(2), 397-413. doi: 10.1109/tsmcb.2010.2056367
- Goudos, S. K., Siakavara, K., Samaras, T., Vafiadis, E. E., & Sahalos, J. N. (2011). Self-adaptive differential evolution applied to real-valued antenna and microwave design problems. *IEEE Transactions on Antennas and Propagation*, 59(4), 1286-1298. doi: 10.1109/tap.2011.2109678
- Griewank, A. O. (1981). Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34(1), 11-39.
- Hansen, N., & Ostermeier, A. (1996). *Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation* Paper presented at the Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Nagoya, Japan.

- Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159-195. doi: 10.1162/106365601750190398
- He, J., & Yao, X. (2002). From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5), 495-511. doi: 10.1109/tevc.2002.800886
- Ho, Y. C., & Pepyne, D. L. (2002). Simple explanation of the no free lunch theorem of optimization *Cybernetics and Systems Analysis*, 38(2), 292–298.
- Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267(1), 66-72.
- Igel, C., & Toussaint, M. (2004). A no-free-lunch theorem for non-uniform distributions of target functions *Journal of Mathematical Modelling and Algorithms*, 3, 313–322.
- Iorio, A. W., & Li, X. D. (2004). Solving rotated multi-objective optimization problems using differential evolution. *Ai 2004: Advances in Artificial Intelligence, Proceedings*, 3339, 861-872.
- Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2012). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 42(2), 482-500. doi: 10.1109/tsmcb.2011.2167966
- Jeyakumar, G., & Shanmugavelayutham, C. (2009). *An empirical comparison of differential evolution variants for high dimensional function optimization*. Paper presented at the Iama: 2009 International Conference on Intelligent Agent & Multi-Agent Systems, Chennai, INDIA.
- Jeyakumar, G., & Shanmugavelayutham, C. (2011). Empirical measurements on the convergence nature of differential evolution variants *Advances in Computer Science and Information Technology, Pt I* (Vol. 131, pp. 472-480). Berlin, Heidelberg: Springer-Verlag.
- Jeyakumar, G., & Velayutham, C. S. (2010). *A comparative study on theoretical and empirical evolution of population variance of differential evolution variants*. Paper presented at the SEAL'10 Proceedings of the 8th international conference on Simulated evolution and learning, Kanpur, India.
- Kaempf, J. H., & Robinson, D. (2009). A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential. *Applied Soft Computing*, 9(2), 738-745. doi: 10.1016/j.asoc.2008.09.009
- Karahan, O., & Binguel, Z. (2008). Modelling and identification of STAUBLI RX-60 robot. *2008 IEEE Conference on Robotics, Automation, and Mechatronics, Vols 1 and 2*, 184-189.

- Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*. Paper presented at the 1995 Proceedings of IEEE International Conference on Neural Networks, Perth, WA.
- Kephart, J. O. (2011). Learning from Nature. *Science*, 331(6018), 682-683. doi: 10.1126/science.1201003
- Khalil, W., Gautier, M., & Lemoine, P. (2007). *Identification of the payload inertial parameters of industrial manipulators*. Paper presented at the IEEE International Conference on Robotics and Automation, Rome, ITALY.
- Kinsheel, A., Taha, Z., Deboucha, A., & Ya, T. M. Y. S. T. (2012). Robust least square estimation of the CRS A465 robot arm's dynamic model parameters. *Journal of Mechanical Engineering Research* 4(3), 89-99.
- Kukkonen, S., & Lampinen, J. (2005). *GDE3: The third evolution step of generalized differential evolution*. Paper presented at the The 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland.
- Lampinen, J. (2001). *Solving problems subject to multiple nonlinear constraints by the differential evolution*. Paper presented at the Proceedings of MENDEL'01 - 7th International Conference on Soft Computing, Brno, Czech Republic.
- Lampinen, J. (2002). *A constraint handling approach for the differential evolution algorithm*. Paper presented at the Proceedings of the 2002 Congress on Evolutionary Computation - CEC2002, Honolulu, HI.
- Lampinen, J., & Zelinka, I. (2000). *On stagnation of the differential evolution algorithm*. Paper presented at the Proceedings of MENDEL'00 - 6th International Mendel Conference on Soft Computing, Brno, Czech Republic.
- Lee, C. Y., & Yao, X. (2004). Evolutionary programming using mutations based on the Levy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1), 1-13. doi: 10.1109/tevc.2003.816583
- Liang, J. J., Suganthan, P. N., & Deb, K. (2005). Novel composition test functions for numerical global optimization. *2005 IEEE Swarm Intelligence Symposium*, 68-75.
- Lin, C., Qing, A., & Feng, Q. (2011). A comparative study of crossover in differential evolution. *Journal of Heuristics*, 17(6), 675-703. doi: 10.1007/s10732-010-9151-1
- Lis, J. (1996). *Parallel genetic algorithm with dynamic control parameter*. Paper presented at the Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Nagoya, Japan.
- Liu, J., & Lampinen, J. (2002a). *Adaptive parameter control of differential evolution*. Paper presented at the Proceedings of MENDEL'02 - 8th International Mendel Conference on Soft Computing, Brno, Czech Republic.

- Liu, J., & Lampinen, J. (2002b). *On setting the control parameter of the differential evolution algorithm*. Paper presented at the Proceedings of the 8th international Mendel conference on soft computing, Brno, Czech Republic.
- Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6), 448-462. doi: 10.1007/s00500-004-0363-x
- Liu, Y., Ni, F. L., Liu, H., & Xu, W. F. (2012). Enhancing pose accuracy of space robot by improved differential evolution. *Journal of Central South University of Technology*, 19(4), 933-943. doi: 10.1007/s11771-012-1095-1
- Lobo, F. G., Lima, C. F., & Michalewicz, Z. (2007). *Parameter setting in evolutionary algorithms* (Vol. 54). Berlin, Germany: Springer-Verlag.
- Mallipeddi, R. (2013). Harmony search based parameter ensemble adaptation for differential evolution. *Journal of Applied Mathematics*. doi: 10.1155/2013/750819
- Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2), 1679-1696. doi: 10.1016/j.asoc.2010.04.024
- Mavroidis, C., Flanz, J., Dubowsky, S., Drouet, P., & Goitein, M. (1998). High performance medical robot requirements and accuracy analysis. *Robotics and Computer-Integrated Manufacturing*, 14(5-6), 329-338. doi: 10.1016/s0736-5845(98)00022-2
- Mezura-Montes, E., Edith Miranda-Varela, M., & del Carmen Gomez-Ramon, R. (2010). Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, 180(22), 4223-4262. doi: 10.1016/j.ins.2010.07.023
- Mezura-Montes, E., Velazquez-Reyes, J., & Coello, C. A. C. (2006). *A comparative study of differential evolution variants for global optimization*. Paper presented at the GECCO 2006: Genetic and Evolutionary Computation Conference, Seattle, WA.
- Mininno, E., Neri, F., Cupertino, F., & Naso, D. (2011). Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, 15(1), 32-54. doi: 10.1109/tevc.2010.2058120
- Mitchell, M. (1998). *An introduction to genetic algorithms* (Third ed.). London, England: MIT Press.
- Mittal, R. K., & Nagrath, I. J. (2003). *Robotics and control* New Delhi: Tata McGraw-Hill.
- Montes, E. M., Coello Coello, C. A., & Tun-Morales, E. I. (2004). *Simple feasibility rules and differential evolution for constrained optimization*. Paper presented at the Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004), New York.

- Neri, F., & Mininno, E. (2010). Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, 5(2), 54-65. doi: 10.1109/mci.2010.936305
- Neri, F., & Tirronen, V. (2008). On memetic differential evolution frameworks: A study of advantages and limitations in hybridization. *2008 IEEE Congress on Evolutionary Computation, Vols 1-8*, 2135-2142. doi: 10.1109/cec.2008.4631082
- Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2), 61-106. doi: 10.1007/s10462-009-9137-2
- Niu, M., & Xu, Z. (2014). Efficiency ranking-based evolutionary algorithm for power system planning and operation. *IEEE Transactions on Power Systems*, 29(3), 1437-1438. doi: 10.1109/tpwrs.2013.2292435
- Noman, N., & Iba, H. (2008). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1), 107-125. doi: 10.1109/tevc.2007.895272
- Oh, S. K., Kim, W.-D., & Pedrycz, W. (2012). Design of optimized cascade fuzzy controller based on differential evolution: Simulation studies and practical insights. *Engineering Applications of Artificial Intelligence*, 25(3), 520-532. doi: 10.1016/j.engappai.2012.01.002
- Pedersen, M. E. H. (2010). Good parameters for differential evolution: Hvas Laboratories.
- Peng, L., Dai, G., Wang, M., Hu, H., Chang, Y., & Chen, F. (2011). *Self-adaptive uniform differential evolution for optimizing the initial integral point of the earth-moon low-energy transfer*. Paper presented at the Proceedings of the Institution of Mechanical Engineers Part G-Journal of Aerospace Engineering.
- Piotrowski, A. P., Napiorkowski, J. J., & Kiczko, A. (2012). Differential evolution algorithm with Separated Groups for multi-dimensional optimization problems. *European Journal of Operational Research*, 216(1), 33-46. doi: 10.1016/j.ejor.2011.07.038
- Pisla, D., Gherman, B., Vaida, C., Suci, M., & Plitea, N. (2013). An active hybrid parallel robot for minimally invasive surgery. *Robotics and Computer-Integrated Manufacturing*, 29(4), 203-221. doi: 10.1016/j.rcim.2012.12.004
- Ponsich, A., & Coello Coello, C. A. (2011). Differential evolution performances for the solution of mixed-integer constrained process engineering problems. *Applied Soft Computing*, 11(1), 399-409. doi: 10.1016/j.asoc.2009.11.030
- Price, K., & Storn, R. (1997). Differential evolution: A simple evolution strategy for fast optimization. *Dr. Dobbs's Journal of Software Tools*, 22(4), 18-24.
- Price, K. V. (1997). *Differential evolution vs. the functions of the 2nd ICEO*. Paper presented at the Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), Indianapolis, In.

- Price, K. V. (1999). An introduction to differential evolution. *New Ideas in Optimization*. London: McGraw-Hill.
- Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization* (First ed.). Berlin, Germany: Springer-Verlag.
- Qasem, S. N., & Shamsuddin, S. M. (2011). Memetic elitist pareto differential evolution algorithm based radial basis function networks for classification problems. *Applied Soft Computing*, *11*(8), 5565-5581. doi: 10.1016/j.asoc.2011.05.002
- Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*(2), 398-417. doi: 10.1109/tevc.2008.927706
- Qin, A. K., & Suganthan, P. N. (2005). *Self-adaptive differential evolution algorithm for numerical optimization*. Paper presented at the Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, SCOTLAND.
- Rahnamayan, S., & Dieras, P. (2008). *Efficiency competition on N-queen problem: DE vs. CMA-ES*. Paper presented at the Canadian Conference on Electrical and Computer Engineering, Niagara Falls, CANADA.
- Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, *12*(1), 64-79. doi: 10.1109/tevc.2007.894200
- Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*: Frommann-Holzboog.
- Renaud, M. (1987). *Quasi-minimal computation of the dynamic model of a robotmanipulator utilizing the Newton-Euler formalism and the notion of augmented body*. Paper presented at the Proc. IEEE Int. Conf. Robot Automation, Raleigh, North Carolina.
- Rönkkönen, J., Kukkonen, S., & Price, V. K. (2005). *Real-parameter optimization with differential evolution*. Paper presented at the The 2005 IEEE Congress on evolutionary computation CEC 2005, Edinburgh, Scotland.
- Rout, B. K., & Mittal, R. K. (2010). Optimal design of manipulator parameter using evolutionary optimization techniques. *Robotica*, *28*, 381-395. doi: 10.1017/s0263574709005700
- Runyon, R. P., Haber, A., Pittenger, D. J., & Coleman, K. A. (1996). *Fundamentals of behavioral statistics* (8th ed.). Boston: McGraw-Hill.
- Sayah, S., & Hamouda, A. (2013). A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems. *Applied Soft Computing*, *13*(4), 1608-1619. doi: 10.1016/j.asoc.2012.12.014

- Schwefel, H. P. (1977). *Numerische optimierung von computer-modellen mittels der evolutionsstrategie: mit einer vergleichenden einführung in die hill-climbing- und zufallsstrategie*: Birkhäuser.
- Shang, Y. W., & Qiu, Y. H. (2006). A note on the extended Rosenbrock function. *Evolutionary Computation*, *14*(1), 119-126. doi: 10.1162/106365606776022733
- Sindhya, K., Ruuska, S., Haanpaa, T., & Miettinen, K. (2011). A new hybrid mutation operator for multiobjective optimization with differential evolution. *Soft Computing*, *15*(10), 2041-2055. doi: 10.1007/s00500-011-0704-5
- Smith, R. E., & Smuda, E. (1995). Adaptively resizing populations: Algorithm, analysis and first results. *Complex Systems*, *9*(1), 47-72.
- Spadoni, M., & Stefanini, L. (2012). A Differential evolution algorithm to deal with box, linear and quadratic-convex constraints for boundary optimization. *Journal of Global Optimization*, *52*(1), 171-192. doi: 10.1007/s10898-011-9695-0
- Storn, R. (1996). *On the usage of differential evolution for function optimization*. Paper presented at the Biennial Conference of the North America Fuzzy Information Processing Society (NAFIPS 1996), Berkeley, CA, New York.
- Storn, R. (2000). Differential evolution (DE), from <http://www1.icsi.berkeley.edu/~storn/code.html>
- Storn, R., & Price, K. (1996). *Minimizing the real functions of the ICEC'96 contest by differential evolution*. Paper presented at the IEEE International Conference on Evolutionary Computation.
- Storn, R., & Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341-359. doi: 10.1023/a:1008202821328
- Storn, R. M., & Price, K. V. (1995). Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Berkeley, CA, USA: International Computer Science Institute.
- Subudhi, B., & Jena, D. (2011). A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, *11*(1), 861-871. doi: 10.1016/j.asoc.2010.01.006
- Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, *10*(8), 673-686. doi: 10.1007/s00500-005-0537-1
- Titare, L. S., Singh, P., Arya, L. D., & Choube, S. C. (2014). Optimal reactive power rescheduling based on EPSDE algorithm to enhance static voltage stability. *International Journal of Electrical Power & Energy Systems*, *63*, 588-599. doi: 10.1016/j.ijepes.2014.05.078
- Torn, A., & Zilinskas, A. (1989). Global optimization *Lecture Notes in Computer Science* (Vol. 350, pp. 1-252).

- Tseng, L. Y., & Chen, C. (2007). *Multiple trajectory search for multiobjective optimization*. Paper presented at the 2007 IEEE Congress on Evolutionary Computation, Singapore, SINGAPORE.
- Tseng, L. Y., & Chen, C. (2008). *Multiple trajectory search for large scale global optimization*. Paper presented at the 2008 IEEE Congress on Evolutionary Computation, Hong Kong, PEOPLES R CHINA.
- Tsutsui, S., Yamamura, M., & Higuchi, T. (1999). *Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms*. Paper presented at the Genetic Evol. Comput. Conf. (GECCO'99)
- Tvrđik, J. (2009). Adaptation in differential evolution: A numerical comparison. *Applied Soft Computing*, 9(3), 1149-1155. doi: 10.1016/j.asoc.2009.02.010
- Urfalioglu, O., & Arıkan, O. (2011). Self-adaptive randomized and rank-based differential evolution for multimodal problems. *Journal of Global Optimization*, 51(4), 607-640. doi: 10.1007/s10898-011-9646-9
- Vesterstrom, J., & Thomsen, R. (2004). *A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems*. Paper presented at the CEC2004: Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004), Portland, OR.
- Vuong, N. D., & Ang, M. H., Jr. (2009). Dynamic Model Identification for Industrial Robots. *Acta Polytechnica Hungarica*, 6(5), 51-68.
- Wang, H., Rahnamayan, S., & Wu, Z. (2013). Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. *Journal of Parallel and Distributed Computing*, 73(1), 62-73. doi: 10.1016/j.jpdc.2012.02.019
- Wang, K. (2009). Application of genetic algorithms to robot kinematics calibration. *International Journal of Systems Science*, 40(2), 147-153. doi: 10.1080/00207720802630644
- Wang, Y. (2011). *Evolutionary algorithms for complex continuous optimization problems*. Doctor of Philosophy, Central South University, China.
- Wang, Y., Cai, Z., & Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1), 55-66. doi: 10.1109/tevc.2010.2087271
- Wang, Y., Cai, Z., & Zhang, Q. (2012). Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, 185(1), 153-177. doi: 10.1016/j.ins.2011.09.001
- Weber, M., Neri, F., & Tirronen, V. (2011). Shuffle or update parallel differential evolution for large-scale optimization. *Soft Computing*, 15(11), 2089-2107. doi: 10.1007/s00500-010-0640-9

- Weber, M., Tirronen, V., & Neri, F. (2010). Scale factor inheritance mechanism in distributed differential evolution. *Soft Computing*, 14(11), 1187-1207. doi: 10.1007/s00500-009-0510-5
- Weise, T. (2009). Global optimization algorithms -Theory and applications (pp. 820). Retrieved from <http://www.it-weise.de/projects/book.pdf>
- Wolpert, D. H., & MacReady, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.
- Xin, B., Chen, J., Zhang, J., Fang, H., & Peng, Z. H. (2012). Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, 42(5), 744-767. doi: 10.1109/tsmcc.2011.2160941
- Yao, X., Liu, Y., & Lin, G. M. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2), 82-102.
- Yin, J., Wang, Y., & Hu, J. (2012). Free search with adaptive differential evolution exploitation and quantum-inspired exploration. *Journal of Network and Computer Applications*, 35(3), 1035-1051. doi: 10.1016/j.jnca.2011.12.004
- Zaharie, D. (2002a). *Critical values for the control parameters of differential evolution algorithms* Paper presented at the Proceedings of MENDEL 2002, 8th International Mendel Conference on Soft Computing, Brno, Czech Republic.
- Zaharie, D. (2002b). *Parameter adaptation in differential evolution by controlling the population diversity*. Paper presented at the Proc. of SYNASC'2002, Analele Univ. Timisoara, Timisoara, Roumania.
- Zaharie, D. (2004). *A multi-population differential evolution algorithm for multi-modal optimization*. Paper presented at the Mendel'04 - 10th International Conference on Soft Computing, Brno, Czech Republic.
- Zaharie, D. (2007). *A comparative analysis of crossover variants in differential evolution*. Paper presented at the Proceedings of the IMCSIT, 2nd International Symposium Advances in Artificial Intelligence and Applications (AAIA'07).
- Zakharov, A., & Halasz, S. (2001). Parameter identification of a robot arm using genetic algorithms. *Periodica Politehnica Ser. Eng*, 45(3-4), 195-209.
- Zhang, J., & Sanderson, A. C. (2009a). *Adaptive differential evolution: A robust approach to multimodal problem optimization* (Vol. 1). Chennai, India: Springer-Verlag.
- Zhang, J., & Sanderson, A. C. (2009b). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5), 945-958. doi: 10.1109/tevc.2009.2014613

- Zhang, X., Chen, W., Dai, C., & Cai, W. (2010). Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization. *International Journal of Electrical Power & Energy Systems*, 32(5), 351-357. doi: 10.1016/j.ijepes.2009.11.009
- Zhao, S.-Z., Suganthan, P. N., & Das, S. (2011). Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing*, 15(11), 2175-2185. doi: 10.1007/s00500-010-0645-4
- Zhao, S., Wang, X., Chen, L., & Zhu, W. (2014). A novel self-adaptive differential evolution algorithm with population size adjustment scheme. *Arabian Journal for Science and Engineering*, 39(8), 6149-6174. doi: 10.1007/s13369-014-1248-7
- Zhu, W., Tang, Y., Fang, J. A., & Zhang, W. (2013). Adaptive population tuning scheme for differential evolution. *Information Sciences*, 223, 164-191. doi: 10.1016/j.ins.2012.09.019
- Zielinski, K., Weitkemper, P., Laur, R., & Kammeyer, K. D. (2006). *Parameter study for differential evolution using a power allocation problem including interference cancellation*. Paper presented at the Proceedings of the IEEE congress on evolutionary computation, Vancouver, BC.
- Zou, D., Liu, H., Gao, L., & Li, S. (2011). A novel modified differential evolution algorithm for constrained optimization problems. *Computers & Mathematics with Applications*, 61(6), 1608-1623. doi: 10.1016/j.camwa.2011.01.029

APPENDIX A

STANDARD BENCHMARK FUNCTIONS

UNIMODAL FUNCTIONS

A. Sphere Function

$$f_1(x) = \sum_{i=1}^D x_i^2 \quad -100 \leq x_i \leq 100$$

$$\text{global minimum } (f_1) = f_1(0, \dots, 0) = 0$$

B. Schwefel 2.22 Function

$$f_2(x) = \sum_{i=1}^D |x_i| + \prod_{i=1}^D |x_i| \quad -10 \leq x_i \leq 10$$

$$\text{global minimum } (f_2) = f_2(0, \dots, 0) = 0$$

C. Schwefel 1.2 Function

$$f_3(x) = \sum_{i=1}^D \left(\sum_{j=1}^i x_j \right)^2 \quad -100 \leq x_i \leq 100$$

$$\text{global minimum } (f_3) = f_3(0, \dots, 0) = 0$$

D. Schwefel 2.21 Function

$$f_4(x) = \max\{|x_i|, 1 \leq x_i \leq D\} \quad -100 \leq x_i \leq 100$$

$$\text{global minimum } (f_4) = f_4(0, \dots, 0) = 0$$

E. Step Function

$$f_5(x) = \sum_{i=1}^D [x_i + 0.5]^2 \quad -100 \leq x_i \leq 100$$

$$\text{global minimum } (f_5) = f_5(0, \dots, 0) = 0$$

F. Quartric Noise Function

$$f_6(x) = \sum_{i=1}^D ix_i^4 + rand[0,1) \quad - 1.28 \leq x_i \leq 1.28$$

$$\text{global minimum}(f_6) = f_6(0, \dots, 0) = 0 + \text{noise}$$

G. Access Parallel Hyper-Ellipsoid Function

$$f_7(x) = \sum_{i=1}^D ix_i^2 \quad - 100 \leq x_i \leq 100$$

$$\text{global minimum}(f_7) = f_7(0, \dots, 0) = 0$$

H. Standard Rosenbrock Function

$$f_8(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \quad - 30 \leq x_i \leq 30$$

$$\text{global minimum}(f_8) = f_8(1, 1) = 0$$

MULTIMODAL FUNCTIONS

I. Generalized Rosenbrock Function

$$f_9(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \quad - 30 \leq x_i \leq 30$$

$$\text{global minimum}(f_9) = f_9(1, \dots, 1) = 0$$

J. Schwefel Function

$$f_{10} = \sum_{i=1}^D [-x_i \sin \sqrt{|x_i|}] \quad - 500 \leq x_i \leq 500$$

$$\text{global minimum}(f_{10}) = f_{10}(420.9687, \dots, 420.9687) = -418.9829 \cdot D$$

K. Rastrigin Function

$$f_{11} = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10] \quad - 5.12 \leq x_i \leq 5.12$$

global minimum (f_{11}) = $f_{11}(0,0,\dots, 0)= 0$

L. Ackley Function

$$f_{12} = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + \exp(1)$$

$-32 \leq x_i \leq 32$ global minimum (f_{12}) = $f_{12}(0,0,\dots, 0)= 0$

M. Griewank Function

$$f_{13} = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$$

$-600 \leq x_i \leq 600$ global minimum (f_{13}) = $f_{13}(0,0,\dots, 0)= 0$

N. Generalized Penalized Functions

$$f_{14} = \frac{\pi}{D} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} + \sum_{i=1}^D u(x_i, 10, 100, 4)$$

$-50 \leq x_i \leq 50$ global minimum (f_{14}) = $f_{14}(-1,-1,\dots, -1)= 0$

$$f_{15}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)] \right\} + \sum_{i=1}^D u(x_i, 5, 100, 4)$$

$-50 \leq x_i \leq 50$ global minimum (f_{15}) = $f_{15}(1,1,\dots, 1)= 0$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$$

O. Nuemaier 3 Function

$$f_{16}(x) = \sum_{i=1}^D (x_i - 1)^2 - \sum_{i=2}^D x_i x_{i-1}$$

$$-D^2 < x_i < D^2 \text{ for } i = 1, 2, 3, \dots, D \quad \text{global minimum}(f_{16}) = -D(D+4)(D-1)/6 \\ \text{at } x_i = i(D+1-i)$$

P. Salomon Function

$$f_{17}(x) = 1 - \cos(2\pi\|x\|) + 0.1\|x\|, \text{ where}$$

$$\|x\| = \sqrt{\sum_{i=1}^D x_i^2}$$

$$-100 \leq x_i \leq 100 \quad \text{global minimum } (f_{17}) = f_{17}(0, 0, \dots, 0) = 0$$

Q. Alpine Function

$$f_{18}(x) = \sum_{i=1}^D |x_i \sin(x_i) + 0.1x_i|$$

$$-10 \leq x_i \leq 10 \quad \text{global minimum } (f_{18}) = f_{18}(0, 0, \dots, 0) = 0$$

R. Easom Function

$$f_{19}(x_1, x_2) = -\cos(x_1) \cos(x_2) \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

$$-100 \leq x_i \leq 100 \quad \text{global minimum}(f_{19}) = f_{19}(\pi, \pi) = -1$$

S. Branin Function

$$f_{20}(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10 \left(1 - \frac{1}{8\pi}\right) \cos(x_1) + 10$$

$$-5 \leq x_1 \leq 10 \quad \text{and} \quad 0 \leq x_2 \leq 15 \quad \text{global minima}(f_{20}) = f_{20}(-\pi, 12.275); f_{20}(\pi,$$

$$2.275); f_{20}(9.42478, 2.475) = 0.397887$$

T. Goldstein-Price Function

$$f_{21}(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

$$-2 \leq x_i \leq 2 \quad \text{global minimum}(f_{21}) = f_{21}(0, -1) = 3.0000$$

U. Six-hump Camel Back Function

$$f_{22}(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

$$-3 \leq x_1 \leq 3 \quad \text{and} \quad -2 \leq x_2 \leq 2 \quad \text{global minima}(f_{22}) = f_{22}(-0.0898, 0.7126) \text{ and} \\ f_{22}(0.0898, -0.7126) = -1.0316$$

V. Shekel Foxholes Function

$$f_{23} = \left[\frac{1}{500} + \sum_{j=1}^m \frac{1}{j + \sum_{i=1}^D (x_i - a_{ij})^6} \right]^{-1}$$

$-65.536 \leq x_i \leq 65.536$ global minimum(f_{23}) = $f_{23}(-32, -32) = 0.998004$
where m is a constant number fixed in advance. It is recommended to set $m=24, 25$, or 30 . In our case $m=25$, and

$$a_{1j} = \{-32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, \\ 16, 32, -32, -16, 0, 16, 32\}$$

$$a_{2j} =$$

$$\{-32, -32, -32, -32, -32, -16, -16, -16, -16, -16, 0, 0, 0, 0, 0, 16, 16, 16, 16, 16\}$$

W. Hartman's Family

$$f(x) = - \sum_{i=1}^4 c_i \exp \left(- \sum_{j=1}^D a_{ij} (x_j - p_{ij})^2 \right)$$

with $D=3, 6$ for $f_{24}(x)$ and $f_{25}(x)$, respectively, $0 \leq x_i \leq 1$. The coefficients a, p and c are defined by Table I. Global minimum (f_{24}) = $f_{24}(0.114, 0.556, 0.852) = -3.86278$. Global minimum (f_{25}) = $f_{25}(0.201, 0.150, 0.477, 0.275, 0.311, 0.657) = -3.3237$.

Table I
Hartman Functions f_{24} and f_{25}

i	$a_{ij}, j=1, \dots, 6$						c_i	$p_{ij}, j=1, \dots, 6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

X. Shekel's Family

$$f(x) = - \sum_{i=1}^m \left(\sum_{j=1}^D (x_j - a_{ji})^2 + c_i \right)^{-1}$$

with $m= 5, 7,$ and 10 for $f_{26}(x), f_{27}(x)$ and $f_{28}(x)$, respectively, $0 \leq x_i \leq 10$. The coefficients a and c are defined by Table II. Global minimum (f_{26}) = $f_{26}(4, 4, 4, 4) = -10.1532$. Global minimum (f_{27}) = $f_{27}(4, 4, 4, 4) = -10.4029$. Global minimum (f_{28}) = $f_{28}(4, 4, 4, 4) = -10.5364$.

Table II
Shekel Functions f_{26}, f_{27} and f_{28}

i	$a_{ij}, j=1, \dots, 4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
0	7	3.6	7	3.6	0.5

APPENDIX B

TRANSFORMED BENCHMARK FUNCTIONS

UNIMODAL FUNCTIONS

A. Shifted Schwefel 1.2 Function

$$F_2(x) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2, \quad z = x - o \quad -100 \leq x_i \leq 100$$

$x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$ is the shifted global optimum
global minimum (F_2) = $F_2(o, \dots, o) = 0$

MULTIMODAL FUNCTIONS

B. Shifted Rotated Ackley Function

$$F_6 = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + \exp(1),$$

$$z = (x - o) \times M \quad -32 \leq x_i \leq 32$$

$x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$: is the shifted global optimum
 M is a linear transformation matrix with condition number =1,
global minimum (F_6) = $F_6(o, \dots, o) = 0$

C. Shifted Rotated Griewank Function without bounds

$$F_8 = \frac{1}{4000} \sum_{i=1}^D z_i^2 - \prod_{i=1}^D \cos \left(\frac{z_i}{\sqrt{i}} \right) + 1, \quad z = (x - o) \times M$$

$x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$: is the shifted global optimum
 $M = \hat{M}(1 + 0.3|N(0,1)|)$ where \hat{M} is a linear transformation matrix with condition
number =3
global minimum (F_8) = $F_8(o, o, \dots, o) = 0$

D. Shifted Rastrigin Function

$$F_9 = \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10], \quad z = x - o \quad -5 \leq x_i \leq 5$$

$x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$: is the shifted global optimum
global minimum (F_9) = $F_9(o, o, \dots, o) = 0$

E. Shifted Rotated Rastrigin Function

$$F_{10} = \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10], \quad z = (x - o) \times M \quad -5 \leq x_i \leq 5$$

$x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$: is the shifted global optimum
 M is a linear transformation matrix with condition number = 2
global minimum (F_{10}) = $F_{10}(o, o, \dots, o) = 0$

APPENDIX C

STANDARD DE/RAND/1/BIN DELPHI 7 SOURCE CODE

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, StdCtrls, Buttons;

type
  TForm1 = class(TForm)
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);
const
  min=-100;
  max=100;
  f= 0.4;
  CR=0.9;
  P=50;
type
  rec=record
    vector:array[1..P] of Real;
    donor_vector: array[1..P] of real;
    trial_vector: array[1..P] of real;
    new_vector: array [1..P] of real;
    fitness: real;
    fitness1:real;
  end;
  population= array[1..500] of rec;

var
  pop:population;
  tempvec:array[1..P] of real;
  i, j, Np, D, r1, r2, r3, k, Irand, t, generation, Gn, ind: Integer;
  r, minn:Real;
  ff:Textfile;
```

```

begin

    Randomize;
    AssignFile(ff, 'Min. dat');
    Rewrite(ff);

    {***** Algorithm's Variables *****)
    D:= StrToInt(Edit1.Text);
    Np:= StrToInt(Edit2.text);
    Gn:= StrToint ( Edit3.Text);

    {***** Step1: Initialization of Population with target vectors *****)
    for i:= 1 to Np do
        for j:= 1 to D do
            pop[i].vector[j]:= min+(max-min)*random;

    {***** Evolution Steps (Mutation, Crossover, Evaluation and Selection)*****)

generation:=0;
while (generation <= Gn) do
    begin

    {***** Step 2: Evaluation of Target Vectors using Sphere Function *****)
        for i:=1 to Np do
            begin
                pop[i].fitness:=0.0;
                for j:=1 to D do
                    pop[i].fitness:= Sqr(pop[i].vector[j])+pop[i].fitness;
                end;

    {***** Print the Minimum Fitness to a file *****)
    minn:= Pop[1].fitness;
    for i:=2 to Np do
        if(pop[i].fitness < minn) then minn:= Pop[i].fitness;

    Edit4.text:= FloatToStr(minn);

    writeln(ff);
    writeln(ff, 'generation'+ INTTOSTR(GENERATION));
    writeln(ff);
    for t:=1 to Np do
        begin
            for j:=1 to D do
                write(ff, pop[t].vector[j], ' ');
            Writeln(ff, ' ', pop[t].fitness);
        end;

    {***** Step 3: Mutation Operation (DE/rand/1) *****)
    i:=1;
    while ( i<= Np ) do
        begin
            repeat
                r1:= Random(Np)+1;
            until (r1 <> i);

```

```

repeat
  r2:= Random(Np)+1;
until (r2 <> i)and(r2 <> r1);

repeat
  r3:= Random(Np)+1;
until (r3 <> i)and (r3 <> r1)and (r3 <> r2);

for k:=1 to D do
  pop[i].donor_vector[k]:=pop[r1].vector[k]+f*(pop[r2].vector[k]-
    pop[r3].vector[k]);

  i:=i+1;
end;

{***** Step 4: Boundary Constraints *****)
for i:=1 to Np do
  for j:=1 to D do
    if(pop[i].donor_vector[j]< min)or( pop[i].donor_vector[j]> max) then

    pop[i].donor_vector[j]:= min+(max-min)*random;

{***** Step 5: Crossover Operation (bin) *****)
for i:=1 to Np do
begin
  Irand:= Random(D)+1;
  for j:=1 to D do
  begin
    r:=Random;
    if ((r <= CR)or(j=Irand))then pop[i].trial_vector[j]:= Pop[i].donor_vector[j]
    else pop[i].trial_vector[j]:= Pop[i].vector[j];
  end;
end;

{***** Step 6: Evaluation of Trial Vectors using Sphere Function *****)
for i:=1 to Np do
begin
  pop[i].fitness1:=0.0;
  for j:=1 to D do
    pop[i].fitness1:= Sqr(pop[i].trial_vector[j])+pop[i].fitness1;
  end;

{*****Step 7: Selection Operation *****)
for i:=1 to Np do
  if (pop[i].fitness1 <= pop[i].fitness) then
  for j:=1 to D do
  begin
    pop[i].new_vector[j]:=pop[i].trial_vector[j];
  end
  else
  for j:=1 to D do
  begin
    pop[i].new_vector[j]:=pop[i].vector[j];
  end;

```

```
{***** Step 8: Exchange the old Target Vectors with New Target Vectors*****}  
  for i:= 1 to Np do  
    for j:=1 to D do  
      pop[i].vector[j]:= Pop[i].new_vector[j];  
  
    generation := generation +1;  
  end;  
  
  Closefile(ff);  
end;
```

LIST OF PUBLICATIONS

ACADEMIC JOURNALS

1. **Rawaa Dawoud Al-Dabbagh**, Azeddien Kinsheel, Mohd Sapiyan Baba and Saad Mekhilef. *A combined compact genetic algorithm and local search method for optimizing the ARMA(1,1) model of a likelihood estimator*. ScienceAsia. 40S(1). February 2014. Pages 78-86. **(ISI-Published)**
2. **Rawaa Dawoud Al-Dabbagh**, Azeddien Kinsheel, Saad Mekhilef, Mohd Sapiyan Baba and Shahab Shamshirband. *System Identification and Control of Robot Manipulator based on Fuzzy Adaptive Differential Evolution Algorithm*. Advances in Engineering Software. Volume 71. December 2014. Pages 60-66. **(ISI-Published)**
3. **Rawaa Dawoud Al-Dabbagh**, Saad Mekhilef and Mohd Sapiyan Baba. *Parameters' fine tuning of differential evolution algorithm*. Computer Systems Science and Engineering. Vol 30 No 2. March 2015. **(ISI-Published)**
4. Mohanad Dawood Al-Dabbagh, **Rawaa Dawoud Al-Dabbagh**, R. S. A. Raja Abdullah, and F. Hashim *A new modified differential evolution algorithm scheme-based linear frequency modulation radar signal de-noising*. Engineering Optimization. 47(6). June 2015. Pages 771-787. **(ISI-Published)**

PROCEEDINGS

1. **Rawaa Dawoud Al-Dabbagh**, Mohd Sapiyan Baba, Saad Mekhilef and Azeddien Kinsheel. *The Compact Genetic Algorithm for Likelihood Estimator of First Order Moving Average Model*. 2012 2nd International Conference on Digital Information and Communication Technology and its Applications, DICTAP 2012. January 2012. Pages 474-481.
2. **Rawaa Dawoud Al-Dabbagh**, Azeddien Kinsheel, Mohd Sapiyan Baba and Saad Mekhilef. *An Integration of Compact Genetic Algorithm and Local Search Method for Optimizing ARMA (1, 1) Model of Likelihood Estimator*. 2nd International Conference on Computer Science and Computational Mathematics (ICCSM 2013). February 2013. Pages 60-67.
3. Mohanad Dawood Al-Dabbagh, R. S. A. Raja Abdullah, **Rawaa Dawoud Al-Dabbagh** and F. Hashim. *Differential Evolution Algorithm for Linear Frequency Modulation Radar Signal Denoising*. IEEE RF and Microwave Conference (RFM) 2013. December 2013. Pages 234-237.

4. **Rawaa Dawoud Al-Dabbagh**, János Botzheim and Mohanad Dawood Al-Dabbagh. *Comparative analysis of a modified differential evolution algorithm based on bacterial mutation scheme*. IEEE SSCI Symposium on DE (SDE) 2014. December 2014. Pages 1-8.
5. **Rawaa Dawoud Al-Dabbagh**. *Differential Evolution with Adaptive Repository of Strategies and Parameter Control Schemes*. IEEE CEC 2015. May 2015.