

ENHANCED VARIANTS OF DIFFERENTIAL EVOLUTION ALGORITHM AND THEIR APPLICATIONS

A THESIS

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

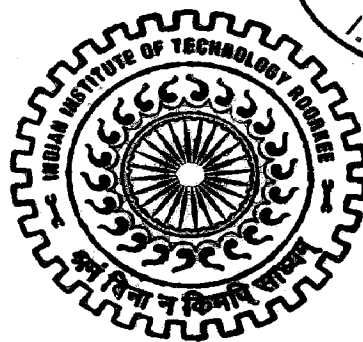
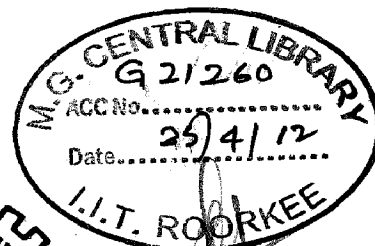
DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

by

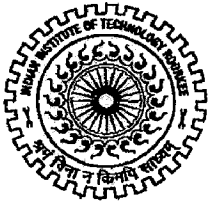
MUSRRAT ALI



DEPARTMENT OF PAPER TECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

JULY, 2011

**©INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, ROORKEE- 2011
ALL RIGHTS RESERVED**

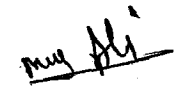


INDIAN INSTITUTE OF TECHNOLOGY ROORKEE ROORKEE

CANDIDATE'S DECLARATION

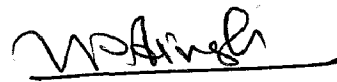
I hereby certify that the work which is being presented in this thesis entitled **ENHANCED VARIANTS OF DIFFERENTIAL EVOLUTION ALGORITHM AND THEIR APPLICATIONS** in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of Paper Technology of Indian Institute of Technology Roorkee is an authentic record of my own work carried out during a period from January, 2007 to July, 2011 under the supervision of Dr. Millie Pant, Asst. Professor and Dr. V. P. Singh, (formerly) Professor, Department of Paper Technology, Indian Institute of Technology Roorkee, Director, Millennium Institute of Technology, Saharanpur, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

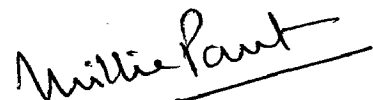

(MUSRRAT ALI)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date: July 15, 2011

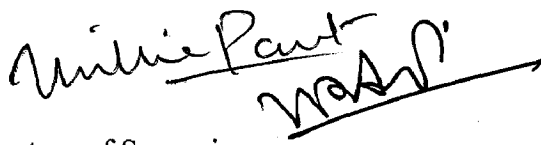


(V. P. Singh)
Supervisor

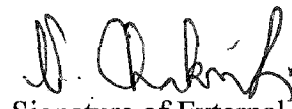


(Millie Pant)
Supervisor

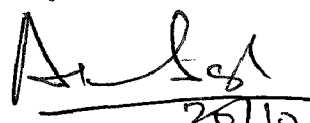
The Ph.D. Viva-Voce Examination of **Mr. Musrrat Ali**, Research Scholar, has been held on...20.10.11....

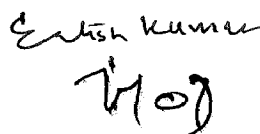


Signature of Supervisors



Signature of External Examiner


20/10
Chairman, etc


21/07

ABSTRACT

Most of the real life optimization problems arising in various fields of science and engineering can be modeled as global optimization problems. In such problems it is desired and is often necessary to determine a global optimal solution rather than a local optimal solution. Determining the global optimal solution of a nonlinear optimization problem is considered to be more difficult as compared to the problem of determining a local optimal solution. The various approaches available for solving the global optimization problems can be broadly categorized as deterministic and probabilistic approaches.

Deterministic approaches extensively use the analytical properties such as continuity, convexity, differentiability etc. of the objective and the constraints to locate a neighborhood of the global optimum. Most of these techniques are designed to solve a particular class of optimization problem. Consequently, these techniques are not generic in nature.

On the other hand stochastic methods, utilize randomness in an efficient way to explore the set over which the objective function is to be optimized. Stochastic methods performed well in the case of the most of the realistic problems over which these have been applied.

Among stochastic approaches, Evolutionary Algorithms (EA) or Nature Inspired Algorithms (NIA) are found to be very promising search techniques for solving global complex optimization problems. Some popular EA/NIA includes Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Differential Evolution (DE) etc.

The focus of the present study is on DE, which has emerged as a powerful optimization tool for solving complex global optimization problems. Comparative studies have confirmed that DE outperforms many other optimizers. Practical experiences however show that DE is not completely flawless. It is vulnerable to problems like slow and/ or premature convergence, is sometimes unable to locate global optima or gets stuck in local optima. Also, like most of the other population based EA/NIA, the performance of DE deteriorates with the increase in the size of the problem. These shortcomings of DE become more persistent in case of multimodal or noisy functions.

This study concentrates on development of efficient variants of DE algorithm for solving global optimization problems. Initially, three variants of DE are proposed named as: Differential Evolution with Cauchy mutation (CDE), Differential Evolution with mixed mutation strategy (MSDE) and Synergetic Differential Evolution (SDE).

The first variant CDE maintains a *failure_counter* (FC) to keep a tab on the performance of the algorithm by scanning or monitoring the individuals and makes use of Cauchy mutation operator in an effective manner which helps in escaping the individual entrapped in local minima.

The second variant MSDE uses the concept of evolutionary game theory to integrate basic DE mutation and quadratic interpolation based mutation to generate a new solution. This is contrast to the basic DE where a single mutation operator is used throughout the algorithm.

The third version which is SDE is based on the rule of synergy that the combined effect is always beneficial than the individualistic effect. In SDE three algorithmic components are fused together. These concepts are three recent modifications in DE (1) opposition based learning (OBL) (2) tournament method for mutation and (3) single population structure. These features have a specific role which helps in improving the performance of DE.

First of all the performance of these algorithms are analyzed on a set of unconstrained benchmark problems. For this purpose, a comprehensive set of well-known complex benchmark functions is employed to experimentally compare and analyze the three proposed algorithms. The test suite consists of 25 traditional/ classical problems and 7 nontraditional shifted benchmark problems. All the algorithms are also compared with some of the recent modified versions of DE and the results showed that the proposed variants are either superior or at par with the competing algorithms.

The algorithms are also analyzed statistically with the help of non parametric tests. It is observed that although all the proposed variants achieve solutions with good accuracy, maintain stable convergence characteristics and are simple to implement within a satisfactory computation time; MSDE and SDE are better than CDE.

MSDE and SDE after suitable modifications are further applied to solve the constrained optimization problems (COP). For this purpose a comprehensive set of 24 constrained benchmark problems is considered over which the proposed algorithms are analyzed and are

ompared with other modified DE versions for solving COP. The numerical and statistical results indicated that both the algorithms performed quite satisfactorily over the considered test suite of COP. However, taking into account the simple structure of SDE algorithm, it is further modified for solving multi objective optimization problems (MOPs). For analyzing the performance of SDE on MOPs, 9 unconstrained, bi-objective MOPs are taken from literature and the obtained results are also compared with some of the contemporary algorithms for solving MOPs. The efficiency of SDE was observed numerically and statistically for dealing with multiple objectives as well.

Finally, the proposed SDE algorithm is applied for solving two real life problems; (1) Trim Loss Problem (TLP) arising in paper industry, which is a highly constrained non-linear, non convex integer programming problem and (2) Image Thresholding Problem which arises frequently in the area of Image Processing. The complex mathematical model of both the problems makes it a challenging task for an optimization algorithm to obtain the global solution and it was observed that SDE was able to deal with both the real life problems in quite an efficient manner giving quality solution while maintaining a reasonably good convergence speed.

ACKNOWLEDGEMENTS

This work would never have been accomplished without God Almighty with His blessings and His power that work within me.

I wish to acknowledge my deep sense of gratitude and indebtedness to Dr. Millie Pant, Assistant Professor, Department of Paper Technology (DPT), IIT Roorkee and Dr. V. P. Singh, (formerly Professor, DPT, IIT Roorkee), Director, MIT, Saharanpur for their invaluable guidance, sincere advice and encouragement throughout the completion of this research work. I feel highly privileged to have worked under them during the course of this work.

Their encouraging support and thorough involvement in formulation of the research work and preparation of manuscript are gratefully acknowledged. I sincerely appreciate their pronounced individualities, humanistic and warm personal approach, which has given me strength to carry out this research work smoothly. I humbly acknowledge a lifetime's gratitude to them.

My heartily gratitude to Prof. J. S. Upadhyaya, former Head of the Department of Paper Technology and Prof. Satish Kumar Head of the Department of Paper Technology and Dr. Kusum Deep, Department of Mathematics, IIT Roorkee, whose humanistic and warm personal approach always helped me from beginning to end of my work.

I express my deep sense of gratitude to my co-authors Prof. Ajith Abraham, MIR Labs, USA and Prof. Atulya Nagar, Liverpool Hope University, Liverpool UK, for their valuable discussions, suggestions and encouragements during the work.

I also owe very special thanks to Prof. Patrick Siarry, Advisor, Sandwich Ph.D. at 'Laboratoire Images, Signaux et Systèmes Intelligents (L.I.S.S.I.)', Université de Paris XII, France for believing in me and inviting me to work under his supervision. The continuous encouragement by Dr. Amir Nakib, in the same laboratory, to implement my work on image processing is thankfully acknowledged.

For the financial support I would like to thank the Ministry of Human Resource Development (MHRD), Govt. of India. I also gratefully acknowledge the Ministry of External Affairs of France for financial support of six months to perform research work in France.

I thanks my friends Manoj Kumar, Kumar Manoj, Sanoj Kumar, Gaurav Bhatnagar,

Parveen Kumar, Abbas El-Dor, Tahar Brahimi, Nagma, Farah, Saba and Tahnga Raj for their positive attitude and cooperation always encouraged me to accomplish my research work. I also thank my seniors Dr. Jagdish Chand Bansal, Dr. K.P. Singh, Dr. Sanjeev Malik, Deepak Kumar and Dr. Ashok Kumar. My sincere thanks also to my colleagues Radha, Tarun, Pinki, Shashi and Parvesh for helping me, being patient and accessible in answering my numerous questions.

My special, sincere, heartfelt gratitude and indebtedness are due to my Mother, brothers and sisters for their sincere prayers, and constant encouragement. Any number of words is simply inadequate to encapsulate a lifetime of gratitude to my Mother. How do I do it in one paragraph? Just I say Thank you Mother. You made me who I am. You deserve the blame.

My only regret however, is for not finishing my Thesis earlier, before my father passed away. I miss watching my father's eyes filled with sincere love and pride. I am confident, though, that in his eternal peace he is proud of me as he was through all my life and as I have been proud of him. With my sincere love, love without ego, this is my gift to the living memory of my father for his exceptional strength, boundless love and support as well as for his unforgettable spirit.

Finally, I would like to thank everybody who was important to the successful realization of Thesis, as well as expressing my apology that I could not mention personally one by one.

Acknowledge Him in all Thy ways and He shall direct Thy paths.

MUSRRAT ALI

CONTENTS

ABSTRACT.....	i
ACKNOWLEDGEMENTS.....	v
CONTENTS.....	vii
LIST OF TABLES.....	xiii
LIST OF FIGURES.....	xix
Chapter 1 INTRODUCTION.....	1
1.1 What is Optimization.....	1
1.2 Formal Definition.....	2
1.3 Local and Global Optimal Solutions.....	3
1.4 Nature Inspired Computational Search Techniques.....	4
1.5 Differential Evolution.....	5
1.6 Working of Differential Evolution.....	11
1.7 Survey of Literature.....	12
1.8 Applications.....	16
1.9 Objective of the Present Work.....	17
1.10 Thesis Overview and Organization.....	17
Chapter 2 ENHANCED DE VARIANTS FOR UNCONSTRAINED OPTIMIZATION.....	19
2.1 Introduction.....	20
2.2 Differential Evolution with Cauchy Mutation (CDE).....	21
2.3 Differential Evolution with Mixed Mutation Strategy (MSDE).....	25
2.4 Synergetic Differential Evolution (SDE).....	28
2.4.1 Opposition Based Learning (OBL) for Generating the Initial Population	28
2.4.2 Randomized Localization for Selecting the Base Vector.....	30
2.4.3 Single Population Structure.....	31

2.5 Benchmark of Unconstrained Problems	33
2.6 Performance Metrics	34
2.7 Parameter Settings	35
2.8 Sensitivity Analysis of Additional Parameters Used in Proposed Algorithms	36
2.8.1 Analysis of Parameter Maximum Failure Counter (MFC).....	37
2.8.2 Sensitivity Analysis of Parameter γ	37
2.9 Comparison of the Proposed Algorithms with Basic DE, ODE, DERL and MDE.....	37
2.9.1 Comparison in Terms of Error and Standard Deviation (Std.)	38
2.9.2 Comparison in Terms of Number of Function Evaluations.....	38
2.9.3 Comparison in Terms of Acceleration Rate (AR)	38
2.9.4 Comparison in Terms of Success Rate (SR)	39
2.9.5 Statistical Analysis.....	39
2.10 Further Analysis of the Proposed Algorithms	41
2.10.1 Influence of Dimensionality.....	41
2.10.2 Influence of Varying the Population Size (NP).....	42
2.10.3 Effect of Jumping on SDE Algorithm.....	42
2.11 Numerical Results for Nontraditional Benchmark Problems	42
2.12 Comparison of SDE and MSDE With Other State of the Art Algorithms	43
2.12.1 Comparison in Terms of Fitness Value and Standard Deviation	43
2.12.2 Comparison in Terms of NFE	43
2.12.3 Comparison in Terms of SR.....	44
2.12.4 Statistical Analysis	44
2.13 Summary	44

Chapter 3 CONstrained Differential Evolution.....67

3.1 Introduction	67
3.1.1 Constraint Handling Techniques Applied to Population Based Search Techniques	68
3.1.2 Application of DE for Solving COP	69

3.2 Constraint Handling Techniques Used in the Present Study	70
3.3 Test Problems	71
3.4 Performance Metrics	72
3.4.1 Algorithms Used for Comparison	72
3.5 Parameter Settings	73
3.6 Results and Discussions.....	73
3.6.1 Results of MSDE and SDE.....	73
3.6.2 Comparison with other Algorithms	74
3.7 Summary.....	75

Chapter 4 SDE FOR MULTI-OBJECTIVE OPTIMIZATION..... 93

4.1 Introduction	93
4.1.1 Application of DE for solving MOP.....	94
4.2 Background.....	95
4.2.1 Multi-Objective Optimization Problem (MOP)	95
4.2.2 Pareto Dominance.....	95
4.2.3 Fast Nondominated Sorting.....	96
4.2.4 Crowding Distance Metric.....	96
4.3 Multi-Objective Synergetic Differential Evolution (MO-SDE).....	97
4.3.1 Working of MO-SDE	97
4.3.2 Proposed Selection Mechanism Used in MO-SDE	97
4.3.3 Effect of Using the Proposed Selection Mechanism	98
4.3.4 Computational Complexity Analysis.....	98
4.4 Experiments.....	100
4.4.1 Test Problems	100
4.4.2 Experimental Setup.....	101
4.4.3 Performance Metric I.....	101
4.4.4 Performance Metric II	102
4.4.5 Algorithms Used for Comparison.....	102
4.5 Results and Discussions.....	103

4.5.1	Results Based on Performance Metric I	103
4.5.2	Results Based on Performance Metric II.....	104
4.6	Summary	105
Chapter 5	TRIM LOSS OPTIMIZATION	113
5.1	Introduction	113
5.2	Mathematical Formulation	117
5.3	Implementation of SDE for Trim Loss Problem	119
5.3.1	Handling of Integers and Binary Variables in SDE	119
5.3.2	Handling Constraint and Boundary Violations	119
5.3.3	Control Parameter Settings.....	120
5.3.4	Numerical Results	120
5.4	Summary	121
Chapter 6	MULTI-LEVEL IMAGE THRESHOLDING	129
6.1	Introduction	129
6.2	Methods Used for Image Thresholding.....	131
6.2.1	Gaussian Curve Fitting.....	132
6.2.2	Entropy Criterion.....	133
6.3	Results and Discussions	134
6.4	Summary	136
Chapter 7	CONCLUSIONS AND FUTURE RESEARCH	145
7.1	Conclusions	145
7.2	Future Research.....	147
BIBLIOGRAPHY		149
Appendix I	LIST OF UNCONSTRAINED TEST PROBLEMS	163
Appendix II	LIST OF SHIFTED TEST PROBLEMS	177
Appendix III	LIST OF CONSTRAINED TEST PROBLEMS	185

Appendix IV LIST OF MULTI-OBJECTIVE TEST PROBLEMS.....	203
Appendix V NON PARAMETRIC TESTS	207
LIST OF PUBLICATION	213

LIST OF TABLES

Table 2.1	Sensitivity analysis of Maximum Failure Counter (MFC) in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	46
Table 2.2	Sensitivity analysis of parameter Gama (γ) in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	47
Table 2.3	Comparison of proposed algorithms with DE and their parent algorithms in terms of error and standard deviation (Std.).....	48
Table 2.4	Comparison of proposed algorithms with DE and their parent algorithms in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	50
Table 2.5	Comparison of proposed algorithms with DE and their parent algorithms in terms of AR. '--' indicates that AR cannot be calculated for them.....	51
Table 2.6	Comparison of proposed algorithms with DE and their parent algorithms in terms of SR.....	52
Table 2.7	Results of Friedman's test based on error of Table 2.3.....	52
Table 2.8	Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.3.....	52
Table 2.9	Results of pairwise comparison based on error of Table 2.3.....	53
Table 2.10	Results of Friedman's test based on NFE of Table 2.4.....	53
Table 2.11	Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.4.....	53
Table 2.12	Results of pairwise comparison based on NFE of Table 2.4.....	53
Table 2.13	Influence of dimensionality on the proposed algorithms and DE in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	54

Table 2.14	Influence of dimensionality on the proposed algorithms and DE in terms of SR.....	54
Table 2.15	Influence of varying population size on the proposed algorithms and DE in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	55
Table 2.16	Influence of varying population size on proposed algorithms and DE in terms of SR.....	56
Table 2.17	Effect of jumping on proposed SDE algorithm with jumping rates as 0.1 and 0.3. The results are tabulated for number of function evaluations (NFE). Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	57
Table 2.18	Comparison of proposed algorithms with DE and ODE for nontraditional shifted functions in terms of error (best median, worst and mean) and standard deviation (Std).....	58
Table 2.19	Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of fitness function value.....	59
Table 2.20	Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.....	60
Table 2.21	Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of SR.....	61
Table 2.22	Results of Friedman's test based on NFE of Table 2.20.....	61
Table 2.23	Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.20.....	61
Table 2.24	Results of pairwise comparison based on NFE of Table 2.20.....	62
Table 3.1	Name of constrained test problems, assigned codes and characteristics...	71
Table 3.2	Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 1-6.....	76
Table 3.3	Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 ,	

	NFEs = 5×10^5 for problems 07-12.....	76
Table 3.4	Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 13-18.....	77
Table 3.5	Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 19-24.....	77
Table 3.6	Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$, Success Rate, Feasibility Rate and Success Performance by MSDE.....	78
Table 3.7	Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 1-6.....	79
Table 3.8	Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 07-12.....	79
Table 3.9	Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 13-18.....	80
Table 3.10	Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 19-24.....	80
Table 3.11	Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$, Success Rate, Feasibility Rate and Success Performance by SDE.....	81
Table 3.12	Comparison of MSDE and SDE on the basis of error and standard deviation.....	82
Table 3.13	Comparison of MSDE and SDE on the basis of Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$	83
Table 3.14	Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$, Success Rate, Feasibility Rate and Success Performance by all the algorithms for problems 01-12.....	84
Table 3.15	Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$, Success Rate, Feasibility Rate and Success Performance by all	

	the algorithms for problems 13-24.....	86
Table 3.16	Results of Friedman's test based on NFEs.....	88
Table 3.17	Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure.....	88
Table 3.18	Results of pairwise comparison based on NFEs.....	88
Table 4.1	Statistics of the results on the test problem SCH.....	107
Table 4.2	Statistics of the results on the test problem FON.....	107
Table 4.3	Statistics of the results on the test problem POL.....	107
Table 4.4	Statistics of the results on the test problem KUR.....	107
Table 4.5	Statistics of the results on the test problem ZDT1.....	107
Table 4.6	Statistics of the results on the test problem ZDT2.....	108
Table 4.7	Statistics of the results on the test problem ZDT3.....	108
Table 4.8	Statistics of the results on the test problem ZDT4.....	108
Table 4.9	Statistics of the results on the test problem ZDT6.....	109
Table 4.10	Statistical results by Wilcoxon test for convergence.....	109
Table 4.11	Statistical results by Wilcoxon test for diversity.....	109
Table 5.1	Example order 1.....	122
Table 5.2	Example order 2.....	122
Table 5.3	Example order 3.....	122
Table 5.4	Example order 4.....	122
Table 5.5	Parameters of the problems.....	122
Table 5.6	Results for the Trim-Loss Problem 1.....	123
Table 5.7	Results for the Trim-Loss Problem 2.....	123
Table 5.8	Results for the Trim-Loss Problem 3.....	124
Table 5.9	Results for the Trim-Loss Problem 4.....	125
Table 5.10	Solution results for problem 1.....	126
Table 5.11	Solution results for problem 2.....	126
Table 5.12	Solution results for problem 3.....	126
Table 5.13	Solution results for problem 4.....	126

Table 5.14	Alternate optimal solutions of problem 1.....	127
Table 5.15	Alternate optimal solutions of problem 2.....	127
Table 5.16	Best, worst, mean fitness and standard deviation, average NFE, time and success rate for all problems.....	127
Table 5.17	Comparison of SDE, ILXPSO and RCGA on average NFE and success rate for all problems.....	127
Table 6.1	Results obtained by SDE_Gaus for images given in Figure 6.1.....	141
Table 6.2	Comparison of SDE_Gaus with basic PSO and GA.....	141
Table 6.3	Experimental results of SDE_Entropy for images given in Figure 6.1.....	141
Table 6.4	Comparison of algorithms in terms of thresholds.....	142
Table 6.5	Comparison of algorithms in terms of fitness corresponding to threshold given in Table 6.4.....	143
Table 6.6	Comparison of algorithms in terms uniformity measure.....	143

LIST OF FIGURES

Figure 1.1	Global and local optima.....	4
Figure 1.2	Illustrating simple DE mutation scheme of equation (1.2).....	7
Figure 1.3	Illustrating binomial crossover scheme of DE.....	8
Figure 1.4	Illustrating exponential crossover scheme of DE.....	9
Figure 1.5	Illustrating selection scheme of DE.....	10
Figure 2.1	Probability density function of Cauchy distribution for different values of y_0 and x_0	23
Figure 2.2	Flow chart of CDE.....	24
Figure 2.3	Parabolic curve of quadratic interpolation.....	26
Figure 2.4	Flow chart of MSDE.....	26
Figure 2.5	Flow chart of SDE.....	32
Figure 2.6	Performance graphs (best solution versus NFE) for performance comparison between DE and SDE. (a) – (j) represents functions $f_1, f_2, f_3, f_4, f_7, f_{10}, f_{11}, f_{12}, f_{13}$, and f_{14}	65
Figure 2.7	Bonferroni-Dunn’s graphic corresponding to error of Table 2.3.....	66
Figure 2.8	Bonferroni-Dunn’s graphic corresponding to NFE of Table 2.4.....	66
Figure 2.9	Bonferroni-Dunn’s graphic corresponding to NFE of Table 2.20.....	66
Figure 3.1	Convergence graph for problems 01-06 of SDE.....	89
Figure 3.2	Convergence graph for problems 07-12 of SDE.....	89
Figure 3.3	Convergence graph for problems 13-18 of SDE.....	89
Figure 3.4	Convergence graph for problems 19-24 of SDE.....	90
Figure 3.5	Convergence graph for problems 01-06 of MSDE.....	90
Figure 3.6	Convergence graph for problems 07-12 of MSDE.....	90
Figure 3.7	Convergence graph for problems 13-18 of MSDE.....	91
Figure 3.8	Convergence graph for problems 19-24 of MSDE.....	91
Figure 3.9	Bonferroni-Dunn’s graphic.....	91
Figure 4.1	True PF and nondominated solutions by MO-SDE on SCH.....	110

Figure 4.2	True PF and nondominated solutions by MO-SDE on FON.....	110
Figure 4.3	True PF and nondominated solutions by MO-SDE on POL.....	110
Figure 4.4	True PF and nondominated solutions by MO-SDE on KUR.....	111
Figure 4.5	True PF and nondominated solutions by MO-SDE on ZDT1.....	111
Figure 4.6	True PF and nondominated solutions by MO-SDE on ZDT2.....	111
Figure 4.7	True PF and nondominated solutions by MO-SDE on ZDT2.....	112
Figure 4.8	True PF and nondominated solutions by MO-SDE on ZDT4.....	112
Figure 4.9	True PF and nondominated solutions by MO-SDE on ZDT6.....	112
Figure 5.1	The pulp and paper supply chain.....	114
Figure 5.2	A schematic illustration of trim loss problem.....	114
Figure 5.3	The cutting pattern.....	115
Figure 5.4	Plot of fitness and constraint violation versus NFE for problem 1.....	128
Figure 5.5	Plot of fitness and constraint violation versus NFE for problem 2.....	128
Figure 6.1	Test images and their normalized histograms. (a) Lena, (b) Pepper, (c) Cameraman, (d) histogram of Lena image, (e) histogram of Pepper image and (f) histogram of Cameraman image.....	137
Figure 6.2	Results of Lena image (a) segmented image with three classes by curve, (b) segmented image with three classes by entropy, (c) segmented image with five classes by curve, (d) segmented image with five classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d).....	138
Figure 6.3	Results of Pepper image (a) segmented image with three classes by curve, (b) segmented image with three classes by entropy, (c) segmented image with four classes by curve, (d) segmented image with four classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d).....	139
Figure 6.4	Results of Cameraman image (a) segmented image with three classes	

by curve, (b) segmented image with three classes by entropy, (c) segmented image with four classes by curve, (d) segmented image with four classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d)..... 140

Figure I.1	3-D map for 2-D function f_1	163
Figure I.2	3-D map for 2-D function f_2	164
Figure I.3	3-D map for 2-D function f_3	164
Figure I.4	3-D map for 2-D function f_4	165
Figure I.5	3-D map for 2-D function f_5	165
Figure I.6	3-D map for 2-D function f_6	166
Figure I.7	3-D map for 2-D function f_8	167
Figure I.8	3-D map for 2-D function f_9	168
Figure I.9	3-D map for 2-D function f_{10}	168
Figure I.10	3-D map for 2-D function f_{11}	169
Figure I.11	3-D map for 2-D function f_{12}	170
Figure I.12	3-D map for 2-D function f_{13}	170
Figure I.13	3-D map for 2-D function f_{14}	171
Figure I.14	3-D map for 2-D function f_{16}	172
Figure I.15	3-D map for 2-D function f_{17}	172
Figure I.16	3-D map for 2-D function f_{18}	173
Figure II.1	3-D map for 2-D function F_1	177
Figure II.2	3-D map for 2-D function F_2	178
Figure II.3	3-D map for 2-D function F_3	179
Figure II.4	3-D map for 2-D function F_4	180
Figure II.5	3-D map for 2-D function F_5	180
Figure II.6	3-D map for 2-D function F_6	181
Figure II.7	3-D map for 2-D function F_7	182

Introduction

This chapter is introductory in nature; besides giving some basic definitions related to optimization, it provides a detailed description of Differential Evolution (DE); literature survey related to DE and its applications to various fields. The scope of the chapter is to introduce the motivation underneath this research work as well as the current challenges in this field. The proposed research objectives and contributions are also discussed followed by the thesis outline.

1.1 What is Optimization?

One of the most fundamental principles in our world is the search for an optimal state. It begins in the microcosm where atoms in physics try to form bonds in order to minimize the energy of their electrons (Pauling, 1960). When molecules form solid bodies during the process of freezing, they try to assume energy- optimal crystal structures. These processes, of course, are not driven by any higher intention but purely result from the laws of physics.

The same goes for the biological principle of survival of the fittest (Spencer, 1864) which, together with the biological evolution (Darwin, 1859), leads to a better adaptation of the species to their environment. Here, a local optimum is a well-adapted species that dominates all other animals in its surroundings. Homosapiens have reached this level, sharing it with ants, bacteria, lice, cockroaches, and all sorts of other creepy creatures.

As long as humankind exists, we strive for perfection in many areas. We want to reach a maximum degree of happiness with the least amount of effort. In our economy, profit and sales must be maximized and costs should be as low as possible. Therefore, optimization is one of the oldest of science which even extends into daily life (Neumaier, 2006). In the most basic sense, it can be defined as an art of selecting the best alternative among a given set of options. Global optimization is the branch of applied mathematics and numerical analysis that deals with the optimization of single or multiple even conflicting criteria. These criteria are called objective

functions. The result of an optimization process is the set of inputs for which these objective functions return optimal values.

1.2 Formal Definition

To formalize the concept of optimization, consider S to be the set of candidate solutions to the optimization problem. Typically S is D -dimensional over some domain, for example in case of binary: $S = \{0, 1\}^D$ or in case of real-values: $S \subseteq R^D$. The domain S is often referred to as the *search-space*.

Let the optimization problem be defined by the function f , which is called the fitness function (or cost function/ error function/ objective function) and rates how well the candidate solutions in S fare on the given problem.

Without lack of generalization this thesis considers minimization problems, that is, to minimize the fitness function f and hence obtain the candidate solution that fares best. Maximization problems can be optimized merely by introducing an auxiliary function. Suppose f is a fitness function to be maximized, then the analogous minimization problem can be considered instead, simply by introducing the function: $h(X) = -f(X)$

The general non-linear unconstrained optimization problem is defined as:

$$\text{Minimize } f(X) : X \in S \rightarrow R$$

that is, to minimize the fitness function f and hence obtain the candidate solution that fares best, find $X \in S$ so that:

$$\forall Y \in S : f(X) \leq f(Y)$$

Such a point X is known as a global minimum for the function f .

The general non-linear constrained optimization problem is defined as:

$$\text{Minimize } f(X) : X \in S \rightarrow R$$

Subject to: $X = (x_1, x_2, \dots, x_D) \in S$

where S is defined by:

$$g_j(X) \leq 0, \quad j = 1, 2, \dots, l$$

$$h_k(X) = 0, \quad k = 1, 2, \dots, m$$

$$l_i \leq x_i \leq u_i, \quad (i = 1, \dots, D)$$

l and m are the number of inequality and equality constraints respectively, l_i and u_i are lower and upper bounds of the decision variable x_i , respectively.

Given a function that contains multiple minima on its feasible set only the smallest minimum will be the global minimum and all others will be classified as local minima. In general, global minimum are difficult to locate and verify. The task of locating the global optimum is referred to as global optimization.

Optimization problems can be categorically differentiated according to the various properties of the objective function, constraints and decision variables. The objective function and the constraints (if any are present) can either be linear or nonlinear. The decision variables can be continuous or discrete or a combination of both.

If a problem has a linear objective function and linear constraints it is considered to be a linear optimization problem. If the solution has an additional requirement that the decision variables are integers then the model is called integer programming optimization problem. Whereas, if a problem has either a nonlinear objection function or constraints or both, it is classified as a nonlinear optimization problem. These definitions apply to problems with continuous decision variables. If however some decision variables are integers and some others are real then the problem is classified as mixed-integer problem, on the other hand if the variables are discrete then the problem is discrete optimization problem.

In this thesis the work is concentrated on continuous variables except for a real life problem in Chapter 5 which contains discrete as well as binary variables.

1.3 Local and Global Optimal Solutions

For a minimization problem, a feasible solution X^* is said to be a global minima of the problem if $f(X^*) \leq f(X)$ for all $X \in S$. If $f(X^*) \leq f(X)$ for all $X \in S \cap N_\varepsilon(X^*)$, where $N_\varepsilon(X^*)$ is called a ε neighborhood of X^* , then X^* is called local minima. A point X^* is a stationary point if the derivative of the function $f(X)$ is zero at X^* . Graphical illustration for local and global optima is given in Figure 1.1.

An optimization problem may have no optimal solution, only one optimal solution or more than one optimal solution. If the problem has a unique local optimal solution, then it is

also the global optimal solution. However, if the problem has more than one local optimal solution, then one or more of these may be global optimal solutions. In an LPP, every local optimal solution is also the global optimal solution, on the other hand in an NLPP, if the objective function (for minimization case) is convex and its domain of definition defined by the set of constraints is also convex, then the local optimal solution is also the global optimal solution.

In most of the NLPP, a global optimal solution rather than a local optimal solution is desired. Determining the global optimal solution of a NLPP is much more difficult than determining the local optimal solution. However, because of the practical reasons, the search for the global optima is often desired.

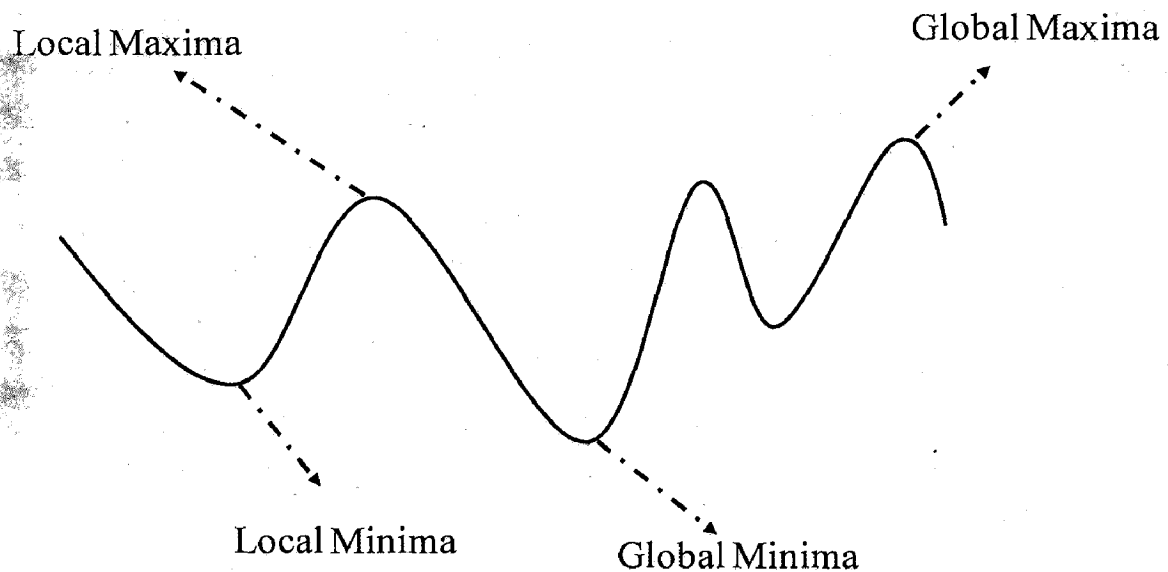


Figure 1.1: Global and local optima.

1.4 Nature Inspired Computational Search Techniques

Considering the practical utility of the global optimization algorithms, efforts have been made to develop efficient global optimization algorithms (Grosan and Abraham, 2009). In the past few decades emphasis has been laid on the development of general purpose algorithms that are problem independent and can efficiently determine the global optimal solution.

Nature Inspired Computational Search Techniques (NICST) are such general purpose algorithms that have gained significant attention of the researchers dealing with global optimization problems. Some popular NICST include Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Differential Evolution (DE) etc. As the name NICST suggests, these algorithms are based on some natural phenomena. For example while GA and DE are based on the Darwin's theory of *survival of the fittest* and *natural selection*; ACO and PSO are based on socio cooperative behaviour displayed by various species (Engelbrecht, 2005).

These algorithms have been developed and modified by several researchers for solving constrained and unconstrained global optimization problems (Liu et al., 2006; Liu et al., 2007; Ali, 2007; Ali and Bagdadi, 2009; Omran et al., 2009; Ghosh et al., 2010; Sabat et al., 2011). These algorithms have also been applied to a wide range of problems like Multi-Objective Optimization Problems (Grosan and Abraham, 2008); noisy functions (Mininno and Neri, 2010); heat exchanger design (Babu and Shaik, 2007); sequence alignment (Jangam and Chakraborty, 2007); assignment problems (Liu and Abraham, 2007); scheduling problems (Liu et al., 2010); Mixed Integer Non Linear Programming Problems (Shaik and Gudi, 2005); power systems (Panigrahi and Pandi, 2010), portfolio optimization (Suganya and Vijayalakshmi, 2010); electronics (Sabat et al., 2010, Sreelaja and Vijayalakshmi, 2010); chemical engineering (Mondal et al., 2011).

The focus of the present study is Differential Evolution (DE) which is relatively a newer NICST. Sections 1.5 – 1.8 are devoted to the detailed description of DE.

1.5 Differential Evolution

Storn and Price developed Differential Evolution (DE) to be a reliable and versatile function optimizer that is also easy to use. The first written publication on DE appeared as a technical report in 1995 (Storn and Price, 1995). Since then, DE has proven itself in competitions like the IEEE's International Contest on Evolutionary Optimization (ICEO) in 1996 and 1997 and in the real world on a broad variety of applications. Differential Evolution (DE) is relatively a new optimization technique in comparison to Evolutionary Algorithms (EAs) such as Genetic Algorithms (Goldberg, 1989), Evolution Strategies (Rochenberg, 1973),

Evolutionary Programming (Fogel, 1965), Ant Colony Optimization and Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995). Within a short span of around fifteen years, DE has emerged as one of the most simple and efficient technique for solving global optimization. Recently, Mathematica[®] added DE to its numerical optimizer package.

A general DE scheme may be defined as $DE/a/b/c$, where DE denotes the Differential Evolution algorithm; 'a' represents a string denoting the vector to be perturbed; it may be the best vector of the population or it may be a random vector. 'b' indicates the number of difference vectors considered for perturbation of a; it may be one or two. And 'c' stands for the type of crossover being used, binomial (*bin*) or exponential (*exp*).

Like all other population based search algorithms, DE starts with a population S of NP candidate solutions: $X_{i,G} = \{x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}\}$, $i = 1, \dots, NP$, where the index i denotes the i^{th} individual of the population, G denotes the generation to which the population belongs and D denotes the dimension of the problem. Each parameter of the problem, may have a certain range within which the value of the parameter should be restricted, often because parameters are related to physical components or measures that have natural bounds (for example if one parameter is a length or mass, it cannot be negative). The initial population should cover this range as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed lower and upper bounds: $l = \{l_1, l_2, \dots, l_D\}$ and $u = \{u_1, u_2, \dots, u_D\}$. Hence, j^{th} component of the i^{th} vector may be initialized as follows:

$$x_{j,i,0} = l_j + rand_{i,j}[0,1] \times (u_j - l_j) \quad (1.1)$$

where $rand_{i,j}[0, 1]$ is a uniformly distributed random number lying between 0 and 1, including both values, and is instantiated independently for each component of the i^{th} vector. The three main operators of DE are mutation, crossover and selection which may be defined as follows:

Mutation: Once the initialization is complete, DE enters the mutation phase. In this phase a *donor* vector $V_{i,G}$ is created corresponding to each member or *target* vector $X_{i,G}$ in the current generation. The method of creating donor vector differentiates one DE scheme from another. The most often used mutation strategies implemented in the DE codes are listed below.

$$\text{DE/rand/1: } V_{i,G} = X_{r_1,G} + F * (X_{r_2,G} - X_{r_3,G}) \quad (1.2)$$

$$\text{DE/rand/2: } V_{i,G} = X_{r_1,G} + F * (X_{r_2,G} - X_{r_3,G}) + F * (X_{r_4,G} - X_{r_5,G}) \quad (1.3)$$

$$\text{DE/best/1: } V_{i,G} = X_{best,G} + F * (X_{r_1,G} - X_{r_2,G}) \quad (1.4)$$

$$\text{DE/best/2: } V_{i,G} = X_{best,G} + F * (X_{r_1,G} - X_{r_2,G}) + F * (X_{r_3,G} - X_{r_4,G}) \quad (1.5)$$

$$\text{DE/rand-to-best/1: } V_{i,G} = X_{r_1,G} + F * (X_{best,G} - X_{r_2,G}) + F * (X_{r_3,G} - X_{r_4,G}) \quad (1.6)$$

The indices r_1, r_2, r_3, r_4 and r_5 are mutually exclusive integers randomly chosen from the range $[1, NP]$ and all are different from the base index i . These indices are randomly generated once for each vector. The scaling factor F is a positive control parameter and is used for scaling the difference vectors. $X_{best,G}$ is the individual having the best fitness function value in the population at generation G . Mutation scheme given in equation (1.2) is represented in Figure 1.2.

Frequently referred strategies implemented in the public-domain DE codes for producing the donor vectors are also available online at: <http://www.icsi.berkeley.edu/~storn/code.html>.

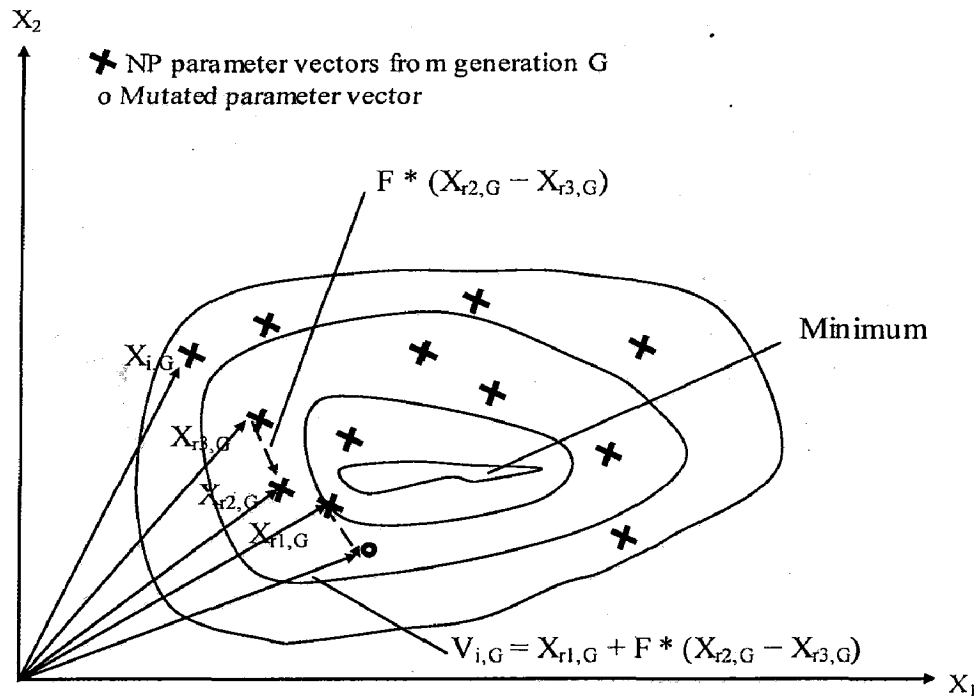


Figure 1.2: Illustrating simple DE mutation scheme of equation (1.2).

Crossover: once the donor vector is generated in the mutation phase, the crossover phase of DE is activated. The crossover operation of DE helps in increasing the potential diversity of the DE population. The DE family of algorithms may use two types of crossover schemes; *exponential (exp)* and *binomial (bin)*. During the crossover operation, the donor vector $V_{i,G} = \{v_{1,i,G}, v_{2,i,G}, v_{3,i,G}, \dots, v_{D,i,G}\}$ exchanges its components with the target vector $X_{i,G} = \{x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \dots, x_{D,i,G}\}$ to form a *trial* vector $U_{i,G} = (u_{1,i,G}, \dots, u_{D,i,G})$.

According to binomial crossover the trial vectors are generated as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } \text{rand}_{i,j}[0,1] \leq C_r, \forall j = j_{rand} \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (1.7)$$

where, $j = 1 \dots D$ and $j_{rand} \in \{1, \dots, D\}$ is a random parameter's index, chosen once for each i . C_r is a positive control parameter (crossover probability) set by the user. DE's version of binomial crossover begins by taking a randomly chosen parameter from the mutant vector so that the trial vector will not simply replicate the target vector. Comparing C_r to $\text{rand}_{i,j}[0,1]$ determines the source for each remaining trial parameter. If $\text{rand}_{i,j}[0,1] \leq C_r$, then the parameter comes from the mutant vector; otherwise, the target vector is the source. Figure 1.3 illustrates the binomial crossover process.

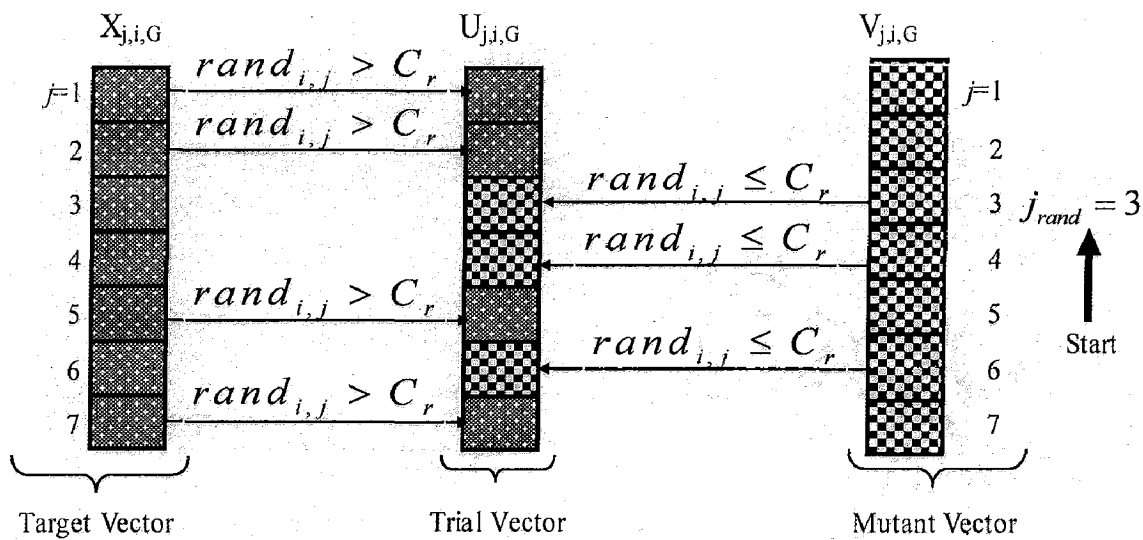


Figure 1.3: Illustrating binomial crossover scheme of DE.

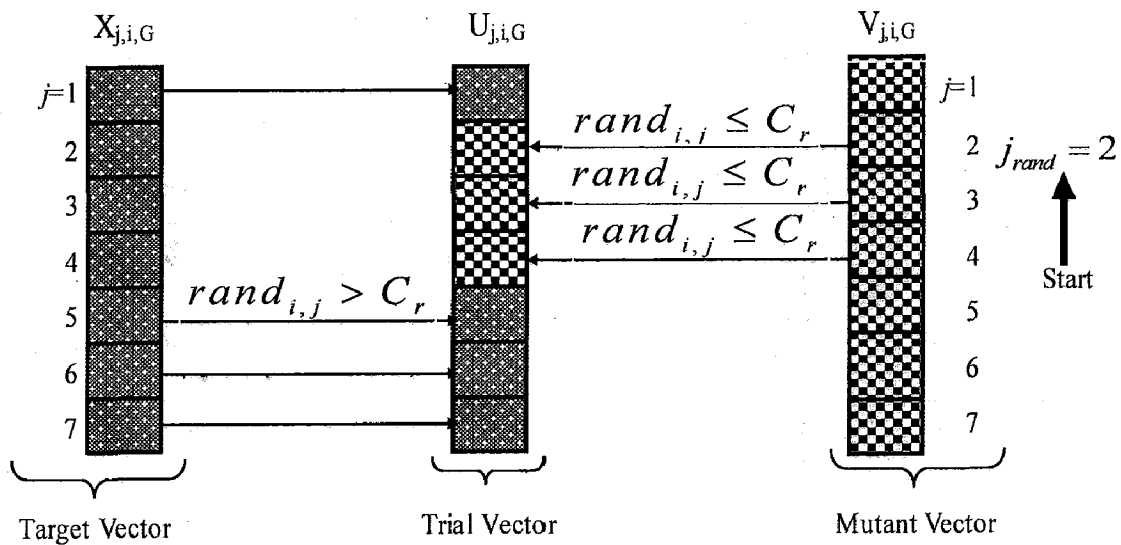


Figure 1.4: Illustrating exponential crossover scheme of DE.

In exponential crossover, as in binomial crossover, one parameter is initially chosen at random and copied from the mutant to the corresponding trial parameter so that the trial vector will be different from the vector with which it will be compared. The source of subsequent trial parameters is determined by comparing C_r to a uniformly distributed random number between 1 and 0 that is generated a new for each parameter. As long as $\text{rand}_{i,j}[0,1] \leq C_r$, parameters are taken from the mutant vector, but the moment, $\text{rand}_{i,j}[0,1] > C_r$, the current and the remaining parameters are taken from the target vector. Its graphical representation is given in Figure 1.4.

Selection: The final phase of the DE algorithm is that of selection, which determines whether the target or the trial vectors generated in the mutation and crossover phases will survive into the next generation. The population for the next generation is selected from the individual in current population and its corresponding trial vector according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (1.8)$$

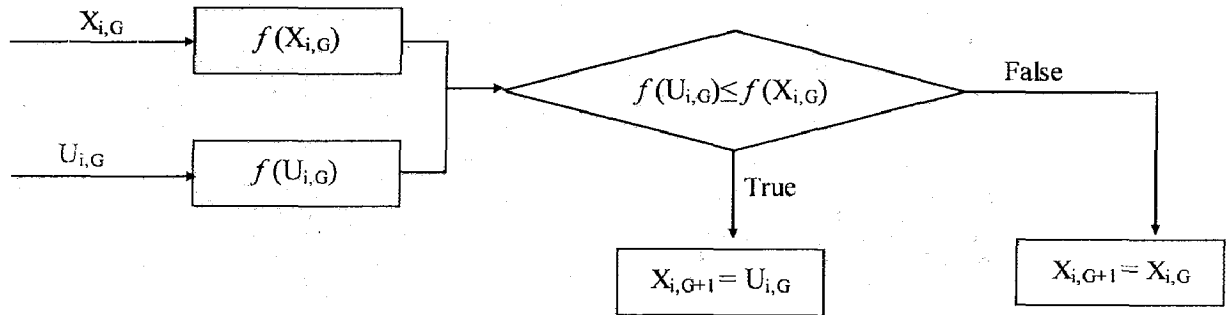


Figure 1.5: Illustrating selection scheme of DE.

Thus, each individual of the advance (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. In DE, the trial vector is not compared against all the individuals in the current generation, but only against its counterpart in the current generation.

Pseudocode of DE is given in Algorithm 1.1.

Algorithm 1.1 Pseudocode of DE illustrating how the procedure acts on a population of individuals, repeating mutation, crossover and selection until the convergence criteria is met.

- Step 1:* Generate randomly NP individuals X_i , $i=1,2,\dots, NP$, using equation (1.1). Set the values of control parameters F , Cr .
- Step 2:* Set $i = 0$.
- Step 3:* $i = i + 1$.
- Step 4:* Corresponding to target individual X_i select three distinct individuals X_{r1} , X_{r2} and X_{r3} such that $i \neq r1 \neq r2 \neq r3$ from population and generate perturbed individual V_i using equation (1.2) and go to step 5.
- Step 5:* Recombine the target vector X_i with perturbed individual V_i generated in step 4 to generate trial vector U_i using equation (1.7) and go to step 6.
- Step 6:* If all parameters of the trial vector is within the given range then go to step 7 otherwise uniformly generate that parameter within given range using equation (1.1) and go to step 7.
- Step 7:* Calculate the objective function value for vector U_i . Choose better of the two (function value at target and trial point) using equation (1.8) for next generation and go to step 8.
- Step 8:* If $i < NP$ then go to step 3 otherwise go to step 9.
- Step 9:* Check whether the termination criterion is met. If yes then stop otherwise go to step 2.
-

1.6 Working of Differential Evolution

Throughout the present study *DE/rand/1/bin* version of DE will be used (unless otherwise mentioned), which is perhaps the most frequently used version and shall be referred as basic version.

With the objective of demonstrating the DE optimization process in continuous spaces, a simple example is analyzed.

$$\min f(X) = x_1^2 + x_2^2 \quad \text{where} \quad -2 \leq x_1, x_2 \leq 2$$

1. Select the control parameter of DE: Problem dimension $n = 2$, population size $NP = 10$, scaling factor $F = 0.5$ and crossover rate $Cr = 0.9$.
2. Initialize the population using equation (1.1).

Parameter	Individual							
	X_1	X_2	X_3	X_4	X_8	X_9	X_{10}
$x_{1,i}$	-1.22	0.92	-1.23	1.23	-1.23	0.15	-1.54
$x_{2,i}$	1.89	-0.14	0.34	0.33	-0.32	-1.34	1.23
Fitness	5.0605	0.866	1.6285	1.6218	1.6153	1.8181	3.8845

3. Select the target vector from current population (say X_I).
4. Select randomly three distinct vectors from current population (say X_4, X_9, X_2).
5. Apply the mutation operation to generate the mutant vector according to (1.2).

$$V_1 = X_4 + F * (X_9 - X_2) = \begin{bmatrix} 0.84 \\ -0.27 \end{bmatrix}$$

6. Create the trial vector by means of the crossover operation according to equation (1.7).

	Target individual	Mutant individual	Random number	Trial individual
$u_{1,i}$	-1.22	0.845	0.78	0.845
$u_{2,i}$	1.89	-0.27	0.46	-0.27
Fitness	5.0605	0.786925	--	0.786925

7. Select the individual that will advance to the next generation according to equation (1.8).

Parameter	Individual							
	X_1	X_2	X_3	X_4	X_8	X_9	X_{10}
$x_{1,i}$	0.845						
$x_{2,i}$	-0.27						
Fitness	0.786925						

8. Return to step 3 and repeat these task for all individual within the current population.
9. Update the current population by trial population.
10. This procedure is executed for several generations until a convergence criterion is satisfied.

1.7 Survey of Literature

Differential Evolution has emerged as one of the most simple and efficient technique for solving global optimization. Practical experience, however, shows that DE is not completely flawless. Like all other population based stochastic search techniques in their basic form, DE also has certain drawbacks associated with it. DE has some unique characteristics that make it different from many others in the family of EAs. The generation of offspring and selection mechanism of DE is completely different from its counterparts. DE uses a one-to-one spawning and selection relationship between each individual and its offspring. These features though help in strengthening the working of DE algorithm; they can sometimes turn into its weakness.

As pointed out by Lampinen and Zelinka (2000), DE may occasionally stop proceeding towards the global optimum even though the population has not converged to a local optimum or any other point. This situation is usually referred to as *stagnation*. Several instances are available in literature which aims at improving the performance of DE.

DE has three main control parameters namely population size, crossover rate Cr and scaling factor F . A number of investigations have been carried out to determine the optimum settings of these parameters. Storn and Price (1995) indicated that a reasonable population size could be between $5D$ and $10D$, where D denotes the dimensionality of the problem. They also recommended that a good initial choice of F can be 0.5.

Later, Gamperle et al. (2002) suggested that the population size should be between $3D$ and $8D$. They also suggested that the scaling factor F should be 0.6 and the crossover rate Cr should be in the range of $[0.3, 0.9]$ for best results. Ronkkonen et al. (2005), on the other hand, recommended the use of F values between $[0.4, 0.95]$ with $F = 0.9$ being a good initial choice. They further pointed out that the Cr values should lie within the range $[0, 0.2]$ when the function is separable while it should lie in $[0.9, 1]$ when the function's parameters are dependent. However, a drawback in their analysis is that in case of real life problems, it is very difficult to

examine beforehand the true nature of the function. Thus, it can be seen that there are different views and recommendations for the choice of parameters.

To avoid the manual tuning of parameters, researchers recommended the use adaptive/self adaptive setting of parameters, where instead of having a fixed value the control parameters are changed dynamically based on some feedback of the search space. Some of the works done in the development of adaptive/self adaptive control parameters are as follows: Abbass (2002) self-adapted the crossover rate Cr for multi-objective optimization problems, by encoding the value of Cr into each individual and simultaneously evolving it with other search variables. In his algorithm, the scaling factor F was generated for each variable by using a Gaussian distribution $N(0, 1)$. Zaharie (2003) proposed a parameter adaptation strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multi-population approach. Later, Zaharie and Petcu (2004) designed an adaptive Pareto DE algorithm for multi-objective optimization and also analyzed its parallel implementation.

Liu and Lampinen (2005) introduced fuzzy logic in DE and developed an algorithm called Fuzzy Adaptive Differential Evolution (FADE). Omran et al. (2005) proposed an algorithm called SDE in which they introduced a self-adaptive scaling factor parameter F and generated the value of Cr for each individual from a normal distribution $N(0.5, 0.15)$. Brest et al. (2007) proposed jDE algorithm using adaptive F and Cr . Epitropakis et al. (2009) suggested evolutionary adaption of the control parameters of differential evolution. Yang et al. (2009) developed an adaptive co-evolutionary DE named JACC-G and applied it for solving large scale global optimization problems. Although most of the self adaptive versions of DE, involve adaption of Cr and F , work has also been done on the adaption of the population size (DESAP by Teng et al., 2009) and on the adaption of trial vector generation strategies (SaDE by Qin et al., 2009).

Das et al. (2005) introduced two schemes for adapting the scale factor F in DE. In the first scheme they varied F randomly between 0.5 and 1.0 in successive iterations. They suggested decreasing F linearly from 1.0 to 0.5 in their second scheme. This encourages the individuals to sample diverse zones of the search space during the early stages of the search. During the later stages, a decaying scale factor helps to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected

global optimum lies. Teo (2006) proposed an attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates. Brest et al. (2006a) encoded control parameters F and Cr into the individual and evolved their values by using two new probabilities τ_1 and τ_2 . In their algorithm, a set of F values were assigned to each individual in the population. With probability τ_1 , F is reinitialized to a new random value in the range of $[0.1, 1.0]$, otherwise it is kept unchanged. The control parameter Cr , assigned to each individual, is adapted in an identical fashion, but with a different reinitialization range of $[0, 1]$ and with the probability τ_2 . With probability τ_2 , Cr takes a random value in $[0, 1]$, otherwise it retains its earlier value in the next generation. Differential Evolution with Preferential Crossover (DEPC) was suggested by Ali (2007). In his work he suggested three changes in the basic DE structure. The DEPC algorithm uses F_i as a random variable in $[-1, -0.4] \cup [0.4, 1]$ for each targeted point. Secondly, DEPC used two population sets S_1 and S_2 containing N points. The function of the auxiliary set S_2 in DEPC is to keep record of the trial points that are discarded in DE. Potential trial points in S_2 are then used for further explorations. Finally, DEPC used a new crossover rule, namely the preferential crossover, which always generates feasible trial points. Ali tested his algorithm on comprehensive set of benchmark problems and showed that DEPC outperforms the basic DE in most of the test cases.

Yang et al. (2008) proposed a self adaptive differential evolution algorithm with neighborhood search (SaNSDE). SaNSDE proposes three self-adaptive strategies: self adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor F , and self-adaptation of the crossover rate Cr . Qin et al. (2009) proposed a Self-adaptive DE algorithm (SaDE), where the choice of learning strategy and the two control parameters F and Cr are not required to be pre-defined. During evolution, the suitable learning strategy and parameter settings are gradually self-adapted according to the learning experience.

Hybridization is another concept which has been applied to DE to enhance its performance. Some hybridized versions of DE available in literature are as follows: Hendtlass, (2001), Zhang and Xie (2003), Talbi and Batchoue (2004) and Kannan et al. (2004) hybridized DE with PSO. Later, Omran et al. (2009) proposed a hybrid version of Bare Bones PSO and DE called BBDE. In their approach, they combined the concept of barebones PSO with self adaptive DE strategies. Zhang et al. (2009) proposed a DE-PSO algorithm in which a random

moving strategy is proposed to enhance the algorithm's exploration abilities and modified DE operators are used to enhance each particle's local tuning ability. Wu and Gu (2009) proposed Particle Swarm Optimization with prior crossover differential evolution (PSOPDE).

Yang et al. (2008a) proposed a Neighborhood Search Differential Evolution or NSDE by including a concept of local neighborhood search in the structure of DE. Later, Yang et al. (2008b) used Self-adaptive NSDE in the cooperative co-evolution framework for optimizing large scale non-separable problems (up to 1000 dimensions).

Caponio et al. (2009) proposed a hybridization of DE with three metaheuristics out of which one was PSO and two were local search methods. Omran and Engelbrecht (2009) proposed Free Search DE (FSDE) in which DE is hybridized with a newly developed 'Free Search Algorithm' and OBL. Menchaca-Mendez and Coello Coello (2009) hybridized Nelder Mead algorithm with DE for solving constrained optimization problems. A hybrid DE based on the one-step k-means clustering, called clustering-based DE (CDE), is presented for the unconstrained global optimization problems by Cai et al. (2011).

Other modifications which aim at improving the performance of DE include development of new mutation schemes; for example Fan and Lampinen (2003) suggested the stochastic application of a newly developed Trigonometric Mutation Operator (TMO) and called their algorithm as Trigonometric DE (TDE). Kaelo and Ali (2006) designed a new mutation scheme for DE based on random localization and named the corresponding algorithm as DERL. Pant et al. (2009a) and Pant et al. (2009b) developed a 'parent centric mutation' operator PCX and Laplace mutation operator for DE and named corresponding versions as DE-PCX and LDE respectively. Pant et al. (2009c) also recommended a mixed strategy DE (MSDE) in which instead of having a single mutation strategy throughout the generations, two mutation strategies were used in a competitive environment. More recently a new mutation operator based on wavelet theory was suggested by Lai et al. (2009).

Besides the variation in mutation scheme, some other interesting modifications in DE include: use of single population structure in DE (Thompson, 2004, Babu and Angira, 2006), crossover based local search method for DE proposed by Noman and Iba (2005, 2008), use of opposition based learning (OBL) for generating the initial population by Rahnamayan et al. (2008). The corresponding algorithm called ODE was also applied for solving large scale

optimization problems by Rahnamayan and Wang (2008); Brest et al. (2009) performed dynamic optimization using DE, Ting and Huang (2009) varied the number of difference vectors in DE. Tasgetiren et al. (2009) included of variable parameter search in DE. A 2- Opt based differential evolution was proposed by Chiang et al. (2010). Some variants of DE can also be found in Chakraborty (2008), Montgomery (2009), Wang et al. (2009) and Peng et al. (2009). A detailed review of literature and recent advances on DE is given in Das and Suganthan (2010) and Neri and Tirronen (2010).

1.8 Applications

Differential Evolution algorithm and its variants have been applied successfully to a wide variety of problems occurring in different fields of science and engineering. It is very difficult to summarize all the applications, therefore considering the brevity of space, in this section a brief review of some of the applications is given.

DE algorithm has been successfully applied to diverse domains of science and engineering, such as mechanical engineering design (Rogalski et al, 1999; Joshi and Sanderson, 1999), signal processing (Das and Konar, 2006), chemical engineering (Wang and Jang, 2000; Lampinen, 1999; Onwubolu and Babu, 2004), machine intelligence, and pattern recognition (Omran et al, 2005), (Das et. al, 2008). Das and Konar (2009) proposed an evolutionary-fuzzy clustering algorithm to automatically group the pixels of an image into different homogeneous regions. Noktehdan et al. (2010) proposed a grouping version of differential evolution (GDE) algorithm and its hybridized version with a local search algorithm (HGDE) to solve benchmarked instances of cell formation problem posing as a grouping problem. A DE based neural network approach to nonlinear system identification was proposed by Subudhi and Jena (2011). Many of the developments in DE algorithm design and applications can be found in Chakraborty (2008).

1.9 Objective of the Present Work

The main objective of the present work is to suggest enhanced DE variants by implying simple and efficient modifications in the basic structure of DE algorithm to improve its performance and to minimize its drawbacks, so that they can effectively solve the test as well as real life application problems.

The major objectives of this dissertation are:

1. To design efficient and reliable variants of DE algorithm for obtaining the global optimal solution of unconstrained non-linear optimization problems.
2. To extend the algorithms for solving constrained and multi-objective problems and test the algorithms on benchmark problems given in literature.
3. To use the algorithms for solving real life optimization problems arising in various fields of science and engineering.

1.10 Thesis Overview and Organization

The thesis is structured as follows:

Chapter 1 details the basic DE method and type of optimum. Besides stating the relevant definitions it gives literature related to the topic.

Chapter 2 describes the three modified versions of the basic Differential Evolution (DE) algorithm namely CDE, MSDE and SDE. The overall features of the aforementioned schemes are explained in details. A rigorous analysis of the algorithms is done numerically and statistically on a comprehensive set of 25 traditional benchmark problems and 7 nontraditional benchmark functions. Several performance metrics are considered to analyze the performance of the proposed algorithms in comparison to basic DE algorithm and to other advanced versions of DE.

Chapter 3 extends MSDE and SDE to solve constrained optimization problems. Pareto ranking constraint handling mechanism is used to handle the constraints. The performance of the algorithms is validated on constrained benchmark set proposed in CEC 2005. The proposed algorithms are also compared with other algorithms.

Chapter 4 extends SDE for solving multi-objective optimization problems. The performance of SDE is validated on nine problems and the numerical results are compared with other algorithms as well.

Chapter 5 further validates the performance of SDE for solving the real life trim loss problems arising in paper industry.

Chapter 6 includes a new optimization-based image thresholding algorithm using SDE. It also compares SDE with other algorithms on thresholding of test images.

Although each chapter in this thesis has its own concluding remarks, **Chapter 7** is dedicated to summarize the entire work and draw some final conclusions. Future research directions have also been discussed in this chapter.

There are 5 **Appendices** in the thesis. Contents of these Appendices are as below:

- Appendix I - List of unconstrained test problems.
- Appendix II - List of shifted test problems.
- Appendix III - List of constrained test problems.
- Appendix IV - List of multi-objective test problems.
- Appendix V - Non parametric tests

Enhanced DE Variants for Unconstrained Optimization

In this chapter three enhanced versions of DE are presented for solving unconstrained optimization problems. The first one is called Differential Evolution with Cauchy mutation (CDE), which employs Cauchy mutation to prevent premature convergence of DE. Second variant is named Differential Evolution with Mixed Mutation Strategy (MSDE) which uses the concept of evolutionary game theory to integrate two different mutation strategies in the basic structure of DE in a competitive gaming environment. The third version, named Synergetic Differential Evolution (SDE), is a fusion of three algorithmic components suggested in the recent modifications of DE.

Similar to other evolutionary algorithms having stochastic nature, a strong convergence proof for DE does not exist (Feoktistov, 2006). It means that even two different versions of DE cannot be compared mathematically (unlike deterministic algorithms which can be compared through algorithm complexity analysis). Therefore, a comprehensive set of benchmark problems is employed to analyze the performance of the proposed DE. The analysis is done on the basis of several performance metrics. Convergence speed, success rate and solution quality are three core measures in the following study.

The chapter is structured as follows. Section 2.1 is the introduction which reviews the available literature regarding the concepts used in the DE versions in the chapter. In section 2.2, Differential Evolution with Cauchy mutation (CDE) is explained. Differential Evolution with Mixed Mutation Strategy (MSDE) is explained in section 2.3. In section 2.4, Synergetic Differential Evolution (SDE) is introduced. Section 2.5 describes the benchmark test suit. Performance evaluation criteria are given in section 2.6. Parameters setting are given in section 2.7. Results are analyzed and discussed in section 2.8- 2.12. Finally the summary of the chapter is given in section 2.13.

2.1 Introduction

The DE versions suggested in this chapter are aimed at minimizing the shortcomings of DE by incorporating simple and efficient changes without disturbing the basic structure of DE.

The first algorithm, CDE, suggested in this work incorporates an additional component of mutation in it. As already discussed in the previous chapter, mutation in DE is significantly different from the mutation operators used in other evolutionary algorithms (like that of Genetic Algorithms), where these are based on some probability distribution. A review of literature shows that though there are several types of mutation operators (Fan and Lampinen, 2003; Kaelo and Ali, 2006; Pant et al., 2009a, 2009b, 2009c; Lai et al., 2009), Gaussian and Cauchy are perhaps the most commonly used distribution for mutating the particles in an evolutionary algorithm. Studies have shown that while applying Gaussian distribution, large mutation steps are very unlikely. This may reduce the convergence speed of a search algorithm and may also increase its risk getting caught in a local minimum. Cauchy distribution, on the other hand, is a more slowly decaying distribution in comparison to Gaussian and allows even the large mutation step sizes. Cauchy distribution has reportedly outperformed Gaussian distribution as per the instances available in literature (Lan and Lan, 2008; Rudolph, 1997; Yao et al., 1999; Birru et al., 1999; Wang et al., 2007; Coelho and Krohling, 2005).

In the previous chapter, a very brief description is given about the various mutation strategies proposed for the basic DE. From there it can be seen that each strategy has a unique feature of its own which guides the working of DE. However, a strategy that works very well for a particular type of function may not be well suited for another function. For example, the greedy mutation strategy (equation 1.4 and 1.5) which works well for a Unimodal function like sphere, may not give good results for a multimodal function like Restringin. A natural remediation for this is to use a combination of strategies instead of a single strategy so that the resulting variant is well suited for all types of problems. Keeping this fact in mind, in the second variant called MSDE, based on the concept of evolutionary game theory (Weibull, 1995; Dong et al., 2007), employs two different strategies in a competitive environment.

The third algorithm, Synergetic DE, presented in this chapter is a combination of three different algorithmic components suggested in the recent past by different researchers for improving the performance of basic DE algorithm. SDE is based on the concept of synergy

according to which “*the combined effort is always better than the individual effort*”. While in CDE and MSDE mutation strategies are targeted, in SDE, besides working on the mutation strategy emphasis is also laid on initial population and two population structure of DE.

In the following sub sections a detailed description of all the three algorithms is given one-by-one.

2.2 Differential Evolution With Cauchy Mutation (CDE)

Differential Evolution algorithm described in section 1.5 typically converges at a rapid pace in the initial stages of the search procedure and then gradually slows down as it approaches global optimum. The performance of DE can easily be viewed by observing the fitness function value. If there is an improvement in fitness in successive generations, it then signifies that the new trial point generated by DE is better than target vector. Consequently, the new trial point replaces the target vector in next generation. However, in case there is no improvement in fitness of an individual, then it is an indication that the individuals are clustered together in a region and their vector difference (second term of equation (1.2)) is either zero or very insignificant to allow any improvement of function value. In such a case, it is necessary to introduce a perturbation in the population which will help the individuals to move to a new location.

CDE introduces a mechanism which not only keeps a track of the progress of individuals but also helps the individuals in escaping the local basin by allowing them to jump to a new region. In order to keep a record of the success of individuals, it uses a ‘*failure_counter*’ (*FC*). The work of *FC* is to monitor the working of individuals in terms of fitness function value for a specified number of generations. If there is no improvement in fitness, then *FC* is increased by unity in each generation. This process is repeated until *FC* achieves user-defined value of *maximum failure counter* (*MFC*). Once *MFC* is attained, it is an indication that perturbation in the population is needed which will allow the individual to *jump* (or move) to a new position.

In order to achieve this, Cauchy mutation (Stacey et al., 2003) is applied for which the probability density function (PDF) is given by the following equation:

$$f(x; x_0, y_0) = \frac{1}{\pi y_0 \left[1 + \left(\frac{x - x_0}{y_0} \right)^2 \right]} = \frac{1}{\pi} \left[\frac{y_0}{(x - x_0)^2 + y_0^2} \right] \quad (2.1)$$

where

x_0 is the location parameter, specifying the location of the peak of the distribution.

y_0 is the scale parameter which specifies the half-width at half-maximum.

Effect of using Cauchy mutation: The graph of PDF for different values of x_0 and y_0 is illustrated in Figure 2.1. From this figure, it is clear that for large values of y_0 a fat tail curve is obtained, where as for smaller values of y_0 , the shape of the curve changes towards a sharper peak. In the present study y_0 is taken as 0.1, which produces a very sharp peak, resulting in a small area around the mean.

A new trial vector $U_{i,G} = (u_{1,i,G}, \dots, u_{D,i,G})$ by CDE is generated as follows:

$$u_{j,i,G} = \begin{cases} x_{j,best,G} + C(y_0, 0) & \text{if } rand(0,1) \leq 0.9 \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (2.2)$$

where $C(y_0, 0)$ stands for random number generated by Cauchy probability distribution with scale parameter y_0 and centered at origin. After generation of a new point, selection process, similar to that of basic DE is used.

This modification enables the algorithm to get a better tradeoff between the convergence rate and robustness. Thus, it is possible to increase the convergence rate of the differential evolution algorithm and thereby obtain an acceptable solution with a lower number of objective function evaluations. Such an improvement can be advantageous in many real-world problems where the evaluation of a candidate solution is a computationally expensive operation and consequently finding the global optimum or a good suboptimal solution with the original differential evolution algorithm is too time-consuming or even impossible within the time available.

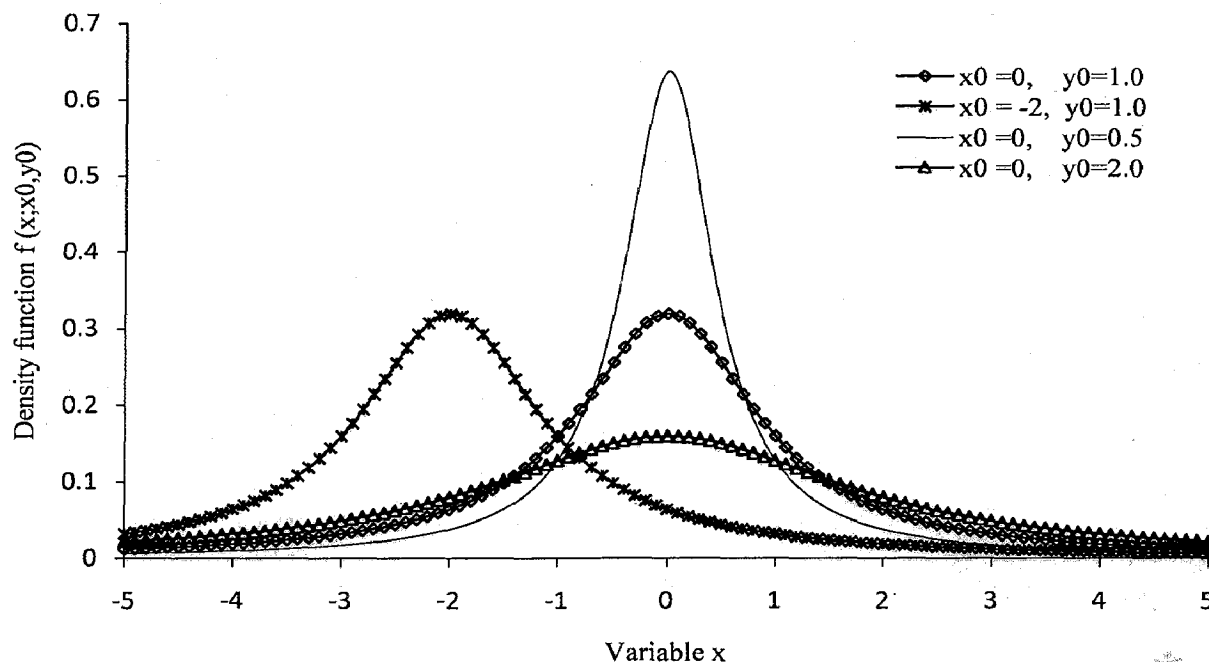


Figure 2.1: Probability density function of Cauchy distribution for different values of y_0 and x_0 .

Working of CDE: working of CDE is same as DE except for the two major differences: first, it is the use of a failure counter to avoid a possible stagnation of performance of an individual and second, the use of two mutation operators depending on the failure counter; basic DE mutation and Cauchy mutation. Sensitivity analysis of the additional parameter MFC of CDE is discussed later in this chapter.

The working procedure of the CDE is outlined below through flow chart and pseudocode:

Algorithm 2.1 Pseudocode of CDE illustrating how the procedure acts on a population of individuals, repeating mutation, crossover and selection until the convergence criteria is met.

Step 1: Generate randomly NP individuals $X_i, i=1,2,\dots, NP$, using equation (1.1). Set the values of control parameters F, Cr , and MFC . Set $failure_counter[i] = 0$.

Step 2: Set $i = 0$.

Step 3: $i = i + 1$.

Step 4: If $failure_counter[i] < MFC$ then go to step 5 otherwise go to step 7.

Step 5: Corresponding to target individual X_i select three distinct individuals X_{r1}, X_{r2} and X_{r3} such that $i \neq r1 \neq r2 \neq r3$ from population and generate perturbed individual V_i using equation (1.2).

Step 6: Recombine the target vector X_i with perturbed individual V_i generated in step 5 to generate trial

vector U_i using equation (1.7) and go to step 8.

Step 7: Generate a trial vector using equation (2.2) and go to step 8.

Step 8: If all parameters of the trial vector is within the given range then go to step 9 otherwise uniformly generate that parameter within given range using equation (1.1) and go to step 9.

Step 9: Calculate the objective function value for vector U_i . Choose better of the two (function value at target and trial point) using equation (1.8) for next generation, if trial vector is selected for next generation then $failure_counter[i] = 0$ otherwise increase the value of $failure_counter[i]$ by one.

Step 10: If $i < NP$ then go to step 3 otherwise go to step 11.

Step 11: Check whether the termination criterion is met. If yes then stop otherwise go to step 2.

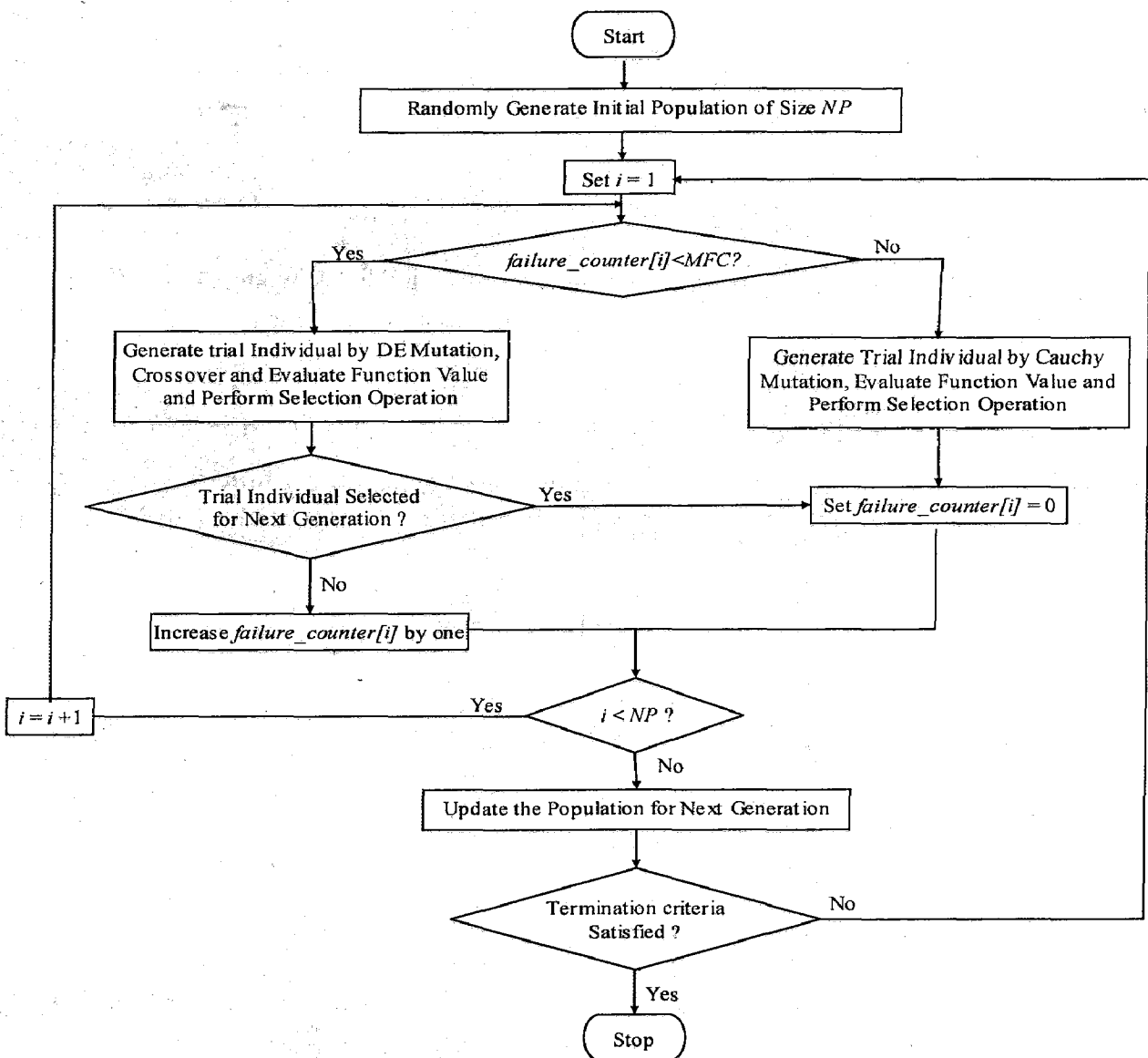


Figure 2.2: Flow chart of CDE.

2.3 Differential Evolution With Mixed Mutation Strategy (MSDE)

The second algorithm proposed in this chapter is Differential Evolution with Mixed mutation strategy or MSDE, based on the classical evolutionary game theory. In MSDE algorithm the individuals are regarded as players in an artificial evolutionary game applying different mutation operators (set of strategy) to generate trial vector. Each player tries to improve its performance by selecting a strategy from the given set and the value of the game changes accordingly. Based on this analogy individuals of the MSDE are referred as players and the mutation operation as the strategy. This is in contrast with the basic DE, where all the individuals are subject to a single mutation operator. In MSDE, every individual of the population may select any one of the two strategies provided to it in order to produce a perturbed (mutant) vector $V_{i,G}$. A single mutation operator is called a pure strategy in the terms of game theory. A strategy profile, vector \vec{S} , is a collection of pure strategies such that $\vec{S} = (s_1, \dots, s_\alpha)$, where s_i is the pure strategy used by individual i .

Strategies used in MSDE:

In the MSDE, only two mutation strategies s_1 and s_2 are considered where s_1 denotes the usual mutation operation as given by equation (1.2) and s_2 is defined as:

$$s_2 = \frac{1 (X_{r1,G}^2 - X_{r2,G}^2)f(X_{r3,G}) + (X_{r2,G}^2 - X_{r3,G}^2)f(X_{r1,G}) + (X_{r3,G}^2 - X_{r1,G}^2)f(X_{r2,G})}{2 (X_{r1,G} - X_{r2,G})f(X_{r3,G}) + (X_{r2,G} - X_{r3,G})f(X_{r1,G}) + (X_{r3,G} - X_{r1,G})f(X_{r2,G})} \quad (2.3)$$

The second strategy s_2 denotes quadratic interpolation, which determines the point of minima of the quadratic curve passing through three selected points. It is clear from the Figure 2.3 of quadratic interpolation that the fitted function passing through these three points is parabola and new point is produced at minimum of this parabola. There is no particular rationale for choosing quadratic interpolation as the second strategy except that it is a well known method that makes use of gradient in a numerical way. It is a direct search optimization method and has given good results in several cases (Mohan and Shanker, 1994; Deep and Das, 2008).

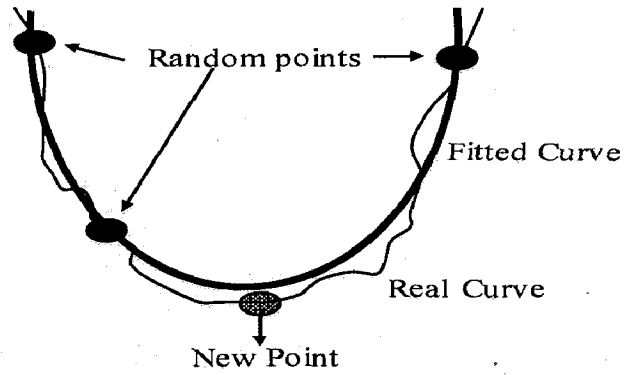


Figure 2.3: Parabolic curve of quadratic interpolation.

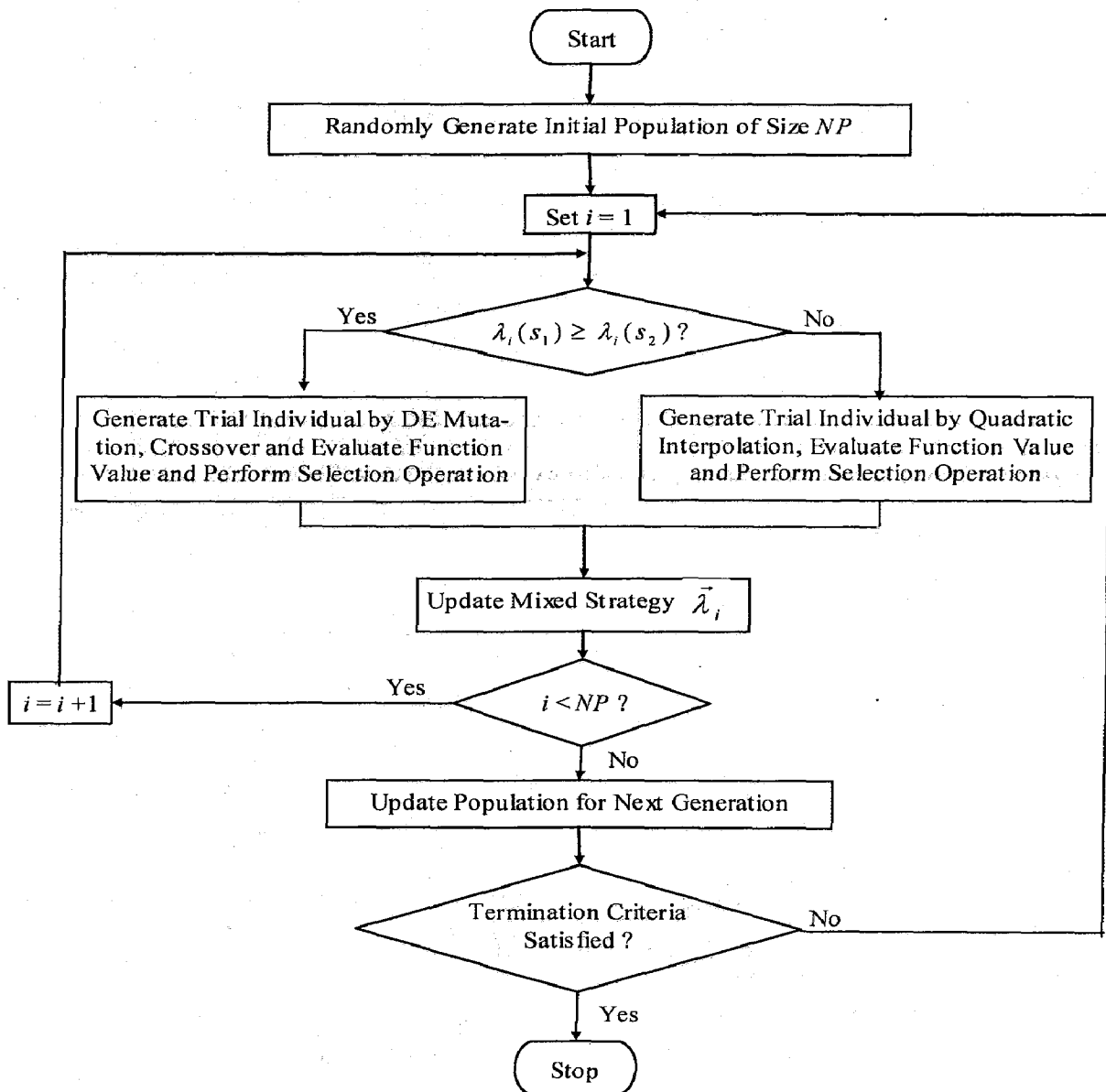


Figure 2.4: Flow chart of MSDE.

Working of MSDE: At each generation, every individual chooses a mutation operator from its strategy set based on a probability distribution. This distribution over the set of pure strategies available to an individual is called the mixed strategy of individual i and is represented by a vector $\vec{\lambda}_i = (\lambda_i(s_1), \dots, \lambda_i(s_\beta))$, where $\beta (=2 \text{ in this case})$ is the number of strategies, and $\lambda_i(a)$ is the probability of individual i applying pure strategy a in mutation. To each individual a payoff is assigned according to its performance using particular mutation strategy. An individual can adjust its mixed strategy based on the payoffs of strategies. Usually, the strategy with a better payoff will be preferred with a higher probability in the next generation. If the target vector X_i uses strategy s_α , where $\alpha = 1, 2$ and new point survive in next generation ($G+1$) then update probability as:

$$\begin{aligned} \text{otherwise } \lambda_i^{G+1}(s_\alpha) &= \lambda_i^G(s_\alpha) + (1 - \lambda_i^G(s_\alpha))\gamma, \quad \lambda_i^{G+1}(s_\beta) = \lambda_i^G(s_\beta) - \lambda_i^G(s_\beta)\gamma \quad \forall \beta \neq \alpha \\ \lambda_i^{G+1}(s_\alpha) &= \lambda_i^G(s_\alpha) - \lambda_i^G(s_\alpha)\gamma, \quad \lambda_i^{G+1}(s_\beta) = \lambda_i^G(s_\beta) + \lambda_i^G(s_\beta)\gamma \quad \forall \beta \neq \alpha \end{aligned} \quad (2.4)$$

For the proposed MSDE, no additional parameter setting is required. For the probability update, the parameter γ has been chosen as suggested in Dong et al. (2007). Sensitivity analysis of γ is done later in the chapter.

The procedure of MSDE through flowchart (Figure 2.4) and pseudocode is outlined as follows:

Algorithm 2.2 Pseudocode of MSDE illustrating how the procedure acts on a population of individuals, repeating mutation, crossover and selection until the convergence criteria is met.

- Step 1:** Generate randomly NP individuals $X_i, i = 1, 2, \dots, NP$, using equation (1.1). Set the values of control parameters F, Cr , and γ . Initially assign mixed strategy as $\vec{\lambda}_i = (\lambda_i(s_1), \lambda_i(s_2)) = (0.5, 0.5)$
- Step 2:** Set $i = 0$.
- Step 3:** $i = i + 1$.
- Step 4:** For target vector X_i (parent vector) choose strategy (mutation operator) according to probability distribution $\vec{\lambda}_i$. If probability of pure strategy s_1 is greater than the probability of strategy s_2 then go to step 5 otherwise go to step 6.
- Step 5:** Corresponding to target individual X_i select three distinct individuals X_{r1}, X_{r2} and X_{r3} such that $i \neq r1 \neq r2 \neq r3$ from population and generate perturbed individual V_i using equation (1.2) and go to step 7.
- Step 6:** Select one best point and other two distinct points from population and generate perturbed individual
-

V_i by quadratic interpolation given in equation (2.3) and go to step 7.

- Step 7:* Recombine each target vector X_i with perturbed individual generated in step 5 or 6 to generate a trial vector U_i using equation (1.7) and go to step 8.
- Step 8:* If all parameters of the trial vector are within the given range then go to step 9 otherwise uniformly generate that parameter within given range using equation (1.1) and go to step 9.
- Step 9:* Calculate the objective function value for vector U_i .
- Step 10:* Choose better of the two (function value at target and trial point) using equation (1.8) for next generation.
- Step 11:* Update the probability according to equation (2.4).
- Step 12:* If $i < NP$ then go to step 3 otherwise go to step 13.
- Step 13:* Check whether the termination criterion is met. If yes then stop otherwise go to step 2.
-

2.4 Synergetic Differential Evolution (SDE)

This section introduces the Synergetic Differential Evolution (SDE), a fusion of three different algorithmic components; Opposition based-learning (OBL) for generating the initial population, random localization for selecting the base vector and a one population DE framework. To make the proposed algorithm self explanatory, these schemes are briefly described in the following subsections.

2.4.1 Opposition Based Learning (OBL) For Generating the Initial Population

The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e., guess and opposite guess) in order to achieve a better approximation for the current candidate solution. Opposition based initial population is based on the concept of opposite numbers. Before explaining the opposition based initial population, the definitions of opposition based random numbers and opposition based optimization are given which form the basis of the opposition based initial population.

Opposition based random numbers: if $x \in [l, u]$ is a real number, then its opposite number x' is defined as

$$x' = l + u - x \quad (2.5)$$

here l and u indicates the lower and upper bounds of the variables. This definition can be extended for higher dimensions also as suggested in Rahnamayan et al. (2008). If $X =$

(x_1, x_2, \dots, x_D) is a point in D -dimensional space, where $x_1, x_2, \dots, x_D \in R$ and $x_i \in [l_i, u_i] \forall i \in \{1, 2, \dots, D\}$, then the opposite point $X' = (x'_1, x'_2, \dots, x'_D)$ is completely defined by its components.

$$x'_i = l_i + u_i - x_i \quad (2.6)$$

Opposition based optimization: by employing the opposite point definition, the opposition-based optimization can be given in three simple steps:

- ✓ Let $X = (x_1, x_2, \dots, x_D)$ be a point (i.e., a candidate solution) in D -dimensional space with $x_i \in [l_i, u_i]$ and assume $f(x)$ is a fitness function which is used to measure candidate optimality. According to the opposite point definition, $X' = (x'_1, x'_2, \dots, x'_D)$ is the opposite point of $X = (x_1, x_2, \dots, x_D)$.
- ✓ Evaluate the fitness of both points $f(X)$ and $f(X')$.
- ✓ If $f(X') \leq f(X)$ (for minimization problem), then replace X with X' ; otherwise, continue with X .

The point and its opposite point are evaluated simultaneously in order to continue with the fitter one.

Opposition based initial population: A random population P of size NP using equation (1.1) and its corresponding opposite population, also of size NP using following equation

$$ox_{j,i,0} = l_j + u_j - x_{j,i,0} \quad (2.7)$$

are generated and merged together to form a population of size $2NP$. Where $x_{j,i,0}$ and $ox_{j,i,0}$ denote the j^{th} variable of the i^{th} individual of the population and its opposite population vector, respectively. Out of this population, NP best candidates are selected as the initial population of the SDE algorithm.

Effect of using opposition based initial population: Purely random re-sampling or selection of solutions from a given population has the chance of visiting or even revisiting unproductive regions of the search space. As demonstrated by Rahnamayan et al. (2008), the chance of this occurring is lower for opposite numbers than it is for purely random ones. In fact, a mathematical proof has been proposed to show that, in general, opposite numbers are more likely to be closer to the optimal solution than purely random ones (Tizhoosh, 2005).

2.4.2 Randomized Localization for Selecting the Base Vector

According to this rule, three distinct points X_{r1} , X_{r2} and X_{r3} are selected randomly from the population corresponding to target point X_i . A tournament is then held among the three points and the region around the best point is explored. That is to say if X_{r1} is the point having the best fitness function value then the region around it is searched with the hope of getting a better solution. For the sake of convenience the tournament best point will be denoted as (say) X_{tb} . Assuming that $X_{tb} = X_{r1}$, the mutation equation (1.2) becomes:

$$V_{i,G+1} = X_{tb,G} + F \times (X_{r2,G} - X_{r3,G}) \quad (2.8)$$

This variation gradually transforms itself into search intensification feature for rapid convergence when the points in S form a cluster around the global minima.

Effect of tournament best for base individual: In order to see the effect of tournament best method for mutation, the two common strategies of DE; DE/best/1/bin and DE/rand/1/bin will be discussed briefly. In DE/best/1/bin, the base vector is always selected as the one having the best fitness function value. It can be seen, that here the probability of selecting the best vector as the base vector is always 1. This strategy may provide a fast convergence in the initial stages. However, as the search procedure progresses it may lead to the loss of diversity in the population due to its greedy nature resulting in premature convergence.

On the other hand, the strategy DE/rand/1/bin is completely random in nature. Here all the points for mutation are randomly selected and the best point of the population may or may not be included in them. This strategy, due to its random nature helps in preserving the diversity but may result in a slower convergence.

Now, if we look at the tournament best method we see that although the three points for mutation are randomly selected, the base vector is always chosen as the one having the best fitness. This makes it neither purely greedy nor purely random in nature but provides a localized effect which helps in exploring the different regions of the search space around the potential candidates. Making use of hyper geometric distribution, we can say that the probability of getting the best vector among the three chosen points for mutation is $\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}$, where M is the number of best points in the population. Initially, it is very much likely that there is one best point but as the evaluation process proceeds the number of best points keeps on increasing.

In case of strategy, DE/rand/1/bin the probability that the best point of the population is among the three chosen points for mutation is $\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}$ and the probability that the best point is also selected as the base vector is $\frac{1}{3} * [\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}]$. When the tournament best strategy is applied, the probability selecting the best point from the three chosen will be 1. Thus the probability that the selected base vector is the best solution of the population becomes $1 * [\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}]$. Thus it can be seen that the probability of selecting the best point of the population as base vector, for tournament best strategy lies between the probabilities of DE/rand/1/bin and DE/best/1/bin. $\frac{1}{3} * [\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}] < [\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}] < 1$. This helps in maintaining the exploration and exploitation capabilities of the proposed SDE ensuring fast convergence and balanced diversity.

2.4.3 Single Population Structure

In a single population DE, only one population is maintained and the individuals are updated as and when a better solution is found. Also, the newly found better solutions can take part in mutation and crossover operation in the current generation itself as opposed to basic DE (where another population is maintained and the better solutions take part in mutation and crossover operations in next generation). This concept was suggested by Thompson (2004) and Babu and Angira (2006).

Effect of using a single population structure: In the basic structure of DE, where two populations (current and advance) are considered simultaneously in all the iterations which results in the consumption of extra memory and CPU time leading to higher number of function evaluations. Updating the single population continuously enhances the convergence speed leading to lesser number of function evaluations as compared to basic DE.

Working of SDE: The structure of SDE is same as of DE but differ from it only in the ways of initialization, mutation and population update.

The working of SDE is given below with the help of flowchart and algorithm:

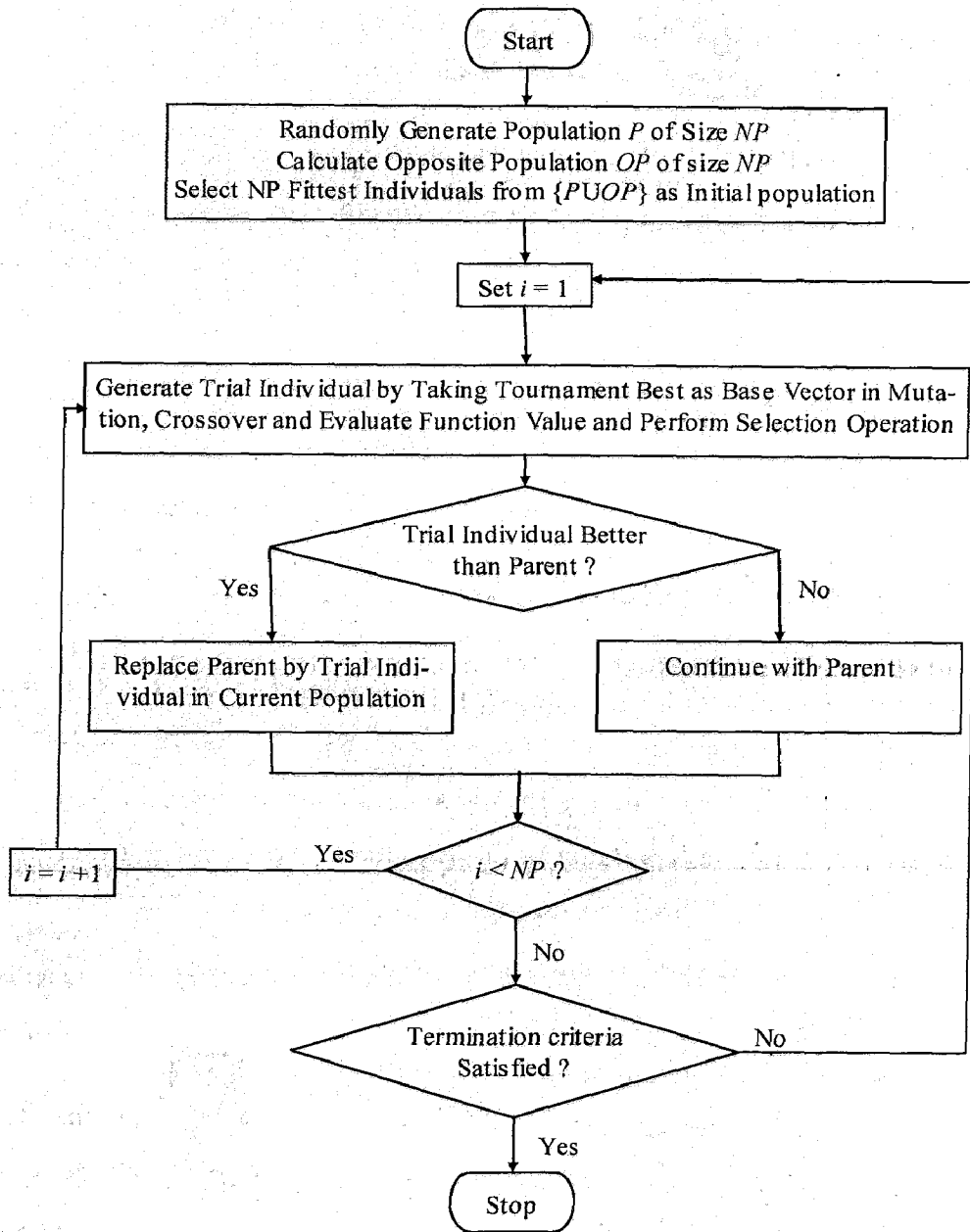


Figure 2.5: Flow chart of SDE.

Algorithm 2.3 Pseudocode of SDE illustrating how the procedure acts on a population of individuals, repeating mutation, crossover and selection until the convergence criteria is met.

- Step 1:* Generate randomly NP individuals X_i , $i = 1, 2, \dots, NP$, using equation (1.1) and (2.7). Set the values of control parameters F and Cr .
- Step 2:* Set $i = 0$.
- Step 3:* $i = i + 1$.
- Step 4:* Corresponding to target individual X_i , select three distinct individuals X_{r1} , X_{r2} and X_{r3} such that $i \neq r1 \neq r2 \neq r3$ from population and generate perturbed individual V_i using equation (2.8).
- Step 5:* Recombine the target vector X_i with perturbed individual V_i generated in step 4 to generate trial vector U_i using equation (1.7).
- Step 6:* If all parameters of the trial vector are within the given range then go to step 7 otherwise uniformly generate that parameter within given range using equation (1.1) and go to step 7.
- Step 7:* Calculate the objective function value for vector U_i . Choose better of the two (function value at target and trial point) using equation (1.8) for current generation if trial vector is better than target individual then update population otherwise continue with old one.
- Step 8:* If $i < NP$ then go to step 3 otherwise go to step 9.
- Step 9:* Check whether the termination criterion is met. If yes then stop otherwise go to step 2.
-

2.5 Benchmark of Unconstrained Problems

Two sets of benchmark problems are considered for analyzing the performance of the proposed algorithms.

The first set consists of a test suite of twenty five standard/traditional benchmark problems taken from Rahnamayan et al. (2008) and Zhang and Sanderson (2009). This test set includes fixed, lower dimension problems as well as scalable problems for which the dimension can be increased to increase the complexity of the problem. Mathematical models of the traditional functions along with the true optimum value are given in APPENDIX I.

The second set consists of seven nontraditional problems selected from the set of recently proposed benchmark test suite for *CEC 2008 special session and competition on large scale global optimization* (Tang et al., 2007). This test suite was specially designed to test the efficiency and robustness of a global optimization algorithm like DE. All the seven problems are tested for dimension 500. Name of these functions and their properties are listed in APPENDIX II.

2.6 Performance Metrics

In order to validate the practicality of the proposed algorithms (CDE, MSDE and SDE) the following two performance metrics were considered.

Performance metric I

➤ **Number of Function Evaluations (NFE):** It is one of the most common performance metric used for evaluating the performance of an algorithm. It is obtained by recording the number of times a function is evaluated for acquiring an error value less than ϵ before the maximum number of function evaluations is reached.

➤ **Average NFE:** Considering the stochastic nature of the algorithm, average NFE obtained

during different runs is calculated as:
$$\frac{\sum_{i=1}^{NR} NFE}{NR}$$

Where NR represents the total numbers of runs.

➤ **Average error:** It is another common criterion used for measuring the reliability of the algorithm. The minimum function error value that an algorithm can find, using predefined maximum NFEs, is recorded in each run and then average of the error values are calculated. The function error value, used for performance measures for an obtained solution x is defined as $|f(x) - f(x^*)| = \epsilon$, where x^* is the global optimum of the function.

➤ **Percentage Acceleration rate (AR):** it is the ratio of the NFE of the algorithm to be compared and the NFE of the algorithm to which it is being compared (Rahnamayan et al., 2008), e.g. the % AR of CDE in comparison to DE will be $\% AR = \left(1 - \frac{(NFE)_{CDE}}{(NFE)_{DE}}\right) * 100$

➤ **Success rate (SR):** The number of times, for which the algorithm succeeds in reaching the desired accuracy (ϵ) for each test function, is measured as the success rate.

$$SR = \frac{\text{Number of times reached accuracy } (\epsilon)}{\text{Total number of trials}}$$

SR is a commonly used metric to quantify the robustness of the algorithms.

➤ Average AR = $\frac{1}{N} \sum_{i=1}^N AR_i$

➤ Average SR = $\frac{1}{N} \sum_{i=1}^N SR_i$

where N denotes the number of problems.

Performance metric II: statistical analysis

The proposed algorithms are also analyzed statistically using non parametric tests (Garcia et al., 2009). A detailed description of non parametric tests used in this study is given in Appendix V.

Graphical illustrations

The performance of algorithms is also illustrated with the help of convergence graphs and statistical plots.

2.7 Parameter Settings

After conducting several experiments and referring to various literatures, following settings have been taken for all the experiments unless otherwise mentioned.

- Population Size (NP) = 100 for traditional benchmark problems and 500 for nontraditional problems (Zhang and Sanderson, 2009; Rahnamayan and Wang, 2008).
- Scaling/ amplitude Factor $F = 0.5$ (Rahnamayan et al., 2008).
- Crossover Rate $Cr = 0.5$ for DE and CDE (Rahnamayan et al., 2008); 0.33 for MSDE and 0.9 for SDE.
- Maximum NFE = $10000 * D$, for traditional problems while it is $5000 * D$ for non-traditional problems, where D is the dimension of the problem (Noman and Iba, 2008; Rahnamayan and Wang, 2008).
- Accuracy (ϵ) = 10^{-8} for all the test problems except noisy function (f_7) for which it is set as 10^{-2} (Zhang and Sanderson, 2009).
- Number of trials = 50 (Rahnamayan et al., 2008).

PC configuration:—

- ✓ Processor: Intel dual core
- ✓ RAM: 1 GB
- ✓ Operating System: Windows vista

Software used:-

- ✓ DEV C++

✓ SPSS 16 (Software Package for Social Science)

Random numbers are generated using inbuilt rand () function with same seed for every algorithm

Handling of boundary violations

Boundary violations, which are often encountered while performing a mutation operation in DE variants, are dealt by replacing each and every member violating variable bounds with a new member generated randomly between the lower and upper bounds of variables. This approach is referred to as random generation.

Algorithms taken for comparison:

- Basic *DE* (Storn and Price, 1997)
- *ODE* (Rahnamayan et al., 2008)
- *DERL* (Kaelo and Ali, 2006)
- *MDE* (Babu and Angira, 2006)
- *SaDE* (Qin et al., 2009),
- *jDE* (Brest et al., 2007; Brest et al., 2006)
- *JADE* (Zhang and Sanderson, 2009).

2.8 Sensitivity Analysis of Additional Parameters Used in Proposed Algorithms

Out of the three DE variants proposed in this chapter, CDE and MSDE use *MFC* and γ as additional parameters respectively. In this section sensitivity analysis of these parameters is done on the basis of numerical results obtained for problems listed in Appendix I. It was observed that simply on the basis of error, no concrete judgment can be made therefore NFE is used as an evaluation criteria for deciding the values of *MFC* and γ .

2.8.1 Analysis of Parameter Maximum Failure Counter (MFC)

A series of experiments conducted to determine an appropriate value for *MFC* showed that very high and very low values for *MFC* are rarely effective. For example if *MFC* is taken as 1, then it is more or less like applying mutation after every iteration. On the other hand, keeping *MFC* very high naturally resulted in higher NFE. In the present study, the NFE obtained for three values of *MFC* viz. 3, 5, 10 is recorded to achieve accuracy mentioned in Section 2.7.

From the corresponding results summarized in Table 2.1, it can be seen that except for f_5 and f_9 for which the desired accuracy was not achieved, 5 is a relatively good choice of *MFC*.

2.8.2 Sensitivity Analysis of Parameter γ

The additional parameter, γ of MSDE acts like a weighting parameter defined by the user at the beginning of the program. Empirical analysis of γ was done for various values between 0 and 1 for which NFE obtained for $\gamma = 0.001, 0.1, 0.25, 0.33$ and 0.95 are recorded in Table 2.2.

An analysis of these results show that smaller values of γ results in slower convergence thereby increasing the number of function evaluations. Values between 0.25 to 0.95 are most suited for the optimization problems taken in the present study. Consequently, the value of γ was taken as 0.33 as suggested in Dong et al. (2007).

2.9 Comparison of the Proposed Algorithms With Basic DE, ODE, DERL and MDE

Initially, the proposed CDE, MSDE and SDE are compared with basic DE, ODE, DERL and MDE. ODE, DERL and MDE are specially taken because their algorithmic components form the basis of the proposed SDE algorithm or it can be said that these are the parent algorithms of SDE. The algorithms are analyzed according to the performance metrics mentioned in section 2.6 and the corresponding results are recorded in Tables 2.3 – 2.12.

2.9.1 Comparison in Terms of Error and Standard Deviation (Std.)

First of all the proposed algorithms are analyzed and compared in terms of error and Std., for which the results are recorded in Table 2.3. From this Table, it can be seen that out of 25 test cases SDE outperformed the other algorithms in 6 test cases while MSDE outperformed others in 7 cases in terms of both error and standard deviation. In the remaining cases all the algorithms gave more or less similar results.

2.9.2 Comparison in Terms of Number of Function Evaluations

On the basis of the NFE, for which the results are given in Table 2.4, it can be seen that MSDE gave a superior performance in comparison to other algorithms in 17 out of 25 cases. The second place went to SDE which gave a performance better than other algorithms in 5 cases and the last place was secured by CDE which outperformed the others only in 1 case.

DE, ODE, MDE, CDE and MSDE were not able to solve the fifth function f_5 , while none of the algorithms, except MSDE, were able to reach the desired accuracy of 10^{-8} for the function f_9 and were therefore terminated when the maximum NFE ($= 10 * D$) was reached.

The performance of the proposed algorithms is illustrated with the help of convergence graphs, drawn according to the fixed NFE (and not according to fixed accuracy), for few selected functions in Figure 2.6.

2.9.3 Comparison in Terms of Acceleration Rate (AR)

Acceleration Rate is another criterion which helps in analyzing the speed of an optimization algorithm. The % AR of all algorithms against DE is calculated and is recorded in Table 2.5.

From this Table it can be observed that for the first proposed version, CDE, the AR is more than 70% for one test problem only (for function f_{15}) and more than 60% for two test cases. The AR was calculated to be more than 40% for one test problem and more than 10% for 7 cases. For 3 cases, it was more than 20% and for 6 cases the AR was more than 1%. For f_7 and f_{24} , the convergence rate of DE was better than CDE and hence the negative AR in these two cases. On an average, the AR for CDE was calculated as 12.6%.

For the second variant, MSDE, the AR is more than 70% for 11 test problems; more than 40% in 6 cases and more than 10% in 5 cases. The average AR for MSDE is calculated to be 58%.

For the third proposed version, that is SDE, it was seen that for 11 test problems the % AR of SDE in comparison to DE is more than 50%. For 7 test problems the % AR is more than 40% and for 4 test problems it is more than 30%. On an average the AR of SDE against DE is 46%.

For the competing algorithms: ODE, DERL and MDE, the average AR was calculated as 1%, 38% and 9% respectively.

It is also to be mentioned that in case of f_9 , AR is not recorded because none of the algorithms were able to meet the desired accuracy criteria while in case of f_5 , AR is not recorded because DE was not able to solve it successfully.

2.9.4 Comparison in Terms of Success Rate (SR)

The successful performance of all the algorithms is summarized in Table 2.6. Here it can be seen that on an average, the SR of the proposed SDE algorithm is 95%, which is the best in comparison to all the algorithms. Second place went to MSDE and DERL, for which the average SR came out to be 92% for both the algorithms. For DE and ODE, the average SR was calculated to be 89% each while for MDE and CDE, the SR was calculated as 88% for both the algorithms.

DE, ODE, SDE, CDE and MSDE were not able to reach the desired accuracy for function f_5 and none of the algorithm except MSDE was able to meet the desired accuracy criteria for function f_9 .

2.9.5 Statistical Analysis

Error values included in Table 2.3 allow us to carry out a rigorous statistical study in order to check whether the results of the algorithms are significant for considering them different in terms of quality on approximation of continuous functions.

Table 2.7 shows the result of applying Friedman's tests in order to see whether there are global differences in the results. Given that the p -value of Friedman test is lower than the level of significance considered $\alpha = 0.05$, there are significant differences among the observed results.

Attending to these results, a *post-hoc* statistical analysis is done to detect concrete differences among algorithms. First of all, Bonferroni-Dunn's test is employed to detect significant differences for the control algorithm SDE. Table 2.8 summarizes the ranking obtained by Friedman's test and the Critical Difference (CD) of Bonferroni-Dunn's procedure.

In Figure 2.7, Bonferroni-Dunn's graphic illustrates difference among rankings obtained for each algorithm. Here, the horizontal cut line represents the threshold for the best performing algorithm, the one with the lowest ranking bar, in order to consider it better than other algorithms. A cut line is drawn for each level of significance considered in the study at height equal to the sum of the ranking of the control algorithm and the corresponding Critical Difference computed by the Bonferroni-Dunn method. The bars which exceed this line are associated to an algorithm with worse performance than the control algorithm. The application of Bonferroni-Dunn's test informs us of the following significant differences with SDE as control algorithm:

- *SDE* is better than *DE* and *CDE* with $\alpha = 0.05$ and $\alpha = 0.10$ (2/6 algorithms).

Until now, procedures for performing multiple comparisons were used to check the behaviour of the algorithms. Now pairwise comparison of SDE is done with the rest of the algorithms using Wilcoxon test. The corresponding results are given in Table 2.9. It displays the statistics, *p*-value and number of +ve ranks (where control algorithm performed better than comparing algorithm), -ve ranks (where control algorithm performed worse than comparing algorithm) and tie (both algorithms performed equivalently), mean and sum of these ranks.

From this Table, it can be seen that in case of *DE*, for 13 problems SDE performed better than it, while for 12 cases both the algorithms performed similarly. In case of SDE and *MDE*, for 11 test cases SDE performed better than it while for 1 case *MDE* performed better than SDE. In the remaining 13 cases both algorithms performed equivalently. SDE outperformed *ODE* in 12 cases while in the remaining 13 cases both algorithms performed equivalently. SDE performed better than *DERL* in 11 cases, in 13 cases there was a tie i.e. both algorithms performed equivalently while in 1 case *DERL* outperformed SDE.

In an interesting observation, it is seen that according to Wilcoxon test there is no significant difference between SDE and *MSDE* algorithms.

Following the procedure given above a similar analysis for NFE was done for which the results are given in Table 2.4. The statistical results based on it are summarized in Tables 2.10-2.12. From these Tables and from the graphical illustration given in Figure 2.8, it can be seen that in an overall comparison MSDE and SDE are at par with each other while the remaining algorithms perform worse than SDE.

2.10 Further Analysis of the Proposed Algorithms

In the following subsections, some further analysis is done for the proposed algorithms. Firstly, the algorithms are analyzed for two important factors; dimension of the problem and population size. The second analysis is done for the proposed SDE algorithm. In Rahnamayan et al. (2008), the authors proposed the concept of *jumping* in their algorithm ODE. Since ODE is the parent algorithm of SDE, it is justified to see if jumping provides any additional benefit in the working of SDE.

2.10.1 Influence of Dimensionality

As already mentioned, the proposed algorithms were first analyzed on scalable problems of dimension 30. To further investigate effect of the size of the problem on the working of the proposed algorithms the dimension of the scalable problems was varied as $D/2$ ($=15$) and $2D$ ($=60$). The corresponding results in terms of average NFE and Success Rate are reported in Tables 2.13 and 2.14. For dimension 15, MSDE outperformed all the remaining algorithms by a significant difference for all the functions except f_9 , for which neither of the algorithms were able to reach the desired accuracy and were therefore terminated when the maximum NFE was reached. It was also observed that the average success rate for basic DE is 0.83 only whereas for SDE and MSDE the average success rates are 0.93 each and for CDE it is 0.85.

When the dimension was increased to 60, the performance of DE deteriorated in comparison to the proposed algorithms. DE was not able to solve problems f_4 and f_5 for dimension 60 under the given parameter settings. None of the algorithms, except MSDE, were able to solve function f_9 .

2.10.2 Influence of Varying the Population Size (NP)

After observing the effect of varying the size (dimension) of the problem on the proposed algorithms, the effect of varying the population size was also investigated by taking two different population sizes as $NP/2$ ($= 50$) and $2NP$ ($= 200$). The dimension of the scalable problems is fixed as 30. The corresponding NFE and SR are recorded in Tables 2.15 and 2.16. It was observed that for smaller population size ($NP = 50$) all the algorithms performed reasonably well in terms of NFE. However, the success rate deteriorated in comparison to the success rate for population size 100. For larger population ($NP = 200$), the SR improved at the cost of increase in NFE. This is an expected outcome as most of the population based search techniques are sensitive to the population size.

2.10.3 Effect of Jumping on SDE Algorithm

The concept of jumping is such that new points based on OBL are generated within the contracted search space obtained after the end of a generation. Instead of applying jumping at the end of every generation it is applied probabilistically with the help of some predefined random number, known as the jumping rate. The same concept is applied on the proposed SDE algorithm, taking jumping rates as 0.1 and 0.3 and the results are recorded in terms of NFE in Table 2.17. From this Table it can be seen that by applying the concept of jumping, the modified SDE algorithms (SDEj0.3 and SDEj0.1) were not able to solve function f_8 besides f_9 . Also, the average NFE increased. This shows that the idea of jumping is not beneficial for the SDE algorithm.

2.11 Numerical Results for Nontraditional Benchmark Problems

After evaluating the performance of the algorithms for solving traditional benchmark problems, their performance is further validated on a set of 7 nontraditional benchmark functions and the corresponding numerical results are reported in Table 2.18 in terms of best, median, worst and mean error and standard deviation. From these results SDE performed better than DE for all the test cases with an improvement of up to 99% in the best function value for F1, F3 and F5 and an improvement up to 75% for F2, F4 and F6. For the last function F7, the improvement is around 8%. After SDE, it was ODE that gave a good performance. Considering

the complexity of the problems, CDE and MSDE also performed reasonably well though not as good as SDE. It is worth mentioning that only ODE has been taken as a competing algorithm because none of the other algorithms that have been used for comparison of traditional benchmark problems have been employed for solving this set of nontraditional benchmark problems.

2.12 Comparison of SDE and MSDE With Other State of the Art Algorithms

According to the various performance metrics, it was observed that MSDE and SDE outperformed the CDE algorithm. These two algorithms are, therefore, further compared with three other state of the art DE algorithms on the basis of average fitness, standard deviation (Std.), number of function evaluations and success rate (SR). The algorithms taken for comparison are jDE, JADE and SaDE. Although these algorithms are adaptive in nature and their comparison with SDE and MSDE may not be completely justified but these are some of the recent variants of DE and have given good performance in comparison to both adaptive and nonadaptive algorithms. All the algorithms are executed according to the maximum number of generations as given in Table 2.19. The remaining parameters are kept same as discussed in the earlier section 2.7.

2.12.1 Comparison in Terms of Fitness Value and Standard Deviation

From Table 2.19 which gives the results on the basis of fitness and standard deviation it can be seen that all the algorithms performed more or less in a similar manner though SDE, MSDE and JADE outperformed the others in certain cases.

2.12.2 Comparison in Terms of NFE

On the basis of NFE the results are given in Table 2.20. From this Table, it can be noticed that MSDE took lesser NFE in most of the test cases. The average NFE of MSDE is around 24676. The worst performance was given by SaDE, which took an average NFE of more than 75000 for solving the 25 test problems.

2.12.3 Comparison in Terms of SR

On the basis of average SR from Table 2.21, JADE gave the best performance for which the SR comes out to be 96%. SDE came in second with an average SR of 95%. For MSDE, the SR came out to be 92% while for SaDE and for jDE, the success rates are 94% and 92% respectively.

2.12.4 Statistical Analysis

Statistical analysis of the algorithms on the basis of NFE are given in Tables 2.22- 2.24. An overall comparison of algorithms is given in Tables 2.22 and 2.23.

Table 2.22 shows that there is a significant difference between the algorithms. From Table 2.23, it can be observed that MSDE, SDE and JADE are at par with each other while the remaining two algorithms jDE and SaDE do not perform as well as these. This is illustrated graphically in Figure 2.9. Pairwise comparison of MSDE and SDE with JADE, jDE and SaDE is summarized in Table 2.24. From this Table it is noted that, Wilcoxon test shows that there is a significant difference between MSDE, jDE and SaDE, while there is no difference between MSDE, SDE and JADE.

2.13 Summary

This chapter presents three enhanced versions of DE named Differential Evolution with Cauchy mutation (CDE), Differential Evolution with mixed mutation strategy (MSDE) and Synergetic Differential Evolution (SDE). The performance of these algorithms vis-à-vis seven other modified DE versions is analyzed analytically with the help of various performance metrics. The conclusions that can be drawn at the end of this chapter can be summarized as follows:

- From the algorithm design point of view, SDE is perhaps the simplest of the proposed algorithms. It employs three efficient and easy to apply algorithmic components and do not require an additional parameter like that of CDE and MSDE.
- Various analysis have been done to validate the performnace of the proposed algorithms in terms of efficiency, reliability and robustness. Effect of the dimension of the problem and the effect of population size is observed for all the algorithms. It was seen that a population

size (NP) =100 is quite efficient for solving problems up to dimension 30. Also, it was observed that although jumping, a concept of ODE, improved its performance it did not have any added benefit on the proposed SDE.

- On the basis of empirical analysis done for the traditional benchmark problems, it was observed that although on the basis of error and standard deviation no concrete conclusion can be drawn about the performance of the algorithms but by analyzing the other performance criteria like NFE, AR and SR the superior performance of the proposed CDE, SDE and MSDE over the basic DE and most of the competent algorithms (ODE, DERL and MDE) can be clearly observed. A comparison of the proposed algorithms among themselves indicates that SDE and MSDE are significantly better than CDE.
- Analysis of AR shows that MSDE is 58% faster while SDE is 46% faster than the basic DE, while CDE is around 13% faster than the basic DE. Also it is observed that SDE was able to solve the 25 traditional benchmark problems with 95% SR while MSDE and CDE gave an SR of 92% and 88% respectively.
- Credibility of the proposed algorithms is further validated by their successful performance for solving the 7 nontraditional benchmark problems.
- The further comparisons of MSDE and SDE (which performed better than the third variant CDE) with some other state of the art DE variants also show the competence of these two variants for solving the unconstrained benchmark problems.
- Statistical analysis of the results strengthens the fact that the proposed algorithms are better or at par with the traditional DE as well as with other algorithms.

Table 2.1: Sensitivity analysis of Maximum Failure Counter (MFC) in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.

Fun	D	MFC = 3	MFC = 5	MFC = 10
f_1	30	93430	85070	87310
f_2	30	164620	145230	133930
f_3	30	111700	99320	98900
f_4	30	269892	268990	272329
f_5	30	--	--	--
f_6	30	7190	9760	13480
f_7	30	283600	133814	250375
f_8	30	85370	78250	78540
f_9	30	--	--	--
f_{10}	30	170340	150620	142370
f_{11}	30	85890	79410	79180
f_{12}	30	75120	70780	71560
f_{13}	30	87790	80300	78890
f_{14}	30	124750	102000	113390
f_{15}	2	1060	1350	1870
f_{16}	2	4940	5640	5770
f_{17}	2	4090	3970	4150
f_{18}	2	5570	4330	5450
f_{19}	2	3500	3560	4000
f_{20}	3	6360	4910	5600
f_{21}	4	10266	10375	10622
f_{22}	4	10642	10230	10922
f_{23}	4	10780	9950	10811
f_{24}	4	35050	11950	33633
f_{25}	6	20650	12580	17550
<i>Average</i>		41049.88	35122.19	37235.5

Table 2.2: Sensitivity analysis of parameter Gama (γ) in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE .

Fun	D	$\gamma = .001$	$\gamma = .1$	$\gamma = .25$	$\gamma = .33$	$\gamma = .95$
f_1	30	25820	26750	25680	24980	26380
f_2	30	37530	36480	37270	36590	37130
f_3	30	28300	28640	27230	27370	28330
f_4	30	113900	110830	112670	111120	113210
f_5	30	--	--	--	--	--
f_6	30	6890	7340	6910	7460	7230
f_7	30	24200	23670	21780	20080	23290
f_8	30	128630	127820	128110	129420	129220
f_9	30	30280	31730	31520	32760	32140
f_{10}	30	40700	40880	39910	39160	40250
f_{11}	30	26880	26670	25770	26230	25420
f_{12}	30	23710	24280	23670	24660	23280
f_{13}	30	28600	28220	29120	28270	27330
f_{14}	30	24520	26390	25640	25320	25240
f_{15}	2	2740	2740	2570	2870	2810
f_{16}	2	1930	1820	1880	1780	1910
f_{17}	2	1300	1370	1320	1400	1410
f_{18}	2	1780	1720	1770	1740	1620
f_{19}	2	2350	2380	2360	2270	2290
f_{20}	3	2460	2510	2630	2620	2490
f_{21}	4	13760	11340	13280	10020	11270
f_{22}	4	8690	8470	8610	8380	8260
f_{23}	4	7530	7690	7480	7880	7560
f_{24}	4	7650	7730	7410	7580	7670
f_{25}	6	11728	11680	11787	12266	12130
Average		19816.4	19822.5	19676.35	19608.3	19641

Table 2.3: Comparison of proposed algorithms with DE and their parent algorithms in terms of error and standard deviation (Std.).

Fun	D	DE	MDE	ODE	DERL	CDE	MSDE	SDE
f_1	30	5.24848e-32 (2.50302e-32)	3.43423e-48 (8.43943e-50)	6.34393e-41 (2.40300e-43)	3.15789e-63 (8.92424e-65)	3.95502e-33 (1.57648e-33)	1.39832e-95 (9.27821e-97)	1.77622e-76 (6.41905e-78)
f_2	30	7.64876e-16 (5.96538e-16)	4.23421e-25 (3.45932e-26)	3.44593e-21 (3.49532e-25)	2.10125e-31 (7.48388e-34)	1.20976e-18 (5.85541e-19)	1.75133e-77 (1.73323e-77)	9.76428e-38 (6.74634e-38)
f_3	30	1.77664e-30 (1.04434e-30)	3.45543e-38 (1.38309e-40)	7.43932e-34 (4.30094e-35)	1.56214e-62 (3.40335e-60)	3.22957e-31 (2.34992e-32)	3.93808e-89 (1.19131e-93)	1.07703e-75 (4.44153e-77)
f_4	30	2.57862e-04 (1.51977e-07)	9.23275e-04 (2.48839e-05)	4.49593e-04 (6.43490e-04)	6.54522e-04 (5.30902e-04)	1.52536e-04 (3.33618e-05)	2.80722e-08 (2.78822e-09)	1.22089e-09 (6.95936e-09)
f_5	30	1.74229e-01 (1.43011e+00)	2.34439e-02 (1.24943e+00)	7.32341e-01 (1.82344e+01)	4.13605e-14 (8.49383e-17)	1.88390e+01 (1.22789e+00)	2.22409e+01 (2.34321e+00)	1.14867e-25 (1.52704e-26)
f_6	30	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_7	30	7.08548e-03 (6.67423e-03)	3.43094e-03 (3.40023e-03)	5.39234e-03 (4.30893e-03)	3.32073e-03 (3.49588e-03)	8.55517e-03 (1.46972e-03)	5.62432e-03 (2.26735e-04)	1.91822e-03 (2.51635e-03)
f_8	30	2.34343e-11 (1.43054e-12)	2.24994e-11 (1.73204e-12)	2.09323e-11 (1.37237e-12)	1.87393e-11 (1.92821e-12)	1.27329e-11 (1.81899e-12)	1.07308e-11 (0)	1.40439e-11 (1.13294e-12)
f_9	30	1.49594e+02 (1.70130e+02)	1.30031e+02 (2.30439e+01)	1.13584e+02 (3.00283e+02)	1.24723e+02 (9.43885e+01)	7.42845e+01 (6.66094e+00)	0 (0)	4.92040e+01 (1.49244e+01)
f_{10}	30	3.69735e-15 (0)	3.69735e-15 (0)	3.69735e-15 (0)	3.69735e-15 (0)	7.25006e-15 (1.66306e-30)	1.44633e-16 (0)	3.69735e-15 (0)
f_{11}	30	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_{12}	30	1.35360e-19 (0)	4.49594e-21 (0)	1.04493e-19 (0)	3.54594e-23 (0)	1.35361e-19 (0)	1.35361e-19 (0)	1.86577e-31 (0)
f_{13}	30	1.29115e-19 (0)	3.45943e-20 (0)	1.03113e-19 (0)	5.43222e-22 (0)	1.29115e-19 (0)	1.29115e-19 (0)	1.46878e-29 (0)
f_{14}	30	1.95165e-32 (2.06563e-32)	5.32332e-42 (8.34993e-43)	3.32494e-38 (7.39920e-41)	1.12489e-64 (5.94992e-67)	4.12296e-30 (3.31649e-30)	1.96421e-86 (2.33493e-94)	1.47748e-76 (5.93376e-77)
f_{15}	2	5.55045e-12 (0)	5.55045e-12 (0)	5.55045e-12 (0)	5.55045e-12 (0)	5.55045e-12 (0)	5.55038e-12 (0)	5.55045e-12 (0)

Table 2.3 Contd.....

f_{16}	2	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)	4.89878e-10 (2.22045e-16)
f_{17}	2	1.56086e-11 (0)	1.56086e-11 (0)	1.56086e-11 (0)	1.56086e-11 (0)	1.56086e-11 (0)	1.56086e-11 (0)	1.56086e-11 (0)
f_{18}	2	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_{19}	2	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_{20}	3	1.73501e-10 (0)	1.73501e-10 (0)	1.73501e-10 (0)	1.73501e-10 (0)	1.73501e-10 (0)	1.73501e-10 (0)	1.73501e-10 (0)
f_{21}	4	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)	9.41771e-10 (3.67257e-10)
f_{22}	4	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)	3.18134e-09 (2.87417e-09)
f_{23}	4	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)	3.30795e-09 (1.77636e-15)
f_{24}	4	2.48594e-08 (4.84870e-08)	2.07859e-08 (6.32116e-09)	2.07859e-08 (6.32116e-09)	2.07859e-08 (6.32116e-09)	2.07859e-08 (6.32116e-09)	2.07859e-08 (6.32116e-09)	2.07859e-08 (6.32116e-09)
f_{25}	6	4.75550e-02 (5.82455e-02)	3.32671e-02 (5.44837e-02)	3.86751e-02 (5.34922e-02)	3.26753e-02 (4.23901e-02)	7.13559e-02 (6.13961e-02)	2.37786e-02 (5.01296e-02)	3.56650e-02 (5.03213e-02)

Table 2.4: Comparison of proposed algorithms with DE and their parent algorithms in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE

Fun	D	NFE						
		DE	MDE	ODE	DERL	CDE	MSDE	SDE
f_1	30	104310	94700	101040	56700	85070	24980	45980
f_2	30	173850	160240	165570	93890	145230	36590	77830
f_3	30	110700	101400	102400	59700	99320	27370	48600
f_4	30	274150	297600	263140	245250	268990	111120	258886
f_5	30	--	--	--	257100	--	--	190600
f_6	30	31890	28770	30030	17080	9760	7460	14850
f_7	30	131640	137370	130680	80660	133814	20080	70680
f_8	30	226850	210986	222033	108800	78250	129420	101067
f_9	30	--	--	--	--	--	32760	--
f_{10}	30	163020	149200	162310	87430	150620	39160	72800
f_{11}	30	108930	99600	106300	58430	79410	26230	48077
f_{12}	30	95400	85600	94460	50910	70780	24660	43340
f_{13}	30	104310	91100	104060	55110	80300	28270	46680
f_{14}	30	104540	91500	100300	55050	102000	25320	46580
f_{15}	2	5220	5360	5260	3640	1350	2870	3330
f_{16}	2	5720	4810	5690	4020	5640	1780	3330
f_{17}	2	6930	6750	7050	4970	3970	1400	4790
f_{18}	2	4470	3930	4460	3200	4330	1740	2850
f_{19}	2	4160	3840	4350	3190	3560	2270	2640
f_{20}	3	5010	4390	4950	3410	4910	2620	2870
f_{21}	4	11990	10350	11920	7570	10375	10020	6640
f_{22}	4	11290	9380	11260	7430	10230	8380	6220
f_{23}	4	11330	10090	11090	7440	9950	7880	6190
f_{24}	4	11220	9750	11800	7780	11950	7580	6050
f_{25}	6	14400	13100	13560	8825	12580	12266	7050

Table 2.5: Comparison of proposed algorithms with DE and their parent algorithms in terms of AR. Here '--' indicates that AR cannot be calculated for them.

Fun	D	Acceleration Rate (AR)					
		MDE vs. DE	ODE vs. DE	DERL vs. DE	CDE vs. DE	MSDE vs. DE	SDE vs. DE
f_1	30	9.21	3.13	45.64	18.45	76.05	55.92
f_2	30	7.83	4.76	45.99	16.46	78.95	55.23
f_3	30	8.40	7.50	46.07	10.28	75.28	56.10
f_4	30	-8.55	4.02	10.54	1.88	59.47	5.57
f_5	30	--	--	--	--	--	--
f_6	30	9.78	5.83	46.44	69.39	76.61	53.43
f_7	30	-4.35	0.73	38.73	-1.65	84.75	46.31
f_8	30	6.99	2.12	52.04	65.51	42.95	55.45
f_9	30	--	--	--	--	--	--
f_{10}	30	8.48	0.44	46.37	7.61	75.98	55.34
f_{11}	30	8.57	2.41	46.36	27.10	75.92	55.86
f_{12}	30	10.27	0.99	46.64	25.81	74.15	54.57
f_{13}	30	12.66	0.24	47.17	23.02	72.90	55.25
f_{14}	30	12.47	4.06	47.34	2.43	75.78	55.44
f_{15}	2	-2.68	-0.77	30.27	74.14	45.02	36.21
f_{16}	2	15.91	0.52	29.72	1.40	68.88	41.78
f_{17}	2	2.60	-1.73	28.28	42.71	79.80	30.88
f_{18}	2	12.08	0.22	28.41	3.13	61.07	36.24
f_{19}	2	7.69	-4.57	23.32	14.42	45.43	36.54
f_{20}	3	12.38	1.20	31.94	2.00	47.70	42.71
f_{21}	4	13.68	0.58	36.86	13.47	16.43	44.62
f_{22}	4	16.92	0.27	34.19	9.39	25.78	44.91
f_{23}	4	10.94	2.12	34.33	12.18	30.45	45.37
f_{24}	4	13.10	-5.17	30.66	-6.51	32.44	46.08
f_{25}	6	9.03	5.83	38.72	12.64	14.82	51.04
Average		9.03	1.51	38.72	12.62	58.11	46.12

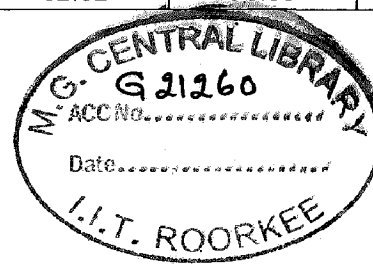


Table 2.6: Comparison of proposed algorithms with DE and their parent algorithms in terms of SR.

Fun	D	Success Rate (SR)						
		DE	MDE	ODE	DERL	CDE	MSDE	SDE
f_1	30	1	1	1	1	1	1	1
f_2	30	1	1	1	1	1	1	1
f_3	30	1	1	1	1	1	1	1
f_4	30	0.36	0.52	0.52	0.44	0.39	0.8	0.9
f_5	30	0	0	0	1	0	0	1
f_6	30	1	1	1	1	1	1	1
f_7	30	1	1	1	1	1	1	1
f_8	30	1	1	1	1	1	1	1
f_9	30	0	0	0	0	0	1	0
f_{10}	30	1	1	1	1	1	1	1
f_{11}	30	1	1	1	1	1	1	1
f_{12}	30	1	1	1	1	1	1	1
f_{13}	30	1	1	1	1	1	1	1
f_{14}	30	1	1	1	1	1	1	1
f_{15}	2	1	1	1	1	1	1	1
f_{16}	2	1	1	1	1	1	1	1
f_{17}	2	1	1	1	1	1	1	1
f_{18}	2	1	1	1	1	1	1	1
f_{19}	2	1	1	1	1	1	1	1
f_{20}	3	1	1	1	1	1	1	1
f_{21}	4	1	1	1	1	0.8	0.9	1
f_{22}	4	1	1	1	1	1	1	1
f_{23}	4	1	1	1	1	1	1	1
f_{24}	4	1	1	1	1	0.9	0.7	1
f_{25}	6	0.84	0.48	0.62	0.44	0.9	0.60	0.8
<i>Average</i>		0.89	0.88	0.89	0.92	0.88	0.92	0.95

Table 2.7: Results of Friedman's test based on error of Table 2.3.

N	Friedman value	df	p-value
25	36.097	6	<0.001

df - Degrees of freedom

N - Total No of functions

Table 2.8: Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.3.

Algorithm	Mean Rank
DE	4.98
SDE	2.88
MDE	4.00
ODE	4.24
DERL	3.48
CDE	5.00
MSDE	3.42
CD for $\alpha = 0.05$	1.611845
CD for $\alpha = 0.10$	1.462758

Table 2.9: Results of pair wise comparison based on error of Table 2.3.

SDE Vs.	Wilcoxon test								
	+ve rank	-ve rank	tie	Mean of +ve rank	Mean of -ve rank	Sum of +ve rank	Sum of -ve rank	Stat.	p-value
DE	13	0	12	7.00	0.00	91.0	0	-3.180	0.001
MDE	11	1	13	6.80	10.0	68.0	10	-2.275	0.023
ODE	12	0	13	6.50	0.00	78.0	0	-3.059	0.002
DERL	11	1	13	6.09	11.0	67.0	11	-2.197	0.028
CDE	14	1	10	8.00	8.00	112.0	8	-2.953	0.003
MSDE	7	9	9	10.29	7.11	72.0	64	-0.207	0.836

Table 2.10: Results of Friedman's test based on NFE of Table 2.4.

N	Friedman value	df	p-value
25	114.339	6	<0.001

df – Degrees of freedom N - Total No of functions

Table 2.11: Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.4.

Algorithm	Mean Rank
DE	6.46
SDE	2.02
SDE	5.78
ODE	2.94
DERL	4.90
CDE	4.22
MSDE	1.68
CD for $\alpha = 0.05$	1.611845
CD for $\alpha = 0.10$	1.462758

Table 2.12: Results of pairwise comparison based on NFE of Table 2.4.

MSDE Vs.	Wilcoxon test								
	+ve rank	-ve rank	tie	Mean of +ve rank	Mean of -ve rank	Sum of +ve rank	Sum of -ve rank	Stat.	p-value
DE	24	0	1	12.50	0	300	0	-4.286	<0.001
MDE	24	0	1	12.50	0	300	0	-4.286	<0.001
ODE	24	0	1	12.50	0	300	0	-4.286	<0.001
DERL	19	6	0	13.95	10	265.0	60.0	-2.758	0.006
CDE	22	2	1	12.82	9.0	282	18	-3.771	<0.001
SDE	18	7	0	13.50	11.71	243	82	-2.166	0.030

Table 2.13: Influence of dimensionality on the proposed algorithms and DE in terms of NFE. Here ‘--’ indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.

Fun	D = 15				D = 60			
	DE	CDE	MSDE	SDE	DE	CDE	MSDE	SDE
f_1	49050	36090	14630	23250	192400	147180	41060	85800
f_2	80440	61860	23150	39350	305120	221170	56720	134890
f_3	52900	40960	15550	24100	215700	199700	45900	96800
f_4	123210	120243	53050	60400	--	--	239800	528000
f_5	--	--	--	62550	--	--	--	546000
f_6	14310	4790	4270	7040	57000	27230	11890	26990
f_7	43740	56400	8510	29580	451600	423820	68920	432000
f_8	57960	33780	41530	40155	594000	222900	201240	516000
f_9	--	--	20330	--	--	--	51620	--
f_{10}	77990	65950	24430	37750	288800	234920	62040	125900
f_{11}	98700	46850	17070	39200	184600	123390	41860	85500
f_{12}	44640	30710	13380	21450	165000	140970	45030	90400
f_{13}	47460	33870	15080	23240	191400	136360	54310	102000
f_{14}	45380	42220	13660	21550	208100	160900	44690	96000

Table 2.14: Influence of dimensionality on the proposed algorithms and DE in terms of SR.

Fun	D = 15				D = 60			
	DE	CDE	MSDE	SDE	DE	CDE	MSDE	SDE
f_1	1	1	1	1	1	1	1	1
f_2	1	1	1	1	1	1	1	1
f_3	1	1	1	1	1	1	1	1
f_4	0.65	0.9	1	1	0	0	0.8	0.73
f_5	0	0	0	1	0	0	0	0.89
f_6	1	1	1	1	1	1	1	1
f_7	1	1	1	1	1	1	1	1
f_8	1	1	1	1	0.9	1	0.5	1
f_9	0	0	1	0	0	0	1	0
f_{10}	1	1	1	1	1	1	1	1
f_{11}	1	1	1	1	0.47	1	1	1
f_{12}	1	1	1	1	1	1	1	1
f_{13}	1	1	1	1	1	1	1	1
f_{14}	1	1	1	1	1	1	1	1
Average	0.83	0.85	0.93	0.93	0.74	0.79	0.88	0.90

Table 2.15: Influence of varying population size on the proposed algorithms and DE in terms of NFE. Here ‘--’ indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.

Fun	D	NP = 50				NP = 200			
		DE	CDE	MSDE	SDE	DE	CDE	MSDE	SDE
f_1	30	40310	41345	11945	19770	286200	171180	50720	128800
f_2	30	61460	69215	16910	29170	--	295220	74920	223940
f_3	30	42350	49600	12995	21445	--	203600	56400	138000
f_4	30	--	278390	52775	--	--	--	222800	--
f_5	30	--	--	--	--	--	--	--	--
f_6	30	12320	5695	3695	6165	87000	15760	14220	39800
f_7	30	70455	150910	14795	65195	260000	--	42980	245600
f_8	30	96650	38605	65288	75362	--	157380	257400	296244
f_9	30	--	--	12805	--	--	--	74400	--
f_{10}	30	62650	72745	18455	32450	--	--	79960	202600
f_{11}	30	41450	39305	12770	20350	297000	157720	53880	133120
f_{12}	30	42177	35040	13810	20438	208321	139640	46260	118560
f_{13}	30	65218	38985	16265	27255	186323	160960	51760	123600
f_{14}	30	52287	52365	11890	21840	165328	144620	51480	128000
f_{15}	2	2572	845	2365	1690	6521	2260	5040	5820
f_{16}	2	2577	2845	900	1540	8723	9280	2920	6740
f_{17}	2	3572	2160	775	2000	13272	7440	2200	9660
f_{18}	2	2466	2785	970	1405	9832	10600	3320	5320
f_{19}	2	2784	1955	1260	1395	6299	6920	4080	5140
f_{20}	3	2943	3045	1425	1485	8734	7320	4620	5560
f_{21}	4	6282	5192	8142	3050	18923	20100	22820	12900
f_{22}	4	7132	5171	4950	2970	20223	19825	16220	12120
f_{23}	4	6980	5570	3815	3005	19890	17422	14880	11920
f_{24}	4	6721	19383	29816	3450	18723	17865	12320	11440
f_{25}	6	6823	13783	6966	3350	19838	20100	28777	14688

Table 2.16: Influence of varying population size on proposed algorithms and DE in terms of SR.

Fun	D	NP = 50				NP = 200			
		DE	CDE	MSDE	SDE	DE	CDE	MSDE	SDE
f_1	30	1	1	1	1	1	1	1	1
f_2	30	1	1	1	1	0	1	1	1
f_3	30	1	1	1	1	0	1	1	1
f_4	30	0	1	1	0	0	0	1	0
f_5	30	0	0	0	1	0	0	0	0
f_6	30	1	1	1	1	1	1	1	1
f_7	30	1	1	1	1	1	0	1	1
f_8	30	0.65	0.56	0.88	0.63	0	1	0.91	0.96
f_9	30	0	0	1	0	0	0	1	0
f_{10}	30	1	1	1	1	1	0	1	1
f_{11}	30	0.87	1	1	1	1	1	1	1
f_{12}	30	1	1	1	1	1	1	1	1
f_{13}	30	1	1	1	1	1	1	1	1
f_{14}	30	1	1	1	1	1	1	1	1
f_{15}	2	1	1	1	1	1	1	1	1
f_{16}	2	1	1	1	1	1	1	1	1
f_{17}	2	1	1	1	1	1	1	1	1
f_{18}	2	1	1	1	1	1	1	1	1
f_{19}	2	1	1	1	1	1	1	1	1
f_{20}	3	1	1	1	1	1	1	1	1
f_{21}	4	1	0.9	0.7	1	1	1	1	1
f_{22}	4	1	0.8	0.9	1	1	1	1	1
f_{23}	4	1	0.5	1	1	1	0.9	1	1
f_{24}	4	1	0.9	0.6	1	1	0.5	0.9	1
f_{25}	6	0.72	0.6	0.6	0.56	0.89	0.8	0.9	0.93
<i>Average</i>		0.85	0.85	0.91	0.89	0.90	0.77	0.95	0.95

Table 2.17: Effect of jumping on proposed SDE algorithm with jumping rates as 0.1 and 0.3. The results are tabulated for number of function evaluations (NFE). Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.

Fun	D	SDE	SDEj0.3	SDEj0.1
f_1	30	45980	48870	48270
f_2	30	77830	80960	78540
f_3	30	48600	53400	50700
f_4	30	258886	267000	261000
f_5	30	190600	248300	212000
f_6	30	14850	15540	14670
f_7	30	70680	82000	75420
f_8	30	101067	--	--
f_9	30	--	--	--
f_{10}	30	72800	76670	73200
f_{11}	30	48077	52383	49444
f_{12}	30	43340	43811	43280
f_{13}	30	46680	48744	46920
f_{14}	30	46580	48300	47700
f_{15}	2	3330	3265	3290
f_{16}	2	3330	3243	3510
f_{17}	2	4790	4840	4520
f_{18}	2	2850	2983	3070
f_{19}	2	2640	6560	2960
f_{20}	3	2870	3045	3560
f_{21}	4	6640	6538	6930
f_{22}	4	6220	6458	6510
f_{23}	4	6190	6245	6350
f_{24}	4	6050	6234	6540
f_{25}	6	7050	7143	7620

Table 2.18: Comparison of proposed algorithms with DE and ODE for nontraditional shifted functions in terms of error (best median, worst and mean) and standard deviation (Std.).

Problem	D	Error value	DE	ODE (Rahnamayan and Wang, 2008)	CDE	MSDE	SDE
F_1	500	Best	2, 636.54	15.66	10.48	4.98	3.48
		Median	3, 181.45	36.61	19.32	5.89	5.32
		Worst	4, 328.80	292.65	23.43	8.28	7.57
		Mean	3, 266.24	80.17	19.54	5.03	4.86
		Std.	409.68	79.24	37.32	5.78	4.34
F_2	500	Best	79.74	3.60	20.42	2.78	19.82
		Median	82.39	4.86	12.32	4.93	11.88
		Worst	85.92	11.91	34.69	7.32	12.26
		Mean	82.93	5.78	27.00	3.92	11.87
		Std.	2.09	2.37	8.50	2.49	1.93
F_3	500	Best	76, 615, 772.08	39, 718.90	807,641.56	763,323.45	727, 996.00
		Median	119, 733, 49.20	137, 279.03	582,743.33	782,301.74	731, 546.21
		Worst	169, 316,779.50	407, 661.64	673,494.23	789,873.56	732, 763.93
		Mean	123, 184, 755.70	154, 306.34	467,232.85	779,289.90	730, 473.25
		Std.	29, 956, 737.58	114, 000.53	124,845.43	117,328.93	116,325.43
F_4	500	Best	5, 209.99	2, 543.51	1,287.74	1,183.84	1, 155.15
		Median	5, 324.57	4, 279.56	4,133.49	4,382.63	3, 243.87
		Worst	5, 388.24	6, 003.94	5,238.32	4,829.43	4, 478.90
		Mean	5, 332.59	4, 216.34	4,141.32	4,132.54	4, 212.76
		Std.	43.82	1, 017.94	61.32	67.32	58.60
F_5	500	Best	24.29	1.25	1.10	1.09	0.31
		Median	24.71	1.55	1.36	1.32	0.87
		Worst	27.59	2.13	1.56	1.58	0.96
		Mean	25.16	1.75	1.52	1.38	0.56
		Std.	1.10	0.37	0.26	0.18	0.05
F_6	500	Best	4.66	2.49	2.52	2.37	1.18
		Median	4.97	4.12	3.98	3.83	1.47
		Worst	5.15	6.73	4.13	5.18	1.56
		Mean	4.94	4.51	4.02	4.27	1.25
		Std.	0.17	1.44	0.14	1.06	0.07
F_7	500	Best	-3683.07	-3957.85	-3961.29	-3983.32	-3992.76
		Median	-3575.13	-3834.07	-3832.43	-3889.49	-3836.65
		Worst	-3565.73	-3830.36	-3830.24	-3738.54	-3833.21
		Mean	-3593.75	-3851.82	-3856.47	-3883.85	-3863.59
		Std.	32.74	38.80	31.34	34.54	29.31

Table 2.19: Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of fitness function value.

Fun	D	#Gen	jDE	JADE	SaDE	MSDE	SDE
f_1	30	1500	2.34343e-28 (1.92383e-28)	1.73443e-60 (7.34344e-60)	3.54533e-20 (5.79432e-20)	4.94619e-70 (1.02143e-70)	2.74298e-36 (3.94901e-36)
f_2	30	2000	3.09343e-23 (8.38772e-24)	2.38353e-25 (8.47876e-25)	1.02398e-14 (1.83421e-15)	1.57077e-51 (1.15935e-51)	1.10654e-24 (8.28990e-25)
f_3	30	5000	3.39041e-14 (3.82921e-14)	4.44584e-61 (1.32743e-60)	9.04322e-37 (3.44302e-37)	0 (0)	4.81002e-131 (0)
f_4	30	5000	0 (0)	8.43245e-24 (4.20037e-23)	6.49202e-11 (1.63430e-10)	6.38436e-43 (3.87941e-43)	5.92984e-11 (8.36854e-10)
f_5	30	20000	0 (0)	8.94840e-02 (5.97326e-01)	2.34993e-01 (2.33498e-01)	8.30564e+00 (9.43304e+00)	0 (0)
f_6	30	1500	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_7	30	3000	2.31545e-03 (7.38443e-04)	8.54564e-04 (2.32534e-04)	3.58832e-03 (1.62992e-03)	7.36002e-04 (4.32394e-03)	2.05093e-04 (1.04551e-03)
f_8	30	9000	-12569.5 (8.00132e-12)	-12569.5 (0)	-12569.5 (8.43901e-08)	-12569.5 (1.45519e-11)	-12569.5 (1.09766e-10)
f_9	30	5000	0 (0)	0 (0)	0 (0)	0 (0)	8.95493e+00 (1.59359e+01)
f_{10}	30	1500	7.09431e-15 (1.72928e-15)	5.65784e-15 (0)	7.38286e-14 (3.48321e-14)	1.44633e-16 (0)	4.05954e-15 (0)
f_{11}	30	2000	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
f_{12}	30	1500	5.93708e-30 (2.32384e-30)	1.06754e-32 (3.43503e-48)	2.43748e-19 (0)	1.3536e-19 (0)	1.35993e-30 (0)
f_{13}	30	1500	6.90221e-29 (3.84204e-29)	4.65656e-32 (4.14394e-48)	2.83043e-19 (0)	1.29115e-19 (2.39491e-20)	1.29390e-29 (0)
f_{14}	30	3000	3.49941e-51 (7.34301e-53)	6.67607e-61 (8.57008e-63)	7.38393e-58 (3.45843e-60)	1.84302e-83 (3.94943e-89)	1.47748e-76 (5.93376e-77)
f_{15}	2	100	0.998004 (1.90023e-16)	0.998004 (1.77884e-16)	0.998004 (1.32943e-16)	0.998004 (1.32943e-16)	0.998004 (1.21077e-16)
f_{16}	2	100	-1.03163 (8.43843e-12)	-1.03163 (2.56648e-15)	-1.03163 (1.48430e-16)	-1.03163 (2.22045e-16)	-1.03163 (2.22875e-16)
f_{17}	2	100	0.397887 (4.43492e-08)	0.397887 (0)	0.397887 (0)	0.397887 (0)	0.397887 (0)
f_{18}	2	100	3.0 (1.98237e-15)	3.0 (2.42127e-17)	3.0 (2.43493e-16)	3.0 (6.8798e-16)	3.0 (1.61278e-16)
f_{19}	2	100	-1 (0)	-1 (0)	-1 (0)	-1 (0)	-1 (0)
f_{20}	3	100	-3.8623 (9.32384e-15)	-3.8626 (7.76755e-14)	-3.8623 (7.34399e-15)	-3.8623 (3.97205e-16)	-3.8623 (4.44089e-16)
f_{21}	4	100	-10.1532 (3.34321e-06)	-10.1532 (4.88743e-13)	-10.1532 (4.23484e-15)	-10.1525 (3.44387e-11)	-10.1532 (1.77532e-15)
f_{22}	4	100	-10.4029 (5.25234e-07)	-10.4029 (8.57584e-13)	-10.4029 (2.43438e-15)	-10.4029 (3.45834e-15)	-10.4029 (1.25879e-15)
f_{23}	4	100	-10.5364 (5.02913e-06)	-10.5364 (8.76765e-11)	-10.5364 (7.43483e-14)	-10.5364 (3.15095e-07)	-10.5364 (1.94865e-15)
f_{24}	4	4000	4.29044e-04 (3.28494e-04)	6.78786e-05 (3.07102e-04)	8.43984e-04 (4.54989e-08)	3.07491e-04 (2.43617e-08)	3.05132e-04 (9.71256e-09)
f_{25}	6	100	-3.2863 (6.34934e-06)	-3.2986 (5.75433e-05)	-3.3182 (6.34348e-03)	-3.28625 (2.34984e-03)	-3.2807 (2.39493e-03)

Table 2.20: Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of NFE. Here '--' indicates that the algorithms were not able to obtain the desired accuracy for specified maximum NFE.

Fun	D	jDE	JADE	SaDE	MSDE	SDE
f_1	30	60100	29900	73490	24980	45980
f_2	30	83220	52550	118932	36590	77830
f_3	30	299399	94840	181673	27370	48600
f_4	30	297650	170890	290380	111120	258886
f_5	30	--	151000	278890	--	190600
f_6	30	24860	11560	28410	7460	14850
f_7	30	98000	30000	128764	20080	70680
f_8	30	88940	130480	121830	129420	101067
f_9	30	118630	131000	170765	32760	--
f_{10}	30	90620	45610	119090	39160	72800
f_{11}	30	64270	34000	80688	26230	48077
f_{12}	30	54310	26950	72346	24660	43340
f_{13}	30	61287	30988	73432	28270	46680
f_{14}	30	71290	50380	89372	25320	46580
f_{15}	2	3578	3455	3672	2870	3330
f_{16}	2	3298	3310	3320	1780	3330
f_{17}	2	4872	4520	4810	1400	4790
f_{18}	2	3389	3080	3010	1740	2850
f_{19}	2	3456	3050	3480	2270	2640
f_{20}	3	3154	2990	3080	2620	2870
f_{21}	4	6829	6745	6770	10020	6640
f_{22}	4	6194	6389	6395	8380	6220
f_{23}	4	6530	6090	6672	7880	6190
f_{24}	4	6648	6532	6438	7580	6050
f_{25}	6	7410	7000	7832	12266	7050

Table 2.21: Comparison of SDE and MSDE with jDE, JADE, and SaDE in terms of SR.

Fun	D	jDE	JADE	SaDE	MSDE	SDE
f_1	30	1	1	1	1	1
f_2	30	1	1	1	1	1
f_3	30	1	1	1	1	1
f_4	30	0.8	0.92	0.86	0.8	0.9
f_5	30	0	0.7	0.29	0	1
f_6	30	1	1	1	1	1
f_7	30	1	1	1	1	1
f_8	30	0.68	1	0.81	1	1
f_9	30	1	1	1	1	0
f_{10}	30	1	1	1	1	1
f_{11}	30	1	1	1	1	1
f_{12}	30	1	1	1	1	1
f_{13}	30	1	1	1	1	1
f_{14}	30	1	1	1	1	1
f_{15}	2	1	1	1	1	1
f_{16}	2	1	1	1	1	1
f_{17}	2	1	1	1	1	1
f_{18}	2	1	1	1	1	1
f_{19}	2	1	1	1	1	1
f_{20}	3	1	1	1	1	1
f_{21}	4	1	1	1	0.9	1
f_{22}	4	1	1	1	1	1
f_{23}	4	1	1	1	1	1
f_{24}	4	1	1	1	0.7	1
f_{25}	6	0.41	0.47	0.44	0.60	0.8
<i>Average</i>		0.92	0.96	0.94	0.92	0.95

Table 2.22: Results of Friedman's test based on NFE of Table 2.20.

N	Friedman value	df	p-value
25	33.852	4	<0.001

df – Degrees of freedom

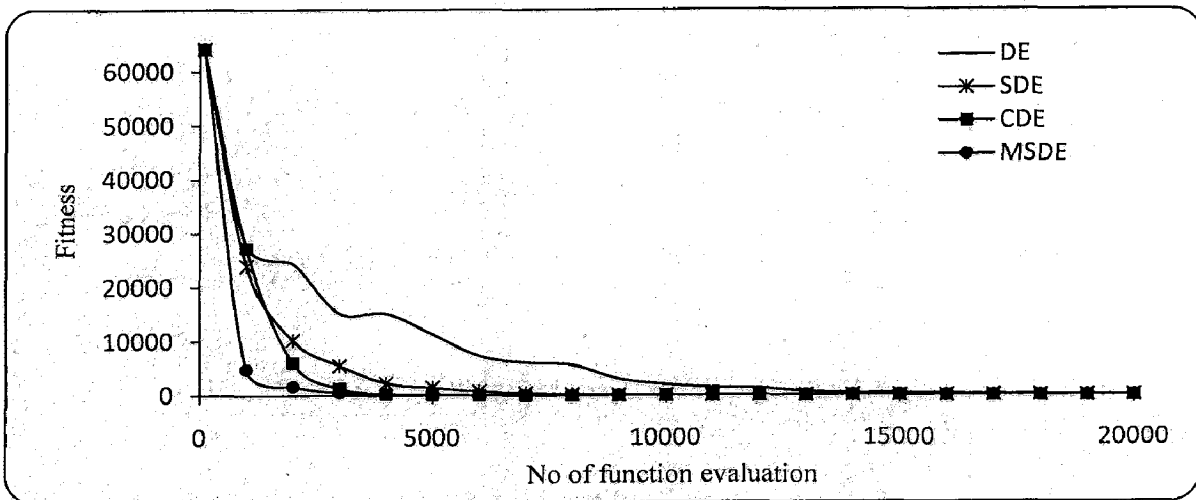
N - Total No of functions

Table 2.23: Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure of Table 2.20.

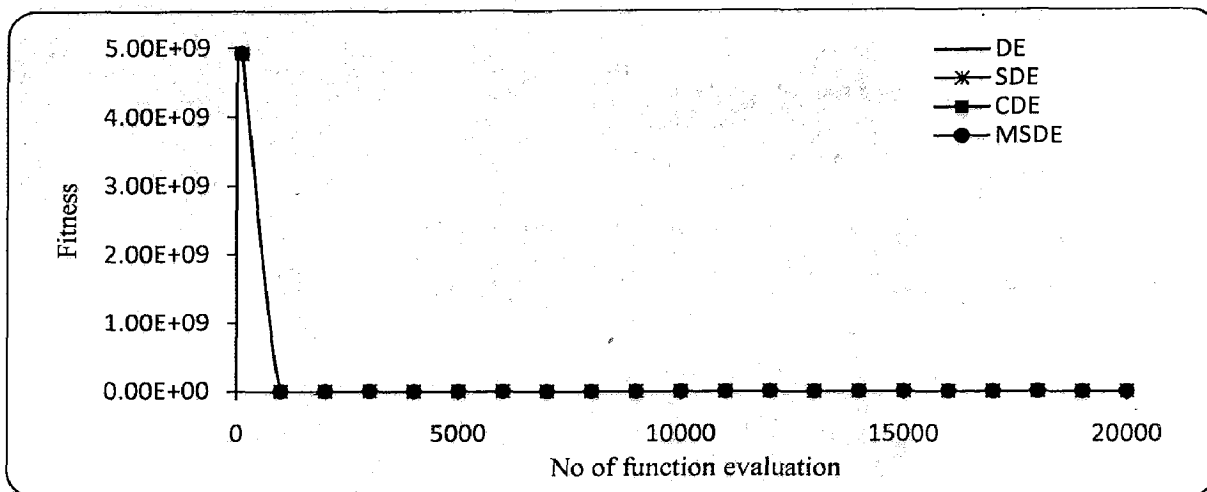
Algorithm	Mean Rank
jDE	3.74
JADE	2.44
SaDE	4.20
SDE	2.56
MSDE	2.06
CD for $\alpha = 0.05$	1.11714
CD for $\alpha = 0.10$	1.002206

Table 2.24: Results of pairwise comparison based on NFE of Table 2.20.

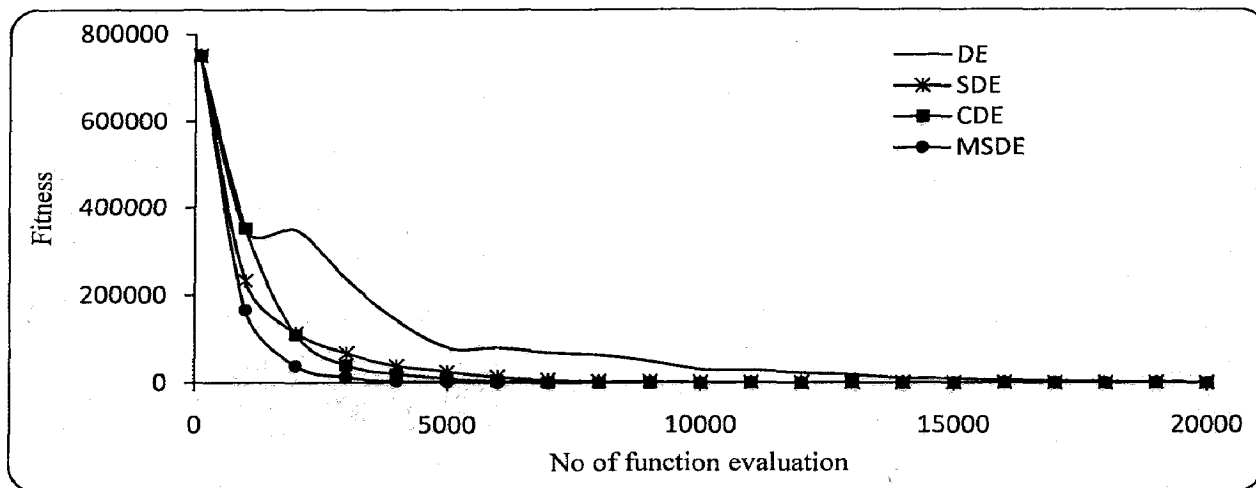
MSDE Vs.	Wilcoxon test								
	+ve rank	-ve rank	tie	Mean of +ve rank	Mean of -ve rank	Sum of +ve rank	Sum of -ve rank	Stat.	p-value
jDE	18	6	1	13.72	8.83	247	53	-2.771	0.006
JADE	19	6	0	13.6	12.5	250	75	-2.354	0.019
SaDE	18	7	0	14.67	8.71	264	61	-2.731	0.006
SDE	18	7	0	13.5	11.71	243	82	-2.166	0.030



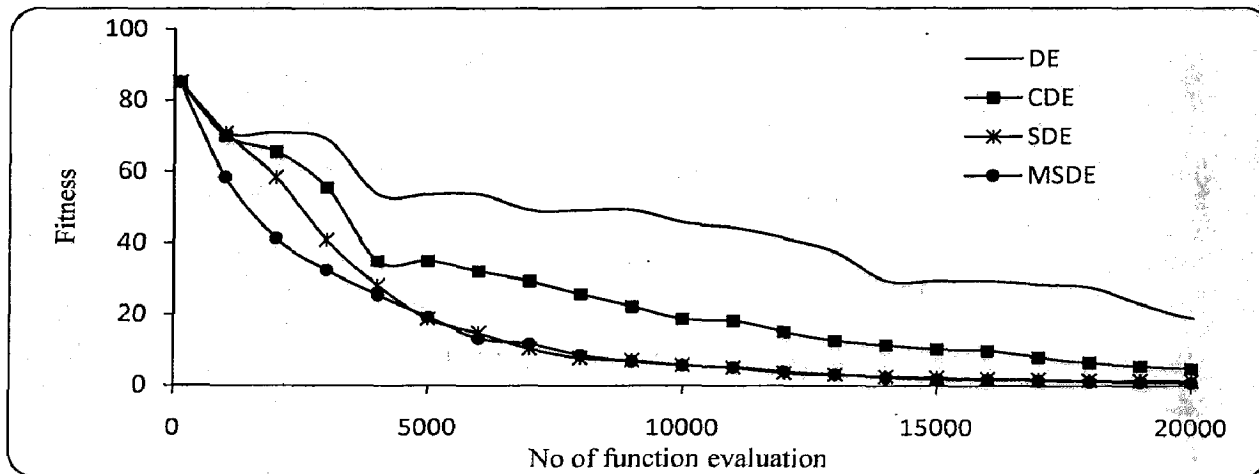
(a)



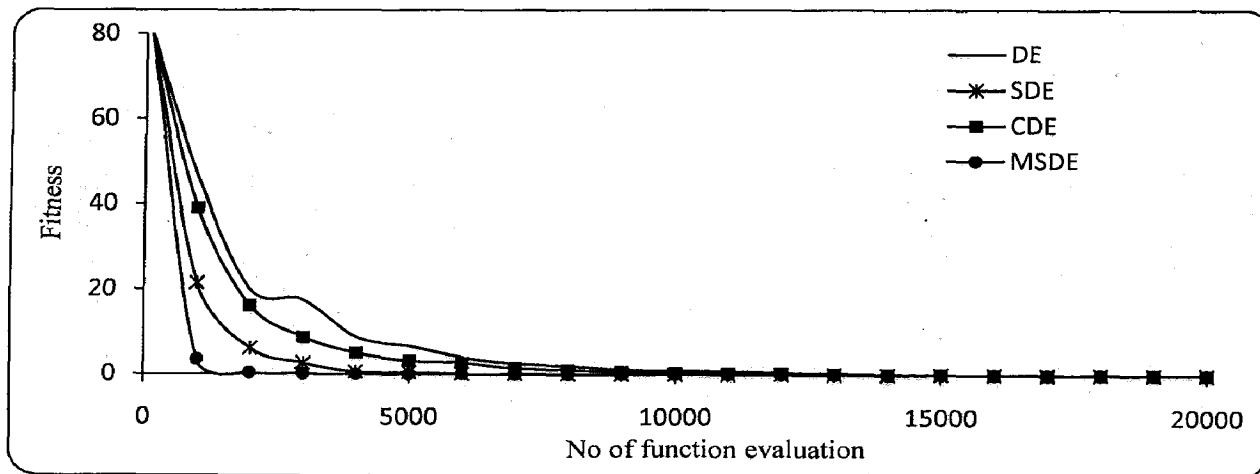
(b)



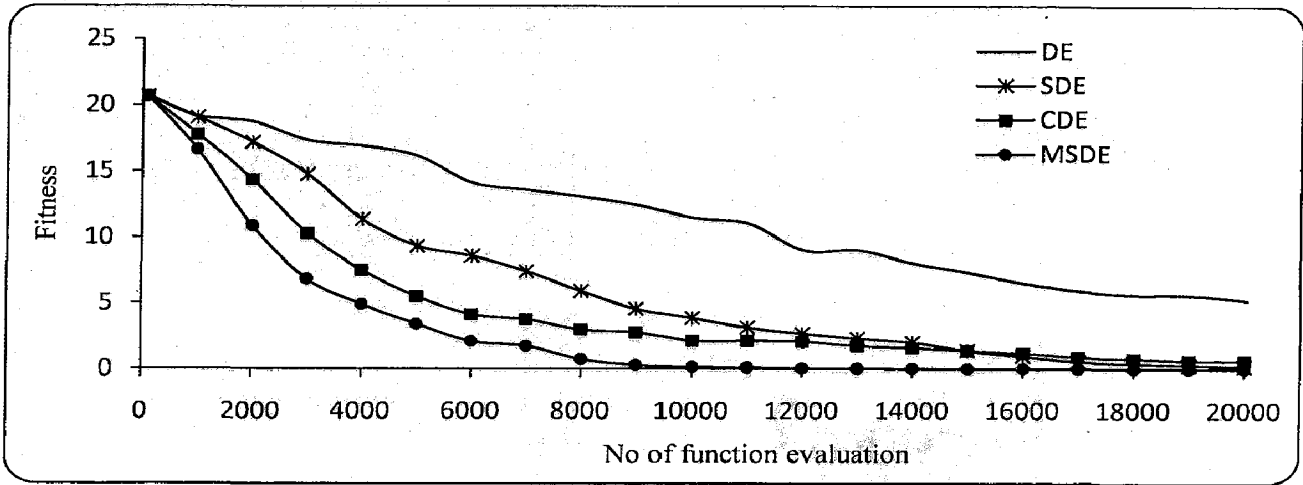
(c)



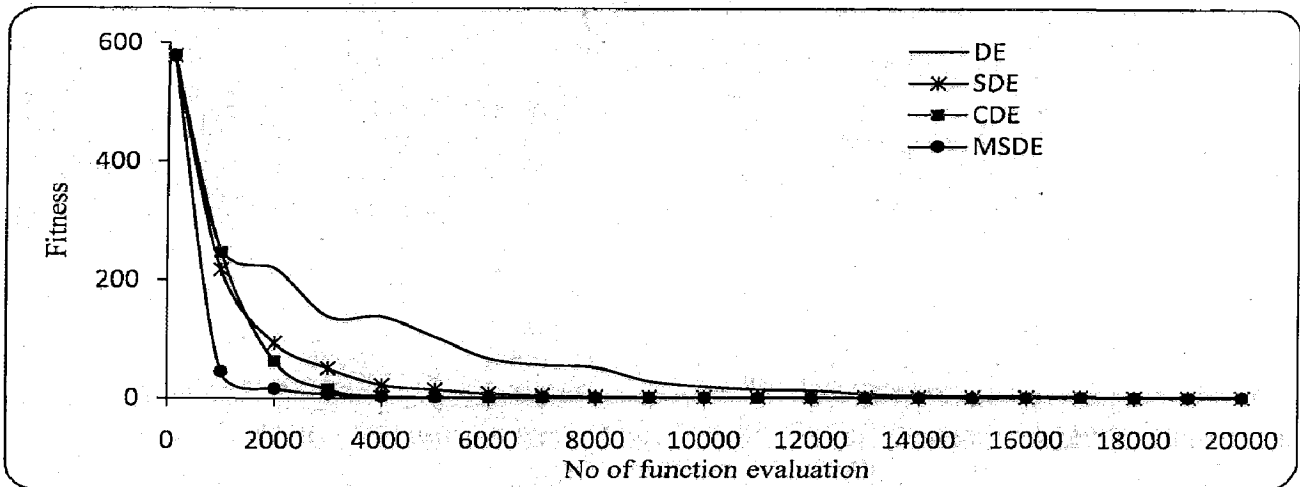
(d)



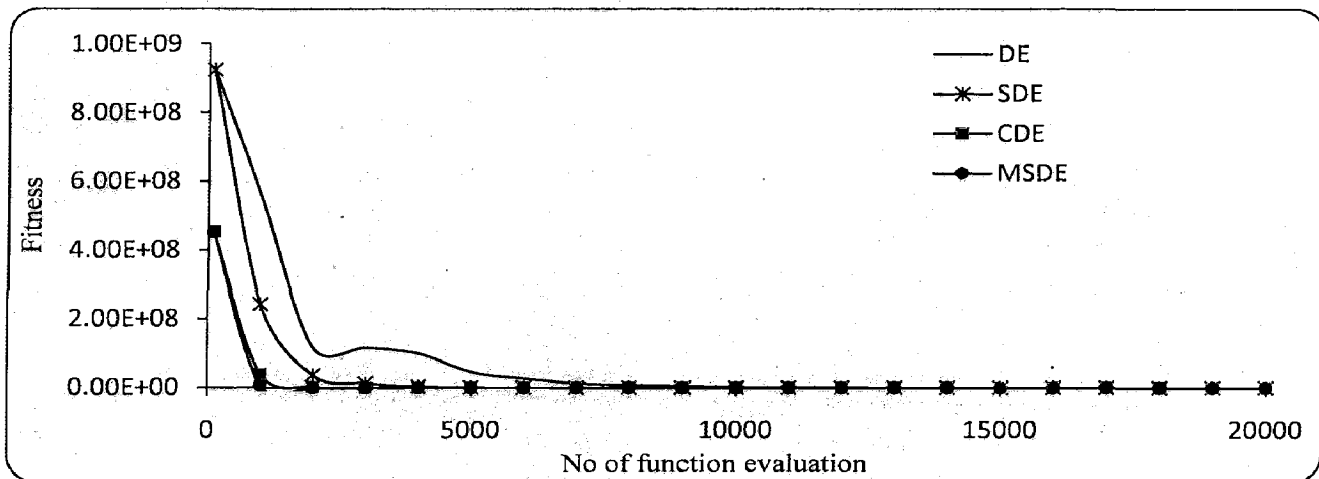
(e)



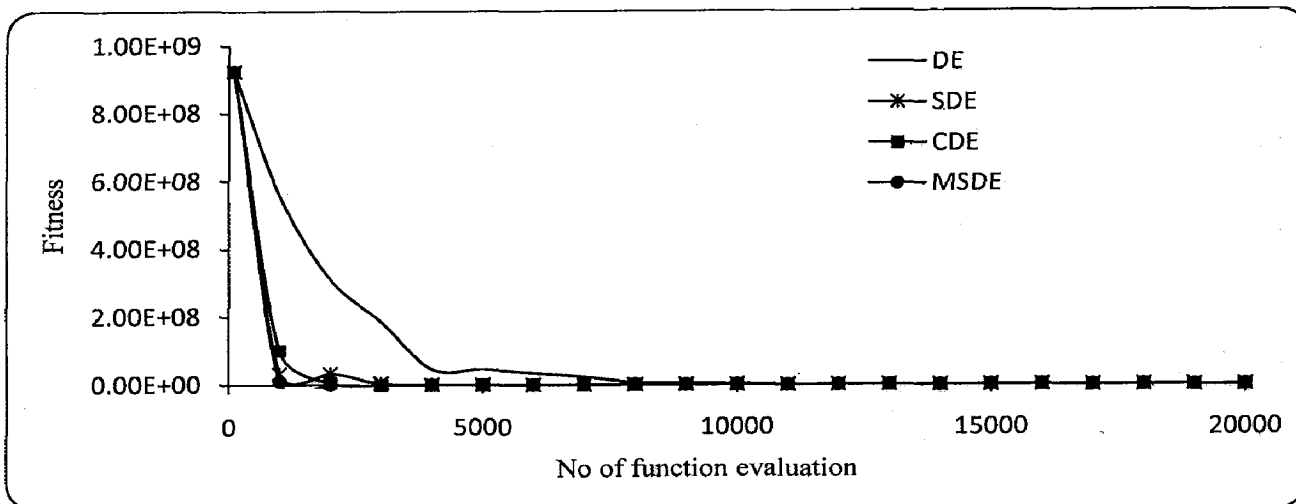
(f)



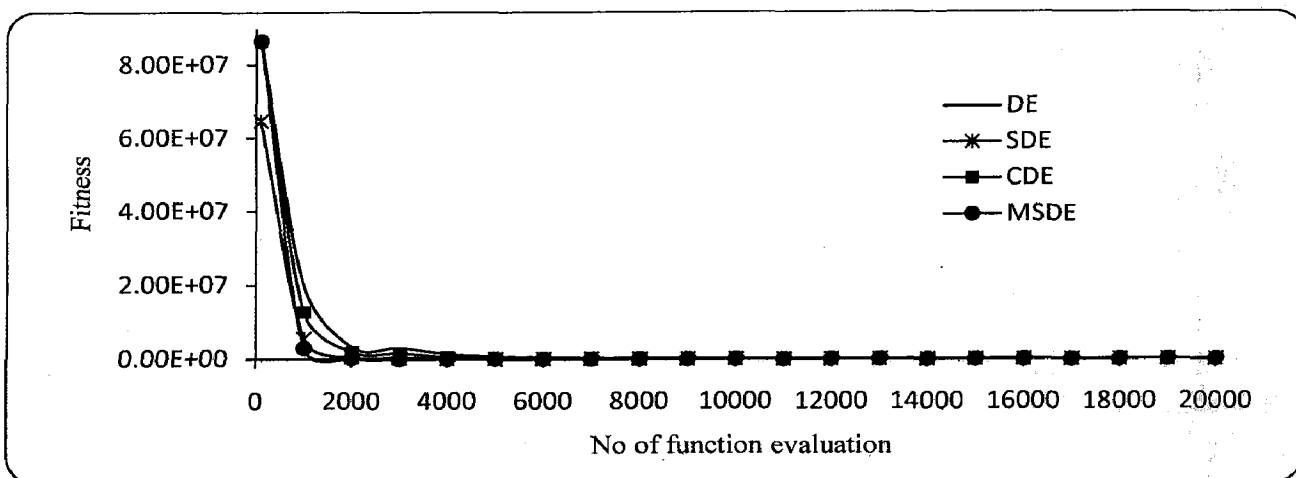
(g)



(h)



(i)



(j)

Figure 2.6: Performance graphs (best solution versus NFE) for performance comparison between DE and SDE. (a) – (j) represents functions $f_1, f_2, f_3, f_4, f_7, f_{10}, f_{11}, f_{12}, f_{13}$ and f_{14} .

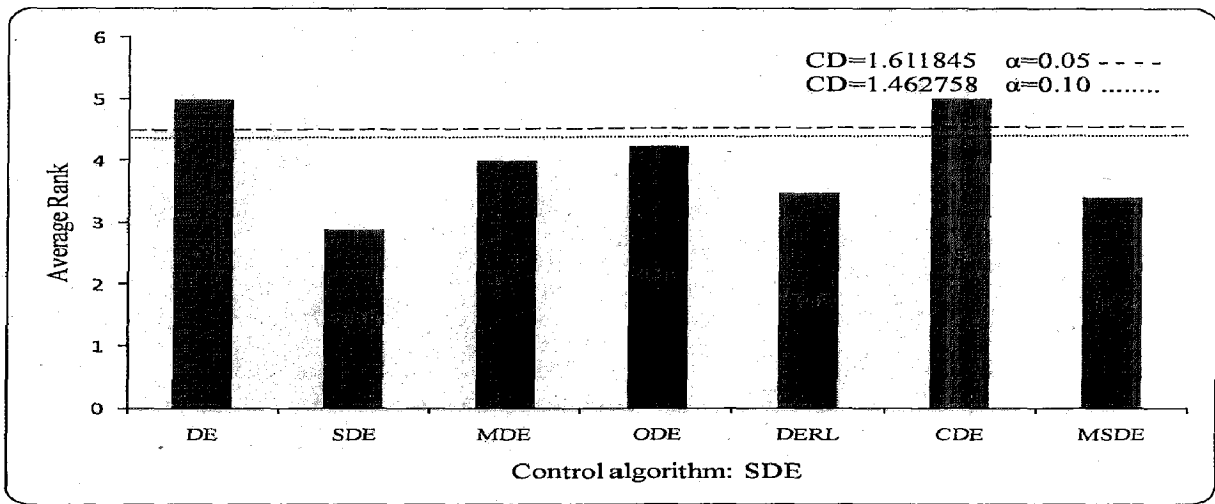


Figure 2.7: Bonferroni-Dunn's graphic corresponding to error of Table 2.3.

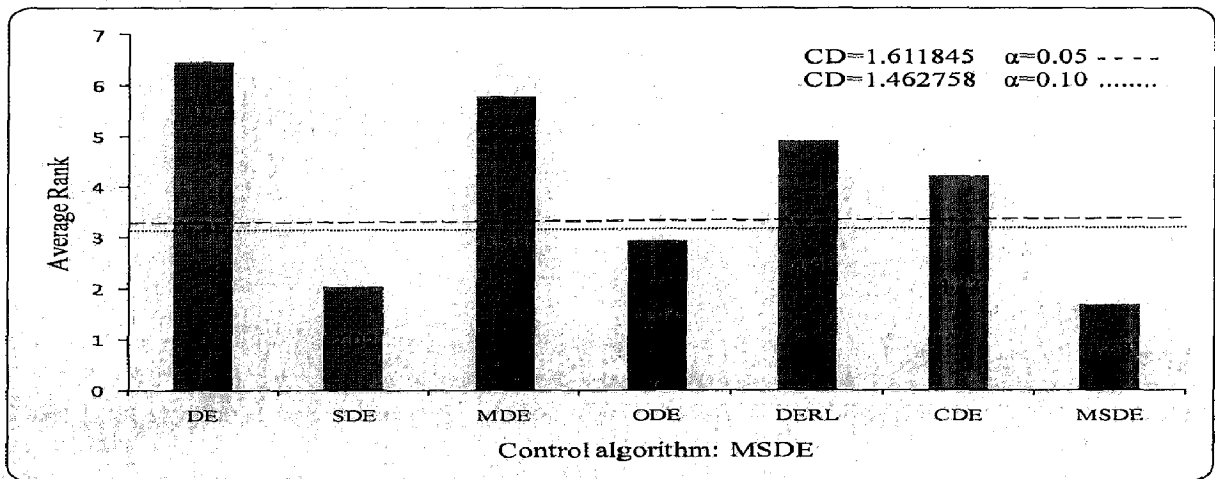


Figure 2.8: Bonferroni-Dunn's graphic corresponding to NFE of Table 2.4.

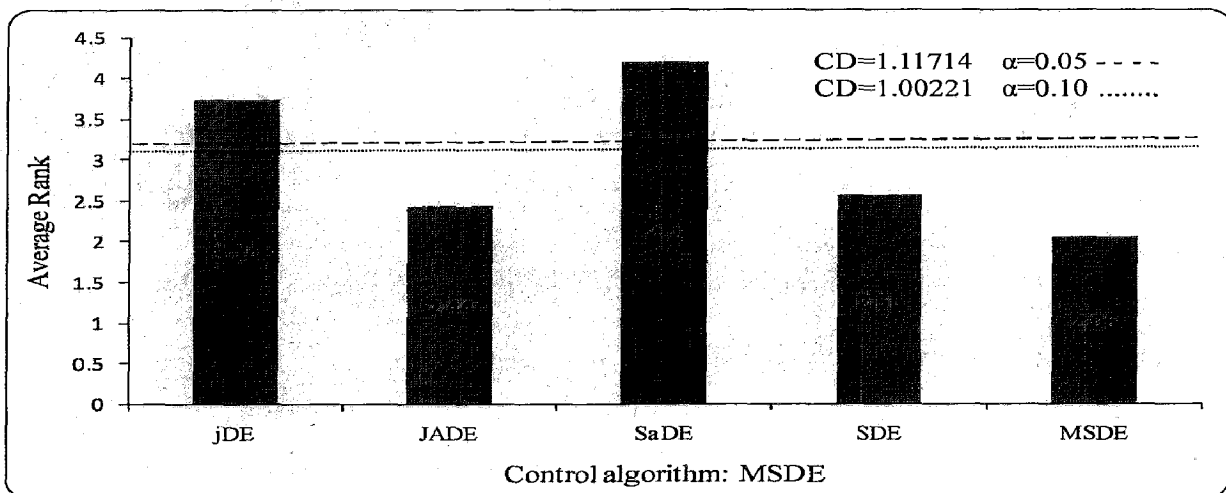


Figure 2.9: Bonferroni-Dunn's graphic corresponding to NFE of Table 2.20.

Constrained Differential Evolution

In chapter 2, three enhanced variants of DE namely DE with Cauchy mutation (CDE), DE with Mixed Mutation Strategy (MSDE), and Synergetic DE (SDE) were proposed and validated on a set of unconstrained continuous optimization problems. In the present chapter, MSDE and SDE algorithms are further extended for solving the constrained optimization problems as it was observed that although all the three algorithms were compatible for solving the unconstrained optimization problems the performance of MSDE and SDE was superior to that of CDE. The performance of these two algorithms is validated on a test suite of 24 constrained benchmark problems proposed in CEC 2006 (Liang et al., 2006) and the algorithms are also compared with some of the contemporary DE variants available in literature for solving the constrained optimization problems.

The chapter is structured as follows. Section 3.1 gives the introduction of constrained optimization. Section 3.2 describes the techniques for handling the constraints. Benchmark test suit is given in section 3.3. Performance evaluation criteria are given in section 3.4. Parameters setting are given in section 3.5. Results are analyzed and discussed in section 3.6 and finally, the chapter is concluded in section 3.7.

3.1 Introduction

Optimization models of most of the real life problems occurring in the field of science of technology are often subjected to constraints and the corresponding problems are called constrained optimization problems (COP). The search space in COPs consists of two kinds of solutions: feasible and infeasible. Feasible points satisfy all the constraints, while infeasible points violate at least one of them. Therefore, the final solution of an optimization problem must satisfy all constraints.

A general COP is formulated in chapter 1, having l inequality and m equality constraints, respectively. If both the objective and the constraints are linear then the problem is said to be a

linear programming problem but if either one of them (or both) is non linear then it is said to be a non linear COP.

Considering that formulation, a solution X is called feasible if

$$g_j(X) \leq 0, \text{ for all } j = 1, \dots, l$$

$$|h_k(X)| - \varepsilon \leq 0, \text{ for all } k = 1, \dots, m$$

In the present chapter equality constraints are transformed into inequality constraints where ε is taken as 0.0001.

A measure of the average constraint violation, which is often useful while handling constraints, is defined as:

$$\bar{v} = \frac{\sum_{j=1}^l G_j(X) + \sum_{k=1}^m H_k(X)}{l + m}$$

$$\text{where } G_j(X) = \begin{cases} g_j(X) & \text{if } g_j(X) > 0 \\ 0 & \text{if } g_j(X) \leq 0 \end{cases}$$

$$H_k(X) = \begin{cases} |h_k(X)| & \text{if } |h_k(X)| - \varepsilon > 0 \\ 0 & \text{if } |h_k(X)| - \varepsilon \leq 0 \end{cases}$$

There are many traditional methods in the literature for solving COP. However, most of the traditional methods require certain auxiliary properties (like convexity, continuity etc.) of the problem and also most of the traditional techniques are suitable for only a particular type of problem (for example Quadratic Programming Problems, Geometric Programming Problems etc). Keeping in view the limitations of traditional techniques researchers have proposed the use of population based stochastic optimization methods and intelligent algorithms like Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE) etc. for solving COP.

3.1.1 Constraint Handling Techniques Applied to Population Based Search Techniques

Based on the research efforts in literature, constraint handling methods have been categorized in a number of classes (Engelbrecht, 2005). These are:

➤ *Reject infeasible solutions*, where solutions are not constrained to the feasible space.

Solutions that find themselves in infeasible space are simply rejected or ignored.

- *Penalty function methods*, which add a penalty to the objective function to discourage search in infeasible areas of the search space.
- *Converting the constrained problem to an unconstrained problem* and then solving the unconstrained problem.
- *Preserving feasibility methods*, which assumes that solutions are initialized in feasible space, and applies specialized operators to transform feasible solutions to new feasible solutions. These methods constrict solutions to move only in feasible space, where all constraints are satisfied at all times.
- *Pareto ranking methods*, which use concepts from multi-objective optimization, such as nondominance, to rank solutions based on degree of violation.
- *Repair methods*, which apply special operators or actions to infeasible solutions to facilitate changing infeasible solutions to feasible solutions.

A state of the art survey on constrained optimization is given in (Coello Coello, 2002).

3.1.2 Application of DE for Solving COP

DE has been successfully applied for solving constrained optimization problems. Several variants of it are available in literature. A feasible region shrinking mechanism was proposed by Storn (1999). The aim was to relax all the constraints of the problems at the beginning of the process. As the time progresses, the pseudo-feasible region shrink at each generation until it matches the real feasible region. Storn also proposed the idea of each parent to generate more than one offspring. In Storn's approach the process finishes when one offspring is better than its parent or when (say) certain numbers of offspring have been generated. He applied his approach on "DE/best/1/bin" version. He also added an aging mechanism to avoid a solution to remain in the population for too long. His approach was well suited for problems having inequality constraints, but was not very efficient while dealing with equality constraints. A static-penalty approach, coupled with DE to solve engineering design problems was proposed by Lampinen and Zelinka (1999a, 1999b, 1999c, 1999d). The main drawback of their approach is that a careful tuning is required for the penalty factors. Chiang et al. (2002) proposed an augmented Lagrangian approach with an adaptive mechanism to update the penalty parameters. The approach performed well against typical EA-based techniques. An extension of DE to solve

constrained optimization problems was proposed by Lampinen (2002). The original DE replacement mechanism (based only on the objective function value of the parent and its corresponding offspring) was substituted by three selection criteria based on feasibility originally proposed by Deb (2000). The difference between Deb's and Lampinen's approach is in the third rule. In Deb's approach the solution with the lowest sum of constraint violation is selected. On the other hand, in Lampinen's technique, the solution which Pareto-dominates the other in the constraints space will be selected. Mezura et al. (2004) proposed a DE-based approach where the newly generated offspring is added to the current generation (instead of including the solution to the next generation). The idea was to allow newly generated solutions to be selected to influence the selection of search directions of the offspring in the current generation and to speed up the convergence. More recently a constrained based approach was suggested by Ali and Bagdadi (2009).

The applications of DE to the constrained optimization problems can also be found in Storn (1999), Lampinen (2002), Koziel and Michalewicz (1999), Lampinen (2001a, 2001b), Lin et al. (2002), Chiou and Wang (1999), Sarimveis and Nikolakopoulos (2005) Becerra and Carlos (2005) and Carlos (2002).

3.2 Constraint Handling Techniques Used in the Present Study

In the present study, Pareto ranking method proposed by Deb (2000) for constraint-handling is adopted for dealing with COP. This method is based on the following three rules:

- Between two feasible vectors, the one with the best value of the objective function is preferred.
- If one vector is feasible and the other one is infeasible, the feasible one is preferred.
- Between two infeasible vectors, the one with the lowest sum of constraint violation is preferred.

Besides following the above three rules, equality constraints were transformed into inequations as explained in Section 3.1 by using the following tolerance value: $\epsilon = 1e^{-4}$.

3.3 Test Problems

24 benchmark problems provided in the special session on constrained problems in CEC 2006 (Liang et al., 2006) are considered in the present chapter for analyzing the performance of MSDE and SDE. This test suite consists of a wide variety of COP including 7 quadratic problems, 7 nonlinear problems, 2 cubic problems, 2 polynomial problems and 6 linear problems. The number of variables vary from 2 to 24 and the problems consist of both equality and inequality type constraints. The list of problems along with the function code, actual minima and other characteristics is summarized in Table 3.1. Mathematical model of the problems is given in Appendix II.

Table 3.1: Name of constrained test problems, assigned codes and characteristics

Function	D	$f(X^*)$	Type of function	N_{EQ}	N_{IEQ}	N_a
g01	13	-15.0000000000	quadratic	0	9	6
g02	20	-0.8036191042	nonlinear	0	2	1
g03	10	-1.0005001000	polynomial	1	0	1
g04	5	-30665.538671783	quadratic	0	6	2
g05	4	5126.4967140071	cubic	3	2	3
g06	2	-6961.8138755802	cubic	0	2	2
g07	10	24.3062090681	quadratic	0	8	6
g08	2	-0.0958250415	nonlinear	0	2	0
g09	7	680.6300573745	polynomial	0	4	2
g10	8	7049.2480205286	linear	0	6	6
g11	2	0.7499000000	quadratic	1	0	1
g12	3	-1.0000000000	quadratic	0	1	0
g13	5	0.0539415140	nonlinear	3	0	3
g14	10	-47.7648884595	nonlinear	3	0	3
g15	3	961.7150222899	quadratic	2	0	2
g16	5	-1.9051552586	nonlinear	0	38	4
g17	6	8853.5396748064	nonlinear	4	0	4
g18	9	-0.8660254038	quadratic	0	13	6
g19	15	32.6555929502	nonlinear	0	5	0
g20	24	0.2049794002	linear	14	6	16
g21	7	193.7245100700	linear	5	1	6
g22	22	236.4309755040	linear	19	1	19
g23	9	-400.0551000000	linear	4	2	6
g24	2	-5.5080132716	linear	0	2	2

D – Dimension of the problem

$f(X^*)$ - best known function value

N_{EQ} – Number of equality constraints

N_{IEQ} – Number of inequality constraints

N_a – Number of active constraints

3.4 Performance Metrics

In order to authenticate the viability of the proposed algorithms a series of experiments have been conducted following various criteria as suggested in CEC 2006 to test their efficiency, robustness and reliability. The performance measures and their definitions are given below:

- Feasible Run: A run during which at least one feasible solution is found in maximum NFE.
- Successful Run: A run during which the algorithm finds a feasible solution X satisfying $(f(X) - f(X^*)) \leq \varepsilon$.
- Feasible Rate = (Number of feasible runs) / total runs.
- Success Rate = (Number of successful runs) / total runs.
- Success Performance = mean (NFEs for successful runs) x (Number of total runs) / (Number of successful runs).
- Convergence graphs: The performance of the proposed algorithms is also illustrated graphically with the help of convergence graphs (or performance curves) where log of error values are plotted against the NFEs.
- Statistical analysis: The proposed algorithms are also analyzed statistically on the basis of non parametric tests.

3.4.1 Algorithms Used for Comparison

The performance of MSDE and SDE is further compared with 6 other variants of DE for solving the COP considered in this chapter.

- ZRDE (Zielinski and Rainer, 2006). Since the authors have not mentioned any particular name for their algorithm it shall be referred to as ZRDE.
- jDE-2 (Brest et al., 2006b).
- ε -DE, (Takahama and Sakai, 2006).
- MDE (Tasgetiren and Suganthan, 2006).
- MDE1 (Mezura-Montes et al., 2006). Here also since the authors have not specified any name of the algorithm it shall be referred to as MDE1.
- SaDE (Huang et al., 2006).

All these algorithms have been successfully used for solving the constrained benchmark problems proposed in the special session of CEC 2006

3.5 Parameter Settings

After conducting several experiments and consulting the literature, following settings have been taken for all the experiments unless otherwise mentioned.

- Maximum NFE = 500000, fixed for all problems (Liang et al., 2006).
- Accuracy (ϵ) = 10^{-04} for all the test problems (Liang et al., 2006).

All other parameters are same as given in chapter 2.

3.6 Results and Discussions

3.6.1 Results of MSDE and SDE

Tables 3.2 – 3.13 are devoted exclusively to the performance of MSDE and SDE. In Tables 3.2 – 3.5 and 3.7-3.10, the performance of MSDE and SDE is recorded in terms of best, worst and average error along with the standard deviation (Std.) while increasing the NFE (number of function evaluations) to three different values 5×10^3 , 5×10^4 and 5×10^5 .

In Tables 3.6 and 3.11, the results of MSDE and SDE are summarized after fixing the accuracy criteria as 0.0001. The comparison of MSDE with SDE algorithm is given in Tables 3.12 and 3.13. In Tables 3.2 – 3.5, and 3.7-3.10 'c' denotes the number of violated constraints (including the number of violations by more than 0.0001, 0.01 and 1) and \bar{v} denotes the mean violation at the median solution and the number in parentheses indicate the unsatisfied constraints.

From Tables 3.2-3.5 and 3.7-3.10 it is clear that both MSDE and SDE found the results very close to global optimal. The error is very small in every case except for g03, g20 and g22. From Tables 3.6 and 3.11, which give the number of function evaluations, it can be seen that the results are quite encouraging. MSDE and SDE gave 100% feasibility rate for all the test problems except for g20 and g22 where it is 0%, implying that none of these algorithms were able to find a feasible solution for these problems.

MSDE gave 100% SR in 17 test problems, 88% SR in two test problems and 72% and 60% SR in one problem each. However, for g03, g20 and g22, it was not able to obtain any success. While for SDE the SR is 100% for 16 test problems, 84%, 80%, 60%, 65% and 24% for one each. However, for g03, g20 and g22, it was not able to achieve any success. On an average the feasibility rate of MSDE is 92% and the SR is 84%. While for SDE the feasibility rate is 91% and SR is 80%.

In terms of convergence speed it can be seen that SDE takes lesser number of function evaluation than MSDE in almost all the cases. Convergence graphs of all the problems are illustrated in Figures 3.1- 3.8 for both the algorithms.

3.6.2 Comparison With Other Algorithms

Results for comparison of MSDE and SDE with other versions of DE are given in Tables 3.14 and 3.15. From these Tables, it can be seen that the success rate is highest for ϵ DE, which is 91%. MSDE performed either at par or better than other algorithms. For jDE-2 and ZRDE, the success rate is 74% and 78%, respectively, and for SaDE, it is 83.5%. For MDE and MDE1, the success rate is same as that of MSDE i.e. 84%.

It should be noted here that for g20 and g22 neither of the algorithms taken in the present study were able to give a successful performance. However, for g03, ϵ DE, MDE, MDE1 and SaDE were able to perform well while jDE and ZRDE along with MSDE and SDE were not successful at all.

In order to further analyze the algorithms they were compared statistically. Non parametric tests were conducted on the algorithms in terms of mean function evaluations to derive some conclusion. First, the algorithms are compared by using non parametric Friedman's and Bonferroni Dunn's test to analyze the overall performance. These tests, for which the results are given in Tables 3.16 and 3.17, show that all the algorithms (except jDE-2 and SaDE) perform at par with each other in terms of average function evaluations and were superior to jDE-2 and SaDE. This can also be observed from Figure 3.9, which illustrates graphically, the behavior of all the algorithms. Secondly, the algorithm SDE was analyzed one by one with all the algorithms using the Wilcoxon test. From this test for which the results are given in Table

3.18, it can be seen that SDE is better than MSDE, ZRDE and jDE2 but it is at par with other algorithms.

3.7 Summary

In the present study the MSDE and SDE are extended for solving COP. The performance of these algorithms is investigated on a set of 24 constrained benchmark problems including linear, non-linear, cubic, polynomial and quadratic programming problems etc. These algorithms are also compared with six other versions of constrained DE available in literature viz. ZRDE, jDE-2, MDE, MDE1, ϵ DE and SaDE. Some conclusions drawn at the end of this chapter are as follows:

- Both MSDE and SDE performed quite satisfactorily while dealing with COP. It was observed that on an average, MSDE gave a success rate of 84% and SDE 80%.
- A statistical comparison of MSDE and SDE with other variants showed that all the algorithms perform at par with each other in terms of average number of NFE except for jDE-2 and SaDE which did not perform as well as the other algorithms.
- It was however also observed that MSDE and SDE were not able to give a successful performance for three test cases; g03, g20 and g22 indicating that further investigation is needed. However all these cases are highly constrained in nature and some other algorithms also (ZRDE and jDE-2) were not able to solve these problems either.

Table 3.2: Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 1-6.

FES \ Prob.		g01	g02	g03	g04	g05	g06
5×10^3	Best	2.2328e+00(0)	3.50263e-01(0)	9.18082e-01(0)	5.48381e+00(0)	7.88670e+00(3)	1.85916e-01(0)
	Median	3.1881e+00(0)	4.08990e-01(0)	9.95901e-01(0)	9.04867e+00(0)	1.99109e+02(3)	4.16645e-01(0)
	Worst	4.2009e+00(0)	4.50712e-01(0)	1.00046e+00(0)	1.72503e+01(0)	7.6431e+02(0)	3.25477e+00(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,2,3	0,0,0
	\bar{v}	0	0	0	0	3.44809e-02	0
	Mean	3.3242e+00	4.18559e-01	9.84616e-01	1.20831e+01	2.75189e+02	8.61851e-01
	Std.	6.2025e-01	2.74252e-02	2.31103e-02	4.05658e+00	2.2038e+02	1.0949e+00
5×10^4	Best	1.7234e-07(0)	1.62894e-02(0)	6.85799e-01(0)	8.00355e-11(0)	0(0)	5.72982e-11(0)
	Median	4.62717e-07(0)	8.11970e-02(0)	9.18852e-01(0)	8.00355e-11(0)	2.28064e+01(0)	5.72982e-11(0)
	Worst	3.20588e-06(0)	1.53001e-01(0)	9.90945e-01(0)	8.36735e-11(0)	4.05407e+02(0)	5.72982e-11(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	6.93423e-07	7.34131e-02	8.53123e-01	8.33097e-11	1.25584e+02	5.72982e-11
	Std.	0	2.68711e-03	1.20395e-01	1.09139e-12	2.72848e-13	0
5×10^5	Best	0(0)	1.34148e-08(0)	5.85904e-01(0)	8.00355e-11(0)	0(0)	5.72982e-11(0)
	Median	0(0)	9.87368e-08(0)	8.69866e-01(0)	8.00355e-11(0)	0(0)	5.72982e-11(0)
	Worst	0(0)	8.95710e-03(0)	9.43275e-01(0)	8.36735e-11(0)	0(0)	5.72982e-11(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	0	8.95757e-04	7.98692e-01	8.03993e-11	0	5.72982e-11
	Std.	0	8.49740e-03	3.80723e-01	3.45129e-12	0	0

Table 3.3: Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 7-12

FES \ Prob.		g07	g08	g09	g10	g11	g12
5×10^3	Best	2.73353e+01(0)	8.83258e-07(0)	8.69717e+00(0)	2.19811e+03(0)	3.70353e-08(0)	2.36478e-14(0)
	Median	3.01337e+01(0)	8.83258e-07(0)	1.05122e+01(0)	2.76555e+03(0)	8.30096e-02(0)	7.60503e-14(0)
	Worst	5.01731e+01(0)	8.83258e-07(0)	1.26877e+01(0)	3.47093e+03(0)	3.52099e-04(0)	5.11036e-13(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	3.12556e+01	8.83258e-07	1.02144e+01	2.9257e+03	1.95120e-02	1.71940e-13
	Std.	6.72901e+00	5.04968e-17	1.24686e+00	4.08190e+02	2.57633e-02	1.50946e-13
5×10^4	Best	6.64148e-03(0)	8.83258e-07(0)	2.32308e-09(0)	1.87936e+00(0)	0(0)	0(0)
	Median	2.1482e-02(0)	8.83258e-07(0)	2.55977e-09(0)	4.17884e+00(0)	0(0)	0(0)
	Worst	2.89947e-02(0)	8.83258e-07(0)	1.38608e-08(0)	9.33135e+00(0)	0(0)	0(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	1.82096e-02	8.83258e-07	6.13596e-09(0)	3.30128e+00	0	0
	Std.	3.90799e-15	0	4.54747e-14(0)	3.08216e-11	0	0
5×10^5	Best	7.98011e-11(0)	8.83258e-07(0)	9.79981e-11(0)	6.91216e-11(0)	0(0)	0(0)
	Median	7.98011e-11(0)	8.83258e-07(0)	9.81117e-11(0)	6.91216e-11(0)	0(0)	0(0)
	Worst	7.98153e-11(0)	8.83258e-07(0)	9.81117e-11(0)	6.91216e-11(0)	0(0)	0(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	7.98106e-11	8.83258e-07	9.80890e-11	6.91216e-11	0	0
	Std.	1.23581e-14	0	1.43804e-13	0	0	0

Table 3.4: Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 13-18

FES		Prob.	g13	g14	g15	g16	g17	g18
5×10^3	Best		2.95759e-01(3)	2.38212e+00(3)	1.08207e-01(2)	1.38310e-02(0)	2.99626e+01(4)	4.52177e-01(0)
	Median		8.71456e-01(3)	4.41679e+00(3)	2.26109e-01(2)	1.69478e-02(0)	1.50391e+02(4)	6.55249e-01(0)
	Worst		9.40891e-01(3)	7.61513e+00(3)	2.44829e+00(2)	2.77160e-02(0)	3.44834e+02(4)	7.68085e-01(0)
	c		0,3,3	0,1,3	0,0,2	0,0,0	1,4,4	0,0,0
	\bar{v}		6.28157e-02	2.32473e-02	1.16100e-03	0	5.498739e-01	6.34650e-01
	Mean		7.27570e-01	5.48375e+00	1.05094e+00	2.06490e-02	1.28984e+02	9.08582e-02
	Std.		2.05594e-01	1.55581e+00	8.38747e-01	5.08476e-03	8.382406e-01	9.08582e-02
5×10^4	Best		7.37830e-04(0)	2.15395e-03(0)	6.09361e-11(0)	6.52225e-11(0)	1.94414e+01(0)	6.36800e-04(0)
	Median		8.12504e-04(0)	5.76329e-03(0)	3.94230e-01(0)	6.53251e-11(0)	8.66369e+01(0)	1.80242e-03(0)
	Worst		4.05454e-01(0)	1.19311e-02(0)	9.14075e-01(0)	6.53251e-11(0)	1.11386e+02(0)	4.50338e-03(0)
	c		0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}		0	0	0	0	0	0
	Mean		2.01164e-01	5.86988e-02	2.46219e-01	6.52582e-11	8.36554e+02	2.18953e-03
	Std.		1.89624e-01	2.13163e-15	3.34211e-10	0	3.39322e+01	5.55112e-17
5×10^5	Best		8.19243e-05(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	5.79999e-08(0)	1.55613e-11(0)
	Median		8.19243e-04(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	2.99626e-07(0)	1.55614e-11(0)
	Worst		3.80067e-01(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	7.40520e-07(0)	1.55614e-11(0)
	c		0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}		0	0	0	0	0	0
	Mean		1.90443e-01	1.05995e-06	6.09361e-11(0)	6.52149e-11	7.66887e-07	1.55614e-11
	Std.		5.99643e-01	6.7408e-015	0	0	2.23362e-08	1.75542e-16

Table 3.5: Error values achieved by MSDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 19-24

FES		Prob.	g19	g20	g21	g22	g23	g24
5×10^3	Best		8.22433e+01(0)	3.20423e+00(20)	1.33383e+02(5)	1.28277e+02(19)	4.07401e+01(5)	9.62840e-07(0)
	Median		8.22433e+01(0)	3.21427e+00(20)	2.70507e+02(5)	1.54448e+04(19)	1.01929e+02(5)	5.51560e-06(0)
	Worst		2.31858e+02(0)	5.46043e+00(20)	7.90876e+02(5)	1.63401e+04(19)	3.40542e+02(5)	6.05468e-06(0)
	c		0,0,0	2,18,20	0,4,5	17,19,19	0,4,5	0,0,0
	\bar{v}		0	1.75721e+00	6.21370e-02	6.20343e+05	1.78671e-01	0
	Mean		1.36668e+02	4.15232e+00	4.81019e+02	9.60174e+03	1.79501e+02	2.81235e-06
	Std.		3.95853e+01	7.91606e-01	2.00387e+02	5.45609e+03	1.06477e+02	1.79679e-06
5×10^4	Best		3.18893e-01(0)	6.58755e-03(20)	6.70068e-05(0)	4.02714e+03(19)	1.20009e+02(0)	4.65317e-12(0)
	Median		7.43132e-01(0)	1.25717e-02(20)	1.29576e-02(0)	6.48001e+03(19)	2.80670e+02(0)	4.65317e-12(0)
	Worst		1.05122e+00(0)	5.02642e-02(20)	1.30982e+02(0)	1.94930e+04(19)	4.46977e+02(0)	4.65317e-12(0)
	c		0,0,0	0,6,20	0,0,0	17,19,19	0,0,0	0,0,0
	\bar{v}		0	2.15111e-02	0	4.08852e+04	0	0
	Mean		5.66698e-01	2.79470e-02	9.03673e+01	9.70419e+03	2.79155e+02	4.65317e-12
	Std.		1.1099e-14	7.76073e-07	6.41660e+01	4.27823e+03	8.33960e-11	0
5×10^5	Best		4.63345e-11(0)	4.76843e-06(1)	3.48166e-11(0)	6.26246e+03(6)	3.97904e-13(0)	4.65317e-12(0)
	Median		4.63345e-11(0)	7.48033e-06(1)	3.48166e-11(0)	1.06714e+04(6)	3.97904e-13(0)	4.65317e-12(0)
	Worst		4.63771e-11(0)	7.48033e-06(1)	1.30978e+02(0)	1.91762e+04(6)	2.71882e-10(0)	4.65317e-12(0)
	c		0,0,0	0,0,1	0,0,0	3,6,6	0,0,0	0,0,0
	\bar{v}		0	7.18618e-03	0	2.17228e+00	0	0
	Mean		4.63459e-11	5.33570e-06	7.85870e+01	1.41397e+04	3.73404e-11	4.65317e-12
	Std.		3.50982e-14	2.45416e-06	2.02911e+02	1.35290e+04	2.63721e-10	0

Table 3.6: Number of NFEs to achieve the fixed accuracy level ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$), Success Rate, Feasibility Rate and Success Performance by MSDE.

Prob.	Best	Median	Worst	Mean	Std.	Feasible Rate	Success Rate	Success Performance
g01	31800	35425	36250	34970	1253	100%	100%	34970
g02	105400	119675	137500	120625	10674	100%	100%	120625
g03	--	--	--	--	--	100%	0%	--
g04	16050	17075	18550	17195	761	100%	100%	17195
g05	16700	117650	218200	122090	55788	100%	100%	122090
g06	7650	8150	9000	8235	361	100%	100%	8235
g07	85200	98550	108250	98070	6540	100%	100%	98070
g08	450	1325	1600	1235	296	100%	100%	1235
g09	24750	28300	29700	27830	1512	100%	100%	27830
g10	120450	127775	135200	128095	4372	100%	100%	128095
g11	3450	15950	30600	15140	8443	100%	100%	15140
g12	750	1000	1250	1010	149	100%	100%	1010
g13	32350	44600	56550	44525	9695	100%	88%	50596
g14	68550	70400	77650	71350	2531	100%	100%	71350
g15	14100	76225	111250	67750	26873	100%	100%	67750
g16	12050	13200	14550	13320	618	100%	100%	13320
g17	204350	228200	299900	240163	36796	100%	88%	272912
g18	73100	82850	101200	84065	8166	100%	100%	84065
g19	152600	168500	181800	167925	9746	100%	100%	167925
g20	--	--	--	--	--	0%	0%	--
g21	46950	49350	62350	51016	5230	100%	72%	70856
g22	--	--	--	--	--	0%	0%	--
g23	198200	228075	271600	231385	22864	100%	60%	385641
g24	3050	3700	4100	3650	264	100%	100%	3650

Table 3.7: Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 1-6.

FES \ Prob.		g01	g02	g03	g04	g0 5	g 06
5×10^3	Best	1.66651e+00(0)	3.38672e-01(0)	5.25821e-01(0)	1.01423e+00(0)	1.56252e+00(3)	5.87306e-04(0)
	Median	2.66785e+00(0)	3.48828e-01(0)	6.87642e-01(0)	1.22702e+00(0)	2.41341e+00(3)	8.93389e-04(0)
	Worst	4.1648e+00(0)	4.53219e-01(0)	8.70172e-01(0)	5.28069e+00(0)	4.77922e+00(3)	2.24387e-02(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,2,3	0,0,0
	\bar{v}	0	0	0	0	9.83244e-02	0
	Mean	3.01379e+00	3.89322e-01	0.728744	3.1947e+00	2.60157e+00	7.71430e-03
	Std.	7.25838e-01	3.15807e-02	0.107681	1.45584e+00	8.48998e-01	6.66068e-03
5×10^4	Best	2.49437e-09(0)	2.68458e-02(0)	7.06243e-02(0)	3.31784e-09(0)	7.10043e-09(0)	1.42791e-10(0)
	Median	1.09052e-08(0)	7.51118e-02(0)	8.72004e-02(0)	3.31784e-09(0)	7.10043e-09(0)	1.42791e-10(0)
	Worst	3.77633e-08(0)	1.08780e-01(0)	9.11032e-02(0)	3.32147e-09(0)	7.10043e-09(0)	1.42791e-10(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	1.46368e-08	6.67891e-02	2.35011e-02	3.32002e-09	7.10043e-09	1.42791e-10
	Std.	0	8.14764e-03	2.89565e-04	1.09139e-12	0	0
5×10^5	Best	0(0)	1.12504e-08(0)	3.18147e-02(0)	8.00355e-11 (0)	0(0)	1.42791e-10(0)
	Median	0(0)	2.86178e-08(0)	3.31784e-02(0)	8.00355e-11 (0)	0(0)	1.42791e-10(0)
	Worst	0(0)	6.28391e-08(0)	3.83409e-02(0)	3.32147e-09(0)	0(0)	1.42791e-10(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	0	3.02811e-08	3.96721e-02	3.32111e-09	0	1.42791e-10
	Std.	0	6.65799e-08	9.00283e-06	3.45129e-12	0	0

Table 3.8: Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 7-12

FES \ Prob.		g07	g08	g09	g10	g11	g12
5×10^3	Best	1.62644e+01(0)	8.83258e-07(0)	4.09591e+00(0)	1.93349e+03(0)	1.50394e-07(0)	9.21485e-15(0)
	Median	2.36368e+01(0)	8.83258e-07(0)	5.8835e+00(0)	2.82773e+03(0)	1.67374e-05(0)	4.39648e-14(0)
	Worst	2.93178e+01(0)	8.83258e-07(0)	1.09018e+01(0)	3.60685e+03(0)	8.10640e-02(0)	2.61569e-13(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	2.31701e+01	8.83258e-07	7.34266e+00	2.61855e+03	9.52244e-03	7.61502e-14
	Std.	4.30983e+00	1.30923e-17	2.28494e+00	5.01995e+02	2.41154e-02	6.85416e-14
5×10^4	Best	9.55995e-03(0)	8.83258e-07(0)	9.81117e-11(0)	6.69388e-10(0)	0(0)	0(0)
	Median	1.41390e-02(0)	8.83258e-07(0)	7.36236e-10(0)	6.55146e-01(0)	0(0)	0(0)
	Worst	2.33647e-02(0)	8.83258e-07(0)	3.1464e-09(0)	1.94167e+00(0)	0(0)	0(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	1.35753e-02	8.83258e-07	9.81004e-11	0.915356	0	0
	Std.	2.13163e-15	1.30923e-17	1.07853e-13	3.63798e-13	0	0
5×10^5	Best	7.98117e-11(0)	8.83258e-07(0)	9.79981e-11(0)	6.91216e-11(0)	0(0)	0(0)
	Median	7.98117e-11(0)	8.83258e-07(0)	9.79981e-11(0)	6.91216e-11(0)	0(0)	0(0)
	Worst	7.98117e-11(0)	8.83258e-07(0)	9.81117e-11(0)	6.91216e-11(0)	0(0)	0(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	7.98117e-11	8.83258e-07	9.80057e-11	6.91216e-11	0	0
	Std.	6.7408e-15	1.30923e-17	3.41061e-14	1.15043e-12	0	0

Table 3.9: Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 13-18

FES \ Prob.		g13	g14	g15	g16	g17	g18
5×10^3	Best	9.62527e-03(3)	1.53957e+01(3)	1.65314e-04(1)	4.58283e-03(0)	2.33195e+01(4)	2.95209e-01(0)
	Median	6.46529e-02(2)	1.80240e+01(3)	1.22720e-03(1)	5.54017e-03(0)	7.43504e+01(4)	3.94191e-01(0)
	Worst	6.08188e-01(3)	4.31482e+01(3)	7.70481e-01(2)	1.15618e-02(0)	2.25439e+02(4)	5.27015e-01(0)
	c	0,2,2	0,3,3	0,0,1	0,0,0	1,4,4	0,0,0
	\bar{v}	5.86143e-02	5.45758e-01	5.32557e-04	0	1.0855e+00	0
	Mean	2.39419e-01	2.38573e+01	8.18152e-02	8.28083e-03	8.90232e+01	4.28404e-01
	Std.	1.98708e-01	8.85313e+00	2.29628e-01	2.27290e-03	5.74606e+01	6.78757e-02
5×10^4	Best	8.19243e-04(0)	1.07811e-03(0)	9.96090e-09(0)	8.53478e-09(0)	5.79999e-03(0)	5.38834e-04(0)
	Median	3.80067e-01(0)	1.25419e-03(0)	9.96101e-09(0)	8.53479e-09(0)	2.55867e+00(0)	6.15036e-04(0)
	Worst	3.80067e-01(0)	9.02366e-03(0)	1.13231e-08(0)	8.53479e-09(0)	9.13463e+01(0)	2.79150e-03(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	3.04217e-01	2.70743e-03(0)	1.00972e-08	8.53478e-09	1.35020e+01	1.38421e-03
	Std.	1.51699e-01	0	3.41061e-14	5.99520e-16	3.70231e+01	5.08768e-17
5×10^5	Best	8.19243e-05(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	5.79999e-08(0)	1.55614e-11(0)
	Median	3.80067e-01(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	5.79999e-03(0)	1.55614e-11(0)
	Worst	3.80067e-01(0)	1.05995e-06(0)	6.09361e-11(0)	6.52149e-11(0)	7.40521e+01(0)	1.55614e-11(0)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	\bar{v}	0	0	0	0	0	0
	Mean	3.04217e-01	1.05995e-06	6.09361e-11	6.52149e-11	3.70289e-01	1.55614e-11
	Std.	4.79715e-01	0	1.07853e-13	1.89585e-15	1.17077e+01	1.60887e-16

Table 3.10: Error values achieved by SDE when NFEs = 5×10^3 , NFEs = 5×10^4 , NFEs = 5×10^5 for problems 19-24

FES \ Prob.		g19	g20	g21	g22	g23	g24
5×10^3	Best	6.60169e+01(0)	1.01651e+00(20)	7.73472e+00(5)	2.00461e+02(18)	4.21079e+02(5)	3.57405e-08(0)
	Median	7.88958e+01(0)	2.24645e+00(20)	3.22758e+01(5)	2.02665e+02(18)	6.15288e+02(5)	6.21463e-08(0)
	Worst	1.28065e+02(0)	2.52606e+00(20)	1.1595e+02(5)	2.33617e+02(18)	6.58727e+02(5)	1.1646e-07(0)
	c	0,0,0	2,18,20	1,5,5	18,18,18	2,5,5	0,0,0
	\bar{v}	0	1.52951e+00	6.15919e-01	1.58915e+06	7.11753e-01	0
	Mean	8.94169e+01	1.81900e+00	6.62669e+01	2.20016e+02	7.53218e+02	7.56426e-08
	Std.	2.03685e+01	4.42047e-01	3.93103e+01	1.25180e+01	6.78559e+00	2.58786e-08
5×10^4	Best	2.16857e-01(0)	5.47738e-02(19)	7.46949e-06(0)	2.31307e+02(18)	1.79542e+01(2)	4.65317e-12(0)
	Median	4.23096e-01(0)	5.55880e-02(20)	7.46949e-06(0)	2.34730e+02(18)	2.12415e+02(2)	4.65317e-12(0)
	Worst	5.37331e-01(0)	7.62731e-02(19)	3.07502e-05(0)	2.36323e+02(18)	3.28436e+02(2)	4.65317e-12(0)
	c	0,0,0	0,4,20	0,0,0	16,18,18	0,2,2	0,0,0
	\bar{v}	0	2.00149e-02	0	5.99771e+04	7.08644e-02	0
	Mean	3.37506e-01	6.51989e-02	7.48647e-06	2.34109e+02	1.82419e-02	4.65317e-12
	Std.	3.25612e-15	9.65038e-03	5.55880e-02	1.2796e-11	9.33119e+01	0
5×10^5	Best	4.63274e-11(0)	2.78755e-02(1)	3.48166e-11(0)	2.36431e+02(9)	3.97904e-13(0)	4.65317e-12(0)
	Median	4.63274e-11(0)	5.33038e-02(1)	3.48166e-11(0)	2.36431e+02(9)	2.48538e+02(2)	4.65317e-12(0)
	Worst	4.63274e-11(0)	6.19949e-02(1)	3.48166e-11(0)	2.36431e+02(9)	3.27808e+02(2)	4.65317e-12(0)
	c	0,0,0	0,1,1	0,0,0	2,6,9	0,2,2	0,0,0
	\bar{v}	0	1.51715e-02	0	7.60051e+01	8.28466e-02	0
	Mean	4.63274e-11	4.43923e-02	3.48166e-11	2.36431e+02	1.80106e+02	4.65317e-12
	Std.	1.02967e-14	3.05172e-02	1.81822e-14	4.04645e-11	2.95078e+02	0

Table 3.11: Number of FES to achieve the fixed accuracy level ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$), Success Rate, Feasibility Rate and Success Performance by SDE.

Prob.	Best	Median	Worst	Mean	Std.	Feasible Rate	Success Rate	Success Performance
g01	28300	29225	32600	29750	1340	100%	100%	29750
g02	104600	119350	133200	120943	8594	100%	80%	151178
g03	--	--	--	--	--	100%	0%	--
g04	13100	13825	15100	13910	545	100%	100%	13910
g05	10100	14250	40650	18280	8970	100%	100%	18280
g06	5500	6125	6500	6105	278	100%	100%	6105
g07	82650	91450	101500	90810	5223	100%	100%	90810
g08	1050	1250	1700	1325	199	100%	100%	1325
g09	24800	25700	28600	26105	1128	100%	100%	26105
g10	99550	109850	117650	109865	5604	100%	100%	109865
g11	3000	6125	16650	7662	4365	100%	100%	7662
g12	1000	1100	1250	1100	83	100%	100%	1100
g13	27000	32600	35700	31766	3600	100%	84%	37816
g14	63050	68025	73250	68575	3448	100%	100%	68575
g15	5850	11325	28000	13312	7992	100%	100%	13312
g16	10150	10950	11700	10945	480	100%	100%	10945
g17	208989	211342	213456	209765	4578	100%	24%	874020
g18	59300	67975	83400	69205	7093	100%	100%	69205
g19	143450	153325	169000	154915	8005	100%	100%	154915
g20	--	--	--	--	--	0%	0%	--
g21	38100	42400	44600	41430	2339	100%	60%	69050
g22	--	--	--	--	--	0%	0%	--
g23	111900	117700	133050	120883	8923	80%	56%	215862
g24	2450	2725	3050	2760	204	100%	100%	2760

Table 3.12: Comparison of MSDE and SDE on the basis of error and standard deviation.

Prob.	Best		Worst		Mean		Std.	
	SDE	MSDE	SDE	MSDE	SDE	MSDE	SDE	MSDE
g01	0	0	0	0	0	0	0	0
g02	1.12504e-08	1.34148e-08	6.28391e-08	8.95710e-03	3.02811e-08	8.95757e-04	6.65799e-08	8.49740e-03
g03	3.18147e-02	5.85904e-01	3.83409e-02	9.43275e-01	3.96721e-02	7.98692e-01	9.00283e-06	3.80723e-01
g04	8.00355e-11	8.00355e-11	3.32147e-09	8.36735e-11	3.32111e-09	8.03993e-11	3.45129e-12	3.45129e-12
g05	0	0	0	0	0	0	0	0
g06	1.42791e-10	5.72982e-11	1.42791e-10	5.72982e-11	1.42791e-10	5.72982e-11	0	0
g07	7.98117e-11	7.98011e-11	7.98117e-11	7.98153e-11	7.98117e-11	7.98106e-11	6.74080e-15	1.23581e-14
g08	8.83258e-07	8.83258e-07	8.83258e-07	8.83258e-07	8.83258e-07	8.83258e-07	1.30923e-17	0
g09	9.79981e-11	9.79981e-11	9.81117e-11	9.81117e-11	9.80057e-11	9.80890e-11	3.41061e-14	1.43804e-13
g10	6.91216e-11	6.91216e-11	6.91216e-11	6.91216e-11	6.91216e-11	6.91216e-11	1.15043e-12	0
g11	0	0	0	0	0	0	0	0
g12	0	0	0	0	0	0	0	0
g13	8.19243e-05	8.19243e-05	3.80067e-01	3.80067e-01	3.04217e-01	1.90443e-01	4.79715e-01	5.99643e-01
g14	1.05995e-06	1.05995e-06	1.05995e-06	1.05995e-06	1.05995e-06	1.05995e-06	0	6.7408e-015
g15	6.09361e-11	6.09361e-11	6.09361e-11	6.09361e-11	6.09361e-11	6.09361e-11	1.07853e-13	0
g16	6.52149e-11	6.52149e-11	6.52149e-11	6.52149e-11	6.52149e-11	6.52149e-11	1.89585e-15	0
g17	5.79999e-08	5.79999e-08	7.40521e+01	7.40520e-07	3.70289e-01	7.66887e-07	1.17070e+01	2.23362e-08
g18	1.55614e-11	1.55613e-11	1.55614e-11	1.55613e-11	1.55614e-11	1.55613e-11	1.60887e-16	1.75542e-16
g19	4.63274e-11	4.63345e-11	4.63274e-11	4.63771e-11	4.63274e-11	4.63459e-11	1.02967e-14	3.50982e-14
g20	2.78755e-02	4.76843e-06	6.19949e-02	7.48033e-06	4.43923e-02	5.33570e-06	3.05172e-02	2.45416e-06
g21	3.48166e-11	3.48166e-11	3.48166e-11	1.30978e+02	3.48166e-11	7.85870e+01	1.81822e-14	2.02910e+02
g22	2.36431e+02	6.26246e+03	2.36431e+02	1.91762e+04	2.36431e+02	1.41390e+04	4.04645e-11	1.35290e+04
g23	3.97904e-13	3.97904e-13	3.27808e+02	2.71882e-10	1.80106e+02	3.73404e-11	2.9507e+02	2.63721e-10
g24	4.65317e-12	4.65317e-12	4.65317e-12	4.65317e-12	4.65317e-12	4.65317e-12	0	0

Table 3.13: Comparison of MSDE and SDE on the basis of Number of NFEs to achieve the fixed accuracy level ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$).

Prob.	Best		Worst		Mean		Std.		Success Performance	
	SDE	MSDE	SDE	MSDE	SDE	MSDE	SDE	MSDE	SDE	MSDE
g01	28300	31800	32600	36250	29750	34970	1340	1253	29750	34970
g02	104600	105400	133200	137500	120943	120625	8594	10674	151178	120625
g03	--	--	--	--	--	--	--	--	--	--
g04	13100	16050	15100	18550	13910	17195	545	761	13910	17195
g05	10100	16700	40650	218200	18280	122090	8970	55788	18280	122090
g06	5500	7650	6500	9000	6105	8235	278	361	6105	8235
g07	82650	85200	101500	108250	90810	98070	5223	6540	90810	98070
g08	1050	450	1700	1600	1325	1235	199	296	1325	1235
g09	24800	24750	28600	29700	26105	27830	1128	1512	26105	27830
g10	99550	120450	117650	135200	109865	128095	5604	4372	109865	128095
g11	3000	3450	16650	30600	7662	15140	4365	8443	7662	15140
g12	1000	750	1250	1250	1100	1010	83	149	1100	1010
g13	27000	32350	35700	56550	31766	44525	3600	9695	37816	50596
g14	63050	68550	73250	77650	68575	71350	3448	2531	68575	71350
g15	5850	14100	28000	111250	13312	67750	7992	26873	13312	67750
g16	10150	12050	11700	14550	10945	13320	480	618	10945	13320
g17	208989	204350	213456	299900	209765	240163	4578	36796	874020	272912
g18	59300	73100	83400	101200	69205	84065	7093	8166	69205	84065
g19	143450	152600	169000	181800	154915	167925	8005	9746	154915	167925
g20	--	--	--	--	--	--	--	--	--	--
g21	38100	46950	44600	62350	41430	51016	2339	5230	69050	70856
g22	--	--	--	--	--	--	--	--	--	--
g23	111900	198200	133050	271600	120883	231385	8923	22864	215862	385641
g24	2450	3050	3050	4100	2760	3650	204	264	2760	3650

Table 3.14: Number of NFEs to achieve the fixed accuracy level $((f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001)$, Success Rate, Feasibility Rate and Success Performance by all the algorithms for problems 01-12.

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g01	MSDE	31800	36250	34970	100	100	34970
	SDE	28300	32600	29750	100	100	29750
	ZRDE	30511	38028	33414	100	100	33414
	jDE-2	46559	56968	50386	100	100	50386
	ϵ DE	57122	61712	59308	100	100	59308
	MDE	37633	49654	43430	100	100	43430
	MDE1	63300	90900	75373	100	100	75373
SaDE	25115	25115	25115	100	100	25115	
g02	MSDE	105400	137500	120625	100	100	120625
	SDE	104600	133200	120943	100	80	151178
	ZRDE	95501	129363	113298	100	84	134879
	jDE-2	101201	173964	123490	100	92	145899
	ϵ DE	126152	175206	149825	100	100	149825
	MDE	260388	301048	280573	100	92	304970
	MDE1	53250	245550	96222	100	16	96250
SaDE	76915	-	188990	100	84	183850	
g03	MSDE	--	--	--	100	0	--
	SDE	-	-	-	100	0	-
	ZRDE	-	-	-	100	0	-
	jDE-2	-	-	-	100	0	-
	ϵ DE	86748	91328	89407	100	100	89407
	MDE	119629	277057	109298	100	84	24860
	MDE1	36000	61350	44988	100	100	44988
SaDE	30000	-	243520	100	96	298960	
g04	MSDE	16050	18550	17195	100	100	17195
	SDE	13100	15100	13910	100	100	13910
	ZRDE	14048	18362	15986	100	100	15986
	jDE-2	38288	42880	40728	100	100	40728
	ϵ DE	24800	28206	26216	100	100	26216
	MDE	19717	22609	20883	100	100	20883
	MDE1	33900	61950	41562	100	100	41562
SaDE	25107	25113	25107	100	100	25107	
g05	MSDE	16700	218200	122090	100	100	122090
	SDE	10100	40650	18280	100	100	18280
	ZRDE	16994	204151	107076	100	100	107076
	jDE-2	133340	482304	206620	100	68	446839
	ϵ DE	96812	98589	97431	100	100	97431
	MDE	39701	477209	216469	100	100	216469
	MDE1	19350	24000	21306	100	100	21306
SaDE	35500	152000	74340	100	100	73000	
g06	MSDE	7650	9000	8235	100	100	8235
	SDE	5500	6500	6105	100	100	6105
	ZRDE	6147	7995	7143	100	100	7143
	jDE-2	26830	31299	29488	100	100	29488
	ϵ DE	6499	8382	7381	100	100	7381
	MDE	9872	11528	10574	100	100	10574
	MDE1	4650	5250	5202	100	100	5202

	SaDE	12546	18347	14394	100	100	12546
g07	MSDE	85200	108250	98070	100	100	98070
	SDE	82650	101500	90810	100	100	90810
	ZRDE	84889	104026	93793	100	100	93793
	jDE-2	114899	141847	127740	100	100	127740
	ϵ DE	69506	78963	74303	100	100	74303
	MDE	52438	63445	57400	100	100	57400
	MDE1	124650	380400	194202	100	100	194202
	SaDE	25195	422860	143090	100	100	27637
g08	MSDE	450	1600	1235	100	100	1235
	SDE	1050	1700	1325	100	100	1325
	ZRDE	831	1337	1086	100	100	1086
	jDE-2	1567	4485	3236	100	100	3236
	ϵ DE	327	1334	1139	100	100	1139
	MDE	990	2068	1515	100	100	1514
	MDE1	900	1350	918	100	100	918
	SaDE	782	1775	1268	100	100	1323
g09	MSDE	24750	29700	27830	100	100	27830
	SDE	24800	28600	26105	100	100	26105
	ZRDE	23828	27424	25805	100	100	25805
	jDE-2	49118	58230	54919	100	100	54919
	ϵ DE	19530	24790	23121	100	100	23121
	MDE	18608	24025	21044	100	100	21044
	MDE1	14850	19200	16152	100	100	16152
	SaDE	12960	33166	18560	100	100	21446
g10	MSDE	120450	135200	128095	100	100	128095
	SDE	99550	117650	109865	100	100	109865
	ZRDE	105673	132270	119217	100	100	119217
	jDE-2	139095	165498	146150	100	100	146150
	ϵ DE	93743	122387	105234	100	100	105234
	MDE	42610	60924	48628	100	100	48628
	MDE1	152400	179850	164160	100	100	164160
	SaDE	26000	15300	58760	100	100	44167
g11	MSDE	3450	30600	15140	100	100	15140
	SDE	3000	16650	7662	100	100	7662
	ZRDE	1384	24356	13380	100	100	13380
	jDE-2	17834	432169	49700	100	96	53928
	ϵ DE	5407	29510	16420	100	100	16420
	MDE	3884	55738	22422	100	96	23356
	MDE1	1200	4950	3000	100	100	3000
	SaDE	12643	25120	23353	100	100	25111
g12	MSDE	750	1250	1010	100	100	1010
	SDE	1000	1250	1100	100	100	1100
	ZRDE	342	7307	5104	100	100	5104
	jDE-2	1820	9693	6355	100	100	6356
	ϵ DE	1645	5540	4124	100	100	4124
	MDE	1044	6510	4238	100	100	4238
	MDE1	1200	1650	1308	100	100	1308
	SaDE	463	2576	1611	100	100	2576

Table 3.15: Number of NFEs to achieve the fixed accuracy level ($(f(\bar{x}) - f(\bar{x}^*)) \leq 0.0001$), Success Rate, Feasibility Rate and Success Performance by all the algorithms for problems 13-24.

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g13	MSDE	32350	56550	44525	100	88	50596
	SDE	27000	35700	31766	100	84	37816
	ZRDE	242289	288226	265703	100	32	830322
	jDE-2	-	-	-	100	0	-
	ϵ DE	8287	68608	34738	100	100	34738
	MDE	268723	429252	356433	88	48	742568
	MDE1	19500	24450	21732	100	100	21732
	SaDE	25161	126080	42372	100	100	25168
g14	MSDE	68550	77650	71350	100	100	
	SDE	63050	73250	68575	100	100	68575
	ZRDE	57727	81392	68226	100	100	68226
	jDE-2	88954	107951	97845	100	100	97845
	ϵ DE	106816	121656	113439	100	100	113439
	MDE	34099	58180	42715	100	100	42715
	MDE1	208236	408036	291642	100	100	291642
	SaDE	32000	-	154320	100	80	45000
g15	MSDE	14100	111250	67750	100	100	67750
	SDE	5850	28000	13312	100	100	13312
	ZRDE	7151	137487	57968	100	100	57968
	jDE-2	51321	432766	222460	100	96	241383
	ϵ DE	57729	90593	84216	100	100	84216
	MDE	12240	490328	200174	100	100	200174
	MDE1	9750	11850	10458	100	100	10458
	SaDE	25500	97000	45240	100	100	27000
g16	MSDE	12050	14550	13320	100	100	13320
	SDE	10150	11700	10945	100	100	10945
	ZRDE	9837	12619	11592	100	100	11592
	jDE-2	28230	34182	31695	100	100	31695
	ϵ DE	12347	13923	12986	100	100	12986
	MDE	11036	14418	13063	100	100	13063
	MDE1	7950	9450	8730	100	100	8730
	SaDE	13144	15797	14545	100	100	14948
g17	MSDE	204350	299900	240163	100	88	272912
	SDE	208989	213456	209765	100	24	874020
	ZRDE	201798	328448	265692	100	20	1328459
	jDE-2	449306	449306	179710	100	4	11232650
	ϵ DE	97274	100144	98861	100	100	98861
	MDE	117917	270372	204791	96	28	731396
	MDE1	20400	34950	26364	100	100	26364
	SaDE	443000	-	497720	100	4	12500000
g18	MSDE	73100	101200	84065	100	100	84065
	SDE	59300	83400	69205	100	100	69205
	ZRDE	70290	96334	79557	100	100	79557
	jDE-2	91049	142674	104460	100	100	104462
	ϵ DE	51035	72112	59153	100	100	59153
	MDE	36211	57603	44045	100	100	44045
	MDE1	54000	133800	103482	100	100	103482

	SaDE	26000	-	65400	100	92	28261
g19	MSDE	152600	181800	167925	100	100	167925
	SDE	143450	169000	154915	100	100	154915
	ZRDE	150864	198377	177229	100	100	177229
	jDE-2	170950	234038	199850	100	100	199850
	ϵ DE	319636	451685	356350	100	100	356350
	MDE	102385	146392	118274	100	100	118274
	MDE1	--	--	--	100	0	--
	SaDE	25531	78048	48733	100	100	52165
g20	MSDE	--	--	--	0	0	--
	SDE	--	--	--	0	0	--
	ZRDE	--	--	--	0	0	--
	jDE-2	--	--	--	4	0	--
	ϵ DE	--	--	--	0	0	--
	MDE	--	--	--	0	0	--
	MDE1	--	--	--	0	0	--
	SaDE	--	--	--	0	0	--
g21	MSDE	46950	62350	51016	100	72	70856
	SDE	38100	44600	41430	100	60	69050
	ZRDE	41559	213297	97614	100	60	162691
	jDE-2	96552	147030	107080	100	92	126507
	ϵ DE	126194	216905	135143	100	100	135143
	MDE	42542	303029	142159	100	68	209057
	MDE1	82350	201150	112566	100	100	112566
	SaDE	98500	--	327660	100	60	164170
g22	MSDE	--	--	--	0	0	--
	SDE	--	--	--	0	0	--
	ZRDE	--	--	--	0	0	--
	jDE-2	--	--	--	0	0	--
	ϵ DE	--	--	--	100	0	--
	MDE	--	--	--	0	0	--
	MDE1	--	--	--	0	0	--
	SaDE	--	--	--	100	0	--
g23	MSDE	198200	271600	231385	100	60	385641
	SDE	111900	133050	120883	80	56	215862
	ZRDE	--	--	--	100	0	--
	jDE-2	205404	495721	302550	100	92	357452
	ϵ DE	158742	281071	200765	100	100	200765
	MDE	109493	293966	210661	100	100	210661
	MDE1	247500	476250	360420	100	100	360420
	SaDE	82500	--	294540	100	88	129550
g24	MSDE	3050	4100	3650	100	100	3650
	SDE	2450	3050	2760	100	100	2760
	ZRDE	2514	3356	3024	100	100	3024
	jDE-2	7587	11550	10196	100	100	10196
	ϵ DE	2661	3474	2952	100	100	2952
	MDE	3623	5099	4342	100	100	4342
	MDE1	1650	1950	1794	100	100	1794
	SaDE	4280	5657	4847	100	100	4624

Table 3.16: Results of Friedman's test based on NFEs.

N	Friedman value	df	p-value
24	30.747	7	<0.001

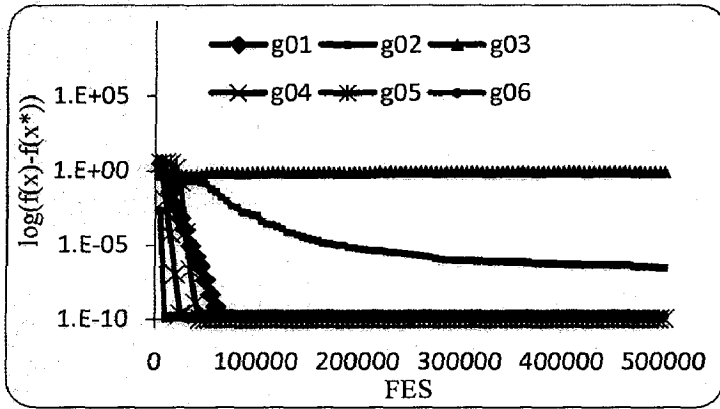
df – Degrees of freedom N - Total No of functions

Table 3.17: Ranking obtained through Friedman's test and Critical Difference (CD) calculated through Bonnferroni-dunn's procedure.

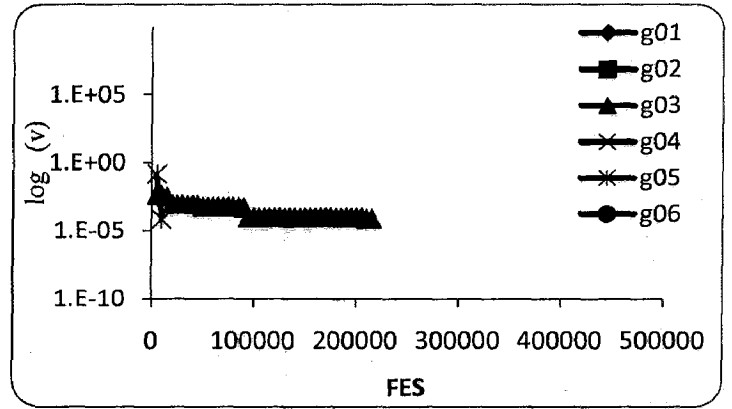
Algorithm	Mean Rank
MSDE	4.60
SDE	3.06
ZRDE	4.31
jDE	6.56
eDE	4.21
MDE	4.60
MDE1	3.83
SaDE	4.81
CD for $\alpha = 0.05$	1.902117
CD for $\alpha = 0.10$	1.732412

Table 3.18: Results of pairwise comparison based on NFEs.

SDE Vs.	Wilcoxon test								
	+ve rank	-ve rank	tie	Mean of +ve rank	Mean of -ve rank	Sum of +ve rank	Sum of -ve rank	Stat.	p-value
MSDE	18	3	3	12.5	2	225	6	-3.806	<0.001
ZRDE	17	4	3	12.41	5	211	20	-3.319	0.001
jDE	20	1	3	11	11	220	11	-3632	0.000
eDE	15	7	2	11.6	11.29	174	79	-1.542	0.123
MDE	14	8	2	10.93	12.50	153	100	-0.860	0.390
MDE1	11	11	2	14.09	8.91	155	98	-0.925	0.355
SaDE	15	7	2	12.20	10	183	70	-1.834	0.067

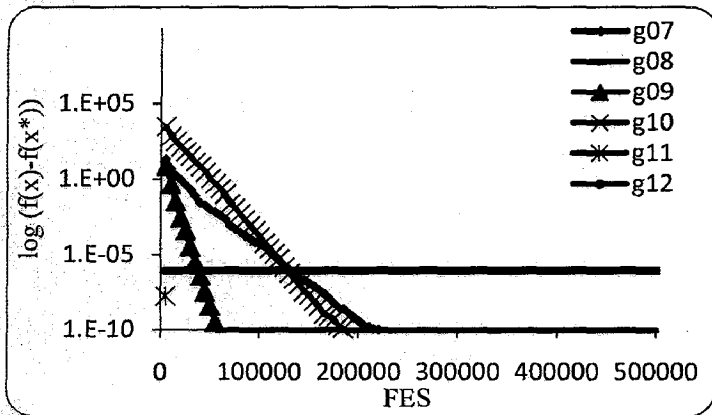


(a). Function error values.

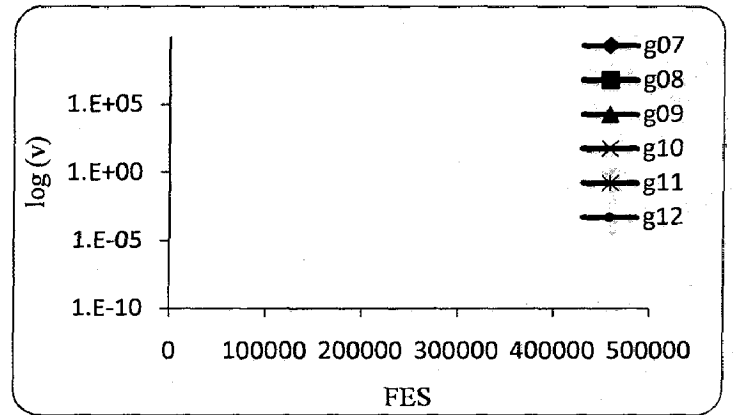


(b). Mean constraint violation.

Figure 3.1: Convergence graph for problems 01-06 of SDE.

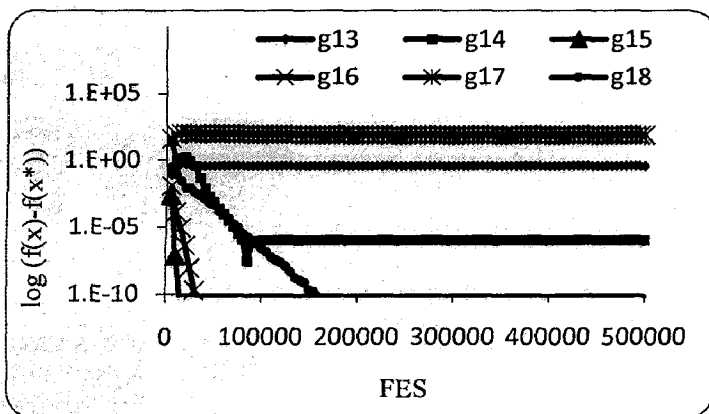


(a). Function error values.

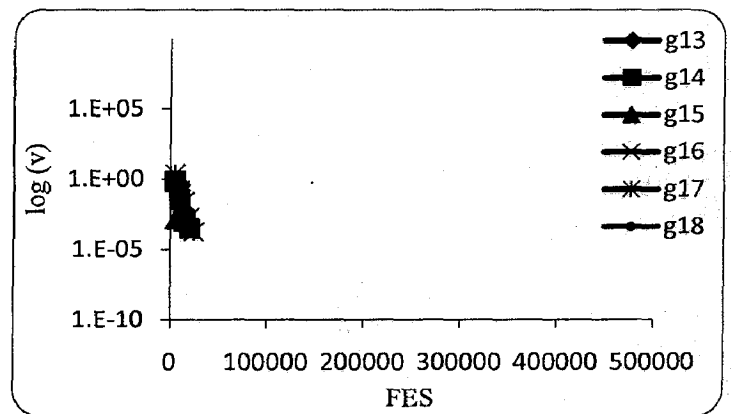


(b). Mean constraint violation.

Figure 3.2: Convergence graph for problems 07-12 of SDE.

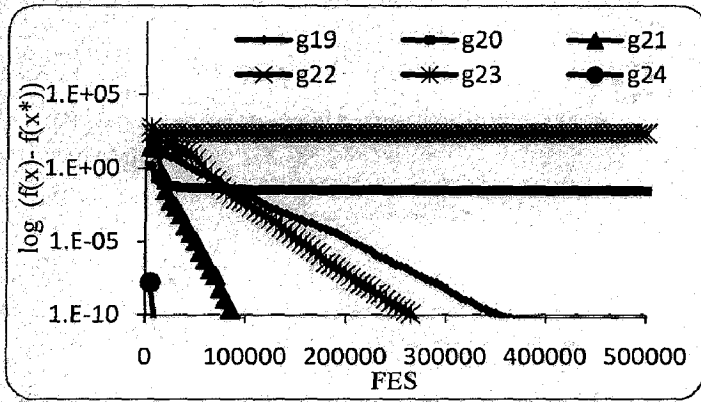


(a). Function error values.

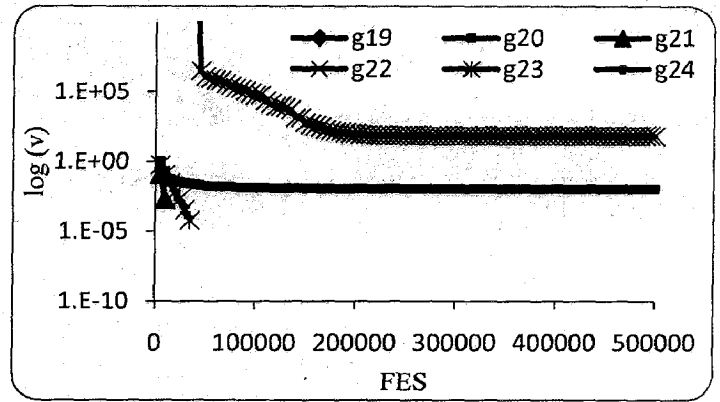


(b). Mean constraint violation.

Figure 3.3: Convergence graph for problems 13-18 of SDE.

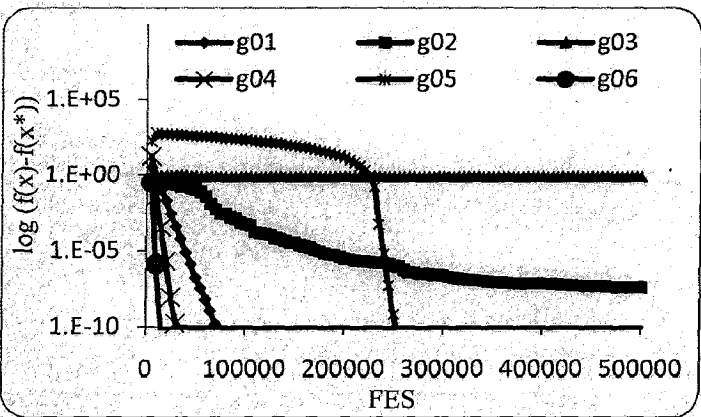


(a). Function error values.

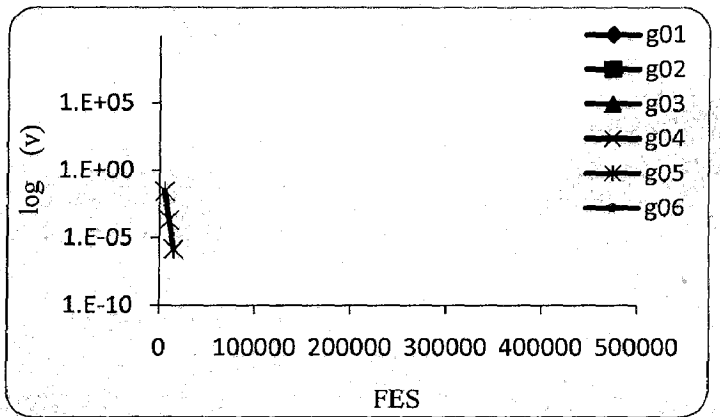


(b). Mean constraint violation.

Figure 3.4: Convergence graph for problems 19-24 of SDE.

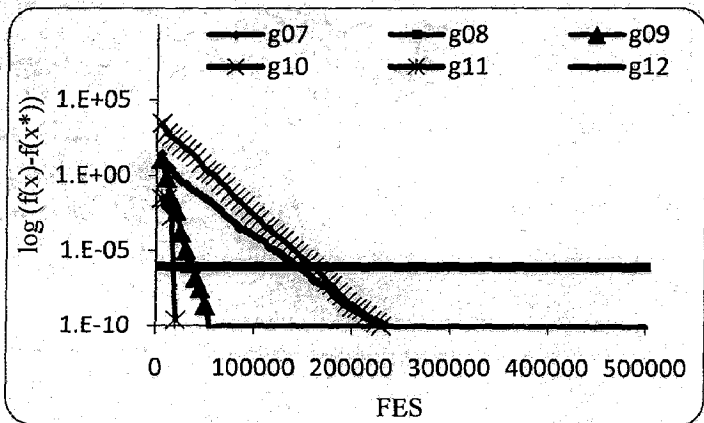


(a). Function error values.

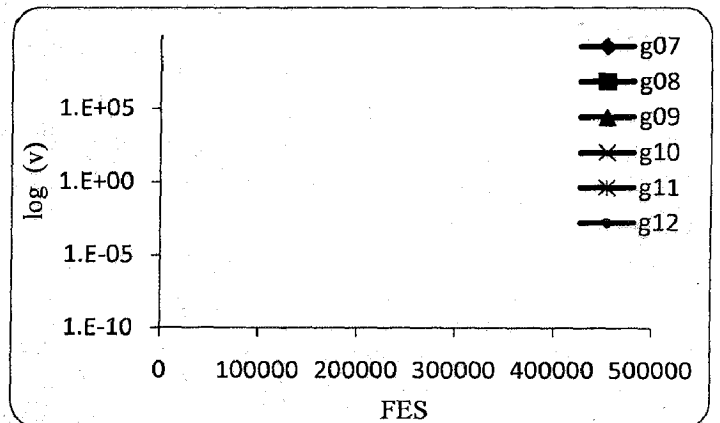


(b). Mean constraint violation.

Figure 3.5: Convergence graph for problems 01-06 of MSDE.

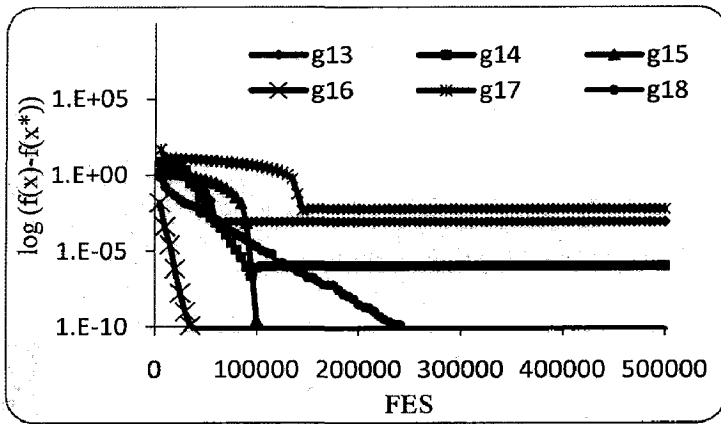


(a). Function error values.

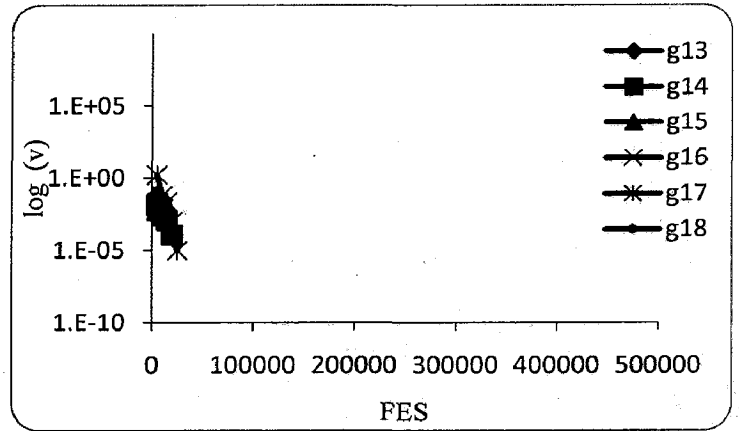


(b). Mean constraint violation.

Figure 3.6: Convergence graph for problems 07-12 of MSDE.

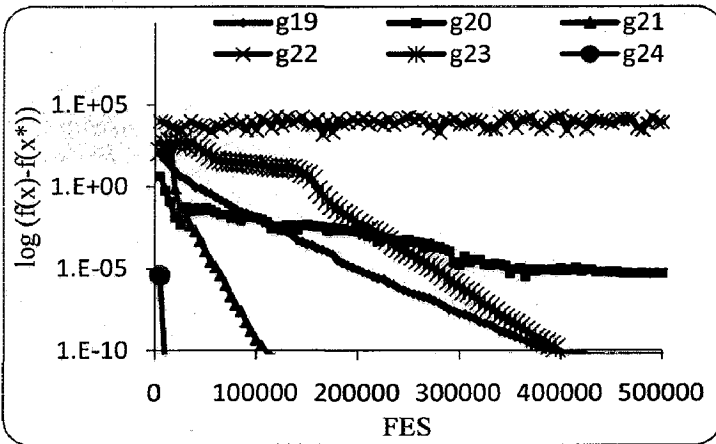


(a). Function error values.

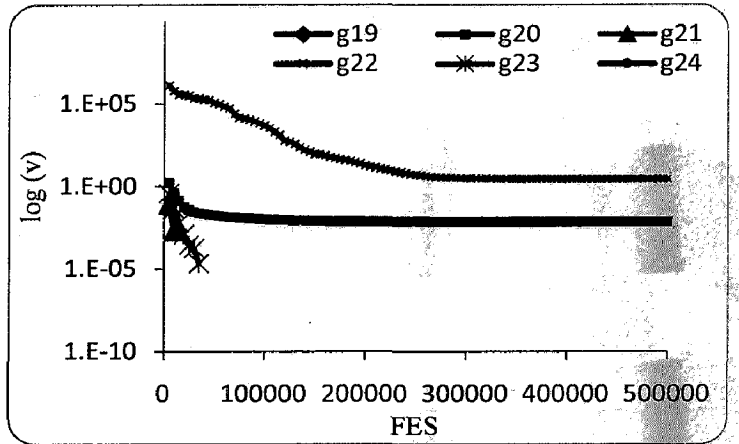


(b). Mean constraint violation.

Figure 3.7: Convergence graph for problems 13-18 of MSDE.



(a). Function error values.



(b). Mean constraint violation.

Figure 3.8: Convergence graph for problems 19-24 of MSDE.

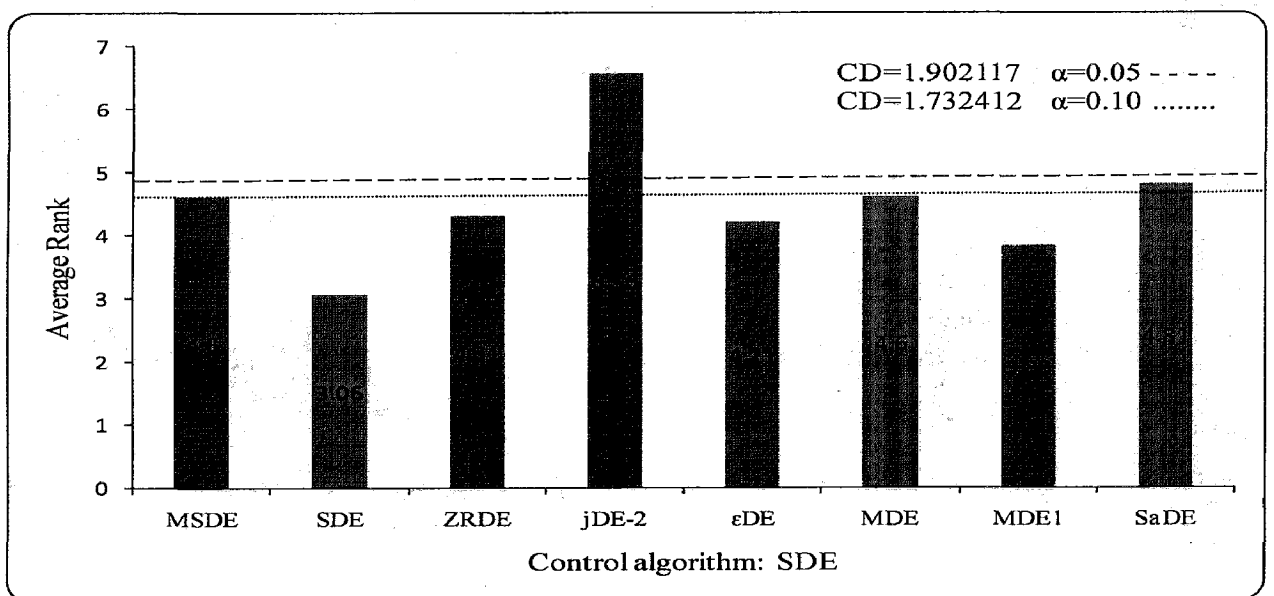


Figure 3.9: Bonferroni-Dunn's graphic.

SDE for Multi-Objective Optimization

Up till now this thesis dealt with the constrained and unconstrained optimization problems having single objective. However, many application problems involve multiple, often conflicting optimization criteria. Such problems are called multi-objective optimization problems (MOPs). The presence of more than one objective hampers the application of classical optimization techniques, which require a certain structure of the problem and are mostly designed to handle only a single objective.

Considering the algorithmic simplicity of the proposed SDE algorithm and its efficient performance for solving the constrained/ unconstrained optimization problems, in the present chapter, it is suitably modified and extended for solving multi-objective optimization problems. The performance of this extended version named MO-SDE is investigated on a set of nine standard benchmark MOPs and is compared with some recently modified versions of DE and some other Multi-Objective Evolutionary Algorithms (MOEAs). The empirical analysis of the numerical results shows the efficiency of the proposed algorithm.

The remainder of the chapter is structured as follows. Literature is given in section 4.1. Section 4.2 provides some definition related to this research. Section 4.3 describes the proposed MO-SDE. Experimental settings, test problems and performance metric are given in section 4.4. Results are discussed in section 4.5. Finally, the conclusions based on the present study are drawn in section 4.6.

4.1 Introduction

In the past few decades there has been a significant rise in the application of Evolutionary Algorithms (EAs) for solving multi-objective optimization problems (MOPs). This is primarily because of the fact that EAs deal with a set of solutions which help in the generation of well distributed Pareto optimal front more quickly and efficiently in comparison to the classical techniques. In mid 80's the application of EA was first suggested by Schaffer for solving MOPs (1984, 1985). Subsequently, several different algorithms have been proposed and

successfully applied to various benchmark and real life problems. For comprehensive overviews and discussions, the interested reader may refer to Fonseca and Fleming (1993, 1995), Horn (1997), Veldhuizen and Lamont (2000), Zitzler and Thiele (1999), Coello Coello (1996, 2000), Srinivas and Deb (1994) and Deb et al. (2002). Many more studies about the development and application of EA for solving MOPs can be found in Nakib et al. (2010), Hammouche et al. (2010), Bader and Zitzler (2008), Deb and Tiwari (2008) and Zitzler and Thiele (2010).

4.1.1 Application of DE for Solving MOP

Several researchers have studied the extension of DE to solve multi-objective optimization problems in continuous domain. Abbass et al. (2001, 2002) were the first to explore the potential of DE for solving MOPs. In their algorithm called Pareto Differential Evolution (PDE), DE is employed to create new solutions and only the nondominated solutions are kept as the basis for next generation. Their results showed the competence of DE for solving MOPs.

Madavan (2002) developed a multi-objective DE using a similar concept as that of PDE of Abbas et al. (2001). His algorithm, called Pareto Differential Evolution Approach (PDEA)¹, applies DE to create new solutions and keeps them in an auxiliary population. It then combines the two populations and calculates the nondominated rank and diversity rank, based on crowding distance for each solution. He demonstrated that with this approach the performance of DE for solving MOPs can be improved further.

Xue et al. (2003) introduced Multi-Objective Differential Evolution (MODE). This algorithm also uses Pareto-based ranking assignment and crowding distance metric, but in a different manner than PDEA. In MODE the fitness of a solution is first calculated using Pareto-based ranking and then reduced with respect to the solution's crowding distance value. This fitness value is then used to select the best solutions for next generation.

Robic et al. (2005) proposed Differential Evolution for Multi-objective Optimization (DEMO) and achieved good results. This algorithm, similar to PDEA algorithm, uses Pareto-based ranking assignment and crowding distance metric, but has a different population update strategy. According to this algorithm, if the newly generated solution dominates the target solution, then there is an immediate replacement of target solution in the current population.

¹ This acronym was not used by Madavan. It was introduced by Robic and Filipic (2005) in their paper.

However, if both are nondominated, then the new solution is added to the population for later sorting, otherwise the target solution is retained.

Multi-objective Differential Evolution Algorithm (MDEA), proposed by Adeyemo and Otieno (2009), generates new solution using DE variant and compares it with target solution. If it dominates target solution then it is added to new population, otherwise target solution is added. As the termination criterion is reached, dominated solutions are removed from last generation using Naïve and Slow approach (Deb, 2001).

Huang et al. (2007, 2009) extended their Self adaptive DE (SaDE) to solve MOPs by a so called Multi-Objective Self adaptive DE (MOSaDE). They further extended MOSaDE by using objective wise learning strategies in WO-MOSaDE.

4.2 Background

This section briefly describes the terminologies and concepts used in the chapter.

4.2.1 Multi-Objective Optimization Problem (MOP)

An unconstrained MOP can be formally defined as the problem of finding all $X = (x_1, x_2, \dots, x_D)$ to optimize the vector function:

$$\text{Minimize } (f_1(X), f_2(X), \dots, f_k(X))$$

In other words, the aim is to determine those decision vectors X in the decision space D which satisfy all the box constraints and optimize the objective function vector. The box constraints define the feasible region 'FR' and any vector X in the feasible region is called a feasible solution.

4.2.2 Pareto Dominance

Pareto dominance can be defined as (Deb, 2001):

A solution $X_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,D})$ is said to dominate the other solution $X_2 = (x_{2,1}, x_{2,2}, \dots, x_{2,D})$ if both the conditions mentioned below are satisfied:

1. $\forall i \in (1, 2, \dots, k) : f_i(X_1) \leq f_i(X_2)$
2. $\exists i \in (1, 2, \dots, k) : f_i(X_1) < f_i(X_2)$

Alternatively, it can be said that a solution dominates the other only if it is strictly better in at least one objective, and not worse in any of them. Thus, while comparing two different solutions X_1 and X_2 , there are three possibilities:

- ✓ X_1 dominates X_2
- ✓ X_1 is dominated by X_2
- ✓ X_1 and X_2 are nondominated

4.2.3 Fast Nondominated Sorting

In this approach introduced by Deb et al. (2002), for each solution i of a set S , two entities are calculated:

1. Domination count n_i , the number of solutions which dominate the solution i .
2. S_i , a set of solutions that the solution i dominates.

At the end of this procedure, all solutions in the first nondominated front F_1 have their domination count as zero. Now, for each solution i with $n_i = 0$, it visits each member (j) of its set S_i and reduces its domination count by one. While doing so, if for any member j the domination count becomes zero then it is put in a separate list P . These members belong to the second nondominated front F_2 . The above procedure is continued with each member of P and the third front F_3 is identified. This process continues until all fronts are identified.

4.2.4 Crowding Distance Metric

Crowding distance (Deb, 2001; Deb et al., 2002) is used to get an estimate of the density of solutions surrounding a particular solution i in the population, it calculates the average distance of two solutions on either side of solution i (i.e. $i+1$ & $i-1$) along each of the objectives. The crowding-distance computation requires sorting the population according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective. Each objective function is normalized before calculating the crowding distance.

4.3 Multi-Objective Synergetic Differential Evolution (MO-SDE)

The main challenging task for modifying the DE algorithm to deal with MOPs is to develop a robust strategy for generating a new point resulting in faster convergence to optimal Pareto front and to replace the points to obtain a solution set as diverse as possible.

4.3.1 Working of MO-SDE

The initial working of MO-SDE is same as that of the SDE algorithm proposed in chapter 2. That is to say, it uses OBL for generating the initial population and tournament based method for generating the trial vector. After the initialization and mutation phases are complete, MO-SDE performs crossover, as defined by equation (1.7), and enters the selection phase. This is perhaps the most important phase in terms of MOPs because a careful selection of candidate solutions helps in the generation of a good Pareto optimal front. The selection process of MO-SDE is defined in the following subsection.

4.3.2 Proposed Selection Mechanism Used in MO-SDE

MO-SDE combines the concept of PDEA (Madavan, 2002) and DEMO (Robic and Filipic, 2005) but with slight difference. In order to explain the selection process of the proposed MO-SDE, first a short description of the selection process of PDEA and DEMO is given.

PDEA applies the DE to create NP new solutions and keeps them in other (advance) population. It then combines the two (current & advance) populations. Note that the total size of the set after combination becomes $2NP$. After that, NP solutions are selected on the basis of nondomination rank and crowding distance rank (crowding distance in last front being accommodated in population) for next generation. PDEA allows a global non-domination check among both the parent and offspring solutions although it requires additional computational effort in sorting the combined rather than only the offspring population, as is done in many of the other approaches.

In DEMO, the trial solution replaces the target solution if it dominates it. If the target solution dominates the trial solution, the trial solution is discarded. Otherwise, the trial solution is added to the population. Thus, at the end of a generation, total size of the population is between NP and $2NP$. This population is truncated for the next step of the algorithm. The

truncation process consists of nondominated sorting and evaluating the solutions of the same front with the crowding distance metric. The truncation procedure keeps only the best (elite) NP solutions in the population. The idea of immediate replacement of solutions in DEMO makes it greedy in nature, therefore though it has a faster convergence, it may lose some important information while discarding the solution.

In MO-SDE, trial solution is compared to the target solution, if it dominates target solution, then it replaces target solution immediately in current population (as in DEMO) and target solution is added to advanced population, otherwise new solution (i.e. the trial solution) is added to the advanced population. After each generation, the two populations (current & advanced) are combined. Note that the total size is $2NP$ (as in PDEA).

Besides the above modifications, the MO-SDE also incorporates an effective elite-preserving and an explicit diversity-preserving strategy, borrowed from NSGA-II (Deb, 2001; Deb et al., 2002) for truncation of $2NP$ solutions to NP .

4.3.3 Effect of Using the Proposed Selection Mechanism

The immediate replacement of the parent solution with the candidate that dominates it is the core of MO-SDE. The newly created solution that enters the population instantly takes part in the creation of the new solutions. This helps in achieving the first goal of multi-objective optimization – convergence to the true Pareto front.

In case, there are many fronts in the current population and the target solution (say X_i) belongs to first front and trial solution dominates it, then it is discarded in DEMO. However, for MO-SDE, instead of discarding X_i , it is stored in the advanced population for latter sorting, because it may get place in subsequent front of the population for next generation.

Pseudocode of MO-SDE is given in Algorithm 4.1.

4.3.4 Computational Complexity Analysis

The proposed MO-SDE approach is simple to implement, yet efficient in yielding true Pareto optimal solutions. The computational complexity of MO-SDE is also reasonable. Here, complexity is defined as the total number of function value comparisons. Basic operations and their worst case complexities are as follows:

Algorithm 4.1 Pseudocode of MO-SDE illustrating how the procedure acts on a population of individuals, repeating mutation, crossover and selection until the convergence criteria is met.

- Step 1:* Generate randomly NP individuals, using equation (1.1) and NP opposite individuals using equation (2.7). Merge these two and select NP fittest solutions as initial population using nondominated sorting and crowding distance metric. Set the values of control parameters F and Cr .
- Step 2:* Set $i = 0$.
- Step 3:* $i = i + 1$.
- Step 4:* Corresponding to target individual X_i select three distinct individuals X_{r1} , X_{r2} and X_{r3} such that $i \neq r1 \neq r2 \neq r3$ from population and generate perturbed individual V_i using equation (2.8). For this operation tournament best is selected on the basis of non domination.
- Step 5:* Recombine the target vector X_i with perturbed individual V_i generated in step 4 to generate trial vector U_i using equation (1.7).
- Step 6:* If all parameters of the trial vector are within the given range then go to step 7 otherwise uniformly generate that parameter within given range using equation (1.1) and go to step 7.
- Step 7:* Calculate the objective function values for vector U_i . If trial vector U_i dominates target vector X_i then it immediately replace target vector in current population and target vector is added to auxiliary population otherwise trial vector is added to auxiliary population.
- Step 8:* If $i < NP$ then go to step 3 otherwise go to step 9.
- Step 9:* Merge these two population (current and auxiliary) and select NP fittest solutions for next generation using nondominated sorting and crowding distance metric.
- Step 10:* Check whether the termination criterion is met. If yes then stop otherwise go to step 2.
-

1. Selection of NP solutions out of $2NP$ solutions for initial population using nondominated and crowding distance sorting: $O(k \times (2NP)^2) + O(k \times (2NP) \times \log(2NP))$.
2. Procedure to check tournament best of three solutions for mutation for one iteration: $O(2 \times k \times NP)$.
3. Procedure to check the domination status of new solution with target solution for one iteration: $O(k \times NP)$.
4. Selection of NP solutions out of $2NP$ solutions (NP old and NP new) for next generation using nondominated and crowding distance sorting: $O(k \times (2NP)^2) + O(k \times (2NP) \times \log(2NP))$.

For large NP , $O(k \times (2NP) \times \log(2NP))$ is smaller than $O(k \times (2NP)^2)$. Therefore, the overall complexity of the MO-SDE is less than or equal to $O(k \times (2NP)^2)$, which is in well agreement with the latest versions of multi-objective evolutionary algorithms (MOEAs). For

example, the overall computational complexity of NSGA-II (Deb, 2001; Deb et al., 2002) is $O(k \times (2NP)^2)$.

Here k and NP represent the number of objective functions and population size respectively.

4.4 Experiments

4.4.1 Test Problems

The performance of the proposed algorithm is tested on a set of nine unconstrained benchmark problems (Deb et al., 2002), generally used to validate the performance of different MOEAs. These are Schaffer's function (SCH), Fonseca and Fleming's function (FON), Poloni's function (POL), Kursawe's function (KUR), ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. All problems have two objective functions.

Out of the considered problems, the first function SCH or Schaffer's objective function is relatively an easier problem because it has only one decision variable, and the Pareto front is convex.

The second Function FON is the three-variable formulation of Fonseca and Fleming. Here, the objective function is scalable and the Pareto front is a single concave curve.

The next function, POL is the two-objective function problem of Poloni with two non convex Pareto fronts that are disconnected in both the objective and decision variable spaces.

KUR is Kursawe's three variable problem where the true Pareto front is made up of three disconnected curves. This is particularly a difficult problem and its solution mapping into dominated objective space is quite convoluted.

ZDT1 is perhaps the easiest of all of the ZDT problems, and the only difficulty an MOEA may face in this problem is that it has a large number of variables.

The next problems ZDT2 is non convex in nature, which makes it difficult for an algorithm to solve it.

In the seventh problem ZDT3, the Pareto optimal front is made up of five disjoint curves which make it challenging for multi-objective optimization algorithms.

The eighth problem ZDT4 is deceptive in nature, having 21^9 different local Pareto-optimal fronts that may mislead the optimization algorithm.

The last problem, ZDT6, is another problem considered to be difficult. The two major difficulties with this problem are (1) thin density of solutions towards the optimal Pareto-front and (2) nonuniform spread of solutions along the front.

Mathematical models of these problems are given in Appendix IV.

4.4.2 Experimental Setup

To be consistent with literature (Abbass, 2002; Madavan, 2002; Xue et al., 2003), the following setting has been taken:

- Crossover probability $Cr = 0.3$
- Maximum number of function evaluations = 25000
- Number of trial = 10

Population size (NP) and scaling factor F are same as in chapter 2. In every case, a run is terminated when the number of function evaluations (NFE) reaches the threshold value of 25000. Also the boundary violation is handled in same manner as in chapter 2.

4.4.3 Performance Metric I

To validate proposed approach, it used the methodology normally adopted in the evolutionary multi-objective optimization literature. The two common metrics used to compare MO-SDE with other MOEAs are (1) Convergence Metric (2) Divergence Metric. They represent both quantitative and qualitative comparisons with MOEAs. For these metrics it needs to know the true Pareto front for a problem. In this chapter experiments use 500 uniformly spaced Pareto optimal solutions as the approximation of the true Pareto front. A brief introduction of these metrics is given here:

- Convergence metric Υ (Deb et al., 2002), measures the distance between the obtained nondominated front NF and optimal Pareto front PF . Mathematically, it may be defined as:

$$\Upsilon = \frac{\sum_{i=1}^N d_i}{N}$$

where N is the number of nondominated solutions found by the algorithm being analyzed and d_i is the minimum Euclidean distance (measured in the objective space) between the i^{th} solution of NF and the solutions in PF.

- Diversity metric Δ (Deb et al., 2002), measures the extent of spread achieved among the nondominated solutions. Its mathematical definition may be given as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^N |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}$$

where d_i is the Euclidean distance (measured in the objective space) between consecutive solutions in the obtained nondominated front, NF and \bar{d} is the average of these distances. The parameters d_f and d_l represent the Euclidean distance between the extreme solutions of the true Pareto-front, PF and the boundary solutions of the obtained NF.

4.4.4 Performance Metric II

The proposed algorithm is also analyzed statistically with other algorithms using non parametric Wilcoxon's signed ranks test (Garcia et al., 2009). Using this test, performed *multiple-problem analysis*, a comparison of algorithms over more than one problem simultaneously. It is a pairwise test that aims at detecting significant difference between the behaviour of two algorithms. A brief description of the test is given in Appendix V.

4.4.5 Algorithms Used for Comparison

- Nondominated Sorting Genetic Algorithm II (NSGA-II) (real) (Deb et al., 2002).
- Nondominated Sorting Genetic Algorithm II (NSGA-II) (binary) (Deb et al., 2002).
- Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999).
- Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 1999).
- Pareto Differential Evolution Approach (PDEA) (Madavan, 2002).
- Pareto-based Multi-Objective Differential Evolution (MODE) (Xue et al., 2003).
- Adaptive Differential Evolution Algorithm (ADEA) (Qian and Li, 2008).
- Differential Evolution for Multi-objective Optimization (DEMO) (Robic and Filipic, 2005).

- Multi-objective Differential Evolution Algorithm (MDEA) (Adeyemo and Otieno, 2009).

4.5 Results and Discussions

In this section, result of nine test problems using the proposed MO-SDE algorithm are compared with the results of nine other algorithms using convergence and diversity metrics. The results of other algorithms are taken from literature (Deb et al., 2002; Adeyemo and Otieno, 2009).

Tables 4.1- 4.9 represent the mean and variance of the values of convergence and diversity metrics of ten runs. The results are also compared statistically and corresponding results are given in Tables 4.10 and 4.11. Figures 4.1- 4.9 illustrate the obtained nondominated Pareto front and optimal Pareto front (PF). Besides the quantitative investigation of MO-SDE, it is also analyzed qualitatively with the help of graphical illustrations.

4.5.1 Results Based on Performance Metric I

From Table 4.1, which gives the result of SCH function, it is clear that MO-SDE was able to attain both the goals. This behaviour of MO-SDE is also evident from Figure 4.1, which illustrates a well predicted Pareto front and a large number of optimal solutions spread out over the entire front.

For the second function, FON, it can be clearly seen from Table 4.2 and Figure 4.2 that the proposed MO-SDE effectively finds diverse solutions along the optimal PF.

For the next function, POL, it can be observed from Figure 4.3 that the proposed method is able to predict the two disconnected Pareto fronts that lie on the boundaries of the search space. However, from Table 4.3 it is seen that although MO-SDE achieves better convergence, better spread of solution is achieved by NSGA-II (real).

For the function, KUR, apparently a difficult function having three disconnected curves, it is observed from Table 4.4 and from Figure 4.4 that MO-SDE performed quite well for it.

For the fifth function, ZDT1, all the algorithms converged to the Pareto optimal front with good spread over the entire front. However, the convergence metric of MO-SDE is much

smaller in comparison to the others, shown in Table 4.5, which demonstrates a superior convergence ability of the proposed MO-SDE.

A nondominated solution obtained by MO-SDE on ZDT2 is shown in Figure 4.6, from which it is clear that MO-SDE attains both the goals of multi-objective optimization; also it is obvious from Table 4.6, that MO-SDE found a better spread with a smaller convergence and diversity metric values than the other algorithms.

From Figure 4.7, it can be observed that the nondominated front obtained by MO-SDE almost converges to the optimal Pareto front. From Table 4.7, it is seen that the best spread is found by MDEA; however in terms of convergence MO-SDE outperforms.

The eighth problem ZDT4 for which the results are given in Table 4.8, it can be observed that, in terms of convergence and diversity metrics, MO-SDE performs much better than all other algorithms. This behaviour can also be seen from Figure 4.8, which illustrates that the Pareto front obtained by the proposed algorithm overlaps the optimal PF.

The last problem, ZDT6, which is once again a considerably difficult problem it can be observed that MO-SDE finds a better spread in comparison to all the other algorithms however, its convergence is not as good in comparison to ADEA, MODE and MDEA. The spread of solutions obtained by MO-SDE is also clear from Figure 4.9.

Additionally, it can be seen from Tables 4.1- 4.9 that variance values resulting from the MO-SDE algorithm are very small in every case, demonstrating the robustness of the MO-SDE algorithm.

4.5.2 Results Based on Performance Metric II

Besides, comparing MO-SDE on the basis of convergence and diversity metrics, it is also analyzed the statistically vis-a-vis other algorithms using Wilcoxon's signed ranks test. The corresponding results are given in Tables 4.10 and 4.11 for convergence and diversity metric respectively.

These tables display the statistics, p -value and sum of positive ranks (where MO-SDE algorithm performed better than the competing algorithm), negative ranks (where the competing algorithm performed better than MO-SDE).

From Table 4.10, it can be seen that in case of convergence MO-SDE outperformed all the algorithms except DEMO and MDEA. This shows that although statistically there is no

significant difference between MO-SDE, DEMO and MDEA, MO-SDE performed comparatively better than the remaining seven algorithms.

Likewise, from Table 4.11, which shows the results on the basis of diversity, it is observed that MO-SDE is better than SPEA and PAES and is equivalent to other algorithms.

4.6 Summary

In the present chapter, SDE algorithm introduced in chapter 2 is suitably modified for solving MOPs. The prime objective of MO-SDE is to attain the true optimal front, by maintaining a careful balance between the convergence and diversity metric.

MO-SDE inherits the basic features of SDE in population initialization (using OBL) and mutation (tournament selection) but employs a different selection procedure which is the immediate replacement of the parent vector with the candidate that dominates it. This helps MO-SDE to obtain a faster convergence. MO-SDE also follows the concept of global nondomination check among the parent and offspring candidates, so that none of the information is lost during the selection process. For truncating the population of $2NP$ solution to NP solutions, it uses the nondominated sorting and crowding distance metric.

The numerical results show that with the help of proposed modifications, SDE can be suitably extended for solving MOPs.

MO-SDE performed better than the competing algorithms or at least at par for most of the problems considered in the present study. The following conclusions can be drawn from the analysis of MO-SDE:

- The overall computational complexity of the MO-SDE is quite reasonable. It is less than or equal to $O(k \times (2NP)^2)$, which is better than or comparable to the overall computational complexity of some of the other competing algorithms for solving MOPs.
- In terms of convergence metric, MO-SDE outperformed the other algorithms for all the test problems, except for ZTD6. For this function, DEMO gave the best performance.
- In terms of diversity metric, which tells about the spread of Pareto front, it is observed that in five cases MO-SDE gave a better result in comparison to other algorithms but, in the remaining problems, some of the other algorithms performed better than MO-SDE. This

shows that improvement can be made further in the performance of the proposed MO-SDE.

- Statistical observation shows that on the basis of convergence metric, MO-SDE is at par with DEMO and MDEA while it is significantly better than the remaining seven algorithms. In terms of diversity metric, MO-SDE is equivalent to the other algorithms except for SPEA and PAES, for which it is significantly better.

Table 4.1: Statistics of the results on the test problem SCH.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.003391±0.000000	0.477899±0.003471
NSGA-II(binary)	0.002833±0.000001	0.449265±0.002062
SPEA	0.003403±0.000000	1.021110±0.004372
PAES	0.001313±0.000003	1.063288±0.002868
MO-SDE	0.000531±0.000000	0.376085±0.000836

Table 4.2: Statistics of the results on the test problem FON.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.001931±0.000000	0.378065±0.000639
NSGA-II(binary)	0.002571±0.000000	0.395131±0.001314
SPEA	0.125692±0.000038	0.792352±0.005546
PAES	0.151263±0.000905	1.162528±0.008945
MO-SDE	0.001642±0.000000	0.286400±0.001003

Table 4.3: Statistics of the results on the test problem POL.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.015553±0.000001	0.452150±0.002868
NSGA-II(binary)	0.017029±0.000003	0.503721±0.004656
SPEA	0.037812±0.000088	0.972783±0.008475
PAES	0.030864±0.000431	1.020007±0.000000
MO-SDE	0.001252±0.000000	0.797450±0.000228

Table 4.4: Statistics of the results on the test problem KUR.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.028964±0.000018	0.411477±0.000992
NSGA-II(binary)	0.028951±0.000016	0.442195±0.001498
SPEA	0.045617±0.000050	0.852990±0.002619
PAES	0.057323±0.011989	1.079838±0.013772
MO-SDE	0.002753±0.000002	0.506228±0.000340

Table 4.5: Statistics of the results on the test problem ZDT1.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.033482±0.004750	0.390307±0.001876
NSGA-II(binary)	0.000894±0.000000	0.463292±0.041622
SPEA	0.001799±0.000001	0.784525±0.004440
PAES	0.082085±0.008679	1.229794±0.000742
PDEA	N/A	0.298567±0.000742
MODE	0.005800±0.000000	N/A
ADEA	0.002741±0.000385	0.382890±0.001435
DEMO	0.001132±0.000136	0.319230±0.031350
MDEA	0.000921±0.000005	0.283708±0.002938
MO-SDE	0.000469±0.000001	0.333027±0.000995

Table 4.6: Statistics of the results on the test problem ZDT2.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.072391±0.031689	0.430776±0.004721
NSGA-II(binary)	0.000824±0.000000	0.435112±0.024607
SPEA	0.001339±0.000000	0.755184±0.004521
PAES	0.126276±0.036877	1.165942±0.007682
PDEA	N/A	0.317958±0.001389
MODE	0.005500±0.000000	N/A
ADEA	0.002203±0.000297	0.345780±0.003900
DEMO	0.000780±0.000035	0.326821±0.021083
MDEA	0.000640±0.000000	0.450482±0.004211
MO-SDE	0.000514±0.000000	0.321888±0.000457

Table 4.7: Statistics of the results on the test problem ZDT3.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.114500±0.004940	0.738540±0.019706
NSGA-II(binary)	0.043411±0.000042	0.575606±0.005078
SPEA	0.047517±0.000047	0.672938±0.003587
PAES	0.023872±0.000010	0.789920±0.001653
PDEA	N/A	0.623812±0.000225
MODE	0.021560±0.000000	N/A
ADEA	0.002741±0.000120	0.525770±0.043030
DEMO	0.001236±0.000091	0.328873±0.019142
MDEA	0.001139±0.000024	0.299354±0.023309
MO-SDE	0.000681±0.000000	0.730341±0.000046

Table 4.8: Statistics of the results on the test problem ZDT4.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.513053±0.118460	0.702612±0.064648
NSGA-II(binary)	3.227636±7.307630	0.479475±0.009841
SPEA	7.340299±6.572516	0.798463±0.014616
PAES	0.854816±0.527238	0.870458±0.101399
PDEA	N/A	0.840852±0.035741
MODE	0.638950±0.500200	N/A
ADEA	0.100100±0.446200	0.436300±0.110000
DEMO	0.041012±0.063920	0.407225±0.094851
MDEA	0.048962±0.536358	0.406382±0.062308
MO-SDE	0.000603±0.000000	0.372672±0.003498

Table 4.9: Statistics of the results on the test problem ZDT6.

Algorithm	Convergence metric	Divergence metric
NSGA-II(real)	0.296564±0.013135	0.668025±0.009923
NSGA-II(binary)	7.806798±0.001667	0.644477±0.035042
SPEA	0.221138±0.000449	0.849389±0.002713
PAES	0.085469±0.006664	1.153052±0.003916
PDEA	N/A	0.473074±0.021721
MODE	0.026230±0.000861	N/A
ADEA	0.000624±0.000060	0.361100±0.036100
DEMO	0.000642±0.000029	0.458641±0.031362
MDEA	0.000436±0.000055	0.305245±0.019407
MO-SDE	0.001474±0.000000	0.302587±0.000076

Table 4.10: Statistical results by Wilcoxon test for convergence.

MO-SDE Vs.	R^+	R^-	Statistics	p -value
NSGA-II(real)	45	0	-2.666	0.008
NSGA-II(binary)	45	0	-2.666	0.008
SPEA	45	0	-2.666	0.008
PAES	45	0	-2.666	0.008
PDEA	--	--	--	--
MODE	15	0	-2.023	0.043
ADEA	14	1	-1.753	0.049
DEMO	11	4	-0.944	0.345
MDEA	11	4	-0.944	0.345

Table 4.11: Statistical results by Wilcoxon test for diversity.

MO-SDE Vs.	R^+	R^-	Statistics	p -value
NSGA-II(real)	33	12	-1.244	0.214
NSGA-II(binary)	29	16	-0.770	0.441
SPEA	44	1	-2.547	0.011
PAES	45	0	-2.666	0.008
PDEA	9	6	-0.405	0.686
MODE	--	--	--	--
ADEA	10	5	-0.674	0.500
DEMO	8	7	-0.135	0.893
MDEA	8	7	-0.135	0.893

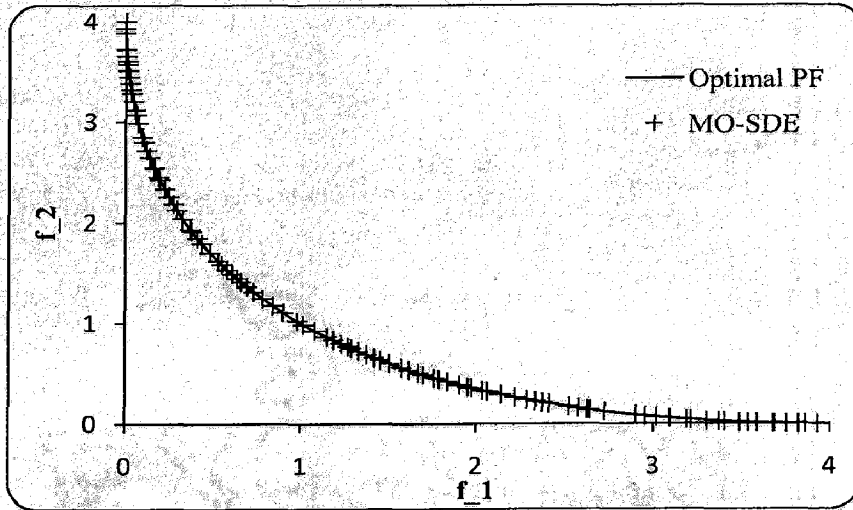


Figure 4.1: True PF and nondominated solutions by MO-SDE on SCH.

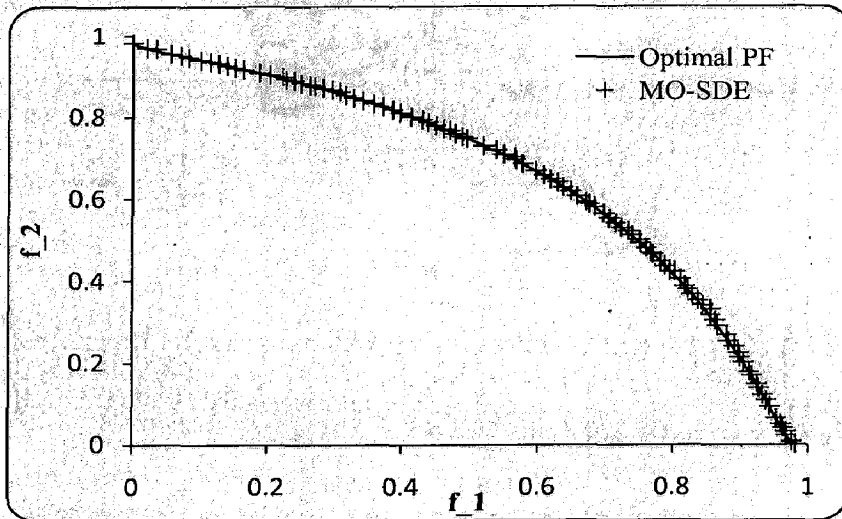


Figure 4.2: True PF and nondominated solutions by MO-SDE on FON.

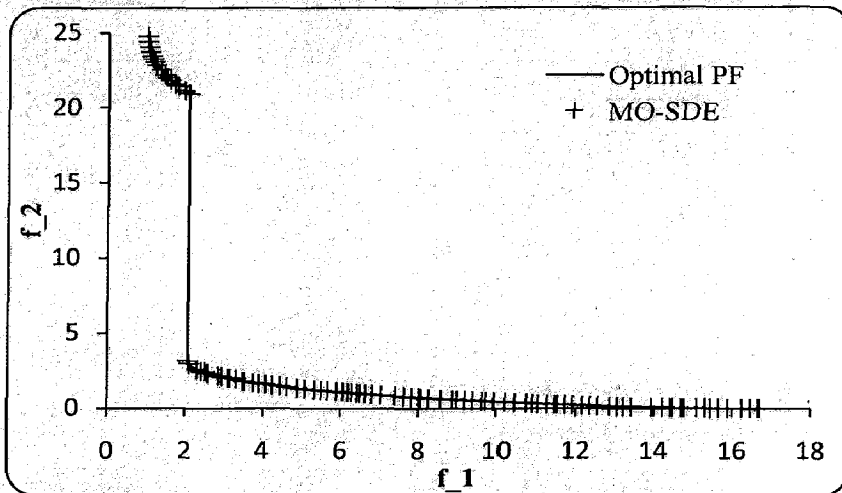


Figure 4.3: True PF and nondominated solutions by MO-SDE on POL.

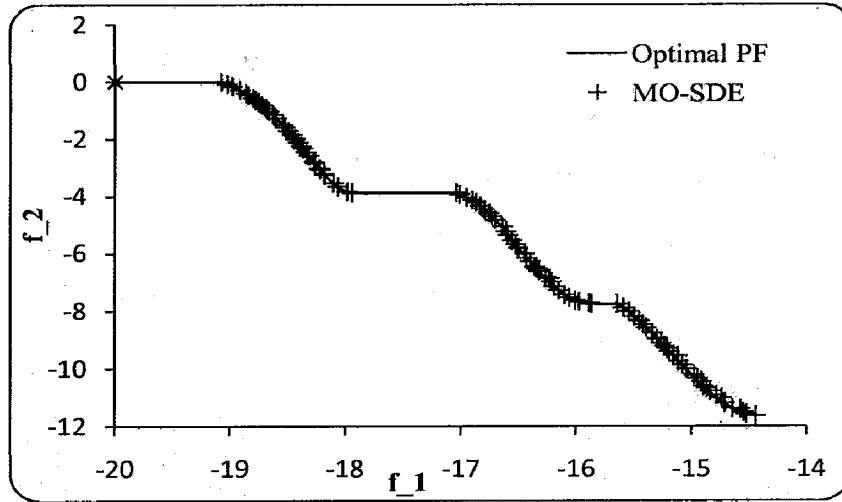


Figure 4.4: True PF and nondominated solutions by MO-SDE on KUR.

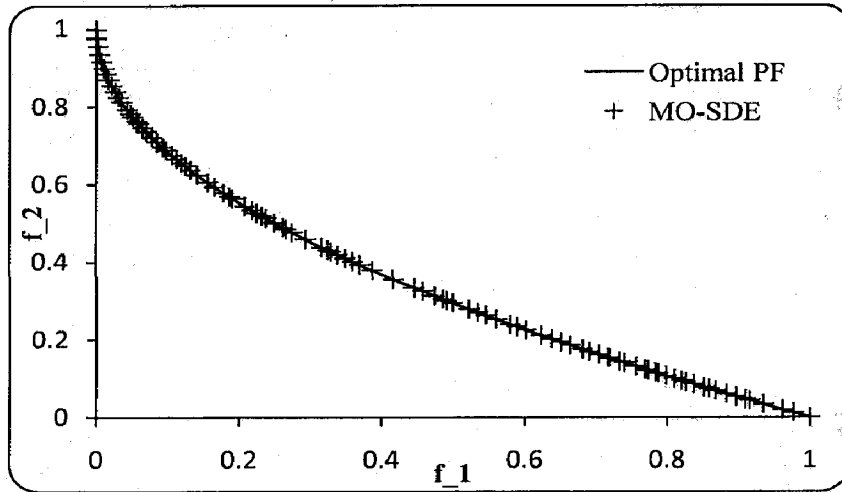


Figure 4.5: True PF and nondominated solutions by MO-SDE on ZDT1.

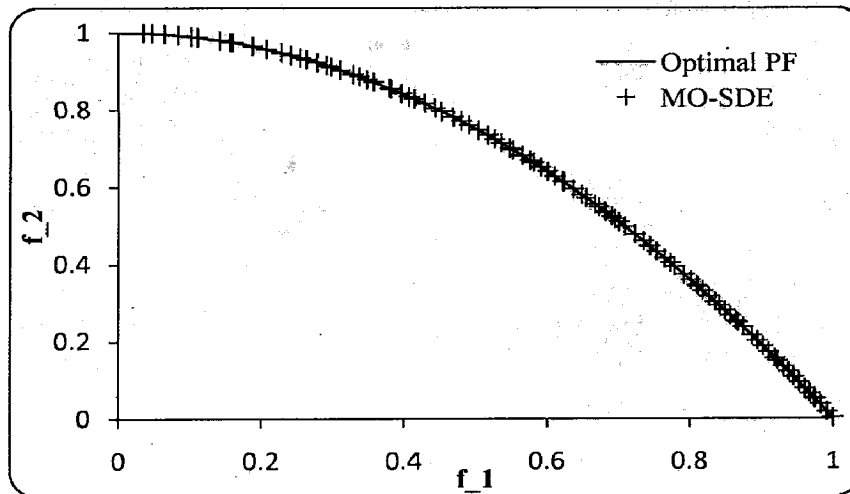


Figure 4.6: True PF and nondominated solutions by MO-SDE on ZDT2.

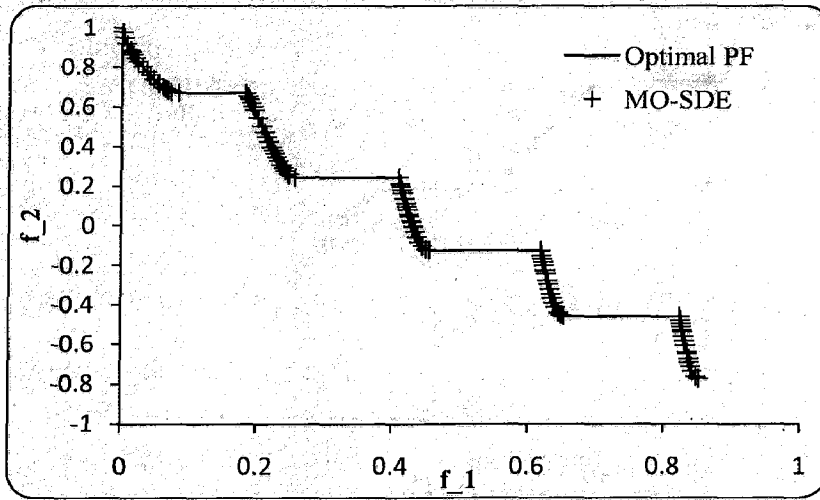


Figure 4.7: True PF and nondominated solutions by MO-SDE on ZDT3

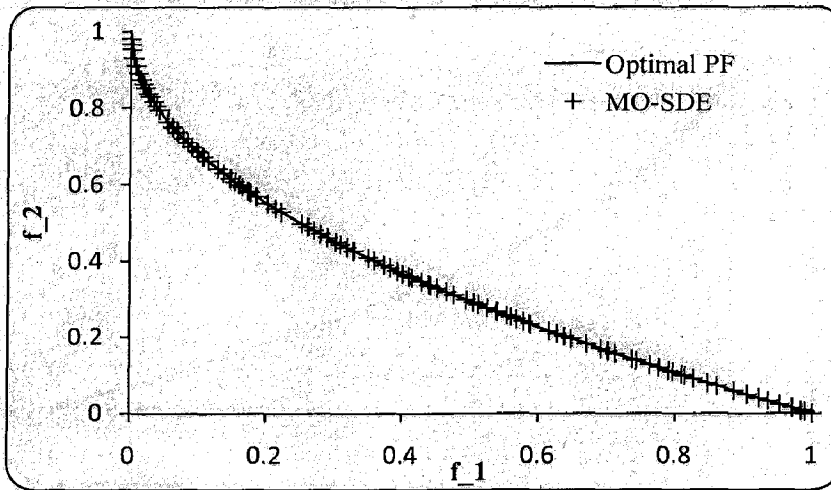


Figure 4.8: True PF and nondominated solutions by MO-SDE on ZDT4

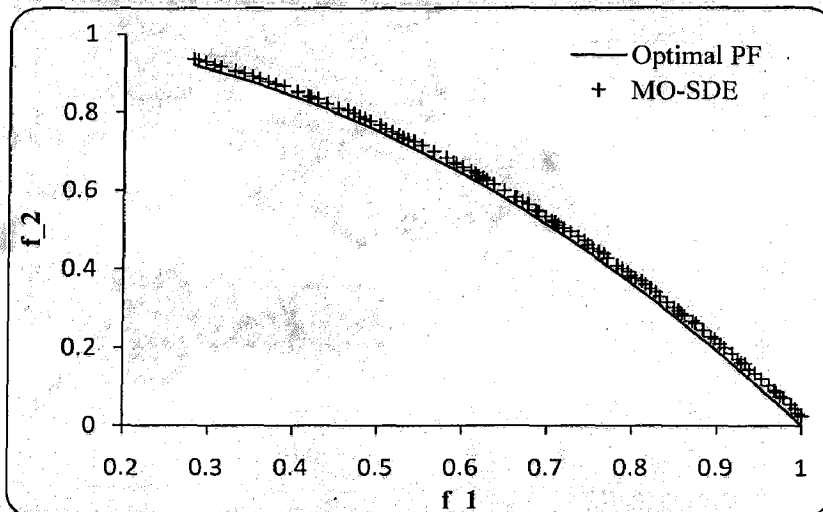


Figure 4.9: True PF and nondominated solutions by MO-SDE on ZDT6

Trim Loss Optimization

In chapters 2 and 3, SDE algorithm was applied to solve unconstrained and constrained numerical benchmark test functions, respectively, where the results clearly indicated the competence of this algorithm for solving such problems. However, in the real world, mathematical model of optimization problems is significantly different from that of benchmark functions. In this chapter the efficiency of SDE algorithm is validated on a well known “trim loss” (TLP) or “cutting-stock” (CSP) problem arising in various industries (glass, fabric and paper industries etc.). In this chapter CSP and TLP will be used alternatively. Mathematically, it can be described as a nonconvex mixed integer nonlinear programming problem subject to several constraints.

Four hypothetical but relevant cases of trim loss problem arising in paper industry are taken for experiment.

The chapter is structured as follows: Section 5.1 reviews the available literature regarding the problem considered in this chapter. Section 5.2 presents the formulation of the trim loss problem. Section 5.3 states the implementation of SDE for solving trim loss problem. Finally, section 5.4 provides summary of the chapter.

5.1 Introduction

Production planning in the fine paper industry is extremely difficult because a huge variety of finished paper products are demanded by customers. The TLP is the most common problem encountered during the cutting of different type of ordered products from the available limited raw material to satisfy the customer’s demands. Therefore the aim of manufacturers is to produce desired products to meet the customer’s demands economically with the maximum utilization of available raw materials, thereby minimizing the inevitable waste of material.

The procedure followed in the paper industry may be broadly described as follows: fixed width *reels* of a given paper grade, also referred to as *jumbo rolls*, are produced on paper machines in a paper mill. These reels are then cut into several *rolls* of smaller diameters and

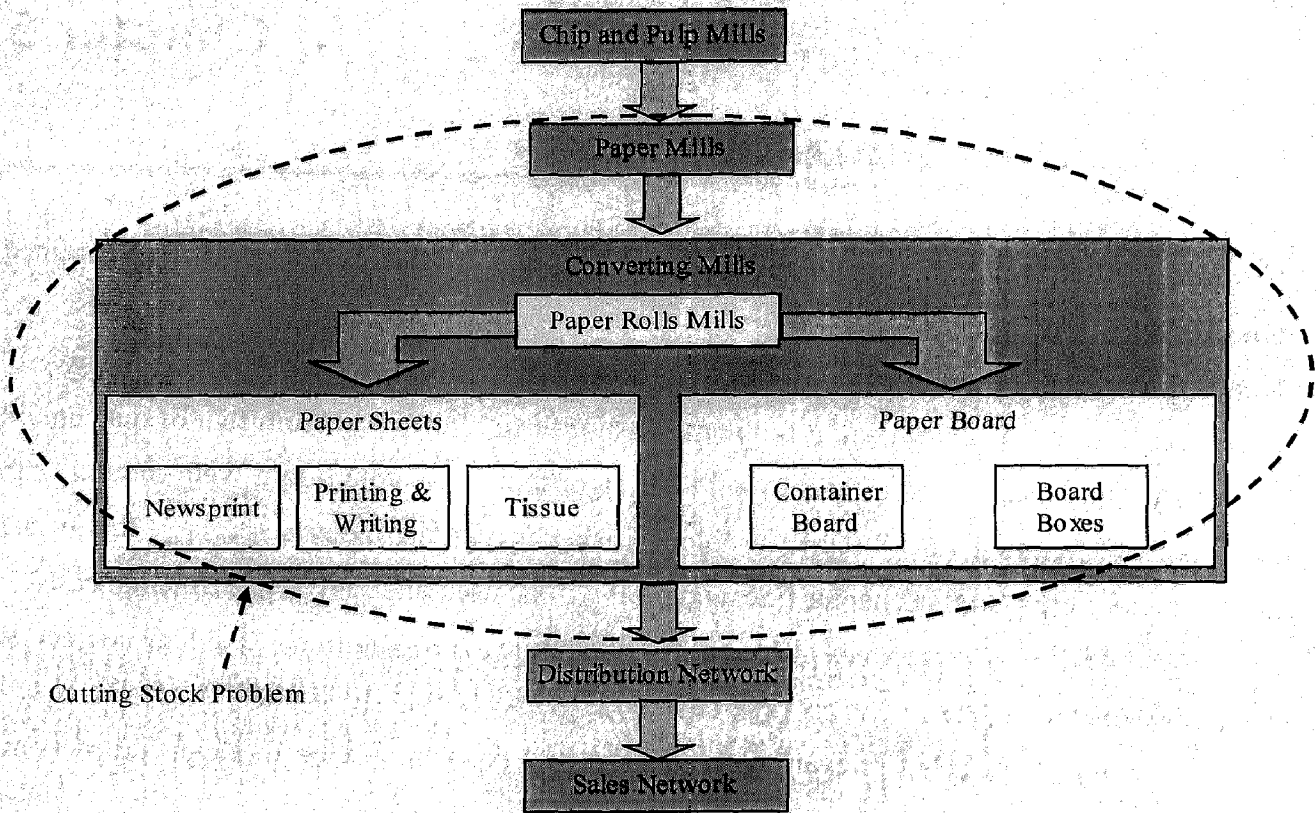


Figure 5.1: The pulp and paper supply chain.

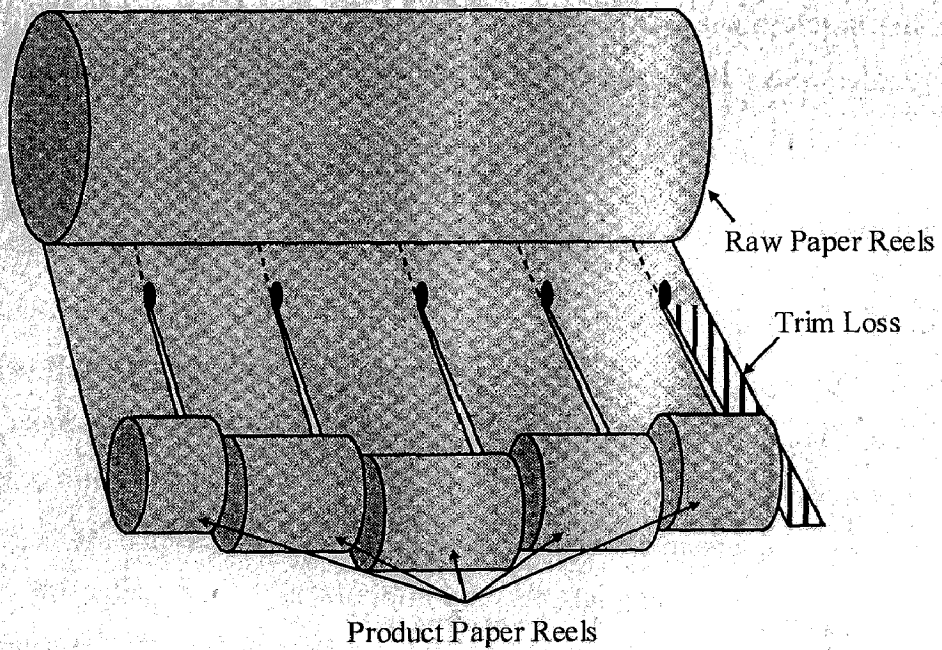


Figure 5.2: A schematic illustration of trim loss problem.

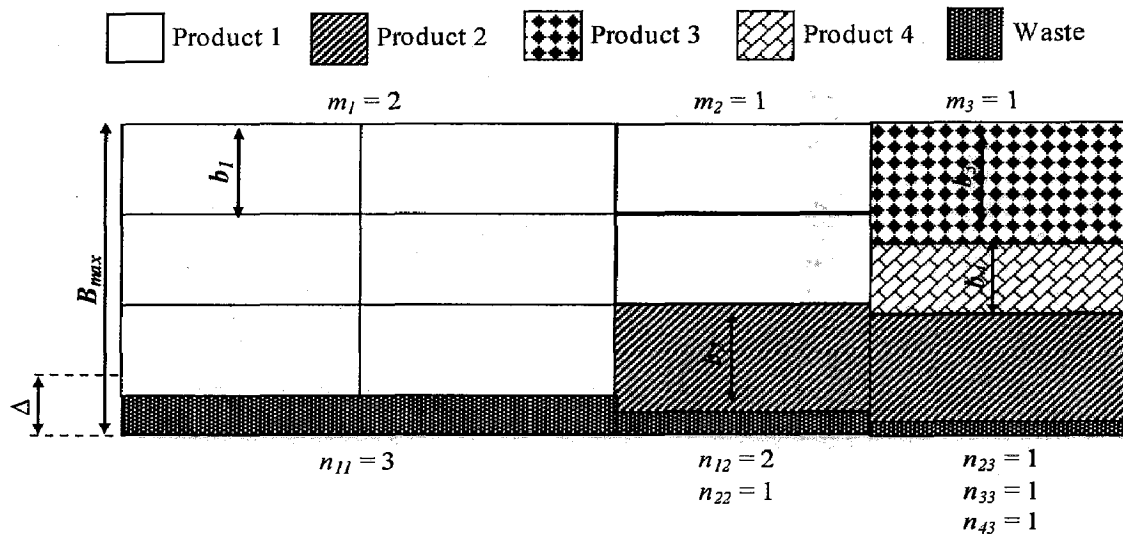


Figure 5.3: The cutting pattern.

widths on a winder. A significant part of the rolls produced in the paper mill are transformed into cut-sheet finished products on sheeters in a converting plant, which may generate some trim loss. The rolls sheeted into finished products are known as parent rolls.

A systematic representation of supply chain is shown in Figure 5.1 (adapted from Rodriguez and Vecchiotti, 2008). The CSP is a common problem in almost every link in pulp and paper supply chain. Consequently, the industries perform their activities in a very competitive market trying to maintain an efficient production plan besides providing convenient product prices and just in time order deliveries.

Considering the practicality and importance of the TLP, it has been given considerable attention by the researchers for developing its model and for recommending various methods to solve it efficiently.

From the mathematical formulation point of view, many articles are available in which the CSP has been studied with different goals such as minimizing trim-loss (Harjunoski et al., 1996; Harjunoski et al., 1999; Trkman and Gradisar, 2007), minimizing the production costs (Harjunoski et al., 1998; Harjunoski et al., 1999), minimizing the number of patterns (Johnston and Sadinlija, 2004), minimizing the total length and overproduction (Correia et al., 2004) etc.

It was observed that TLP can be modelled as a global optimization problem having a complex formulation (mathematical formulation of TLP is discussed in the next section) and therefore efficient techniques are required for finding its solution. Basically three different classes of solution methods exist for solving the CSP (i) Algorithmic methods which guarantee the optimal solution for the problem to be found but the drawback of these methods is computational complexity which often results into high running times, especially with large dimension problems. For this reason, purely algorithmic approaches for solving the CSP are usually avoided. (ii) The second class of solutions is heuristic methods, which may not find the exact optimal solution, but usually generate a faster and an acceptable solution. A heuristic is considered acceptable if the solutions are close enough to the known or expected optimal solution. Heuristic methods are also highly domain-dependent i.e. to say that they use information about the particular problems for which they are developed. Thus they may appear to be little of use on apparently similar problems (Hinxman, 1980), (iii) finally there are Metaheuristic methods which have an ability of not being trapped into local optima as might happen with traditional heuristics. In metaheuristic methods, the solution process is often guided by some lower level heuristic.

A series of articles are available in literature using heuristic and metaheuristic methods. Some of these are: linear programming approximations for the reel cutting stock (Gilmore and Gomory, 1961). However it was observed that linear approximation was not a very pragmatic approach for solving such a complex problem. Therefore efforts were made to solve the non-linear models heuristically (Haessler, 1971; Coverdale and Wharton, 1976; Johnston, 1979).

Considering the fact that the decision variables of a TLP are of integer type therefore, mixed integer linear programming (MILP) has also been applied for solving trim loss problems. A good overview of different formulations and solution methods is given in Hinxman (1980). Instances of application of heuristic techniques for solving TLP can be found in Beasley (2004), Riehme et al. (1996), Harjunoski et al. (1996), Harjunoski et al. (1998) and Westerlund and Isaksson (1998) etc.

Taking into account the complexity of TLP, metaheuristic techniques are probably a more pragmatic approach for solving such problems. Different metaheuristics that have been used include Tabu Search (Alvares-Valdés et al, 2002), Simulated Annealing (Lai and Chan,

1997), PSO (Xianjun et al, 2007), Genetic Algorithm (Wagner, 1999), Hybrid Genetic Algorithm (Ostermark, 1999) etc.

5.2 Mathematical Formulation

The trim-loss problem appears when a set of ordered product reels are to be cut from raw paper reels or from other reels having specified widths. The cutting process is simply a winding process, where the raw paper is wound through the slit and is cut by a set of knives positioned on the line (Figure 5.2). Product widths can rarely be combined to the exact raw paper widths; therefore waste appears during the cutting process. The objective is to minimize the trim loss while satisfying the demand specifications. In this chapter the mathematical formulation of TLP suggested by Adjiman et al. (2000) and Yen et al. (2004) is taken into consideration.

A raw paper roll of width B_{\max} must be cut to satisfy the following order specifications. There are $i = 1, \dots, N$ different products, n_i rolls of order i with a width b_i must be cut. All products rolls are assumed to be of equal length. In order to identify the best overall scheme, a maximum of $j = 1, \dots, P$; different cutting patterns are postulated. A pattern is defined by the position of the knives. The number of times pattern j is repeated is given by an integer variable m_j . The existence of a product in a given pattern is denoted by an integer variable r_{ij} . The binary variable y_j is introduced to a change in pattern.

If a new pattern is introduced ($m_j > 0$), then y_j is equal to one. A sample cutting pattern is shown in Figure 5.3.

The actual cost of the trim loss is the total amount of raw materials used, that is, the sum all repeated patterns multiplied by a cost factor C_j , in addition to the cost of changing knife positions between patterns.

Let the pattern change be weighted by a coefficient c_j .

The trim loss problem may now be defined as:

$$\text{Minimize } \sum_{j=1}^P (C_j \cdot m_j + c_j \cdot y_j) \quad (5.1)$$

restricted by the following constraints:

The number of rolls of each product must be greater than customer's order.

$$\sum_j m_j r_{ij} \geq n_i, \quad i = 1, \dots, N \quad (5.2)$$

The width of each pattern must be less than the width of raw paper roll and width of cut product in each pattern must exceed a certain minimum Δ :

$$(B_{\max} - \Delta)y_j \leq \sum_i b_i r_{ij} \leq B_{\max} y_j, \quad j = 1, \dots, P \quad (5.3)$$

This constraint imposes a lower bound on the total number of patterns made:

$$\sum_{j=1}^P m_j \geq \max \left\{ \left\lceil \sum_{i=1}^N n_i / Nk_{\max} \right\rceil, \left\lceil \sum_{i=1}^N n_i b_i / B_{\max} \right\rceil \right\} \quad (5.4)$$

There must be at least one product in a pattern and total number of knives limited to Nk_{\max} :

$$y_j \leq \sum_{i=1}^N r_{ij} \leq Nk_{\max} y_j, \quad j = 1, \dots, P \quad (5.5)$$

There must be at least one pattern after a knife change and the maximum number of pattern repetition is limited to M_j :

$$y_j \leq m_j \leq M_j y_j, \quad j = 1, \dots, P \quad (5.6)$$

Constraints (5.7) and (5.8) introduce an order on y and m variables to reduce degeneracy:

$$y_{k+1} \leq y_k, \quad k = 1, \dots, P-1 \quad (5.7)$$

$$m_{k+1} \leq m_k, \quad k = 1, \dots, P-1 \quad (5.8)$$

$$y_j \in \{0, 1\}, \quad j = 1, \dots, P \quad (5.9)$$

$$m_j \in [0, M_j] \cap Z, \quad j = 1, \dots, P \quad (5.10)$$

$$r_{ij} \in [0, Nk_{\max}] \cap Z, \quad i = 1, \dots, N, \quad j = 1, \dots, P \quad (5.11)$$

$$c_j = 1, \quad j = 1, \dots, P \quad \text{and} \quad C_j = 0.1, \quad j = 1, \dots, P \quad (5.12)$$

where Z is a set of integers.

The maximum number of knives is $Nk_{\max} + 1$, since edge cuts need to be made at both sides of raw paper reel. The width of an edge cut depends entirely on the product quality and the machine used and can't be controlled by the schedule. For this reason, it is not considered in the mathematical model. All the variables are either integer or binary in nature. The presence of bilinear inequality (5.6) makes the problem nonlinear and nonconvex.

5.3 Implementation of SDE for Trim Loss Problem

Up till now in this thesis SDE has been applied for solving problems having continuous variables. The TLP however turns out to be a MINLP problem (having binary variables as well) therefore suitable changes are made in SDE to adapt it for dealing with integer as well as binary variables. This is described in the following section:

5.3.1 Handling of Integers and Binary Variables in SDE

In its canonical form SDE, as DE, is only capable of handling continuous variables. Extending it to optimize integer variables is, however, an easy task and requires only a couple of simple modifications. Integer values can be used to evaluate the objective function and constraints, while SDE itself works internally with continuous floating points.

According to the literature, getting integer values for evaluating objective function and constraints can be done in two ways (1) by rounding the continuous variables (Srinivas and Rangaiah, 2007) to the nearest integers and (2) by truncating the values to integers (Angira and Babu, 2006).

In the present study, *rounding off* method is used because it has an equal probability to choose between nearest lower and nearest upper integer values. For example, if the continuous variable has a value of 5.7 in one case and 5.4 in the second case, then rounding off the digits takes the nearest highest integer as 6 in the first case and the nearest lowest integer as 5 in the second case. Truncation, on the other hand, takes the value as 5 in both the cases since it always takes the nearest lowest integer value. Thus, it can be seen that the former method is unbiased and is therefore more reasonable.

Binary variables are also handled in the same fashion as that for integers except that in this case the bounds are restricted between 0 and 1.

5.3.2 Handling Constraint and Boundary Violations

Constraints for TLP are handled using the method described in chapter 3. Boundary violations, are dealt as described in chapter 2. Before checking the boundary violation the variable is rounded off to the nearest integer.

5.3.3 Control Parameter Settings

Fine tuning of SDE parameters was done to obtain the appropriate value of control parameters for solving the TLP.

A series of experiments were conducted and it was observed that a smaller crossover rate (< 0.5) gave good results for TLP. In this study, the value of Cr is therefore taken as 0.3.

Considering the complexity of the problem the numbers of function evaluations (NFE) were kept quite high as 1,500,000.

Finally, a reasonable accuracy of 10^{-04} was taken to analyze the performance of SDE.

All the other parameters are kept same as given in chapter 2.

5.3.4 Numerical Results

SDE has been tested on four problems for the model described in section 5.2. The orders of the customers for the problems are given in Tables 5.1-5.4 and problem parameters are given in Table 5.5. In order to minimize the effect of the stochastic nature of the algorithm each problem is executed 50 times with different random seeds and the average fitness value of the best solutions throughout the optimization run is recorded. An Intel Dual Core personal computer with 1GB RAM is used in this experiment. The experimental best results with fitness value are given in Tables 5.6-5.9 and the cutting patterns are given in Tables 5.10-5.13. Alternate solutions obtained by the SDE are listed in Tables 5.14 and 5.15. Other results which consist of the best and the worst results, standard deviation (Std.) and average values of the obtained results for all problems are recorded in Table 5.16. Additionally, the computational times, number of functions evaluation (NFE) and success rate (SR) are also included in Table 5.16.

The convergence graphs of SDE, illustrating best fitness vs. number of function evaluation are given in Figures 5.4 and 5.5. Further, the convergence graphs of constraints violation are also illustrated in the same figures on secondary axis.

Comparison of SDE with other algorithms on the basis of NFE and SR, are given in Table 5.17.

From Tables 5.6 and 5.7 it can be seen that for problems 1 and 2 all the algorithms give same optimal solution. While from Tables 5.8 and 5.9 it can be said that ILXPSO (), a PSO variant used for solving this problem, does not achieve the global optimum.

From Tables 5.10-5.13, which gives trim loss in terms of width, it can be said that trim loss is very less. Success rate of SDE for problems 1-4 are 93, 100, 89 and 93, respectively. Also variance is very less. SDE is also compared with GMIN- α BB (Adjiman et al., 2000), ILXPSO (Deep et al., 2009), RCGA (Deep et al., 2009) in term of NFE and SR. From Table 5.17 it is clear that performance of SDE is better than other algorithms in both criteria. CPU absorbances time for the algorithms is not compared here, since they were implemented in completely different environments while it is recorded for SDE in Table 5.16.

5.4 Summary

In this chapter, the performance of SDE is analyzed on a real life problem of trim loss (or TLP), arising frequently in paper industries. TLP is specially suited for testing the efficiency of an optimization algorithm like that of SDE because of its complex mathematical model which is nonlinear and non-convex and contain integer as well as binary variables. Also, it has several constraints associated with it. Conclusions drawn at the end of this chapter can be summarized as follows:

- SDE can be easily modified for solving the problems having integer or/and binary restrictions imposed on it.
- SDE can deal efficiently with nonlinear/ nonconvex optimization problems subject to several constraints. This is important in real life scenarios where usually the problems are complex in nature.
- SDE outperformed some of the contemporary optimization algorithms in terms of solution quality as well as convergence rate.

Table 5.1: Example order 1.

Product	Width	Quantity
1	290	15
2	315	28
3	350	21
4	455	30

Table 5.2: Example order 2.

Product	Width	Quantity
1	330	9
2	360	7
3	385	12
4	415	11

Table 5.3: Example order 3.

Product	Width	Quantity
1	330	12
2	360	6
3	370	15
4	415	6
5	435	9

Table 5.4: Example order 4.

Product	Width	Quantity
1	330	8
2	360	16
3	380	12
4	430	7
5	490	14
6	530	16

Table 5.5: Parameters of the problems.

Pro. no.	B_{\max}	Δ	c_j	C_j	Nk_{\max}	$M_j \quad j = 1, \dots, P$
1	1850	100	0.1	1	5	$\in [0, 30]^4 \cap z^4$
2	1900	200	0.1	1	5	$\in [0, 15] \times [0, 12] \times [0, 9] \times [0, 6] \cap z^4$
3	2000	200	0.1	1	5	$\in [0, 15] \times [0, 12] \times [0, 9] \times [0, 6] \times [0, 6] \cap z^5$
4	2200	100	0.1	1	5	$\in [0, 15] \times [0, 12] \times [0, 8] \times [0, 7] \times [0, 4] \times [0, 2] \cap z^6$

Table 5.6: Results for the Trim-Loss Problem 1.

Algorithm	Objective function value	y	m	r
GMIN- α BB	19.6	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 3 \\ 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 2 & 0 & 1 & 0 \end{pmatrix}$
ILXPSO	19.6	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 14 \\ 3 \\ 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 2 & 0 & 1 & 0 \end{pmatrix}$
SDE	19.6	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 9 \\ 7 \\ 3 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 2 & 1 & 1 & 0 \\ 0 & 3 & 0 & 0 \\ 2 & 1 & 2 & 0 \end{pmatrix}$

Table 5.7: Results for the Trim-Loss Problem 2.

Algorithm	Objective function value	y	m	r
GMIN- α BB	8.6	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 11 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
ILXPSO	8.6	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 5 \\ 2 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \end{pmatrix}$
SDE	8.6	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 3 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 3 & 0 \\ 0 & 2 & 1 & 0 \\ 3 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 \end{pmatrix}$

Table 5.8: Results for the Trim-Loss Problem 3.

Algorithm	Objective function value	y	m	r
GMIN- α BB	10.3	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 15 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$
ILXPSO	11.5	$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 3 & 3 & 4 \\ 0 & 0 & 1 & 2 & 0 \\ 2 & 2 & 0 & 0 & 0 \end{pmatrix}$
SDE	10.3	$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 6 \\ 4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$

Table 5.9: Results for the Trim-Loss Problem 4.

Algorithm	Objective function value	y	m	r
GMIN- α BB	15.3	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 8 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
ILXPSO	16.3	$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 9 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
SDE	15.3	$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 8 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Table 5.10: Solution results for problem 1.

Cutting pattern no.	Cutting pattern generated	Trim loss
1	$(290 \times 1) + (315 \times 2) + (455 \times 2)$	20
2	$(315 \times 1) + (350 \times 3) + (455 \times 1)$	30
3	$(290 \times 2) + (315 \times 1) + (455 \times 2)$	45
Total trim loss		95

Table 5.11: Solution results for problem 2.

Cutting pattern no.	Cutting pattern generated	Trim loss
1	$(330 \times 1) + (385 \times 3) + (415 \times 1)$	0
2	$(330 \times 1) + (360 \times 2) + (415 \times 2)$	20
3	$(330 \times 3) + (360 \times 1) + (415 \times 1)$	135
Total trim loss		155

Table 5.12: Solution results for problem 3.

Cutting pattern no.	Cutting pattern generated	Trim loss
1	$(330 \times 2) + (360 \times 1) + (415 \times 1) + (435 \times 1)$	130
2	$(370 \times 4) + (435 \times 1)$	85
Total trim loss		215

Table 5.13: Solution results for problem 4.

Cutting pattern no.	Cutting pattern generated	Trim loss
1	$(330 \times 1) + (360 \times 2) + (530 \times 2)$	90
2	$(380 \times 2) + (430 \times 1) + (490 \times 2)$	30
Total trim loss		120

Table 5.14: Alternate optimal solutions of problem 1.

S.no.	m1	m2	m3	r11	r12	r13	r21	r22	r23	r31	r32	r33	r41	r42	r43
1	9	7	3	1		2	2	1	1		3		2	1	2
2	11	5	3	1	1		2		2		3	2	2	1	1
3	12	4	3	1		1	2	1			3	3	2	1	1
4	8	6	5		1	2	2	2		2		1	1	2	2
5	14	3	2	1		1	2				5	3	2		1

Table 5.15: Alternate optimal solutions of problem 2.

S.no.	m1	m2	m3	r11	r12	r13	r21	r22	r23	r31	r32	r33	r41	r42	r43
1	5	2	1	1	2	1	1		2	2	1		1	2	2
2	4	3	1	1	1	2		2	2	3			1	2	1

Table 5.16: Best, worst, mean fitness and standard deviation, average NFE, time and success rate for all problems.

Problem no.	Best	Worst	Mean	Std.	Average NFE	Time	% SR
1	19.60	20.60	19.75	3.57071e-01	205500	2.01	92
2	8.60	8.60	8.60	1.77636e-15	188000	0.90	100
3	10.30	11.30	11.05	4.33013e-01	1091500	8.35	89
4	15.30	16.30	15.85	4.97494e-01	1335000	11.80	93

Table 5.17: Comparison of SDE, ILXPSO and RCGA on average NFE and success rate for all problems.

Problem no.	NFE			% SR		
	SDE	ILXPSO	RCGA	SDE	ILXPSO	RCGA
1	205500	375100	600000	92	85	60
2	188000	375100	600000	100	80	55
3	1091500	900200	1050000	89	65	30
4	1335000	1200200	1920000	93	40	10

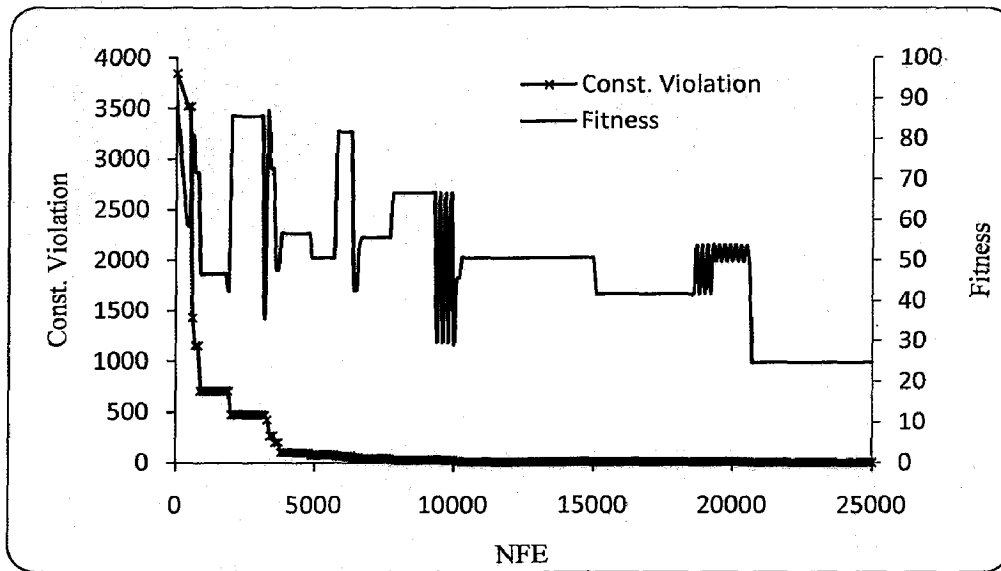


Figure 5.4: Plot of fitness and constraint violation versus NFE for problem 1.

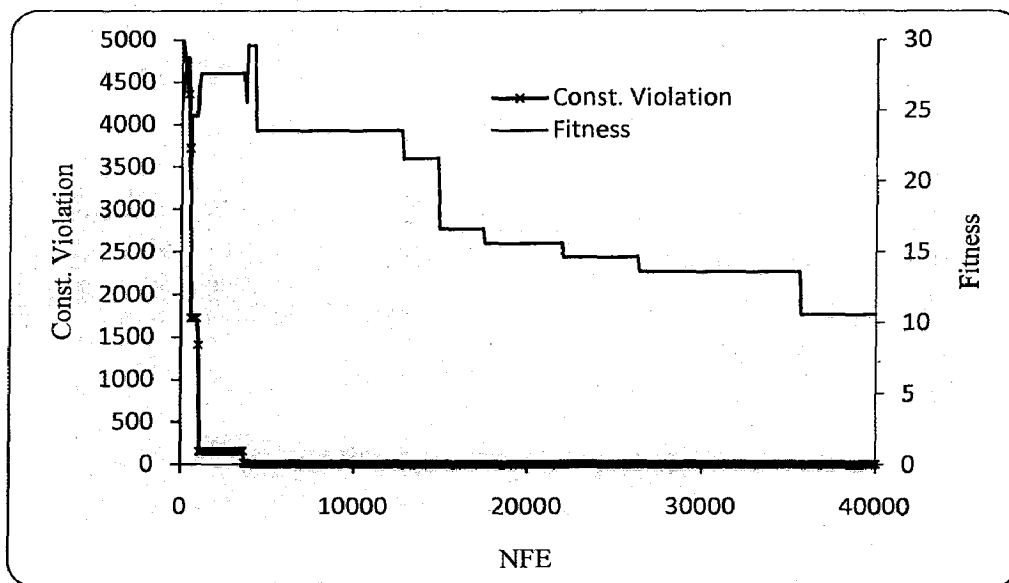


Figure 5.5: Plot of fitness and constraint violation versus NFE for problem 2.

Multi-level Image Thresholding

In the previous chapter, SDE algorithm was used for solving Trim Loss Problem (TLP) by considering the case of paper industry. TLP was modelled as a constrained global optimization problem having integer and binary variables. It was observed that SDE was able to solve such problem successfully. To further investigate the efficiency of SDE, in the present chapter another important area of that of image thresholding is considered.

Image thresholding is a challenging task in image processing field. Image segmentation is the process of dividing an image into different regions based on some criterion such that each region is homogeneous. Many efforts have already been made to propose universal and robust methods to handle a wide range of images. Among them two popular methods are (1) the maximum entropy thresholding criterion and (2) the approximation of normalized histogram of an image by a mixture of Gaussian distribution. Typically, finding the parameters of Gaussian distribution leads to a nonlinear optimization problem, for which obtaining the solution is computationally expensive and time-consuming. Such problems provide a useful platform for testing the efficiency of an optimization algorithm like that of SDE.

The chapter is structured as follows. Section 6.1 gives the introduction of image thresholding. Section 6.2 describes the techniques for image thresholding. Results are analyzed and discussed in section 6.3 and finally, chapter is concluded in section 6.4.

6.1 Introduction

Image thresholding is definitely one of the most popular segmentation approaches for extracting objects from the background, or for discriminating objects from objects that have distinct gray-levels. It is typically simple and computationally efficient. It is based on the assumption that the objects can be distinguished by their gray levels. The optimal threshold is the one that can separate different objects from each other or from the background to such an extent that a decision can be made without further processing (Gonzalez and Woods, 2002;

Otsu, 1979). The automatic fitting of this threshold is one of the main challenges of image segmentation. There are a lot of approaches classifying thresholding methods.

Sezgin and Sankur (2004) presented a survey of a variety of thresholding techniques. They labeled the methods according to the information they exploited, such as histogram shape, space measurement clustering, entropy, object attributes, spatial information and local gray-level surface. They also classified the thresholding techniques in terms of parametric and non parametric approaches.

Parametric thresholding methods exploit the first-order statistical characterization of the image to be segmented. An attempt to find an estimate of the parameters of the distribution that best fit the given histogram data is made by using the least-squares estimation method. Typically, it leads to a nonlinear optimization problem, its solution is computationally expensive and time consuming. Over the years, many researchers have proposed several algorithms to solve the objective function of Gaussian curve fitting for multi-level thresholding. Some instances of parametric thresholding methods in literature are: Weszka and Azriel (1979) proposed a parametric method where the gray-level distribution of each class is assumed to be a Gaussian distribution. Other examples of the use of parametric methods for thresholding are as follows: Snyder et al., (1990) presented an alternative method for fitting curves based on a heuristic method called tree annealing; Nakib et al. (2007, 2008) proposed a fast scheme for optimal thresholding using a simulated annealing algorithm; Zahara et al. (2005) proposed a hybrid Nelder-Mead Particle Swarm Optimization (NM-PSO) method. More recently a hybrid method based on Expectation Maximization (EM) and Particle Swarm Optimization (PSO+EM) is proposed by Fan and Lin (2007) and application of basic Differential Evolution (DE) for solving image segmentation problem is shown in Cuevas et al. (2010).

Non parametric approaches find the thresholds that separate the gray-level regions of an image in an optimal manner based on some discriminating criteria such as the between class variance, entropy and cross entropy. The most popular method is that of Otsu's (1979) in which optimal thresholds are selected by maximizing the *between class* variance. Sahoo et al. (1988) showed that the Otsu's method is one of the better threshold selection methods for real world images with regard to uniformity and shape measures. However, inefficient formulation of *between class variance* makes the method very time-consuming in multilevel threshold

selection. To overcome this problem, Liao et al. (2001) proposed a fast recursive algorithm called Fast Otsu's method, along with a look-up-table and implemented it for multilevel thresholding. Ye et al. (2008) proposed the use of PSO algorithm to optimize the Otsu's criterion. Kapur et al. (1985) proposed a method for gray-level picture thresholding by using the entropy of the histogram. Abutaleb (1989) proposed a 2-D maximum entropy thresholding method for separating the regions of image. Zhang and Liu (2006) adopted PSO to maximize the entropy for underwater image segmentation. Madhubanti and Amitava (2008) proposed a hybrid cooperative-comprehensive learning based PSO algorithm based on maximum entropy criterion. Li and Lee (1993) proposed a method which selects the threshold by minimizing the cross entropy between the original and segmented images. Yin (2007) developed a recursive programming technique to reduce the order of magnitude of computing the multilevel thresholds and further used the PSO algorithm to minimize the cross entropy. Most recently Horng (2010) proposed honey bee mating optimization based multilevel thresholding using maximum entropy.

All these metaheuristic based methods are shown to be efficient in solving the multi-level thresholding problem and provide better effectiveness than the other traditional methods (local search and deterministic methods). However, finding of threshold in multilevel thresholding is a time taking process which indicates that further improvement is needed to enhance the efficiency of existing methods while maintaining quality effectiveness. In the present chapter, the proposed SDE is employed for improving the thresholding techniques.

6.2 Methods Used for Image Thresholding

Several objective functions have been proposed in the literature devoted to thresholding (Sezgin and Sankur, 2004). These functions are determined generally from the histogram of the image I , denoted by $g(i)$, for $i = 0, \dots, L-1$, where $g(i)$ represents the number of pixels having the gray level i and L is the total number of gray levels. The normalized probability at gray level i is defined by the ratio $h(i) = (g(i)/N)$ where N is the total number of pixels. Among these functions, the objective function defined by Kapur et al. (1985) and Gaussian approximation of normalized histogram of the image are most popular. These methods are briefly described in the next subsections.

6.2.1 Gaussian Curve Fitting

A properly normalized multimodal histogram of an image I can be fitted with the sum of d probability density functions (pdf's) for finding the optimal thresholds for use in image segmentation (Snyder et al., 1990). The case where the Gaussian pdf's are used is defined by

$$p(x) = \sum_{i=1}^d P_i \exp \left[-\frac{(x - \mu_i)^2}{\sigma_i^2} \right] \quad (6.1)$$

where P_i is the amplitude of Gaussian pdf on μ_i , μ_i is the mean and σ_i^2 is the variance of mode i and d is number of Gaussians used to approximate the original histogram corresponding to the number of the segmentation classes. A pdf model must be fitted to the histogram data, typically by using the maximum likelihood or mean-squared error approach, in order to locate the optimal threshold. Our goal is to find a set of parameters, Θ , that minimizes the fitting error J , given by the following expression (Nakib et al., 2007, 2008):

$$\text{Minimize } J = \frac{\sum_i |h(i) - p(\Theta, x_i)|}{\sum_i h(i)} \quad (6.2)$$

and i ranges over the bins in the measured histogram. Here, J is the objective function to be minimized with respect to Θ , a set of parameters defining the Gaussian pdf's and the probabilities, is given by

$$\Theta = \{P_i, \mu_i, \sigma_i; i = 1, 2, \dots, d\}$$

The standard process of setting the partial derivatives to zero results in a set of non-linear coupled equations, the system usually being solved through numerical techniques.

After fitting the multimodal histogram, the optimal threshold could be determined by minimizing the overall probability of error, for two adjacent Gaussian pdf's, given by

$$e(t_i) = P_i \int_{-\infty}^{t_i} p_i(x) dx + P_{i+1} \int_{t_i}^{\infty} p_{i+1}(x) dx, \quad i = 1, 2, \dots, d-1 \quad (6.3)$$

with respect to the threshold t_i , where $p_i(x)$ is the i th pdf (Gonzalez and Woods, 2002). Then the overall probability to minimize is:

$$E(T) = \sum_{i=1}^{d-1} e(t_i) \quad (6.4)$$

where T is the vector of thresholds: $0 < t_1 < t_2 < \dots < t_{d-1} < L - 1$. In this case L is equal to 256. To find the thresholds values for which this error is minimal requires differentiating $e(t_i)$ with respect to t_i (using Leibniz's rule) and equating the result to zero. The obtained equation is as follows:

$$P_i \cdot p_i(t_i) = P_{i+1} \cdot p_{i+1}(t_i) \quad (6.5)$$

This equation is solved for t_i to find the optimum threshold. Using equation (6.1) in the general solution of equation (6.5) results in the following solution for the threshold t_i :

$$At_i^2 + Bt_i + C = 0 \quad (6.6)$$

where

$$\begin{aligned} A &= \sigma_i^2 - \sigma_{i+1}^2 \\ B &= 2(\mu_i \sigma_{i+1}^2 - \mu_{i+1} \sigma_i^2) \\ C &= \sigma_i^2 \mu_{i+1}^2 - \sigma_{i+1}^2 \mu_i^2 + 4\sigma_i^2 \sigma_{i+1}^2 \ln(\sigma_{i+1} P_i / \sigma_i P_{i+1}) \end{aligned} \quad (6.7)$$

Out of the two possible solutions of above quadratic equation, only one will be feasible.

The optimal thresholding methods search for the thresholds such that the segment classes on the histogram satisfying the desired property. This is performed by minimizing or maximizing an objective function which uses the selected thresholds as parameters. This function is often referred to as objective function or criterion measure.

6.2.2 Entropy Criterion

The original algorithm was developed for bi-level thresholding and was later extended for multiple levels. The bi-level algorithm can be described as follows:

Objective is to maximize the fitness function:

$$f(t) = H_0 + H_1 \quad (6.8)$$

where

$$\begin{aligned} H_0 &= -\sum_{i=0}^{t-1} \frac{h(i)}{\omega_0} \ln \frac{h(i)}{\omega_0}, & \omega_0 &= \sum_{i=0}^{t-1} h(i) \\ H_1 &= -\sum_{i=t}^{L-1} \frac{h(i)}{\omega_1} \ln \frac{h(i)}{\omega_1}, & \omega_1 &= \sum_{i=t}^{L-1} h(i) \end{aligned}$$

The optimum threshold is t which maximizes $f(t)$. The multilevel thresholding problem consists in selecting the $(d-1)$ - dimensional vector $T = \{(t_1, t_2, \dots, t_{d-1}) : t_1 < t_2 < \dots < t_{d-1}\}$ of threshold which optimizes the objective function $f(T)$:

$$f(T) = H_0 + H_1 + H_2 \dots + H_{d-1} \quad (6.9)$$

where

$$\begin{aligned} H_0 &= -\sum_{i=0}^{t_1-1} \frac{h(i)}{\omega_0} \ln \frac{h(i)}{\omega_0}, & \omega_0 &= \sum_{i=0}^{t_1-1} h(i) \\ H_1 &= -\sum_{i=t_1}^{t_2-1} \frac{h(i)}{\omega_1} \ln \frac{h(i)}{\omega_1}, & \omega_1 &= \sum_{i=t_1}^{t_2-1} h(i) \\ H_2 &= -\sum_{i=t_2}^{t_3-1} \frac{h(i)}{\omega_2} \ln \frac{h(i)}{\omega_2}, & \omega_2 &= \sum_{i=t_2}^{t_3-1} h(i) \\ H_{d-1} &= -\sum_{i=t_{d-1}}^{L-1} \frac{h(i)}{\omega_c} \ln \frac{h(i)}{\omega_c}, & \omega_c &= \sum_{i=t_{d-1}}^{L-1} h(i) \end{aligned}$$

This entropy criterion based measure tries to achieve more and more centralized distribution for each histogram based segmentation region in the image.

6.3 Results and Discussions

This section evaluates the performance of SDE with some other algorithms while implementing Gaussian curve fitting for multi-level thresholding and entropy criterion. These variants are referred to as SDE_Gaus and SDE_Entropy. The test images "Lena" and "Pepper" are of size 512×512 each and "Cameraman" is of size 256×256 pixels with 8 bit gray-levels, taken under natural lighting without the support of any special light source. Test images and their respective normalized histograms are given in Figure 6.1. The coding is done in Matlab 7. The stopping criterion used is the maximum number of iteration. SDE has 4 parameters that must be well fitted. Preliminary testing has been done for the purpose of getting suitable values of these parameters and the fine tuned results, which are different from the parameters given in chapter 2, are as follows:

- Population size NP= (10*dimension),
- Scaling factor = crossover rate = 0.25.

➤ Both SDE_Gaus and SDE_Entropy are halted after 200 generations.

To judge the quality of segmented image, uniformity measure (Sathya and Kayalvizhi, 2010) is utilized which has also been extensively used in several literatures. This uniformity measure is given as:

$$U = 1 - 2 \times (nc - 1) \times \frac{\sum_{j=0}^{nc-1} \sum_{i \in R_j} (f_i - \mu_j)^2}{N \times (f_{max} - f_{min})}$$

where, nc denotes number of classes, R_j denotes the j th segmented region, f_i indicates the gray level of the pixel i , μ_j implies gray level of pixels in j th region, N denotes the total number of pixels in the given image, f_{max} gives the maximum gray level of pixels in the given image and f_{min} gives minimum gray level of pixels in the given image. The value of the uniformity measure, U , should be a positive fraction lying between 0 and 1. A higher value of U indicates that there is better uniformity in the thresholded image, depicting better quality of thresholding and vice versa.

The experimental results obtained by SDE_Gaus are listed in Table 6.1, in terms of number of classes, the threshold values, parameter values and CPU time. To measure the performance of the SDE_Gaus algorithm, experimental results of previous multilevel thresholding methods based on GA and PSO (Sathya and Kayalvizhi, 2010) are included in Table 6.2. The quality of the thresholded images is evaluated by the uniformity measure which is a broadly used criterion. It can be observed from Table 6.2 that the results of the SDE_Gaus method have higher uniformity than those of the other two multilevel thresholding methods.

Results of SDE_Entropy method are given in Table 6.3. It includes number of thresholds, threshold values, fitness, time and uniformity measure. This table provides quantitative standard for evaluating several aspect. The computation times of SDE_Entropy algorithm are relatively irrelevant to the size of image and the number of the thresholds. Furthermore, as the number of thresholds increase, the uniformity and the fitness value are enhanced. To provide the visual comparison between SDE_Gaus and SDE_Entropy the results of multi-level thresholding are illustrated by indicating the derived thresholds on the histograms and displaying the corresponding thresholded images (Figures 6.2, 6.3 and 6.4). As can be seen (Figures 2(e, g), 3(e, g) and 4(e, g)), SDE_Gaus method tend to produce thresholds at the valleys of the fitted histograms. The quality of segmented images obtained by SDE_Gauss is

superior to the quality of those obtained by SDE_Entropy. Comparison results of the SDE_Gaus method to SDE_Entropy demonstrate that SDE_Gaus offers higher quality in visualization. Not surprisingly, SDE_Gaus incurs higher computation time than the SDE_Entropy method, since curve fitting needs to search for the optimum values of more parameters (Tables 6.1 and 6.3).

Finally, SDE_Entropy method is compared with four other algorithms (Horn, 2010): (1) Maximum entropy based honey bee mating optimization thresholding (MEHBMOT) method, (2) Particle swarm optimization (PSO), (3) Hybrid cooperative-comprehensive learning based PSO algorithm (HCOCLPSO) and (4) Fast Otsu's method. Results for the comparison with the other algorithms are given in Tables 6.4-6.6. The CPU time for SDE_Entropy algorithm is given in Table 6.3 but is not compared with other algorithms, as these are executed on different platform. It can also be seen from Tables 6.5 and 6.6 that the proposed SDE_Entropy algorithm could achieve significantly better segmentation results as demonstrated by its higher values of U and fitness values in most of the cases, compared to other methods.

6.4 Summary

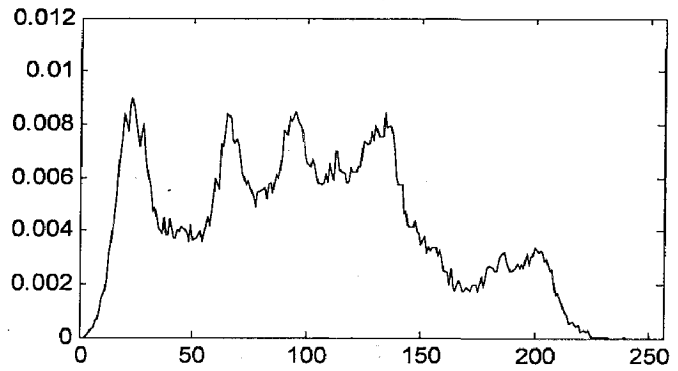
In this chapter, the efficiency of SDE is validated on the global optimization problem of image thresholding. The objects and background components within the image are assumed to fit into Gaussian distributions exhibiting non-equal mean and standard deviation. The histogram is thus approximated by a mix of Gaussian probability functions. SDE is used to estimate the parameters for the mixing density function as it seeks to get a minimum error between the density function and the original histogram. It is referred to as SDE_Gauss. On the other hand, SDE_Entropy has used the objective function of maximum entropy.

Conclusions that can be drawn at the end of this chapter are as follows:

- SDE algorithm can be successfully applied to the area of image thresholding which is an important part of image segmentation.
- Experimental results show that SDE_Entropy produces more satisfactory results in comparison to SDE_Gauss, in terms of computational efficiency. These results are quite expected because the segmentation results depend heavily on the objective function selected. However, in terms of picture quality, Gaussian method is preferred over the entropy method.



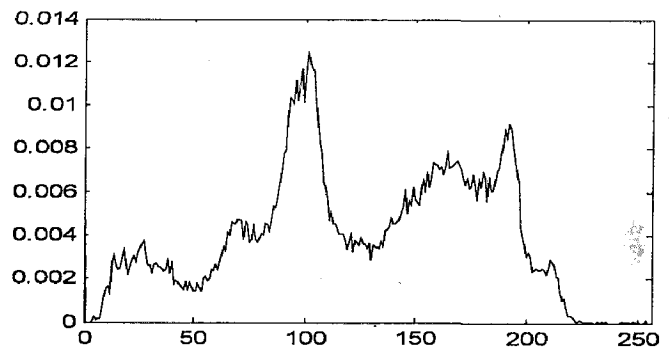
(a)



(d)



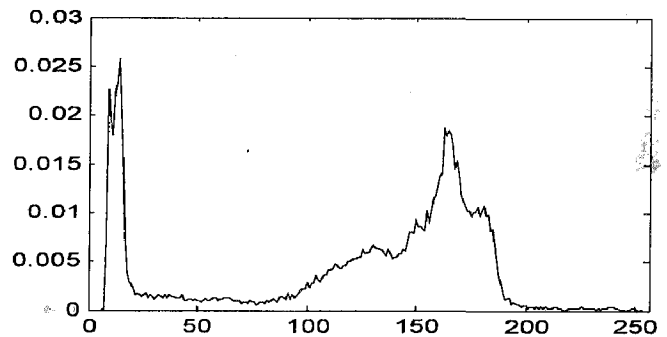
(b)



(e)



(c)

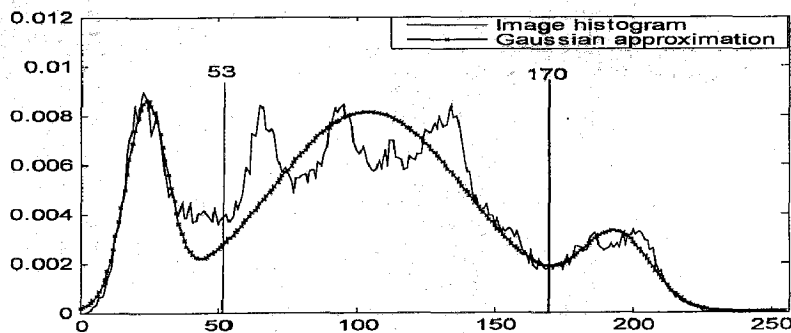


(f)

Figure 6.1: Test images and their normalized histograms. (a) Lena, (b) Pepper, (c) Cameraman, (d) histogram of Lena image, (e) histogram of Pepper image and (f) histogram of Cameraman image.



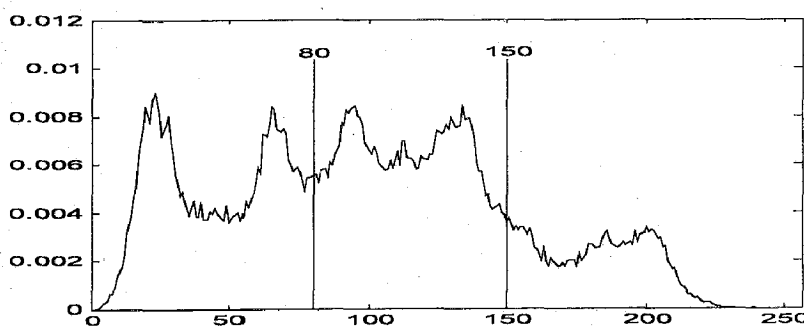
(a)



(e)



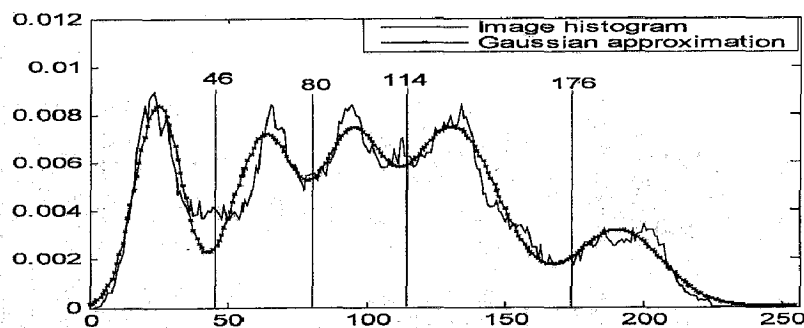
(b)



(f)



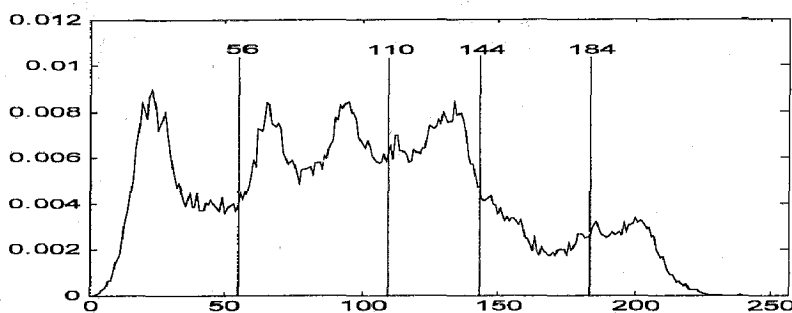
(c)



(g)



(d)



(h)

Figure 6.2: Results of Lena image (a) segmented image with three classes by curve, (b) segmented image with three classes by entropy, (c) segmented image with five classes by curve, (d) segmented image with five classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d).



(a)



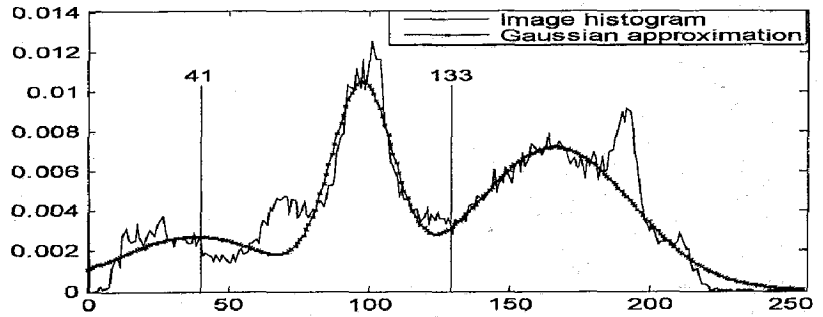
(b)



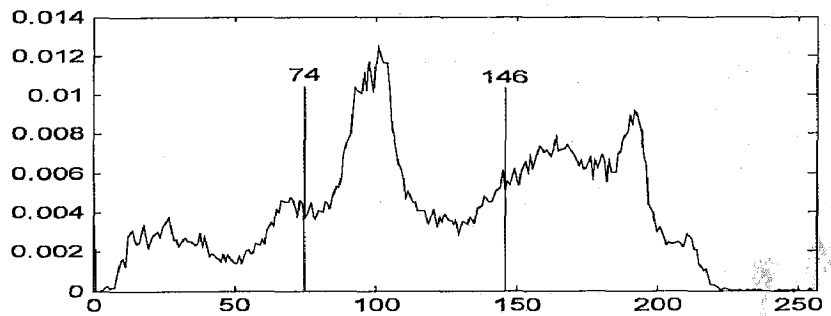
(c)



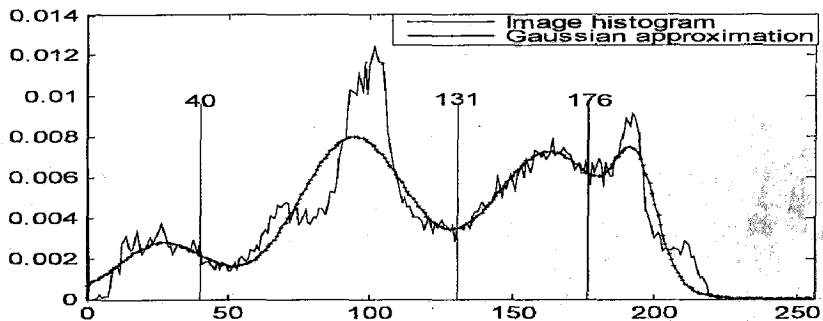
(d)



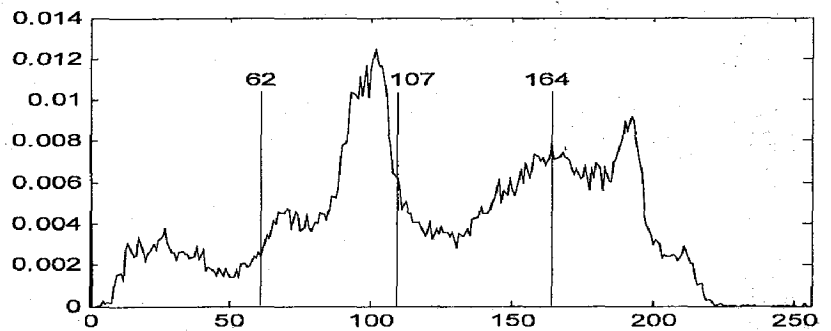
(e)



(f)



(g)



(h)

Figure 6.3: Results of Pepper image (a) segmented image with three classes by curve, (b) segmented image with three classes by entropy, (c) segmented image with four classes by curve, (d) segmented image with four classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d).

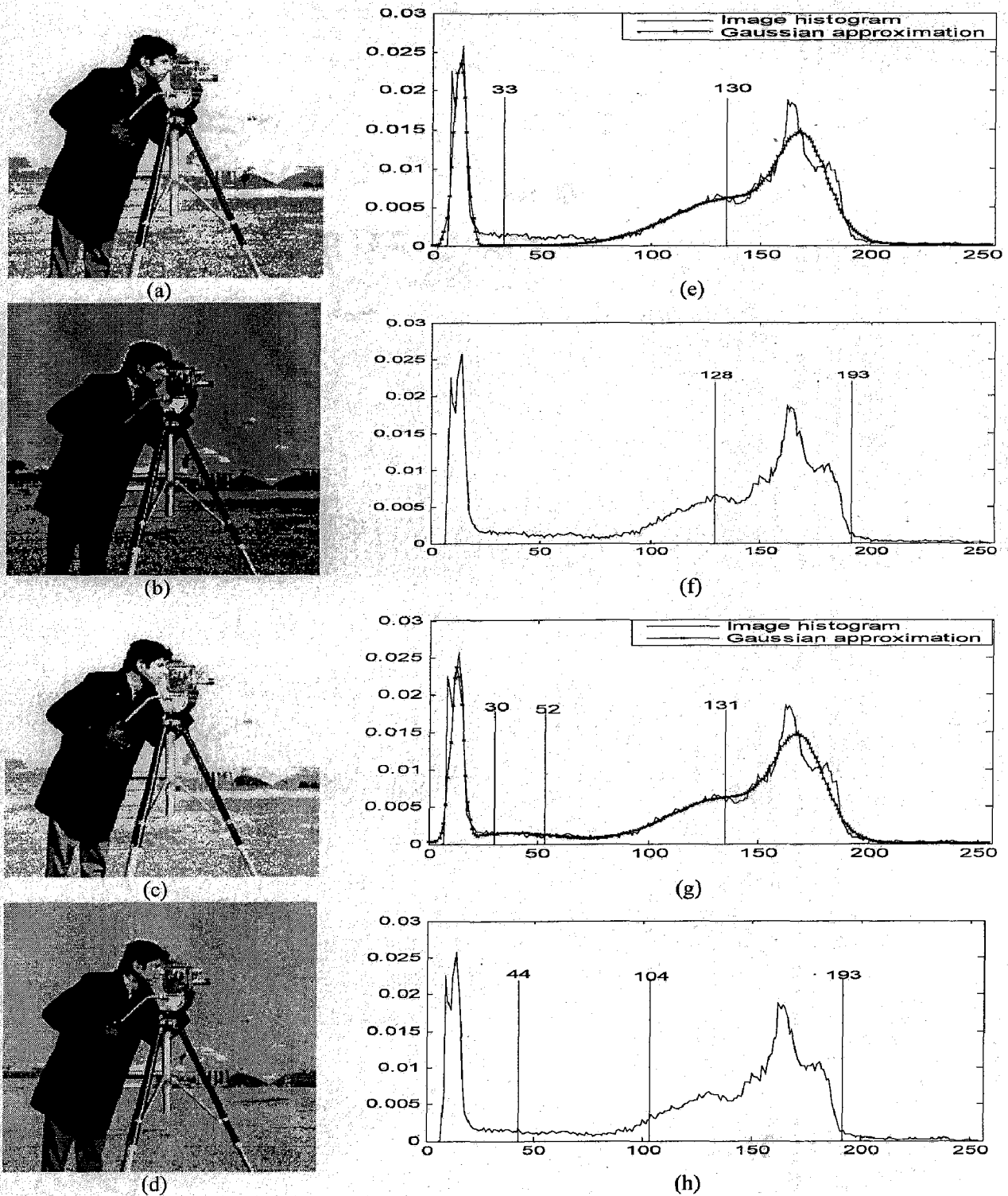


Figure 6.4: Results of Cameraman image (a) segmented image with three classes by curve, (b) segmented image with three classes by entropy, (c) segmented image with four classes by curve, (d) segmented image with four classes by entropy, (e) fitted histogram and threshold of (a), (f) threshold of (b), (g) fitted histogram and threshold of (c) and (h) threshold of (d).

Table 6.1: Results obtained by SDE_Gaus for images given in Figure 6.1.

Image	Number of classes	Parameters of Gaussian approximations	Time (Sec)	Threshold
Lena	3	$P(0.0079, 0.0081, 0.0029)$ $\mu(24, 104, 194)$ $\sigma(10.7663, 49.9796, 17.4342)$	5.8968	53, 170
	5	$P(0.0084, 0.0070, 0.0066, 0.0074, 0.0032)$ $\mu(25, 63, 94, 131, 191)$ $\sigma(12.1269, 15.6615, 16.0053, 23.8994, 22.1594)$	12.8077	46, 80, 114, 176
Pepper	3	$P(0.0027, 0.0079, 0.0078)$ $\mu(28, 96, 170)$ $\sigma(21.2210, 29.5961, 30.0024)$	5.7408	41, 133
	4	$P(0.0027, 0.0079, 0.0072, 0.0048)$ $\mu(27, 94, 162, 193)$ $\sigma(22.9351, 27.2802, 29.3592, 10.2470)$	8.6113	40, 131, 176
Cameraman	3	$P(0.0235, 0.0061, 0.0114)$ $\mu(13, 137, 169)$ $\sigma(4.3845, 39.2224, 15.4275)$	5.7876	33, 130
	4	$P(0.0231, 0.0014, 0.0061, 0.0118)$ $\mu(13, 39, 136, 169)$ $\sigma(3.9712, 29.6954, 37.0420, 15.5644)$	8.8609	30, 52, 131

Table 6.2: Comparison of SDE_Gaus with basic PSO and GA.

Image	No. of classes	Threshold			Uniformity measure		
		PSO	GA	SDE_Gaus	PSO	GA	SDE_Gaus
Lena	3	61,166	53,178	53, 170	0.9597	0.9490	0.9533
	5	46,84,119,186	46,77,115,186	46, 80, 114, 176	0.9774	0.9758	0.9807
Pepper	3	42, 135	38, 136	41, 133	0.9740	0.9738	0.9741
	4	38, 141, 178	38, 134, 181	40, 131, 176	0.9749	0.9746	0.9752
Camera man	3	30, 135	30, 142	33, 130	0.9752	0.9744	0.9764
	4	28, 48, 145	28, 50, 145	30, 52, 131	0.9735	0.9732	0.9736

Table 6.3: Experimental results of SDE_Entropy for images given in Figure 6.1.

Image	Number of threshold	Threshold	Fitness	Time (s)	Uniformity
Lena	2	80, 150	12.6990	1.17	0.9716
	3	60, 109,162	15.7667	1.19	0.9790
	4	56,110, 144, 184	18.5106	1.23	0.9777
	5	47, 78, 112, 137, 183	21.2446	1.28	0.9843
Pepper	2	74, 146	12.6348	1.46	0.9778
	3	62, 107,164	15.6896	1.62	0.9813
	4	57, 88, 143, 194	18.5406	1.87	0.9869
	5	45,77,114, 155, 195	21.2910	1.98	0.9887
Cameraman	2	128, 193	12.1688	1.98	0.9363
	3	44, 104,193	15.2274	2.01	0.9625
	4	42,95, 156, 196	18.2461	2.34	0.9807
	5	40, 84, 119, 154, 201	21.0971	2.62	0.9857

Table 6.4: Comparison of algorithms in terms of thresholds.

Image	Number of thre.	MEHBMOT	PSO	HCOCLPSO	Fast Otsu's method	SDE_Entropy
Lena	2	80, 150	80, 150	80, 150	77, 145	80, 150
	3	60, 109, 160	60, 109, 160	60, 109, 160	56, 106, 159	60, 109, 162
	4	56, 100, 144, 182	56, 100, 144, 182	56, 100, 144, 182	74, 112, 144, 179	56, 110, 144, 184
	5	44, 80, 115, 150, 185	43, 79, 114, 150, 185	46, 83, 118, 153, 187	45, 79, 109, 138, 173	47, 78, 112, 137, 183
Pepper	2	74, 146	74, 146	74, 146	67, 134	74, 146
	3	61, 112, 164	72, 135, 193	61, 112, 164	61, 117, 165	62, 107, 164
	4	57, 104, 148, 194	58, 105, 148, 194	57, 104, 148, 194	46, 85, 125, 168	57, 88, 143, 194
	5	42, 77, 113, 153, 194	43, 77, 113, 153, 194	42, 77, 114, 154, 194	41, 77, 111, 145, 176	45, 77, 114, 155, 195
Cameraman	2	128, 193	128, 193	128, 193	69, 143	128, 193
	3	44, 104, 193	44, 104, 193	44, 104, 193	58, 118, 155	44, 104, 193
	4	44, 97, 146, 197	44, 97, 146, 197	44, 97, 146, 197	41, 94, 139, 169	42, 95, 156, 196
	5	40, 84, 119, 155, 197	40, 85, 120, 155, 197	40, 84, 119, 155, 197	35, 81, 121, 148, 172	40, 84, 119, 154, 201

Table 6.5: Comparison of algorithms in terms of fitness corresponding to threshold given in Table 6.4.

Image	Number of threshold	MEHBMOT	PSO	HCOCLPSO	Fast Otsu's method	SDE_Entropy
Lena	2	12.6990	12.6990	12.6990	12.6920	12.6990
	3	15.7658	15.7658	15.7658	15.7591	15.7667
	4	18.5875	18.5875	18.5875	18.5084	18.5106
	5	21.2425	21.2400	21.2358	21.1301	21.2446
Pepper	2	12.6348	12.6348	12.6348	12.6044	12.6348
	3	15.6892	15.5772	15.6892	15.6829	15.6896
	4	18.5397	18.5395	18.5397	18.4699	18.5406
	5	21.2830	21.2806	21.2804	21.1905	21.2910
Cameraman	2	12.1688	12.1688	12.1688	11.3367	12.1688
	3	15.2274	15.2274	15.2274	14.3235	15.2274
	4	18.3955	18.3955	18.3955	17.2633	18.2461
	5	21.0691	21.0680	21.0691	19.8241	21.0971

Table 6.6: Comparison of algorithms in terms uniformity measure.

Image	Number of threshold	MEHBMOT	PSO	HCOCLPSO	Fast Otsu's method	SDE_Entropy
Lena	2	0.9716	0.9716	0.9716	0.9702	0.9716
	3	0.9789	0.9789	0.9789	0.9760	0.9790
	4	0.9792	0.9792	0.9792	0.9773	0.9777
	5	0.9840	0.9839	0.9838	0.9827	0.9843
Pepper	2	0.9778	0.9778	0.9778	0.9752	0.9778
	3	0.9805	0.9792	0.9805	0.9802	0.9813
	4	0.9854	0.9881	0.9854	0.9838	0.9869
	5	0.9881	0.9879	0.9876	0.9853	0.9887
Cameraman	2	0.9363	0.9363	0.9363	0.9243	0.9363
	3	0.9625	0.9625	0.9625	0.9560	0.9625
	4	0.9825	0.9825	0.9825	0.9759	0.9807
	5	0.9852	0.9851	0.9852	0.9816	0.9857

Conclusions and Future Research

This chapter provides a summary of the research presented in this thesis. The general conclusions, including encountered challenges and limitations, are discussed, followed by a description of future research work. It consists of two sections; section 7.1 brings out the overall conclusions of the research work carried out in this thesis and section 7.2 suggests some research directions and possible extensions of the work presented in the thesis.

7.1 Conclusions

The objective of this research work was to design general purpose algorithms for solving global optimization problems (both unconstrained and constrained) and to apply them for solving multiple objectives and real life optimization problems. The goal was to create a solver that could easily and effectively deal with a wide assortment of problems without imposing any restrictions on the problems. DE has been taken as the base algorithm on which enhancements have been done to develop efficient and robust optimization algorithms.

Three algorithms namely CDE, MSDE and SDE have been proposed in the second chapter for solving unconstrained global optimization problems. These algorithms are rigorously analyzed and are compared with basic DE algorithm and also with some of its enhanced variants with the help of various performance measures including a non parametric statistical analysis. For the purpose of analysis, 25 classical benchmark problems and 7 nontraditional (shifted) functions were taken.

It was observed that all the three proposed variants performed satisfactorily. However, MSDE and SDE performed better than CDE and were therefore compared with JADE, SaDE and jDE, some other advanced versions of DE. Here it was observed that MSDE and SDE perform at par with JADE, while they were better than SaDE and jDE. The results are also justified with the help of statistical analysis.

Since MSDE and SDE performed better than CDE for solving unconstrained optimization problems MSDE and SDE algorithms were suitably modified for constrained optimization problems. Pareto ranking approach is used for constraint handling. One of the salient features of this approach is that aside from the standard parameters required for MSDE and SDE no additional user input is required as in penalty approach. Each algorithm was tested on a set of twenty four test problems. The results for both algorithms were found to be extremely promising, with SDE showing a slight dominance over the MSDE algorithm. The results of proposed algorithms were also compared with six other versions of constrained DE available in literature. These versions are ZRDE, jDE-2, MDE, MDE-1, ϵ DE and SaDE. A statistical comparison of the algorithms has shown that all the algorithms perform at par with each other in terms of average number of NFE except for jDE-2 and SaDE which did not perform as well as the other algorithms.

The efficiency of SDE for solving constrained and unconstrained benchmark and its simple algorithmic structure was the motivation to extend and modify it for solving multi-objective optimization problems (MOPs). The modified SDE called as MO-SDE for solving MOPs is described in Chapter 4. The performance of MO-SDE was validated on a set of nine unconstrained MOPs. Its comparison with some recently modified versions of differential evolution and some other Multi-Objective Evolutionary Algorithms (MOEAs) once again showed the efficiency of the proposed SDE algorithm.

The superior performance of SDE for solving unconstrained/ constrained and MOPs was an encouragement to further apply it for solving real life problems. For this purpose two real problems were considered: (1) Trim Loss Problem (TLP) and (2) Image thresholding which are described in chapter 5 and 6 respectively.

The first problem (TLP) is a popular problem arising in various industries like glass, ceramic and textile etc. The particular industry that has been taken as an example here is the TLP arising in paper industry. Mathematically, it has a complex non linear/ non convex structure having integer as well as binary variables.

The second problem is of image thresholding, which is a complex and difficult task in image processing field. The SDE algorithm was applied to solve the image thresholding problem was tested on three test images.

For both the real life problems, SDE once again showed its competence by finding a quality solution with a reasonably good convergence rate.

An overall conclusion that can be drawn at the end of this study is that the suggested modifications helped in improving the performance of basic DE in terms of solution quality as well as convergence rate. SDE, which was the simplest of the three algorithms, emerged as a clear winner. However, it is worth mentioning that some cases were encountered where the proposed algorithms were not able to achieve the solution. For example, in case of unconstrained benchmark problems for Rosenbrock (f_5) and Rastrigin (f_9) function the proposed algorithms did not perform well under the given parameter settings. In case of constrained problems MSDE and SDE were not able to give a successful performance for three test cases; g03, g20 and g22. This shows that some further investigation is needed to make these algorithms compatible for solving all type of problems.

The present work can be extended in several ways. Some future directions and suggestions regarding this work are given in the next section.

7.2 Future Research

Research is an iterative and everlasting process. The work presented in this thesis is not an exception. Based on the experiments, several suggestions can be implemented for future work. A few of them are listed below:

1. Population size plays an important role in algorithms success. Large populations give the more opportunity to find the desired solution since it can evaluate more thoroughly the feasible space at the expense of computational time. Small population tends to converge to a solution faster than the larger populations, but is more susceptible to local minima. So population can be taken in an adaptive manner instead of taking a fixed size population.
2. An extensive empirical analysis of numerical results has been done in the present work. It would be interesting to do research on the theoretical analysis of the operators used in the proposed algorithms of the thesis.
3. Fine tuning of parameters of proposed algorithms can be replaced with some suitable adaptive technique. Effects of adding some local search technique can also be observed.

4. In CDE, Cauchy mutation may be applied adaptively.
5. MSDE has taken only two strategies in the present study; it may be extended with more strategies.
6. SDE has been evaluated on unconstrained MOPs benchmark functions. It may be evaluated for solving constrained MOPs and also can be applied for solving multi-objective real life application problems.
7. The algorithms developed in this thesis may be hybridized among themselves or with other nature inspired algorithms like genetic algorithm, particle swarm optimization, ant colony optimization, bacterial foraging and honey bee mating optimization etc.
8. Parallelization of these techniques is another viable option to increase overall performance and more easily handle larger data sets.

Bibliography

- Abbass, H. (2002), The Self-Adaptive Pareto Differential Evolution Algorithm, In Proc. of IEEE Congress on Evolutionary Computation 2002 (CEC 2002), pp. 831–836.
- Abbass, H.A., Sarker, R., Newton, C. (2001), PDE: A Pareto-Frontier Differential Evolution Approach for Multi-Objective Optimization Problems, In Proc. of IEEE Congress on Evolutionary Computation 2001 (CEC 2001), pp. 971–978.
- Abutaleb, A.S. (1989), Automatic Thresholding of Gray-Level Pictures Using Two-Dimensional Entropy, Computer Vision Graphics Image Process, Vol. 47, pp. 22–32.
- Adeyemo, J.A., Otiño, F.A.O. (2009), Multi-Objective Differential Evolution for Solving Engineering Problems, Journal of Applied Sciences, Vol. 9(20), pp. 3652–3661.
- Adjiman, C.S., Androulakis, I.P., Floudas, C.A. (2000), Global Optimization of Mixed Integer Nonlinear Problems, AIChE Journal, Vol. 46, pp. 1769–1797.
- Ali, M.M. (2007), Differential Evolution with Preferential Crossover, European Journal of Operational Research, Vol. 181, pp. 1137–1147.
- Ali, M.M., Bagdadi, Z.K. (2009), A Local Exploration-Based Differential Evolution Algorithm for Constrained Global Optimization, Applied Mathematics and Computation, Vol. 208, pp. 31–48.
- Alvares-Valdés, A., Parajón, A., Tamarit, J.M. (2002), A Tabu search algorithm for large scale guillotine (un)constrained two-dimensional cutting problems, Computers and Operations Research, Vol. 29, pp. 925–947.
- Angira, R., Babu, B.V. (2006), Optimization of process synthesis and design problems: A modified differential approach, Chemical Engineering Science, Vol. 61, pp. 4707–4721.
- Babu, B.V., Angira, R. (2006), Modified differential evolution (MDE) for optimization of non-linear chemical processes, Computer and Chemical Engineering, Vol. 30, pp. 989–1002.
- Babu, B.V., Shaik, M.A. (2007), Differential Evolution Strategies for Optimal Design of Shell-and-Tube Heat Exchangers, Chemical Engineering Science, Vol. 62, pp. 3720–3739.
- Bader, J., Zitzler, E. (2008), HypE: An algorithm for fast hypervolume-Based many-objective optimization, TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.
- Beasley, J.E. (2004), A population heuristic for constrained two-dimensional non-guillotine cutting, European Journal of Operational Research, Vol. 156, pp. 601–627.
- Becerra, R.L., Coello, A.C. (2006), Cultural Differential Evolution for Constrained Optimization, Computer Methods in Applied Mechanics Engineering, Vol. 195, pp. 4303–4322.
- Birru, H.K., Chellapilla, K., Rao, S.S. (1999), Local search operators in fast evolutionary programming, In Proc. of IEEE Congress on Evolutionary Computation 1999 (CEC 1999), pp.1506–1513.

- Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V. (2006a), Self Adapting Control parameters in Differential Evolution: A comparative study of numerical benchmark problems, *IEEE Transactions on Evolutionary Computation*, Vol. 10, pp. 646 – 657.
- Brest, J., Zumer, V., Maucec, M.S. (2006b), Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization, In *Proc. of IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*, pp. 215 - 222.
- Brest, J., Boskovic, B., Greiner, S., Zumer, V., Maucec, M.S. (2007), Performance comparison of self-adaptive and adaptive differential evolution algorithms, *Soft Computing*, Vol. 11, pp. 617–629.
- Brest, J., Zamuda, A., Boskovic, B., Maucec, M.S., Zumer, V. (2009), Dynamic Optimization using self-adaptive Differential Evolution, In *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 415–422.
- Cai, Z., Gong, W., Ling, C.X., Zhang, H. (2011), A clustering-based differential evolution for global optimization, *Applied Soft Computing*, Vol. 11, pp. 1363–1379.
- Caponio, A., Neri, F., Tirronen, V. (2009), Superfit control adaptation in memetic differential evolution frameworks, *Soft Computing*, Vol. 13, pp. 811–831.
- Chakraborty, U.K. (Ed.) (2008), *Advances in Differential Evolution*, Springer-Verlag, Heidelberg.
- Chiang, C-L., Su, C-T., Wang, F-S. (2002), Augmented Lagrangian Method for Evolutionary Optimization of Mixed-Integer Nonlinear Constrained Problems, *International Mathematical Journal*, Vol. 2, pp. 119 - 154.
- Chiang, C.W., Lee, W.P., Heh, J-S. (2010), A 2-Opt based differential evolution for global optimization, *Applied Soft Computing*, Vol. 10, pp. 1200–1207.
- Chiou, J-P, Wang, F-S. (1999), Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Applications to a Fed-Batch fermentation Process, *Computers and Chemical Engineering*, Vol. 23, pp. 1277–1291.
- Coelho, L.S., Krohling, R.A. (2005), Predictive controller tuning using modified particle swarm optimization based on Cauchy and Gaussian distributions, *Advances in intelligent and soft computing*, Vol. 32, pp. 287–298.
- Coello Coello, C.A. (1996), *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*, Ph.D. Thesis, Department of Computer Science, Tulane University, New Orleans, LA.
- Coello Coello, C.A. (2000), An updated survey of GA based multiobjective optimization techniques. *ACM Computing survey*, Vol. 32, pp. 109– 143.
- Coello Coello, C.A.. (2002), Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art, *Computer Methods Applied Mechanics and Engineering*, Vol. 191, pp. 1245–1287.
- Correia, M.H., Oliveira, J.F., Ferreira, J.S. (2004), Reel and sheet cutting at a paper mill, *Computers and Chemical Engineering*, Vol. 31, pp. 1223–1243.
- Coverdale, I., Wharton, F. (1976), An Improved Heuristic Procedure for a Nonlinear Cutting Stock Problem, *Management Science*, Vol. 23, pp. 78–86.

- Cuevas, E., Zaldivar, D., Cisneros, Pérez, M. (2010), A Novel Multi-threshold Segmentation Approach Based on Differential Evolution Optimization, *Expert Systems with Applications*, Vol. 37, pp. 265–5271.
- Darwin, C. (1859), *On the Origin of Species*, John Murray, Sixth (Ed.), November 1859, Online available at: <http://www.gutenberg.org/etext/1228>.
- Das, S., Konar, A. (2006), Design of two dimensional IIR filters with modern search heuristics: A comparative study, *International Journal of Computational and Intelligence Application*, Vol. 6, pp. 329–355.
- Das, S., Konar, A. (2009), Automatic Image Pixel Clustering with an Improved Differential Evolution, *Applied Soft Computing*, Vol. 9, pp. 226–236.
- Das, S., Suganthan, P.N. (2010), Differential Evolution: A Survey of the State-of-the-Art, *IEEE Transactions on Evolutionary Computation*, DOI- 10.1109/TEVC.2010.2059031.
- Das, S., Abraham, A., Konar, A. (2008), Adaptive clustering using improved differential evolution algorithm, *IEEE Transactions on System, Man and Cybernetics*, Vol. 38, pp. 218–237.
- Das, S., Konar, A., Chakraborty, U.K. (2005), Two Improved Differential Evolution Schemes for Faster Global Search, In *Proc. of GECCO*, Washington D.C., pp. 991–998.
- Deb, K. (2001), *Multi-objective Optimization Using Evolutionary Algorithms*, Chichester, U.K., Wiley.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002), A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, pp. 182–197.
- Deb, K., Tiwari, S. (2008), Omni-optimizer: A generic evolutionary algorithm for single and multiobjective optimization, *European Journal of Operational Research*, Vol. 185, pp. 1062–1087.
- Deb, K. (2000), An Efficient Constraint Handling Method for Genetic Algorithms, *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, pp. 311–338.
- Deep, K., Chauhan, P., Bansal, J.C. (2009), Solving Nonconvex Trim Loss Problem Using an Efficient Hybrid Particle Swarm Optimization, In *Proc. of Nature and Biologically Inspired Computing 2009 (NaBIC 2009)*, pp. 1608 – 1611.
- Deep, K., Das, K.N. (2008), quadratic approximation based hybrid genetic algorithm for function optimization, *applied mathematics and computation*, Vol. 203, pp. 86–98.
- Dong, H., He, J., Huang, H., Hou, W. (2007), Evolutionary programming using a mixed mutation strategy, *Information Science*, Vol. 177, pp. 312–327.
- Engelbrecht, A.P. (2005), *Fundamentals of Computational Swarm Intelligence*, England: John Wiley & Sons Ltd.
- Epitropakis, M.G., Plagianakos, V.P., Vrahatis, M.N. (2009), Evolutionary Adaption of the Differential Evolution Control Parameters, In *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 1359–1366.
- Fan, H-Y., Lampinen, J. (2003), A Trigonometric Mutation Operation to Differential Evolution, *Journal of Global Optimization*, Vol. 27, pp. 105–129.
- Fan, S-K.S., Lin, Y. (2007), A Multi-Level Thresholding Approach Using a Hybrid Optimal Estimation Algorithms, *Pattern Recognition Letters*, Vol. 28, pp. 662– 669.
- Feoktistov, V. (2006), *Differential Evolution: In Search of Solutions*, Springer, USA.

- Fogel, L.J., Owens, A.J., Walsh, M.J. (1965), Artificial intelligence through a simulation of evolution, In M. Maxfield, A. Callahan and L. J. Fogel, editors, *Biophysics and Cybernetic systems*, In Proc. of the 2nd Cybernetic Sciences Symposium, pp. 131 – 155.
- Fonseca, C.M., Fleming, P.J. (1993), Genetic algorithms for multiobjective optimization: formulation, discussion, and generalization, *Genetic Algorithms: Proc. of the Fifth International Conference*, pp. 416–423.
- Fonseca, C.M., Fleming, P.J. (1995), An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*, Vol. 3, pp. 1–16.
- Gamperle, R., Muller, S.D., Koumoutsakos, P. (2002), A Parameter study for differential evolution, In WSEAS NNA-FSFS-EC 2002, Interlaken, Switzerland, pp. 293–298.
- García, S., Molina, D., Lozano, M., Herrera, F. (2009), A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization, *Journal of Heuristics*, Vol. 15, pp. 617–644.
- Ghosh, S., Das, S., Kundu, D., Suresh, K., Panigrahi, B.K., Zhihua Cui, Z. (2010), An Inertia-Adaptive Particle Swarm System With Particle Mobility Factor for Improved Global Optimization, *Neural Computing and Applications*, DOI: 10.1007/s00521-010-0356-x.
- Gilmore, P.C., Gomory, R.E. (1961), A Linear Programming Approach to the Cutting-Stock Problem, *Operations Research*, Vol. 9, pp. 849–859.
- Goldberg, D.E. (1989), *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley.
- Gonzalez, R.C., Woods, R.E. (2002), *Digital Image Processing*, Prentice Hall, Upper Saddle River, NJ.
- Grosan, C., Abraham, A. (2009), A Novel Global Optimization Technique for High Dimensional Functions, *International Journal of Intelligent Systems*, John Wiley and Sons, USA, Vol. 24, pp. 421 – 440.
- Grosan, C., Abraham, A. (2008), Evolutionary Multiobjective Optimization for Large Scale Systems of Equations, *IEEE Transaction on Systems, Man and Cybernetics Part A*, IEEE Press, USA, Vol. 38, pp. 698–714.
- Haessler, R.W. (1971), A Heuristic Programming Solution to a Non-Linear Cutting Stock Problem, *Management Science*, Vol. 17, pp. 793–802.
- Hammouche, K., Diaf, M., Siarry, P. (2010), A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem, *Engineering Applications of Artificial Intelligence*, Vol. 23, pp. 676–688.
- Harjunkski, I., Westerlund, T., Porn, R. (1999), Numerical and environmental considerations on a complex industrial mixed integer non-linear programming (MINLP) problem, *Computers and Chemical Engineering*, Vol. 23, pp. 1545–1561.
- Harjunkski, I., Westerlund, T., Porn, R., Skrifvars, H. (1998), Different transformations for solving non-convex trim-loss problems by MINLP, *European Journal of Operational Research*, Vol. 105, pp. 594–603.
- Harjunkski, I., Westerlund, T., Isaksson, J., Skrifvars, H. (1996), Different formulations for solving trim loss problems in a paper-converting mill with ILP, *Computers and Chemical Engineering*, Vol. 20, pp. 121–126.

- Hendtlass, T. (2001), A Combined Swarm Differential Evolution Algorithm for Optimization Problems, In Proc. of 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Lecture Notes in Computer Science, Springer Verlag, Vol. 2070, pp. 11 – 18.
- Hinxman, A.I. (1980), The Trim-Loss and Assortment Problems: A Survey, European Journal of Operational Research, Vol. 5, pp. 8–18.
- Horn, J. (1997), Multi criteria decision making, In T Back, D. B. Fogel and Z. Michalewicz (Edt). Handbook of Evolutionary Computation, 97/1, F1.9, IOP Publishing Ltd. and Oxford University Press.
- Hornig, M-H. (2010), A Multilevel image thresholding using the honey bee mating optimization, Applied Mathematics and Computation, Vol. 215, pp. 3302–3310.
- Huang, V.L., Qin, A.K., Suganthan, P.N. (2006), Self-adaptive Differential Evolution Algorithm for constrained Real-Parameter Optimization, In Proc. of IEEE Congress on Evolutionary Computation 2006 (CEC 2006), pp. 223 – 230.
- Huang, V.L., Qin, A.K., Suganthan, P.N., Tasgetiren, M.F. (2007), Multi-objective optimization based on self adaptive differential evolution algorithm, In Proc. of the Congress on Evolutionary Computation 2007 (CEC 2007), pp. 3601–2608.
- Huang, V.L., Zhao, S.Z., Mallipeddi, R., Suganthan, P.N. (2009), Multi-objective optimization using self adaptive differential evolution algorithm, In Proc. of the Congress on Evolutionary Computation 2009 (CEC 2009), pp. 190–194.
- Jangam, S.R., Chakraborti, N. (2007), A Novel Method for Alignment of Two Nucleic Acid Sequences Using Ant Colony Optimization and Genetic Algorithms, Applied Soft Computing, Vol. 7, pp. 1121–1130.
- Johnston, R., Sadinlija, E. (2004), A new model for complete solutions to one-dimensional cutting stock problems, European Journal of Operational Research, Vol. 153, pp. 176–183.
- Johnston, R.E. (1979). Extensions to Haessler's Heuristics for the Trim Problem, Centre Technique du Papier CR No. 1366, Grenoble, France.
- Joshi, R., Sanderson, A.C. (1999), Minimal representation multi-sensor fusion using differential evolution, IEEE Transaction on Systems, Man and Cybernetics, Part A, Vol. 29, pp. 63–76.
- Kaelo, P., Ali, M.M. (2006), A numerical study of some modified differential evolution algorithms, European Journal Of Operational Research, Vol. 169, pp. 1176–1184.
- Kannan, S., Slochanal, S.M.R., Subbaraj, P., Padhy, N.P. (2004), Applications of Particle Swarm Optimization Techniques and its Variants to Generation Expansion Planning, Electric Power Systems Research, Vol. 70, pp. 203 – 210.
- Kapur, J.N., Sahoo, P.K., Wong, A.K.C. (1985), A new method for gray-level picture thresholding using the entropy of the histogram, Computer Vision Graphics Image Processing, Vol. 29, pp. 273–285.
- Kennedy, J., Eberhart, R.C. (1995), Particle Swarm Optimization, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, pp. 1942–1948.

- Knowles, J.D., Corne, D.W. (1999), The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation, In Proc. of the Congress on Evolutionary Computation 1999 (CEC 1999), pp. 98–105.
- Koziel, S., Michalewicz, Z. (1999), Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, *Evolutionary Computation*, Vol. 7, pp. 19–44.
- Lai, K.K., Chan, J.W.M. (1997), Developing a simulated annealing algorithm for the cutting stock problem, *Computers and Industrial Engineering*, Vol. 32, pp. 115–127.
- Lai, J.C.Y., Leung, F.H.F., Ling, S.H. (2009), A new Differential Evolution Algorithm with Wavelet Theory Based Mutation Operation, In Proc. of the IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 1116–1122.
- Lampinen, J. (1999), A bibliography of differential evolution algorithm. Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing, Tech. Report, Online available at: <http://www.lut.fi/jlampine/debiblio.htm>.
- Lampinen, J. (2001a), Multi-Constrained Optimization by the Differential Evolution, In Proc. of the IASTED International Conference Artificial Intelligence Applications (AIA 2001), pp. 177–184.
- Lampinen, J. (2001b), Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution, In Proc. of the MENDEL 2001, 7th International Conference on Soft Computing, pp. 50–57.
- Lampinen, J. (2002), A Constraint Handling Approach for the Differential Evolution Algorithm, In Proc. of the Congress on Evolutionary Computation 2002 (CEC 2002), pp. 1468–1473.
- Lampinen, J., Zelinka, I. (2000), On stagnation of the Differential Evolution Algorithm, In: Pavel Ošmera, (Ed.), In Proc. of MENDEL 2000, 6th International Mendel Conference on Soft Computing, pp. 76 – 83.
- Lampinen, J., Zelinka, I. (1999a), Mechanical Engineering Design Optimization by Differential Evolution, In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pp. 127–146. Mc Graw-Hill, UK.
- Lampinen, J., Zelinka, I. (1999b), Mixed Integer-Discrete-Continuous Optimization by Differential Evolution, Part 1: the optimization method. In Pavel Osmera, editor, Proc. of MENDEL'99, 5th International Mendel Conference on Soft Computing, pp. 71– 76. Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, June 1999. ISBN 80-214-1131-7.
- Lampinen, J., Zelinka, I. (1999c), Mixed Integer-Discrete-Continuous Optimization by Differential Evolution, Part 2: a practical example. In Pavel Osmera, editor, Proc. of MENDEL'99, 5th International Mendel Conference on Soft Computing, pp. 77–81. Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, June 1999. ISBN 80-214-1131-7.
- Lampinen, J., Zelinka, I. (1999d), Mixed Variable Non-Linear Optimization by Differential Evolution. In Proc. of Nostradamus'99, 2nd International Prediction Conference, pp. 45–55. Zlin, Czech Republic. Technical University of Brno, Faculty of Technology Zlin, Department of Automatic Control, ISBN 80-214-1424- 3.

- Mininno, E., Neri, F. (2010), A Memetic Differential Evolution Approach in Noisy Optimization, *Memetic Computing Journal*, Springer, Vol. 2, pp. 111–135.
- Mohan, C., Shanker, K. (1994), A Controlled Random Search Technique For Global Optimization using Quadratic Approximation, *Asia-Pacific Journal of Operational Research*, Vol. 11, pp. 93–101.
- Mondal, D.N., Sarangi, K., Pettersson, F., Sen, P.K., Saxén, H., Chakraborti, N. (2011), Cu-Zn Separation by Supported Liquid Membrane Analyzed through Multi-objective Genetic Algorithms, *Hydrometallurgy*, Vol. 107, pp.112–123.
- Montgomery, J. (2009), Differential Evolution: Difference Vectors and Movement in Solution Space, In Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 2833–2840.
- Nakib, A., Oulhadj, H., Siarry, P. (2007), Image histogram thresholding beased on multiobjective optimization, *Signal Processing*, Vol. 87, pp. 2516– 2534.
- Nakib, A., Oulhadj, H., Siarry, P. (2008), Non supervised image segmentation beased on multiobjective optimization, *Pattern Recognition Letters*, Vol. 29, pp. 161– 172.
- Nakib, A., Oulhadj, H., Siarry, P. (2010), Image thresholding based on Pareto multiobjective optimization, *Engineering Applications of Artificial Intelligence*, Vol. 23, pp. 313–320.
- Neri, F., Tirronen, V. (2010), Recent Advances in Differential Evolution: A Review and Experimental Analysis, *Artificial Intelligent Review*, Springer, Vol. 33, pp. 61–106.
- Neumaier, A. (2006), Global Optimization and Constraint Satisfaction, In I. Bomze, I. Emiris, Arnold Neumaier, and L. Wolsey, editors, Proc. of GICOLAG workshop (of the research project Global Optimization, Integrating Convexity, Optimization, Logic Programming and Computational Algebraic Geometry), December 2006. Online available at <http://www.mat.univie.ac.at/~neum/glopt.html>
- Noktehdan, A., Karimi, B., Kashan, A.H. (2010), A Differential Evolution Algorithm for the Manufacturing Cell Formation Problem Using Group Based Operators, *Expert System with Application*, Vol. 37, pp. 4822–4829.
- Noman, N., Iba, H. (2005), Enhancing differential evolution performance with local search for high dimensional function optimization, in Proc. of the 2005 Conference on Genetic and Evolutionary Computation, pp. 967–974.
- Noman, N., Iba, H. (2008), Accelerating Differential Evolution Using an Adaptive Local Search, *IEEE Transactions on Evolutionary Computation*, Vol. 12, pp. 107 – 125.
- Omran, M.G.H, Engelbrecht, A.P., Salman, A. (2005), Differential evolution methods for unsupervised image classification, In Proc. of IEEE Congress on Evolutionary Computation 2005 (CEC 2005), Piscataway, NJ: IEEE Press, pp. 966–973.
- Omran, M.G.H., Engelbrecht, A.P., Salman, A. (2009), Bare bones differential evolution, *European Journal of Operational Research*, Vol. 196, pp. 128–139.
- Omran, M.G.H., Engelbrecht, A.P. (2009), Free Search Differential Evolution, in Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 110–117.

- Lan, K-T., Lan, C-H. (2008), Notes on the distinction of Gaussian and Cauchy mutations, In Proc. Eighth international conference on intelligent systems design and applications, Vol. 1, pp. 272–277.
- Li, G.H., Lee, C.K. (1993), Minimum cross entropy thresholding, *Pattern Recognition*, Vol. 26, pp. 617–625.
- Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, N., Coello, C.A.C., Deb, K. (2006), Problem Definitions and Evaluation Criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization, Technical Report Report #2006005, Nanyang Technological University, Singapore, Online available at: <http://www.ntu.edu.sg/home/EPNSugan>.
- Liao, P.S., Chen, T.S., Chung, P.C.(2001), A fast algorithm for multilevel thresholding, *Journal of Information Science and Engineering*, Vol. 17, pp. 713–727.
- Lin, Y-C., Hwang, K-S., Wang, F-S. (2002), Hybrid Differential Evolution with Multiplier updating method for Nonlinear Constrained Optimization, In Proc. of the Congress on Evolutionary Computation 2002 (CEC 2002), pp. 872–877.
- Liu, H., Abraham, A. (2007), An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems, *Journal of Universal Computer Science*, Vol. 13, pp. 1032–1054.
- Liu, H., Abraham, A., Hassanien, A.E. (2010), Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, *Future Generation Computer Systems*, Vol. 26, pp. 1336–1343.
- Liu, H., Abraham, A., Zhang, W. (2007), A Fuzzy Adaptive Turbulent Particle Swarm Optimization, *International Journal of Innovative Computing and Applications*, Vol. 1, pp. 39–47.
- Liu, H., Abraham, A., Li, Y., Yang, X. (2006), Role of Chaos in Swarm Intelligence: A Preliminary Analysis, WSC10: 10th Online World Conference on Soft Computing in Industrial Applications, *Advances in Soft Computing: Recent Trends*, A. Tiwari et al. (Eds.), ISBN 3-540-29123-7, Springer Verlag, Germany.
- Liu, J., Lampinen, J. (2005), A fuzzy adaptive differential evolution algorithm, *Soft computing*, Vol. 9, pp. 448–462.
- Madavan, N.K. (2002), Multiobjective optimization using a Pareto differential evolution approach, In Proc. of the Congress on Evolutionary Computation 2002 (CEC 2002), pp. 1145–1150.
- Madhubanti, M., Amitava, A. (2008), A hybrid cooperative-comprehensive learning based algorithm for image segmentation using multilevel thresholding, *Expert Systems with Application*, Vol. 34, pp. 1341–1350.
- Menchaca-Mendez, A., Coello Coello, C.A. (2009), A new proposal to hybridize the Nelder Mead method to a Differential Evolution Algorithm for Constrained Optimization, In Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 2598–2605.
- Mezura-Montes, E., Coello Coello, C.A. (2004), An Improved Diversity Mechanism for Solving Constrained Optimization Problems Using a Multimembered Evolution Strategy, In Kalyanmoy Deb et. al., editor, In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2004), pp. 700–712, Heidelberg, Germany, June 2004. Seattle, WA.
- Mezura-Montes, E, Reyes, J.V., Coello, C.A.C. (2006), Modified Differential Evolution for Constrained Optimization, In Proc. of IEEE Congress on Evolutionary Computation 2006 (CEC 2006), pp. 25 – 32.

- Onwubolu, G.C., Babu, B.V. (2004), *New optimization techniques in engineering*, Springer-Verlag, Heidelberg, Germany.
- Ostermark, R. (1999), Solving a nonlinear non-convex trim loss problem with a genetic hybrid algorithm, *Computers and Operations research*, Vol. 26, pp. 623–635.
- Otsu, N. (1979), A Threshold Selection Method for Gray-Level Histogram, *IEEE Transactions on System, Man and Cybernetics*, Vol. 9, pp. 62–66.
- Panigrahi, B.K., Pandi, V.R. (2010), A Comparative Study of Evolutionary Computing Methods for Parameter Estimation of Power Quality Signals, *International Journal of Applied Evolutionary Computation*, Vol. 1, pp. 28 – 59.
- Pant, M., Ali, M., Singh, V.P. (2009a), Parent centric differential evolution algorithm for global optimization, *Opsearch*, Vol. 46, pp. 153–168.
- Pant, M., Thangaraj, R., Abraham, A., Grosan, C. (2009b), Differential Evolution with Laplace Mutation Operator, in *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 2841–2849.
- Pant, M., Ali, M., Abraham, A. (2009c), Mixed Strategy Embedded Differential Evolution, in *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 1240–1246.
- Pauling, L. (1960), *The Nature of the Chemical Bond*, Cornell Univ. Press, Ithaca, New York, ISBN: 0-8014-0333-2. Online available at: <http://osulibrary.oregonstate.edu/specialcollections/coll/pauling/>
- Peng, F., Tang, K., Chen, G., Yao, X. (2009), Multistart JADE with Knowledge Transfer for Numerical Optimization, in *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 1889–2895
- Qian, W., Li, A. (2008), Adaptive differential evolution algorithm for multiobjective optimization problems, *Applied Mathematics and Computation*, Vol. 201, 431–440.
- Qin, K., Huang, V.L., Suganthan, P.N. (2009), Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computations*, Vol. 13, pp. 398–417.
- Rahnamayan, S., Wang, G.G. (2008), Solving large scale optimization problems by opposition based differential evolution (ODE), *WSEAS Transaction on Computers*, Vol. 7, pp. 1792–1804.
- Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A. (2008), Opposition-Based Differential Evolution, *IEEE Transactions on Evolutionary Computation*, Vol. 12, pp. 64 – 79.
- Riehme, J., Scheithauer, G., Terno, J. (1996), The solution of two-stage guillotine cutting stock problems having extremely varying order demands, *European Journal of Operational Research*, Vol. 91, pp. 543–552.
- Robic, T., Filipic, B. (2005), DEMO: Differential Evolution for Multiobjective Optimization, In *Proc. of the 3rd International Conference on Evolutionary Multi Criterion Optimization (EMO 2005)*, LNCS 3410, pp. 520–533.
- Rochenberg, I. (1973), *Evolution Strategy: Optimization of Technical Systems by means of biological evolution*, Fromman-Holzboog, Stuttgart, Germany.
- Rodríguez, M.A., Vecchiotti, A. (2008), Enterprise optimization for solving an assignment and trim-loss non-convex problem, *Computers and Chemical Engineering*, Vol. 32, pp. 2812–2822.

- Rogalsky, T., Derksen, R.W., Kocabiyik, S. (1999), Differential evolution in aerodynamic optimization, in Proc. 46th Annual Conference, Canadian Aeronautics Space Institute, pp. 29–36.
- Ronkkonen, J., Kukkonen, S., Price, K.V. (2005), Real parameter optimization with differential evolution, in Proc. of IEEE Congress on Evolutionary Computation 2005 (CEC-2005), pp. 506 – 513.
- Rudolph G (1997), Local convergence rates of simple evolutionary algorithms with Cauchy mutations, IEEE Transactions on Evolutionary Computation, Vol. 1, pp. 249–258.
- Sabat, S.L., Ali, L., Udgata, S.K. (2011), Targeted Learning Particle Swarm Optimizer for Global Optimization, Applied Soft Computing, Vol. 11, pp. 574– 584.
- Sabat, S., Udgata, S.K., Abraham, A. (2010), Artificial Bee Colony Algorithm for Small Signal Model Parameter Extraction of MESFET, Engineering Applications of Artificial Intelligence, Elsevier Science, Netherlands, Vol. 23, pp. 689–694.
- Sahoo, P.K., Soltani, S., Wong, A.K.C., Chen, Y.C. (1988), A survey of thresholding techniques, Computer Vision Graphics and Image Processing, Vol. 41, pp. 233–236.
- Sarimveis, H., Nikolakopoulos, A. (2005), A Line Up Evolutionary Algorithm for Solving Nonlinear Constrained Optimization Problems, Computers and Operations Research, Vol. 32, pp. 1499–1514.
- Sathya, P.D., Kayalvizhi, R. (2010), Development of a New Optimal Multilevel Thresholding Using Improved Particle Swarm Optimization Algorithm for Image Segmentation, International Journal of Electronics Engineering, Vol. 2, pp. 63–67.
- Schaffer, J.D. (1984), Multiple objective optimization with vector evaluated genetic algorithms, Ph.D. thesis, Vanderbilt University, Nashville, TN.
- Schaffer, J.D. (1985), Multiple objective optimization with vector evaluated genetic algorithm, In Proc. of the First International Conference on Genetic Algorithms, pp. 93–100.
- Sezgin, M., Sankur, B. (2004), Survey over image thresholding techniques and quantitative performance evaluation, Journal of Electronic Imaging, Vol. 13, pp. 146–165.
- Shaik, M.A., Gudi, R.D. (2005), A Nonlinear Transformation based Hybrid Evolutionary Method for MINLP Solution, Chemical Engineering Research and Design, Vol. 83, pp. 1218–1236.
- Spencer, H. (1864), The Principles of Biology, Vol. 1. London & Edinburgh: Williams and Norgate, First (Ed.) Online available at: <http://www.archive.org/details/ThePrinciplesOfBiology>.
- Sreelaja, N.K., Vijayalakshmi, G.A. (2010), Ant Colony Optimization Based Approach for Efficient Packet Filtering In Firewall, Applied Soft Computing, Vol. 10, pp. 1222–1236.
- Srinivas, N., Deb, K. (1994), Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation, Vol. 2, pp. 221–248.
- Srinivas, M., Rangaiah, G.P. (2007), Differential Evolution with Tabu List for Solving Nonlinear and Mixed-Integer Nonlinear Programming Problems, Ind. Eng. Chem. Res. Vol. 46, pp. 7126–7135.
- Stacey, A., Jancic, M., Grundy, I. (2003), Particle swarm optimization with mutation, In: Proc. of IEEE congress on evolutionary computation 2003 (CEC 2003), pp. 1425–1430.

- Storn, R., Price, K. (1995), Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, Berkeley, CA.
- Storn, R. (1999), System Design by Constraint Adaptation and Differential Evolution, *IEEE Transactions on Evolutionary Computation*, Vol. 3, pp. 22–34.
- Storn, R., Price, K. (1997), Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, Vol. 11, pp. 341–359.
- Subudhi, B., Jena, D. (2011), A Differential Evolution Based Neural Network Approach to Nonlinear System Identification, *Applied Soft Computing*, Vol. 11, pp. 861–871.
- Suganya, N.C., Vijayalakshmi, G.A. (2010), Pareto-Archived Evolutionary Wavelet Network for Financial Constrained Portfolio Optimization, *Intelligent Systems in Accounting, Finance and Management*, Vol. 17, pp. 59–90.
- Synder, W., Bilbro, G., Logenthiran, A., Rajala, S. (1990), Optimal thresholding-a new approach, *Pattern Recognition Letters*, Vol. 11, pp. 803–810.
- Takahama, T., Sakai, S. (2006), Constrained Optimization by the ϵ -Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites, In *Proc. of IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*, pp. 1 – 8.
- Talbi, H., Batouche, M. (2004), Hybrid particle swarm with differential evolution for multimodal image registration, In *Proc. of the IEEE international conference on industrial technology*, pp. 1567–1572.
- Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M., Yang, Z. (2007), Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, Online available at: <http://nical.ustc.edu.cn/cec08ss.php>.
- Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N., Liang, Y.C. (2009), A Differential Evolution Algorithm with Variable Parameter Search for Real Parameter Continuous Function Optimization, in *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 1247–1254.
- Tasgetiren, F.M., Suganthan, P.N. (2006), A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem, In *Proc. of IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*, pp. 33 – 40.
- Teng, N.S., Teo, J., Hijazi, M.H.A. (2009), Self-adaptive population sizing for a tune-free differential evolution, *Soft Computing*, Vol. 13, pp. 709–724.
- Teo, J. (2006), Exploring Dynamic Self-adaptive Populations in Differential Evolution, *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Vol. 10, pp. 673 – 686.
- Thompsen, R. (2004), Multimodal optimization using crowding-based differential evolution, In *Proc. of IEEE Congress on Evolutionary Computation 2004 (CEC 2004)*, pp. 1382–1389.
- Ting, C.K., Huang, C-H. (2009), Varying numbers of Difference Vectors in Differential Evolution, in *Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009)*, Norway, pp. 1351–1358.

- Tizhoosh, H.R. (2005), Opposition-Based Learning: a New Scheme for Machine Intelligence, in Proc. International Conference on Computational Intelligence, Modeling Control and Automation (CIMCA 2005), pp. 695–701.
- Trkman, P., Gradisar, M. (2007), One-Dimensional Cutting Stock Optimization in Consecutive Time Periods, European Journal of Operational Research, Vol. 179, pp. 291–301.
- Veldhuizen, D.A., Van, Lamont, G.B. (2000), On Measuring Multiobjective Evolutionary Algorithm Performance, In Proc. of the Congress on Evolutionary Computation 2000 (CEC 2000), pp. 204–211.
- Wagner, B.J. (1999), A genetic Algorithm Solution for One-Dimensional Bundled Stock Cutting, European Journal of Operational Research, Vol. 117, pp. 368–381.
- Wang, H., Liu, Y., Li, C., Zeng, S. (2007), A Hybrid Particle Swarm Algorithm With Cauchy Mutation, In IEEE swarm intelligence symposium 2007 (SIS 2007), Honolulu, Hawaii, USA.
- Wang, F.S., Jang, H.J. (2000), Parameter Estimation of a Bio-Reaction Model by Hybrid Differential Evolution, In Proc. of the Congress on Evolutionary Computation 2000 (CEC 2000), 1. Piscataway, NJ: IEEE Press, pp. 410–417.
- Wang, Y., Li, B., Lai, X. (2009), Variance Priority based Cooperative Co-evolution Differential Evolution for large scale Global Optimization, In Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 1232–1239.
- Weibull, J.W. (1995), Evolutionary Game Theory, MIT press, Cambridge, MA.
- Westerlund, T., Isaksson, J. (1998), Some efficient Formulation for the Simultaneous Solution of Trim-Loss and Scheduling Problems in the Paper-Converting Industry, Chemical Engineering Research and Design, Vol. 76, pp. 677–684.
- Weszka, J.S., Azriel, R. (1979), Histogram Modifications for Threshold Selection, IEEE Transactions on System, Man and Cybernetics, Vol. 9, pp. 38–52.
- Xianjun, S., Li, Y., Zheng, B., Dai, Z. (2007), General Particle Swarm Optimization based on simulated Annealing for Multi-Specification One-dimensional cutting stock problem, CIS 2006, LNAI4456, pp. 67–76.
- Xu, W., Gu, X. (2009), A Hybrid Particle Swarm Optimization Approach With Prior Crossover Differential Evolution, In Proc. of GEC09, pp. 671 – 677.
- Xue, F., Sanderson, A.C., Graves, R.J. (2003), Pareto-Based Multi-Objective Differential Evolution, In Proc. of the IEEE Congress on Evolutionary Computation 2003 (CEC 2003), pp. 862–869.
- Yang, Z., Tang, K., Yao, X. (2008a), Self-adaptive Differential Evolution with Neighborhood Search, In Proc. of the IEEE Congress on Evolutionary Computation 2008 (CEC 2008), Hong Kong, pp. 1110–1116.
- Yang, Z., Tang, K., Yao, X. (2008b), Large Scale Evolutionary Optimization Using Cooperative Coevolution, Information Sciences, Vol. 178, pp. 2985–2999.
- Yang, Z., Zhang, J., Tang, K., Yao, X., Sanderson, A.C. (2009), An adaptive Co-evolutionary Differential Evolution Algorithm for large Scale optimization, in Proc. of IEEE Congress on Evolutionary Computation 2009 (CEC 2009), Norway, pp. 102 – 109.

- Yao, X., Liu, Y., Lin, G. (1999), Evolutionary Programming Made Faster, *IEEE Transactions on Evolutionary Computation*, Vol. 3, pp. 82–102.
- Ye, Z.W., Chen, H.W., Li, W., Zhang, J.P. (2008), Automatic Threshold Selection Based on Particle Swarm Optimization Algorithm, in *Proc. of International Conference on Intelligent Computation Technology and Automation*, pp. 36–39.
- Yen, C.H., Wong, D.S.H., Jang, S.S. (2004), Solution of Trim Loss Problem by an Integrated Simulated Annealing and Ordinal Optimization Approach, *Journal of intelligent manufacturing*, Vol. 15, pp. 701–709.
- Yin, P.Y. (2007), Multilevel Minimum Cross Entropy Threshold Selection Based on Particle Swarm Optimization, *Applied Mathematics and Computation*, Vol. 184, pp. 503–513.
- Zahara, E., Fan, S-K. S., Tsai, D-M. (2005), Optimal Multi-Thresholding Using a Hybrid Optimization Approach, *Pattern Recognition Letters*, Vol. 26, pp. 1082–1095.
- Zaharie, D. (2003), Control Of Population Diversity and Adaptation in Differential Evolution Algorithms, In D. Matousek, P. Osmera (eds.), *Proc. of MENDEL 2003, 9th International Conference on Soft Computing*, Brno, Czech Republic, pp. 41–46.
- Zaharie, D., Petcu, D. (2004), Adaptive Pareto Differential Evolution and its Parallelization, In *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, 3019, pp. 261 – 268.
- Zhang, C., Ning, J., Lu, S., Ouyang, D., Ding, T. (2009), A Novel Hybrid Differential Evolution and Particle Swarm Optimization Algorithm for Unconstrained Optimization, *Operations Research Letters*, Vol. 37, pp. 117 – 122.
- Zhang, J., Sanderson, A.C. (2009), JADE: Adaptive Differential Evolution With Optional External Archive, *IEEE Transactions on Evolutionary Computation*, Vol. 13, pp. 945–958.
- Zhang, R. and Liu, J. (2006), Underwater Image Segmentation with Maximum Entropy Based on Particle Swarm Optimization (PSO), In *Proc. of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 06)*, pp. 360–363.
- Zhang, W-J., Xie, X-F. (2003), DEPSO: Hybrid Particle Swarm With Differential Evolution Operator, In *Proc. of IEEE International Conference on Systems Man and Cybernetics*, Vol. 4, pp. 3816 – 3821.
- Zielinski, K., Laur, R. (2006), Constrained Single - Objective Optimization Using Differential Evolution, In *Proc. of the IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*, pp. 223 – 230.
- Zitzler, E., Thiele, L. (1999), Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation*, Vol. 3, pp. 257–271.
- Zitzler, E., Thiele, L., Bader, J. (2010), On Set-Based Multi-objective Optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 14, pp. 58–79.

List of Unconstrained Test Problems

All these test problems are taken from (Rahnamayan et al., 2008; Zhang and Sanderson, 2009).

1. Dejong's function (f_1)

$$\text{Minimize } f(x) = \sum_{i=1}^D x_i^2$$

Properties:

- $-100 \leq x_i \leq 100$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- This function is the dream of every optimization algorithm. It is also called the sphere model. It is smooth and symmetric. Also this function is continuous, convex and unimodal.

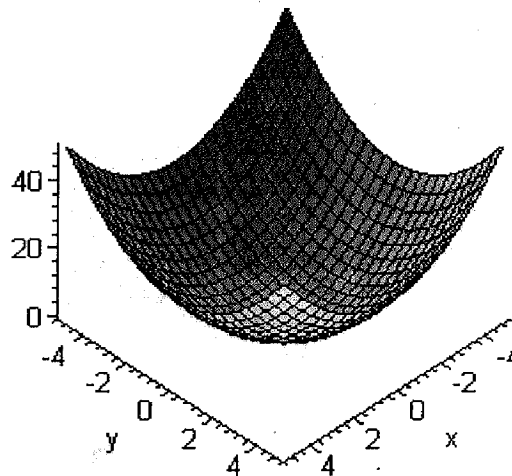


Figure I.1: 3-D map for 2-D function f_1 .

2. Schwefel's function 2.22 (f_2)

$$\text{Minimize } f(x) = \sum_{i=1}^D |x_i| + \prod_{i=1}^D |x_i|$$

Properties:

- $-10 \leq x_i \leq 10$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- This function is a unimodal function.

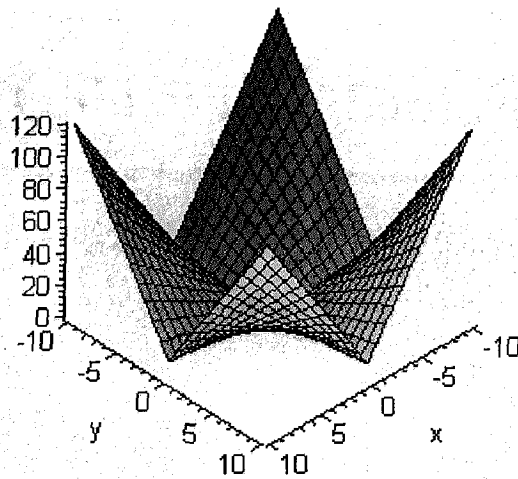


Figure I.2: 3-D map for 2-D function f_2 .

3. Schwefel's function 1.2 (f_3)

Minimize $f(x) = \sum_{i=1}^D \left(\sum_{j=0}^i x_j \right)^2$

Properties:

- $-100 \leq x_i \leq 100$, $x^* = (0, 0, \dots, 0)$, $f(x^*) = 0$
- This function is a unimodal function.

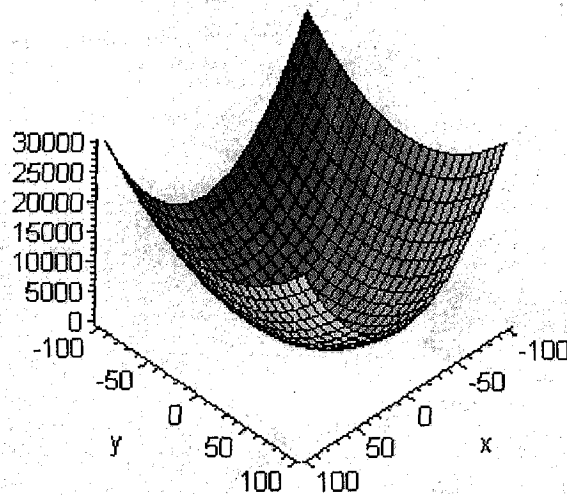


Figure I.3: 3-D map for 2-D function f_3 .

4. Schwefel's function 2.21 (f_4)

Minimize $f(x) = \max |x_i|$

Properties:

- $1 \leq i \leq D$, $-100 \leq x_i \leq 100$, $x^* = (0,0,0,\dots,0)$, $f(x^*) = 0$
- This function is a unimodal function.

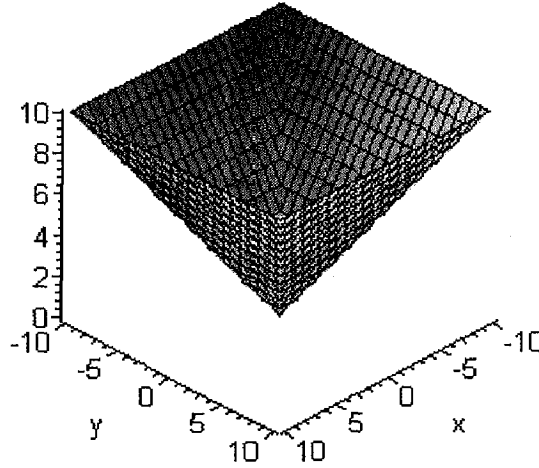


Figure I.4: 3-D map for 2-D function f_4

5. Rosenbrock function (f_5)

Minimize $f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$

Properties:

- $-30 \leq x_i \leq 30$, $x^* = (1,1,\dots,1)$, $f(x^*) = 0$
- It is a classic optimization problem with a narrow global optimum hidden inside a long, narrow and curved flat valley. It is unimodal, yet due to a saddle point it is very difficult to locate the minimum. This function is also known as banana valley function.

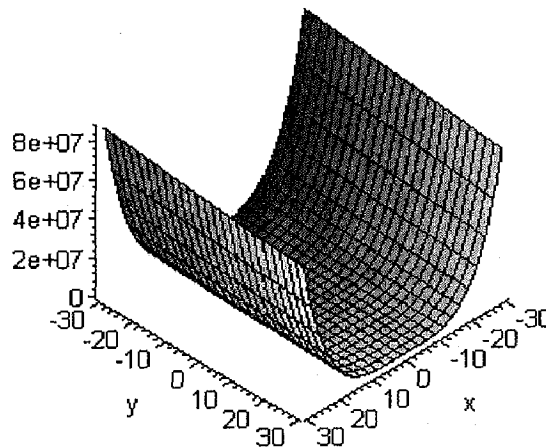


Figure I.5: 3-D map for 2-D function f_5

6. Step function (f_6)

$$\text{Minimize } f(x) = \sum_{i=1}^D [x_i + 1/2]^2$$

Properties:

- $-100 \leq x_i \leq 100$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- It is the representative of the problem of flat surfaces. Flat surfaces are obstacles for optimization algorithms, because they do not give any information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateaus. It has one global minimum and is discontinuous.

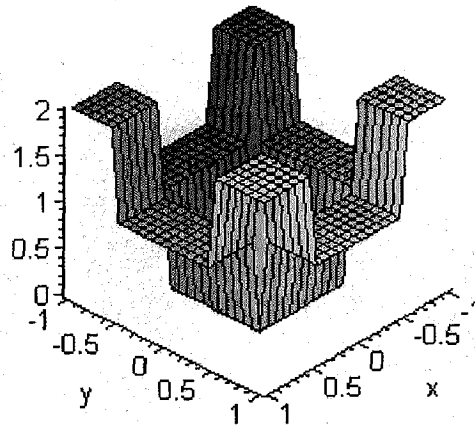


Figure I.6: 3-D map for 2-D function f_6 .

7. Dejong's function with noise (f_7)

$$\text{Minimize } f(x) = \left(\sum_{i=1}^D ix_i^4 \right) + \text{rand}[0,1]$$

Properties:

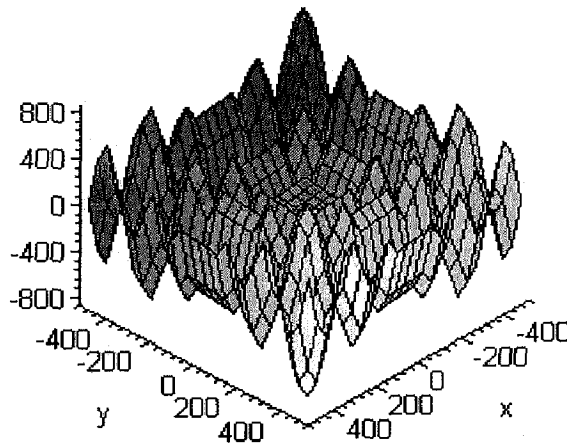
- $-1.28 \leq x_i \leq 1.28$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- This function is a simple unimodal function padded with noise. Algorithms that do not do well on this test function will do poorly on noisy data.

8. Schwefel function (f_8)

$$\text{Minimize } f(x) = -\sum_{i=1}^D x_i \sin(\sqrt{|x_i|})$$

Properties:

- $-500 \leq x_i \leq 500$, $x^* = (420.97, 420.947, \dots, 420.947)$, $f(x^*) = -418.9829 * D$
- This function is deceptive in nature. Here the global minimum is geometrically distant, over the parameter space, from the next best global minima. Therefore the search algorithms are prone to converge in wrong direction.

Figure 1.7: 3-D map for 2-D function f_8 .

9. Rastrigin function (f_9)

$$\text{Minimize } f(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

Properties:

- $-5.12 \leq x_i \leq 5.12$, $x^* = (0, 0, \dots, 0)$, $f(x^*) = 0$
- This function is highly multimodal with regularly distributed many local minima. The total number of minima for this function is not exactly known but the global minimum is located at the origin. For 2 dimension, it has about 50 local minimas arranged in a lattice like configuration.

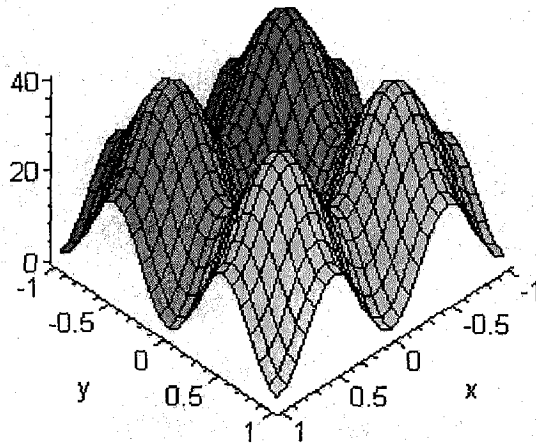


Figure I.8: 3-D map for 2-D function f_9 .

10. Ackley's path function (f_{10})

$$\text{Minimize } f(x) = 20 + e - 20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right)$$

Properties:

- $-32 \leq x_i \leq 32$, $x^* = (0, 0, \dots, 0)$, $f(x^*) = 0$.
- This function is widely used multimodal function. The number of local minima is not known, but the global minimum is located at the origin.

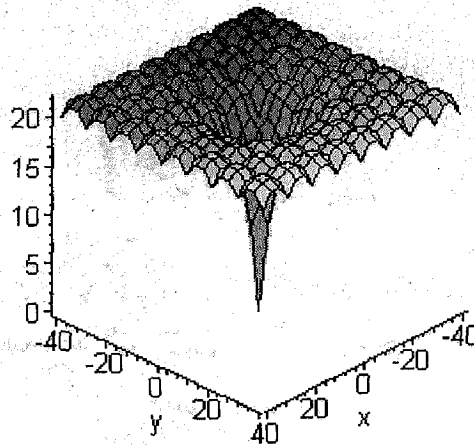


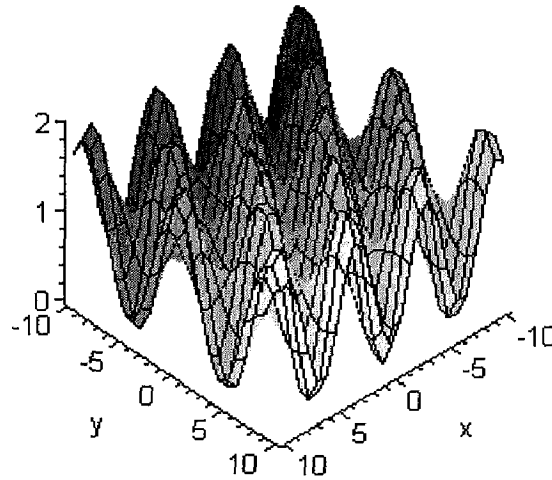
Figure I.9: 3-D map for 2-D function f_{10} .

11. Griewank function (f_{11})

$$\text{Minimize } f(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Properties:

- $-600 \leq x_i \leq 600$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- This test problem is similar to Rastrigin function. It has thousands of local minima. However the locations of minima are regularly distributed.

Figure I.10: 3-D map for 2-D function f_{11} .

12. Generalized penalized function 1 (f_{12})

$$\text{Minimize } f(x) = \frac{\pi}{D} \{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(y_{i+1}\pi)] \\ + (y_D - 1)^2 \} + \sum_{i=1}^D u(x_i, 10, 100, 4), \text{ Where } y_i = 1 + \frac{1}{4}(x_i + 1)$$

Properties:

- $-50 \leq x_i \leq 50$, $x^* = (0,0,\dots,0)$, $f(x^*) = 0$
- This function is a multimodal function where the number of local minima increases exponentially with the problem dimension. It appears to be the most difficult class of problems for many optimization algorithms.

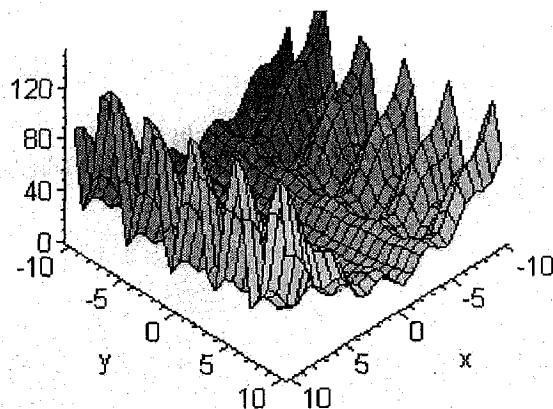


Figure I.11: 3-D map for 2-D function f_{12} .

13. Generalized penalized function 2 (f_{13})

$$\text{Minimize } f(x) = (0.1)\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} ((x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})))\} \\ + (x_D - 1)(1 + \sin^2(2\pi x_D))\} + \sum_{i=1}^{D-1} u(x_i, 5, 100, 4)$$

Properties:

- $-50 \leq x_i \leq 50$, $x^* = (0, 0, \dots, 0)$, $f(x^*) = 0$
- This function is similar to Generalized penalized function 1. It is a multimodal function where the number of local minima increases exponentially with the problem dimension. It appears to be the most difficult class of problems for many optimization algorithms.

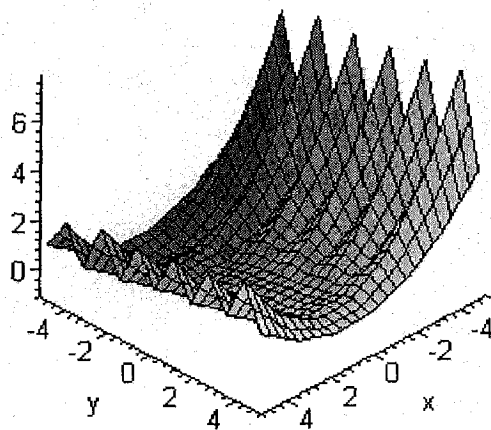


Figure I.12: 3-D map for 2-D function f_{13} .

In problems 12 and 13,

$$u(x, a, b, c) = b(x-a)^c \quad \text{if } x > a,$$

$$u(x, a, b, c) = b(-x-a)^c \quad \text{if } x < -a,$$

$$u(x, a, b, c) = 0 \quad \text{if } -a \leq x \leq a.$$

14. Zhakarov function (f_{14})

$$\text{Minimize } f(x) = \sum_{i=1}^D x_i^2 + \left(\sum_{i=1}^D 0.5ix_i\right)^2 + \left(\sum_{i=1}^D 0.5ix_i\right)^4$$

Properties:

➤ $-5 \leq x_i \leq 10$, $x^* = (0,0,0,\dots,0)$, $f(x^*) = 0$

➤ This function has no local minima, it has one global minima at the origin.

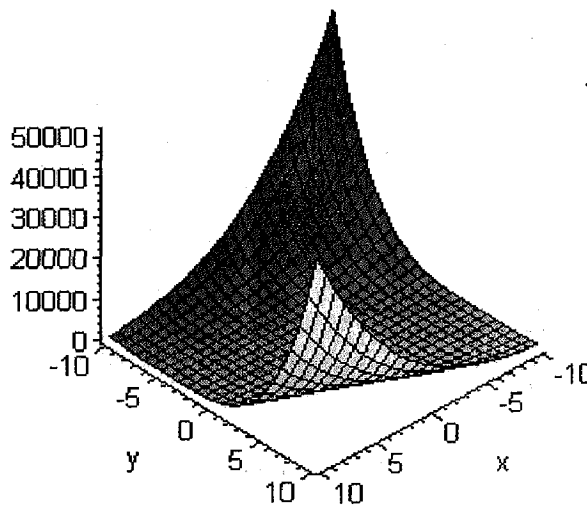


Figure I.13: 3-D map for 2-D function f_{14} .

15. Shekel's Foxholes function (f_{15})

$$\text{Minimize } f(x) = \left(\frac{1}{500} + \sum_{j=0}^{24} \left(j+1 + \frac{1}{\sum_{i=0}^6 (x_i - a_{ij})^6}\right)^{-1}\right)^{-1}$$

Properties:

➤ $-65.54 \leq x_i \leq 65.54$, $x^* = (-31.95, -31.95)$, $f(x^*) = 1$

where $a = \begin{pmatrix} -32, -16, 0, 16, 32, \dots, -32, -16, 0, 16, 32 \\ -32, \dots, -16, \dots, 0, \dots, 16, \dots, 32, \dots \end{pmatrix}$

➤ This function is a low dimensional function; it has only a few local minima.

16. Six hump camel back function (f_{16})

$$\text{Minimize } f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

Properties:

- $-5 \leq x_i \leq 5$, $x^* = (0.09, -0.71)$, $f(x^*) = -1.03163$
- This function has two global minima with the minimum of -1.03163.

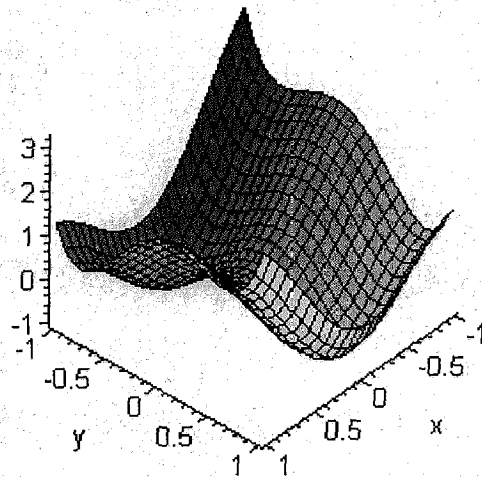


Figure I.14: 3-D map for 2-D function f_{16} .

17. Branin function (f_{17})

$$\text{Minimize } f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$$

Properties:

- $-10 \leq x_i \leq 10$, $x^* = (9.42, 2.47)$, $f(x^*) = 0.397886$
- This function has three global minima.

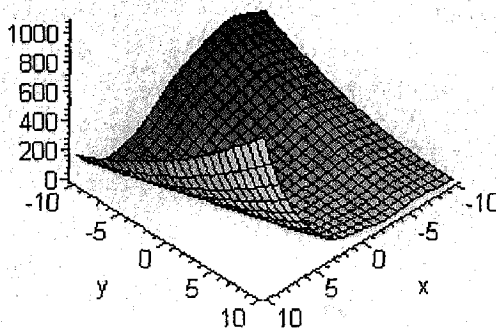


Figure I.15: 3-D map for 2-D function f_{17} .

18. Goldstein and price function (f_{18})

$$\text{Minimize } f(x) = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \\ \{30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_2^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$$

Properties:

- $-2 \leq x_i \leq 2$, $x^* = (0,1)$, $f(x^*) = 3$
- This problem has four local minima and one global minima.

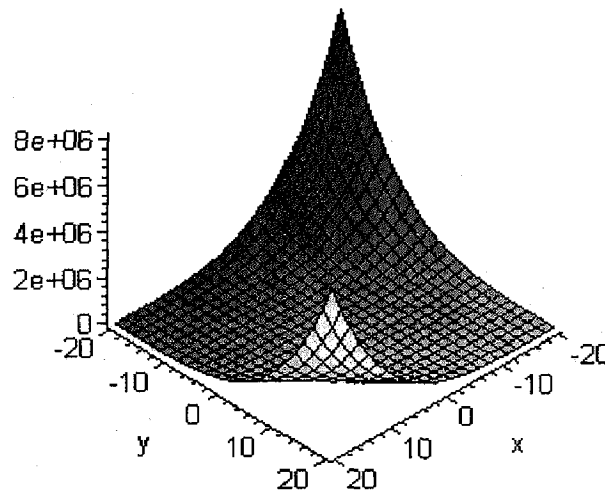


Figure I.16: 3-D map for 2-D function f_{18} .

19. Easom function (f_{19})

$$\text{Minimize } f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 - (x_2 - \pi)^2]$$

Properties:

- $-10 \leq x_i \leq 10$, $x^* = (\pi, \pi)$, $f(x^*) = -1$
- The function value rapidly approaches zero, when away from (π, π) .

20. Hartmann function 1 (f_{20})

$$\text{Minimize } f(x) = -\sum_{i=1}^4 \alpha_i \exp\left(-\sum_{j=1}^3 A_{ij}(x_j - P_{ij})^2\right),$$

Properties:

- $0 \leq x_i \leq 1$, $x^* = (0.114614, 0.555649, 0.852547)$, $f(x^*) = -3.86278$

$$\text{where, } \alpha = [1 \quad 1.2 \quad 3 \quad 3.2], \quad A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},$$

$$P = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$$

➤ This function has four local minima and one global minima.

21. Shekel 5 function (f_{21})

$$\text{Minimize } f(x) = -\sum_{i=1}^5 \left[\sum_{j=1}^4 (x_j - a_{ij})(x_j - a_{ij})^T + c_i \right]^{-1}$$

Properties:

$$\text{➤ } 0 \leq x_j \leq 10, \quad x^* = (4, 4, 4, 4) \quad f(x^*) = -10.1499$$

22. Shekel 7 function (f_{22})

$$\text{Minimize } f(x) = -\sum_{i=1}^7 \left[\sum_{j=1}^4 (x_j - a_{ij})(x_j - a_{ij})^T + c_i \right]^{-1}$$

Properties:

$$\text{➤ } 0 \leq x_j \leq 10, \quad x^* = (4, 4, 4, 4) \quad f(x^*) = -10.3999$$

23. Shekel 10 function (f_{23})

$$\text{Minimize } f(x) = -\sum_{i=1}^{10} \left[\sum_{j=1}^4 (x_j - a_{ij})(x_j - a_{ij})^T + c_i \right]^{-1}$$

Properties:

$$\text{➤ } 0 \leq x_j \leq 10, \quad x^* = (4, 4, 4, 4) \quad f(x^*) = -10.5319$$

➤ The values of constant a_{ij} and c_i for problem 21, 22 and 23 are given below.

$$a = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix} \quad c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$$

24. Kowalik function (f_{24})

Minimize
$$f(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$$

Properties:

- $-5 \leq x_i \leq 5$, $x^* = (0.192, 0.190, 0.123, 0.135)$, $f(x^*) = 0.0003075$
- The values of constant a_i and b_i are given as below:

$$a = \begin{bmatrix} 0.1957 \\ 0.1947 \\ 0.1735 \\ 0.1600 \\ 0.0844 \\ 0.0627 \\ 0.0456 \\ 0.0342 \\ 0.0323 \\ 0.0235 \\ 0.0246 \end{bmatrix} \quad b = \begin{bmatrix} 0.25 \\ 0.5 \\ 1 \\ 2 \\ 4 \\ 6 \\ 8 \\ 10 \\ 12 \\ 14 \\ 16 \end{bmatrix}$$

25. Hartmann function 2 (f_{25})

Minimize
$$f(x) = -\sum_{i=1}^4 \alpha_i \exp\left(-\sum_{j=1}^6 B_{ij}(x_j - Q_{ij})^2\right)$$

Properties:

➤ $0 \leq x_i \leq 1$, $x^* = (0.20169, 0.50011, 0.476874, 0.275332, 0.311652, 0.6573)$,

$f(x^*) = -3.32237$ Where $\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]$,

$$B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$$

$$Q = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

➤ This function has four local minima and one global minima.

List of Shifted Test Problems

All these problems are taken from (Tang et al., 2007). For all the problems:

- D: Dimension
- $x = [x_1, x_2, \dots, x_D]$
- $o = [o_1, o_2, \dots, o_D]$: Shifted global optimum
- $z = x - o$

1. F_1 : Shifted Sphere Function

$$F_1(x) = \sum_{i=1}^D z_i^2 + f_{\text{bias}_1}$$

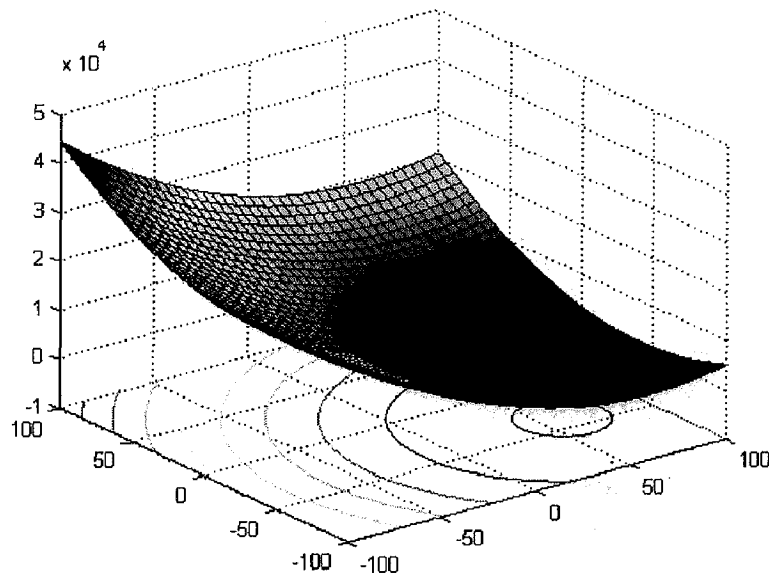


Figure II.1: 3-D map for 2-D function F_1 .

Properties:

- Unimodal
- Shifted
- Separable
- Scalable

Appendix II

➤ $-100 \leq x_i \leq 100$, $x^* = 0$, $F_1(x^*) = f_bias_1 = -450$

2. F_2 : Schwefel's Function 2.21

$$F_2(x) = \max\{|z_i|, 1 \leq i \leq D\} + f_bias_2$$

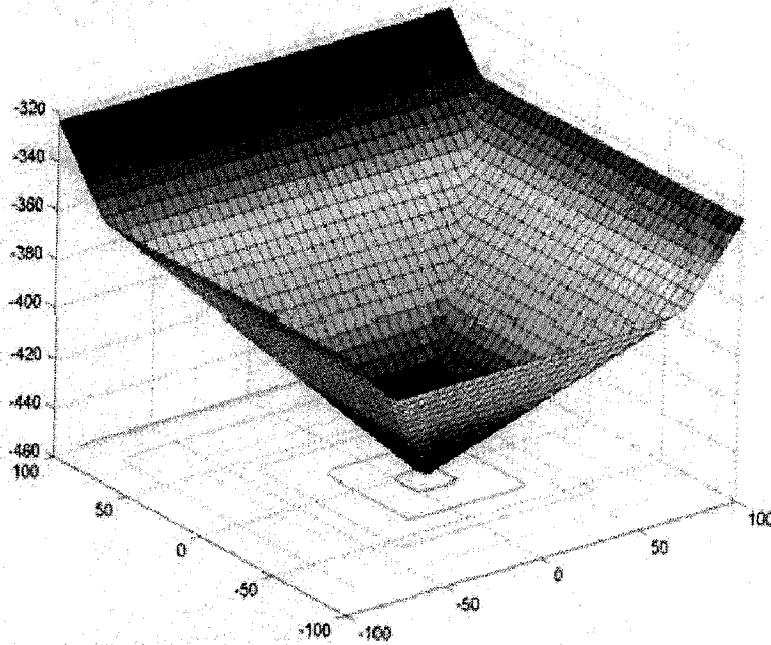


Figure II.2: 3-D map for 2-D function F_2

Properties:

- Unimodal
- Shifted
- Non-Separable
- Scalable
- $-100 \leq x_i \leq 100$, $x^* = 0$, $F_1(x^*) = f_bias_1 = -450$

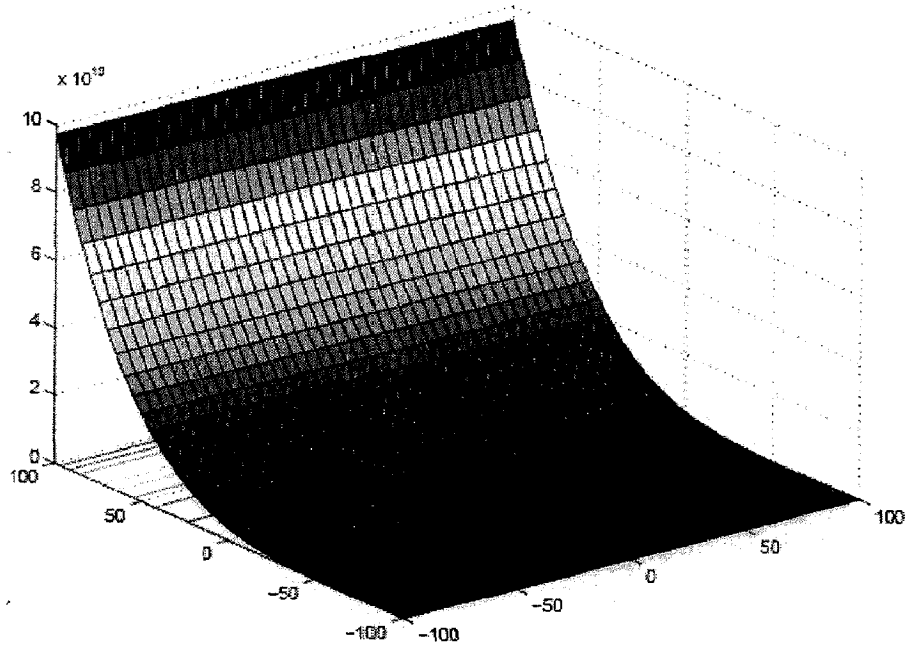
3. F_3 : Shifted Rosenbrock's Function

$$F_3(x) = \sum_{i=1}^D (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_bias_3$$

Properties:

- Multi-modal

- Shifted
- Non-Separable
- Scalable
- Having a very narrow valley from local optimum to global optimum
- $-100 \leq x_i \leq 100$, $x^* = 0$, $F_3(x^*) = f_bias_3 = 390$

Figure II.3: 3-D map for 2-D function F_3 .

4. F_4 : Shifted Rastrigin's Function

$$F_4(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_bias_4$$

Properties:

- Multi-modal
- Shifted
- Separable
- Scalable
- Local optima's number is huge
- $-5 \leq x_i \leq 5$, $x^* = 0$, $F_4(x^*) = f_bias_4 = -330$

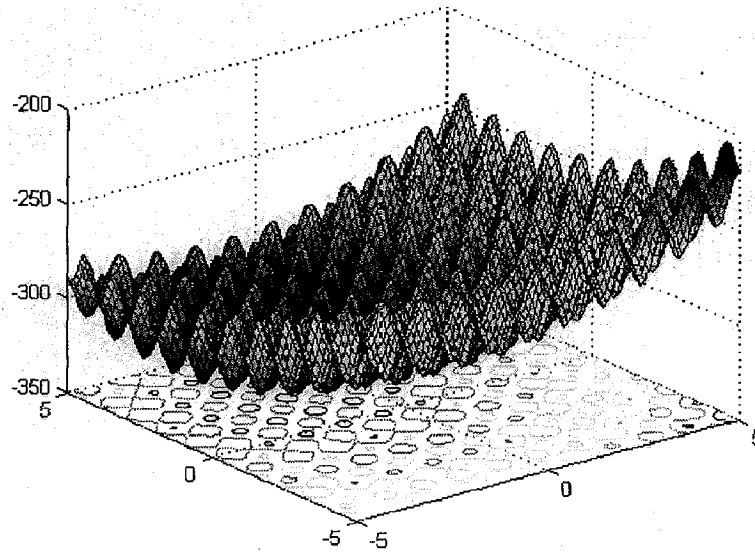


Figure II.4: 3-D map for 2-D function F_4 .

5. F_5 : Shifted Griewank's Function

$$F_5(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_bias_5$$

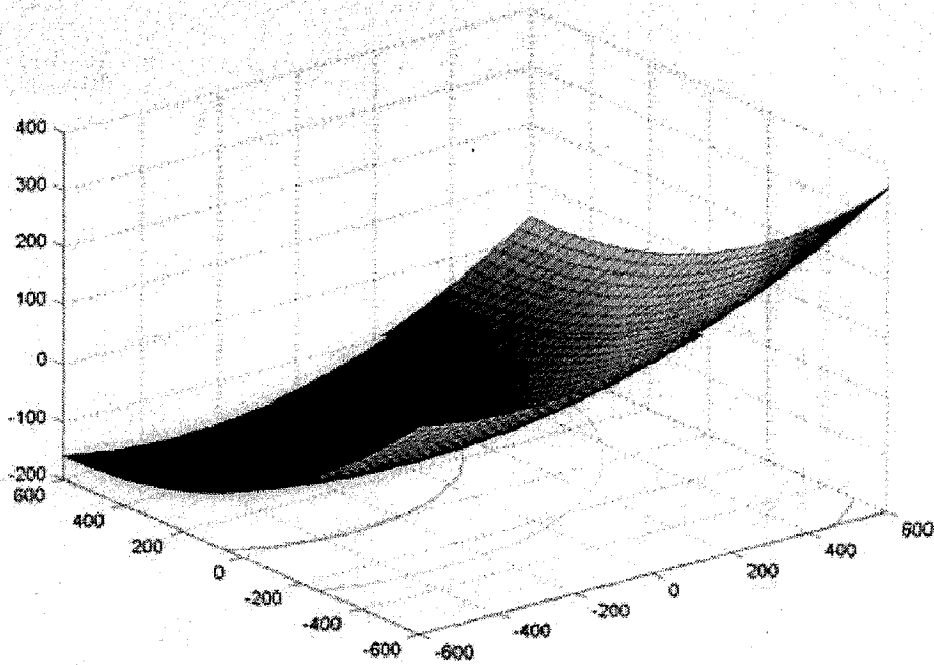


Figure II.5: 3-D map for 2-D function F_5 .

Properties:

- Multi-modal
- Shifted
- Non-Separable
- Scalable

➤ $-600 \leq x, \leq 600, x^* = 0, F_5(x^*) = f_bias_5 = -180$

6. F_6 : Shifted Ackley's Function

$$F_6(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)\right) + 20 + e + f_bias_6$$

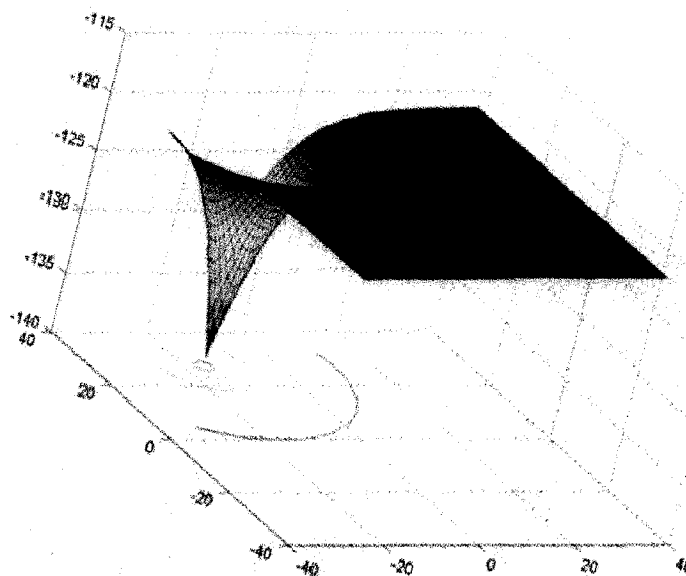


Figure II.6: 3-D map for 2-D function F_6

Properties:

- Multi-modal
- Shifted
- Separable
- Scalable

➤ $-32 \leq x_i \leq 32, x^* = 0, F_6(x^*) = f_bias_6 = -140$

7. F_7 : FastFractal "DoubleDip" Function

$$F_7(x) = \sum_{i=1}^D \text{fractal1D}(x_i + \text{twist}(x_{(i \bmod D)+1}))$$

$$\text{twist}(y) = 4(y^4 - 2y^3 + y^2)$$

$$\text{fractal1D}(x) \approx \sum_{k=1}^3 \sum_{l=1}^{2^{k-1}} \sum_{m=1}^{\text{ran2}(o)} \text{doubledip}(x, \text{ran1}(o), \frac{1}{2^{k-1}(2 - \text{ran1}(o))})$$

$$\text{doubledip}(x, c, s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s, & -0.5 \leq x \leq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

ran1(o): double, pseudorandomly chosen, with seed o , with equal probability from the interval $[0,1]$

ran2(o): integer, pseudorandomly chosen, with seed o , with equal probability from the set $\{0,1,2\}$

fractal1D(x) is an approximation to a recursive algorithm, it does not take account of wrapping at the boundaries, or local re-seeding of the random generators - please use the executable provided

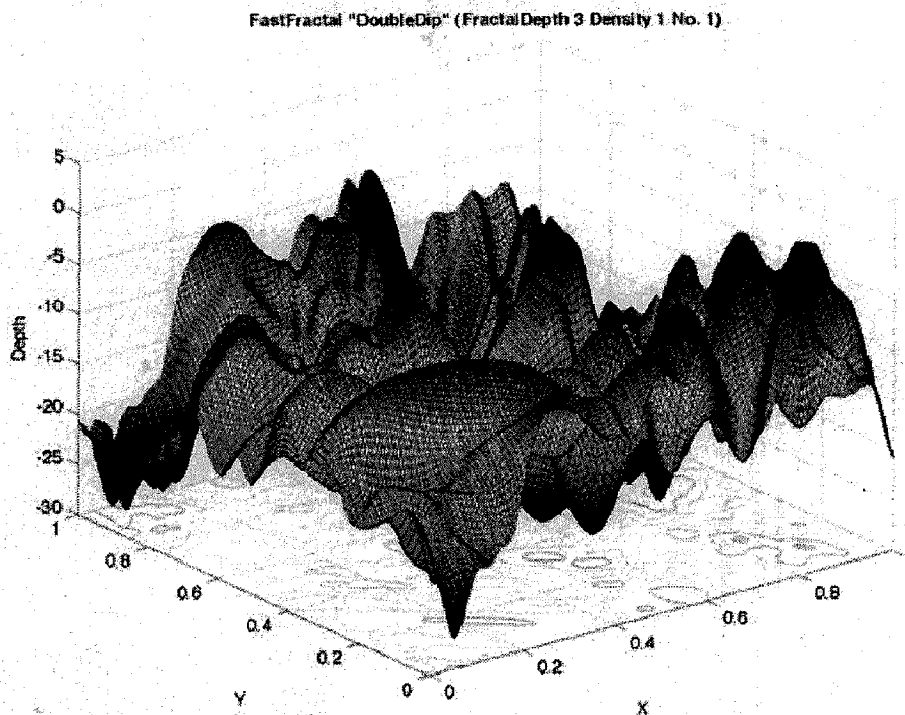


Figure II.7: 3-D map for 2-D function F_7 .

Properties:

- Multi-modal
- Non-Separable
- Scalable
- $-1 \leq x_i \leq 1$, Global optimum unknown, $F_7(x^*)$ unknown

APPENDIX III

List of Constrained Test Problems

These problems are taken from (Liang et al., 2006).

1. Problem 1 (g01)

$$\text{Minimize } f(x) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i,$$

Subject to:

$$g_1(x) = (2x_1 + 2x_2 + x_{10} + x_{11}) - 10 \leq 0,$$

$$g_2(x) = (2x_1 + 2x_3 + x_{10} + x_{12}) - 10 \leq 0,$$

$$g_3(x) = (2x_2 + 2x_3 + x_{11} + x_{12}) - 10 \leq 0,$$

$$g_4(x) = -8x_1 + x_{10} \leq 0,$$

$$g_5(x) = -8x_2 + x_{11} \leq 0,$$

$$g_6(x) = -8x_3 + x_{12} \leq 0,$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0,$$

$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0,$$

$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0,$$

Properties:

$$\triangleright 0 \leq x_i \leq 1, \quad i = 1, \dots, 9, \quad 0 \leq x_i \leq 100, \quad i = 10, 11, 12, \quad 0 \leq x_{13} \leq 1.$$

$$\triangleright x^* = (1, 1, \dots, 1, 3, 3, 3, 1)$$

$$\triangleright f(x^*) = -15.$$

2. Problem 2 (g02)

$$\text{Minimize } f(x) = \frac{\left| \sum_{i=1}^D \cos^4(x_i) - 2 \prod_{i=1}^D \cos^2(x_i) \right|}{\sqrt{\sum_{i=1}^D ix_i^2}},$$

Subject to:

$$g_1(x) = 0.75 - \prod_{i=1}^D x_i \leq 0,$$

$$g_2(x) = \sum_{i=1}^D x_i - 7.5n \leq 0,$$

Properties:

- $0 \leq x_i \leq 10$ where $(i = 1, \dots, D)$ and $D = 20$
- $x^* = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865, 0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317)$
- $f(x^*) = -0.80361910412559$ (the best found till date).

3. Problem 3 (g03)

Minimize $f = -(\sqrt{n})^D \prod_{i=1}^D x_i$

Subject to:

$h_1(x) = \sum_{i=1}^n x_i^2 - 1 = 0$

Properties:

- $0 \leq x_i \leq 1$ where $(i = 1, 2, \dots, D)$ and $D = 10$
- $x^* = (0.31624357647283069, 0.316243577414338339, 0.316243578012345927, 0.316243575664017895, 0.316243578205526066, 0.31624357738855069, 0.316243575472949512, 0.316243577164883938, 0.316243578155920302, 0.316243576147374916)$
- $f(x^*) = -1.00050010001$.

4. Problem 4 (g04)

Minimize $f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$

Subject to:

$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$

$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$

$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$

$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$

$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$

$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$

Properties:

- $78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ ($i = 3, 4, 5$)

$$\triangleright x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$$

$$\triangleright f(x^*) = -30665.539$$

5. Problem 5 (g05)

$$\text{Minimize } f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

Subject to:

$$g_1(x) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(x) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(x) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(x) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(x) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

Properties:

$$\triangleright 0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200, -0.55 \leq x_3 \leq 0.55 \text{ and } -0.55 \leq x_4 \leq 0.55.$$

$$\triangleright x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$$

$$\triangleright f(x^*) = 5126.49671.$$

6. Problem 6 (g06)

$$\text{Minimize } f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Subject to:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

Properties:

$$\triangleright 13 \leq x_1 \leq 100 \text{ and } 0 \leq x_2 \leq 100$$

$$\triangleright x^* = (14.095, 0.84296)$$

$$\triangleright f(x^*) = -6961.81388.$$

7. Problem 7 (g07)

$$\begin{aligned} \text{Minimize } f(x) = & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 \\ & + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \end{aligned}$$

Subject to:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

Appendix III

$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

Properties:

$$\triangleright -10 \leq x_i \leq 10 \quad (i = 1, \dots, 10)$$

$$\triangleright x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, \\ 1.321644, 9.828726, 8.280092, 8.375927)$$

$$\triangleright f(x^*) = 24.3062091.$$

8. Problem 8 (g08)

$$\text{Minimize } f(x) = -\frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to:

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

Properties:

$$\triangleright 0 \leq x_1 \leq 10 \text{ and } 0 \leq x_2 \leq 10$$

$$\triangleright x^* = (1.22797135260752599, 4.24537336612274885)$$

$$\triangleright f(x^*) = -0.0958250414180359.$$

9. Problem 9 (g09)

$$\text{Minimize } f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Subject to:

$$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

Properties:

- $-10 \leq x_i \leq 10$ for $(i = 1, \dots, 7)$
- $x^* = (2.33049935147405174, 1.95137236847114592, -.477541399510615805,$
 $4.36572624923625874, -0.624486959100388983, 1.03813099410962173,$
 $1.5942266780671519)$
- $f(x^*) = 680.630057374402$

10. Problem 10 (g10)

Minimize $f(x) = x_1 + x_2 + x_3$

Subject to:

- $g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$
- $g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$
- $g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$
- $g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$
- $g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$
- $g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$

Properties:

- $100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000$ ($i = 2, 3$) and $10 \leq x_i \leq 1000$ ($i = 4, \dots, 8$)
- $x^* = (579.306685017979589, 1359.97067807935605, 5109.97065743133317,$
 $182.01769963061534, 295.601173702746792, 217.982300369384632,$
 $286.41652592786852, 395.601173702746735)$
- $f(x^*) = 7049.24802052867$.

11. Problem 11 (g11)

Minimize $f(x) = x_1^2 + (x_2 - 1)^2$

Subject to:

$$h_1(x) = x_2 - x_1^2 = 0.$$

Properties:

- $-1 \leq x_i \leq 1, i = 1, 2.$
- $x^* = (-0.707036070037170616, 0.500000004333606807)$
- $f(x^*) = 0.7499$

12. Problem 12 (g12)

Minimize $f(x) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

Subject to:

$g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

Properties:

- $0 \leq x_i \leq 10 (i = 1, 2, 3)$ and $p, q, r = 1, 2, \dots, 9$.
- The feasible region of the search space consists of 9^3 disjoint spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds.
- $x^* = (5, 5, 5)$
- $f(x^*) = -1$

13. Problem 13 (g13)

Minimize $f(x) = e^{(x_1 x_2 x_3 x_4 x_5)}$

Subject to:

$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$

$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$

$h_3(x) = x_1^3 + x_2^3 + 1 = 0$

Properties:

- $-2.3 \leq x_i \leq 2.3 (i = 1, 2)$ and $-3.2 \leq x_i \leq 3.2 (i = 3, 4, 5)$
- $x^* = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498)$
- $f(x^*) = 0.053941514041898$

14. Problem 14 (g14)

Minimize $f(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$

Subject to:

$h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$

$$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

Properties:

$$\begin{aligned} &\triangleright 0 \leq x_i \leq 10 \quad (i=1, \dots, 10), \text{ and } c_1 = -6.089, \quad c_2 = -17.164, \quad c_3 = -34.054, \\ &\quad c_4 = -5.914, \quad c_5 = -24.721, \quad c_6 = -14.986, \quad c_7 = -24.1, \quad c_8 = -10.708, \\ &\quad c_9 = -26.662, \quad c_{10} = -22.179. \end{aligned}$$

$$\begin{aligned} &\triangleright x^* = (0.0406684113216282, \quad 0.147721240492452, \quad 0.783205732104114, \\ &\quad 0.00141433931889084, \quad 0.485293636780388, \quad 0.000693183051556082, \\ &\quad 0.0274052040687766, \quad 0.0179509660214818, \quad 0.0373268186859717, \\ &\quad 0.0968844604336845) \end{aligned}$$

$$\triangleright f(x^*) = -47.7648884594915$$

15. Problem 15 (g15)

$$\text{Minimize } f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

Subject to:

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

Properties:

$$\triangleright 0 \leq x_i \leq 10 \quad (i=1, 2, 3)$$

$$\triangleright x^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$$

$$\triangleright f(x^*) = 961.715022289961$$

16. Problem 16 (g16)

$$\text{Minimize } f(x) = -0.0000005843y_{17} + 0.000117y_{14} + 0.1365 + 0.00002358y_{13}$$

$$+ 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001 \frac{c_{15}}{c_{16}} + 37.48 \frac{y_2}{c_{12}}$$

Subject to:

$$g_1(x) = y_4 - \frac{0.28}{0.72} y_5 \geq 0$$

$$g_2(x) = 1.5x_2 - x_3 \geq 0$$

$$g_3(x) = 21 - 3496 \frac{y_2}{c_{12}} \geq 0$$

Appendix III

$$g_4(x) = \frac{62,212}{c_{17}} - 110.6 - y_1 \geq 0$$

$$g_5(x) = y_1 - 213.1 \geq 0$$

$$g_6(x) = 405.23 - y_1 \geq 0$$

$$g_7(x) = y_2 - 17.505 \geq 0$$

$$g_8(x) = 1053.6667 - y_2 \geq 0$$

$$g_9(x) = y_3 - 11.275 \geq 0$$

$$g_{10}(x) = 35.03 - y_3 \geq 0$$

$$g_{11}(x) = y_4 - 214.228 \geq 0$$

$$g_{12}(x) = 665.585 - y_4 \geq 0$$

$$g_{13}(x) = y_5 - 7.458 \geq 0$$

$$g_{14}(x) = 584.463 - y_5 \geq 0$$

$$g_{15}(x) = y_6 - 0.961 \geq 0$$

$$g_{16}(x) = 265.916 - y_6 \geq 0$$

$$g_{17}(x) = y_7 - 1.612 \geq 0$$

$$g_{18}(x) = 7.046 - y_7 \geq 0$$

$$g_{19}(x) = y_8 - 0.146 \geq 0$$

$$g_{20}(x) = 0.222 - y_8 \geq 0$$

$$g_{21}(x) = y_9 - 107.99 \geq 0$$

$$g_{22}(x) = 273.366 - y_9 \geq 0$$

$$g_{23}(x) = y_{10} - 922.693 \geq 0$$

$$g_{24}(x) = 1286.105 - y_{10} \geq 0$$

$$g_{25}(x) = y_{11} - 926.832 \geq 0$$

$$g_{26}(x) = 1444.046 - y_{11} \geq 0$$

$$g_{27}(x) = y_{12} - 18.766 \geq 0$$

$$g_{28}(x) = 537.141 - y_{12} \geq 0$$

$$g_{29}(x) = y_{13} - 1072.163 \geq 0$$

$$g_{30}(x) = 3247.039 - y_{13} \geq 0$$

$$g_{31}(x) = y_{14} - 8961.448 \geq 0$$

$$g_{32}(x) = 26844.086 - y_{14} \geq 0$$

$$g_{33}(x) = y_{15} - 0.063 \geq 0$$

$$g_{34}(x) = 0.386 - y_{15} \geq 0$$

$$g_{35}(x) = y_{16} - 71084.33 \geq 0$$

$$g_{36}(x) = 140000 - y_{16} \geq 0$$

$$g_{37}(x) = y_{17} - 2802713 \geq 0$$

$$g_{38}(x) = 12146108 - y_{17} \geq 0$$

where:

$$y_1 = x_2 + x_3 + 41.6$$

$$c_1 = 0.024x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12$$

$$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$$

$$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$$

$$y_3 = \frac{c_2}{c_3}$$

$$y_4 = 19y_3$$

$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$$

$$c_5 = 100x_2$$

$$c_6 = x_1 - y_3 - y_4$$

$$c_7 = 0.950 - \frac{c_4}{c_5}$$

$$y_5 = c_6c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3$$

$$c_8 = (y_5 + y_4)0.995$$

$$y_7 = \frac{c_8}{y_1}$$

$$y_8 = \frac{c_8}{3798}$$

$$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$$

$$y_9 = \frac{96.82}{c_9} + 0.321y_1$$

$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$$

$$y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3$$

$$c_{10} = \frac{12.3}{752.3}$$

$$c_{11} = (1.75y_2)(0.995x_1)$$

$$c_{12} = 0.995y_{10} + 1998$$

Appendix III

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2$$

$$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146.312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

Properties:

$$\triangleright 704.4148 \leq x_1 \leq 906.3855, 68.6 \leq x_2 \leq 288.88, 0 \leq x_3 \leq 134.75,$$

$$193 \leq x_4 \leq 287.0966 \text{ and } 25 \leq x_5 \leq 84.1988$$

$$\triangleright x^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$$

$$\triangleright f(x^*) = -1.90515525853479$$

17. Problem 17 (g17)

Minimize $f(x) = f(x_1) + f(x_2)$

where

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

Subject to:

$$h_1(x) = -x_1 + 300 - \frac{x_3 x_4}{131.078} \cos(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} \cos(1.47588)$$

$$h_2(x) = -x_2 - \frac{x_3 x_4}{131.078} \cos((1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} \cos(1.47588)$$

$$h_3(x) = -x_5 - \frac{x_3 x_4}{131.078} \sin((1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} \sin(1.47588)$$

$$h_4(x) = 200 - \frac{x_3 x_4}{131.078} \sin((1.48477 + x_6) + \frac{0.90798 x_3^2}{131.078} \sin(1.47588)$$

Properties:

$$\begin{aligned} &\triangleright 0 \leq x_1 \leq 400, \quad 0 \leq x_2 \leq 1000, \quad 340 \leq x_3 \leq 420, \quad 340 \leq x_4 \leq 420, \\ &\quad -1000 \leq x_5 \leq 1000 \text{ and } 0 \leq x_6 \leq 0.5236 \end{aligned}$$

$$\triangleright x^* = (201.784467214523659, 99.999999999999005, 383.071034852773266, 420, 10.9076584514292652, 0.0731482312084287128)$$

$$\triangleright f(x^*) = 8853.53967480648$$

18. Problem 18 (g18)

$$\text{Minimize } f(x) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$$

Subject to:

$$g_1(x) = x_3^2 + x_4^2 - 1 \leq 0$$

$$g_2(x) = x_9^2 - 1 \leq 0$$

$$g_3(x) = x_5^2 + x_6^2 - 1 \leq 0$$

$$g_4(x) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$$

$$g_5(x) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$

$$g_6(x) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$

$$g_7(x) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0$$

$$g_8(x) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0$$

$$g_9(x) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$

$$g_{10}(x) = x_2 x_3 - x_1 x_4 \leq 0$$

$$g_{11}(x) = -x_3 x_9 \leq 0$$

$$g_{12}(x) = x_5 x_9 \leq 0$$

$$g_{13}(x) = x_6 x_7 - x_5 x_8 \leq 0$$

Properties:

$$\triangleright -10 \leq x_i \leq 10 (i = 1, \dots, 8) \text{ and } 0 \leq x_9 \leq 20$$

- $x^* = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$
- $f(x^*) = -0.866025403784439$

19. Problem 19 (g19)

$$\text{Minimize } f(x) = \sum_{j=1}^5 \sum_{i=1}^5 c_{ij} x_{(10+i)} x_{(10+j)} + 2 \sum_{j=1}^5 d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i$$

Subject to:

$$g_j(x) = -2 \sum_{i=1}^5 c_{ij} x_{(10+i)} - 3d_j x_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij} x_i \leq 0 \quad j=1, \dots, 5$$

Properties:

- $b = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1], 0 \leq x_i \leq 10 (i = 1, \dots, 15)$ and the remaining data is in Table III.1
- $x^* = (1.66991341326291344e-17, 3.95378229282456509e-16, 3.94599045143233784, 1.06036597479721211e-16, 3.2831773458454161, 9.99999999999999822, 1.12829414671605333e-17, 1.2026194599794709e-17, 2.50706276000769697e-15, 2.24624122987970677e-15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579)$
- $f(x^*) = 32.6555929502463$

Table III.1: Data set for test problem 19.

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	-10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0.4	2
a_{3j}	-3.5	0	2	0	0

a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

20. Problem 20 (g20)

Minimize $f(x) = \sum_{i=1}^{24} a_i x_i$

Subject to:

$$g_i(x) = \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 1, 2, 3$$

$$g_i(x) = \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 4, 5, 6$$

$$h_i(x) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} \leq 0 - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1, 2, \dots, 12$$

$$h_{13}(x) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(x) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

Properties:

➤ $k = (0.7302)(530) \left(\frac{14.7}{40} \right)$, $0 \leq x_i \leq 10$ ($i = 1, \dots, 24$) and the data set is detailed

in Table III.2.

➤ $x^* = (1.28582343498528086e - 18, 4.83460302526130664e - 34, 0, 0, 6.30459929660781851e - 18, 7.57192526201145068e - 34, 5.03350698372840437e - 34, 9.28268079616618064e - 34, 0, 1.76723384525547359e - 17, 3.55686101822965701e - 34, 2.99413850083471346e - 34, 0.158143376337580827,$

2.29601774161699833e-19, 1.06106938611042947e-18,
 1.31968344319506391e-18, 0.530902525044209539, 0,
 2.89148310257773535e-18, 3.34892126180666159e-18, 0,
 0.310999974151577319, 5.41244666317833561e-05,
 4.84993165246959553e-16)

➤ This solution is a little infeasible and no feasible solution is found so far.

21. Problem 21 (g21)

Minimize $f(x) = x_1$.

Subject to:

$$g_1(x) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$

$$h_1(x) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(x) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(x) = -x_5 + \ln(-x_4 + 900) = 0$$

$$h_4(x) = -x_6 + \ln(x_4 + 300) = 0$$

$$h_5(x) = -x_7 + \ln(-2x_4 + 700) = 0$$

Properties:

- $0 \leq x_1 \leq 1000, 0 \leq x_2, x_3 \leq 40, 100 \leq x_4 \leq 300, 6.3 \leq x_5 \leq 6.7, 5.9 \leq x_6 \leq 6.4$
 and $4.5 \leq x_7 \leq 6.25$.
- $x^* = (193.724510070034967, 5.56944131553368433e-27,$
 $17.3191887294084914, 100.047897801386839, 6.68445185362377892,$
 $5.99168428444264833, 6.21451648886070451)$
- $f(x^*) = 193.724510070035$

Table III.2: Data set for test problem 20.

1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	

12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

22. Problem 22 (g22)

Minimize $f(x) = x_1$

Subject to:

$$g_1(x) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$$

$$h_1(x) = x_5 - 100000x_8 + 1 \times 10^7 = 0$$

$$h_2(x) = x_6 + 100000x_8 - 100000x_9 = 0$$

$$h_3(x) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$

$$h_4(x) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$$

$$h_5(x) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$$

$$h_6(x) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$$

$$h_7(x) = x_5 - 120x_2x_{13} = 0$$

$$h_8(x) = x_6 - 80x_3x_{14} = 0$$

$$h_9(x) = x_7 - 40x_4x_{15} = 0$$

$$h_{10}(x) = x_8 - x_{11} + x_{16} = 0$$

$$h_{11}(x) = x_9 - x_{12} + x_{17} = 0$$

$$h_{12}(x) = -x_{18} + \ln(x_{10} - 100) = 0$$

$$h_{13}(x) = -x_{19} + \ln(-x_8 + 300) = 0$$

$$h_{14}(x) = -x_{20} + \ln(x_{16}) = 0$$

$$h_{15}(x) = -x_{21} + \ln(-x_9 + 400) = 0$$

$$h_{16}(x) = -x_{22} + \ln(x_{17}) = 0$$

$$h_{17}(x) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$

$$h_{18}(x) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$

$$h_{19}(x) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

Properties:

➤ $0 \leq x_1 \leq 20000, 0 \leq x_2, x_3, x_4 \leq 1 \times 10^6, 0 \leq x_5, x_6, x_7 \leq 4 \times 10^7, 100 \leq x_8 \leq 299.99, 100 \leq x_9 \leq 399.99, 100.01 \leq x_{10} \leq 300, 100 \leq x_{11} \leq 400, 100 \leq x_{12} \leq 600, 0 \leq x_{13}, x_{14}, x_{15} \leq 500, 0.01 \leq x_{16} \leq 300, 0.01 \leq x_{17} \leq 400, -4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25.$

➤ $x^* = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395)$

➤ $f(x^*) = 236.430975504001$

23. Problem 23 (g23)

Minimize $f(x) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$

Subject to:

$$g_1(x) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(x) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(x) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(x) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(x) = x_3 + x_6 - x_5 = 0$$

$$h_4(x) = x_4 + x_7 - x_8 = 0$$

Properties:

➤ $0 \leq x_1, x_2, x_6 \leq 300, 0 \leq x_3, x_5, x_7 \leq 100, 0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$

➤ $x^* = (0.00510000000000259465, 99.9947000000000514, 9.01920162996045897e - 18, 99.9999000000000535, 0.000100000000027086086, 2.75700683389584542e - 14, 99.999999999999574, 2000.0100000100000100008)$

➤ $f(x^*) = -400.055099999999584$

24. Problem 24 (g24)

Minimize $f(x) = -x_1 - x_2$

Subject to:

$$x_2 \leq 2x_1^4 - 8x_1^3 + 8x_1^2 + 2$$

$$x_2 \leq 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 + 36$$

Properties:

➤ $0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4$

➤ $x^* = (2.32952019747762, 3.17849307411774)$

➤ $f(x^*) = -5.50801327159536$

List of Multi-Objective Test Problems

All these test problems are taken from (Deb, 2001).

1. *Schaffer's function (SCH)*

$$f_1(x) = x^2, \quad f_2(x) = (x - 2)^2$$

Properties:

- $-1000 \leq x \leq 1000$
- Optimal solutions at $x \in [0, 2]$
- Convex

2. *Fonseca and Fleming function (FON)*

$$f_1(x) = 1 - \exp\left(-\sum_{i=1}^3 \left(x_i - \frac{1}{\sqrt{3}}\right)^2\right), \quad f_2(x) = 1 - \exp\left(-\sum_{i=1}^3 \left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$$

Properties:

- $-4 \leq x_i \leq 4, i = 1, 2, 3$
- Optimal solutions at $(x_1 = x_2 = x_3) \in [-1/\sqrt{3}, 1/\sqrt{3}]$
- Non convex

3. *Poloni's function (POL)*

$$f_1(x) = [1 + (A_1 - B_1)^2 + (A_2 - B_2)^2], \quad f_2(x) = [(x_1 + 3)^2 + (x_2 + 1)^2]$$

$$A_1 = 0.5 \sin 1 - 2 \cos 1 + \sin 2 - 1.5 \cos 2, \quad A_2 = 1.5 \sin 1 - \cos 1 + 2 \sin 2 - 0.5 \cos 2,$$

$$B_1 = 0.5 \sin x_1 - 2 \cos x_1 + \sin x_2 - 1.5 \cos x_2, \quad B_2 = 1.5 \sin x_1 - \cos x_1 + 2 \sin x_2 - 0.5 \cos x_2,$$

Properties:

- $-\pi \leq x_i \leq \pi, i = 1, 2$
- Optimal solutions at (Deb, 2001)
- Non convex
- Disconnected

4. Kursawe's function (KUR)

$$f_1(x) = \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right),$$

$$f_2(x) = \sum_{i=1}^n \left(|x_i|^{0.8} + 5 \sin(x_i^3) \right)$$

Properties:

- $-5 \leq x_i \leq 5, i = 1, 2, 3$
- Optimal solutions at (Deb, 2001)
- Non convex

5. ZDT1 function (ZDT1)

$$f_1(x) = x_1, \quad f_2(x) = g(x) \left[1 - \sqrt{x_1 / g(x)} \right], \quad g(x) = 1 + 9 \left(\sum_{i=2}^n x_i \right) / (n - 1)$$

Properties:

- $0 \leq x_i \leq 1, i = 1, 2, \dots, 30$
- Optimal solutions at $x_1 \in [0, 1], x_i = 0, i = 2, 3, \dots, 30$
- Convex

6. ZDT2 function (ZDT2)

$$f_1(x) = x_1, \quad f_2(x) = g(x) \left[1 - (x_1 / g(x))^2 \right]$$

$$g(x) = 1 + 9 \left(\sum_{i=2}^n x_i \right) / (n - 1)$$

Properties:

- $0 \leq x_i \leq 1, i = 1, 2, \dots, 30$
- Optimal solutions at $x_1 \in [0, 1], x_i = 0, i = 2, 3, \dots, 30$
- Non convex

7. ZDT3 function (ZDT3)

$$f_1(x) = x_1, \quad f_2(x) = g(x) \left[1 - \sqrt{x_1 / g(x)} - \frac{x_1}{g(x)} \sin(10\pi x_1) \right]$$

$$g(x) = 1 + 9 \left(\sum_{i=2}^n x_i \right) / (n - 1)$$

Properties:

- $0 \leq x_i \leq 1, i = 1, 2, \dots, 30$
- Optimal solutions at $x_1 \in [0, 1], x_i = 0, i = 2, 3, \dots, 30$
- This function is convex and disconnected in nature.

8. ZDT4 function (ZDT4)

$$f_1(x) = x_1 \quad f_2(x) = g(x) \left[1 - \sqrt{x_1 / g(x)} \right]$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^n [x_i^2 - 10 \cos(4\pi x_i)]$$

Properties:

- $0 \leq x_i \leq 1, i = 1, 2, \dots, 10$
- Optimal solutions at $x_1 \in [0, 1], x_i = 0, i = 2, 3, \dots, 10$
- Non convex

9. ZDT6 function (ZDT6)

$$f_1(x) = 1 - \exp(-4x_1) \sin^6(6\pi x_1) \quad f_2(x) = g(x) \left[1 - (f_1(x) / g(x))^2 \right]$$

$$g(x) = 1 + 9 \left[\left(\sum_{i=2}^n x_i \right) / (n - 1) \right]^{0.25}$$

Properties:

- $0 \leq x_i \leq 1, i = 1, 2, \dots, 10$
- Optimal solutions at $x_1 \in [0, 1], x_i = 0, i = 2, 3, \dots, 10$
- Non convex
- Non uniformly spaced

Non Parametric Tests

Due to the lack of strong theoretical convergence proofs in the field of evolutionary algorithms, the researchers mainly rely on empirical analysis to compare two algorithms. Statistical analysis of algorithms can be done on the basis of (i) parametric tests or (ii) non parametric tests.

According to the literature, parametric tests have been used more frequently in comparison to non parametric tests. The most commonly used statistical criterion is the use of parametric *paired t-test*. However, the application of paired t-test requires the fulfilment of the following three conditions:

1. The data in the set should be independent
2. The sample should be normally distributed
3. Variance of the samples should be equal

If any of the above three conditions are not satisfied then the paired t-test may give a wrong conclusion.

Non parametric tests on the other hand are more suitable for comparing the algorithms because they do not require any specific condition for their application. A detailed study on the use of non parametric tests is given in Garcia et al. (2009).

Non parametric tests can be used for comparing algorithms whose results represent average values for each problem, in spite of the inexistence of relationships among them. In this thesis, results are analyzed statistically for *multiple-problem analysis* (a comparison of algorithms over more than one problem simultaneously) by using the following non parametric tests.

- Friedmann test
- Bonferrani Dunn test
- Wilcoxon test

All the tests used here find the associated p -value, which represents the dissimilarity of the sample of results. Hence, a low p -value points out a critical difference. The definitions relevant to the present study are described as follows:

Hypothesis and p-value

Hypothesis testing: Hypothesis testing is a procedure in which sample data are employed to evaluate a hypothesis. In order to evaluate a research hypothesis, it is restated within the framework of two statistical hypotheses; the null hypothesis (H_0) and the alternative hypothesis (H_1).

The null hypothesis is generally a hypothesis that the researcher expects to be rejected. The alternative hypothesis represents a statistical statement indicating the presence of an effect or a difference. In this case, the researcher generally expects the alternative hypothesis to be supported.

Once the data is collected, it is evaluated by means of an appropriate inferential statistical test to obtain a test statistic interpreted by employing special tables that contain information with regard to the expected distribution of the test statistic.

The conventional hypothesis testing model employed in inferential statistics assumes that prior to conducting a study, a researcher stipulates whether a directional or nondirectional alternative hypothesis is employed, as well as at what level of significance is represented the null hypothesis to be evaluated. The probability value which identifies the level of significance is represented by α . When one employs the term significance in the context of scientific research, it is instructive to make a distinction between statistical significance and practical significance. Statistical significance only implies that the outcome of a study is highly unlikely to have occurred as a result of chance, but it does not necessarily suggest that any difference or effect detected in a set of data is of any practical value.

p-value: A p -value is obtained when instead of stipulating *a priori* a level of significance α , one calculates the smallest level of significance that results in the rejection of the null hypothesis. p -value is a useful datum for many while conducting statistical analysis. It provides information about the significance of the statistical hypothesis. The smaller the p -value, the stronger is the evidence against the null hypothesis.

Friedman's test

It is applied to see if there are global differences in the results of k samples ($k \geq 2$). It is a multiple comparison test that aims to detect significant differences between the behaviour of two or more algorithms.

The null hypothesis for Friedman's test is $H_0: \theta_1 = \theta_2 = \dots = \theta_k$; the median of the population i represents the median of the population j , $i \neq j$, $1 \leq i \leq k$, $1 \leq j \leq k$. The alternative hypothesis is $H_1: \text{Not } H_0$.

It computes the ranking of the observed results for algorithm (r_j for the algorithm j with k algorithms) for each function, assigning first rank to the best performing algorithm, and k^{th} rank to the worst performing algorithm. Under the null hypothesis, formed from supposing that the results of the algorithms are equivalent and therefore their rankings are also similar, the Friedman's statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]$$

is distributed according to χ_F^2 with $k - 1$ degrees of freedom, being $R_j = \frac{1}{N} \sum_i r_i^j$ and N the number of functions. The critical values for the Friedman's statistic coincide with the values established in the χ^2 distribution when $N > 10$ and $k > 5$.

The rejection of the null hypothesis does not involve the detection of the existing differences among the algorithms being compared. It only informs that there are differences among the samples of results compared.

In order to conduct pair wise comparisons within the framework of multiple comparisons, a post-hoc procedure is employed. In this case, a control algorithm (maybe a proposal to be compared) is chosen. Then, the post-hoc procedures proceed to compare the control algorithm with the remaining $k - 1$ algorithms. It is done by the Bonferroni-Dunn's test given below.

Bonferroni-Dunn's Test

According to this test the performance of two algorithms is significantly different if the corresponding average of rankings is at least as great as its critical difference (CD).

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

Where q_{α} is the critical value.

A post-hoc statistical analysis helps to detect concrete differences among algorithms. In Bonferroni-Dunn's graphic the difference among rankings obtained for each algorithm is illustrated by drawing a horizontal cut line which represents the threshold for the best performing algorithm, that one with the lowest ranking bar, in order to consider it better than other algorithm. A cut line is drawn for each level of significance considered in the study at height equal to the sum of the ranking of the control algorithm and the corresponding CD computed by the Bonferroni-Dunn method. The bars which exceed this line are associated to an algorithm with worse performance than the control algorithm.

Wilcoxon signed-ranks test

It is a pairwise test that aims to detect significant differences between the behavior of two algorithms. Let d_i be the difference between the performance scores of the two algorithms on i^{th} out of N functions. The differences are ranked according to their absolute values; average ranks are assigned in case of ties.

Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N degrees of freedom, the null hypothesis of equality of means is rejected. The obtaining of the p -value associated to a comparison is performed by means of the normal approximation for the Wilcoxon T statistics.

In this study SPSS software package is used for statistical analysis. The level of significance is considered at $\alpha = 0.05$ (5%) and 0.10 (10%). A p -value greater than α indicates that there is no significant difference between the results.

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

Where q_{α} is the critical value.

A post-hoc statistical analysis helps to detect concrete differences among algorithms. In Bonferroni-Dunn's graphic the difference among rankings obtained for each algorithm is illustrated by drawing a horizontal cut line which represents the threshold for the best performing algorithm, that one with the lowest ranking bar, in order to consider it better than other algorithm. A cut line is drawn for each level of significance considered in the study at height equal to the sum of the ranking of the control algorithm and the corresponding CD computed by the Bonferroni-Dunn method. The bars which exceed this line are associated to an algorithm with worse performance than the control algorithm.

Wilcoxon signed-ranks test

It is a pairwise test that aims to detect significant differences between the behavior of two algorithms. Let d_i be the difference between the performance scores of the two algorithms on i^{th} out of N functions. The differences are ranked according to their absolute values; average ranks are assigned in case of ties.

Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N degrees of freedom, the null hypothesis of equality of means is rejected. The obtaining of the p -value associated to a comparison is performed by means of the normal approximation for the Wilcoxon T statistics.

List of Publication

1. **Musrrat Ali**, Millie Pant and Ajith Abraham, Improving Differential Evolution Algorithm by Synergizing Different Improvement Mechanisms, ACM Transaction on Autonomous and Adaptive Systems. **Accepted**
2. **Musrrat Ali**, Millie Pant, Ajith Abraham and Vaclav Snasel, Differential Evolution Using Mixed Strategies in Competitive Environment, International Journal of Innovative Computing, Information and Control. **Accepted**
3. **Musrrat Ali**, Millie Pant and Ajith Abraham, A Simplex Differential Evolution Algorithm: Development and Applications, Transactions of the Institute of Management and Control. DOI: 10.1177/0142331211403032. (2011).
4. **Musrrat Ali** and Millie Pant, Improving the Performance of Differential Evolution Algorithm Using Cauchy Mutation, Soft Computing, Vol. 15, pp. 991- 1007, (2011).
5. **Musrrat Ali**, Millie Pant and Ajith Abraham, Improved Differential Evolution Algorithm With Decentralization of Population, International Journal of Bio-Inspired Computation, Vol. 3(1), pp. 17- 30, (2010).
6. **Musrrat Ali**, Millie Pant and Atulya Nagar, Interpolated Differential Evolution for Global Optimisation Problems, International Journal of Computing Science and Mathematics, Vol. 3(3), pp. 298 - 315, (2010).
7. **Musrrat Ali**, Millie Pant and V. P. Singh, Two Modified Differential Evolution Algorithms and Their Applications to Engineering Design Problems, World Journal of Modeling and Simulation, Vol. 6(1), pp. 72-80, (2010).
8. **Musrrat Ali**, Millie Pant and Ajith Abraham, Simplex Differential Evolution, International Journal Acta Polytechnica Hungarica , Vol. 6(5), pp. 95- 115, (2009).
9. Millie Pant, **Musrrat Ali** and V. P. Singh, Parent Centric Differential Evolution Algorithm for Global Optimization Problems, Opsearch, Springer, Vol. 46(2), pp. 153-168, (2009).

10. **Musrrat Ali**, Millie Pant and V. P. Singh, An Improved Differential Evolution Algorithm for Real Parameter Optimization Problems, *International Journal of Recent Trends in Engineering*, Vol. 1(5), pp. 63-65, (2009).
11. Millie Pant, **Musrrat Ali** and V. P. Singh, Modified Differential Evolution Algorithms for Solving Unconstrained Global Optimization Problems, *International Journal of Mathematical Modeling, Simulation and Applications (IJMMSA)*, Vol. 2(3), pp. 245-256, (2009).

International Conferences

1. **Musrrat Ali**, Millie Pant, Ajith Abraham and Vaclav Snasel, Modified Differential Evolution Algorithm for Parameter Estimation in Mathematical Models, In Proc. of IEEE International Conference on Systems Man and Cybernetics (SMC 2010), Istanbul Turkey, pp. 2767- 2772, (2010).
2. **Musrrat Ali**, Millie Pant and V.P. Singh, Differential Evolution Using Interpolated Local Search, In Proc. of International Conference on Contemporary Computing (IC3, 2010), Springer, Noida, India, pp. 94-106, (2010).
3. **Musrrat Ali**, Millie Pant and Atulya Nagar, Two Local Search Strategies for Differential Evolution, In Proc. of IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2010), Liverpool UK, pp. 1429-1435, (2010).
4. Millie Pant, **Musrrat Ali** and V. P. Singh, A New Differential Evolution Algorithm and Its Application to Real Life Problems, *International Conference on Modeling and Engineering and Technological Problems (ICMETP) and the 9th Biennial National Conference of Indian Society of Industrial and Applied Mathematics (ISIAM 2009)*, In AIP Proc, Agra, India, Vol. 1146, pp. 177-185, (2009).
5. Millie Pant, **Musrrat Ali** and Ajith Abraham, Mixed Mutation Strategy Embedded Differential Evolution, In Proc. of IEEE congress on Evolutionary Computation (CEC 2009), Norway, pp. 1240-1246, (2009).

6. Millie Pant, **Musrrat Ali** and V. P. Singh, Differential Evolution Using Quadratic Interpolation for Initializing the population, In Proc. of IEEE International Advance Computing Conference (IACC 2009), Patiala, India, pp. 375-380, (2009).
7. **Musrrat Ali**, Millie Pant and V. P. Singh, A Modified Differential Evolution Algorithm With Cauchy Mutation for Global Optimization, In Proc. of International Conference on Contemporary Computing (IC3, 2009), Springer, Noida, India, pp. 127-137, (2009).
8. **Musrrat Ali**, Millie Pant and Ajith Abraham, A Modified Differential Evolution Algorithm and Its Application to Engineering Problems, In Proc. of International Conference of Soft Computing and Pattern Recognition (SoCPaR 2009), Malaysia, pp. 196-201, (2009).
9. **Musrrat Ali**, Millie Pant and Ajith Abraham, Inserting Information Sharing Mechanism of PSO to Improve the Convergence of DE, In Proc. of World Congress on Nature and Biological Inspired Computing (NaBIC 2009), Coimbatore, India, pp. 282-287, (2009).
10. **Musrrat Ali** and Millie Pant, Modified Differential Evolution Algorithms for Global Optimization, In Proc. of World Congress on Nature and Biological Inspired Computing (NaBIC 2009), Coimbatore, India, pp. 1686-1689, (2009).
11. **Musrrat Ali**, Millie Pant and Ajith Abraham, A Hybrid Ant Colony Differential Evolution and its Application to Water Resources Problems, In Proc. of International Symposium on Biologically Inspired Computing and Applications (BICA 2009), Bhubaneswar, India, pp. 1133-1138, (2009).
12. Millie Pant, **Musrrat Ali** and V. P. Singh, Differential Evolution With Parent Centric Crossover, In Proc. of Second UKSIM European Symposium on Computer Modelling and Simulation, Liverpool UK, pp. 141-146, (2008).